Modeling the Geometry of Neural Network Representation Spaces

by

Joshua David Robinson

MMath, University of Warwick (2018)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

Authored by: Joshua David Robinson
Department of Electrical Engineering and Computer Science
August 7, 2023

Certified by: Stefanie Jegelka
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by: Suvrit Sra
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Modeling the Geometry of Neural Network Representation Spaces

by

Joshua David Robinson

Submitted to the Department of Electrical Engineering and Computer Science
on August 7, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Neural networks automate the process of representing objects and their relations on a computer, including everything from household items to molecules. New representations are obtained by transforming different instances into a shared representation space, where variations in data can be measured using simple geometric quantities such as Euclidean distances. This thesis studies the geometric structure of this space and its influence on key properties of the learning process, including how much data is needed to acquire new skills, when predictions will fail, and the computational cost of learning. We examine two foundational aspects of the geometry of neural network representations.

Part I designs and studies learning algorithms that take into account the location of data in representation space. Focusing on contrastive self-supervised learning, we design a) hard instance sampling strategies and b) methods for controlling what features models learn. Each produces improvements in key characteristics, such as training speed, generalization, and model reliability.

Part II studies how to use non-Euclidean geometries to build network architectures that respect symmetries and structures arising in physical data, providing a powerful inductive bias for learning. Specifically, we use geometric spaces such as the real projective plane and the spectraplex to build a) provably powerful neural networks that respect the symmetries of eigenvectors, which is important for building Transformers on graph structured data, and b) neural networks that solve combinatorial optimization problems on graphs such as finding big cliques or small cuts, which arise in molecular engineering and network science.

Thesis Supervisor: Stefanie Jegelka
Title: Associate Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Suvrit Sra
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

The person graduating today is very different from the student who arrived at MIT five years ago. The process of change I have undergone over this time reflects the unparalleled research environment I have taken great pleasure in immersing myself in. Two factors stand out as especially important: the brilliance and kindness of many mentors and collaborators, and the intellectual freedom, and encouragement, to explore. In both cases I am indebted to others.

First and foremost comes my advisors Stefanie Jegelka and Suvrit Sra. More than anyone, their patient mentorship shaped every stage of PhD life. Starting from hiring a hopeful student of mathematics to work on machine learning, they were always generous with their time and ideas as I gradually explored the landscape of machine learning research. They were also incredibly trusting, giving freedom to explore new ideas that excited me, a privilege I will always be grateful for.

Next comes my collaborators, including Derek Lim, Ching-Yao Chuang, Nikos Karalias, Sharut Gupta, Shlok Mishra, Dilip Krishnan, Andreas Loukas, Zelda Mariet, Yen-Chen Lin, Li Sun, Ke Yu, Aaron Maschinot, Lingxiao Zhao, Haggai Maron, Soledad Villar and more. I am grateful to all of you for shaping the day-to-day experience of doing research, and teaching me so much in the process.

I am especially grateful to the members of Stefanie and Suvrit's research groups (formerly collectively known as LOGSS, but then everyone forgot what the acronym stood for). Many conversations and interactions stand out, and the breadth of knowledge within the groups influenced my interests greatly. For instance, Keyulu Xu taught me what a graph neural network was back in 2018, and more recently Michael Murphy (very patiently) explained all the ways in which molecules are far more complex and interesting than their 2D graph descriptions. Matt Staib taught me many things, including what NeurIPS and ICML are, what PhD students are supposed to do, and many many optimization problems, including submodular ones.

Every time I spoke with Derek Lim I learned something new, and over the course of months and years he taught me about symmetries in neural networks, a topic I am

very glad to now consider as an *interest* of mine. His unbelievably detailed knowledge of both the mathematical and engineering landscapes, and seemingly effortless ability to produce new mathematical facts or intuition based on what he already knew made him a world-class one-stop shop for any idea or question I may have. On top of all of that, he has also become a close friend.

Ching-Yao Chuang played a pivotal role in my time at graduate school. Introducing me to the world of self-supervised learning, including contrastive learning, a pair of papers written with Ching-Yao marked a key shift in my work from pure theory to ideas and methods at the intersection of empirical and theoretical deep learning.

With Nikos Karalias I explored a new world of neural combinatorial optimization—a mathematically beautiful synthesis of two topics that I had been interested in, but never managed to bring together before. Nikos always had the knack of pulling out exactly the right mathematical tool—often a theorem of object with a fancy name, often gifted to us long ago by László Lovász. Working asynchronously from the US and Switzerland, we were able to continually engineer our approach and synchronize often last thing at night. This close collaboration was a lot of fun, and its a cruel irony that Nikos is arriving in Cambridge exactly at the same time I am leaving!

I also learned an extraordinary amount about self-supervised learning in particular and how to think intuitively about learning at scale in general from Dilip Krishnan. The many discussions over many hours have been both a joy, and influential on my thinking.

Outside of research life, many people have helped me enjoy life and keep it balanced. From visit days and the start of SidPac include Matthieu, Sarah, John, Camille, Jules, Ola, Peter, James, Serena, Nico, Charlotte. From there, I met many other great friends at MIT, including Graeme, Chandler, Aspen, Kevin, Isaac, Karima, Ondrej, Lukas and so many more. Other friends who have meant a lot to me include Matt and Sizi, Matt West, Phil.

As well as new friends since moving to the US, I have been able to enjoy the company of many friends from home, despite the distance, including Dillon, Darshan, Ned, Rob, George, Charlie, and Guy.

I owe a debt of thanks that I know can never be repaid to my mum, dad, and Isabel. Over the course of my lifetime they have poured more love and care into me than I can comprehend. And, just like the ideal PhD advisor, they knew exactly when I needed support, and when to let me explore. I am especially proud of Isabel, who is growing into academic life so successfully.

And finally, Jess, who more than anyone has made my life fulfilling and full of joy. Of course, this thesis would have been much harder to finish without your companionship. But more importantly you helped me discover what I want from life, and how to get there—Kocham cię małpo!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

How observations from the world are described on a computer influences all parts of the computational problem solving process. For this reason, discovering descriptions, or *representations*, that are more amenable to simple computational processes has been a key and consistent driver of advances in computational science for many decades.

As with all other computational process, what representation of data an AI system extracts and uses critically affects its behavior, such as how much data is needed to acquire new skills, when predictions will fail, and the speed at which it can learn. However, one of the most important strengths of modern AI systems driven by deep learning is their ability to operate on basic perceptual inputs such as individual pixels of an image, and determine internally what information from the pixel-level features to keep, and what to discard.

Besides automating representation acquisition, another key strength of deep learning systems is the ability to represent different data points in a *single, coherent space*. For instance, images living in extremely large, high-dimensional spaces (for instance, immunohistochemistry images are commonly of size $1024 \times 1024 \times 3$ or larger) and converted into comparatively small, compact spaces of a few thousand dimensions. This space of representations allows comparisons between data, and geometric properties of this space such as distances, directions, and angles capture not only individual objects, but how objects relate to one another.

However, this geometry is often emergent. That is, it is not directly learned, and

instead learned as an intermediate step in an end-to-end system using low-granularity supervision such as class labels. The objective of this thesis is to explore ideas, methods, and new directions for more explicitly programming this geometry. Through explicit handling, the hope is to develop a greater control over the properties of representation space, making it easier to use this space to reason about the world.

1.1 Neural Network Representation Space

The central promise of deep learning is to learn a map

$$f : \mathcal{X} \to \mathbb{R}^d \tag{1.1}$$

from the space of objects $\mathcal{X}$ expressed in a basic form in which we are able to perceive them (as images, molecular strings, etc.) to a *representation space* $\mathbb{R}^d$ where everything that is hard to do with raw perceptual data becomes easy. For instance, quantifying the similarity between two objects expressed as tensors $x_1, x_2 \in \mathbb{R}^{h \times w \times 3}$ of pixel intensities is non-trivial, but becomes easy if f maps image data into a space where simple Euclidean distances $\|f(x_1) - f(x_2)\|_2$ constitute a meaningful measure of similarity.

Just as learning f is the central *promise* of deep learning, how to learn f is the central *question*. It is challenging not least because it is hard to precisely specify what properties the ideal representation has in order to be useful for unknown downstream tasks. In this thesis we address both how to learn f, and what should f encode from the point of view of the geometric structure of representation space.

1.2 Part I: Learning a Representation Geometry

Part I begins with the question of *how* to learn Euclidean structured representation space. This question covers a huge breadth of work in general, and a large part of this thesis in particular. The more precise goal of this section is to design learning algorithms that explicitly depend on the location of data in embedding space. This is in contrast, for instance, with supervised pretrained models, which do learn representations, but

only implicitly—there is no loss function on the embeddings themselves, the loss only depends on embedding via their connection to the output class logits. This also holds from more contemporary self-supervised learning methods such as the masked autoencoder [He et al., 2022], which likewise does not directly optimize the learned embeddings themselves, but computes the loss after a Transformer decoder. Our approach differs by focusing on learning algorithms that not only use the locations of embeddings, but the location of embeddings *relative* to one another to design better sampling procedures and objective functions.

1.3 Part II: Programming Special Geometries to Reflect Problem Structure

In Part I of this thesis, and typically in deep learning, it is assumed that the neural network feature extractor $f : \mathcal{X} \to \mathbb{R}^d$ has Euclidean output space. There are good reasons for this, and it is often not necessary to look further. Indeed, Euclidean space is unique among finite dimensional vector spaces. A linear space that 1) finite-dimensional (i.e., tractable for storing on a computer), and 2) to have the basic geometry needed to compare data—i.e., distances and angles—and 3) to be *complete*, so that it is possible to define derivatives for gradient-based training, then Euclidean space is in fact the only option as this standard result in functional analysis states.

Theorem 1. *Every finite dimensional Hilbert space over real numbers is isometrically isomorphic to Euclidean space.*

In other words, the basic properties required for learning representations with neural networks are possessed by Euclidean space, and moreover Euclidean space is the only space with these properties. For these reasons and more, Euclidean space is rightfully *the* default for deep learning. In the absence of strong prior knowledge to lead the engineer to believe that another geometry is required, there is no reason to look beyond Euclidean space.

That said, the goal of Part II of this thesis is to demonstrate that there *are*

systematic cases in which non-Euclidean geometries are not only useful but *essential* to the model's ability to learn new skills. Why this is the case is non-obvious, and one of the key contributions of this thesis is to simply understand when geometries beyond Euclidean space are desirable. We explore two key problem classes which call for a different choice of geometry:

1. When there are symmetries in input space $\mathcal{X}$ (Section 1.3.1, Chapter 7).

2. When f is representing combinatorial objects such as paths and cliques in a graph (Section 1.3.2, Chapter 8).

1.3.1 Encoding Symmetries of Data as Specialized Representation Geometry

Symmetries of input space $\mathcal{X}$ can be formalized as the existence of a group G acting on $\mathcal{X}$. That is, there is a map $\cdot : G \times \mathcal{X} \to \mathcal{X}$ that a) respects the group identity operation: $e \cdot x = x$ for the identity $e \in G$ and any $x \in \mathcal{X}$, and b) is consistent under compositions: $g_1 \cdot (g_2 \cdot x) = (g_1 g_2) \cdot x$ for all $g_1, g_2 \in G$ and $x \in \mathcal{X}$. In this formalism, x and $g \cdot x$ are considered equivalent for all $x \in \mathcal{X}$ and $g \in G$—that is, all points in the same *orbit*—and therefore we wish $f(x) = f(g \cdot x)$ to be *invariant* to the action of G.

For instance, $\mathcal{X}$ could be the space of images, and G translations and rotations of Euclidean space. A translated or rotated object is still the same object, so a model identifying objects should be invariant to translations. Caution is advised, however, since incorrectly enforced invariances reduce the expressive power of models in a way that can be severely limiting. For instance, a translation and rotation invariant model will be completely unable to solve certain tasks, for instance distinguishing the digits 6 and 9.

What approaches are there for encoding f to be invariant to G? We explore several options considering their advantages and disadvantages.

Group Averaging. One natural option is to parameterize f using a secondary

model $h : \mathcal{X} \to \mathbb{R}^d$ and defining

$$f(x) = \frac{1}{|G|} \sum_{g \in G} h(g \cdot x) \quad \text{for all } x \in \mathcal{X}. \tag{1.2}$$

This addresses the group symmetry in output space though an average. In certain circumstances this approach works well, and it is always worth considering this strategy since it is simple. However, it has severe limitations for many applications. The most significant limitation is that this approach only works for relatively small finite groups. Exponentially large groups and continuous groups do not permit efficient computation of the average over G in general. A second reason is that the parameterization of h has an effect on the gradient dynamics of f. Specifically, although a function class may appear expressive, the fact that the function is parameterized as a sum can lead to "degenerate" gradients that make it impossible to learn a good model via gradient descent. An example is the following: $G = S_n$, the permutation group on n elements, and $h(x) = Wx$ for $x \in \mathbb{R}^n$ and $W \in \mathbb{R}^{n \times d}$ is the family of linear models. Then:

$$f(x) = \frac{1}{n!} \sum_{\sigma \in S_n} W(\sigma \cdot x) = \frac{W}{n!} \sum_{\sigma \in S_n} (\sigma \cdot x) = W\big(\sum_i x_i\big)\mathbf{1} \tag{1.3}$$

where $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector. In other words, the only linear functions that are also permutation invariant can be written as a function of $\sum_i x_i$ alone. That is, the function f we have parameterized with h is now considerably less expressive than h—the number of free parameters is only d, not $n \times d$. The takeaway is that it is not a given that the group average f is an expressive model just because h is, and careful checking is called for.

Canonicalization. Group averaging deals with the group symmetry in the output space of h. It is also possible to deal with symmetry in the input space of h via canonicalization. Letting $[x] = \{g \cdot x : g \in G\}$ denote the orbit of x in G, one option is to define a hardcoded canonicalizing map $\phi : \mathcal{X} \to \mathcal{X}$ such that $\phi(x) = \phi(x')$ for all $x' \in [x]$, then the model

$$f(x) = h(\phi(x)) \tag{1.4}$$

is invariant to G. Furthermore, if h is universally expressive then so is f. This situation appears good. However, not all canonicalizing maps are created equal, and different choices lead to subtle differences in behavior. The *smoothness* of the canonicalizing map ϕ critically affects the generalization of $f(x) = h(\phi(x))$. For instance, consider again the case of $G = \{+1, -1\}$ and $\mathcal{X} = \mathbb{R}$, and $h_a(x) = ax$ parameterized by $a \in \mathbb{R}$, leading to an overall hypothesis family $f(x) = h_a(\phi(x)) = a\phi(x) = a|x|$. Now suppose also that the ground truth target function is $F(x) = 4|x|$, so our hypothesis class is aligned well to the task at hand, but that we only observe training data pairs $(x_i, F(x_i))$ with x_i in the interval $[-1, +1]$. In this case the limited coverage of the input space is no issue, since the empirical loss minimizer is

$$a^* = \arg\min_{a \in \mathbb{R}} \sum_i \big(h_a((\phi(x_i)) - F(x_i)\big)^2 \tag{1.5}$$

$$= \arg\min_{a \in \mathbb{R}} \sum_i \big(a|x_i| - 4|x_i|\big)^2 \tag{1.6}$$

$$= \arg\min_{a \in \mathbb{R}} (a-4)^2 \sum_i x_i^2 \tag{1.7}$$

$$= 4 \tag{1.8}$$

and the empirically optimal model $f(x) = a\phi(x) = 4|x|$ generalizes perfectly to all of the real line.

However, another valid choice of canonicalizing function is:

$$\phi(x) = \begin{cases} |x| & \text{if } |x| \leq 1 \\ -|x| & \text{otherwise.} \end{cases} \tag{1.9}$$

If again we assume that $h_a(x) = ax$, and we only observe data data pairs $(x_i, F(x_i))$ with x_i in the interval $[-1, +1]$, then as before we find that the empirical minimizer of the loss is $a^* = 4$. However, in this case the learned function is $f(x) = 4|x| \cdot \text{sign}(1 - |x|)$ which approximates $F(x) = 4|x|$ well within the training distribution, but completely fails to generalize when $|x| > 1$.

The takeaway from this example is that different canonicalizations ϕ can have very

different generalization properties. One reason is that the overall family $f(x) = h(\phi(x))$ may change—indeed the families $f(x) = a|x|$ and $f(x) = 4|x| \cdot \text{sign}(1 - |x|)$ are different, and therefore are able to learn different target functions. However, changing the hypothesis class is not the only way different canonicalization can affect learning. Suppose that instead of $h(x) = ax$ we took $h(x) = \text{MLP}(x)$ where MLP is a feed-forward neural network that is allowed to be any width and depth and so is a universal approximator of continuous functions. In this case, the hypothesis classes induced by the two different cannonicalizations are:

$$f(x) = \text{MLP}(|x|) \quad \text{and} \quad f(x) = \text{MLP}(|x| \cdot \text{sign}(1 - |x|)). \tag{1.10}$$

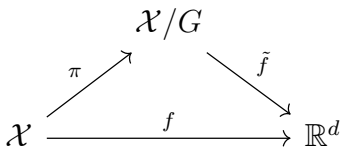
Both of these models universally approximate continuous functions $\mathbb{R} \to \mathbb{R}^d$ (more precisely, functions with compact domains $C \subseteq \mathbb{R}$) that are invariant to the sign group $G = \{-1, +1\}$. But, whilst their expressive power is the same, their generalization out-of-distribution will be very different. It is known that feed-forward neural networks extrapolate linearly [Xu et al., 2021], so the model $\text{MLP}(|x|)$ trained on input data from $[-1, 1]$ and tested on data from all of $\mathbb{R}$ should still approximate $F(x) = 4|x|$ well, but $\text{MLP}(|x| \cdot \text{sign}(1 - |x|))$ will fail outside of the training interval $[-1, 1]$.

A guiding principle for selecting a suitable ϕ is to find the simplest—e.g., smoothest—canonicalization possible. However, a precise formulation for how to do this, when it is even possible, and the precise implications for generalization remain unexplored.

Specialized Representation Space Geometry. In this thesis we take a different approach. The key mathematical insight is that the underlying object of interest in this setting is the quotient space $\mathcal{X}/G$, and that defining a function $f : \mathcal{X} \to \mathbb{R}^d$ that is invariant to G is the same as defining an arbitrary function $\tilde{f} : \mathcal{X}/G \to \mathbb{R}^d$. More specifically, every G invariant function f can be written as $f = \tilde{f} \circ \pi$ where $\pi : \mathcal{X} \to \mathcal{X}/G$ is the quotient map, which maps each x to its equivalent class under G. If our intention is to parameterize both π and $\tilde{f}$ with neural networks, this means that one of the internal representation spaces of our network is $\mathcal{X}/G$, a space with interesting and potentially non-Euclidean geometry. For instance, if we are interested

in building a permutation invariant model, then $\mathcal{X} = \mathbb{R}^n$ is Euclidean space, and $G = S_n$, the permutation group on n elements. Then the quotient $\mathbb{R}^n/S_n$ is an Orbifold, a generalization of a manifold.

Note also that both maps π and $\tilde{f}$ are continuous whenever f is continuous. This relation can be expressed in terms of the following commutative diagram:



The key property of π is that it is injective up to equivalences. That is, if $x' \notin [x]$ then $\pi(x) \neq \pi(x')$. This suggests a general recipe for designing a G-invariant neural network: 1) Show that $\mathcal{X}/G$ is homeomorphic to some subspace of a Euclidean space $\mathbb{R}^{d'}$ for some d', 2) parameterize a function $\phi : \mathcal{X} \to \mathbb{R}^{d'}$ that is G-invariant and injective up to G-equivalence, and 3) parameterize $\tilde{f} : \mathbb{R}^{d'} \to \mathbb{R}^d$ to be any universal approximator, such as an MLP. If all steps are completed, then the resulting model is guaranteed to be universally expressive and G invariant.

The first step requires use of geometric properties of the quotient space $\mathcal{X}/G$. For instance, in Chapter 7 on eigenvector symmetries we have $\mathbb{S}^{n-1}$ (the unit sphere in $\mathbb{R}^n$), and $G = \{+1, -1\}$. In this case $\mathbb{S}^{n-1}/\{+1, -1\} = \mathbb{RP}^{n-1}$, the real projective plane. The well-known fact that $\mathbb{RP}^{n-1}$ is a smooth manifold implies that 1) is true by the Whitney Embedding theorem [Whitney, 1944]. Of the second and third steps, which still remain, the third is trivial, leaving only the second. The second step is the point where the designer must inject knowledge of the symmetry group G into the network architecture. Currently this step largely relies on the cunning of the designer, but it is an interesting and important future direction to develop general processes for construction architectures that work for larger classes of symmetry groups.

1.3.2 Representations that Encode Combinatorial Structure

Problems with combinatorial structure are notoriously hard for neural networks to learn to solve. This is unsurprising for several reasons, not least that many combinatorial problems are provably hard in the sense of complexity theory. Furthermore, there appears a natural, fundamental, divide between the discrete, algebraic, nature of many combinatorial tasks, and the differentiable—or at least continuous—structure of neural network loss landscapes and representation spaces. Indeed, as noted earlier, one key reason that Euclidean space is the default representation space is that it is a continuous and complete space, allowing gradients to be defined and computed. The mathematical field of representation theory [Curtis and Reiner, 1966] is well placed to design differentiable expressions of combinatorial structures, since it provides many tools for embedding discrete combinatorial objects as matrices. Chapters 5 and 8 both explore this theme, in two very different contexts.

1.4 Thesis Outline

This thesis has two main parts. Part I studies contrastive representation learning, a general framework for learning Euclidean structured representations. This part begins with Chapter 2 which outlines a general introduction to contrastive learning that is of use throughout the thesis. Then, Chapter 3 asks the following question: how should negative samples be produced for contrastive learning? Negative samples are one of the two key design choices in contrastive learning, but most prior negative sampling approaches are naive, e.g., sampling uniformly at random. We introduce techniques for sampling negatives according to their location representation space. By sampling negatives in a location-dependent way we are able to generate more informative samples that accelerate learning and improve the quality of the final representation space for downstream tasks.

In Chapter 4 we examine more carefully which input data features—out of the many possibilities—do contrastive learning models actually encode? Unsurprisingly, we show that how positive and negative samples are generated affects feature learning

and sketch an intuitive approach to understanding this relation. More surprisingly, we also find that obvious ways to adjust sampling—e.g., by adjusting the hardness of negatives—does not lead to a clear winner. Different sampling methods do better on some downstream tasks and worse on others. In response to this we explore a new principle for modifying samples using the representation space geometry. Our *implicit feature modification* method perturbs embeddings to remove whatever features are being used to solve the contrastive task, then asks the model to re-solve the contrastive task using the new features. This forces the model to look for new and different features to use, thereby encouraging the model not to ignore aspects of the input data that it previously might have.

Chapter 5 takes a step back to reconsider what structure representation space should have. As well as encoding similarities between data, Chapter 5 argues that it is also valuable to encode complex transformations of data as predictable, simple transformations of embedding space. Specifically, we introduce a training method that forces augmentations of data—cropping, rotation etc.—to correspond to orthogonal transformations of the spherical embedding space. We find that our approach leads to improved downstream performance, as well as increased sensitivity to data features that other contrastive methods do not.

The final chapter of this section, Chapter 6, explores contrastive learning at scale. As model parameters and dataset size are scaled into the hundreds of millions and billions it is important to consider not only how the performance of models scales, but also the cost of training them. Training methods whose performance scales reliably, but whose cost grows excessively will ultimately lose out to methods whose performance scales more slowly, but whose cost grows even slower. Contrastive learning is relatively costly at scale since it uses two entire copies of each batch sample. Another re-emerging training method—the *masked autoencoder* (MAE)—is highly efficient since it uses only 25% of one copy of each batch sample, but has relatively poor performance, especially for few-shot downstream tasks. Our contribution is to introduce CAN, a *hybrid* of contrastive learning and MAE that enjoys better efficiency-performance trade-offs. It is 70% less costly than contrastive learning, whilst outperforming both contrastive

learning and MAE. An important interpretation of CAN is geometric: MAE learns powerful embeddings, but fails to arrange their geometry in a linearly separable way since it has a non-linear Vision Transformer decoder, which can easily undo non-linear transforms. By adding a contrastive component to the embedding space of MAE (i.e., before the decoder), CAN forces the Euclidean distances between embeddings to be meaningful, leading to much improved few-shot performance.

Part II of this thesis explores the relation between representation space geometry and problem structure. Chapter 7 considers symmetry, specifically the symmetries of eigenvectors (e.g., sign symmetry: if v is an eigenvector then so is $-v$). Eigenvectors are widely used in machine learning. This is especially the case for learning on graphs, for which spectral theory has provided a firm mathematical foundation for many years. This continues in the present day, where eigenvectors of the graph Laplacian are widely used as node positional encodings for building Tranformers on graphs. However, the sign symmetry (and a more general basis symmetry which occurs when eigenvalues have multiplicity greater than 1) is a problem since neural networks are not invariant to changes in sign, and therefore make predictions that change unreliably under irrelevant changes to input data.

To address this we introduce a sign invariant architecture SignNet—and a basis invariant BasisNet—which have the symmetries of eigenvectors built in. Our models considerably improve the performance when using Laplacian positional encodings, and are mathematically powerful: SignNet can approximate all sign invariant functions. The expressive power of SignNet is based on the geometry of its embedding space. We show that the ideal space for eigenvectors is the space $\mathbb{R}^n/\{-1,+1\}$ that quotients out the sign ambiguity by gluing v and $-v$ together into a single point. Critically, this quotient space is a well studied manifold known as the real projective plane. Using geometric properties of the real projective plane—specifically, that it can be rendered in a $2n$-dimensional Euclidean space—we are able to design the pieces of SignNet so that they are processing vectors in Euclidean space at all times, but these Euclidean spaces are connected up in such a way as to mirror the real projective plane, from which we derive the universal expressivity result. The analysis for BasisNet also proceeds in

the same way, but using the corresponding quotient space, the Grassmanian.

The final chapter of this thesis, Chapter 8, designs representation geometries that make neural networks more suitable for solving combinatorial optimization problems such as finding large cliques and small cuts in a graph. Combinatorial tasks such as these can be formulated as searching for a set of items—such as a set of nodes—which can be described as a Boolean vector $S \in \{0,1\}^n$ with 1 in coordinate i denotes the inclusion of object i in the set, and 0 non-inclusion. However, this choice of space immediately runs into difficulties since it is discrete, interfering with gradient based training of neural networks. A natural step to avoid this is to *extend* the space to the cube $[0,1]^n$, and decode set solutions at inference time using heuristics such as thresholding. However, the key insight of our work is that this choice of space is still sub-optimal for neural network training. The reason is that each item—e.g., each node—is now described by a single scalar value between $[0,1]$. This misses out on one of the key strengths of neural networks: representation of objects as *high-dimensional* vectors. But it is non-obvious what high-dimensional space is suitable and viable. Our contribution was to identify the spectraplex, the space of positive-semidefinite matrices with trace equal to 1, as a suitable choice. With this choice of space each object is represented by a high-dimensional vector, and so is better suited to neural network training. The reasoning that led us to choose the spectraplex is to start from the cube extension, and so that it can be re-expressed as the solution to a certain linear program (LP—linear optimization over vectors). We then move to matrices by specifying a semi-definite program (SDPs are linear optimization problems over matrices with non-negative eigenvalues) that is a counterpart to the LP, which is an optimization problem over vectors. We then specify our high-dimensional extension as the solution to this SDP.

1.5 Additional Related Publications

The author also published several other works during the PhD program. These cover mathematical foundations of learning, including Monte Carlo methods, discrete

probability, symmetry, and statistical learning theory. They have been omitted from this thesis in order to maintain focus, and keep to a reasonable length. Here we very briefly discuss each work.

1. **Flexible Modeling of Diversity with Strongly Log-Concave Distributions**, Joshua Robinson, Suvrit Sra, and Stefanie Jegelka. NeurIPS 2019 [Robinson et al., 2019]. This work studies the class of *strongly log-concave* (SLC) distributions on subsets of the set $\{1, , \ldots, n\}$ of n elements. SLC distributions subsume the well-known determinantal point process (DPPs). SLC distributions, like DPPs, model objects with a repulsive, or negative dependence. Arising in particle physics, these distributions have found applications in machine learning where it is desirable to encourage a *diverse* set of samples or outcomes. In this work we study foundational statistcial algorithms such as MCMC sampling and finding the mode of a distribution.

2. **Strength from Weakness: Fast Learning Using Weak Supervision**, Joshua Robinson, Stefanie Jegelka, and Suvrit Sra. ICML 2020 [Robinson et al., 2020]. This paper studies the following fundamental question in statistical learning theory: when can pretraining on a large dataset, followed by finetuning on a small amount n of labeled data, achieve an excess risk that decreases at the *fast rate* of $O(1/n)$, as opposed to the typical rate of $O(1/\sqrt{n})$. In this paper we show that fast rates are achieved when the pretranining task is related to the downstream task in the following intuitive way: there exists an embedding z of data such that there is a linear model $W_1 z$ that achieves zero population risk on the pretraining task, and another linear model $W_2 z$ that achieves zero downstream population risk.

3. **Optimal Batch Variance with Second-Order Marginals**, Zelda Mariet, Joshua Robinson, Jamie Smith, Suvrit Sra, Stefanie Jegelka. ICML 2020 Workshop on Real World Experiment Design and Active Learning [Mariet et al., 2020]. This paper considers a problem at the core of learning: which minibatch of data to select to compute stochastic gradients on? We formulate this problem as

searching for the distribution over minibatches of k for which the variance of the stochastic gradient is minimized. We show that this implies specific linear and quadratic constraints on the distribution, and optimize over a parametric class (DPPs) for the distribution most closely matching these conditions. We find that this can lead to faster convergence on small-scale problems.

4. **Expressive Sign Equivariant Networks for Spectral Geometric Learning**, Derek Lim, Joshua Robinson, Stefanie Jegelka, Yaron Lipman, Haggai Maron. ICLR 2023 Workshop Physics4ML [Lim et al., 2023a]. This work directly extends Chapter 7. The work begins by exploring a previously not understood property of the sign invariant networks we introduce in Chapter 7. We show that if two nodes in a graph are automorphic, then a sign invariant network will give both nodes the same embedding. This is a problem for tasks that require distinguishing automorphic nodes—such as link prediction tasks and certain node classification tasks. We show that sign *equivariant* networks overcome this issue, and design powerful classes of sign equivariant models. Additionally, we show that orthogonal equivariance—which is of high interest in the physical sciences since particle systems have this symmetry—can be reduced to sign equivariance. This is especially remarkable since the orthogonal group is continuous, whilst the sign group is finite.

Part I

Contrastive Learning

Chapter 2

The Basics of Contrastive Learning

Given access only to samples from a distribution $p(x)$ on input space $\mathcal{X}$ (e.g., images), the goal of contrastive self-supervised learning is to train a feature extracting model $f : \mathcal{X} \to \mathbb{S}^{d-1}$ mapping to the unit sphere $\mathbb{S}^{d-1} = \{z \in \mathbb{R}^d : \|z\|_2 = 1\}$. Contrastive learning achieves this by automatically generating supervision from the data in the form of a space of augmentations $\mathcal{A}$, containing maps $a : \mathcal{X} \to \mathcal{X}$ which slightly perturb inputs $\bar{x}$ (blurring, cropping, jittering, etc.) in such as way that the perceptual meaning of the input does not change much. For instance, a cropped image of a dog is still a dog. This enables the generation of similar datapoints $x = a(\bar{x})$ and $x^+ = a^+(\bar{x})$, with $\bar{x} \sim p(x)$ and $a \sim \mathcal{A}$.

How can the similarity between x and x^+ be used to guide neural network training? Siamese self-supervised learning methods cover the spectrum of all methods that train f by computing $f(x)$ and $f(x^+)$ in parallel (hence the name *Siamese*, after Siamese twins), and then designing training criteria based on comparisons between the embeddings.

One of the most natural principles is *invariance*: train f to embed x and x^+ nearby—i.e., so that $f(x) = f(x^+)$. A loss that achieves this is to simply optimize $\mathcal{L}_{\text{inv}}(f) = \|f(x) - f(x^+)\|_2$. However this loss alone is insufficient since it immediately leads to trivial solutions which fail to extract useful features such as $f(x) = \text{constant}$. In order to avoid trivial solutions, many self-supervised methods have been proposed that can be formulated as adding a second loss term alongside $\mathcal{L}_{\text{inv}}$ that rules out

trivial solutions. Contrastive learning achieves this by adding a term $\mathcal{L}_{\text{unif}}(f)$ that is zero if and only if $z = f(x)$ is uniformly distributed on the sphere $\mathbb{S}^{d-1}$ [Wang and Isola, 2020b]. By minimizing the sum $\mathcal{L}_{\text{inv}}(f) + \mathcal{L}_{\text{unif}}(f)$, f has to balance two competing but not contradictory goals: enforcing $f(x) \approx f(x^+)$ whilst ensuring $f(x)$ and $f(x')$ are approximately orthogonal for two independent samples.

The InfoNCE loss [van den Oord et al., 2018, Gutmann and Hyvärinen, 2010] used in contrastive learning achieves precisely this:

$$\mathcal{L}_{\text{InfoNCE}}(f) = \mathbb{E}_{x,x^+,\{x_i^-\}_{i=1}^N}\left[-\log \frac{e^{f(x)^\top f(x^+)/\tau}}{e^{f(x)^\top f(x^+)/\tau} + \sum_{i=1}^N e^{f(x)^\top f(x_i^-)/\tau}} \right], \tag{2.1}$$

where $\tau > 0$ is a temperature hyperparameter, and $x_i^- \sim p$ are negative samples from the marginal distribution on $\mathcal{X}$.

The formal connection between the InfoNCE loss and uniformity was first established by Wang and Isola [2020b], who shows that in the limit as the number N of negative pairs goes to infinity,

$$\lim_{N\to\infty} \left\{ \mathcal{L}_{\text{InfoNCE}} - \log N \right\} = \mathbb{E}_{x,x^+} \|f(x) - f(x^+)\|^2 + \mathbb{E}_x \log \mathbb{E}_{x^-} e^{f(x)^\top f(x^-)} \tag{2.2}$$

up to a constant, where x, x^- are independent samples from the marginal $p(x)$ (we also set $\tau = 1$ for simplicity). Critically they analyze,

$$\mathcal{L}_{\text{unif}} = \mathbb{E}_x \log \mathbb{E}_{x^-} e^{f(x)^\top f(x^-)} \tag{2.3}$$

and show that it is minimizied when $f(x)$ is uniformly distributed on the sphere, so this loss is indeed a uniformity-promoting loss, ruling out trivial solutions such as constant functions.

Finally, the InfoNCE loss can also be interpreted as a classification task with cross-entropy loss [van den Oord et al., 2018]. The task is as follows: given x and a bag of candidates $\mathcal{N} \subseteq \mathcal{X}$ with $|\mathcal{N}| = N + 1$, exactly one of which is *similar* to x, identify the unique similar sample (i.e., positive) $x^+ \in \mathcal{N}$. This task can be connected to the InfoNCE loss by computing a similarity score for each $x' \in \mathcal{N}$ as $f(x)^\top f(x')$

and simply taking $\{f(x)^\top f(x')\}_{x' \in \mathcal{N}}$ to be the logits that are passed to the standard cross-entropy loss.

The considerable number of interpretations of the contrastive learning mechanism is a good thing, suggesting many connections to prior ideas that has and will continue to come in use when adapting the contrastive learning setup to new problem settings. Finally, whilst there have been a number of interesting and high profile *non-contrastive* Siamese methods that suggest alternative non-collapse loss terms $\mathcal{L}_{\text{inv}} + \mathcal{L}_{\text{non-collapse}}$, recent works have shown that both empirically and theoretically the differences are less significant than they first appeared. For instance, Tao et al. [2022a] show that although the loss functions may look different, their gradients are nonetheless similar, all taking the form of *decorrelation* of embeddings, either across different samples as in the InfoNCE loss, or across different coordinates of the embedding vector, as in Barlow Twins [Zbontar et al., 2021]. Additionally, Garrido et al. [2022] formally demonstrate an equivalence between contrastive and non-contrastive objectives. As a consequence, we do not lose any generality by exclusively considering contrastive learning as our prototypical Siamese self-supervised learning approach.

Chapter 3

Negative Sampling in Contrastive Learning

Which samples make good negatives for contrastive learning? The goal of this chapter is to argue that, as with metric learning, contrastive learning benefits from samples that a) are hard negatives (i.e., points that are difficult to distinguish from an anchor point) but b) are not false negatives. This is in contrast to the primary existing paradigm for negative sampling, which is to draw them uniformly at random from the training dataset. The key challenge toward sampling more informative hard negatives is that contrastive methods must remain unsupervised, making it infeasible to adopt existing negative sampling strategies that use *true* similarity information. In this chapter we develop a new family of unsupervised sampling methods for selecting hard negative samples where the user can control the hardness. A limiting case of this sampling results in a representation that tightly clusters each class, and pushes different classes as far apart as possible. The proposed method improves downstream performance across multiple modalities, requires only few additional lines of code to implement, and introduces no computational overhead.

Acknowledgements. This work is in collaboration with Ching-Yao Chuang, Yen-Chen Lin, Antonio Torralba, Suvrit Sra and Stefanie Jegelka. In particular, Ching-Yao led the initial exploration of false negative samples [Chuang et al., 2020], which we include in this chapter as Robinson et al. [2021a] extends these results.

3.1 Background and Motivation

Contrastive learning recipes involve two key ingredients: notions of similar (positive) (x, x^+) and dissimilar (negative) (x, x^-) pairs of data points. The training dynamics guide the learned representation f to balance to conflicting objectives: mapping positive pairs to nearby locations, and negative pairs far apart. The success of the associated methods depends on the design of informative of the positive and negative pairs, but the challenge is the lack of ground truth similarity information since we are in the self-supervised setting.

Much research effort has addressed sampling strategies for positive pairs, and has been a key driver of recent progress in multi-view and contrastive learning [Blum and Mitchell, 1998, Xu et al., 2013, Bachman et al., 2019, Chen et al., 2020b, Tian et al., 2020b]. For image data, positive sampling strategies often apply transformations that preserve semantic content, e.g., jittering, random cropping, separating color channels, etc. [Chen et al., 2020b,e, Tian et al., 2019]. Such transformations have also been effective in learning control policies from raw pixel data [Srinivas et al., 2020]. Positive sampling techniques have also been proposed for sentence, audio, and video data [Logeswaran and Lee, 2018, van den Oord et al., 2018, Purushwalkam and Gupta, 2020, Sermanet et al., 2018].

Surprisingly, the choice of negative pairs has drawn much less attention in contrastive learning. Often, given an "anchor" point x, a "negative" x^- is simply sampled uniformly from the training data, independent of how informative it may be for the learned representation. In supervised and metric learning settings, "hard" (true negative) examples can help guide a learning method to correct its mistakes more quickly [Schroff et al., 2015, Song et al., 2016]. For representation learning, informative negative examples are intuitively those pairs that are mapped nearby but should be far apart. This idea is successfully applied in metric learning, where true pairs of dissimilar points are available, as opposed to unsupervised contrastive learning.

With this motivation, we address the challenge of selecting informative negatives for contrastive representation learning. In response, we propose a solution that builds

a tunable sampling distribution that prefers negative pairs whose representations are currently very similar. This solution faces two challenges: (1) we do not have access to any true similarity or dissimilarity information; (2) we need an efficient sampling strategy for this tunable distribution. We overcome (1) by building on ideas from positive-unlabeled learning [Elkan and Noto, 2008, Du Plessis et al., 2014], and (2) by designing an efficient, easy to implement importance sampling technique that incurs no computational overhead.

Our theoretical analysis shows that, as a function of the tuning parameter, the optimal representations for our new method place similar inputs in tight clusters, whilst spacing the clusters as far apart as possible. Empirically, our hard negative sampling strategy improves the downstream task performance for image, graph and text data, supporting that indeed, our negative examples are more informative.

Contributions. In summary, we make the following contributions:

1. We propose a simple distribution over hard negative pairs for contrastive representation learning, and derive a practical importance sampling strategy with zero computational overhead that takes into account the lack of true dissimilarity information;

2. We theoretically analyze the hard negatives objective and optimal representations, showing that they capture desirable generalization properties;

3. We empirically observe that the proposed sampling method improves the downstream task performance on image, graph and text data.

Next we move onto the problem formulation and our approach.

We begin with the setup and the idea of contrastive representation learning. We wish to learn an embedding $f : \mathcal{X} \to \mathbb{S}^{d-1}/t$ that maps an observation x to a point on a hypersphere $\mathbb{S}^{d-1}/t$ in $\mathbb{R}^d$ of radius $1/t$, where t is the "temperature" scaling hyperparameter.

3.2 Mathematical Problem Formulation

Following the setup of Arora et al. [2019], we assume an underlying set of discrete latent classes $\mathcal{C}$ that represent semantic content, so that similar pairs (x, x^+) have the same latent class. Denoting the distribution over latent classes by $\rho(c)$ for $c \in \mathcal{C}$, we define the joint distribution $p_{x,c}(x, c) = p(x|c)\rho(c)$ whose marginal $p(x)$ we refer to simply as p, and assume $\text{supp}(p) = \mathcal{X}$. For simplicity, we assume $\rho(c) = \tau^+$ is uniform, and let $\tau^- = 1 - \tau^+$ be the probability of another class. Since the class-prior τ^+ is unknown in practice, it must either be treated as a hyperparameter, or estimated [Christoffel et al., 2016, Jain et al., 2016].

Let $h : \mathcal{X} \to \mathcal{C}$ be the true underlying hypothesis that assigns class labels to inputs. We write $x \sim x'$ to denote the label equivalence relation $h(x) = h(x')$. We denote by $p_x^+(x') = p(x'|h(x') = h(x))$, the distribution over points with same label as x, and by $p_x^-(x') = p(x'|h(x') \neq h(x))$, the distribution over points with labels different from x. We drop the subscript x when the context is clear. Following the usual convention, we overload '$\sim$' and also write $x \sim p$ to denote a point sampled from p.

For each data point $x \sim p$, the noise-contrastive estimation (NCE) objective [Gutmann and Hyvärinen, 2010] for learning the representation f uses a *positive* example x^+ with the same label as x, and *negative* examples $\{x_i^-\}_{i=1}^N$ with (supposedly) different labels, $h(x_i^-) \neq h(x)$, sampled from q:

$$\mathbb{E}_{\substack{x\sim p,\ \ x^+\sim p_x^+ \\ \{x_i^-\}_{i=1}^N \sim q}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{N} \sum_{i=1}^N e^{f(x)^T f(x_i^-)}} \right]. \tag{3.1}$$

The weighting parameter Q is introduced for the purpose of analysis. When N is finite we take $Q = N$, yielding the usual form of the contrastive objective. The negative sample distribution q is frequently chosen to be the marginal distribution p, or, in practice, an empirical approximation of it [Tian et al., 2019, Chen et al., 2020b,e, He et al., 2020b, Chen et al., 2020e, van den Oord et al., 2018, Henaff, 2020]. In this chapter we ask: is there a better way to choose q?

3.3 Principles for Negative Sampling

In this section we describe our approach for hard negative sampling. We begin by asking *what makes a good negative sample?* To answer this question we adopt the following two guiding principles:

Principle 1. *q should only sample "true negatives" x_i^- whose labels differ from that of the anchor x.*

Principle 2. *The most useful negative samples are ones that the embedding currently believes to be similar to the anchor.*

In short, negative samples that have different label from the anchor, but that are embedded nearby are likely to be most useful and provide significant gradient information during training. In metric learning there is access to true negative pairs, automatically fulfilling the first principle.

In unsupervised contrastive learning there is no supervision, so upholding Principle 1 is impossible to do exactly. In this paper we propose a method that upholds Principle 1 approximately, and simultaneously combines this idea with the key additional conceptual ingredient of "hardness" (encapsulated in Principle 2). The level of "hardness" in our method can be smoothly adjusted, allowing the user to select the hardness that best trades-off between an improved learning signal from hard negatives, and the harm due to the correction of false negatives being only approximate. This important since the hardest points are those closest to the anchor, and are expected to have a high propensity to have the same label. Therefore the damage from the approximation not removing all false negatives becomes larger for harder samples, creating the trade-off. As a special case our our method, when the hardness level is tuned fully down, we obtain the method proposed in [Chuang et al., 2020] that only upholds Principle 1 (approximately) but not Principle 2. Finally, beyond Principles 1 and 2, we wish to design an efficient sampling method that does not add additional computational overhead during training.

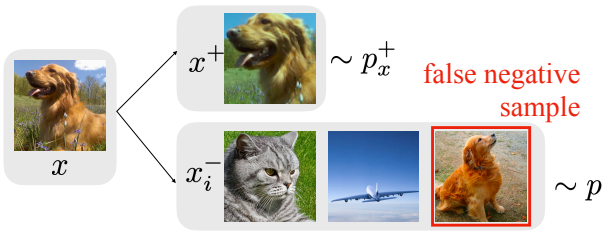


Figure 3-1: **Sampling bias.** The common practice of drawing negative examples x_i^- from the data distribution $p(x)$ may result in x_i^- that are actually similar to x.

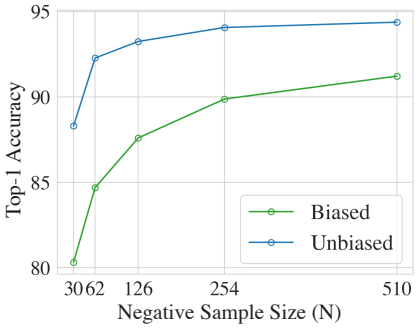


Figure 3-2: **Sampling bias leads to performance drop.** Results on CIFAR-10 for drawing x_i^- from $p(x)$ (biased) and from data with different labels, i.e., truly semantically different data (unbiased).

3.3.1 Principle I: Avoid Sampling False Negatives

Intuitively, the ideal contrastive loss will use positive and negative pairs correspond to the desired latent classes. In other words, negative samples x^- will not belong to the same class as x. Hence, the ideal loss to optimize would be

$$L^N_{\text{Unbiased}}(f) = \mathbb{E}_{\substack{x\sim p, x^+\sim p_x^+ \\ x_i^-\sim p_x^-}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{N}\sum_{i=1}^N e^{f(x)^T f(x_i^-)}} \right], \tag{3.2}$$

which we will refer to as the *unbiased loss*. Here, we introduce a weighting parameter Q for the analysis. When the number N of negative examples is finite, we set $Q = N$, in agreement with the standard contrastive loss. In practice, however, $p_x^-(x_i^-) = p(x_i^-|h(x_i^-) \neq h(x))$ is not accessible. The standard approach is thus to sample negative examples x_i^- from the (unlabeled) $p(x)$ instead. We refer to the resulting loss as the *biased* loss L^N_{Biased}. When drawn from $p(x)$, the sample x_i^- will come from the same class as x with probability τ^+.

Lemma 1 shows that in the limit, the standard InfoNCE loss $\mathcal{L}_{\text{InfoNCE}} = L^N_{\text{Biased}}$—which for the purposes of this section we denote using new notation emphasizing the possibility for false negatives—we upper bounds the ideal, unbiased loss.

Lemma 1. *For any embedding f and finite N, we have*

$$L^N_{\text{Biased}}(f) \geq L^N_{\text{Unbiased}}(f) + \mathbb{E}_{x\sim p}\left[0 \wedge \log \frac{\mathbb{E}_{x^+\sim p^+_x} \exp f(x)^\top f(x^+)}{\mathbb{E}_{x^-\sim p^-_x} \exp f(x)^\top f(x^-)}\right] - e^{3/2}\sqrt{\frac{\pi}{2N}}. \tag{3.3}$$

where $a \wedge b$ denotes the minimum of two real numbers a and b.

Recent works often use large N, e.g., $N = 65536$ in He et al. [2020a], making the last term negligible. While, in general, minimizing an upper bound on a target objective is a reasonable idea, two issues arise here: (1) the smaller the unbiased loss, the larger is the second term, widening the gap; and (2) the empirical results in Figure 3-2 show that minimizing the upper bound L^N_{Biased} and minimizing the ideal loss L^N_{Unbiased} can result in very different learned representations.

Debiased Contrastive Loss Next, we derive a loss that is closer to the ideal L^N_{Unbiased}, while only having access to positive samples and samples from p. Figure 3-2 shows that the resulting embeddings are closer to those learned with L^N_{Unbiased}. We begin by decomposing the data distribution as

$$p(x') = \tau^+ p^+_x(x') + \tau^- p^-_x(x').$$

An immediate approach would be to replace p^-_x in L^N_{Unbiased} with $p^-_x(x') = (p(x') - \tau^+ p^+_x(x'))/\tau^-$ and then use the empirical counterparts for p and p^+_x. The resulting objective can be estimated with samples from only p and p^+_x, but is computationally expensive for large N:

$$\frac{1}{(\tau^-)^N}\sum_{k=0}^{N}\binom{N}{k}(-\tau^+)^k \mathbb{E}_{\substack{x\sim p, x^+\sim p^+_x \\ \{x^-_i\}_{i=1}^k \sim p^+_x \\ \{x^-_i\}_{i=k+1}^N \sim p}}\left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \sum_{i=1}^N e^{f(x)^T f(x^-_i)}}\right], \tag{3.4}$$

where $\{x^-_i\}_{i=k}^j = \varnothing$ if $k > j$. It also demands at least N positive samples. To obtain a more practical form, we consider the asymptotic form as the number N of negative examples goes to infinity.

Lemma 2. *For fixed Q and $N \to \infty$, it holds that*

$$\mathbb{E}_{\substack{x\sim p, x^+\sim p_x^+ \\ \{x_i^-\}_{i=1}^N \sim p_x^{-N}}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{N}\sum_{i=1}^N e^{f(x)^T f(x_i^-)}} \right] \tag{3.5}$$

$$\longrightarrow \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{\tau^-}\left(\mathbb{E}_{x^-\sim p}\left[e^{f(x)^T f(x^-)}\right] - \tau^+ \mathbb{E}_{v\sim p_x^+}\left[e^{f(x)^T f(v)}\right]\right)} \right]. \tag{3.6}$$

The limiting objective (3.6), which we denote by $\widetilde{L}^Q_{\text{Debiased}}$, still samples examples x^- from p, but corrects for that with additional positive samples v. This essentially reweights positive and negative terms in the denominator.

The empirical estimate of $\widetilde{L}^Q_{\text{Debiased}}$ is much easier to compute than the straightforward objective (3.5). With N samples $\{u_i\}_{i=1}^N$ from p and M samples $\{v_i\}_{i=1}^M$ from p_x^+, we estimate the expectation of the second term in the denominator as

$$g(x, \{u_i\}_{i=1}^N, \{v_i\}_{i=1}^M) = \max\left\{\frac{1}{\tau^-}\left(\frac{1}{N}\sum_{i=1}^N e^{f(x)^T f(u_i)} - \tau^+ \frac{1}{M}\sum_{i=1}^M e^{f(x)^T f(v_i)}\right),\ e^{-1/t}\right\}. \tag{3.7}$$

We constrain the estimator g to be greater than its theoretical minimum $e^{-1/t} \leq \mathbb{E}_{x^-\sim p_x^-} e^{f(x)^T f(x_i^-)}$ to prevent calculating the logarithm of a negative number. The resulting population loss with fixed N and M per data point is

$$L^{N,M}_{\text{Debiased}}(f) = \mathbb{E}_{\substack{x\sim p;\, x^+\sim p_x^+ \\ \{u_i\}_{i=1}^N \sim p^N \\ \{v_i\}_{i=1}^N \sim p_x^{+M}}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + N g\left(x, \{u_i\}_{i=1}^N, \{v_i\}_{i=1}^M\right)} \right], \tag{3.8}$$

where, for simplicity, we set Q to the finite N. The class prior τ^+ can be estimated from data [Jain et al., 2016, Christoffel et al., 2016] or treated as a hyperparameter. Theorem 2 bounds the error due to finite N and M as decreasing with rate $\mathcal{O}(N^{-1/2} + M^{-1/2})$.

Theorem 2. *For any embedding f and finite N and M, we have*

$$\left|\widetilde{L}^N_{\text{Debiased}}(f) - L^{N,M}_{\text{Debiased}}(f)\right| \leq \frac{e^{3/2}}{\tau^-}\sqrt{\frac{\pi}{2N}} + \frac{e^{3/2}\tau^+}{\tau^-}\sqrt{\frac{\pi}{2M}}. \tag{3.9}$$

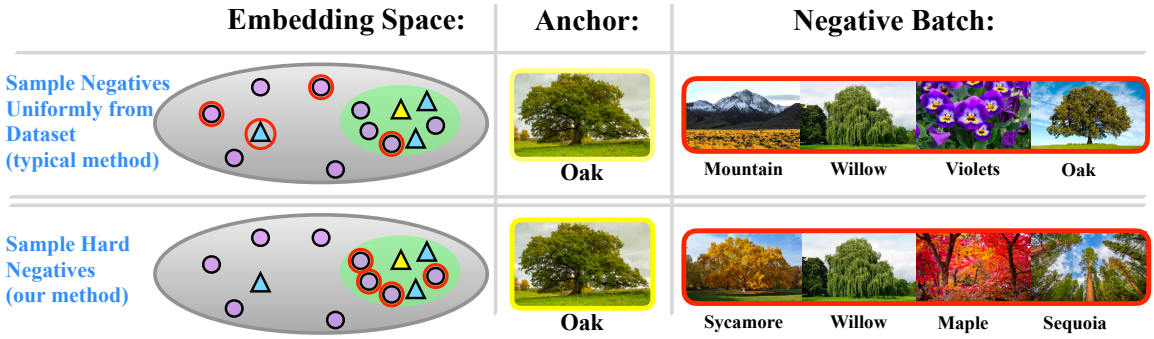


Figure 3-3: Schematic illustration of negative sampling methods for the example of classifying species of tree. Top row: uniformly samples negative examples (red rings); mostly focuses on very different data points from the anchor (yellow triangle), and may even sample examples from the same class (triangles, vs. circles). Bottom row: Hard negative sampling prefers examples that are (incorrectly) close to the anchor.

Empirically, the experiments in Section 3.5 also show that larger N and M consistently lead to better performance. In the implementations, we use a full empirical estimate for $L^{N,M}_{\text{Debiased}}$ that averages the loss over T points x, for finite N and M.

3.3.2 Principle II: Sample Hard, Informative Negatives

Next we move on to incorporating the idea of *hardness* into the debiased loss. Our goal is to design a distribution q on $\mathcal{X}$ that is allowed to depend on the embedding f and the anchor x. From q we sample a batch of negatives $\{x_i^-\}_{i=1}^N$ according to the principles noted above. We propose sampling negatives from the distribution q_β^- defined as

$$q_\beta^-(x^-) := q_\beta(x^-|h(x) \neq h(x^-)), \quad \text{where} \quad q_\beta(x^-) \propto e^{\beta f(x)^\top f(x^-)} \cdot p(x^-),$$

for $\beta \geq 0$. Note that q_β^- and q_β both depend on x, but we suppress the dependance from the notation. The exponential term in q_β is an unnormalized von Mises–Fisher distribution with mean direction $f(x)$ and "concentration parameter" β [Mardia and Jupp, 2000]. There are two key components to q_β^-, corresponding to each principle: 1) conditioning on the event $\{h(x) \neq h(x^-)\}$ which guarantees that (x, x^-) correspond to different latent classes (Principle 1); 2) the concentration parameter β term controls

the degree by which q_β up-weights points x^- that have large inner product (similarity) to the anchor x (Principle 2). Since f lies on the surface of a hypersphere of radius $1/t$, we have $\|f(x) - f(x')\|^2 = 2/t^2 - 2f(x)^\top f(x')$ so preferring points with large inner product is equivalent to preferring points with small squared Euclidean distance.

Geometric Intuition. The hard negatives distribution q_β can be interpreted geometrically. The debiasing, which addresses false negatives, down-weights negative samples that are very close to the anchor embedding $f(x)$, and the hard negative reweighting term down-weights samples that are very far from $f(x)$. The result is that q_β targets samples in a ring surrounding $f(x)$.

Although we have designed q_β^- to have all of the desired components, it is not clear how to sample efficiently from it. To work towards a practical method, note that we can rewrite this distribution by adopting a PU-learning viewpoint [Elkan and Noto, 2008, Du Plessis et al., 2014, Chuang et al., 2020]. That is, by conditioning on the event $\{h(x) = h(x^-)\}$ we can split $q_\beta(x^-)$ as

$$q_\beta(x^-) = \tau^- q_\beta^-(x^-) + \tau^+ q_\beta^+(x^-), \tag{3.10}$$

where $q_\beta^+(x^-) = q_\beta(x^-|h(x) = h(x^-)) \propto e^{\beta f(x)^\top f(x^-)} \cdot p^+(x^-)$. Rearranging (3.10) yields a formula $q_\beta^-(x^-) = \big(q_\beta(x^-) - \tau^+ q_\beta^+(x^-)\big)/\tau^-$ for the negative sampling distribution q_β^- in terms of two distributions that are tractable since we have samples from p and can approximate samples from p^+ using a set of semantics-preserving transformations, as is typical in contrastive learning methods.

It is possible to generate samples from q_β and approximately from q_β^+ using rejection sampling and data augmentations to generate positives. However, rejection sampling involves an algorithmic complication since the procedure for sampling batches must be modified. To avoid this, we instead take an importance sampling approach. To obtain this, first note that fixing the number Q and taking the limit $N \to \infty$ in the objective (3.1) yields,

$$\mathcal{L}(f, q) = \mathbb{E}_{\substack{x \sim p \\ x^+ \sim p_x^+}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + Q \mathbb{E}_{x^- \sim q}[e^{f(x)^T f(x^-)}]} \right]. \tag{3.11}$$

The original objective (3.1) can be viewed as a finite negative sample approximation to $\mathcal{L}(f,q)$ (note implicitly $\mathcal{L}(f,q)$ depends on Q) . Inserting $q = q_\beta^-$ and using the rearrangement of equation (3.10) we obtain the following hardness-biased objective:

$$\mathop{\mathbb{E}}_{\substack{x\sim p\\ x^+\sim p_x^+}}\left[-\log\frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)}+\frac{Q}{\tau^-}(\mathbb{E}_{x^-\sim q_\beta}[e^{f(x)^T f(x^-)}]-\tau^+\mathbb{E}_{v\sim q_\beta^+}[e^{f(x)^T f(v)}])}\right]. \quad (3.12)$$

This objective suggests that we need only to approximate *expectations* $\mathbb{E}_{x^-\sim q_\beta}[e^{f(x)^T f(x^-)}]$ and $\mathbb{E}_{v\sim q_\beta^+}[e^{f(x)^T f(v)}]$ over q_β and q_β^+ (rather than explicily sampling). This can be achieved using classical Monte-Carlo importance sampling techniques using samples from p and p^+ as follows:

$$\mathbb{E}_{x^-\sim q_\beta}[e^{f(x)^T f(x^-)}] = \mathbb{E}_{x^-\sim p}[e^{f(x)^T f(x^-)}q_\beta/p] = \mathbb{E}_{x^-\sim p}[e^{(\beta+1)f(x)^T f(x^-)}/Z_\beta],$$
$$\mathbb{E}_{v\sim q_\beta^+}[e^{f(x)^T f(v)}] = \mathbb{E}_{v\sim p^+}[e^{f(x)^T f(v)}q_\beta^+/p^+] = \mathbb{E}_{v\sim p^+}[e^{(\beta+1)f(x)^T f(v)}/Z_\beta^+],$$

where Z_β, Z_β^+ are the partition functions of q_β and q_β^+ respectively. The right hand terms readily admit empirical approximations by replacing p and p^+ with $\hat{p}(x) = \frac{1}{N}\sum_{i=1}^N \delta_{x_i^-}(x)$ and $\hat{p}^+(x) = \frac{1}{M}\sum_{i=1}^M \delta_{x_i^+}(x)$ respectively (δ_w denotes the Dirac delta function centered at w). The only unknowns left are the partition functions, $Z_\beta = \mathbb{E}_{x^-\sim p}[e^{\beta f(x)^T f(x^-)}]$ and $Z_\beta^+ = \mathbb{E}_{x^+\sim p^+}[e^{\beta f(x)^T f(x^+)}]$ which themselves are expectations over p and p^+ and therefore admit empirical estimates,

$$\widehat{Z}_\beta = \frac{1}{N}\sum_{i=1}^N e^{\beta f(x)^\top f(x_i^-)}, \qquad \widehat{Z}_\beta^+ = \frac{1}{M}\sum_{i=1}^M e^{\beta f(x)^\top f(x_i^+)}.$$

It is important to emphasize the simplicity of the implementation of our proposed approach. Since we propose to reweight the objective instead of modifying the sampling procedure, only two extra lines of code are needed to implement our approach, with no additional computational overhead.

3.4 Theoretical Analysis of Hard Negative Sampling

3.4.1 Hard Sampling Interpolates Between Marginal and Worst-Case Negatives

Intuitively, the concentration parameter β in our proposed negative sample distribution q_β^- controls the level of "hardness" of the negative samples. As discussed earlier, the debiasing method of Chuang et al. [2020] can be recovered as a special case: taking $\beta = 0$ to obtain the distribution q_0^-. This case amounts to correcting for the fact that some samples in a negative batch sampled from p will have the same label as the anchor. But what interpretation does large β admit? Specifically, what does the distribution q_β^- converge to in the limit $\beta \to \infty$, if anything? We show that in the limit q_β^- approximates an inner solution to the following zero-sum two player game.

$$\inf_f \sup_{q \in \Pi} \left\{ \mathcal{L}(f, q) = \mathbb{E}_{\substack{x \sim p \\ x^+ \sim p_x^+}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + Q \mathbb{E}_{x^- \sim q}[e^{f(x)^T f(x^-)}]} \right] \right\}. \tag{3.13}$$

where $\Pi = \{q = q(\cdot; x, f) : \text{supp}\,(q(\cdot; x, f)) \subseteq \{x' \in \mathcal{X} : x' \not\sim x\}, \forall x \in \mathcal{X}\}$ is the set of distributions with support that is disjoint from points with the same class as x (without loss of generality we assume $\{x' \in \mathcal{X} : x' \not\sim x\}$ is non-empty). Since $q = q(\cdot; x, f)$ depends on x and f it can be thought of as a family of distributions. The formal statement is as follows.

Proposition 1. *Let $\mathcal{L}^*(f) = \sup_{q \in \Pi} \mathcal{L}(f, q)$. Then for any $t > 0$ and $f : \mathcal{X} \to \mathbb{S}^{d-1}/t$ we observe the convergence $\mathcal{L}(f, q_\beta^-) \longrightarrow \mathcal{L}^*(f)$ as $\beta \to \infty$.*

Proof. See Appendix A.1.1. □

To develop a better intuitive understanding of the worst case negative distribution objective $\mathcal{L}^*(f) = \sup_{q \in \Pi} \mathcal{L}(f, q)$, we note that the supremum can be characterized analytically. Indeed,

$$\sup_{q\in\Pi}\mathcal{L}(f,q) = -\mathbb{E}_{\substack{x\sim p\\ x^+\sim p_x^+}} f(x)^T f(x^+) + \sup_{q\in\Pi}\mathbb{E}_{\substack{x\sim p\\ x^+\sim p_x^+}} \log\left\{e^{f(x)^T f(x^+)} + Q\mathbb{E}_{x^-\sim q}[e^{f(x)^T f(x^-)}]\right\}$$
$$= -\mathbb{E}_{\substack{x\sim p\\ x^+\sim p_x^+}} f(x)^T f(x^+) + \mathbb{E}_{\substack{x\sim p\\ x^+\sim p_x^+}} \log\left\{e^{f(x)^T f(x^+)} + Q\cdot\sup_{q\in\Pi}\mathbb{E}_{x^-\sim q}[e^{f(x)^T f(x^-)}]\right\}.$$

The supremum over q can be pushed inside the expectation since q is a family of distribution indexed by x, reducing the problem to maximizing $\mathbb{E}_{x^-\sim q}[e^{f(x)^T f(x^-)}]$, which is solved by any q^* whose support is a subset of $\arg\sup_{x^-:x^-\not\sim x} e^{f(x)^T f(x^-)}$ if the supremum is attained. However, computing such points involves maximizing a neural network. Instead of taking this challenging route, using q_β^- defines a lower bound by placing higher probability on x^- for which $f(x)^T f(x^-)$ is large. This lower bound becomes tight as $\beta\to\infty$ (Proposition 1).

3.4.2 Optimal Embeddings on the Hypersphere for Worst-Case Negative Samples

What desirable properties does an optimal contrastive embedding (global minimizer of $\mathcal{L}$) possess that make the representation generalizable? To study this question, we first analyze the distribution of an optimal embedding f^* on the hypersphere when negatives are sampled from the adversarial worst-case distribution. We consider a different limiting viewpoint of objective (3.1) as the number of negative samples $N\to\infty$. Following the formulation of Wang and Isola [2020a] we take $Q=N$ in (3.1), and subtract $\log N$. This changes neither the set of minimizers, nor the geometry of the loss surface. Taking the number of negative samples $N\to\infty$ yields the limiting objective,

$$\mathcal{L}_\infty(f,q) = \mathbb{E}_{\substack{x\sim p\\ x^+\sim p_x^+}}\left[-\log\frac{e^{f(x)^T f(x^+)}}{\mathbb{E}_{x^-\sim q}[e^{f(x)^T f(x^-)}]}\right]. \tag{3.14}$$

Theorem 3. *Suppose the downstream task is classification (i.e. $\mathcal{C}$ is finite), and let $\mathcal{L}_\infty^*(f)=\sup_{q\in\Pi}\mathcal{L}_\infty(f,q)$. The infimum $\inf_{f:\ measurable}\mathcal{L}_\infty^*(f)$ is attained, and any f^* achieving the global minimum is such that $f^*(x)=f^*(x^+)$ almost surely. Furthermore,*

letting $\mathbf{v}_c = f^*(x)$ *for any* x *such that* $h(x) = c$ *(so* $\mathbf{v}_c$ *is well defined up to a set of* x *of measure zero),* f^* *is characterized as being any solution to the following ball-packing problem,*

$$\max_{\{\mathbf{v}_c \in \mathbb{S}^{d-1}/t\}_{c \in \mathcal{C}}} \sum_{c \in \mathcal{C}} \rho(c) \cdot \min_{c' \neq c} \|\mathbf{v}_c - \mathbf{v}_{c'}\|^2. \tag{3.15}$$

Proof. See Appendix A.1.2. □

Interpretation. The first component of the result is that $f^*(x) = f^*(x^+)$ almost surely for an optimal f^*. That is, an optimal embedding f^* must be invariant across pairs of similar inputs x, x^+. The second component is characterizing solutions via the classical geometrical Ball-Packing Problem of Tammes [1930] (Eq. 3.15) that has only been solved exactly for uniform ρ, for specific of $|\mathcal{C}|$ and typically for $\mathbb{S}^2$ [Schütte and Van der Waerden, 1951, Musin and Tarasov, 2015, Tammes, 1930]. When the distribution ρ over classes is uniform this problem is solved by a set of $|\mathcal{C}|$ points on the hypersphere such that the average squared-ℓ_2 distance from a point to the nearest other point is as large as possible. In other words, suppose we wish to place $|\mathcal{C}|$ number of balls[1] on $\mathbb{S}^{d-1}$ so that they do not intersect. Then solutions to Tammes' Problem (3.15) expresses (twice) the largest possible average squared radius that the balls can have. So, we have a ball-packing problem where instead of trying to pack as many balls as possible of a fixed size, we aim to pack a fixed number of balls (one for each class) to have as big radii as possible. Non-uniform ρ adds importance weights to each fixed ball. In summary, solutions of the problem $\min_f \mathcal{L}^*_\infty(f)$ are a maximum margin clustering.

This understanding of global minimizers of $\mathcal{L}^*_\infty(f) = \sup_{q \in \Pi} \mathcal{L}_\infty(f, q)$ can further developed into a better understanding of generalization on downstream tasks. The next result shows that representations that achieve small excess risk on the objective $\mathcal{L}^*_\infty$ still separate clusters well in the sense that a simple 1-nearest neighbor classifier achieves low classification error.

[1]For a manifold $\mathcal{M} \subseteq \mathbb{R}^d$, we say $C \subset \mathcal{M}$ is a ball if it is connected, and there exists a Euclidean ball $\mathcal{B} = \{x \in \mathbb{R}^d : \|x\|_2 \leq R\}$ for which $C = \mathcal{M} \cap \mathcal{B}$.

Theorem 4. *Suppose ρ is uniform on $\mathcal{C}$ and f is such that $\mathcal{L}^*_\infty(f) - \inf_{\bar{f}\ measurable} \mathcal{L}^*_\infty(\bar{f}) \leq \varepsilon$ with $\varepsilon \leq 1$. Let $\{\mathbf{v}^*_c \in \mathbb{S}^{d-1}/t\}_{c\in\mathcal{C}}$ be a solution to Problem 3.15, and define the constant $\xi = \min_{c,c^-:c\neq c^-} \|\mathbf{v}^*_c - \mathbf{v}^*_{c-}\| > 0$. Then there exists a set of vectors $\{\mathbf{v}_c \in \mathbb{S}^{d-1}/t\}_{c\in\mathcal{C}}$ such that the 1-nearest neighbor classifier $\hat{h}(x) = \arg\min_{\bar{c}\in\mathcal{C}} \|f(x) - \mathbf{v}_{\bar{c}}\|$ (ties broken arbitrarily) achieves misclassification risk,*

$$\mathbb{P}_{x,c}(\hat{h}(x) \neq c) \leq \frac{8\varepsilon}{(\xi^2 - 2\,|\mathcal{C}|\,(1 + 1/t)\varepsilon^{1/2})^2}$$

Proof. See Appendix A.1.3. □

In particular, $\mathbb{P}(\hat{h}(x) \neq c) = \mathcal{O}(\varepsilon)$ as $\varepsilon \to 0$, and in the limit $\varepsilon \to 0$ we recover the invariance claim of Theorem 3 as a special case. The result can be generalized to arbitrary ρ by replacing $|\mathcal{C}|$ in the bound by $1/\min_c \rho(c)$. The result also implies that it is possible to build simple classifiers for tasks that involve only a subset of classes from $\mathcal{C}$, or classes that are a union of classes from $\mathcal{C}$. The constant $\xi = \min_{c,c^-:c\neq c^-} \|\mathbf{v}^*_c - \mathbf{v}^*_{c-}\| > 0$ is a purely geometrical property of spheres, and describes the minimum separation distance between a set of points that solves the Tammes' ball-packing problem.

3.4.3 Generalization Bounds for False Negatives Loss (Principle I)

For the final part of this theoretical section we return to the debiased contrastive objective, without hard negatives. We connect this loss to a corresponding supervised loss, and show how our contrastive learning approach leads to a generalization bound for downstream supervised learning tasks.

We consider a supervised classification task $\mathcal{T}$ with K classes $\{c_1, \ldots, c_K\} \subseteq \mathcal{C}$. After contrastive representation learning, we fix the representations $f(x)$ and then train a linear classifier $q(x) = Wf(x)$ on task $\mathcal{T}$ with the standard multiclass softmax

cross entropy loss $L_{\text{Softmax}}(\mathcal{T}, q)$. Hence, we define the supervised loss for f as

$$L_{\text{Sup}}(\mathcal{T}, f) = \inf_{W \in \mathbb{R}^{K \times d}} L_{\text{Softmax}}(\mathcal{T}, Wf). \tag{3.16}$$

In line with the approach of Arora et al. [2019] we analyze the supervised loss of a mean classifier [Snell et al., 2017], where for each class c, the rows of W are set to the mean of the representations $\mu_c = \mathbb{E}_{x \sim p(\cdot|c)}[f(x)]$. We will use $L^{\mu}_{\text{Sup}}(\mathcal{T}, f)$ as shorthand for its loss. Note that $L^{\mu}_{\text{Sup}}(\mathcal{T}, f)$ is always an upper bound on $L_{\text{Sup}}(\mathcal{T}, f)$. To allow for uncertainty about the task $\mathcal{T}$, we will bound the average supervised loss for a uniform distribution $\mathcal{D}$ over K-way classification tasks with classes in $\mathcal{C}$.

$$L_{\text{Sup}}(f) = \mathbb{E}_{\mathcal{T} \sim \mathcal{D}} L_{\text{Sup}}(\mathcal{T}, f). \tag{3.17}$$

We begin by showing that the asymptotic unbiased contrastive loss is an upper bound on the supervised loss of the mean classifier.

Lemma 3. *For any embedding f, whenever $N \geq K - 1$ we have*

$$L_{\text{Sup}}(f) \leq L^{\mu}_{\text{Sup}}(f) \leq \widetilde{L}^{N}_{\text{Debiased}}(f).$$

Lemma 3 uses the asymptotic version of the debiased loss. Together with Theorem 2 and a concentration of measure result, it leads to a generalization bound for debiased contrastive learning, as we show next.

Generalization Bound. In practice, we use an empirical estimate $\widehat{L}^{N,M}_{\text{Debiased}}$, i.e., an average over T data points x, with M positive and N negative samples for each x. Our algorithm learns an empirical risk minimizer $\hat{f} \in \arg\min_{f \in \mathcal{F}} \widehat{L}^{N,M}_{\text{Debiased}}(f)$ from a function class $\mathcal{F}$. The generalization depends on the *empirical Rademacher complexity* $\mathcal{R}_{\mathcal{S}}(\mathcal{F})$ of $\mathcal{F}$ with respect to our data sample $\mathcal{S} = \{x_j, x_j^+, \{u_{i,j}\}_{i=1}^N, \{v_{i,j}\}_{i=1}^M\}_{j=1}^T$. Let $f_{|\mathcal{S}} = (f_k(x_j), f_k(x_j^+), \{f_k(u_{i,j})\}_{i=1}^N, \{f_k(v_{i,j})\}_{i=1}^M)_{j \in [T], k \in [d]} \in \mathbb{R}^{(N+M+2)dT}$ be the

restriction of f onto $\mathcal{S}$, using $[T] = \{1, \ldots, T\}$. Then $\mathcal{R}_{\mathcal{S}}(\mathcal{F})$ is defined as

$$\mathcal{R}_{\mathcal{S}}(\mathcal{F}) := \mathbb{E}_{\sigma} \sup_{f \in \mathcal{F}} \langle \sigma, f_{|\mathcal{S}} \rangle \tag{3.18}$$

where $\sigma \sim \{\pm 1\}^{(N+M+1)dT}$ are Rademacher random variables. Combining Theorem 2 and Lemma 3 with a concentration of measure argument yields the final generalization bound for debiased contrastive learning.

Theorem 5. *With probability at least* $1 - \delta$*, for all* $f \in \mathcal{F}$ *and* $N \geq K - 1$*,*

$$L_{\text{Sup}}(\hat{f}) \leq L^{N,M}_{\text{Debiased}}(f) + \mathcal{O}\left(\frac{1}{\tau^-} \sqrt{\frac{1}{N}} + \frac{\tau^+}{\tau^-} \sqrt{\frac{1}{M}} + \frac{\lambda \mathcal{R}_{\mathcal{S}}(\mathcal{F})}{T} + B \sqrt{\frac{\log \frac{1}{\delta}}{T}} \right) \tag{3.19}$$

where $\lambda = \sqrt{\frac{1}{(\tau^-)^2}(\frac{M}{N} + 1) + (\tau^+)^2(\frac{N}{M} + 1)}$ *and* $B = \log N \left(\frac{1}{\tau^-} + \tau^+ \right)$.

The bound states that if the function class $\mathcal{F}$ is sufficiently rich to contain some embedding for which $L^{N,M}_{\text{Debiased}}$ is small, then the representation encoder $\hat{f}$, learned from a large enough dataset, will perform well on the downstream classification task. The bound also highlights the role of the positive and unlabeled sample sizes M and N in the objective function, in line with the observation that a larger number of negative/positive examples in the objective leads to better results [He et al., 2020a, Chen et al., 2020b]. The last two terms in the bound grow slowly with N, but the effect of this on the generalization error is small if the dataset size T is much larger than N and M, as is commonly the case. The dependence on on N and T in Theorem 5 is roughly equivalent to the result in [Arora et al., 2019], but the two bounds are not directly comparable since the proof strategies differ.

3.5 Empirical Results

Next, we evaluate our hard negative sampling method empirically, and apply it as a modification to state-of-the-art contrastive methods on image, graph, and text data. For all experiments β is treated as a hyper-parameter (see ablations in Fig. 3-4 for

more understanding of how to pick β). Values for M and τ^+ must also be determined. We fix $M = 1$ for all experiments, since taking $M > 1$ would increase the number of inputs for the forward-backward pass. Lemma 7 in the appendix gives a theoretical justification for the choice of $M = 1$. Choosing the class-prior τ^+ can be done in two ways: estimating it from data [Christoffel et al., 2016, Jain et al., 2016], or treating it as a hyper-parameter. The first option requires the possession of labeled data *before* contrastive training.

3.5.1 Image Representations

We begin by testing the hard sampling method on vision tasks using the STL10, CIFAR100 and CIFAR10 data. We use SimCLR [Chen et al., 2020b] as the baseline method, and all models are trained for 400 epochs. The results in Fig. 3-4 show consistent improvement over SimCLR ($q = p$) and the particular case of our method with $\beta = 0$ proposed in [Chuang et al., 2020] (called debiasing) on STL10 and CIFAR100. For $N = 510$ negative examples per data point we observe absolute improvements of 3% and 7.3% over SimCLR on CIFAR100 and STL10 respectively, and absolute improvements over the best debiased baseline of 1.9% and 3.2%. On tinyImageNet (Tab. 3.1) we observe an absolute improvement of 3.6% over SimCLR, while on CIFAR10 there is a slight improvement for smaller N, which disappears at larger N. See Appendix A.3.1 results using MoCo-v2 for large negative batch size, and Appendix A.4.1 for full setup details.

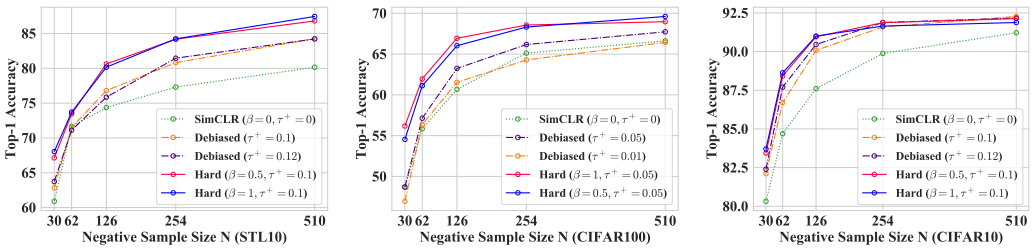


Figure 3-4: **Classification accuracy on downstream tasks.** Embeddings trained using hard, debiased, and standard ($\beta = 0, \tau^+ = 0$) versions of SimCLR, and evaluated using linear readout accuracy.

SimCLR	Debiased	Hard ($\beta = 1$)
53.4%	53.7%	57.0%

Table 3.1: Top-1 linear readout on tinyImageNet. Class prior is set to $\tau^+ = 0.01$.

3.5.2 Graph Representations

Second, we consider hard negative sampling in the context of learning graph representations. We use the state-of-the-art InfoGraph method introduced by Sun et al. [2020] as the baseline, which is suitable for downstream graph-level classification. The objective is of a slightly different form from the NCE loss. Because of this we use a generalization of the formulation presented in Section 3.3 (See Appendix A.2 for details). In doing so, we illustrate that it is easy to adapt our hard sampling method to other contrastive frameworks.

Fig. 3-5 shows the results of fine-tuning an SVM [Boser et al., 1992, Cortes and Vapnik, 1995] on the fixed, learned embedding for a range of different values of β. Hard sampling does as well as InfoGraph in all cases, and better in 6 out of 8 cases. For ENZYMES and REDDIT, hard negative samples improve the accuracy by 3.2% and 2.4%, respectively, for DD and PTC by $1 - 2\%$, and for IMDB-B and MUTAG by at least 0.5%. Usually, multiple different choices of $\beta > 0$ were competitive with the InfoGraph baseline: 17 out of the 24 values of $\beta > 0$ tried (across all 8 datasets) achieve accuracy as high or better than InfoGraph ($\beta = 0$).

3.5.3 Sentence Representations

Third, we test hard negative sampling on learning representations of sentences using the *quick-thoughts* (QT) vectors framework introduced by Logeswaran and Lee [2018], which uses adjacent sentences (before/after) as positive samples. Embeddings are trained using the unlabeled BookCorpus dataset [Kiros et al., 2015], and evaluated following the protocol of Logeswaran and Lee [2018] on six downstream tasks. The results are reported in Table 3.2. Hard sampling outperforms or equals the QT baseline in 5 out of 6 cases, the debiased baseline [Chuang et al., 2020] in 4 out of 6, and both

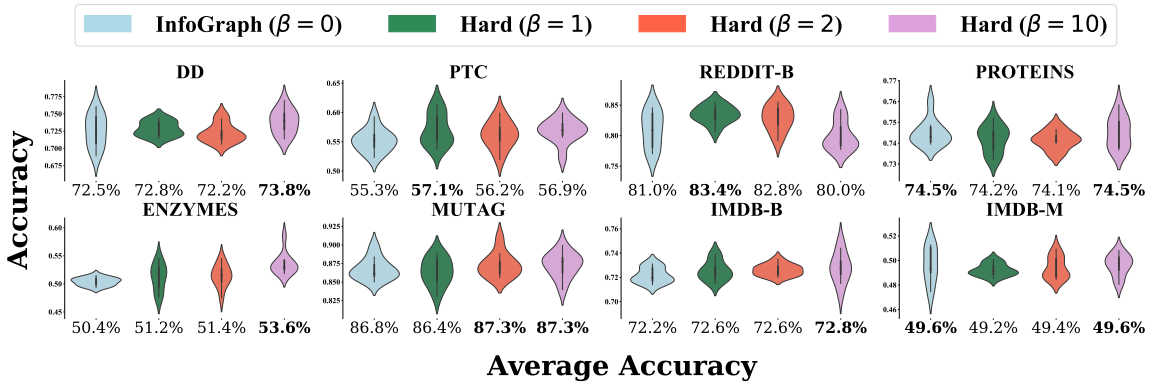


Figure 3-5: **Classification accuracy on downstream tasks.** We compare graph representations on four classification tasks. Accuracies are obtained by fine-tuning an SVM readout function, and are the average of 10 runs, each using 10-fold cross validation. Results in **bold** indicate best performer.

in 3 out of 6 cases. Setting $\tau^+ > 0$ led to numerical issues in optimization for hard sampling.

Objective	MR	CR	SUBJ	MPQA	TREC	MSRP (Acc)	MSRP (F1)
QT ($\beta = 0, \tau^+ = 0$)	76.8	81.3	86.6	93.4	**89.8**	73.6	81.8
Debiased ($\tau^+ = 0.01$)	76.2	82.9	86.9	**93.7**	89.1	**74.7**	**82.7**
Hard ($\beta = 1, \tau^+ = 0$)	77.1	82.5	**87.0**	92.9	89.2	73.9	82.2
Hard ($\beta = 2, \tau^+ = 0$)	**77.4**	**83.6**	86.8	93.4	88.7	73.5	82.0

Table 3.2: **Classification accuracy on downstream tasks.** Sentence representations are learned using quick-thoughts (QT) vectors on the BookCorpus dataset and evaluated on six classification tasks. Evaluation of binary classification tasks (MR, CR, SUBJ, MPQA) uses 10-fold cross validation.

3.6 Ablations: A Closer Look at Hard Negatives

3.6.1 Are Harder Samples Necessarily Better?

By setting β to large values, one can focus on only the hardest samples in a training batch. But is this desirable? Fig. 3-6 (left, middle) shows that for vision problems, taking larger β does not necessarily lead to better representations. In contrast, when one uses true positive pairs during training (green curve, uses label information for

positive but not negative pairs), the downstream performance monotonically increases with β until convergence (Fig. 3-6 , middle). Interestingly, this is achieved without using label information for the negative pairs. This observation suggests an explanation for why bigger β hurts performance in practice. Debiasing (conditioning on the event $\{h(x) \neq h(x^-)\}$) using the true p^+ corrects for sampling x^- with the same label as x. However, since in practice we approximate p^+ using a set of data transformations, we can only partially correct. This is harmful for large β since this regime strongly prefers x^- for which $f(x^-)$ is close to $f(x)$, many of whom will have the same label as x if not corrected for. We note also that by annealing β (gradually decreasing β to 0 throughout training; see Appendix A.4.1 for details) it is possible to be more robust to the choice of initial β, with marginal impact on downstream accuracy compared to the best fixed value of β.

3.6.2 Does Avoiding False Negatives Improve Hard Sampling?

Our proposed hard negative sampling method conditions on the event $\{h(x) \neq h(x^-)\}$ in order to avoid false negatives (termed "debiasing" [Chuang et al., 2020]). But does this help? To test this, we train four embeddings: hard sampling with and without debiasing, and uniform sampling ($\beta = 0$) with and without debiasing. The results in Fig. 3-6 (right) show that hard sampling with debiasing obtains the highest linear readout accuracy on STL10, only using hard sampling or only debiasing yields (in this case) similar accuracy. All improve over the SimCLR baseline.

Fig. 3-7 compares the histograms of cosine similarities of positive and negative pairs for the four learned representations. The representation trained with hard negatives and debiasing assigns much lower similarity score to a pair of negative samples than other methods. On the other hand, the SimCLR baseline assigns higher cosine similarity scores to pairs of positive samples. However, to discriminate positive and negative pairs, a key property is the amount of *overlap* of positive and negative histograms. Our hard sampling method achieves less overlap than SimCLR, by better trading off higher dissimilarity of negative pairs with less similarity of positive pairs. Similar tradeoffs are observed for the debiased objective, and hard sampling without

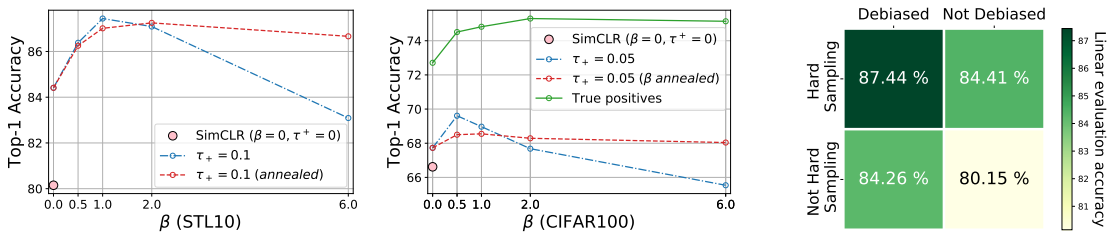


Figure 3-6: Left: the effect of varying concentration parameter β on linear readout accuracy. Middle: linear readout accuracy as concentration parameter β varyies, in the case of contrastive learning (fully unsupervised), using true positive samples (uses label information), and an annealing method that improves robustness to the choice of β (see Appendix A.4.1 for details). Right: STL10 linear readout accuracy for hard sampling with and without debiasing, and non-hard sampling ($\beta = 0$) with and without debiasing. Best results come from using both simultaneously.

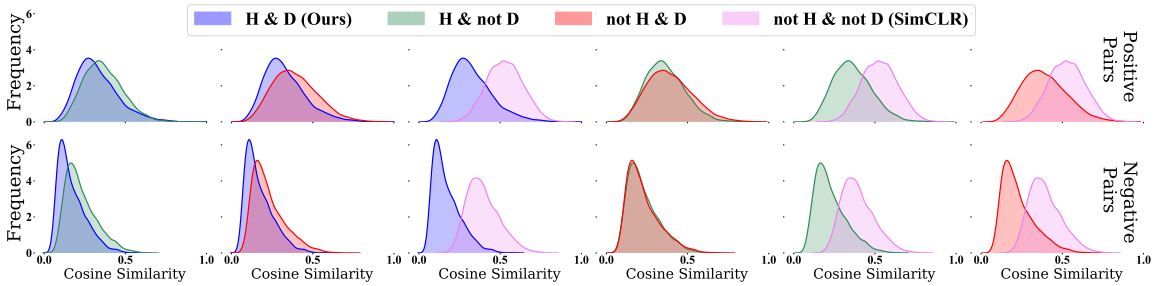


Figure 3-7: Histograms of cosine similarity of pairs of points with the same label (top) and different labels (bottom) for embeddings trained on STL10 with four different objectives. H=Hard Sampling, D=Debiasing. Histograms overlaid pairwise to allow for convenient comparison.

debiasing.

3.6.3 Qualitative Study of Hard Negatives

We compare hard negative samples to uniformly samples negatives in Fig. 3-11. The top row selects the 10 images with highest inner product with anchor in latent space from a batch of 128 inputs. The bottom row displays a set of random samples from the same batch. Hard negatives are semantically much more similar to the anchor than uniformly sampled negatives - hard negatives possess many similar characteristics to the anchor, including texture, colors, animals vs machinery.

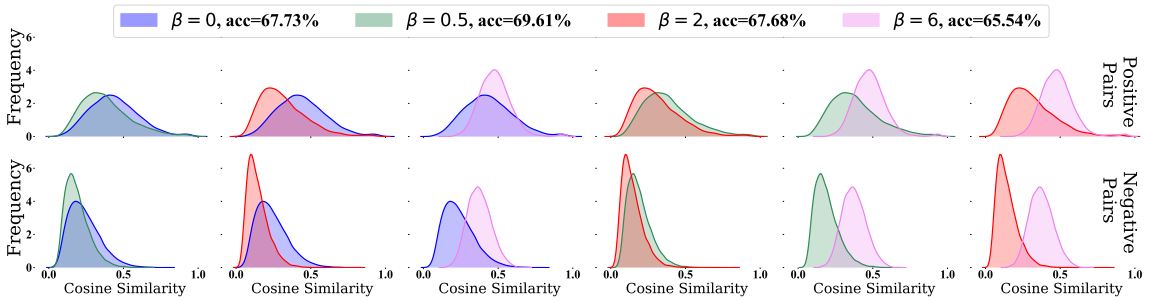


Figure 3-8: Histograms of cosine similarity of pairs of points with different label (bottom) and same label (top) for embeddings trained on CIFAR100 with different values of β. Histograms overlaid pairwise to allow for easy comparison.

