Representation Learning Through the Lens of Science: Symmetry, Language and Symbolic Inductive Biases

by

Rumen Rumenov Dangovski

B. S., Massachusetts Institute of Technology (2018)
S. M., Massachusetts Institute of Technology (2020)

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: Rumen Rumenov Dangovski
Department of Electrical Engineering and Computer Science
August 31, 2023

Certified by: Marin Soljačić
Professor of Physics
Thesis Supervisor

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

Representation Learning Through the Lens of Science: Symmetry, Language and Symbolic Inductive Biases

by

Rumen Rumenov Dangovski

Submitted to the Department of Electrical Engineering and Computer Science
on August 31, 2023, in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

Abstract

In this thesis, we explore representation learning, a key technique in machine learning and artificial intelligence that has led to remarkable advancements in fields such as speech, vision, language perception and generation, as well as solving complex scientific problems like protein folding. Despite its success, the prevailing method of end-to-end supervised learning faces challenges, including the need for large datasets, non-interpretable classifications, and difficulties in transferring representations.

To address these limitations, we adopt a scientific perspective, focusing on machine learning tasks that are particularly affected by these issues, and developing benchmarks inspired by scientific principles. Our approach centers on the identification and development of novel inductive biases (assumptions made by the learning algorithm to improve generalization) based on symmetry, language, and symbolic properties. These inductive biases prove beneficial for both solving scientific problems using machine learning and enhancing representation learning methods.

We term this methodology "Representation Learning through the Lens of Science" and demonstrate its effectiveness in various applications. Finally, we discuss the limitations of our approach and propose directions for future research to further refine and expand upon the concepts introduced in this thesis.

Thesis Supervisor: Marin Soljačić
Title: Professor of Physics

Dedicated to Mom Tsetska and Dad Rumen
whose unconditional love gave me the courage
to become a researcher and live my dreams.

"One day Sendov said that we live on a small planet and the only way to stay on it is to live as friends. I would very much like this to come true."

—John Vincent Atanasoff (1985)

Children in the Information Era Conference

Varna, Bulgaria

Translated from Bulgarian

Acknowledgments

I am grateful to Professor Marin Soljačić who supervised, and Professors Isaac Chuang and Pulkit Agrawal who read the thesis.

To my cherished academic family:

Marin, thank you for the plethora of conversations about numerous ideas, from automating science journalism, through discovering laws in social sciences, to representation learning for materials science, etc. From the first minutes of our first meeting when I declared a double major in physics and you became my undergraduate advisor, I knew it would be an incredible journey doing research together. Little did I know that my life would be impacted in a profound way. A major reason why I wanted to pursue a Ph.D. is because I wanted to be part of the unique group you are leading. I learned not only how to be a researcher striving for impactful science and thorough engineering, but also how to be a supportive mentor and pass my experience effectively to fellow students. I look forward to our future collaborations and weekly reflections!

Charlotte, thank you for being the most caring and inspiring collaborator! Without you, my Ph.D. would not have been so memorable and meaningful. I look forward to many more of K-pop classes, walks with JJ and Xia, OccamAI, coffee breaks, life talks, etc.

Li, thank you for bringing me into machine learning research. Working with you is always an honor. Apart from a great mentor, you are my friend, and I hope

our collaborations continue to be fruitful. I hope to have many more energizing conversations in the Bay Area.

JJ, thank you for providing vital guidance on how to present my research to a wider audience who can benefit from my work. I truly appreciate your inspiring suggestions and look forward to future discussions.

Ike and Pulkit, thank you for encouraging me toward becoming an academic leader. I am working on building my "rocket" for "moonshots" by following a "back-propagation" approach that helps me to follow reflective and impactful research. Thank you for the fruitful discussions!

Ido and Nick, thank you for being my first mentors in Soljačić's Lab and helping me make my first steps. I admire your enthusiasm for science and that motivates me to work even harder.

Charles, thank you for being my meditation buddy and collaborator. Without you, I would not have felt so welcomed to Soljačić's group. I look forward to many more fruitful conversations in the Bay Area.

The entire Soljačić's lab, a huge thank you for making my work at MIT my dream job since 2016 when I first entered the lab. Special thanks to Jamison, Peter, Sam, Thomas, Ali, Di, Zhuo, Andrew, Yannick, Sachin, Shiekh, André, Seou, Yi, Simo, Josué, Tena, Hrvoje for all the collaborations, conversations, joint struggles and laughts, and everything memorable and dull that could happen to us Ph.D. students and postdocs :).

Preslav, Peter, Mohamed, Allan, Yung-Sung, Geeticka, Elyssa, Ileana, Ge, Brian, Akash and Seungwook and the MIT-IBM folks, Olga and Nick and Mark and the Air Force and Lincoln Lab folks, Daniel and the Meta folks, MLxMIT folks, thanks for all the collaboration and guidance!

Owen, Isaac, Viggo, Evan, Guillem, Varun, Ali, Anugrah, the Michaels, Michelle, Lay, Pawan, Dawson, and many other talented students I have the privilege to work with: many thanks for joining the journey of research with me! I hope we continue collaborating :).

Yichen, Paul, Spencer and the Lightelligence folks. Thank you for making me at

home in one of the most amazing ventures.

Google DeepMind, thank you for accepting me during a particularly challenging economic time. I look forward to our work :).

To my fellow friends:

Rumen and Momchil, thank you for being the most caring roommates. Growing up, in elementary and high school I dreamt to learn from you how to become a better programmer. It has truly been a dream coming true to live with you in the Boston area and also learn to be a better person.

GANG GANG—Yusu, Stan, Alex, and Ivan—thank you for the most "prestigious" and memorable friendship time of the last few years and onward. Yusu, let us devour more and more books together and learn about life! Stan, let us hunt more raccoons and dance till exhaustion with the New York gang! Alex, let us continue our walks and deep conversations about physics and life! Ivan, let us continue our conversations in the style of the Bulgarian events.

Kate, Gordon and the Baty Family, Rayna, Kris, Victor, Momchil, Noah, Allan, The Number Six Club, The Bulgarians at Boston, Sean, Jeremy, Liz and Darius and the Dog Gang, thank you for your friendship and all the happy moments across the US and Europe! :)

Microtonal Marauders—Polly, Kamen—Moni, George, the Alexes, Deni, Kalina, Milena, Stefan, Oleg and his Princesses and The Academy folks, thank you for being my Bulgarian support through the last few years. I look forward to many Odd Jobs and Mob Psychos together :).

János, Dora and Jason, thank you for being so helpful in the beginning of my Ph.D. Without your help, I would not have oriented myself toward research that fulfills me.

The Capoeira, Korean Karate, MIT Gymnastics, Ludo Mlado Folk Ensemble, Flamenco, and Tai Chi folks, thank you for keeping me sane throughout all these years through continual exercises of the body and mind!

Valeria, Sudhanshu, Edgar, Anton, Attilio, Thiago, Amanda, Eric, Emma, Shurong, Christina, Laura, Langelo, Eric, Yu-Na and the K-Pop gang — thank you for being

integral parts of my journey at MIT and the Boston area. I cherish our moments together!

Jenny and Pesho and The RSI folks, Krasi, Jenny and Vessi et al., thank you for all of your wisdom and unconditional friendship!

To my dearest family:

Mom, thank you for being with me always and loving me unconditionally in such a strong way. Our deep bond defies the physical distance between us. From *Treasure Island* to artificial (and natural:) intelligence, we will be discussing various states of affairs in person for many decades ahead.

Dad, thank you for lighting the fire within me to pursue crazy projects with grand and beautiful impacts —- from the "sink" project to shipbuilding *Santa Maria*. Your love and support are greatly appreciated. We will continue holding our conversations over walks in nature for many decades ahead.

Dragana Holding—Baba Genka, Dad, Mom, Itcheto, Tomi, Tsveti, Yuli, Kosio—thank you for your love and support. You have been a foundation for me.

Xia, I am blessed to have met you and found a soulmate. Our love sparked a vision for AI and education that is truly one of the brightest I have felt in my life.

"Онзи ден Сендов каза, че ние живеем на малка планета и че единственият начин да останем на нея е да живеем като приятели. Много бих искал това да се осъществи."

—John Vincent Atanasoff (1985)

Children in the Information Era Conference

Varna, Bulgaria

Contents

1 Introduction **31**

1.1 Big Question 31

1.2 Goals 33

1.3 Prior Art 34

1.4 New Question 38

1.5 Ideas 39

1.6 Results 40

2 Addressing the Need for Data with the Symmetry Inductive Bias **43**

2.1 Introduction 43

2.1.1 Symmetry 43

2.1.2 Need for Data 44

2.2 Equivariant Contrastive Learning 47

2.2.1 Introduction 47

2.2.2 Related Work 51

2.2.3 Method 53

2.2.4 Experiments 56

2.2.5 Discussion 60

2.2.6 Appendix: Summary of Main Text and Layout of Appendix . 62

2.2.7 Appendix A: Proof of Proposition 1 63

2.2.8 Appendix B: Rotation prediction and I-SSL benefit from similar data augmentation. 64

2.2.9 Appendix C: CIFAR-10 Experiments 65

2.2.10 Appendix E: ImageNet Experiments 69
2.2.11 Appenix F: PhC Experiments 70
2.3 Flowers-102 Experiments 72
2.3.1 Appendix G: Relative Orientation Prediction 74
2.4 Multi-Symmetry Ensembles 76
2.5 DiffCSE: Difference-Based Contrastive Learning for Sentence Embeddings 81
2.6 Surrogate- and Invariance-Boosted Contrastive Learning for Data-Scarce Applications in Science 86
2.7 Contrastive Learning for Stormy Event Imagery 90
2.7.1 Storm Event Imagery 90
2.7.2 Benchmark Construction 91
2.7.3 Methods 92
2.7.4 Experimental Details 93
2.7.5 Self-supervised Pre-training 94
2.8 Conclusion 97

3 Addressing the Ability to Transfer with the Language Inductive Bias 99
3.1 Introduction 99
3.2 Towards Automating Science Journalism at Scale 101
3.2.1 Introduction 101
3.2.2 Related Work 103
3.2.3 The *Science Daily* Dataset 105
3.2.4 Evaluation 111
3.2.5 Experiments 112
3.2.6 Discussion 118
3.2.7 Conclusion and Future Work 120
3.3 Conclusion 122

4 Addressing the Lack of Interpretability with the Symbolic Inductive Bias 123

4.1 Introduction . . . 123
4.2 OccamNet: A Fast Neural Model for Symbolic Regression at Scale . . 125
4.2.1 Introduction . . . 125
4.2.2 Model Architecture . . . 127
4.2.3 Training . . . 130
4.2.4 Results . . . 131
4.2.5 Discussion . . . 139
4.2.6 Methods . . . 141
G PMLB Experiment Results . . . 169
H Analysis of Fits to PMLB Datasets . . . 169
I Analysis of PMLB Scaling Tests . . . 174
J Ablation Studies . . . 176
K Neural Approaches to Benchmarks . . . 178
L Small Experiments . . . 180
M Related Work . . . 181
N Symbolic Regression Benchmarks . . . 183
O Code . . . 185
4.3 AI-Assisted Discovery of Quantitative and Formal Models in Social Science . . . 186
4.4 Phase Transitions and Representation Geometry in Contrastive Learning 196
4.5 Conclusion . . . 203

5 Beyond the Limitations of *Representation Learning Through the Lens of Science*. Promising Directions of Future Work. 205
5.1 Future Work on the Symmetry Inductive Bias: *Multimodality* . . . 206
5.2 Future Work on the Language Inductive Bias: *Large Language Models for Science Education and Research.* . . . 210
5.3 Future Work on the Symbolic Inductive Bias: *OccamNet as Representation in Novel Domains* . . . 211
5.4 Final Remarks . . . 213

List of Figures

2-1 SSL representations should be encouraged to be either insensitive or sensitive to transformations. The baseline is SimCLR with random resized cropping only. Each transformation on the horizontal axis is combined with random resized cropping. The dataset is CIFAR-10 and the kNN accuracy is on the test set. More experimental details can be found in Section 4.2.4. 49

2-2 E-SSL framework. Left: framework. Right: methods. Egomotion, Context, Colorization and Jigsaw use other transformations than rotations, but their patterns looks like that of RotNet's. Likewise, for E-SSL can use transformations different from rotation. 53

2-3 Sketch of E-SSL with four-fold rotations prediction, resulting in a backbone that is sensitive to rotations and insensitive to flips and blurring. ImageNet example n01534433:169. 55

2-4 Reducing the labels for training and the data augmentation for pretraining on CIFAR-10. Error bars for 5 different training data splits. 60

2-5 PhC datasets with transformations for sensitivity. The regression task is to predict the DOS labels (an example of a label in $\mathbb{R}^{400}$ is shown on the right) from 2D square periodic unit cells (examples of the inputs in $\mathbb{R}^{32\times32}$ are shown on the left). We consider two types of input unit cells; at the top is the Blob dataset where the feature variation is always centered; at the bottom is the Group pm (Gpm) dataset where inputs have a horizontal mirror symmetry. 61

2-6 Demonstration of the evolution of the invariance (top) and equivariance (bottom) measures during training. Left is E-SimCLR and right is E-SimSiam. 69

2-7 E-SimCLR gives sizable improvements for the Flowers-102 SSL pre-training. kNN accuracy (%) is on the validation set. 74

2-8 The Flowers-102 is not completely invariant to rotation. The top row shows data points which are roughly invariant to four-fold translations. The bottom row shows counterexamples to that hypothesis. 75

2-9 **(a)** A comparative illustration of the diversity in the hypothesis space that traditional deep ensembles and our MSE can achieve. While deep ensembles are effective at capturing different solutions around one hypothesis, MSE can learn diverse solutions around inherently opposing hypotheses. **(b)** Schematic visualization of invariance (top) v.s. equivariance (bottom) for the four-fold rotation. The spheres denote the representation space of the models. 78

2-10 **Ensembles with opposing hypotheses have significantly larger potential.** Ensembles constructed only from a single hypothesis very quickly give marginal ensembling gains from adding more members. . 79

2-11 **Understanding the effectiveness of including the opposing hypothesis.** Plot shows the proportion of classes in each dataset where each hypothesis dominates. The remaining proportions (not shown) are classes where Eq and Inv are equally performant. Gains are minimal in datasets with a high level of imbalance between the leading and opposing hypothesis. 81

2-12 **Examples of images from the "jellyfish" class in ImageNet (left) and ImageNet-R (right).** Samples visualized using `https://knowyourdata-tfds.withgoogle.com/` 82

2-13 Illustration of DiffCSE. On the left-hand side is a standard SimCSE model trained with regular contrastive loss on dropout transformations. On the right hand side is a conditional difference prediction model which takes the sentence vector **h** as input and predict the difference between x and x''. During testing we discard the discriminator and only use **h** as the sentence embedding. 83

2-14 **Overcoming data scarcity with SIB-CL.** We propose to overcome data scarcity by leveraging a) an abundance of unlabeled data, b) prior knowledge of the underlying physics (e.g., symmetries and invariances of the data) and c) knowledge from a possibly-approximate surrogate data which is faster and cheaper to generate (e.g., coarse-grained computations or special-case analytical solutions). d) SIB-CL incorporates these auxiliary information into a single framework to accelerate training in data-scarce settings. 87

2-15 **Network prediction results for PhC-DOS problem.** a) Network prediction error against fine-tuning dataset sizes, N_{t}, between 50 and 3000, when using our SIB-CL framework (based on SimCLR [Chen et al., 2020a] during contrastive learning) compared against the baselines: direct supervised learning (SL) and standard approaches involving transfer learning (TL) or involving invariances via data augmentation (SL-I). A 9-fold (7-fold) reduction in target data requirements is obtained by using SIB-CL over SL (SL-I, TL) at a relative error target of $\sim 5.1\%$. Error bars show the 1σ uncertainty level estimated from varying the data selection of D_{t}. b) Examples of the DOS spectrum predicted by the SIB-CL-trained network compared against the actual DOS at various error levels (insets depict associated unit cells, shown here using the network-inputs' resolution of 32×32). 88

2-16 Comparison of SIB-CL with an equivariant network E2CNN [Weiler and Cesa, 2019]. E2CNN is trained using supervised learning (E2CNN-SL) as well as fine-tuned after an additional pre-training stage using the surrogate dataset (E2CNN-TL). The supervised baseline (SL) using the non-equivariant architecture that SIB-CL uses is also included for comparison. Error bars show the 1σ uncertainty level when varying the data selection of the fine-tuning dataset. 90

2-17 **Frame from The Storm Event Imagery (SEVIR) dataset.** We use four of the five available modalities: 2 IR, VIL, and lightning information. 91

2-18 **Augmentations for the contrastive learning experiment** By indicating "more" we show examples of a larger magnitude of the augmentation being applied. 96

2-19 **Contrastive Learning for SEVIR.** For mean absolute error lower is better. For every other evaluation measure, higher is better. 96

2-20 **Pretrained encoder - generated samples** Pretrained models better identify the sparse high VIL values. 97

3-1 Histogram for number of articles vs. number of words for the selected publishers in *Science Daily*. Stars indicate modes of the histograms (excluding the outliers with fewer than 1,000 words for Elsevier). . . . 108

3-2 Density vs. coverage of source-target pairs for *Science Daily*, *ArXiv*, *PubMed*, and *CNN and Daily Mail*. Warmer colors show more data entries, and # is number of pairs. Outliers with extreme densities are omitted. Arrows indicate the modes of the datasets. 109

3-3 Positions of the source sentences that maximize the NLI entailment of the summary sentences for *Science Daily*. On the left are gold summaries, and on the right are summaries by our model (*Story+Parts*). The counts are normalized, so that the bin with the highest counts is at 1.0. 110

4-1 OccamNet architecture and training. **a.** OccamNet is a stack of "symbolic layers" each described by a collection of learned distributions (over the neurons from the previous layer) for each neuron within the layer, as well as non-linearities that are collections of symbolic expressions. **b.** By sampling from each distribution independently, we are able to sample paths from OccamNet that represent symbolic expressions, ready for evaluation. **c.** We evaluate each expression by feeding the observations' support data and comparing the outputs with the ground truth. The probability of the best paths is increased and the process is repeated until convergence. 127

4-2 Experiment on analytic functions. **a.** A sketch of the function $\sum_{n=1}^{3} \sin(nx)$ as an example of the analytics functions we consider in our work. **b.** Success rate (out of 10 trials) for each of the five methods considered: OccamNet, Eureqa, Eplex, AI Feynman 2.0 (AIF) and Deep Symbolic Regression (DSR) (at the top). Training time for the methods (at the bottom). Eureqa almost always finishes much more quickly than the other methods, so we do not provide training times for Eureqa. We enumerate the functions to ease the discussion. **c.** The "worst-case" performance for each methods, showing the minimal success rate across the six tasks. 132

4-3 Experiments on non-analytic functions. **a.** Two prominent examples of non-analytic functions: The challenging recursion $g(x) = x^2$ if $x < 2$, else $x/2$, $y(x) = g^{\circ 4}(x)$ (top) and a sorting circuit of three numbers (bottom). **b.** Success rate (out of 10 trails) and training time for OccamNet and Eplex. We enumerate the functions to ease the discussion. 134

4-4 Experiments on implicit functions and standard vision benchmarks. **a.** Examples of implicit functions' loci (left) and the corresponding success rate on a suite of implicit functions (right). **b.** Examples of image recognition tasks (left) and the best accuracy from 10 trials for both OccamNet and the baseline. The baseline for MNIST Binary/ Trinary and ImageNet Binary is HeuristicLab Wagner et al. [2014]. The baseline for Backprop OccamNet and Finetune ResNet is a feed-forward neural network with the same number of parameters as OccamNet. . 135

4-5 A bar chart showing the relative performance between OccamNet and two baseline methods, Eplex and AIF. The x-axis is the dataset involved. The y-axis is the relative performance according to the given metric: the MSE on the training, validation, or testing set or the training time. To compute this relative performance, we divide the higher (worse) performance value by the lower (better) performance value for each dataset. The green bars represent datasets where OccamNet has a lower (better) performance value than the comparison baseline method, and the red bars represent the datasets where the comparison method has a better performance than OccamNet. 138

4-6 *Left:* The run time for OccamNet V100 or Eplex as a function of the number of functions sampled per epoch. Each curve represents one of the 15 datasets. *Right:* Gradual modularity with training. Dark blue is the correct function. Light blue is a suboptimal fit with a high probability early in training. Red corresponds to the correct function. The insets show the first sample of the correct function. 139

4-7 (a) A two-output network model with depth $L = 2$, $\vec{x} = [x_0, x_1]$, user-selected constants $\mathcal{C} = [1, \pi]$, and basis functions $\mathbf{\Phi} = (+(\cdot,\cdot), \sin(\cdot), (\cdot)^2, \times(\cdot,\cdot))$. Highlighted are the arguments sublayer, composed of P-nodes, and the images sublayer, composed of the basis functions from $\mathbf{\Phi}$. Together, these two sublayers define a single layer of our model. (b) An example of function-specifying directed acyclic graphs (DAGs) that can be sampled from the network in (a). These DAGs represent the functions $y_0 = \sin^2(x_0 + 1)$ and $y_1 = \sin(\pi^2 \sin(x_1))$. 142

4-8 A demonstration of the dropped connections from sampled paths in OccamNet. All red paths are dropped from the final symbolic form of the sampled function because they are not directly connected to the outputs. These paths are unnecessarily computed during OccamNet's training process, leading to potential slowdowns in training. 144

4-9 Skip connections. Dotted lines and color: the origin of the reused neurons. 148

4-10 OccamNet V100 and Eplex's Training, Validation, and Testing MSE as a function of run time for the 15 PMLB datasets discussed above. . 170

4-11 OccamNet V100 and Eplex's Training, Validation, and Testing MSE as a function of run time for the 15 PMLB datasets discussed above. For this figure, we only consider the losses for a restricted subset of hyperparameter combinations. 171

4-12 In this figure, we present two video frames for the target $\sin(3x + 2)$, which could be accessed via `videos/sin(3x + 2).mp4` in our code files. We show the beginning of the fitting (left) and the end, where OccamNet has almost converged (right). 185

4-13 In this figure we present two video frames for the target $\mathsf{SORT}(x_0, x_1, x_2)$, which could be accessed via `videos/sorting.mp4` in our code files. We show the beginning of the fitting (left) and the end, where OccamNet has almost converged (right). 186

4-14 Using neuro-symbolic regression to systematically guide model discovery in social science. Analogous to the inductive-deductive reasoning process, a dataset of interest (1) – which may be time-series, cross-sectional, or longitudinal – is supplied to OccamNet. The user can provide inductive biases (2), such as the choice of key variables, known constants, or specific functional forms to constrain the search space. OccamNet finds interpretable and compact solutions that model the input data by sampling functions from an internal probability distribution represented using *P-nodes* [Costa et al., 2021]. In this example, OccamNet recovers the governing equation of the Solow-Swan model of economic growth [Solow, 1956] from a synthetic dataset. Each formal model is characterized by its error distribution in the training set (3), allowing the user to identify outliers and interrogate its internal validity. The symbolic model is then used to generate predictions (4) to perform deductive tests across unseen data, either by partitioning a test set or informing experimental designs (5). 187

4-15 Regression of coupled dynamical models using noisy real-world data. **(a)** Time-series plot of a simulated Lotka-Volterra predator-prey system. OccamNet was able to correctly reconstruct the functional form and constants with high accuracy. **(b)** Using cubic spline interpolation, our system was able to learn the two differential equations from noisy, real-world data of lynx and hare populations with just 21 data points each. The inferred non-linear model can then be used to extend predictions of future populations. **(c)** The symbolic regression system is used to infer the SIR model of pandemic spread in synthetic data and **(d)** an ensemble of real-world measles infection data in the UK. . 189

4-16 Ensemble learning of longitudinal (panel) macroeconomic data. **(a)** Country-level macroeconomic data on capital and income per capita, savings rates, and population growth for 18 OECD member countries. **(b)** Ensemble learning of the Solow economic growth model. The error distribution of the differential equation, applied to each country, is shown for three expressions generated with increasing levels of complexity regularization. The identification of outliers may inform alternative explanations, hidden parameters, or higher-order corrections to the economic model. 194

4-17 Main result. Augmentations used in contrastive learning typically exhibit a tradeoff between robustness (how much noise is removed) and destructiveness (how much signal is removed). In order to explore the effect of robustness alone, we study non-destructive augmentations which preserve all relevant data. We observe in multiple settings that increased robustness speeds up training by speeding up the occurrence of phase transitions, points in training at which representation geometry changes suddenly and significantly. Each numbered point in the first plot corresponds to the augmentations of the idealized training curve on the right. 197

4-18 How the data generation for our Kepler dataset works. Images on bottom right are real examples sourced from the dataset. 199

4-19 Phase transition from Twisted Disk to Bowl showing transitory intermediate embedding geometry. 201

4-20 Visual comparison showing poor alignment in Twisted Disk, but not in the Bowl. Black outline is embedding of a single orbit. 202

4-21 Number of epochs needed until the phase transition for Kepler charted against percentage of trajectory seen during training. 16 trials for each α plotted, with a 95% confidence interval shown. 202

4-22 The phase transition in this run on the Kepler dataset can be observed from epochs 65-85 where there is a sudden and surprising increase in the R^2 metric. At the same time, loss does not sharply decrease. . . . 202

5-1 Multimodal representation learning for crystaline materials. 207

5-2 Our fine-tuning of LLaMA on an example question. The LLM can understand the question and remember the correct formula for the hydrostatic pressure exerted on the submarine. 211

5-3 OccamNet as a representation for LLMs. **a.** The standard information flow for Transformers where the encoder layers updates the representations of each token in the input sequence. **b.** Extracting the numerical tokens from the tokenized input. **c.** A special token representation from the LLM is used to configure the weights of an OccamNet that receives the numerical tokens as input and produces a numerical output that is used by the following encoder layer. The process is repeated for all the layers of the Transformer. 213

List of Tables

2.1 Parallel between self-supervised learning stages and physics process. . 46

2.2 Linear probe accuracy (%) on CIFAR-10. Models are pre-trained for 800 epochs. Baseline results are from Appendix D in [Chen and He, 2021b]. Standard deviations are from 5 different random initializations for the linear head. Deviations are small because the linear probe is robust to the seed. 59

2.3 Linear probe accuracy (%) on ImageNet. Each model is pre-trained for 100 epochs. Baseline results are from Table B.1 in [Chen et al., 2020a] from Table 4 in [Chen and He, 2021b]. Numbers marked with * use a less optimal setting than our reproduction for SimCLR (see ImageNet setup). 60

2.4 Linear probe accuracy (%) on ImageNet with longer pre-training. "BT" is short for "Barlow Twins." 61

2.5 Fine-tuning the backbone on PhC datasets using 3000/ 2000 labeled train/ test samples. Relative error (%) is $\ell_{\mathrm{DOS}} = (\sum_{\omega} \left|\mathrm{DOS}^{\mathrm{pred}} - \mathrm{DOS}\right|)/(\sum_{\omega} \mathrm{DOS})$. Lower is better. SimCLR for Blob includes C_{4v} (rotations and flips); SimCLR for Gpm includes rolling translations and mirrors. E-SimCLR encourages the features to be sensitive to the selected transformation explained in the text (four-fold translations for Blob and four-fold rotations for Gpm). "+ Transform" means adding this transformation to SimCLR. Error bars are for 3 different training data splits. 62

2.6 RotNet's augmentation sweet spot. kNN and Rotation Prediction have the same sweep spot (Level 4) which gives best accuracy in both columns. RotNet is trained on CIFAR-10 for 100 epochs with the same optimization setup as in our I-SSL experiments. Accuracies are on the test split. ($\downarrow \cdot$) marks the deviation from the sweet spot. Every new level adds a new augmentation to the previous level incrementally. . . 65

2.7 Tuning the λ parameter for CIFAR-10. 66

2.8 Comparing the augmentation sensitivity for CIFAR-10. Levels: 0 is no transformations; 1 adds random resized cropping; 2 adds horizontal flips; 3 adds color jitter; 4 adds grayscale. 67

2.9 Studying the effect of disjoint representations on CIFAR-10. Split Representation means that we encourage similarity only on one half of the backbone representation. Disentangled Representation means that one half of the representation is trained to be insensitive to four-fold rotations and the other half is sensitive four-fold rotations. Linear probe accuracy (%) after 800 epochs. 67

2.10 Overhead in doing rotation prediction. Reported GPU hours for an experiment on 100 epochs. 70

2.11 Frozen backbone experiment on PhC datasets for 3000/ 2000 labelled train/ test samples. 72

2.12 Fine-tuning the backbone on PhC datasets using 3000/ 2000 labelled train/ test samples. Relative error (%) is $\ell_{\text{DOS}} = (\sum_{\omega} \left|\text{DOS}^{\text{pred}} - \text{DOS}\right|)/(\sum_{\omega} \text{DOS})$. Lower is better. E-SimCLR encourages the features to be sensitive to scaling. "+ Scaling" means adding scaling to SimCLR. Error bars are for 3 different training data splits. 73

2.13 **Most suitable functional class differs within a dataset.** The top-half shows the overall accuracy for models from the SimCLR baseline and each of the opposing hypotheses wrt 4-fold rotations. The bottom-half shows the proportion of classes within each dataset where each hypotheses dominate (i.e. averaged over all samples within the class), suggesting that hypotheses apart from the one with the highest individual accuracy are still beneficial. 77

2.14 **Capturing opposing hypotheses across transformations for $M = 6$.** The upper three rows are ensembles that consist of both equivariant and invariant learners with respect to a single transformation and the bottom row greedily searches over all models across the three transformations. 80

2.15 The performance on STS tasks (Spearman's correlation) for different sentence embedding models. ♣: results from Reimers and Gurevych [2019]; ♡: results from Zhang et al. [2020]; ♢: results from Gao et al. [2021]; ♠: results from Yang et al. [2020]; †: results from Kim et al. [2021b]; ∗: results from our experiments. 86

2.16 Computational times for building the network and performing a single forward pass of the network in SIB-CL vs in E2CNN models of equal number of parameters (approx. 8M parameters). 89

3.1 Summary from our dataset (*short Science Daily) using our model (*SciBertSumAbs*).* We see the need for extreme paraphrasing and coherent generation. . 102

3.2 Statistics about the *Science Daily* datasets. 106

3.3 *Science Daily* covers diverse journals. 106

3.4 Complexity of related datasets' sources based on readability scores such as SMOG, CLI, and LIX. The datasets from scientific sources (the top half) use more complex language (bigger numbers indicate higher complexity). 110

3.5 *Short Science Daily*: SciBERT pre-training improves over vanilla BERT (ROUGE scores in %). LEAD takes the first 45 words from the input. 113

3.6 *Long Science Daily*: baselines. *Fconv* outperforms *Story* in ROUGE 1/2/L and Prompt Ranking (PR); top-k sampling generally helps for *Fconv*. PR does not depend on the decoding scheme. LEAD takes the first 488 input words. 114

3.7 *Long Science Daily*: Training with *ArXiv*. We can observe sizeable and consistent improvements. 115

3.8 Training in parts yields improvements: sizable for Prompt Ranking, but partial for ROUGE 1/2/L. 116

3.9 Summary from *long Science Daily.* Shown are some snippets (generated, gold, and original) when using the *Story* model with top-k sampling and data augmentation using *ArXiv* (*StoryTopK+Arxiv*). . 119

3.10 Manual expert analysis of the utility of models trained with *SciBertSumAbs* on *short Science Daily.* See the text for a definition of the criteria and their abbreviations. Legend: Y=Yes, N=No, P=Probably, ML=Most Likely, NMT=Needs Minor Tweaks, NFT=Needs Few Tweaks. 121

4.1 Analytic Functions. The proportion of 10 trials that converge to the correct analytic function for OccamNet, Eureqa, Eplex, AI Feynman 2.0, and Deep Symbolic Regression (DSR). *sec.* is the average number of seconds for convergence. Eureqa almost always finishes much more quickly than the other methods, so we do not provide training times for Eureqa. 142

4.2 Non-analytic Functions. The proportion of 10 trials that converge to the correct function for OccamNet, Eureqa, and Eplex. *sec.* is the average number of seconds for convergence. Eureqa almost always finishes much more quickly than OccamNet and Eplex, so we do not provide training times for Eureqa. *For program #6, Eplex fits y_1 every time and never fits y_0 correctly, so we give it a score of 0.5. . . 143

4.3 Implicit Functions: The proportion of 10 trials that converge to the correct implicit function for OccamNet and Eureqa. Image Recognition: The best accuracy from 10 trials for both OccamNet and the baseline. The baseline above the mid-line is HeuristicLab [Wagner et al., 2014], and the baseline below the mid-line is a feed-forward neural network with the same number of parameters as OccamNet. *sec.* is the average number of seconds for convergence. The baselines almost always finish much more quickly than OccamNet, so we do not provide baseline training times. 144
4.4 Basis functions tested for clustering 159
4.5 Hyperparameters for Experiments Where $E = 0$ 165
4.6 Hyperparameters for Experiments Where $E = 1$ 165
4.7 Datasets Tested 166
4.8 OccamNet Hyperparameters 166
4.9 Number of Functions Sampled Per Epoch 167
4.10 Ablation studies on representative experiments 177
4.11 Minimal configurations to sort list of length "input size." 179

Chapter 1

Introduction

1.1 Big Question

Inductive Biases. In the era of vast data repositories and powerful computers, it is undeniable that learning algorithms play a crucial role in automating mundane tasks and creative endeavors. A central question to researchers in machine learning is "*How to represent data and design algorithms?*" Representation learning is a subfield of machine learning focused on developing methods for automatically learning feature representations from raw data, which can be used to improve the performance of various tasks and domains [Bengio et al., 2013]. The central question in representation learning is determining the most effective inductive biases to facilitate learning these representations. Inductive biases refer to the assumptions or predispositions made by a learning algorithm, which influence the model's ability to generalize from limited training data [Mitchell, 1980]. These biases can be encouraged through factors such as *architecture*, *training objectives*, *choice of data*, etc.