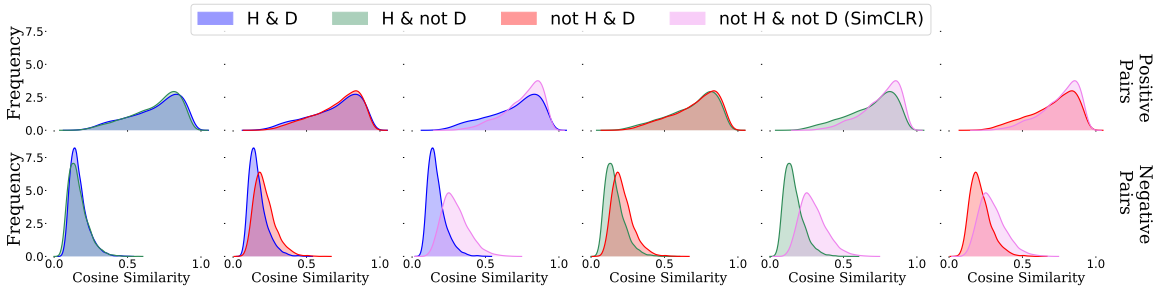


Figure 3-9: Histograms of cosine similarity of pairs of points with the same label (top) and different labels (bottom) for embeddings trained on CIFAR10 with four different objectives. H=Hard Sampling, D=Debiasing. Histograms overlaid pairwise to allow for convenient comparison.

3.6.4 How do Hard Negatives Affect Optimization?

Fig. 3-12 shows the performance on STL10 and CIFAR100 of SimCLR versus using hard negatives throughout training. We use weighted k-nearest neighbors with $k = 200$ as the classifier and evaluate each model once every five epochs. Hard sampling with $\beta = 1$ leads to much faster training: on STL10 hard sampling takes only 60 epochs to reach the same performance as SimCLR does in 400 epochs. On CIFAR100 hard sampling takes only 125 epochs to reach the same performance as SimCLR does in 400 epochs. We speculate that the speedup is, in part, due to hard negatives providing non-negligible gradient information during training.

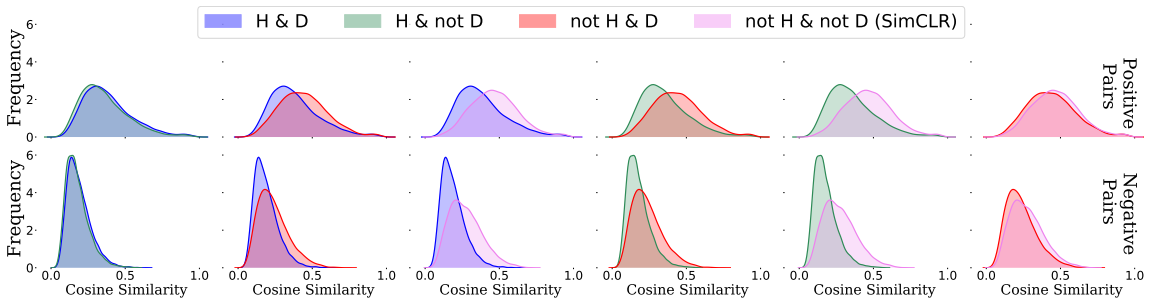


Figure 3-10: Histograms of cosine similarity of pairs of points with the same label (top) and different labels (bottom) for embeddings trained on CIFAR100 with four different objectives. H=Hard Sampling, D=Debiasing. Histograms overlaid pairwise to allow for convenient comparison.

Figure 3-11: Qualitative comparison of hard negatives and uniformly sampled negatives for embedding trained on STL10 for 400 epochs using SimCLR.

3.6.5 Visualizing Embedding Space for Debiased Loss

We consider the debiasing method alone, without hard negatives. Figure 3-13 shows t-SNE visualizations of the representations learned by the biased and debiased objectives ($N = 256$) on CIFAR10. The debiased contrastive loss leads to better class separation than the contrastive loss, and the result is closer to that of the ideal, unbiased loss.

3.7 Discussion of Related Work

Contrastive Representation Learning. Various frameworks for contrastive learning of visual representations have been proposed, including SimCLR [Chen et al., 2020b,d], which uses augmented views of other items in a minibatch as negative samples, and MoCo [He et al., 2020b, Chen et al., 2020e], which uses a momentum updated memory bank of old negative representations to enable the use of very large batches of negative samples. Most contrastive methods are unsupervised, however there exist some that use label information [Sylvain et al., 2020, Khosla et al., 2020].

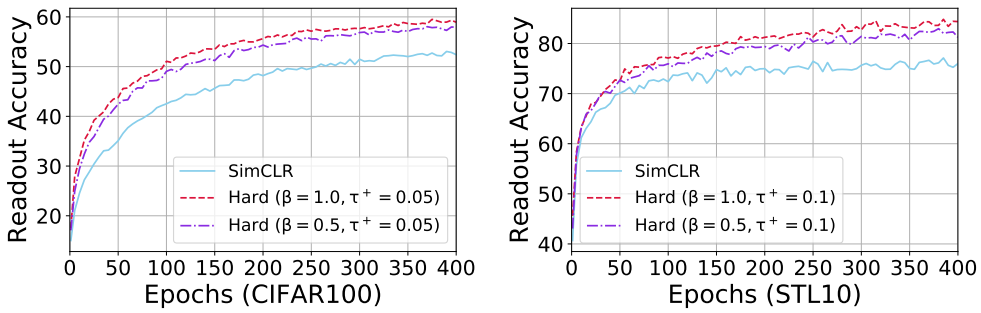


Figure 3-12: Hard sampling takes much fewer epochs to reach the same accuracy as SimCLR does in 400 epochs; for STL10 with $\beta = 1$ it takes only 60 epochs, and on CIFAR100 it takes 125 epochs (also with $\beta = 1$).

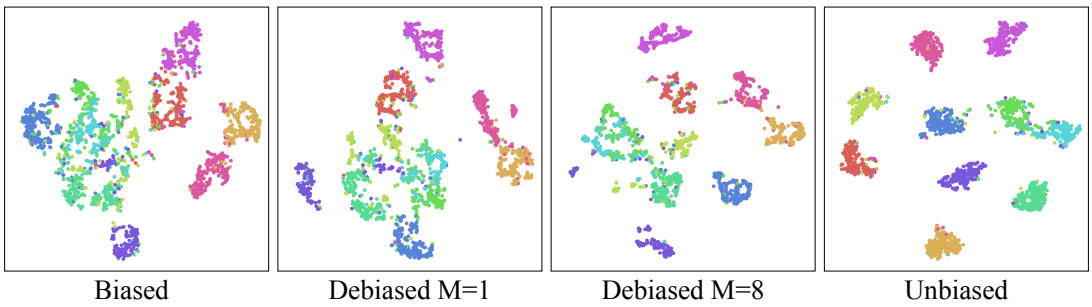


Figure 3-13: **t-SNE visualization of learned representations on CIFAR10.** Classes are indicated by colors. The debiased objective ($\tau^+ = 0.1$) leads to better data clustering than the (standard) biased loss; its effect is closer to the supervised unbiased objective.

Many works study the role of positive pairs, and, e.g., propose to apply large perturbations for images Chen et al. [2020b,e], or argue to minimize the mutual information within positive pairs, apart from relevant information for the ultimate prediction task [Tian et al., 2020b]. Beyond visual data, contrastive methods have been developed for sentence embeddings [Logeswaran and Lee, 2018], sequential data [van den Oord et al., 2018, Henaff, 2020], graph [Sun et al., 2020, Hassani and Khasahmadi, 2020, Li et al., 2019b] and node representation learning [Velickovic et al., 2019], and learning representations from raw images for off-policy control [Srinivas et al., 2020]. The role of negative pairs has been much less studied. Recent work by Kalantidis et al. [2020] consider applying Mixup [Zhang et al., 2018a] to generate hard negatives in latent space, and Jin et al. [2018] exploit the specific temporal structure of video to generate

negatives for object detection.

Negative Mining in Deep Metric Learning. As opposed to the contrastive representation learning literature, selection strategies for negative samples have been thoroughly studied in (deep) metric learning [Schroff et al., 2015, Song et al., 2016, Harwood et al., 2017, Wu et al., 2017, Ge, 2018, Suh et al., 2019]. Most of these works observe that it is helpful to use negative samples that are difficult for the current embedding to discriminate. Schroff et al. [2015] qualify this, observing that some examples are simply too hard, and propose selecting "semi-hard" negative samples. The well known importance of negative samples in metric learning, where (partial) true dissimilarity information is available, raises the question of negative samples in contrastive learning, the subject of this paper.

Positive-unlabeled Learning. For the debiased loss, we approximate the contrastive loss with only unlabeled data from $p(x)$ and positive examples, our work is also related to *Positive-Unlabeled* (PU) learning, i.e., learning from only positive (P) and unlabeled (U) data. Common applications of PU learning are retrieval or outlier detection [Elkan and Noto, 2008, Du Plessis et al., 2014, 2015]. Our approach is related to *unbiased PU learning*, where the unlabeled data is used as negative examples, but down-weighted appropriately Kiryo et al. [2017], Du Plessis et al. [2014, 2015]. While these works focus on zero-one losses, we here address the contrastive loss, where existing PU estimators are not directly applicable.

Chapter 4

Understanding Shortcuts in Contrastive Learning

How do we know if a contrastive-trained model will learn features that are able to solve a newly encountered tasks of interest? Since contrastive leanrning does not directly train for extracting features suited to a particularly downstream task it is non-obvious what these models actually learn. In this chapter we explore the factors driving *which* features contrastive models learn.

To begin, we observe that the contrastive loss does not always sufficiently guide which features are extracted, a behavior that can negatively impact the performance on downstream tasks via "shortcuts", i.e., by inadvertently suppressing important predictive features. We find that feature extraction is influenced by the *difficulty* of the so-called instance discrimination task (i.e., the task of discriminating pairs of similar points from pairs of dissimilar ones).

Although harder pairs improve the representation of some features, the improvement comes at the cost of suppressing previously well represented features. In response, we propose *implicit feature modification* (IFM), a method for altering positive and negative samples in order to guide contrastive models towards capturing a wider variety of predictive features. The principle underlying IFM is the following: for an embeddings, and identify the direction in embedding space that would most hurt the pretraining, and move the point in this direction. In doing so we rforbit the model

from using the features currently being used to solve the contrastive task, and ask it to learn different features instead. By leveraging the geometry of representation space to change which data features the model can use, we observe that IFM reduces feature suppression, and as a result improves performance on vision and medical imaging tasks.

Acknowledgements. This chapter is based on [Robinson et al., 2021b], which is in collaboration with Li Sun, Ke Yu, Kayhan Batmanghelich, Stefanie Jegelka, and Suvrit Sra. Many of the experimental result in this work were developed in close collaboration with Li and Ke.

4.1 Background and Motivation

The contrastive learning pretraining task forces the learned mode to extract features capable of distinguishing a pair of data augmented inputs from random samples fro the trainign data.

In other words, learning features that are discriminative during training does not guarantee a model will generalize. Many studies find inductive biases in supervised learning toward *simple* "shortcut" features and decision rules [Hermann and Lampinen, 2020, Huh et al., 2021, Nguyen et al., 2021] which result in unpredictable model behavior under perturbations [Ilyas et al., 2019, Szegedy et al., 2014] and failure outside the training distribution [Beery et al., 2018, Recht et al., 2019]. Simplicity bias has various potential sources [Geirhos et al., 2020] including training methods [Chizat and Bach, 2020, Lyu and Li, 2020, Soudry et al., 2018] and architecture design [Geirhos et al., 2019, Hermann et al., 2019]. Bias towards shortcut decision rules also hampers transferability in contrastive learning [Chen and Li, 2020], where it is in addition influenced by the instance discrimination task. These difficulties lead us to ask: can the contrastive instance discrimination task itself be modified to avoid learning shortcut solutions?

We approach this question by studying the relation between contrastive instance discrimination and feature learning. First, we theoretically explain why optimizing

the InfoNCE loss alone does not guarantee avoidance of shortcut solutions that *suppress* (i.e., discard) certain input features [Chen and Li, 2020, Geirhos et al., 2020]. Second, despite this negative result, we show that it is still possible to trade off representation of one feature for another using simple methods for adjusting the difficulty of instance discrimination. However, these methods have an important drawback: improved learning of one feature often comes at the cost of harming another. That is, feature suppression is still prevalent. In response, we propose *implicit feature modification*, a technique that encourages encoders to discriminate instances using multiple input features. Our method introduces no computational overhead, reduces feature suppression (without trade-offs), and improves generalization on various downstream tasks.

Contributions. In summary, this chapter makes the following main contributions:

1. It analyzes feature suppression in contrastive learning, and explains why feature suppression can occur when optimizing the InfoNCE loss.
2. It studies the relation between instance discrimination tasks and feature learning; concretely, adjustments to instance discrimination difficulty leads to different features being learned.
3. It proposes *implicit feature modification*, a simple and efficient method that reduces the tendency to use feature suppressing shortcut solutions and improves generalization.

4.2 Feature Suppression in Contrastive Learning

Feature suppression refers to the phenomenon where, in the presence of multiple predictive input features, a model uses only a subset of them and ignores the others. The selected subset often corresponds to intuitively "simpler" features, e.g., color as opposed to shape. Such features lead to "shortcut" decision rules that might perform well on training data, but can harm generalization and lead to poor robustness to data shifts. Feature suppression has been identified as a common problem in deep learning

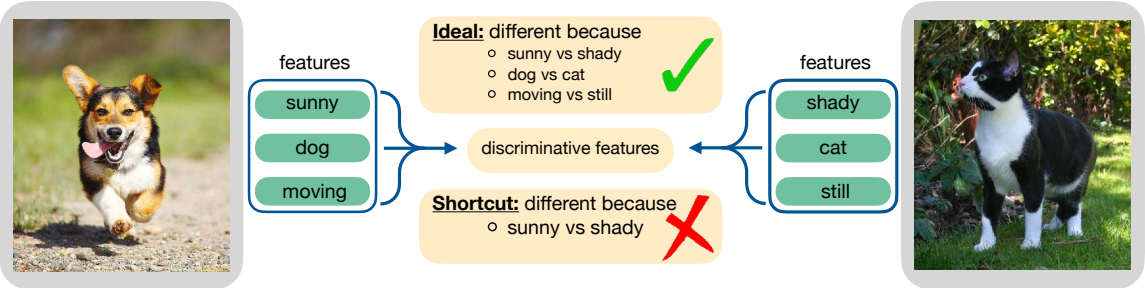


Figure 4-1: An ideal encoder would discriminate between instances using multiple distinguishing features instead of finding simple shortcuts that suppress features. We show that InfoNCE-trained encoders can suppress features (Sec. 4.2.2). However, making instance discrimination harder during training can trade off representation of different features (Sec. 4.2.3). To avoid the need for trade-offs we propose *implicit feature modification* (Sec. 4.3), which reduces suppression in general, and improves generalization (Sec. 4.4).

[Geirhos et al., 2020], and both supervised and contrastive learning suffer from biases induced by the choice of optimizer and architecture. However, contrastive learning bears an additional potential source of bias: *the choice of instance discrimination task*. Which positive and negative pairs are presented critically affects which features are discriminative, and hence which features are learned. In this work we study the relation between feature suppression and instance discrimination.

First, we explain why optimizing the InfoNCE loss is insufficient in general to avoid feature suppression, and show how it can lead to counter-intuitive generalization (Sec. 4.2.2). Given this negative result, we then ask if it is at least possible to *control* which features a contrastive encoder learns? We find that this is indeed the case, and that adjustments to the instance discrimination task lead to different features being learned (Sec. 4.2.3). However, the primary drawback of these adjustments is that improving one feature often comes at the cost of harming representation of another. That is, feature suppression is still prevalent. Addressing this drawback is the focus of Sec. 4.3.

4.2.1 A Formal Definition of Feature Suppression

Formally, we assume that the data has underlying feature spaces $\mathcal{Z}^1, \ldots, \mathcal{Z}^n$ with a distribution p_j on each $\mathcal{Z}^j$. Each $j \in [n]$, corresponding to a latent space $\mathcal{Z}^j$, models a distinct feature. We write the product as $\mathcal{Z}^S = \prod_{j \in S} \mathcal{Z}^j$, and simply write $\mathcal{Z}$ instead

of $\mathcal{Z}^{[n]}$ where $[n] = \{1, \ldots, n\}$. A set of features $z = (z^j)_{j\in[n]} \in \mathcal{Z}$ is generated by sampling each coordinate $z^j \in \mathcal{Z}^j$ independently, and we denote the measure on $\mathcal{Z}$ induced by z by λ. Further, let $\lambda(\cdot|z^S)$ denote the conditional measure on $\mathcal{Z}$ for fixed z^S. For $S \subseteq [n]$ we use z^S to denote the projection of z onto $\mathcal{Z}^S$. Finally, an injective map $g : \mathcal{Z} \to \mathcal{X}$ produces observations $x = g(z)$.

Our aim is to train an encoder $f : \mathcal{X} \to \mathbb{S}^{d-1}$ to map input data x to the surface of the unit sphere $\mathbb{S}^{d-1} = \{u \in \mathbb{R}^d : \|u\|_2 = 1\}$ in such a way that f extracts useful information. To formally define feature suppression, we need the *pushforward* $h\#\nu(V) = \nu(h^{-1}(V))$ of a measure ν on a space $\mathcal{U}$ for a measurable map $h : \mathcal{U} \to \mathcal{V}$ and measurable $V \subseteq \mathcal{V}$, where $h^{-1}(V)$ denotes the preimage.

Definition 1. *Consider an encoder $f : \mathcal{X} \to \mathbb{S}^{d-1}$ and features $S \subseteq [n]$. For each $z^S \in \mathcal{Z}^S$, let $\mu(\cdot|z^S) = (f \circ g)\#\lambda(\cdot|z^S)$ be the pushforward measure on $\mathbb{S}^{d-1}$ by $f \circ g$ of the conditional $\lambda(\cdot|z^S)$.*

1. *f* suppresses *S if for any pair $z^S, \bar{z}^S \in \mathcal{Z}^S$, we have $\mu(\cdot|z^S) = \mu(\cdot|\bar{z}^S)$.*
2. *f* distinguishes *S if for any pair of distinct $z^S, \bar{z}^S \in \mathcal{Z}^S$, measures $\mu(\cdot|z^S), \mu(\cdot|\bar{z}^S)$ have disjoint support.*

Feature suppression is thus captured in a distributional manner, stating that S is suppressed if the encoder distributes inputs in a way that is invariant to the value z^S. Distinguishing features, meanwhile, asks that the encoder f separates points with different features z^S into disjoint regions. We consider training an encoder $f : \mathcal{X} \to \mathbb{S}^{d-1}$ to optimize the InfoNCE loss [van den Oord et al., 2018, Gutmann and Hyvärinen, 2010],

$$\mathcal{L}_m(f) = \mathbb{E}_{x,x^+,\{x_i^-\}_{i=1}^m}\left[-\log \frac{e^{f(x)^\top f(x^+)/\tau}}{e^{f(x)^\top f(x^+)/\tau} + \sum_{i=1}^m e^{f(x)^\top f(x_i^-)/\tau}}\right], \tag{4.1}$$

where τ is known as the *temperature*. Positive pairs x, x^+ are generated by first sampling $z \sim \lambda$, then independently sampling two random augmentations $a, a^+ \sim \mathcal{A}$, $a : \mathcal{X} \to \mathcal{X}$ from a distribution $\mathcal{A}$, and setting $x = a(g(z))$ and $x^+ = a^+(g(z))$. We assume $\mathcal{A}$ samples the identity function $a(x) = x$ with non-zero probability ("x is

similar to itself"), and that there are no collisions: $a(x) \neq a'(x')$ for all a, a', and all $x \neq x'$. Each negative example x_i^- is generated as $x_i^- = a_i(g(z_i))$, by independently sampling features $z_i \sim \lambda$ and an augmentation $a_i \sim \mathcal{A}$.

4.2.2 Why Feature Suppression Occurs in Contrastive Learning

Do optimal solutions to the InfoNCE loss automatically avoid shortcut solutions? Unfortunately, as we show in this section, this is not the case in general; there exist both optimal solutions of the InfoNCE loss that do and solutions that do not suppress a given feature. Following previous work [Robinson et al., 2021a, Wang and Isola, 2020b, Zimmermann et al., 2021], we analyze the loss as the number of negatives goes to infinity,

$$\mathcal{L} = \lim_{m\to\infty} \left\{\mathcal{L}_m(f) - \log m - \tfrac{2}{\tau}\right\} = \tfrac{1}{2\tau}\mathbb{E}_{x,x^+}\|f(x) - f(x^+)\|^2 + \mathbb{E}_{x^+} \log\left[\mathbb{E}_{x^-} e^{f(x^+)^\top f(x^-)/\tau}\right].$$

We subtract $\log m$ to ensure the limit is finite, and use x^- to denote a random sample with the same distribution as x_i^-. Prop. 2 shows that, assuming the marginals p_j are uniform, the InfoNCE loss is optimized both by encoders that suppress feature j, and by encoders that distinguish j.

Proposition 2. *Suppose that p_j is uniform on $\mathcal{Z}^j = \mathbb{S}^{d-1}$ for all $j \in [n]$. Then for any feature $j \in [n]$ there exists an encoder f_{supp} that suppresses feature j and encoder f_{disc} that discriminates j but both attain $\min_{f:\ measurable} \mathcal{L}(f)$.*

Proof. The existence of the encoders f_{supp} and f_{disc} is demonstrated by constructing explicit examples. Before defining f_{supp} and f_{disc} themselves, we begin by constructing a family $\{f^k\}_{k\in[n]}$ of optimal encoders.

Since g is injective, we know there exists a left inverse $h : \mathcal{X} \to \mathcal{Z}$ such that $h \circ g(z) = z$ for all $z \in \mathcal{Z}$. For any $k \in [n]$ let $\Pi^k : \mathcal{Z} \to \mathbb{S}^{d-1}$ denote the projection $\Pi^k(z) = z^k$. Since p_k is uniform on the sphere $\mathbb{S}^{d-1}$, we know that $\Pi^k \circ h \circ g(z) = z^j$ is uniformly distributed on $\mathbb{S}^{d-1}$. Next we partition the space $\mathcal{X}$. Since we assume that for all $a \neq a'$ and $z \neq z'$ that $a(z) \neq a'(z')$, the family $\{\mathcal{X}_z\}_{z\in\mathcal{Z}}$ where $\mathcal{X}_z = \{a \circ g(z) : z \in \mathcal{Z}\}$ is guaranteed to be a partition (and in particular, disjoint). We may therefore

define an encoder $f_k : \mathcal{X} \to \mathbb{S}^{d-1}$ to be equal to $f_k(x) = \Pi^k \circ h \circ g(z) = z^k$ for all $x \in \mathcal{X}_z$.

First we check that this f_k is optimal. Since for any z, and any $a \sim \mathcal{A}$, by definition we have $a \circ g(z) \in \mathcal{X}_z$, we have that $f_k(x) = f_k(a(x))$ almost surely, so $\mathcal{L}_{\text{align}}(f_k) = 0$ is minimized. To show f_k minimizes $\mathcal{L}_{\text{unif}}$ note that the uniformity loss can be re-written as

$$\begin{aligned}
\mathcal{L}_{\text{unif}}(f_k) &= \int_a \int_z \log \int_{a^-} \int_{z^-} e^{f_k \circ a(g(z))^\top f_k \circ a^-(g(z^-))/\tau} \lambda(\mathrm{d}z)\lambda(\mathrm{d}z^-)\mathcal{A}(\mathrm{d}a)\mathcal{A}(\mathrm{d}a^-) \\
&= \int_z \log \int_{z^-} e^{f_k \circ g(z)^\top f_k \circ g(z^-)/\tau} \lambda(\mathrm{d}z)\lambda(\mathrm{d}z^-) \\
&= \int_{\mathbb{S}^{d-1}} \log \int_{\mathbb{S}^{d-1}} e^{u^\top v/\tau} \mu(\mathrm{d}u)\mu(\mathrm{d}v)
\end{aligned}$$

where $\mu = f_k \circ g \# \lambda$ is the pushforward measure on $\mathbb{S}^{d-1}$, and the second equality follows from the fact that $\mathcal{L}_{\text{align}}(f_k) = 0$. Theorem 1 of [Wang and Isola, 2020b] establishes that the operator,

$$\mu \mapsto \int_{\mathbb{S}^{d-1}} \log \int_{\mathbb{S}^{d-1}} e^{u^\top v/\tau} \mu(\mathrm{d}u)\mu(\mathrm{d}v)$$

is minimized over the space of Borel measures on $\mathbb{S}^{d-1}$ if and only if $\mu = \sigma_d$, the uniform distribution on $\mathbb{S}^{d-1}$, as long as such an f exists. However, since by construction $f_k(x) = \Pi^k \circ h \circ g(z) = z^k$ is uniformly distributed on $\mathbb{S}^{d-1}$, we know that $(f_k \circ g)\#\lambda = \sigma_d$, and hence that f_k minimizes $\mathcal{L}_{\text{align}}$ and $\mathcal{L}_{\text{unif}}$ and hence also the sum $\mathcal{L} = \mathcal{L}_{\text{align}} + \mathcal{L}_{\text{unif}}$.

Recall that we seek encoder f_{supp} that suppress feature j, and f_{disc} that distinguishes feature j. We have a family $\{f^k\}_{k \in [n]}$ that are optimal, and select the two encoders we seen from this collection. First, for f_{supp} define $f_{\text{supp}} = f^k$ for any $k \neq j$. Then by construction $f_{\text{supp}}(x) = z^k$ (where $x \in \mathcal{X}_z$) depends only on z^k, which is independent of z^j. Due to independence, we therefore know that for any pair $z^j, \bar{z}^j \in \mathcal{Z}^j$, we have $\mu(\cdot|z^j) = \mu(\cdot|\bar{z}^j)$, i.e., that f_{supp} is optimal but suppresses feature j. Similarly, simply define $f_{\text{disc}} = f^j$. So $f_{\text{disc}}(x) = z^j$ where $x \in \mathcal{X}_z$, and for any $z^j, \bar{z}^j \in \mathcal{Z}^j$ with $z^j \neq \bar{z}^j$

the pushforwards $\mu(\cdot|z^j), \mu(\cdot|\bar{z}^j)$ are the Dirac measures $\delta_{z^j}, \delta_{\bar{z}^j}$, which are disjoint. □

The condition that p_j is uniformly distributed on $\mathcal{Z}^j = \mathbb{S}^{d-1}$ is similar to conditions used in previous work [Zimmermann et al., 2021]. Prop. 2 shows that empirical observations of feature suppression [Chen and Li, 2020] (see also Fig. 4-3) are not simply due to a failure to sufficiently optimize the loss, but that the possibility of feature suppression is *built into* the loss. What does Prop. 2 imply for the generalization behavior of encoders? Besides explaining why feature suppression can occur, Prop. 2 also suggests another counter-intuitive possibility: *lower InfoNCE loss may actually lead to worse performance on some tasks.*

To empirically study whether this possibility manifests in practice, we use two datasets with known semantic features: (1) In the Trifeature data, [Hermann and Lampinen, 2020] each image is 128×128 and has three features: color, shape, and texture, each taking possible 10 values. See Fig. B-1, App. B.2 for sample images. (2) In the STL-digits data, samples combine MNIST digits and STL10 objects by placing copies of a randomly selected MNIST digit on top of an STL10 image. See Fig. B-2 App. B.2 for sample images.

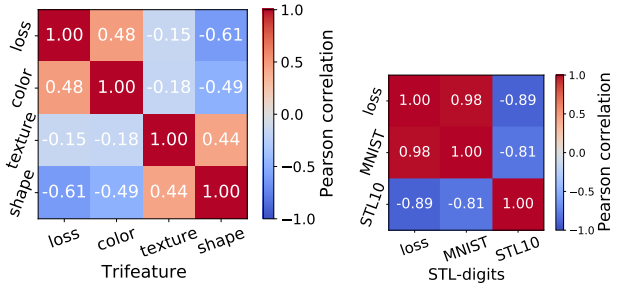


Figure 4-2: Linear readout error on different downstream tasks can be negatively correlated. Further, lower InfoNCE loss does not always yield not lower error: error rates on texture, shape and STL10 prediction are *negatively correlated* with InfoNCE loss.

We train encoders with ResNet-18 backbone using SimCLR [Chen et al., 2020b]. To study correlations between the loss value and error on downstream tasks, we train 33 encoders on Trifeature and 7 encoders on STL-digits with different hyperparameter settings (see App. B.2.2 for full details on training and hyperparameters). For Trifeature, we compute the Pearson correlation between InfoNCE loss and linear readout error when predicting {color, shape, texture}. Likewise, for STL-digits we compute correlations between the InfoNCE loss and MNIST and STL10 prediction error.

Fig. 4-2 shows that performance on different downstream tasks is not always positively correlated. For Trifeature, color error is negatively correlated with shape and texture, while for STL-digits there is a strong negative correlation between MNIST digit error and STL10 error. Importantly, lower InfoNCE loss is correlated with lower prediction error for color and MNIST-digit, but with *larger* error for shape, texture and STL10. Hence, lower InfoNCE loss can improve representation of some features (color, MNIST digit), but may actually *hurt* others. This conflict is likely due to the simpler color and MNIST digit features being used as shortcuts. Our observation is an important addition to the statement of [Wang and Isola, 2020b] that lower InfoNCE loss improves generalization: the situation is more subtle – whether lower InfoNCE helps generalization on a task depends on the use of shortcuts.

4.2.3 Controlling Feature Learning via the Difficulty of Instance Discrimination

The previous section showed that the InfoNCE objective has solutions that suppress features. Next, we ask what factors determine which features are suppressed? Is there a way to target *specific* features and ensure they are encoded? One idea is to use *harder* positive and negative examples. Hard examples are precisely those that are not easily distinguishable using the currently extracted features. So, a focus on hard examples may change the scope of the captured features. To test this hypothesis, we consider two methods for adjusting the difficulty of positive and negative samples:

1. Temperature τ in the InfoNCE loss (Eqn. 4.1). Smaller τ places higher importance on positive an negative pairs with high similarity [Wang and Liu, 2021].
2. Hard negative sampling method of Robinson et al. [Robinson et al., 2021a], which uses importance sampling to sample harder negatives. The method introduces a hardness concentration parameter β, with larger β corresponding to harder negatives (see [Robinson et al., 2021a] for full details).

Results reported in Fig. 4-3 (also Fig. B-4 in App. B.2.2) show that varying instance

discrimination difficulty—i.e., varying temperature τ or hardness concentration β—enables trade-offs between which features are represented. On Trifeature, easier instance discrimination (large τ, small β) yields good performance on 'color'—an "easy" feature for which a randomly initialized encoder already has high linear readout accuracy—while generalization on the harder texture and shape features is poor. The situation *reverses* for harder instance discrimination (small τ, large β). We hypothesize that the use of "easy" features with easy instance discrimination is analogous to simplicity biases in supervised deep networks [Hermann et al., 2019, Huh et al., 2021]. As with supervised learning [Geirhos et al., 2019, Hermann et al., 2019], we observe a bias for texture over shape in convolutional networks, with texture prediction always outperforming shape.

That there are simple levers for controlling which features are learned already distinguishes contrastive learning from supervised learning, where attaining such control is less easy (though efforts have been made [Jacobsen et al., 2018]). However, these results show that representation of one feature must be sacrificed in exchange for learning another one better. To understand how to develop methods for improving feature representation without suppressing others, the next result examines more closely *why* there is a relationship between (hard) instance discrimination tasks and feature learning.

Proposition 3 (Informal)**.** *Suppose that p_j is uniform on $\mathcal{Z}^j = \mathbb{S}^{d-1}$ for all $j \in [n]$. Further, for $S \subseteq [n]$ suppose that $x, x^+, \{x_i^-\}_i$ are conditioned on the event that they have the same features S. Then any f that minimizes the (limiting) InfoNCE loss suppresses features S.*

The formal version of the result is as follows.

Proposition 4. *For a set $S \subseteq [n]$ of features let*

$$\mathcal{L}_S(f) = \mathcal{L}_{align}(f) + \mathbb{E}_{x^+}\big[-\log \mathbb{E}_{x^-}[e^{f(x^+)^\top f(x^-)} | z^S = z^{S-}]\big]$$

denote the (limiting) InfoNCE conditioned on x^+, x^- having the same features S.

Suppose that p_j is uniform on $\mathcal{Z}^j = \mathbb{S}^{d-1}$ for all $j \in [n]$. Then the infimum $\inf \mathcal{L}_S$ is attained, and every $f \in \min_{f'} \mathcal{L}_S(f')$ suppresses features S almost surely.

Proof. By Prop 3, we know that for each z^S there is a measurable f such that $\mathcal{L}_{\text{align}}(f) = 0$ and f achieves perfect uniformity $(f \circ g)\#\lambda(\cdot|z^S) = \sigma_d$ conditioned on z^S. So consider such an f. Since $\mathcal{L}_{\text{align}}(f) = 0$ we may write,

$$\begin{aligned}
\mathcal{L}_S(f) &= \mathbb{E}_{x^+}\big[-\log \mathbb{E}_{x^-}[e^{f(x^+)^\top f(x^-)}|z^S = z^{S-}]\big] \\
&= \mathbb{E}_{z^S}\mathbb{E}_{z^{S-}}\big[-\log \mathbb{E}_{z^-}[e^{f\circ g(z)^\top f\circ g(z^-)}|z^S = z^{S-}]\big] \\
&= \mathbb{E}_{z^S}\mathcal{L}(f; z^S).
\end{aligned}$$

Where we have introduced the conditional loss function

$$\mathcal{L}(f; z^S) = \mathbb{E}_{z^{S-}}\big[-\log \mathbb{E}_{z^-}[e^{f\circ g(z)^\top f\circ g(z^-)}|z^S = z^{S-}]\big]$$

We shall show that any minimizer f of $\mathcal{L}_S$ is such that f minimizes $\mathcal{L}(f; z^S)$ for all values of z^S. To show this notice that $\min_f \mathcal{L}_S(f) = \min_f \mathbb{E}_{z^S}\mathcal{L}(f; z^S) \geq \mathbb{E}_{z^S} \min_f \mathcal{L}(f; z^S)$ and if there is an f such that f minimizes $\mathcal{L}(f; z^S)$ for each z^S then the inequality is tight. So we make it our goal to show that there is an f such that f minimizes $\mathcal{L}(f; z^S)$ for each z^S.

For fixed z^S, by assumption there is an f_{z^S} such that $(f_{z^S} \circ g)\#\lambda(\cdot|z^S) = \sigma_d$. That is, f_{z^S} achieves perfect uniformity given z^S. Theorem 1 of Wang and Isola [Wang and Isola, 2020b] implies that f_{z^S} must minimize $\mathcal{L}(f; z^S)$. Given $\{f_{z^S}\}_{z^S}$ we construct an $f : \mathcal{X} \to \mathbb{S}^{d-1}_\tau$ that minimizes $\mathcal{L}(f; z^S)$ *for all* z^S. By injectivity of g we may partition $\mathcal{X}$ into pieces $\bigcup_{z^S \in \mathcal{Z}^S} \mathcal{X}_{z^S}$ where $\mathcal{X}_{z^S} = \{x : x = g((z^S, z^{S^c})) \text{ for some } z^{S^c} \in \mathcal{Z}^{S^c}\}$. So we may simply define f on domain $\mathcal{X}$ as follows: $f(x) = f_{z^S}(x)$ if $x \in \mathcal{X}_{z^S}$.

This construction allows us to conclude that the minimum of $\mathcal{L}_S$ is attained, and any minimizer f of $\mathcal{L}_S$ also minimizes $\mathcal{L}(f; z^S)$ for each z^S. By Theorem 1 of Wang and Isola [Wang and Isola, 2020b] any such f is such that $(f_{z^S} \circ g)\#\lambda(\cdot|z^S) = \sigma_d$ for all z^S, which immediately implies that f suppresses features S. □

The positive and negative instances in Prop. 3 must be distinguished with features

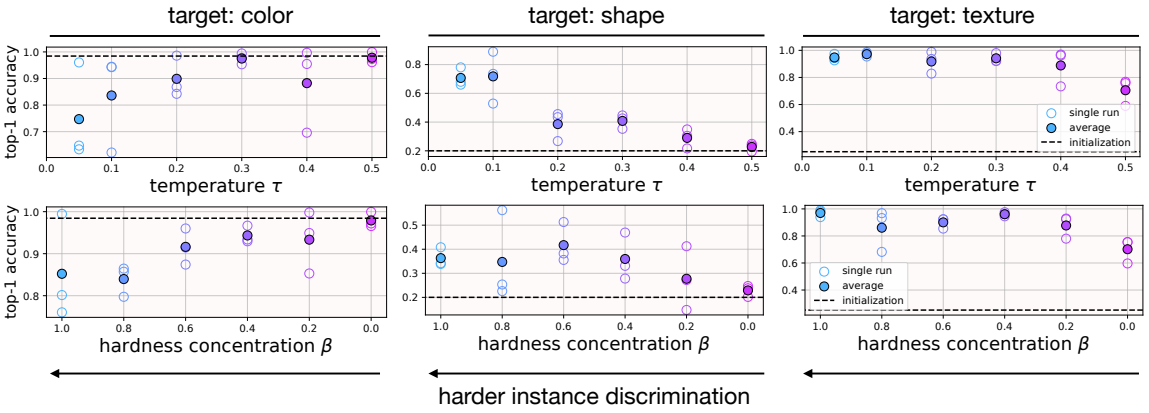


Figure 4-3: Trifeature dataset [Hermann and Lampinen, 2020]. The *difficulty* of instance discrimination affects which features are learned (Sec. 4.2.3). When instance discrimination is easy (big τ, small β), encoders represent color well and other features badly. When instance discrimination is hard (small τ, big β), encoders represent more challenging shape and texture features well, at the expense of color.

in S^c. Relating this point to the above observations, assume that an encoder exclusively uses features S. Any positives and negatives that do not (much) differ in features S are difficult for the encoder. By Prop. 3, focusing the training on these difficult examples pushes the encoder to instead use features in S^c, i.e., to learn *new* features. But at the same time, the proposition also says that a strong focus on such hard negative pairs leads to suppressing the originally used features S, explaining the results in Fig. 4-3. While the two techniques for adjusting instance difficulty we studied were unable to avoid feature suppression, this insight forms the motivation for *implicit feature modification*, which we introduce next.

4.3 Implicit Feature Modification: A Method for Reducing Feature Suppression

The previous section found that simple adjustments to instance discrimination *difficulty* could significantly alter which features a model learns. Prop. 3 suggests that this ability to modify which features are learned stems from holding features constant across positive and negative samples. However, these methods were unable to avoid trade-offs in feature representation (Fig. 4-3) since features that are held constant are

themselves suppressed (Prop. 3).

To avoid this effect, we develop a technique that *adaptively* modifies samples to remove whichever features are used to discriminate a particular positive pair from negatives, then trains an encoder to discriminate instances using *both* the original features, and the features left over after modification. While a natural method for modifying features is to directly transform raw input data, it is very challenging to modify the semantics of an input in this way. So instead we propose modifying features by applying transformations to encoded samples $v = f(x)$. Since we modify the encoded samples, instead of raw inputs x, we describe our method as *implicit*.

We set up our notation. Given batch $x, x^+, \{x_i^-\}_{i=1}^m$ we write $v = f(x)$, $v^+ = f(x^+)$, and $v_i^- = f(x_i^-)$ to denote the corresponding embeddings. As in Eqn. 4.1, the point-wise InfoNCE loss is,

$$\ell(v, v^+, \{v_i^-\}_{i=1}^m) = -\log \frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}}.$$

Definition 2 (Implicit feature modification)**.** *Given budget* $\boldsymbol{\varepsilon} \in \mathbb{R}_+^m$*, and encoder* $f : \mathcal{X} \to \mathbb{S}^d$*, an adversary removes features from* f *that discriminates batch* $x, x^+, \{x_i^-\}_{i=1}^m$ *by maximizing the point-wise InfoNCE loss,* $\ell_{\boldsymbol{\varepsilon}}(v, v^+, \{v_i^-\}_{i=1}^m) = \max_{\delta^+ \in \mathcal{B}_{\varepsilon^+}, \{\delta_i^- \in \mathcal{B}_{\varepsilon_i}\}_{i=1}^m} \ell(v, v^+ + \delta^+, \{v_i^- + \delta_i^-\}_{i=1}^m)$.

Here $\mathcal{B}_\varepsilon$ denotes the ℓ_2-ball of radius ε. Implicit feature modification (IFM) removes components of the current representations that are used to discriminate positive and negative pairs. In other words, the embeddings of positive and negative samples are modified to remove well represented features. So, if the encoder is currently using a simple shortcut solution, IFM removes the features used, thereby encouraging the encoder to also discriminate instances using other features. By applying perturbations in the embedding space IFM can modify high level semantic features (see Fig. 4-4), which is extremely challenging when applying perturbations in input space. In order to learn new features using the perturbed loss while still learning potentially complementary information using the original InfoNCE objective,

we propose optimizing the the multi-task objective $\min_f\{\mathcal{L}(f) + \alpha\mathcal{L}_{\boldsymbol{\varepsilon}}(f)\}/2$ where $\mathcal{L}_{\boldsymbol{\varepsilon}} = \mathbb{E}\ell_{\boldsymbol{\varepsilon}}$ is the adversarial perturbed loss, and $\mathcal{L}$ the standard InfoNCE loss. For simplicity, all experiments set the balancing parameter $\alpha = 1$ unless explicitly noted, and all take $\varepsilon^+, \varepsilon_i^-$ to be equal, and denote this single value by ε. Crucially, $\ell_{\boldsymbol{\varepsilon}}$ can be computed analytically and efficiently.

Lemma 4. *For any $v, v^+, \{v_i^-\}_{i=1}^m \in \mathbb{R}^d$ we have,*

$$\nabla_{v_j^-}\ell = \frac{e^{v^\top v_j^-/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} \cdot \frac{v}{\tau} \quad \textit{and} \quad \nabla_{v^+}\ell = \left(\frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} - 1\right) \cdot \frac{v}{\tau}.$$

In particular, $\nabla_{v_j^-}\ell \propto v$ and $\nabla_{v^+}\ell \propto -v$.

This expression shows that the adversary perturbs v_j^- (resp. v^+) in the direction of the anchor v (resp $-v$). Since the derivative directions are *independent* of $\{v_i^-\}_{i=1}^m$ and v^+, we can analytically compute optimal perturbations in $\mathcal{B}_\varepsilon$. Indeed, following the constant ascent direction shows the optimal updates are simply $v_i^- \leftarrow v_i^- + \varepsilon_i v$ and $v^+ \leftarrow v^+ - \varepsilon^+ v$. The positive (resp. negative) perturbations increase (resp. decrease) cosine similarity to the anchor $\mathrm{sim}(v, v_i^- + \varepsilon_i v) \to 1$ as $\varepsilon_i \to \infty$ (resp. $\mathrm{sim}(v, v^+ - \varepsilon^+ v) \to -1$ as $\varepsilon^+ \to \infty$). In Fig. 4-4 we visualize the newly synthesized v_i^-, v^+ and find meaningful interpolation of semantics. Plugging the update rules for v^+ and v_i^- into the point-wise InfoNCE loss yields,

$$\ell_{\boldsymbol{\varepsilon}}(v, v^+, \{v_i^-\}_{i=1}^m) = -\log \frac{e^{(v^\top v^+ - \varepsilon^+)/\tau}}{e^{(v^\top v^+ - \varepsilon^+)/\tau} + \sum_{i=1}^m e^{(v^\top v_i^- + \varepsilon_i)/\tau}}. \tag{4.2}$$

In other words, IFM amounts to simply perturbing the logits – reduce the positive logit by ε^+/τ and increase negative logits by ε_i/τ. From this we see that $\ell_{\boldsymbol{\varepsilon}}$ is automatically symmetrized in the positive samples: perturbing v instead of v^+ results in the exact same objective. Eqn. 4.2 shows that IFM re-weights each negative sample by a factor $e^{\varepsilon_i/\tau}$ and positive samples by $e^{-\varepsilon^+/\tau}$.

4.3.1 Visualizing Implicit Feature Modification

With implicit feature modification, newly synthesized data points do not directly correspond to any "true" input data point. However it is still possible to visualize the effects of implicit feature modification. To do this, assume access to a memory bank of input data $\mathcal{M} = \{x_i\}_i$. A newly synthesized sample s can be approximately visualized by retrieving the 1-nearest neighbour using cosine similarity $\arg\min_{x \in \mathcal{M}} \text{sim}(s, f(x))$ and viewing the image x as an approximation to s.

Fig. 4-4 shows results using a ResNet-50 encoder trained using MoCo-v2 on ImageNet1K using the training set as the memory bank. For positive pair v, v^+ increasing ε causes the semantics of v and v^+ to diverge. For $\varepsilon = 0.1$ a different car with similar pose and color is generated, for $\varepsilon = 0.2$ the pose and color then changes, and finally for $\varepsilon = 1$ the pose, color and type of vehicle changes. For negative pair v, v^- the reverse occurs. For $\varepsilon = 0.1$, v^- is a vehicle with similar characteristics (number of windows, color etc.), and with $\varepsilon = 0.2$, the pose of the vehicle v^+ aligns with v. Finally for $\varepsilon = 1$ the pose and color of the perturbed negative sample become aligned to the anchor v. In summary, implicit feature modification successfully *modifies the feature content in positive and negative samples*, thereby altering which features can be used to discriminate instances.

Related Work. Several works consider adversarial contrastive learning [Ho and Nvasconcelos, 2020, Jiang et al., 2020, Kim et al., 2020] using PGD (e.g. FGSM) attacks to alter samples in input space. Unlike our approach, PGD-based attacks require costly inner-loop optimization. Other work takes an adversarial viewpoint in input space for other self-supervised tasks e.g., rotations and jigsaws but uses an image-to-image network to simulate FGSM/PGD attacks [Minderer et al., 2020], introducing comparable computation overheads. They note that low-level (i.e., pixel-level) shortcuts can be avoided using their method. All of these works differ from ours by applying attacks in input space, thereby focusing on lower-level features, whereas ours aims to modify high-level features. Fig. 4-5 compares IFM to this family of input-space adversarial methods by comparing to a top performing method ACL(DS)

[Jiang et al., 2020]. We find that ACL improves robust accuracy under ℓ_∞-attack on input space (see [Jiang et al., 2020] for protocol details), whereas IFM improves standard accuracy (full details and discussion in Appdx. B.2.3). Synthesizing harder negatives in latent space using Mixup [Zhang et al., 2018a] has also been considered [Kalantidis et al., 2020] but does not take an adversarial perspective. Other work, AdCo [Hu et al., 2020a], also takes an adversarial viewpoint in latent space. There are several differences to our approach. AdCo perturbs all negatives using the same weighted combination of all the queries, whereas IFM perturbations are query specific. In other words, IFM makes instance discrimination harder point-wise, whereas AdCo perturbation makes the InfoNCE loss larger *on average* (see Fig. 4-4 for visualizations of instance dependent perturbation using IFM). AdCo also treats the negatives as learnable parameters, introducing $\sim 1M$ more parameters and $\sim 7\%$ computational overhead, while IFM has no computational overhead and is implemented with only two lines of code (see Tab. 4.1 for empirical comparison). Finally, no previous work makes the connection between suppression of semantic features and adversarial methods in contrastive learning (see Fig. 4-6).

4.4 Experimental results

Implicit feature modification (IFM) can be used with any InfoNCE-based contrastive framework, and we write IFM-SimCLR, IFM-MoCo-v2 etc. to denote IFM applied within a specific framework. Code for IFM will be released publicly, and is also available in the supplementary material.

4.4.1 Does Implicit Feature Modification Actually Help Avoid Feature Suppression?

We study the effect IFM has on feature suppression by training ResNet-18 encoders for 200 epochs with $\tau \in \{0.05, 0.2, 0.5\}$ on the Trifeature dataset [Hermann and Lampinen, 2020]. Results are averaged over three seeds, with IFM using $\varepsilon = 0.1$ for simplicity. Fig. 4-6 shows that IFM improves the linear readout accuracy across *all* three features

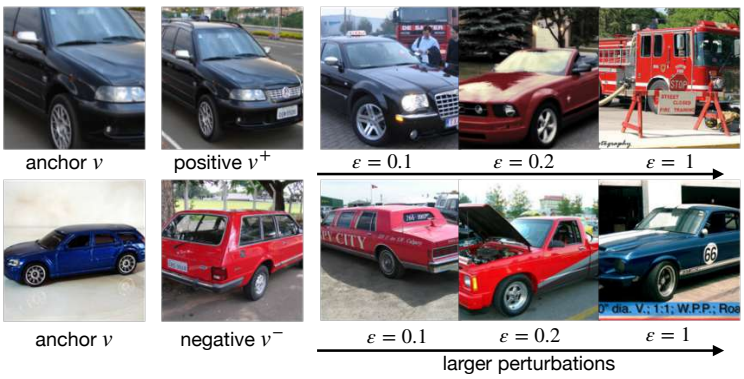


Figure 4-4: Visualizing implicit feature modification. **Top row:** progressively moving positive sample away from anchor. **Bottom row:** progressively moving negative sample towards anchor. In both cases, semantics such as color, orientation, and vehicle type are modified, showing the suitability of implicit feature modification for altering instance discrimination tasks.

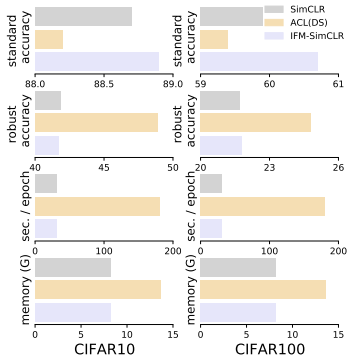


Figure 4-5: Comparison between IFM and ACL(DS). Under standard linear evaluation IFM performs best. ACL is suited to adversarial evaluation.

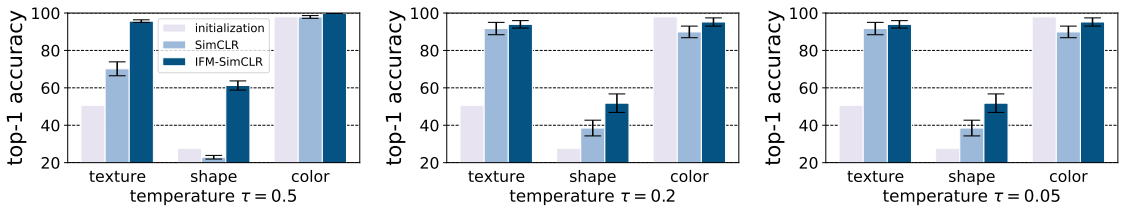


Figure 4-6: **Trifeature dataset.** Implicit feature modification reduces feature suppression, enhancing the representation of texture, shape and color features simultaneously. All results are average linear readout accuracy over three seeds and use a fixed value $\varepsilon = 0.1$ to illustrate robustness to ε.

for all temperature settings. The capability of IFM to enhance the representation of all features – i.e. reduce reliance on shortcut solutions – is an important contrast with tuning temperature τ or using hard negatives, which Fig. 4-3 shows only *trades-off* which features are learned.

4.4.2 Performance on Downstream Tasks

Sec. 4.3.1 and Sec. 4.4.1 demonstrate that implicit feature modification is adept at altering high-level features of an input, and combats feature suppression. This section shows that these desirable traits translate into improved performance on object

–	MoCo-v2	AdCo [Hu et al., 2020a]	IFM-MoCo-v2		
ε	N/A	N/A	0.05	0.1	0.2
top-1	$80.4_{\pm 0.11}$	$78.9_{\pm 0.21}$	$\mathbf{81.1}_{\pm 0.02}$	$80.9_{\pm 0.25}$	$80.7_{\pm 0.13}$

Table 4.1: Linear readout (%) on ImageNet100, averaged over five seeds. IFM improves over MoCo-v2 for all settings of ε.

classification and medical imaging tasks.

Experimental setup for classification tasks. Having observed the positive effect IFM has on feature suppression, we next test if this feeds through to improved performance on real tasks of interest. We benchmark using both SimCLR and MoCo-v2 [Chen et al., 2020b,f] with standard data augmentation [Chen et al., 2020b]. All encoders have ResNet-50 backbones and are trained for 400 epochs (with the exception of on ImageNet100, which is trained for 200 epochs). All encoders are evaluated using the test accuracy of a linear classifier trained on the full training dataset (see Appdx. B.2.4 for full setup details).

Classification tasks. Results given in Fig. 4-7 and Tab. 4.1 find that every value of $0 < \varepsilon \leq 0.2$ *improves performance across all datasets using both MoCo-v2 and SimCLR frameworks.* We find that optimizing $\mathcal{L}_\varepsilon$ (76.0% average score across all eight runs in Fig. 4-7) performs similarly to the standard contrastive loss (75.9% average score), and does worse than the IFM loss $(\mathcal{L} + \mathcal{L}_\varepsilon)/2$. This suggests that $\mathcal{L}$ and $\mathcal{L}_\varepsilon$ learn complementary features. Tab. 4.1 benchmarks IFM on ImageNet100 [Tian et al., 2019] using MoCo-v2, observing improvements of 0.9%. We also compare results on ImageNet100 to AdCo [Hu et al., 2020a], another adversarial method for contrastive learning. We adopt the official code and use the exact same training and finetuning hyperparameters as for MoCo-v2 and IFM. For the AdCo-specific hyperparamters – negatives learning rate lr_{neg} and negatives temperature τ_{neg} – we use a grid search over all combinations $lr_{\text{neg}} \in \{1, 2, 3, 4\}$ and $\tau_{\text{neg}} \in \{0.02, 0.1\}$, which includes the AdCo default ImageNet1K recommendations $lr_{\text{neg}} = 3$ and $\tau_{\text{neg}} = 0.02$ [Hu et al., 2020a]. The resulting AdCo performance of 78.9% is slightly below MoCo-v2. However using

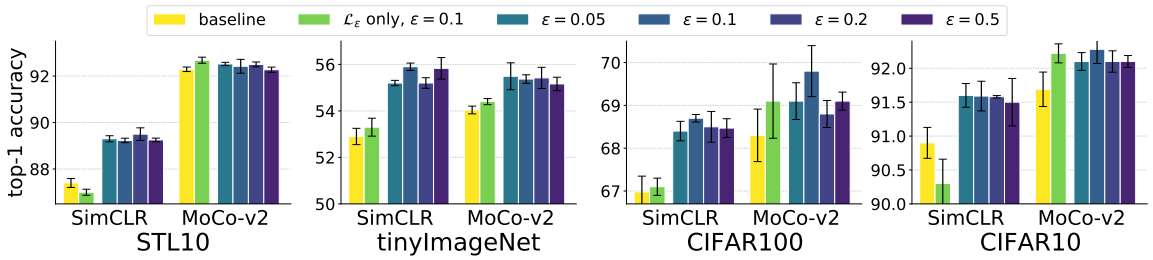


Figure 4-7: IFM improves linear readout performance on all datasets for all $\varepsilon \in \{0.05, 0.1, 0.2\}$ compared to baselines. Protocol uses 400 epochs of training with ResNet-50 backbone.

their respective ImageNet1K default parameters AdCo and MoCo-v2 achieve 72.4% and 71.8% respectively, suggesting that the discrepancy between AdCo and MoCo-v2 may in part be due to the use of improved hyperparameters tuned on MoCo-v2. Note importantly, IFM is robust to the choice of ε: all values $\varepsilon \in \{0.05, 0.1, 0.2\}$ were found to boost performance across all datasets and all frameworks. We emphasize that the MoCo-v2 baseline performance of 80.5% on ImageNet100 is strong. Our hyperparameters, which we detail in Appdx. B.2.4, may be of interest to other works benchmarking MoCo-v2 on ImageNet100.

Medical images. To evaluate our method on a modality differing significantly from object-based images we consider the task of learning representations of medical images. We benchmark using the approach proposed by [Sun et al., 2021] which is a variant of MoCo-v2 that incorporates the anatomical context in the medical images. We evaluate our method on the COPDGene dataset [Regan et al., 2011], which is a multi-center observational study focused on the genetic epidemiology of Chronic obstructive pulmonary disease (COPD). See Appdx. B.2.5 for full background details on the COPDGene dataset, the five COPD related outcomes we use for evaluation, and our implementation. We perform regression analysis for continuous outcomes in terms of coefficient of determination (R-square), and logistic regression to predict ordinal outcomes and report the classification accuracy and the *1-off* accuracy, i.e., the probability of the predicted category is within one class of true value.

Tab. 4.2 reports results. For fair comparison we use same experimental configuration for the baseline approach [Sun et al., 2021] and our method. We find that IFM yields

Method	logFEV1pp	logFEV$_1$FVC	CLE	CLE *1-off*	Para-septal	Para-septal *1-off*	mMRC	mMRC *1-off*
Loss	R-Square		Accuracy (%)					
$\mathcal{L}$ (baseline)	$0.566_{\pm.005}$	$0.661_{\pm.005}$	$49.6_{\pm0.4}$	$81.8_{\pm0.5}$	$55.7_{\pm0.3}$	$84.4_{\pm0.2}$	$50.4_{\pm0.5}$	$72.5_{\pm0.3}$
$\mathcal{L}_\varepsilon$, $\varepsilon = 0.1$	$0.591_{\pm.008}$	$0.681_{\pm.008}$	$49.4_{\pm0.4}$	$81.9_{\pm0.3}$	$55.6_{\pm0.3}$	$85.1_{\pm0.2}$	$50.3_{\pm0.8}$	$72.7_{\pm0.4}$
IFM, $\varepsilon = 0.1$	$\mathbf{0.615_{\pm.005}}$	$\mathbf{0.691_{\pm.006}}$	$48.2_{\pm0.8}$	$80.6_{\pm0.4}$	$55.3_{\pm0.4}$	$84.7_{\pm0.3}$	$50.4_{\pm0.5}$	$72.8_{\pm0.2}$
IFM, $\varepsilon = 0.2$	$0.595_{\pm.006}$	$0.683_{\pm.006}$	$48.5_{\pm0.6}$	$80.5_{\pm0.6}$	$55.3_{\pm0.3}$	$85.1_{\pm0.1}$	$49.8_{\pm0.8}$	$72.0_{\pm0.3}$
IFM, $\varepsilon = 0.5$	$0.607_{\pm.006}$	$0.683_{\pm.005}$	$49.6_{\pm0.4}$	$82.0_{\pm0.3}$	$54.9_{\pm0.2}$	$84.7_{\pm0.2}$	$\mathbf{50.6_{\pm0.4}}$	$\mathbf{73.1_{\pm0.2}}$
IFM, $\varepsilon = 1.0$	$0.583_{\pm.005}$	$0.675_{\pm.006}$	$\mathbf{50.0_{\pm0.5}}$	$\mathbf{82.9_{\pm0.4}}$	$\mathbf{56.3_{\pm0.6}}$	$\mathbf{85.7_{\pm0.2}}$	$50.3_{\pm0.6}$	$71.9_{\pm0.3}$

Table 4.2: Linear readout performance on COPDGene dataset. The values are the average of 5-fold cross validation with standard deviations. The bold face indicates the best average performance. IFM yields improvements on all phenotype predictions.

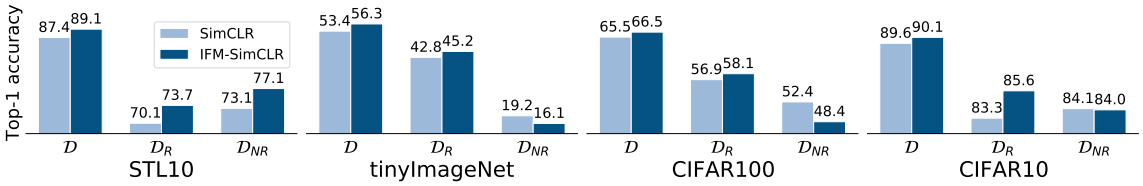


Figure 4-8: Label $\{\mathcal{D}, \mathcal{D}_\text{R}, \mathcal{D}_\text{NR}\}$ indicates which dataset was used to train the linear readout function. Improved performance of IFM on standard data $\mathcal{D}$ can be attributed to improved representation of *robust* features $\mathcal{D}_\text{R}$. See Sec. 4.4.3 for construction of robust ($\mathcal{D}_\text{R}$) and non-robust ($\mathcal{D}_\text{NR}$) datasets.

improvements on all outcome predictions. The gain is largest on spirometry outcome prediction, particularly logFEV1pp with improvement of 8.7% with $\varepsilon = 0.1$. We found that at least $\varepsilon = 0.5$ and 1.0 improve performance on all tasks. However, we note that not all features yield a statistically significant improvement with IFM.

4.4.3 Further study on the impact of IFM on feature learning

This section further studies the effect implicit feature modification has on *what type* of features are extracted. Specifically, we consider the impact on learning of robust (higher-level) vs. non-robust features (pixel-level features). Our methodology, which is similar to that of Ilyas et al. [Ilyas et al., 2019] for deep supervised learning, involves carefully perturbing inputs to obtain non-robust features.

Constructing non-robust features. Given encoder f we finetune a linear probe (classifier) h on-top of f using training data (to avoid smoothing effects we do not use data augmentation). Once h is trained, we consider each labeled example (x, y) from training data $\mathcal{D}_\text{train} \in \{\text{tinyImageNet, STL10, CIFAR10, CIFAR100}\}$. A

hallucinated target label t is sampled uniformly at random, and we perturb $x = x_0$ until $h \circ f$ predicts t using repeated FGSM attacks [Goodfellow et al., 2015] $x_k \leftarrow x_{k-1} - \varepsilon \text{sign}(\nabla_x \ell(h \circ f(x_{k-1}), t))$. At each step we check if $\arg\max_i h \circ f(x_k)_i = t$ (we use the maximum of logits for inference) and stop iterating and set $x_{\text{adv}} = x_k$ for the first k for which the prediction is t. This usually takes no more than a few FGSM steps with $\varepsilon = 0.01$. We form a dataset of "robust" features by adding (x_{adv}, y) to $\mathcal{D}_R$, and a dataset of "non-robust" features by adding (x_{adv}, t) to $\mathcal{D}_{NR}$. To a human the pair (x_{adv}, t) will look mislabeled, but for the encoder x_{adv} contains features predictive of t. Finally, we re-finetune (i.e. re-train) linear classifier g using $\mathcal{D}_R$ (resp. $\mathcal{D}_{NR}$) as training data.

Fig. 4-8 compares accuracy of the re-finetuned models on a test set of *standard* $\mathcal{D}_{\text{test}}$ examples (no perturbations are applied to the test set). Note that $\mathcal{D}_R$, $\mathcal{D}_{NR}$ depend on the original encoder f. When re-finetuning f we always use datasets $\mathcal{D}_R$, $\mathcal{D}_{NR}$ formed via FGSM attacks on f itself. So there is one set $\mathcal{D}_R, \mathcal{D}_{NR}$ for SimCLR, and another set for IFM. Fig. 4-8 shows that IFM achieves superior generalization ($\mathcal{D}$) compared to SimCLR *by better representing robust features* ($\mathcal{D}_R$). Representation of non-robust features ($\mathcal{D}_{NR}$) is similar for IFM (55.5% average across all datasets) and SimCLR (56.7% average). IFM is juxtaposed to the supervised adversarial training of Madry et al., which *sacrifices* standard supervised performance in exchange for not using non-robust features [Madry et al., 2018, Tsipras et al., 2018].

4.5 Discussion

This chapter studies the relation between contrastive instance discrimination and feature learning. While we focus specifically on contrastive learning, it would be of interest to also study any possible differences in feature learning for other empirically successful self-supervised methods [Bardes et al., 2021, Chen and He, 2021a, Grill et al., 2020, Zbontar et al., 2021]. Understanding differences in feature learning biases between different methods may inform which methods are best suited for a given task, as well as point the way to further improved self-supervised techniques.

Chapter 5

Contrastive Learning with Rotational Equivariance

So far this thesis has focused on the processes by which contrast learning produces a representation space where simple Euclidean *distances* measure meaningful variations in data. In this chapter, we extend this formulation adding additional geometric structure to the embedding space so that as well as *distances* being meaningful, *transformations* of input space to correspond to simple (i.e., linear) transformations of embedding space. Specifically, in the contrastive learning setting, we introduce an equivariance-promoting objective and theoretically prove that its minima forces augmentations on input space to correspond to rotations on the spherical embedding space. We show that merely combining our equivariant loss with a non-collapse term results in non-trivial representations, without requiring invariance to data augmentations. Optimal performance is achieved by also encouraging approximate invariance, where input augmentations correspond to small rotations. Our method, CARE: **C**ontrastive **A**ugmentation-induced **R**otational **E**quivariance, leads to improved performance on downstream tasks, and ensures sensitivity in embedding space to important variations in data (e.g., color) that standard contrastive methods do not achieve.

Acknolwedgements. This work is in collaboration with Sharut Gupta, Derek Lim, Soledad Villar, and Stefanie Jegelka. In particular, the entire project involved a highly collaborative effort with Sharut, with both parties making equal contributions.

5.1 Background and Motivation

What structure do neural network representation spaces need to possess in order to enable intelligent behavior to efficiently emerge [Ma et al., 2022]? One known key ingredient is to learn low-dimensional spaces in which simple Euclidean distances effectively measure the similarity between data. A standout success of recent years has been the development of powerful methods for achieving this at web-scale using self-supervision [Chen et al., 2020c, Schneider et al., 2021, Radford et al., 2021]. However, many use cases require the use of richer structural relationships that similarities between data cannot capture. One example that has enjoyed considerable success is the encoding of relations between objects (*X is a parent of Y*, *A is a treatment for B*) as simple transformations of embeddings (e.g., translations), which has driven learning with knowledge graphs [Bordes et al., 2013, Sun et al., 2019, Yasunaga et al., 2022]. But similar capabilities have been notably absent from existing self-supervised learning recipes.

Recent contrastive self-supervised learning approaches have explored ways to close this gap by ensuring representation spaces are sensitive to certain transformations of input data (e.g., variations in color) [Dangovski et al., 2022, Devillers and Lefort, 2023, Garrido et al., 2023, Bhardwaj et al., 2023]. Encouraging sensitivity is especially important in contrastive learning, as it is known to learn shortcuts that forget features that are not needed to solve the pretraining task [Robinson et al., 2021b]. This line of work formalizes sensitivity in terms of *equivariance*: transformations of input data correspond to predictable transformations in representation space. Equivariance requires specifying a family of transformations $a \in \mathcal{A}$ in the input space, a corresponding transformation T_a in representation space and training f so that $f(a(x)) \approx T_a f(x)$. A typical choice of T_a is a learnable feed-forward network, which acts non-linearly on embeddings [Devillers and Lefort, 2023, Garrido et al., 2023]. This approach has the disadvantage of encoding the relation between the embeddings of x and $a(x)$ in a complex and hard to interpret manner. It also suffers from geometric pathologies, such as inconsistency under compositions: $T_{a_2 \circ a_1} f(x) \neq T_{a_2} T_{a_1} f(x)$.

To address these concerns we propose CARE, an equivariant contrastive learning framework that learns to translate augmentations in the input space (such as cropping, blurring, and jittering) into simple *linear* transformations in feature space. Here, we use the sphere as our feature space (the standard space for contrastive learning), so we specifically consider transformations that are isometries of the sphere: rotations and reflections, i.e., orthogonal transformations. As orthogonal transformations are (intentionally) less expressive than prior non-linear formulations, our learning problem is more constrained and prior approaches for learning non-linear transforms do not apply (see Section 5.3). CARE trains f to preserve angles, i.e., $f(a(x))^\top f(a(x')) \approx f(x)^\top f(x')$, a property that must hold if f is orthogonally equivariant. We show that achieving low error on this seemingly weaker property also implies approximate equivariance and enjoys consistency under compositions. Critically, we can easily integrate CARE into contrastive learning workflows since both operate by comparing pairs of data.

The key contributions of this chapter include:

1. Introducing CARE, a novel equivariant contrastive learning framework that trains transformations (cropping, jittering, blurring, etc.) in input space to approximately correspond to local orthogonal transformations in representation space.

2. Theoretically proving and empirically demonstrating that CARE places an orthogonally equivariant structure on the embedding space.

3. Showing that CARE increases sensitivity to features (e.g., color) compared to invariance-based contrastive methods, and also improves performance on image recognition tasks.

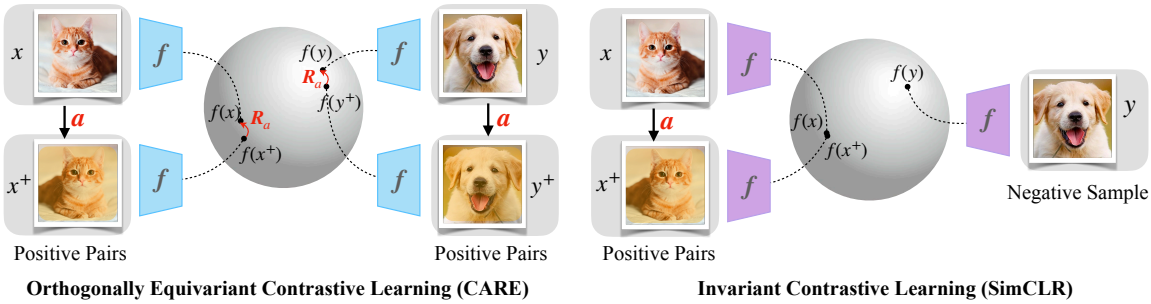


Figure 5-1: CARE is an equivariant contrastive learning approach that trains augmentations (cropping, blurring, etc.) of input data to correspond to orthogonal transformations of embedding space.

5.2 Rethinking how Augmentations are used in Contrastive Learning

Given access only to samples from a marginal distribution $p(x)$ on some input space $\mathcal{X}$ such as images, the goal of representation learning is commonly to train a feature extracting model $f : \mathcal{X} \to \mathbb{S}^{d-1}$ mapping to the unit sphere $\mathbb{S}^{d-1} = \{z \in \mathbb{R}^d : \|z\|_2 = 1\}$. A common strategy to automatically generate supervision from the data is to additionally introduce a space of augmentations $\mathcal{A}$, containing maps $a : \mathcal{X} \to \mathcal{X}$ which slightly perturb inputs $\bar{x}$ (blurring, cropping, jittering, etc.). Siamese self-supervised methods learn representation spaces that reflect the relationship between the embeddings of $x = a(\bar{x})$ and $x^+ = a^+(\bar{x})$, commonly by training f to be invariant or equivariant to the augmentations in the input space [Chen and He, 2021b].

Invariance to augmentation. The approach considered thusfar in this thesis is to train f to embed x and x^+ nearby—i.e., so that $f(x) = f(x^+)$ is *invariant* to augmentations. The InfoNCE loss [van den Oord et al., 2018, Gutmann and Hyvärinen, 2010] used in contrastive learning achieves precisely this:

$$\mathcal{L}_{\text{InfoNCE}}(f) = \mathbb{E}_{x,x^+,\{x_i^-\}_{i=1}^N}\left[-\log \frac{e^{f(x)^\top f(x^+)/\tau}}{e^{f(x)^\top f(x^+)/\tau} + \sum_{i=1}^N e^{f(x)^\top f(x_i^-)/\tau}}\right], \tag{5.1}$$

where $\tau > 0$ is a temperature hyperparameter, and $x_i^- \sim p$ are negative samples from the marginal distribution on $\mathcal{X}$. As noted by Wang and Isola [2020b], the contrastive

training mechanism balances invariance to augmentations with a competing objective: uniformly distributing embeddings over the sphere, which rules out trivial solutions such as constant functions.

Whilst contrastive learning has produced considerable advances in large-scale learning [Radford et al., 2021], several lines of work have begun to probe the fundamental role of invariance in contrastive learning. Two key conclusions of recent investigations include: 1) invariance limits the expressive power of features learned by f, as it removes information about features or transformations that may be relevant in fine-grained tasks [Lee et al., 2021, Xie et al., 2022a], and 2) contrastive learning actually benefits from not having exact invariance. For instance, a critical role of the projection head is to expand the feature space so that f is not fully invariant [Jing et al., 2022], suggesting that it is preferable for the embeddings of x and x^+ to be close, but not identical.

Equivariance to augmentation. To address the limitations of invariance, recent work has additionally proposed to control *equivariance* (i.e., sensitivity) of f to data transformations [Dangovski et al., 2022, Devillers and Lefort, 2023, Garrido et al., 2023]. Prior works can broadly be viewed as training a set of features f (sometimes alongside the usual invariant features) so that $f(a(x)) \approx T_a f(x)$ for samples $x \sim p$ from the data distribution where T_a is some transformation of the embedding space. A common choice is to take $T_a f(x) = \text{MLP}(f(x), a)$, a learnable feed-forward network, and optimize a loss $\|\text{MLP}(f(x), a) - f(a(x))\|_2$. Whilst a learnable MLP ensures that information about a is encoded into the embedding of $a(x)$, it permits complex non-linear relations between embeddings and hence does not necessarily encode relations in a linearly separable way. Furthermore, it does not enjoy the beneficial properties of equivariance in the formal group-theoretic sense, such as consistency under compositions in general: $T_{a_2 \circ a_1} f(x) \neq T_{a_2} T_{a_1} f(x)$.

Instead, this work introduces CARE, an equivariant contrastive learning approach respecting two key design principles:

Principle 3. *The map T_a satisfying $f(a(x)) = T_a f(x)$ should be linear.*

Principle 4. *Equivariance should be learned from* pairs *of data, as in invariant contrastive learning.*

The first principle asks that f converts complex perturbations a of input data into much simpler (i.e., linear) transformations in embedding space. Specifically, we constrain the complexity of T_a by considering isometries of the sphere, $O(d) = \{Q \in \mathbb{R}^{d\times d} : QQ^T = Q^TQ = I\}$, containing all rotations and reflections. Throughout this paper we define $f(a(x)) = T_a f(x)$ for $T_a \in O(d)$ to be *orthogonal equivariance*. This approach draws heavily from ideas in linear representation theory [Curtis and Reiner, 1966, Serre et al., 1977], which studies how to convert abstract group structures into matrix spaces equipped with standard matrix multiplication as the group operation.

The second principle stipulates *how* we want to learn orthogonal equivariance. Naively following previous non-linear approaches is challenging as our learning problem is more constrained, requiring learning a mapping $a \mapsto R_a$ to orthogonal matrices. Furthermore, for a single (a, x) pair, the orthogonal matrix R_a such that $f(a(x)) = R_a f(x)$ is not unique, making it hard to directly learn R_a. We sidestep these challenges by, instead of explicitly learning R_a, training f so that an augmentation a applied to two different inputs x, x^+ produces the same change in embedding space.

Our method, CARE, encodes data augmentations (cropping, blurring, jittering, etc.) as $O(d)$ transformations of embeddings using an equivariance-promoting objective function. CARE can be viewed as an instance of *symmetry regularization*, a term introduced by Shakerinava et al. [2022].

5.3 CARE: Contrastive Augmentation-induced Rotational Equivariance

This section introduces a simple and practical approach for training a model $f : \mathcal{X} \to \mathbb{S}^{d-1}$ so that f is orthogonally equivariant: i.e., a data augmentation $a \sim \mathcal{A}$ (cropping, blurring, jittering, etc.) applied to any input $x \in \mathcal{X}$ causes the embedding $f(x)$ to transformed by the same $R_a \in O(d)$ for all $x \in \mathcal{X}$: $f(a(x)) = R_a f(x)$.

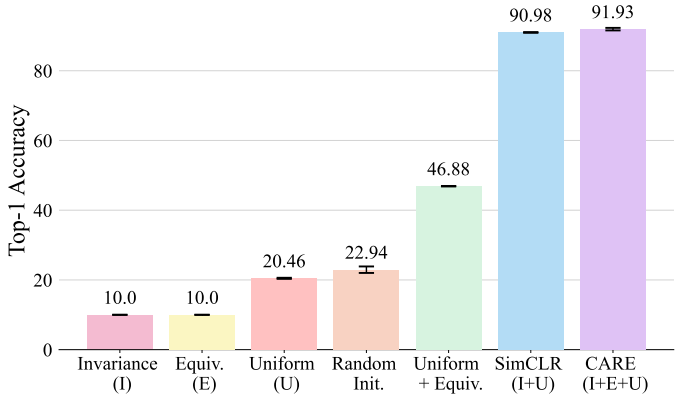


Figure 5-2: Ablating different loss terms. Combining $\mathcal{L}_{\text{equi}}$ with a uniformity promoting non-collapses term suffices to learn non-trivial features. However, optimal performance is achieved when encouraging *smaller* rotations, as in CARE. ResNet-50 models pretrained on CIFAR10 and evaluated with linear probes.

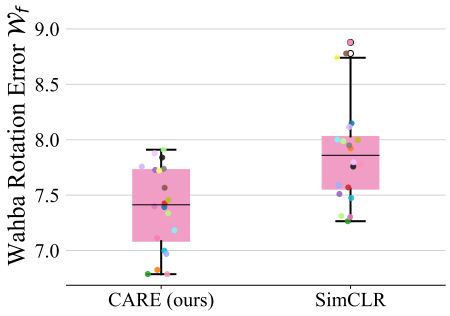


Figure 5-3: CARE learns a representation space with better rotational equivariance. We compare the models by the error of optimally rotating a set of embeddings to match the embeddings of augmented inputs, known as Wahba's problem (Section 5.4).

To achieve this, we consider the following loss:

$$\mathcal{L}_{\text{equi}}(f) = \mathbb{E}_{a\sim\mathcal{A}}\mathbb{E}_{x,x'\sim\mathcal{X}}\left[f(a(x'))^\top f(a(x)) - f(x)^\top f(x')\right]^2 \tag{5.2}$$

Since inner products describe angles on the sphere, this objective enforces the angles between the embeddings of independent samples x and x' to be the same as those between their transformed counterparts $a(x)$ and $a(x')$. This is necessarily true if f is orthogonally equivariant or, more generally, $R_a \in O(d)$ exists. But the converse—that $\mathcal{L}_{\text{equi}} = 0$ implies orthogonal equivariance—is non-obvious. In Section 5.3.1 we theoretically analyze $\mathcal{L}_{\text{equi}}$, demonstrating that it does indeed enforce mapping input augmentations to orthogonal transformations of embeddings. In practice, we replace the $f(x)^\top f(x')$ term with $f(a'(x))^\top f(a'(x'))$ for a freshly sampled $a' \sim \mathcal{A}$, noting that minimizing this variant also minimizes $\mathcal{L}_{\text{equi}}$, if we assume a' can be the identity function with non-zero probability. A trivial but undesirable solution that minimizes $\mathcal{L}_{\text{equi}}$ is to collapse the embeddings of all points to be the same (see Figure 5-2). One natural approach to avoiding trivial solutions is to combine the equivariance loss with a non-collapse term such as the uniformity $\mathcal{L}_{\text{unif}}(f) = \log \mathbb{E}_{x,x'\sim\mathcal{X}} \exp\left(f(x)^\top f(x')\right)$

[Wang and Isola, 2020b] whose optima f distribute points uniformly over the sphere:

$$\mathcal{L}(f) = \mathcal{L}_{\text{equi}}(f) + \mathcal{L}_{\text{unif}}(f). \tag{5.3}$$

This is directly comparable to the InfoNCE loss, which can similarly be decomposed into two terms:

$$\mathcal{L}_{\text{InfoNCE}}(f) = \mathcal{L}_{\text{inv}}(f) + \mathcal{L}_{\text{unif}}(f) \tag{5.4}$$

where $\mathcal{L}_{\text{inv}}(f) = \mathbb{E}_{a,a' \sim \mathcal{A}} \|f(a(x)) - f(a'(x))\|$ is minimized when f is invariant to $\mathcal{A}$—i.e., $f(a(x)) = f(x)$. Figure 5-2 shows that training using $\mathcal{L}_{\text{equi}} + \mathcal{L}_{\text{unif}}$ yields non-trivial representations. However, the performance is below that of invariance-based contrastive learning approaches. We hypothesize that this is because data augmentations—which make small perceptual changes to data—should correspond to *small* perturbations of embeddings, which $\mathcal{L}_{\text{equi}}$ does not enforce.

To rule out this possibility, we introduce CARE: **C**ontrastive **A**ugmentation-induced **R**otational **E**quivariance. CARE additionally enforces the orthogonal transformations in embedding space to be *localized* by reintroducing an invariance loss term $\mathcal{L}_{\text{inv}}$ to encourage f to be approximately invariant. Doing so breaks the indifference of $\mathcal{L}_{\text{equi}}$ between large and small rotations, biasing towards small. Specifically, we propose the following objective that combines our equivariant loss with InfoNCE:

$$\mathcal{L}_{\text{CARE}}(f) = \mathcal{L}_{\text{inv}}(f) + \mathcal{L}_{\text{unif}}(f) + \lambda \mathcal{L}_{\text{equi}}(f) \tag{5.5}$$

where λ weights the equivariant loss. We note that many variations of this approach are possible. For instance, the equivariant loss and InfoNCE loss could use different augmentations, resulting in invariance to specific transformations while maintaining rotational equivariance to others, similar to Dangovski et al. [2022]. The InfoNCE loss can also be replaced by other Siamese self-supervised losses. We leave further exploration of these possibilities to future work. In all, CARE consists of three components: (i) a term to induce orthogonal equivariance; (ii) a non-collapse term; and (iii) an invariance term to enforce localized transformations on the embedding

space.

5.3.1 Theoretical Properties of the Orthogonally Equivariant loss

In this section, we establish that matching angles via $\mathcal{L}_{\text{equi}}$ leads to a seemingly stronger property. Specifically, $\mathcal{L}_{\text{equi}} = 0$ implies the existence of an orthogonal matrix $R_a \in O(d)$ for any augmentation a, such that $f(a(x)) = R_a f(x)$ holds for all x. The converse also holds and is easy to see. Indeed, suppose such an $R_a \in O(d)$ exists. Then, $f(a(x'))^\top f(a(x)) = f(x')^\top R_a^\top R_a f(x) = f(x)^\top f(x')$, which implies $\mathcal{L}_{\text{equi}}(f) = 0$. We formulate the first direction as a proposition.

Proposition 5. *Suppose $\mathcal{L}_{equi}(f) = 0$. Then for almost every $a \in \mathcal{A}$, there is an orthogonal matrix $R_a \in O(d)$ such that $f(a(x)) = R_a f(x)$ for almost all $x \in \mathcal{X}$.*

Figure 5-1 illustrates this result. Crucially R_a is independent of x, without which the Proposition 5 would be trivial. That is, a single orthogonal transformation R_a captures the impact of applying a across the entire input space $\mathcal{X}$. Consequently, low $\mathcal{L}_{\text{equi}}$ loss converts "unstructured" augmentations in input space to have a structured geometric interpretation as rotations in the embedding space.

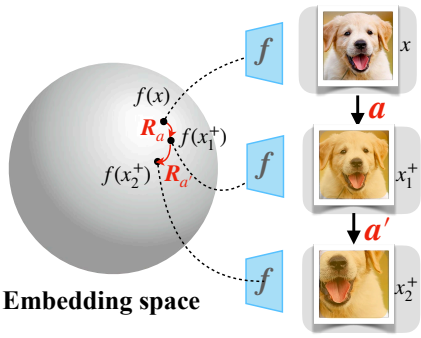


Figure 5-4: When $\mathcal{L}_{\text{equi}} = 0$, compositions of augmentations correspond to compositions of rotations.

This result can be expressed as the existence of a mapping $\rho : \mathcal{A} \to O(d)$ that encodes the space of augmentations within $O(d)$. This raises a natural question: how much of the structure of $\mathcal{A}$ does this encoding preserve? For instance, assuming $\mathcal{A}$ is a semi-group (i.e., closed under compositions $a' \circ a \in \mathcal{A}$), does this transformation respect compositions: $f(a'(a(x)) = R_{a'} R_a f(x)$? This property does not hold for non-linear actions [Devillers and Lefort, 2023], but does for orthogonal equivariance:

Corollary 1. *If $\mathcal{L}_{equi}(f) = 0$, then $\rho : \mathcal{A} \to O(d)$ given by $\rho(a) = R_a$ satisfies $\rho(a' \circ a) = \rho(a')\rho(a)$ for almost all a, a'. That is, ρ defines a group action on $\mathbb{S}^{d-1}$ up to a set of measure zero.*

Formally, this result states that if $\mathcal{A}$ is a semi-group, then $\rho : \mathcal{A} \to O(d)$ defines a group homomorphism (or linear representation of $\mathcal{A}$ in the sense of representation theory [Curtis and Reiner, 1966, Serre et al., 1977], a branch of mathematics that studies the encoding of abstract groups as spaces of linear maps).

To exactly attain $\mathcal{L}_{\text{equi}}(f) = 0$, the space of augmentations $\mathcal{A}$ needs to have a certain structure, but this becomes less restrictive if d is large. Assuming for simplicity that $\mathcal{A}$ is a group, the first isomorphism theorem for groups states that $\rho(\mathcal{A}) \simeq \mathcal{A}/\ker(\rho)$. For instance, if $\ker(\rho)$ is trivial, the equivariant loss can be exactly zero when the group of augmentations is a subgroup of the orthogonal group. Examples include orthogonal transformations or rotations that fix a subspace—i.e., $O(d')$ or $SO(d')$ with $d' \leq d$—or subgroups of the permutation group on d elements. Furthermore, the Peter-Weyl theorem implies that any compact Lie group can be realized as a closed subgroup of $O(d)$ for some d [Peter and Weyl, 1927]. In practice, we are learning equivariance, so do not expect to achieve exactly zero loss. Instead, the primary focus is on achieving better approximate equivariance (see Figure 5-7), while enforcing small transformations that remain local.

5.3.2 Extensions to Other Groups

Proposition 5 states that perfectly optimizing $\mathcal{L}_{\text{equi}} = 0$ produces an f that is equivariant, encoding augmentations in the input space as orthogonal transformation in the embedding space. Notably, since the computation of $\mathcal{L}_{\text{equi}}$ solely relies on pairwise data instances $x, x' \in \mathcal{X}$, it naturally aligns with the contrastive learning paradigm that already works with pairs of data. However, this alignment does not hold in cases where orthogonal transformations in the embedding space are replaced by arbitrary group actions.

Mathematically, invariants of the action of $O(d)$ on n points—seen in $(\mathbb{R}^d)^n$ as

$Q\,(x_1, \ldots x_n) = (Q\,x_1, \ldots, Q\,x_n)$—can be expressed as a function of pairs of objects $(x_i^\top x_j)_{i,j=1\ldots n}$. This is because the orthogonal group is defined as the stabilizer of a bilinear form. In other words, letting $B(x, x') = x^\top x'$ denote the standard inner product, we have

$$O(d) = \{A \in GL(d) : B(Ax, Ax') = B(x, x') \text{ for all } x, x' \in \mathbb{R}^d\}. \tag{5.6}$$

This argument applies more generally to other groups that are defined as stabilizers of bilinear forms. For instance, the Lorentz group, which has applications in the context of special relativity, can be defined as the stabilizer of the Minkowski inner product. Additionally, the symplectic group, which is used to characterize Hamiltonian dynamical systems, can be defined in a similar manner.

Such extensions to other groups allow us to use CARE for different embedding space geometries. For instance, several recent works have used a hyperbolic space as an embedding space for self-supervised learners [Ge et al., 2022, Yue et al., 2023, Desai et al., 2023]. If we constrain our embedding to a hyperboloid model of hyperbolic space, then linear isometries of this space are precisely the Lorentz group. Hence, using our equivariance loss with the Minkowski inner product replacing the Euclidean inner product would allow us to learn hyperbolic representations that transform the embeddings according to the action of the Lorentz group when an augmentation is applied to the input space. Further discussions on extensions to other groups are given in Appendix C.3.

5.4 Measuring Orthogonal Action on Embedding Space

To probe the geometric properties of CARE, we consider two efficiently computable metrics for empirically measuring the orthogonal equivariance in the embedding space. We report empirical results with these measures in Section 5.5.2.

Wahba's problem. Proposition 5 states that a single orthogonal matrix $R_a \in O(d)$ describes the effect of augmentation a for all input points x—i.e., R_a does not depend

on x. Hence, a natural way to assess the equivariance of f is to sample a batch of data $\{x_i\}_{i=1}^n$ and an augmentation a and test to what extent applying a transforms the embeddings of each x_i the same way. To measure this we compute a single rotation that approximates the map from $f(x_i)$ to $f(a(x_i))$ for all i. Let F and $F_a \in \mathbb{R}^{d\times n}$ have ith columns $f(x_i)$ and $f(a(x_i))$ respectively, then we compute the error

$$\mathcal{W}_f = \min_{R\in SO(d)} \|RF - F_a\|_{\text{Fro}}, \tag{5.7}$$

where $\|\cdot\|_{\text{Fro}}$ denotes the Frobenius norm. If $\mathcal{W}_f = 0$, then $f(a(x_i)) = R_a f(x_i)$ for all i. Problem (5.7) is a well-studied problem known as *Wahba's problem*. The analytic solution to Wahba's problem is easily computed. It is nearly $R^* = UV^\top$ where $U\Sigma V^\top$ is a singular value decomposition of $F_a F^\top$. However, a slight modification is required as this R^* could have determinant ± 1, and therefore may not belong to $SO(d)$. Fortunately, the only modification needed is to re-scale so that the determinant is one: $R^* = U \cdot \text{diag}\big(\mathbf{1}_{(n-1)}, \det(U)\det(V)\big) \cdot V^\top$ where $\mathbf{1}_n$ denotes the vector in $\mathbb{R}^n$ of all ones. This method of computing the solution R^* to Wahba's problem is known as Kabsch's algorithm [Kabsch, 1976], and has been used for aligning point clouds to, e.g., compare molecular and protein structures and spacecraft attitude determination [Markley and Crassidis, 2014, Kneller, 1991]. We use this algorithm to compute the optimal solution R^* and further compare the error of interest as $\mathcal{W}_f = \|R^*F - F_a\|$.

Relative rotational equivariance. Optimizing for the CARE objective may potentially result in learning invariance rather than equivariance. Specifically, for input image x, $f(a(x)) = f(x)$ for $a \in \mathcal{A}$ is a trivial optimal solution of $\arg\min_f \mathcal{L}_{\text{equi}}(f)$. To check that our model is learning non-trivial equivariance, we consider a metric similar to one proposed by Bhardwaj et al. [2023] for measuring the equivariance *relative* to the invariance of f:

$$\gamma_f = \mathbb{E}_{a\sim\mathcal{A}}\mathbb{E}_{x,x'\sim\mathcal{X}}\left\{\frac{(\|f(a(x')) - f(a(x))\|^2 - \|f(x') - f(x)\|^2)^2}{(\|f(a(x')) - f(x')\|^2 + \|f(a(x)) - f(x)\|^2)^2}\right\}. \tag{5.8}$$

Here, the denominator measures the invariance of the representation, with smaller

values corresponding to greater invariance to the augmentations. The numerator, on the other hand, measures equivariance and can be simplified to $[f(a(x'))^\top f(a(x)) - f(x)^\top f(x')]^2$ (i.e., $\mathcal{L}_{\text{equi}}(f)$) up to a constant, because f maps to the unit sphere. The ratio γ_f of these two terms measures the non-trivial equivariance, with a lower value implying greater non-trivial orthogonal equivariance.

5.5 Experiments

We examine the representations learned by CARE, as well as those obtained from purely invariance-based contrastive approaches. We study three aspects of our model: 1) quantitative measures of orthogonal equivariance, 2) qualitative evaluation of the effect of equivariance on sensitivity to data transforms, and 3) performance of features learned by CARE on image classification tasks. We describe our experiment configurations in detail in Appendix C.4.

5.5.1 Qualitative assessment of equivariance

A key property promised by equivariant contrastive models is sensitivity to specific augmentations. To qualitatively evaluate the sensitivity, or equivariance, of our models, we consider an image retrieval task on the Flowers-102 dataset [Nilsback and Zisserman, 2008], as considered by Bhardwaj et al. [2023]. Specifically, when presented with an input image x, we extract the top 5 nearest neighbors based on the Euclidean distance of $f(x)$ and $f(a(x))$, where $a \in \mathcal{A}$. We report the results of using color jitter as a transformation of the input, comparing the invariant (SimCLR) and our equivariant (CARE) models in Figure 5-6. We see that retrieved results for the CARE model exhibit

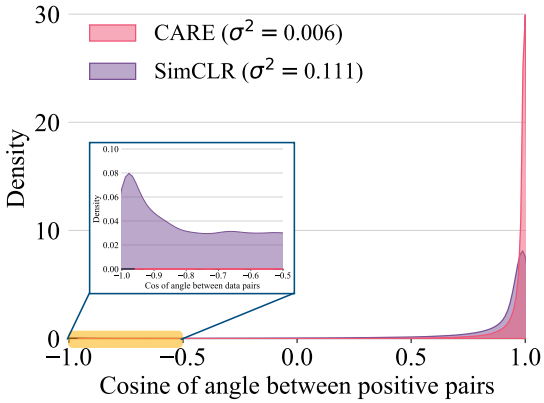


Figure 5-5: Histogram of the cosine of angles between data pairs for CARE and SimCLR. CARE exhibits a significantly lower variance of cosine similarity values compared to SimCLR.

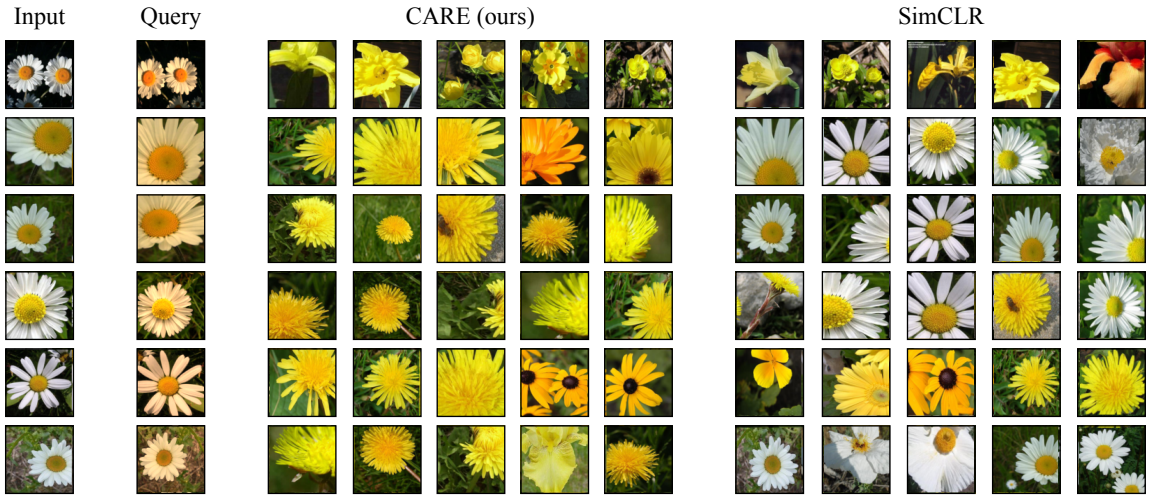


Figure 5-6: CARE exhibits sensitivity to features that invariance-based contrastive methods (e.g., SimCLR) do not. For each input we apply color jitter to produce the query image. We then retrieve the 5 nearest neighbors in the embedding space of CARE and SimCLR.

greater variability in response to a change in query color compared to the SimCLR model. Notably, the color of the retrieved results for all queries in the SimCLR model remains largely invariant, thereby confirming its robustness to color changes.

5.5.2 Quantitative Measures for Orthogonal Equivariance

Wahba's Problem We compare ResNet-18 models pretrained with CARE and with SimCLR on CIFAR10. For each model, we compute the optimal value $\mathcal{W}_f$ of Wahba's problem, as introduced in Section 5.4, over repeated trials. In each trial, we sample a single augmentation $a \sim \mathcal{A}$ at random and compute $\mathcal{W}_f$ for $f = f_{\text{CARE}}$ and $f = f_{\text{SimCLR}}$ over the test data. We repeat this process 20 times and plot the results in Figure 5-3, where the colors of dots indicate the sampled augmentation. Results show that CARE has a lower average error and worst-case error. Furthermore, comparing point-wise for a single augmentation, CARE achieves lower error in nearly all cases.

Relative rotational equivariance. We measure the relative rotational equivariance for both CARE and SimCLR over the course of pretraining by following the approach outlined in Section 5.4. Specifically, we compare ResNet-18 models trained using CARE and SimCLR on CIFAR10. From Figure 5-7, we observe that both the

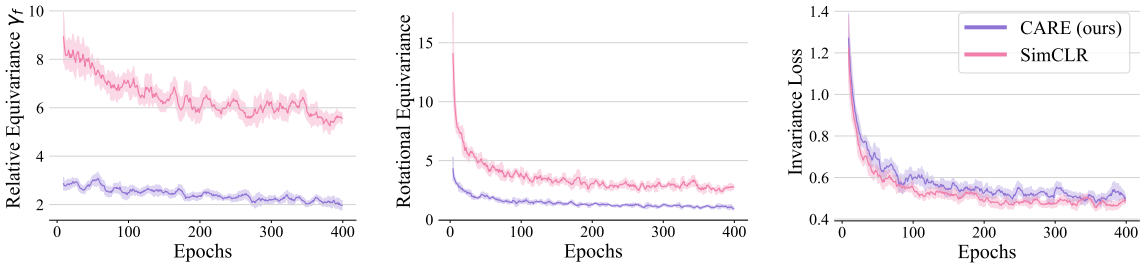


Figure 5-7: **Relative rotational equivariance** (lower is more equivariant). Both CARE and invariance-based contrastive methods (e.g., SimCLR) produce *approximately* invariant embeddings. However, they differ in their residual sensitivity to augmentations. CARE learns a considerably more rotationally structured embedding space. We note that this is in part because CARE is less invariant to augmentations (higher invariance loss).

models produce embeddings with comparable non-zero invariance loss $\mathcal{L}_{\text{inv}}$, indicating approximate invariance. However, they differ in their sensitivity to augmentations, with CARE attaining a much lower relative equivariance error. Importantly, this shows that CARE is *not* achieving lower equivariance error $\mathcal{L}_{\text{equi}}$ by collapsing to invariance, a trivial form of equivariance.

Analyzing Structure on a 2D manifold.

To further study $\mathcal{L}_{\text{equi}}$, we train an encoder f that projects the input onto $\mathbb{S}^1$, the unit circle in the 2D plane. In this case, orthogonal transformations are characterized by *angles*. We sample an augmentation $a \sim \mathcal{A}$ and measure the cosine of the angle between pairs $f(x)$ and $f(a(x))$ for all x in the test set. This process is repeated for 20 distinct sampled augmentations, and the density of all recorded cosine angles is recorded in Figure 5-5. Both CARE and SimCLR exhibit high density close to 1, demonstrating approximate invariance. However, unlike CARE, SimCLR exhibits non-zero density in the region -0.5 to -1.0, indicating that the application of augmentations significantly displaces the embeddings. Additionally, CARE consistently exhibits lower variance σ^2 of the cosine angles between $f(x)$ and $f(a(x))$ for a fixed augmentation, as expected given that it is supposed to transform all embeddings in the same way.

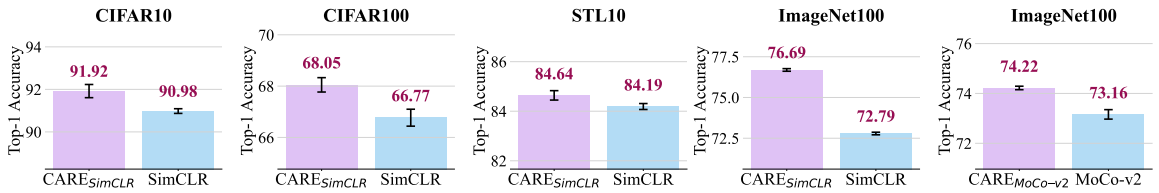


Figure 5-8: Top-1 linear readout accuracy (%) on CIFAR10, CIFAR100, STL10 and ImageNet100. All results are from 5 independent seed runs for the linear probe.

5.5.3 Linear Probe for Image Classification

Next, we examine the quality of features learned by CARE for solving image classification tasks. We train ResNet-50 models on four datasets: CIFAR10, CIFAR100, STL10, and ImageNet100 using CARE and SimCLR. To illustrate that CARE can also be integrated into other self-supervised frameworks, we train MoCo-v2 models on ImageNet100 (with and without CARE). We refer to the model trained using CARE with SimCLR or MoCo-v2 backbone as $\text{CARE}_{\text{SimCLR}}$ and $\text{CARE}_{\text{MoCo-v2}}$ respectively. For each method and dataset, we evaluate the quality of the learned features by training a linear classifier (i.e., probe [Alain and Bengio, 2017]) on the frozen features of f and report the test set performances in Figure 5-8. In all cases, we run the linear probe training for five random seeds and report averages. We find consistent improvements in performance using CARE, showing the benefits of our structured embedding approach for image recognition tasks.

5.5.4 Ablation of Loss Terms

The CARE loss $\mathcal{L}_{\text{CARE}}$ is a weighted sum of the InfoNCE loss $\mathcal{L}_{\text{InfoNCE}}$ and the orthogonal equivariance loss $\mathcal{L}_{\text{equi}}$. Furthermore, as outlined in Section 5.3, the InfoNCE loss is itself a combination of an invariance inducing loss $\mathcal{L}_{\text{inv}}$ and a non-collapse term $\mathcal{L}_{\text{unif}}$. To study each loss component, we pretrain ResNet-50 models on CIFAR10 using different combinations of the three losses. The results in Figure 5-2 suggest that simply optimizing for $\mathcal{L}_{\text{inv}}$ and $\mathcal{L}_{\text{equi}}$ leads to collapse, while optimizing $\mathcal{L}_{\text{unif}}$ alone prevents collapse but performs similar to random initialization. Interestingly, $\mathcal{L}_{\text{unif}} + \mathcal{L}_{\text{equi}}$ yields non-trivial representations without directly enforcing invariance. But the performance falls below that of invariance-based contrastive baselines. In

combination with the invariance term $\mathcal{L}_{\text{inv}}$—which biases rotations to be small—we achieve superior performance to the invariance-only counterpart.

5.6 Related work

Equivariance is a key tool for encoding geometric structure—e.g., symmetries—into neural network representations [Cohen and Welling, 2016, Bronstein et al., 2021]. Whilst hard-coding equivariance into model architectures is very successful, approximate learned equivariance [Kaba et al., 2022, Shakerinava et al., 2022], has certain advantages: 1) when the symmetry is provided only by data, with no closed-form expression, 2) can still be used when it is unclear how to hard code equivariance into the architecture, and 3) can exploit standard high capacity architectures [He et al., 2016, Dosovitskiy et al., 2021a], benefiting from considerable engineering efforts to optimize their performance. Shakerinava et al. [2022] also consider learning orthogonal equivariance, but consider problems where both input and embedding space are acted on by $O(d)$. Our setting differs from this in two key ways: 1) we consider a very different set of transforms of input space—jitter, crops, etc.—and 2) can be naturally integrated into contrastive learning, and 3) theoretically study the minima of the angle-preserving loss. A related line of work, *mechanistic interpretability*, hypothesizes that algorithmic structure—possibly including group symmetries—emerge naturally within network connections during training [Chughtai et al., 2023]. Our approach is very different from this as we directly *train* models to have the desired structure without relying on implicit processes. Finally, the geometry of representation space has been used in a very different sense in prior contrastive learning approaches, for instance bootstrapping useful negatives Chuang et al. [2020], Robinson et al. [2021a] based on their location in embedding space during training.

Chapter 6

A Simple, Efficient and Scalable Contrastive Masked Autoencoder

This chapter introduces CAN, a simple, efficient and scalable method for self-supervised learning of visual representations. The framework is a minimal and conceptually clean synthesis of (C) contrastive learning, (A) masked autoencoders, and (N) the noise prediction approach used in diffusion models. The learning mechanisms are complementary to one another: contrastive learning shapes the embedding space across a batch of images; masked autoencoders reconstruct low-frequency spatial correlations in a single image; and noise prediction reconstructs high-frequency components of an image. The combined approach outperforms its MAE and SimCLR constituent parts on an extensive set of downstream transfer learning and robustness tasks under both linear probe and finetune protocols, and pre-training on large datasets such as JFT-300M and ImageNet-21K. Importantly, CAN masks 50% of patches in *both* views, meaning that the overall FLOPs load of SimCLR is 70% higher than CAN for ViT-L backbones. Code can be found at `https://github.com/shlokk/mae-contrastive`.

Acknowledgements. This chapter is based on [Mishra* et al., 2023], which is work is in collaboration with Shlok Mishra, Huiwen Chang, David Jacobs, Aaron Sarna, Aaron Maschinot, Dilip Krishnan. In particular, Shlok, Aaron Machinot and Dilip all made significant contributions.

6.1 Background and Motivation

Self-supervised learning promises continued advances in the state of the art by enabling the use of increasingly large models and datasets without reliance on human annotations. However, interest in larger datasets has precipitated an increased reliance on web-scraped data collection processes, which result in heterogeneous, uncurated datasets [Yu et al., 2022, Radford et al., 2021, Jia et al., 2021]. Extreme image heterogeneity has made scaling vision models to large uncurated datasets a non-trivial challenge [Tian et al., 2021, Cole et al., 2022]. There are two families of self-supervised methods for images which have both proven highly effective on curated datasets (e.g., ImageNet), and are therefore natural candidates for scaling to large, uncurated data. First, masked image models such as the masked autoencoder (MAE) [He et al., 2022] are a nascent approach based on a mask-and-reconstruct training mechanism. This classical idea [Ballard, 1987] is enjoying a rejuvenation thanks to favourable efficiency when combined with the vision transformer architecture [Dosovitskiy et al., 2021c]. Second, contrastive learning [van den Oord et al., 2018, Chen et al., 2020c, He et al., 2020b] trains an encoder to distinguish between pairs of positive samples generated with data augmentations and negative pairs sampled at random. Both approaches have proven to be very powerful self-supervised methods.

Contrastive learning and masked autoencoders (MAE) employ very different learning mechanisms: the former trains the encoder to be invariant to semantics-preserving data variations, while MAE learns spatial statistical correlations. Furthermore, MAE methods treat each sample independently in the loss function, while contrastive methods explicitly look at the relationship between all samples in the batch, by either reducing or increasing embedding distance. Given this, we hypothesize that these two approaches are *complementary*, extracting different discriminative features. If this hypothesis holds, then combining the two approaches presents itself as a promising way to build a reinforced training mechanism.

Advances in diffusion models [Ho et al., 2020, Song et al., 2021] have been driven by denoising models that predict the *noise* added to an input image. While denoising

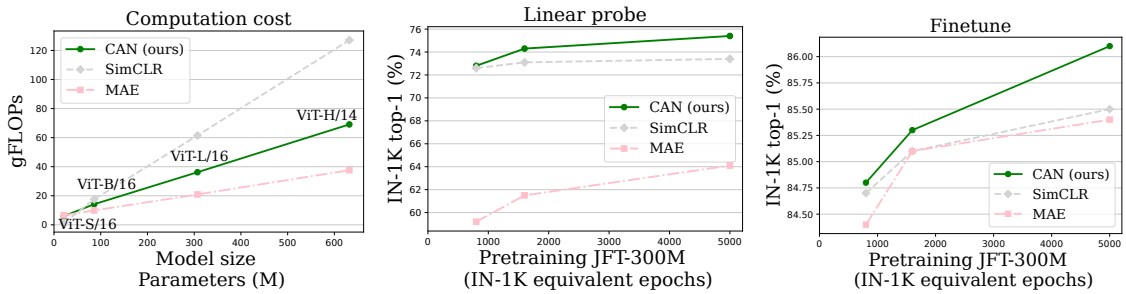


Figure 6-1: **Left:** CAN scales better than SimCLR since it uses masked inputs. **Middle and right:** CAN outperforms SimCLR and MAE on ImageNet linear probe and finetune evaluations for ViT-L models when pre-training on uncurated data such as JFT-300M.

has not yet enjoyed wide adoption for self-supervised learning on its own, we believe it offers a promising *third* complementary learning mechanism to contrastive learning and masked autoencoding. Specifically, denoising forces the model to learn high-frequency information, whereas autoencoder reconstructions focus on low-frequency information [Hou et al., 2017].

This chapter presents CAN, a minimal fusion of contrastive learning, masked autoencoders, and the diffusion denoising loss. Our method enjoys stronger performance than its constituent parts on their own, especially when pre-training on large datasets such as JFT-300M and ImageNet-21K, which consist of 300M and 14M images, respectively. For instance, evaluating JFT-trained ViT-L models using the top-1 accuracy of an ImageNet-trained linear probe, MAE achieves 64.1% and SimCLR achieves 73.4%, while CAN achieves 75.4%. The advantages of CAN are:

1. **Simplicity.** CAN is a minimal synthesis of three powerful self-supervised learning methods.

2. **Efficiency.** CAN enjoys a favourable efficiency-performance trade-off (Figure 6-1), e.g., SimCLR uses 70% more FLOPs than CAN with ViT-L backbones.

3. **Scalability.** CAN scales well to training on large image datasets, such as JFT-300M and ImageNet-21K.

CAN is more efficient than SimCLR since it masks 50% of patches in each view.

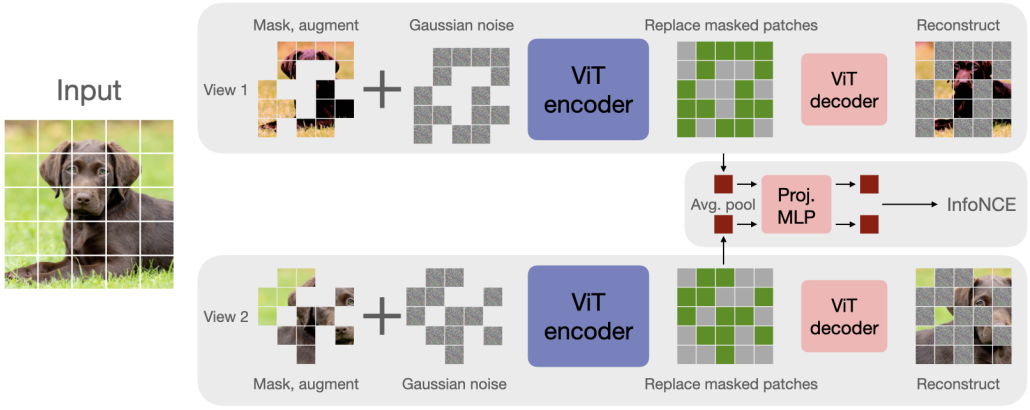


Figure 6-2: **The CAN framework:** Two views of an image are generated, 50% of patches randomly masked in each, and noise is added to patches. An encoder is trained to solve three tasks: 1) **Reconstruction:** encoded patches are passed to a decoder that reconstructs missing patches, 2) **Denoise:** reconstructs the noise added to unmasked patches, and 3) **Contrast:** pooled patches are passed to a contrastive loss, using in-batch samples as negatives [Chen et al., 2020c].

This also translates to faster run-times, with our largest training (ViT-L 5000 epochs) taking 2 weeks for SimCLR, and 1 week for CAN on our hardware.

6.2 A Simple Contrastive Masked Autoencoder

Our approach is a minimal synthesis of contrastive learning, the masked autoencoder (MAE) [He et al., 2022], and the denoising loss used in the training of diffusion models. We focus on simplicity and scalability, aiming to design a hybrid with as few complex or costly components as possible. We also aim to minimize *wasted* computation: in particular, the MAE decoder requires reconstructions of all patches, but only those of masked patches are used in the loss, a fact that CAN exploits. Below, first we detail the basic pipeline of generating views and passing masked inputs through the encoder and decoder, then explain the three objectives we use: contrastive, reconstruction, and denoising. The penultimate section describes the combined objective, and the final section discusses scalability.

6.2.1 Overview of Method

Given a batch of n images $\{\mathbf{x}\}_{i=1}^{n}$, we generate two views $\mathbf{x}_i^1, \mathbf{x}_i^2 \in \mathbb{R}^{h\times w\times 3}$ of each image without supervision using the same data augmentations as Chen et al. [2020c]. Each image is then split into $T = (h/p) \times (w/p)$ non-overlapping patches of size $p \times p$: $\mathbf{x}_{i,\text{patch}}^1, \mathbf{x}_{i,\text{patch}}^2 \in \mathbb{R}^{T\times p\times p\times 3}$ in preparation for input to the ViT encoder. We always assume that p divides h and w. Two masks $\mathbf{M}_i^1, \mathbf{M}_i^2 \in \{0,1\}^T$ are independently generated, with a 1 in coordinate $t \in \{1,\dots T\}$ indicating that the t-th patch is masked. Each patch is masked independently with probability r, conditioned on always having exactly $T' = r \cdot T$ patches masked, which we assume is an integer. In all CAN experiments our default masking rate is $r = 50\%$ unless explicitly stated otherwise (note that for all MAE results we follow the exact settings as in [He et al., 2022] using the default $r = 75\%$). Following He et al. [2022], only the $T - T'$ *unmasked* patches are passed to the ViT encoder, which processes the two views in parallel. Masking a large fraction of patches from both views makes our method much more efficient (see Table 6-1) than contrastive methods that use two full views and recent works that use one full view and one masked view [Assran et al., 2022, Huang et al., 2022]. Finally, we collect the embeddings of unmasked tokens $\mathbf{z}_i^1, \mathbf{z}_i^2 \in \mathbb{R}^{(T-T')\times d}$ and reshape into $T \times d$ tensors by adding a learned `[M]` embedding to positions corresponding to masked tokens. The result is passed through a comparatively lightweight ViT decoder to produce outputs $\hat{\mathbf{x}}_i^1, \hat{\mathbf{x}}_i^2$ in image space $\mathbb{R}^{h\times w\times 3}$.

6.2.2 Contrastive Learning Objective

The embeddings $\mathbf{z}_i^1, \mathbf{z}_i^2 \in \mathbb{R}^{(T-T')\times d}$ returned by the encoder are pooled via a simple mean along the first dimension to form d-dimensional embeddings, which are passed through a lightweight MLP projection head that maps into a lower dimension space $\mathbb{R}^r$, $r < d$, and normalized to unit length to produce embeddings $\mathbf{u}_i^1, \mathbf{u}_i^2 \in \mathbb{R}^r$ for $i = 1,\dots n$. For the ith batch item we collect the other $2n - 2$ samples in-batch

$\mathcal{N}_i = \{\mathbf{u}_j^1, \mathbf{u}_j^2\}_{j \neq i}$ to use as negatives, and compute the $\mathcal{L}_{\text{InfoNCE}}$ loss:

$$\frac{1}{2n} \sum_{v=1,2} \sum_{i=1}^{n} -\log \frac{e^{\mathbf{u}_i^{1\top} \mathbf{u}_i^2 / \tau}}{e^{\mathbf{u}_i^{1\top} \mathbf{u}_i^2 / \tau} + \sum_{\mathbf{u}^- \in \mathcal{N}_i} e^{\mathbf{u}_i^{v\top} \mathbf{u}^- / \tau}}$$

where $\tau > 0$ is a temperature parameter, defaulting to 0.1. Our choice of InfoNCE objective is justified by recent work [Koppula et al., 2022] that found that a simple InfoNCE objective as in SimCLR scales to large dataset better than methods such as BYOL [Grill et al., 2020] or DINO [Caron et al., 2020].

6.2.3 Patch Reconstruction Objective

The outputs $\hat{\mathbf{x}}_i^1, \hat{\mathbf{x}}_i^2$, $i = 1, \ldots, n$ of the ViT decoder are trained to reconstruct the missing patches of each image. As in He et al. [2022], we find it best to only compute the reconstruction loss on masked patches:

$$\mathcal{L}_{\text{rec}} = \frac{1}{2n} \sum_{v=1,2} \sum_{i=1}^{n} \|\mathbf{M}_i^v \circ (\mathbf{x}_i^v - \hat{\mathbf{x}}_i^v)\|_2^2$$

where $\circ$ multiplies all pixels in the tth patch of the residual image $\mathbf{x}_i^v - \hat{\mathbf{x}}_i^v$ by $(\mathbf{M}_i^v)_t \in \{0, 1\}$.

Whilst computing the loss only on masked patches gives better performance, it indicates wasted computation since the decoder also produces reconstructions for unmasked patches. To avoid waste we propose an alternative objective specifically for unmasked patches, which we discuss next.

6.2.4 Denoising Objective

Inspired by the significant advances in diffusion modelling using *denoising* training objectives [Ho et al., 2020, Kingma et al., 2021] and their equivalent score-based counterparts [Song et al., 2021, Vincent, 2011] we revisit the suitability of denoising for self-supervised learning. We add independent isotropic Gaussian noise to each image $\mathbf{x}_i^v \leftarrow \mathbf{x}_i^v + \sigma_i^v \mathbf{e}_i^v$ with $\mathbf{e}_i^v \sim \mathcal{N}(\mathbf{0}, I)$ and σ_i^v uniformly sampled from an interval

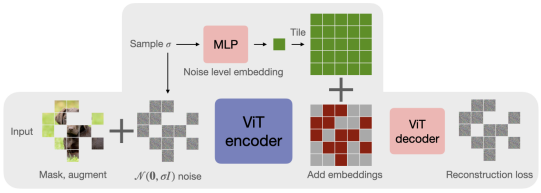


Figure 6-3: **Denoising:** Both the encoded patches and the noise level σ are passed to the decoder by passing σ through an MLP, and adding the result to each embedded token.

$[0, \sigma_{\max}]$. This noisy input is masked and passed to the encoder as described in Section 6.2.1. When passing encoded patches to the decoder we make a small addition to the method in Section 6.2.1 to provide the decoder with information on the noise level σ_i^v to help it separate noise from the ground truth image. This is motivated by denoising diffusion methods, which pass both the noisy image and the noise level as inputs to the denoising model [Ho et al., 2020]. We approach this by using σ_i^v as a positional encoding in the decoder, similarly to Vaswani et al. [2017a]. First we produce a sinusoidal embedding of $\sigma_i^v \in \mathbb{R}^d$, which is passed through a lightweight 2 layer MLP with ReLU activations of constant width d to produce a (learnable) embedding $\mathbf{p}_i^v \in \mathbb{R}^d$, whose dimension matches the latent dimension of $\mathbf{z}_i^v \in \mathbb{R}^{T \times d}$. We add the result to each embedded token (including missing tokens `[M]`) to provide noise-level information: $(\mathbf{z}_i^v)_t \leftarrow (\mathbf{z}_i^v)_t + \mathbf{p}_i^v$ for $t = 1 \ldots, T$, and pass the result to the decoder producing $\hat{\mathbf{x}}_i^v$. We define our denoising loss function, which is computed only on unmasked pixels:

$$\mathcal{L}_{\text{denoise}} = \frac{1}{2n} \sum_{v=1,2} \sum_{i=1}^{n} \|(1 - \mathbf{M}_i^v) \circ (\sigma_i^v \mathbf{e}_i^v - \hat{\mathbf{x}}_i^v)\|_2^2$$

where, $\circ$ multiplies pixels by the patch-level masking as in Section 6.2.3. Note that the reconstruction loss $\mathcal{L}_{\text{rec}}$ still uses the *clean* input $\mathbf{x}$ as its target, with no noise added. The denoising loss is extremely lightweight, introducing only a very small overhead due to the MLP. We emphasize that the reconstruction of noise patches comes at zero additional cost since the decoder produces reconstructions of all patches, both masked and unmasked, but only reconstructions of masked patches are used in $\mathcal{L}_{\text{rec}}$. Finally, it has been observed in the diffusion modelling literature that although

it is equivalent to train a denoising model to estimate the noise $\mathbf{e}$, or to estimate the clean input $\mathbf{x}$ [Vincent, 2011], there is an empirical gap, with noise target faring better. While we do not pursue it further, our testing corroborates this.

6.2.5 The Combined Objective Function

The overall CAN objective trains the encoder and decoder to optimize three losses combined:

$$\mathcal{L}_{\text{CAN}} = \lambda_{\text{InfoNCE}}\mathcal{L}_{\text{InfoNCE}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{denoise}}\mathcal{L}_{\text{denoise}}$$

where $0 \leq \lambda_{\text{InfoNCE}}, \lambda_{\text{rec}}, \lambda_{\text{denoise}}$, and $\lambda_{\text{InfoNCE}} + \lambda_{\text{rec}} + \lambda_{\text{denoise}} = 1$ weight the objectives. In practice we parameterize the weights by eliminating one variable using the equality constraint, taking: $\lambda_{\text{rec}} = (1 - \lambda_{\text{InfoNCE}}) \cdot \lambda$ and $\lambda_{\text{denoise}} = (1 - \lambda_{\text{InfoNCE}}) \cdot (1 - \lambda)$ where $0 \leq \lambda \leq 1$. This parameterization makes it easy to control the weighting between the two reconstruction losses $\mathcal{L}_{\text{rec}}, \mathcal{L}_{\text{denoise}}$ on the one hand, and the contrastive loss $\mathcal{L}_{\text{InfoNCE}}$ on the other. We find that performance is robust to the choice of λ, and many choices of λ_{InfoNCE} also work well (see Section 6.4).

6.2.6 Discussion on Efficiency

The efficiency of CAN arises from masking 50% of both views. We also omit certain design choices in the interests of efficiency: we do not use a momentum encoder or multiple views (multi-crop). Each of these components tends to add significant ($2\times$ or more) expense to training. Even without these components CAN achieves strong performance, outperforming its key constituent parts SimCLR and MAE.

6.3 Experimental Results

6.3.1 Pre-training on Uncurated Data: JFT-300M

A key promise of self-supervised learning is to allow models to be trained on extremely large scale image datasets collected from the Web. Not only is such data likely to be

	Architecture	Epochs	IN-1K top-1
MoCLR [Tian et al., 2021]	R50	5000	67.6
BYOL [Grill et al., 2020]	R50	5000	67.9
DnC [Tian et al., 2021]	R50	1000	67.9
DnC [Tian et al., 2021]	R50	4500	70.7
MoCLR [Tian et al., 2021]	R200×2	5000	74.2
DnC [Tian et al., 2021]	R200×2	3000	**77.3**
MAE† [He et al., 2022]	ViT-L	1600	50.5
MAE† [He et al., 2022]	ViT-L	5000	64.1
SimCLR† [Chen et al., 2020c]	ViT-B	800	65.8
SimCLR† [Chen et al., 2020c]	ViT-L	800	72.6
SimCLR† [Chen et al., 2020c]	ViT-L	1600	73.1
SimCLR† [Chen et al., 2020c]	ViT-L	5000	73.4
CAN (ours)	ViT-B	800	67.1
CAN (ours)	ViT-L	800	72.8
CAN (ours)	ViT-L	1600	74.3
CAN (ours)	ViT-L	3000	75.3
CAN (ours)	ViT-L	5000	75.4

Table 6.1: **JFT-300M pre-training:** Comparison to the state of the art on ImageNet linear probe. CAN outperforms all methods except DnC, which uses a complicated multi-stage training process. Computation is measured as ImageNet-equivalent epochs. †Our implementation of [Chen et al., 2020c] and [He et al., 2022].

unannotated, but also *uncurated*: images containing many objects, variable lighting, artifacts (e.g., watermarks) and so on. The large variation in images found online presents a major challenge to self-supervised learning, and it is not guaranteed that methods that work well on curated (and comparatively smaller) datasets such as ImageNet will work equally well on less curated data. To study how CAN scales to large datasets we use JFT-300M [Sun et al., 2017], a dataset of around 300 million images.

Setup. Training time is measured in ImageNet-equivalent epochs: 1 epoch equals 1281167/[batch size] steps, the number of steps in one IN-1K epoch. Models are evaluated using linear probe and finetuning on IN-1K. All hyperparameers were tuned on IN-1K, besides learning rate and weight decay which we cut by a factor of 4 and 2 respectively to stabilize training on JFT-300M. See Appendix D.3 and Section 6.4 for details.

Results. Figure 6-1 compares CAN to SimCLR and MAE baselines using ViT-L models. CAN achieves a much better trade-off between efficiency (measured in FLOPs) and performance using ViT-L models for all three methods: SimCLR uses 70% more FLOPs than CAN, which consistently outperforms both SimCLR and MAE: for training ViT-L models for 5000 epochs, CAN achieves an IN-1K linear probe performance of 75.4%, compared to 73.4% for SimCLR and 64.1% for MAE. The relatively poorer linear probe performance of MAE on JFT-300M highlights the non-triviality of scaling from IN-1K to larger datasets and suggests that while MAE is scalable for *model size*, scalability to larger *datasets* requires further study. Figure 6-1 (right) gives finetuning results. CAN performs favourably: for a 5000 epoch pre-training schedule, CAN achieves an IN-1K linear probe performance of 86.1%, compared to 85.5% for SimCLR and 85.4% for MAE. CAN also enjoys better scaling with training schedule length than either MAE or SimCLR, with the difference in performance becoming *larger* for longer schedules. We hypothesize that this is not coincidental, and that strong pre-training tasks like CAN play an important role in scalability.

We also compare CAN to the current state of the art on JFT-300M pre-training in

Table 6.1. Our best performance, 75.4% with ViT-L outperforms all methods besides DnC, with 77.3% [Tian et al., 2021] with R200×2. However we note that CAN is *considerably* simpler than DnC, which involves training 10 separate "expert" models (each as large as the final model), and then using MoCLR (an improvement of SimCLR that adds a momentum encoder and more), using distillation to produce a single final model. Our calculations suggest that training a ViT-L with CAN is about 3× faster than training the considerably smaller ResNet50 with DnC in terms of wall clock time (see Appendix D.2 for explanation). CAN on ViT-L outperforms MoCLR with R200×2 backbone (similar parameter counts), where we note that MoCLR performs as well or better than BYOL and MoCo-v3 on IN-1K [Tian et al., 2021].

6.3.2 Pre-training on ImageNet-21K

We also consider the performance of CAN on pre-training on ImageNet-21K (IN-21K), a publicly available dataset of 14.2 million images Deng et al. [2009]. We use the same hyperparameter settings as JFT-300M. We run a full set of evaluations on linear probe (Table D.1), robustness (Figure D-8), and few-shot learning (Figure D-9) (see Sections 6.3.4 and 6.3.5 for details on few-shot and robustness evaluations). Results are reported in Appendix D.1.1. CAN also performs well with IN-21K pre-training, with CAN finetuned on IN-1K showing better robustness than MAE and SimCLR in 8 out of 8 cases, and CAN achieving best 25-shot performance on 6 out of 9 datasets.

6.3.3 Pre-training on ImageNet-1K

Next we evaluate our method using ImageNet (IN-1K) pre-training to verify that it is also competitive in this setting. Results in Table 6.2 record the top-1 accuracy on IN-1K classification of finetuned models and linear probes. Finetuning CAN achieves 83.6% with ViT-B, outperforming other contrastive approaches such as MoCo-v3 (83.0%), and is competitive with other state-of-the-art approaches such as CAE (83.9%). The linear probe performance of CAN is 74.8% using ViT-B, beating all masked image modelling methods, the best of which is CAE with 70.4% [Chen et al., 2022]. CAN

Method	Pre-training epochs	Encoder	No Additional params.	Masked image	Finetune	Linear probe
from scratch	100	ViT-B	✓	✗	79.1	—
MoCo-v3 [Chen et al., 2021b]	300	ViT-B	✗	✗	83.0	76.7
DINO [Caron et al., 2021]	1600	ViT-B	✗	✗	82.8	78.2
CIM [Fang et al., 2022]	300	ViT-B	✗	✗	83.1	—
CAE [Chen et al., 2022]	800	ViT-B	✗	✗	83.8	68.6
CAE [Chen et al., 2022]	1600	ViT-B	✗	✗	83.9	70.4
BEiT [Bao et al., 2022]	800	ViT-B	✗	✗	83.2	37.6*
SimMIM [Xie et al., 2022b]	800	ViT-B	✓	✗	83.8	56.7
MAE [He et al., 2022]	800	ViT-B	✓	✓	83.1	—
MAE [He et al., 2022]	1600	ViT-B	✓	✓	83.6	68.0
CAN (ours)	800	ViT-B	✓	✓	83.4	74.0
CAN (ours)	1600	ViT-B	✓	✓	83.6	74.8
SimCLR† [Chen et al., 2020c]	800	ViT-L	✓	✗	83.4	73.9
MAE [He et al., 2022]	800	ViT-L	✓	✓	84.9	73.5
MAE† [He et al., 2022]	800	ViT-L	✓	✓	83.7	71.4
CAN (ours)	800	ViT-L	✓	✓	84.7	76.2

Table 6.2: **Finetune and linear probe results with pre-training on ImageNet-1K.** Note that CAN does not use multi-crop augmentation or momentum encoder. †Our implementation of [Chen et al., 2020c] and [He et al., 2022]. *Quoted from Chen et al. [2022].

is only outperformed by MoCo-v3 and DINO, which use momentum encoders and two full image views, and in the case of DINO 10 multi-crop views. Note that the *masked image* column indicates whether a method uses one or more full image views as input to the model, and the *no additional parameters* column indicates whether a method relies on other parameters besides the main encoder, e.g., from a pre-trained tokenizer, or a momentum updated target encoder. We also report results for our MAE implementation, which approximately matches the numbers reported in He et al. [2022], validating our MAE results on JFT-300M.

6.3.4 Few-shot Learning

We use linear probes to evaluate suitability of CAN for few-shot learning, following the protocol of Dosovitskiy et al. [2021b]. We use the models pre-trained on JFT-300M for 5000 epochs whose ImageNet performance is recorded in Figure 6-1. Results in Figure 6-4 for few-shot transfer learning on 9 other datasets show that the superior performance on IN-1K translates to strong performance on other tasks. We also note that our 25-shot ViT-L models beat *full-shot* both DnC and BYOL ResNet50 models

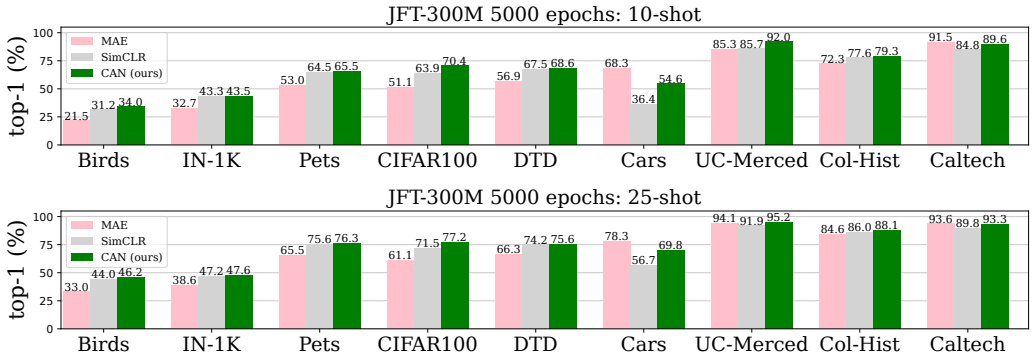


Figure 6-4: **Few-shot:** ViT-L models pre-trained on JFT-300M for 5000 epochs are evaluated on 9 datasets in few-shot setting (10-shot and 25-shot). CAN outperforms MAE and SimCLR.

(also trained for 5000 epochs on JFT-300M) on 6 out of 8 datasets [Tian et al., 2021]. See Appendix D.1 for many additional results, including pre-training on IN-21K.

6.3.5 Robustness to Distribution Shift

Finally, we consider the robustness of CAN to distribution shifts. We use ViT-L backbones trained for 5000 epochs on JFT-300M, which have been finetuned on IN-1K. Model performance is evaluated on a number of different validation sets with the same 1000 classes as IN-1K Mao et al. [2022]. Figure 6-5 reports results on the following 7 validation sets, which cover a large variety of distribution shifts: original IN-1K [Deng et al., 2009], IN-v2 [Recht et al., 2019], IN-ReaL [Beyer et al., 2020], IN-Adversarial [Hendrycks et al., 2021b], IN-Rendition [Hendrycks et al., 2021a], ObjectNet [Barbu et al., 2019]. CAN performs favourably under both JFT-300M, IN-21K and IN-1K pre-training, beating SimCLR and MAE baselines in nearly all cases. See Appendix D.1 for additional results.

6.4 Hyperparameter Analysis

We study the different components of CAN to better understand the effect of the different mechanisms, and to determine optimal parameter configurations. All ablations use ViT-B models trained for 100 epochs on IN-1K and evaluated with a linear probe

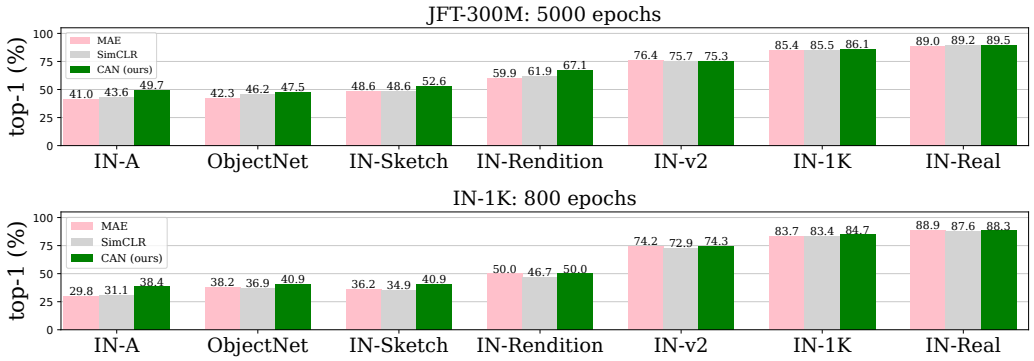


Figure 6-5: **Robustness:** Evaluating performance under distribution shifts with respect to models finetuned on IN-1K. Validation performance of ViT-L models is reported on 7 different datasets.

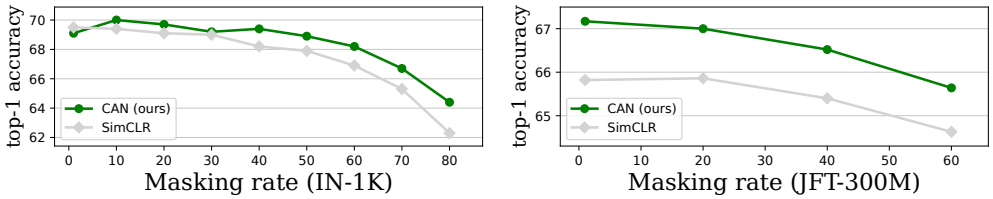


Figure 6-6: CAN and SimCLR with different masking rates. ViT-B models are pre-trained for 100 epochs on IN-1K (left), and 800 epochs on JFT-300M (right).

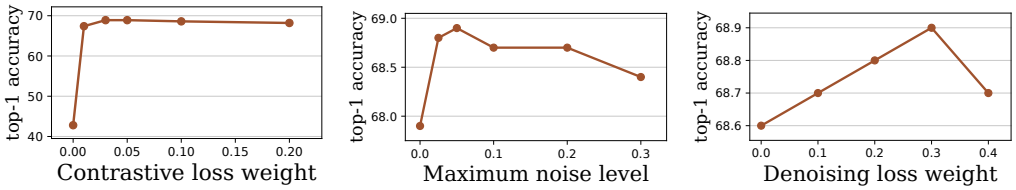


Figure 6-7: ViT-B models pre-trained on IN-1K for 100 epochs. **Left:** The best contrastive loss weight is small but non-negative. **Middle:** A wide range of σ_{max} values improve over no-noise. **Right:** Performance is not sensitive to the denoising loss weight.

Method	Contrastive loss ↓	Reconstruction loss ↓
SimCLR	9.157	—
MAE	—	0.1658
CAN (ours)	**9.143**	**0.1633**

Table 6.3: **Loss complementarity.** CAN training achieves *lower* training loss for both contrastive and reconstruction than individual training. All methods use 50% masking for fair comparison.

on IN-1K unless explicitly said otherwise. We use the best loss weights and noise level in these experiments for experiments in Section 6.3.

Complementarity of contrastive and reconstruction losses. A key hypothesis motivating our work is that contrastive learning and masked autoencoder reconstruction may not only be compatible training objectives, but are *complementary* ones. Table 6.3 compares the final training value of the contrastive $\mathcal{L}_{\text{InfoNCE}}$ and reconstruction $\mathcal{L}_{\text{rec}}$ when jointly trained (i.e., CAN) compared to only optimizing $\mathcal{L}_{\text{InfoNCE}}$ (SimCLR) or only $\mathcal{L}_{\text{rec}}$ (MAE). The results support the hypothesis: joint training achieves a lower loss on *both* objectives compared to individual training.

Ablating CAN loss terms. CAN is comprised of three components: (C) contrastive, (A) masked autoencoder, and (N) denoising losses. We ablate each of the three components in Table 6.5, setting the loss weight to zero to "remove" a component. We use ViT-B models pre-trained for 100 epochs. Removing any component leads to worse performance, with contrastive loss hurting the most.

Denoising method. Table 6.4 studies the effect of each of the components of the denoising method. We use ViT-B models trained for 100 epochs on ImageNet, and consider four settings, each adding in more parts of the method: 1) CAN with no denoising, 2) adding noise to the input only, 3) adding noise and using the denoising loss, and 4) the full method with all of the described components, including using σ_i^v as a positional encoding in the decoder. Results show that simply adding noise as a data augmentation improves performance by 0.7%, which can be improved to 1% by adding a reconstruction loss with noise level passed as an argument. The noise level

argument is necessary: the reconstruction loss without noise level argument performs worse (68.4%) than noise with no reconstruction at all (68.6%). We emphasize that the improvement from denoising comes at minimal run time and memory cost, since it uses reconstructions produced by the decoder, which in the case of MAE are simply thrown away unused. We also tried predicting the clean patches instead of noise, and found it worked poorly, corroborating similar findings in the diffusion literature.

Masking rate. Figure 6-6 reports the behavior of CAN and SimCLR under different masking rates on IN-1K and JFT-300M pre-training (for JFT-300M we use 800 epochs). The performance of SimCLR decreases as the masking rate increases, suggesting that masking is not an effective data augmentation. In contrast, performance of CAN peaks at a non-zero masking rate, but at a much lower rate than the 75% used by MAE on IN-1K. This occurs since very low masking rates are preferred by the contrastive part of CAN, but severely damage the autoencoder part as it can learn trivial solutions. The considerable efficiency improvement from masking 50% of patches more than compensates for the small drop in performance for a fixed number of epochs.

Contrastive loss weight. We vary the weighting λ_{InfoNCE} used to weight the contribution of the contrastive and reconstruction losses. Recall that larger λ_{InfoNCE} places higher weight on the contrastive loss. Results in Figure 6-7 show that the best weight is $\lambda_{\text{InfoNCE}} = 0.03$, which approximately balances the magnitudes of the two terms (see Table 6.3).

None	+noise	+noise, +loss	Full
67.9	68.6	68.4	68.9

Table 6.4: **Denoising objective.** "Full" denotes the entire method as described in Section 6.2.4

AN	CN	CA	CAN (full)
42.8	68.5	67.9	68.9

Table 6.5: **CAN loss terms.** We remove each of the three loss terms in CAN one by one.

Denoising loss weight and noise level. We study the noise level interval $[0, \sigma_{\max}]$ from which to sample input noise, and the weight λ balancing the denoising and reconstruction losses. Results in Fig. 6-7 show that the best maximum noise level is $\sigma_{\max} = 0.05$, and that similar performance

is attained for different weights on the denoising loss.

6.5 Related Work

Masked image models with Vision Transformers. The advent of the Vision Transformer (ViT) [Dosovitskiy et al., 2021c] provoked a focused effort to develop strong self-supervised learning frameworks for ViT backbones. Works such as DINO [Caron et al., 2021] and MoCo-v3 [Chen et al., 2021b] demonstrated that techniques developed with ConvNet backbones in mind could also perform competitively using ViTs after proper tuning to suit the new architecture. ViT-specific methods have emerged since then, particularly masked image modelling [Bao et al., 2022, Chen et al., 2022, Xie et al., 2022b], which use a mask-and-reconstruct training mechanism, taking inspiration from pre-training methods used in NLP [Devlin et al., 2018]. This classical idea [Ballard, 1987] is enjoying a rejuvenation thanks to favourable efficiency when combined with the vision transformer architecture [Dosovitskiy et al., 2021c]. Most notably MAE [He et al., 2022] showed that classical masked autoencoding approaches could be used to pre-train ViTs *without* passing masked tokens through the encoder. This provides a significant efficiency boost; our method similarly takes advantage of this.

Contrastive learning in computer vision. Self-supervision has received significant attention in computer vision as it offers a way to extract general purpose features without supervision. In particular, contrastive learning [van den Oord et al., 2018, Hénaff et al., 2020, Chen et al., 2020c, He et al., 2020b, Tian et al., 2020a, Chuang et al., 2020, Hénaff et al., 2021] has achieved state of the art performance by enforcing invariance to augmentations, whilst using negative samples [Robinson et al., 2021a, Ge et al., 2021] to avoid trivial solutions by spreading the embedding out uniformly on the sphere [Wang and Isola, 2020b]. The contrastive pre-training task is conceptually very different from masked image models such as MAE, which learn spatial statistical dependencies. Another distinction is that autoencoders encourage information preservation in latent representations, whilst contrastive learning could

suppress features [Chen et al., 2021a, Robinson et al., 2021b]. This leads us to hypothesize that the two approaches learn different, complementary data features. This motivates us to combine contrastive learning and masked image modelling so as to develop a reinforced pre-training task that enjoys the merits of each.

Denoising diffusion models. Denoising autoencoders (DAE) [Vincent et al., 2010] learn to reconstruct clean data given a noisy input. By learning to map low-density data regions to high-density regions, DAE learns the shape of the data manifold. This connection was made precise by Vincent [2011], who showed that DAEs learn the score-function $s(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x})$. This key observation underpins the significant recent advances in generative diffusion models, which use an estimate of the score-function to generate samples [Ho et al., 2020, Song et al., 2021]. The recent success of DAEs in generative modelling has not yet translated to representation learning, with some exceptions [Asiedu et al., 2022, Zaidi et al., 2022]. In this work we exploit a denoising autoencoder to eliminate the MAE inefficiency of reconstructing unmasked patches but never using them.

Siamese masked image modelling. Several recent works propose approaches that combine ideas from masked image modelling and Siamese self-supervised learning. For instance, Huang et al. [2022] propose a combination of contrastive and masked reconstruction objectives using one masked view, and one full (unmasked) view. Other recent works [Tao et al., 2022b, Chen et al., 2022, Assran et al., 2022] use similar asymmetric designs. The key distinction between CAN and these works is that we strike a different balance, focusing on developing a *simple*, and *efficient* method. For instance we use *two masked views* and no momentum encoder. We hope the simplicity and efficiency of CAN, and our experiments showing it's scalability, will make it easy to adapt and modify in future work.

Part II

Encoding Problem Structure into Representation Geometry

Chapter 7

Neural Networks for Eigenvector Data

Numerous machine learning models process eigenvectors, which arise in various settings including principal component analysis, matrix factorizations, and operators associated to graphs or manifolds. An important example is the use of Laplacian eigenvectors to encode information about the structure of a graph or manifold [Belkin and Niyogi, 2003, Von Luxburg, 2007, Lévy, 2006]. Positional encodings that involve Laplacian eigenvectors have recently been used to generalize Transformers to graphs [Kreuzer et al., 2021, Dwivedi and Bresson, 2021], and to improve the expressive power and empirical performance of graph neural networks (GNNs) [Dwivedi et al., 2022]. Furthermore, these eigenvectors are crucial for defining spectral operations on graphs that are foundational to graph signal processing and spectral GNNs [Ortega et al., 2018, Bruna et al., 2014].

However, there are nontrivial symmetries that should be accounted for when processing eigenvectors, as has been noted in many fields [Eastment and Krzanowski, 1982, Rustamov et al., 2007, Bro et al., 2008, Ovsjanikov et al., 2008]. For instance, if v is an eigenvector, then so is $-v$, with the same eigenvalue. More generally, if an eigenvalue has higher multiplicity, then there are infinitely many unit-norm eigenvectors that can be chosen. Indeed, a full set of linearly independent eigenvectors is only defined up to a change of basis in each eigenspace. In the case of sign invariance, for any k eigenvectors there are 2^k possible choices of sign. Accordingly, prior works on graph positional encodings randomly flip eigenvector signs during training in order

to approximately learn sign invariance [Kreuzer et al., 2021, Dwivedi et al., 2020, Kim et al., 2022]. However, learning all 2^k invariances is challenging and limits the effectiveness of Laplacian eigenvectors for encoding positional information. Sign invariance is a special case of basis invariance when all eigenvalues are distinct, but general basis invariance is even more difficult to deal with. In Appendix E.3.2, we show that higher dimensional eigenspaces are abundant in real datasets; for instance, 64% of molecule graphs in the ZINC dataset have a higher dimensional eigenspace.

In this chapter, we address the sign and basis ambiguity problems by developing new neural networks—SignNet and BasisNet. Under certain conditions, our networks are universal and can approximate any continuous function of eigenvectors with the proper invariances. Moreover, our networks are theoretically powerful for graph representation learning—they can provably approximate and go beyond both spectral graph convolutions and powerful spectral invariants, which allows our networks to express graph properties like subgraph counts that message passing neural networks cannot. Laplacian eigenvectors with SignNet and BasisNet can provably approximate many previously proposed graph positional encodings, so our networks are general and remove the need for choosing one of the many positional encodings in the literature. Experiments on molecular graph regression tasks, learning expressive graph representations, and texture reconstruction on triangle meshes illustrate the empirical benefits of our models' approximation power and invariances.

Acknowledgements. This chapter is based on [Lim et al., 2023b], which is work is in collaboration with Derek Lim, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. In particular Derek Lim made very significant contributions to this work, and it was a joy working with him.

7.1 Sign and Basis Invariant Networks

For an $n \times n$ symmetric matrix, let $\lambda_1 \leq \ldots \leq \lambda_n$ be the eigenvalues and $v_1, \ldots, v_n$ the corresponding eigenvectors, which we may assume to form an orthonormal basis. For instance, we could consider the normalized graph Laplacian $L = I - D^{-1/2}AD^{-1/2}$,

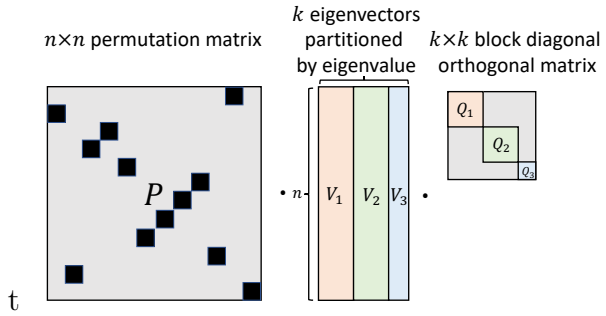


Figure 7-1: Symmetries of eigenvectors of a symmetric matrix with permutation invariances (e.g. a graph Laplacian). A neural network applied to the eigenvectors matrix (middle) should be invariant or equivariant to permutation of the rows (left product with a permutation matrix P) and invariant to the choice of eigenvectors in each eigenbasis (right product with a block diagonal orthogonal matrix $\text{Diag}(Q_1, Q_2, Q_3)$).

where $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix and D is the diagonal degree matrix of some underlying graph. For undirected graphs, L is symmetric. Nonsymmetric matrices can be handled very similarly, as we show in Appendix E.2.1.

Motivation. Our goal is to parameterize a class of models $f(v_1, \ldots, v_k)$ taking k eigenvectors as input in a manner that respects the eigenvector symmetries. This is because eigenvectors capture much information about data; for instance, Laplacian eigenvectors of a graph capture clusters, subgraph frequencies, connectivity, and many other useful properties [Von Luxburg, 2007, Cvetković et al., 1997].

A major motivation for processing eigenvector input is for graph positional encodings, which are additional features appended to each node in a graph that give information about the position of that node in the graph. These additional features are crucial for generalizing Transformers to graphs, and also have been found to improve performance of GNNs [Dwivedi et al., 2020, 2022]. Figure 7-2 illustrates a standard pipeline and the use of our SignNet within it: the input adjacency, node features, and eigenvectors of a graph are used to compute a prediction about the graph. Laplacian eigenvectors are processed before being fed into this prediction model. Laplacian eigenvectors have been widely used as positional encodings, and many works have noted that sign and/or basis invariance should be addressed in this case [Dwivedi and Bresson, 2021, Beaini et al., 2021, Dwivedi et al., 2020, Kreuzer et al., 2021, Mialon et al., 2021, Dwivedi et al., 2022, Kim et al., 2022].

Sign invariance. For any eigenvector v_i, the sign flipped $-v_i$ is also an eigenvector, so a function $f : \mathbb{R}^{n\times k} \to \mathbb{R}^{d_{\text{out}}}$ (where d_{out} is an arbitrary output dimension) should be *sign invariant*:

$$f(v_1, \ldots, v_k) = f(s_1 v_1, \ldots, s_k v_k) \tag{7.1}$$

for all sign choices $s_i \in \{-1, 1\}$. That is, we want f to be invariant to the product group $\{-1, 1\}^k$. This captures all eigenvector symmetries if the eigenvalues λ_i are distinct and the eigenvectors are unit-norm.

Basis invariance. If the eigenvalues have higher multiplicity, then there are further symmetries. Let $V_1, \ldots, V_l$ be bases of eigenspaces—i.e., $V_i = \begin{bmatrix} v_{i_1} & \ldots & v_{i_{d_i}} \end{bmatrix} \in \mathbb{R}^{n\times d_i}$ has orthonormal columns and spans the eigenspace associated with the shared eigenvalue $\mu_i = \lambda_{i_1} = \ldots = \lambda_{i_{d_i}}$. Any other orthonormal basis that spans the eigenspace is of the form $V_i Q$ for some orthogonal $Q \in O(d_i) \subseteq \mathbb{R}^{d_i \times d_i}$ (see Appendix E.6.2). Thus, a function $f : \mathbb{R}^{n\times \sum_{i=1}^{l} d_i} \to \mathbb{R}^{d_{\text{out}}}$ that is invariant to changes of basis in each eigenspace satisfies

$$f(V_1, \ldots, V_l) = f(V_1 Q_1, \ldots, V_l Q_l), \qquad Q_i \in O(d_i). \tag{7.2}$$

In other words, f is invariant to the product group $O(d_1) \times \ldots \times O(d_l)$. The number of eigenspaces l and the dimensions d_i may vary between matrices; we account for this in Section 7.1.2. As $O(1) = \{-1, 1\}$, sign invariance is a special case of basis invariance when all eigenvalues are distinct.

Permutation equivariance. For GNN models that output node features or node predictions, one typically further desires f to be invariant or equivariant to permutations of nodes, i.e., along the rows of each vector. Thus, for $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d_{\text{out}}}$, we typically require $f(PV_1, \ldots, PV_l) = Pf(V_1, \ldots, V_l)$ for any permutation matrix $P \in \mathbb{R}^{n\times n}$. Figure 7-1 illustrates all of the symmetries.

Universal Approximation. We desire universal models, which can approximate any function in a target class. Formally, we say that a class of functions $\mathcal{F}_{\text{model}}$ of domain $\mathcal{X}$ and output space $\mathcal{Y}$ *universally approximates* a class of functions $\mathcal{F}_{\text{target}}$ if for any $\epsilon > 0$, any compact $\Omega \subseteq \mathcal{X}$, and any target function $\tilde{f} \in \mathcal{F}_{\text{target}}$, there exists

an $f \in \mathcal{F}_{\text{model}}$ such that $\|f(x) - \tilde{f}(x)\| < \epsilon$ for all $x \in \Omega$.

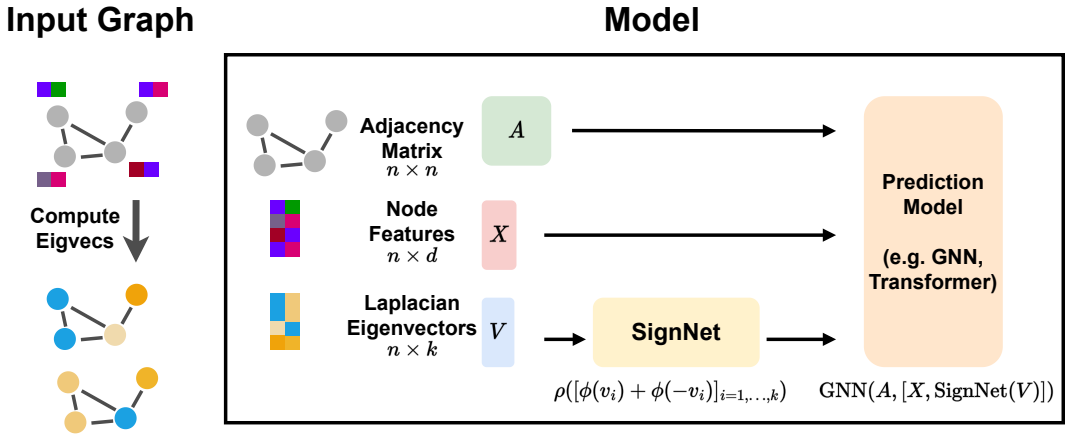


Figure 7-2: Pipeline for using node positional encodings. After processing by our SignNet, the learned positional encodings from the Laplacian eigenvectors are added as additional node features of an input graph ($[X, \text{SignNet}(V)]$ denotes concatenation). These positional encodings along with the graph adjacency and original node features are passed to a prediction model (e.g. a GNN). Not shown here, SignNet can also take in eigenvalues, node features and adjacency information if desired.

7.1.1 Warmup: Neural Networks on One Eigenspace

Before considering the general setting, we design neural networks that take a single eigenvector or eigenspace as input and are sign or basis invariant. These single subspace architectures will become building blocks for the general architectures. For one subspace, a sign invariant function is merely an even function, and is easily parameterized.

Proposition 6. *A continuous function $h : \mathbb{R}^n \to \mathbb{R}^{d_{\text{out}}}$ is sign invariant if and only if*

$$h(v) = \phi(v) + \phi(-v) \tag{7.3}$$

for some continuous $\phi : \mathbb{R}^n \to \mathbb{R}^{d_{\text{out}}}$. A continuous $h : \mathbb{R}^n \to \mathbb{R}^n$ is sign invariant and permutation equivariant if and only if (7.3) holds for a continuous permutation equivariant $\phi : \mathbb{R}^n \to \mathbb{R}^n$.

In practice, we parameterize ϕ by a neural network. Any architecture choice will ensure sign invariance, while permutation equivariance can be achieved using

elementwise MLPs, DeepSets [Zaheer et al., 2017], Transformers [Vaswani et al., 2017b], or most GNNs [Gilmer et al., 2017].

Next, we address basis invariance for a single d-dimensional subspace, i.e., we aim to parameterize maps $h : \mathbb{R}^{n\times d} \to \mathbb{R}^n$ that are (a) invariant to right multiplication by $Q \in O(d)$, and (b) equivariant to permutations along the row axis. For (a), we use the mapping $V \mapsto VV^\top$ from V to the orthogonal projector of its column space, which is $O(d)$ invariant. Mapping $V \mapsto VV^\top$ does not lose information if we treat V as equivalent to VQ for any $Q \in O(d)$. This is justified by the classical first fundamental theorem of $O(d)$ [Kraft and Procesi, 1996], which has recently been applied in machine learning by Villar et al. [2021].

Regarding (b), permuting the rows of V permutes rows and columns of $VV^\top \in \mathbb{R}^{n\times n}$. Hence, we desire the function $\phi : \mathbb{R}^{n\times n} \to \mathbb{R}^n$ on $VV^\top$ to be equivariant to simultaneous row and column permutations: $\phi(PVV^\top P^\top) = P\phi(VV^\top)$. To parameterize such a mapping from matrices to vectors, we use an invariant graph network (IGN) [Maron et al., 2018]—a neural network mapping to and from tensors of arbitrary order $\mathbb{R}^{n^{d_1}} \to \mathbb{R}^{n^{d_2}}$ that has the desired permutation equivariance. We thus parameterize a family with the requisite invariance and equivariance as follows:

$$h(V) = \mathrm{IGN}(VV^\top). \tag{7.4}$$

Proposition 7 states that this architecture universally approximates $O(d)$ invariant and permutation equivariant functions. The full approximation power requires high order tensors to be used for the IGN; in practice, we restrict the tensor dimensions for efficiency, as discussed in the next section.

Proposition 7. *Any continuous, $O(d)$ invariant $h : \mathbb{R}^{n\times d} \to \mathbb{R}^{d_{\text{out}}}$ is of the form $h(V) = \phi(VV^\top)$ for a continuous ϕ. For a compact $\mathcal{Z} \subseteq \mathbb{R}^{n\times d}$, maps of the form $V \mapsto \mathrm{IGN}(VV^\top)$ universally approximate continuous $h : \mathcal{Z} \subseteq \mathbb{R}^{n\times d} \to \mathbb{R}^n$ that are $O(d)$ invariant and permutation equivariant.*

7.1.2 Neural Networks on Multiple Eigenspaces

To develop a method for processing multiple eigenvectors (or eigenspaces), we first prove a general decomposition theorem (see Appendix E.1 for more details). Our result reduces invariance for a large product group $G_1 \times \ldots \times G_k$ to the much simpler invariances for the smaller constituent groups G_i.

Theorem 6 (Informal)**.** *Let a product of groups $G = G_1 \times \ldots \times G_k$ act on $\mathcal{X}_1 \times \ldots \times \mathcal{X}_k$. Under mild conditions, any continuous G-invariant function f can be written $f(x_1, \ldots, x_k) = \rho(\phi_1(x_1), \ldots, \phi_k(x_k))$, where ϕ_i is G_i invariant, and ϕ_i and ρ are continuous If $\mathcal{X}_i = \mathcal{X}_j$ and $G_i = G_j$, then we can take $\phi_i = \phi_j$.*

The key consequence of this result is that if we know how to design invariant models for the smaller groups G_i (of size 2 for sign invariance), then we can combine them in a simple way to get invariant models for the larger and more complex G (of size 2^k for sign invariance), without losing any expressive power. For eigenvector data, the ith eigenvector (or eigenspace) is in $\mathcal{X}_i$, and its symmetries are described by G_i. Thus, we can reduce the multiple-eigenspace case to the single-eigenspace case, and leverage the models we developed in the previous section.

SignNet. We parameterize our sign invariant network $f : \mathbb{R}^{n \times k} \to \mathbb{R}^{d_{\text{out}}}$ on eigenvectors $v_1, \ldots, v_k$ as:

$$f(v_1, \ldots, v_k) = \rho\left([\phi(v_i) + \phi(-v_i)]_{i=1}^{k}\right), \tag{7.5}$$

where ϕ and ρ are unrestricted neural networks, and $[\cdot]_i$ denotes concatenation of vectors. The form $\phi(v_i) + \phi(-v_i)$ induces sign invariance for each eigenvector. Since we do not yet impose permutation equivariance here, we term this model *Unconstrained-SignNet*.

To obtain a sign invariant *and* permutation equivariant f that outputs vectors in $\mathbb{R}^{n \times d_{\text{out}}}$, we restrict ϕ and ρ to be permutation equivariant networks from vectors to vectors, such as elementwise MLPs, DeepSets [Zaheer et al., 2017], Transformers [Vaswani et al., 2017b], or most standard GNNs. We name this permutation

equivariant version *SignNet*. If desired, we can use eigenvalues λ_i, an adjacency matrix $A \in \mathbb{R}^{n \times n}$, and node features $X \in \mathbb{R}^{n \times d_{\text{feat}}}$ by adding them as arguments to ϕ:

$$f(v_1, \ldots, v_k, \lambda_1, \ldots, \lambda_k, X) = \rho\left([\phi(v_i, \lambda_i, A, X,) + \phi(-v_i, \lambda_i, A, X)]_{i=1}^{k}\right). \tag{7.6}$$

BasisNet. For basis invariance, let $V_i \in \mathbb{R}^{n \times d_i}$ be an orthonormal basis of a d_i dimensional eigenspace. Then we parameterize our *Unconstrained-BasisNet* f by

$$f(V_1, \ldots, V_l) = \rho\left([\phi_{d_i}(V_i V_i^\top)]_{i=1}^{l}\right), \tag{7.7}$$

where each ϕ_{d_i} is shared amongst all subspaces of the same dimension d_i, and l is the number of eigenspaces (i.e., number of distinct eigenvalues, which can differ from the number of eigenvectors k). As l differs between graphs, we may use zero-padding or a sequence model like a Transformer to parameterize ρ. Again, ϕ_{d_i} and ρ are generally unrestricted neural networks. To obtain permutation equivariance, we make ρ permutation equivariant and let $\phi_{d_i} = \text{IGN}_{d_i} : \mathbb{R}^{n^2} \to \mathbb{R}^n$ be IGNs from matrices to vectors. For efficiency, we will only use matrices and vectors in the IGNs (that is, no tensors in $\mathbb{R}^{n^p}$ for $p > 2$), i.e., we use 2-IGN [Maron et al., 2018]. Our resulting *BasisNet* is

$$f(V_1, \ldots, V_l) = \rho\left([\text{IGN}_{d_i}(V_i V_i^\top)]_{i=1}^{l}\right). \tag{7.8}$$

Expressive-BasisNet. While we restrict SignNet to only use vectors and BasisNet to only use vectors and matrices, higher order tensors are generally required for universally approximating permutation equivariant or invariant functions [Keriven and Peyré, 2019, Maron et al., 2019, Maehara and NT, 2019]. Thus, we will consider a theoretically powerful but computationally impractical variant of our model, in which we replace ρ and IGN_{d_i} in BasisNet with IGNs of arbitrary tensor order. We call this variant *Expressive-BasisNet*. Universal approximation requires $\mathcal{O}(n^n)$ sized intermediate tensors [Ravanbakhsh, 2020]. We study Expressive-BasisNet due to its theoretical interest, and to juxtapose with the computational efficiency and strong expressive power of SignNet and BasisNet.

In the multiple subspace case, we can prove universality for some instances of our models through our decomposition theorem—see Section E.1 for details. For a summary of properties and more details about our models, see Appendix E.2.

7.2 Theoretical Power for Graph Representation Learning

Next, we establish that our SignNet and BasisNet can go beyond useful basis invariant and permutation equivariant functions on Laplacian eigenvectors for graph representation learning, including: spectral graph convolutions, spectral invariants, and existing graph positional encodings. Expressive-BasisNet can of course compute these functions, but this section shows that the practical invariant architectures SignNet and BasisNet can compute them as well.

7.2.1 SignNet and BasisNet strictly Generalize Spectral Graph Convolution

For node features $X \in \mathbb{R}^{n \times d_{\text{feat}}}$ and an eigendecomposition $V\Lambda V^\top$, a *spectral graph convolution* takes the form $f(V, \Lambda, X) = \sum_{i=1}^{n} \theta_i v_i v_i^\top X = V\text{Diag}(\theta)V^\top X$, for some parameters θ_i, that may optionally be continuous functions $h(\lambda_i) = \theta_i$ of the eigenvalues [Bruna et al., 2014, Defferrard et al., 2016]. This family includes important functions like heat kernels and generalized PageRanks on graphs [Li et al., 2019a]. A spectral GNN is defined as multiple layers of spectral graph convolutions and node-wise linear maps, e.g. $V\text{Diag}(\theta_2)V^\top \sigma\left(V\text{Diag}(\theta_1)V^\top XW_1\right)W_2$ is a two layer spectral GNN. It can be seen (in Appendix E.8.1) that spectral graph convolutions are permutation equivariant and sign invariant, and if $\theta_i = h(\lambda_i)$ (i.e. the transformation applied to the diagonal elements is parametric) they are additionally invariant to a change of bases in each eigenspace.

Our SignNet and BasisNet can be viewed as generalizations of spectral graph convolutions, as our networks universally approximate all spectral graph convolutions of

the above form. For instance, SignNet with $\rho(a_1, \ldots, a_k) = \sum_{i=1}^{k} a_k$ and $\phi(v_i, \lambda_i, X) = \frac{1}{2}\theta_i v_i v_i^\top X$ directly yields the spectral graph convolution. This is captured in Theorem 7, which we prove in Appendix E.8.1. In fact, we may expect SignNet to learn spectral graph convolutions well, according to the principle of algorithmic alignment [Xu et al., 2020] (see Appendix E.8.1); this is supported by numerical experiments in Appendix E.10.3, in which our networks outperform baselines in learning spectral graph convolutions.

Theorem 7. *SignNet universally approximates all spectral graph convolutions. BasisNet universally approximates all parametric spectral graph convolutions.*

In fact, SignNet and BasisNet are strictly stronger than spectral graph convolutions; there are functions computable by SignNet and BasisNet that cannot be approximated by spectral graph convolutions or spectral GNNs. This is captured in Proposition 8: our networks can distinguish bipartite graphs from non-bipartite graphs, but spectral GNNs cannot for certain choices of graphs and node signals.[1]

Proposition 8. *There exist infinitely many pairs of non-isomorphic graphs that SignNet and BasisNet can distinguish, but spectral graph convolutions or spectral GNNs cannot distinguish.*

7.2.2 BasisNet can Compute Spectral Invariants

Many works measure the expressive power of graph neural networks by comparing their power for testing graph isomorphism [Xu et al., 2019, Sato, 2020], or by comparing their ability to compute certain functions on graphs like subgraph counts [Chen et al., 2020g, Tahmasebi et al., 2020]. These works often compare GNNs to combinatorial invariants on graphs, especially the k-Weisfeiler-Leman (k-WL) tests of graph isomorphism [Morris et al., 2021].

While we may also compare with these combinatorial invariants, as other GNN works that use spectral information have done [Beaini et al., 2021], we argue that

[1] A function class $\mathcal{F}_{\text{model}}$ *distinguishes* graphs G_1, G_2 if there is an $f \in \mathcal{F}_{\text{model}}$ such that $f(G_1) \neq f(G_2)$.

it is more natural to analyze our networks in terms of *spectral invariants*, which are computed from the eigenvalues and eigenvectors of graphs. There is a rich literature of spectral invariants from the fields of spectral graph theory and complexity theory [Cvetković et al., 1997]. For a spectral invariant to be well-defined, it must be invariant to permutations and changes of basis in each eigenspace, a characteristic shared by our networks.

The simplest spectral invariant is the multiset of eigenvalues, which we give as input to our networks. Another widely studied, powerful spectral invariant is the collection of graph angles, which are defined as the values $\alpha_{ij} = \|V_i V_i^\top e_j\|_2$, where $V_i \in \mathbb{R}^{n \times d_i}$ is an orthonormal basis for the ith adjacency matrix eigenspace, and e_j is the jth standard basis vector, which is zero besides a one in the jth component. These are easily computed by our networks (Appendix E.8.3), so our networks inherit the strength of these invariants. We capture these results in the following theorem, which also lists a few properties that graph angles determine [Cvetković, 1991].

Theorem 8. *BasisNet universally approximates the graph angles α_{ij}. The eigenvalues and graph angles (and thus BasisNet) can determine the number of length 3, 4, or 5 cycles, whether a graph is connected, and the number of length k closed walks from any vertex to itself.*

Relation to WL and message passing. In contrast to this result, message passing GNNs are not able to express any of these properties (see [Arvind et al., 2020, Garg et al., 2020] and Appendix E.8.3). Although spectral invariants are strong, Fürer [2010] shows that the eigenvalues and graph angles—as well as some strictly stronger spectral invariants—are not stronger than the 3-WL test (or, equivalently, the 2-Folklore-WL test). Using our networks for node positional encodings in message passing GNNs allows us to go beyond graph angles, as message passing can distinguish all trees, but there exist non-isomorphic trees with the same eigenvalues and graph angles [Fürer, 2010, Cvetković, 1988].

7.2.3 SignNet and BasisNet Generalize Existing Graph Positional Encodings

Many graph positional encodings have been proposed, without any clear criteria on which to choose for a particular task. We prove (in Appendix E.8.2) that our efficient SignNet and BasisNet can approximate many previously used graph positional encodings, as we unify these positional encodings by expressing them as either a spectral graph convolution matrix or the diagonal of a spectral graph convolution matrix.

Proposition 9. *SignNet and BasisNet can approximate node positional encodings based on heat kernels [Feldman et al., 2022] and random walks [Dwivedi et al., 2022]. BasisNet can approximate diffusion and p-step random walk relative positional encodings [Mialon et al., 2021], and generalized PageRank and landing probability distance encodings [Li et al., 2020].*

7.3 Experiments

We demonstrate the strength of our networks in various experiments. Appendix E.2 shows simple pseudo-code and Figure 7-2 is a diagram detailing the use of SignNet as a node positional encoding.

7.3.1 Graph Regression

We study the effectiveness of SignNet for learning positional encodings (PEs) from the eigenvectors of the graph Laplacian on the ZINC dataset of molecule graphs [Irwin et al., 2012] (using the subset of 12,000 graphs from Dwivedi et al. [2020]). We primarily consider three settings: 1) No positional encoding, 2) Laplacian PE (LapPE)—the k eigenvectors of the graph Laplacian with smallest eigenvalues are concatenated with existing node features, 3) SignNet positional features—passing the eigenvectors through a SignNet and concatenating the output with node features. We parameterize SignNet by taking ϕ to be a GIN [Xu et al., 2019] and ρ to be an MLP. We sum over

Table 7.1: Results on the ZINC dataset with a 500k parameter budget. All models use edge features besides the Sparse Transformer. Numbers are the mean and standard deviation over 4 runs, each with different seeds.

Base model	Positional encoding	k	#param	Test MAE (↓)
GatedGCN	No PE	N/A	492k	$0.252_{\pm 0.007}$
	LapPE (flip)	8	492k	$0.198_{\pm 0.011}$
	LapPE (abs.)	8	492k	$0.204_{\pm 0.009}$
	LapPE (can.)	8	505k	$0.298_{\pm 0.019}$
	SignNet ($\phi(v)$ only)	8	495k	$0.148_{\pm 0.007}$
	SignNet	8	495k	$0.121_{\pm 0.005}$
	SignNet	All	491k	$\mathbf{0.100_{\pm 0.007}}$
Sparse Transformer	No PE	N/A	473k	$0.283_{\pm 0.030}$
	LapPE (flip)	16	487k	$0.223_{\pm 0.007}$
	SignNet	16	479k	$0.115_{\pm 0.008}$
	SignNet	All	486k	$\mathbf{0.102_{\pm 0.005}}$
GINE	No PE	N/A	470k	$0.170_{\pm 0.002}$
	LapPE (flip)	16	470k	$0.178_{\pm 0.004}$
	SignNet	16	470k	$0.147_{\pm 0.005}$
	SignNet	All	417k	$\mathbf{0.102_{\pm 0.002}}$
PNA	No PE	N/A	474k	$0.133_{\pm 0.011}$
	LapPE (flip)	8	474k	$0.132_{\pm 0.010}$
	SignNet	8	476k	$0.105_{\pm 0.007}$
	SignNet	All	487k	$\mathbf{0.084_{\pm 0.006}}$

ϕ outputs before the MLP when handling variable numbers of eigenvectors, so the SignNet is of the form MLP $\left(\sum_{i=1}^{l} \phi(v_i) + \phi(-v_i)\right)$ (see Appendix E.11.2 for further details). We consider four different base models that process the graph data and positional encodings: GatedGCN [Bresson and Laurent, 2017], a Transformer with sparse attention only over neighbours [Kreuzer et al., 2021], PNA [Corso et al., 2020], and GIN [Xu et al., 2019] with edge features (i.e. GINE) [Hu et al., 2020c]. The total number of parameters of the SignNet and the base model is kept within a 500k budget.

Table 7.1 shows the results. For all 4 base models, the PE learned with SignNet yields the best test MAE (mean absolute error)—lower MAE is better. This includes

Table 7.2: Comparison with SOTA methods on graph-level regression tasks. Numbers are test MAE, so lower is better. Best models within a standard deviation are bolded.

	ZINC (10K) ↓	ZINC-full ↓	Alchemy (10k) ↓
GIN [Xu et al., 2019]	$.170_{\pm.002}$	$.088_{\pm.002}$	$.180_{\pm.006}$
δ-2-GNN [Morris et al., 2020b]	$.374_{\pm.022}$	$.042_{\pm.003}$	$.118_{\pm.001}$
δ-2-LGNN [Morris et al., 2020b]	$.306_{\pm.044}$	$.045_{\pm.006}$	$.122_{\pm.003}$
SpeqNet [Morris et al., 2022]	—	—	$\mathbf{.115}_{\pm.001}$
GNN-IR [Dupty and Lee, 2022]	$.137_{\pm.010}$	—	$.119_{\pm.002}$
PF-GNN [Dupty et al., 2021]	$.122_{\pm.01}$	—	$\mathbf{.111}_{\pm.01}$
Recon-GNN [Cotta et al., 2021]	$.170_{\pm.006}$	—	$.125_{\pm.001}$
SignNet (ours)	$\mathbf{.084}_{\pm.006}$	$\mathbf{.024}_{\pm.003}$	$\mathbf{.113}_{\pm.002}$

the cases of PNA and GINE, for which Laplacian PE with random sign flipping was unable to improve performance over using no PE. Our best model is a PNA model combined with SignNet, which achieves 0.084 test MAE. Besides SignNet, we consider two non-learned approaches to resolving eigenvector sign ambiguity—sign canonicalization and element-wise absolute values (see Appendix E.11.2 for details). Results with GatedGCN show that these alternatives are not more effective than random sign flipping. We also consider an ablation of our SignNet architecture where we remove the sign invariance, using simply $\text{MLP}([\phi(v_i)]_{i=1}^k)$. Although the resulting architecture is no longer sign invariant, ϕ still processes eigenvectors independently, meaning that only two invariances (± 1) need be learned, significantly fewer than the 2^k total sign flip configurations. Accordingly, this non-sign-invariant learned positional encoding achieves a test MAE of 0.148, improving over the Laplacian PE (0.198) but falling short of the fully sign invariant SignNet (0.121). In all cases, using all available eigenvectors in SignNet significantly improves performance over using a fixed number of eigenvectors; this is notable as other works typically truncate to a fixed number of eigenvectors.

Efficiency. These significant performance improvements from SignNet come with only a slightly higher computational cost. For example, GatedGCN with no PE takes about 8.2 seconds per training iteration on ZINC, while GatedGCN with 8

eigenvectors and SignNet takes about 10.6 seconds; this is only a 29% increase in time, for a reduction of test MAE by over 50%. Also, eigenvector computation time is neglible, as we need only precompute and save the eigenvectors once, and it only takes 15 seconds to do this for the 12,000 graphs of ZINC.

Comparison with SOTA. In Table 7.2, we compare SignNet with other domain-agnostic state-of-the-art methods on graph-level molecular regression tasks on ZINC (10,000 training graphs), ZINC-full (about 250,000 graphs), and Alchemy [Chen et al., 2019a] (10,000 training graphs). SignNet outperforms all methods on ZINC and ZINC-full. Our mean score is the second best on Alchemy, and is within a standard deviation of the best. We perform much better on ZINC (.084) than other state-of-the-art positional encoding methods, like GNN-LSPE (.090) [Dwivedi et al., 2022], SAN (.139) [Kreuzer et al., 2021], and Graphormer (.122) [Ying et al., 2021].

7.3.2 Counting Substructures and Regressing Graph Properties

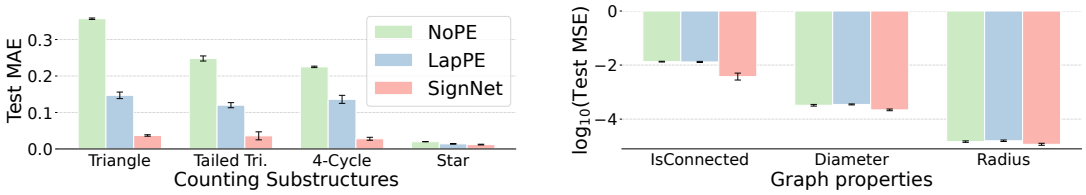


Figure 7-3: Counting substructures and regressing graph properties (lower is better). With Laplacian PEs, SignNet improves performance, while sign flip data augmentation (LapPE) is less consistent. Mean and standard deviations are reported on 3 runs. All runs use the same 4-layer GIN base model.

Substructure counts (e.g. of cycles) and global graph properties (e.g. connectedness, diameter, radius) are important graph features that are known to be informative for problems in biology, chemistry, and social networks [Chen et al., 2020g, Holland and Leinhardt, 1977]. Following the setting of Zhao et al. [2022], we show that SignNet with Laplacian positional encodings boosts the ability of simple GNNs to count substructures and regress graph properties. We take a 4-layer GIN as the base model for all settings, and for SignNet we use GIN as ϕ and a Transformer as ρ to

Table 7.3: Test results for texture reconstruction experiment on cat and human models, following the experimental setting of [Koestler et al., 2022]. We use 1023 eigenvectors of the cotangent Laplacian.

		Cat			Human		
Method	Params	PSNR ↑	DSSIM ↓	LPIPS ↓	PSNR ↑	DSSIM ↓	LPIPS ↓
Intrinsic NF	329k	34.25	.099	.189	32.29	**.119**	.330
Absolute value	329k	34.67	.106	.252	**32.42**	.132	.363
Sign flip	329k	23.15	1.28	2.35	21.52	1.05	2.71
SignNet	324k	**34.91**	**.090**	**.147**	**32.43**	.125	**.316**

handle variable numbers of eigenvectors (see Appendix E.11.4 for details). As shown in Figure 7-3, Laplacian PEs with sign-flip data augmentation improve performance for counting substructures but not for regressing graph properties, while Laplacian PEs processed by SignNet significantly boost performance on all tasks.

7.3.3 Neural Fields on Manifolds

Discrete approximations to the Laplace-Beltrami operator on manifolds have proven useful for processing data on surfaces, such as triangle meshes [Lévy, 2006]. Recently, Koestler et al. [2022] propose intrinsic neural fields, which use eigenfunctions of the Laplace-Beltrami operator as positional encodings for learning neural fields on manifolds. For generalized eigenfunctions $v_1, \dots, v_k$, at a point p on the surface, they parameterize functions $f(p) = \text{MLP}(v_1(p), \dots, v_k(p))$. As these eigenfunctions have sign ambiguity, we use our SignNet to parameterize $f(p) = \text{MLP}(\, \rho(\, [\phi(v_i(p)) + \phi(-v_i(p))]_{i=1,\dots,k}\,)\,)$, with ρ and ϕ being MLPs.