The Norm. *End-to-end supervised learning* has become a very popular approach to representation learning. This process involves training deep neural networks using gradient descent on a dataset containing raw data points and their corresponding labels [Rumelhart et al., 1985]. This approach has gained popularity because, given enough data and training, it allows the model to automatically learn representations

of data that are optimal for the task at hand, often leading to improved performance compared to manually designed features [LeCun et al., 2015a].

In 2012, a pivotal event in the field of deep learning, often referred to as the "ImageNet moment," marked a significant turning point for the advancement of machine learning. This breakthrough was achieved by AlexNet, a deep convolutional neural network (CNN) [LeCun et al., 1998a], which outperformed all other competitors by a wide margin in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Deng et al., 2009b, Russakovsky et al., 2015]. The ILSVRC, based on the extensive ImageNet dataset, consisted of over 14 million labeled images spanning 20,000 classes, and aimed at evaluating the performance of algorithms for object detection and image classification. AlexNet's remarkable success in this competition not only demonstrated the potential of deep learning models in the realm of computer vision, but also laid the foundation for further research and development in the field. As a result, end-to-end supervised learning has become the standard approach (the *norm*) in the field.

Limitations. End-to-end supervised learning is not without limitations. Specifically, this approach often:

(i) requires extensive human annotation via crowd-sourcing efforts [Russell et al., 2007, Russakovsky et al., 2015], which sometimes can also yield wrong or ambiguous labels;

(ii) leads to a lack of interpretability in representations despite their confident predictions. This lack of transparency and understanding in decision-making is a critical concern, especially when errors are made with high confidence [Nguyen et al., 2015, Arazo et al., 2020].

(iii) may result in learned representations that can only be modestly transferred to other tasks with less available data [Huh et al., 2016, He et al., 2020].

The problems *(i-iii)* are especially pronounced in the realm of Science. Regarding *(i)*: While fields like computer vision and natural language processing successfully ex-

ploit the vast Internet data for training and development, scientific domains such as bioinformatics and materials science face unique challenges due to the need for highly-specialized expertise in data labeling. For instance, predicting a protein's folding can be so intricate that it becomes the core focus of a PhD thesis, as exemplified in the AlphaFold study by Jumper et al. [2021]. Similarly, in materials science, identifying and characterizing novel materials often demands specialized knowledge and extensive research projects. As for *(ii)*: In the medical domain, predictions must be interpretable and well-calibrated to gain acceptance from doctors. Finally, regarding *(iii)*, complex systems like materials have been investigated across various scientific disciplines, such as chemistry, quantum physics, and lab experiments. However, these fields have diverse representations, complicating transfer learning and establishing interdisciplinary connections for learning algorithms.

To tackle issues *(i-iii)*, it is crucial to emphasize the deliberate development of inductive biases that promote generalization. Additionally, considering the pronounced impact of *(i-iii)* in Science, examining these challenges from the perspective of specific scientific fields serves as a key motivation for the goals outlined in our study.

1.2 Goals

Considering the challenges associated with end-to-end supervised learning, our research aims to enhance representation learning by developing novel inductive biases. The key goals are:

1. Minimize dependence on extensive human annotation, enabling models to learn effectively from limited or unsupervised data, accelerating the learning process, and reducing the burden on human experts.

2. Increase classifier interpretability, making them more useful for practical deployment in computer vision and natural language processing, as well as for making scientific discoveries with machine learning.

3. Improve the transferability of learned representations, facilitating the creation

of more general and versatile models that can efficiently adapt to new tasks and domains across multiple scientific fields.

4. Deploy our work on Goals 1-3 to make advancement in scientific fields, either by accelerating prediction or providing novel insights by inspecting the learned representations.

Objectives 1-3 can be accomplished through the development of innovative neural network architectures, training objectives, and judicious data selection. Additionally, Goal 4 can leverage the progress made in these areas. In the following section, we will review the relevant literature around 1-4.

1.3 Prior Art

Developing neural network architectures, refining self-supervised learning techniques, and enhancing transfer learning approaches are all effective strategies for addressing the challenges posed by Problems 1-3. We will begin discussing those research direction, and will conclude with some applications in Science.

Neural Network Architectures. Over the years, a variety of neural network architectures have been developed to address different tasks and modalities. Convolutional Neural Networks (CNNs) [LeCun et al., 1998a] have been widely used for computer vision to encourage the inductive bias of equivariance to translations. Equivariance to more general transformations has been pioneered by the more general equivariant neural networks [Cohen and Welling, 2016] and explored more broadly in the framework of geometric deep learning [Bronstein et al., 2021].

For sequential data Recurrent Neural Networks (RNNs) [Hochreiter and Schmidhuber, 1997, Pascanu et al., 2013], in which a hidden state is recurrently updated by the neural network through the time dimension, have been leading the fields in speech recognition, natural language processing, etc. In contrast to the local updates of internal representations found in CNNs and RNNs, attention mechanisms, such as

those introduced by Bahdanau et al. [2014] and Hermann et al. [2015], enable global updates. Most notably, the self-attention mechanism in Transformers [Vaswani et al., 2017b] has achieved remarkable success in natural language processing, effectively supplanting RNNs as the dominant architecture.

A longstanding problem in the field has been the issue of exploding and vanishing gradients in deep computational graphs, exacerbated by the chain rule of backpropagation. Residual Networks (ResNets) [He et al., 2016b] have been developed to tackle vanishing gradient issues in deep networks, by allowing for a simple residual connection that adds the input to the output of each intermediate layer.

Neurosymbolic architectures [Trask et al., 2018b, Martius and Lampert, 2016, Sahoo et al., 2018b, Kim et al., 2020a] integrate symbolic reasoning into representation learning with neural networks, which allows for more interpretable solutions. Other architectures aim for compositions of the internal representations as a gated mixture of experts [Shazeer et al., 2017].

There is a prevailing trend in neural network design towards universality, aiming to apply a single architecture across various modalities. Notable examples include the adaptation of Transformers to vision tasks [Dosovitskiy et al., 2020] and the Perceiver, which employs iterative attention on any tokenized input [Jaegle et al., 2021]. This interplay between modalities has fostered the development of increasingly efficient architectures. However, when it comes to scientific advancements, specialized architectures, such as AlphaFold's customized Transformer for protein folding [Jumper et al., 2021], have also demonstrated significant effectiveness. The quest for a universally applicable architecture remains an open challenge.

Self-supervised Learning. Self-supervised learning methods have emerged as powerful alternatives to supervised learning because they do not require human annotated and can be trained on large datasets, typically extracted from the Internet. [Balestriero et al., 2023].

Autoencoders [Hinton and Zemel, 1993] learn compact representations by reconstructing input data, while word2vec [Mikolov et al., 2013] captures semantic rela-

tionships between words in vector space. Contrastive learning [Oord et al., 2018, He et al., 2020, Chen et al., 2020a] maximizes the similarity between semantically similar data points, and masked autoencoding [Devlin et al., 2018, He et al., 2022] learns to reconstruct masked portions of input data. ELMo [Peters et al., 2018b], BERT [Devlin et al., 2018], and GPT [Radford et al., 2018, Brown et al., 2020] leverage language modeling and masked autoencoding to learn rich contextual language representations, generalizing the static representations obtained from word2vec. Researchers aim to develop a unified training objective for self-supervised learning, which remains an open challenge.

Transformer-based Large Language Models (LLMs) [Brown et al., 2020, Chowdhery et al., 2022] have demonstrated remarkable versatility and performance across various tasks due to their training on the simple general objective of next token prediction and increased scaling of the model and the training data. This has led to a pursuit of increasingly larger models, which exhibit more sophisticated abilities and result in significant performance improvements when they reach a certain size threshold [Wei et al., 2022]. Whether the size of data and model needs to grow is debated [Hoffmann et al., 2022] and remains an open problem.

One aspect that stands clear amidst the ongoing debate is the crucial role of self-supervised pre-training in enabling large models to adapt to a wide variety of tasks. This pre-training process facilitates the extraction of valuable patterns and representations from large amounts of unlabeled data, which in turn contributes to the models' performance and versatility. Consequently, self-supervised pre-training serves as the backbone for the subsequent transfer learning phase, where the acquired knowledge is fine-tuned and adapted to specific tasks.

Transfer Learning. Transfer learning has been explored to adapt pre-trained models to new tasks or domains. Techniques such as computer vision pre-training [Krizhevsky et al., 2012], meta-learning [Vinyals et al., 2016, Finn et al., 2017, Mishra et al., 2017], and multitask learning [Caruana, 1997] have been developed to leverage shared knowledge across tasks.

The question about the best way to adapt the pre-trained representations, beyond fine-tuning all parameters on new the new data, has been heavily researched, with ideas ranging from prompting methods [Radford et al., 2018], linear probing [He et al., 2020], and adapter modules [Houlsby et al., 2019]. Proper data curation and scale are essential for successful transfer learning. The research community is actively working to find the best approach for solving new tasks, such as determining whether to use one general model versus many specialized models, or prompting versus fine-tuning. This remains an open problem.

Representation Learning in Science. Representation learning has been successfully applied to various scientific domains, such as protein folding with AlphaFold [Jumper et al., 2021], medical text mining using Med-PaLM [Singhal et al., 2022], and quantum physics with Restricted Boltzmann Machines (RBMs) [Carleo and Troyer, 2017]. Other examples include training neural networks to guide mathematical and physical discoveries [Davies et al., 2021], deep reinforcement learning for novel matrix multiplication [Fawzi et al., 2022], using supervised learning for inverse design in optics [Peurifoy et al., 2018], and transfer learning for migrating physical knowledge [Qu et al., 2019].

Taking a Deeper Dive. We notice that almost all of the representation learning methods above deploy end-to-end supervised learning as a representation learning method. Many of these methods do not utilize specialized architectures, and in some instances, they even benefit from operating on handcrafted features, as exemplified in the prediction of topological insulators from chemical formulas [Ma et al., 2023], and prediction of stability of materials [Manti et al., 2023]. Self-supervised learning and transfer learning have not been properly introduced in the above scientific tasks, while high-dimensional data for pre-training is becoming more accessible at scale [Jain et al., 2013].

Concurrently, these challenging domains can provide a fertile ground for the development of representation learning methods. The significant acceleration in rep-

resentation learning progress can be attributed to the introduction of public benchmarks [Deng et al., 2009b, Russakovsky et al., 2015]. This prompts the inquiry of whether it is possible to establish scientific benchmarks that could similarly propel advancements in representation learning. Moreover, can we derive inspiration from concepts that are effective in science, such as symmetry, to devise innovative architectures for general machine learning applications? Indeed, some examples in Science have inspired the creation of novel neural networks, which in turn facilitate enhanced representation learning [Jing et al., 2017].

We continue with this inquiry in the subsequent section.

1.4 New Question

Can the development of representation learning within the context of Science contribute to the overall improvement of representation learning, beyond enhancing machine learning applications in scientific domains?

From studying the Prior Art it is clear that representation learning has not been extensively explored in the realm of Science. Most applications thus far have relied on supervised learning, which inherits the limitations *(i-iii)* discussed earlier. However, recent advances in novel architectures, self-supervised learning, and transfer learning may prompt a reconsideration of representation learning's limited application in scientific domains.

The development of novel inductive biases related to symmetry, language, and symbolic representations can help enhance representation learning for scientific disciplines. Just as progress in NLP has proven beneficial for computer vision (and vice versa), we anticipate that advances in representation learning for science could foster improvements across various modalities, including NLP and others prevalent on the internet. We propose the development of a bridge between representation learning and science, coining this research direction as "Representation Learning through the Lens of Science."

Our focus on symmetry, language, and symbolic biases stems from their demon-

strated success in both standard machine learning and representation learning within scientific contexts. By integrating these biases, we aim to create a more robust and comprehensive approach to representation learning that can drive progress in various scientific fields while addressing the aforementioned limitations.

1.5 Ideas

The development of representation learning within the context of Science presents several key ideas that could contribute to its overall improvement.

Firstly, symmetry as an inductive bias plays a crucial role. While CNNs have effectively harnessed symmetry to generalize, real-world symmetry is often more nuanced. Equivariant neural networks, as a generalization of CNNs, have been proposed, but their specialized hard coding of symmetry is not well-suited for existing hardware accelerators. Thus, there is a need to explore novel ways of incorporating symmetry into neural networks. We hypothesize that self-supervised learning can efficiently achieve this through data augmentation, encouraging neural networks to learn symmetry in a general function approximation that is the neural network. While this approach offers potential benefits in terms of speed and generalization, the dependence on training data is a limitation that warrants further investigation. Addressing problems *(i)* and *(iii)*, symmetry can help reduce the reliance on labeled data and enhance transferability.

Secondly, Occam's razor, a principle stating that simpler explanations are preferable to more complex ones when accounting for a given set of observations, serves as an essential inductive bias for promoting sparsity in representation learning. By favoring sparse and interpretable solutions, we can facilitate the discovery of new scientific laws and improve our understanding of complex phenomena. In this context, compositionality is also vital, as it enables the construction of more expressive models by combining simpler components, akin to the mixture of experts approach.

Lastly, language is a powerful bias for transferring ideas efficiently. The vast amount of scientific data available in natural language and programming offers signif-

icant potential for transfer learning. To leverage this, the development of novel neural networks capable of handling challenges specific to scientific datasets is necessary. For instance, networks should be able to maintain a long context in memory and utilize associative recall effectively.

By exploring these ideas, representation learning in Science can be advanced, improving machine learning applications within scientific domains and potentially beyond.

1.6 Results

In this thesis, we have taken preliminary steps in exploring "Representation Learning through the lens of Science." Our work is structured into three chapters, each focusing on one inductive bias to tackle one of the problems (need for data, lack of interpretability, and lack of transferability), as follows:

1. **Addressing the Need for Data with the Symmetry Inductive Bias (Chapter 2):** The primary focus in this chapter is on our work on "Equivariant Contrastive Learning" [Dangovski et al., 2021a], published at *International Conference on Learning Representations (ICLR) 2022*. The motivation of this work is to reconsider symmetry in self-supervised learning, which directly tackles Problems *(i,iii)*. In our work we improve representation learning for computer vision and nanophotonics. Furthermore, we present a novel testbed, based on nanophotonics data, that is helpful to the development of Equivariant Contrastive Learning. Then we will discuss the impact of Equivariant Contrastive Learning to our further work:

    - "Surrogate-and invariance-boosted contrastive learning for data-scarce applications in science", published in *Nature Communications* [Loh et al., 2022a]. In this work, we use contrastive learning to induce the inductive bias of symmetry in problems in nanophotonic and quantum systems, in order to learn more useful representations with fewer labeled data.

- "DiffCSE: Difference-based contrastive learning for sentence embeddings", published at *North American Chapter of the Association for Computational Linguistics - Human and Language Technologies (NAACL-HLT) 2022* [Chuang et al., 2022]. In this work, we apply Equivariant Contrastive Learning to the learning of representations of sentences.
- "Multi-Symmetry Ensembles: Improving Diversity and Generalization via Opposing Symmetries", published at *International Conference on Machine Learning (ICML)* 2023 [Loh et al., 2023]. In this work, we ensemble multiple models trained with Equivariant Contrastive Learning to obtain more accurate and calibrated classifiers.
- "Meta-Learning and Self-Supervised Pretraining for Real World Image Translation", to appear at *IEEE High Performance Extreme Computing Conference (IEEE HPEC)* 2023 [Rugina et al., 2021]. In this work, we apply our work on contrastive learning to a realistic weather system.

We also briefly discuss our works as listed: [Liao et al., 2022] to appear at *TMLR*, [Luo et al., 2022] submitted to *NeuRIPS 2023*, [Dangovski et al., 2019a] at *TACL 2023*, [Jing et al., 2018] to appear at *IEEE HPEC 2023*, [Loh et al., 2022b] to appear at *TMLR*, [Dugan et al., 2023] at *ICML 2023*, and [Han et al., 2023] at *Generative Models for Computer Vision Workshop 2023*.

2. **Addressing the Ability to Transfer with the Language Inductive Bias (Chapter 3):** In this chapter we will discuss the role of language for transfer learning through our work "We Can Explain Your Research in Layman's Terms: Towards Automating Science Journalism at Scale", published at *AAAI Conference on Artificial Intelligence 2021* [Dangovski et al., 2021b]. The work focused on transfer learning for the summarization of scientific text.

   We also briefly discuss our works as listed: [Ramírez et al., 2020] submitted to *IJCNLP-AACL 2023*, [Khoury et al., 2020] at *EMNLP 2020*, [Rugina et al., 2020] submitted to *IJCNLP-AACL 2023*, and [Vogelbaum et al., 2020] to appear at *IEEE HPEC 2023*.

3. **Addressing the Lack of Interpretability with the Symbolic Inductive Bias (Chapter 4):** The main focus in this chapter is on our work on "Fast neural models for symbolic regression at scale", prepared for submission to *Nature Machine Intelligence* [Costa et al., 2020], which proposes a novel neurosymbolic inductive bias that is faster and more scalable on modern hardware. Then we will discuss the impact of OccamNet to our work "AI-Assisted Discovery of Quantitative and Formal Models in Social Science", under review at *Nature Human Behavior* [Balla et al., 2022]. In this work, we apply the neurosymbolic inductive bias from above to search for quantitative and formal models in social science. We will conclude with our work on "Phase Transitions and Representation Geometry in Contrastive Learning", submitted to *NeurIPS 2023*, where discover phase transitions in the representation dynamics of contrastive learning. We will also birefly mention [Lu et al., 2022], to appear at *Nature Communications* and [Hernandez et al., 2023] at *NeurIPS SVRHM Workshop 2022*.

In the following chapters, we will discuss these biases and their implications for representation learning in science, with each chapter dedicated to a particular bias. There are many limitations to our results and we will discuss addressing them through future work in Chapter 5.

Chapter 2

Addressing the Need for Data with the Symmetry Inductive Bias

2.1 Introduction

2.1.1 Symmetry

Symmetry, the phenomenon in which the properties of a system are preserved after that system is transformed, is an indispensable tool for helping us understand Nature. For example, symmetry benefits the study of complex physical systems, which are modeled in a high-dimensional space and/ or with complex dynamics. In such type of physics knowing the exact solution describing the state of the system is unattainable due to high-dimensionality space of possible solutions and the complex dynamics. However, we could infer a large portion of the properties of the solution by looking at the global or local symmetries of the system: e.g. materials that have certain symmetries induce corresponding symmetries to the light that interacts with the materials [Joannopoulos et al., 2008]. Symmetry effectively limits the space of solution we consider when studying a system. Likewise, symmetry could also be *broken*, which further constrains the space of possible solutions by forcing all considered properties of the system to have a particular stance to a symmetric operation. One famous example of breaking the symmetry is the study of spin glasses that was instrumental

in the Nobel Prize in Physics to Giorgio Parizi [Parisi, 1979].

Symmetry is crucial not only in deciphering the solutions of complex physical phenomena, but also in building physical theories from ground up. Einstein famously built his theory of General Relativity by starting from first-principle symmetry arguments and following these principles in constructing the equations in the theory [Einstein, 1922].

So far we have seen two useful sides of the the study of symmetry: one is using symmetry to help us decipher complex systems (materials), and the other is using symmetry from ground-up to help us develop understadning that can explain a general class of phenomena (motion in space). To support such useful applications of symmetry, certain mathematical abstractions have been developed. Namely, two of these abstractions stand out:

1. Group theory [Borel and Tits, 1965] — the study of symmetry by abstracting transformations in mathematical objects and using these grouping these transformations in a set that satisfies symmetries between the operations. Group theory is an indispencible mathematical tool that has revlolutionized all the natural sciences, ranging from physics, chemistry, biology, etc.

2. Representation theory [Serre et al., 1977] — a single Group, an abstract mathematical object, can appear in Nature in many forms of manifestation. In order to understand the possible ways a group can interact with Nature, we study all the possible ways a group transformation can be presented as a linear operator in vector spaces, hence we aim to find "representations" of the Group. The symmetries of the Group constrain the possible representations. Similartly to Group theory, Representation theory has been instrumental in understadingn how the abstract notion of symmetry can be manifested in Nature.

2.1.2 Need for Data

Mirroring our discussion of physical phenomena, neural networks with a large number of parameters *can be considered* as complex systems with high-dimensional solutions

space and complicated dynamics.

Given the extensive nature of the machine learning solution space and the complex dynamics involved, a significant volume of data is required for traditional supervised learning to discover meaningful solutions. This necessity can often lead to several challenges, such as data scarcity or the immense computational resources needed to process large datasets. One potential solution to mitigate this issue is to leverage the concept of symmetry, which has been instrumental in understanding complex physical systems, to influence the inductive bias in neural networks.

Symmetry can effectively reduce the complexity of a problem by limiting the possible solutions that need to be considered. Similar to how symmetry aids in understanding physical phenomena, symmetry can guide the structure and function of neural networks, thereby constraining their solution space and making them more efficient and robust.

Supervised learning, despite its strengths, comes with its limitations. For instance, it requires labeled data, which can be expensive and time-consuming to produce, and it is prone to overfitting on the available training data, potentially yielding models that fail to generalize well to new data. One alternative approach is self-supervised learning (SSL), which can be broken down into two stages: representation learning and fine-tuning.

During the first stage, an objective is established that encourages symmetry as the inductive bias in the neural network. Here, we do not concern ourselves with the final task the network will perform; rather, we seek to expose the network to vast amounts of data so that it can learn robust, generalizable representations. This process, in essence, is akin to a physicist identifying the symmetries of a system to form an initial solution framework or ansatz.

The second stage involves using the learned representations from the first stage and fine-tuning them for a specific task using a limited amount of labeled data. This echoes the physics process where, after identifying the symmetries and formulating an ansatz, the physicist proceeds to solve the equations within this framework, refining and specifying the solution to the particular context at hand.

Thus, we can view self-supervised learning as a two-step process parallel to the practice of physics. This parallelism is outlined in the table below:

Stage	**Self-Supervised Learning**	**Physics Process**
First Stage	Learning representations	Identifying symmetries
Second Stage	Task-specific fine-tuning	Solution refinement

Table 2.1: Parallel between self-supervised learning stages and physics process.

As depicted in Table 2.1, there is a clear parallel between the two-stage process in self-supervised learning and that of problem-solving in physics.

In the first stage of both processes, the goal is to establish a general understanding or a framework. In self-supervised learning, we aim to learn general representations of the data. Similarly, in physics, we seek to identify the fundamental symmetries of a system, which helps us formulate an ansatz, a starting point that constrains the possible solutions.

The second stage revolves around the utilization and fine-tuning of the previously established framework. For self-supervised learning, the representations learned in the first stage are fine-tuned on a specific task with a smaller amount of labeled data. In physics, we use the ansatz to solve the equations and refine the solution in the specific context of interest.

This parallelism is more than just an interesting comparison; it provides valuable insights into how we might approach problem-solving in machine learning. By taking cues from the methodical, principles-based approach in physics, we could improve the efficiency and efficacy of machine learning models, especially in scenarios where data may be scarce or costly to acquire. Utilizing symmetry as an inductive bias in neural networks could be a powerful tool for constraining the solution space and improving generalization, much like it is in the study of physical systems.

In summary, leveraging symmetry as an inductive bias in machine learning models, particularly within a self-supervised learning paradigm, offers an efficient and effective approach to manage the high-dimensionality and complex dynamics of these systems. Drawing parallels from physics, we can potentially guide the formulation and

execution of such models, reducing data dependency, and increasing their potential for generalization.

We continue our discussion in the following section with our most extensive fundamental contribution to SSL through the lens of symmetry, which aims at mitigating the need for data in representation learning.

2.2 Equivariant Contrastive Learning

2.2.1 Introduction

Human knowledge about what makes a good representation and the abundance of unlabeled data has enabled the learning of useful representations via self-supervised learning (SSL) pretext tasks. State-of-the-art SSL methods encourage the representations not to contain information about the way the inputs are transformed, i.e. to be invariant to a set of manually chosen transformations. One such method is contrastive learning, which sets up a binary classification problem to learn invariant features. Given a set of data points (say images), different transformations of the same data point constitute positive examples, whereas transformations of other data points constitute the negatives [He et al., 2020, Chen et al., 2020a]. Beyond contrastive learning, many SSL methods also rely on learning representations by encouraging invariance [Grill et al., 2020b, Chen and He, 2021b, Caron et al., 2021, Zbontar et al., 2021]. Here, we refer to such methods as Invariant-SSL (I-SSL).

The natural question in I-SSL is to what transformations should the representations be insensitive [Chen et al., 2020a, Tian et al., 2020b, Xiao et al., 2020]. Chen et al. [2020a] highlighted the importance of transformations and empirically evaluated which transformations are useful for contrastive learning (e.g., see Figure 5 in their paper). Some transformations, such as *four-fold rotations*, despite preserving semantic information, were shown to be harmful for contrastive learning. This does not mean that four-fold rotations are not useful for I-SSL at all. In fact, predicting four-fold rotations is a good proxy task for evaluating the representations produced

with contrastive learning [Reed et al., 2021]. Furthermore, instead of being insensitive to rotations (invariance), training a neural network to predict them, i.e. to be *sensitive* to four-fold rotations, results in good image representations [Gidaris et al., 2018, 2019]. These results indicate that the choice of making features *sensitive* or *insensitive* to a particular group of transformations can have a substantial effect on the performance of downstream tasks. However, the prior work in SSL has exclusively focused on being either entirely insensitive [Grill et al., 2020b, Chen and He, 2021b, Caron et al., 2021, Zbontar et al., 2021] or sensitive [Agrawal et al., 2015, Doersch et al., 2015, Zhang et al., 2016, Noroozi and Favaro, 2016, Gidaris et al., 2018] to a set of transformations. In particular, the I-SSL literature has proposed to simply remove transformations that hurt performance when applied as invariance.

To understand how sensitivity/ insensitivity to a particular transformation affects the resulting features, we ran a series of experiments summarized in Figure 2-1. We trained and tested a simple I-SSL baseline, SimCLR [Chen et al., 2020a], on CIFAR-10 using only the *random resized cropping transformation* (solid yellow line). The test accuracy is calculated as the retrieval accuracy of a k-nearest neighbors (kNN) classifier with a memory bank consisting of the representations on the training set obtained after pre-training for 800 epochs. Next, in addition to being invariant to resized cropping, we additionally encouraged the model to be either sensitive (shown in pink) or insensitive (shown in blue) to a second transformation. We encourage insensitivity by adding the transformation to the SimCLR data augmentation, and sensitivity by predicting it (see Section 4.2.4). We varied the choice of this second transformation. We found that for some transformations, such as *horizontal flips* and *grayscale*, insenstivity results in better features, but is detrimental for transformations, such as *four-fold rotations*, *vertical flips*, *2x2 jigsaws* (4! = 24 classes), *four-fold Gaussian blurs* (4 levels of blurring) and *color inversions*. When we encourage sensitivity to these transformations, the trend is reversed. In summary, we observe that if invariance to a particular transformation hurts feature learning, then imposing sensitivity to the same transformation may improve performance. This leads us to conjecture that instead of choosing the features to be only invariant or only sensitive as done

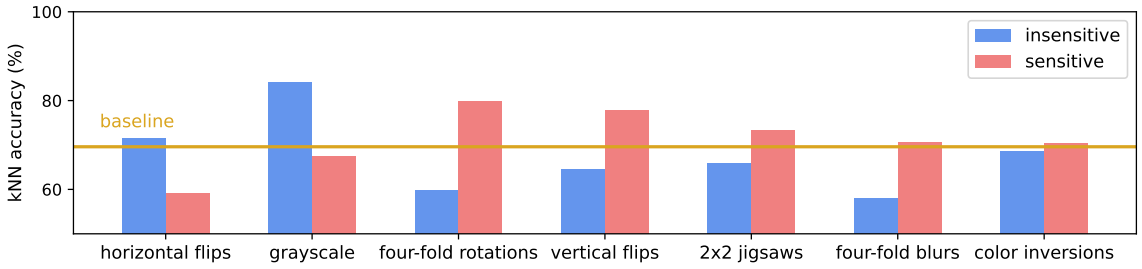

Figure 2-1: SSL representations should be encouraged to be either insensitive or sensitive to transformations. The baseline is SimCLR with random resized cropping only. Each transformation on the horizontal axis is combined with random resized cropping. The dataset is CIFAR-10 and the kNN accuracy is on the test set. More experimental details can be found in Section 4.2.4.

in prior work, it may be possible to learn better features by imposing invariance to certain transformations (e.g., cropping) and sensitivity to other transformations (e.g., four-fold transformations).

The concepts of sensitivity and insensitivity are both captured by the mathematical idea of equivariance [Agrawal et al., 2015, Jayaraman and Grauman, 2015, Cohen and Welling, 2016]. Let G be a group of transformations. For any $g \in G$ let $T_g(\boldsymbol{x})$ denote the function with which g transforms an input image $\boldsymbol{x}$. For instance, if G is the group of four-fold rotations then $T_g(\boldsymbol{x})$ rotates the image $\boldsymbol{x}$ by a multiple of $\pi/2$. Let f be the encoder network that computes feature representation, $f(\boldsymbol{x})$. I-SSL encourages the property of "invariance to G," which states $f(T_g(\boldsymbol{x})) = f(\boldsymbol{x})$, i.e. the output representation, $f(\boldsymbol{x})$, does not vary with T_g. Equivariance, a generalization of invariance, is defined as, $\forall \boldsymbol{x} : f(T_g(\boldsymbol{x})) = T'_g(f(\boldsymbol{x}))$, where T'_g is a fixed transformation (i.e., without any parameters). Intuitively, equivariance encourages the feature representation to change in a well defined manner to the transformation applied to the input. Thus, invariance is a trivial instance of equivariance, where T'_g is the identity function, i.e. $T'_g(f(\boldsymbol{x})) = f(\boldsymbol{x})$. While there are many possible choices for T'_g [Cohen and Welling, 2016, Bronstein et al., 2021], I-SSL uses only the trivial choice that encourages f to be insensitive to G. In contrast, if T'_g is not the identity, then f will be sensitive to G and we say that the "equivariance to G" will be non-trivial.

Therefore, in order to encourage potentially more useful equivariance properties,

we generalize SSL to an Equivariant Self-Supervised Learning (E-SSL) framework. In our experiments on standard computer vision data, such as the small-scale CIFAR-10 [Torralba et al., 2008, Krizhevsky, 2009] and the large-scale ImageNet [Deng et al., 2009b], we show that extending I-SSL to E-SSL by also predicting four-fold rotations improves the semantic quality of the representations. We show that this approach works for other transformations too, such as vertical flips, 2x2 jigsaws, four-fold Gaussian blurs and color inversions, but focus on four-fold rotations as the most promising improvement we obtain with initial E-SSL experiments in Figure 2-1.

We also note that the applications of E-SSL in this paper are task specific, meaning that the representations from E-SSL may work best for a particular downstream task that benefits from equivariance dictated by the available data. E-SSL can be further extended to applications in science; in particular, we focus on predictive tasks using (unlabelled and labelled) data collected via experiments or simulations. The downstream tasks in prediction problems in science are often fixed and can be aided by incorporating scientific insights. Here, we also explore the generality of E-SSL beyond computer vision, on a different application: regression problems in photonics science and demonstrate examples where E-SSL is effective over I-SSL.

Our contributions can be summarized as follows:

- We introduce E-SSL, a generalization of popular SSL methods that highlights the complementary nature of invariance and equivariance. To our knowledge, we are the first to create a method that benefits from such complementarity.

- We improve state-of-the-art SSL methods on CIFAR-10 and ImageNet by encouraging equivariance to four-fold rotations. We also show that E-SSL is more general and works for many other transformations, previously unexplored in related works.

- We demonstrate the usefulness of E-SSL beyond computer vision with experiments on regression problems in photonics science. We also show that our method works both for finite and infinite groups.

The rest of this section is organized as follows. In Subsection M we elaborate on related work. In Subsection 2.2.3 we introduce our experimental method for E-SSL. In Subsection 4.2.4 we present our main experiments in computer vision. In Subsection 4.2.5 provide a discussion around our work that extends our study beond computer vision. Beginning from Appendix 2.2.6, we provide more details behind our findings and discuss several potential avenues of future work.

2.2.2 Related Work

To encourage non-trivial equivariance, we observe that a simple task that predicts the synthetic transformation applied to the input, works well and improves I-SSL already; some prediction tasks create representations that can be transferred to other tasks of interest, such as classification, object detection and segmentation. While prediction tasks alone have been realized successfully before in SSL [Agrawal et al., 2015, Doersch et al., 2015, Zhang et al., 2016, Misra et al., 2016, Noroozi and Favaro, 2016, Zamir et al., 2016, Lee et al., 2017, Mundhenk et al., 2018, Gidaris et al., 2018, Zhang et al., 2019, Zhang, 2020], to our knowledge we are the first to combine simple predictive objectives of synthetic transformations with I-SSL, and successfully improve the semantic quality of representations. We found that the notion of equivariance captures the generality of our method.

To improve representations with pretext tasks, Gidaris et al. [2018] use four-fold rotations prediction as a pretext task for learning useful visual representations via a new model named RotNet. Feng et al. [2019] learn decoupled representations: one part trained with four-fold rotations prediction and another with non-parametric instance discrimination [Wu et al., 2018] and invariance to four-fold rotations. Yamaguchi et al. [2021] use a joint training objective between four-fold rotations prediction and image enhancement prediction. Xiao et al. [2020] propose to learn representations as follows: for each atomic augmentation from the contrastive learning's augmentation policy, they leave it out and project to a new space on which I-SSL encourages invariance to all augmentations, but the left-out one. The resulting representation could either be a concatenation of all projected left-out views' representations, or the

representation in the shared space, before the individual projections. Our method differs from the above contributions in that E-SSL is the only hybrid framework that encourages both insensitive representations for some transformations and sensitive representations for others and does not require representations to be sensitive and insensitive to a particular transformation at the same time. Thus, what distinguishes our work is the complementary nature of invariance and equivariance for multiple transformations, including finite and infinite groups.

To obtain performance gains from transformations, Tian et al. [2020b] study which transformations are the best for contrastive learning through the lens of mutual information. Reed et al. [2021] use four-fold rotations prediction as an evaluation measure to tune optimal augmentations for contrastive learning. Wang and Qi [2021] use strong augmentations to improve contrastive learning by matching the distributions of strongly and weakly augmented views' representation similarities to a memory bank. Wang et al. [2021] provide an effective way to bridge transformation-insensitive and transformation-sensitive approaches in self-superived learning methods via residual relaxation. A growing body of work encourages invariance to domain agnostic transformations [Tamkin et al., 2021, Lee et al., 2021, Verma et al., 2021] or strengthens invariance with regularization [Foster et al., 2021]. Our framework is different from the above works, because we work with transformations that encourage equivariance beyond invariance.

To understand and improve equivariant properties of neural networks, Lenc and Vedaldi [2015] study emerging equivariant properties of neural networks and [Cohen and Welling, 2016, Bronstein et al., 2021] construct equivariant neural networks. In contrast, our work does not enforce strict equivariance, but only encourages equivariant properties for the encoder network through the choice of the loss function. While strict equivariance is concerned with groups, some of the transformations, such as random resized cropping and Gaussian blurs, may not even be groups, but they could still be analyzed in the E-SSL framework. Thus, ours is a flexible framework, which allows us to consider a variety of transformations and how the encoder might exhibit equivariant properties to them.

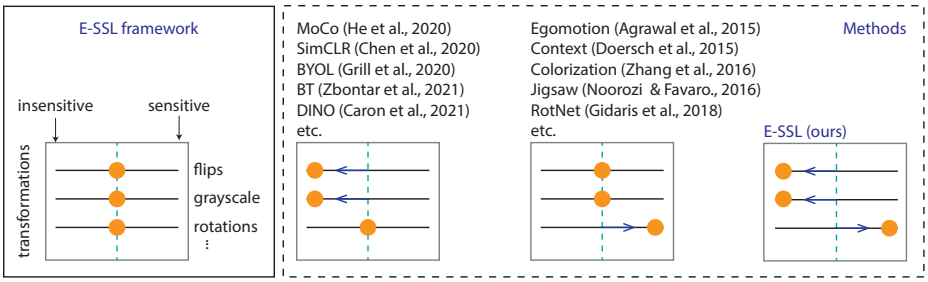

Figure 2-2: E-SSL framework. Left: framework. Right: methods. Egomotion, Context, Colorization and Jigsaw use other transformations than rotations, but their patterns looks like that of RotNet's. Likewise, for E-SSL can use transformations different from rotation.

2.2.3 Method

Our method is designed to test our primary conjecture that a *hybrid approach* of sensitive and insensitive representations learns better features. Surprisingly, this hybrid approach is not yet present in SSL, as Figure 2-2 illustrates. In this figure, we can view transformations in SSL as "levers." Each downstream task has an optimal configuration of the levers, which should be tuned in the SSL objective: left for insensitive and right for sensitive representations. E.g., make representations insensitive to horizontal flips and grayscale and sensitive to four-fold rotations, vertical flips, 2x2 jigsaws, Gaussian blurs or color inversions. Formally, insensitive and sensitive features correspond to trivial and regular group representations, respectively. Here, we present an effective method to achieve this control.

Let $f(\cdot;\boldsymbol{\theta}_f)$ with trainable parameters $\boldsymbol{\theta}_f$ be a backbone encoder. Analogously, let $p_1(\cdot;\boldsymbol{\theta}_{p_1})$ be a projector network for the I-SSL loss. There might be an extra prediction head and parameters, depending on the objective, which we omit for simiplicity. Let $p_2(\cdot;\boldsymbol{\theta}_{p_2})$ be the predictor network for encouraging sensitivity, which we will call "predictor for equivariance." We share the backbone encoder f jointly for I-SSL and the objective of predicting the transformations from the backbone representations. Let $\ell_{\text{I-SSL}}$ be the I-SSL loss and $\ell_{\text{E-SSL}}$ be the added E-SSL loss that encourages sensitivity to a particular transformation. Let the parameter λ be the strength of the E-SSL loss. The optimization objective for an image $\boldsymbol{x}$ with views $\{\boldsymbol{x}'\}$ in the batch

is given as follows

$$\arg\min_{\boldsymbol{\theta}_f,\boldsymbol{\theta}_{p_1},\boldsymbol{\theta}_{p_2}} \ell_{\text{SSL}}(p_1(f(\{\boldsymbol{x}'\};\boldsymbol{\theta}_f);\boldsymbol{\theta}_{p_1})) + \lambda \mathbb{E}_{g\in G}\left[\text{PredictionLoss}(g, p_2(f(T_g(\boldsymbol{x}');\boldsymbol{\theta}_f);\boldsymbol{\theta}_{p_2})\right] \tag{2.1}$$

where $\ell_{\text{E-SSL}}$ (the expectation in the second summand) can take either one or all of the views, but we take only one for simplicity. The goal of $\ell_{\text{E-SSL}}$ is to predict g from the representation $p_2(f(T_g(\boldsymbol{x}');\boldsymbol{\theta}_f);\boldsymbol{\theta}_{p_2})$, which encourages equivariance to the group of transformations G. The PredictionLoss could be either a cross entropy loss for finite groups or L1/ MSE loss for infinite groups. In practice we replace the expectation with an unbiased estimate. Most of our experiments in this paper focus on finite groups, but we show one example for an infinite group in Appendix 2.2.11.

E-SSL can be constructed for any semantically meaningful transformation (see for example, Figure 2-1). From Figure 2-1 we choose four-fold rotations as the most promising transformation and we fix it for the upcoming section. As a minor motivation, we also present empirical results about the similarities between four-fold rotations prediction and I-SSL in Appendix 2.2.8. In particular, both tasks benefit from the same data augmentation. Figure 2-3 sketches how our construction works for predicting four-fold rotations. In particular, we sample each of the 4 possible rotations uniformly and use the cross entropy loss for the PredictionLoss in Equation 2.1.

What transformations could work for E-SSL? A common property of the successful transformations we have studied up to this point is that they form groups in the mathematical sense, i.e. (*i*) each transformation is invertible, (*ii*) composition of two transformations is part of the set of transformations and (*iii*) compositions are associative.

In this paper, we encourage equivariance to a group of transformation by predicting them. This does not guarantee that the encoder we learn will be strictly equivariant to the group. In practice we observe that invariance and equivariance is well encouraged by the training objectives we use (see Appendix 2.2.9 for detailed analysis). In fact, even strict equivariance is possible, i.e. there exists an encoder

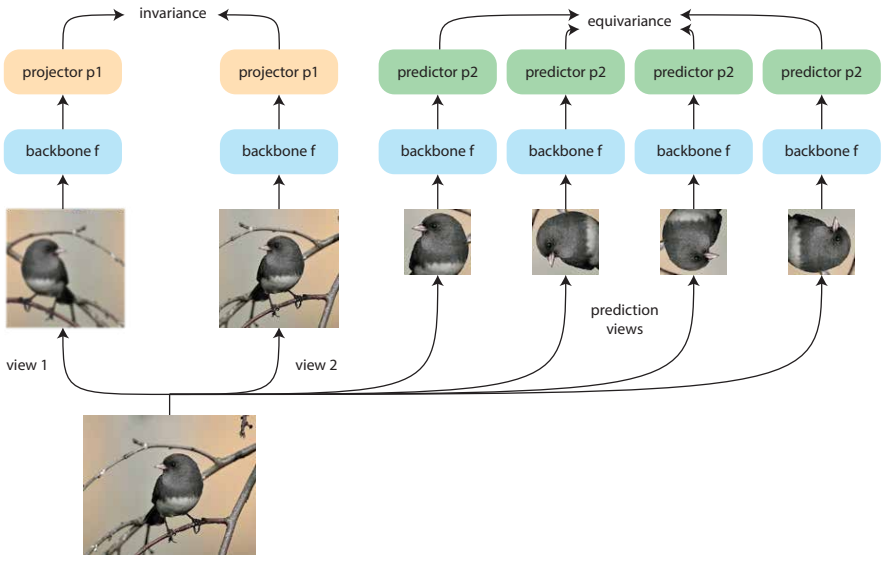

Figure 2-3: Sketch of E-SSL with four-fold rotations prediction, resulting in a backbone that is sensitive to rotations and insensitive to flips and blurring. ImageNet example n01534433:169.

that is non-trivially equivariant, under a reasonable assumption which is formulated as follows. Let X be the set of all images. Let G be a group whose elements $g \in G$ transform X via the function $T_g \colon X \to X$. Let $X' = \{T_g(\boldsymbol{x}) \mid g \in G, \boldsymbol{x} \in X\}$ be the set of all transformed images. Let $f(\cdot;\boldsymbol{\theta}) \colon X' \to S$ be an encoder network that we learn with parameters $\boldsymbol{\theta}$. We write $f(\cdot) \equiv f(\cdot;\boldsymbol{\theta})$ for simplicity. Finally, let $S = \{f(\boldsymbol{x}') \mid \boldsymbol{x}' \in X\}$ be the set of all representations of the images in X'. The following is our statement.

Proposition 1 (Non-trivial Equivariance)**.** *Given $T_g \colon X' \to X'$ for the group G, there exists an encoder $f \colon X' \to S$ that is non-trivially equivariant to the group G under the assumption that if $f(T_g(\boldsymbol{x})) = f(T_{g'}(\boldsymbol{x}'))$ then $g = g'$ and $\boldsymbol{x} = \boldsymbol{x}'$ for all $g, g' \in G$ and $\boldsymbol{x}, \boldsymbol{x}' \in X$.*

We defer the proof to Appendix 2.2.7. The significance of this proof is that it explicitly constructs a non-trivially equivariant encoder network for groups G if the assumption is satisfied. The intuition of the assumption is that if the representations of two transformed inputs are the same, the inputs should coincide, and likewise the transformations. More formally, this assumption reflects the condition when the

dataset contains only one element of each group orbit. We speculate that satisfying this assumption is reasonable for the datasets in this work, since we observe a natural setting of the data, e.g. horizontal mirror symmetry in Gpm, and we consider transformations that disturb this natural setting. In Appendix 2.3 we also show that E-SSL is crucial for the Flowers-102 dataset for which this assumption might be less clear. In Appendix 2.3.1 we also present a natural modification of E-SSL for scenarios, where that assumption is violated.

Could other transformations still help? To motivate our work, in Figure 2-1 we observed additional transformations that could be useful, such as vertical flips, 2x2 jigsaws, four-fold Gaussian blurs and color inversions. All of these transformations are groups, except for four-fold Gaussian blurs. Each element of Gaussian blurs is invertible (de-blurring), but the inverse is not a transformation in the set. Interestingly, we observe that four-fold Gaussian blurs still improve the baseline, which means the success of E-SSL may not be limited to groups.

We might also consider combining the prediction of multiple transformations to encourage sensitivity to all of them. However, the gains we saw in Figure 2-1 may not add up when we combine transformations, because they may not be independent. The gains may also depend on the transformations that we choose for I-SSL. While we see combinations of transformations as promising future work, we focus on a single transformation to make a clear presentation of E-SSL.

2.2.4 Experiments

Setups

CIFAR-10 setup. We use the CIFAR-10 experimental setup from [Chen and He, 2021b]. We consider two simple I-SSL methods: SimCLR (with InfoNCE loss [Oord et al., 2018] and temperature 0.5) and SimSiam [Chen and He, 2021b]. We were able to obtain baseline results close to those in [Chen and He, 2021b]. The predictor for equivariance takes a smaller crop with size 16x16. We report performance on the standard linear probe. We tune λ to 0.4 both for SimCLR and SimSiam (full tuning

in Table 2.7 in Appendix 2.2.9). Remaining experimental details can be found in Appendix 2.2.9.

ImageNet setup. We use the original augmentation setting for each method. The predictor for equivariance takes a smaller crop with size 96x96. We use a ResNet-50 [He et al., 2016a] backbone for each method. In terms of optimizer and batch size settings, we follow the standard training recipe for each method. For our SimCLR experiments we use a slightly more optimal implementation that uses BYOL's augmentations (i.e. it includes *solarization*), initializes the ResNet with zero BatchNorm weights and uses the InfoNCE loss with temperature 0.2.

Photonic-crystals setup. Photonic crystals (PhC) are periodically-structured materials engineered for wide ranging applications by manipulating light waves[Yablonovitch, 1987, Joannopoulos et al., 2008]. The density-of-states (DOS) is often used as a design metric to engineer the desired properties of these crystals and thus here, we consider the regression task of predicting the DOS of PhCs. Examples of this dataset are depicted in Section 4.2.5 and further details can be found in Appendix 2.2.11. The use of symmetry or invariance knowledge is common in scientific problems; here, the DOS labels are invariant to several physical transformations of the unit cell, namely, rolling translations (due to its periodicity), operations arising from the symmetry group (C_{4v}) of the square lattice, i.e. rotations and mirror flips, and refractive scaling. We construct an encoder network comprising of simple convolutional and fully-connected layers (see Appendix 2.2.11) and create various synthetic datasets to investigate encouragement of equivariance. After SSL/ E-SSL, we fine-tune the network with L1 loss; for better interpretability of prediction accuracies, we use a relative error metric [Liu et al., 2018a, Loh et al., 2021] for evaluation, given by $\ell_{\text{DOS}} = (\sum_\omega \left|\text{DOS}^{\text{pred}} - \text{DOS}\right|)/(\sum_\omega \text{DOS})$, reported in (%). We defer the results to Section 4.2.5, because of the novelty of the experimental setup.

The predictor p_2 for E-SSL. The predictor is a 2 layer MLP for CIFAR-10 and Photonic-crystals, and a 3 layer MLP for ImageNet, followed by a linear head that pro-

Algorithm 1 PyTorch-style pseudocode for E-SSL, predicting four-fold rotations.

```
# f: backbone encoder network
# p1: projector network for I-SSL
# p2: predictor network for E-SSL
# ssl_loss: loss function for I-SSL
# lambda: weight of the E-SSL


for x in loader:
    # large views for SSL and small view for EE
    V_large = augment(x, small_crop=False) # list of views
    v_small = augment(x, small_crop=True) # change: crop with size=96 and scale=(0.05, 0.14)

    # loss
    loss_invariance = ssl_loss(p1(f(V_large))
    labels = [0] * N + [1] * N + [2] * N + [3] * N # 4Nx1
    v_cat = cat([v_small] * 4, dim=0) # 4Nx3x96x96
    v_equivariance = rot90(v_cat, labels) # constructing the rotated views

    logits = p2(f(v_equivariance)) # 4Nx4
    loss_equivariance = CrossEntropyLoss(logits, labels) # rotation prediction
    loss = loss_invariance + lambda * loss_equivariance

    # optimization step
    loss.backward()
    optimizer.step()
```

duces the logits for the an n-way classification (for example four-fold rotations is 4-way classification), or a single node for the continuous group experiment. The predictor's hidden dimension is shared across all layers and it equals 2048 for CIFAR-10 and ImageNet and 512 for PhC. After each linear layer, there is a Layer Normalization [Ba et al., 2016] followed by ReLU. We experimented with Batch Normalization [Ioffe and Szegedy, 2015] (with trainable affine parameters) instead of Layer Normalization, but did not observe any significant changes. For some experiments, we discovered that removing the last ReLU from the MLP improves the results slightly. In particular, for SimSiam on CIFAR-10 and for all models on ImageNet we omit the last ReLU.

Finally, Algorithm 1 presents pseudocode for E-SSL with four-fold rotations on ImageNet. In our implementation, we use smaller resolution for the rotated images, so that we can fit all views on the same batch and have minimal overhead for pre-training (additional details in Table 2.10 in Appendix 2.2.10).

Main results

CIFAR-10 results. To highlight the benefits of our method, Table 2.2 demonstrates the improvement we obtain by using E-SSL on top of SimCLR and SimSiam and then shows different ablations and alternative methods. We label the E-SSL extensions as E-SimCLR and E-SimSiam respectively. We observe that we can increase

Table 2.2: Linear probe accuracy (%) on CIFAR-10. Models are pre-trained for 800 epochs. Baseline results are from Appendix D in [Chen and He, 2021b]. Standard deviations are from 5 different random initializations for the linear head. Deviations are small because the linear probe is robust to the seed.

Method	SimCLR [Chen et al., 2020a]	SimSiam [Chen and He, 2021b]
Baseline [Chen and He, 2021b]	91.1	91.8
Baseline (our reproduction)	92.0± 0.0	91.6± 0.0
E-SSL (ours)	**94.1**± 0.0	**94.2**± 0.1
Ablating E-SSL		
Single random rotation	93.4± 0.0 (↓ 0.7)	92.6± 0.0 (↓ 1.6)
Linear predictor for equivariance	93.3± 0.0 (↓ 0.8)	93.4± 0.0 (↓ 0.8)
No SSL augmentation in equivariance views	92.7± 0.1 (↓ 1.4)	92.0± 0.1 (↓ 2.2)
Alternatives to E-SSL		
Disentangled representations	91.3± 0.0 (↓ 2.7)	91.1± 0.0 (↓ 3.1)
Insensitive instead of sensitive	86.3± 0.1 (↓ 7.8)	86.1± 0.1 (↓ 8.1)

a tuned baseline accuracy by about $2-3\%$. When ablating E-SSL, we see that each component of E-SSL is important. Most useful is the SSL augmentation applied on top of the rotated views. We also study alternatives to E-SSL. With "Disentangled representations" we investigate whether a "middle ground" is optimal for E-SSL: half of the representation to be insensitive to a transformation and the other half to be sensitive to the same transformation. This results in degradation of performance, which reflects our hypothesis that the representations should be either insensitive or sensitive. We conducted this experiment by using four-fold rotations in I-SSL for half of the representation and E-SSL for the other half. Finally, making the representations "Insensitive instead of sensitive" to four-fold-rotations hurts the performance significantly, as it is also observed in Figure 2-1, and in [Chen et al., 2020a, Xiao et al., 2020].

Figure 2-4 reveals that E-SSL is more robust to removing transformations for I-SSL or reducing the labels for training. For example, E-SimCLR and E-SimSiam with only random resized cropping obtain 83.5% and 84.6% accuracies. Encouraging sensitivity to one transformation, namely four-fold-rotations, can reduce the need for selecting many transformations for I-SSL and with only 1% of the training data,

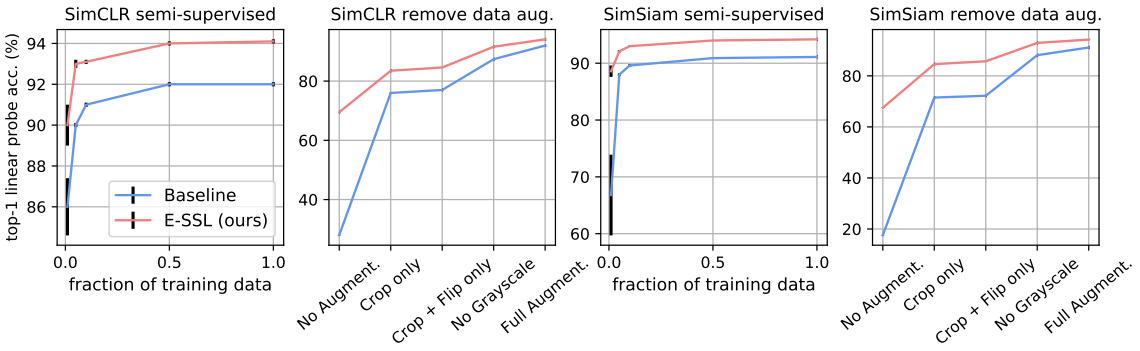

Figure 2-4: Reducing the labels for training and the data augmentation for pre-training on CIFAR-10. Error bars for 5 different training data splits.

Table 2.3: Linear probe accuracy (%) on ImageNet. Each model is pre-trained for 100 epochs. Baseline results are from Table B.1 in [Chen et al., 2020a] from Table 4 in [Chen and He, 2021b]. Numbers marked with * use a less optimal setting than our reproduction for SimCLR (see ImageNet setup).

Method	SimCLR [Chen et al., 2020a]	SimSiam [Chen and He, 2021b]	Barlow Twins [Zbontar et al., 2021]
Baseline [Chen et al., 2020a]	64.7*	-	-
Baseline [Chen and He, 2021b]	66.5*	68.1	-
Baseline (our reproduction)	67.3	68.1	66.9
E-SSL (ours)	**68.3**	**68.6**	**68.2**

E-SimCLR and E-SimSiam achieve 90.0± 1.0% and 88.6± 1.0% respectively.

ImageNet results. Table 2.3 demonstrates our main results on the linear probe on ImageNet after pre-training with various state-of-the-art I-SSL methods and their E-SSL versions. By only sweeping λ and slightly reducing the original learning rate for SimSiam we obtain consistent 1%/ 0.5%/ 1.3% improvements for SimCLR/ SimSiam/ Barlow Twins respectively. Additionally, in Table 2.4 we observe consistent benefits of using E-SSL with longer pre-training. Finally, after 800 epochs of pre-training E-SimCLR achieves **72.5%**, which is 0.6% better than SimCLR's 71.9% baseline.

2.2.5 Discussion

To show that other domains benefit from E-SSL in a qualitatively similar way to the applications in the previous section, here we introduce two datasets in photonics science. Figure 2-5 depicts the datasets, i.e. input-label pairs consisting of 2D square

Table 2.4: Linear probe accuracy (%) on ImageNet with longer pre-training. "BT" is short for "Barlow Twins."

Method	pre-training epochs		
	100	200	300
SimCLR (repro)	67.3	69.7	70.6
E-SimCLR (ours)	**68.3**	**70.5**	**71.5**
BT (repro)	66.9	70.0	71.1
E-BT (ours)	**68.2**	**71.0**	**71.9**

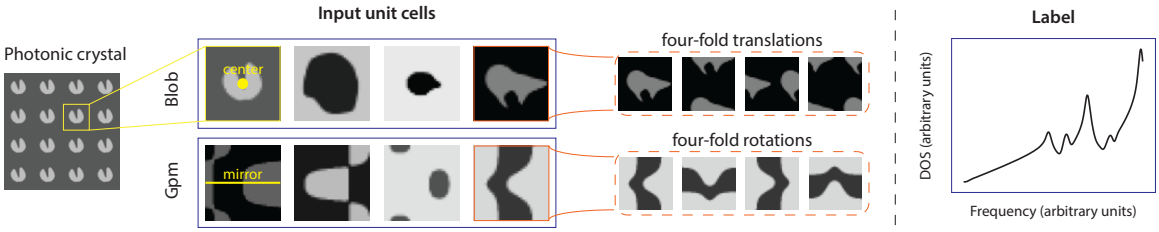

Figure 2-5: PhC datasets with transformations for sensitivity. The regression task is to predict the DOS labels (an example of a label in $\mathbb{R}^{400}$ is shown on the right) from 2D square periodic unit cells (examples of the inputs in $\mathbb{R}^{32\times32}$ are shown on the left). We consider two types of input unit cells; at the top is the Blob dataset where the feature variation is always centered; at the bottom is the Group pm (Gpm) dataset where inputs have a horizontal mirror symmetry.

periodic unit cells of PhCs and their associated DOS. The physics of the problem dictates that the DOS is invariant to (rolling) translations, scaling of all pixels by a fixed positive factor, and operations of the C_{4v} symmetry group, i.e. rotations and mirror flips. In choosing the transformations that E-SSL should encourage sensitivity to, we observe that the transformations that have worked for CIFAR-10 and ImageNet disturb the natural setting of the data (e.g. rotations disturb the natural upright setting of images). Thus, we encourage sensitivity to transformations that fit this observation, and insensitivity to the rest of the transformations.

In Figure 2-5, the top dataset is a "Blob" dataset where the shape variation in each image is centered. We encourage sensititivy to the group of *four-fold translations*, given by $G = \{e, h, v, hv\}$, where h and v are 1/2-unit cell translations in the horizontal and vertical axis, respectively, e is the unit element (no transformation) and hv is the composition of h and v. In the bottom dataset of Figure 2-5, the

Table 2.5: Fine-tuning the backbone on PhC datasets using 3000/ 2000 labeled train/ test samples. Relative error (%) is $\ell_{\text{DOS}} = (\sum_\omega \left|\text{DOS}^{\text{pred}} - \text{DOS}\right|)/(\sum_\omega \text{DOS})$. Lower is better. SimCLR for Blob includes C_{4v} (rotations and flips); SimCLR for Gpm includes rolling translations and mirrors. E-SimCLR encourages the features to be sensitive to the selected transformation explained in the text (four-fold translations for Blob and four-fold rotations for Gpm). "+ Transform" means adding this transformation to SimCLR. Error bars are for 3 different training data splits.

PhC Dataset	Supervised	SimCLR	SimCLR + Transform	E-SimCLR (ours)
Blob	$1.068_{\pm\ 0.015}$	$0.987_{\pm\ 0.005}$	$0.999_{\pm\ 0.005}$	$\mathbf{0.974}_{\pm\ 0.009}$
Gpm	$3.212_{\pm\ 0.041}$	$3.122_{\pm\ 0.002}$	$3.139_{\pm\ 0.005}$	$\mathbf{3.091}_{\pm\ 0.006}$

PhC unit cells are generated to have a horizontal mirror symmetry, i.e. we use the 2D wallpaper (or crystallographic plane) group **pm**. We encourage sensitivity to the group of *four-fold rotations* (the same group we used for CIFAR-10 and ImageNet), since rotating any of the images disturbs the (horizontal) mirror symmetry. More accurately, since only $\pm\pi/2$ rotations disturb the symmetry, we separate them in two classes, $\{\pi/2, -\pi/2\}$ and $\{0, \pi\}$, and perform binary prediction in E-SSL.

Table 2.5 shows the results of fine-tuning the backbone and an additional DOS-predictor head (see Appendix 2.2.11) with 3000 labelled samples for this regression task. We observe that encouraging sensitivity to the selected transformations (via E-SimCLR) leads to the largest reduction in the error. On the contrary, including these transformations to SimCLR (indicated by "+ Transform") increases the error. Furthermore, we explore scaling transformations and show that E-SSL can be generalized to infinite groups (see Appendix 2.2.11). This supports our observations about the usefulness of E-SSL over I-SSL and demonstrates E-SSL's generality beyond computer vision.

2.2.6 Appendix: Summary of Main Text and Layout of Appendix

In this paper we motivated the generalization of state-of-the-art methods in self-supervised learning to the more general framework of equivariant self-supervised learning (E-SSL). In E-SSL rather than using only invariance as a trivial case of

equivariance, we encouraged non-trivial equivariance and improved state-of-the-art methods on common computer vision benchmarks and regressions tasks in photonics science. We also discussed that there are many types of equivariance we can consider for E-SSL. We observed that most of the successful transformations for E-SSL that we explored form groups, but forsee that potentially many more transformations could be explored.

For future work one could learn transformations that are equivariances, instead of setting them manually. Thus, the concept of E-SSL could potentially be extended to natural language processing or other science domains, whose transformations for SSL are less well-understood. To facilitate further research in E-SSL, below we provide additional details and analysis of the experiments in the main text. We also discuss interesting avenues for future work.

2.2.7 Appendix A: Proof of Proposition 1

Proof. To construct a non-trivially equivariant f, we first need to show that both X' and S are G-sets, i.e. that there is a group action T_g of G on X', which is given by the statement of the proposition, and another (non-trivial) group action T'_g of G on S, which we will construct. Then, we need to show that f commutes with the group action, i.e. that $f(T_g(\boldsymbol{x}')) = T'_g(f(\boldsymbol{x}'))$.

Group actions. Note that by the setup of the problem, we are already given how G acts on the input X', i.e. T_g is known. For example, if G is the group of four-fold rotations, then T_g is the rotation of the input by a multiple of $\pi/2$. We proceed to construct the non-trivial group action T'_g of G on S.

Define the function $T'\colon G \times S \to S$ as $T'(g, s) = f(T_g(T_{g'}(\boldsymbol{x}')))$, where $s = f(T_{g'}(\boldsymbol{x}'))$. Note that T' is well-defined, because $gg' \in G$ by the closure of the group and s is uniquely written as $s = f(T_{g'}(\boldsymbol{x}'))$. To see why s is uniquely written, it suffices to show that if $f(T_{g'}(\boldsymbol{x}')) = f(T_{g''}(\boldsymbol{x}''))$ then both $g' = g''$ and $\boldsymbol{x}' = \boldsymbol{x}''$, which follows directly from our assumption in the statement.

Now, to prove that T' is a group action, it suffices to show two properties.

- Identity: $T'(e, s) = s$ for $s = f(g'(\boldsymbol{x}'))$ and e is the unit element of the group. To show that, note that by definition $T'(e, s) = f(T_e(T_{g'}(\boldsymbol{x}'))) = f(T_{g'}(\boldsymbol{x}'))$, because $eg' = g'$.

- Compositionality: $T'(g, T'(h, f(T_{g'}(\boldsymbol{x}')))) = T'(gh, f(T_{g'}(\boldsymbol{x}')))$. To show this, we expand the LHS and use the definition of T' to obtain as follows $T'(g, T'(h, f(T_{g'}(\boldsymbol{x}')))) = T'(g, f(T_h(T_{g'}(\boldsymbol{x}')))) = f(T_g(T_{hg'}(\boldsymbol{x}'))) = f(T_{ghg'}(\boldsymbol{x}')) = f(T_{gh}(T_{g'}(\boldsymbol{x}'))) = T'(gh, f(T_{g'}(\boldsymbol{x}')))$, because the group operation is associative.

Hence, T' is a group action, and thus S is a G-set, and we can write $T'(g, \cdot) \equiv T'_g(\cdot)$.

Commuting with the group action. To see this property, note that $T'_{g'}(f(\boldsymbol{x}')) = T'_{g'}(f(T_g(\boldsymbol{x}))) = f(T_{g'}(T_g(\boldsymbol{x}))) = f(T_{g'}(\boldsymbol{x}'))$ as desired. Note that T'_g is non-trivial.

Therefore, we can conclude that f, which satisfies the constructed group action T'_g, is not-trivially equivariant to the group G. □

2.2.8 Appendix B: Rotation prediction and I-SSL benefit from similar data augmentation.

Recently, rotation prediction with a linear head from the frozen backbone representations proved to be useful for validating the augmentation policies of contrastive learning [Reed et al., 2021]. This shows that the two tasks of classification of *ground truth classes* and *synthetic rotation classes* from frozen backbone representations benefit from similar augmentation policies. We took this experiment a step further, and performed rotation prediction with the augmentation policies, typically used in contrastive learning.

The result is in Table 2.6. Interestingly, RotNet benefits from augmentations, typically used in contrastive learning, and the RotNet training shares the same sweet spot [Tian et al., 2020b] as kNN classification. There are several takeaways from this experiment: (*i*) we can find good augmentations for contrastive learning by doing RotNet *alone*, i.e. without doing *any* contrastive learning; (*ii*) RotNet benefits from

augmentations needed in contrastive learning; (*iii*) we may be able to combine four-fold rotations prediction and contrastive learning.

Table 2.6: RotNet's augmentation sweet spot. kNN and Rotation Prediction have the same sweep spot (Level 4) which gives best accuracy in both columns. RotNet is trained on CIFAR-10 for 100 epochs with the same optimization setup as in our I-SSL experiments. Accuracies are on the test split. $(\downarrow \cdot)$ marks the deviation from the sweet spot. Every new level adds a new augmentation to the previous level incrementally.

Level	Added Augmentation	Supervised kNN Acc. (%)	Rotation Prediction Acc. (%)
0	none	44.8 (↓ 19.8)	90.2 (↓ 4.8)
1	random resized cropping	59.2 (↓ 5.4)	93.7 (↓ 1.3)
2	horizontal flips w.p. 0.5	59.4 (↓ 5.2)	94.5 (↓ 0.5)
3	color jitter w.p. 0.8	64.3 (↓ 0.3)	94.9 (↓ 0.1)
4	grayscale w.p. 0.2	**64.6**	**95.0**
5	Gaussian blur w.p. 0.2	64.1 (↓ 0.5)	94.5 (↓ 0.5)
6	random rotation ($\pm\pi/6$)	59.4 (↓ 5.2)	93.1 (↓ 1.9)
7	vertical flip w.p. 0.5	51.9 (↓ 12.7)	90.6 (↓ 4.4)

2.2.9 Appendix C: CIFAR-10 Experiments

Experimental setup

Our experiments use the following architectural choices: ResNet-18 backbone (the CIFAR-10 version has kernel size 3, stride 1, padding 1 and there is no max pooling afterwards); 512 batch size (only our baseline SimSiam model uses batch size 1024); 0.03 base learning rate for the baseline SimCLR and SimSiam and 0.06 base learning rate for E-SimCLR and E-SimSiam; 800 pre-training epochs; standard cosine decayed learning rate; 10 epochs for the linear warmup; two layer projector with hidden dimension 2048 and output dimension 2048; for SimSiam a two layer (bottleneck) predictor with hidden dimension 512 whose learning rate is not decayed; the last batch normalization for the projector does not have learnable affine parameters; 0.0005 weight decay value; SGD with momentum 0.9 optimizer. The augmentation is Random Resized Cropping with scale (0.2, 1.0), aspect ratio (3/4, 4/3) and size 32x32, Random horizontal Flips with probability 0.5, Color Jittering (0.4, 0.4, 0.4, 0.1) with probability 0.8 and Grayscale with probability 0.2. Some of our evaluations use a kNN-classifer with 200 neighbors, cosine similarity and Gaussian kernel with temperature 0.1. This evaluation correlates well with the standard linear probe, but it is more efficient to calculate. We report the kNN accuracy in % at the end of the

800 epochs of training. For our main results, we report a linear probe accuracy from training a linear classifier for 100 epochs on top of the frozen representations with SGD with momentum 0.9 and cosine decay of the learning rate, batch size 256 and initial laerning rate of 30. For linear probe experiments we try 5 different initalizations of the linear head and report mean and standard deviations. The deviations are negligible because the linear probe is robust to the random seed. All parameters are reported in a Pytorch-like style.