Table 7.3 shows our results for texture reconstruction experiments on all models from Koestler et al. [2022]. The total number of parameters in our SignNet-based model is kept below that of the original model. We see that the SignNet architecture improves over the original Intrinsic NF model and over other baselines — especially in the LPIPS metric, which is often a better perceptual metric than PSNR or DSSIM [Zhang et al., 2018b]. While we have not yet tested this, we believe that SignNet would allow even more improvement when learning over eigenfunctions of different models, as it

could improve transfer and generalization. See Appendix E.4.1 for visualizations and Appendix E.11.5 for more details.

7.3.4 Visualization of Learned Positional Encodings

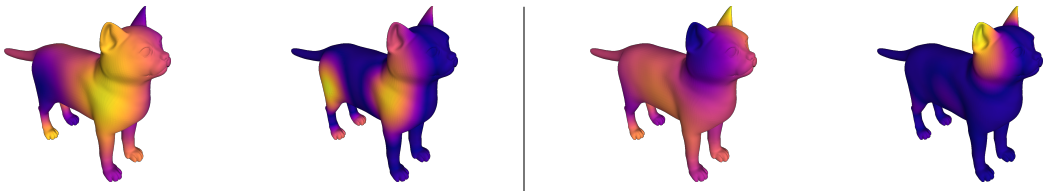


Figure 7-4: Cotangent Laplacian eigenvectors of the cat model and first principal component of $\phi(v) + \phi(-v)$ from our trained SignNet. Our SignNet encodes bilateral symmetry, which is useful for reconstruction of the bilaterally symmetric texture.

To better understand SignNet, we plot the first principal component of $\phi(v)+\phi(-v)$ for two eigenvectors on the cat model in Figure 7-4. We see that SignNet encodes bilateral symmetry and structural information on the cat model. See Appendix E.4 for more visualizations and further details.

7.4 Related Work

In this section, we review selected related work. A more thorough review is deferred to Appendix E.5.

Laplacian eigenvectors in GNNs. Various recently proposed methods in graph deep learning have directly used Laplacian eigenvectors as node positional encodings that are input to a message passing GNN [Dwivedi et al., 2020, 2022], or some variant of a Transformer that is adapted to graphs [Dwivedi and Bresson, 2021, Kreuzer et al., 2021, Mialon et al., 2021, Dwivedi et al., 2022, Kim et al., 2022]. None of these methods address basis invariance, and they only partially address sign invariance for node positional encodings by randomly flipping eigenvector signs during training.

Graph positional encodings. Other recent methods use positional encodings besides Laplacian eigenvectors. These include positional encodings based on random

walks [Dwivedi et al., 2022, Mialon et al., 2021, Li et al., 2020], diffusion kernels on graphs [Mialon et al., 2021, Feldman et al., 2022], shortest paths [Ying et al., 2021, Li et al., 2020], and unsupervised node embedding methods [Wang et al., 2022]. In particular, Wang et al. [2022] use Laplacian eigenvectors for relative positional encodings in an invariant way, but they focus on robustness, so they have stricter invariances that significantly reduce expressivity (see Appendix E.5.2 for more details). These previously used positional encodings are mostly ad-hoc, less general since they can be provably expressed by SignNet and BasisNet (see Section 7.2.3), and/or are expensive to compute (e.g., all pairs shortest paths).

7.5 Conclusion and Discussion

SignNet and BasisNet are novel architectures for processing eigenvectors that are invariant to sign flips and choices of eigenspace bases, respectively. Both architectures are provably universal under certain conditions. When used with Laplacian eigenvectors as inputs they provably go beyond spectral graph convolutions, spectral invariants, and a number of other graph positional encodings. These theoretical results are supported by experiments showing that SignNet and BasisNet are highly expressive in practice, and learn effective graph positional encodings that improve the performance of message passing graph neural networks. Initial explorations show that SignNet and BasisNet can be useful beyond graph representation learning, as eigenvectors are ubiquitous in machine learning.

Chapter 8

Learning with Discrete Functions in High Dimensions

This final chapter considers the following problem: given a discrete space—for instance sets of nodes in a graph—how might we design a neural network representations that describe this discrete space? Answering this question is important, since it would allow design of end-to-end differentiable architectures capable of learning to perform discrete computations and reason about discrete objects. But a solution is non-obvious since discrete domains are (I) not naturally differentiable, and so not immediately amenable to gradient-based optimization, and (II) incompatible with deep learning architectures that rely on representations in high-dimensional vector spaces. In this section, we address both difficulties for set functions, which capture many important discrete problems. First, we develop a framework for extending set functions onto low-dimensional continuous domains, where many extensions are naturally defined. Our framework subsumes many well-known extensions as special cases. Second, to avoid undesirable low-dimensional neural network bottlenecks, we convert low-dimensional extensions into representations in high-dimensional spaces, taking inspiration from the success of semidefinite programs for combinatorial optimization. Empirically, we observe benefits of our extensions for unsupervised neural combinatorial optimization, in particular, with high-dimensional representations.

Acknowledgements. This chapter is based on [Karalias et al., 2022], which is

work is in collaboration with Nikolaos Karalias, Andreas Loukas, and Stefanie Jegelka. In particular this work was completed in close and constant collaboration with Nikos.

8.1 Background and Motivation

Although neural networks are highly effective in solving tasks grounded in basic perception [Chen et al., 2020b, Vaswani et al., 2017b], discrete algorithmic and combinatorial tasks such as partitioning graphs, and finding optimal routes or shortest paths have proven to be more challenging. This is, in part, due to the difficulty of integrating discrete operations into neural network architectures [Battaglia et al., 2018, Bengio et al., 2021, Cappart et al., 2021a]. One immediate difficulty with functions on discrete spaces is that they are not amenable to standard gradient-based training. Another is that discrete functions are typically expressed in terms of scalar (e.g., Boolean) variables for each item (e.g., node, edge to be selected), in contrast to the high-dimensional and continuous nature of neural networks' internal representations. A natural approach to addressing these challenges is to carefully choose a function on a continuous domain that *extends* the discrete function and can be used as a drop-in replacement.

There are several important desiderata that such an extension should satisfy in order to be suited to neural network training. First, an extension should be valid, i.e., agree with the discrete function on discrete points. It should also be amenable to gradient-based optimization, and should avoid introducing spurious minima. Beyond these requirements, there is one additional critical consideration. In both machine learning and optimization, it has been observed that high-dimensional representations can make problems "easier". For example, neural networks rely on high-dimensional internal representations for representational power and to allow information to flow through gradients, and performance suffers considerably when undesirable low-dimensional bottlenecks are introduced into network architectures [Belkin et al., 2019, Veličković and Blundell, 2021]. In optimization, *lifting* to higher-dimensional spaces can make the problem more well behaved [Goemans and Williamson, 1995, Shawe-Taylor et al.,

2004, Du et al., 2018]. Therefore, extending discrete functions to *high-dimensional* domains may be critical to the effectiveness of the resulting learning process, yet it remains a relatively open problem.

With those considerations in mind, we propose a framework for constructing extensions of discrete set functions onto high-dimensional continuous spaces. The core idea is to view a continuous point $\mathbf{x}$ in space as an expectation over a distribution (that depends on $\mathbf{x}$) supported on a few carefully chosen discrete points, to retain tractability. To evaluate the discrete function at $\mathbf{x}$, we compute the expected value of the set function over this distribution. The method resulting from a principled formalization of this idea is computationally efficient and addresses the key challenges of building continuous extensions. Namely, our extensions allow gradient-based optimization and address the dimensionality concerns, allowing any function on sets to be used as a computation step in a neural network.

First, to enable gradient computations, we present a method based on a linear programming (LP) relaxation for constructing extensions on continuous domains where exact gradients can be computed using standard automatic differentiation software [Abadi et al., 2016, Bastien et al., 2012, Paszke et al., 2019]. Our approach allows task-specific considerations (e.g., a cardinalilty constraint) to be built into the extension design. While our initial LP formulation handles gradients, and is a natural formulation for explicitly building extensions, it replaces discrete Booleans with scalars in the unit interval $[0, 1]$, and hence does not yet address potential dimensionality bottlenecks. Second, to enable higher-dimensional representations, we take inspiration from classical SDP relaxations, such as the celebrated Goemans-Williamson maximum cut algorithm [Goemans and Williamson, 1995], which recast low-dimensional problems in high-dimensions. Specifically, our key contribution is to develop an SDP analog of our original LP formulation, and show how to *lift* LP-based extensions into a corresponding high-dimensional SDP-based extensions. Our general procedure for lifting low-dimensional representations into higher dimensions aligns with the neural algorithmic reasoning blueprint [Veličković and Blundell, 2021], and suggests that classical techniques such as SDPs may be effective tools for combining deep learning

with algorithmic processes more generally.

8.2 Problem Setup

Consider a ground set $[n] = \{1, \dots, n\}$ and an arbitrary function $f : 2^{[n]} \to \mathbb{R} \cup \{\infty\}$ defined on subsets of $[n]$. For instance, f could determine if a set of nodes or edges in a graph has some structural property, such as being a path, tree, clique, or independent set [Bello et al., 2016, Cappart et al., 2021a]. Our aim is to build neural networks that use such discrete functions f as an intermediate layer or loss. In order to produce a model that is trainable using standard auto-differentiation software, we consider a continuous domain $\mathcal{X}$ onto which we would like to extend f, with sets embedded into $\mathcal{X}$ via an injective map $e : 2^{[n]} \to \mathcal{X}$. For instance, when $\mathcal{X} = [0, 1]^n$ we may take $e(S) = \mathbf{1}_S$, the Boolean vector whose ith entry is 1 if $i \in S$, and 0 otherwise. Our approach is to design an extension

$$\mathfrak{F} : \mathcal{X} \to \mathbb{R} \tag{8.1}$$

of f and consider the neural network $\mathrm{NN}_2 \circ \mathfrak{F} \circ \mathrm{NN}_1$ (if f is used as a loss, NN_2 is simply the identity). To ensure that the extension is *valid* and amenable to automatic differentiation, we require that 1) it agrees with f on all discrete points: $\mathfrak{F}(e(S)) = f(S)$ for all $S \subseteq [n]$ with $f(S) < \infty$, and 2) $\mathfrak{F}$ is continuous.

There is a rich existing literature on extensions of functions on discrete domains, particularly in the context of discrete optimization [Lovász, 1983, Grötschel et al., 1981, Calinescu et al., 2011, Vondrák, 2008, Bach, 2019, Obozinski and Bach, 2012, Tawarmalani and Sahinidis, 2002]. These works provide promising tools to reach our goal of neural network training. Building on these, our method is the first to use semi-definite programming (SDP) to combine neural networks with set functions. There are, however, different considerations in the neural network setting as compared to optimization. The optimization literature often focuses on a class of set functions and aims to build extensions with desirable optimization properties, particularly

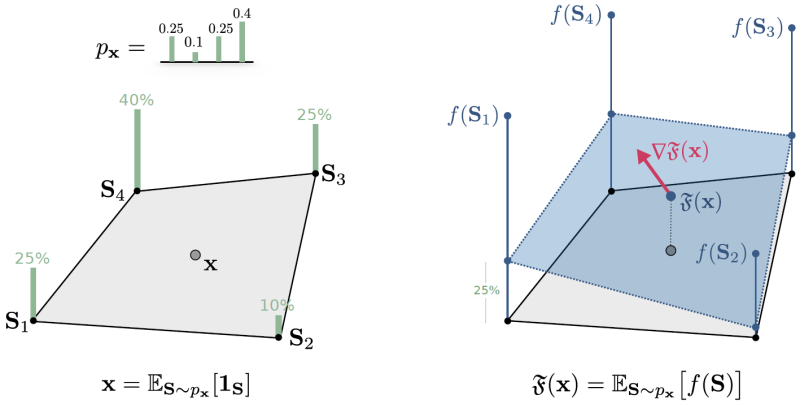


Figure 8-1: **SFEs:** Fractional points $\mathbf{x}$ are reinterpreted as expectations $\mathbf{x} = \mathbb{E}_{S\sim p_{\mathbf{x}}}[\mathbf{1}_S]$ over the distribution $p_{\mathbf{x}}(S)$ on sets. A value is assigned at $\mathbf{x}$ by exchanging the order of f and the expectation: $\mathfrak{F}(\mathbf{x})_{S\sim p_{\mathbf{x}}}[f(S)]$. Unlike f, the extension $\mathfrak{F}$ is amenable to gradient-based optimization.

convexity. We do not focus on convexity, aiming instead to develop a formalism that is as flexible as possible. Doing so maximizes the applicability of our method, and allows extensions adapted to task-specific desiderata (see Section 8.3.1).

8.3 Scalar Set Function Extensions

We start by presenting a general framework for extending set functions onto $\mathcal{X} = [0,1]^n$, where a set $S \subseteq [n]$ is viewed as the Boolean indicator vector $e(S) = \mathbf{1}_S \in \{0,1\}^n$ whose ith entry is 1 if $i \in S$ and 0 otherwise. We call extensions onto $[0,1]^n$ *scalar* since each item i is represented by a single scalar value—the ith coordinate of $\mathbf{x} \in \mathcal{X}$. These scalar extensions will become the core building blocks in developing high-dimensional extensions in Section 8.4.

A classical approach to extending discrete functions on sets represented as Boolean indicator vectors $\mathbf{1}_S$ is by computing the convex-envelope, i.e., the point-wise supremum over linear functions that lower bound f [Falk and Hoffman, 1976, Bach, 2019]. Doing so yields a convex function whose value at a point $\mathbf{x} \in [0,1]^n$ is the solution of the

following linear program (LP):

$$\widetilde{\mathfrak{F}}(\mathbf{x}) = \max_{\mathbf{z},b\in\mathbb{R}^n\times\mathbb{R}} \{\mathbf{x}^\top \mathbf{z} + b\} \text{ subject to } \mathbf{1}_S^\top \mathbf{z} + b \leq f(S) \text{ for all } S \subseteq [n]. \text{ (primal LP)}$$

The set $\mathcal{P}_f$ of all feasible solutions $(\mathbf{z}, b)$ is known as the *(canonical) polyhedron of* f [Obozinski and Bach, 2012] and can be seen to be non-empty by taking the coordinates of $\mathbf{z}$ to be sufficiently small (possibly negative). Variants of this optimization program are frequently encountered in the theory of matroids and submodular functions [Edmonds, 2003] where $\mathcal{P}_f$ is commonly known as the *submodular polyhedron* (see Appendix F.1 for an extended discussion). By strong duality, we may solve the primal LP by instead solving its dual:

$$\widetilde{\mathfrak{F}}(\mathbf{x}) = \min_{\{y_S\geq 0\}_{S\subseteq[n]}} \sum_{S\subseteq[n]} y_S f(S) \text{ subject to } \sum_{S\subseteq[n]} y_S \mathbf{1}_S = \mathbf{x}, \sum_{S\subseteq[n]} y_S = 1, \quad \text{for all } S \subseteq [n], \text{ (dual LP)}$$

whose optimal value is the same as the primal LP. The dual LP is always feasible (see e.g., the Lovász extension in Section 8.3.1). However, $\widetilde{\mathfrak{F}}$ does not necessarily agree with f on discrete points in general, unless the function is convex-extensible [Murota, 1998].

To address this important missing piece, we relax our goal from solving the dual LP to instead seeking a *feasible* solution to the dual LP that *is* an extension of f. Since the dual LP is defined for a fixed $\mathbf{x}$, a feasible solution must be a function $y_S = p_{\mathbf{x}}(S)$ of $\mathbf{x}$. If $p_{\mathbf{x}}$ were to be continuous and a.e. differentiable in $\mathbf{x}$ then the value $\sum_S p_{\mathbf{x}}(S) f(S)$ attained by the dual LP would also be continuous and a.e. differentiable in $\mathbf{x}$ since gradients flow through the coefficients $y_S = p_{\mathbf{x}}(S)$, while $f(S)$ is treated as a constant in $\mathbf{x}$. This leads us to the following definition:

Definition 3 (Scalar SFE)**.** *A scalar SFE* $\mathfrak{F}$ *of* f *is defined at a point* $\mathbf{x} \in [0,1]^n$ *by coefficients* $p_{\mathbf{x}}(S)$ *such that* $y_S = p_{\mathbf{x}}(S)$ *is a feasible solution to the dual LP. The*

extension value is given by

$$\mathfrak{F}(\mathbf{x}) = \sum_{S \subseteq [n]} p_{\mathbf{x}}(S) f(S) \tag{8.2}$$

and we require the following properties to hold for all $S \subseteq [n]$: 1) $p_{\mathbf{x}}(S)$ is a continuous function of $\mathbf{x}$ and 2) $\mathfrak{F}(\mathbf{1}_S) = f(S)$ for all $S \subseteq [n]$.

Efficient evaluation of $\mathfrak{F}$ requires that $p_{\mathbf{x}}(S)$ is supported on a small collection of carefully chosen sets S. This choice is a key inductive bias of the extension, and Section 8.3.1 gives many examples with only $O(n)$ non-zero coefficients. Examples include well-known extensions, such as the Lovász extension, as well as a number of novel extensions, illustrating the versatility of the SFE framework.

Thanks to the constraint $\sum_S y_S = 1$ in the dual LP, scalar SFEs have a natural probabilistic interpretation. An SFE is defined by a probability distribution $p_{\mathbf{x}}$ such that fractional points $\mathbf{x}$ can be written as an expectation $\mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S] = \mathbf{x}$ over discrete points using $p_{\mathbf{x}}$. The extension itself can be viewed as arising from exchanging f and the expectation operation: $\mathfrak{F}(\mathbf{x}) = \mathbb{E}_{S \sim p_{\mathbf{x}}}[f(S)]$. This interpretation is summarized in Figure 8-1.

Scalar SFEs also enjoy the property of not introducing any spurious minima. That is, the minima of $\mathfrak{F}$ coincide with the minima of f up to convex combinations. This property is especially important when training models of the form $f \circ \mathrm{NN}_1$ (i.e., f is a loss function) since $\mathfrak{F}$ will guide the network NN_1 towards the same solutions as f.

Proposition 10 (Scalar SFEs have no bad minima)**.** *If $\mathfrak{F}$ is a scalar SFE of f then:*

1. $\min_{\mathbf{x} \in \mathcal{X}} \mathfrak{F}(\mathbf{x}) = \min_{S \subseteq [n]} f(S)$
2. $\arg\min_{\mathbf{x} \in \mathcal{X}} \mathfrak{F}(\mathbf{x}) \subseteq \mathit{Hull}\big(\arg\min_{\mathbf{1}_S : S \subseteq [n]} f(S)\big)$

See Appendix F.2 for proofs.

Obtaining set solutions. Given an architecture $\mathfrak{F} \circ \mathrm{NN}_1$ and input problem instance G, we often wish to produce sets as outputs at inference time. To do this, we simply compute $\mathbf{x} = \mathrm{NN}_1(G)$, and select the set S in $\mathrm{supp}_S\{p_{\mathbf{x}}(S)\}$ with the smallest

value $f(S)$. This can be done efficiently if, as is typically the case, the cardinality of $\text{supp}_S\{p_{\mathbf{x}}(S)\}$ is small.

8.3.1 Constructing Scalar Set Function Extensions

A key characteristic of scalar SFEs is that there are many potential extensions of any given f. In this section, we provide examples of scalar SFEs, illustrating the capacity of the SFE framework for building knowledge about f into the extension. See Appendix F.3 for all proofs and further discussion.

Lovász extension. Re-indexing the coordinates of $\mathbf{x}$ so that $x_1 \geq x_2 \ldots \geq x_n$, we define $p_{\mathbf{x}}$ to be supported on the sets $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_n$ with $S_i = \{1, 2, \ldots, i\}$ for $i = 1, 2, \ldots, n$. The coefficient are defined as $y_{S_i} = p_{\mathbf{x}}(S_i) := x_i - x_{i+1}$ and $p_{\mathbf{x}}(S) = 0$ for all other sets. The resulting *Lovász extension*—known as the *Choquet integral* in decision theory [Choquet, 1954, Marichal, 2000]—is a key tool in combinatorial optimization due to a seminal result: the Lovász extension is convex if and only if f is submodular [Lovász, 1983], implying that submodular minimization can be solved in polynomial-time [Grötschel et al., 1981].

Bounded cardinality Lovász extension. A collection $\{S_i\}_{i=1}^n$ of subsets of $[n]$ can be encoded in an $n \times n$ matrix $\mathbf{S} \in \{0,1\}^{n \times n}$ whose ith column is $\mathbf{1}_{S_i}$. In this notation, the dual LP constraint $\sum_{S \subseteq [n]} y_S \mathbf{1}_S = \mathbf{x}$ can be written as $\mathbf{Sp} = \mathbf{x}$, where the ith coordinate of $\mathbf{p}$ defines $p_{\mathbf{x}}(S_i)$. The *bounded cardinality* extension generalizes the Lovász extension to focus only on sets of cardinality at most $k \leq n$. Again, re-index $\mathbf{x}$ so that $x_1 \geq x_2 \ldots \geq x_n$. Use the first k sets $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_k$, where $S_i = \{1, 2, \ldots, i\}$, to populate the first k columns of matrix $\mathbf{S}$. We add further $n - k$ sets: $S_{k+i} = \{j + i \mid j \in S_k\}$ for $i = 1, \ldots, n-k$, to fill the rest of $\mathbf{S}$. Finally, $p_{\mathbf{x}}(S_i)$ can be analytically calculated from $\mathbf{p} = \mathbf{S}^{-1}\mathbf{x}$, where $\mathbf{S}$ is invertible since it is a Toeplitz banded upper triangular matrix.

Permutations and involutory extensions. We use the same $\mathbf{S}, \mathbf{p}$ notation. Let $\mathbf{S}$ be an elementary permutation matrix. Then it is involutory, i.e., $\mathbf{SS} = \mathbf{I}$, and we may easily determine $\mathbf{p} = \mathbf{Sx}$ given $\mathbf{S}$ and $\mathbf{x}$. Note that $p_{\mathbf{x}}(S_i) = \mathbf{p}_i$ must be non-negative since $\mathbf{x}$ and $\mathbf{S}$ are non-negative entry-wise. Finally, restricting $\mathbf{x}$ to the

n-dimensional Simplex guarantees that $\|\mathbf{p}\|_1 \leq 1$, which ensures $p_{\mathbf{x}}$ is a probability distribution (any remaining mass is placed on the empty set). The extension property can be guaranteed on singleton sets as long as the chosen permutation admits a fixed point at the argmax of $\mathbf{x}$. Any elementary permutation matrix $\mathbf{S}$ with such a fixed point yields a valid SFE.

Singleton extension. Consider a set function f for which $f(S) = \infty$ unless S has cardinality one. To ensure $\mathfrak{F}$ is finite valued, $p_{\mathbf{x}}$ must be supported only on the sets $S_i = \{i\}$, $i = 1, \ldots, n$. Assuming $\mathbf{x}$ is sorted so that $x_1 \geq x_2 \ldots \geq x_n$, define $p_{\mathbf{x}}(S_i) = x_i - x_{i+1}$. It is shown in Appendix F.3 that this defines a scalar SFE, except for the dual LP feasibility. However, when using $\mathfrak{F}$ as a loss function, minimization drives $\mathbf{x}$ towards the minima $\min_{\mathbf{x}} \mathfrak{F}(\mathbf{x})$ which *are* dual feasible. So dual infeasibility is benign in this instance and we approach the feasible set from the outside.

Multilinear extension. The multilinear extension, widely used in combinatorial optimization [Calinescu et al., 2011], is supported on all sets with coefficients $p_{\mathbf{x}}(S) = \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i)$, the product distribution. In general, evaluating the multilinear extension exactly requires 2^n calls to f, but for several interesting set functions, e.g., graph cut, set cover, and facility location, it can be computed efficiently in $\widetilde{\mathcal{O}}(n^2)$ time [Iyer et al., 2014].

8.4 Neural Set Function Extensions

This section builds on the scalar SFE framework—where each item i in the ground set $[n]$ is represented by a single scalar—to develop extensions that use high-dimensional embeddings to avoid introducing low-dimensional bottlenecks into neural network architectures. The core motivation that lifting problems into higher dimensions can make them easier is not unique to deep learning. For instance, it also underlies kernel methods [Shawe-Taylor et al., 2004] and the *lift-and-project* method for integer programming [Lovász and Schrijver, 1991].

Our method takes inspiration from prior successes of semi-definite programming for combinatorial optimization [Goemans and Williamson, 1995] by extending onto

$\mathcal{X} = \mathbb{S}^n_+$, the set of $n \times n$ positive semi-definite (PSD) matrices. With this domain, each item is represented by a vector, not a scalar.

8.4.1 Lifting Set Function Extensions to Higher Dimensions

We embed sets into $\mathbb{S}^n_+$ via the map $e(S) = \mathbf{1}_S \mathbf{1}_S^\top$. To define extensions on this matrix domain, we translate the linear programming approach of Section 8.3 into an analogous SDP formulation:

$$\max_{\mathbf{Z} \succeq 0, b \in \mathbb{R}} \{\mathrm{Tr}(\mathbf{X}^\top \mathbf{Z}) + b\} \text{ subject to } \frac{1}{2}\mathrm{Tr}((\mathbf{1}_S \mathbf{1}_T^\top + \mathbf{1}_T \mathbf{1}_S^\top)\mathbf{Z}) + b \leq f(S \cap T) \text{ for } S, T \subseteq [n], \tag{primal SDP}$$

where we switch from lower case letters to upper case since we are now using matrices. Next, we show that this choice of primal SDP is a natural analog of the original LP that provides the right correspondences between vectors and matrices by proving that primal LP feasible solutions correspond to primal SDP feasible solutions with the same objective value (see Appendix F.1 for a discussion on the SDP and its dual). To state the result, note that the embedding $e(S) = \mathbf{1}_S \mathbf{1}_S^\top$ is a particular case of the correspondence $\mathbf{x} \in [0,1]^n \mapsto \sqrt{\mathbf{x}}\sqrt{\mathbf{x}}^\top$.

Proposition 11. *(Containment of LP in SDP) For any $\mathbf{x} \in [0,1]^n$, define $\mathbf{X} = \sqrt{\mathbf{x}}\sqrt{\mathbf{x}}^\top$ with the square-root taken entry-wise. Then, for any $(\mathbf{z}, b) \in \mathbb{R}^n_+ \times \mathbb{R}$ that is primal LP feasible, the pair $(\mathbf{Z}, b)$ where $\mathbf{Z} = \mathit{diag}(\mathbf{z})$, is primal SDP feasible and the objective values agree: $\mathit{Tr}(\mathbf{X}^\top \mathbf{Z}) = \mathbf{z}^\top \mathbf{x}$.*

Proposition 11 establishes that the primal SDP feasible set is a *spectrahedral lift* of the positive primal LP feasible set, i.e., feasible solutions of the primal LP lead to feasible solutions of the primal SDP. As with scalar SFEs, to define neural SFEs we consider the dual SDP:

$$\min_{\{y_{S,T} \geq 0\}} \sum_{S,T \subseteq [n]} y_{S,T} f(S \cap T) \text{ subject to } \mathbf{X} \preceq \sum_{S,T \subseteq [n]} \frac{1}{2} y_{S,T} (\mathbf{1}_S \mathbf{1}_T^\top + \mathbf{1}_T \mathbf{1}_S^\top) \text{ and } \sum_{S,T \subseteq [n]} y_{S,T} = 1 \tag{dual SDP}$$

We demonstrate that for suitable $\mathbf{X}$ this SDP has feasible solutions via an explicit construction in Section 8.4.2. This leads us to define a neural SFE which, as with scalar SFEs, is given by a feasible solution to the dual SDP that satisfies the extension property whose coefficients are continuous in $\mathbf{X}$:

Definition 4 (Neural SFE)**.** *A neural set function extension of f at a point $\mathbf{X} \in \mathbb{S}_+^n$ is defined as*

$$\mathfrak{F}(\mathbf{X}) \triangleq \sum_{S,T \subseteq [n]} p_{\mathbf{X}}(S,T) f(S \cap T),$$

where $y_{S,T} = p_{\mathbf{X}}(S,T)$ is a feasible solution to the dual SDP and for all $S, T \subseteq [n]$: 1) $p_{\mathbf{X}}(S,T)$ is continuous at $\mathbf{X}$ and 2) it is valid, i.e., $\mathfrak{F}(\mathbf{1}_S \mathbf{1}_S^\top) = f(S)$ for all $S \subseteq [n]$.

8.4.2 Constructing Neural Set Function Extensions

We constructed a number of explicit examples of scalar SFEs in Section 8.3.1. For neural SFEs we employ a different strategy. Instead of providing individual examples of neural SFEs, we develop a single recipe for converting *any* scalar SFE into a corresponding neural SFE. Doing so allows us to build on the variety of scalar SFEs and provides an additional connection between scalar and neural SFEs. In Section 8.5 we show the empirical superiority of neural SFEs over their scalar counterparts. Our construction is given in the following proposition:

Proposition 12. *Let $p_{\mathbf{x}}$ induce a scalar SFE of f. For $\mathbf{X} \in \mathbb{S}_+^n$, consider a decomposition $\mathbf{X} = \sum_{i=1}^n \lambda_i \mathbf{x}_i \mathbf{x}_i^\top$ and fix*

$$p_{\mathbf{X}}(S,T) = \sum_{i=1}^n \lambda_i \, p_{\mathbf{x}_i}(S) p_{\mathbf{x}_i}(T) \text{ for all } S, T \subseteq [n]. \tag{8.3}$$

Then, $p_{\mathbf{X}}$ defines a neural SFE $\mathfrak{F}$ at $\mathbf{X}$.

See Appendix F.4 for proof. The choice of decomposition will give rise to different extensions. Here, we instantiate our neural extensions using the eigendecomposition of $\mathbf{X}$. Since eigenvectors may not belong to $[0,1]^n$ we reparameterize by first applying a sigmoid function before computing the scalar extension distribution $p_{\mathbf{x}}$. In practice

we found that neural SFEs work just as well even without this sigmoid function—i.e., allowing scalar SFEs to be evaluated outside of $[0,1]^n$. The continuity of the neural SFE $\mathfrak{F}$ when using the eigendecomposition follows from a variant of the Davis–Kahan theorem [Yu et al., 2015], which requires the additional assumption that the eigenvalues of $\mathbf{x}$ are distinct. For efficiency, in practice we do not use all n eigenvectors, and use only the k with largest eigenvalue. This is justified by Figure 8-3, which shows that in practical applications $\mathbf{X}$ often has a rapidly decaying spectrum.

Evaluating a neural SFE requires an accessible closed-form expression, the precise form of which depends on the underlying scalar SFE. Further, from the definition of Neural SFEs we see that if a scalar SFE is supported on sets with a property that is closed under intersection (e.g., bounded cardinality), then the supporting sets of the corresponding neural SFE will also inherit that property. This implies that the neural counterparts of the Lovász, bounded cardinality Lovász, and singleton/permutation extensions have the same support as their scalar counterparts. An immediate corollary is that we can easily compute the neural counterpart of the Lovász extension which has a simple closed form:

Corollary 2. *For $\mathbf{X} \in \mathbb{S}^n_+$ consider the eigendecomposition $\mathbf{X} = \sum_{i=1}^n \lambda_i \mathbf{x}_i \mathbf{x}_i^\top$. Let $p_{\mathbf{x}_i}$ be as in the Lovász extension: $p_{\mathbf{x}_i}(S_{ij}) = \sigma(x_{i,j}) - \sigma(x_{i,j+1})$, where σ is the sigmoid function, and $\mathbf{x}_i$ is sorted so $x_{i,1} \geq \ldots \geq x_{i,n}$ and $S_{ij} = \{1, \ldots, j\}$, with $p_{\mathbf{x}_i}(S) = 0$ for all other sets. Then, the neural Lovász extension is:*

$$\mathfrak{F}(\mathbf{X}) = \sum_{i,j=1}^{n} \lambda_i p_{\mathbf{x}_i}(S_{ij}) \cdot \left(p_{\mathbf{x}_i}(S_{ij}) + 2 \sum_{\ell:\ell>j} p_{\mathbf{x}_i}(S_{i\ell}) \right) \cdot f(S_{ij}). \tag{8.4}$$

Complexity and obtaining sets as solutions. In general, the neural SFE relies on all pairwise intersections $S \cap T$ of the scalar SFE sets, requiring $O(m^2)$ evaluations of f when the scalar SFE is supported on m sets. However, when the scalar SFE is supported on a family of sets that is closed under intersection—e.g., the Lovász and singleton extensions—the corresponding neural SFE requires only $O(m)$ function evaluations. Discrete solutions can be obtained efficiently by returning the best set out of all scalar SFEs $p_{\mathbf{x}_i}$.

	Maximum Clique				
	ENZYMES	PROTEINS	IMDB-Binary	MUTAG	COLLAB
Straight-through [Bengio et al., 2013]	$0.725_{\pm 0.268}$	$0.722_{\pm 0.26}$	$0.917_{\pm 0.253}$	$0.965_{\pm 0.162}$	$0.856_{\pm 0.221}$
Erdős [Karalias and Loukas, 2020]	$0.883_{\pm 0.156}$	$0.905_{\pm 0.133}$	$0.936_{\pm 0.175}$	$1.000_{\pm 0.000}$	$0.852_{\pm 0.212}$
REINFORCE [Williams, 1992]	$0.751_{\pm 0.301}$	$0.725_{\pm 0.285}$	$0.881_{\pm 0.240}$	$1.000_{\pm 0.000}$	$0.781_{\pm 0.316}$
Lovász scalar SFE	$0.723_{\pm 0.272}$	$0.778_{\pm 0.270}$	$0.975_{\pm 0.125}$	$0.977_{\pm 0.125}$	$0.855_{\pm 0.225}$
Lovász neural SFE	$0.933_{\pm 0.148}$	$0.926_{\pm 0.165}$	$0.961_{\pm 0.143}$	$1.000_{\pm 0.000}$	$0.864_{\pm 0.205}$
	Maximum Independent Set				
	ENZYMES	PROTEINS	IMDB-Binary	MUTAG	COLLAB
Straight-through [Bengio et al., 2013]	$0.505_{\pm 0.244}$	$0.430_{\pm 0.252}$	$0.701_{\pm 0.252}$	$0.721_{\pm 0.257}$	$0.331_{\pm 0.260}$
Erdős [Karalias and Loukas, 2020]	$0.821_{\pm 0.124}$	$0.903_{\pm 0.114}$	$0.515_{\pm 0.310}$	$0.939_{\pm 0.069}$	$0.886_{\pm 0.198}$
REINFORCE [Williams, 1992]	$0.617_{\pm 0.214}$	$0.579_{\pm 0.340}$	$0.899_{\pm 0.275}$	$0.744_{\pm 0.121}$	$0.053_{\pm 0.164}$
Lovász scalar SFE	$0.311_{\pm 0.289}$	$0.462_{\pm 0.260}$	$0.716_{\pm 0.269}$	$0.737_{\pm 0.154}$	$0.302_{\pm 0.238}$
Lovász neural SFE	$0.775_{\pm 0.155}$	$0.729_{\pm 0.205}$	$0.679_{\pm 0.287}$	$0.854_{\pm 0.132}$	$0.392_{\pm 0.253}$

Table 8.1: **Unsupervised neural combinatorial optimization**: Approximation ratios for combinatorial problems. Values closer to 1 are better (↑). Neural SFEs are competitive with other methods, and consistently improve over vector SFEs.

8.5 Experiments

We experiment with SFEs as loss functions in neural network pipelines on discrete objectives arising in combinatorial and vision tasks. For combinatorial optimization, SFEs network training with a continuous version of the objective without supervision. For supervised image classification, they allow us to directly relax the training error instead of optimizing a proxy like cross entropy.

8.5.1 Unsupervised Neural Combinatorial Optimization

We begin by evaluating the suitability of neural SFEs for unsupervised learning of neural solvers for combinatorial optimization problems on graphs. We use the ENZYMES, PROTEINS, IMDB, MUTAG, and COLLAB datasets from the TUDatasets benchmark [Morris et al., 2020a], using a 60/30/10 split for train/test/val. We test on two problems: finding maximum cliques, and maximum independent sets. We compare with three neural network based methods. We compare to two common approaches for

backpropogating through discrete functions: the REINFORCE algorithm [Williams, 1992], and the Straight-Through estimator [Bengio et al., 2013]. The third is the recently proposed probabilistic penalty relaxation [Karalias and Loukas, 2020] for combinatorial optimization objectives. All methods use the same GNN backbone, comprising a single GAT layer [Veličković et al., 2018] followed by multiple gated graph convolution layers [Li et al., 2015].

In all cases, given an input graph $G = (V, E)$ with $|V| = n$ nodes, a GNN produces an embedding for each node: $\mathbf{X} \in \mathbb{R}^{n \times d}$. For scalar SFEs $d = 1$, while for neural SFEs we consider $\mathbf{X}\mathbf{X}^\top$ in order to produce an $n \times n$ PSD matrix, which is passed as input to the SFE $\mathfrak{F}$. The set function f used is problem dependent, which we discuss below. Finally, see Appendix F.6 for training and hyper-parameter optimization details.

Maximum Clique. A set $S \subseteq V$ is a clique of $G = (V, E)$ if $(i, j) \in E$ for all $i, j \in S$. The MaxClique problem is to find the largest set S that is a clique: i.e., $f(S) = |S| \cdot \mathbf{1}\{S \text{ a clique}\}$.

Maximum Independent Set (MIS). A set $S \subseteq V$ is an independent set of $G = (V, E)$ if $(i, j) \notin E$ for all $i, j \in S$. The goal is to find the largest S in the graph that is independent, i.e., $f(S) = |S| \cdot \mathbf{1}\{S \text{ an ind. set}\}$. MIS differs significantly from MaxClique due to its high heterophily.

Results. Table 8.1 displays the mean and standard deviation of the approximation ratio $f(S)/f(S^*)$ of the solver solution S and an optimal S^* on the test set graphs. The neural Lovaśz extension outperforms its scalar counterpart in 8 out of 10 cases, often by significant margins, for instance improving a score of 0.778 on PROTEINS MaxClique to 0.926. The neural SFE proved effective at boosting poor scalar SFE performance, e.g., 0.311 on ENZYMES MIS, to the competitive performance of 0.775. Neural Lovaśz outperformed or equalled and straight-through in 9 out of 10 cases, and the method of Karalias and Loukas [2020] in 6 out of 10.

8.5.2 Constraint Satisfaction Problems

Constraint satisfaction problems ask if there exists a set satisfying a given set of conditions [Kumar, 1992, Cappart et al., 2021b]. In this section, we apply SFEs to the

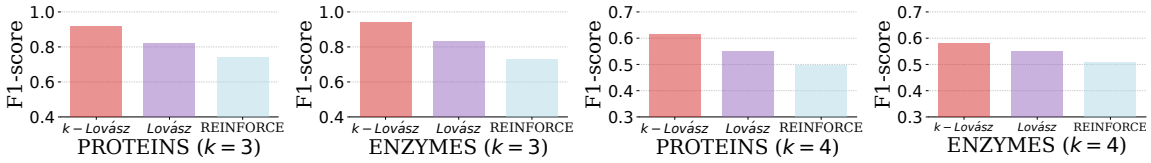


Figure 8-2: k**-clique constraint satisfaction:** higher F1-score is better. The k-bounded cardinality Lovasz extension is better aligned with the task and significantly improves over the Lovász extension.

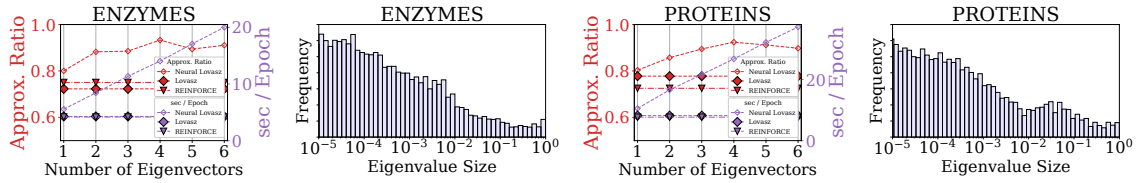


Figure 8-3: **Left:** Runtime and performance of neural SFEs on MaxClique using different numbers of eigenvectors. **Right:** Histogram of spectrum of matrix $\mathbf{X}$, outputted by a GNN trained on MaxClique.

k-clique problem: given a graph, determine if it contains a clique of size k or more. We test on the ENZYMES and PROTEINS datasets. Since satisfiability is a binary classification problem we evaluate using F1 score.

Results. Figure 8-2 shows that by specifically searching over sets of size k using the cardinality constrained Lovász extension from Section 8.3.1, we significantly improve performance compared to the Lovász extension, and REINFORCE. This illustrates the value of SFEs in allowing task-dependent considerations (in this case a cardinality constraint) to be built into extension design.

8.5.3 Training Error as a Classification Objective

During training the performance of a classifier h is typically assessed using the training error $\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{y_i \neq h(x_i)\}$. Since training error itself is non-differentiable, it is standard to train h to optimize a differentiable surrogate such as the cross-entropy loss. Here we offer an alternative training method by continuously extending the non-differentiable mapping $\hat{y} \mapsto \mathbf{1}\{y_i \neq \hat{y}\}$. This map is a set function defined on single item sets, so we use the singleton extension (definition in Section 8.3.1). Our goal is to demonstrate that the resulting differentiable loss function closely tracks the training

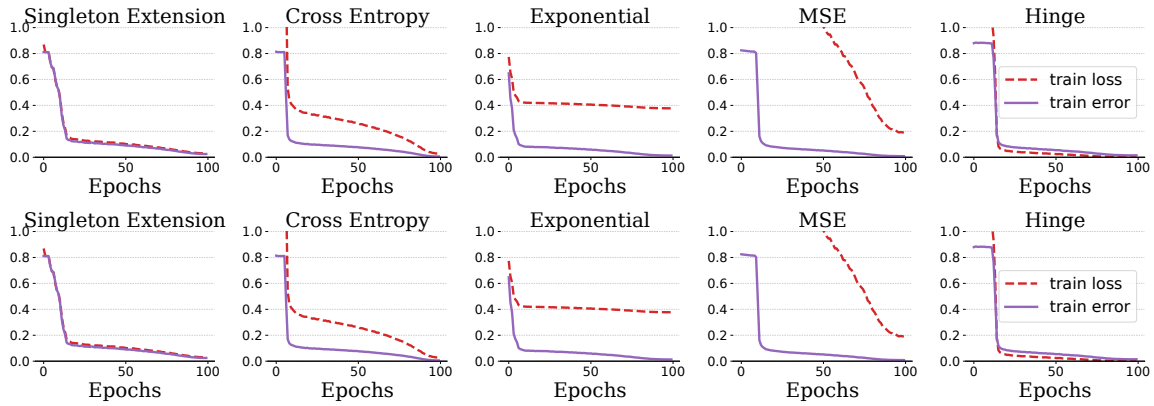


Figure 8-4: Top: CIFAR10. Bottom: SVHN. The singleton extension loss (left) is the only loss that approximates the true non-differentiable training error at the same numerical scale.

error, and can be used to minimize it. We do not focus on test time generalization. Figure 8-4 shows the results. The singleton extension loss (left plot) closely tracks the true training error at the same numerical scale, unlike other common loss functions (see Appendix F.7 for setup details). While we leave further consideration to future work, training error extensions may be useful for model calibration [Kennedy and O'Hagan, 2001] and uncertainty estimation [Abdar et al., 2021].

8.5.4 Ablations

Number of Eigenvectors. Figure 8-3 compares the runtime and performance of neural SFEs using only the top-k eigenvectors from the eigendecomposition $\mathbf{X} = \sum_{i=1}^{n} \lambda_i \mathbf{x}_i \mathbf{x}_i^\top$ with $k \in \{1, 2, 3, 4, 5, 6\}$ on the maximum clique problem. For both ENZYMES and PROTEINS, performance increases with k—easily outperforming scalar SFEs and REINFORCE—until saturation around $k = 4$, while runtime grows linearly with k. Histograms of eigenvalues produced by trained networks show a rapid decay in the spectrum, suggesting that smaller eigenvalues have little effect on $\mathfrak{F}$.

Comparison to Naive High-Dimensional Extension. We compare neural SFEs to a naive high-dimensional alternative which, given an $n \times d$ matrix $\mathbf{X}$ simply computes a scalar SFE on each column independently and sums them up. This naive function design is not an extension, and the dependence on the d dimensions is linearly

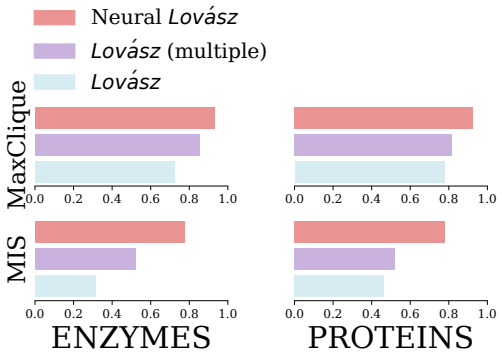


Figure 8-5: Neural SFEs outperform a naive alternative high-dimensional extension.

separable, in contrast to the complex non-linear interactions between columns of $\mathbf{X}$ in neural SFEs. Figure 8-5 shows that this naive extension, whilst improving over one-dimensional extensions, performs considerably worse than neural SFEs.

8.6 Related Work

Neural combinatorial optimization Our experimental setup largely follows recent work on unsupervised neural combinatorial optimization [Karalias and Loukas, 2020, Schuetz et al., 2022, Toenshoff et al., 2021, Amizadeh et al., 2018], where continuous relaxations of discrete objectives are utilized. In that context, it is important to take into account the key conceptual and methodological differences of our approach. For instance, in the unsupervised *Erdős goes neural* (EGN) framework from Karalias and Loukas [2020], the probabilistic relaxation and the proposed choice of distribution can be viewed as instantiating a multilinear extension. As explained earlier, this extension is costly in the general case (since f must be evaluated 2^n times, and summed) but can be computed efficiently in closed form in certain cases. On the other hand, our extension framework offers multiple options for efficiently computable extensions without imposing any further conditions on the set function. For example, one could efficiently (linear time in n) compute the scalar and neural Lovász extensions of any set function with only black-box access to the function. This renders our framework more broadly applicable. Furthermore, EGN incorporates the problem constraints additively in the loss function. In contrast to that, our extension framework does not

require any commitment to a specific formulation in order to obtain a differentiable loss. For general background on neural combinatorial optimization, we refer the reader to the surveys [Bengio et al., 2021, Cappart et al., 2021a, Mazyavkina et al., 2021].

Chapter 9

Conclusion

This thesis explores the relationship between problem structure and the internal geometry of neural network representations. Along the way, we encountered close connections to a number of core considerations in AI system development, including learning speed, data efficiency, model reliability, and discrete reasoning capabilities.

Due to the centrality of representation geometry in learning, there are necessarily many directions and open questions that this thesis did not cover in detail. The goal of this final chapter is to layout several interesting features of this wider landscape by revisiting some threads that were not picked up earlier in the thesis.

9.1 What is the Full Potential of Hard Negatives?

The first major part of this thesis studies ways to generate hard negative samples in contrastive learning. Hard negatives were largely motivated from the point of view of *training efficiency*, namely that harder negatives are more "informative", and therefore more useful to focus on the model during training.

But efficiency is not the only reason why hard negatives may be useful. A completely different viewpoint is that judicious selection of negatives can be used to control which features of the data the model does and does not learn. This was explored in a preliminary way in Chapter 4, however the full potential of this idea remains largely untapped.

Concretely, a common problem in many machine learning contexts is that there are too many features that spuriously correlate with the target of interest. Careful selection of negatives may offer a tool for pruning the space of spuriously correlating features by providing signal to the model on which correlated features are spurious and which are not. Indeed, by selecting negative pairs x, and x^- that have certain overlapping features, the model is forced to *not* use these features to distinguish samples, guiding the learning process to search for other features that the designed *does* want the model to learn.

But how to obtain such carefully drawn negatives in the first place? This may be a significant challenge, since necessarily certain prior knowledge is required. However, there are guiding examples in the literature. One example is provided by Murphy et al. [2022], who seek to learn representations of immunohistochemical images in order to identify the cell type specificity of certain protein markers. In this setting, there are multiple images per donor—i.e., the same section of tissue—but stained with different antibodies so that different cells are highlighted for different stains. A natural, but spurious, positive-pair correlation arises when using images from the same donor as positive pairs for contrastive learning, namely that the pattern of cells in the common underlying tissue can be used to identify the similarity of the positive pair. This is a *spurious* correlation since it is independent of the downstream question of interest: which antibodies stain which types of cell? In order to address this, Murphy et al. [2022] notice that it is possible to draw negative pairs from the *same donor*. This approach produces negative pairs share the exact same pattern of cells, thereby forcing the models to look for features that account for the similarity of positive pairs to look beyond the spurious pattern of cells.

How can the success of this approach be extended to other settings? One option, as in the case of Murphy et al. [2022], is to keep an eye out for useful structure in the data that is available—in their case, the presence of multiple stains for the same donor tissue. Another option may be to create *synthetic* negatives. Indeed suppose you could design an augmentation $a : \mathcal{X} \to \mathcal{X}$ such that x and $a(x)$ *do not* share the same underlying semantics. Note, this is the exact opposite role that augmentations usually

perform in contrastive learning—the production of pairs of semantically matching samples. This means that x and $a(x)$ should be treated as a negative pair. So long as the augmentation a is small, so that x and $a(x)$ still share many features, then the slight differences in the pair $(x, a(x))$ pinpoints features that are relevant to the task of interest. A concrete example of this is in contrastive learning of SAT formula embeddings, as considered by Duan et al. [2022]. The ultimate goal in this setting is to determine whether a Boolean formula is satisfiable—i.e., whether there exists a true/false assignment for the variables such that the entire formula evaluates to true—or not. SAT formulas can be represented as graphs. Importantly small local changes in the graph, such as adding and removing nodes and edges, may change the satisfiability of the formula, and whats more, whether the satisfiabilitiy has changed can be efficiently computed from the original problem. More broadly, many combinatorial problems enjoy *efficient re-computation* of solutions under local modifications of a problem instance. Local re-computation makes combinatorial problems an especially promising setting for testing this hard negatives approach.

9.2 Nuances in Understanding of Neural Networks for Eigenvector Data

Chapter 7 introduced provably powerful architectures for processing eigenvector data. Although our proposed design makes significant progress in characterizing and designing neural networks for eigenvectors, there are remain a number of interesting questions worth further study in this area that we were not able to address.

Robustness. How robust are our models to errors or perturbations to input eigenvectors, and do we even want robustness? One setting in which robust models are desirable is in learning on noisy networks, whose connectivity structure is not to be trusted entirely. In this setting it is desirable to know whether or not our SignNet more, or another sign invariant architecture is likely to produce wildly different outputs as nodes or edges are modified slightly. Wang et al. [2022] design PEG, an (non-universal)

eigenvector architecture that is guaranteed to be Lipschitz with respect to perturbation of both the eigenvectors and the underlying node features.

However, two questions remain. First, what guarantees does PEG yield on stability with respect to pertubations of edges and nodes of the underlying graph, as opposed to pertubations of the Laplacian as the current analysis considers. This can be understood by directly drawing from existing spectral graph theory under graph perturbations.

Second, there are settings in which robustness is the opposite of what is needed. A salient example arises in molecular chemistry, where certain molecules are highly similar in their structure, but have very significant differences in potency [van Tilborg et al., 2022]. These rapid variations in the landscape of molecules are known as *activity cliffs*. Activity cliffs pose a major challenge in computational chemistry—especially deep learning approaches—as predicting activity cliffs requires fitting a highly non-smooth target function. This raises an interesting question for neural network design—how to design neural networks that are suited to learning such cliffs.

9.3 Laplacian Eigenvectors as Universal Descriptions of Positions

We used our eigenvector networks, such as SignNet, to learn positional encodings for graph data. Is this a specialist positional encoding that fundamentally from positional encodings used for words and images?

In fact there is a close connection between Laplacian eigenvectors and *sinusoidal* positional encodings, that are often used in language modeling as their cyclical nature enables extrapolation to sequences during inference that are longer than the sequences seen during training [Vaswani et al., 2017b]. Sinusoidal positional encodings $p_n = \sin(\omega n)$ constitute the terms of the solution of Laplace's equation

$$\nabla^2 f = 0$$

Laplace's equation is in turn intimately tied to the Laplacian matrix, which can be

thought of as the negative of the discretization of the Laplace operator ∇^2. By making this connection more precise, it appears that eigenvector positional encodings may also generalize to structures beyond graphs, including sequences and images (2D lattices of patches).

This connection points to a possible consolidation of machine learning methodologies across data modalities. Indeed considerable consolidation has already occurred, with many divers problems solvable using the same network architecture—the Transformer—combined with a domain specific positional encoding.

9.4 Broad Outlook

Where does representation geometry fit into the wider landscape of machine learning methods? This question is especially pertinent in an era of methodological consolidation into a few key architectures and training pipleines that dominate due to favorable scaling laws. Here I suggest three paths through which geometric thinking can play a part.

1. Pre-training objective. Underlying a broad scope of pre-training approaches is a *Goldilocks principle*: pre-training tasks must be hard enough that it requires extraction of complex and rich data features, but not too hard that it is not possible to learn. One key knob that controls difficulty is the amount of redundancy between between parts of an input. Redundancy arises from multiple sources, one important example of which is geometric structure of data. For instance, predicting masked nodes in a homophilous graph is easier, potentially too easy, if many of its neighbours are known. In the setting of data with complex structure, pre-training objectives that take advantage of the problem structure remain underdeveloped, but promising directions for integrating domain knowledge into otherwise largely generic Transformer-based training pipelines.

2. Positional encoding. Our sign invariant networks (Chapter 7) provide one perspective on how to integrate geometric structure, in this case symmetry, into positional encodings, and Section 9.3 elaborates on the potential broader impact of

such methods. But this is not the only possibility. Generally speaking, a Transformer with positional encoding decomposes the learning task into two parts: processing *structure* or *location* of data (the positional encoding) and processing the *substance* of different parts of data (self-attention). I believe that the positional encoding is much more amenable to geometric thinking than self-attention. Indeed, positional encodings are a general framework for describing the structure of data, and in cases that data has non-trivial structure (as in the case of graphs) the question of how to design positional encodings becomes both interesting and important.

3. Tokenizers. Most existing Transformer tokenizers identify commonly co-occurring pairs and tuples of letters to form tokens (for instance, the popular byte pair encoding, or BPE, used in GPT-3). Whilst BPE and its variations are without doubt well suited to natural language, as GPT-style models are developed across increasingly broad problem settings more domain specific alternatives appear likely to emerge. Indeed, whilst the tokenizer may seem a low-level detail, it is empirically vitally important. Furthermore it is an opportunity to inject domain knowledge into the training pipeline, since choosing tokens amounts to deciding what constitutes a semantically meaningful chunk of information. Situations in which sequence data has structure—for instance the many equivalent SMILES descriptions of a molecule—are prime candidates for specialized tokenizers that better reflect their structure. Furthermore, tokenization interacts with positional encoding approaches—if one token describes multiple atoms, how can the spatial position of atoms be included as input to the model? Integrating complex positional encodings with complex tokenizers remains largely unexplored.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *Int. Conference on Learning Representations (ICLR)*, 2017.

Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Learning to solve circuit-sat: An unsupervised differentiable approach. In *International Conference on Learning Representations*, 2018.

Federico Ardila, Carolina Benedetti, and Jeffrey Doker. Matroid polytopes and their volumes. *Discrete & Computational Geometry*, 43(4):841–854, 2010.

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. In *Int. Conference on Machine Learning (ICML)*, pages 5628–5637, 2019.

Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On

weisfeiler-leman invariance: Subgraph counts and related graph properties. In *Journal of Computer and System Sciences*, volume 113, pages 42–59. Elsevier, 2020.

Emmanuel Brempong Asiedu, Simon Kornblith, Ting Chen, Niki Parmar, Matthias Minderer, and Mohammad Norouzi. Decoder denoising pretraining for semantic segmentation. *preprint arXiv:2205.11423*, 2022.

Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Michael Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *preprint arXiv:2204.07141*, 2022.

László Babai, Dmitry Y Grigoryev, and David M Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 310–324, 1982.

Francis Bach. Submodular functions: from discrete to continuous domains. *Mathematical Programming*, 175(1):419–459, 2019.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15535–15545, 2019.

Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *Int. Conference on Learning Representations (ICLR)*, volume 8, 2020.

Fabian Ball and Andreas Geyer-Schulz. How symmetric are real-world graphs? a large-scale study. *Symmetry*, 10(1):29, 2018.

Dana H Ballard. Modular learning in neural networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*, volume 647, pages 279–284, 1987.

Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image transformers. In *Int. Conference on Learning Representations (ICLR)*, 2022.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. *preprint arXiv:2105.04906*, 2021.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. In *Int. Conference on Machine Learning (ICML)*, pages 748–758. PMLR, 2021.

Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473, 2018.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio.

Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *Int. Conference on Learning Representations (ICLR)*, volume 10, 2022.

Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with ImageNet? In *preprint arXiv:2006.07159*, 2020.

Sangnie Bhardwaj, Willie McClinton, Tongzhou Wang, Guillaume Lajoie, Chen Sun, Phillip Isola, and Dilip Krishnan. Steerable equivariant representation learning. *preprint arXiv:2302.11349*, 2023.

Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. In *IEEE transactions on pattern analysis and machine intelligence*. IEEE, 2021.

Jeff Bilmes. Submodularity in machine learning and artificial intelligence. *arXiv preprint arXiv:2202.00132*, 2022.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

Ben Blum-Smith and Soledad Villar. Equivariant maps from invariant functions. *preprint arXiv:2209.14991*, 2022.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Advances in Neural Information Processing Systems*, 34:2625–2640, 2021.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 26, 2013.

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

Xavier Bresson and Thomas Laurent. Residual gated graph ConvNets. In *preprint arXiv:1711.07553*, 2017.

Rasmus Bro, Evrim Acar, and Tamara G Kolda. Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 22(2):135–140, 2008.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *preprint arXiv:2104.13478*, 2021.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *Int. Conference on Learning Representations (ICLR)*, volume 2, 2014.

Chen Cai and Yusu Wang. Convergence of invariant graph networks. In *preprint arXiv:2201.10129*, 2022.

G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SIAM J. Computing*, 40(6), 2011.

Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544*, 2021a.

Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4348–4355. International Joint Conferences on Artificial Intelligence Organization, 8 2021b. doi: 10.24963/ijcai.2021/595. URL https://doi.org/10.24963/ijcai.2021/595. Survey Track.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9912–9924, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Int. Conference on Computer Vision (ICCV)*, pages 9650–9660, 2021.

Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. In *preprint arXiv:2005.03675*, 2020.

Guangyong Chen, Pengfei Chen, Chang-Yu Hsieh, Chee-Kong Lee, Benben Liao, Renjie Liao, Weiwen Liu, Jiezhong Qiu, Qiming Sun, Jie Tang, et al. Alchemy: A quantum

chemistry dataset for benchmarking ai models. *arXiv preprint arXiv:1906.09427*, 2019a.

Mark Chen, Alec Radford, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *Int. Conference on Machine Learning (ICML)*, 2020a.

Ting Chen and Lala Li. Intriguing properties of contrastive losses. *preprint arXiv:2011.02803*, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020b.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Int. Conference on Machine Learning (ICML)*, pages 1597–1607. PMLR, 2020c.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020d.

Ting Chen, Calvin Luo, and Lala Li. Intriguing properties of contrastive losses. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 11834–11845, 2021a.

Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. In *preprint arXiv:2202.03026*, 2022.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021a.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758, 2021b.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020e.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *preprint arXiv:2003.04297*, 2020f.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Int. Conference on Computer Vision (ICCV)*, pages 9640–9649, 2021b.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with GNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 1589–15902, 2019b.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 10383–10395, 2020g.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *Int. Conference on Learning Representations (ICLR)*, volume 9, 2021.

Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory (COLT)*, pages 1305–1338, 2020.

Gustave Choquet. Theory of capacities. In *Annales de l'institut Fourier*, volume 5, pages 131–295, 1954.

Marthinus Christoffel, Gang Niu, and Masashi Sugiyama. Class-prior estimation for learning from positive and unlabeled data. In *Asian Conference on Machine Learning*, pages 221–236, 2016.

Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8765–8775, 2020.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. In *ICLR Workshop on Physics for Machine Learning*, 2023.

Fan Chung. *Spectral graph theory*. American Mathematical Soc., 1997.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Int. Conference on Machine Learning (ICML)*, pages 2990–2999. PMLR, 2016.

Elijah Cole, Xuan Yang, Kimberly Wilber, Oisin Mac Aodha, and Serge Belongie. When does contrastive visual representation learning work? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14755–14764, 2022.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 13260–13271, 2020.

Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine learning*, 20(3):273–297, 1995.

Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Ekin Dogus Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3008–3017, 2020.

Charles W Curtis and Irving Reiner. *Representation theory of finite groups and associative algebras*, volume 356. American Mathematical Soc., 1966.

Dragoš Cvetković. Constructing trees with given eigenvalues and angles. *Linear Algebra and its Applications*, 105:1–8, 1988.

Dragoš Cvetković. Some comments on the eigenspaces of graphs. *Publ. Inst. Math.(Beograd)*, 50(64):24–32, 1991.

Dragoš Cvetković, Peter Rowlinson, and Slobodan Simic. *Eigenspaces of graphs*. Number 66 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1997.

Rumen Dangovski, Li Jing, Charlotte Loh, Seungwook Han, Akash Srivastava, Brian Cheung, Pulkit Agrawal, and Marin Soljačić. Equivariant contrastive learning. In *Int. Conference on Learning Representations (ICLR)*, 2022.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 3844–3852, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. Ieee, 2009.

Karan Desai, Maximilian Nickel, Tanmay Rajpurohit, Justin Johnson, and Ramakrishna Vedantam. Hyperbolic image-text representations. *ICLR Workshop on Multimodal Representation Learning*, 2023.

Alexandre Devillers and Mathieu Lefort. Equimod: An equivariance module to improve self-supervised learning. In *Int. Conference on Learning Representations (ICLR)*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics, 2004.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conference on Learning Representations (ICLR)*, 2021a.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conference on Learning Representations (ICLR)*, 2021b.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conference on Learning Representations (ICLR)*, 2021c.

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *Int. Conference on Machine Learning (ICML)*, pages 1386–1394, 2015.

Marthinus C Du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 703–711, 2014.

Haonan Duan, Pashootan Vaezipoor, Max B Paulus, Yangjun Ruan, and Chris J Maddison. Augment with care: Contrastive learning for the boolean satisfiability problem. In *Int. Conference on Machine Learning (ICML)*, 2022.

Mohammed Haroon Dupty and Wee Sun Lee. Graph representation learning with individualization and refinement. *arXiv preprint arXiv:2203.09141*, 2022.

Mohammed Haroon Dupty, Yanfei Dong, and Wee Sun Lee. Pf-gnn: Differentiable particle filtering based approximation of universal graph representations. In *Int. Conference on Learning Representations (ICLR)*, 2021.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. In *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. In *preprint arXiv:2003.00982*, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *Int. Conference on Learning Representations (ICLR)*, volume 10, 2022.

HT Eastment and WJ Krzanowski. Cross-validatory choice of the number of components from a principal component analysis. *Technometrics*, 24(1):73–77, 1982.

Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Optimization—Eureka, You Shrink!*, pages 11–26. Springer, 2003.

Marwa El Halabi. Learning with structured sparsity: From discrete to convex and back. Technical report, EPFL, 2018.

Marwa El Halabi, Francis Bach, and Volkan Cevher. Combinatorial penalties: Which structures are preserved by convex relaxations? In *International Conference on Artificial Intelligence and Statistics*, pages 1551–1560. PMLR, 2018.

Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220, 2008.

Paul Erdos and Alfréd Rényi. Asymmetric graphs. *Acta Math. Acad. Sci. Hungar*, 14 (295-315):3, 1963.

James E Falk and Karla R Hoffman. A successive underestimation method for concave minimization problems. *Mathematics of operations research*, 1(3):251–259, 1976.

Yuxin Fang, Li Dong, Hangbo Bao, Xinggang Wang, and Furu Wei. Corrupted image modeling for self-supervised visual pre-training. *preprint arXiv:2202.03382*, 2022.

Or Feldman, Amit Boyarski, Shai Feldman, Dani Kogan, Avi Mendelson, and Chaim Baskin. Weisfeiler and leman go infinite: Spectral and combinatorial pre-colorings. In *preprint arXiv:2201.13410*, 2022.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. In *ICLR (Workshop on Representation Learning on Graphs and Manifolds)*, volume 7, 2019.

Matthias Fey, Jan-Gin Yuen, and Frank Weichert. Hierarchical inter-message passing for learning on molecular graphs. *arXiv preprint arXiv:2006.12179*, 2020.

Martin Fürer. On the power of combinatorial and spectral invariants. *Linear algebra and its applications*, 432(9):2373–2380, 2010.

Jean Gallier and Jocelyn Quaintance. *Differential geometry and Lie groups: a computational perspective*, volume 12. Springer Nature, 2020.

V. K. Garg, S. Jegelka, and T. Jaakkola. Generalization and representational limits of graph neural networks. In *Int. Conference on Machine Learning (ICML)*, 2020.

Quentin Garrido, Yubei Chen, Adrien Bardes, Laurent Najman, and Yann Lecun. On the duality between contrastive and non-contrastive self-supervised learning. *arXiv preprint arXiv:2206.02574*, 2022.

Quentin Garrido, Laurent Najman, and Yann Lecun. Self-supervised learning of split invariant equivariant representations. *preprint arXiv:2302.10283*, 2023.

Songwei Ge, Shlok Kumar Mishra, Haohan Wang, Chun-Liang Li, and David Jacobs. Robust contrastive learning using negative samples with diminished semantics. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume abs/2110.14189, 2021.

Songwei Ge, Shlok Mishra, Simon Kornblith, Chun-Liang Li, and David Jacobs. Hyperbolic contrastive learning for visual representations beyond objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Europ. Conference on Computer Vision (ECCV)*, pages 269–285, 2018.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *Int. Conference on Learning Representations (ICLR)*, 2019.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

Juan A Navarro González and Juan B Sancho de Salas. *C^∞-differentiable spaces*, volume 1824. Springer, 2003.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Int. Conference on Learning Representations (ICLR)*, 2015.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *preprint arXiv:1706.0267*, 2017.

Jean-Bastien Grill, Florian Strub, Florent Altch'e, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo A Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 21271–21284, 2020.

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid algorithm and its consequences in combinatorial optimization. *Combinatorica*, 1:499–513, 1981.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL https://www.gurobi.com.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. Int. Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 297–304, 2010.

Will Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

Ben Harwood, Vijay Kumar BG, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *Int. Conference on Computer Vision (ICCV)*, pages 2821–2829, 2017.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *Int. Conference on Machine Learning (ICML)*, pages 3451–3461, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020a.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738, 2020b.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.

Mingguo He, Zhewei Wei, Hongteng Xu, et al. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *Int. Conference on Machine Learning (ICML)*, pages 4182–4192. PMLR, 2020.

Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. In *Int. Conference on Machine Learning (ICML)*, pages 4182–4192. PMLR, 2020.

Olivier J Hénaff, Skanda Koppula, Jean-Baptiste Alayrac, Aäron Van den Oord, Oriol Vinyals, and João Carreira. Efficient visual pretraining with contrastive detection. In *Int. Conference on Computer Vision (ICCV)*, 2021.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Int. Conference on Computer Vision (ICCV)*, pages 8340–8349, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15262–15271, 2021b.

Katherine L Hermann and Andrew K Lampinen. What shapes feature representations? Exploring datasets, architectures, and training. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9995–10006, 2020.

Katherine L Hermann, Ting Chen, and Simon Kornblith. The origins and prevalence of texture bias in convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 19000–19015, 2019.

Charles Herrmann, Kyle Sargent, Lu Jiang, Ramin Zabih, Huiwen Chang, Ce Liu, Dilip Krishnan, and Deqing Sun. Pyramid adversarial training improves ViT performance. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13419–13429, 2022.

Chih-Hui Ho and Nuno Nvasconcelos. Contrastive learning with adversarial examples. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 17081–17093, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6840–6851, 2020.

Paul W Holland and Samuel Leinhardt. A method for detecting structure in sociometric data. In *Social networks*, pages 411–432. Elsevier, 1977.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 1133–1141. IEEE, 2017.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004.

Qianjiang Hu, Xiao Wang, Wei Hu, and Guo-Jun Qi. Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020a.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 22118–22133, 2020b.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *Int. Conference on Learning Representations (ICLR)*, volume 8, 2020c.

Leo Huang, Andrew J Graven, and David Bindel. Density of states graph kernels. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 289–297. SIAM, 2021.

Zhicheng Huang, Xiaojie Jin, Chengze Lu, Qibin Hou, Ming-Ming Cheng, Dongmei Fu, Xiaohui Shen, and Jiashi Feng. Contrastive masked autoencoders are stronger vision learners. *arXiv:2207.13532v1*, 2022.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *preprint arXiv:2103.10427*, 2021.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 125–136, 2019.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

Rishabh Iyer, Stefanie Jegelka, and Jeff Bilmes. Monotone closure of relaxed constraints in submodular optimization: Connections between minimization and maximization: Extended version. In *UAI*, 2014.

Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. In *Int. Conference on Learning Representations (ICLR)*, 2018.

Shantanu Jain, Martha White, and Predrag Radivojac. Estimating the class prior and posterior from noisy positives and unlabeled data. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2693–2701, 2016.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021.

Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 16199–16210, 2020.

SouYoung Jin, Aruni RoyChowdhury, Huaizu Jiang, Ashish Singh, Aditya Prasad, Deep Chakraborty, and Erik Learned-Miller. Unsupervised hard example mining from videos for improved object detection. In *Europ. Conference on Computer Vision (ECCV)*, pages 307–324, 2018.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *Int. Conference on Learning Representations (ICLR)*, 2022.

Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In *NeurIPS*, 2020.

Nikolaos Karalias, Joshua Robinson, Andreas Loukas, and Stefanie Jegelka. Neural set function extensions: Learning with discrete functions in high dimensions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 15338–15352, 2022.

Marc C Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3): 425–464, 2001.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv:2004.11362*, 2020.

Jinwoo Kim, Saeyoon Oh, and Seunghoon Hong. Transformers generalize deepsets and can be extended to graphs & hypergraphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *arXiv preprint arXiv:2207.02505*, 2022.

Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 21696–21707, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Int. Conference on Learning Representations (ICLR)*, volume 5, 2017.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3294–3302, 2015.

Ryuichi Kiryo, Gang Niu, Marthinus C du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1675–1685, 2017.

Gerald R Kneller. Superposition of molecular structures using quaternions. *Molecular Simulation*, 7(1-2):113–119, 1991.

Lukas Koestler, Daniel Grittner, Michael Moeller, Daniel Cremers, and Zorah Lähner. Intrinsic neural fields: Learning functions on manifolds. *arXiv preprint arXiv:2203.07967*, 2022.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *ECCV*, 2020.

Skanda Koppula, Yazhe Li, Evan Shelhamer, Andrew Jaegle, Nikhil Parthasarathy, Relja Arandjelovic, João Carreira, and Olivier Hénaff. Where should i spend my flops? efficiency evaluations of visual pre-training methods. *arXiv preprint arXiv:2209.15589*, 2022.

Hanspeter Kraft and Claudio Procesi. Classical invariant theory, a primer. *Lecture Notes.*, 1996.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Vipin Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI magazine*, 13(1):32–32, 1992.

Hankook Lee, Kibok Lee, Kimin Lee, Honglak Lee, and Jinwoo Shin. Improving transferability of representations via augmentation-aware self-supervision. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 17710–17722, 2021.

John M Lee. Smooth manifolds. In *Introduction to Smooth Manifolds*. Springer, 2013.

F. Thomson Leighton and Gary l. Miller. Certificates for graphs with distinct eigen values. Orginal Manuscript, 1979.

Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

Bruno Lévy. Laplace-beltrami eigenfunctions towards an algorithm that" understands" geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 13–13. IEEE, 2006.

Pan Li, Eli Chien, and Olgica Milenkovic. Optimizing generalized pagerank methods for seed-expansion community detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019a.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 4465–4478, 2020.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *Int. Conference on Machine Learning (ICML)*, pages 3835–3845, 2019b.

Derek Lim, Joshua Robinson, Stefanie Jegelka, Yaron Lipman, and Haggai Maron. Expressive sign equivariant networks for spectral geometric learning. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023a.

Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *Int. Conference on Learning Representations (ICLR)*, 2023b.

Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. In *Int. Conference on Learning Representations (ICLR)*, 2018.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in Adam. *preprint arXiv:1711.05101*, 2017a.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *Int. Conference on Learning Representations (ICLR)*, 2017b.

László Lovász. Submodular functions and convexity. In *Mathematical programming the state of the art*, pages 235–257. Springer, 1983.

László Lovász and Alexander Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM journal on optimization*, 1(2):166–190, 1991.

Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *Int. Conference on Learning Representations (ICLR)*, 2020.

Yi Ma, Doris Tsao, and Heung-Yeung Shum. On the principles of parsimony and self-consistency for the emergence of intelligence. *Frontiers of Information Technology & Electronic Engineering*, 23(9):1298–1323, 2022.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Int. Conference on Learning Representations (ICLR)*, 2018.

Takanori Maehara and Hoang NT. A simple proof of the universality of invariant/equivariant graph neural networks. *arXiv preprint arXiv:1910.03802*, 2019.

Chengzhi Mao, Lu Jiang, Mostafa Dehghani, Carl Vondrick, Rahul Sukthankar, and Irfan Essa. Discrete representations strengthen vision transformer robustness. In *Int. Conference on Learning Representations (ICLR)*, 2022.

K. V. Mardia and P. Jupp. *Directional Statistics*. John Wiley and Sons Ltd., second edition, 2000.

J-L Marichal. An axiomatic approach of the discrete choquet integral as a tool to aggregate interacting criteria. *IEEE transactions on fuzzy systems*, 8(6):800–807, 2000.

Zelda Mariet, Joshua Robinson, Jamie Smith, Suvrit Sra, and Stefanie Jegelka. Optimal batch variance with second-order marginals. In *ICML Workshop*, volume 2, 2020.

F Landis Markley and John L Crassidis. *Fundamentals of spacecraft attitude determination and control*, volume 1286. Springer, 2014.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *Int. Conference on Learning Representations (ICLR)*, volume 6, 2018.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *Int. Conference on Machine Learning (ICML)*, pages 4363–4371. PMLR, 2019.

Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

D.S. Meek. The inverses of toeplitz band matrices. *Linear Algebra and its Applications*, 49:117–129, 1983. ISSN 0024-3795. doi: https://doi.org/10.1016/0024-3795(83)90097-6. URL https://www.sciencedirect.com/science/article/pii/0024379583900976.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. GraphiT: Encoding graph structure in transformers. In *preprint arXiv:2106.05667*, 2021.

Matthias Minderer, Olivier Bachem, Neil Houlsby, and Michael Tschannen. Automatic shortcut removal for self-supervised representation learning. In *International Conference on Machine Learning*, pages 6927–6937, 2020.

Shlok Mishra*, Joshua Robinson*, Huiwen Chang, David Jacobs, Aaron Sarna, Aaron Maschinot, and Dilip Krishnan. A simple, efficient and scalable contrastive masked autoencoder for learning visual representations. In *preprint arXiv:2210.16870*, 2023.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning

with graphs. In *ICML workshop on Graph Representation Learning and Beyond*, 2020a.

Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. *Advances in Neural Information Processing Systems*, 33:21824–21840, 2020b.

Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: The story so far. *arXiv preprint arXiv:2112.09992*, 2021.

Christopher Morris, Gaurav Rattan, Sandra Kiefer, and Siamak Ravanbakhsh. Speqnets: Sparsity-aware permutation-equivariant graph networks. *arXiv preprint arXiv:2203.13913*, 2022.

Kazuo Murota. Discrete convex analysis. *Mathematical Programming*, 83(1):313–371, 1998.

Michael Murphy, Stefanie Jegelka, and Ernest Fraenkel. Self-supervised learning of cell type specificity from immunohistochemical images. *Bioinformatics*, 38 (Supplement_1):i395–i403, 2022.

Oleg R Musin and Alexey S Tarasov. The tammes problem for n= 14. *Experimental Mathematics*, 24(4):460–468, 2015.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *Int. Conference on Learning Representations (ICLR)*, 2021.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 271–279, 2016.

Guillaume Obozinski and Francis Bach. *Convex Relaxation for Combinatorial Penalties*. PhD thesis, INRIA, 2012.

Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

Maks Ovsjanikov, Jian Sun, and Leonidas Guibas. Global intrinsic symmetries of shapes. In *Computer graphics forum*, volume 27, pages 1341–1348. Wiley Online Library, 2008.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035. Curran Associates, Inc., 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

Fritz Peter and Hermann Weyl. Die Vollständigkeit der primitiven Darstellungen einer geschlossenen kontinuierlichen Gruppe. *Mathematische Annalen*, 97(1):737–755, 1927.

Omri Puny, Matan Atzmon, Heli Ben-Hamu, Edward J Smith, Ishan Misra, Aditya Grover, and Yaron Lipman. Frame averaging for invariant and equivariant network design. In *Int. Conference on Learning Representations (ICLR)*, volume 10, 2022.

Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv:2007.13916*, 2020.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Int. Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021.

Siamak Ravanbakhsh. Universal equivariant multilayer perceptrons. In *International Conference on Machine Learning*, pages 7996–8006. PMLR, 2020.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *Int. Conference on Machine Learning (ICML)*, pages 5389–5400, 2019.

Elizabeth A Regan, John E Hokanson, James R Murphy, Barry Make, David A Lynch, Terri H Beaty, Douglas Curran-Everett, Edwin K Silverman, and James D Crapo.

Genetic epidemiology of COPD (COPDGene) study design. *COPD: Journal of Chronic Obstructive Pulmonary Disease*, 7(1):32–43, 2011.

Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 287–297. Springer, 2008.

Joshua Robinson, Suvrit Sra, and Stefanie Jegelka. Flexible modeling of diversity with strongly log-concave distributions. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Joshua Robinson, Stefanie Jegelka, and Suvrit Sra. Strength from weakness: Fast learning using weak supervision. In *Int. Conference on Machine Learning (ICML)*, pages 8127–8136, 2020.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *Int. Conference on Learning Representations (ICLR)*, 2021a.

Joshua Robinson, Li Sun, Ke Yu, Kayhan Batmanghelich, Stefanie Jegelka, and Suvrit Sra. Can contrastive learning avoid shortcut solutions? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 4974–4986, 2021b.

Raif M Rustamov et al. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Symposium on geometry processing*, volume 257, pages 225–233, 2007.

Horst Sachs and M Stiebitz. Automorphism group and spectrum of a graph. In *Studies in pure mathematics*, pages 587–604. Springer, 1983.

Ryoma Sato. A survey on the expressive power of graph neural networks. In *preprint arXiv:2003.04078*, 2020.

Barbara J Schmid. Finite groups and invariant theory. In *Topics in Invariant Theory: Séminaire d'Algèbre P. Dubreil et M.-P. Malliavin 1989–1990 (40ème Année)*, pages 35–66. Springer, 2006.

Alexei Schneider, Alexei Baevski, Shu-wen Chen, Sanjeev Khudanpur, and Andrew Davis. Wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Proceedings of the 2021 Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1657–1661, 2021.

Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.

K Schütte and BL Van der Waerden. Auf welcher kugel haben 5, 6, 7, 8 oder 9 punkte mit mindestabstand eins platz? *Mathematische Annalen*, 123(1):96–124, 1951.

Nimrod Segol and Yaron Lipman. On universal equivariant set networks. In *Int. Conference on Learning Representations (ICLR)*, volume 7, 2019.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3): 93–93, 2008.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141, 2018.

Jean-Pierre Serre et al. *Linear representations of finite groups*, volume 42. Springer, 1977.

Mehran Shakerinava, Arnab Kumar Mondal, and Siamak Ravanbakhsh. Structuring representations using group invariants. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *NeurIPS Workshop on Relational Representation Learning*, 2018.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4077–4087, 2017.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4004–4012, 2016.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Int. Conference on Learning Representations (ICLR)*, 2021.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Int. Conference on Machine Learning (ICML)*, pages 10360–10371, 2020.

Balasubramaniam Srinivasan and Bruno Ribeiro. On the equivalence between positional node embeddings and structural graph representations. In *Int. Conference on Learning Representations (ICLR)*, 2019.

Andreas Steiner, Alexander Kolesnikov, , Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? Data, augmentation, and regularization in vision transformers. In *Transactions on Machine Learning Research (TMLR)*, 2021.

Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7251–7259, 2019.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Int. Conference on Computer Vision (ICCV)*, pages 843–852, 2017.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *Int. Conference on Learning Representations (ICLR)*, 2020.

Li Sun, Ke Yu, and Kayhan Batmanghelich. Context matters: Graph-based self-supervised representation learning for medical images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4874–4882, 2021.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *Int. Conference on Learning Representations (ICLR)*, 2019.

Tristan Sylvain, Linda Petrini, and Devon Hjelm. Locality and compositionality in zero-shot learning. In *Int. Conference on Learning Representations (ICLR)*, 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Int. Conference on Learning Representations (ICLR)*, 2014.

Behrooz Tahmasebi, Derek Lim, and Stefanie Jegelka. Counting substructures with higher-order graph neural networks: Possibility and impossibility results. In *preprint arXiv:2012.03174*, 2020.

Edric Tam and David Dunson. Multiscale graph comparison via the embedded laplacian distance. In *preprint arXiv:2201.12064*, 2022.

Pieter Merkus Lambertus Tammes. On the origin of number and arrangement of the places of exit on the surface of pollen-grains. *Recueil des travaux botaniques néerlandais*, 27(1):1–84, 1930.

Chenxin Tao, Honghui Wang, Xizhou Zhu, Jiahua Dong, Shiji Song, Gao Huang, and Jifeng Dai. Exploring the equivalence of siamese self-supervised learning via a unified gradient framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14431–14440, 2022a.

Chenxin Tao, Xizhou Zhu, Gao Huang, Yu Qiao, Xiaogang Wang, and Jifeng Dai. Siamese image modeling for self-supervised vision representation learning. In *preprint arXiv:2206.01204*, 2022b.

Terence Tao and Van Vu. Random matrices have simple spectrum. *Combinatorica*, 37 (3):539–553, 2017.

Mohit Tawarmalani and Nikolaos V Sahinidis. Convex extensions and envelopes of lower semi-continuous functions. *Mathematical Programming*, 93(2):247–263, 2002.

Yasuo Teranishi. Eigenvalues and automorphisms of a graph. *Linear and Multilinear Algebra*, 57(6):577–585, 2009.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive Multiview Coding. In *Europ. Conference on Computer Vision (ECCV)*, pages 770–786, 2019.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Europ. Conference on Computer Vision (ECCV)*, pages 776–794, 2020a.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *arXiv:2005.10243*, 2020b.

Yonglong Tian, Olivier J Henaff, and Aäron van den Oord. Divide and contrast: Self-supervised learning from uncurated data. In *Int. Conference on Computer Vision (ICCV)*, pages 10063–10074, 2021.

Jan Toenshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph neural networks for maximum constraint satisfaction. *Frontiers in artificial intelligence*, 3: 98, 2021.

Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. SIAM, 1997.

William F Trench. Inversion of toeplitz band matrices. *Mathematics of computation*, 28(128):1089–1095, 1974.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *Int. Conference on Learning Representations (ICLR)*, 2018.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller. Netlsd: hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2347–2356, 2018.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *preprint arXiv:1807.03748*, 2018.

Derek van Tilborg, Alisa Alenicheva, and Francesca Grisoni. Exposing the limitations of molecular machine learning with activity cliffs. *Journal of Chemical Information and Modeling*, 62(23):5938–5951, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017a.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 5998–6008, 2017b.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Int. Conference on Learning Representations (ICLR)*, volume 6, 2018.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep Graph Infomax. In *Int. Conference on Learning Representations (ICLR)*, 2019.

Petar Veličković and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2(7): 100273, 2021. ISSN 2666-3899.

Saurabh Verma and Zhi-Li Zhang. Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 88–98, 2017.

Soledad Villar, David Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17 (4):395–416, 2007.

J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *ACM Symposium on Theory of Computing (STOC)*, 2008.

Ellen M Voorhees and Donna Harman. Overview of trec 2002. In *Trec*, 2002.

Feng Wang and Huaping Liu. Understanding the behaviour of contrastive loss. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *Int. Conference on Learning Representations (ICLR)*, volume 10, 2022.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Int. Conference on Machine Learning (ICML)*, pages 9574–9584, 2020a.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Int. Conference on Machine Learning (ICML)*, pages 9929–9939. PMLR, 2020b.

Hermann Weyl. *The classical groups: their invariants and representations*. Princeton university press, 1946.

Hassler Whitney. The self-intersections of a smooth n-manifold in $2n$-space. In *Annals of Mathematics*, pages 220–246, 1944.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210, 2005.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Int. Conference on Computer Vision (ICCV)*, pages 2840–2848, 2017.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Yuyang Xie, Jianhong Wen, Kin Wai Lau, Yasar Abbas Ur Rehman, and Jiajun Shen. What should be equivariant in self-supervised learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4111–4120, 2022a.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9653–9663, 2022b.

Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv:1304.5634*, 2013.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Int. Conference on Learning Representations (ICLR)*, volume 7, 2019.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *Int. Conference on Learning Representations (ICLR)*, volume 8, 2020.

Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *Int. Conference on Learning Representations (ICLR)*, 2021.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. Deep bidirectional language-knowledge graph pretraining. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 37309–37323, 2022.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*, volume 35, pages 10737–10745, 2021.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. In *ECCV*, 2017.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. CoCa: Contrastive captioners are image-text foundation models. *preprint arXiv:2205.01917*, 2022.

Yi Yu, Tengyao Wang, and Richard J Samworth. A useful variant of the davis–kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015.

Yun Yue, Fangzhou Lin, Kazunori D Yamada, and Ziming Zhang. Hyperbolic contrastive learning. *preprint arXiv:2302.01409*, 2023.

Li Yujia, Tarlow Daniel, Brockschmidt Marc, Zemel Richard, et al. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 3391–3401, 2017.

Sheheryar Zaidi, Michael Schaarschmidt, James Martens, Hyunjik Kim, Yee Whye Teh, Alvaro Sanchez-Gonzalez, Peter Battaglia, Razvan Pascanu, and Jonathan Godwin. Pre-training via denoising for molecular property prediction. *preprint arXiv:2206.00133*, 2022.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Int. Conference on Machine Learning (ICML)*, pages 12310–12320. PMLR, 2021.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *Int. Conference on Machine Learning (ICML)*, pages 7472–7482, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Int. Conference on Learning Representations (ICLR)*, 2018a.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-BERT: Only attention is needed for learning graph representations. In *preprint arXiv:2001.05140*, 2020.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings*

of the IEEE conference on computer vision and pattern recognition, pages 586–595, 2018b.

Zhen Zhang, Mianzhi Wang, Yijian Xiang, Yan Huang, and Arye Nehorai. RetGK: Graph kernels based on return probabilities of random walks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 3964–3974, 2018c.

Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *Int. Conference on Learning Representations (ICLR)*, volume 10, 2022.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

Roland S Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive learning inverts the data generating process. In *Int. Conference on Machine Learning (ICML)*, 2021.

Appendix A

Further Discussion and Proofs for Negative Sampling in Contrastive Learning

A.1 Analysis of Hard Sampling

A.1.1 Hard Sampling Interpolates Between Marginal and Worst-Case Negatives

We begin by proving Proposition 1. Recall that the proposition stated the following.

Proposition 13. *Let* $\mathcal{L}^*(f) = \sup_{q \in \Pi} \mathcal{L}(f, q)$. *Then for any* $t > 0$ *and measurable* $f : \mathcal{X} \to \mathbb{S}^{d-1}/t$ *we observe the convergence* $\mathcal{L}(f, q_\beta^-) \longrightarrow \mathcal{L}^*(f)$ *as* $\beta \to \infty$.

Proof. Consider the following essential supremum,

$$M(x) = \operatorname*{ess\,sup}_{x^- \in \mathcal{X} : x^- \not\sim x} f(x)^T f(x^-) = \sup\{m > 0 : m \geq f(x)^T f(x^-) \text{ a.s. for } x^- \sim p^-\}.$$

The second inequality holds since $\operatorname{supp}(p) = \mathcal{X}$. We may rewrite

$$\mathcal{L}^*(f) = \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}}\left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + Qe^{M(x)}}\right],$$
$$\mathcal{L}(f, q_\beta^-) = \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}}\left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + Q\mathbb{E}_{x^-\sim q_\beta^-}[e^{f(x)^T f(x^-)}]}\right].$$

The difference between these two terms can be bounded as follows,

$$\begin{aligned}
\left|\mathcal{L}^*(f) - \mathcal{L}(f, q_\beta^-)\right| &\leq \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}}\left|-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + Qe^{M(x)}} + \log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + Q\mathbb{E}_{x^-\sim q_\beta^-}[e^{f(x)^T f(x^-)}]}\right| \\
&= \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}}\left|\log\left(e^{f(x)^T f(x^+)} + Q\mathbb{E}_{x^-\sim q_\beta^-}[e^{f(x)^T f(x^-)}]\right) - \log\left(e^{f(x)^T f(x^+)} + Qe^{M(x)}\right)\right| \\
&\leq \frac{e^{1/t}}{Q+1}\cdot \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}}\left|e^{f(x)^T f(x^+)} + Q\mathbb{E}_{x^-\sim q_\beta^-}[e^{f(x)^T f(x^-)}] - e^{f(x)^T f(x^+)} - Qe^{M(x)}\right| \\
&= \frac{e^{1/t}Q}{Q+1}\cdot \mathbb{E}_{x\sim p}\left|\mathbb{E}_{x^-\sim q_\beta^-}[e^{f(x)^T f(x^-)}] - e^{M(x)}\right| \\
&\leq e^{1/t}\cdot \mathbb{E}_{x\sim p}\mathbb{E}_{x^-\sim q_\beta^-}\left|e^{M(x)} - e^{f(x)^T f(x^-)}\right|
\end{aligned}$$

where for the second inequality we have used the fact that f lies on the hypersphere of radius $1/t$ to restrict the domain of the logarithm to values greater than $(Q+1)e^{-1/t}$. Because of this the logarithm is Lipschitz with parameter $e^{1/t}/(Q+1)$. Using again the fact that f lies on the hypersphere we know that $\left|f(x)^T f(x^-)\right| \leq 1/t^2$ and hence have the following inequality,

$$\mathbb{E}_{x\sim p}\mathbb{E}_{q_\beta^-}\left|e^{M(x)} - e^{f(x)^T f(x^-)}\right| \leq e^{1/t^2}\mathbb{E}_{x\sim p}\mathbb{E}_{q_\beta^-}\left|M(x) - f(x)^T f(x^-)\right|$$

Let us consider the inner expectation $E_\beta(x) = \mathbb{E}_{q_\beta^-}\left|M(x) - f(x)^T f(x^-)\right|$. Note that since f is bounded, $E_\beta(x)$ is uniformly bounded in x. Therefore, in order to show the convergence $\mathcal{L}(f, q_\beta^-) \to \mathcal{L}^*(f)$ as $\beta \to \infty$, it suffices by the dominated convergence theorem to show that $E_\beta(x) \to 0$ pointwise as $\beta \to \infty$ for arbitrary fixed $x \in \mathcal{X}$.