For Figure 2-1 we use resolution of 32x32 for the transformations studied. The 4 levels of the Gaussian blur are for kernel sizes 0, 5, 9 and 15 in the default Gaussian blur torchvision implementation. The prediction of the transformations follows the experimental setup in Section 2.2.3. When we apply the transformations in I-SSL, we add them in the beginning of the augmentation policy with probability 1. The same setup is used for "Disentangled representations" and "Insensitive instead of sensitive" in Table 2.2.

Additional experiments

Explored hyperparameters. Both for SimCLR and SimSiam we ran a grid search over the following hyperparameters: base learning rate: {0.01, 0.03, 0.06}, batch size: {512, 1024}, λ (for E-SSL): {0.0, 0.2, 0.4, 0.6, 0.8, 1.0}, predictor's MLP depth: {2, 3, 4}, predictor's normalization: {None, BatchNorm, LayerNorm}, nonlinearity at the last MLP layer of the predictor: {True, False}.

Tuning λ. Table 2.7 shows tuning of the CIFAR-10 results. We observe noticeable improvements over the SSL baselines by using E-SSL instead.

Table 2.7: Tuning the λ parameter for CIFAR-10.

Method	Baseline	E-SSL				
	0.0	0.2	0.4	0.6	0.8	1.0
SimCLR	92.0± 0.0	93.6± 0.0	**94.1± 0.0**	94.0± 0.0	**94.1± 0.0**	93.5± 0.0
SimSiam	91.1± 0.0	94.1± 0.0	**94.2± 0.1**	93.7± 0.0	93.8± 0.0	93.3± 0.0

Sensitivity to transformations for I-SSL. Table 2.8 demonstrates that E-SSL can produce good representation with as few SSL transformations for I-SSL as possible. We observe that E-SSL is less sensitive than SSL to the choice of data augmentation.

Table 2.8: Comparing the augmentation sensitivity for CIFAR-10. Levels: 0 is no transformations; 1 adds random resized cropping; 2 adds horizontal flips; 3 adds color jitter; 4 adds grayscale.

Method	Augmentation Level				
	0	1	2	3	4
SimCLR	28.0± 0.1	76.0± 0.1	77.0± 0.0	87.4± 0.0	92.0± 0.0
E-SimCLR	69.5± 0.0	83.5± 0.0	84.6± 0.1	91.6± 0.0	**94.1**± 0.0
SimSiam	17.6± 0.1	71.5± 0.0	72.2± 0.0	88.1± 0.0	91.1± 0.0
E-SimSiam	67.5± 0.1	84.6± 0.0	85.7± 0.1	92.9± 0.0	**94.2**± 0.1

The importance of complete invariance or sensitivity. Table 2.9 studies whether a middle ground for the representations exist, i.e. whether it is possible to have part of the representation invariant and the other part sensitive to the transformation. If we apply the E-SSL loss only to half of the representation, then there is a very small drop in the performance. Furthermore, we observe that having a disjoint mix between insensitivity and sensitivity in the representation is noticeably harmful.

Table 2.9: Studying the effect of disjoint representations on CIFAR-10. Split Representation means that we encourage similarity only on one half of the backbone representation. Disentangled Representation means that one half of the representation is trained to be insensitive to four-fold rotations and the other half is sensitive four-fold rotations. Linear probe accuracy (%) after 800 epochs.

Method	Baseline	Split Representation	Disentangled Representation
E-SimCLR	**94.1**± 0.0	94.1± 0.0 (↓ 0.0)	91.3± 0.0 (↓ 2.7)
E-SimSiam	**94.2**± 0.1	93.8± 0.0 (↓ 0.4)	91.1± 0.0 (↓ 3.1)

Fully connected backbone. We perform a simple experiment with a fully connected backbone, instead of a ResNet-18. The hidden dimensions of the backbone

are listed in order as {3×32×32, 2048, 2048, 512} with Batch Normalization and ReLUs in between. The rest of the experimental setup is exactly the same. On the linear probe (%), we obtain 70.5±0.0 for SimCLR and 73.8±0.1 for E-SimCLR, and 70.9±0.0 for SimSiam and 73.5±0.1 for E-SimSiam, highlighting noticeable gains from using E-SSL.

CIFAR-100 experiments. We test our CIFAR-10 experimental setup directly on CIFAR-100. On the linear probe (%), we obtain 65.8±0.0 for SimCLR and 69.5±0.1 for E-SimCLR, and 65.8±0.1 for SimSiam and 69.3±0.1 for E-SimSiam, highlighting sizable gains from using E-SSL.

Large crop study. We study whether using a large crop with a single rotation on CIFAR-10 can be just as good as a small crop. We obtain 93.9±0.0 on the linear probe using E-SimCLR, which is only 0.2 absolute points below our best result of 94.1±0.0 using four small crops.

Appendix D: Norm-differences Analysis

In Figure 2-6 we present analysis that shows our training objectives encourage invariance and equivariance to transformations. We take our best performing E-SimCLR and E-SimSiam methods on CIFAR-10. During training we keep track of two measures that can capture how invariant/ equivariant the backbone representations are.

The "invariance measure" computes the negative cosine similarity between two views of the backbone representations. The lower this measure is, the higher the similarity between the two views, and thus the more invariant the backbone representations are to the transformations in I-SSL. We observe that during training high similarity between the two views is maintained (roughly between 0.8 and 0.9), which indicates that invariance is encouraged in the backbone representations, as desired.

Likewise, the "equivariance measure" computes the average cosine similarity of the backbone represenations, between all six pairs of the four rotated views. The lower this measure is, the lower the similarity between the four views, and thus the more

non-trivially equivariant the backbone representations are to the transformations for equivariance. We observe that the measure decays to about 0.3 during training, which indicates that the backbone representations are encouraged to be equivariant to four-fold rotations, as desired.

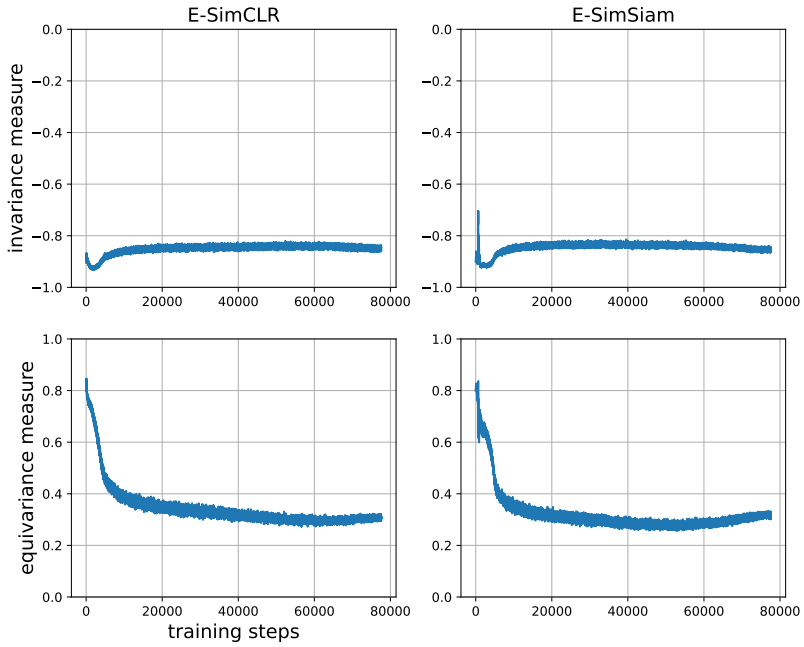

Figure 2-6: Demonstration of the evolution of the invariance (top) and equivariance (bottom) measures during training. Left is E-SimCLR and right is E-SimSiam.

2.2.10 Appendix E: ImageNet Experiments

We had limited computational resources, so we kept the learning rates the same as in the original methods. Only for SimSiam we found that choosing a smaller learning rate 0.08 leads to better results for E-SimSiam. We only swept the λ parameter, where for SimCLR and SimSiam the sweep was between 0 and 1 and for Barlow Twins it was between 0 and 100. The optimal λ is 0.4 for SimCLR, 0.08 for SimSiam, 8 for Barlow Twins. We use (0.05, 0.14) scale range for 100 pre-training epochs. For more pre-training epochs we use (0.05, 0.14) for SimCLR and (0.08, 1.0) for Barlow Twins.

Table 2.10 lists the overhead from using rotation prediction in our experiments.

Table 2.10: Overhead in doing rotation prediction. Reported GPU hours for an experiment on 100 epochs.

	SimCLR	SimSiam	Barlow Twins
Baseline	256	295	246
E-SSL (ours)	307	364	294
Overhead	20%	23%	19%

2.2.11 Appenix F: PhC Experiments

Dataset generation. 2D Photonic crystals (PhCs) are characterized by a periodically varying permitivitty $\varepsilon(x, y)$; here, for simplicity we consider a "two-tone" permitivitty profile i.e. $\varepsilon \in \{\varepsilon_1, \varepsilon_2\}$, with $\varepsilon_i \in [1, 20]$ discretized to a resolution of 32×32. To generate the unit cells in the "blob" dataset, we follow the proceedure in Christensen et al. [2020]. For the Gpm dataset, the unit cells are defined using a level set of a 2D Fourier sum function like in Kim et al. [2021a], Loh et al. [2021], with additional constraints applied to the lattice to create the mirror symmetry adopted from the method in Christensen et al. [2021]. We then follow the procedure in Loh et al. [2021] to compute, and subsequently process, the density-of-states (DOS) of each unit cell, specifically, via the MIT Photonics Bands (MPB) software [Johnson and Joannopoulos, 2001] and the Generalized Gilat-Raubenheimer method in an implementation from Liu et al. [2018a].

Network architecture. We use an encoder network composing of simple convolutional (CNN) and fully-connected (FC) layers for the backbone; specifically, our backbone begins with 3 CNN layers, all with a kernel size of 7 and channel dimensions given by $[64, 256, 256]$. The output is flattened and fed into 2 FC layers each with 1024 nodes (i.e. the representations have dimension 1024). We include BatchNorm [Ioffe and Szegedy, 2015], ReLU and MaxPooling for the CNNs, and ReLU only for the first FC layer. The projector and predictor networks, p_1 and p_2 are 2-layer MLPs with hidden dimension 512, with BatchNorm and ReLU between each layer

except the last and the projection dimension for p_1 is 256. Additionally, since this is a regression task and the label space is much larger than in image classification tasks, we include a dense DOS-predictor head after the representations, which is fine-tuned with 3000 labelled samples after SSL or E-SSL. The DOS-predictor has 4 FC layers, with number of nodes given by $[1024, 1024, 512, 400]$. We explore two fine-tuning protocols of the DOS-predictor: freezing the backbone (discussed later in the Appendix) or fine-tuning the backbone (discussed in the main text).

Hyperparameters. For SSL and E-SSL, we performed 250 pre-training epochs using the SGD optimizer with a standard cosine decayed learning rate; the batch size was fixed to 512. The pre-trained model was saved at various epochs {20, 50, 100, 180, 250} for further fine-tuning. Fine-tuning was performed for 100 epochs using Adam optimizer and a fixed batch size of 64. No transformations were applied to the input during fine-tuning for both freezing or fine-tuning the backbone. We ran a grid search over the following hyperparameters; for pre-training, base learning rate: $\{10^{-3}, 10^{-4}, 10^{-5}\}$, λ (for E-SSL): {0.2, 1.0, 2.0, 5.0, 10.0}, and for fine-tuning: a learning rate in $\{10^{-3}, 10^{-4}, 10^{-5}\}$.

Frozen backbone experiment. In Table 2.11 we present our results from freezing the backbone encoder while fine-tuning the DOS-predictor head. We observe similar trends as in Table 2.5 where we allowed fine-tuning of the backbone. Relative error is reported in % and the lower the error is, the better. SimCLR for Blob includes C_{4v} (rotations and flips) and SimCLR for Gpm includes rolling translations and flips. E-SimCLR encourages the features to be sensitive to the selected transformation (four-fold translations for Blob and four-fold rotations for Gpm), which improves the performance of SimCLR. On the contrary, adding the selected transformation to SimCLR, as indicated by "+ Transform", increases the error of SimCLR. Error bars are reported for 3 different choices of training data. Supervised (frozen) refers to the impractical situation of freezing a random backbone and fine-tuning the DOS-predictor.

Table 2.11: Frozen backbone experiment on PhC datasets for 3000/ 2000 labelled train/ test samples.

PhC Dataset	Supervised (frozen)	SimCLR	SimCLR + Transform	E-SimCLR (ours)
Blob	1.686± 0.014	1.237± 0.005	1.242± 0.013	**1.165**± 0.020
Gpm	5.450± 0.077	3.214± 0.048	3.313± 0.029	**3.187**± 0.000

Continuous group experiment. In all experiments shown so far, we dealt with finite groups of transformations. To show that E-SSL generalizes beyond the finite group setting, we also explore transformations from a continuous group. An example is the scaling transformation where every pixel of the input unit cell is scaled by the same positive factor. More specifically, this set of positive scaling transformations $g(s)\boldsymbol{x} = s\boldsymbol{x}$ defines a continuous group $G = \{g(s)|s \in \mathbb{R}_+\}$ which leaves the DOS labels invariant due to the physics of the problem and normalization applied when pre-processing the dataset [Loh et al., 2021]. In our experiment, we uniformly sample $s \in (1, s_{max}]$ and apply the inverse with probability 0.5 (i.e. we cap the scaling factor to a maximum of $s_{max} = \{5, 10\}$ for numerical stability during training and we apply up-scaling and down-scaling with equal probability). To encourage equivariance to this group, we simply predict the scale factor applied to the input using L1 loss (i.e. the final layer of the predictor p_2 is a single node). In Table 2.12, we show results after fine-tuning the backbone and the DOS predictor network with 3000 labelled samples. We observe similar trends to Table 2.5; encouraging sensitivity to scaling produces the lowest error and including scaling to SimCLR increases the error. To isolate the effect of scaling transformation, the remaining physics-governed invariances excluding scaling (translations, rotations and mirrors) are used in SimCLR and the invariance part of E-SimCLR for both datasets.

2.3 Flowers-102 Experiments

In order to study the importance of the assumption in Proposition 1 we perform an experiment with a dataset that might not be amenable to such an assumption at first sight. We choose the Flowers-102 dataset [Nilsback and Zisserman, 2008], because at first sight the dataset might not benefit from four-fold rotations in E-SSL. Therefore,

Table 2.12: Fine-tuning the backbone on PhC datasets using 3000/ 2000 labelled train/ test samples. Relative error (%) is $\ell_{\text{DOS}} = (\sum_\omega \left|\text{DOS}^{\text{pred}} - \text{DOS}\right|)/(\sum_\omega \text{DOS})$. Lower is better. E-SimCLR encourages the features to be sensitive to scaling. "+ Scaling" means adding scaling to SimCLR. Error bars are for 3 different training data splits.

PhC Dataset	Supervised	SimCLR	SimCLR + Scaling	E-SimCLR (ours)
Blob ($s_{max} = 10$)	1.068 $\pm$ 0.015	0.988$\pm$ 0.001	1.005$\pm$ 0.006	**0.974**$\pm$ 0.000
Blob ($s_{max} = 5$)	1.068 $\pm$ 0.015	0.988$\pm$ 0.001	1.000$\pm$ 0.014	**0.987**$\pm$ 0.017
Gpm ($s_{max} = 10$)	3.212$\pm$ 0.041	3.073$\pm$ 0.003	3.112$\pm$0.011	**3.062**$\pm$ 0.005
Gpm ($s_{max} = 5$)	3.212$\pm$ 0.041	3.073$\pm$ 0.003	3.082$\pm$0.013	**3.058**$\pm$ 0.008

this dataset complements the experiments we performed for CIFAR-10 and ImageNet.

Experimental setup. We train SimCLR and E-SimCLR. We use the same optimization hyperparameters from the experimental setup for our CIFAR-10 experiments. We downsize the images to 96x96 resolution and use the standard ResNet-18, instead of its modified version for CIFAR-10. For the data augmentation in I-SSL, we use the same RandomResizedCropping as in CIFAR-10 (with size 96 of the crops being the only difference), the same Color Jittering and Random horizontal flips as in the CIFAR-10 experiment. We report the kNN accuracy in (%) on the validation set. We study both four-fold rotations and four-fold translations as transformations for invariance/ equivariance. Four-fold rotations are chosen following the hypothesis that most of the data points should be invariant to rotation. Four-fold translations are chosen, because of our observation that most of the data points are centered, just like in the Blob PhC dataset. The λ for predicting four-fold translations is 0.01 and for four-fold rotations is 0.5 (chosen from a grid search among {0.001, 0.01, 0.1, 0.5, 1.0, 2.0}).

Results. Following our observations in Figure 2-1, we observe that encouraging insensitivity to four-fold rotations and translations, by adding the transformations to the SimCLR data augmentation, worsens the SimCLR baseline. In contrast, using these transformations for E-SSL improves the baselines and further shows the utility of E-SSL for real-world data. We even observe benefit from encouraging equivariance

to four-fold rotations, which is against the intuition that rotations should be invariant. This is probably due to the fact that some images in the dataset are not truly rotationally invariant (see examples of the data points in Figure 2-8).

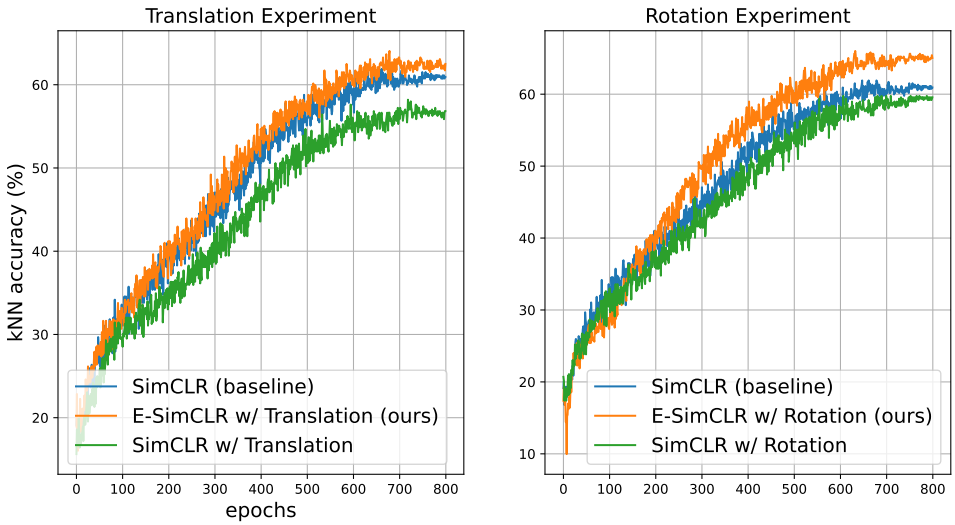

Figure 2-7: E-SimCLR gives sizable improvements for the Flowers-102 SSL pre-training. kNN accuracy (%) is on the validation set.

2.3.1 Appendix G: Relative Orientation Prediction

In our experiments we demonstrate that if a shared biased between the train and test sets exists, we should exploit it via the E-SSL training objective. However, in some scenarios, the class label of the downstream tasks depends on the orientation (e.g., classifying road signs) and the current E-SSL method may not be very useful, because both $\boldsymbol{x}$ and $T_g(\boldsymbol{x})$ exist in the data. This situation invites a natural generalization of our method in the spirit of [Agrawal et al., 2015]. If $\boldsymbol{x}$ and $T_g(\boldsymbol{x})$ exist in the data then we can modify E-SSL minimally to form a useful objective. In particular, we can set the objective for p_2 to predict the relative orientation between two data points, i.e. given $\boldsymbol{x}$, we form $T_{g'}(\boldsymbol{x})$ by sampling $T_{g'}$ from G uniformly, and then predict g' from $p_2(\boldsymbol{z}; \boldsymbol{\theta}_{p_2})$, where $z = [f(\boldsymbol{x}), f(T_{g'}(\boldsymbol{x}))]$ is the concatenation of the two representations. This modification requires minimal change to our framework.

Figure 2-8: The Flowers-102 is not completely invariant to rotation. The top row shows data points which are roughly invariant to four-fold translations. The bottom row shows counterexamples to that hypothesis.

To test the usefulness of the modified method when $\boldsymbol{x}$ and $T_{g'}(\boldsymbol{x})$ exist in the data, we artificially modify CIFAR-10 so that any rotation of an image can appear in the dataset. We consider the downstream task of predicting the rotation orientation of an image, which clearly depends on the orientation of the image. Our hypothesis is that the modified E-SimCLR will be better than SimCLR, which is what we observe in our results below. Intuitively, SimCLR is not able to capture the orientation of the image, while the modified E-SimCLR is able to, because the latter predicts relative orientation.

The experimental setting is as follows: we pretrain for 100 epochs. The predictor for equivariance's input dimension is doubled, because we concatenate two representations. We set λ to 0.4. All other hyperparameters are the same as the rest of the CIFAR-10 experiments. The downstream task is 4-way rotation orientation classification. Using pre-training with SimCLR on the standard CIFAR-10 dataset, we obtain

baseline linear probe (%) accuracy 67.1±0.1. Using our modification of E-SSL on the same experimental setting, we obtain **71.2±0.1**. linear probe accuracy, which is a sizable gain and points to the promise of the relative orientation prediction as future work.

The relative orientation prediction scenario is also highly relevant in other domains such as in photonic crystals. For example, we can modify the PhC setup and remove the "orientation bias" of the datasets while predicting a different property, the band structures. Unlike the DOS, the PhC band structures are not invariant to rotations and so we can consider the group of four-fold rotations. This setup would fit the scenario described above since 1) both $\boldsymbol{x}$ and $T_{g'}(\boldsymbol{x})$ exist in the data due to the lack of bias and 2) the downstream task is sensitive to rotation. We will explore the above framework on this setup in future work.

Outlook. In this section we present equivariant contrastive learning as a framework for using transformations in various ways, equivariant or invariant. We showed that each dataset has a preference for invariance or equivariance to a particular transformation. However, we only considered the downstream task performance aggregated across the entire dataset. What if a single dataset benefits from a *variety of symmetries*, both invariant and equivariant? We address the limitation of equivariant contrastive learning in the following section.

2.4 Multi-Symmetry Ensembles

To motivate the study of multiple symmetries, we consider the transformation of 4-fold rotations from the previous section. In Table 2.13 we consider various pretraining strategies ("Baseline" SimCLR training, "Eq", E-SimCLR and "Inv", I-SimCLR), and show their accuracies at the top of the table. For each of the 1,000 classes in the dataset, we compute the accuracy per each pretraining strategy after fine-tuning on ImageNet. Then we check the proportion of the classes, where Eq performs the best, Inv performs the best, or where both pretraining strategies are similar. We observe

Table 2.13: **Most suitable functional class differs within a dataset.** The top-half shows the overall accuracy for models from the SimCLR baseline and each of the opposing hypotheses wrt 4-fold rotations. The bottom-half shows the proportion of classes within each dataset where each hypotheses dominate (i.e. averaged over all samples within the class), suggesting that hypotheses apart from the one with the highest individual accuracy are still beneficial.

Model Accuracy on ImageNet (%)	
Baseline	76.5
Eq	76.9
Inv	76.0
Proportion of Classes (%)	
Eq > Inv	47.7
Eq < Inv	36.3
Eq == Inv	16.0

that a sizable portion of the classes prefer Eq, and likewise for Inv. This experiment suggests that both symmetries create useful hypothesis spaces during pre-training that could be exploited in the fine-tuning stage. However, what is a useful way of combining these symmetries?

In Figure 2-9 we present a very simple and natural method to combine multiple symmetries, which we dub *Multi-Symmetry Ensembles* (MSE). In Panel (a) we introduce the notion of our method. In green we show multiple solutions around one hypothesis, which forms the methods of deep ensembles *Deep Ensembles*: averages of the predicted distributions of multiple solutions around one hypothesis (in green) [Lakshminarayanan et al., 2016]. While this method offers sizable gains in accuracy and improved calibration, it is only limited to a single hypothesis, which seems sub-optimal as judged by our experiments presented in Table 2.13. Thus, we introduce multiple axes of hypotheses labelled by H_i for $i = 1, 2, 3$ where we collect multiple solutions around a hypothesis that is either equivariant or invariant to a corresponding transformation T_i. We could consider multiple types of transformations T. In Panel (b) we present the four-fold rotation.

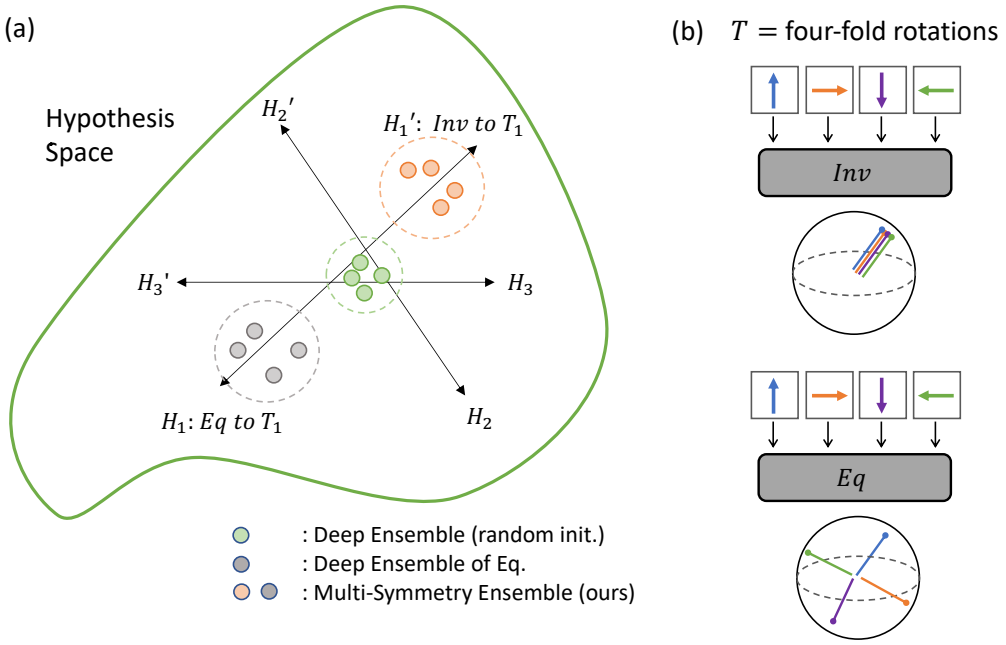

Figure 2-9: **(a)** A comparative illustration of the diversity in the hypothesis space that traditional deep ensembles and our MSE can achieve. While deep ensembles are effective at capturing different solutions around one hypothesis, MSE can learn diverse solutions around inherently opposing hypotheses. **(b)** Schematic visualization of invariance (top) v.s. equivariance (bottom) for the four-fold rotation. The spheres denote the representation space of the models.

Scalability. In Figure 2-10 we present the effect of having a variety of hypotheses in an ensemble (MSE) on scaling the accuracy of the ensemble. Namely, we observe that using both equivariant and invariant members of the ensemble (orange line, our MSE method) is the *only* method that continues scaling noticeably by increasing the number of ensemble members.

Mixing transformations. In Table 2.14 we also demonstrate a new capability of equivariant contrastive learning, where we can combine multiple transformations in a single classifier in the form of an MSE. Recall that we found it challenging to combine multiple transformations in a single model [Dangovski et al., 2021a]. Using MSE is a

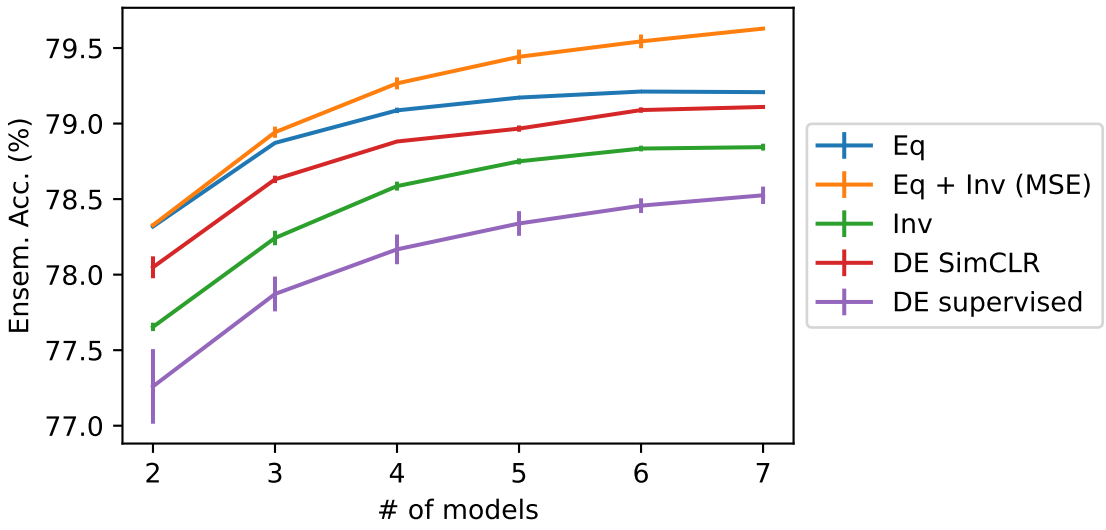

Figure 2-10: **Ensembles with opposing hypotheses have significantly larger potential.** Ensembles constructed only from a single hypothesis very quickly give marginal ensembling gains from adding more members.

more scalable alternative that enables noticeable improvements (of about 1%) when all transformations are combined.

Effectiveness of MSE depends on dataset diversity. In Figure 2-11 we present the *ensemble efficiency*, the change in the performance of the ensemble relative to the mean accuracy of the individual models in the ensemble. we aim to provide empirical guidance on when the inclusion of opposing hypotheses in an ensemble is beneficial. We evaluate the proportion of classes dominated by each of the opposing hypotheses (invariant and equivariant symmetries) for different datasets, including iNaturalist-1k, CIFAR-100, ImageNet-V2, and ImageNet-R. These results are shown in 2-11. Our findings indicate that on datasets such as iNaturalist-1k, the inclusion of opposing hypotheses in the ensemble improves performance. However, on datasets like CIFAR-100 and ImageNet-R, the opposing hypotheses do not provide significant gains. This is because these datasets have a high level of imbalance between the dominance of the two hypotheses, with one hypothesis dominating in a majority of the classes. For example, in ImageNet-R, the equivariant hypothesis dominates in 76.5% of the classes

Table 2.14: **Capturing opposing hypotheses across transformations for $M = 6$.** The upper three rows are ensembles that consist of both equivariant and invariant learners with respect to a single transformation and the bottom row greedily searches over all models across the three transformations.

Ensemble Accuracy on ImageNet-100 (%)	
Rotate	86.60
Halfswap	86.00
ColorInvert	86.26
Rotate + Halfswap + ColorInvert	87.22

while the invariant hypothesis only dominates in 18% of classes. These datasets are poorly described by the opposing hypothesis and thus including them in the ensemble provides little to no improvement in performance.

Empirical intuition on datasets where MSE are effective. In our work, we found the effectiveness of MSE to be highly dependent on dataset diversity. In particular, if the datasets are poorly described by the opposing hypothesis (i.e. ImageNet-R, i.e. IN-R in Figure 2-11), the gains from MSE would be negligible. Here, we provide some intuition on why this may be so. Following the intuition provided in the previous paragraph, we conjecture that this could be related to the existence of a dominant pose of images in the dataset. An example of the class of "jellyfish" in ImageNet (IN) and IN-R is shown in 2-12. In IN-R which contains renditions of the images, such as in cartoon and art, many images assume a conventional "upright" pose of the jellyfish with its head on top and its tentacles trailing below vertically. However, in IN where real-life jellyfish are imaged, they often occur in multiple poses. We believe this is true for other classes as well, since artists often draw objects in their 'conventional pose'. Thus, for IN, invariant models are useful for 36.3% (v.s. equivariant models being useful in 47.7%). In contrast, for IN-R, invariant models are dominant only for 18% of the classes (v.s. equivariant models being dominant in 76.5%). Given the existence of an upright pose in IN-R, equivariant models that encode pose information are likely more useful than invariant models leading to this stark difference.

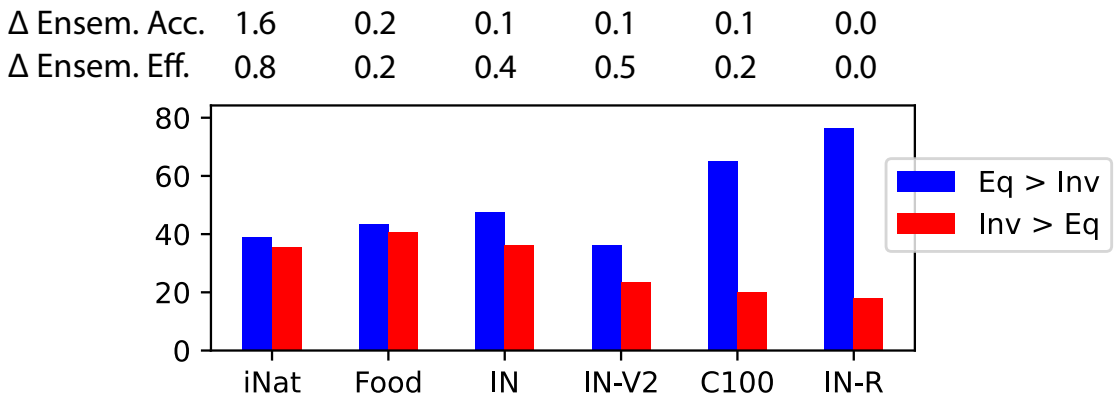

Figure 2-11: **Understanding the effectiveness of including the opposing hypothesis.** Plot shows the proportion of classes in each dataset where each hypothesis dominates. The remaining proportions (not shown) are classes where Eq and Inv are equally performant. Gains are minimal in datasets with a high level of imbalance between the leading and opposing hypothesis.

Outlook. Our work on MSE addressed the limitation of our work on equivariant contrastive learning by allowing a single classifier (an ensemble) to contain different types of hypotheses (equivariant or invariant). However, we used our background knowledge for what *might* be a good transformations, such as Rotation, Halfswap, or ColorInvert (see Table 2.14). In reality our intuition of what might be a useful synthetic transformation could fail, and we might be challenged to use any meaningful synthetic transformation at all. We address that problem in the following section by studying contrastive learning for sentence embedding.

2.5 DiffCSE: Difference-Based Contrastive Learning for Sentence Embeddings

Motivation. Learning "universal" sentence representations that capture rich semantic information and are at the same time performant across a wide range of downstream NLP tasks without task-specific finetuning is an important open issue in the field [Conneau et al., 2017, Cer et al., 2018, Kiros et al., 2015, Logeswaran and

Figure 2-12: **Examples of images from the "jellyfish" class in ImageNet (left) and ImageNet-R (right).** Samples visualized using `https://knowyourdata-tfds.withgoogle.com/`

Lee, 2018, Giorgi et al., 2020, Yan et al., 2021, Gao et al., 2021]. Recent work has shown that finetuning pretrained language models with *contrastive learning* makes it possible to learn good sentence embeddings without any labeled data [Giorgi et al., 2020, Yan et al., 2021, Gao et al., 2021]. Contrastive learning uses multiple augmentations on a single datum to construct positive pairs whose representations are trained to be more similar to one another than negative pairs. While different data augmentations (random cropping, color jitter, rotations, etc.) have been found to be crucial for pretraining vision models [Chen et al., 2020a], such augmentations have generally been unsuccessful when applied to contrastive learning of sentence embeddings. Indeed, Gao et al. [2021] find that constructing positive pairs in SimCLR via a simple dropout-based augmentation works much better than more complex augmentations such as word deletions or replacements based on synonyms or masked language models. This is perhaps unsurprising in hindsight; while the training objective in contrastive learning encourages representations to be *invariant* to augmentation transformations, direct augmentations on the input (e.g., deletion, replacement) often change the meaning of the sentence. That is, ideal sentence embeddings should *not* be invariant to such transformations.

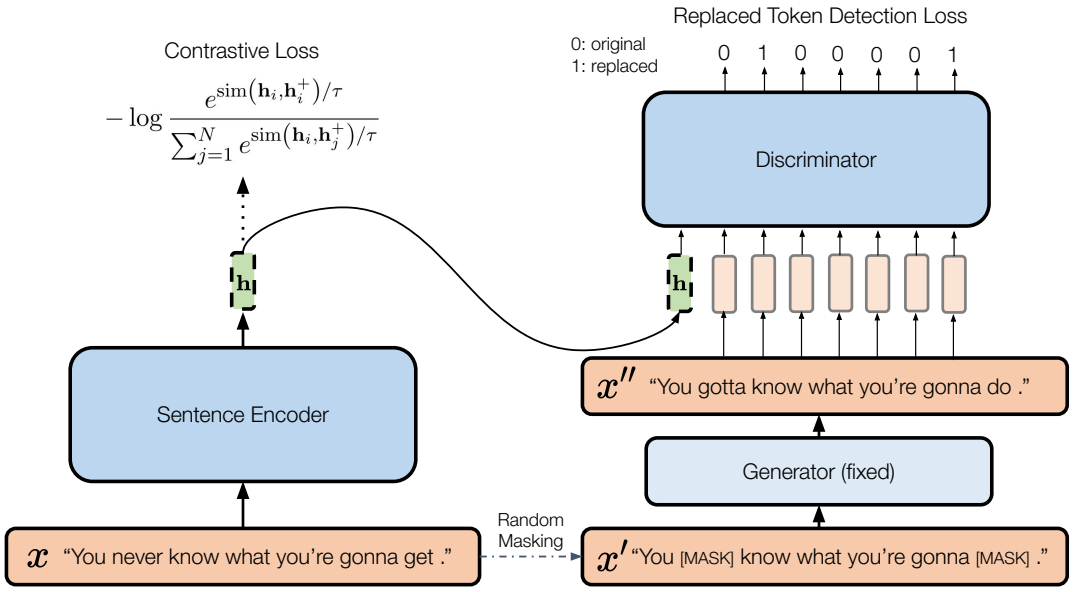

Figure 2-13: Illustration of DiffCSE. On the left-hand side is a standard SimCSE model trained with regular contrastive loss on dropout transformations. On the right hand side is a conditional difference prediction model which takes the sentence vector $\mathbf{h}$ as input and predict the difference between x and x''. During testing we discard the discriminator and only use $\mathbf{h}$ as the sentence embedding.

Our method: DiffCSE. In Figure 2-13 we demonstrate the application of equivariant contrastive learning to sentence embeddings. We present an additional training objective of the Sentence Encoder[1] from the SimCSE "Contrastive Loss" objective (left). Namely, we apply random masking to the input sentence and use a small fixed Generator[2] to replace the masked tokens. Then we condition a Discriminator (the same model as the Sentence Encoder) on the hidden state from the Sentence Encoder and the modified sentence from the Generator to perform the "Replaced Token Detection Loss" (right).

We are inspired by a recent generalization of contrastive learning in computer vision (CV) called equivariant contrastive learning [Dangovski et al., 2021a]. We now explain how this CV technique can be adapted to natural language.

[1]We also use the checkpoints of BERT Devlin et al. [2019] and RoBERTa Liu et al. [2019] as the initialization of our sentence encoder.

[2]DistilBERT or DistilRoBERTa [Sanh et al., 2019].

Understanding the role of input transformations is crucial for successful contrastive learning. Past empirical studies have revealed useful transformations for contrastive learning, such as random resized cropping and color jitter for computer vision Chen et al. [2020a] and dropout for NLP Gao et al. [2021]. Contrastive learning encourages representations to be insensitive to these transformations, i.e. the encoder is trained to be invariant to a set of manually chosen transformations. The above studies in CV and NLP have also revealed transformations that are *harmful* for contrastive learning. For example, Chen et al. [2020a] showed that making the representations insensitive to rotations decreases the ImageNet linear probe accuracy, and Gao et al. [2021] showed that using an MLM to replace 15% of the words drastically reduces performance on STS-B. While previous works simply omit these transformations from contrastive pre-training, here we argue that we should still make use of these transformations by learning representations that are *sensitive* (but not necessarily invariant) to such transformations.

The notion of (in)sensitivity can be captured by the more general property of equivariance in mathematics. Let T be a transformation from a group G and let $T(x)$ denote the transformation of a sentence x. Equivariance is the property that there is an induced group transformation T' on the output features [Dangovski et al., 2021a]:

$$f(T(x)) = T'(f(x)).$$

In the special case of contrastive learning, T''s target is the identity transformation, and we say that f is trained to be "invariant to T." However, invariance is just a trivial case of equivariance, and we can design training objectives where T' is not the identity for some transformations (such as MLM), while it is the identity for others (such as dropout). Dangovski et al. [2021a] show that generalizing contrastive learning to equivariance in this way improves the semantic quality of features in CV, and here we show that the complementary nature of invariance and equivariance extends to the NLP domain. The key observation is that the encoder should be equivariant to MLM-based augmentation instead of being invariant. We can operationalize this by using a conditional discriminator that combines the sentence representation with an

edited sentence, and then predicts the *difference* between the original and edited sentences. This is essentially a conditional version of the ELECTRA model [Clark et al., 2020], which makes the encoder equivariant to MLM by using a binary discriminator which detects whether a token is from the original sentence or from a generator. We hypothesize that conditioning the ELECTRA model with the representation from our sentence encoder is a useful objective for encouraging f to be "equivariant to MLM."

To the best of our knowledge, we are the first to observe and highlight the above parallel between CV and NLP. In particular, we show that equivariant contrastive learning extends beyond CV, and that it works for transformations even without algebraic structures, such as diff operations on sentences. Further, insofar as the canonical set of useful transformations is less established in NLP than is in CV, DiffCSE can serve as a diagnostic tool for NLP researchers to discover useful transformations.

Data. For unsupervised pretraining, we use the same 10^6 randomly sampled sentences from English Wikipedia that are provided by the source code of SimCSE. We evaluate our model on 7 semantic textual similarity (STS) and 7 transfer tasks in SentEval.[3] STS tasks includes STS 2012–2016 [Agirre et al., 2016], STS Benchmark [Cer et al., 2017] and SICK-Relatedness [Marelli et al., 2014]. All the STS experiments are fully unsupervised, which means no STS training datasets are used and all embeddings are fixed once they are trained.

Results. We compare our model with many strong unsupervised baselines including SimCSE [Gao et al., 2021], IS-BERT [Zhang et al., 2020], CMLM [Yang et al., 2020], DeCLUTR [Giorgi et al., 2020], CT-BERT [Carlsson et al., 2021], SG-OPT [Kim et al., 2021b] and some post-processing methods like BERT-flow [Li et al., 2020] and BERT-whitening [Su et al., 2021] along with some naive baselines like averaged GloVe embeddings [Pennington et al., 2014] and averaged first and last layer BERT embeddings.

We show the results of STS tasks in Table 2.15 including $\text{BERT}_{\texttt{base}}$ (upper part)

[3] `https://github.com/facebookresearch/SentEval`

and RoBERTa$_{\texttt{base}}$ (lower part). We also reproduce the previous state-of-the-art SimCSE [Gao et al., 2021]. DiffCSE-BERT$_{\texttt{base}}$ can significantly outperform SimCSE-BERT$_{\texttt{base}}$ and raise the averaged Spearman's correlation from 76.25% to 78.49%. For the RoBERTa model, DiffCSE-RoBERTa$_{\texttt{base}}$ can also improve upon SimCSE-RoBERTa$_{\texttt{base}}$ from 76.57% to 77.80%.

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
GloVe embeddings (avg.)♣	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
BERT$_{\text{base}}$ (first-last avg.)♢	39.70	59.38	49.67	66.03	66.19	53.87	62.06	56.70
BERT$_{\text{base}}$-flow♢	58.40	67.10	60.85	75.16	71.22	68.66	64.47	66.55
BERT$_{\text{base}}$-whitening♢	57.83	66.90	60.90	75.08	71.31	68.24	63.73	66.28
IS-BERT$_{\text{base}}$ ♡	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
CMLM-BERT$_{\text{base}}$ ♠ (1TB data)	58.20	61.07	61.67	73.32	74.88	76.60	64.80	67.22
CT-BERT$_{\text{base}}$ ♢	61.63	76.80	68.47	77.50	76.48	74.31	69.19	72.05
SG-OPT-BERT$_{\text{base}}$ †	66.84	80.13	71.23	81.56	77.17	77.23	68.16	74.62
SimCSE-BERT$_{\text{base}}$ ♢	68.40	82.41	74.38	80.91	78.56	76.85	**72.23**	76.25
∗ SimCSE-BERT$_{\text{base}}$(reproduce)	70.82	82.24	73.25	81.38	77.06	77.24	71.16	76.16
∗ DiffCSE-BERT$_{\text{base}}$	**72.28**	**84.43**	**76.47**	**83.90**	**80.54**	**80.59**	71.23	**78.49**
RoBERTa$_{\text{base}}$ (first-last avg.)♢	40.88	58.74	49.07	65.63	61.48	58.55	61.63	56.57
RoBERTa$_{\text{base}}$-whitening♢	46.99	63.24	57.23	71.36	68.99	61.36	62.91	61.73
DeCLUTR-RoBERTa$_{\text{base}}$ ♢	52.41	75.19	65.52	77.12	78.63	72.41	68.62	69.99
SimCSE-RoBERTa$_{\text{base}}$ ♢	**70.16**	81.77	73.24	81.36	80.65	80.22	68.56	76.57
∗ SimCSE-RoBERTa$_{\text{base}}$(reproduce)	68.60	81.36	73.16	81.61	80.76	80.58	68.83	76.41
∗ DiffCSE-RoBERTa$_{\text{base}}$	70.05	**83.43**	**75.49**	**82.81**	**82.12**	**82.38**	**71.19**	**78.21**

Table 2.15: The performance on STS tasks (Spearman's correlation) for different sentence embedding models. ♣: results from Reimers and Gurevych [2019]; ♡: results from Zhang et al. [2020]; ♢: results from Gao et al. [2021]; ♠: results from Yang et al. [2020]; †: results from Kim et al. [2021b]; ∗: results from our experiments.

Outlook. So far in our discussion we have used symmetry-based inductive biases to improve representation learning in computer vision and natural language processing. A critical part of our new question is whether we could utilize these improvements in representation learning to mitigate the issues with representation learning for Science. We address that challenge in the following section by applying our insights for data-scarce applications in Science.

2.6 Surrogate- and Invariance-Boosted Contrastive Learning for Data-Scarce Applications in Science

In Figure 2-14 we demonstrate the power of contrastive learning to improve data-scarce applications. Since we have a limited number of labeled data for predicting

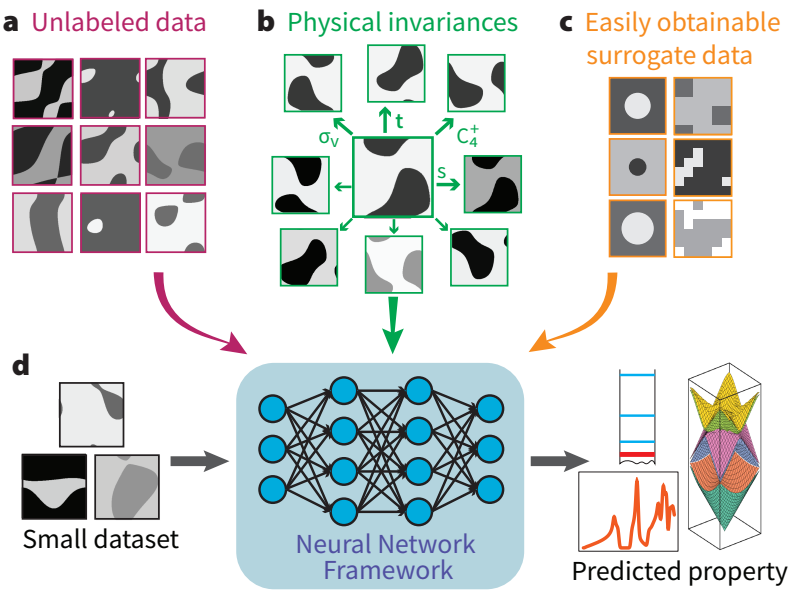

Figure 2-14: **Overcoming data scarcity with SIB-CL.** We propose to overcome data scarcity by leveraging a) an abundance of unlabeled data, b) prior knowledge of the underlying physics (e.g., symmetries and invariances of the data) and c) knowledge from a possibly-approximate surrogate data which is faster and cheaper to generate (e.g., coarse-grained computations or special-case analytical solutions). d) SIB-CL incorporates these auxiliary information into a single framework to accelerate training in data-scarce settings.

the desired physical property (density of states, bandgaps, ground states, etc.), we make use of useful background knowledge and resources that are cheap to obtain and do not require extensive human annotation. Namely, in Panel a) we show examples of 2D PhC unit cells that are unlabeled but easy to obtain. In Panel b) we demonstrate the transformations under which the physical properties should be invariant. We encourage that property with a contrastive learning objective. In Panel c) we use a simpler, easily obtainable, surrogate dataset that is labeled with the desired physical properties and so it could also be used for pretraining. We use a-c) to pretrain a neural network framework, which we further finetune on a small dataset. The objective of our work is to improve the prediction performance of the neural network with our pretraining strategy.

Our method, *Surrogate- and Invariance-Boosted Contrastive Learning (SIB-CL)* is sketched in Panel d). SIB-CL applies the physical invariances from Panel b) as

data augmentation for contrastive learning on the unlabeled dataset from Panel a). Additionally, jointly we pretrain the neural network encoder to predict the desired properties on the surrogate data from Panel c).

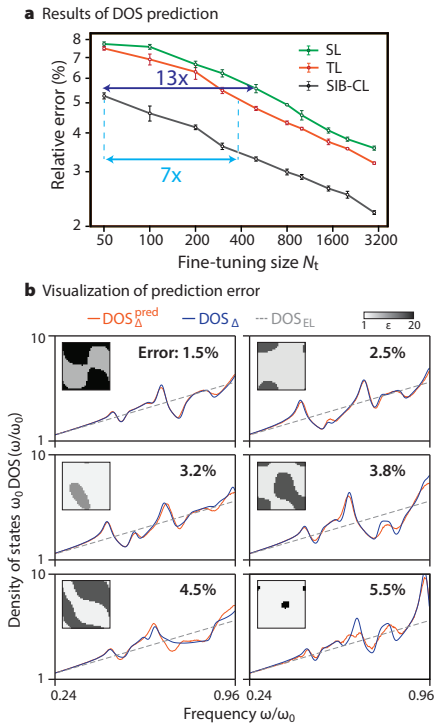

Figure 2-15: **Network prediction results for PhC-DOS problem.** a) Network prediction error against fine-tuning dataset sizes, N_t, between 50 and 3000, when using our SIB-CL framework (based on SimCLR [Chen et al., 2020a] during contrastive learning) compared against the baselines: direct supervised learning (SL) and standard approaches involving transfer learning (TL) or involving invariances via data augmentation (SL-I). A 9-fold (7-fold) reduction in target data requirements is obtained by using SIB-CL over SL (SL-I, TL) at a relative error target of $\sim 5.1\%$. Error bars show the 1σ uncertainty level estimated from varying the data selection of D_t. b) Examples of the DOS spectrum predicted by the SIB-CL-trained network compared against the actual DOS at various error levels (insets depict associated unit cells, shown here using the network-inputs' resolution of 32×32).

Results. We focus our results on photonic crystals, which we studied in Section 4.2.5. In Figure 2-15 we demonstrate the performance gains from SIB-CL. In Panel a) we observe that we can obtain an order of magnitude reduction in the number of datapoints we need in order to match the relative error. In Panel b) we further demonstrate that

the predictions of the SIB-CL network are highly accurate across the various error levels.

Comparison with neural networks that have the inductive bias of invariance. While with SIB-CL we are encouraging the inductive bias of physical invariances during pretraining, there are neural networks that are parameterized to be *always* invariant by design. These are equivariant neural networks [Cohen and Welling, 2016, Bronstein et al., 2021]. Since we are enforcing the inductive bias through a training objective on a training dataset and not through a parameterization, it is natural to hypothesize that equivariant neural networks could generalize better than SIB-CL.

Method	**Time to build/init. model (s)**	**Time for a forward pass(s)**
SIB-CL	0.046	0.0098
E2CNN	56.5	1.33

Table 2.16: Computational times for building the network and performing a single forward pass of the network in SIB-CL vs in E2CNN models of equal number of parameters (approx. 8M parameters).

However, we observe that the generalization of equivariant neural networks comes at a trade-off cost. First of all, in Table 2.16 we compare the time efficiency of SIB-CL with E2CNN [Weiler and Cesa, 2019], a state-of-the-art neural network. We observe that SIB-CL is several orders of magnitude faster. Furthermore, in Figure 2-16 we observe that SIB-CL (orange) can even *improve* upon E2CNN when there are enough datapoints for fine-tuning. These findings indicate that the use of contrastive learning to encourage the property of equivariance is a fruitful area of research.

Outlook. Our study has successfully showcased the employment of contrastive learning to induce an inductive bias that adheres to physical invariances. Nonetheless, our work does harbor a limitation; the labels we utilize are derived synthetically by conducting physics simulations for the DOS values of the photonic crystals. An intriguing prospect arises as to whether the efficacy of contrastive learning would re-

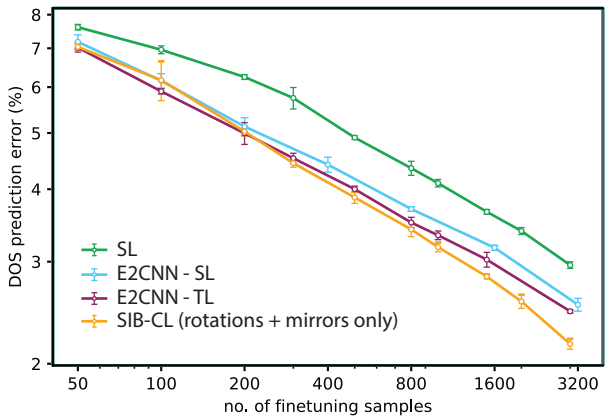

Figure 2-16: Comparison of SIB-CL with an equivariant network E2CNN [Weiler and Cesa, 2019]. E2CNN is trained using supervised learning (E2CNN-SL) as well as fine-tuned after an additional pre-training stage using the surrogate dataset (E2CNN-TL). The supervised baseline (SL) using the non-equivariant architecture that SIB-CL uses is also included for comparison. Error bars show the 1σ uncertainty level when varying the data selection of the fine-tuning dataset.

main observable in a more pragmatic context, where scientific data is not artificially generated, but systematically recorded.

2.7 Contrastive Learning for Stormy Event Imagery

To address the limitation from the previous section, we study a dataset of real scientific recordings and a novel few-shot learning benchmark as follows.

2.7.1 Storm Event Imagery

The Storm Event Imagery (SEVIR) [Veillette et al., 2020] is a radar and satellite meteorology dataset (Figure 2-17). It is a collection of over 10,000 weather events, each of which tracks 5 sensor modalities within 384 km $\times$ 384 km patches for 4 hours. The events are uniformly sampled so that there are 49 frames for each 4 hour period, and the 5 channels consist of: *i)* 1 visible and 2 IR sensors from the GOES-16 advanced baseline [Schmit et al., 2017] *ii)* vertically integrated liquid (VIL) from NEXTRAD *iii)* lightning flashes from GOES-16 . Figure 2-17 shows examples of the two IR and the VIL modalities. We disregard the visible channel because it often contains no

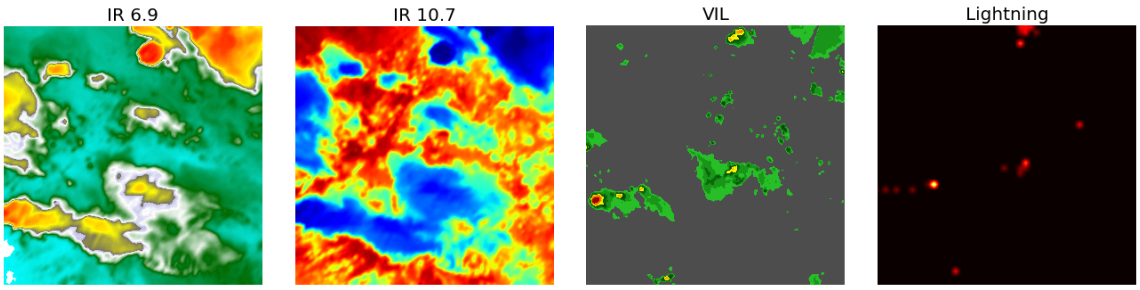

Figure 2-17: **Frame from The Storm Event Imagery (SEVIR) dataset.** We use four of the five available modalities: 2 IR, VIL, and lightning information.

information as visible radiation is easily occluded. Veillette et al. [2020] suggested several machine learning problems that can be studied on SEVIR and provided baselines for two of these: nowcasting and synthetic weather radar generation. In both cases they train U-Net models [Ronneberger et al., 2015] to predict VIL information and experiment with various loss functions.

Evaluation We review common evaluation metrics used in the satellite and radar literature to analyse artificially-generated VIL imagery. They all compare the target and generated samples after binarizing them with an arbitrarily threshold in $[0, 255]$ and look at counts in the associated confusion matrix. Let H denote the number of true positives, M denote the number of false negatives and F the number of false positives. Veillette et al. [2020] define four evaluation metrics: Critical Succes Index **(CSI)** is equivalent to the intersection over union $\frac{H}{H+M+F}$; Probability of detection **(POD)** is equivalent to recall $\frac{H}{H+M}$; Succes Ratio **(SUCR)** is equivalent to precision $\frac{H}{H+F}$.

2.7.2 Benchmark Construction

We leverage the SEVIR [Veillette et al., 2020] dataset to construct a few-shot multi-task image-to-image translation problem where each task corresponds to one event. From the 49 available frames we keep the first N_{support} frames to form the task's support set and the next N_{query} to be the query. Throughout the following experiments

we set $N_{\text{support}} = N_{\text{query}} = 10$.

For the sake of this discussion let's assume we have re-scaled all input modalities to the maximum observed resolution 384×384 so that we can view all of SEVIR as a simple input tensor $\mathcal{D}_1 \in \mathbb{R}^{N_{\text{event}} \times N_{\text{frames}} \times C \times w \times h}$, where: *i)* $N_{\text{event}} = 11479$; *ii)* $N_{\text{frames}} = N_{\text{support}} + N_{\text{query}}$; *iii)* $C = 4$; *iv)* $w = h = 384$. The four input channels are split into three input modalities $C_{\text{in}} = 3$ and one target $C_{\text{out}} = 1$. For joint training we ignore the hierarchical dataset structure and collapse the first two axis $\mathcal{D}_2 \in \mathbb{R}^{N \times C \times w \times h}$, where $N = N_{\text{event}} \times N_{\text{frames}}$ — the total number of frames.

2.7.3 Methods

We solve the aforementioned task using either first-order or second-order gradient descent methods on U-Nets trained using either reconstruction or adversarial objectives. Note that in the case when we train GANs using MAML we are searching for a good initialization for multiple related saddle-point problems. Despite this challenging task, we still obtain good performance.

Below we present the meta-train loop for adversarial networks, which is a novel contribution of our work. For simplicity, we only present the variant with a single SGD inner-loop adaptation step. We train a U-Net generator G with model weights w_G jointly with an extranous patch discriminator D with model weights w_D using data $\mathcal{D} \in \mathbb{R}^{N_{\text{event}} \times N_{\text{frames}} \times C \times w \times h}$. We use batched alternating gradient descent as our optimization algorithm and consider batches $\mathcal{B} \in \mathbb{R}^{B \times N_{\text{frames}} \times C \times w \times h}$, where B is the meta-batch size. Each of these can be split along the second axis into the support and query sets, and along the third axis into the source (S) and target tensors (T) to create $S^{\text{support}} \in \mathbb{R}^{B \times N_{\text{support}} \times C_{\text{in}} \times w \times h}$, $S^{\text{query}} \in \mathbb{R}^{B \times N_{\text{query}} \times C_{\text{in}} \times w \times h}$, $T^{\text{support}} \in \mathbb{R}^{B \times N_{\text{support}} \times C_{\text{out}} \times w \times h}$, $T^{\text{query}} \in \mathbb{R}^{B \times N_{\text{query}} \times C_{\text{out}} \times w \times h}$. For any of these tensors $X \in \{S^{\text{support}}, S^{\text{query}}, T^{\text{support}}, T^{\text{query}}\}$ we refer to the four-dimensional tensor given by the i^{th} task or event as X_i. We use such four-dimensional tensor quantities to evaluate the generator and discriminator loss functions:

$$\hat{\mathcal{L}}_G(t^{\text{generated}}, t, s; w_G, w_D) = -\log D(s, t^{\text{generated}}) + \lambda ||t^{\text{generated}} - t||_1 \tag{2.2}$$

$$\hat{\mathcal{L}}_D(t^{\text{generated}}, t, s; w_G, w_D) = \frac{\log D(s, t^{\text{generated}}) - \log D(s, t)}{2}, \tag{2.3}$$

where $t^{\text{generated}} = G(s)$ is a generated target sample, t and s are corresponding input and output modalities, $||x||_1$ is the mean absolute error. Note the slight abuse of notation where by $\log D(x, y)$ with $x, y \in \mathbb{R}^{N \times C \times w \times h}$ we mean the average $\frac{1}{N}\sum_{i=1}^{N} \log D(x_i, y_i)$. This formulation also uses the trick of replacing $\max \log (1 - D(G(z)))$ with $\min \log D(G(z))$ to obtain a non-saturating generator objective. We wrote the loss functions above such that both players want to minimize their respective objectives.

For each task in a meta-batch size we evaluate the losses above on the support set frames and adapt to this event using SGD to obtain parameters ϕ. We then evaluate the same losses on the task's query set using finetuned models. We repeat these two steps for each event in the meta-batch and perform a second-order gradient update to the initial parameters to optimize the average loss across all events in the meta-batch. The procedure for the Reconstruction loss requires minimal modification. In particular, we modify Equation 2.2 by removing the first term for the discriminator.

2.7.4 Experimental Details

We run experiments using a single 32GB Nvidia Volta V100 GPU. For MAML optimization [Arnold et al., 2020] we use meta-batch sizes of 2, 3 or 4 events. For the corresponding joint training baselines we used $N_{\text{support}} + N_{\text{query}}$ frames from each event and comparable number of events to keep comparisons fair. We randomly split all SEVIR events into 9169 train, 1162 validation, and 1148 test tasks. Joint training baselines and MAML outer loop optimizations are both performed using the Adam optimizer [Kingma and Ba, 2017] with learning rate 0.0002 and momentum 0.5.

We resize input modalities to all have 192×192 resolution and keep the target at 384×384. The generator's encoder has four convolutional blocks, and the decoder has five. All generator blocks except for the last decoder layer use ReLU activation functions. The very last layer uses linear activation functions to support z-score normalization for all four image modalities.

2.7.5 Self-supervised Pre-training

Method

We follow recent work in self-supervised pretraining which applies contrastive learning to convolutional networks before finetuning on classification tasks and improves downstream performance and data efficiency. We ask if these improvements extrapolate to our image-to-image setup. The main distinction between our scenario and those in previous work is that we can initialize only a fraction of our parameters through contrastive pretraining.

We restrict our attention to the U-Net encoder parameters during the pretraining stage and follow the same network architecture as the standard U-Net from the previous section. Our experiments are inspired by the large-scale study on unsupervised spatiotemporal representation learning, conducted by [Feichtenhofer et al., 2021]. In particular, we focus on MoCoV3 [Chen et al., 2021], which is a state-of-the-art contrastive learning method, because [Feichtenhofer et al., 2021] identify the momentum contrast (MoCo) contrastive learning method as the most useful for our data.

Pre-training objective. For a given representation q of a query frame from the dataset, a positive key representation k^+ and a negative key representation k^-, the loss function increases the similarity between the representations within the positive pair (q, k^+) and decreases the similarity within the negative pair (q, k^-) respectively. All representations are normalized on the unit sphere and the similarity is the dot product (i.e. the cosine similarity, because the representations are normalized). The loss is the InfoNCE loss [Oord et al., 2018]:

$$\hat{\mathcal{L}}_q = -\log \frac{e^{p(q)\cdot \mathrm{sg}(k^+)/\tau}}{e^{p(q)\cdot \mathrm{sg}(k^+)/\tau} + \sum_{k^-} e^{p(q)\cdot \mathrm{sg}(k^-)/\tau}} \tag{2.4}$$

for a temperature parameter τ and a predictor MLP p, which is a two layer MLP, with input dimension 128, hidden dimension 2048, output dimension 128, BatchNorm and ReLU in the hidden layer activation, and where "sg" is the stopgradient operation. Following [Chen et al., 2021], the gradients are not backpropagated through $k^{\{+,-\}}$ and

the encoder representations both for keys and queries are obtained after a composition of the backbone and the projector (which is a two layer MLP, with dimensions [256, 2048, 128] with BatchNorm and ReLUs in between the hidden layers, and ending with a BatchNorm with no trainable affine parameters). Additionally, the branch for key representations follows the momentum update policy $\theta_{\text{k}} \leftarrow m\theta_{\text{k}} + (1-m)\theta_{\text{q}}$ from [He et al., 2020] with momentum parameter $m = 0.999$, where θ_{k} are the weights in the key branch and θ_{q} are the weights in the query branch.

Data Augmentations. Another difficulty particular to our setup is the problem of choosing data augmentations the input domain is invariant to because weather modalities have different invariances than natural images: for example, the popular color jitter transformation is not applicable here, because image-to-image translation is sensitive to color. From the standard augmentations, the ones we consider are: random resized crops, random horizontal flips, gaussian noise, gaussian blur, random vertical flips and random rotation. We also further exploit the temporal structure of SEVIR to obtain "natural" augmentations, which we introduce next.

Natural augmentations. We further consider using the temporal structure of SEVIR for augmentations, as follows. Each event consists of 49 frames, so we anchor every even frame as query frame and use every odd frame as key frame. For each query frame, to obtain q and k^+ we apply the following stochastic transformations to the frame twice: random resized crops using scale (0.8, 1.0); random horizontal flips with probability 0.5, pixel-wise gaussian noise sampled from the normal distribution $\mathcal{N}(0, 0.1)$ with probability 0.5, gaussian blur with kernel size 19, random vertical flips with probability 0.2, random rotation by angle unformly chosen in $(-\pi/6, \pi/6)$. The rest of the augmentation arguments follow the default in the Torchvision library[4]. In Figure 2-18 we present a conceptual visualization of the transforms. To obtain k^- we apply the above stochastic transformations to the corresponding key frame once.

Training hyperparameters. Our experiments use the following architectural choices: mini-batch, consisting of 3 events with 24 frames for queries and key 24 frames for keys each; 0.015 base learning rate; 100 pre-training epochs; standard cosine

[4] `https://pytorch.org/vision/stable/transforms.html`

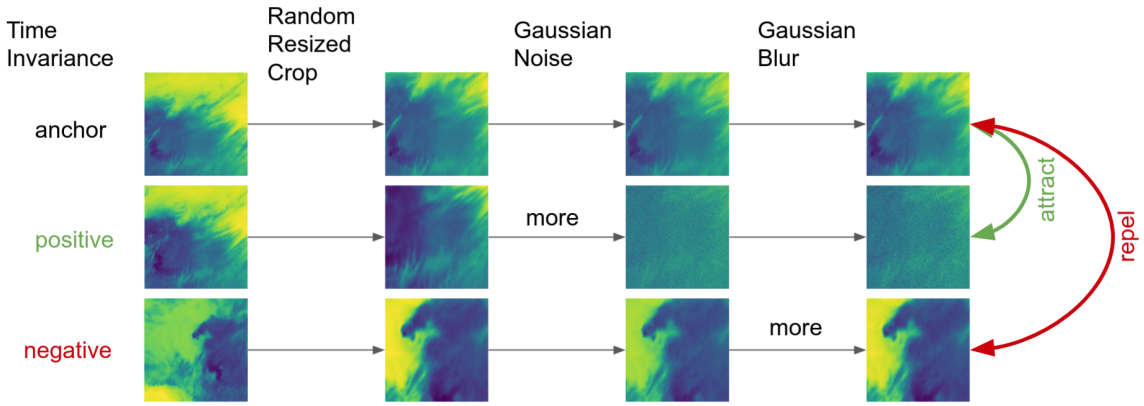

Figure 2-18: **Augmentations for the contrastive learning experiment** By indicating "more" we show examples of a larger magnitude of the augmentation being applied.