From now on we denote $M = M(x)$ for brevity, and consider a fixed $x \in \mathcal{X}$. From

the definition of q_β^- it is clear that $q_\beta^- \ll p^-$. That is, since $q_\beta^- = c \cdot p^-$ for some (non-constant) c, it is absolutely continuous with respect to p^-. So $M(x) \geq f(x)^T f(x^-)$ almost surely for $x^- \sim q_\beta^-$, and we may therefore drop the absolute value signs from our expectation. Define the following event $\mathcal{G}_\varepsilon = \{x^- : f(x)^\top f(x^-) \geq M - \varepsilon\}$ where $\mathcal{G}$ is refers to a "good" event. Define its complement $\mathcal{B}_\varepsilon = \mathcal{G}_\varepsilon^c$ where $\mathcal{B}$ is for "bad". For a fixed $x \in \mathcal{X}$ and $\varepsilon > 0$ consider,

$$\begin{aligned}
E_\beta(x) &= \mathbb{E}_{x^- \sim q_\beta^-} \left| M(x) - f(x)^T f(x^-) \right| \\
&= \mathbb{P}_{x^- \sim q_\beta^-}(\mathcal{G}_\varepsilon) \cdot \mathbb{E}_{x^- \sim q_\beta^-} \left[\left| M(x) - f(x)^T f(x^-) \right| | \mathcal{G}_\varepsilon \right] \\
&\quad + \mathbb{P}_{x^- \sim q_\beta^-}(\mathcal{B}_\varepsilon) \cdot \mathbb{E}_{x^- \sim q_\beta^-} \left[\left| M(x) - f(x)^T f(x^-) \right| | \mathcal{B}_\varepsilon \right] \\
&\leq \mathbb{P}_{x^- \sim q_\beta^-}(\mathcal{G}_\varepsilon) \cdot \varepsilon + 2\mathbb{P}_{x^- \sim q_\beta^-}(\mathcal{B}_\varepsilon) \\
&\leq \varepsilon + 2\mathbb{P}_{x^- \sim q_\beta^-}(\mathcal{B}_\varepsilon).
\end{aligned}$$

We need to control $\mathbb{P}_{x^- \sim q_\beta^-}(\mathcal{B}_\varepsilon)$. Expanding,

$$\mathbb{P}_{x^- \sim q_\beta^-}(\mathcal{B}_\varepsilon) = \int_{\mathcal{X}} \mathbf{1}\left\{ f(x)^T f(x^-) < M(x) - \varepsilon \right\} \frac{e^{\beta f(x)^T f(x^-)} \cdot p^-(x^-)}{Z_\beta} \mathrm{d}x^-$$

where $Z_\beta = \int_{\mathcal{X}} e^{\beta f(x)^T f(x^-)} p^-(x^-) \mathrm{d}x^-$ is the partition function of q_β^-. We may bound this expression by,

$$\begin{aligned}
\int_{\mathcal{X}} \mathbf{1}\left\{ f(x)^T f(x^-) < M - \varepsilon \right\} \frac{e^{\beta(M-\varepsilon)} \cdot p^-(x^-)}{Z_\beta} \mathrm{d}x^- &\leq \frac{e^{\beta(M-\varepsilon)}}{Z_\beta} \int_{\mathcal{X}} \mathbf{1}\left\{ f(x)^T f(x^-) < M - \varepsilon \right\} p^-(x^-) \mathrm{d}x^- \\
&= \frac{e^{\beta(M-\varepsilon)}}{Z_\beta} \mathbb{P}_{x^- \sim p^-}(\mathcal{B}_\varepsilon) \\
&\leq \frac{e^{\beta(M-\varepsilon)}}{Z_\beta}
\end{aligned}$$

Note that

$$Z_\beta = \int_{\mathcal{X}} e^{\beta f(x)^T f(x^-)} p^-(x^-) \mathrm{d}x^- \geq e^{\beta(M-\varepsilon/2)} \mathbb{P}_{x^- \sim p^-}(f(x)^T f(x^-) \geq M - \varepsilon/2).$$

By the definition of $M = M(x)$ the probability $\rho_\varepsilon = \mathbb{P}_{x^- \sim p^-}(f(x)^T f(x^-) \geq M - \varepsilon/2) > 0$, and we may therefore bound,

$$\begin{aligned}\mathbb{P}_{x^- \sim q_\beta^-}(\mathcal{B}_\varepsilon) &= \frac{e^{\beta(M-\varepsilon)}}{e^{\beta(M-\varepsilon/2)}\rho_\varepsilon} \\ &= e^{-\beta\varepsilon/2}/\rho_\varepsilon \\ &\longrightarrow 0 \;\text{ as }\; \beta \to \infty.\end{aligned}$$

We may therefore take β to be sufficiently big so as to make $\mathbb{P}_{x^- \sim q_\beta^-}(\mathcal{B}_\varepsilon) \leq \varepsilon$ and therefore $E_\beta(x) \leq 3\varepsilon$. In other words, $E_\beta(x) \longrightarrow 0$ as $\beta \to \infty$. □

A.1.2 Optimal Embeddings on the Hypersphere for Worst-Case Negative Samples

In order to study properties of global optima of the contrastive objective using the adversarial worst case hard sampling distribution recall that we have the following limiting objective,

$$\mathcal{L}_\infty(f, q) = \mathbb{E}_{\substack{x \sim p \\ x^+ \sim p_x^+}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{\mathbb{E}_{x^- \sim q_\beta}\left[e^{f(x)^T f(x^-)}\right]} \right]. \tag{A.1}$$

We may separate the logarithm of a quotient into the sum of two terms plus a constant,

$$\mathcal{L}_\infty(f, q) = \mathcal{L}_{\text{align}}(f) + \mathcal{L}_{\text{unif}}(f, q) - 1/t^2$$

where $\mathcal{L}_{\text{align}}(f) = \mathbb{E}_{x,x^+} \|f(x) - f(x^+)\|^2/2$ and $\mathcal{L}_{\text{unif}}(f, q) = \mathbb{E}_{x \sim p} \log \mathbb{E}_{x^- \sim q} e^{f(x)^\top f(x^-)}$. Here we have used the fact that f lies on the boundary of the hypersphere of radius $1/t$, which gives us the following equivalence between inner products and squared Euclidean norm,

$$2/t^2 - 2f(x)^\top f(x^+) = \|f(x)\|^2 + \|f(x^+)\|^2 - 2f(x)^\top f(x^+) = \|f(x) - f(x^+)\|^2. \tag{A.2}$$

Taking supremum to obtain $\mathcal{L}^*_\infty(f) = \sup_{q\in\Pi} \mathcal{L}_\infty(f,q)$ we find that the second expression simplifies to,

$$\mathcal{L}^*_{\text{unif}}(f) = \sup_{q\in\Pi} \mathcal{L}_{\text{unif}}(f,q) = \mathbb{E}_{x\sim p} \log \sup_{x^- \not\sim x} e^{f(x)^\top f(x^-)} = \mathbb{E}_{x\sim p} \sup_{x^-\not\sim x} f(x)^\top f(x^-).$$

Using Eqn. (A.2), this can be re-expressed as,

$$\mathbb{E}_{x\sim p} \sup_{x^-\not\sim x} f(x)^\top f(x^-) = -\mathbb{E}_{x\sim p} \inf_{x^-\not\sim x} \|f(x) - f(x^-)\|^2/2 + 1/t^2. \tag{A.3}$$

The forthcoming theorem exactly characterizes the global optima of $\min_f \mathcal{L}^*_\infty(f)$

Theorem 9. *Suppose the downstream task is classification (i.e. $\mathcal{C}$ is finite), and let $\mathcal{L}^*_\infty(f) = \sup_{q\in\Pi} \mathcal{L}_\infty(f,q)$. The infimum $\inf_{f:\ measurable} \mathcal{L}^*_\infty(f)$ is attained, and any f^* achieving the global minimum is such that $f^*(x) = f^*(x^+)$ almost surely. Furthermore, letting $\mathbf{v}_c = f^*(x)$ for any x such that $h(x) = c$ (so $\mathbf{v}_c$ is well defined up to a set of x of measure zero), f^* is characterized as being any solution to the following ball-packing problem,*

$$\max_{\{\mathbf{v}_c\in\mathbb{S}^{d-1}/t\}_{c\in\mathcal{C}}} \sum_{c\in\mathcal{C}} \rho(c) \cdot \min_{c'\neq c} \|\mathbf{v}_c - \mathbf{v}_{c'}\|^2. \tag{A.4}$$

Proof. Any minimizer of $\mathcal{L}_{\text{align}}(f)$ has the property that $f(x) = f(x^+)$ almost surely. So, in order to prove the first claim, it suffices to show that there exist functions $f \in \arg\inf_f \mathcal{L}^*_{\text{unif}}(f)$ for which $f(x) = f(x^+)$ almost surely. This is because, at that point, we have shown that $\arg\min_f \mathcal{L}_{\text{align}}(f)$ and $\arg\min_f \mathcal{L}^*_{\text{unif}}(f)$ intersect, and therefore any solution of $\mathcal{L}^*_\infty(f) = \mathcal{L}_{\text{align}}(f) + \mathcal{L}^*_{\text{unif}}(f)$ must lie in this intersection.

To this end, suppose that $f \in \arg\min_f \mathcal{L}^*_{\text{unif}}(f)$ but that $f(x) \neq f(x^+)$ with non-zero probability. We shall show that we can construct a new embedding $\hat{f}$ such that $f(x) = f(x^+)$ almost surely, and $\mathcal{L}^*_{\text{unif}}(\hat{f}) \leq \mathcal{L}^*_{\text{unif}}(f)$. Due to Eqn. (A.3) this last condition is equivalent to showing,

$$\mathbb{E}_{x\sim p} \inf_{x^- \not\succ x} \|\hat{f}(x) - \hat{f}(x^-)\|^2 \geq \mathbb{E}_{x\sim p} \inf_{x^- \not\succ x} \|f(x) - f(x^-)\|^2. \tag{A.5}$$

Fix a $c \in \mathcal{C}$, and let $x_c \in \arg\max_{x:h(x)=c} \inf_{x^- \not\succ x} \|f(x) - f(x^-)\|^2$. The maximum is guaranteed to be attained, as we explain now. Indeed we know the maximum is attained at some point in the closure $\partial\{x : h(x) = c\} \cup \{x : h(x) = c\}$. Since $\mathcal{X}$ is compact and connected, any point $\bar{x} \in \partial\{x : h(x) = c\} \setminus \{x : h(x) = c\}$ is such that $\inf_{x^- \not\succ \bar{x}} \|f(\bar{x}) - f(x^-)\|^2 = 0$ since $\bar{x}$ must belong to $\{x : h(x) = c'\}$ for some other c'. Such an $\bar{x}$ cannot be a solution unless all points in $\{x : h(x) = c\}$ also achieve 0, in which case we can simply take x_c to be a point in the interior of $\{x : h(x) = c\}$.

Now, define $\hat{f}(x) = f(x_c)$ for any x such that $h(x) = c$ and $\hat{f}(x) = f(x)$ otherwise. Let us first aim to show that Eqn. (A.5) holds for this $\hat{f}$. Let us begin to expand the left hand side of Eqn. (A.5),

$$\begin{aligned}
&\mathbb{E}_{x\sim p} \inf_{x^- \not\succ x} \|\hat{f}(x) - \hat{f}(x^-)\|^2 \\
&\qquad = \mathbb{E}_{\hat{c}\sim\rho}\mathbb{E}_{x\sim p(\cdot|\hat{c})} \inf_{x^- \not\succ x} \|\hat{f}(x) - \hat{f}(x^-)\|^2 \\
&\qquad = \rho(c)\mathbb{E}_{x\sim p(\cdot|c)} \inf_{x^- \not\succ x} \|\hat{f}(x) - \hat{f}(x^-)\|^2 \\
&\qquad\qquad + (1-\rho(c))\mathbb{E}_{\hat{c}\sim\rho(\cdot|\hat{c}\neq c)}\mathbb{E}_{x\sim p(\cdot|\hat{c})} \inf_{x^- \not\succ x} \|\hat{f}(x) - \hat{f}(x^-)\|^2 \\
&\qquad = \rho(c)\mathbb{E}_{x\sim p(\cdot|c)} \inf_{x^- \not\succ x} \|f(x_c) - f(x^-)\|^2 \\
&\qquad\qquad + (1-\rho(c))\mathbb{E}_{\hat{c}\sim\rho(\cdot|\hat{c}\neq c)}\mathbb{E}_{x\sim p(\cdot|\hat{c})} \inf_{x^- \not\succ x} \|\hat{f}(x) - \hat{f}(x^-)\|^2 \\
&\qquad = \rho(c) \inf_{x^- \not\succ x_c} \|f(x_c) - f(x^-)\|^2 \\
&\qquad\qquad + (1-\rho(c))\mathbb{E}_{\hat{c}\sim\rho(\cdot|\hat{c}\neq c)}\mathbb{E}_{x\sim p(\cdot|\hat{c})} \inf_{h(x^-)\neq\hat{c}} \|\hat{f}(x) - \hat{f}(x^-)\|^2
\end{aligned} \tag{A.6}$$

By construction, the first term can be lower bounded by $\inf_{x^- \not\succ x_c} \|f(x_c) - f(x^-)\|^2 \geq \mathbb{E}_{x\sim p(\cdot|c)} \inf_{h(x^-)\neq c} \|f(x) - f(x^-)\|^2$ for any x such that $h(x) = c$. To lower bound the second term, consider any fixed $\hat{c} \neq c$ and $x \sim p(\cdot|\hat{c})$ (so $h(x) = \hat{c}$). Define the following two subsets of the input space $\mathcal{X}$

$$\mathcal{A} = \{f(x^-) : f(x^-) \neq \hat{c} \text{ for } x^- \in \mathcal{X}\} \qquad \widehat{\mathcal{A}} = \{f(x^-) \in \mathcal{X} : \hat{f}(x^-) \neq \hat{c} \text{ for } x^- \in \mathcal{X}\}.$$

Since by construction the range of $\hat{f}$ is a subset of the range of f, we know that $\widehat{\mathcal{A}} \subseteq \mathcal{A}$. Combining this with the fact that $\hat{f}(x) = f(x)$ whenever $h(x) = \hat{c} \neq c$ we see,

$$\begin{aligned}
\inf_{h(x^-)\neq\hat{c}} \|\hat{f}(x) - \hat{f}(x^-)\|^2 &= \inf_{h(x^-)\neq\hat{c}} \|f(x) - \hat{f}(x^-)\|^2 \\
&= \inf_{u\in\widehat{\mathcal{A}}} \|f(x) - u\|^2 \\
&\geq \inf_{u\in\mathcal{A}} \|f(x) - u\|^2 \\
&= \inf_{h(x^-)\neq\hat{c}} \|f(x) - f(x^-)\|^2
\end{aligned}$$

Using these two lower bounds we may conclude that Eqn. (A.6) can be lower bounded by,

$$\rho(c)\mathbb{E}_{x\sim p(\cdot|c)} \inf_{h(x^-)\neq c} \|f(x) - f(x^-)\|^2 + (1-\rho(c))\mathbb{E}_{\hat{c}\sim\rho(\cdot|\hat{c}\neq c)}\mathbb{E}_{x\sim p(\cdot|\hat{c})} \inf_{h(x^-)\neq\hat{c}} \|f(x) - f(x^-)\|^2$$

which equals $\mathbb{E}_{x\sim p} \inf_{x^-\not\sim x} \|f(x) - f(x^-)\|^2$. We have therefore proved Eqn. (A.5). To summarize the current progress; given an embedding f we have constructed a new embedding $\hat{f}$ that attains lower $\mathcal{L}_{\text{unif}}$ loss and which is constant on x such that $\hat{f}$ is constant on $\{x : h(x) = c\}$. Enumerating $\mathcal{C} = \{c_1, c_2 \ldots, c_{|\mathcal{C}|}\}$, we may repeatedly apply the same argument to construct a sequence of embeddings $f_1, f_2, \ldots, f_{|\mathcal{C}|}$ such that f_i is constant on each of the following sets $\{x : h(x) = c_j\}$ for $j \leq i$. The final embedding in the sequence $f^* = f_{|\mathcal{C}|}$ is such that $\mathcal{L}^*_{\text{unif}}(f^*) \leq \mathcal{L}^*_{\text{unif}}(f)$ and therefore f^* is a minimizer. This embedding is constant on each of $\{x : h(x) = c_j\}$ for $j = 1, 2, \ldots, |\mathcal{C}|$. In other words, $f^*(x) = f^*(x^+)$ almost surely. We have proved the first claim.

Obtaining the second claim is a matter of manipulating $\mathcal{L}^*_\infty(f^*)$. Indeed, we know that $\mathcal{L}^*_\infty(f^*) = \mathcal{L}^*_{\text{unif}}(f^*) - 1/t^2$ and defining $\mathbf{v}_c = f^*(x) = f(x_c)$ for each $c \in \mathcal{C}$, this expression is minimized if and only if f^* attains,

$$\begin{aligned}\max_f \mathbb{E}_{x\sim p} \inf_{x^- \not\sim x} \|f(x) - f(x^-)\|^2 &= \max_f \mathbb{E}_{c\sim\rho}\mathbb{E}_{x\sim p(\cdot|c)} \inf_{h(x^-)\neq c} \|f(x) - f(x^-)\|^2 \\ &= \max_f \sum_{c\in\mathcal{C}} \rho(c) \cdot \inf_{h(x^-)\neq c} \|f(x) - f(x^-)\|^2 \\ &= \max_{\{\mathbf{v}_c \in \mathbb{S}^{d-1}/t\}_{c\in\mathcal{C}}} \sum_{c\in\mathcal{C}} \rho(c) \cdot \min_{c'\neq c} \|\mathbf{v}_c - \mathbf{v}_{c'}\|^2\end{aligned}$$

where the final equality inserts f^* as an optimal f and reparameterizes the maximum to be over the set of vectors $\{\mathbf{v}_c \in \mathbb{S}^{d-1}/t\}_{c\in\mathcal{C}}$. □

A.1.3 Downstream Generalization

Theorem 1. *Suppose ρ is uniform on $\mathcal{C}$ and f is such that $\mathcal{L}^*_\infty(f) - \inf_{\bar{f} \text{ measurable}} \mathcal{L}^*_\infty(\bar{f}) \leq \varepsilon$ with $\varepsilon \leq 1$. Let $\{\mathbf{v}^*_c \in \mathbb{S}^{d-1}/t\}_{c\in\mathcal{C}}$ be a solution to Problem 3.15, and define $\xi = \min_{c,c^-:c\neq c^-} \|\mathbf{v}^*_c - \mathbf{v}^*_{c^-}\| > 0$. Then there exists a set of vectors $\{\mathbf{v}_c \in \mathbb{S}^{d-1}/t\}_{c\in\mathcal{C}}$ such that the following 1-nearest neighbor classifier,*

$$\hat{h}(x) = \hat{c}, \quad \textit{where} \quad \hat{c} = \arg\min_{\bar{c}\in\mathcal{C}} \|f(x) - \mathbf{v}_{\bar{c}}\| \quad \textit{(ties broken arbitrarily)}$$

achieves misclassification risk,

$$\mathbb{P}(\hat{h}(x) \neq c) \leq \frac{8\varepsilon}{(\xi^2 - 2\,|\mathcal{C}|\,(1 + 1/t)\varepsilon^{1/2})^2}$$

Proof. To begin, using the definition of $\hat{h}$ we know that for any $0 < \delta < \xi$,

$$\begin{aligned}\mathbb{P}_{x,c}(\hat{h}(x)=c) &= \mathbb{P}_{x,c}\left(\|f(x)-\mathbf{v}_c\| \leq \min_{c^-:c^-\neq c}\|f(x)-\mathbf{v}_{c^-}\|\right)\\ &\geq \mathbb{P}_{x,c}\left(\|f(x)-\mathbf{v}_c\| \leq \delta, \quad \text{and} \quad \delta \leq \min_{c^-:c^-\neq c}\|f(x)-\mathbf{v}_{c^-}\|\right)\\ &\geq 1-\mathbb{P}_{x,c}\left(\|f(x)-\mathbf{v}_c\|>\delta\right)-\mathbb{P}_{x,c}\big(\min_{c^-:c^-\neq c}\|f(x)-\mathbf{v}_{c^-}\|<\delta\big)\end{aligned}$$

So to prove the result, our goal is now to bound these two probabilities. To do so, we use the bound on the excess risk. Indeed, combining the fact $\mathcal{L}^*_\infty(f) - \inf_{\bar{f}\text{ measurable}}\mathcal{L}^*_\infty(\bar{f}) \leq \varepsilon$ with the notational rearrangements before Theorem 9 we observe that $\mathbb{E}_{x,x^+}\|f(x)-f(x^+)\|^2 \leq 2\varepsilon$.

We have,

$$2\varepsilon \geq \mathbb{E}_{x,x^+}\|f(x)-f(x^+)\|^2 = \mathbb{E}_{c\sim\rho}\mathbb{E}_{x^+\sim p(\cdot|c)}\mathbb{E}_{x\sim p(\cdot|c)}\|f(x)-f(x^+)\|^2.$$

For fixed c, x^+, let $x_c \in \arg\min_{\overline{\{x^+:h(x^+)=c\}}}\mathbb{E}_{x\sim p(\cdot|c)}\|f(x)-f(x^+)\|^2$ where we extend the minimum to be over the closure, a compact set, to guarantee it is attained. Then we have

$$2\varepsilon \geq \mathbb{E}_{c\sim\rho}\mathbb{E}_{x^+\sim p(\cdot|c)}\mathbb{E}_{x\sim p(\cdot|c)}\|f(x)-f(x^+)\|^2 \geq \mathbb{E}_{c\sim\rho}\mathbb{E}_{x\sim p(\cdot|c)}\|f(x)-\mathbf{v}_c\|^2$$

where we have now defined $\mathbf{v}_c = f(x_c)$ for each $c \in \mathcal{C}$. Note in particular that $\mathbf{v}_c$ lies on the surface of the hypersphere $\mathbb{S}^{d-1}/t$. This enables us to obtain the follow bound using Markov's inequality,

$$\begin{aligned}\mathbb{P}_{x,c}\left(\|f(x)-\mathbf{v}_c\|>\delta\right) &= \mathbb{P}_{x,c}\left(\|f(x)-\mathbf{v}_c\|^2>\delta^2\right)\\ &\leq \frac{\mathbb{E}_{x,c}\|f(x)-\mathbf{v}_c\|^2}{\delta^2}\\ &\leq \frac{2\varepsilon}{\delta^2}.\end{aligned}$$

so it remains still to bound $\mathbb{P}_{x,c}\big(\min_{c^-:c^-\neq c}\|f(x)-\mathbf{v}_{c^-}\|<\delta\big)$. Defining $\xi' =$

$\min_{c,c^-:c\neq c^-} \|\mathbf{v}_c - \mathbf{v}_{c^-}\|$, we have the following fact (proven later).

Fact (see lemma 5): $\xi' \geq \sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\sqrt{\varepsilon}}$.

Using this fact we are able to get control over the tail probability as follows,

$$\begin{aligned}
\mathbb{P}_{x,c}\left(\min_{c^-:c^-\neq c} \|f(x) - \mathbf{v}_{c^-}\| < \delta\right) &\leq \mathbb{P}_{x,c}\left(\|f(x) - \mathbf{v}_c\| > \xi' - \delta\right) \\
&\leq \mathbb{P}_{x,c}\left(\|f(x) - \mathbf{v}_c\| > \xi - \sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\varepsilon^{1/2}} - \delta\right) \\
&= \mathbb{P}_{x,c}\left(\|f(x) - \mathbf{v}_c\|^2 > (\sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\varepsilon^{1/2}} - \delta)^2\right) \\
&\leq \frac{2\varepsilon}{(\sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\varepsilon^{1/2}} - \delta)^2}.
\end{aligned}$$

where this inequality holds for for any $0 \leq \delta \leq \sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\varepsilon^{1/2}}$.

Gathering together our tail probability bounds we find that $\mathbb{P}_{x,c}(\hat{h}(x) = c) \geq 1 - \frac{2\varepsilon}{\delta^2} - \frac{2\varepsilon}{(\sqrt{\xi^2 - 2|\mathcal{C}|(1+1/t)\varepsilon^{1/2}} - \delta)^2}$ for any $0 \leq \delta \leq \sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\varepsilon^{1/2}}$. That is,

$$\mathbb{P}_{x,c}(\hat{h}(x) \neq c) \leq \frac{2\varepsilon}{\delta^2} + \frac{2\varepsilon}{(\sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\varepsilon^{1/2}} - \delta)^2}$$

Since this holds for any $0 \leq \delta \leq \sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\varepsilon^{1/2}}$,

$$\mathbb{P}_{x,c}(\hat{h}(x) \neq c) \leq \min_{0\leq\delta\leq\sqrt{\xi^2-2|\mathcal{C}|\varepsilon}} \left\{\frac{2\varepsilon}{\delta^2} + \frac{2\varepsilon}{(\sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\varepsilon^{1/2}} - \delta)^2}\right\}.$$

Elementary calculus shows that the minimum is attained at $\delta = \frac{\sqrt{\xi^2-2|\mathcal{C}|(1+1/t)\varepsilon^{1/2}}}{2}$. Plugging this in yields the final bound,

$$\mathbb{P}(\hat{h}(x) \neq c) \leq \frac{8\varepsilon}{(\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\varepsilon^{1/2})^2}.$$

□

Lemma 5. *Consider the same setting as introduced in Theorem 4. In particular define*

$$\xi' = \min_{c,c^-:c\neq c^-} \|\mathbf{v}_c - \mathbf{v}_{c-}\|, \qquad \xi = \min_{c,c^-:c\neq c^-} \|\mathbf{v}_c^* - \mathbf{v}_{c-}^*\|.$$

where $\{\mathbf{v}_c^* \in \mathbb{S}^{d-1}/t\}_{c\in\mathcal{C}}$ *is a solution to Problem 3.15, and* $\{\mathbf{v}_c \in \mathbb{S}^{d-1}/t\}_{c\in\mathcal{C}}$ *is defined via* $\mathbf{v}_c = f(x_c)$ *with* $x_c \in \arg\min_{\{x^+:h(x^+)=c\}} \mathbb{E}_{x\sim p(\cdot|c)}\|f(x) - f(x^+)\|^2$ *for each* $c \in \mathcal{C}$*. Then we have,*

$$\xi' \geq \sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1+1/t)\varepsilon^{1/2}}.$$

Proof. Define,

$$X = \min_{c^-:c^-\neq c} \|\mathbf{v}_c - \mathbf{v}_{c-}\|^2, \qquad X^* = \min_{c^-:c^-\neq c} \|\mathbf{v}_c^* - \mathbf{v}_{c-}^*\|^2.$$

X and X^* are random due to the randomness of $c \sim \rho$. We can split up the following expectation by conditioning on the event $\{X \leq X^*\}$ and its complement,

$$\mathbb{E}\,|X - X^*| = \mathbb{P}(X \geq X^*)\mathbb{E}[X - X^*] + \mathbb{P}(X \leq X^*)\mathbb{E}[X^* - X]. \tag{A.7}$$

Using $\mathcal{L}_\infty^*(f) - \inf_{\bar{f}\text{ measurable}} \mathcal{L}_\infty^*(\bar{f}) \leq \varepsilon$ and the notational re-writing of the objective $\mathcal{L}_\infty^*$ introduced before Theorem 9, we observe the following fact, whose proof we give in a separate lemma after the conclusion of this proof.

Fact (see lemma 6): $\mathbb{E}X^* - 2(1+1/t)\sqrt{\varepsilon} \leq \mathbb{E}X \leq \mathbb{E}X^*$.

This fact implies in particular $\mathbb{E}[X - X^*] \leq 0$ and $\mathbb{E}[X^* - X] \leq 2(1+1/t)\sqrt{\varepsilon}$. Inserting both inequalities into Eqn. A.7 we find that $\mathbb{E}\,|X - X^*| \leq 2(1+1/t)\sqrt{\varepsilon}$. In other words, since ρ is uniform,

$$\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \left| \min_{c^-:c^-\neq c} \|\mathbf{v}_c - \mathbf{v}_{c-}\|^2 - \min_{c^-:c^-\neq c} \|\mathbf{v}_c^* - \mathbf{v}_{c-}^*\|^2 \right| \leq 2(1+1/t)\sqrt{\varepsilon}.$$

From which we can say that for any $c \in \mathcal{C}$,

$$\left| \min_{c^- : c^- \neq c} \|\mathbf{v}_c - \mathbf{v}_{c-}\|^2 - \min_{c^- : c^- \neq c} \|\mathbf{v}_c^* - \mathbf{v}_{c-}^*\|^2 \right| \leq 2\,|\mathcal{C}|\,(1 + 1/t)\sqrt{\varepsilon}.$$

So

$$\begin{aligned} \min_{c^- : c^- \neq c} \|\mathbf{v}_c - \mathbf{v}_{c-}\| &\geq \sqrt{\min_{c^- : c^- \neq c} \|\mathbf{v}_c^* - \mathbf{v}_{c-}^*\|^2 - 2\,|\mathcal{C}|\,(1 + 1/t)\varepsilon^{1/2}} \\ &\geq \sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1 + 1/t)\varepsilon^{1/2}} \end{aligned}$$

Since this holds for any $c \in \mathcal{C}$, we conclude that $\xi' \geq \sqrt{\xi^2 - 2\,|\mathcal{C}|\,(1 + 1/t)\varepsilon^{1/2}}$. □

Lemma 6. *Consider the same setting as introduced in Theorem 4. Define also,*

$$X = \min_{c^- : c^- \neq c} \|\mathbf{v}_c - \mathbf{v}_{c-}\|^2, \qquad X^* = \min_{c^- : c^- \neq c} \|\mathbf{v}_c^* - \mathbf{v}_{c-}^*\|^2,$$

where $\mathbf{v}_c = f(x_c)$ *with* $x_c \in \arg\min_{\overline{\{x^+ : h(x^+) = c\}}} \mathbb{E}_{x \sim p(\cdot|c)} \|f(x) - f(x^+)\|^2$ *for each* $c \in \mathcal{C}$. *We have,*

$$\mathbb{E}X^* - 2(1 + 1/t)\sqrt{\varepsilon} \leq \mathbb{E}X \leq \mathbb{E}X^*.$$

Proof. By Theorem 3.15 we know there is an f^* attaining the minimum $\inf_{\bar{f} \text{ measurable}} \mathcal{L}_\infty^*(\bar{f})$ and that this f^* attains $\mathcal{L}_{\text{align}}^*(f^*) = 0$, and also minimizes the uniformity term $\mathcal{L}_{\text{unif}}^*(f)$, taking the value $\mathcal{L}_{\text{unif}}^*(f^*) = \mathbb{E}_{c \sim \rho} \max_{c^- : c^- \neq c} \mathbf{v_c^*}^\top \mathbf{v}_{c^-}^*$. Because of this we find,

$$\begin{aligned} \mathcal{L}_{\text{unif}}^*(f) &\leq \big(\mathcal{L}_\infty^*(f) - \mathcal{L}_\infty^*(f^*)\big) + \big(\mathcal{L}_{\text{align}}^*(f^*) - \mathcal{L}_{\text{align}}^*(f)\big) + \mathcal{L}_{\text{unif}}^*(f^*) \\ &\leq \big(\mathcal{L}_\infty^*(f) - \mathcal{L}_\infty^*(f^*)\big) + \mathcal{L}_{\text{unif}}^*(f^*) \\ &\leq \varepsilon + \mathcal{L}_{\text{unif}}^*(f^*) \\ &= \varepsilon + \mathbb{E}_{c \sim \rho} \max_{c^- : c^- \neq c} \mathbf{v_c^*}^\top \mathbf{v}_{c^-}^*. \end{aligned}$$

Since we would like to bound $\mathbb{E}_{c \sim \rho} \max_{c^- : c^- \neq c} \mathbf{v_c}^\top \mathbf{v}_{c^-}$ in terms of $\mathbb{E}_{c \sim \rho} \max_{c^- : c^- \neq c} \mathbf{v_c^*}^\top \mathbf{v}_{c^-}^*$, this observation means that is suffices to bound $\mathbb{E}_{c \sim \rho} \max_{c^- : c^- \neq c} \mathbf{v_c}^\top \mathbf{v}_{c^-}$ in terms of

$\mathcal{L}^*_{\text{unif}}(f)$. To this end, note that for a fixed c, and x such that $h(x) = c$ we have,

$$\begin{aligned}
\sup_{x^- \not\succ x} f(x)^\top f(x^-) &= \sup_{x^- \not\succ x} \left\{ \mathbf{v}_c{}^\top f(x^-) + (f(x) - \mathbf{v}_c)^\top f(x^-) \right\} \\
&= \sup_{x^- \not\succ x} \mathbf{v}_c{}^\top f(x^-) - \|f(x) - \mathbf{v}_c\|/t \\
&\geq \max_{x^- \in \{x_c\}_{c \in \mathcal{C}}} \mathbf{v}_c{}^\top f(x^-) - \|f(x) - \mathbf{v}_c\|/t \\
&= \max_{c^- \neq c} \mathbf{v}_c{}^\top \mathbf{v}_{c^-} - \|f(x) - \mathbf{v}_c\|/t
\end{aligned}$$

where the inequality follows since $\{x_c\}_{c \in \mathcal{C}}$ is a subset of the closure of $\{x^- : x^- \not\succ x\}$. Taking expectations over c, x,

$$\begin{aligned}
\mathcal{L}^*_{\text{unif}}(f) &= \mathbb{E}_{x,c} \sup_{x^- \not\succ x} f(x)^\top f(x^-) \\
&\geq \mathbb{E}_{c \sim \rho} \max_{c^- \neq c} \mathbf{v}_c{}^\top \mathbf{v}_{c^-} - \mathbb{E}_{x,c} \|f(x) - \mathbf{v}_c\|/t \\
&\geq \mathbb{E}_{c \sim \rho} \max_{c^- \neq c} \mathbf{v}_c{}^\top \mathbf{v}_{c^-} - \sqrt{\mathbb{E}_{x,c} \|f(x) - \mathbf{v}_c\|^2}/t \\
&\geq \mathbb{E}_{c \sim \rho} \max_{c^- \neq c} \mathbf{v}_c{}^\top \mathbf{v}_{c^-} - \sqrt{\varepsilon}/t.
\end{aligned}$$

So since $\varepsilon \leq \sqrt{\varepsilon}$, we have found that

$$\mathbb{E}_{c \sim \rho} \max_{c^- \neq c} \mathbf{v}_c{}^\top \mathbf{v}_{c^-} \leq \sqrt{\varepsilon}/t + \varepsilon + \mathbb{E}_{c \sim \rho} \max_{c^- : c^- \neq c} \mathbf{v}_{\mathbf{c}}^{*\top} \mathbf{v}^*_{c^-} \leq (1 + 1/t)\sqrt{\varepsilon} + \mathbb{E}_{c \sim \rho} \max_{c^- : c^- \neq c} \mathbf{v}_{\mathbf{c}}^{*\top} \mathbf{v}^*_{c^-}.$$

Of course we also have,

$$\mathbb{E}_{c \sim \rho} \max_{c^- : c^- \neq c} \mathbf{v}_{\mathbf{c}}^{*\top} \mathbf{v}^*_{c^-} = \mathcal{L}^*_{\text{unif}}(f^*) \leq \mathbb{E}_{c \sim \rho} \max_{c^- : c^- \neq c} \mathbf{v}_{\mathbf{c}}{}^\top \mathbf{v}_{c^-}$$

since the embedding $f(x) = \mathbf{v}_c$ whenever $h(x) = c$ is also a feasible solution.

Combining these two inequalities with the simple identity $\mathbf{x}^\top \mathbf{y} = 1/t^2 - \|\mathbf{x}-\mathbf{y}\|^2/2$ for all length $1/t$ vectors $\mathbf{x}, \mathbf{y}$, we find,

$$\begin{aligned} 1/t^2 - \mathbb{E}_{c\sim\rho} \max_{c^-:c^-\neq c} \|\mathbf{v}_{\mathbf{c}}^* - \mathbf{v}_{c^-}^*\|^2/2 &\leq 1/t^2 - \mathbb{E}_{c\sim\rho} \max_{c^-:c^-\neq c} \|\mathbf{v}_{\mathbf{c}} - \mathbf{v}_{c^-}\|^2/2 \\ &\leq 1/t^2 - \mathbb{E}_{c\sim\rho} \max_{c^-:c^-\neq c} \|\mathbf{v}_{\mathbf{c}}^* - \mathbf{v}_{c^-}^*\|^2/2 + (1+1/t)\sqrt{\varepsilon}. \end{aligned}$$

Subtracting $1/t^2$ and multiplying by -2 yields the result. □

A.1.4 Proofs of Theoretical Results on Debiased Contratstive Loss

A.1.5 Proof of Lemma 1

The first result we give shows the relation between the unbiased, and conventional (sample biased) objective.

Lemma 1. *1 For any embedding f and finite N, we have*

$$L_{\text{Biased}}^N(f) \geq L_{\text{Unbiased}}^N(f) + \mathbb{E}_{x\sim p}\left[0 \wedge \log \frac{\mathbb{E}_{x^+\sim p_x^+} \exp f(x)^\top f(x^+)}{\mathbb{E}_{x^-\sim p_x^-} \exp f(x)^\top f(x^-)}\right] - e^{3/2}\sqrt{\frac{\pi}{2N}}.$$

where $a \wedge b$ denotes the minimum of two real numbers a and b.

Proof. We use the notation $h(x, \bar{x}) = \exp^{f(x)^\top f(\bar{x})}$ for the critic. We will use Theorem 3 to prove this lemma. Setting $\tau^+ = 0$, Theorem 3 states that

$$\begin{aligned} &\mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}}\left[-\log \frac{h(x,x^+)}{h(x,x^+) + N\mathbb{E}_{x^-\sim p} h(x,x_i^-)}\right] \\ &\qquad - \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+ \\ \{x_i^-\}_{i=1}^N \sim p^N}}\left[-\log \frac{h(x,x^+)}{h(x,x^+) + \sum_{i=1}^N h(x,x_i^-)}\right] \leq e^{3/2}\sqrt{\frac{\pi}{2N}}. \end{aligned}$$

Equipped with this inequality, the biased objective can be decomposed into the

sum of the debiased objective and a second term as follows:

$$
\begin{aligned}
L^N_{\text{Biased}}(f) &= \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+ \\ \{x_i^-\}_{i=1}^N\sim p^N}}\left[-\log\frac{h(x,x^+)}{h(x,x^+)+\sum_{i=1}^N h(x,x_i^-)}\right] \\
&\geq \mathbb{E}_{x\sim p, x^+\sim p_x^+}\left[-\log\frac{h(x,x^+)}{h(x,x^+)+N\mathbb{E}_{x^-\sim p_x}h(x,x^-)}\right] - e^{3/2}\sqrt{\frac{\pi}{2N}} \\
&= \mathbb{E}_{x\sim p, x^+\sim p_x^+}\left[-\log\frac{h(x,x^+)}{h(x,x^+)+N\mathbb{E}_{x^-\sim p_x^-}h(x,x^-)}\right] \\
&\qquad + \mathbb{E}_{x\sim p, x^+\sim p_x^+}\left[\log\frac{h(x,x^+)+N\mathbb{E}_{x^-\sim p_x}h(x,x^-)}{h(x,x^+)+N\mathbb{E}_{x^-\sim p_x^-}h(x,x^-)}\right] - e^{3/2}\sqrt{\frac{\pi}{2N}} \\
&= L^N_{\text{Debiased}}(f) + \mathbb{E}_{x\sim p, x^+\sim p_x^+}\left[\log\frac{h(x,x^+)+N\mathbb{E}_{x^-\sim p_x}h(x,x^-)}{h(x,x^+)+N\mathbb{E}_{x^-\sim p_x^-}h(x,x^-)}\right] - e^{3/2}\sqrt{\frac{\pi}{2N}} \\
&= L^N_{\text{Debiased}}(f) + \mathbb{E}_{x\sim p, x^+\sim p_x^+}\left[\log\frac{h(x,x^+)+\tau^-N\mathbb{E}_{x^-\sim p_x^-}h(x,x^-)+\tau^+N\mathbb{E}_{x^-\sim p_x^+}h(x,x^-)}{h(x,x^+)+\tau^-N\mathbb{E}_{x^-\sim p_x^-}h(x,x^-)+\tau^+N\mathbb{E}_{x^-\sim p_x^-}h(x,x^-)}\right] \\
&\qquad - e^{3/2}\sqrt{\frac{\pi}{2N}}.
\end{aligned}
$$

If $\mathbb{E}_{x^-\sim p_x^+}h(x,x^-) \geq \mathbb{E}_{x^-\sim p_x^-}h(x,x^-)$, then this expression can be lower bounded by $L^N_{\text{Debiased}}(f) + \log 1 = L^N_{\text{Debiased}}(f)$. Otherwise, if $\mathbb{E}_{x^-\sim p_x^+}h(x,x^-) \leq \mathbb{E}_{x^-\sim p_x^-}h(x,x^-)$, we can use the elementary fact that $\frac{a+c}{b+c} \geq \frac{a}{b}$ for $a \leq b$ and $a,b,c \geq 0$. Combining these two cases, we conclude that

$$
L^N_{\text{Biased}}(f) \geq L^N_{\text{Unbiased}}(f) + \mathbb{E}_{x\sim p}\left[0 \wedge \log\frac{\mathbb{E}_{x^+\sim p_x^+}\exp f(x)^\top f(x^+)}{\mathbb{E}_{x^-\sim p_x^-}\exp f(x)^\top f(x^-)}\right] - e^{3/2}\sqrt{\frac{\pi}{2N}},
$$

where we replaced the dummy variable x^- in the numerator by x^+. □

A.1.6 Proof of Lemma 2

The next result is a consequence of the dominated convergence theorem.

Lemma 2. *2 For fixed Q and $N \to \infty$, it holds that*

$$\mathbb{E}_{\substack{x\sim p, x^+\sim p_x^+ \\ \{x_i^-\}_{i=1}^N \sim p_x^{-N}}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{N}\sum_{i=1}^N e^{f(x)^T f(x_i^-)}} \right]$$
$$\longrightarrow \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}} \left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{\tau^-}(\mathbb{E}_{x^-\sim p}[e^{f(x)^T f(x^-)}] - \tau^+ \mathbb{E}_{v\sim p_x^+}[e^{f(x)^T f(v)}])} \right].$$

Proof. Since the contrastive loss is bounded, applying the Dominated Convergence Theorem completes the proof:

$$\begin{aligned}
&\lim_{N\to\infty} \mathbb{E}\left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{N}\sum_{i=1}^N e^{f(x)^T f(x_i^-)}} \right] \\
&= \mathbb{E}\left[\lim_{N\to\infty} -\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \frac{Q}{N}\sum_{i=1}^N e^{f(x)^T f(x_i^-)}} \right] \quad \text{(Dominated Convergence Theorem)} \\
&= \mathbb{E}\left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + Q\mathbb{E}_{x^-\sim p_x^-} e^{f(x)^T f(x^-)}} \right].
\end{aligned}$$

Since $p_x^-(x') = (p(x') - \tau^+ p_x^+(x'))/\tau^-$ and by the linearity of the expectation, we have

$$\mathbb{E}_{x^-\sim p_x^-} e^{f(x)^T f(x^-)} = \tau^-(\mathbb{E}_{x^-\sim p}[e^{f(x)^T f(x^-)}] - \tau^+ \mathbb{E}_{x^-\sim p_x^+}[e^{f(x)^T f(x^-)}]),$$

which completes the proof. □

A.1.7 Proof of Theorem 3

In order to prove Theorem 3, which shows that the empirical estimate of the asymptotic debiased objective is a good estimate, we first seek a bound on the tail probability that the difference between the integrands of the asymptotic and non-asymptotic objective functions i slarge. That is, we wish to bound the probability that the following quantity is greater than ε:

$$\Delta = \left| -\log \frac{h(x,x^+)}{h(x,x^+) + Qg(x,\{u_i\}_{i=1}^N, \{v_i\}_{i=1}^M)} + \log \frac{h(x,x^+)}{h(x,x^+) + Q\mathbb{E}_{x^-\sim p_x^-} h(x,x^-)} \right|,$$

where we again write $h(x, \bar{x}) = \exp^{f(x)^\top f(\bar{x})}$ for the critic. Note that implicitly, Δ depends on x, x^+ and the collections $\{u_i\}_{i=1}^N$ and $\{v_i\}_{i=1}^M$. We achieve control over the tail via the following lemma.

Lemma 3. *A.2 Let x and x^+ in $\mathcal{X}$ be fixed. Further, let $\{u_i\}_{i=1}^N$ and $\{v_i\}_{i=1}^M$ be collections of i.i.d. random variables sampled from p and p_x^+ respectively. Then for all $\varepsilon > 0$,*

$$\mathbb{P}(\Delta \geq \varepsilon) \leq 2\exp\left(-\frac{N\varepsilon^2(\tau^-)^2}{2e^3}\right) + 2\exp\left(-\frac{M\varepsilon^2(\tau^-/\tau^+)^2}{2e^3}\right).$$

We delay the proof until after we prove Theorem 3, which we are ready to prove with this fact in hand.

Theorem 1. *3 For any embedding f and finite N and M, we have*

$$\left|\widetilde{L}^N_{\text{Debiased}}(f) - L^{N,M}_{\text{Debiased}}(f)\right| \leq \frac{e^{3/2}}{\tau^-}\sqrt{\frac{\pi}{2N}} + \frac{e^{3/2}\tau^+}{\tau^-}\sqrt{\frac{\pi}{2M}}.$$

Proof. By Jensen's inequality, we may push the absolute value inside the expectation to see that $|\widetilde{L}^N_{\text{Unbiased}}(f) - L^{N,M}_{\text{Debiased}}(f)| \leq \mathbb{E}\Delta$. All that remains is to exploit the exponential tail bound of Lemma *A*.2.

To do this we write the expectation of Δ for fixed x, x^+ as the integral of its tail probability,

$$\begin{aligned}\mathbb{E}\ \Delta = \mathbb{E}_{x,x^+}\left[\mathbb{E}[\Delta|x, x^+]\right] &= \mathbb{E}_{x,x^+}\left[\int_0^\infty \mathbb{P}(\Delta \geq \varepsilon|x, x^+)\mathrm{d}\varepsilon\right] \\ &\leq \int_0^\infty 2\exp\left(-\frac{N\varepsilon^2(\tau^-)^2}{2e^3}\right)\mathrm{d}\varepsilon + \int_0^\infty 2\exp\left(-\frac{M\varepsilon^2(\tau^-/\tau^+)^2}{2e^3}\right)\mathrm{d}\varepsilon.\end{aligned}$$

The outer expectation disappears since the tail probably bound of Theorem 3 holds uniformly for all fixed x, x^+. Both integrals can be computed analytically using the classical identity

$$\int_0^\infty e^{-cz^2}\mathrm{d}z = \frac{1}{2}\sqrt{\frac{\pi}{c}}.$$

Applying the identity to each integral we finally obtain the claimed bound,

$$\sqrt{\frac{2e^3\pi}{(\tau^-)^2 N}} + \sqrt{\frac{2e^3\pi}{(\tau^-/\tau^+)^2 M}} = \frac{e^{3/2}}{\tau^-}\sqrt{\frac{2\pi}{N}} + \frac{e^{3/2}\tau^+}{\tau^-}\sqrt{\frac{2\pi}{M}}.$$

□

We still owe the reader a proof of Lemma 3, which we give now.

Proof of Lemma 3. We first decompose the probability as

$$\begin{aligned}
&\mathbb{P}\bigg(\bigg| - \log \frac{h(x,x^+)}{h(x,x^+) + Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)} + \log \frac{h(x,x^+)}{h(x,x^+) + Q\mathbb{E}_{x^-\sim p_x^-} h(x,x^-)}\bigg| \geq \varepsilon\bigg) \\
&= \mathbb{P}\bigg(\bigg| \log\big\{h(x,x^+) + Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)\big\} - \log\big\{h(x,x^+) + Q\mathbb{E}_{x^-\sim p_x^-} h(x,x^-)\big\}\bigg| \geq \varepsilon\bigg) \\
&= \mathbb{P}\bigg(\log\big\{h(x,x^+) + Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)\big\} - \log\big\{h(x,x^+) + Q\mathbb{E}_{x^-\sim p_x^-} h(x,x^-)\big\} \geq \varepsilon\bigg) \\
&\quad + \mathbb{P}\bigg(- \log\big\{h(x,x^+) + Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)\big\} + \log\big\{h(x,x^+) + Q\mathbb{E}_{x^-\sim p_x^-} h(x,x^-)\big\} \geq \varepsilon\bigg)
\end{aligned}$$

where the final equality holds simply because $|X| \geq \varepsilon$ if and only if $X \geq \varepsilon$ or $-X \geq \varepsilon$. The first term can be bounded as

$$\begin{aligned}
&\mathbb{P}\bigg(\log\big\{h(x,x^+) + Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)\big\} - \log\big\{h(x,x^+) + Q\mathbb{E}_{x^-\sim p_x^-} h(x,x^-)\big\} \geq \varepsilon\bigg) \\
&= \mathbb{P}\bigg(\log \frac{h(x,x^+) + Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)}{h(x,x^+) + Q\mathbb{E}_{x^-\sim p_x^-} h(x,x^-)} \geq \varepsilon\bigg) \\
&\leq \mathbb{P}\bigg(\frac{Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M) - Q\mathbb{E}_{x^-\sim p_x^-} h(x,x^-)}{h(x,x^+) + Q\mathbb{E}_{x^-\sim p_x^-} h(x,x^-)} \geq \varepsilon\bigg) \\
&= \mathbb{P}\bigg(g(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M) - \mathbb{E}_{x^-\sim p_x^-} h(x,x^-) \geq \varepsilon\bigg\{\frac{1}{Q}h(x,x^+) + \mathbb{E}_{x^-\sim p_x^-} h(x,x^-)\bigg\}\bigg) \\
&\leq \mathbb{P}\bigg(g(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M) - \mathbb{E}_{x^-\sim p_x^-} h(x,x^-) \geq \varepsilon e^{-1}\bigg). \qquad \text{(A.8)}
\end{aligned}$$

The first inequality follows by applying the fact that $\log x \leq x - 1$ for $x > 0$. The second inequality holds since $\frac{1}{Q}h(x,x^+) + \mathbb{E}_{x^-\sim p_x^-} h(x,x^-) \geq 1/e$. Next, we move on

to bounding the second term, which proceeds similarly, using the same two bounds.

$$
\begin{aligned}
&\mathbb{P}\Big\{ -\log\left(h(x,x^+) + Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)\right\} + \log\left\{h(x,x^+) + Q\mathbb{E}_{x^-\sim p_x^-}h(x,x^-)\right\} \geq \varepsilon\Big) \\
&= \mathbb{P}\left(\log\frac{h(x,x^+) + Q\mathbb{E}_{x^-\sim p_x^-}h(x,x^-)}{h(x,x^+) + Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)} \geq \varepsilon\right) \\
&\leq \mathbb{P}\left(\frac{Q\mathbb{E}_{x^-\sim p_x^-}h(x,x^-) - Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)}{h(x,x^+) + Qg(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)} \geq \varepsilon\right) \\
&= \mathbb{P}\left(\mathbb{E}_{x^-\sim p_x^-}h(x,x^-) - g(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M) \geq \varepsilon\left\{\frac{1}{Q}h(x,x^+) + g(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)\right\}\right) \\
&\leq \mathbb{P}\left(\mathbb{E}_{x^-\sim p_x^-}h(x,x^-) - g(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M) \geq \varepsilon e^{-1}\right). \qquad \text{(A.9)}
\end{aligned}
$$

Combining equation (A.8) and equation (A.9), we have

$$
\mathbb{P}(\Delta \geq \varepsilon) \leq \mathbb{P}\left(\left|g(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M) - \mathbb{E}_{x^-\sim p_x^-}h(x,x^-)\right| \geq \varepsilon e^{-1}\right).
$$

It therefore suffices to bound the right hand tail probability. We are bounding the tail of a difference of the form $|\max(a,b)-c|$ where $c \geq b$. Notice that $|\max(a,b)-c| \leq |a-c|$. If $a > b$ then this relation is obvious, while if $a \leq b$ we have $|\max(a,b)-c| = |b-c| = c-b \leq c-a \leq |a-c|$. Using this elementary observation, we can decompose the random variable whose tail we wish to control as follows:

$$
\begin{aligned}
&\left|g(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M) - \mathbb{E}_{x^-\sim p_x^-}h(x,x^-)\right| \\
&\quad\leq \frac{1}{\tau^-}\left|\frac{1}{N}\sum_{i=1}^N \mathbb{E}_{x\sim p}h(x,u_i) - \mathbb{E}_{\substack{x^-\sim p\\ x\sim p}}h(x,x^-)\right| + \frac{\tau^+}{\tau^-}\left|\frac{1}{M}\sum_{i=1}^M \mathbb{E}_{x\sim p}h(x,v_i) - \mathbb{E}_{\substack{x^-\sim p_x^+\\ x\sim p}}h(x,x^-)\right|
\end{aligned}
$$

Using this observation, we find that

$$
\begin{aligned}
&\mathbb{P}\left(\left|g(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M) - \mathbb{E}_{x^-\sim p_x^-}h(x,x^-)\right| \geq \varepsilon e^{-1}\right) \\
&\leq \mathbb{P}\left(\left|\frac{1}{\tau^-}\left(\frac{1}{N}\sum_{i=1}^N e^{f(x)^T f(u_i)} - \tau^+\frac{1}{M}\sum_{i=1}^M e^{f(x)^T f(v_i)}\right) - \mathbb{E}_{x^-\sim p_x^-}h(x,x^-)\right| \geq \varepsilon e^{-1}\right) \\
&\leq \mathrm{I}(\varepsilon) + \mathrm{II}(\varepsilon).
\end{aligned}
$$

where

$$\mathrm{I}(\varepsilon) = \mathbb{P}\left(\frac{1}{\tau^-}\left|\frac{1}{N}\sum_{i=1}^{N} h(x,u_i) - \mathbb{E}_{x^-\sim p} h(x,x^-)\right| \geq \frac{\varepsilon e^{-1}}{2}\right)$$
$$\mathrm{II}(\varepsilon) = \mathbb{P}\left(\frac{\tau^+}{\tau^-}\left|\frac{1}{M}\sum_{i=1}^{M} h(x,v_i) - \mathbb{E}_{x^-\sim p_x^+} h(x,x^-)\right| \geq \frac{\varepsilon e^{-1}}{2}\right).$$

Hoeffding's inequality states that if $X, X_1, \ldots, X_N$ are i.i.d random variables bounded in the range $[a,b]$, then

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{N} X_i - \mathbb{E}X\right| \geq \varepsilon\right) \leq 2\exp\left(-\frac{2N\varepsilon^2}{b-a}\right).$$

In our particular case, $e^{-1} \leq h(x,\bar{x}) \leq e$, yielding the following bound on the tails of both terms:

$$\mathrm{I}(\varepsilon) \leq 2\exp\left(-\frac{N\varepsilon^2(\tau^-)^2}{2e^3}\right) \quad \text{and} \quad \mathrm{II}(\varepsilon) \leq 2\exp\left(-\frac{M\varepsilon^2(\tau^-/\tau^+)^2}{2e^3}\right).$$

□

A.1.8 Proof of Lemma 4

Lemma 4. *4 For any embedding f, whenever $N \geq K-1$ we have*

$$L_{\mathrm{Sup}}(f) \leq L^{\mu}_{\mathrm{Sup}}(f) \leq \widetilde{L}^{N}_{\mathrm{Debiased}}(f).$$

Proof. We first show that $N = K-1$ gives the smallest loss:

$$\begin{aligned}
\widetilde{L}^{N}_{\mathrm{Unbiased}}(f) &= \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}}\left[-\log\frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + N\mathbb{E}_{x^-\sim p_x^-} e^{f(x)^T f(x^-)}}\right] \\
&\geq \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}}\left[-\log\frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + (K-1)\mathbb{E}_{x^-\sim p_x^-} e^{f(x)^T f(x^-)}}\right] \\
&= L^{K-1}_{\mathrm{Unbiased}}(f)
\end{aligned}$$

To show that $L^{K-1}_{\mathrm{Unbiased}}(f)$ is an upper bound on the supervised loss $L_{\mathrm{sup}}(f)$, we addi-

tionally introduce a task specific class distribution $\rho_{\mathcal{T}}$ which is a uniform distribution over the classes in task $\mathcal{T}$.

$$
\begin{aligned}
&L^{K-1}_{\text{Unbiased}}(f) \\
&= \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}}\left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + (K-1)\mathbb{E}_{x^-\sim p_x^-} e^{f(x)^T f(x^-)}}\right] \\
&= \mathbb{E}_{\mathcal{T}\sim\mathcal{D}}\mathbb{E}_{\substack{c\sim\rho_{\mathcal{T}};x\sim p(\cdot|c) \\ x^+\sim p(\cdot|c)}}\left[-\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + (K-1)\mathbb{E}_{\mathcal{T}\sim\mathcal{D}}\mathbb{E}_{\rho_{\mathcal{T}}(c^-\sim|c^-\neq h(x))}\mathbb{E}_{x^-\sim p(\cdot|c^-)} e^{f(x)^T f(x^-)}}\right] \\
&\geq \mathbb{E}_{\mathcal{T}\sim\mathcal{D}}\mathbb{E}_{c\sim\rho_{\mathcal{T}};x\sim p(\cdot|c)}\left[-\log \frac{e^{f(x)^T \mathbb{E}_{x^+\sim p(\cdot|c)} f(x^+)}}{e^{f(x)^T \mathbb{E}_{x^+\sim p^+_{x,\mathcal{T}}} f(x^+)} + (K-1)\mathbb{E}_{\mathcal{T}\sim\mathcal{D}}\mathbb{E}_{\rho_{\mathcal{T}}(c^-|c^-\neq h(x))}\mathbb{E}_{x^-\sim p(\cdot|c^-)} e^{f(x)^T f(x^-)}}\right] \\
&\geq \mathbb{E}_{\mathcal{T}\sim\mathcal{D}}\mathbb{E}_{c\sim\rho_{\mathcal{T}};x\sim p(\cdot|c)}\left[-\log \frac{e^{f(x)^T \mathbb{E}_{x^+\sim p(\cdot|c)} f(x^+)}}{e^{f(x)^T \mathbb{E}_{x^+\sim p(\cdot|c)} f(x^+)} + (K-1)\mathbb{E}_{\rho_{\mathcal{T}}(c^-|c^-\neq h(x))}\mathbb{E}_{x^-\sim p(\cdot|c^-)} e^{f(x)^T f(x^-)}}\right] \\
&= \mathbb{E}_{\mathcal{T}\sim\mathcal{D}}\mathbb{E}_{c\sim\rho_{\mathcal{T}};x\sim p(\cdot|c)}\left[-\log \frac{e^{f(x)^T \mathbb{E}_{x^+\sim p(\cdot|c)} f(x^+)}}{e^{f(x)^T \mathbb{E}_{x^+\sim p(\cdot|c)} f(x^+)} + (K-1)\mathbb{E}_{\rho_{\mathcal{T}}(c^-|c^-\neq h(x))}\mathbb{E}_{x^-\sim p(\cdot|c^-)} e^{f(x)^T f(x^-)}}\right] \\
&\geq \mathbb{E}_{\mathcal{T}\sim\mathcal{D}}\mathbb{E}_{c\sim\rho_{\mathcal{T}};x\sim p(\cdot|c)}\left[-\log \frac{e^{f(x)^T \mathbb{E}_{x^+\sim p(\cdot|c)} f(x^+)}}{e^{f(x)^T \mathbb{E}_{x^+\sim p(\cdot|c)} f(x^+)} + (K-1)\mathbb{E}_{\rho_{\mathcal{T}}(c^-|c^-\neq h(x))} e^{f(x)^T \mathbb{E}_{x^-\sim p(\cdot|c^-)} f(x^-)}}\right] \\
&= \mathbb{E}_{\mathcal{T}\sim\mathcal{D}}\mathbb{E}_{c\sim\rho_{\mathcal{T}};x\sim p(\cdot|c)}\left[-\log \frac{\exp(f(x)^T\mu_c)}{\exp(f(x)^T\mu_c) + \sum_{c^-\in\mathcal{T},c^-\neq c}\exp(f(x)^T\mu_{c^-})}\right] \\
&= \mathbb{E}_{\mathcal{T}\sim\mathcal{D}} L^{\mu}_{\text{Sup}}(\mathcal{T}, f) \\
&= \bar{L}^{\mu}_{\text{Sup}}(f)
\end{aligned}
$$

where the three inequalities follow from Jensen's inequality. The first and third inequality shift the expectations $\mathbb{E}_{x^+\sim p^+_{x,\mathcal{T}}}$ and $\mathbb{E}_{x^-\sim p(\cdot|c^-)}$, respectively, via the convexity of the functions and the second moves the expectation $\mathbb{E}_{\mathcal{T}\sim\mathcal{D}}$ out using concavity. Note that $\bar{L}_{\text{Sup}}(f) \leq \bar{L}^{\mu}_{\text{Sup}}(f)$ holds trivially. □

A.1.9 Proof of Theorem 5

We wish to derive a data dependent bound on the downstream supervised generalization error of the debiased contrastive objective. Recall that a sample $(x, x^+, \{u_i\}_{i=1}^N, \{v_i\}_{i=1}^M)$

yields loss

$$-\log\left\{\frac{e^{f(x)^\top f(x^+)}}{e^{f(x)^\top f(x^+)}+Ng(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)}\right\}=\log\left\{1+N\frac{g(x,\{u_i\}_{i=1}^N,\{v_i\}_{i=1}^M)}{e^{f(x)^\top f(x^+)}}\right\}$$

which is equal to $\ell\left(\left\{f(x)^\top\big(f(u_i)-f(x^+)\big)\right\}_{i=1}^N,\left\{f(x)^\top\big(f(v_i)-f(x^+)\big)\right\}_{i=1}^M\right)$, where we define

$$\ell(\{a_i\}_{i=1}^N,\{b_i\}_{i=1}^M)=\log\left\{1+N\max\left(\frac{1}{\tau^-}\frac{1}{N}\sum_{i=1}^N a_i-\tau^+\frac{1}{M}\sum_{i=1}^M b_i, e^{-1}\right)\right\}.$$

To derive our bound, we will exploit a concentration of measure result due to Arora et al. [2019]. They consider an objective of the form

$$L_{un}(f)=\mathbb{E}\left[\ell(\{f(x)^\top\big(f(x_i)-f(x^+)\big)\}_{i=1}^k)\right],$$

where $(x,x^+,x_1^-,\ldots,x_k^-)$ are sampled from any fixed distribution on $\mathcal{X}^{k+2}$ (they were particularly focused on the case where $x_i^-\sim p$, but the proof holds for arbitrary distributions). Let $\mathcal{F}$ be a class of representation functions $\mathcal{X}\to\mathbb{R}^d$ such that $\|f(\cdot)\|\le R$ for $R>0$. The corresponding empirical risk minimizer is

$$\hat{f}\in\arg\min_{f\in\mathcal{F}}\frac{1}{T}\sum_{j=1}^T\ell\left(\{f(x_j)^\top\big(f(x_{ji})-f(x^+)\big)\}_{i=1}^k\right)$$

over a training set $\mathcal{S}=\{(x_j,x_j^+,x_{j1}^-,\ldots,x_{jk}^-)\}_{j=1}^T$ of i.i.d. samples. Their result bounds the loss of the empirical risk minimizer as follows.

Lemma 5. *A.3* [Arora et al., 2019] *Let $\ell:\mathbb{R}^k\to\mathbb{R}$ be η-Lipschitz and bounded by B. Then with probability at least $1-\delta$ over the training set $\mathcal{S}=\{(x_j,x_j^+,x_{j1}^-,\ldots,x_{jk}^-)\}_{j=1}^T$, for all $f\in\mathcal{F}$*

$$L_{un}(\hat{f})\le L_{un}(f)+\mathcal{O}\left(\frac{\eta R\sqrt{k}\mathcal{R}_{\mathcal{S}}(\mathcal{F})}{T}+B\sqrt{\frac{\log\frac{1}{\delta}}{T}}\right)$$

where

$$\mathcal{R}_{\mathcal{S}}(\mathcal{F}) = \mathbb{E}_{\sigma\sim\{\pm 1\}^{(k+2)dT}}\left[\sup_{f\in\mathcal{F}}\langle\sigma, f_{|\mathcal{S}}\rangle\right],$$

and $f_{|\mathcal{S}} = \left(f_t(x_j), f_t(x_j^+), f_t(x_{j1}^-), \ldots, , f_t(x_{jk}^-)\right)_{\substack{j\in[T]\\ t\in[d]}}$.

In our context, we have $k = N + M$ and $R = e$. So, it remains to obtain constants η and B such that $\ell(\{a_i\}_{i=1}^N, \{b_i\}_{i=1}^M)$ is η-Lipschitz, and bounded by B. Note that since we consider normalized embeddings f, we have $\|f(\cdot)\| \leq 1$ and therefore only need to consider the domain where $e^{-1} \leq a_i, b_i \leq e$.

Lemma 6. *A.4 Suppose that* $e^{-1} \leq a_i, b_i \leq e$. *The function* $\ell(\{a_i\}_{i=1}^N, \{b_i\}_{i=1}^M)$ *is* η*-Lipschitz, and bounded by* B *for*

$$\eta = e \cdot \sqrt{\frac{1}{(\tau^-)^2 N} + \frac{(\tau^+)^2}{M}}, \qquad B = \mathcal{O}\left(\log N\left(\frac{1}{\tau^-} + \tau^+\right)\right).$$

Proof. First, it is easily observed that ℓ is upper bounded by plugging in $a_i = e$ and $b_i = e^{-1}$, yielding a bound of

$$\log\left\{1 + N\max\left(\frac{1}{\tau^-}e - \tau^+ e^{-1}, e^{-1}\right)\right\} = \mathcal{O}\left(\log N\left(\frac{1}{\tau^-} + \tau^+\right)\right).$$

To bound the Lipschitz constant we view ℓ as a composition $\ell(\{a_i\}_{i=1}^N, \{b_i\}_{i=1}^M) = \phi\left(g\left(\ell(\{a_i\}_{i=1}^N, \{b_i\}_{i=1}^M\right)\right)$ where[1],

$$\phi(z) = \log\left(1 + N\max(z, e^{-1}\right)$$
$$g(\{a_i\}_{i=1}^N, \{b_i\}_{i=1}^M) = \frac{1}{\tau^-}\frac{1}{N}\sum_{i=1}^N a_i - \tau^+\frac{1}{M}\sum_{i=1}^M b_i.$$

If $z < e^{-1}$ then $\partial_z\phi(z) = 0$, while if $z \geq e^{-1}$ then $\partial_z\phi(z) = \frac{N}{1+Nz} \leq \frac{N}{1+Ne^{-1}} \leq e$. We therefore conclude that ϕ is e-Lipschitz. Meanwhile, $\partial_{a_i} g = \frac{1}{\tau^- N}$ and $\partial_{b_i} g = \frac{\tau^+}{M}$. The Lipschitz constant of g is bounded by the Forbenius norm of the Jacobian of g,

[1]Note the definition of g is slightly modified in this context.

which equals

$$\sqrt{\sum_{i=1}^{N} \frac{1}{(\tau^- N)^2} + \sum_{j=1}^{M} \frac{(\tau^+)^2}{M^2}} = \sqrt{\frac{1}{(\tau^-)^2 N} + \frac{(\tau^+)^2}{M}}.$$

□

Now we have control on the bound on ℓ and its Lipschitz constant, we are ready to prove Theorem 5 by combining several of our previous results with Lemma A.3.

Theorem 2. *5 With probability at least $1-\delta$, for all $f \in \mathcal{F}$ and $N \geq K-1$,*

$$L_{\text{Sup}}(\hat{f}) \leq L^{\mu}_{\text{Sup}}(f) \leq L^{N,M}_{\text{Debiased}}(f) + \mathcal{O}\left(\frac{1}{\tau^-}\sqrt{\frac{1}{N}} + \frac{\tau^+}{\tau^-}\sqrt{\frac{1}{M}} + \frac{\lambda \mathcal{R}_{\mathcal{S}}(\mathcal{F})}{T} + B\sqrt{\frac{\log \frac{1}{\delta}}{T}}\right)$$

where $\lambda = \sqrt{\frac{1}{\tau^{-2}}(\frac{M}{N}+1) + \tau^{+2}(\frac{N}{M}+1)}$ *and* $B = \log N \left(\frac{1}{\tau^-} + \tau^+\right)$.

Proof. By Lemma 4 and Theorem 3 we have

$$L_{\text{sup}}(\hat{f}) \leq \widetilde{L}^{N}_{\text{Unbiased}}(\hat{f}) \leq L^{N,M}_{\text{Debiased}}(\hat{f}) + \frac{e^{3/2}}{\tau^-}\sqrt{\frac{\pi}{2N}} + \frac{e^{3/2}\tau^+}{\tau^-}\sqrt{\frac{\pi}{2M}}.$$

Combining Lemma A.3 and Lemma A.4, with probability at least $1-\delta$, for all $f \in \mathcal{F}$, we have

$$L^{N,M}_{\text{Debiased}}(\hat{f}) \leq L^{N,M}_{\text{Debiased}}(f) + \mathcal{O}\left(\frac{\lambda \mathcal{R}_{\mathcal{S}}(\mathcal{F})}{T} + B\sqrt{\frac{\log \frac{1}{\delta}}{T}}\right),$$

where $\lambda = \eta\sqrt{k} = \sqrt{\frac{1}{\tau^{-2}}(\frac{M}{N}+1) + \tau^{+2}(\frac{N}{M}+1)}$ and $B = \log N \left(\frac{1}{\tau^-} + \tau^+\right)$. □

A.1.10 Derivation of Equation 4

In Section 3.3.1, we mentioned that the obvious way to approximate the unbiased objective is to replace p^-_x with $p^-_x(x') = (p(x') - \tau^+ p^+_x(x'))/\tau^-$ and then use the empirical counterparts for p and p^+_x, and that this yields an objective that is a sum of

$N+1$ expectations. To give the derivation of this claim, let

$$\ell(x, x^+, \{x_i^-\}_{i=1}^N, f) = -\log \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \sum_{i=1}^N e^{f(x)^T f(x_i^-)}}.$$

We plug in the decomposition as follows:

$$\begin{aligned}
&\mathbb{E}_{\substack{x\sim p, x^+\sim p_x^+ \\ \{x_i^-\}_{i=1}^N \sim p_x^-}} \left[\ell(x, x^+, \{x_i^-\}_{i=1}^N, f)\right] \\
&= \int p(x) p_x^+(x^+) \prod_{i=1}^N p_x^-(x_i^-) \ell(x, x^+, \{x_i^-\}_{i=1}^N, f) \mathrm{d}x \mathrm{d}x^+ \prod_{i=1}^N \mathrm{d}x_i^- \\
&= \int p(x) p_x^+(x^+) \prod_{i=1}^N \frac{p(x_i^-) - \tau^+ p_x^+(x_i^-)}{\tau^-} \ell(x, x^+, \{x_i^-\}_{i=1}^N, f) \mathrm{d}x \mathrm{d}x^+ \prod_{i=1}^N \mathrm{d}x_i^- \\
&= \frac{1}{(\tau^-)^N} \int p(x) p_x^+(x^+) \prod_{i=1}^N \left(p(x_i^-) - \tau^+ p_x^+(x_i^-)\right) \ell(x, x^+, \{x_i^-\}_{i=1}^N, f) \mathrm{d}x \mathrm{d}x^+ \prod_{i=1}^N \mathrm{d}x_i^-.
\end{aligned}$$

By the Binomial Theorem, the product can be separated into $N+1$ groups corresponding to how many x_i^- are sampled from p.

$$\begin{aligned}
&(1) \qquad \prod_{i=1}^N p(x_i^-) \\
&(2) \qquad \binom{N}{1} (-\tau^+) p_x^+(x_1^-) \prod_{i=2}^N p(x_i^-) \\
&(3) \qquad \binom{N}{2} \prod_{j=1}^2 (-\tau^+) p_x^+(x_j^-) \prod_{i=3}^N p(x_i^-) \\
&\cdots \\
&(k+1) \qquad \binom{N}{k} \prod_{j=1}^k (-\tau^+) p_x^+(x_j^-) \prod_{i=k+1}^N p(x_i^-) \\
&\cdots \\
&(N+1) \qquad \prod_{i=1}^N (-\tau^+) p_x^+(x_i^-)
\end{aligned}$$

In particular, the objective becomes

$$\frac{1}{(\tau^-)^N}\sum_{k=0}^{N}\binom{N}{k}(-\tau^+)^k\mathbb{E}_{\substack{x\sim p, x^+\sim p_x^+\\ \{x_i^-\}_{i=1}^k\sim p_x^+\\ \{x_i^-\}_{i=k+1}^N\sim p}}\left[-\log\frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)}+\sum_{i=1}^N e^{f(x)^T f(x_i^-)}}\right],$$

where $\{x_i^-\}_{i=k}^j=\varnothing$ if $k>j$. Note that this is exactly the *Inclusion–exclusion principle*. The numerical value of this objective is extremely small when N is large. We tried various approaches to optimize this objective, but none of them worked.

A.2 Graph Representation Learning

We describe in detail the hard sampling method for graphs whose results are reported in Section 3.5.2. Before getting that point, in the interests of completeness we cover some required background details on the InfoGraph method of Sun et al. [2020]. For further information see the original paper [Sun et al., 2020].

A.2.1 Background on Graph Representations

We observe a set of graphs $\mathbf{G}=\{G_j\in\mathbb{G}\}_{j=1}^n$ sampled according to a distribution p over an ambient graph space $\mathbb{G}$. Each node u in a graph G is assumed to have features $h_u^{(0)}$ living in some Euclidean space. We consider a K-layer graph neural network, whose k-th layer iteratively computes updated embeddings for each node $v\in G$ in the following way,

$$h_v^{(k)}=\text{COMBINE}^{(k)}\left(h_v^{(k-1)},\text{AGGREGATE}^{(k)}\left(\left\{\left(h_v^{(k-1)},h_u^{(k-1)},e_{uv}\right):u\in\mathcal{N}(v)\right\}\right)\right)$$

where $\text{COMBINE}^{(k)}$ and $\text{AGGREGATE}^{(k)}$ are parameterized learnable functions and $\mathcal{N}(v)$ denotes the set of neighboring nodes of v. The K embeddings for a node u are collected together to obtain a single final summary embedding for u. As recommended by Xu et al. [2019] we use concatenation, $h^u=h^u(G)=\text{CONCAT}\left(\{h_u^{(k)}\}_{k=1}^K\right)$ to

obtain an embedding in $\mathbb{R}^d$. Finally, the node representations are combined together into a length d graph level embedding using a readout function,

$$H(G) = \text{READOUT}\left(\{h^u\}_{u \in G}\right)$$

which is typically taken to be a simple permutation invariant function such as the sum or mean. The InfoGraph method aims to maximize the mutual information between the graph level embedding $H(G)$ and patch-level embeddings $h^u(G)$ using the following objective,

$$\max_h \mathbb{E}_{G \sim p} \frac{1}{|G|} \sum_{u \in G} I\left(h^u(G); H(G)\right)$$

In practice the population distribution p is replaced by its empirical counterpart, and the mutual information I is replaced by a variational approximation I_T. In line with Sun et al. [2020] we use the Jensen-Shannon mutual information estimator as formulated by Nowozin et al. [2016]. It is defined using a neural network discriminator $T : \mathbb{R}^{2d} \to \mathbb{R}$ as,

$$I_T\left(h^u(G); H(G)\right) = \mathbb{E}_{G \sim p}\left[-\text{sp}(-T\left(h^u(G), H(G)\right))\right] - \mathbb{E}_{(G,G') \sim p \times p}\left[\text{sp}(T\left(h^u(G), H(G')\right))\right]$$

where $\text{sp}(z) = \log(1 + e^z)$ denotes the softplus function. The finial objective is the joint maximization over h and T,

$$\max_{\theta,\psi} \mathbb{E}_{G \sim p} \frac{1}{|G|} \sum_{u \in G} I_T\left(h^u(G); H(G)\right)$$

A.2.2 Hard Negative Sampling for Learning Graph Representations

In order to derive a simple modification of the NCE hard sampling technique that is appropriate for use with InfoGraph, we first provide a mildly generalized view of hard sampling. Recall that the NCE contrastive objective can be decomposed into

two constituent pieces,

$$\mathcal{L}(f, q) = \mathcal{L}_{\text{align}}(f) + \mathcal{L}_{\text{unif}}(f, q)$$

where q is in fact a family of distributions $q(x^-; x)$ over x^- that is indexed by the possible values of the anchor x. $\mathcal{L}_{\text{align}}$ performs the role of "aligning" positive pairs (embedding near to one-another), while $\mathcal{L}_{\text{unif}}$ repels negative pairs. The hard sampling framework aims to solve,

$$\inf_f \sup_q \mathcal{L}(f, q).$$

In the case of NCE loss we take,

$$\begin{aligned}
\mathcal{L}_{\text{align}}(f) &= -\mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}} f(x)^T f(x^+), \\
\mathcal{L}_{\text{unif}}(f, q) &= \mathbb{E}_{\substack{x\sim p \\ x^+\sim p_x^+}} \log \left\{ e^{f(x)^T f(x^+)} + Q\mathbb{E}_{x^-\sim q}[e^{f(x)^T f(x^-)}] \right\}.
\end{aligned}$$

View this view, we can easily adapt to the InfoGraph framework, taking

$$\begin{aligned}
\mathcal{L}_{\text{align}}(h, T) &= -\mathbb{E}_{G\sim p} \frac{1}{|G|} \sum_{u\in G} \text{sp}(-T\left(h^u(G), H(G)\right)), \\
\mathcal{L}_{\text{unif}}(h, T, q) &= -\mathbb{E}_{G\sim p} \frac{1}{|G|} \sum_{u\in G} \mathbb{E}_{G'\sim q} \text{sp}(T\left(h^u(G), H(G')\right))
\end{aligned}$$

Denote by $\hat{p}$ the distribution over nodes $u \in \mathbb{R}^s$ defined by first sampling $G \sim p$, then sampling $u \in G$ uniformly over all nodes of G. Then these two terms can be simplified to

$$\begin{aligned}
\mathcal{L}_{\text{align}}(h, T) &= -\mathbb{E}_{u\sim \hat{p}} \text{sp}(-T\left(h^u(G), H(G)\right)), \\
\mathcal{L}_{\text{unif}}(h, T, q) &= -\mathbb{E}_{(u,G')\sim \hat{p}\times q} \text{sp}(T\left(h^u(G), H(G')\right))
\end{aligned}$$

At this point it becomes clear that, just as with NCE, a distribution $q^* \in$

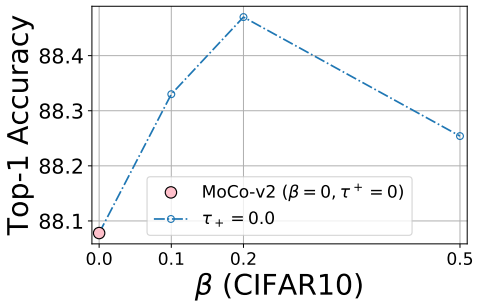


Figure A-1: Hard negative sampling using MoCo-v2 framework. Results show that hard negative samples can still be useful when the negative memory bank is very large (in this case $N = 65536$).

$\arg\max_q \mathcal{L}(f, q)$ in the InfoGraph framework if it is supported on $\arg\max_{G' \in \mathbb{G}} \text{sp}(T\left(h^u(G), H(G')\right))$. Although this is still hard to compute exactly, it can be approximated by,

$$q_u^\beta(G') \propto \exp\left(\beta T(h^u(G), H(G))\right) \cdot p(G').$$

A.3 Additional Experiments

A.3.1 Hard negatives with large batch sizes

The vision experiments in the main body of the paper are all based off the SimCLR framework [Chen et al., 2020b]. They use a relatively small batch size (up to 512). In order to test whether our hard negatives sampling method can help when the negative batch size is very large, we also run experiments using MoCo-v2 with standard negative memory bank size $N = 65536$ [He et al., 2020b, Chen et al., 2020e]. We adopt the official MoCo-v2 code[2]. Embeddings are trained for 200 epochs, with batch size 128. Figure A-1 summarizes the results. We find that hard negative sampling can still improve the generalization of embeddings trained on CIFAR10: MoCo-v2 attains linear readout accuracy of 88.08%, and MoCo-v2 with hard negatives ($\beta = 0.2$, $\tau^+ = 0$) attains 88.47%.

[2] https://github.com/facebookresearch/moco

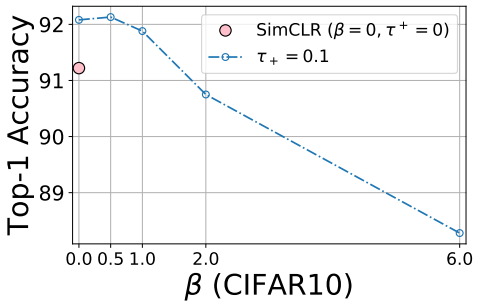


Figure A-2: The effect of varying concentration parameter β on linear readout accuracy for CIFAR10. (Complements the left and middle plot from Figure 3-6.)

A.3.2 Ablations

To study the affect of varying the concentration parameter β on the learned embeddings Figure 3-10 plots cosine similarity histograms of pairs of similar and dissimilar points. The results show that for β moving from 0 through 0.5 to 2 causes both the positive and negative similarities to gradually skew left. In terms of downstream classification, an important property is the *relative* difference in similarity between positive and negative pairs. In this case $\beta = 0.5$ find the best balance (since it achieves the highest downstream accuracy). When β is taken very large ($\beta = 6$), we see a change in conditions. Both positive and negative pairs are assigned higher similarities in general. Visually it seems that the positive and negative histograms for $\beta = 6$ overlap a lot more than for smaller values, which helps explain why the linear readout accuracy is lower for $\beta = 6$.

Figure 3-11 gives real examples of hard vs. uniformly sampled negatives. Given an anchor x (a monkey) and trained embedding f (trained on STL10 using standard SimCLR for 400 epochs), we sample a batch of 128 images. The top row shows the ten negatives x^- that have the largest inner product $f(x)^\top f(x^-)$, while the bottom row is a random sample from from the same batch. Negatives with the largest inner product with the anchor correspond to the items in the batch are the most important terms in the objective since they are given the highest weighting by q_β^-. Figure 3-11 shows that "real" hard negatives are conceptually similar to the idea as proposed in Figure 1: hard negatives are semantically similar to the anchor, possessing various

similarities, including color (browns and greens), texture (fur), and objects (animals vs machinery).

A.4 Experimental Details

A.4.1 Visual Representations

We implement SimCLR in PyTorch. We use a ResNet-50 [He et al., 2016] as the backbone with embedding dimension 2048 (the representation used for linear readout), and projection head into the lower 128-dimensional space (the embedding used in the contrastive objective). We use the Adam optimizer [Kingma and Ba, 2014] with learning rate 0.001 and weight decay 10^{-6}. Code available at `https://github.com/joshr17/HCL`. Since we adopt the SimCLR framework, the number of negative samples $N = 2(\text{batch size} - 1)$. Since we always take the batch size to be a power of 2 $(16, 32, 64, 128, 256)$ the negative batch sizes are $30, 62, 126, 254, 510$ respectively. Unless otherwise stated, all models are trained for 400 epochs.

Annealing β Method: We detail the annealing method whose results are given in Figure 3-6. The idea is to reduce the concentration parameter down to zero as training progresses. Specifically, suppose we have e number of total training epochs. We also specify a number ℓ of "changes" to the concentration parameter we shall make. We initialize the concentration parameter $\beta_1 = \beta$ (where this β is the number reported in Figure 3-6), then once every e/ℓ epochs we reduce β_i by β/ℓ. In other words, if we are currently on β_i, then $\beta_{i+1} = \beta_i - \beta/\ell$, and we switch from β_i to β_{i+1} in epoch number $i \cdot e/\ell$. The idea of this method is to select particularly difficult negative samples early on order to obtain useful gradient information early on, but later (once the embedding is already quite good) we reduce the "hardness" level so as to reduce the harmful effect of only approximately correcting for false negatives (negatives with the same labels as the anchor).

We also found the annealing in the opposite direction ("down") achieved similar performance.

Bias-variance of empirical estimates in hard-negative objective: Recall the final hard negative samples objective we derive is,

$$\mathop{\mathbb{E}}_{\substack{x\sim p\\ x^+\sim p_x^+}}\left[-\log\frac{e^{f(x)^Tf(x^+)}}{e^{f(x)^Tf(x^+)}+\frac{Q}{\tau^-}\left(\mathbb{E}_{x^-\sim q_\beta}\left[e^{f(x)^Tf(x^-)}\right]-\tau^+\mathbb{E}_{v\sim q_\beta^+}\left[e^{f(x)^Tf(v)}\right]\right)}\right]. \quad \text{(A.10)}$$

This objective admits a practical counterpart by using empirical approximations to $\mathbb{E}_{x^-\sim q_\beta}[e^{f(x)^Tf(x^-)}]$ and $\mathbb{E}_{v\sim q_\beta^+}[e^{f(x)^Tf(v)}]$. In practice we use a fairly large number of samples (e.g. $N = 510$) to approximate the first expectation, and only $M = 1$ samples to approximate the second. Clearly in both cases the resulting estimator is unbiased. Further, since the first expectation is approximated using many samples, and the integrand is bounded, the resulting estimator is well concentrated (e.g. apply Hoeffding's inequality out-of-the-box). But what about the second expectation? This might seem uncontrolled since we use only one sample, however it turns out that the random variable $X = e^{f(x)^Tf(v)}$ where $x \sim p$ and $v \sim q_\beta^+$ has variance that is bounded by $\mathcal{L}_{\text{align}}(f)$.

Lemma 7. *Consider the random variable $X = e^{f(x)^Tf(v)}$ where $x \sim p$ and $v \sim q_\beta^+$. Then $Var(X) \leq \mathcal{O}\big(\mathcal{L}_{align}(f)\big)$.*

Recall that $\mathcal{L}_{\text{align}}(f) = \mathbb{E}_{x,x^+}\|f(x) - f(x^+)\|^2/2$ is termed *alignment*, and Wang and Isola [2020a] show that the contrastive objective jointly optimize *alignment* and *uniformity*. Lemma 7 therefore shows that as training evolves, the variance of the $X = e^{f(x)^Tf(v)}$ where $x \sim p$ and $v \sim q_\beta^+$ is bounded by a term that we expect to see becoming small, suggesting that using a single sample ($M = 1$) to approximate this expectation is not unreasonable. We cannot, however, say more than this since we have no guarantee that $\mathcal{L}_{\text{align}}(f)$ goes to zero.

Proof. Fix an x and recall that we are considering $q_\beta^+(\cdot) = q_\beta^+(\cdot; x)$. First let X' be an i.i.d. copy of X, and note that, conditioning on x, we have $2\text{Var}(X|x) = \text{Var}(X|x) + \text{Var}(X'|x) = \text{Var}(X - X'|x) \leq \mathbb{E}\big[(X - X')^2|x\big]$. Bounding this difference,

$$
\begin{aligned}
\mathbb{E}\big[(X - X')^2 | x\big] &= \mathbb{E}_{v,v' \sim q_\beta^+} \left(e^{f(x)^\top f(v)} - e^{f(x)^\top f(v')} \right)^2 \\
&\leq \mathbb{E}_{v,v' \sim q_\beta^+} \left(e^{1/t^2} \left[f(x)^\top f(v) - f(x)^\top f(v') \right] \right)^2 \\
&\leq e^{1/t^4} \mathbb{E}_{v,v' \sim q_\beta^+} \left(\left[\|f(x)\| \|f(v) - f(v')\| \right] \right)^2 \\
&= \frac{e^{1/t^4}}{t^2} \mathbb{E}_{v,v' \sim q_\beta^+} \|f(v) - f(v')\|^2 \\
&\leq \mathcal{O}\left(\mathbb{E}_{v,v' \sim p^+} \|f(v) - f(v')\|^2 \right)
\end{aligned}
$$

where the first inequality follows since f lies on the sphere of radius $1/t$, the second inequality by Cauchy–Schwarz, the third again since f lies on the sphere of radius $1/t$, and the fourth since q_β^+ is absolutely continuous with respect to p^+ with bounded ratio.