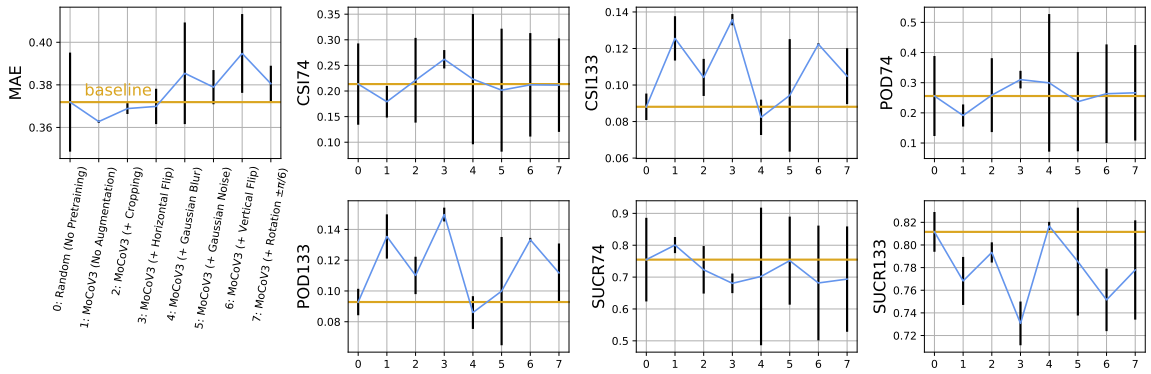

Figure 2-19: **Contrastive Learning for SEVIR.** For mean absolute error lower is better. For every other evaluation measure, higher is better.

decayed learning rate; 5 epochs for the linear warmup; 0.0005 weight decay value; SGD with momentum 0.9 optimizer. We report the joint training reconstruction loss experimental setup by finetuning the checkpoint obtained from pretraining.

Results

In Figure 2-19 we report our results. Firstly, for mean absolute error we find marginal yet somewhat consistent gains up to level 3 augmentation. Secondly, we also evaluate on meteorological metrics. We find that even though pretraining has a marginal effect on the reconstruction loss train objective, it often provides important gains on

Figure 2-20: **Pretrained encoder - generated samples** Pretrained models better identify the sparse high VIL values.

domain-specific evaluation criteria. We highlight the large improvement in CSI133 and POD133, which stems mostly from significant improvements in precision. We observe that up to level 4 MoCoV3 augmentation we obtain improvements throughout all measures with the contrastive pretraining. Finally, we show example samples in figure 2-20 and find that pretraining the U-Net encoder leads to better performance in high-VIL regions.

2.8 Conclusion

Chronologically, in the discussion above we explored a contribution that generalizes the symmetries used in self-supervised learning [Dangovski et al., 2021a] for computer vision, efficiently combined such symmetries in a single classifier with solutions to multiple hypotheses [Loh et al., 2023], enabled synthetically meaningful symmetries for NLP [Chuang et al., 2022], provided an application to scarce data in science [Loh et al., 2022a], as well as an application to severe weather events from real-world recordings [Rugina et al., 2021].

Several of our contributions are related to the importance of symmetry in reducing the need for data, which we have not discussed above.

Firstly, we show that symmetry is useful for improving the sampling complexity of optimization algorithms. Namely, in [Liao et al., 2022] we develop an optimizer

that learns during optimization and is parameterized through efficient unitary neural networks [Jing et al., 2017]. Furthermore, we speed up quantum optimization and machine learning through Koopman's operator learning theory [Luo et al., 2022].

Secondly, we show that architectural modifications can boost symmetry and reduce the need for data for learning representations. In [Dangovski et al., 2019a] we develop a phase-coded norm-preserving memory mechanism that enables better associative memory and recall for NLP tasks. In [Jing et al., 2018] we use logarithmic asymmetry to improve efficient models for computer vision. In [Loh et al., 2022b] we develop a Bayesian approach to reducing confirmation bias during semi-supervised learning. In [Dugan et al., 2023] we reformulate the quantum evolution of complex-valued quantum mechanics to the evolution of real-valued probability distributions to speed up simulations of open quantum mechanics by orders of magnitude. We also explore how using generative models can improve the quality of contrastive learning in computer vision [Han et al., 2023].

Finally, a promising direction for future research is to view representation learning as invariant to the modalities that correspond to a given datapoint object. Throughout the Internet and Science there is a natural correspondence between a variety of modalities, such as images, captions, videos and waveforms in the Internet, and varieties of scientific measurement for a single scientific phenomenon. It is only natural to learn representations with the objective to maintain invariance across modalities. We saw in the above discussion that representation learning in one domain, say vision, can yield useful methods for other domains, such as NLP and materials science. Thus, we present promising directions for multimodal learning in Chapter 5.

Chapter 3

Addressing the Ability to Transfer with the Language Inductive Bias

3.1 Introduction

Language is a very efficient means of communication. With only very few bits of information, we can transfer knowledge through the communication channel of language. For example, we could be observing a less popular animal that could be found on safari, such as a *pangolin*[1]. However, we could describe the animal to anyone around the world in an efficient way, using natural language, and detailing the unique features of pangolins. The person on the other side of the phone would be able to provide a meaningful reconstruction of a pangolin. In linguistics, Chomsky's theory of *universal grammar* posits that there is a fundamental inductive bias of language. However, in machine learning, communication similar to language emerges from similar interactions between AI agents in the safari environment above [Evtimova et al., 2017].

An example of the role of language in scientific discourse is the contrast between the development of mathematical methodologies in Greek and Chinese mathematics [Crossley et al., 1999].

Greek mathematical language is formal and symbolic, which is instrumental in

[1] A pangolin is a unique mammal known for its protective keratin scales, which cover its body.

the development of abstract and deductive mathematical reasoning. Mathematical concepts are often presented in the form of definitions, postulates, and proofs, representing a language of logic and precision. This axiomatic approach enables Greek mathematics to form and solve problems irrespective of their real-world application. This linguistic method provides a strong basis for the rigorous logical framework required in disciplines such as geometry and calculus. The Greek emphasis on logical completeness is also manifested in their mathematical language, with the Pythagorean theorem or Euclid's *Elements* serving as examples of a detailed logical construct.

However, Chinese mathematics exhibits a more descriptive and concrete language, emphasizing practical problem solving. This is reflected in the structure and style of mathematical problems in key texts such as *The Nine Chapters on the Mathematical Art*, which frequently present mathematical problems embedded in real-world scenarios. The language of Chinese mathematics tends to be more intuitive and less formal, favoring direct computational procedures and empirical approaches over abstract proofs. This allows for a close alignment of mathematics with practical applications, leading to the development of innovative methods in areas like algebra and arithmetic. Furthermore, the inherent flexibility of the Chinese language, with its reliance on context, aligns well with this approach, lending itself to multiple interpretations and thereby encouraging an empirical approach to mathematical problem solving.

In the context of machine learning, understanding these linguistic differences could shed light on the interaction between the formal symbolic language used in algorithm design and the more descriptive context-dependent language used in problem definition and data interpretation. The balance between these two modes of thought: logical abstraction and empirical problem solving can be crucial in designing and refining machine learning systems.

How can we efficiently use the inductive bias of language for representation learning? The power of *merging* concepts formed by language in self-similarity patterns is arguably a way to create an infinite number of meaningful constructs [Adger, 2019]. Furthermore, we can communicate the "infinity" of constructs in a very efficient man-

ner, using language, i.e. just a few bits of information.

Looking at some of the most influential works in representation learning, we can find connections to the power of merging language concepts and the transferability of the representations learned in that way. Some examples include:

- Using language as weak supervision for image representation learning in the CLIP model [Radford et al., 2021]. The representations of the images have been used to construct state-of-the-art image generative models [Ramesh et al., 2022].

- Using language to describe a wide range of tasks and benefit from language modeling as a pre-training strategy of language representations [Radford et al., 2018, Brown et al., 2020, Chowdhery et al., 2022, OpenAI, 2023, Anil et al., 2023]. The hypothesis in such works is that langugage is a natural representation of abstract internal representations that "understand about the world."

In all these works, formulating the training task of the neural network representation learners with language has been instrumental. In the next section, we use that inspiration to show how we can improve transfer learning by formulating a variety of representation learning tasks with proper constructions in natural language. The task of "automating science journalism" is a fruitful bed to study such transfer learning techniques, as we demonstrate next.

3.2 Towards Automating Science Journalism at Scale

3.2.1 Introduction

Recent years have been characterized by rapid growth of published scientific research. Coping with this quantity is increasingly challenging, which has led to the emergence of a number of initiatives, including software applications that try to summarize and to organize research articles. For example, *Scholarcy* helps researchers and students by summarizing relevant portions of academic papers. Likewise, *Mendeley* establishes

meaningful links between research papers. Furthermore, there are emerging tools, such as *Litmaps* that place scientific research in a broader perspective, thus making it accessible to layman readers.

Generated: it 's no secret that women are as good as men . but when it comes to job satisfaction , a new study shows that this gender equality can affect one 's own job and make the impression that women experience higher levels of gender equity among women .
Target: male workers appear to support women becoming ceos even more than female workers do , finds new research on the proverbial glass ceiling and job satisfaction in six formerly socialist countries .
Source snippets: ... moreover , recent data show that , in spite of significant barriers , more women reach the upper managerial ranks in the workplace ... does gender equality in workplace promotion opportunities have consequences for job satisfaction ? we address this question by examining the link between job satisfaction and perceived prospects for women to become top manager at the firm .

Table 3.1: Summary from our dataset (*short Science Daily) using our model (*SciBertSumAbs*).* We see the need for extreme paraphrasing and coherent generation.

Traditionally, this was the task of science journalism, led by media outlets such as *Science Daily*, *Scientific American*, and *Popular Science*, which establish some of the few direct connections between scientific research and the general public. As demonstrated in Table 3.1, this is a tremendously difficult task: it requires writing factual summaries, while also paraphrasing complex scientific concepts using a language that is accessible to the general public.

We argue that the abundance of science journalism articles enables a variety of learning approaches, most notably neural text summarization [Rush et al., 2015]. The latter has undergone strong evolution recently [Lin and Ng, 2019]: from extractive [Nallapati et al., 2017] through abstractive [Nallapati et al., 2016] to hybrid [See et al., 2017] models; from maximum likelihood to reinforcement learning objectives [Celikyilmaz et al., 2018, Chen and Bansal, 2018]; from small to large datasets [Grusky

et al., 2018], which are also abstractive [Sharma et al., 2019]; from short to orders of magnitude longer sources and targets [Liu et al., 2018b]; from models trained from scratch to using pre-trained representations [Edunov et al., 2019, Liu and Lapata, 2019].

From a modelling perspective, these advances are yet to be challenged with an abstractive summarization task (*i*) from *long source* research articles into *long targets*, and (*ii*) using *extreme paraphrasing*. Here, we argue that automating science journalism is a natural testbed for this.

The task is defined as follows: *Given a scientific article, produce a layman summary of that article.*

Our contributions can be summarized as follows:

- We introduce a text summarization task: generate a layman's terms summary of a research article in the form of a press release.
- We create a specialized dataset for the task and we experiment with a number of models.
- We focus on story generation as a way to model press releases, and we propose suitable data augmentation methods, which we validate extensively.

3.2.2 Related Work

Summarization of Scientific Documents Abu-Jbara and Radev [2011] produced readable and coherent citation-based summaries improving upon a history of related work [Nanba et al., 2000, Nakov et al., 2004, Elkiss et al., 2008, Qazvinian and Radev, 2008, Mei and Zhai, 2008, Mohammad et al., 2009, Divoli et al., 2012]. Collins et al. [2017] studied extractive summarization of scientific papers to highlights, following a history of predominantly extractive summarization of scientific documents [Kupiec et al., 1995, Saggion et al., 2016]. Yasunaga et al. [2019] proposed hybrid summarization of well-annotated datasets, thus extending work by [Jaidka et al., 2016, 2017, 2018]. Beltagy et al. [2019] fine-tuned BERT on scientific articles and improved the baselines for some downstream scientific tasks. Subramanian et al. [2020]

performed summarization of very long documents, but did not address the task of extreme paraphrasing, nor did they use a seq2seq architecture. Luu et al. [2020] explained the relationship between two scientific documents via citations. Finally, recent advances in efficient Transformers [Beltagy et al., 2020] made it possible to process long scientific documents efficiently, and thus scale ASJ. Recent proof-of-concept work has approached automating scientific reviewing [Wang et al., 2020, Yuan et al., 2021]. Furthermore, workshops, such as CL-SciSumm/ CL-LaySumm Chandrasekaran et al. [2019] and LongSumm Chandrasekaran et al. [2020] have offered opportunities for developing summarization of scientific documents.

Unlike the above work, we use orders of magnitude larger datasets with diverse content domains, and we generate meaningful abstractive summaries in layman's terms. To our knowledge, we are the first to explore scaling automating science journalism through summarization of long sources, which require extreme paraphrasing and long generation.

Scientific Datasets Dangovski et al. [2019b] presented pioneering results on the *Science Daily* dataset using a seq2seq model with novel RNN units, based on rotation. However, their work was limited to short source and target pairs. Moreover, they performed summarization from a *journalistic* article in Science Daily article to the highlight of that article, again in Science Daily.

In contrast, we perform summarization from a *research journal* article to Science Daily highlights. This is an important distinction, as research articles use very different style, language, and terminology compared to journalistic articles.

Other work preserved the style of the source [Teufel and Moens, 2002, Nikolov et al., 2018, Cohan et al., 2018] or generated very short targets taking the form of blog titles [Vadapalli et al., 2018]. Sharma et al. [2019] introduced BigPatent as a new challenge for abstractive summarization, which is a good parallel to our task, as it still summarizes scientific content in an abstractive manner. Lev et al. [2019] proposed a dataset, TalkSumm, for generating summaries using conference talks. Recently, Cachola et al. [2020] introduced SciTldr for extreme summarization of scientific papers

in Computer Science. Gidiotis and Tsoumakas [2020] used the RNN units from [Dangovski et al., 2019b] and a divide-and-conquer approach to improve summarization of ArXiv and PubMed [Cohan et al., 2018] articles to abstracts. However, none of the above work addressed our task of producing a press release for a research article in layman's terms.

Data-Augmentation and Multitask Learning for Language Generation Our task and the corresponding datasets made it possible to use recent advances in transfer learning for NLP [Raffel et al., 2020, Ruder, 2019]. Namely, we combine datasets sharing a source domain, i.e., scientific articles, with different target domains, i.e., abstracts and press releases. We take inspiration from recent work on automatically generating news articles [Zellers et al., 2019], trained on multiple variations of the same dataset, e.g., in some instances, the headline might be used to generate the body, while in other, the body can be used to generate the headline. Similarly, via a special tag, we can signal to the decoder to generate either an abstract or a press release, or to generate the target in several steps by conditioning on intermediate outputs. Other ways to signal to the decoder were proposed in the context of summarization with user preferences [Fan et al., 2018a], neural machine translation [Lample and Conneau, 2019, Aharoni et al., 2019], and controllable text generation [Keskar et al., 2019] that contain tags, similarly to pre-training contextual word embeddings [Peters et al., 2018a, Delvin et al., 2019]. Finally, we should mention multitask learning Raffel et al. [2020], Lewis et al. [2019], Cachola et al. [2020] for improving summarization.

3.2.3 The *Science Daily* Dataset

Dataset

We introduce two versions of *Science Daily*: (*i*) for long summarization, consisting of pairs of full-text scientific papers and their corresponding *Science Daily* press releases, and (*ii*) for short summarization, made of pairs of scientific papers cut after the first 400 words and corresponding short highlights in the press releases. Even though

Science Daily	short	long
# pairs	50,308	50,134
# source words	400 ± 0	5,975 ± 2,731
# target words	45 ± 19	488 ± 219
train/ dev/ test	90%/5%/5%	80%/10%/10%

Table 3.2: Statistics about the *Science Daily* datasets.

Rank	Journal	# dataset entries
1	PNAS	5,482
2	Science	4,006
14	Nature Geoscience	472
15	Nature Medicine	425
16	Nature Neuroscience	397
17	Nature Climate Change	396

Table 3.3: *Science Daily* covers diverse journals.

in this paper we put emphasis on long summarization, the *short Science Daily* is a task that is closer, in terms of length of sources and targets, to the one considered in [Dangovski et al., 2019b].

Moreover, they both contain another challenging aspect, that is the difference in the style of language between the source and the target. See Table 3.2 for some statistics.

Note that the number of pairs in these datasets do not match, as not all *Science Daily* articles had highlights. The training split for *long Science Daily* is lower by 10% since its pairs contain more tokens than their counterparts in the short dataset. Below, we explain how we created our datasets.

Note also that our *Science Daily* dataset differs from existing datasets for summarization of scientific content as it is extremely diverse and covers a wide range of scientific fields, as shown in Table 3.3, and as it features a drastic change in style between the source and the target.

Press Releases. *Science Daily*[2] is a website that aggregates and publishes lightly edited press releases about science. We were granted access to download about 100,000 HTML pages from their website, each containing a public story about a recent research paper. From each HTML page, we extracted the main content, a short highlight, and a title.

Scientific Articles. We further parsed each HTML page of the press releases to obtain information about the target scientific article: title, short description, main content and DOI. Then, we sent the DOI to the *Crossref API*[3] to obtain the meta information about the target paper. We downloaded the papers as PDF files, and we then converted them to raw text. These papers span a large range of publishers including *American Association for the Advancement of Science* (AAAS), *Elsevier*, *Public Library of Science* (PLOS), *Proceedings of the National Academy of Sciences* (PNAS), *Springer* and *Wiley*. We ignored publishers with fewer than 100 papers.

There are many such publishers and the style of their PDFs is not consistent; hence, we opted to convert to text and to use scientific papers from the most prevalent publishers only.

Figure 3-1 shows statistics about the publishers. The figure gives a peek into the differences in style among the publishers. For example, AAAS publishes short letters, while PNAS publishes longer articles. We treat articles with fewer then 1,000 words as outliers, and we do not include them in the dataset.

Analysis: Comparison to Related Datasets

Compared to other datasets, *Science Daily* summaries are significantly more *abstractive*. To see this, we compare to the *ArXiv* dataset, which summarizes scientific articles to their abstracts. We use two statistics from [Grusky et al., 2018]:

$$\text{coverage}(A, S) = (1/|S|) \sum_{f \in \mathcal{F}(A,S)} |f|$$

and

$$\text{density}(A, S) = (1/|S|) \sum_{f \in \mathcal{F}(A,S)} |f|^2$$

[2] `http://www.sciencedaily.com/`

[3] `https://www.crossref.org/`

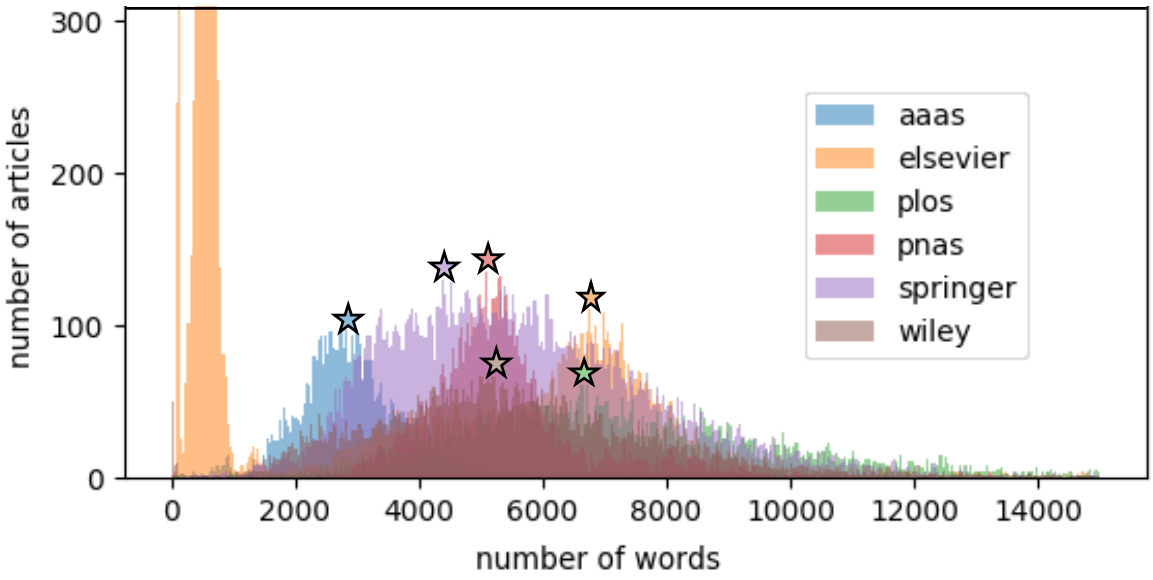

Figure 3-1: Histogram for number of articles vs. number of words for the selected publishers in *Science Daily*. Stars indicate modes of the histograms (excluding the outliers with fewer than 1,000 words for Elsevier).

where $\mathcal{F}(A, S)$ is the set of extractive fragments, a sequence of words that is shared between the source and the target for a set of articles $\{A\}$ and a corresponding set of summaries $\{S\}$, $|f|$ is the number of words in fragment f, and $|S|$ is the number of words in summary S.

The *coverage* represents the fraction of words in an extractive fragment, and the *density* is the average length of these fragments. Figure 3-2 compares *Science Daily* to established datasets. We can see that the coverage is around 0.4 for *Science Daily* vs. 0.8 for *ArXiv*. Moreover, while the density for *Science Daily* is on the order of a few absolute density points, it is in the hundreds for *ArXiv*.

Another important characteristic of our *Science Daily* dataset is that both the source and the target are relatively long, with source articles and target press releases containing about 6,000 and 500 word tokens, respectively. For comparison, the *CNN/ Daily Mail* dataset is much shorter, with sources of 800 word tokens and targets of just 50 word tokens, and even the *ArXiv* dataset has substantially shorter targets of around 200 word tokens.

We further computed standard measures of language complexity such as SMOG,

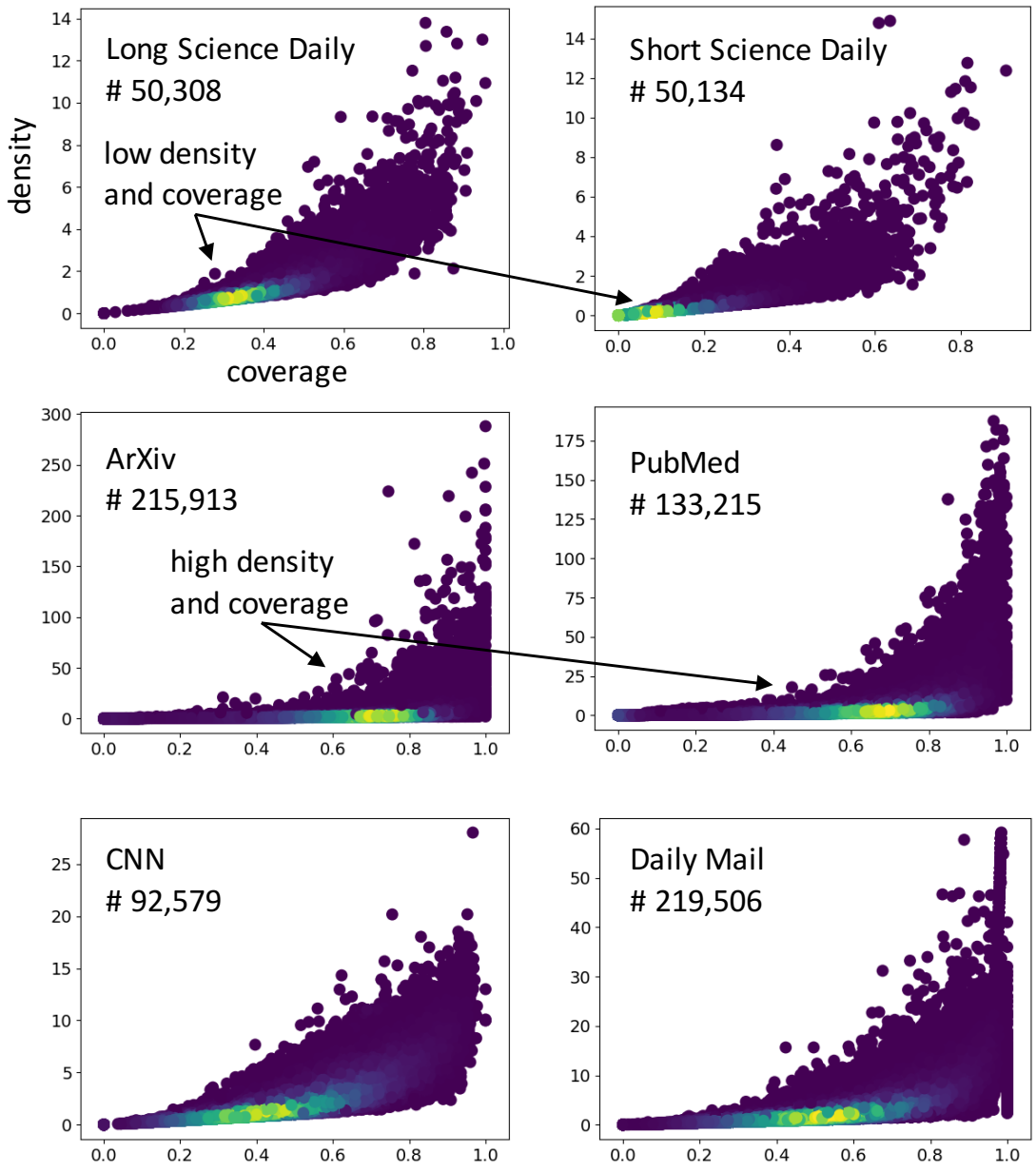

Figure 3-2: Density vs. coverage of source-target pairs for *Science Daily*, *ArXiv*, *PubMed*, and *CNN and Daily Mail*. Warmer colors show more data entries, and # is number of pairs. Outliers with extreme densities are omitted. Arrows indicate the modes of the datasets.

CLI, and LIX, as implemented in the NELA toolkit [Horne et al., 2018]. The results are shown in Table 3.4, where we can see that the texts from scientific sources use more complex language.

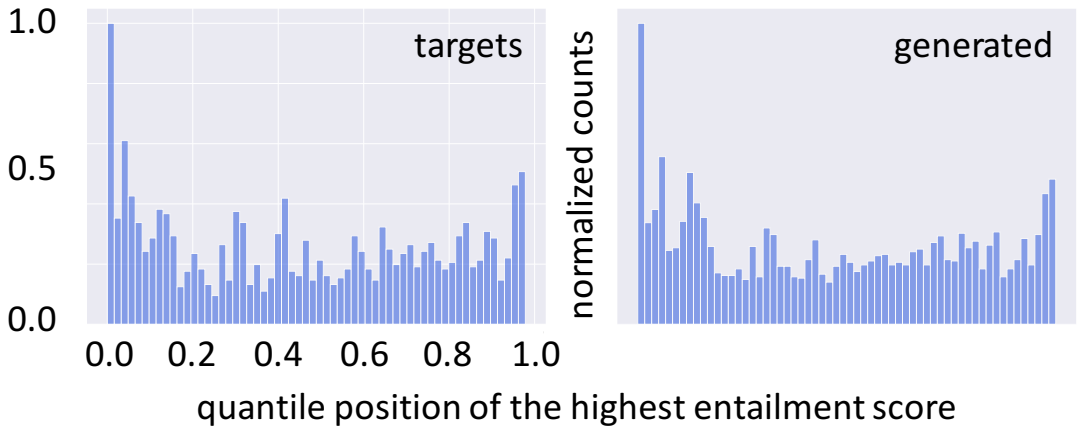

Figure 3-3: Positions of the source sentences that maximize the NLI entailment of the summary sentences for *Science Daily*. On the left are gold summaries, and on the right are summaries by our model (*Story+Parts*). The counts are normalized, so that the bin with the highest counts is at 1.0.

Dataset	SMOG	CLI	LIX
Science Daily	15.23 ± 1.51	14.34 ± 1.21	55.60 ± 4.66
PubMed	16.98 ± 1.65	14.21 ± 1.67	59.00 ± 6.73
ArXiv	13.74 ± 1.53	12.09 ± 1.64	50.26 ± 6.19
CNN	12.01 ± 1.67	10.66 ± 1.87	45.31 ± 8.20
Daily Mail	12.29 ± 1.61	10.35 ± 1.50	49.01 ± 7.80

Table 3.4: Complexity of related datasets' sources based on readability scores such as SMOG, CLI, and LIX. The datasets from scientific sources (the top half) use more complex language (bigger numbers indicate higher complexity).

We further used natural language inference (NLI) to explore which parts of the source text contain the most relevant information for summarizing *Science Daily* research articles. For each sentence from the target summary, we found a corresponding one in the source text that entailed it with the highest probability, and we marked the relative position of that sentence in the source text.

We repeated the procedure for all summaries, and we generated aggregated statistics about the relative positions of these source sentences (in bins), as shown in Figure 3-3. We can see on the left side of the figure that the gold journalistic summaries use information not only from the introduction and from the conclusion of the input

research articles, but also from the entire input text. On the right side of the figure, we show a similar analysis for summaries generated by our model: we can see a similar pattern, (albeit different from the left histogram, the right histogram spreads throughout its entire range too), which means that the model learns to look at the entire input when generating a summary.

3.2.4 Evaluation

ROUGE. We use the standard ROUGE 1/2/L scores [Lin and Hovy, 2003].

Natural Language Inference (NLI). Ideally, each summary should be fully entailed from the source text. With this in mind, Falke et al. [2019] proposed an evaluation measure for text summarization that uses NLI and tries to find for each sentence in the summary the maximal probability of it being entailed from some sentence in the source text. The final score is calculated as the average of these probabilities:

$$\sigma(S) = \frac{1}{n} \sum_{j=1}^{n} \max_{d \in D} N(d, s_j) \tag{3.1}$$

where $N(d, s_j)$ is the probability that sentence s_j from the summary S is entailed from sentence d in the source document D, and n is the number of sentences in the summary.

This approach resembles the NLI analysis method we used above, but here the focus is on the score, while above we were interested in the relative position of the best-matching source sentence.

Prompt Ranking (PR). For *long Science Daily*, we further used an evaluation measure, inspired by the *prompt ranking measure* from [Fan et al., 2018b]. For a target in the dataset, it takes the source and nine additional sources of different targets. Then, it tests whether the generator assigns higher probability to the target when conditioned on the correct source (by feeding the source into the encoder) compared to conditioning on the incorrect sources, and measures the success rate of

that test on a selected number of targets from the dataset.

Here, we follow the same procedure, but with the important modification that instead of taking the full source, we select a random substring of 100 words to feed into the encoder. To provide results in the context of existing work on prompt ranking, our aim is to mirror the original prompt ranking measure, which was used to rank the prompts (short prompts, such as a title of a movie) based on the probability that the true story (long generation) has, when conditioned on the prompts. In the *long Science Daily*, the press releases are similar to the stories in [Fan et al., 2018b], but the sources (the scientific papers) are not similar to the prompts. Hence, we take 100-word random substrings to form prompts for the press releases. We calculate the prompt ranking score on a hold-out set of 1,000 long *Science Daily* pairs, and we report its value in percentage points.

3.2.5 Experiments

Given that the size of the Science Daily dataset is not that large compared to existing summarization corpora, our task should benefit from using pre-trained models or from augmenting the data. Below, we present experiments that demonstrate techniques in both directions, which lay the foundations for our task.

Summarization with Pre-trained BERT

We begin by exploring familiar ground: short summarization using the *short Science Daily* (Table 3.2) à la CNN/ Daily Mail [See et al., 2017], i.e., our sources are up to 512 tokens long, and the targets are up to 140 tokens long. We choose an abstractive seq2seq model, following a strong neural summarization baseline with pre-trained BERT [Liu and Lapata, 2019]. In particular, we experiment with their BertSumAbs, which uses a pre-trained BERT model as an encoder and a Transformer [Vaswani et al., 2017a] trained from scratch as a decoder. We denote this experiment with *BertSumAbs* as well.

Model	1	2	L
LEAD	19.7	3.7	13.1
BertSumAbs	27.16	4.54	21.45
SciBertSumAbs	**30.30**	**6.24**	**24.00**

Table 3.5: *Short Science Daily*: SciBERT pre-training improves over vanilla BERT (ROUGE scores in %). LEAD takes the first 45 words from the input.

Scientific Pre-training Since we are in the scientific domain, we replace the BERT [Delvin et al., 2019] encoder with SciBERT Beltagy et al. [2019], which is fine-tuned on scientific papers, and we dub the resulting model *SciBertSumAbs*. We train the model for 200K steps. The hyper-parameter values coincide with those for *BertSumAbs*.

In Table 3.5, we show how *BertSumAbs* and *SciBertSumAbs* compare in terms of ROUGE 1/2/L scores using beam search decoding and trigram blocking [Paulus et al., 2018], thereby following the decoding setup in [Liu and Lapata, 2019], but limiting the generation to 50–200 tokens. We observe sizable gains from using SciBERT. This result is expected since *Science Daily* focuses on the scientific articles (Table 3.3).

In the following experiments, we focus on the *long Science Daily* (Table 3.2) dataset.

Efficiency with CNN seq2seq

For the *long Science Daily*, we use CNN-based seq2seq architectures, which can handle long input. We start with a small vanilla convolutional seq2seq model [Gehring et al., 2017], corresponding to fairseq's ISWLT German–English (de-en) model [Ott et al., 2019], which we take directly from the library.

We train the model until convergence on the dev set with a learning rate of 0.25, Nesterov accelerated gradient (NAG) descent, 0.2 dropout, and a 0.1 gradient threshold. We name this experiment *Fconv*.

ASJ as Story Generation

We can frame ASJ as story generation, since a press release can be viewed as a story shaped around a scientific paper. The scientific paper itself can be viewed as a "writing

Model	1	2	L	NLI	PR
LEAD	39.6	10.1	16.1	N/A	N/A
Fconv	39.2	9.5	36.9	0.23	38.0
FconvTopK	39.2	**10.8**	**37.0**	0.23	38.0
Story	38.9	7.8	36.4	0.12	22.7
StoryTopK	38.2	**8.5**	36.0	**0.14**	22.7

Table 3.6: *Long Science Daily*: baselines. *Fconv* outperforms *Story* in ROUGE 1/2/L and Prompt Ranking (PR); top-k sampling generally helps for *Fconv*. PR does not depend on the decoding scheme. LEAD takes the first 488 input words.

prompt" for the story. Hence, our second model is a modification of a state-of-the-art model for neural story generation [Fan et al., 2018b, 2019]. It introduces attention [Bahdanau et al., 2015] between the output of the encoder and the decoder layers, as well as multi-head self-attention on the decoder layers [Vaswani et al., 2017a] that is gated [Dauphin et al., 2017] and equipped with a multi-scale mechanism for down-sampling [Fan et al., 2018b]. Since our sources are three orders of magnitude larger than the writing prompt sources for which the original story model has been used, we *additionally equip the encoders* with gated multi-scale multi-head self-attention. Thus, we extend the fairseq implementation with additional four-gated self-attention heads both on the encoders and on the decoders with projected inputs and down-sampling. We train the model until convergence on Dev with a learning rate of 0.25, NAG, dropout of 0.2, and a gradient threshold of 1.0. We call this experiment *Story*.

Training for all our fairseq models takes about 20-30 epochs depending on the batch size, which is around 30-40. In preprocessing, we only keep words that appear at least ten times in the source, or at least ten times in the target. Moreover, for these models we converted all textual data to *byte pair encoding* (BPE) [Sennrich et al., 2016] with 32,000 BPE tokens both on the source and on the target side following the guidelines for fairseq.

Table 3.6 shows a comparison between *Fconv* and *Story*. Surprisingly, the simple *Fconv* baseline outperforms the *Story* model both on ROUGE scores and Prompt Ranking. We speculate that this might be due to *Fconv* being more extractive, which

Model	1	2	L	NLI	PR
LEAD	39.6	10.1	16.1	N/A	N/A
Fconv	39.2	9.5	36.9	0.23	38.0
Fconv+ArXiv	**41.2**	**10.2**	**38.6**	**0.28**	**77.8**
FconvTopK	39.2	10.8	37.0	0.23	38.0
FconvTopK+ArXiv	**41.8**	**11.6**	**38.6**	**0.25**	**77.8**
Story	38.9	7.8	36.4	0.12	22.7
Story+ArXiv	**41.0**	**9.2**	**38.6**	**0.15**	**64.1**
StoryTopK	38.2	8.5	36.0	0.13	22.7
StoryTopK+ArXiv	**41.4**	**10.6**	**38.8**	**0.14**	**64.1**

Table 3.7: *Long Science Daily*: Training with *ArXiv*. We can observe sizeable and consistent improvements.

might influence the scores marginally.

I.e., the model might optimize for generating words that overlap between the source paper and the target, e.g., by copying scientific terms. Thus, high ROUGE scores do not necessarily imply a good story Fan et al. [2018b], and we will proceed with both models as baselines.

Moreover, sampling from the top-k candidates ($k = 10$) has been shown useful for story generation [Fan et al., 2018b], and we try it here as well. We label such experiments by appending *TopK*; Table 3.6 shows that top-k decoding yields sizable improvements for ROUGE-2.

Data Augmentation with ArXiv

As summarization in *Arxiv* to generate abstracts and our ASJ task share similar domains for their sources, namely scientific papers, it is natural to try to augment our *Science Daily* dataset with the *ArXiv* dataset. We do so using specially designed tags: (*i*) we prepend the tag **[begin-paper]** and we append the tags **[end-paper] [begin-press]** for *Science Daily*.

For *ArXiv* examples, we do the same, but we replace **press** with **abstract**. (*ii*) We also append the target with **[end-press]** or **[end-abstract]**, respectively. These tags indicate the source domain (*ArXiv* or *Science Daily*) and the target domain (*abstract*

Model	1	2	L	NLI	PR
LEAD	39.6	10.1	16.1	N/A	N/A
Fconv	**39.2**	**9.5**	**36.9**	0.23	38.0
Fconv+Parts	32.8	7.8	31.2	**0.25**	**77.1**
FconvTopK	**39.2**	**10.8**	**37.0**	0.23	38.0
FconvTopK+Parts	31.1	9.0	29.6	**0.27**	**77.1**
Story	38.9	7.8	36.4	0.12	22.7
Story+Parts	**42.8**	**10.6**	**40.2**	**0.17**	**73.8**
StoryTopK	38.2	8.5	36.0	0.14	22.7
StoryTopK+Parts	**41.4**	**11.0**	**39.1**	**0.16**	**73.8**

Table 3.8: Training in parts yields improvements: sizable for Prompt Ranking, but partial for ROUGE 1/2/L.

or *press release*). To ensure equal balance between the two datasets, we take 40,000 examples from their training sets, 5,000 from their test, and 5,000 from their dev set, for a final train/dev/test split of 80,000/10,000/10,000.

We hypothesize that the encoder layers and the decoder attention mechanism will focus on these tags while processing the source and while generating the output, respectively. Table 3.7 shows that using *ArXiv* yields sizable improvements both for ROUGE 1/2/L and for our Prompt Ranking score. Note that we did not use ArXiv source-target paris for generation and calculation of ROUGE, NLI and PR. We only used the originally designated *Science Daily* test source-target pairs (even though the model has been trained using *ArXiv* source-target pairs too). We believe that the ability to co-train with other datasets offers important flexibility in our experimental setup.

Data Augmentation with Targets in Parts

In order to increase the total number of training examples and to focus the summarization on particular parts, we experimented with augmenting *Science Daily* with partitioned targets as follows:

1. For each source–target pair in *Science Daily*, we preserve the source **body**, and we divide the target into three equal parts: **part-1**, **part-2**, and **part-3**.

2. We construct the source-target pairs as follows: for all bodies **body**, for indices **i** equal to 2 or 3, the source is

   [begin-body]body[end-body][begin-part-(i-1)]

   part-(i-1)[end-part-(i-1)][begin-part-i]

   and for **i** equal to 1, the source is

   [begin-body]body[end-body][begin-part-i],

   where the corresponding target to the source is **part-i [end-part-i]**.

3. At inference, we generate the parts **part-i** autoregressively from **part-1** to **part-3**.

Instead of training the model to generate the full press release, we train it to generate specific sections only. Thus, we increase the data split threefold, which yields a train/dev/test split of size 120,741/15,087/15,087. Recently, similar divide-and-conquer approaches have improved the state of the art on scientific summarization [Gidiotis and Tsoumakas, 2020]. Table 3.8 shows results when using this partition.

Note that to compute ROUGE, NLI, and PR, we generate each designated part, concatenate the generations, and then we calculate the scores. We can see in Table 3.8 sizable improvements over the baselines for the in-parts training method, both for ROUGE 1/2/L and for PR, which confirms that this data augmentation scheme is indeed helpful.

NLI Scores. We computed the NLI scores using RoBERTa-large [Liu et al., 2019], fine-tuned for natural language inference on the MNLI dataset. We noted an increase in the scores when training with *ArXiv* (+ArXiv) compared to the baseline models. Although the *TopK* strategy also improves the scores for the baseline models, the *ArXiv* (+ArXiv) models performed better on their own. Training parts (+Parts) also yielded a higher score for both the *Story* and the *Fconv* models. However, we should note that there is a significant difference between the scores of the *Story* and of the *Fconv* models due to the more extractive nature of the *Fconv* model, which ultimately yields higher NLI scores.

PR Scores. For *Fconv* models, training with *ArXiv* (+ArXiv) and in parts (+Parts) outperforms the baseline Fconv/ FconvTopK significantly by 39.8 and 39.1 absolute percentage points, respectively. For *Story* models, both training with *ArXiv* (+ArXiv) and in parts (+Parts) outperforms the *Story* baseline significantly by 42.6 and 51.1 percentage points absolute, respectively. Tables 3.6, 3.7 and 3.8 further show that, in general, Prompt Ranking is in agreement with the ROUGE scores, but it is more sensitive to training using data augmentation with *ArXiv* or using parts.

3.2.6 Discussion

Short Science Daily For *short Science Daily*, we observe that the results are particularly coherent and fluent, given the short sources. For example, in Table 3.1, in contrast to the gold summary, the source does not mention *male* or *socialist countries*. Yet, *SciBertSumAbs* gets it correctly even though *SciBERT* and *Science Daily* are biased towards biomedical topics, which is not the case here.

Long Science Daily Table 3.9 shows a summary from *long Science Daily*, which is fluent and logical, and focuses on specific information relevant to the source paper. It demonstrates structured and concise writing with sections that are both relevant and conceptually accurate. For example, it mentions that *x-ray crystallography* was used to determine the three-dimensional structure of the proteins. The target article says that this was done by the study's authors in previous work, but this technique is not mentioned in the source, which is all the model sees. This demonstrates a very important and promising phenomenon: similarly to [Tshitoya et al., 2019], where unsupervised word embeddings captured information about materials, the model learns representations of key concepts such as *x-ray crystallography*, and applies this knowledge at generation time. In contrast, the baseline *Fconv* generates fragments like *in the new study , the scientists used a technique called " dna, " the researchers say*, which misreads the meaning of DNA.

Overall, the advantages of our transfer learning experiments include (*i*) topical and factual generation, (*ii*) memorization and utilization of scientific concepts beyond

Generated: . . . histone proteins are the building blocks of proteins , and they are involved in a variety of biological functions , marino said . histones are the amino acids that make up the cell 's dna . when dna is copied , the proteins are copied to form proteins , which are called histones . histones have been studied for more than a decade , but until now it has been difficult to determine how these histone proteins assemble and how histones are assembled in the cell . . . . the researchers used a technique called **x-ray crystallography** , which allows scientists to determine the atomic structure of proteins . . . other co-authors of the paper are postdoctoral researcher zachary sandman , a former ohio state graduate student in biochemistry and molecular biology and a member of the marino lab.
Target: the colorado state university researcher studies how these hardy microbes – which constitute one of three surviving domains of life – express their genes , produce their energy , and thrive in hot , lightless environments . . . . in 1997 , luger and colleagues first reported the exact structure of eukaryotic nucleosomes via x-ray crystallography . . .
Source snippets: . . . small basic proteins present in most archaea share a common ancestor with the eukaryotic core histones . we report the crystal structure of an archaeal histone-dna complex . . . our data establish that most features of eukaryotic dna compaction into nucleosomes are conserved in archaeal histone-based chromatin . . . .

Table 3.9: Summary from *long Science Daily*. Shown are some snippets (generated, gold, and original) when using the *Story* model with top-k sampling and data augmentation using *ArXiv* (*StoryTopK+Arxiv*).

the current source, and (*iii*) clear semantic and syntactic structure.

Limitations We found that in some cases, the output of *Fconv+ArXiv*, *Story+ArXiv*, and *Fconv+Parts* is repetitive, unable to match named entities (e.g., *Zachary Sandman* in Table 3.9 is not a real person), diverging from the topic, and limited in the sense that it only has access to a single scientific paper. Moreover, the *Story* model sometimes overfits to a set of concepts, and then creates a story around those concepts rather than based on the input sequence. For example, a source paper about

the structural similarities of DNA in archaea and eukaryotes might not be accurately summarized by story-based experiments: they might elaborate on related topics, even though still focusing on DNA.

Human Evaluation on IEEE Articles Using our *SciBertSumAbs* model on *short Science Daily*, we generated summaries for five IEEE articles, randomly selected by an IEEE expert. The summaries were manually evaluated by that expert using the following criteria, which he selected independently from us:

- (Rel.) *Is the generated summary relevant to the article in context?*
- (Read.) *Is the generated summary readable by the market of interest?*
- (Compr.) *Can the summary be comprehended by the market of interest?*
- (As-is) *Is the summary acceptable As-Is?*
- (Cons.) *Can the summary be consumed by the market of interest as is (leads to effort level required from IEEE to polish the summaries before they are market-ready)?*

We present the evaluation results in Table 3.10. Overall, our summaries appear deployable after some polishing by IEEE experts. Note that, in general, human evaluation is hard, as it requires a domain expert, as opposed to evaluating topics that are common sense [Chang et al., 2009]. Evaluating even a small number of articles properly is a difficult task.

In our setting, it took more than an hour per paper by an expert. Naturally, such settings are very difficult to scale, and they take up a sizable portion of the expert's time and effort. The challenge becomes even more acute when we recognize that outsourcing such evaluations would be harder than for domains closer to a layman.

3.2.7 Conclusion and Future Work

We have proposed to study *Automating Science Journalism* (ASJ), which is the process of producing a layman's terms summary of a research article, as a new benchmark

#	**Rel.**	**Read.**	**Compr.**	**As-is**	**Cons.**
1	Y	Y	Y	Y	ML
2	Y	P	Y	N	NMT
3	Y	P	Y	N	NMT
4	Y	P	Y	N	NMT
5	Y	P	Y	N	NFT

Table 3.10: Manual expert analysis of the utility of models trained with *SciBert-SumAbs* on *short Science Daily*. See the text for a definition of the criteria and their abbreviations. Legend: Y=Yes, N=No, P=Probably, ML=Most Likely, NMT=Needs Minor Tweaks, NFT=Needs Few Tweaks.

for long neural abstractive summarization and story generation. We further created a specialized dataset that contains scientific papers and their *Science Daily* press releases: short and long versions. We demonstrated numerous *sequence to sequence* (seq2seq) applications using Science Daily with the aim of facilitating further research on language generation, which requires extreme paraphrasing and coping with long research articles. We further improved the quality of the press releases using co-training with scientific abstracts of sources or partitioned press releases. Finally, we further confirmed our results using quantitative and qualitative evaluation, including manual evaluation and analysis by a domain expert. The results suggested that our model is potentially usable in practice, possibly after post-editing.

There are many exciting directions that we plan to explore in future work. One possibility is to use more efficient linear Transformers that can model long sequences better. Another option is to encourage factuality more explicitly during training and inference, e.g., by combining variants of the NLI score and Prompt Ranking measures with the maximum likelihood objective at training time, and with the generation method at inference time. More explicit text simplification and style transfer methods could also improve the performace. Finally, we could apply our models directly to many practical problems, which would truly test generalization, and could serve as the basis of fruitful applications of automating science journalism.

3.3 Conclusion

In the discussion above, we demonstrated how we could use natural language constructions to guide the training of representations used in creating abstractive summarizes of scientific input text. A few other of our contributions are focused on the power of language as an inductive bias in improving representation learning tasks and transfer learning, in particular. In [Ramírez et al., 2020] we develop a novel method to translate between vocabularies of different languages without any supervision. In [Khoury et al., 2020] we provide a novel matrix-vector parameterization for neural networks, which enables efficient applications in machine translation and language modeling. In [Rugina et al., 2020] we developed a simple statistical method that can reduce the computational cost of running neural networks with attention mechanisms, by introducing a data-informed global sparseness in attention mechanisms. Finally, in [Vogelbaum et al., 2020] we demonstrate how the mechanism of contextualization using self-attention in multiple stages improves gradient based meta learning.

A promising direction for future work is to take inspiration from education based on exploration with the computer. A middle ground between Chinese and Greek mathematics is the educational program of using intuitive programming languages, such as Logo [Abelson and DiSessa, 1986], to allow students to explore mathematical concepts and formulate their own hypotheses for exploration. Having an intuitive language interface allows students to develop their hypotheses in well-defined "microworlds" that allow explorations and well-defined solutions [Papert, 2020]. For example, Elementary school education in Bulgaria has been hugely influenced by allowing students to explore hypotheses with educational programming languages [Henriksen et al., 2018, Stager, 2021]. In a similar fashion, a curriculum learning protocol that enables large language models to a wide range of programming tools and formulate their own hypothesis could be a fruitful bed for learning more transferable representations. Thus, we present promising directions for learning to use tools in Chapter **??**.

Chapter 4

Addressing the Lack of Interpretability with the Symbolic Inductive Bias

4.1 Introduction

In his profound endeavor writing the "Principia," Sir Isaac Newton meticulously documented a plethora of observations and elegantly derived the equations of *classical mechanics* that encapsulate these phenomena [Newton, 1687]. The significance of his discoveries was profound, given the universal applicability of his laws across diverse systems—from the diminutive dynamics of a falling apple to the colossal movements of celestial bodies. One can speculate that the immense power of Newton's laws is fueled by their simplistic, intuitive, symbolic representations.

Symbolic expressions stand at the core of scientific revelations. The challenge lies in deciphering the correct symbolic expression that accurately describes the observed data, a task known as symbolic regression. Its potential benefits are noteworthy, including interpretability and generalizability, which grant scientists an opportunity to deduce meaningful insights and extrapolate to new, uncharted situations.

However, the pursuit of symbolic regression presents its own set of formidable

challenges. The search navigates through a combinatorially vast space of expressions – a task that appears hopelessly daunting if undertaken at random. Furthermore, there is no straightforward method to define gradients, as is the case for neural network optimization, which compounds the complexity of the problem.

How then do we venture through this labyrinthine search space efficiently? A brute force enumeration of solutions is computationally prohibitive. Here, reinforcement learning, especially deep reinforcement learning, offers a glimmer of hope. By intelligently exploring the search space, these methods have yielded profound discoveries. The Monte Carlo Tree Search (MCTS) was instrumental in shaping DeepMind's revolutionary game-playing techniques such as AlphaGo [Silver et al., 2017], AlphaZero [Silver et al., 2018], MuZero [Schrittwieser et al., 2020], as well as in conceptualizing optimization challenges as games, evident in matrix mulitplaciatoin algorithms, i.e. AlphaTensor [Fawzi et al., 2022] and sorting algorithms, i.e. AlphaDev [Mankowitz et al., 2023]. In all of these accomplishments, a neural network is trained to make actions in a well-defined environment and the observed behavior of the network led to discoveries that changed the way experts think about the games/ optimization challenges.

Yet, while neural networks offer a means to traverse the solution space, the representations they learn during this quest are often cryptic and provide limited utility. What if we could foster interpretable representations within the neural network during the search process? By making minimal modifications to the network, we could exploit the extensive engineering and theoretical advancements in training and deploying neural networks, but repurpose them for interpretable internal representations.

It is worth noting that Newton, with his "biological neural network," was able to conceive symbolic expressions that dictate the dynamics of our world. What distinguished him from DeepMind's deep reinforcement learning was his ability to articulate intermediate representations in a symbolic form, as evidenced by his authorship of scientific treatises.

The introduction of symbolic bias in neural networks could potentially enable us to conduct an efficient and interpretable search. The remaining part of this chapter

focuses on presenting our contributions in that direction of research.

4.2 OccamNet: A Fast Neural Model for Symbolic Regression at Scale

4.2.1 Introduction

Deep learning has revolutionized a variety of complex tasks, ranging from language modeling to computer vision [LeCun et al., 2015b]. Key to this success is designing a large search space in which many local minima sufficiently approximate given data [Choromanska et al., 2015]. This requires large, complex models, which conflicts with the goals of sparsity and interpretability, making neural nets ill-suited for a myriad of physical and computational problems with compact and interpretable underlying mathematical structures [Lample and Charton, 2020]. Neural networks also might not preserve desired physical properties (e.g., time invariance) and are unable to generalize beyond observed data.

In contrast, Evolutionary Algorithms (EAs), in particular genetic programming, can find interpretable, compact models that explain observed data [Schmidt and Lipson, 2009, Udrescu and Tegmark, 2020, Poli et al., 2008]. EAs have been employed as an alternative to gradient descent for optimizing neural networks in what is known as *neuroevolution* [Angeline et al., 1994, Arnold and Hansen, 2012, Such et al., 2017]. Recently, evolutionary strategies that model a probability distribution over parameters, updating this distribution according to their own best samples (i.e., selecting the fittest), were found advantageous for optimization on high-dimensional spaces, including neural networks' hyperparameters [Hansen, 2016, Loshchilov and Hutter, 2016].

A number of evolution-inspired, probability-based models have been explored for Symbolic Regression [Mckay et al., 2010]. Along these lines, Petersen et al. [2021] explore deep symbolic regression by using an RNN to define a probability distribution over a space of expressions and sample from it using autoregressive expression

generation. More recently, Biggio et al. [2021] have pretrained Transformer models that receive input-output pairs as input and return functional forms that could fit the data. In the related field of program synthesis, probabilistic program induction using domain-specific languages [Ellis et al., 2018b,a, 2019] has proven successful. Balog et al. [2016] first train a machine learning model to predict a DSL based on input-output pairs and then use methods from satisfiability modulo theory [Solar Lezama, 2008] to search the space of programs built using the predicted DSL.

One approach to symbolic regression which can integrate with deep learning is the Neural Arithmetic Logic Unit (NALU) and related models [Trask et al., 2018a, Madsen and Johansen, 2020], which provide neural inductive bias for arithmetic in neural networks by shaping a neural network towards a gating interpretation of the linear layers. Neural Turing Machines [Graves et al., 2014, 2016] and their stable versions [Collier and Beel, 2018] can also discover interpretable programs, simulated by neural networks connected to external memory, via observations of input-output pairs. Another option is Equation Learner (EQL) Networks [Martius and Lampert, 2016, Sahoo et al., 2018a, Kim et al., 2020b], which identify symbolic fits to data by training a neural network with symbolic activation functions, such as multiplication or trigonometric functions. However, these methods require strong regularization to be interpretable. NALUs and to a lesser extent EQL Networks can also only use a restricted set of differentiable basis functions, and Neural Turing Machines do not include the concept of a "basis." Additionally, these methods often converge to local minima and often converge to uninterpretable models unless they are carefully regularized for sparsity.

In this paper, we consider a mixed approach of connectionist and sample-based optimization for symbolic regression. We propose a neural network architecture, OccamNet, which preserves key advantages of EQL networks and other neural-integrable symbolic regression frameworks while addressing many of these architectures' limitations. Inspired by neuroevolution, our architecture uses a neural network to model a probability distribution over functions. We optimize the model by sampling to compute a REINFORCE-type loss, tunable for different tasks, based on the train-

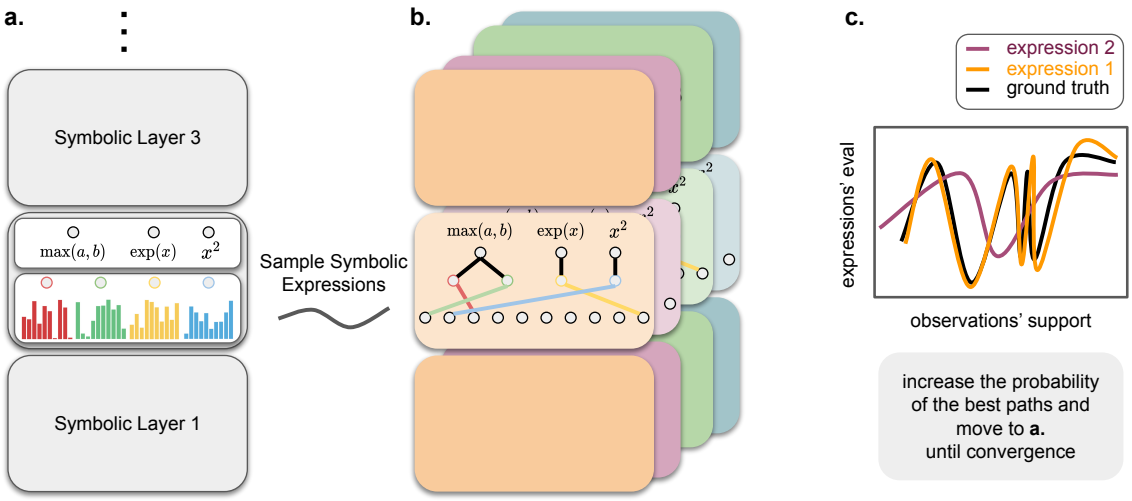

Figure 4-1: OccamNet architecture and training. **a.** OccamNet is a stack of "symbolic layers" each described by a collection of learned distributions (over the neurons from the previous layer) for each neuron within the layer, as well as non-linearities that are collections of symbolic expressions. **b.** By sampling from each distribution independently, we are able to sample paths from OccamNet that represent symbolic expressions, ready for evaluation. **c.** We evaluate each expression by feeding the observations' support data and comparing the outputs with the ground truth. The probability of the best paths is increased and the process is repeated until convergence.

ing method presented in Risk-Seeking Policy Gradients [Petersen et al., 2021]. Our method handles non-differentiable and implicit functions, converges to sparse, interpretable symbolic expressions, and can work across a wide range of symbolic regression problems. Further, OccamNet consistently outperforms other symbolic regression algorithms in testing on real-world regression datasets. We also introduce a number of strategies to induce compactness and simplicity, à la Occam's Razor.

4.2.2 Model Architecture

In Figure 4-1 we sketch the OccamNet architecture and the method for training it, before following with a more detailed description. We can view OccamNet as a standard feed-forward network, a stack of fully connected linear layers with non-linearities. The unique features of OccamNet are two-fold. First, the parameters of the linear layer are

substituted with a learned probability distribution associated with the neurons from the preceding layer for each neuron within the layer. Second, the non-linearities form a collection of symbolic expressions. Thus we obtain a collection of "symbolic layers" that form OccamNet (panel **a.**). Panel **b.** shows a variety of symbolic expressions, representing paths within OccamNet from sampling each probability distribution independently. Panel **c.** shows OccamNet's training objective, which increases the probability of the paths that are closest to the ground truth. Below we formalize OccamNet in details.

Layer structure

A dataset $\mathcal{D} = \{(\vec{x}_p, \vec{y}_p)\}_{p=1}^{|\mathcal{D}|}$ consists of pairs of inputs $\vec{x}_p$ and targets $\vec{y}_p = \vec{f}^*(\vec{x}_p) = [f^*_{(0)}(\vec{x}_p), \ldots, f^*_{(v-1)}(\vec{x}_p)]^\top$. Our goal is to compose either $f^*_{(i)}(\cdot)$ or an approximation of $f^*_{(i)}(\cdot)$ using a predefined collection of N basis functions $\mathbf{\Phi} = \{\phi_i(\cdot)\}_{i=1}^N$, which act as primitives. Note that bases can be repeated, their arity (number of arguments) is not restricted to one, and they may operate over different domains. The concept of bases Φ is similar to that of DSL, *domain-specific languages* [Fowler, 2010].

To solve this problem, we follow a similar approach as in EQL networks [Sahoo et al., 2018a, Martius and Lampert, 2016, Kim et al., 2020b], in which the bases act as activation functions on the nodes of a neural network. Specifically, each hidden layer consists of an *arguments* sublayer and an *images* sublayer, as shown in Figure 4-7a (in the SM). The bases are stacked in the images sublayer and act as activation functions for their respective nodes. Each basis takes in nodes from the arguments sublayer. Additionally, we use skip connections similar to those in DenseNet [Huang et al., 2017] and ResNet [He et al., 2016b], concatenating image states with those of subsequent layers.

To overcome EQL networks' challenges, we introduce a probabilistic modification. Instead of computing the inputs to the arguments sublayers using dense feed-forward layers, we compute them probabilistically and sample through the network. This addresses all of the issues with EQL described above: *(i)* sampling enforces sparsity and interpretability without requiring regularization, *(ii,iii)* sampling allows us to avoid

backpropagating through the activation functions, thereby allowing non-differentiable and fast-growing functions in the bases, and *(iv)* sampling helps our model avoid premature convergence to local minima.

Because they behave probabilistically, we call nodes in the arguments sublayer *P-nodes*. Figure 4-7 highlights this sublayer structure, while the supplemental material (SM) describes the complete mathematical formalism behind it.

Temperature-controlled connectivity

Instead of dense linear layers, we use *T-softmax layers*. For any temperature $T > 0$, we define a T-softmax layer as a standard T-controlled softmax layer with weighted edges connecting an images sublayer and the subsequent arguments sublayer, in which each P-node from the arguments sublayer probabilistically samples a single edge between itself and a node in the images sublayer. Each node's sampling distribution is given by

$$\mathbf{p}^{(l,i)}(T) = \text{softmax}(\mathbf{w}^{(l,i)}; T),$$

where $\mathbf{w}^{(l,i)}$ and $\mathbf{p}^{(l,i)}$ are the weights and probabilities for edges leading to the ith P-node of the lth layer. Selecting these edges for all T-softmax layers produces a sparse DAG specifying a function $\vec{f}$, as seen in Figure 4-7b. OccamNet thus automatically enforces sparsity.

A neural network as a probability distribution over functions

Let $\mathbf{W} = \left\{\mathbf{w}^{(l,i)}; 1 \leq l \leq L, 1 \leq i \leq N\right\}$. The probability of the model sampling $f_{(i)}$ as its ith output, $q_i(f_{(i)}|\mathbf{W})$, is the product of the probabilities of the edges of $f_{(i)}$'s DAG. Similarly, $q(\vec{f}|\mathbf{W})$, the probability of the model sampling $\vec{f}$, is given by the product of $\vec{f}$'s edges, or $q(\vec{f}|\mathbf{W}) = \prod_{i=0}^{v-1} q_i(f_{(i)}|\mathbf{W})$.

In practice, we compute an approximate of this probability which we denote $\tilde{q}$, as described in Methods Section 4.2.6. We find that OccamNet performs well with this approximation. For all other sections of this paper, unless explicitly mentioned, we use q to mean $\tilde{q}$.

Our model thus represents a probability distribution $q(\cdot|\mathbf{W})$ over a function space of all functions sampleable by the network,

$$\mathcal{F}_{\mathbf{\Phi}}^{L} = \{\text{all function compositions up to nesting depth } L \text{ of } \mathbf{\Phi}\}.$$

We initialize the network with weights $\mathbf{W}_{\text{i}}$ such that $\tilde{q}(\vec{f}_1|\mathbf{W}_{\text{i}}) = \tilde{q}(\vec{f}_2|\mathbf{W}_{\text{i}})$ for all $\vec{f}_1$ and $\vec{f}_2$ in $\mathcal{F}_{\mathbf{\Phi}}^{L}$. After training (Section 4.2.3), the network has weights $\mathbf{W}_{\text{f}}$. The network then selects the function $\vec{f}_{\text{f}}$ with the highest probability $\tilde{q}(\vec{f}_{\text{f}}|\mathbf{W}_{\text{f}})$. We discuss our algorithms for initialization and function selection in the Methods section. A key benefit of OccamNet is that, unlike other approaches such as Petersen et al. [2021], it allows for efficiently identifying the function with the highest probability using a dynamic programming algorithm.

4.2.3 Training

To express a wide range of functions, we include non-differentiable and fast-growing bases. Additionally, in symbolic regression, we are interested in finding global minima. To address these constraints, we implement a loss function and training method that combine gradient-based optimization and sampling-based strategies for efficient global exploration of the function space. Our loss function and training procedure are closely related to those proposed by Petersen et al. [2021], differing mainly in the fitness function and regularization terms.

Loss Function

Consider a mini-batch $\mathcal{M} = (X, Y)$ and a sampled function from the network $\vec{f}(\cdot) \sim q(\cdot|\mathbf{W})$. We compute the *fitness* of each $f_{(i)}(\cdot)$ with respect to a training pair $(\vec{x}, \vec{y})$ by evaluating

$$k_i\left(f_{(i)}(\vec{x}), \vec{y}\right) = (2\pi\sigma^2)^{-1/2} \exp(-\left[f_{(i)}(\vec{x}) - (\vec{y})_i\right]^2 / (2\sigma^2)),$$

which measures how close $f_{(i)}(\vec{x})$ is to the target $(\vec{y})_i$. The total fitness is determined by summing over the entire mini-batch: $K_i\left(\mathcal{M}, f_{(i)}\right) = \sum_{(\vec{x},\vec{y})\in\mathcal{M}} k_i\left(f_{(i)}(\vec{x}), \vec{y}\right)$.

We then define the loss function

$$H_{q_i}[f_{(i)}, \mathbf{W}, \mathcal{M}] = -K_i\left(\mathcal{M}, f_{(i)}\right) \cdot \log\left[q_i(f_{(i)}|\mathbf{W})\right]. \tag{4.1}$$

as in Petersen et al. [2021]. As in Petersen et al. [2021], we train the network by sampling functions, selecting the λ functions with the highest fitness for each output, and perform a gradient step based on these highest-fitness functions using the loss defined in Equation 4.1.

To improve implicit function fitting, we implement regularization terms that punish trivial solutions by reducing the fitness K, as discussed in the Methods Section. We also introduce regularization to restrict OccamNet to solutions that preserve units.

Recurrence

OccamNet can also be trained to find recurrence relations. To augment the training algorithm, for each sampled function, we compute its recurrence to a maximum depth D, obtaining a collection of RD functions. Training continues similarly to Petersen et al. [2021], in which we compute the corresponding fitnesses, select the best $v\lambda$ functions, and update the weights. See the Methods section for more details.

4.2.4 Results

To empirically validate our model, we first develop a diverse collection of benchmarks in four categories: *Analytic Functions*, simple, smooth functions; *Implicit Functions*, functions specifying an implicit relationship between inputs; *Non-Analytic Functions*, discontinuous and/or non-differentiable functions; *Image/Pattern Recognition*, patterns explained by analytic expressions. We then test OccamNet's performance and ability to scale on real-world symbolic regression datasets. The purpose of these experiments is to demonstrate that OccamNet can perform competitively with other symbolic regression frameworks in a diverse range of applications.