Since $p^+(x^+) = p(x^+|h(x))$ only depends on $c = h(x)$, rather than x itself, taking expectations over $x \sim p$ is equivalent to taking expectations over $c \sim \rho$. Further, $\rho(c)p(v|c)p(v'|c) = p(v)p(v'|c) = p(v)p_v^+(v')$. So $\mathbb{E}_{c \sim \rho} \mathbb{E}_{v,v' \sim p^+} \|f(v) - f(v')\|^2 = \mathbb{E}_{x,x^+} \|f(x) - f(x^+)\|^2 = 2\mathcal{L}_{\text{align}}(f)$, where $x \sim p$ and $x^+ \sim p_x^+$. Thus we obtain the lemma. □

A.4.2 Graph Representations

All datasets we benchmark on can be downloaded at `www.graphlearning.io` from the TUDataset repository of graph classification problems [Morris et al., 2020a]. Information on basic statistics of the datasets is included in Tables A.1 and A.2. For fair comparison to the original InfoGraph method, we adopt the official code, which can be found at `https://github.com/fanyun-sun/InfoGraph`. We modify only the `gan_losses.py` script, adding in our proposed hard sampling via reweighting. For simplicity we trained all models using the same set of hyperparameters: we used the

GIN architecture [Xu et al., 2019] with $K = 3$ layers and embedding dimension $d = 32$. Each model is trained for 200 epochs with batch size 128 using the Adam optimizer [Kingma and Ba, 2014]. with learning rate 0.001, and weight decay of 10^{-6}. Each embedding is evaluated using the average accuracy 10-fold cross-validation using an SVM as the classifier (in line with the approach taken by Morris et al. [2020a]). Each experiment is repeated from scratch 10 times, and the distribution of results from these 10 runs is plotted in Figure 3-5.

Since the graph embeddings are not constrained to lie on a hypersphere, for a batch we clip all the inner products to live in the interval $[-2, 2]$ while computing the reweighting. We found this to be important for stabilizing optimization.

Dataset	**DD**	**PTC**	**REDDIT-B**	**PROTEINS**
No. graphs	1178	344	2000	1113
No. classes	2	2	2	2
Avg. nodes	284.32	14.29	429.63	39.06
Avg. Edges	715.66	14.69	497.75	72.82

Table A.1: Basic statistics for graph datasets.

Dataset	**ENZYMES**	**MUTAG**	**IMDB-B**	**IMDB-M**
No. graphs	600	188	1000	1500
No. classes	6	2	2	3
Avg. nodes	32.63	17.93	19.77	13.00
Avg. Edges	62.14	19.79	96.53	65.94

Table A.2: Basic statistics for graph datasets.

A.4.3 Sentence Representations

We adopt the official quick-thoughts vectors experimental settings, which can be found at `https://github.com/lajanugen/S2V`. We keep all hyperparameters at the default values and change only the `s2v-model.py` script. Since the official BookCorpus dataset

Kiros et al. [2015] is not available, we use an unofficial version obtained using the following repository: https://github.com/soskek/bookcorpus. Since the sentence embeddings are also not constrained to lie on a hypersphere, we use the same clipping trick as for the graph embeddings.

After training on the BookCorpus dataset, we evaluate the embeddings on six different classification tasks: paraphrase identification (MSRP) [Dolan et al., 2004], question type classification (TREC) [Voorhees and Harman, 2002], opinion polarity (MPQA) [Wiebe et al., 2005], subjectivity classification (SUBJ) [Pang and Lee, 2004], product reviews (CR) [Hu and Liu, 2004], and sentiment of movie reviews (MR) [Pang and Lee, 2005].

Appendix B

Further Discussion of Shortcuts in Contrastive Learning

B.1 Computation of implicit feature modification updates

This section gives detailed derivations of two simple but key facts used in the development of IFM. The first result derives an analytic expression for the gradient of the InfoNCE loss with respect to positive sample in latent space, and the second result computes the gradient with respect to an arbitrary negative sample. The analysis is very simple, only requiring the use of elementary tools from calculus. Despite its simplicity, this result is very important, and forms the core of our approach. It is thanks to the analytic expressions for the gradients of the InfoNCE loss that we are able to implement our adversarial method *without introducing any memory or run-time overheads*. This is a key distinction from previous adversarial methods for contrastive learning, which introduce significant overheads (see Fig. 4-5).

Recall the statement of the lemma.

Lemma 8. *For any* $v, v^+, \{v_i^-\}_{i=1}^m \in \mathbb{R}^d$ *we have,*

$$\nabla_{v_j^-}\ell = \frac{e^{v^\top v_j^-}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} \cdot \frac{v}{\tau} \quad \textit{and} \quad \nabla_{v^+}\ell = \left(\frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} - 1\right) \cdot \frac{v}{\tau}.$$

In particular, $\nabla_{v_j^-}\ell \propto v$ *and* $\nabla_{v^+}\ell \propto -v$.

Proof. Both results follow from direct computation. First we compute $\nabla_{v_j^-}\ell(v, v^+, \{v_i^-\}_{i=1}^m)$. Indeed, for any $j \in \{1, 2, \ldots, m\}$ we have,

$$\begin{aligned}
\nabla_{v_j^-}\left\{ -\log \frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} \right\} &= \nabla_{v_j^-} \log \left\{ e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau} \right\} \\
&= \frac{\nabla_{v_j^-}\left\{ e^{v^\top v^+} + \sum_{i=1}^m e^{v^\top v_i^-/\tau} \right\}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} \\
&= \frac{e^{v^\top v_j^-/\tau} \cdot v/\tau}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}}
\end{aligned}$$

the quantity $\frac{e^{v^\top v_j^-/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} > 0$ is a strictly positive scalar, allowing us to conclude the derivative $\nabla_{v_j^-}\ell$ is proportional to v. We also compute $\nabla_{v^+}\ell(v, v^+, \{v_i^-\}_{i=1}^m)$ in a similar fashion,

$$\begin{aligned}
\nabla_{v^+}\left\{ -\log \frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} \right\} &= \nabla_{v^+}\left\{ -\log e^{v^\top v^+/\tau} \right\} + \nabla_{v^+} \log \left\{ e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau} \right\} \\
&= -\frac{v}{\tau} + \frac{\nabla_{v^+}\left\{ e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau} \right\}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} \\
&= -\frac{v}{\tau} + \frac{e^{v^\top v^+/\tau} \cdot v/\tau}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} \\
&= \left(\frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} - 1 \right) \cdot \frac{v}{\tau}.
\end{aligned}$$

Since $0 < \frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} < 1$ we conclude in this case that the derivative $\nabla_{v^+}\ell$ points in the direction $-v$. □

B.1.1 Alternative formulations of implicit feature modification

This section contemplates two simple modifications to the IFM method with the aim of confirming that these modifications do not yield superior performance to the default proposed method. The two alternate methods focus around the following observation: IFM perturbs embeddings of unit length, and returns a modified version that will no longer be of unit length in general. We consider two alternative variations of IFM that yield normalized embeddings. The first is the most simple solution possible: simply re-normalize perturbed embeddings to have unit length. The second is slightly more involved, and involves instead applying perturbations *before* normalizing the embeddings. Perturbing unnormalized embeddings, then normalizing, guarantees the final embeddings have unit length. The key property we observed in the original formulation was the existence of an analytic, easily computable closed form expressions for the derivatives. This property enables efficient computation of newly synthesized "adversarial" samples in latent space. Here we derive corresponding formulae for the pre-normalization attack.

For clarity, we introduce the slightly modified setting in full detail. We are given positive pair x, x^+ and a batch of negative samples $\{x_i^-\}_{i=1}^m$ and denote their encodings via f as $v = f(x), v^+ = f(x^+)$, and $v_i^- = f(x_i^-)$ for $i = 1, \dots m$ where we *do not* assume that f returns normalized vectors. That is, f is allowed to map to anywhere in the ambient latent space $\mathbb{R}^d$. The re-parameterized point-wise contrastive loss for this batch of samples is

$$\ell(v, v^+, \{v_i^-\}_{i=1}^m) = -\log \frac{e^{\mathrm{sim}(v,v^+)/\tau}}{e^{\mathrm{sim}(v,v^+)/\tau} + \sum_{i=1}^m e^{\mathrm{sim}(v,v_i^-)/\tau}},$$

where $\mathrm{sim}(u, v) = u \cdot v / \|u\| \|v\|$ denotes the cosine similarity measure. As before we wish to perturb v^+ and negative encodings v_j^- to increase the loss, thereby making the negatives harder. Specifically we wish to solve $\max_{\delta^+ \in \mathcal{B}_{\varepsilon^+}, \{\delta_i^- \in \mathcal{B}_{\varepsilon_i}\}_{i=1}^m} \ell(v, v^+ + \delta^+, \{v_i^- + \delta_i^-\}_{i=1}^m)$. The following lemma provides the corresponding gradient directions.

Lemma 9. *For any $v, v^+, \{v_i^-\}_{i=1}^m \in \mathbb{R}^d$ we have*

$$\nabla_{v_j^-}\ell \propto \frac{v}{\|v\|} - sim(v_j^-, v)\frac{v_j^-}{\|v_j^-\|} \quad and \quad \nabla_{v^+}\ell \propto \frac{v}{\|v\|} - sim(v^+, v)\frac{v^+}{\|v^+\|}.$$

To prove this lemma we rely on the following well-known closed form expression for the derivative of the cosine similarity, whose proof we omit.

Lemma 10. $\nabla_v sim(v, u) = \frac{u}{\|v\|\|u\|} - sim(v, u)\frac{v}{\|v\|^2}$.

Proof of Lemma 9. We compute,

$$\begin{aligned}\nabla_{v_j^-}\ell &= \nabla_{v_j^-}\log\left(e^{\mathrm{sim}(v,v^+)} + \sum_{i=1}^m e^{\mathrm{sim}(v,v_i^-)}\right)\\ &= \frac{e^{\mathrm{sim}(v,v_j^-)}}{e^{\mathrm{sim}(v,v^+)} + \sum_{i=1}^m e^{\mathrm{sim}(v,v_i^-)}} \cdot \nabla_{v_j^-}\mathrm{sim}(v, v_j^-)\end{aligned}$$

Using the formula for the derivative of the cosine similarity, we arrive at a closed form formula,

$$\begin{aligned}\nabla_{v_j^-}\ell &= \frac{e^{\mathrm{sim}(v,v_j^-)}}{e^{\mathrm{sim}(v,v^+)} + \sum_{i=1}^m e^{\mathrm{sim}(v,v_i^-)}} \cdot \left(\frac{v}{\|v_j^-\|\|v\|} - \mathrm{sim}(v_j^-, v)\frac{v_j^-}{\|v_j^-\|^2})\right).\\ &\propto \frac{v}{\|v\|} - \mathrm{sim}(v_j^-, v)\frac{v_j^-}{\|v_j^-\|}\end{aligned}$$

Similar computations yield

$$\begin{aligned}\nabla_{v^+}\ell &= -\nabla_{v^+}\log\frac{e^{\mathrm{sim}(v,v^+)}}{e^{\mathrm{sim}(v,v^+)} + \sum_{i=1}^m e^{\mathrm{sim}(v,v_j^-)}}\\ &= \nabla_{v^+}\left(-\mathrm{sim}(v, v^+) + \log\left(e^{\mathrm{sim}(v,v^+)} + \sum_{i=1}^m e^{\mathrm{sim}(v,v_i^-)}\right)\right)\\ &= \left(\frac{e^{\mathrm{sim}(v,v^+)}}{e^{\mathrm{sim}(v,v^+)} + \sum_{i=1}^m e^{\mathrm{sim}(v,v_i^-)}} - 1\right) \cdot \nabla_{v^+}\mathrm{sim}(v, v^+)\\ &= \left(\frac{e^{\mathrm{sim}(v,v^+)}}{e^{\mathrm{sim}(v,v^+)} + \sum_{i=1}^m e^{\mathrm{sim}(v,v_i^-)}} - 1\right) \cdot \left(\frac{v}{\|v^+\|\|v\|} - \mathrm{sim}(v^+, v)\frac{v^+}{\|v^+\|^2})\right)\\ &\propto \frac{v}{\|v\|} - \mathrm{sim}(v^+, v)\frac{v^+}{\|v^+\|}\end{aligned}$$

□

Lemma 9 provides precisely the efficiently computable formulae for the derivatives we seek. One important difference between this pre-normalization case and the original setting is that the direction vector depends on v_j^- and v^+ respectively. In the original (unnormalized) setting the derivatives depend only on v, which allowed the immediate and exact discovery of the worst case perturbations in an ε-ball. Due to these additional dependencies in the pre-normalized case the optimization is more complex, and must be approximated iteratively. Although only approximate, it is still computationally cheap since we have simple analytic expressions for gradients.

It is possible give an interpretation to the pre-normalization derivatives $\nabla_{v_j^-}\ell$ by considering the ℓ_2 norm,

$$\begin{aligned}\|\nabla_{v_j^-}\|_2 &= \sqrt{\left(\frac{v}{\|v\|} - \frac{v^\top v_j^-}{\|v\|\|v_j^-\|}\frac{v_j^-}{\|v_j^-\|}\right)\cdot\left(\frac{v}{\|v\|} - \frac{v^\top v_j^-}{\|v\|\|v_j^-\|}\frac{v_j^-}{\|v_j^-\|}\right)} \\ &= \sqrt{1+\mathrm{sim}(v,v_i^-)^2 - 2\mathrm{sim}(v,v_i^-)^2} \\ &= \sqrt{1-\mathrm{sim}(v,v_i^-)^2}\end{aligned}$$

So, samples v_i^- with higher cosine similarity with anchor v receive smaller updates. Similar calculations for v^+ show that higher cosine similarity with anchor v leads to larger updates. In other words, the pre-normalization version of the method automatically adopts an adaptive step size based on sample importance.

Experimental results using alternative formulations

In this section we test the two alternative implementations to confirm that these simple alternatives do not obtain superior performance to IFM. We consider only object-based images, so it remains possible that other modalities may benefit from alternate formulations. First note that f encodes all points to the boundary of the same hypersphere, while perturbing $v_i^- \leftarrow v_i^- + \varepsilon_i v$ and $v^+ \leftarrow v^+ - \varepsilon_+ v$ moves adversarial samples off this hypersphere. We therefore consider normalizing all points

Dataset	MoCo-v2	IFM-MoCo-v2		
–	–	default	+ norm	+ pre-norm
STL10	92.4%	92.9%	92.9%	**93.0%**
CIFAR10	91.8%	**92.4%**	92.2%	92.0%
CIFAR100	69.0%	**70.3%**	70.1%	70.2%

Table B.1: Linear readout performance of alternative latent space adversarial methods. We report the best performance over runs for $\varepsilon \in \{0.05, 0.1, 0.2, 0.5\}$. We find that the two modifications to IFM we considered do not improve performance compared to the default version of IFM.

again *after* perturbing (*+ norm*). The second method considers applying attacks *before* normalization (*+ pre-norm*), whose gradients were computed in the Lem. 9. It is still possible to compute analytic gradient expressions in this setting; we refer the reader to Appendix B.1.1 for full details and derivations. Results reported in Tab. B.1, suggest that all versions improve over MoCov2, and both alternatives perform comparably to the default implementation based on Eqn. 4.2.

B.2 Supplementary experimental results and details

B.2.1 Hardware and setup

Experiments were run on two internal servers. The first consists of 8 NVIDIA GeForce RTX 2080 Ti GPUs (11GB). The second consists of 8 NVIDIA Tesla V100 GPUs (32GB). All experiments use the PyTorch deep learning framework Paszke et al. [2019]. Specific references to pre-existing code bases used are given in the relevant sections below.

B.2.2 Feature suppression experiments

This section gives experimental details for all experiments in Sec. 4.2 in the main manuscript, the section studying the relation between feature suppression and instance discrimination.

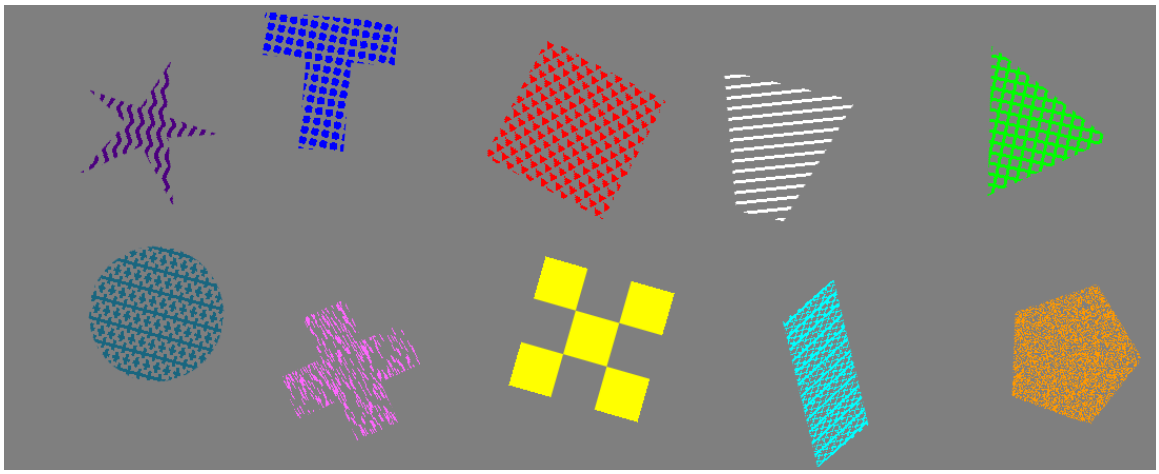


Figure B-1: Sample images from the Trifeature dataset Hermann and Lampinen [2020]. There are three features: shape, color, and texture. Each feature has 10 different possible values. We show exactly one example of each feature.

Figure B-2: Sample images from the STL-digits dataset. There are two features: object class, and MNIST digit. Both features have 10 different possible values.

Datasets

Trifeature Hermann and Lampinen [2020] Introduced by Hermann and Lampinen, each image is 128×128 and has three features: color, shape, and texture each taking 10 values. For each (color, shape, texture) triplet (1000 in total) Trifeature contains 100 examples, forming a dataset of 100K examples in total. Train/val sets are obtained by a random 90/10 split. See Fig. B-1, Appdx. B.2 for sample images.

STL10-digits dataset We artificially combine MNIST digits and STL10 object to produce data with two controllable semantic features. We split the STL10 image into a 3×3 grid, placing a copy of the MNIST digit in the center of each sector. This is done by masking all MNIST pixels with intensity lower than 100, and updating non-masked pixels in the STL10 image with the corresponding MNIST pixel value.

Experimental protocols

Training We train ResNet-18 encoders using SimCLR with batch size 512. We use standard data SimCLR augmentations Chen et al. [2020b], but remove grayscaling and color jittering when training on Trifeature in order to avoid corrupting color features. We use Adam optimizer, learning rate 1×10^{-3} and weight decay 1×10^{-6}. Unless stated otherwise, the temperature τ is set to 0.5.

Linear evaluation For fast linear evaluation we first extract features from the trained encoder (applying the same augmentations to inputs as used during pre-training) then use the `LogisticRegression` function in scikit-learn Pedregosa et al. [2011] to train a linear classifier. We use the Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm with a maximum iteration of 500 for training.

Details on results

Correlations Fig. 4-2 For the Trifeature heatmap 33 encoders are used to compute correlations. The encoders are precisely encoders used to plot Fig. 4-3. Similarly, the 7 encoders used to generate the STL-digits heatmap are precisely the encoders whose training is shown in Fig. B-4. When computing the InfoNCE loss for Fig. 4-2, for fair comparison all losses are computed using temperature normalization value $\tau = 0.5$. This is independent of training, and is necessary only in evaluation to ensure loss values are comparable across different temperatures.

Fig. B-4 displays results for varying instance discrimination difficult on the STL-digits dataset. These results are complementing the Trifeature results in Fig. 4-3 in Sec. 4.2 in the main manuscript. For STL-digits we report only a single training run per hyperparameter setting since performance is much more stable on STL-digits compared to Trifeature (see Fig. B-3). See Sec. 4.2 for discussion of STL-digits results, which are qualitatively the same as on Trifeature. Finally, Fig. B-5 shows the effect of IFM on encoders trained on STL-digits. As with Trifeature, we find that IFM improves the performance on suppressed features (STL10), but only slightly. Unlike hard instance discrimination methods, IFM does not harm MNIST performance in

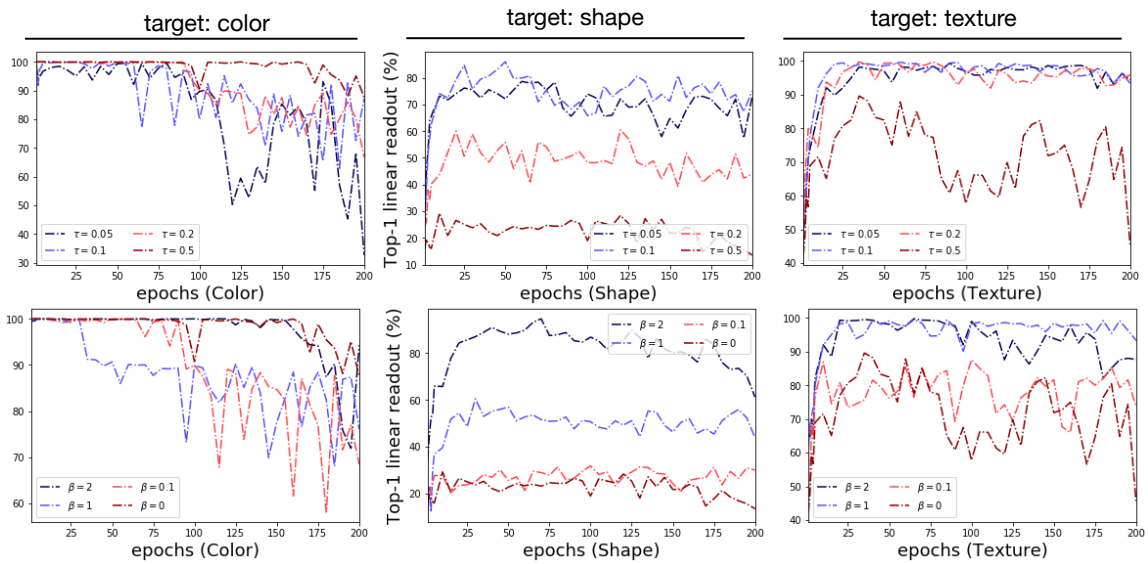


Figure B-3: Single run experiments showing training dynamics of Trifeature contrastive training. Linear readout performance on color prediction is particularly noisy.

the process.

B.2.3 Comparing IFM and ACL(DS)

We give details for Fig 4-5. Similarly to concurrent work Ho and Nvasconcelos [2020], Kim et al. [2020], ACL Jiang et al. [2020] directly performs PGD attacks in input space. We compare to the top performing version ACL(DS) – which uses a duel stream structure and combines standard and adversarial loss terms. We use the official ACL implementation[1] and for fair comparison run IFM by changing only the loss function. All hyperparameters are kept the same for both runs, and follow the ACL recommendations.

Training We use the SimCLR framework with a ResNet-18 backbone and train for 1000 epochs. We use a base learning rate of 5 with cosine annealing scheduling and batch size 512. LARS optimizer is used. For ACL(DS), we run the PGD for 5 steps in the pre-training stage following the practice of Jiang et al. [2020].

[1] https://github.com/VITA-Group/Adversarial-Contrastive-Learning

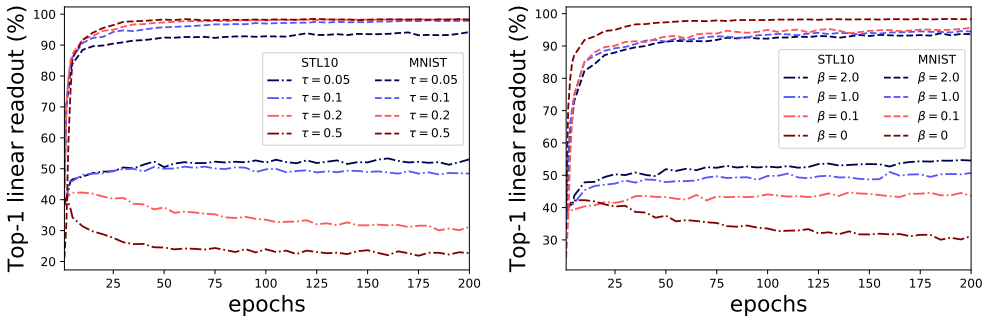


Figure B-4: STL-digits dataset. **Left:** performance on STL10 and MNIST linear readout for different temperature τ values. **Right:** performance on STL10 and MNIST linear readout for different hardness concentration β values Robinson et al. [2021a]. In both cases harder instance discrimination (smaller τ, bigger β) improves STL10 performance at the expense of MNIST. When instance discrimination is too easy (big τ, small β) STL10 features are *suppressed*: achieving worse linear readout after training than at initialization.

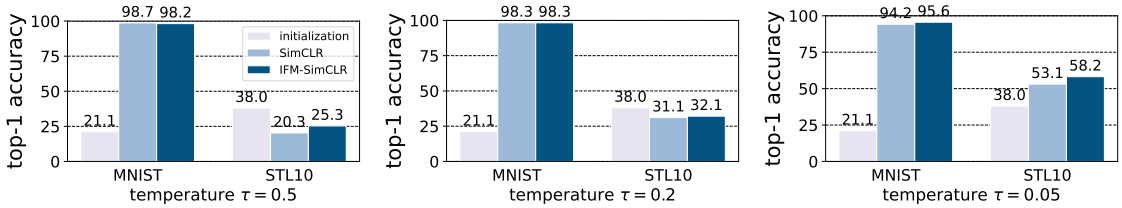


Figure B-5: STL-digits dataset. Implicit feature modification reduces feature suppression, enhancing the representation of both MNIST and STL10 features simultaneously. All IFM runs use a fixed value $\varepsilon = 0.1$, and loss $\mathcal{L} + 0.5 \cdot \mathcal{L}_\varepsilon$ (i.e. weighting parameter $\alpha = 0.5$) to illustrate robustness to the choice of parameters.

Linear evaluation We use two schemes to evaluate the quality of learnt representation: standard accuracy and robust accuracy. Robust accuracy reports the accuracy in the setting where an adversary is allowed to apply an ℓ_∞ attack to each input. For standard accuracy, we only finetune the last layer and test on clean images following the practice of MoCo-v2 Chen et al. [2020f]. The initial learning rate is set as 0.1 and we tune for 100 epochs for CIFAR10, 25 epochs for CIFAR100 respectively. An SGD optimizer is used to finetune the model. We use a step scheduler that decreases the learning rate by a factor of 10 after epochs: $40, 60$ for CIFAR10; $15, 20$ for CIFAR100 respectively. For robust accuracy, we finetune the model using the loss in TRADE Zhang et al. [2019], and evaluate classification accuracy on adversarially perturbed testing images. We use the same hyperparameters as ACL Jiang et al. [2020] for adversarial finetuning. We perform experiments on CIFAR10 and CIFAR100

and the results are shown in Fig. 4-5.

Results See Fig. 4-5 in the main manuscript for the results. There are significant qualitative differences between the behaviour of IFM and ACL(DS). IFM improves (standard) linear readout accuracy with zero memory or compute time cost increase, whereas ACL(DS) has improved adversarial linear readout performance, but at the cost of worse standard linear readout and 2× memory and 6× time per epoch. This shows that these two method are addressing two distinct problems. ACL(DS) is suitable for improving the adversarial robustness of a model, whereas IFM improves the generalization of a representation.

B.2.4 Object classification experiments

We first describe the protocol used for evaluating IFM on the following datasets: CIFAR10, CIFAR100, STL10, tinyImageNet. For simplicity, the objective weighting parameter is fixed at $\alpha = 1$. For MoCo-v2, we performed 5-fold cross validation for CIFAR10/CIFAR100 datasets, and 3 replicated runs on official train/val data splits for tinyImageNet and STL10 datasets.

Training All encoders have ResNet-50 backbones and are trained for 400 epochs with temperature $\tau = 0.5$ for SimCLR and $\tau = 0.1$ for MoCo-v2. Encoded features have dimension 2048 and are followed by a two layer MLP projection head with output dimension 128. Batch size is taken to be 256, yielding negative batches of size $m = 510$ for SimCLR. For MoCo-v2, we use a queue size of $k = 4096$ (except for STL10 dataset we use $k = 8192$), and we use batch size of 256 for CIFAR10, CIFAR100 and tinyImageNet, 128 for STL10. For both SimCLR and MoCo-v2 we use the Adam optimizer.

SimCLR uses initial learning rate 1×10^{-3} and weight decay 1×10^{-6} for CIFAR10, CIFAR100 and tinyImageNet, while STL10 uses 1×10^{-1} learning rate, and weight decay 5×10^{-4} (since we found these settings boosted performance by around 5% in absolute terms). MoCo-v2 training uses weight decay 5×10^{-4}, and an initial learning

rate 3×10^{-2} for CIFAR10 and CIFAR100; and learning rate 1×10^{-1} for STL10 and tinyImageNet. Cosine learning rate schedule is used for MoCo-v2.

Linear evaluation Evaluation uses test performance of a linear classifier trained ontop of the learned embedding (with embedding model parameters kept fixed) trained for 100 epochs.

For SimCLR, the batch size is set as 512, and the linear classifier is trained using the Adam optimizer with learning rate 1×10^{-3} and weight decay 1×10^{-6}, and default PyTorch settings for other hyperparameters. For CIFAR10 and CIFAR100 the same augmentations as SimCLR are used for linear classifier training, while for STL10 and tinyImageNet no augmentations were used (since we found this improves performance).

For MoCo-v2, the batch size is set as 256. Training uses SGD with initial learning rate set to 30, momentum is set as 0.9 and a scheduler that reduces the learning rate by a factor of 10% at epoch 30 and 60. The weight decay is 0. For CIFAR10 and CIFAR100, we normalize images with mean of $[0.4914, 0.4822, 0.4465]$ and standard deviation of $[0.2023, 0.1994, 0.2010]$. For STL10 and tinyImageNet, we normalize images with mean of $[0.485, 0.456, 0.406]$ and standard deviation of $[0.229, 0.224, 0.225]$. The same augmentations as the official MoCo-v2 implementation are used for linear classifier training.

ImageNet100

We adopt the official MoCo-v2 code[2] (CC-BY-NC 4.0 license), modifying only the loss function. For comparison with AdCo method, we adopt the official code[3] (MIT license) and use the exact same hyperparmeters as for MoCo-v2. For the AdCo specific parameters we perform a simple grid search for the following two hyperparameters: negatives learning rate lr_{neg} and negatives temperature τ_{neg}. We search over all combinations $lr_{\text{neg}} \in \{1, 2, 3, 4\}$ and $\tau_{\text{neg}} \in \{0.02, 0.1\}$, which includes the AdCo

[2] https://github.com/facebookresearch/moco
[3] https://github.com/maple-research-lab/AdCo

default ImageNet1K recommendations $lr_{\text{neg}} = 3$ and $\tau_{\text{neg}} = 0.02$ Hu et al. [2020a]. The result reported for AdCo in Tab. 4.1 is the best performance over all 8 runs.

Training We use ResNet-50 backbones, and train for 200 epochs. We use a base learning rate of 0.8 with cosine annealing scheduling and batch size 512. The MoCo momentum is set to 0.99, and temperature to $\tau = 0.2$. All other hyperparameters are kept the same as the official defaults.

Linear evaluation We train for 60 epochs with batch size 128. We use initial learning rate of 30.0 and a step scheduler that decreases the learning rate by a factor of 10 after epochs: $30, 40, 50$. All other hyperparameters are kept the same as the official MoCo-v2 defaults.

As noted in the manuscript, our combination of training and linear evaluation parameters leads to 80.5% top-1 linear readout for standard MoCo-v2, and 81.4% with IFM-MoCo-v2. The standard MoCo-v2 performance of 80.5% is, to the best of our knowledge, state-of-the-art performance on ImageNet100 using 200 epoch training with MoCo-v2. For comparison, we found that using the default recommended MoCo-v2 ImageNet1k parameters (both training and linear evaluation) achieves ImageNet100 performance of 71.8%. This choice of parameters maybe useful for other researchers using MoCo-v2 as a baseline on ImageNet100.

B.2.5 COPDGene dataset

The dataset Regan et al. [2011] in our experiments includes 9,180 subjects. Each subject has a high-resolution inspiratory CT scan and five COPD related outcomes, including two continuous spirometry measures: (1) FEV1pp: the forced expiratory volume in one second, (2) FEV_1/FVC: the FEV1pp and forced vital capacity (FVC) ratio, and three ordinal variables: (1) six-grade centrilobular emphysema (CLE) visual score, (2) three-grade paraseptal emphysema (Para-septal) visual score, (3) five-grade dyspnea symptom (mMRC) scale. The dataset is publicly available.

For fair comparison, we use the same encoder and data augmentation described in the baseline approach Sun et al. [2021]. We set the representation dimension to 128 in

all experiments. For simplicity, instead of using a GNN, we use average pooling to aggregate patch representations into image representation. The learning rate is set as 0.01. We use Adam optimizer and set momentum as 0.9 and weight decay as 1×10^{-4}. The batch size is set as 128, and the model is trained for 10 epochs.

B.2.6 Further discussion of feature robustness experiments (Sec. 4.4.3)

Ilyas et al. Ilyas et al. [2019] showed that deep networks richly represent so-called "non-robust" features, but that adversarial training can be used to avoid extracting non-robust features at a modest cost to downstream performance. Although in-distribution performance is harmed, Ilyas et al. argue that the reduction in use of non-robust features – which are highly likely to be statistical coincidences due to the high dimensionality of input data in computer vision – may be desirable from the point of view of trustworthiness of a model under input distribution shifts. In this section we consider similar questions on the effect implicit feature modification on learning of robust vs. non-robust features during self-supervised pre-training.

Compared to supervised adversarial training Ilyas et al. [2019], Madry et al. [2018] our approach has the key conceptual difference of being applied in feature space. As well as improved computation efficiency (no PGD attacks required) Fig. 4-8 shows that this difference translates into different behavior when using implicit feature modification. Instead of suppressing non-robust features as Ilyas et al. observe for supervised representations, IFM *enhances the representation of robust features*. This suggests that the improved generalization of encoders trained with IFM can be attributed to improved extraction of features aligned with human semantics (robust features). However, we also note that IFM has no significant effect on learning of non-robust features.

Appendix C

Further Discussion of Contrastive Learning with Rotational Equivariance

C.1 Proofs of Theoretical Results

The aim of this section is to detail the proofs of the theoretical results presented in the main manuscript. The key theoretical tools driving our analysis are prepared separately in Section C.2.

Throughout our analysis, we assume that all spaces (e.g., $\mathcal{A}$ and $\mathcal{X}$) are subspaces of Euclidean space and therefore admit a Lebesgue measure. We also assume that all distributions (e.g., $a \sim \mathcal{A}$ and $x \sim \mathcal{X}$) admit a density with respect to the Lebesgue measure. With these conditions in mind, we recall the loss function that is the main object of study:

$$\mathcal{L}_{\text{equi}}(f) = \mathbb{E}_{a\sim\mathcal{A}}\mathbb{E}_{x,x'\sim\mathcal{X}}\left[f(a(x'))^\top f(a(x)) - f(x)^\top f(x')\right]^2 \tag{C.1}$$

Next, we re-state and prove Proposition 5, our first key result.

Proposition 5. *Suppose $\mathcal{L}_{equi}(f) = 0$. Then for almost every $a \in \mathcal{A}$, there is an orthogonal matrix $R_a \in O(d)$ such that $f(a(x)) = R_a f(x)$ for almost all $x \in \mathcal{X}$.*

Proof. Suppose that $\mathcal{L}_{\text{equi}}(f) = 0$. This means that $f(a(x'))^\top f(a(x)) = f(x)^\top f(x')$ for almost all $a \in G$, and $x, x' \in \mathcal{X}$. Setting $g_a(x) = f(a(x))$, we have that $g_a(x')^\top g_a(x) =$

$f(x)^\top f(x')$. The continuous version of the First Fundamental Theorem of invariant theory for the orthogonal group (see Proposition 15) implies that there is an $R_a \in O(d)$ such that $f(a(x)) = g_a(x) = R_a f(x)$. □

As discussed in greater detail in the main manuscript, these results show that minimizing $\mathcal{L}_{\text{equi}}$ produces a model where an augmentation a corresponds to a single orthogonal transformation of embeddings R_a, independent of the input. This result is continuous in flavor as it studies the loss over the full data distribution $p(x)$. There exists a corresponding result for the finite sample loss

$$\mathcal{L}_{\text{equi},n}(f) = \mathbb{E}_{a\sim\mathcal{A}} \sum_{i,j=1}^{n} \left[f(a(x_j))^\top f(a(x_i)) - f(x_i)^\top f(x_j)\right]^2.$$

Proposition 14. *Suppose $\mathcal{L}_{equi,n}(f) = 0$. Then for almost every $a \in \mathcal{A}$, there is an orthogonal matrix $R_a \in O(d)$ such that $f(a(x_i)) = R_a f(x_i)$ for all $i = 1, \ldots, n$.*

As for the population counterpart, the proof of this result directly follows from the application of the First Fundamental Theorem of invariant theory for the orthogonal group.

Proof of Proposition 14. Suppose that $\mathcal{L}_{\text{equi}}(f) = 0$. This means that for almost every $a \in G$, and every $i, j = 1, \ldots, n$ we have $f(a(x_j))^\top f(a(x_i)) = f(x_i)^\top f(x_j)$. In other words $AA^T = BB^T$ where $A, B \in \mathbb{R}^{n\times d}$ are matrices whose ith rows are $A_i = f(a(x_i))^\top$ and $B_i = f(x_i)^\top$ respectively. This implies, by the First Fundamental Theorem of invariant theory for the orthogonal group (see Corollary 3), that there is an $R_a \in O(d)$ such that $A = BR_a$. Considering only the ith rows of A and B leads us to conclude that $f(a(x_i)) = R_a f(x_i)$. □

A corollary of Proposition 5 is that compositions of augmentations correspond to compositions of rotations.

Corollary 1. *If $\mathcal{L}_{equi}(f) = 0$, then $\rho : \mathcal{A} \to O(d)$ given by $\rho(a) = R_a$ satisfies $\rho(a' \circ a) = \rho(a')\rho(a)$ for almost all a, a'. That is, ρ defines a group action on $\mathbb{S}^{d-1}$ up to a set of measure zero.*

Proof. Applying Proposition 5 on $a' \circ a$ as the sampled augmentation, we have that $f(a' \circ a(x_i)) = R_{a' \circ a} f(x_i) = \rho(a' \circ a) f(x_i)$. However, taking $\bar{x} = a(x_i)$ and applying Proposition 5 twice we also know that $f(a' \circ a(x_i)) = f(a'(\bar{x})) = R_a f(\bar{x}) = R_{a'} f(a(x_i)) = R_{a'} R_a f(x) = \rho(a')\rho(a) f(x_i)$. That is, $\rho(a' \circ a) f(x_i) = f(a' \circ a(x_i)) = \rho(a')\rho(a) f(x_i)$. Since this holds for all i, we have that $\rho(a' \circ a) = \rho(a')\rho(a)$. □

This corollary requires us to assume that $\mathcal{A}$ is a semi-group. That is, $\mathcal{A}$ is closed under compositions, but group elements do not necessarily have inverses and it does not need to include an identity element.

C.2 Background on Invariance Theory for the Orthogonal Group

This section recalls some classical theory on orthogonal groups and an extension that we use for proving results over continuous data distributions.

A function $f : (\mathbb{R}^d)^n \to \mathbb{R}$ is said to be $O(d)$-invariant if $f(Rv_1, \ldots, Rv_n) = f(v_1, \ldots, v_n)$ for all $R \in O(d)$. Throughout this section, we are especially interested in determining easily computed statistics that *characterize* an $O(d)$ invariant function f. In other words, we would like to write f as a function of these statistics. The following theorem was first proved by Hermann Weyl using Capelli's identity [Weyl, 1946] and shows that the inner products $v_i^\top v_j$ suffice.

Theorem 2 (First fundamental theorem of invariant theory for the orthogonal group)**.** *Suppose that $f : (\mathbb{R}^d)^n \to \mathbb{R}$ is $O(d)$-invariant. Then there exists a function $g : \mathbb{R}^{n \times n} \to \mathbb{R}$ for which*

$$f(v_1, \ldots, v_n) = g\big([v_i^\top v_j]_{i,j=1}^n\big).$$

In other words, to compute f at a given input, it is not necessary to know all of $v_1, \ldots, v_n$. Computing the value of f at a point can be done using only the inner products $v_i^\top v_j$, which are invariant to $O(d)$. Letting V be the $n \times d$ matrix whose ith

row is $v_i^\top$, we may also write $f(v_1, \ldots, n_n) = g(VV^\top)$. The map $V \mapsto VV^\top$ is known as the orthogonal projection of V.

A corollary of this result has recently been used to develop $O(d)$ equivariant architectures in machine learning [Villar et al., 2021].

Corollary 3. *Suppose that A, B are $n \times d$ matrices and $AA^\top = BB^\top$. Then $A = BR$ for some $R \in O(d)$.*

Villar et al. [2021] use this characterization of orthogonally equivariant functions to *parameterize* function classes of neural networks that have the same equivariance. This result is also useful in our context; However, we put it to use for a very different purpose: studying $\mathcal{L}_{\text{equi}}$.

Intuitively this result says the following: given two point clouds A, B of unit length vectors with some fixed correspondence (bijection) between each point in A and a point in B, if the *angles* between the ith and jth points in cloud A always equal the angle between the ith and jth point in cloud B, then A and B are the same up to an orthogonal transformation.

This is the main tool we use to prove the finite sample version of the main result for our equivariant loss (Proposition 14). However, to analyze the population sample loss $\mathcal{L}_{\text{equi}}$ (Proposition 5), we require an extended version of this result to the continuous limit as $n \to \infty$. To this end, we develop a simple but novel extension to Theorem 2 to the case of continuous data distributions. This result may be useful in other contexts independent of our setting.

Proposition 15. *Let $\mathcal{X}$ be any set and $f, h : \mathcal{X} \to \mathbb{R}^d$ be functions on $\mathcal{X}$. If $f(x)^\top f(y) = h(x)^\top h(y)$ for all $x, y \in \mathcal{X}$, then there exists $R \in O(d)$ such that $Rf(x) = h(x)$ for all $x \in \mathcal{X}$.*

The proof of this result directly builds on the finite sample version. The key idea of the proof is that since the embedding space $\mathbb{R}^d$ is finite-dimensional we may select a set of points $\{f(x_i)\}_i$ whose span has maximal rank in the linear space spanned by the outputs of f. This means that any arbitrary point $f(x)$ can be written as a linear combination of the $f(x_i)$. This observation allows us to apply the finite sample result

on each $f(x_i)$ term in the sum to conclude that $f(x)$ is also a rotation of a sum of $h(x_i)$ terms. Next, we give the formal proof.

Proof of Proposition 15. Choose $x_1, \ldots, x_n \in \mathcal{X}$ such that $F = [f(x_1) \mid \ldots \mid f(x_n)]^\top \in \mathbb{R}^{n \times d}$ and $h = [h(x_1) \mid \ldots \mid h(x_n)]^\top \in \mathbb{R}^{n \times d}$ have maximal rank. Note we use "|" to denote the column-wise concatenation of vectors. Note that such x_i can always be chosen. Since we have $FF^\top = HH^\top$, we know by Corollary 3 that $F = HR$ for some $R \in O(d)$.

Now consider an arbitrary $x \in \mathcal{X}$ and define $\tilde{F} = [F \mid f(x)]^\top$ and $\tilde{H} = [H \mid h(x)]^\top$, both of which belong to $\mathbb{R}^{(n+1) \times d}$. Note that again we have $\tilde{F}\tilde{F}^\top = \tilde{H}\tilde{H}^\top$ so also know that $\tilde{F} = \tilde{H}\tilde{R}$ for some $\tilde{R} \in O(d)$. Since x_i were chosen so that F and H are of maximal rank, we know that $h(x) = \sum_{i=1}^n c_i h(x_i)$ for some coefficients $c_i \in \mathbb{R}$, since if this were not the case then we would have $\operatorname{rank}(\tilde{H}) = \operatorname{rank}(H) + 1$.

From this, we know that

$$\begin{aligned} R^\top h(x) &= \sum_{i=1}^n c_i R^\top h(x_i) \\ &= \sum_{i=1}^n c_i f(x_i) \\ &= \sum_{i=1}^n c_i \tilde{R}^\top h(x_i) \\ &= \tilde{R}^\top \sum_{i=1}^n c_i h(x_i) \\ &= \tilde{R}^\top h(x) \\ &= f(x). \end{aligned}$$

So we have that $Rf(x) = RR^\top h(x) = h(x)$ for all $x \in \mathcal{X}$. □

C.3 Extensions to Other Groups: Further Discussion

In Section 5.3.2, we explore the possibility of formulating an equivariant loss $\mathcal{L}_{\text{equi}}$ for pairs of points that fully captures equivariance by requiring the group to be the stabilizer of a bilinear form. In this context, the invariants are generated by polynomials of degree two in two variables, and the equivariant functions can be obtained by computing gradients of these invariants [Blum-Smith and Villar, 2022]. Section 5.3.2 notes that this holds true not only for the orthogonal group, which is the primary focus of our research but also for the Lorentz group and the symplectic group, suggesting natural extensions of our approach.

It is worth noting that the group of rotations $SO(d)$ does not fall into this framework. It can be defined as the set of transformations that preserve both inner products (a 2-form) and determinants (a d-form). Consequently, some of its generators have degree 2 while others have degree d (see [Weyl, 1946], Section II.A.9).

Weyl's theorem states that if a group acts on n copies of a vector space (in our case, $(\mathbb{R}^d)^n$ for consistency with the rest of the paper), its action can be characterized by examining how it acts on k copies (i.e., $(\mathbb{R}^d)^k$) when the maximum degree of its irreducible components is k (refer to Section 6 of [Schmid, 2006] for a precise statement of the theorem). Since our interest lies in understanding equivariance in terms of pairs of objects, we desire invariants that act on pairs of points. One way to guarantee this is to restrict ourselves to groups that act through representations where the irreducible components have degrees of at most two (though this is not necessary in all cases, such as the orthogonal group $O(d)$ that we consider in the main paper). An example of such groups is the product of finite subgroups of the unitary group $U(2)$, which holds relevance in particle physics. According to Weyl's theorem, the corresponding invariants can be expressed as *polarizations* of degree-2 polynomials on two variables. Polarizations represent an algebraic construction that enables the expression of homogeneous polynomials in multiple variables by introducing additional variables to polynomials with fewer variables. In our case, the base polynomials consist of degree-2 polynomials in two variables, while the polarizations incorporate additional

variables. Notably, an interesting open problem lies in leveraging this formulation for contrastive learning.

C.3.1 Experimental Protocols

We first outline the training protocol adopted for training our proposed approach on a variety of datasets, namely CIFAR10, CIFAR100, STL10, and ImageNet100.

CIFAR10, CIFAR100 and STL10 All encoders have ResNet-50 backbones and are trained for 400 epochs with temperature $\tau = 0.5$ for SimCLR and $\tau = 0.1$ for MoCo-v2 [1]. The encoded features have a dimension of 2048 and are further processed by a two-layer MLP projection head, producing an output dimension of 128. A batch size of 256 was used for all datasets. For CIFAR10 and CIFAR100, we employed the Adam optimizer with a learning rate of $1e^{-3}$ and weight decay of $1e^{-6}$. For STL10, we employed the SGD optimizer with a learning rate of 0.06, utilizing cosine annealing and a weight decay of $5e^{-4}$, with 10 warmup steps. We use the same set of augmentations as in SimCLR [Chen et al., 2020c]. To train the encoder using $\mathcal{L}_{\text{CARE-SimCLR}}$, we use the same hyper-parameters for InfoNCE loss. Additionally, we use 4, 8 and 16 batch splits for CIFAR100, STL10 and CIFAR10, respectively. This allows us to sample multiple augmentations per batch, effectively reducing the batch size of equivariance loss whilst retaining the same for InfoNCE loss. Furthermore, for the equivariant term, we find it optimal to use a weight of $\lambda = 0.01, 0.001$, and 0.01 for CIFAR10, CIFAR100, and STL10, respectively.

ImageNet100 We use ResNet-50 as the encoder architecture and pretrain the model for 200 epochs. A base learning rate of 0.8 is used in combination with cosine annealing scheduling and a batch size of 512. For MoCo-v2, we use 0.99 as the momentum and $\tau = 0.2$ as the temperature. All remaining hyperparameters were maintained at their respective official defaults as in the official MoCo-v2 code. While training with $\mathcal{L}_{\text{CARE-SimCLR}}$ and $\mathcal{L}_{\text{CARE-MoCo}}$, we find it optimal to use splits of 4 and 8 and weight of $\lambda = 0.005$ and 0.01 respectively on the equivariant term.

[1] https://github.com/facebookresearch/moco

Linear evaluation We train a linear classifier on frozen features for 100 epochs with a batch size of 512 for CIFAR10, CIFAR100, and STL10 datasets. To optimize the classifier, we employ the Adam optimizer with a learning rate of $1e^{-3}$ and a weight decay of $1e^{-6}$. In the case of ImageNet100, we train the linear classifier for 60 epochs using a batch size of 128. We initialize the learning rate to 30.0 and apply a step scheduler with an annealing rate of 0.1 at epochs 30, 40, and 50. The remaining hyper-parameters are retained from the official code.

C.4 Additional experiments

Histogram for loss ablation. To accompany Figure 5-2, this section plots the cosine similarity between positive pairs. We provide two plots for each experiment: the first plots the *histogram* of similarities of positive pairs drawn from the test set; the second plots the *average* positive cosine similarity throughout training. The results are reported in Figures C-1, C-2, C-3, C-4, C-5, C-6.

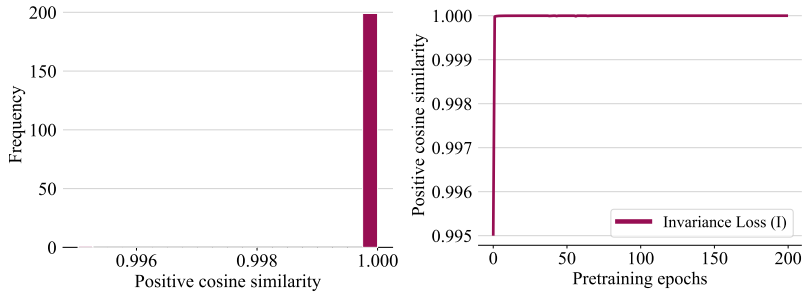


Figure C-1: (left) Histogram of positive cosine similarity values at the end of pre-training using the invariance loss; (right) Evolution of positive cosine similarity values over pre-training epochs using the invariance loss

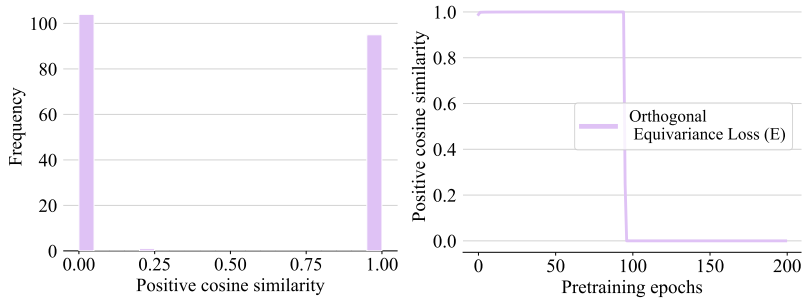


Figure C-2: (left) Histogram of positive cosine similarity values at the end of pre-training using the orthogonal equivariance loss; (right) Evolution of positive cosine similarity values over pre-training epochs using the orthogonal equivariance loss

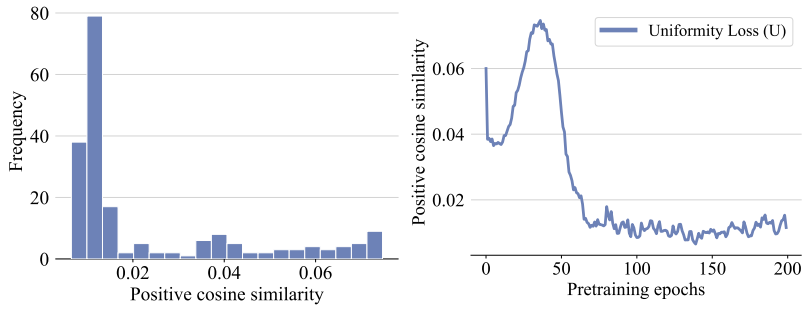


Figure C-3: (left) Histogram of positive cosine similarity values at the end of pre-training using the uniformity loss; (right) Evolution of positive cosine similarity values over pre-training epochs using the uniformity loss

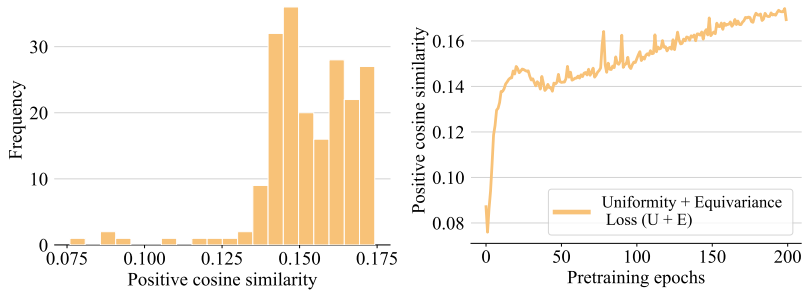


Figure C-4: (left) Histogram of positive cosine similarity values at the end of pre-training using the Uniformity + Equivariance loss; (right) Evolution of positive cosine similarity values over pre-training epochs using the Uniformity + Equivariance loss

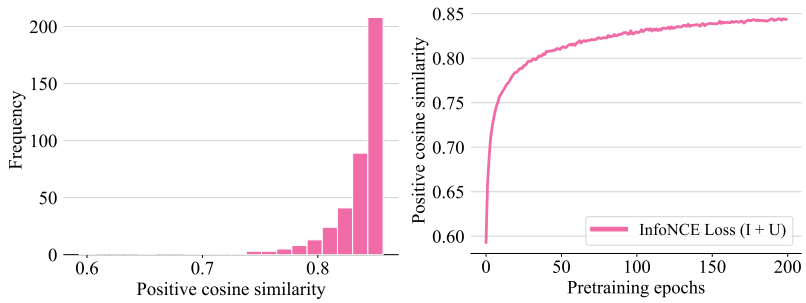


Figure C-5: (left) Histogram of positive cosine similarity values at the end of pre-training using the InfoNCE (invariance + uniformity) loss; (right) Evolution of positive cosine similarity values over pre-training epochs using the InfoNCE loss

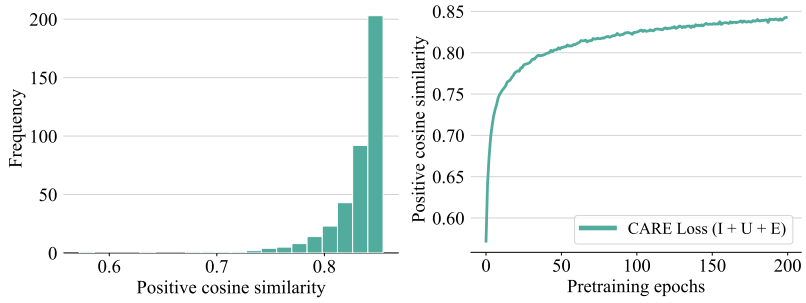


Figure C-6: (left) Histogram of positive cosine similarity values at the end of pre-training using the CARE (InfoNCE + orthogonal equivariance) loss; (right) Evolution of positive cosine similarity values over pre-training epochs using the CARE loss

Appendix D

Further Discussion for Contrastive Masked Auotencoders

D.1 Additional Transfer Learning Results

We report additional results for few-shot learning and robustness.

Robustness: Section 6.3.5 reports robustness results for ViT-L models pre-trained on JFT-300M for 5000 epochs, and ViT-L models pre-trained on IN-1K for 800 epochs. In both cases we report the performance of the models after finetuning on IN-1K.

Here we report the same robustness results for ViT-L models trained on JFT-300M for 1600 and 800 epochs (Figure D-1), and ViT-B models pre-trained for 800 epochs (Figure D-2). Figure D-2 also compares our ViT-B model to ViT-B models trained from scratch on ImageNet. We find that our model is considerably more robust than training with cross-entropy and Mixup from scratch, and also outperforms PyramidAT [Herrmann et al., 2022], an adversarial training method that introduces significant overheads compared to standard cross-entropy training. We emphasize that here there are two differences in the training: a) the training algorithm itself, and b) the data seen by the model. Our model sees extra JFT-300M data not seen by the other two approaches. This means that the methods are not exactly comparable. It is, however, a realistic setting showing the benefits to robustness of pre-training on large datasets.

Few shot: Section 6.3.4 reports 10- and 25-shot results for ViT-L models pre-

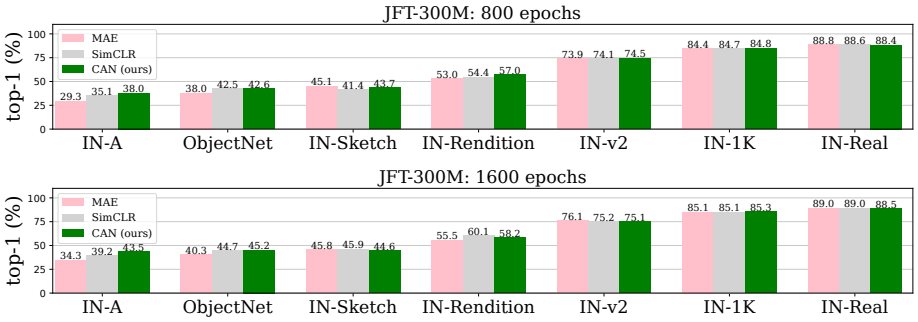


Figure D-1: ViT-L models pre-trained on JFT-300M for 800 and 1600 epochs respectively, evaluated on 7 datasets with distribution shifts from IN-1K.

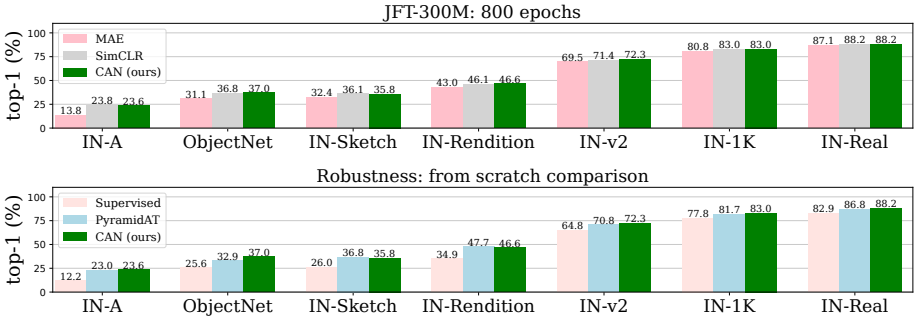


Figure D-2: **Top:** ViT-B models pre-trained on JFT-300M for 800 epochs, evaluated on 7 datasets with distribution shifts from IN-1K. **Bottom:** Comparison of our JFT-300M pre-trained ViT-B model to training ViT-B from scratch on IN-1K. We compare to standard supervised cross-entropy training with Mixup, and to PyramidAT [Herrmann et al., 2022], which uses an adversarial training method. CAN considerably outperforms supervised training, and beats PyramidAT in 6 out of 7 cases without requiring adversarial training.

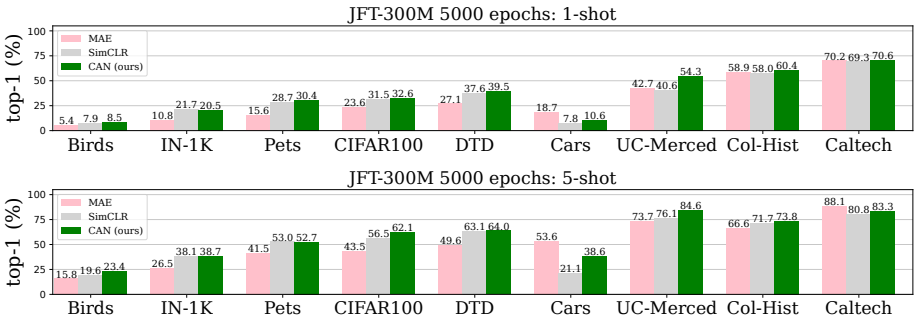


Figure D-3: **Few shot:** ViT-L models pre-trained on JFT-300M for 5000 epochs evaluated on 9 few-shot learning tasks. Results accompany the 10- and 25-shot results in Figure 6-4.

trained on JFT-300M for 5000 epochs. Here we report 1- and 5-shot results for the same models in Figure D-3. We additionally show the full set of $\{1, 5, 10, 25\}$-shot results for ViT-L models pre-trained on JFT-300M for 800 and 1600 epochs (Figures D-4 and D-5 respectively), ViT-B models pre-trained on JFT-300M for 800 epochs (Figure D-6), and ViT-L models pre-trained on IN-1K for 800 epochs (Figure D-7).

We make a number of observations.

1. JFT-300M pre-training often outperforms IN-1K pre-training. Comparing Figures D-4 and D-7, JFT-300M yields better 25-shot CAN performance on 6 out of 9 datasets.

2. Model scale helps. Comparing Figures D-6 and D-4, ViT-L models perform best in nearly all cases.

3. Across all settings CAN generally performs the best on JFT-300M pre-training.

4. The situation is less consistent on IN-1K pre-training. For instance, although MAE has comparatively poor few-shot performance on IN-1K, it is competitive on others for 25-shot evaluation: in this setting CAN only beats MAE on 5 out of 9 datasets. However, on 10-shot CAN outperforms MAE and SimCLR in 7 out of 9 cases, showing a subtle picture.

Figure D-4: **Few shot:** ViT-L models pre-trained on JFT-300M for 800 epochs are evaluated on 9 few-shot learning tasks.

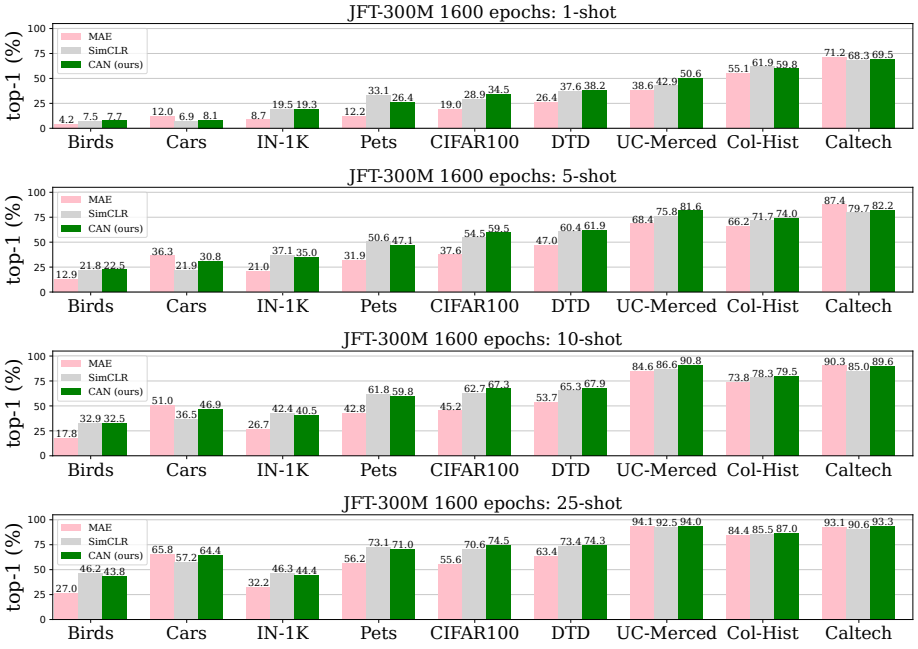


Figure D-5: **Few shot:** ViT-L models pre-trained on JFT-300M for 1600 epochs are evaluated on 9 few-shot learning tasks.

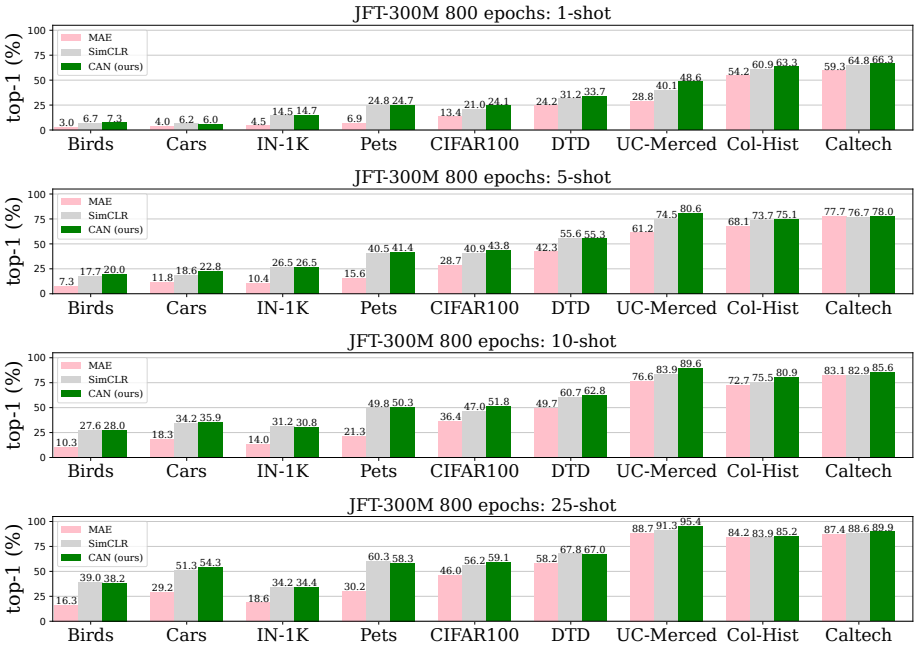


Figure D-6: **Few shot:** ViT-B models pre-trained on JFT-300M for 800 epochs are evaluated on 9 few-shot learning tasks.

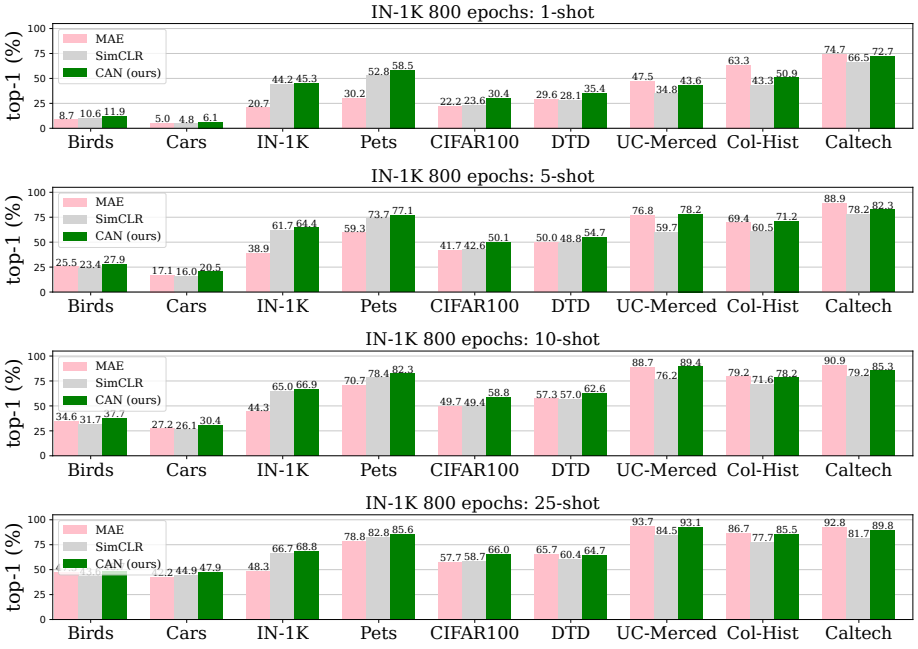


Figure D-7: **Few shot:** ViT-L models pre-trained on IN-1K for 800 epochs are evaluated on 9 few-shot learning tasks.

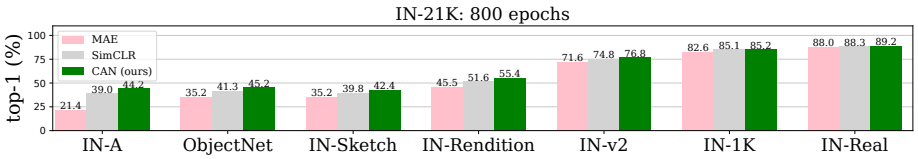


Figure D-8: **Robustness:** ViT-L models pre-trained on IN-21K for 800 (IN-1K equivalent) epochs are first finetuned on IN-1K. The models are then evaluated on 7 test datasets with different distribution shifts from IN-1K.

MAE	SimCLR	CAN
57.8	76.5	76.5

Table D.1: **IN-21K pre-training.** Linear probe performance on IN-1K. ViT-B models pre-trained for 800 IN-1K equivalent epochs.

D.1.1 ImageNet-21K pre-training.

We also consider the performance of CAN on pre-training on ImageNet-21K (IN-21K), a publicly available dataset of 14.2 million images, grouped into 21,000 different classes Deng et al. [2009]. We use the same hyperparamter settings as JFT-300M training to train ViT-L models on IN-21K for 800 (IN-1K equivalent) epochs.

We run a full set of evaluations on finetuning, linear probe, robustness (Figure D-8), and few-shot learning (Figure D-9), and linear probe in Table D.1. We note that while CAN and SimCLR achieve similar linear probe performance, CAN remains much more efficient (see Figure 6-1). Furthermore, we only ran IN-21K training for 800 epochs. This is a training schedule which also led to similar linear probe for CAN and SimCLR when pre-training on JFT-300M. The difference in linear probe between CAN and SimCLR emerged on JFT-300M with longer training runs. Despite this, CAN pre-training on IN-21K with 800 epoch schedule still performs favourably on other downstream robustness and few-show evaluations.

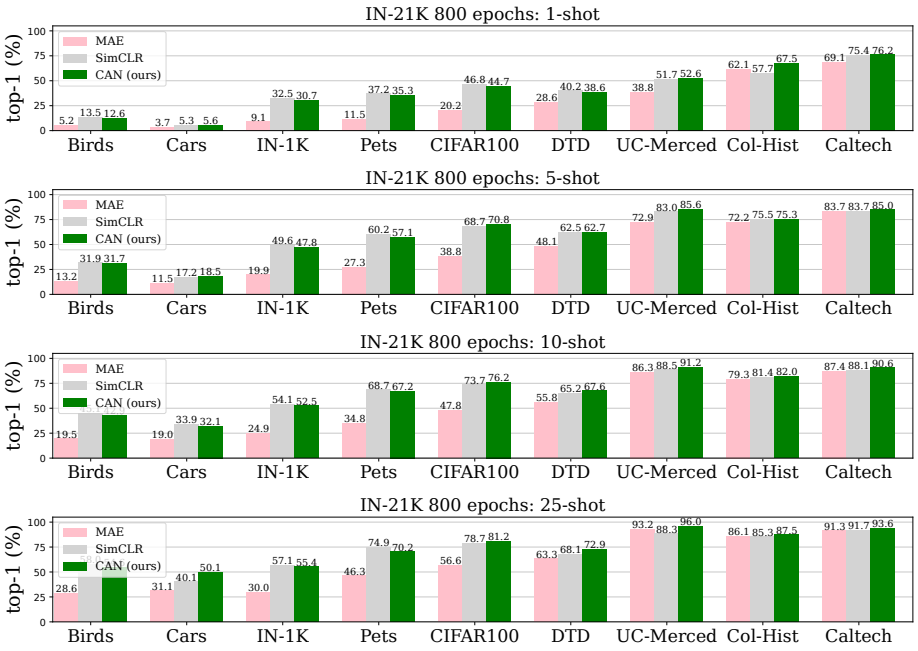


Figure D-9: **Few shot:** ViT-L models pre-trained on IN-21K for 800 (IN-1K equivalent) epochs are evaluated on 9 few-shot learning tasks.

D.2 Runtime of CAN compared to DnC

In the main paper we estimate our method is significantly faster than DnC [Tian et al., 2021]. We determined this approximate comparison from the following two pieces of information: 1) DnC reports that 3000 ImageNet epochs takes 29 hours on 512 TPUs for a ResNet-50 model (∼25M parameters), and 2) 3000 ImageNet epochs of CAN take 78 hours on 64 TPUs for a ViT-L model (∼300M parameters). We assume a linear relationship between number of TPUs and runtime. Under this assumption, we estimate that CAN would take approximately 10 hours to train with 512 TPUs, compared to the 29 hours reported by Tian et al. [2021] for a model with 1/10th the number of parameters. We emphasize that this is far from an exact comparison and is only intended as a very approximate guide.

D.3 Hyperparameter settings

We list hyperparameters used for CAN pre-training in Table D.2 and Table D.3. For preprocessing we closely follow SimCLR Chen et al. [2020c]. We use the same hyperparameters for SimCLR pre-training. For MAE pre-training, we use the same hyperparameters as listed in He et al. [2022], except for the use of Glorot uniform initialization instead of LeCun initialization as done in He et al. [2022]. We found that this provided better performance for our JAX-based MAE implementation. Table D.6 lists the hyperparameters for finetuning evaluations. We use the same set of hyperparameters for each finetuning each pre-training method, and for both ViT-B and ViT-L model sizes. For linear probing we list the hyperparameters in Table D.7 for which we followed the settings in He et al. [2022]. We use global average pool of the final representation instead of the cls token.

MAE longer training: MAE pre-training for longer training (5000 epochs) on JFT becomes unstable after about 500k steps (training loss oscillates); this results in poorer fine-tuning performance. To overcome this, we decrease the base learning rate by 75% as shown in Table D.5. However our model CAN is more stable and we use the same hyperparameters across different numbers of epochs.

Few shot training: For few-shot learning we use the same hyperparameters and pipeline as Dosovitskiy et al. [2021b]. We use the same pre-processing as was done in [Kolesnikov et al., 2020]. We use a base learning rate of 0.01 and train for 2500 steps, using an input resolution of 384×384.

Hardware details: We use TPU-v4 for all of our experiments. CAN on ViT-B uses 64 TPUs for a batch size of 4096. SimCLR, on the other hand, uses 128 TPUs for the same batch size, and is more compute intensive than CAN.

Decoder architecture: Our decoder architecture is the same as He et al. [2022]. We use standard ViT with a decoder depth of 8 and decoder width of 512. We use 16

heads and 2048 as the dimension of the MLP.

Projection head architecture: We use 2 hidden layers in our projection heads. Each layer has a Fully-Connected (FC) layer (dim 4096) followed by BatchNorm (momentum=0.9) followed by ReLU. After these 2 layers we have a FC layer which transforms the features to 128 dimensions. We apply contrastive learning on top of these 128 dimensional features.

Noise positional encoding architecture: We use a 1 hidden layer MLP with ReLU activations of hidden width and output dimension of 768, equal to the input dimensions of the decoder. During testing we tried different depths and widths, finding performance to be robust to these variations. The input dimension is 256 sinusoidal features, exactly as in Vaswani et al. [2017a].

JFT-300M and IN-21K hyperparameters: All hyperparameters were determined by training on IN-1K, and directly transferred to JFT-300M and IN-21K pre-training, with the exception of learning rate and weight decay, which found needed to be at a lower level for JFT-300M and IN-21K. For all methods we divided the learning rate by a factor of 4, and the weight decay by a factor of 2, except for MAE where we found that the original weight decay tuned on ImageNet worked better. We also tried dividing the learning rate by factors of 2 and 8, but found that 4 worked best for all methods. Specifically, for CAN and SimCLR we used following parameter choices: $wd = 0.1/2 = 0.05$ and $lr = 1.25 \times 10^{-4}/4 = 3.125 \times 10^{-5}$ and for MAE we used $lr = 1.5 \times 10^{-4}/4 = 3.75 \times 10^{-5}$, and tried $wd = 0.05/2 = 0.025$, but found that the original $wd = 0.05$ worked better, so kept this value.

D.3.1 Pre-training hyperparameters

Config	Value
optimizer	AdamW [Loshchilov and Hutter, 2017a]
base learning rate(ViT-B)	2.5e-4
base learning rate (ViT-L)	1.25e-4
weight decay (ViT-B)	0.05
weight decay (ViT-L)	0.1
optimizer momentum	β_1, β_2=0.9, 0.95 [Chen et al., 2020a]
batch size	4096
learning rate schedule	cosine decay [Loshchilov and Hutter, 2017b]
warmup epochs [Goyal et al., 2017]	40
augmentation	RandomResizedCrop, Color Jittering(strength=1.0), GrayScale(probability=0.2), Gaussian Blurring (probability=0.5)
masking rate	0.5

Table D.2: Hyperparameters for CAN pre-training on ImageNet. Note that we use lower learning rate for ViT-L as compared to ViT-B, following Steiner et al. [2021]. We use the same hyper-parameters for SimCLR pre-training, except for masking rate that is set to 0.

Config	Value
optimizer	AdamW [Loshchilov and Hutter, 2017a]
base learning rate (ViT-B)	2.5e-4
base learning rate (ViT-L)	3.125e-5
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.95 [Chen et al., 2020a]
batch size	4096
learning rate schedule	cosine decay [Loshchilov and Hutter, 2017b]
warmup epochs [Goyal et al., 2017]	40
augmentation	RandomResizedCrop, Color Jittering(strength=1.0), GrayScale(probability=0.2), Gaussian Blurring (probability=0.5)
masking rate	0.5

Table D.3: Hyperparameters for CAN pre-training on JFT-300M and IN-21K. Note that we use lower learning rate for ViT-L as compared to ViT-B, following Steiner et al. [2021]. We use the same hyper-parameters for SimCLR pre-training, except for masking rate that is set to 0.

D.3.2 Finetuning and linear probe hyperparameters

Table D.6 shows the finetuning recipie we use for CAN and SimCLR. This is the same as the recipe reported for MAE, except for the learning rate, which we found beneficial to lower, and the augmentations, which we found marginally beneficial to strengthen

Config	Value
optimizer	AdamW [Loshchilov and Hutter, 2017a]
base learning rate (ViT-L)	1.5e-4
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.95 [Chen et al., 2020a]
batch size	4096
learning rate schedule	cosine decay [Loshchilov and Hutter, 2017b]
warmup epochs [Goyal et al., 2017]	40
augmentation	RandomResizedCrop
masking rate	0.75

Table D.4: Hyperparameters for MAE pre-training on IN-1K. These parameters are exactly the same as in He et al. [2022].

Config	Value
optimizer	AdamW [Loshchilov and Hutter, 2017a]
base learning rate (ViT-L)	3.75e-5
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.95 [Chen et al., 2020a]
batch size	4096
learning rate schedule	cosine decay [Loshchilov and Hutter, 2017b]
warmup epochs [Goyal et al., 2017]	40
augmentation	RandomResizedCrop
masking rate	0.75

Table D.5: Hyperparameters for MAE pre-training on JFT-300M and IN-21K with ViT-L models. The only difference from the IN-1K configuration is the learning rate, which we reduced to stabilize training.

(He et al. [2022] uses only RandAug). The only other difference is that we always use layer-wise learning rate decay of 0.65, whereas MAE uses 0.65 for ViT-B, and 0.75 for ViT-L. For 5000 epochs of JFT-300M, and 800 epochs of IN-21K pre-training we noticed that CAN and SimCLR finetunne was over-fitting, so we reduced the finetuning from 100 epochs to 40 epochs. For all MAE finetuning runs we follow the recipe of the original paper He et al. [2022] as we found adjusting the learning rate and augmentations not to help for MAE. We did not observe any over-fitting in MAE, so kept the schedule fixed.

For linear probe training we follow the exact same recipe of He et al. [2022] for all models (see Table D.7 for details).

Config	Value
optimizer	AdamW [Loshchilov and Hutter, 2017a]
base learning rate	5e-4
weight decay	0.005
optimizer momentum	β_1, β_2=0.9, 0.999 [Chen et al., 2020a]
batch size	1024
learning rate schedule	cosine decay [Loshchilov and Hutter, 2017b]
warmup epochs [Goyal et al., 2017]	5
training epochs	100
label smoothing	0.1
drop path	0.1
layer-wise lr decay	0.65
augmentation	RandomResizedCrop, Flip, RandAug(layers=2, magnitude=9) [Cubuk et al., 2020], Random Erase [Zhong et al., 2020](probability=0.25)

Table D.6: Hyperparameters for finetuning CAN and SimCLR pre-trained models on ImageNet. We use the same hyperparameters for ViT-B and ViT-L, for JFT-300M, IN-21K and IN-1K pre-trainined models. For MAE, we use the settings from Table 9 of [He et al., 2022].

.

Config	Value
optimizer	LARS [You et al., 2017]
base learning rate	0.1
weight decay	0
optimizer momentum	0.9
batch size	16384
learning rate schedule	cosine decay [Loshchilov and Hutter, 2017b]
warmup epochs [Goyal et al., 2017]	10
training epochs	100
batch norm momentum	0.9
label smoothing	0
augmentation	RandomResizedCrop

Table D.7: Hyperparameters for linear probing pre-trained models on ImageNet. We use the same hyperparameters for ViT-B and ViT-L, and for JFT-300M, IN-21K and IN-1K pre-trainined models. Note that these hyperparamters are same as reported in He et al. [2022].

Appendix E

Further Discussion For Neural Networks for Eigenvector Data

E.1 Universality for Multiple Spaces

While the networks introduced in the Section 7.1.2 possess the desired invariances, it is not immediately obvious whether they are powerful enough to express *all* functions with these invariances. Under certain conditions, the universality of our architectures follows as a corollary of the following general decomposition result, which may enable construction of universal architectures for other invariances as well.

Theorem 10 (Decomposition Theorem)**.** *Let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ be topological spaces, and let G_i be a group acting on $\mathcal{X}_i$ for each i. We assume mild topological conditions on $\mathcal{X}_i$ and G_i hold. For any continuous $f : \mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \mathbb{R}^{d_{\text{out}}}$ that is invariant to the action of $G = G_1 \times \ldots \times G_k$, there exists continuous ϕ_i and a continuous $\rho : \mathcal{Z} \subseteq \mathbb{R}^a \to \mathbb{R}^{d_{\text{out}}}$ such that*

$$f(v_1, \ldots, v_k) = \rho(\phi_1(v_1), \ldots, \phi_k(v_k)). \tag{E.1}$$

Furthermore: (1) each ϕ_i can be taken to be invariant to G_i, (2) the domain $\mathcal{Z}$ of ρ is compact if each $\mathcal{X}_i$ is compact, (3) if $\mathcal{X}_i = \mathcal{X}_j$ and $G_i = G_j$, then ϕ_i can be taken to be equal to ϕ_j.

This result says that when a product of groups G acts on a product of spaces $\mathcal{X}$, for invariance to the product group G it suffices to individually process each smaller group G_i on $\mathcal{X}_i$ and then aggregate the results. Along with the proof of Theorem 10, the mild topological assumptions are explained in Appendix E.7.1. The assumptions hold for sign invariance and basis invariance, when not enforcing permutation equivariance. By applying this theorem, we can prove universality of some instances of our networks:

Corollary 4. *Unconstrained-SignNet can represent any sign invariant function and Unconstrained-BasisNet can represent any basis invariant function. Expressive-BasisNet is a universal approximator of functions that are both basis invariant and permutation equivariant.*

This result shows that Unconstrained-SignNet, Unconstrained-BasisNet, and Expressive-BasisNet take the correct functional form for their respective invariances (proofs in Appendix E.7.2). Note that Expressive-BasisNet approximates all sign invariant functions as a special case, by treating all inputs as one dimensional eigenspaces. Further, note that we require Expressive-BasisNet's high order tensors to achieve universality when enforcing permutation equivariance. Universality under permutation equivariance is generally difficult to achieve when dealing with matrices with permutation symmetries [Maron et al., 2019, Keriven and Peyré, 2019], but it may be possible that more efficient architectures can achieve it in our setting.

Accompanying the decomposition result, we show a corresponding universal approximation result (proof in Appendix E.7.3). Similarly to Theorem 10, the problem of approximating $G = G_1 \times \ldots \times G_k$ invariant functions is reduced to approximating several G_i-invariant functions.

E.2 More Details on SignNet and BasisNet

In Figure 7-2, we show a diagram that describes how SignNet is used as a node positional encoding for a graph machine learning task. In Table E.1, we compare and contrast properties of the neural architectures that we introduce. In Figure E-1,

Table E.1: Properties of our architectures: Unconstrained-SignNet, SignNet, Unconstrained-BasisNet, and Expressive-BasisNet. The properties are: permutation equivariance, universality (for the proper class of continuous invariant functions), and computational tractability.

	Unconstr.-SignNet	SignNet	Unconstr.-BasisNet	BasisNet	Expr.-BasisNet
Perm. equivariant	×	✓	×	✓	✓
Universal	✓	×	✓	×	✓
Tractable	✓	✓	✓	✓	×

PyTorch-like pseudo-code for SignNet

```
class SignNetGNN(nn.Module):

    def __init__(self, d, k, D1, D2, out_dim):
        self.phi = GIN(1, D1) # in dim=1, out dim=D1
        self.rho = MLP(k*D1, D2)
        self.base_model = GNN(d+D2, out_dim)

    def forward(self, g, x, eigvecs):
        # g contains graph information
        # x shape: n x d
        # eigvecs shape: n x k

        n, k = eigvecs.shape
        eigvecs = eigvecs.reshape(n, k, 1)
        pe = self.phi(g, eigvecs) + self.phi(g, -eigvecs)
        pe = pe.reshape(n, -1) # n x k x D1 -> n x k*D1
        pe = self.rho(pe)

        return self.base_model(g, x, pe)
```

Figure E-1: PyTorch-like pseudo-code for using SignNet with a GNN prediction model, where $\phi = \text{GIN}$ and $\rho = \text{MLP}$ as in the ZINC molecular graph regression experiments. Reshaping eigenvectors from $n \times k$ to $n \times k \times 1$ allows ϕ to process each eigenvector (and its negation) independently in PyTorch-like deep learning libraries.

we give pseudo-code of SignNet for learning node positional encodings with a GNN prediction model.

E.2.1 Generalization Beyond Symmetric Matrices

In the main paper, we assume that the eigenspaces come from a symmetric matrix. This holds for many cases of practical interest, as e.g. the Laplacian matrix of an undirected graph is symmetric. However, we may also want to process directed graphs, or other data that have associated nonsymmetric matrices. Our SignNet and BasisNet generalize in a straightforward way to handle nonsymmetric diagonalizable matrices, as we detail here. Let $A \in \mathbb{R}^{n \times n}$ be a matrix with a diagonalization

$A = V\Lambda V^{-1}$, where $\Lambda = \text{Diag}(\lambda_1, \ldots, \lambda_n)$ contains the eigenvalues λ_i, and the columns of $V = \begin{bmatrix} v_1 & \ldots & v_n \end{bmatrix}$ are eigenvectors. Suppose we want to learn a function on the eigenvectors $v_1, \ldots, v_k$. Unlike in the symmetric matrix case, the eigenvectors are not necessarily orthonormal, and both the eigenvalues and eigenvectors can be complex.

Real eigenvectors. First, we assume the eigenvectors v_i are all real vectors in $\mathbb{R}^n$. We can take the eigenvectors to be real if A is symmetric, or if A has real eigenvalues (see Horn and Johnson [2012] Theorem 1.3.29). Also, suppose that we choose the real numbers $\mathbb{R}$ as our base field for the vector space in which eigenvectors lie. Note that for any scaling factor $c \in \mathbb{R} \setminus \{0\}$ and eigenvector v, we have that cv is an eigenvector of the same eigenvalue. If the eigenvalues are distinct, then the eigenvectors of the form cv are the only other eigenvectors in the same eigenspace as v. Thus, we want a function to be invariant to scalings:

$$f(v_1, \ldots, v_k) = f(c_1 v_1, \ldots, c_k v_k) \qquad c_i \in \mathbb{R} \setminus \{0\}. \tag{E.2}$$

This can be handled by SignNet, by giving unit normalized vector inputs:

$$f(v_1, \ldots, v_k) = \rho\left([\phi(v_i/\|v_i\|) + \phi(-v_i/\|v_i\|)]_{i=1,\ldots,k} \right). \tag{E.3}$$

Now, say have bases of eigenspaces $V_1, \ldots, V_l$ with dimensions $d_1, \ldots, d_l$. For a basis V_i, we have that any other basis of the same space can be obtained as $V_i W$ for some $W \in \text{GL}_{\mathbb{R}}(d_i)$, the set of real invertible matrices in $\mathbb{R}^{d_i \times d_i}$. Indeed, the orthonormal projector for the space spanned by the columns of V_i is given by $V_i(V_i^\top V_i)^{-1} V_i^\top$. Thus, if $Z \in \mathbb{R}^{n \times d_i}$ is another basis for the column space of V_i, we have that $V_i(V_i^\top V_i)^{-1} V_i^\top = Z(Z^\top Z)^{-1} Z^\top$, so

$$V_i(V_i^\top V_i)^{-1} V_i^\top Z = Z(Z^\top Z)^{-1} Z^\top Z = Z, \tag{E.4}$$

so let $W = (V_i^\top V_i)^{-1} V_i^\top Z \in \mathbb{R}^{d_i \times d_i}$. Note that W is invertible, because it has inverse $(Z^\top Z)^{-1} Z^\top V_i$, so indeed $V_i W = Z$ for $W \in \text{GL}_{\mathbb{R}}(d_i)$. Thus, basis invariance in this

case is of the form

$$f(V_1 \ldots, V_l) = f(V_1 W_1, \ldots, V_l W_l) \qquad W_i \in \mathrm{GL}_{\mathbb{R}}(d_i). \tag{E.5}$$

Note that the distinct eigenvalue invariance is a special case of this invariance, as $\mathrm{G}_{\mathbb{R}}(1) = \mathbb{R} \setminus \{0\}$. We can again achieve this basis invariance by using a BasisNet, where the inputs to the ϕ_{d_i} are orthogonal projectors of the corresponding eigenspace:

$$f(V_1, \ldots, V_l) = \rho\left(\left[\phi_{d_i}(V_i(V_i^\top V_i)^{-1} V_i^\top)\right]_{i=1,\ldots,l}\right). \tag{E.6}$$

Recall that if V_i is an orthonormal basis, then the orthogonal projector is just $V_i V_i^\top$, so this is a direct generalization of BasisNet in the symmetric case.

Complex eigenvectors. More generally, suppose $V \in \mathbb{C}^{n \times n}$ are complex eigenvectors, and we take the base field of the vector space to be $\mathbb{C}$. The above arguments generalize to the complex case; in the case of distinct eigenvalues, we want

$$f(v_1, \ldots, v_k) = f(c_1 v_1, \ldots, c_k v_k) \qquad c_i \in \mathbb{C} \setminus \{0\}. \tag{E.7}$$

However, this symmetry can not be as easily reduced to a unit normalization and a discrete sign invariance, as it can be in the real case. Nonetheless, the basis invariant architecture directly generalizes, so we can handle the case of distinct eigenvalues by a more general basis invariant architecture as well. The basis invariance is

$$f(V_1, \ldots, V_l) = f(V_1 W_1, \ldots, V_l W_l) \qquad W_i \in \mathrm{GL}_{\mathbb{C}}(d_i). \tag{E.8}$$

The orthogonal projector of the image of V_i is $V_i(V_i^* V_i)^{-1} V_i^*$, where there are now conjugate transposes replacing the transposes. Thus, BasisNet takes the form:

$$f(V_1, \ldots, V_l) = \rho\left(\left[\phi_{d_i}(V_i(V_i^* V_i)^{-1} V_i^*)\right]_{i=1,\ldots,l}\right). \tag{E.9}$$

E.2.2 Complexity of SignNet and BasisNet

Here, we give a simplified but intuitive analysis of the complexity of SignNet and BasisNet. Suppose we have a graph of n nodes, with k eigenvectors $v_1, \ldots, v_k$. A standard GNN that naively inputs the eigenvectors as node features forms tensors of size $\mathcal{O}(nk + nd)$, where d is the hidden dimension of the learned node features. SignNet forms tensors of size $\mathcal{O}(nkd)$, where d is the hidden dimension or output dimension of ϕ. This is because for each of the $2k$ eigenvectors v_i and $-v_i$, we must put it through our ϕ network. Similarly, BasisNet forms tensors of size $\mathcal{O}(n^2ld)$, where l is the number of eigenspaces and d is the hidden dimension or output dimension of the ϕ_{d_i}. Thus, there is an extra multiplicative factor of n when compared with SignNet. If we instead use p-IGNs with order p tensors, then the complexity is $\mathcal{O}(n^pld)$. Moreover, note that a naive version of BasisNet requires a separate IGN to be learned for each multiplicity d_i. This may be intractable for datasets with eigenspaces of many sizes. One way to get around this would be to parameterize a single IGN, and define $\phi_{d_i}(V_iV_i^\top) = \text{IGN}(V_iV_i^\top, d_i)$; in other words, we simply input the dimension to the shared IGN. We have not tested the learning capabilities of this more efficient model in this work, but it could be promising for future work.

E.2.3 Other Architectural Notes

There are several alternatives available in the design of SignNet and BasisNet that we now discuss. Our approach, as outlined in Figure 7-2, processes the eigenvectors independently to compute learned positional encodings and then uses these learned positional encodings along with the node features X in a final base model (say, a GNN) to get a prediction. Another possibility is to process eigenvectors and node features jointly. One way to do this is to add X as input to ϕ, so for instance SignNet would include $\phi(v_i, X) + \phi(-v_i, X)$. However, this requires processing X $2k$ times with ϕ, which may be inefficient.

Another possibility to parameterize a sign invariant architecture is through taking elementwise absolute values of eigenvectors, and then composing with arbitrary

functions, e.g. $\text{MLP}(|v_1|, \ldots, |v_k|)$, where the MLP acts independently on each node. Empirically, this often does not work well (see our results on ZINC as well as those of Dwivedi et al. [2020]). Intuitively, these elementwise absolute values remove distance information, since for instance nodes i and j in which $v_2^{(i)} = -v_2^{(j)}$ are typically far in the graph, but they will have the same value in this eigenvector under the absolute value mapping. Nonetheless, if the ϕ in SignNet is taken to be an elementwise function, meaning $\phi : \mathbb{R}^n \to \mathbb{R}^{n \times d}$ satisfies $\phi(v)_i = \psi(v_i)$ for some ψ applied independently to each node, then SignNet is equivalent in expressiveness to $\text{MLP}(|v_1|, \ldots, |v_k|)$, where the MLP acts independently on each node.

E.3 More on Eigenvalue Multiplicities

In this section, we study the properties of eigenvalues and eigenvectors computed by numerical algorithms on real-world data.

E.3.1 Sign and Basis Ambiguities in Numerical Eigensolvers

When processing real-world data, we use eigenvectors that are computed by numerical algorithms. These algorithms return specific eigenvectors for each eigenspace, so there is some choice of sign or basis of each eigenspace. The general symmetric matrix eigensolvers `numpy.linalg.eigh` and `scipy.linalg.eigh` both call LAPACK routines. They both proceed as follows: for a symmetric matrix A, they first decompose it as $A = QTQ^\top$ for orthogonal Q and tridiagonal T, then they compute the eigendecomposition of $T = W\Lambda W^\top$, so the eigendecomposition of A is $A = (QW)\Lambda(W^\top Q^\top)$. There are multiple ambiguities here: for diagonal sign matrices $S = \text{Diag}(s_1, \ldots, s_n)$ and $S' = \text{Diag}(s'_1, \ldots, s'_n)$, where $s_i, s'_i \in \{-1, 1\}$, we have that $A = QS(STS)SQ^\top$ is also a valid tridiagonalization, as QS is still orthogonal, $SS = I$, and STS is still tridiagonal. Also, $T = (WS')\Lambda(S'W^\top)$ is a valid eigendecomposition of T, as WS' is still orthogonal.

In practice, we find that the general symmetric matrix eigensolvers `numpy.linalg.eigh` and `scipy.linalg.eigh` differ between frameworks but are consistent with the same

framework. More specifically, for a symmetric matrix A, we find that the eigenvectors computed with the default settings in numpy tend to differ by a choice of sign or basis from those that are computed with the default settings in scipy. On the other hand, the called LAPACK routines are deterministic, so the eigenvectors returned by numpy are the same in each call, and the eigenvectors returned by scipy are likewise the same in each call.

Eigensolvers for sparse symmetric matrices like `scipy.linalg.eigsh` are required for large scale problems. This function calls ARPACK, which uses an iterative method that starts with a randomly sampled initial vector. Due to this stochasticity, the sign and basis of eigenvectors returned differs between each call.

Bro et al. [2008] develop a data-dependent method to choose signs for each singular vector of a singular value decomposition. Still, in the worst case the signs chosen will be arbitrary, and they do not handle basis ambiguities in higher dimensional eigenspaces. Other works have made choices of sign, such as by picking the sign so that the eigenvector's entries are in the largest lexicographic order [Tam and Dunson, 2022]. This choice of sign may work poorly for learning on graphs, as it is sensitive to permutations on nodes. For some graph regression experiments in Section 7.3.1, we try a choice of sign that is permutation invariant, but we find it to work poorly.

E.3.2 Higher Dimensional Eigenspaces in Real Graphs

Here, we investigate the normalized Laplacian eigenspace statistics of real-world graph data. For any graph that has distinct Laplacian eigenvalues, only sign invariance is required in processing eigenvectors. However, we find that graph data tends to have higher multiplicity eigenvalues, so basis invariance would be required for learning symmetry-respecting functions on eigenvectors.

Indeed, we show statistics for multi-graph datasets in Table E.2 and for single-graph datasets with more nodes per graph in Table E.3. For multi-graph datasets, we consider :

- Molecule graphs: ZINC [Irwin et al., 2012, Dwivedi et al., 2020], ogbg-molhiv [Wu

et al., 2018, Hu et al., 2020b]

- Social networks: IMDB-M, COLLAB [Yanardag and Vishwanathan, 2015, Morris et al., 2020a],
- Bioinformatics graphs: PROTEINS [Morris et al., 2020a]
- Computer vision graphs: COIL-DEL [Riesen and Bunke, 2008, Morris et al., 2020a].

For single-graph datasets, we consider:

- The 32×32 image grid as in Section E.10.3
- Citation networks: Cora, Citeseer [Sen et al., 2008]
- Co-purchasing graphs with Amazon Photo [McAuley et al., 2015, Shchur et al., 2018].

We see that these datasets all contain higher multiplicity eigenspaces, so sign invariance is insufficient for fully respecting symmetries. The majority of graphs in each multi-graph dataset besides COIL-DEL contain higher multiplicity eigenspaces. Also, the dimension of these eigenspaces can be quite large compared to the size of the graphs in the dataset. The single-graph datasets have a large proportion of their eigenvectors belonging to higher dimensional eigenspaces. Thus, basis invariance may play a large role in processing spectral information from these graph datasets.

E.3.3 Relationship to Graph Automorphisms

Higher multiplicity eigenspaces are related to automorphism symmetries in graphs. For an adjacency matrix A, the permutation matrix P is an automorphism of the graph associated to A if $PAP^\top = A$. If P is an automorphism, then for any eigenvector v of A with eigenvalue λ, we have

$$APv = PAP^\top Pv = PAv = P\lambda v = \lambda Pv, \tag{E.10}$$

Table E.2: Eigenspace statistics for datasets of multiple graphs. From left to right, the columns are: dataset name, number of graphs, range of number of nodes per graph, largest multiplicity, and percent of graphs with an eigenspace of dimension > 1.

Dataset	Graphs	# Nodes	Max. Mult	% Graphs mult. > 1
ZINC	12,000	9-37	9	64.1
ZINC-full	249,456	6-38	10	63.8
ogbg-molhiv	41,127	2 - 222	42	68.0
IMDB-M	1,500	7 - 89	37	99.9
COLLAB	5,000	32 - 492	238	99.1
PROTEINS	1,113	4 - 620	20	77.3
COIL-DEL	3,900	3 - 77	4	4.00

Table E.3: Eigenspace statistics for single graphs. From left to right, the columns are: dataset name, number of nodes, distinct eigenvalues (i.e. distinct eigenspaces), number of unique multiplicities, largest multiplicity, and percent of eigenvectors belonging to an eigenspace of dimension > 1.

Dataset	Nodes	Distinct λ	# Mult.	Max Mult.	% Vecs mult. > 1
32×32 image	1,024	513	3	32	96.9
Cora	2,708	2,187	11	300	19.7
Citeseer	3,327	1,861	12	491	44.8
Amazon Photo	7,650	7,416	8	136	3.71

so Pv is an eigenvector of A with the same eigenvalue λ. If Pv and v are linearly independent, then λ has a higher dimensional eigenspace. Thus, under certain additional conditions, automorphism symmetries of graphs lead to repeated eigenvalues [Sachs and Stiebitz, 1983, Teranishi, 2009].

E.3.4 Multiplicities in Random Graphs

It is known that almost all random graphs under the Erdős-Renyi model have no repeated eigenvalues in the infinite number of nodes limit [Tao and Vu, 2017]. Likewise, almost all random graphs under the Erdős-Renyi model are asymmetric in the sense of having no nontrivial automorphism symmetries [Erdos and Rényi, 1963]. These results contrast sharply with the high eigenvalue multiplicities that we see in real-world data in Section E.3.2. Likewise, many types of real-world graph data have been found to possess nontrivial automorphism symmetries [Ball and Geyer-Schulz, 2018]. This demonstrates a potential downside of using random graph models to study real-world data: the eigenspace dimensions and automorphism symmetries of random graphs may not agree with those of real-world data.

E.4 Visualization of SignNet output

E.4.1 Cat Model Visualization

In Figure E-2, we plot the eigenvectors of the cotangent Laplacian on a cat model, as well as the first principal component of the corresponding learned $\phi(v) + \phi(-v)$ from our SignNet model that was trained on the texture reconstruction task. Interestingly, this portion of our SignNet encodes bilateral symmetry; for instance, while some eigenvectors differ between left feet and right feet, this portion of our SignNet gives similar values for the left and right feet. This is useful for the texture reconstruction task, as the texture regression target has bilateral symmetry.

We also show principal components of outputs for the full SignNet model in Figure E-3. This is not as interpretable, as the outputs are high frequency and appear to be close to the texture that is the regression target. If instead we trained the network on a task involving eigenvectors of multiple models, then we may expect the SignNet to learn more structurally interpretable mappings (as in the case of the molecule tasks).

E.4.2 Molecule visualization

To better understand SignNet, in Figure E-5 we visualize the learned positional encodings of a SignNet with $\phi = \text{GIN}$, $\rho = \text{MLP}$ (with a summation to handle variable eigenvector numbers) trained on ZINC as in Section 7.3.1. SignNet learns interesting structural information such as cut nodes (PC 3) and appendage atoms (PC 2) that qualitatively differ from any single eigenvector of the graph.

For this visualization we use a SignNet trained with a GatedGCN base model on ZINC, as in Section 7.3.1. This SignNet uses GIN as ϕ and ρ as an MLP (with a sum before it to handle variable numbers of eigenvectors), and takes in all eigenvectors of each graph. See Figure E-4 for all of the eigenvectors of fluorescein.

E.5 More Related Work

E.5.1 Graph Positional Encodings

Various graph positional encodings have been proposed, which have been motivated for increasing expressive power or practical performance of graph neural networks, and for generalizing Transformers to graphs. Positional encodings are related to so-called position-aware network embeddings [Chami et al., 2020], which capture distances between nodes in graphs. These include network embedding methods like Deepwalk [Perozzi et al., 2014] and node2vec [Grover and Leskovec, 2016], which have been recently integrated into GNNs that respect their invariances by Wang et al. [2022]. Further, Li et al. [2020] studies the theoretical and practical benefits of incorporating distance features into graph neural networks. Dwivedi et al. [2022] proposes a method to inject learnable positional encodings into each layer of a graph neural network, and uses a simple random walk based node positional encoding. You et al. [2021] proposes a node positional encoding $\text{diag}(A^k)$, which captures the number of closed walks from a node to itself. Dwivedi et al. [2020] propose to use Laplacian eigenvectors as positional encodings in graph neural networks, with sign ambiguities alleviated by sign flipping data augmentation. Srinivasan and Ribeiro [2019] theoretically analyze node positional embeddings and structural representations in graphs, and show that most-expressive structural representations contain the information of any node positional embedding.

While positional encodings in sequences as used for Transformers [Vaswani et al., 2017b] are able to leverage the canonical order in sequences, there is no such useful canonical order for nodes in a graph, due in part to permutation symmetries. Thus, different permutation equivariant positional encodings have been proposed to help generalize Transformers to graphs. Dwivedi and Bresson [2021] directly add in linearly projected Laplacian eigenvectors to node features before processing these features with a graph Transformer. Kreuzer et al. [2021] propose an architecture that uses attention over Laplacian eigenvectors and eigenvalues to learn node or edge positional encodings. Mialon et al. [2021] uses spectral kernels such as the diffusion kernel to define relative positional encodings that modulate the attention matrix. Ying et al.

[2021] achieve state-of-the-art empirical performance with simple Transformers that incorporate shortest-path based relative positional encodings. Zhang et al. [2020] also utilize shortest-path distances for positional encodings in their graph Transformer. Kim et al. [2021] develop higher-order transformers (that generalize invariant graph networks), which interestingly perform well on graph regression using sparse higher-order transformers without positional encodings.

E.5.2 Eigenvector Symmetries in Graph Representation Learning

Many works that attempt to respect the invariances of eigenvectors solely focus on sign invariance (by using data augmentation) [Dwivedi et al., 2020, Dwivedi and Bresson, 2021, Dwivedi et al., 2022, Kreuzer et al., 2021]. This may be reasonable for continuous data, where eigenvalues of associated matrices may be usually distinct and separated (e.g. Puny et al. [2022] finds that this empirically holds for covariance matrices of n-body problems). However, discrete graph Laplacians are known to have higher multiplicity eigenvalues in many cases, and in Appendix E.3.2 we find this to be true in various types of real-world graph data. Graphs without higher multiplicity eigenspaces are easier to deal with; in fact, graph isomorphism can be tested in polynomial time on graphs of bounded multiplicity for adjacency matrix eigenvalues [Babai et al., 1982, Leighton and l. Miller, 1979], with a time complexity that is lower for graphs with lower maximum multiplicities.

A recent work of Wang et al. [2022] proposes full orthogonal group invariance for functions that process positional encodings. In particular, for positional encodings $Z \in \mathbb{R}^{n \times k}$, they parameterize functions $f(Z)$ such that $f(Z) = f(ZQ)$ for all $Q \in O(k)$. This indeed makes sense for network embeddings like node2vec [Grover and Leskovec, 2016], as their objective functions are based on inner products and are thus orthogonally invariant. While they prove stability results when enforcing full orthogonal invariance for eigenvectors, this is a very strict constraint compared to our basis invariance. For instance, when $k = n$ and all eigenvectors are used in V, the condition $f(V) = f(VQ)$

implies that f is a constant function on orthogonal matrices, since any orthogonal matrix W can be obtained as $W = VQ$ for $Q = V^\top W \in O(n)$. In other words, for bases of eigenspaces $V_1, \ldots, V_l$ and $V = \begin{bmatrix} V_1 & \ldots & V_l \end{bmatrix}$, Wang et al. [2022] enforces $VQ \cong V$, while we enforce $V\text{Diag}(Q_1, \ldots, Q_l) \cong V$. While the columns of $V\text{Diag}(Q_1, \ldots, Q_l)$ are still eigenvectors, the columns of VQ generally are not.

E.5.3 Graph Spectra and Learning on Graphs

More generally, graph spectra are widely used in analyzing graphs, and spectral graph theory [Chung, 1997] studies the connection between graph properties and graph spectra. Different graph kernels have been defined based on graph spectra, which use robust and discriminative notions of generalized spectral distance [Verma and Zhang, 2017], the spectral density of states [Huang et al., 2021], random walk return probabilities [Zhang et al., 2018c], or the trace of the heat kernel [Tsitsulin et al., 2018]. Graph signal processing relies on spectral operations to define Fourier transforms, frequencies, convolutions, and other useful concepts for processing data on graphs [Ortega et al., 2018]. The closely related spectral graph neural networks [Wu et al., 2020, Balcilar et al., 2020] parameterize neural architectures that are based on similar spectral operations.

E.6 Definitions, Notation, and Background

E.6.1 Basic Topology and Algebra Definitions

We will use some basic topology and algebra for our theoretical results. A topological space $(\mathcal{X}, \tau)$ is a set $\mathcal{X}$ along with a family of subsets $\tau \subseteq 2^{\mathcal{X}}$ satisfying certain properties, which gives useful notions like continuity and compactness. From now on, we will omit mention of τ, and refer to a topological space as the set $\mathcal{X}$ itself. For topological spaces $\mathcal{X}$ and $\mathcal{Y}$, we write $\mathcal{X} \cong \mathcal{Y}$ and say that $\mathcal{X}$ is homeomorphic to $\mathcal{Y}$ if there exists a continuous bijection with continuous inverse from $\mathcal{X}$ to $\mathcal{Y}$. We will say $\mathcal{X} = \mathcal{Y}$ if the underlying sets and topologies are equal as sets (we will often

use this notion of equality for simplicity, even though it can generally be substituted with homeomorphism). For a function $f : \mathcal{X} \to \mathcal{Y}$ between topological spaces $\mathcal{X}$ and $\mathcal{Y}$, the image $\mathrm{im} f$ is the set of values that f takes, $\mathrm{im} f = \{f(x) : x \in \mathcal{X}\}$. This is also denoted $f(\mathcal{X})$. A function $f : \mathcal{X} \to \mathcal{Y}$ is called a topological embedding if it is a homeomorphism from $\mathcal{X}$ to its image.

A group G is a set along with a multiplication operation $G \times G \to G$, such that multiplication is associative, there is a multiplicative identity $e \in G$, and each $g \in G$ has a multiplicative inverse g^{-1}. A topological group is a group that is also a topological space such that the multiplication and inverse operations are continuous.

A group G may act on a set $\mathcal{X}$ by a function $\cdot : G \times \mathcal{X} \to \mathcal{X}$. We usually denote $g \cdot x$ as gx. A topological group is said to act continuously on a topological space $\mathcal{X}$ if $\cdot$ is continuous. For any group G and topological space $\mathcal{X}$, we define the coset $Gx = \{gx : g \in G\}$, which can be viewed as an equivalance class of elements that can be transformed from one to another by a group element. The quotient space $\mathcal{X}/G = \{Gx : x \in \mathcal{X}\}$ is the set of all such equivalence classes, with a topology induced by that of $\mathcal{X}$. The quotient map $\pi : \mathcal{X} \to \mathcal{X}/G$ is a surjective continuous map that sends x to its coset, $\pi(x) = Gx$.

For $x \in \mathbb{R}^d$, $\|x\|_2$ denotes the standard Euclidean norm. By the ∞ norm of functions $f : \mathcal{Z} \to \mathbb{R}^d$ from a compact $\mathcal{Z}$ to a Euclidean space $\mathbb{R}^d$, we mean $\|f\|_\infty = \sup_{z \in \mathcal{Z}} \|f(z)\|_2$.

E.6.2 Background on Eigenspace Invariances

Let $V = \begin{bmatrix} v_1 & \dots & v_d \end{bmatrix}$ and $W = \begin{bmatrix} w_1 & \dots & w_d \end{bmatrix} \in \mathbb{R}^{n \times d}$ be two orthonormal bases for the same d dimensional subspace of $\mathbb{R}^n$. Since V and W span the same space, their orthogonal projectors are the same, so $VV^\top = WW^\top$. Also, since V and W have orthonormal columns, we have $V^\top V = W^\top W = I \in \mathbb{R}^{d \times d}$. Define $Q = V^\top W$. Then Q is orthogonal because

$$Q^\top Q = W^\top V V^\top W = W^\top W W^\top W = I \tag{E.11}$$

Moreover, we have that

$$VQ = VV^\top W = WW^\top W = W \tag{E.12}$$

Thus, for any orthonormal bases V and W of the same subspace, there exists an orthogonal $Q \in O(d)$ such that $VQ = W$.

For another perspective on this, define the Grassmannian $\mathrm{Gr}(d, n)$ as the smooth manifold consisting of all d dimensional subspaces of $\mathbb{R}^n$. Further define the Stiefel manifold $\mathrm{St}(d, n)$ as the set of all orthonormal tuples $\begin{bmatrix} v_1 & \dots & v_d \end{bmatrix} \in \mathbb{R}^{n \times d}$ of d vectors in $\mathbb{R}^n$. Letting $O(d)$ act by right multiplication, it holds that $\mathrm{St}(d, n)/O(d) \cong \mathrm{Gr}(d, n)$. This implies that any $O(d)$ invariant function on $\mathrm{St}(d, n)$ can be viewed as a function on subspaces. See e.g. Gallier and Quaintance [2020] Chapter 5 for more information on this. We will use this relationship in our proofs of universal representation.

When we consider permutation invariance or equivariance, the permutation acts on dimensions of size n. Then a tensor $X \in \mathbb{R}^{n^k \times d}$ is called an order k tensor with respect to this permutation symmetry, where order 0 are called scalars, order 1 tensors are called vectors, and order 2 tensors are called matrices. Note that this does not depend on d; in this work, we only ever consider vectors and scalars with respect to the $O(d)$ action.

E.7 Proofs of Universality

We begin by proving the two propositions for the single subspace case from Section 7.1.1.

. *A continuous function $h : \mathbb{R}^n \to \mathbb{R}^{d_{\text{out}}}$ is sign invariant if and only if*

$$h(v) = \phi(v) + \phi(-v) \tag{7.3}$$

for some continuous $\phi : \mathbb{R}^n \to \mathbb{R}^{d_{\text{out}}}$. A continuous $h : \mathbb{R}^n \to \mathbb{R}^n$ is sign invariant and permutation equivariant if and only if (7.3) holds for a continuous permutation equivariant $\phi : \mathbb{R}^n \to \mathbb{R}^n$.

Proof. If $h(v) = \phi(v) + \phi(-v)$, then h is obviously sign invariant. On the other hand, if h is sign invariant, then letting $\phi(v) = h(v)/2$ gives that $h(v) = \phi(v) + \phi(-v)$, and ϕ is of course continuous.

If $h(v) = \phi(v) + \phi(-v)$ for a permutation equivariant ϕ, then $h(-Pv) = \phi(-Pv) + \phi(Pv) = P\phi(-v) + P\phi(v) = P(\phi(v) + \phi(-v)) = Ph(v)$, so h is permutation equivariant and sign invariant. If h is permutation equivariant and sign invariant, then define $\phi(v) = h(v)/2$ again; it is clear that ϕ is continuous and permutation equivariant. □

. *Any continuous, $O(d)$ invariant $h : \mathbb{R}^{n\times d} \to \mathbb{R}^{d_{\text{out}}}$ is of the form $h(V) = \phi(VV^\top)$ for a continuous ϕ. For a compact domain $\mathcal{Z} \subseteq \mathbb{R}^{n\times d}$, maps of the form $V \mapsto \text{IGN}(VV^\top)$ universally approximate continuous functions $h : \mathcal{Z} \subseteq \mathbb{R}^{n\times d} \to \mathbb{R}^n$ that are $O(d)$ invariant and permutation equivariant.*

Proof. The case without permutation equivariance holds by the First Fundamental Theorem of $O(d)$ (Lemma 12).

For the permutation equivariant case, let $\mathcal{Z}' = \{VV^\top : V \in \mathcal{Z}\}$ and let $\epsilon > 0$. Note that $\mathcal{Z}'$ is compact, as it is the continuous image of a compact set. Since h is $O(d)$ invariant, the first fundamental theorem of $O(d)$ shows that there exists a continuous function $\phi : \mathcal{Z}' \subseteq \mathbb{R}^{n\times n} \to \mathbb{R}^n$ such that $h(V) = \phi(VV^\top)$. Since h is permutation equivariant, for any permutation matrix P we have that

$$h(PV) = P \cdot h(V) \tag{E.13}$$

$$\phi(PVV^\top P^\top) = P \cdot \phi(VV^\top), \tag{E.14}$$

so ϕ is a continuous permutation equivariant function from matrices to vectors. Then note that Keriven and Peyré [2019] show that invariant graph networks (of generally high tensor order in hidden layers) universally approximate continuous permutation equivariant functions from matrices to vectors on compact sets of matrices. Thus, an IGN can ϵ-approximate ϕ, and hence $V \mapsto \text{IGN}(VV^\top)$ can ϵ-approximate h. □

E.7.1 Proof of Decomposition Theorem

Here, we give the formal statement of Theorem 10, which provides the necessary topological assumptions for the theorem to hold. In particular, we only require the G_i be a topological group that acts continuously on $\mathcal{X}_i$ for each i, and that there exists a topological embedding of each quotient space into some Euclidean space. That the group action is continuous is a very mild assumption, and it holds for any finite or compact matrix group, which all of the invariances we consider in this paper can be represented as.

A topological embedding of the quotient space into a Euclidean space is desired, as we know how to parameterize neural networks with Euclidean outputs and inputs, whereas dealing with a quotient space is generally difficult. Many different conditions can guarantee existence of such an embedding. For instance, if the quotient space is a smooth manifold, then the Whitney Embedding Theorem (Lemma 15) guarantees such an embedding. Also, if the base space $\mathcal{X}_i$ is a Euclidean space and G_i is a finite or compact matrix Lie group, then a map built from G-invariant polynomials gives such an embedding (González and de Salas [2003] Lemma 11.13).

Figure E-6 provides a commutative diagram representing the constructions in our proof.

Theorem 10 (Decomposition Theorem)**.** *Let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ be topological spaces, and let G_i be a topological group acting continuously on $\mathcal{X}_i$ for each i. Assume that there is a topological embedding $\psi_i : \mathcal{X}_i/G_i \to \mathbb{R}^{a_i}$ of each quotient space into a Euclidean space $\mathbb{R}^{a_i}$ for some dimension a_i. Then, for any continuous function $f : \mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \mathbb{R}^{d_{\text{out}}}$ that is invariant to the action of $G = G_1 \times \ldots \times G_k$, there exists continuous functions $\phi_i : \mathcal{X}_i \to \mathbb{R}^{a_i}$ and a continuous function $\rho : \mathcal{Z} \subseteq \mathbb{R}^a \to \mathbb{R}^{d_{\text{out}}}$, where $a = \sum_i a_i$ such that*

$$f(v_1, \ldots, v_k) = \rho(\phi_1(v_1), \ldots, \phi_k(v_k)). \tag{E.15}$$

Furthermore: (1) each ϕ_i can be taken to be invariant to G_i, (2) the domain $\mathcal{Z}$ is compact if each $\mathcal{X}_i$ is compact, (3) if $\mathcal{X}_i = \mathcal{X}_j$ and $G_i = G_j$, then ϕ_i can be taken to be

equal to ϕ_j.

Proof. Let $\pi_i : \mathcal{X}_i \to \mathcal{X}_i/G_i$ denote the quotient map for $\mathcal{X}_i/G_i$. Since each G_i acts continuously, Lemma 13 gives that the quotient of the product space is the product of the quotient spaces, i.e. that

$$(\mathcal{X}_1 \times \ldots \times \mathcal{X}_k)/(G_1 \times \ldots G_k) \cong (\mathcal{X}_1/G_1) \times \ldots \times (\mathcal{X}_k/G_k), \tag{E.16}$$

and the corresponding quotient map $\pi : \mathcal{X}/G$ is given by

$$\pi = \pi_1 \times \ldots \times \pi_k, \qquad \pi(x_1, \ldots, x_k) = (\pi_1(x_1), \ldots, \pi_k(x_k)). \tag{E.17}$$

By passing to the quotient (Lemma 11), there exists a continuous $\tilde{f} : \mathcal{X}/G \to \mathbb{R}^{d_{\text{out}}}$ on the quotient space such that $f = \tilde{f} \circ \pi$. By Lemma 14, each $\mathcal{X}_i/G_i$ is compact if $\mathcal{X}_i$ is compact. Defining the image $\mathcal{Z}_i = \psi_i(\mathcal{X}_i/G_i) \subseteq \mathbb{R}^{a_i}$, we thus know that $\mathcal{Z}_i$ is compact if $\mathcal{X}_i$ is compact.

Moreover, as ψ_i is a topological embedding, it has a continuous inverse ψ_i^{-1} on its image $\mathcal{Z}_i$. Further, we have a topological embedding $\psi : \mathcal{X}/G \to \mathcal{Z} = \mathcal{Z}_1 \times \ldots \times \mathcal{Z}_k$ given by $\psi = \psi_1 \times \ldots \times \psi_k$, with continuous inverse $\psi^{-1} = \psi_1^{-1} \times \ldots \times \psi_k^{-1}$.

Note that

$$f = \tilde{f} \circ \pi = (\tilde{f} \circ \psi^{-1}) \circ (\psi \circ \pi). \tag{E.18}$$

So we define

$$\rho = \tilde{f} \circ \psi^{-1} \qquad \rho : \mathcal{Z} \to \mathbb{R}^{d_{\text{out}}} \tag{E.19}$$

$$\phi_i = \psi_i \circ \pi_i \qquad \phi_i : \mathcal{X}_i \to \mathcal{Z}_i \tag{E.20}$$

$$\phi = \psi \circ \pi = \phi_1 \times \ldots \times \phi_k \qquad \phi : \mathcal{X} \to \mathcal{Z} \tag{E.21}$$

Thus, $f = \rho \circ \phi = \rho \circ (\phi_1 \times \ldots \times \phi_k)$, so equation (E.1) holds. Moreover, the ρ and ϕ_i are continuous, as they are compositions of continuous functions. Furthermore, (1) holds as each ϕ_i is invariant to G_i because each π_i is invariant to G_i. Since each $\mathcal{Z}_i$ is compact if $\mathcal{X}_i$ is compact, the product $\mathcal{Z} = \mathcal{Z}_1 \times \ldots \times \mathcal{Z}_k$ is compact if each $\mathcal{X}_i$ is

compact, thus proving (2).

To show the last statement (3), note simply that if $\mathcal{X}_i = \mathcal{X}_j$ and $G_i = G_j$, then the quotient maps are equal, i.e. $\pi_i = \pi_j$. Moreover, we can choose the embeddings to be equal, so say $\psi_i = \psi_j$. Then, $\phi_i = \psi_i \circ \pi_i = \psi_j \circ \pi_j = \phi_j$, so we are done. □

E.7.2 Universality of SignNet and BasisNet

Here, we prove Corollary 4 on the universal representation and approximation capabilities of our Unconstrained-SignNets, Unconstrained-BasisNets, and Expressive-BasisNets. We proceed in several steps, first proving universal representation of continuous functions when we do not require permutation equivariance, then proving universal approximation when we do require permutation equivariance.

Sign Invariant Universal Representation

Recall that $\mathbb{S}^{n-1}$ denotes the unit sphere in $\mathbb{R}^n$. As we normalize eigenvectors to unit norm, the domain of our functions on k eigenvectors are on the compact space $(\mathbb{S}^{n-1})^k$.

Corollary 5 (Universal Representation for SignNet)**.** *A continuous function $f : (\mathbb{S}^{n-1})^k \to \mathbb{R}^{d_{\text{out}}}$ is sign invariant, i.e. $f(s_1 v_1, \ldots, s_k v_k) = f(v_1, \ldots, v_k)$ for any $s_i \in \{-1, 1\}$, if and only if there exists a continuous $\phi : \mathbb{R}^n \to \mathbb{R}^{2n-2}$ and a continuous $\rho : \mathbb{R}^{(2n-2)k} \to \mathbb{R}^{d_{\text{out}}}$ such that*

$$f(v_1, \ldots, v_k) = \rho\left([\phi(v_i) + \phi(-v_i)]_{i=1}^k\right). \tag{E.22}$$

Proof. It can be directly seen that any f of the above form is sign invariant.

Thus, we show that any sign invariant f can be expressed in the above form. First, we show that we can apply the general Theorem 10. The group $G_i = \{1, -1\}$ acts continuously and satisfies that $\mathbb{S}^{n-1}/\{1, -1\} = \mathbb{RP}^{n-1}$, where $\mathbb{RP}^{n-1}$ is the real projective space of dimension $n-1$. Since $\mathbb{RP}^{n-1}$ is a smooth manifold of dimension $n-1$, Whitney's embedding theorem states that there exists a (smooth) topological embedding $\psi_i : \mathbb{RP}^{n-1} \to \mathbb{R}^{2n-2}$ (Lemma 15).

Thus, we can apply the general theorem to see that $f = \rho \circ \tilde{\phi}^k$ for some continuous ρ and $\tilde{\phi}^k$. Note that each $\tilde{\phi}_i = \tilde{\phi}$ is the same, as each $\mathcal{X}_i = \mathbb{S}^{n-1}$ and $G_i = \{1, -1\}$ is the same. Also, Theorem 10 says that we may assume that $\tilde{\phi}$ is sign invariant, so $\tilde{\phi}(x) = \tilde{\phi}(-x)$. Letting $\phi(x) = \tilde{\phi}(x)/2$, we are done with the proof. □

Sign Invariant Universal Representation with Extra Features

Recall that we may want our sign invariant functions to process other data besides eigenvectors, such as eigenvalues or node features associated to a graph. Here, we show universal representation for when we have this other data that does not possess sign symmetry. The proof is a simple extension of Corollary 5, but we provide the technical details for completeness.

Corollary 6 (Universal Representation for SignNet with features)**.** *For a compact space of features $\Omega \subseteq \mathbb{R}^d$, let $f(v_1, \ldots, v_k, x_1, \ldots, x_k)$ be a continuous function $f : (\mathbb{S}^{n-1} \times \Omega)^k \to \mathbb{R}^{d_{\text{out}}}$.*

Then f is sign invariant for the inputs on the sphere, i.e.

$$f(s_1 v_1, \ldots, s_k v_k, x_1, \ldots, x_k) = f(v_1, \ldots, v_k, x_1, \ldots, x_k) \qquad s_i \in \{1, -1\}, \tag{E.23}$$

if and only if there exists a continuous $\psi : \mathbb{R}^{n+d} \to \mathbb{R}^{2n-2+d}$ and a continuous $\rho : \mathbb{R}^{(2n-2+d)k} \to \mathbb{R}^{d_{\text{out}}}$ such that

$$f(v_1, \ldots, v_k) = \rho\left(\phi(v_1, x_1) + \phi(-v_1, x_1),\ \ldots,\ \phi(v_k, x_k) + \phi(-v_k, x_k)\right). \tag{E.24}$$

Proof. Once again, the sign invariance of any f in the above form is clear.

We follow very similar steps to the proof of Corollary 5 to show that we may apply Theorem 10. We can view Ω as a quotient space, after quotienting by the trivial group that does nothing, $\Omega \cong \Omega/\{1\}$. The corresponding quotient map is id_Ω, the identity map. Also, Ω trivially topologically embeds in $\mathbb{R}^d$ by the inclusion map.

As $G_i = \{-1, 1\} \times \{1\}$ acts continuously, by Lemma 13 we have that

$$(\mathbb{S}^{n-1} \times \Omega)/(\{1, -1\} \times \{1\}) \cong (\mathbb{S}^{n-1}/\{1, -1\}) \times (\Omega/\{1\}) \cong \mathbb{RP}^{n-1} \times \Omega, \tag{E.25}$$

with corresponding quotient map $\pi \times \mathrm{id}_\Omega$, where π is the quotient map to $\mathbb{RP}^{n-1}$.

Letting $\tilde{\psi}$ be the embedding of $\mathbb{RP}^{n-1} \to \mathbb{R}^{2n-2}$ guaranteed by Whitney's embedding theorem (Lemma 15), we have that $\psi = \tilde{\psi} \times \mathrm{id}_\Omega$ is an embedding of $\mathbb{RP}^{n-1} \times \Omega \to \mathbb{R}^{2n-2+d}$. Thus, we can apply Theorem 10 to write $f = \rho \circ \tilde{\phi}^k$ for $\tilde{\phi} = (\tilde{\psi} \times \mathrm{id}_\Omega) \circ (\pi \times \mathrm{id}_\Omega)$, so

$$\tilde{\phi}(v_i, x_i) = (\tilde{\psi}(v_i), x_i), \tag{E.26}$$

where $\tilde{\phi}(v_i, x_i) = \tilde{\phi}(-v_i, x_i)$. Letting $\phi(v_i, x_i) = \tilde{\phi}(v_i, x_i)/2$, we are done. □

Basis Invariant Universal Representation

Recall that $\mathrm{St}(d, n)$ is the Stiefel manifold of d-tuples of vectors $(v_1, \ldots, v_d)$ where $v_i \in \mathbb{R}^n$ and $v_1, \ldots, v_d$ are orthonormal. This is where our inputs lie, as our eigenvectors are unit norm and orthogonal. We will also make use of the Grassmannian $\mathrm{Gr}(d, n)$, which consists of all d-dimensional subspaces in $\mathbb{R}^n$. This is because the Grassmannian is the quotient space for the group action we want, $\mathrm{Gr}(d, n) \cong \mathrm{St}(d, n)/O(d)$, where $Q \in O(d)$ acts on $V \in \mathrm{St}(d, n) \subseteq \mathbb{R}^{n \times d}$ by mapping V to VQ [Gallier and Quaintance, 2020].

Corollary 7 (Universal Representation for BasisNet)**.** *For dimensions $d_1, \ldots, d_l \leq n$ let f be a continuous function on $\mathrm{St}(d_1, n) \times \ldots \times \mathrm{St}(d_l, n)$. Further assume that f is invariant to $O(d_1) \times \ldots \times O(d_l)$, where $O(d_i)$ acts on $\mathrm{St}(d_i, n)$ by multiplication on the right.*

Then there exist continuous $\rho : \mathbb{R}^{\sum_{i=1}^{l} 2d_i(n-d_i)} \to \mathbb{R}^{d_{\text{out}}}$ and continuous $\phi_i : \mathrm{St}(d_i, n) \to \mathbb{R}^{2d_i(n-d_i)}$ such that

$$f(V_1, \ldots, V_l) = \rho\left(\phi_1(V_1), \ldots, \phi_l(V_l)\right), \tag{E.27}$$

where the ϕ_i are $O(d_i)$ invariant functions, and we can take $\phi_i = \phi_j$ if $d_i = d_j$.

Proof. Letting $\mathcal{X}_i = \mathrm{St}(d_i, n)$ and $G_i = O(d_i)$, it can be seen that G_i acts continuously on $\mathcal{X}_i$. Also, we have that the quotient space $\mathrm{St}(d_i, n)/O(d_i) = \mathrm{Gr}(d_i, n)$ is the Grassmannian of d_i dimensional subspaces in $\mathbb{R}^n$, which is a smooth manifold of dimension $d_i(n - d_i)$. Thus, the Whitney embedding theorem (Lemma 15) gives a topological embedding $\psi_i : \mathrm{Gr}(d_i, n) \to \mathbb{R}^{2d_i(n-d_i)}$.

Hence, we may apply Theorem 10 to obtain continuous $O(d_i)$ invariant $\phi_i : \mathrm{St}(d_i, n) \to \mathbb{R}^{2d_i(n-d_i)}$ and continuous $\rho : \mathbb{R}^{\sum_{i=1}^{l} 2d_i(n-d_i)} \to \mathbb{R}^{d_{\text{out}}}$, such that $f = \rho \circ (\phi_1 \times \ldots \times \phi_l)$. Also, if $d_i = d_j$, then $\mathcal{X}_i = \mathcal{X}_j$ and $G_i = G_j$, so we can take $\phi_i = \phi_j$. □

Basis Invariant and Permutation Equivariant Universal Approximation

With the restriction that $f(V_1, \ldots, V_l) : \mathbb{R}^{n \times \sum_i d_i} \to \mathbb{R}^n$ be permutation equivariant and basis invariant, we need to use the impractically expensive Expressive-BasisNet to approximate f. Universality of permutation invariant or equivariant functions from matrices to scalars or matrices to vectors is difficult to achieve in a computationally tractable manner [Maron et al., 2019, Keriven and Peyré, 2019, Maehara and NT, 2019]. One intuitive reason to expect this is that universally approximating such functions allows solution of the graph isomorphism problem [Chen et al., 2019b], which is a computationally difficult problem. While we have exact representation of basis invariant functions by continuous ρ and ϕ_i when there is no permutation equivariance constraint, we can only achieve approximation up to an arbitrary $\epsilon > 0$ when we require permutation equivariance.

Corollary 8 (Universal Approximation for Expressive-BasisNets)**.** *Let $f(V_1, \ldots, V_l) : \mathrm{St}(d_1, n) \times \ldots \times \mathrm{St}(d_l, n) \to \mathbb{R}^n$ be continuous, $O(d_1) \times \ldots \times O(d_l)$ invariant, and permutation equivariant. Then f can be ϵ-approximated by an Expressive-BasisNet.*

Proof. By invariance, Corollary 7 of the decomposition theorem shows that f can be written as

$$f(V_1, \ldots, V_l) = \rho\left(\varphi_{d_1}(V_1), \ldots, \varphi_{d_l}(V_l)\right) \tag{E.28}$$

for some continuous $O(d_i)$ invariant φ_{d_i} and continuous ρ. By the first fundamental

theorem of $O(d)$ (Lemma 12), each φ_{d_i} can be written as $\varphi_{d_i}(V_i) = \phi_{d_i}(V_i V_i^\top)$ for some continuous ϕ_{d_i}. Let

$$\mathcal{Z} = \{(V_1 V_1^\top, \ldots, V_l V_l^\top) : V_i \in \mathrm{St}(d_i, n)\} \subseteq \mathbb{R}^{n^2 \times l}, \tag{E.29}$$

which is compact as it is the image of the compact space $\mathrm{St}(d_1, n) \times \ldots \times \mathrm{St}(d_l, n)$ under a continuous function. Define $h : \mathcal{Z} \subseteq \mathbb{R}^{n^2 \times l} \to \mathbb{R}^n$ by

$$h(V_1 V_1^\top, \ldots, V_l V_l^\top) = \rho\left(\phi_{d_1}(V_1 V_1^\top), \ldots, \phi_{d_l}(V_l V_l^\top)\right). \tag{E.30}$$

Then note that h is continuous and permutation equivariant from matrices to vectors, so it can be ϵ-approximated by an invariant graph network [Keriven and Peyré, 2019], call it $\widetilde{\mathrm{IGN}}$. If we define $\tilde{\rho} = \widetilde{\mathrm{IGN}}$ and $\mathrm{IGN}_{d_i}(V_i V_i^\top) = V_i V_i^\top$ (this identity operation is linear and permutation equivariant, so it can be exactly expressed by an IGN), then we have ϵ-approximation of f by

$$\widetilde{\mathrm{IGN}}(V_1 V_1^\top, \ldots, V_l V_l^\top) = \tilde{\rho}\left(\mathrm{IGN}_{d_1}(V_1 V_1^\top), \ldots, \mathrm{IGN}_{d_l}(V_l V_l^\top)\right). \tag{E.31}$$

□

E.7.3 Proof of Universal Approximation for General Decompositions

Theorem 11. *Consider the same setup as Theorem 10, where $\mathcal{X}_i$ are also compact. Let Φ_i be a family of G_i-invariant functions that universally approximate G_i-invariant continuous functions $\mathcal{X}_i \to \mathbb{R}^{a_i}$, and let $\mathcal{R}$ be a set of continuous function that universally approximate continuous functions $\mathcal{Z} \subseteq \mathbb{R}^a \to \mathbb{R}^{d_{\text{out}}}$ for every compact $\mathcal{Z}$, where $a = \sum_i a_i$. Then for any $\varepsilon > 0$ and any G-invariant continuous function $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \mathbb{R}^{d_{\text{out}}}$ there exists $\phi \in \Phi$ and $\rho \in \mathcal{R}$ such that $\|f - \rho(\phi_1, \ldots, \phi_k)\|_\infty < \varepsilon$.*

Proof. Consider a particular G-invariant continuous function $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \mathbb{R}^{d_{\text{out}}}$. By Theorem 10 there exists G_i-invariant continuous functions $\phi_i' : \mathcal{X}_i \to \mathbb{R}^{a_i}$ and a

continuous function $\rho' : \mathcal{Z} \subseteq \mathbb{R}^a \to \mathbb{R}^{d_{\text{out}}}$ (where $a = \sum_i a_i$) such that

$$f(v_1, \ldots, v_k) = \rho'(\phi'_1(v_1), \ldots, \phi'_k(v_k)).$$

Now fix an $\varepsilon > 0$. For any $\rho \in \mathcal{R}$ and any $\phi_i \in \Phi_i$ $(i = 1, \ldots k)$ we may bound the difference from f as follows (suppressing the v_i's for brevity),

$$\begin{aligned}
&\|f - \rho(\phi_1, \ldots, \phi_k)\|_\infty \\
&= \|\rho'(\phi'_1, \ldots, \phi'_k) - \rho(\phi_1, \ldots, \phi_k)\|_\infty \\
&= \|\rho'(\phi'_1, \ldots, \phi'_k) - \rho(\phi'_1, \ldots, \phi'_k) + \rho(\phi'_1, \ldots, \phi'_k) - \rho(\phi_1, \ldots, \phi_k)\|_\infty \\
&\leq \|\rho'(\phi'_1, \ldots, \phi'_k) - \rho(\phi'_1, \ldots, \phi'_k)\|_\infty + \|\rho(\phi'_1, \ldots, \phi'_k) - \rho(\phi_1, \ldots, \phi_k)\|_\infty \\
&= \mathrm{I} + \mathrm{II}
\end{aligned}$$

Now let $K' = \prod_{i=1}^k \mathrm{im}\phi'_i$. Since each ϕ'_i is continuous and defined on a compact set $\mathcal{X}_i$ we know that $\mathrm{im}\phi'_i$ is compact, and so the product K is also compact. Since K' is compact, it is contained in a closed ball $B(r)$ of radius $r > 0$ centered at the origin. Let K be the closed ball $B(r+1)$ of radius $r+1$ centered at the origin, so K contains K' and a ball of radius 1 around each point of K'. We may extend ρ' continuously to K as needed, so assume $\rho' : K \to \mathbb{R}^{d_{\text{out}}}$. By universality of $\mathcal{R}$ we may pick a particular $\rho : K \to \mathbb{R}^{d_{\text{out}}}$, $\rho \in \mathcal{R}$ such that

$$\mathrm{I} = \sup_{\{v_i \in \mathcal{X}_i\}_{i=1}^k} \|\rho'(\phi'_1, \ldots, \phi'_k) - \rho(\phi'_1, \ldots, \phi'_k)\|_\infty \leq \sup_{z \in K} \|\rho'(z) - \rho(z)\|_2 < \varepsilon/2.$$

Keeping this choice of ρ, it remains only to bound II. As ρ is continuous on a compact domain, it is in fact uniformly continuous. Thus, we can choose a $\delta' > 0$ such that if $\|y - z\|_2 \leq \delta'$, then $\|\rho(y) - \rho(z)\|_\infty < \epsilon/2$, and then we define $\delta = \min(\delta', 1)$.

Since Φ_i universally approximates ϕ'_i we may pick $\phi_i \in \Phi_i$ such that $\|\phi_i - \phi'_i\|_\infty < \delta/\sqrt{k}$, and thus $\|(\phi_1, \ldots, \phi_k) - (\phi'_1, \ldots \phi'_k)\|_\infty \leq \delta$. With this choice of ϕ_i, we know that $\prod_{i=1}^k \mathrm{im}\phi_i \subseteq K$ (because each $\phi_i(x_i)$ is within distance 1 of $\phi'_i(x_i)$). Thus,

$\rho(\phi_1(x_1), \ldots, \phi_k(x_k))$ is well-defined, and we have

$$\begin{aligned}
\text{II} &= \|\rho(\phi_1', \ldots, \phi_k') - \rho(\phi_1, \ldots, \phi_k)\|_\infty \\
&= \sup_{\{x_i \in \mathcal{X}_i\}_{i=1}^k} \|\rho(\phi_1'(x_1), \ldots, \phi_k'(x_k)) - \rho(\phi_1(x_1), \ldots, \phi_k(x_k))\|_2 \\
&< \varepsilon/2
\end{aligned}$$

due to our choice of δ, which completes the proof. □

E.8 Basis Invariance for Graph Representation Learning

E.8.1 Spectral Graph Convolution

In this section, we consider spectral graph convolutions, which for node features $X \in \mathbb{R}^{n \times d_{\text{feat}}}$ take the form $f(V, \Lambda, X) = \sum_{i=1}^n \theta_i v_i v_i^\top X$ for some parameters θ_i. We can optionally take $\theta_i = h(\lambda_i)$ for some continuous function $h : \mathbb{R} \to \mathbb{R}$ of the eigenvalues. This form captures most popular spectral graph convolutions in the literature [Bruna et al., 2014, Hamilton, 2020, Bronstein et al., 2017]; often, such convolutions are parameterized by taking h to be some analytic function such as a simple affine function [Kipf and Welling, 2017], a linear combination in a polynomial basis [Defferrard et al., 2016, Chien et al., 2021], or a parameterization of rational functions [Levie et al., 2018, Bianchi et al., 2021].

First, it is well known and easy to see that spectral graph convolutions are permutation equivariant, as for a permutation matrix P we have

$$f(PV, \Lambda, PX) = \sum_i \theta_i P v_i v_i^\top P^\top P X = \sum_i \theta_i P v_i v_i^\top X = P f(V, \Lambda, X). \tag{E.32}$$

Also, it is easy to see that they are sign invariant, as $(-v_i)(-v_i)^\top = v_i v_i^\top$. However, if the θ_i do not depend on the eigenvalues, then the spectral graph convolution is not necessarily basis invariant. For instance, if v_1 and v_2 are in the same eigenspace, and

we change basis by permuting $v_1' = v_2$ and $v_2' = v_1$, then if $\theta_1 \neq \theta_2$ the spectral graph convolution will generally change as well.

On the other hand, if $\theta_i = h(\lambda_i)$ for some function $h : \mathbb{R} \to \mathbb{R}$, then the spectral graph convolution is basis invariant. This is because if v_i and v_j belong to the same eigenspace, then $\lambda_i = \lambda_j$ so $h(\lambda_i) = h(\lambda_j)$. Thus, if $v_{i_1}, \ldots, v_{i_d}$ are eigenvectors of the same eigenspace with eigenvalue λ, we have that $\sum_{l=1}^{d} h(\lambda_{i_l}) v_{i_l} v_{i_l}^\top = h(\lambda) \sum_{l=1}^{d} v_{i_l} v_{i_l}^\top$. Now, note that $\sum_{l=1}^{d} v_{i_l} v_{i_l}^\top$ is the orthogonal projector onto the eigenspace [Trefethen and Bau III, 1997]. A change of basis does not change this orthogonal projector, so such spectral graph convolutions are basis invariant.

Another way to see this basis invariance is with a simple computation. Let $V_1, \ldots, V_l$ be the eigenspaces of dimension $d_1, \ldots, d_l$, where $V_i \in \mathbb{R}^{n \times d_i}$. Let the corresponding eigenvalues be $\mu_1, \ldots, \mu_l$. Then for any orthogonal matrices $Q_i \in O(d_i)$, we have

$$\sum_{i=1}^{n} h(\lambda_i) v_i v_i^\top = \sum_{j=1}^{l} V_j h(\mu_j) I_{d_j} V_j^\top \tag{E.33}$$

$$= \sum_{j=1}^{l} V_j h(\mu_j) I_{d_j} Q_j Q_j^\top V_j^\top \tag{E.34}$$

$$= \sum_{j=1}^{l} (V_j Q_j) h(\mu_j) I_{d_j} (V_j Q_j)^\top, \tag{E.35}$$

so the spectral graph convolution is invariant to substituting $V_j Q_j$ for V_j.

Now, we give the proof that shows SignNet and BasisNet can universally approximate spectral graph convolutions.

Theorem 7 (Learning Spectral Graph Convolutions)**.** *Suppose the node features $X \in \mathbb{R}^{n \times d_{\text{feat}}}$ take values in compact sets. Then SignNet can universally approximate any spectral graph convolution, and both BasisNet and Expressive-BasisNet can universally approximate any parametric spectral graph convolution.*

Proof. Note that eigenvectors and eigenvalues of normalized Laplacian matrices take values in compact sets, since the eigenvalues are in $[0, 2]$ and we take eigenvectors to have unit-norm. Thus, the whole domain of the spectral graph convolution is compact.

Let $\varepsilon > 0$. First, consider a spectral graph convolution $f(V, \Lambda, X) = \sum_{i=1}^{n} \theta_i v_i v_i^\top X$. For SignNet, let $\phi(v_i, \lambda_i, X)$ approximate the function $\tilde{\phi}(v_i, \lambda_i, X) = \theta_i v_i v_i^\top X$ to within ε/n error, which DeepSets can do since this is a continuous permutation equivariant function from vectors to vectors [Segol and Lipman, 2019] (note that we can pass λ_i as a vector in $\mathbb{R}^n$ by instead passing $\lambda_i \mathbf{1}$, where $\mathbf{1}$ is the all ones vector). Then $\rho = \sum_{i=1}^{n}$ is a linear permutation equivariant operation that can be exactly expressed by DeepSets, so the total error is within ε. The same argument applies when $\theta_i = h(\lambda_i)$ for some continuous function h.

For the basis invariant case, consider a parametric spectral graph convolution $f(V, \Lambda, X) = \sum_{i=1}^{n} h(\lambda_i) v_i v_i^\top X$. Note that if the eigenspace bases are $V_1, \ldots, V_l$ with eigenvalues $\mu_1, \ldots, \mu_l$, we can write the $f(V, \Lambda, X) = \sum_{i=1}^{l} h(\mu_j) V_j V_j^\top X$. Again, we will let $\rho = \sum_{i=1}^{l}$ be a sum function, which can be expressed exactly by DeepSets. Thus, it suffices to show that $h(\mu_j) V_j V_j^\top X$ can be ϵ/n approximated by a 2-IGN (i.e. an IGN that only uses vectors and matrices).

Note that since h is continuous, we can use an elementwise MLP (which IGNs can learn) to approximate $f_1(\mu \mathbf{1}\mathbf{1}^\top, VV^\top, X) = (h(\mu)\mathbf{1}\mathbf{1}^\top, VV^\top, X)$ to arbitrary precision (note that we represent the eigenvalue μ as a constant matrix $\mu \mathbf{1}\mathbf{1}^\top$). Also, since a 2-IGN can learn matrix vector multiplication (Cai and Wang [2022] Lemma 10), we can approximate $f_2(h(\mu)\mathbf{1}\mathbf{1}^\top, VV^\top, X) = (h(\mu)\mathbf{1}\mathbf{1}^\top, VV^\top X)$, as $V_i V_i^\top \in \mathbb{R}^{n^2}$ is a matrix and $X \in \mathbb{R}^{n \times d_{\text{feat}}}$ is a vector with respect to permutation symmetries. Finally, we use an elementwise MLP to approximate the scalar-vector multiplication $f_3(h(\mu)\mathbf{1}\mathbf{1}^\top, VV^\top, X) = h(\mu)VV^\top X$. Since $f_3 \circ f_2 \circ f_1(\mu \mathbf{1}\mathbf{1}^\top, VV^\top, X) = h(\mu)VV^\top X$, and since 2-IGNs universally approximate each f_i, applying Lemma 16 shows that a 2-IGN can approximate $h(\mu)VV^\top X$ to ϵ/n accuracy, so we are done. Since Expressive-BasisNet is stronger than BasisNet, it can also universally approximate these functions.

□

From the proof, we can see that SignNet and BasisNet need only learn simple functions for the ρ and ϕ when h is simple, or when the filter is non-parametric and we need only learn θ_i. Xu et al. [2020] propose the principle of algorithmic alignment, and show that if separate modules of a neural network each need only learn simple

functions (that is, functions that are well-approximated by low-order polynomials with small coefficients), then the network may be more sample efficient. If we do not require permutation equivariance, and parameterize SignNet and BasisNet with simple MLPs, then algorithmic alignment may suggest that our models are sample efficient. Indeed, $\rho = \sum$ is a simple linear function with coefficients 1, and $\phi(V, \lambda, X) = h(\lambda)VV^\top X$ is quadratic in V and linear in X, so it is simple if h is simple.

. *There exist infinitely many pairs of non-isomorphic graphs that SignNet and BasisNet can distinguish, but spectral graph convolutions or spectral GNNs cannot distinguish.*

Proof. The idea is as follows: we will take graphs G and give them the node feature matrix $X_G = D^{1/2}\mathbf{1}$, i.e. each node has as feature the square root of its degree. Then any spectral graph convolution (or, the first layer of any spectral GNN) will map $V\text{Diag}(\theta)V^\top X$ to something that only depends on the degree sequence and number of nodes. Thus, any spectral graph convolution or spectral GNN will have the same output (up to permutation) for any such graphs G with node features X_G and the same number of nodes and same degree sequence. On the other hand, SignNet and BasisNet can distinguish between infinitely many pairs of graphs $(G^{(1)}, G^{(2)})$ with node features $(X_{G^{(1)}}, X_{G^{(2)}})$ and the same number of nodes and degree sequence; this is because SignNet and BasisNet can tell when a graph is bipartite.

For each $n \geq 5$, we will define $G^{(1)}$ and $G^{(2)}$ as connected graphs with n nodes, with the same degree sequence. Also, we define $G^{(1)}$ to have node features $X_i^{(1)} = \sqrt{d_i^{(1)}}$, where $d_i^{(1)}$ is the degree of node i in $G^{(1)}$, and similarly $G^{(2)}$ has node features $X_i^{(2)} = \sqrt{d_i^{(2)}}$. Now, note that $X^{(1)}$ is an eigenvector of the normalized Laplacian of $G^{(1)}$, and it has eigenvalue 0. As we take the eigenvectors to be orthonormal (since the normalized Laplacian is symmetric), for any spectral graph convolution we have that

$$\sum_{i=1}^{n} \theta_i v_i v_i^\top X^{(1)} = \theta_1 v_1 v_1^\top X^{(1)} = \theta_1 D_1^{1/2}\mathbf{1}(D_1^{1/2}\mathbf{1})^\top D_1^{1/2}\mathbf{1} = \theta_1 \sum_{j=1}^{n}(d_j^{(1)})D_1^{1/2}\mathbf{1}. \quad \text{(E.36)}$$

Where D_1 is the diagonal degree matrix of $G^{(1)}$. Likewise, any spectral graph convolution outputs $\theta_1 \sum_j (d_j^{(2)})D_2^{1/2}\mathbf{1}$ for $G^{(2)}$. Since D_1 and D_2 are the same up to a

permutation, we have that any spectral graph convolution has the same output for $G^{(1)}$ and $G^{(2)}$, up to a permutation. In fact, this also holds for spectral GNNs, as the first layer will always have the same output (up to a permutation) on $G^{(1)}$ and $G^{(2)}$, so the latter layers will also have the same output up to a permutation.

Now, we concretely define $G^{(1)}$ and $G^{(2)}$. This is illustrated in Figure E-7 and Figure E-8. For $n = 5$, let $G^{(1)}$ contain a triangle with nodes w_1, w_2, w_3, and have a path of length 2 coming out of one of the nodes in the triangle, say w_1 connects to w_4, and w_4 connects to w_5. This is not bipartite, as there is a triangle. Let $G^{(2)}$ be a bipartite graph that has 2 nodes on the left (v_1, v_2) and 3 nodes on the right (v_3, v_4, v_5). Connect v_1 with all nodes on the right, and connect v_2 with v_3 and v_4.

Note that both $G^{(1)}$ and $G^{(2)}$ have the same number of nodes and the same degree sequence $\{3, 2, 2, 2, 1\}$. Thus, spectral graph convolutions or spectral GNNs cannot distinguish them. However, SignNet and BasisNet can distinguish them, as they can tell whether a graph is bipartite by checking the highest eigenvalue of the normalized Laplacian. This is because the multiplicity of the eigenvalue 2 is the number of bipartite components. In particular, SignNet can approximate the function $\phi(v_i, \lambda_i, X) = \lambda_i$ and $\rho \approx \max_{i=1}^{n}$. Likewise, BasisNet can approximate the function $\phi_{d_i}(V_i V_i^\top, \lambda_i) = \lambda_i$ and $\rho \approx \max_{i=1}^{l}$.

This in fact gives an infinite family of graphs that SignNet / BasisNet can distinguish, but spectral graph convolutions or spectral graph GNNs cannot. To see why, suppose we have $G^{(1)}$ and $G^{(2)}$ for some $n \geq 5$. Then we construct a pair of graphs on $n + 1$ nodes with the same degree sequence. To do this, we add another node to the path of $G^{(1)}$, thus giving it degree sequence $\{3, 2, \ldots, 2, 1\}$. For $G^{(2)}$, we add a node v_{n+1} to the side that v_n is not contained on (e.g. for $n = 5$, we add v_6 to the left side, as v_5 was on the right), then connect v_n to v_{n+1} to also give a degree sequence $\{3, 2, \ldots, 2, 1\}$. Note that the non-bipartiteness of $G^{(1)}$ and bipartiteness of $G^{(2)}$ are preserved.

□

E.8.2 Existing Positional Encodings

Here, we show that our SignNets and BasisNets universally approximate various types of existing graph positional encodings. The key is to show that these positional encodings are related to spectral graph convolution matrices and the diagonals of these matrices, and to show that our networks can approximate these matrices and diagonals.

Proposition 16. *If the eigenvalues take values in a compact set, SignNets and BasisNets universally approximate the diagonal of any spectral graph convolution matrix* $f(V,\Lambda) = \text{diag}\left(\sum_{i=1}^n h(\lambda_i) v_i v_i^\top\right)$. *BasisNets can additionally universally approximate any spectral graph convolution matrix* $f(V,\Lambda) = \sum_{i=1}^n h(\lambda_i) v_i v_i^\top$.

Proof. Note that the v_i come from a compact set as they are of unit norm. The λ_i are from a compact set by assumption; this assumption holds for the normalized Laplacian, as $\lambda_i \in [0,2]$. Also, as diag is linear, the spectral graph convolution diagonal can be written $\sum_{i=1}^n h(\lambda_i)\text{diag}(v_i v_i^\top)$.

Let $\epsilon > 0$. For SignNet, let $\rho = \sum_{i=1}^n$, which can be exactly expressed as it is a permutation equivariant linear operation from vectors to vectors. Then $\phi(v_i, \lambda_i)$ can approximate the function $\lambda_i \text{diag}(v_i v_i^\top)$ to arbitrary precision, as it is a permutation equivariant function from vectors to vectors [Segol and Lipman, 2019]. Thus, letting ϕ approximate the function to ϵ/n accuracy, SignNet can approximate f to ϵ accuracy.

Let l be the number of eigenspaces $V_1, \ldots, V_l$, so $f(V,\Lambda) = \sum_{i=1}^l h(\mu_i) V_i V_i^\top$. For BasisNet, we need only show that it can approximate the spectral graph convolution matrix to ϵ/l accuracy, as a 2-IGN can exactly express the diag function in each ϕ_{d_i}, since it is a linear permutation equivariant function from matrices to vectors. A 2-IGN can universally approximate the function $f_1(\mu_i, V_i V_i^\top) = (h(\mu_i), V_i V_i^\top)$, as it can express any elementwise MLP. Also, a 2-IGN can universally approximate the scalar-matrix multiplication $f_2(h(\mu_i), V_i V_i^\top) = h(\mu_i) V_i V_i^\top$ by another elementwise MLP. Since $h(\mu_i) V_i V_i^\top = f_2 \circ f_1(\mu_i, V_i V_i^\top)$, Lemma 16 shows that a single 2-IGN can approximate this composition to ϵ/l accuracy, so we are done.

□

. *SignNet and BasisNet can approximate node positional encodings based on heat kernels [Feldman et al., 2022] and random walks [Dwivedi et al., 2022]. BasisNet can approximate diffusion and p-step random walk relative positional encodings [Mialon et al., 2021], and generalized PageRank and landing probability distance encodings [Li et al., 2020].*

Proof. We will show that we can apply the above Proposition 16, by showing that all of these positional encodings are spectral graph convolutions. The heat kernel embeddings are of the form $\operatorname{diag}\left(\sum_{i=1}^{n} \exp(-t\lambda_i) v_i v_i^\top\right)$ for some choices of the parameter t, so they can be approximated by SignNets or BasisNets. Also, the diffusion kernel [Mialon et al., 2021] is just the matrix of this heat kernel, and the p-step random walk kernel is $\sum_{i=1}^{n}(1-\gamma\lambda_i)^p v_i v_i^\top$ for some parameter γ, so BasisNets can universally approximate both of these.

For the other positional encodings, we let v_i be the eigenvectors of the random walk Laplacian $I - D^{-1}A$ instead of the normalized Laplacian $I - D^{-1/2}AD^{-1/2}$. The eigenvalues of these two Laplacians are the same, and if $\tilde{v}_i$ is an eigenvector of the normalized Laplacian then $D^{-1/2}\tilde{v}_i$ is an eigenvector of the random walk Laplacian with the same eigenvalue [Von Luxburg, 2007].

Then with v_i as the eigenvectors of the random walk Laplacian, the random walk positional encodings (RWPE) in Dwivedi et al. [2022] take the form

$$\operatorname{diag}\left((D^{-1}A)^k\right) = \operatorname{diag}\left(\sum_{i=1}^{n}(1-\lambda_i)^k v_i v_i^\top\right), \tag{E.37}$$

for any choices of integer k.

The distance encodings proposed in Li et al. [2020] take the form

$$f_3(AD^{-1}, (AD^{-1})^2, (AD^{-1})^3, \ldots), \tag{E.38}$$

for some function f_3. We restrict to continuous f_3 here; shortest path distances can be obtained by a discontinuous f_3 that we discuss below. Their generalized PageRank

based distance encodings can be obtained by

$$\sum_{i=1}^{n} \left(\sum_{k \geq 1} \gamma_k (1 - \lambda_i)^k \right) v_i v_i^\top \tag{E.39}$$

for some $\gamma_k \in \mathbb{R}$, so this is a spectral graph convolution. They also define so-called landing probability based positional encodings, which take the form

$$\sum_{i=1}^{n} (1 - \lambda_i)^k v_i v_i^\top, \tag{E.40}$$

for some choices of integer k. Thus, BasisNets can approximate these distance encoding matrices. □

Another powerful class of positional encodings is based on shortest path distances between nodes in the graph [Ying et al., 2021, Li et al., 2020]. Shortest path distances can be expressed in a form similar to the spectral graph convolution, but require a highly discontinuous function. If we define $f_3(x_1, \ldots, x_n) = \min_{i:x_i \neq 0} i$ to be the lowest index such that x_i is nonzero, then we can write the shortest path distance matrix as $f_3(D^{-1}A, (D^{-1}A)^2, \ldots, (D^{-1}A)^n)$, where f_3 is applied elementwise to return an $n \times n$ matrix. As $(D^{-1}A)^k = \sum_{i=1}^{n} (1 - \lambda_i)^k v_i v_i^\top$, BasisNets can learn the inside arguments, but cannot learn the discontinuous function f_3.

E.8.3 Spectral Invariants

Here, we consider the graph angles $\alpha_{ij} = \|V_i V_i^\top e_j\|_2$, for $i = 1, \ldots, l$ where l is the number of eigenspaces, and $j = 1, \ldots, n$. It is clear that graph angles are permutation equivariant and basis invariant. These graph angles have been extensively studied, so we cite a number of interesting properties of them. That graph angles determine the number of length 3, 4 and 5 cycles, the connectivity of a graph, and the number of length k closed walks is all shown in Chapter 4 of Cvetković et al. [1997]. Other properties may be of use for graph representation learning as well. For instance, the eigenvalues of node-deleted subgraphs of a graph $\mathcal{G}$ are determined by the eigenvalues

and graph angles of $\mathcal{G}$; this may be useful in extending recent graph neural networks that are motivated by node deletion and the reconstruction conjecture [Cotta et al., 2021, Bevilacqua et al., 2022, Papp et al., 2021, Tahmasebi et al., 2020].

Now, we prove that BasisNet can universally approximate the graph angles. The graph properties we consider in the theorem are all integer valued (e.g. the number of cycles of length 3 in a graph is an integer). Thus, any two graphs that differ in these properties will differ by at least 1, so as long as we have approximation to $\varepsilon < 1/2$, we can distinguish any two graphs that differ in these properties. Recall the statement of Theorem 8.

Theorem 8. *BasisNet can universally approximate the graph angles α_{ij}. The eigenvalues and graph angles (and thus BasisNets) can determine the number of length 3, 4, and 5 cycles, whether a graph is connected, and the number of length k closed walks from any vertex to itself.*

Proof. Note that the graph angles satisfy

$$\alpha_{ij} = \|V_i V_i^\top e_j\|_2 = \sqrt{e_j^\top V_i V_i^\top V_i V_i^\top e_j} = \sqrt{e_j^\top V_i V_i^\top e_j}, \tag{E.41}$$

where V_i is a basis for the ith adjacency matrix eigenspace, and $e_j^\top V_i V_i^\top e_j$ is the (j,j)-entry of $V_i V_i^\top$. These graph angles are just the elementwise square roots of the diagonals of the matrices $V_i V_i^\top$. As $f_1(V_i V_i^\top) = \mathrm{diag}(V_i V_i^\top)$ is a permutation equivariant linear function from matrices to vectors, 2-IGN on $V_i V_i^\top$ can exactly compute this with 0 error. Then a 2-IGN can learn an elementwise MLP to approximate the elementwise square root $f_2(\mathrm{diag}(V_i V_i^\top)) = \sqrt{\mathrm{diag}(V_i V_i^\top)}$ to arbitrary precision. Finally, there may be remaining operations f_3 that are permutation invariant or permutation equivariant from vectors to vectors; for instance, the α_{ij} are typically gathered into a matrix of size $l \times n$ where the columns are lexicographically sorted (l is the number of eigenspaces) [Cvetković et al., 1997], or we may have a permutation invariant readout to compute a subgraph count. A DeepSets can approximate f_3 without any higher order tensors besides vectors [Zaheer et al., 2017, Segol and Lipman, 2019].

As 2-IGNs can approximate each f_i individually, a single 2-IGN can approximate

$f_3 \circ f_2 \circ f_1$ by Lemma 16. Also, since the graph properties considered in the theorem are integer-valued, BasisNet can distinguish any two graphs that differ in one of these properties. □

To see that message passing graph neural networks (MPNNs) cannot determine these quantities, we use the fact that MPNNs cannot distinguish between two graphs that have the same number of nodes and where each node (in both graphs) has the same degree. For $k \geq 3$, let C_k denote the cycle graph of size k, and $C_k + C_k$ denote the graph that is the union of two disjoint cycle graphs of size k. MPNNs cannot distinguish between C_{2k} and $C_k + C_k$ for $k \geq 3$, because they have the same number of nodes, and each node has degree 2. Thus, MPNNs cannot tell whether a graph is connected, as C_{2k} is but $C_k + C_k$ is not. Also, it cannot count the number of 3, 4, or 5 cycles, as $C_k + C_k$ has two k cycles while C_{2k} has no k cycles. Likewise, any node in $C_k + C_k$ has more length k closed walks than any node in C_{2k}. This is because any length k closed walk in C_{2k} has an analogous closed walk in $C_k + C_k$, but the nodes in $C_k + C_k$ also have a closed walk that completely goes around a cycle.

E.9 Useful Lemmas

In this section, we collect useful lemmas for our proofs. These lemmas generally only require basic tools to prove. Our first lemma is a crucial property of quotient spaces.

Lemma 11 (Passing to the quotient)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be topological spaces, and let $\mathcal{X}/G$ be a quotient space, with corresponding quotient map π. Then for every continuous G-invariant function $f : \mathcal{X} \to \mathcal{Y}$, there is a unique continuous $\tilde{f} : \mathcal{X}/G \to \mathcal{Y}$ such that $f = \tilde{f} \circ \pi$.*

Proof. For $z \in \mathcal{X}/G$, by surjectivity of π we can choose an $x_z \in \mathcal{X}$ such that $\pi(x_z) = z$. Define $\tilde{f} : \mathcal{X}/G \to \mathcal{Y}$ by $\tilde{f}(z) = f(x_z)$. This is well-defined, since if $\pi(x_z) = \pi(x)$ for any other $x \in \mathcal{X}$, then $gx_z = x$ for some $g \in G$, so

$$f(x) = f(gx_z) = f(x_z) = \tilde{f}(z), \tag{E.42}$$

where the second equality uses the G-invariance of f. Note that $\tilde{f}$ is continuous by the universal property of quotient spaces. Also, $\tilde{f}$ is the unique function such that $f = \tilde{f} \circ \pi$; if there were another function $h : \mathcal{X}/G \to \mathcal{Y}$ with $h(z) \neq \tilde{f}(z)$, then $h(z) \neq f(x_z)$, so $h(\pi(x_z)) = h(z) \neq f(x_z)$. □

Next, we give the First Fundamental Theorem of $O(d)$, a classical result that has been recently used for machine learning by Villar et al. [2021]. This result shows that an orthogonally invariant $f(V)$ can be expressed as a function $h(VV^\top)$. We give a proof that if f is continuous, then h is also continuous.

Lemma 12 (First Fundamental Theorem of $O(d)$)**.** *A continuous function $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{d_{\text{out}}}$ is orthogonally invariant, i.e. $f(VQ) = f(V)$ for all $Q \in O(d)$, if and only if $f(V) = h(VV^\top)$ for some continuous h.*

Proof. If $f(V) = h(VV^\top)$, then we have $f(VQ) = h(VQQ^\top V^\top) = h(VV^\top)$ so f is orthogonally invariant.

For the other direction, invariant theory shows that the $O(d)$ invariant polynomials are generated by the inner products $v_i^\top v_j$, where $v_i \in \mathbb{R}^d$ are the rows of V [Kraft and Procesi, 1996]. Let $p : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times n}$ be the map $p(V) = VV^\top$. Then González and de Salas [2003] Lemma 11.13 shows that the quotient space $\mathbb{R}^{n\times d}/O(d)$ is homeomorphic to a closed subset $p(\mathbb{R}^{n\times d}) = \mathcal{Z} \subseteq \mathbb{R}^{n\times n}$. Let $\tilde{p}$ refer to this homeomorphism, and note that $\tilde{p} \circ \pi = p$ by passing to the quotient (Lemma 11). Then any continuous $O(d)$ invariant f passes to a unique continuous $\tilde{f} : \mathbb{R}^{n\times d}/O(d) \to \mathbb{R}^{d_{\text{out}}}$ (Lemma 11), so $f = \tilde{f} \circ \pi$ where π is the quotient map. Define $h : \mathcal{Z} \to \mathbb{R}^{d_{\text{out}}}$ by $h = \tilde{f} \circ \tilde{p}^{-1}$, and note that h is a composition of continuous functions and hence continuous. Finally, we have that $h(VV^\top) = h(\tilde{p} \circ \pi(V)) = \tilde{f} \circ \pi(V) = f(V)$, so we are done. □

The next lemma allows us to decompose a quotient of a product space into a product of smaller quotient spaces.

Lemma 13. *Let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ be topological spaces and $G_1, \ldots, G_k$ be topological groups such that each G_i acts continuously on $\mathcal{X}_i$. Denote the quotient maps by $\pi_i : \mathcal{X}_i \to$*

$\mathcal{X}_i/G_i$. *Then the quotient of the product is the product of the quotient, i.e.*

$$(\mathcal{X}_1 \times \ldots \times \mathcal{X}_k)/(G_1 \times \ldots \times G_k) \cong (\mathcal{X}_1/G_1) \times \ldots \times (\mathcal{X}_k/G_k), \tag{E.43}$$

and $\pi_1 \times \ldots \times \pi_k : \mathcal{X}_1 \times \ldots \mathcal{X}_k \to (\mathcal{X}_1/G_1) \times \ldots \times (\mathcal{X}_k/G_k)$ *is quotient map.*

Proof. First, we show that $\pi_1 \times \ldots \times \pi_k$ is a quotient map. This is because 1. the quotient map of any continuous group action is an open map, so each π_i is an open map, 2. the product of open maps is an open map, so $\pi_1 \times \ldots \times \pi_k$ is an open map and 3. a continuous surjective open map is a quotient map, so $\pi_1 \times \ldots \times \pi_k$, which is continuous and surjective, is a quotient map.

Now, we need only apply the theorem of uniqueness of quotient spaces to show (E.43) (see e.g. Lee [2013], Theorem A.31). Letting $q : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to (\mathcal{X}_1 \times \ldots \times \mathcal{X}_k)/(G_1 \times \ldots \times G_k)$ denote the quotient map for this space, it is easily seen that $q(x_1, \ldots, x_k) = q(y_1 \ldots, y_k)$ if and only if $\pi_1 \times \ldots \times \pi_k(x_1, \ldots, x_k) = \pi_1 \times \ldots \times \pi_k(y_1, \ldots, y_k)$, since either of these is true if and only if there exist $g_i \in G_i$ such that $x_i = g_i y_i$ for each i. Thus, we have an isomorphism of these quotient spaces. □

The following lemma shows that quotients of compact spaces are also compact, which is useful for universal approximation on quotient spaces.

Lemma 14 (Compactness of quotients of compact spaces)**.** *Let* $\mathcal{X}$ *be a compact space. Then the quotient space* $\mathcal{X}/G$ *is compact.*

Proof. Denoting the quotient map by $\pi : \mathcal{X} \to \mathcal{X}/G$ and letting $\{U_\alpha\}_\alpha$ be an open cover of $\mathcal{X}/G$, we have that $\{\pi^{-1}(U_\alpha)\}_\alpha$ is an open cover of $\mathcal{X}$. By compactness of $\mathcal{X}$, we can choose a finite subcover $\{\pi^{-1}(U_{\alpha_i})\}_{i=1,\ldots,n}$. Then $\{\pi(\pi^{-1}(U_{\alpha_i}))\}_{i=1,\ldots,n} = \{U_{\alpha_i}\}_{i=1,\ldots,n}$ by surjectivity, and $\{U_{\alpha_i}\}_{i=1,\ldots,n}$ is thus an open cover of $\mathcal{X}/G$. □

The Whitney embedding theorem gives a nice condition that we apply to show that the quotient spaces $\mathcal{X}/G$ that we deal with embed into Euclidean space. It says that when $\mathcal{X}/G$ is a smooth manifold, then it can be embedded into a Euclidean space of double the dimension of the manifold. The proof is outside the scope of this paper.

Lemma 15 (Whitney Embedding Theorem [Whitney, 1944]). *Every smooth manifold $\mathcal{M}$ of dimension $n > 0$ can be smoothly embedded in $\mathbb{R}^{2n}$.*

Finally, we give a lemma that helps prove universal approximation results. It says that if functions f that we want to approximate can be written as compositions $f = f_L \circ \ldots \circ f_1$, then it suffices to universally approximate each f_i and compose the results to universally approximate the f. This is especially useful for proving universality of neural networks, as we may use some layers to approximate each f_i, then compose these layers to approximate the target function f.

Lemma 16 (Layer-wise universality implies universality). *Let $\mathcal{Z} \subseteq \mathbb{R}^{d_0}$ be a compact domain, let $\mathcal{F}_1, \ldots, \mathcal{F}_L$ be families of continuous functions where $\mathcal{F}_i$ consists of functions from $\mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ for some $d_1, \ldots, d_L$. Let $\mathcal{F}$ be the family of functions $\{f_L \circ \ldots f_1 : \mathcal{Z} \to \mathbb{R}^{d_L}, f_i \in \mathcal{F}_i\}$ that are compositions of functions $f_i \in \mathcal{F}_i$.*

For each i, let Φ_i be a family of continuous functions that universally approximates $\mathcal{F}_i$. Then the family of compositions $\Phi = \{\phi_L \circ \ldots \circ \phi_1 : \phi_i \in \Phi_i\}$ universally approximates $\mathcal{F}$.

Proof. Let $f = f_L \circ \ldots \circ f_1 \in \mathcal{F}$. Let $\tilde{\mathcal{Z}}_1 = \mathcal{Z}$, and then for $i \geq 2$ let $\tilde{\mathcal{Z}}_i = f_{i-1}(\tilde{\mathcal{Z}}_{i-1})$. Then each $\tilde{\mathcal{Z}}_i$ is compact by continuity of the f_i. For $1 \leq i < L$, let $\mathcal{Z}_i = \tilde{\mathcal{Z}}_i$, and for $i = L$ let $\mathcal{Z}_L$ be a compact set containing $\tilde{\mathcal{Z}}_L$ such that every ball of radius one centered at a point in $\tilde{\mathcal{Z}}_L$ is still contained in $\mathcal{Z}_L$.

Let $\epsilon > 0$. We will show that there is a $\phi \in \Phi$ such that $\|f - \phi\|_\infty < \epsilon$ by induction on L. This holds trivially for $L = 1$, as then $\Phi = \Phi_1$.

Now, let $L \geq 2$, and suppose it holds for $L - 1$. By universality of Φ_L, we can choose a $\phi_L : \mathcal{Z}_L \to \mathbb{R}^{d_L} \in \Phi_L$ such that $\|\phi_L - f_L\|_\infty < \epsilon/2$. As ϕ_L is continuous on a compact domain, it is also uniformly continuous, so we can choose a $\tilde{\delta} > 0$ such that $\|y - z\|_2 < \tilde{\delta} \implies \|\phi_L(y) - \phi_L(z)\|_2 < \epsilon/2$.

Let $\delta = \min(\tilde{\delta}, 1)$. By induction, we can choose $\phi_{L-1} \circ \ldots \circ \phi_1, \phi_i \in \Phi_i$ such that

$$\|\phi_{L-1} \circ \ldots \circ \phi_1 - f_{L-1} \circ \ldots \circ f_1\|_\infty < \delta. \tag{E.44}$$

Note that $\phi_{L-1} \circ \ldots \circ \phi_1(\mathcal{Z}) \subseteq \mathcal{Z}_L$, because for each $x \in \mathcal{Z}$, $\phi_{L-1} \circ \ldots \circ \phi_1(x)$ is

within $\delta \leq 1$ Euclidean distance to $f_{L-1} \circ \ldots \circ f_1(x) \in \tilde{\mathcal{Z}}_L$, so it is contained in $\mathcal{Z}_L$ by construction. Thus, we may define $\phi = \phi_L \circ \ldots \circ \phi_1 : \mathcal{Z} \to \mathbb{R}^{d_L}$, and compute that

$$\|\phi - f\|_\infty \leq \|\phi - \phi_L \circ f_{L-1} \circ \ldots \circ f_1\|_\infty + \|\phi_L \circ f_{L-1} \circ \ldots \circ f_1 - f\|_\infty \tag{E.45}$$

$$< \|\phi - \phi_L \circ f_{L-1} \circ \ldots \circ f_1\|_\infty + \epsilon/2, \tag{E.46}$$

since $\|\phi_L - f_L\|_\infty < \epsilon/2$. To bound this other term, let $x \in \mathcal{Z}$, and for $y = \phi_{L-1} \circ \ldots \circ \phi_1(x)$ and $z = f_{L-1} \circ \ldots \circ f_1(x)$, we know that $\|y - z\|_2 < \delta$, so $\|\phi_L(y) - \phi_L(z)\|_2 < \epsilon/2$ by uniform continuity. As this holds for all x, we have $\|\phi - \phi_L \circ f_{L-1} \circ \ldots \circ f_1\|_\infty \leq \epsilon/2$, so $\|\phi - f\|_\infty < \epsilon$ and we are done. □

E.10 Further Experiments

E.10.1 Graph Regression with no Edge Features

Table E.4: Results on the ZINC dataset with 500k parameter budget and no edge features. Numbers are the mean and standard deviation over 4 runs each with different seeds.

Base model	Positional encoding	k	#params	Test MAE (↓)
GIN	No PE	16	497k	$0.348_{\pm 0.014}$
	LapPE (flip)	16	498k	$0.341_{\pm 0.011}$
	SignNet	16	500k	$\mathbf{0.238_{\pm 0.012}}$
GAT	No PE	16	501k	$0.464_{\pm 0.011}$
	LapPE (flip)	16	502k	$0.462_{\pm 0.013}$
	SignNet	16	499k	$\mathbf{0.243_{\pm 0.008}}$

All graph regression models in Table 7.1 use edge features for learning and inference. To show that SignNet is also useful when no edge features are available, we ran ZINC experiments without edge features as well. The results are displayed in Table E.4. In this setting, SignNet still significantly improves the performance over message passing

networks without positional encodings, and over Laplacian positional encodings with sign flipping data augmentation.

E.10.2 Comparison with Domain Specific Molecular Graph Regression Models

Table E.5: Comparison with domain specific methods on graph-level regression tasks. Numbers are test MAE, so lower is better. Best models within a standard deviation are bolded.

	ZINC (10K) ↓	ZINC-full ↓
HIMP † [Fey et al., 2020]	$.151_{\pm.006}$	$.036_{\pm.002}$
CIN-small † [Bodnar et al., 2021]	$.094_{\pm.004}$	$.044_{\pm.003}$
CIN † [Bodnar et al., 2021]	$\mathbf{.079}_{\pm.006}$	$\mathbf{.022}_{\pm.002}$
SignNet (ours)	$\mathbf{.084}_{\pm.006}$	$\mathbf{.024}_{\pm.003}$

In Table E.5, we compare our model against methods that have domain-specific information about molecules built into them: HIMP [Fey et al., 2020] and CIN [Bodnar et al., 2021]. We see that SignNet is better than HIMP and CIN-small on these tasks, and is within a standard deviation of CIN. The SignNet models are the same as the ones reported in Table 7.2. Once again, we emphasize that SignNet is domain-agnostic.

E.10.3 Learning Spectral Graph Convolutions

To numerically test the ability of our basis invariant networks for learning spectral graph convolutions, we follow the experimental setups of Balcilar et al. [2020], He et al. [2021]. We take the dataset of 50 images in He et al. [2021] (originally from the Image Processing Toolbox of MATLAB), and resize them from 100×100 to 32×32. Then we apply the same spectral graph convolutions on them as in He et al. [2021], and train neural networks to learn these as regression targets. As in prior work, we report sum of squared errors on the training set to measure expressivity.

We compare against message passing GNNs [Kipf and Welling, 2017, Veličković et al., 2018] and spectral GNNs [Chien et al., 2021, Bianchi et al., 2021, Defferrard et al.,

Table E.6: Sum of squared errors for spectral graph convolution regression (with no test set). Lower is better. Numbers are mean and standard deviation over 50 images from He et al. [2021].

	Low-pass	High-pass	Band-pass	Band-rejection	Comb
GCN	$.111_{\pm.068}$	$3.092_{\pm5.11}$	$1.720_{\pm3.15}$	$1.418_{\pm1.03}$	$1.753_{\pm1.17}$
GAT	$.113_{\pm.065}$	$.954_{\pm.696}$	$1.105_{\pm.964}$	$.543_{\pm.340}$	$.638_{\pm.446}$
GPR-GNN	$.033_{\pm.032}$	$.012_{\pm.007}$	$.137_{\pm.081}$	$.256_{\pm.197}$	$.369_{\pm.460}$
ARMA	$.053_{\pm.029}$	$.042_{\pm.024}$	$.107_{\pm.039}$	$.148_{\pm.089}$	$.202_{\pm.116}$
ChebNet	$.003_{\pm.002}$	$\mathbf{.001}_{\pm.001}$	$.005_{\pm.003}$	$.009_{\pm.006}$	$.022_{\pm.016}$
BernNet	$\mathbf{.001}_{\pm.002}$	$\mathbf{.001}_{\pm.001}$	$\mathbf{.000}_{\pm.000}$	$.048_{\pm.042}$	$.027_{\pm.019}$
Transformer	$3.662_{\pm1.97}$	$3.715_{\pm1.98}$	$1.531_{\pm1.30}$	$1.506_{\pm1.29}$	$3.178_{\pm1.93}$
Transformer Eig Flip	$4.454_{\pm2.32}$	$4.425_{\pm2.38}$	$1.651_{\pm1.53}$	$2.567_{\pm1.73}$	$3.720_{\pm1.94}$
Transformer Eig Abs	$2.727_{\pm1.40}$	$3.172_{\pm1.61}$	$1.264_{\pm.788}$	$1.445_{\pm.943}$	$2.607_{\pm1.32}$
DeepSets SignNet	$.004_{\pm.013}$	$.086_{\pm.405}$	$.021_{\pm.115}$	$.008_{\pm.037}$	$\mathbf{.003}_{\pm.016}$
Transformer SignNet	$.003_{\pm.016}$	$.004_{\pm.025}$	$.001_{\pm.004}$	$.006_{\pm.023}$	$.093_{\pm.641}$
DeepSets BasisNet	$.009_{\pm.018}$	$.003_{\pm.015}$	$.008_{\pm.030}$	$\mathbf{.004}_{\pm.011}$	$.015_{\pm.060}$
Transformer BasisNet	$.079_{\pm.471}$	$.014_{\pm.038}$	$.005_{\pm.018}$	$.006_{\pm.016}$	$.014_{\pm.051}$

2016, He et al., 2021]. Also, we consider standard Transformers with only node features, with eigenvectors and sign flip augmentation, and with absolute values of eigenvectors. These models are all approximately sign invariant (they either use eigenvectors in a sign invariant way or do not use eigenvectors). We use DeepSets [Zaheer et al., 2017] in SignNet and 2-IGN [Maron et al., 2018] in BasisNet for ϕ, use a DeepSets for ρ in both cases, and then feed the features into another DeepSets or a standard Transformer [Vaswani et al., 2017b] to make the final predictions. That is, we are only given graph information through the eigenvectors and eigenvalues, and we do not use message passing.

Table E.6 displays the results, which validate our theoretical results in Section 7.2.1. Without any message passing, SignNet and BasisNet allow DeepSets and Transformers to perform strongly, beating the spectral GNNs GPR-GNN and ARMA on all tasks. Also, our networks outperform all other methods on the band-rejection and comb filters, and are mostly close to the best model on the other filters.

Runtimes. In these experiments, for 100 epochs on the same machine, GCN takes .435 seconds, ChebNet takes .675, DeepSets SignNet takes 3.741 seconds, and DeepSets BasisNet takes 6.196 seconds. SignNet and BasisNet are significantly more expensive than the GNN methods in this experiment, in large part because we use all 1024 eigenvectors of the 1024 node graph here. In contrast, recall that in Section 7.3.1, SignNet does not have much overhead over base GNNs on the task with smaller molecular graphs.

E.11 Further Experimental Details

E.11.1 Hardware, Software, and Data Details

All experiments could fit on one GPU at a time. Most experiments were run on a server with 8 NVIDIA RTX 2080 Ti GPUs. We run all of our experiments in Python, using the PyTorch [Paszke et al., 2019] framework (license URL). We also make use of Deep Graph Library (DGL) [Wang et al., 2019] (Apache License 2.0), and PyTorch Geometric (PyG) [Fey and Lenssen, 2019] (MIT License) for experiments with graph data.

The data we use are all freely available online. The datasets we use are ZINC [Irwin et al., 2012], Alchemy [Chen et al., 2019a], the synthetic counting substructures dataset [Chen et al., 2020g], the multi-task graph property regression synthetic dataset [Corso et al., 2020] (MIT License), the images dataset used by Balcilar et al. [2020] (GNU General Public License v3.0), the cat mesh from `free3d.com/3d-model/cat-v1--522281.html` (Personal Use License), and the human mesh from `turbosquid.com/3d-models/water-park-slides-3d-max/1093267` (TurboSquid 3D Model License). If no license is listed, this means that we cannot find a license for the dataset. As they appear to be freely available with permissive licenses or no licenses, we do not ask for permission from the creators or hosts of the data.

We do not believe that any of this data contains offensive content or personally identifiable information. The 50 images used in the spectral graph convolution

experiments are mostly images of objects, with a few low resolution images of humans that do not appear to have offensive content. The only other human-related data appears to be the human mesh, which appears to be from a 3D scan of a human.

E.11.2 Graph Regression Details

ZINC. In Section 7.3.1 we study the effectiveness of SignNet for learning positional encodings to boost the expressive power, and thereby generalization, on the graph regression problem ZINC. In all cases we take our ϕ encoder to be an 8 layer GIN with ReLU activation. The input eigenvector $v_i \in \mathbb{R}^n$, where n is the number of nodes in the graph, is treated as a single scalar feature for each node. In the case of using a fixed number of eigenvectors k, the aggregator ρ is taken to be an 8 layer MLP with batch normalization and ReLU activation. The aggregator ρ is applied separately to the concatenatation of the k different embeddings for each node in a graph, resulting in one single embedding per node. This embedding is concatenated to the node features for that node, and the result passed as input to the base (predictor) model. We also consider using all available eigenvectors in each graph instead of a fixed number k. Since the total number of eigenvectors is a variable quantity, equal to the number of nodes in the underlying graph, an MLP cannot be used for ρ. To handle the variable sized input in this case, we take ρ to be an MLP preceded by a sum over the ϕ outputs. In other words, the SignNet is of the form MLP $\left(\sum_{i=1}^{k} \phi(v_i) + \phi(-v_i)\right)$ in this case.

As well as testing SignNet, we also checked whether simple transformations that resolve the sign ambiguity of the Laplacian eigenvectors $p = (v_1, \ldots, v_k)$ could serve as effective positional encoding. We considered three options. First is to randomly flip the sign of each $\pm v_i$ during training. This is a common heuristic used in prior work on Laplacian positional encoding [Kreuzer et al., 2021, Dwivedi et al., 2020]. Second, take the element-wise absolute value $|v_i|$. This is a non-injective map, creating sign invariance at the cost of destroying positional information. Third is a different canonicalization that avoids stochasticity and use of absolute values by selecting the sign of each v_i so that the majority of entries are non-negative, with ties broken by comparing the ℓ_1-norm of positive and negative parts. When the tie-break also fails,

the sign is chosen randomly. Results for GatedGCN base model on ZINC in Table 7.1 show that all three of these approaches are significantly poorer positional encodings compared to SignNet.

Our training pipeline largely follows that of Dwivedi et al. [2022], and we use the GatedGCN and PNA base models from the accompanying implementation (see `https://github.com/vijaydwivedi75/gnn-lspe`). The Sparse Transformer base model architecture we use, which like GAT computes attention only across neighbouring nodes, is introduced by Kreuzer et al. [2021]. Finally, the GINE implementation is based on the PyTorch Geometric implementation [Fey and Lenssen, 2019]. For the state-of-the-art comparison, all baseline results are from their respective papers, except for GIN, which we run.

We used edge features for all models except the Sparse Transformer. For the Sparse Transformer, we found our method of using edge features to somewhat increase training instability, so standard deviation was higher, though mean test MAE was mostly similar to the runs without edge features.

ZINC-full. We also run our method on the full ZINC dataset, termed ZINC-full. The result we report for SignNet is a larger version of the GatedGCN base model with a SignNet that takes in all eigenvectors. This model has 994,113 parameters in total. All baseline results are from their respective papers, except for GIN, which is from [Bodnar et al., 2021].

Alchemy. We run our method and compare with the state-of-the-art on Alchemy (with 10,000 training graphs). We use the same data split as Morris et al. [2020b]. Our base model is a GIN that takes in edge features (i.e. a GINE). The SignNet consists of GIN for ϕ and a Transformer for ρ, as in the counting substructures and graph property regression experiments in Section 7.3.2. The model has 907,371 parameters in total. Our training setting is very similar to that of Morris et al. [2022], as we build off of their code. We train with an Adam optimizer [Kingma and Ba, 2014] with a starting learning rate of .001, and a minimum learning rate of .000001. The learning rate schedule cuts the learning rate in half with a patience of 20 epochs, and training ends when we reach the minimum learning rate. All baseline results are from their

respective papers, except for GIN, which is from [Morris et al., 2022].

E.11.3 Spectral Graph Convolution Details

In Appendix E.10.3, we conduct node regression experiments for learning spectral graph convolutions. The experimental setup is mostly taken from He et al. [2021]. However, we resize the 100×100 images to 32×32. Thus, each image is viewed as a 1024-node graph. The node features $X \in \mathbb{R}^n$ are the grayscale pixel intensities of each node. Just as in He et al. [2021], we only train and evaluate on nodes that are not connected to the boundary of the grid (that is, we only evaluate on the 28×28 middle section). For all experiments we limit each model to 50,000 parameters. We use the Adam [Kingma and Ba, 2014] optimizer for all experiments. For each of the GNN baselines (GCN, GAT, GPR-GNN, ARMA, ChebNet, BernNet), we select the best performing out of 4 hyperparameter settings: either 2 or 4 convolution layers, and a hidden dimension of size 32 or D, where D is just large enough to stay with 50,000 parameters (for instance, $D = 128$ for GCN, GPR-GNN, and BernNet).

We use DeepSets or standard Transformers as our prediction network. This takes in the output of SignNet or BasisNet and concatenates it with the node features, then outputs a scalar prediction for each node. We use a 3 layer output network for DeepSets SignNet, and 2 layer output networks for all other configurations. All networks use ReLU activations.

For SignNet, we use DeepSets for both ϕ and ρ. Our ϕ takes in eigenvectors only, then our ρ takes the outputs of ϕ and the eigenvalues. We use three layers for ϕ and ρ.

For BasisNet, we use the same DeepSets for ρ as in SignNet, and 2-IGNs for the ϕ_{d_i}. There are three distinct multiplicities for the grid graph (1, 2, and 32), so we only need 3 separate IGNs. Each IGN consists of an $\mathbb{R}^{n^2 \times 1} \to \mathbb{R}^{n \times d'}$ layer and two $\mathbb{R}^{n \times d''} \to \mathbb{R}^{n \times d'''}$ layers, where the d' are hidden dimensions. There are no matrix to matrix operations used, as the memory requirements are intensive for these ≥ 1000 node graphs. The ϕ_{d_i} only take in $V_i V_i^\top$ from the eigenspaces, and the ρ takes the output of the ϕ_{d_i} as well as the eigenvalues.

E.11.4 Substructures and Graph Properties Regression Details

We use the random graph dataset from Chen et al. [2020g] for counting substructures and the synthetic dataset from Corso et al. [2020] for regressing graph properties. For fair comparison we fix the base model as a 4-layer GIN model with hidden size 128. We choose ϕ as a 4-layer GIN (independently applied to every eigenvector) and ρ as a 1-layer Transformer (independently applied to every node). Combined with proper batching and masking, we have a SignNet that takes Laplacian eigenvectors $V \in \mathbb{R}^{n \times n}$ and outputs fixed size sign-invariant encoding node features $f(V, \Lambda, X) \in \mathbb{R}^{n \times d}$, where n varies between graphs but d is fixed. We use this SignNet in our experiments and compare with other methods of handling PEs.

E.11.5 Texture Reconstruction Details

Table E.7: Parameter settings for the texture reconstruction experiments.

	Params	Base MLP width	Base MLP layers	ϕ out dim	ρ out dim	ρ, ϕ width
Intrinsic NF	328,579	128	6	—	—	—
SignNet	323,563	108	6	4	64	8

We closely follow the experimental setting of Koestler et al. [2022] for the texture reconstruction experiments. In this work, we use the cotangent Laplacian [Rustamov et al., 2007] of a triangle mesh with the lowest 1023 eigenvectors besides the trivial eigenvector of eigenvalue 0. We implemented SignNet in the authors' original code, which was privately shared with us. Both ρ and ϕ are taken to be MLPs. Hyperparameter settings and number of parameters are given in Table E.7. We chose hyperparameters so that the total number of parameters in the SignNet model was no larger than that of the original model.

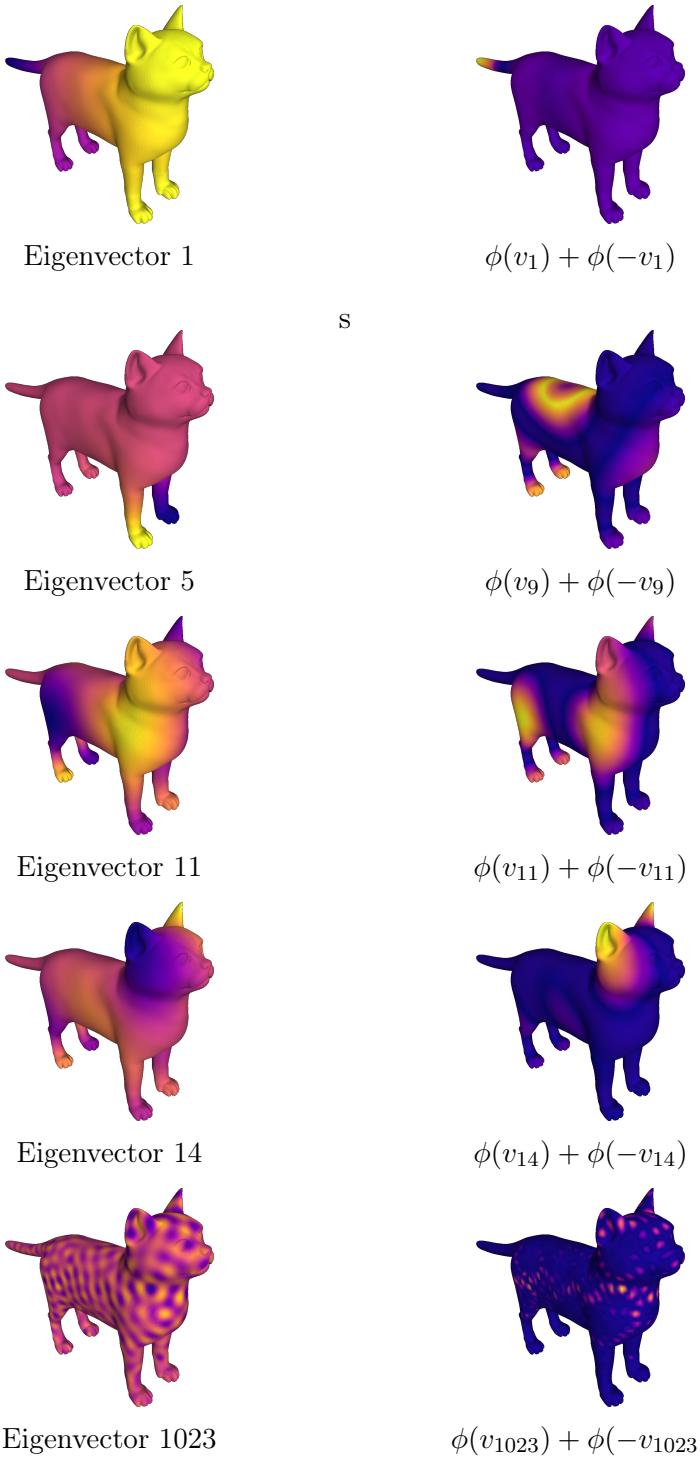


Figure E-2: (Left) Cotangent Laplacian eigenvectors of the cat model. (Right) First principal component of $\phi(v) + \phi(-v)$ from our trained SignNet.

Figure E-3: First three principal components of the full SignNet output on the cat model.

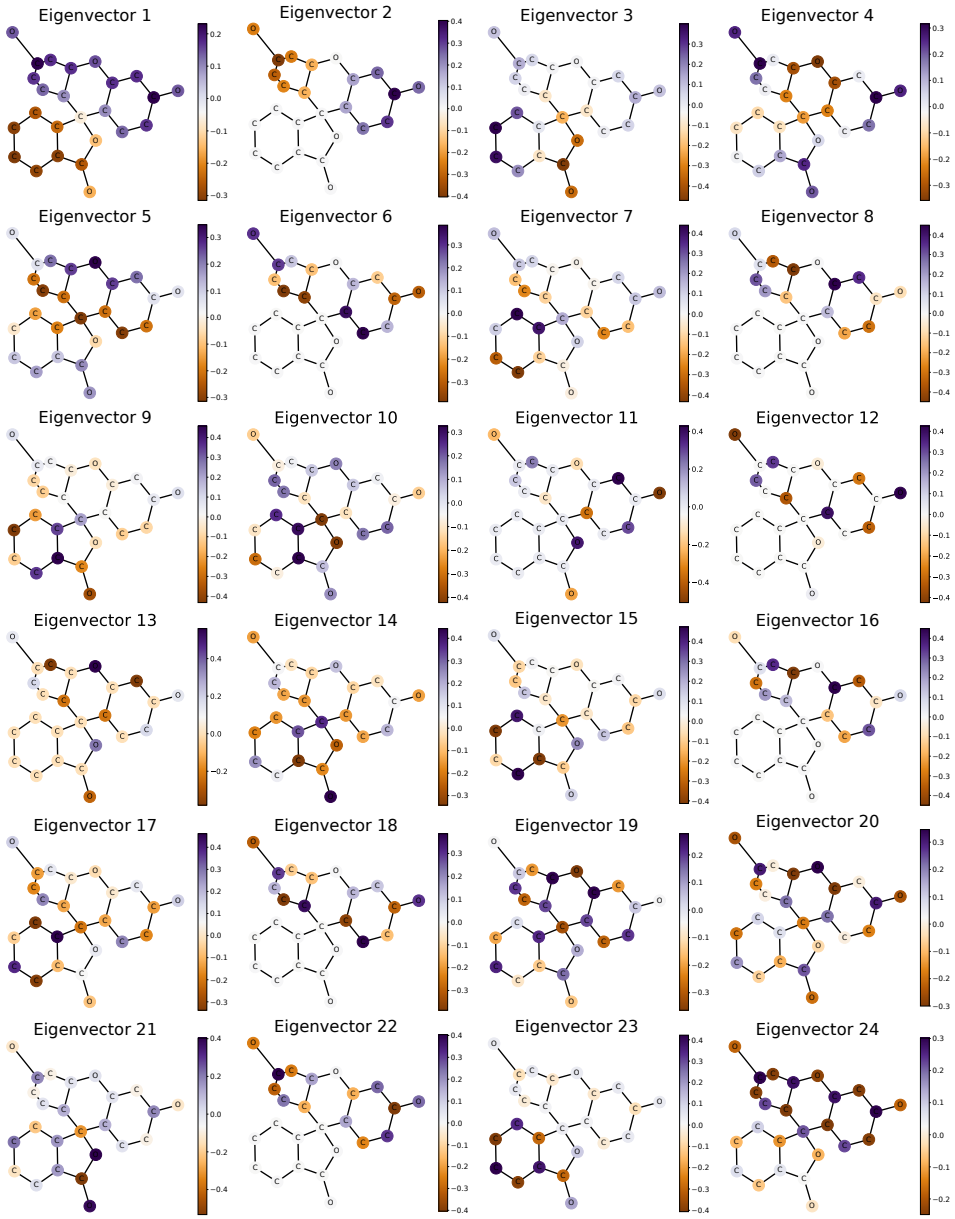


Figure E-4: All normalized Laplacian eigenvectors of the fluorescein graph. The first principal components of SignNet's learned positional encodings do not exactly match any eigenvectors.

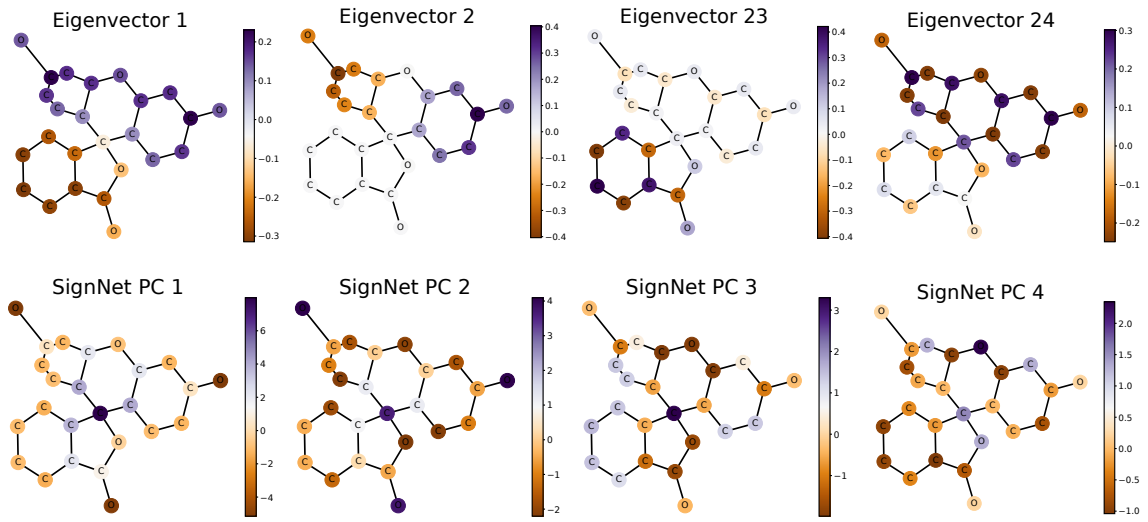


Figure E-5: Normalized Laplacian eigenvectors and learned positional encodings for the graph of fluorescein. (Top row) From left to right: smallest and second smallest nontrivial eigenvectors, then second largest and largest eigenvectors. (Bottom row) From left to right: first four principal components of the output $\rho([\phi(v_i)+\phi(-v_i)]_{i=1,\ldots,n})$ of SignNet.

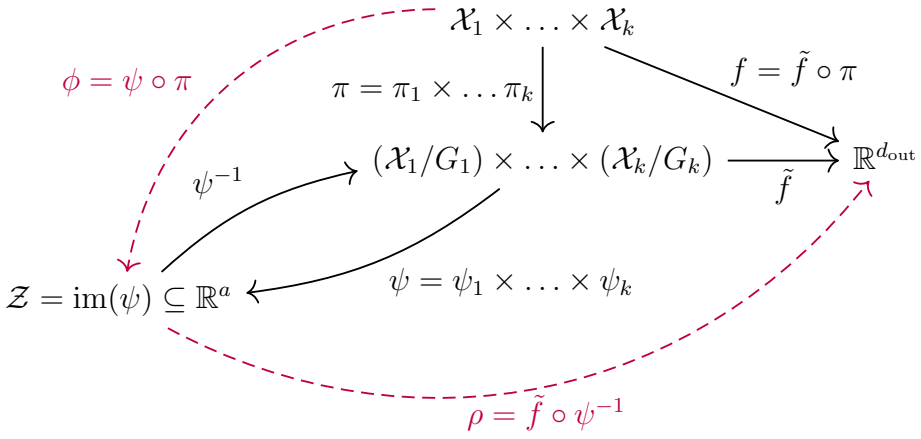
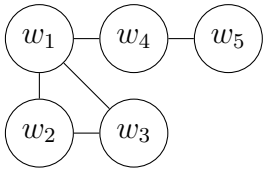
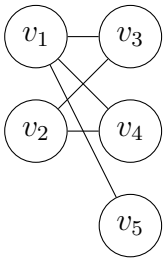


Figure E-6: Commutative diagram for our proof of Theorem 10. Black arrows denote functions from topological constructions, and red dashed lines denote functions that we parameterize by neural networks ($\phi = \phi_1 \times \ldots \times \phi_k$ and ρ).

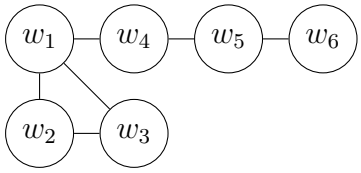


$G^{(1)}$

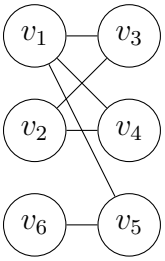


$G^{(2)}$

Figure E-7: Illustration of our constructed $G^{(1)}$ and $G^{(2)}$ for $n = 5$, as used in the proof of Proposition 8.



$G^{(1)}$



$G^{(2)}$

Figure E-8: Illustration of our constructed $G^{(1)}$ and $G^{(2)}$ for $n = 6$, as used in the proof of Proposition 8.

Appendix F

Learning with Discrete Functions in High Dimensions: Further Discussion

F.1 Optimization programs: extended discussion

In this section, we provide an extended discussion of the key components of our LP and SDP formulations and the relationships between them. Apart from supplying derivations, another goal of this section is to illustrate that there is in fact flexibility in the exact choice of formulation for the LP (and consequently the SDP). We provide details on possible variations as part of this discussion as a guide to users who may wish to adapt the SFE framework.

F.1.1 LP formulation: Derivation of the dual.

First, recall that our primal LP is defined as

$$\max_{\mathbf{z},b\in\mathbb{R}^n\times\mathbb{R}} \{\mathbf{x}^\top \mathbf{z} + b\} \;\text{ subject to }\; \mathbf{1}_S^\top \mathbf{z} + b \leq f(S) \;\text{ for all } S \subseteq [n].$$

The dual is

$$\min_{\{y_S \geq 0\}_{S\subseteq[n]}} \sum_{S\subseteq[n]} y_S f(S) \text{ subject to } \sum_{S\subseteq[n]} y_S \mathbf{1}_S = \mathbf{x}, \; \sum_{S\subseteq[n]} y_S = 1, \;\; \text{for all } S \subseteq [n].$$

In order to standardize the derivation, we first convert the primal maximization problem into minimization (this will be undone at the end of the derivation). We have

$$\min_{\mathbf{z},b\in\mathbb{R}^n\times\mathbb{R}}\{-\mathbf{x}^\top\mathbf{z}-b\} \text{ subject to } \mathbf{1}_S^\top\mathbf{z}+b\leq f(S) \text{ for all } S\subseteq[n].$$

The Lagrangian is

$$\begin{aligned}\mathcal{L}(\mathbf{z},y_S,b) &= -\mathbf{x}^\top\mathbf{z}-b-\sum_{S\subseteq[n]} y_S(f(S)-\mathbf{1}_S^\top\mathbf{z}-b)\\ &= -\sum_{S\subseteq[n]} y_S f(S)+(\sum_{S\subseteq[n]} y_S\mathbf{1}_S^\top-\mathbf{x}^\top)\mathbf{z}+b(\sum_{S\subseteq[n]} y_S-1)\end{aligned}$$

The optimal solution $\mathbf{p}^*$ to the primal problem is then

$$\begin{aligned}\mathbf{p}^* &= \min_{\mathbf{z},b}\max_{y_S\geq 0}\mathcal{L}(\mathbf{z},y_S,b)\\ &= \max_{y_S\geq 0}\min_{\mathbf{z},b}\mathcal{L}(\mathbf{z},y_S,b) && \text{(strong duality)}\\ &= \mathbf{d}^*,\end{aligned}$$

where $\mathbf{d}^*$ is the optimal solution to the dual. From the Lagrangian,

$$\min_{\mathbf{z},b}\mathcal{L}(\mathbf{z},y_S,b)=\begin{cases}-\sum_{S\subseteq[n]} y_S f(S), \text{ if } \sum_{S\subseteq[n]} y_S\mathbf{1}_S=\mathbf{x} \text{ and } \sum_{S\subseteq[n]} y_S=1,\\ -\infty, \text{ otherwise.}\end{cases}$$

Thus, we can write the dual problem as

$$\mathbf{d}^*=\max_{y_S\geq 0}-\sum_{S\subseteq[n]} y_S f(S) \text{ subject to } \sum_{S\subseteq[n]} y_S\mathbf{1}_S=\mathbf{x} \text{ and } \sum_{S\subseteq[n]} y_S=1.$$

Our proposed dual formulation is then obtained by switching from maximization to minimization and negating the objective. It can also be verified that by taking the dual of our dual, the primal is recovered (see El Halabi [2018, Def. 20] for the derivation).

F.1.2 Connections to submodularity, related linear programs, and possible alternatives.

Our LP formulation depends on a linear program known to correspond to the convex closure [Murota, 1998, Eq. 3.57] (convex envelope) of a discrete function. Some readers may recognize the formal similarities of this formulation with the one used to define the Lovász extension [Bilmes, 2022]. Namely, for $\mathbf{x} \in \mathbb{R}^n$ we can define the Lovász Extension as

$$\mathfrak{F}(\mathbf{x}) = \max_{\mathbf{z} \in \mathcal{B}_f} \mathbf{x}^\top \mathbf{z},$$

where the feasible set, known as the base polytope of a submodular function, is defined as $\mathcal{B}_f = \{\mathbf{z} \in \mathbb{R}^n : \mathbf{z}^\top \mathbf{1}_S \leq f(S) \; S \subset [n], \text{ and } \mathbf{z}^\top \mathbf{1}_S = f(S) \text{ when } S = [n]\}$. Base polytopes are also known as *generalized permutahedra* and have rich connections to the theory of matroids, since matroid polytopes belong to the class of generalized permutahedra Ardila et al. [2010].

An alternative option is to consider $\mathbf{x} \in \mathbb{R}^n_+$, then the Lovász extension is given by

$$\mathfrak{F}(\mathbf{x}) = \max_{\mathbf{z} \in \mathcal{P}_f} \mathbf{x}^\top \mathbf{z},$$

where $\mathcal{P}_f$ is the submodular polyhedron as defined in our original primal LP. The subtle differences between those formulations lead to differences in the respective dual formulations. In principle, those formulations can be just as easily used to define set function extensions. Overall, there are three key considerations when defining a suitable LP:

- The constraints of the primal.
- The domain of the primal variables $\mathbf{z}, b$ and the cost $\mathbf{x}$.
- The properties of the function being extended.

Below, we describe a few illustrative example cases for different choices of the above:

- Adding the constraint $\mathbf{z}^\top \mathbf{1}_S = f(S)$ when $S = [n]$ leads to $y_{[n]} \in \mathbb{R}^n$ for the dual. This implies that the coefficients cannot be interpreted as probabilities in general which is what provides the guarantee that the extension will not introduce any spurious minima. $\sum_{S \subseteq [n]} y_S = 1$ is just an affine hull constraint in that case.

- For $b = 0$, the constraint $\sum_{S \subseteq [n]} y_S = 1$ is not imposed in the dual and the probabilistic interpretation of the extension cannot be guaranteed. Examples that do not rely on this constraint include the homogeneous convex envelope [El Halabi et al., 2018] and the Lovász extension as presented above. However, even for $b = 0$, from the definition of the Lovász extension it is easy to see that it retains the probabilistic interpretation when $\mathbf{x} \in [0, 1]$.

- Consider a feasible set defined by $\mathcal{P}_f \bigcap \mathbb{R}^n_+$ and let $\mathbf{x} \in \mathbb{R}^n_+$. If the function f is submodular, non-decreasing and normalized so that $f(\varnothing) = 0$ (e.g., the rank function of a matroid), then the feasible set is called polymatroid and f is a polymatroid function. Again, in that case the Lovász extension achieves the optimal objective value [Schrijver et al., 2003, Eq. 44.32]. In that case, the constraint $\sum_{S \subseteq [n]} y_S \mathbf{1}_S = \mathbf{x}$ of the dual is relaxed to $\sum_{S \subseteq [n]} y_S \mathbf{1}_S \geq \mathbf{x}$. This feasible set of the dual will allow for more flexible definitions of an extension but it comes at the cost of generality. For instance, for a submodular function that is not non-decreasing, one cannot obtain the Lovász extension as a feasible solution to the primal LP, and the solutions to this LP will not be the convex envelope in general.

F.1.3 SDP formulation: The geometric intuition of extensions and deriving the dual.

In order to motivate the SDP formulation, first we have to identify the essential ingredients of the LP formulation. First, the constraint $\sum_{S \subseteq [n]} y_S \mathbf{1}_S = \mathbf{x}$ captures the simple idea that each continuous point is expressed as a combination of discrete ones, each representing a different set, which is at the core of our extensions. Then,

ensuring that the continuous point lies in the convex hull of those discrete points confers additional benefits w.r.t. optimization and offers a probabilistic perspective.

Consider the following example. The Lovász extension identifies each continuous point in the hypercube with a simplex. Then the continuous point is viewed as an expectation over a distribution supported on the simplex corners. The value of the set function at a continuous point is then the expected value of the function over those corners under the same distribution, i.e., $\mathbb{E}_{S\sim p_{\mathbf{x}}}[\mathbf{1}_S] = \mathbf{x}$ leads to $\mathbb{E}_{S\sim p_{\mathbf{x}}}[f(S)] = \mathfrak{F}(\mathbf{x})$. As long as the distribution $p_{\mathbf{x}}$ can be differentiated w.r.t $\mathbf{x}$, we obtain an extension that can be used with gradient-based optimization. It is clear that the construction depends on being able to identify a small convex set of discrete vectors that can express the continuous one.

This can be formulated in higher dimensions, particularly in the space of PSD matrices. A natural way to represent sets in high dimensions is through rank one matrices that are outer products of the indicator vectors of the sets, i.e., $\mathbf{1}_S\mathbf{1}_S^\top$ is the matrix representation of S similar to how $\mathbf{1}_S$ is the vector representation. Hence, in the space of matrices, our goal will be again to identify a set of discrete *matrices* that represents sets that can express a matrix of continuous values.

The above considerations set the stage for a transition from linear programming to semidefinite programming, where the feasible sets are spectrahedra. Our SDP formulation attempts to capture the intuition described in the previous paragraphs while also maintaining formal connections to the LP by showing that feasible LP regions correspond to feasible SDP regions by simply projecting the LP regions on the space of diagonal matrices (see Proposition 11).

Derivation of the dual. Recall that our primal SDP is defined as

$$\max_{\mathbf{Z}\succeq 0, b\in\mathbb{R}} \{\mathrm{Tr}(\mathbf{X}^\top\mathbf{Z}) + b\} \text{ subject to } \frac{1}{2}\mathrm{Tr}((\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top)\mathbf{Z}) + b \leq f(S\cap T) \text{ for } S,T\subseteq[n].$$

We will show that the dual is

$$\min_{\{y_{S,T}\geq 0\}} \sum_{S,\subseteq[n]} y_{S,T} f(S\cap T) \text{ subject to } \mathbf{X} \preceq \sum_{S,T\subseteq[n]} \frac{1}{2} y_{S,T}(\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top) \text{ and } \sum_{S,T\subseteq[n]} y_{S,T} = 1.$$

As before, we convert the primal to a minimization problem:

$$\max_{\mathbf{Z}\succeq 0, b\in\mathbb{R}} \{-\text{Tr}(\mathbf{X}^\top\mathbf{Z}) - b\} \text{ subject to } \frac{1}{2}\text{Tr}((\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top)\mathbf{Z}) + b \leq f(S\cap T) \text{ for } S,T\subseteq[n].$$

First, we will standardize the formulation by converting the inequality constraints into equality constraints. This can be achieved by adding a positive slack variable $d_{S,T}$ to each constraint such that

$$\frac{1}{2}\text{Tr}((\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top)\mathbf{Z}) + b + d_{S,T} = f(S\cap T).$$

In matrix notation this is done by introducing the positive diagonal slack matrix $\mathbf{D}$ to the decision variable $\mathbf{Z}$, and extending the symmetric matrices in each constraint

$$\mathbf{Z}' = \begin{bmatrix} \mathbf{Z} & 0 \\ 0 & \mathbf{D} \end{bmatrix}, \quad \mathbf{X}' = \begin{bmatrix} \mathbf{X} & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{A}'_{S,T} = \begin{bmatrix} \frac{1}{2}(\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top) & 0 \\ 0 & \text{diag}(\mathbf{e}_{S,T}) \end{bmatrix},$$

where $\text{diag}(\mathbf{e}_{S,T})$ is a diagonal matrix where all diagonal entries are zero except at the diagonal entry corresponding to the constraint on S,T which has a 1. Using this reformulation, we obtain an equivalent SDP in standard form:

$$\max_{\mathbf{Z}'\succeq 0, b\in\mathbb{R}} \{-\text{Tr}(\mathbf{X}'^\top\mathbf{Z}') - b\} \text{ subject to } \text{Tr}(\mathbf{A}'_{S,T}\mathbf{Z}') + b = f(S\cap T) \text{ for } S,T\subseteq[n].$$

Next, we form the Lagrangian which features a decision variable $y_{S,T}$ for each inequality, and a dual matrix variable $\mathbf{\Lambda}$. We have

$$\mathcal{L}(\mathbf{Z}', b, y_{S,T}, \mathbf{\Lambda}) = -\mathrm{Tr}(\mathbf{X}'^{\top}\mathbf{Z}') - b - \sum_{S,T\subseteq[n]} y_{S,T}\left(2f(S\cap T) - \mathrm{Tr}(\mathbf{A}'_{S,T}\mathbf{Z}') - b\right) - \mathrm{Tr}(\mathbf{\Lambda}\mathbf{Z}')$$

$$= \mathrm{Tr}\left(((\sum_{S,T\subseteq[n]} y_{S,T}\mathbf{A}'_{S,T}) - \mathbf{X}' - \mathbf{\Lambda})\mathbf{Z}'\right) + b(\sum_{S,T\subseteq[n]} y_{S,T} - 1) - \sum_{S,T\subseteq[n]} y_{S,T}f(S\cap T)$$

For the solution to the primal $\mathbf{p}^*$, we have

$$\begin{aligned}
\mathbf{p}^* &= \min_{\mathbf{Z}',b}\max_{\mathbf{\Lambda},y_{S,T}} \mathcal{L}(\mathbf{Z}', b, y_{S,T}, \mathbf{\Lambda}) \\
&\geq \max_{\mathbf{\Lambda},y_{S,T}}\min_{\mathbf{Z}',b} \mathcal{L}(\mathbf{Z}', b, y_{S,T}, \mathbf{\Lambda}) && \text{(weak duality)} \\
&= \mathbf{d}^*.
\end{aligned}$$

For our Lagrangian we have the dual function

$$\min_{\mathbf{Z}',b} \mathcal{L}(\mathbf{Z}', b, y_{S,T}, \mathbf{\Lambda}) = \begin{cases} 0, \text{ if } \mathbf{\Lambda} \succeq 0, \\ -\infty, \text{ otherwise .} \end{cases}$$

Thus, the dual function $\min_{\mathbf{Z}',b} \mathcal{L}(\mathbf{Z}', b, y_{S,T}, \mathbf{\Lambda})$ takes non-infinite values under the conditions

$$\begin{aligned}
(\sum_{S,T\subseteq[n]} y_{S,T}\mathbf{A}'_{S,T}) - \mathbf{X}' - \mathbf{\Lambda} = 0, \\
\mathbf{\Lambda} \succeq 0, \\
\text{and } \sum_{S,T\subseteq[n]} y_{S,T} - 1 = 0.
\end{aligned}$$

The first two conditions imply the linear matrix inequality (LMI)

$$\sum_{S,T\subseteq[n]} y_{S,T}\mathbf{A}'_{S,T} - \mathbf{X}' \succeq 0. \qquad (\mathbf{\Lambda} \succeq 0)$$

From the definition of $\mathbf{A}'_{S,T}$ we know that its additional diagonal entries will correspond to the variables $y_{S,T}$. Combined with the conditions above, we arrive at the constraints of the dual

$$\begin{aligned} y_{S,T} &\geq 0, \\ \sum_{S,T\subseteq[n]} \frac{1}{2} y_{S,T}(\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top) &\succeq \mathbf{X}, \\ \sum_{S,T\subseteq[n]} y_{S,T} &= 1. \end{aligned}$$

This leads us to the dual formulation

$$\max_{y_{S,T}\geq 0} -\sum_{S,T\subseteq[n]} y_{S,T} f(S\cap T) \text{ subject to } \sum_{S,T\subseteq[n]} \frac{1}{2} y_{S,T}(\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top) \succeq \mathbf{X} \text{ and } \sum_{S,T\subseteq[n]} y_{S,T} = 1.$$

Then, we can obtain our original dual by switching to minimization and negating the objective.

F.2 Scalar Set Function Extensions Have No Bad Minima

In this section we re-state and prove the results from Section 8.3. The first result concerns the minima of $\mathfrak{F}$, showing that the minimum value is the same as that of f, and no additional minima are added (besides convex combinations of discrete minimizers). These properties are especially desirable when using an extension $\mathfrak{F}$ as a loss function (see Section 4.4) since it is important that $\mathfrak{F}$ drive the neural network NN_1 towards producing discrete $\mathbf{1}_S$ outputs.

Proposition 17 (Scalar SFEs have no bad minima)**.** *If $\mathfrak{F}$ is a scalar SFE of f then:*

1. $\min_{\mathbf{x}\in\mathcal{X}} \mathfrak{F}(\mathbf{x}) = \min_{S\subseteq[n]} f(S)$

2. $\arg\min_{\mathbf{x}\in\mathcal{X}} \mathfrak{F}(\mathbf{x}) \subseteq \mathit{Hull}\big(\arg\min_{\mathbf{1}_S : S\subseteq[n]} f(S)\big)$

Proof. The inequality $\min_{\mathbf{x}\in\mathcal{X}} \mathfrak{F}(\mathbf{x}) \leq \min_{S\subseteq[n]} f(S)$ automatically holds since $\min_{S\subseteq[n]} f(S) = \min_{\mathbf{1}_S : S\subseteq[n]} \mathfrak{F}(\mathbf{1}_S)$, and $\{\mathbf{1}_S : S \subseteq [n]\} \subseteq \mathcal{X}$. So it remains to show the reverse. Indeed, letting $\mathbf{x} \in \mathcal{X}$ be an arbitrary point we have,

$$\begin{aligned}
\mathfrak{F}(\mathbf{x}) &= \mathbb{E}_{S\sim p_{\mathbf{x}}}[f(S)] \\
&= \sum_{S\subseteq[n]} p_{\mathbf{x}}(S)\cdot f(S) \\
&\geq \sum_{S\subseteq[n]} p_{\mathbf{x}}(S)\cdot \min_{S\subseteq[n]} f(S) \\
&= \min_{S\subseteq[n]} f(S)
\end{aligned}$$

where the last equality simply uses the fact that $\sum_{S\subseteq[n]} p_{\mathbf{x}}(S) = 1$. This proves the first claim.

To prove the second claim, suppose that $\mathbf{x}$ minimizes $\mathfrak{F}(\mathbf{x})$ over $\mathbf{x} \in \mathcal{X}$. This implies that the inequality in the above derivation must be tight, which is true if and only if

$$p_{\mathbf{x}}(S)\cdot f(S) = p_{\mathbf{x}}(S)\cdot \min_{S\subseteq[n]} f(S) \quad \text{for all } S \subseteq [n].$$

For a given S, this implies that either $p_{\mathbf{x}}(S) = 0$ or $f(S) = \min_{S\subseteq[n]} f(S)$. Since $\mathbf{x} = \mathbb{E}_{p_{\mathbf{x}}}[\mathbf{1}_S] = \sum_{S\subseteq[n]} p_{\mathbf{x}}(S)\cdot\mathbf{1}_S = \sum_{S:p_{\mathbf{x}}(S)>0} p_{\mathbf{x}}(S)\cdot\mathbf{1}_S$. This is precisely a convex combination of points $\mathbf{1}_S$ for which $f(S) = \min_{S\subseteq[n]} f(S)$. Since $\mathfrak{F}$ is a convex combination of exactly this set of points $\mathbf{1}_S$, we have the second claim.

□

F.3 Examples of Vector Set Function Extensions

This section re-defines the vector SFEs given in Section 8.3.1, and prove that they satisfy the definition of an SFEs. One of the conditions we must check is that $\mathfrak{F}$ is continuous. A sufficient condition for continuity (and almost everywhere differentiability) that we shall use for a number of constructions is to show that $\mathfrak{F}$ is Lipschitz. A very simple computation shows that it suffices to show that $\mathbf{x} \in \mathcal{X} \mapsto p_{\mathbf{x}}(S)$ is Lipschitz continuous.

Lemma 17. *If the mapping $\mathbf{x} \in [0,1]^n \mapsto p_{\mathbf{x}}(S)$ is Lipschitz continuous and $f(S)$ is finite for all S in the support of $p_{\mathbf{x}}$, then $\mathfrak{F}$ is also Lipschitz continuous. In particular, $\mathfrak{F}$ is continuous and almost everywhere differentiable.*

Proof. The Lipschitz continuity of $\mathfrak{F}(\mathbf{x})$ follows directly from definition:

$$\begin{aligned}\left|\mathfrak{F}(\mathbf{x}) - \mathfrak{F}(\mathbf{x}')\right| &= \left| \sum_{S\subseteq[n]} p_{\mathbf{x}}(S)\cdot f(S) - \sum_{S\subseteq[n]} p_{\mathbf{x}'}(S)\cdot f(S)\right| \\ &= \left| \sum_{S\subseteq[n]} \big(p_{\mathbf{x}}(S) - p_{\mathbf{x}'}(S)\big)\cdot f(S)\right| \leq \left(2kL \max_{S\subseteq[n]} f(S)\right)\cdot \|\mathbf{x}-\mathbf{x}'\|,\end{aligned}$$

where L is the maximum Lipschitz constant of $\mathbf{x} \mapsto p_{\mathbf{x}}(S)$ over any S in the support of $p_{\mathbf{x}}$, and k is the maximal cardinality of the support of any $p_{\mathbf{x}}$. □

In general k can be trivially bounded by 2^n, so $\mathfrak{F}$ is always Lipschitz. However in may cases the cardinality of the support of any $p_{\mathbf{x}}$ is much smaller than 2^n, leading too a smaller Lipschitz constant. For instance, $k = n$ in the case of the Lovász extension.

F.3.1 Lovász extension.

Recall the definition: $\mathbf{x}$ is sorted so that $x_1 \geq x_2 \geq \ldots \geq x_d$. Then the Lovász extension corresponds to taking $S_i = \{1,\ldots,i\}$, and letting $p_{\mathbf{x}}(S_i) = x_i - x_{i+1}$, the non-negative increments of $\mathbf{x}$ (where recall we take $x_{n+1} = 0$). All other sets have zero probability. For convenience, we introduce the shorthand notation $a_i = p_{\mathbf{x}}(S_i) = x_i - x_{i+1}$

Feasibility. Clearly all $a_i = x_i - x_{i+1} \geq 0$, and $\sum_{i=1}^n a_i = \sum_{i=1}^n (x_i - x_{i+1}) = x_1 \leq 1$. Any remaining probability mass is assigned to the empty set: $p_{\mathbf{x}}(\varnothing) = 1 - x_1$, which contributes nothing to the extension $\mathfrak{F}$ since $f(\varnothing) = 0$ by assumption. All that remains is to check that

$$\sum_{i=1}^{n} p_{\mathbf{x}}(S_i) \cdot \mathbf{1}_{S_i} = \mathbf{x}.$$

For a given $k \in [n]$, note that the only sets S_i with non-zero kth coordinate are $S_1, \ldots, S_k$, and in all cases $(\mathbf{1}_{S_i})_k = 1$. So the kth coordinate is precisely $\sum_{i=1}^k p_{\mathbf{x}}(S_i) = \sum_{i=1}^k (x_i - x_{i+1}) = x_k$, yielding the desired formula.

Extension. Consider an arbitrary $S \subseteq [n]$. Since we assume $\mathbf{x} = \mathbf{1}_S$ is sorted, it has the form $\mathbf{1}_S = (\underbrace{1, 1, \ldots, 1}_{k \text{ times}}, 0, 0, \ldots 0)^\top$. Therefore, for each $j < k$ we have $a_j = x_j - x_{j+1} = 1 - 1 = 0$ and for each $j > k$ we have $a_j = x_j - x_{j+1} = 0 - 0 = 0$. The only non-zero probability is $a_k = x_k - x_{k+1} = 1 - 0 = 1$. So,

$$\mathfrak{F}(\mathbf{1}_S) = \sum_{i=1}^{n} a_i f(S_i) = \sum_{i : i \neq k} a_i f(S_i) + a_k f(S_k) = 0 + 1 \cdot f(S_k) = f(S)$$

where the the final equality follows since by definition S_k corresponds exactly to the vector $(\underbrace{1, 1, \ldots, 1}_{k \text{ times}}, 0, 0, \ldots 0)^\top = \mathbf{1}_S$ and so $S_k = S$.

Continuity. The Lovász is a well-known extension, whose properties have been carefully studied. In particular it is well known to be a Lipschitz function Bach [2019]. However, for completeness we provide a simple proof here nonetheless.

Lemma 18. *Let $p_{\mathbf{x}}$ be as defined for the Lovász extension. Then $\mathbf{x} \mapsto p_{\mathbf{x}}(S)$ is Lipschitz for all $S \subseteq [n]$.*

Proof. First note that $p_{\mathbf{x}}$ is piecewise linear, with one piece per possible ordering $x_1 \geq x_2 \geq \ldots \geq x_n$ (so $n!$ pieces in total). Within the interior of each piece $p_{\mathbf{x}}$ is linear, and therefore Lipschitz. So in order to prove global Lipschitzness, it suffices to show that $p_{\mathbf{x}}$ is continuous at the boundaries between pieces (the Lipschitz constant is then the maximum of the Lipschitz constants for each linear piece).

Now consider a point $\mathbf{x}$ with $x_1 \geq \ldots \geq x_i = x_{i+1} \geq \ldots \geq x_n$. Consider the perturbed point $\mathbf{x}_\delta = \mathbf{x} - \delta \mathbf{e}_i$ with $\delta > 0$, and $\mathbf{e}_i$ denoting the ith standard basis vector. To prove continuity of $p_{\mathbf{x}}$ it suffices to show that for any $S \in \Omega$ we have $p_{\mathbf{x}_\delta}(S) \to p_{\mathbf{x}}(S)$ as $\delta \to 0^+$.

There are two sets in the support of $p_{\mathbf{x}}$ whose probabilities are different under $p_{\mathbf{x}_\delta}$, namely: $S_i = \{1, \ldots, i\}$ and $S_{i+1} = \{1, \ldots, i, i+1\}$. Similarly, there are two sets in the support of $p_{\mathbf{x}_\delta}$ whose probabilities are different under $p_{\mathbf{x}}$, namely: $S'_i = \{1, \ldots, i-1, i+1\}$ and $S'_{i+1} = \{1, \ldots, i, i+1\} = S_{i+1}$. So it suffices to show the convergence $p_{\mathbf{x}_\delta}(S) \to p_{\mathbf{x}}(S)$ for these four S. Consider first S_i:

$$\left|p_{\mathbf{x}_\delta}(S_i) - p_{\mathbf{x}}(S_i)\right| = \left|0 - (x_i - x_{i+1})\right| = 0$$

where the final equality uses the fact that $x_i = x_{i+1}$. Next consider $S_{i+1} = S'_{i+1}$:

$$\left|p_{\mathbf{x}_\delta}(S_{i+1}) - p_{\mathbf{x}}(S_{i+1})\right| = \left|(x'_{i+1} - x'_{i+2}) - (x_{i+1} - x_{i+2})\right| = \left|(x'_{i+1} - x_{i+1}) - (x'_{i+2} - x_{i+2})\right| = 0$$

Finally, we consider S'_i:

$$\begin{aligned}
\left|p_{\mathbf{x}_\delta}(S'_i) - p_{\mathbf{x}}(S'_i)\right| &= \left|(x'_i - x'_{i+1}) - (x_i - x_{i+1})\right| \\
&= \left|(x'_{i+1} - x_{i+1}) - (x'_{i+1} - x_{i+1})\right| \\
&= \left|(x_{i+1} - \delta - x_{i+1}) - (x'_{i+1} - x_{i+1})\right| \\
&= \delta \to 0
\end{aligned}$$

completing the proof. □

F.3.2 Bounded cardinality Lovaśz extension.

The bounded cardinality extension considers n sets S of cardinality at most k, with $n \geq k \geq 2$. We collect $\{S_i\}_{i=1}^n$ of subsets of $[n]$ in an $n \times n$ matrix $\mathbf{S} \in \{0,1\}^{n \times n}$

whose ith column is $\mathbf{1}_{S_i}$:

$$\mathbf{S} = \begin{bmatrix} \overbrace{1 & \dots & 1}^{\text{k}} & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & 1 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The matrix will contain k sets of gradually increasing cardinality, from 1 up until k, and $n-k$ sets of cardinality exactly k. In this notation, the dual LP constraint $\sum_{S\subseteq[n]} y_S \mathbf{1}_S = \mathbf{x}$ can be written as $\mathbf{Sp} = \mathbf{x}$, where the ith coordinate of $\mathbf{p}$ defines $p_{\mathbf{x}}(S_i)$. Then, the bounded cardinality extension coefficients $p_{\mathbf{x}}(S)$ are the coordinates of the vector $\mathbf{y}$, where $\mathbf{y} = \mathbf{S}^{-1}\mathbf{x}$. To calculate the inverse, we will leverage the fact that $\mathbf{S}$ will be triangular Toeplitz by construction. Clearly, its inverse will also be triangular.

Lemma 19. *The entries (i,j) of the inverse are*

$$\mathbf{S}^{-1}(i,j) = \begin{cases} 1, \ \textit{if } (j-i) \ \textit{mod } k = 0 \ \textit{and } i \le j, \\ -1, \ \textit{if } (j-i) \ \textit{mod } k = 1 \ \textit{and } i \le j, \\ 0, \ \textit{otherwise}, \end{cases}$$

for $i = 1, 2, \dots, n$.

Proof. The proof relies on known results for banded Toeplitz matrices. A banded Toeplitz matrix of bandwidth r and superdiagonal s is an $n \times n$ matrix that has the

following form:

$$\mathbf{T}_{r,s} = \begin{bmatrix} c_{s+1} & c_s & \dots & c_1 & & 0 \\ c_{s+2} & c_{s+1} & c_s & \dots & \ddots & \\ \vdots & \ddots & \ddots & \ddots & & c_1 \\ c_r & & \ddots & \ddots & & \vdots \\ & \ddots & & \ddots & \ddots & \vdots \\ 0 & & c_r & \dots & \dots & c_{s+1} \end{bmatrix}.$$

Note here that the (i, j) entry of $\mathbf{T}$, due to its Toeplitz structure, is going to be $\mathbf{T}(i, j) = c_{i-j+s+1}$. For convenience, we are going to invert $\mathbf{S}^\top$ and the result straightforwardly transfers to $\mathbf{S}$. For $\mathbf{S}^\top$, we have superdiagonal $s = 0$ and bandwidth $r = k$. It is known [Meek, 1983, Trench, 1974] that the entries $g_{i-j+1} = (\mathbf{S}^\top)^{-1}(i, j)$ of the inverse will obey the following difference equation:

$$c_k g_{l-k} + c_{k-1} g_{l-k+1} + \cdots = 0, \quad l \geq 3, \quad g_1 = 1,$$

with $g_0 = g_{-1} = \cdots = g_{3-k} = 0$. Considering the conditions above and the fact that $c_1 = c_2 = \cdots = c_k = 1$, the difference equation simplifies to

$$\sum_{t=0}^{k-1} g_{l-k+t} = 0.$$

As an example, let us compute the case for $k = 3, l = 3$. We obtain $g_0 + g_1 + g_2 = 0$, which implies $g_2 = -1$. It is easy to see that for any k, computing the difference equation for $l = 3$ yields $g_2 = -1$ since all the negative indices do not contribute to the sum, reducing it to $g_1 + g_2 = 0$.

We continue with $k = 3, l = 4$ and obtain $g_1 + g_2 + g_3 = 0$, which implies $g_3 = 0$. By incrementing l, observe that we are shifting the terms in the sum by one, so this straightforwardly implies that $l = 5$ yields $g_4 = 1$ for $k = 3$, and so on. Generalizing

this observation, we obtain the following cases:

- $g_t = 1$, for $t = mk + 1$,
- $g_t = -1$, for $t = mk + 2$,
- $g_t = 0$, otherwise.

Here, m is a non-negative integer. The lemma follows straightforwardly from that observation. □

Equivalence to the Lovaśz extension. We want to show that the bounded cardinality extension is equivalent to the Lovaśz extension when $k = n$. Let $T_{i,k} = \{j \mid (j - i) \bmod k = 0, \text{ for } i \leq j \leq n, \}$, i.e., $T_{i,k}$ stores the indices where $j - i$ is perfectly divided by k. From the analytic form of the inverse, observe that the i-th coordinate of $\mathbf{y}$ is $p_{\mathbf{x}}(S_i) = \sum_{j \in T_{i,k}} (x_j - x_{j+1})$. For $k = n$, we have $T_{i,n} = \{j \mid (j - i) \bmod n = 0\} = \{i\}$, and therefore $p_{\mathbf{x}}(S_i) = x_i - x_{i+1}$, which are the coefficients of the Lovász extension.

Feasibility. The equation $\mathbf{y} = \mathbf{S}^{-1}\mathbf{x}$ guarantees that the constraint $\mathbf{x} = \sum_{i=1}^{n} y_{S_i} \mathbf{1}_{S_i}$ is obeyed. Recall that $\mathbf{x}$ is sorted in descending order like in the case of the Lovász extension. Then, it is easy to see that $p_{\mathbf{x}}(S_i) = \sum_{j \in T_{i,k}} (x_j - x_{j+1}) \leq x_i$, because $x_i - x_{i+1}$ is always contained in the summation for $p_{\mathbf{x}}(S_i)$. Therefore, by restricting $\mathbf{x}$ in the probability simplex it is easy to see that $\sum_{i=1}^{n} p_{\mathbf{x}}(S_i) \leq \sum_{i=1}^{n} x_i = 1$. To secure tight equality, we allocate the rest of the mass to the empty set, i.e., $p_{\mathbf{x}}(\varnothing) = 1 - \sum_{i=1}^{n} p_{\mathbf{x}}(S_i)$, which does not affect the value of the extension since the corresponding Boolean is the zero vector.

Extension. To prove the extension property we need to show that $\mathfrak{F}(\mathbf{1}_S) = f(S)$ for all S with $|S| \leq k$. Consider any such set S and recall that we have sorted $\mathbf{1}_S$ with arbitrary tie breaks, such that $x_i = 1$ for $i \leq |S|$ and $x_i = 0$ otherwise. Due to the equivalence with the Lovaśz extension, the extension property is guaranteed when $k = n$ for all possible sets. For $k < n$, consider the following three cases for $T_{i,k}$.

- When $i > |S|$, $T_{i,k} = \varnothing$ because for sorted $\mathbf{x}$ of cardinality at most k, we know for the coordinates that $x_i = x_{i+1} = 0$. For $i > k$, this implies that $p_{\mathbf{x}}(S_i) = 0$.

- When $i < |S|$, $\sum_{j \in T_{i,k}} (x_j - x_{j+1}) = 0$ because $x_j = x_{j+1} = 1$ and we have again $p_{\mathbf{x}}(S_i) = 0$.

- When $i = |S|$, observe that $\sum_{j \in T_{i,k}} (x_j - x_{j+1}) = x_i - x_{i+1} = x_i$. Therefore, $p_{\mathbf{x}}(S_i) = 1$. in that case.

Bringing it all together, $\mathfrak{F}(\mathbf{1}_S) = \sum_{i=1}^{n} p_{\mathbf{x}} f(S_i) = p_{\mathbf{x}}(S) f(S) = f(S)$ since the sum contains only one nonzero term, the one that corresponds to $i = |S|$.

Continuity. Similar to the Lovaśz extension, $p_{\mathbf{x}}$ in the bounded cardinality extension is piecewise linear and therefore a.e. differentiable with respect to $\mathbf{x}$, where each piece corresponds to an ordering of the coordinates of $\mathbf{x}$. On the other hand, unlike the Lovaśz extension, the mapping $\mathbf{x} \mapsto p_{\mathbf{x}}(S)$ is not necessarily globally Lipschitz when $k < n$, because it is not guaranteed to be Lipschitz continuous at the boundaries.

F.3.3 Singleton extension.

Feasibility. The singleton extension is not dual LP feasible. However, one of the key reasons why feasibility is important is that it implies Proposition 10, which show that optimizing $\mathfrak{F}$ is a reasonable surrogate to f. In the case of the singleton extension, however, Proposition 10 still holds even without feasibility for f. This includes the case of the training accuracy loss, which can be viewed as minimizing the set function $f(\{\hat{y}\}) = -\mathbf{1}\{y_i = \hat{y}\}$.

Here we give an alternative proof of Proposition 10 for the singleton extension. Consider the same assumptions as Proposition 10 with the additional requirement that $\min_S f(S) < 0$ (this merely asserts hat $S = \varnothing$ is not a trivial solution to the minimization problem, and that the minimizer of f is unique. This is true, for example, for the training accuracy objective we consider in Section 4.4.

Proof of Proposition 10 for singleton extension. For $\mathbf{x} \in \mathcal{X} = [0,1]^n$,

$$\begin{aligned}
\mathfrak{F}(\mathbf{x}) &= \sum_{i=1}^{n} p_{\mathbf{x}}(S_i) f(S_i) \\
&= \sum_{i=1}^{n} (x_i - x_{i+1}) f(S_i) \\
&\geq \sum_{i=1}^{n} (x_i - x_{i+1}) \min_{j \in [n]} f(S_j) \\
&\geq (x_1 - x_{n+1}) \min_{j \in [n]} f(S_j) \\
&\geq x_1 \cdot \min_{j \in [n]} f(S_j) \\
&\geq \min_{j \in [n]} f(S_j)
\end{aligned}$$

where the final inequality follows since $\min_{j \in [n]} f(S_j) < 0$. Taking $\mathbf{x} = (1, 0, 0, \ldots, 0)^\top$ shows that all the inequalities can be made tight, and the first statement of Proposition 10 holds. For the second statement, suppose that $\mathbf{x} \in \mathcal{X} = [0,1]^n$ minimizes $\mathfrak{F}$. Then all the inequality in the preceding argument must be tight. In particular, tightness of the final inequality implies that $x_1 = 1$. Meanwhile, tightness of the first inequaliity implies that $x_i - x_{i+1} = 0$ for all i for which $f(S_i) \neq \min_{j \in [n]} f(S_j)$, and tightness of the second inequality implies that $x_{n+1} = 0$. These together imply that $\mathbf{x} = \mathbf{1} \oplus \mathbf{0}_{n-1}$ where $\mathbf{1}$ is a 1×1 vector with entry equal to one, and $\mathbf{0}_{n-1}$ is an all zeros vectors of length $n-1$, and $\oplus$ denotes concatenation. Since $f(S_1) = \min_{j \in [n]} f(S_j)$ is the unique minimize we have that $\mathbf{x} = \mathbf{1}_{S_1} \in \mathrm{Hull}\big(\arg\min_{\mathbf{1}_{S_i} : i \in [n]} f(S_i)\big)$, completing the proof. □

Extension. Consider an arbitrary $i \in [n]$. Since we assume $\mathbf{x} = \mathbf{1}_{\{i\}}$ is sorted, we are without loss of generality considering $\mathbf{1}_{\{1\}} = (1, 0, \ldots, 0, 0, \ldots 0)^\top$. Therefore, we have $p_{\mathbf{x}}(S_1) = x_1 - x_2 = 1 - 0 = 1$ and for each $j > 1$ we have $p_{\mathbf{x}}(S_j) = x_j - x_{j+1} = 0 - 0 = 0$.

The only non-zero probability is $p_{\mathbf{x}}(S_1)$, and so

$$\mathfrak{F}(\mathbf{1}_{\{1\}}) = \sum_{j=1}^{n} p_{\mathbf{x}}(S_j) f(S_j) = f(S_1) = f(\{1\}).$$

Continuity. The proof of continuity of the singleton extension is a simple adaptation of the proof used for the Lovaśz extension, which we omit.

F.3.4 Permutations and Involutory Extension.

Feasibility. It is known that every elementary permutation matrix is involutory, i.e., $\mathbf{SS} = \mathbf{I}$. Given such an elementary permutation matrix $\mathbf{S}$, since $\mathbf{S}(\mathbf{Sx}) = \mathbf{S}p_{\mathbf{x}} = \mathbf{x}$, the constraint $\sum_{S\subseteq[n]} y_S \mathbf{1}_S = \mathbf{x}$ is satisfied. Furthermore, $\sum_{S\subseteq[n]} y_S = 1$ can be secured if $\mathbf{x}$ is in the simplex, since the sum of the elements of a vector is invariant to permutations of the entries.

Extension. If the permutation has a fixed point at the maximum element of $\mathbf{x}$, i.e., it maps the maximum element to itself, then any elementary permutation matrix with such a fixed point yields an extension on singleton vectors. Without loss of generality, let $\mathbf{x} = \mathbf{e}_1$, where $\mathbf{e}_1$ is the standard basis vector in $\mathbb{R}^n$. Then $\mathbf{Se}_1 = \mathbf{e}_1$ and therefore $p_{\mathbf{x}}(\mathbf{e}_1) = 1$. This in turn implies $\mathfrak{F}(\mathbf{e}_1) = 1 \cdot f(\mathbf{e}_1)$. This argument can be easily applied to all singleton vectors.

Continuity. The permutation matrix $\mathbf{S}$ can be chosen in advance for each $\mathbf{x}$ in the simplex. Since $p_{\mathbf{x}} = \mathbf{Sx}$, the probabilities are piecewise-linear and each piece is determined by the fixed point induced by the maximum element of $\mathbf{x}$. Consequently, $p_{\mathbf{x}}$ depends continuously on $\mathbf{x}$.

F.3.5 Multilinear extension.

Recall that the multiliniear extension is defined via $p_{\mathbf{x}}(S) = \prod_{i\in S} x_i \prod_{i\notin S}(1 - x_i)$ supported on all subsets $S \subseteq [n]$ in general.

Feasibility. The definition of $p_{\mathbf{x}}(S)$ is equivalent to:

$$p_{\mathbf{x}}(S) = \prod_{i=1}^{n} x_i^{y_i}(1-x_i)^{1-y_i}$$

where $y_i = 1$ if $i \in S$ and zero otherwise. That is, $p_{\mathbf{x}}(S)$ is the product of n independent Bernoulli distributions. So we clearly have $p_{\mathbf{x}}(S) \geq 0$ and $\sum_{S\subseteq[n]} p_{\mathbf{x}}(S) = 1$. The final feasibility condition, that $\sum_{S\subseteq[n]} p_{\mathbf{x}}(S) \cdot \mathbf{1}_S = \mathbf{x}$ can be checked by induction on n. For $n = 1$ there are only two sets: $\{1\}$ and the empty set. And clearly $p_{\mathbf{x}}(\{1\}) \cdot \mathbf{1}_{\{1\}} = x_1(1-x_1)^0 = x_1$, so we have the base case.

Extension. For any $S \subseteq [n]$ we have $p_{\mathbf{1}_S}(S) = \prod_{i\in S} x_i \prod_{i\notin S}(1-x_i) = \prod_{i\in S} 1 \prod_{i\notin S}(1-0) = 1$. So $\mathfrak{F}(\mathbf{1}_S) = \mathbb{E}_{T\sim p_{\mathbf{x}}} f(T) = f(S)$.

Continuity. Fix and $S \subseteq [n]$. Again we check Lipschitzness. We use ∂_{x_k} to denote the derivative operator with respect to x_k. If $k \in S$ we have

$$|\partial_{x_k} p_{\mathbf{1}_S}(S)| = \left|\partial_{x_k} \prod_{i\in S} x_i \prod_{i\notin S}(1-x_i)\right| = \prod_{i\in S\setminus\{k\}} x_i \prod_{i\notin S}(1-x_i) \leq 1.$$

Similarly, if $k \notin S$ we have,

$$|\partial_{x_k} p_{\mathbf{1}_S}(S)| = \left|\partial_{x_k} \prod_{i\in S} x_i \prod_{i\notin S}(1-x_i)\right| = \left|-\prod_{i\in S} x_i \prod_{i\notin S\cup\{k\}} (1-x_i)\right| \leq 1.$$

Hence the spectral norm of the Jacobian $Jp_{\mathbf{x}}(S)$ is bounded, and so $\mathbf{x} \mapsto p_{\mathbf{x}}(S)$ is a Lipschitz map.

F.4 Neural Set Function Extensions

This section re-states and proves the results from Section 8.4. To start, recall the definition of the primal LP:

$$\max_{\mathbf{z},b}\{\mathbf{x}^\top\mathbf{z}+b\}, \quad \text{where} \quad (\mathbf{z},b)\in\mathbb{R}^n\times\mathbb{R} \text{ and } \mathbf{1}_S^\top\mathbf{z}+b\leq f(S) \quad \text{for all } S\subseteq[n].$$

and primal SDP:

$$\max_{\mathbf{Z}\succeq 0, b\in\mathbb{R}}\{\mathrm{Tr}(\mathbf{X}^\top\mathbf{Z})+b\} \text{ subject to } \frac{1}{2}\mathrm{Tr}((\mathbf{1}_S\mathbf{1}_T^\top+\mathbf{1}_T\mathbf{1}_S^\top)\mathbf{Z})+b\leq f(S\cap T) \text{ for } S,T\subseteq[n].$$

Proposition 18. *(Containment of LP in SDP) For any $\mathbf{x}\in[0,1]^n$, define $\mathbf{X}=\sqrt{\mathbf{x}}\sqrt{\mathbf{x}}^\top$ with the square-root taken entry-wise. Then, for any $(\mathbf{z},b)\in\mathbb{R}^n_+\times\mathbb{R}$ that is primal LP feasible, the pair $(\mathbf{Z},b)$ where $\mathbf{Z}=\mathrm{diag}(\mathbf{z})$, is primal SDP feasible and the objective values agree: $\mathrm{Tr}(\mathbf{X}^\top\mathbf{Z})=\mathbf{z}^\top\mathbf{x}$.*

Proof. We start with the feasibility claim. Suppose that $(\mathbf{z},b)\in\mathbb{R}^n_+\times\mathbb{R}$ is a feasible solution to the primal LP. We must show that $(\mathbf{Z},b)$ is a feasible solution to the primal SDP with $\mathbf{X}=\sqrt{\mathbf{x}}\sqrt{\mathbf{x}}^\top$ and where $\mathbf{Z}=\mathrm{diag}(\mathbf{z})$.

Recall the general formula for the trace of a matrix product: $\mathrm{Tr}(\mathbf{AB})=\sum_{i,j}A_{ij}B_{ji}$. With this in mind, and noting that the (i,j) entry of $\mathbf{1}_S\mathbf{1}_T^\top$ is equal to 1 if $i,j\in S\cap T$,

and zero otherwise, we have for any $S, T \subseteq [n]$ that

$$\begin{aligned}
\frac{1}{2}\mathrm{Tr}((\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top)\mathbf{Z}) + b = \mathrm{Tr}(\mathbf{1}_S\mathbf{1}_T^\top\mathbf{Z}) + b &= \sum_{i,j=1}^{n}(\mathbf{1}_S\mathbf{1}_T^\top)_{ij} \cdot \mathrm{diag}(\mathbf{z})_{ij} + b \\
&= \sum_{i,j\in S\cap T}(\mathbf{1}_S\mathbf{1}_T^\top)_{ij} \cdot \mathrm{diag}(\mathbf{z})_{ij} + b \\
&= \sum_{i,j\in S\cap T}\mathrm{diag}(\mathbf{z})_{ij} + b \\
&= \sum_{i\in S\cap T} z_i + b \\
&= \mathbf{1}_{S\cap T}^\top\mathbf{z} + b \\
&\leq f(S\cap T)
\end{aligned}$$

showing SDP feasibility. That the objective values agree is easily seen since:

$$\mathrm{Tr}(\mathbf{ZX}) = \sum_{i,j=1}^{n}\mathrm{diag}(\mathbf{z})_{ij} \cdot \sqrt{x_i}\sqrt{x_j} = \sum_{i=1}^{n} z_i \cdot \sqrt{x_i}\sqrt{x_i} = \mathbf{x}^\top\mathbf{z}.$$

□

Next, we provide a proof for the construction of neural extensions. Recall the statement of the main result.

Proposition 19. *Let $p_{\mathbf{x}}$ induce a scalar SFE of f. For $\mathbf{X} \in \mathbb{S}_+^n$ with distinct eigenvalues, consider the decomposition $\mathbf{X} = \sum_{i=1}^n \lambda_i \mathbf{x}_i\mathbf{x}_i^\top$ and fix*

$$p_{\mathbf{X}}(S,T) = \sum_{i=1}^{n}\lambda_i\, p_{\mathbf{x}_i}(S)p_{\mathbf{x}_i}(T) \text{ for all } S, T \subseteq [n].$$

Then, $p_{\mathbf{X}}$ defines a neural SFE $\mathfrak{F}$ at $\mathbf{X}$.

Proof. We begin by showing through the eigendecomposition of $\mathbf{X}$ that the $\mathfrak{F}$ defined by $p_{\mathbf{X}}(S,T)$ is dual SDP feasible. It is clear that $\sum_{S,T} p_{\mathbf{X}}(S,T) = 1$ as long as $\sum_{i=1}^n \lambda_i = 1$, which can be easily enforced by appropriate normalization of $\mathbf{X}$. Recall from the eigendecomposition we have $\mathbf{X} = \sum_{i=1}^n \lambda_i\mathbf{v}_i\mathbf{v}_i^\top$ where we have fixed each $\mathbf{v}_i \in [0,1]^n$ through a sigmoid. Using the scalar SFE $p_{\mathbf{x}}$ we may write each $\mathbf{v}_i$ as a

convex combination $\mathbf{v}_i = \sum_S p_{\mathbf{v}_i}(S)\mathbf{1}_S$. For each i we may use this representation to re-express the outer product of $\mathbf{v}_i$ with itself:

$$\begin{aligned}\mathbf{v}_i\mathbf{v}_i^\top &= \big(\sum_S p_{\mathbf{v}_i}(S)\mathbf{1}_S\big)\big(\sum_T p_{\mathbf{v}_i}(T)\mathbf{1}_T\big)^\top \\ &= \sum_S p_{\mathbf{v}_i}(S)^2\mathbf{1}_S\mathbf{1}_S^\top + \sum_{S\neq T} p_{\mathbf{v}_i}(S)p_{\mathbf{v}_i}(T)(\mathbf{1}_T\mathbf{1}_S^\top + \mathbf{1}_S\mathbf{1}_T^\top) \\ &= \sum_{S,T\subseteq[n]} p_{\mathbf{v_i}}(S)p_{\mathbf{v_i}}(T)(\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top)\end{aligned}$$

Summing over all eigenvectors $\mathbf{v}_i$ yields the relation $\mathbf{X} = \sum_{S,T\subseteq[n]} p_{\mathbf{X}}(S,T)(\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top)$, proving dual SDP feasibility.

Next, consider an input $\mathbf{X} = \mathbf{1}_S\mathbf{1}_S^\top$. In this case, the only eigenvector is $\mathbf{1}_S$ with eigenvalue $\lambda = |S|$ since $\mathbf{X}\mathbf{1}_S = \mathbf{1}_S(\mathbf{1}_S^\top\mathbf{1}_S) = \mathbf{1}_S|S|$. That is, $p_{\mathbf{X}}(T',T) = p_{\mathbf{1}_S}(T')p_{\mathbf{1}_S}(T)$.

For $\mathbf{X} = \mathbf{1}_S\mathbf{1}_S^\top$, $\mathbf{1}_S$ is clearly an eigenvector with eigenvalue $\lambda = |S|$ because $\mathbf{X}\mathbf{1}_S = \mathbf{1}_S(\mathbf{1}_S^\top\mathbf{1}_S) = \mathbf{1}_S|S|$. So, taking $\bar{\mathbf{1}}_S = \mathbf{1}_S/\sqrt{|S|}$ to be the normalized eigenvector of $\mathbf{X}$, we have $\mathbf{X} = |S|\bar{\mathbf{1}}_S\bar{\mathbf{1}}_S^\top = |S|\left(\frac{\mathbf{1}_S}{\sqrt{|S|}}\right)\left(\frac{\mathbf{1}_S}{\sqrt{|S|}}\right)^\top = p_{\mathbf{X}}(S,S)\mathbf{1}_S\mathbf{1}_S^\top$ for $p_{\mathbf{X}}(S,S) = 1$. Therefore, the corresponding neural SFE is

$$\mathfrak{F}(\mathbf{1}_S\mathbf{1}_S^\top) = p_{\mathbf{X}}(S,S)f(S\cap S) = f(S).$$

All that remains is to show continuity of neural SFEs. Since the scalar SFE $p_{\mathbf{x}}$ is continuous in $\mathbf{x}$ by assumption, all that remains is to show that the map sending $\mathbf{X}$ to its eigenvector with i-th largest eigenvalue is continuous. We handle sign flip invariance of eignevectors by assuming a standard choice for eigenvector signs—e.g., by flipping the sign where necessary to ensure that the first non-zero coordinate is greater than zero. The continuity of the mapping $\mathbf{X} \mapsto \mathbf{v}_i$ follows directly from Theorem 2 from Yu et al. [2015], which is a variant of the Davis–Kahan theorem. The result shows that the angle between the i-th eigenspaces of two matrices $\mathbf{X}$ and $\mathbf{X}'$ goes to zero in the limit as $\mathbf{X} \to \mathbf{X}'$. □

F.5 General Experimental Background Information

F.5.1 Hardware and Software Setup

All training runs were done on a single GPU at a time. Experiments were either run on 1) a server with 8 NVIDIA RTX 2080 Ti GPUs, or 2) 4 NVIDIA RTX 2080 Ti GPUs. All experiments are run using Python, specifically the PyTorch [Paszke et al., 2019] framework (see licence here). For GNN specific functionality, such as graph data batching, use the PyTorch Geometric (PyG) [Fey and Lenssen, 2019] (MIT License).

F.5.2 Data Details

This paper uses five graph datasets: ENZYMES, PROTEINS, IMDB-BINARY, MUTAG, and COLLAB. All data is accessed via the standardized PyG API. In the case of COLLAB, which has 5000 samples available, we subsample the first 1000 graphs only for training efficiency. All experiments Use a train/val/test split ratio of 60/30/10, which is done in exactly one consistent way across all experiments for each dataset.

F.6 Unsupervised Neural Combinatorial Optimization Experiments

All methods use the same GNN backbone: a combination of GAT Veličković et al. [2018] and Gated Graph Convolution layer [Yujia et al., 2016]. We use the Adam optimizer Kingma and Ba [2014] with initial $lr = 10^{-4}$ and default PyTorch settings for other parameters Paszke et al. [2019]. We use grid search HPO over batch size $\{4, 32, 64\}$, number of GNN layers $\{6, 10, 16\}$ network width $\{64, 128, 256\}$. All models are trained for 200 epochs. For the model with the best validation performance, we report the test performance and the standard deviation of performance over test graphs as a measure of method reliability.

F.6.1 Discrete Objectives

Maximum Clique. For the maximum clique problem, we could simply take f to compute the clique size (with the size being zero if S is not a clique). However, we found that this objective led to poor results and unstable training dynamics. So, instead, we select a discrete objective that yielded the much more stable results across datasets. It is defined for a graph $G = ([n], E)$ as,

$$f_{\text{MaxClique}}(S; G) = w(S)q^c(S),$$

where w is a measure of size of S and q measures the density of edges within S (i.e., distance from being a clique). The scalar c is a constant, taken to be $c = 2$ in all cases except REINFORCE for which $c = 2$ proved ineffective, so we use $c = 4$ instead. Specifically, $w(S) = \sum_{i,j \in S} \mathbf{1}\{(i, j) \in E\}$ simply counts up all the edges between nodes in S, and $q(S) = -2w(S)/(|S|^2 - |S|)$ is the ratio (with a sign flip) between the number of edges in S, and the number of undirected edges $(|S|^2 - |S|)/2$ there would be in a clique of size $|S|$. If G were directed, simply remove the factor of 2. Note that this f is minimized when S is a maximum clique.

Maximum Independent Set. Similarly for maximum independent set we use the discrete objective,

$$f_{\text{MIS}}(S; G) = w(S)q^c(S),$$

where w is a measure of size of S and q measures the number of edges between nodes in S (the number should be zero for an independent set), and $c = 2$ as before. Specifically, we take $w(S) = |S|/n$, and $q(s) = 2\sum_{i,j \in S} \mathbf{1}\{(i, j) \in E\}/(|S|^2 - |S|)$, as before.

F.6.2 Neural SFE details.

All Neural SFEs, unless otherwise stated, use the top $k = 4$ eigenvectors corresponding to the largest eigenvalues. This is an important efficiency saving step, since with

$k = n$, i.e., using all eigenvectors, the resulting Neural Lovaśz extension requires $O(n^2)$ set function evaluations, compared to $O(n)$ for the scalar Lovaśz extension. By only using the top k we reduce the number of evaluations to $O(kn)$. Wall clock runtime experiments given in Figure 8-3 show that the runtime of the Neural Lovaśz extension is around $\times k$ its scalar counterpart, and that the performance of the neural extension gradually increases then saturates when k gets large. To minimize compute overheads we pick the smallest k at which performance saturation approximately occurs.

Instead of calling the pre-implemented PyTorch eigensolver `torch.linalg.eigh`, which calls LAPACK routines, we use the power method to approximate the first k eignevectors of $\mathbf{X}$. This is because we found the PyTorch function to be too numerically unstable in our case. In contrast, we found the power method, which approximates eigenvectors using simple recursively defined polynomials of $\mathbf{X}$, to be significantly more reliable. In all cases we run the power method for 5 iterations, which we found to be sufficient for convergence.

F.6.3 Baselines.

This section discusses various implementation details of the baseline methods we used. The basic training pipeline is kept identical to SFEs, unless explicitly said otherwise. Namely, we use nearly identical model architectures, identical data loading, and identical HPO parameter grids.

REINFORCE. We compared with REINFORCE (Williams [1992]) which enables backpropagation through (discrete) black-box functions. We opt for a simple instantiation for the score estimator

$$\hat{g}_{\text{REINFORCE}} = f(S)\frac{\partial}{\partial\theta}\log p(S|\theta),$$

where $p(S|\theta) = \prod_{i\in S} p_i \prod_{j\notin S}(1-p_j)$, i.e., each node is selected independently with probability $p_i = g_\theta(\mathbf{y})$ for $i = 1, 2, \ldots, n$, where g_θ is a neural network and $\mathbf{y}$ some

input attributes. We maximize the expected reward, i.e.,

$$L_{\text{REINFORCE}}(\theta) = \mathbb{E}_{S\sim\theta}[\hat{g}_{\text{REINFORCE}}].$$

For all experiments with REINFORCE, the expected reward is computed over 250 sampled actions S which is approximately the number of function evaluations of neural SFEs in most of the datasets. Here, f is taken to be the corresponding discrete objective of each problem (as described earlier in section F.6.1). For maximum clique, we normalize rewards $f(S)$ by removing the mean and dividing by the standard deviation. For the maximum independent set, the same strategy led to severe instability during training. To alleviate the issue, we introduced an additional modification to the rewards: among the sampled actions S, only the ones that achieved higher than average reward were retained and the rewards of the rest were set to 0. This led to more stable results in most datasets, with the exception of COLLAB were the trick was not sufficient.

These issues highlight the instability of the score function estimator in this kind of setting. Additionally, we experimented by including simple control variates (baselines). These were: i) a simple greedy baseline obtained by running a greedy algorithm on each input graph ii) a simple uniform distribution baseline, where actions S were sampled uniformly at random. Unfortunately, we were not able to obtain any consistent boost in either performance or stability using those techniques. Finally, to improve stability, the architectures employed with REINFORCE were slightly modified according to the problem. For example, for the independent set we additionally applied a sigmoid to the outputs of the final layer.

Erdos Goes Neural. We compare with recent work on unsupervised combinatorial optimization [Karalias and Loukas, 2020]. We use the probabilistic methodology described in the paper to obtain a loss function for each problem. For the MaxClique, we use the loss provided in the paper, where for an input graph $G = ([n], E)$ and

learned probabilities $\mathbf{p}$ it is calculated by

$$L_{\text{Clique}}(\mathbf{p}; G) = (\beta + 1) \sum_{(i,j)\in E} w_{ij} p_i p_j + \frac{\beta}{2} \sum_{v_i \neq v_j} p_i p_j.$$

We omit additive constants as in practice they not affect the optimization. For the maximum independent set, we follow the methodology from the paper to derive the following loss:

$$L_{\text{IndepSet}}(\mathbf{p}; G) = \beta \sum_{(i,j)\in E} w_{ij} p_i p_j - \sum_{v_i \in V} p_i.$$

β was tuned through a simple line search over a few possible values in each case. Following the implementation of the original paper, we use the same simple decoding algorithm to obtain a discrete solution from the learned probabilities.

Straight Through Estimator. We also compared with the Straight-Through gradient estimator [Bengio et al., 2013]. This estimator can be used to pass gradients through sampling and thresholding operations, by assuming in the backward pass that the operation is the identity. In order to obtain a working baseline with the straight-through estimator, we generate level sets according to the ranking of elements in the output vector $\mathbf{x}$ of the neural network. Specifically, given $\mathbf{x} \in [0, 1]^n$ outputs from a neural network, we generate indicator vectors $\mathbf{1}_{S_k}$, where $S_k = \{j |\ x_j \geq x_k\}$ for $k = 1, 2, \ldots, n$. Then our loss function was computed as

$$L_{ST}(\mathbf{x}; G) = \frac{1}{n} \sum_{k=1}^{n} f(\mathbf{1}_{S_k}),$$

where f is the corresponding discrete objective from section F.6.1. At inference, we select the set that achieves the best value in the objective while complying with the constraints.

Ground truths. We obtain the maximum clique size and the maximum independent set size s for each graph by expressing it as a mixed integer program and using the

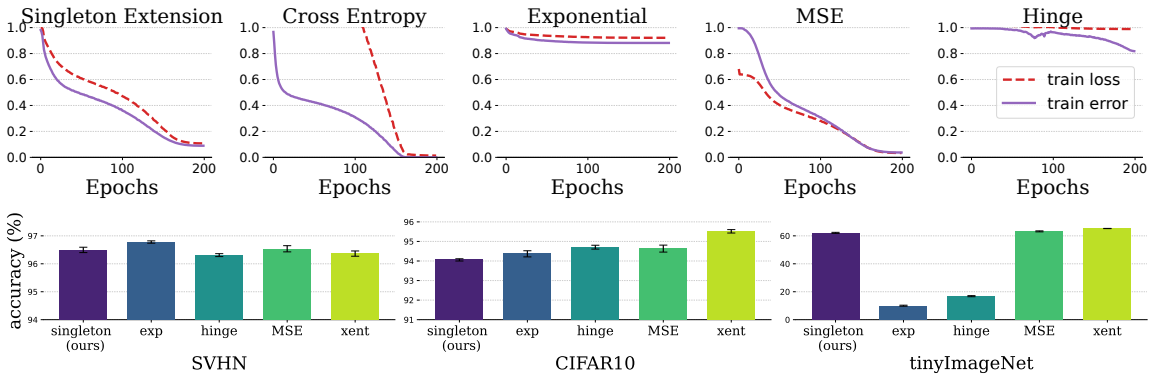


Figure F-1: Top: Additional experimental results on the tinyImageNet dataset. Bottom: test accuracies of different losses. The singleton extension performs broadly comparably to other losses.

Gurobi solver [Gurobi Optimization, LLC, 2021].

F.6.4 k-Clique Constraint Satisfaction

Ground truths. As before, we obtain the maximum clique size s for each graph by expressing it as a mixed integer program and using the Gurobi solver [Gurobi Optimization, LLC, 2021]. This is converted into a binary label $\mathbf{1}\{s \geq k\}$ indicating if there is a clique of size k or bigger.

Implementation details. The training pipeline, including HPO, is identical to the MaxClique setup. The only difference comes in the evaluation—at test time the GNN produces an embedding $\mathbf{x}$, and the largest clique S in the support of $p_{\mathbf{x}}$ is selected. The model prediction for the constraint satisfaction problem is then $\mathbf{1}\{|S| \geq k\}$, indicating whether the GNN found a clique of size k or more. Since this problem is. binary classification problem we compute the F1-score on a validation set, and report as the final result the F1-score of that same model on the test set.

F.7 Training error as an objective

Recall that for a K-way classifier $h : \mathcal{X} \to \mathbb{R}^K$ with $\hat{y}(x) = \arg\max_{k=1,\dots,K} h(x)_k$, we consider the training error $\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i \neq \hat{y}(x_i)\}$ calculated over a labeled training dataset $\{(x_i, y_i)\}_{i=1}^{n}$ to be a discrete non-differentiable loss. The set function in question is $y \mapsto \mathbf{1}\{y_i \neq y\}$, which we relax using the singleton method described in Section 8.3.1.

Training details. For all datasests we use a standard ResNet-18 backbone, with a final layer to output a vector of the correct dimension depending on the number of classes in the dataset. CIFAR10 and tinyImageNet models are trained for 200 epochs, while SVHN uses 100 (which is sufficient for convergence). We use SGD with momentum $mom = 0.9$ and weight decay $wd = 5 \times 10^{-4}$ and a cosine learning rate schedule. We tune the learning rate for each loss via a simple grid search of the values $lr \in \{0.01, 0.05, 0.1, 0.2\}$. For each loss we select the learning rate with highest accuracy on a validation set, then display the training loss and accuracy for this run.