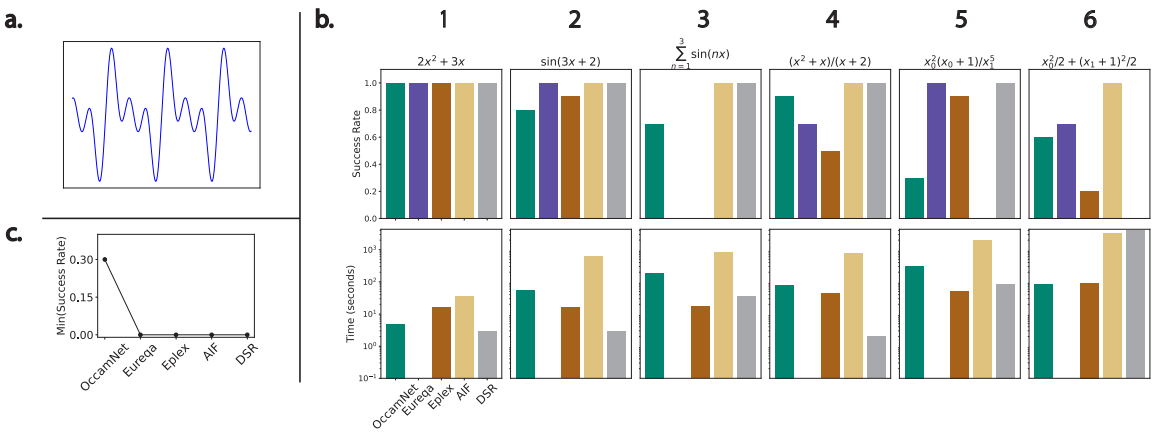

Figure 4-2: Experiment on analytic functions. **a.** A sketch of the function $\sum_{n=1}^{3}\sin(nx)$ as an example of the analytics functions we consider in our work. **b.** Success rate (out of 10 trials) for each of the five methods considered: OccamNet, Eureqa, Eplex, AI Feynman 2.0 (AIF) and Deep Symbolic Regression (DSR) (at the top). Training time for the methods (at the bottom). Eureqa almost always finishes much more quickly than the other methods, so we do not provide training times for Eureqa. We enumerate the functions to ease the discussion. **c.** The "worst-case" performance for each methods, showing the minimal success rate across the six tasks.

We compare OccamNet with several other symbolic regression methods: Eureqa [Schmidt and Lipson, 2009], a genetic algorithm with Epsilon-Lexicase (Eplex) selection [La Cava et al., 2016], AI Feynman 2.0 (AIF) [Udrescu and Tegmark, 2020, Udrescu et al., 2020], and Deep Symbolic Regression (DSR) [Petersen et al., 2021]. We do not compare to Transformer-based models such as Biggio et al. [2021] because, unlike our method, these methods utilize a prespecified and immutable set of basis functions which are not always sufficiently general to fit our experiments. The results are shown in Tables 4.1, 4.2, and 4.3, and we discuss them below. More details about the experimental setup are given in the SM.

Analytic functions

In Figure 4.2.4 and Table 4.1 (in the Methods) we present our results on analytic functions. Panel **a.** presents an analytic function that is particularly challenging for Eureqa. Panel **b.** shows that OccamNet is competetive to state-of-the-art symbolic regression methods, while it has the only non-zero minimal success rate across the

considered functions (panel **c.**).

We highlight the large success rate for function 4, which we originally speculated could easily trick the network with the local minimum $f(x) \approx x + 1$ for large enough x. In contrast, as with the difficulties faced by AI Feynman, we find that OccamNet often failed to converge for function 5 because it approximated the factor $x_0^2(x_0 + 1)$ to x_0^3; even when convergence did occur, it required a relatively large number of steps for the network to resolve this additional constant factor. Notably, Eureqa and Eplex had difficulty finding function 3.

AI Feynman consistently identifies many of the functions, but it struggles with function 5 and is also generally much slower than other approaches. Eplex also performs well on most functions and is fast. However, like Eureqa, Eplex struggles with functions 3 and 6. We suspect that this is because evolutionary approaches require a larger sample size than OccamNet's training procedure to adequately explore the search space. DSR consistently identifies many of the functions and is very fast. However, DSR struggles to fit Equation 6, which we suspect is because such an equation is complex but can be simplified using feature reuse. OccamNet's architecture allows such feature reuse, demonstrating an advantage of OccamNet's inductive biases.

Non-analytic functions

In Figure 4.2.4 and Table 4.2 (in the Methods) we benchmark the ability to find several non-differentiable, potentially recursive/iterative functions. From our experiments, we highlight both the network's fast convergence to the correct functional form and the discovery of the correct recurrence depth of the final expression. This is pronounced for function 7 in, which is a challenging chaotic series on which Eureqa and Eplex struggle. Interestinly, Eplex fails to identify the simpler functions 1-3 correctly. We suspect that this may be because, for these experiments, we restrict both OccamNet and Eplex to smaller expression depths. Although OccamNet is able to identify the correct functions with small expression depth, we suspect that Eplex often identifies expressions by producing more complex equivalents to the correct program and so cannot identify the correct function when restricted to simpler expressions.

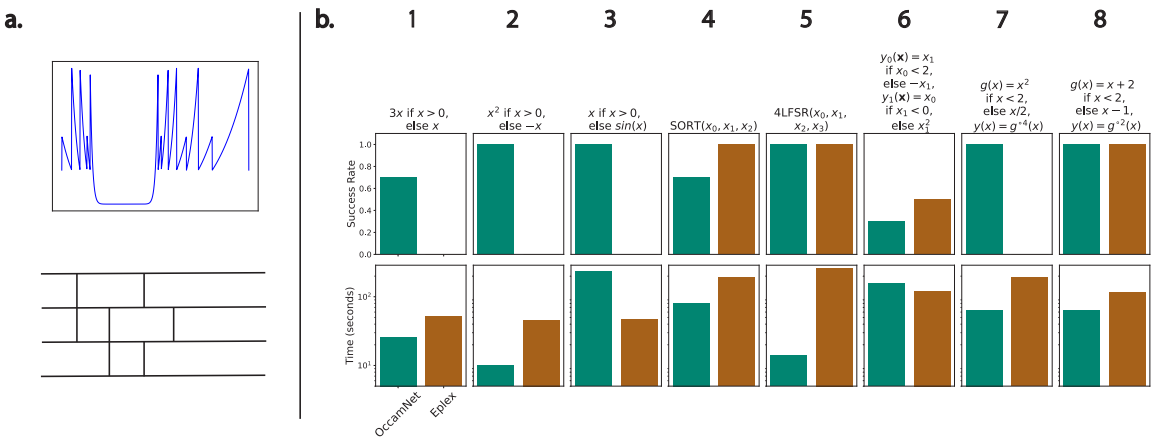

Figure 4-3: Experiments on non-analytic functions. **a.** Two prominent examples of non-analytic functions: The challenging recursion $g(x) = x^2$ if $x < 2$, else $x/2$, $y(x) = g^{\circ 4}(x)$ (top) and a sorting circuit of three numbers (bottom). **b.** Success rate (out of 10 trails) and training time for OccamNet and Eplex. We enumerate the functions to ease the discussion.

We also investigated the usage of bases such as MAX and MIN to sort numbers (function 4), obtaining relatively well-behaved final solutions: the few solutions that did not converge fail only in deciding the second component, y_2, of the output vector. Finally, we introduced binary operators and discrete input sets for testing a simple 4-bit LFSR (function 5), the function $(x_0, x_1, x_2, x_3) \rightarrow (x_0 + x_3 \mod 2, x_0, x_1, x_2)$, which converges fast with a high success rate.

We do not compare to AI Feynman in these experiments because AI Feynman does not support the required basis functions.

Implicit Functions and Image Recognition

Figure 4-4 and Table 4.3 demonstrate applications of OccamNet in domains that are not natural for standard symbolic regression baselines, but are quite natural for OccamNet due to its interpretation as a feed-forward neural network.

OccamNet demonstrates a sizable advantage on all of the implicit functions. Notably, Eureqa is unsuccessful in fitting $m_1 v_1 - m_2 v_2 = 0$ (conservation of momentum). Note that we only compare OccamNet to Eureqa for Implicit Functions because none of the other methods include regularization to fit such functions.

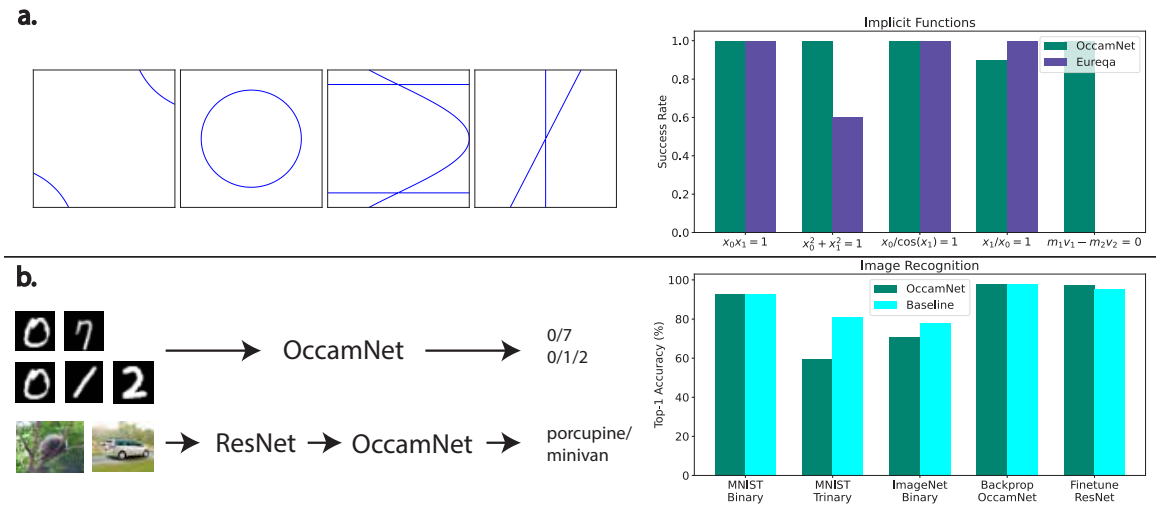

Figure 4-4: Experiments on implicit functions and standard vision benchmarks. **a.** Examples of implicit functions' loci (left) and the corresponding success rate on a suite of implicit functions (right). **b.** Examples of image recognition tasks (left) and the best accuracy from 10 trials for both OccamNet and the baseline. The baseline for MNIST Binary/ Trinary and ImageNet Binary is HeuristicLab Wagner et al. [2014]. The baseline for Backprop OccamNet and Finetune ResNet is a feed-forward neural network with the same number of parameters as OccamNet.

We train OccamNet to classify MNIST [LeCun et al., 1998b][1] in a binary setting between the digits 0 and 7 (*MNIST Binary*). For this high-dimensionality task, we implement OccamNet on an Nvidia V100 GPU, yielding a sizable 8x speed increase compared to a CPU. For MNIST Binary, one of the successful functional fits that OccamNet finds is $y_0\left(\vec{x}\right) = \tanh\left(10(\max(x_{715}, x_{747}) + \tanh(x_{435}) + 2x_{710} + 2x_{713})\right)$ and $y_1\left(\vec{x}\right) = \tanh\left(10\tanh(10\left(x_{512} + x_{566}\right))\right)$. The model learns to incorporate pixels into the functional fit that are indicative of the class: here x_{512} and x_{566} are indicative of the digit 7. These observations hold when we further benchmark the integration of OccamNet with deep feature extractors. We extract features from ImageNet [Deng et al., 2009a][2] images using a ResNet 50 model, pre-trained on ImageNet [He et al., 2016b]. For simplicity, we select two classes, "minivan" and "porcupine" (*ImageNet Binary*). OccamNet significantly improves its accuracy by backpropagating through our model using a standard cross-entropy signal. We either freeze the ResNet weights

[1]Creative Commons Attribution Share Alike 3.0 License
[2]The Creative Commons Attribution (CC BY) License

(*Backprop OccamNet*) or finetune ResNet through OccamNet (*Finetune ResNet*). In both cases, the converged OccamNet represents simple rules, ($y_0(\vec{x}) = x_{1838}$, $y_1(\vec{x}) = x_{1557}$), suggesting that replacing the head in deep neural networks with OccamNet might be promising.

Real-world regression datasets

We also test OccamNet's ability to fit real-world datasets, selecting 15 datasets with 1667 or fewer datapoints from the Penn Machine Learning Benchmarks (PMLB[3]) regression datasets Olson et al. [2017]. These are real-world datasets, and based on their names, we infer that many are from social science, suggesting that they are inherently noisy and likely to follow no known symbolic law. Additionally, 1/3 of the datasets we choose have feature sizes of 10 or greater. These factors make the PMLB datasets challenging symbolic regression tasks. We again compare OccamNet to Eplex and AI Feynman 2.0.[4]

We test OccamNet twice. For the first test, "OccamNet," we test exactly 1,000,000 functions, the same number as we test for Eplex. For the second test, "V100," we exploit our architecture's integration with the deep learning framework by running OccamNet on an Nvidia V100 GPU and testing a much larger number of functions. We allow AIF to run for approximately as long or longer than OccamNet for each dataset.

As discussed in the SM, we perform grid search on hyperparameters and identify the fits with the best training, validation, and testing Mean Squared Error (MSE) losses. The raw data from these experiments are shown in the SM.

Figure 4-5 shows the relative performance of OccamNet and comparison datasets according to several metrics. As shown in Figure 4-5a,b,c, overall, Eplex outperforms OccamNet in training and testing MSE loss, but OccamNet outperforms Eplex in validation loss. We speculate that OccamNet's performance drop between the validation

[3]Creative Commons Attribution 4.0 International License

[4]AIF's regression algorithm examines all possible feature subsets, the number of which grows exponentially with the number of features. Accordingly, we only test the datasets with ten or fewer features. AI Feynman failed to run on a few datasets. All remaining datasets are included in tables and figures.

and testing datasets results from overfitting from the larger set of hyperparameter combinations for OccamNet (details in the SM).

Additionally, OccamNet runs faster than Eplex in nearly all datasets tested, often by an order of magnitude (Figure 4-5d). Furthermore, OccamNet is highly parallel and can easily scale on a GPU. Thus, a major advantage of OccamNet is its speed and scalability (see Section 4.2.4 for a further discussion of OccamNet's scaling). Comparing V100 and Eplex demonstrates that OccamNet continues to improve when testing more functions. The testing MSE is where V100 performs worst in comparison to Eplex (see Figure 4-5e), but it still outperforms Eplex at 10/15 of the datasets while running more than nine times faster. Thus OccamNet's speed and scalability can be exploited to greatly increase its accuracy at symbolic regression. This demonstrates that OccamNet is a powerful alternative to genetic algorithms for interpretable data modeling.

OccamNet also outperforms AIF for training, validation, and testing while running faster. OccamNet achieves a lower training and validation MSE than AIF for every dataset tested. For training loss, OccamNet performs better than AIF in 4/7 datasets (Figure 4-5f). Additionally, unlike OccamNet, AIF performs polynomial fitting, giving it an additional advantage. However, the datasets we test are likely a worst-case for AIF; the datasets are small, have no known underlying formula, and we normalize the data prior to training, meaning that AIF will likely struggle not to overfit with its neural network and will also be unlikely to identify graph modularities.

Scaling on real-world regression datasets

As discussed in Section 4.2.4, OccamNet runs far more quickly than Eplex, meaning that it can scale to testing far more functions per epoch than Eplex in the same runtime. To explore this advantage, we compare OccamNet running on an Nvidia V100 GPU (OccamNet) against Eplex while varying the number of functions sampled per epoch for each method. We benchmark both methods on the same 15 PMLB datasets (see the methods section for more details).

We include and discuss the complete results of this experiment in Appendix I.

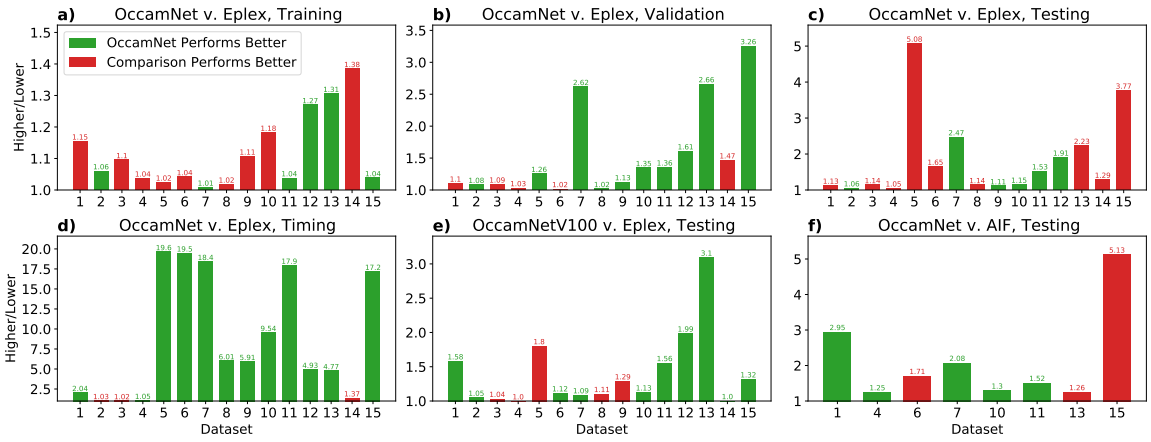

Figure 4-5: A bar chart showing the relative performance between OccamNet and two baseline methods, Eplex and AIF. The x-axis is the dataset involved. The y-axis is the relative performance according to the given metric: the MSE on the training, validation, or testing set or the training time. To compute this relative performance, we divide the higher (worse) performance value by the lower (better) performance value for each dataset. The green bars represent datasets where OccamNet has a lower (better) performance value than the comparison baseline method, and the red bars represent the datasets where the comparison method has a better performance than OccamNet.

In this section, we highlight key results. Figure 4-6 shows that OccamNet is often more than an order of magnitude faster than Eplex. Eplex scales quadratically with the number of functions, whereas OccamNet's runtime asymptotes to linear growth. However, the V100 GPU's extreme parallelism initially suppresses OccamNet's linear growth, demonstrating an advantage of OccamNet's ability to scale on a GPU.

In all of the 15 datasets, OccamNet's training loss decreases with larger runtimes, demonstrating that OccamNet can utilize the greater number of sampled functions that its efficient scaling allows. Additionally, for 11 of the training datasets, the OccamNet best fit has a loss that is lower than or equal to the Eplex best fit. Interestingly, OccamNet's validation and training loss do not always show such a clear trend of improvement with increasing sample size. Given that the training loss does improve, we suspect that this is a case of overfitting. OccamNet's validation loss does decrease for most of the datasets.

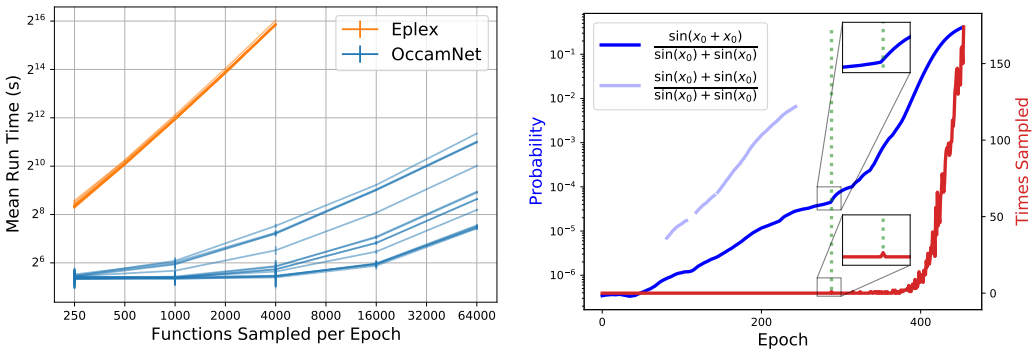

Figure 4-6: *Left:* The run time for OccamNet V100 or Eplex as a function of the number of functions sampled per epoch. Each curve represents one of the 15 datasets. *Right:* Gradual modularity with training. Dark blue is the correct function. Light blue is a suboptimal fit with a high probability early in training. Red corresponds to the correct function. The insets show the first sample of the correct function.

4.2.5 Discussion

Since our experimental settings did not require very large depths, we have not tested the limits of OccamNet in terms of depth rigorously (preliminary results on increasing the depth for pattern recognition are in the SM). We expect increasing depth to yield significant complications as the search space grows exponentially. We recognize the need to create symbolic regression benchmarks that would require expressions that are large in depth. We believe that other contributions to symbolic regression would also benefit from such benchmarks. Another direction where OccamNet might be improved is low-level optimization that would make the method more efficient to train. For example, in our PMLB experiments, we estimate that OccamNet performs >8x as many computations as necessary. Eplex may also benefit from optimization. Finally, similarly to other symbolic regression methods, OccamNet requires a specified basis to fit a dataset. While it is a notable advantage of OccamNet to have non-differentiable bases, further work needs to be done to explore optimization at a meta level that discovers appropriate bases for the datasets of interest.

OccamNet's learning procedure allows it to combine partial solutions into better results. For example in Figure 4-6, the correct function's probability increases by more than 100 times *before being sampled* because OccamNet samples similar approximate

solutions.

OccamNet successfully fits many implicit functions that other neurosymbolic architectures struggle to fit because of the non-differentiable regularization terms required to avoid trivial solutions. Although Eureqa also fits many of these equations, we find that it sometimes requires the data to be ordered by some latent variable and struggles when the dataset is very small. This is likely because Eureqa numerically evaluates implicit derivatives from the dataset [Schmidt and Lipson, 2010], which can be noisy when the data is sparse. While Schmidt and Lipson [2010] propose methods for analyzing unordered data, it is unclear whether these methods have been implemented in Eureqa. Thus, OccamNet seems to shine in its ability to fit unordered and small datasets described by implicit equations (e.g., momentum conservation in line 5 in Table 4.3).

To our knowledge, a unique advantage of our method is that OccamNet represents complete analytic expressions with a single forward pass, which allows sizable gains when using an AI accelerator, as demonstrated by our experiments on a V100 GPU (Figure 4-5). Furthermore, because of this property, OccamNet can be easily integrated with components from the standard deep learning toolkit. For example, lines 9-10 in Table 4.3 demonstrate integration and joint optimization with neural networks which is not possible with Eureqa. We also conjecture that such integration with autoregressive approaches [Petersen et al., 2021] might be challenging as the memory and latency would increase.

An advantage of OccamNet over transformer-based approaches to symbolic regression is that OccamNet is basis-agnostic. In particular, OccamNet can find fits to data regardless of the basis functions it is given, whereas transformer-based models [Biggio et al., 2021] can only fit functions that contain a certain set of basis functions chosen at pretraining time. This makes OccamNet and other similar approaches more flexible and broadly applicable than transformer-based models. As discussed above, this is the reason that we do not compare against transformer-based methods in our experiments.

4.2.6 Methods

We divide our methods section into two parts. In Section 4.2.6, we provide a more detailed description of OccamNet, and in 4.2.6 we fully describe the setup for all of our experiments.

Complete Model Description

We divide this section as follows:

1. In Section 4.2.6, we present additional materials that support the figures from the main text.
2. In Section 4.2.6, we describe OccamNet's probability distribution.
3. In Section 4.2.6, we describe of OccamNet's sampling process.
4. In Section 4.2.6, we describe OccamNet's initialization process.
5. In Section 4.2.6, we describe OccamNet's function selection.
6. In Section 4.2.6, we describe OccamNet's loss function.
7. In Section 4.2.6, we describe OccamNet's training.
8. In Section 4.2.6, we describe OccamNet's two-step training method for fitting constants.
9. In Section 4.2.6, we describe OccamNet's handling of recurrence.
10. In Section 4.2.6, we describe OccamNet's regularization for fitting implicit functions.
11. In Section 4.2.6, we describe OccamNet's method for regularizing to respect units.
12. In Section 4.2.6, we describe OccamNet's procedure for handling functions with undefined outputs.

Supporting Materials for the Main Figures

Figure 4-7 presents our OccamNet architecture in more detail. Tables 4.1,4.2, and 4.3 present our experiments in a tabular format.

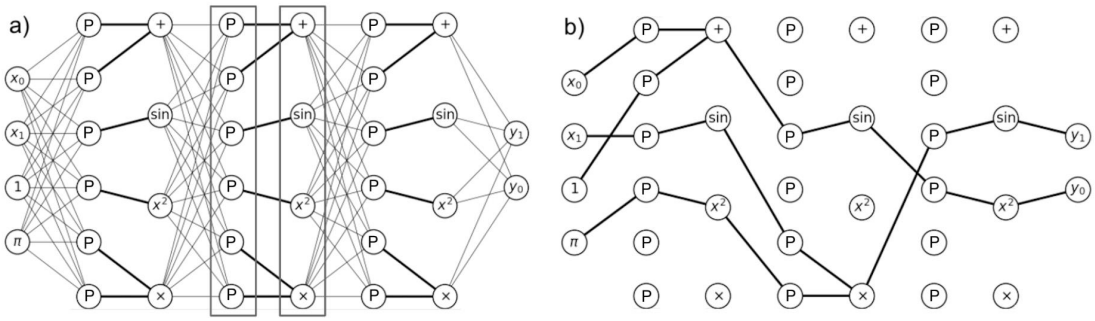

Figure 4-7: (a) A two-output network model with depth $L = 2$, $\vec{x} = [x_0, x_1]$, user-selected constants $\mathcal{C} = [1, \pi]$, and basis functions $\mathbf{\Phi} = (+(\cdot,\cdot), \sin(\cdot), (\cdot)^2, \times(\cdot,\cdot))$. Highlighted are the arguments sublayer, composed of P-nodes, and the images sublayer, composed of the basis functions from $\mathbf{\Phi}$. Together, these two sublayers define a single layer of our model. (b) An example of function-specifying directed acyclic graphs (DAGs) that can be sampled from the network in (a). These DAGs represent the functions $y_0 = \sin^2(x_0 + 1)$ and $y_1 = \sin(\pi^2 \sin(x_1))$.

Table 4.1: Analytic Functions. The proportion of 10 trials that converge to the correct analytic function for OccamNet, Eureqa, Eplex, AI Feynman 2.0, and Deep Symbolic Regression (DSR). *sec.* is the average number of seconds for convergence. Eureqa almost always finishes much more quickly than the other methods, so we do not provide training times for Eureqa.

Analytic Functions										
#	Targets	OccamNet	sec.	Eureqa	Eplex	sec.	AI Feynman	sec.	DSR	sec.
1	$2x^2 + 3x$	1.0	5	1.0	1.0	16	1.0	35	1.0	3
2	$\sin(3x + 2)$	0.8	56	1.0	0.9	16	1.0	620	1.0	3
3	$\sum_{n=1}^{3} \sin(nx)$	0.7	190	0.0	0.0	17	1.0	815	1.0	36
4	$(x^2 + x)/(x + 2)$	0.9	81	0.7	0.5	44	1.0	807	1.0	2
5	$x_0^2(x_0 + 1)/x_1^5$	0.3	305	1.0	0.9	53	0.0	1918	1.0	84
6	$x_0^2/2 + (x_1 + 1)^2/2$	0.6	83	0.7	0.2	92	1.0	3237	0.0	3935

OccamNet's Probability Distribution As discussed in the main text, OccamNet represents not only the probability of sampling a given function $\vec{f} = (f_{(0)}, \ldots, f_{(v-1)})^\top$ but also the probability of sampling each $f_{(i)}$ independently of the other components of $\vec{f}$. Because $q(\cdot|\cdot)$ is a probability distribution we have $\sum_{\vec{f} \in \mathcal{F}_{\mathbf{\Phi}}^L} q(\vec{f}|\mathbf{W}) = 1$ and

Table 4.2: Non-analytic Functions. The proportion of 10 trials that converge to the correct function for OccamNet, Eureqa, and Eplex. *sec.* is the average number of seconds for convergence. Eureqa almost always finishes much more quickly than OccamNet and Eplex, so we do not provide training times for Eureqa. *For program #6, Eplex fits y_1 every time and never fits y_0 correctly, so we give it a score of 0.5.

Non-analytic Functions						
#	Targets	OccamNet	sec.	Eureqa	Eplex	sec.
1	$3x$ if $x > 0$, else x	0.7	26	1.0	0.0	52
2	x^2 if $x > 0$, else $-x$	1.0	10	1.0	0.0	46
3	x if $x > 0$, else $\sin(x)$	1.0	236	1.0	0.0	47
4	$\mathsf{SORT}(x_0, x_1, x_2)$	0.7	81	1.0	1.0	191
5	$\mathsf{4LFSR}(x_0, x_1, x_2, x_3)$	1.0	14	1.0	1.0	262
6	$y_0(\vec{x}) = x_1$ if $x_0 < 2$, else $-x_1$ $y_1(\vec{x}) = x_0$ if $x_1 < 0$, else x_1^2	0.3	157	0.1	*0.5	121
7	$g(x) = x^2$ if $x < 2$, else $x/2$ $y(x) = g^{\circ 4}(x)$	1.0	64	0.0	0.0	189
8	$g(x) = x + 2$ if $x < 2$, else $x - 1$ $y(x) = g^{\circ 2}(x)$	1.0	64	0.6	1.0	116

$q(\vec{f}|\mathbf{W}) \geq 0$ for all $\vec{f}$ in $\mathcal{F}_{\mathbf{\Phi}}^{L}$. Similar results hold for the probability distributions of each component $f_{(i)}$.

OccamNet's sampling process involves independently sampling connections from each layer. Although each of OccamNet's layers represents an independent probability distribution, when sampling a function, the layers do not act independently. This is because the samples from layers closer to the outputs inform which of the sampled connections from previous layers are used. In particular, the full DAG that OccamNet samples has many disconnected components, and all components of the DAG which are not connected to any of the output nodes are effectively trimmed (See Figure 4-8). This is advantageous as it allows OccamNet to produce very different distributions of functions for different choices of connections in the final few layers, thereby allowing OccamNet to explore multiple classes of functions simultaneously.

Table 4.3: Implicit Functions: The proportion of 10 trials that converge to the correct implicit function for OccamNet and Eureqa. Image Recognition: The best accuracy from 10 trials for both OccamNet and the baseline. The baseline above the mid-line is HeuristicLab [Wagner et al., 2014], and the baseline below the mid-line is a feed-forward neural network with the same number of parameters as OccamNet. *sec.* is the average number of seconds for convergence. The baselines almost always finish much more quickly than OccamNet, so we do not provide baseline training times.

Implicit Functions					Image Recognition				
#	Target	OccamNet	sec.	Eureqa	#	Target	OccamNet	sec.	Baseline
1	$x_0 x_1 = 1$	1.0	294	1.0	6	MNIST Binary	92.9	150	92.8
2	$x_0^2 + x_1^2 = 1$	1.0	153	0.6	7	MNIST Trinary	59.6	400	81.2
3	$x_0 / \cos(x_1) = 1$	1.0	131	1.0	8	ImageNet Binary	70.7	400	78.0
4	$x_1 / x_0 = 1$	0.9	232	1.0	9	Backprop OccamNet	98.1	37	97.7
5	$m_1 v_1 - m_2 v_2 = 0$	1.0	270	0.0	10	Finetune ResNet	97.3	200	95.4

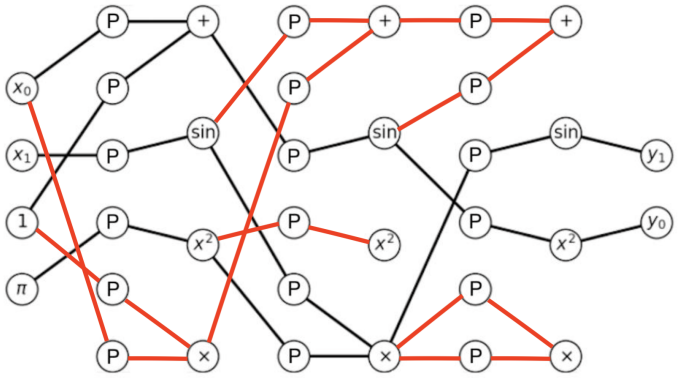

Figure 4-8: A demonstration of the dropped connections from sampled paths in OccamNet. All red paths are dropped from the final symbolic form of the sampled function because they are not directly connected to the outputs. These paths are unnecessarily computed during OccamNet's training process, leading to potential slowdowns in training.

As discussed in the main text, $q(\vec{f}|\mathbf{W})$ is the product of the probabilities of the sampled connections in $\vec{f}$'s DAG which are connected to the output nodes. However, in practice, we compute probabilities of functions in a feed-forward manner. This computation underestimates some probabilities; it actually computes an estimate $\tilde{q}(\vec{f}|\mathbf{W})$ of $q(\vec{f}|\mathbf{W})$.

To compute the probability of a given function, we assign each image and argument node a probability given this function's DAG. We denote the probability of the i'th node of the l'th image layer with $p_i^{(l)}$ and the probability of the i'th node of the l'th argument layer with $\tilde{p}_i^{(l)}$.

We propagate probabilities as follows. If the i'th image node in layer $l-1$ is connected to the j'th argument node in layer l, the probability of the j'th argument node in layer l is

$$\tilde{p}_j^{(l)} = p_i^{(l-1)} \cdot p_i^{(l,j)}(T). \tag{4.2}$$

The ith image node of the jth layer then has probability given by

$$p_i^{(l)} = \prod_{k=n+1}^{n+\alpha(\phi_i)} \tilde{p}_k^{(l)}, \quad n = \sum_{j=1}^{i-1} \alpha(\phi_j). \tag{4.3}$$

Finally, to calculate the probability of a function, we multiply the probabilities of the output nodes.

This algorithm computes function probabilities correctly unless a function's DAG has multiple nodes connecting to the same earlier node in the DAG. In this case, the probability of the earlier node is included multiple times in the final function probability, producing an estimate that is below the true probability of sampling the function.

In practice, we find that this biased evaluation of probabilities does not substantially affect OccamNet training. Note that when we equalize all functions to have the same probability (Section 4.2.6) or sample the highest probability function (Section 4.2.6), we do so with respect to the probability estimate $\tilde{q}$, not with respect to q. In this paper, we use q to mean $\tilde{q}$ unless otherwise specified.

Sampling from OccamNet Here we introduce the full mathematical formalism behind OccamNet. As described in the main text, we start from a predefined collection of N basis functions $\mathbf{\Phi} = \{\phi_i(\cdot)\}_{i=1}^N$. Each neural network layer is defined by two sublayers, the *arguments* and *image* sublayers. For a network of depth L, each of these sublayers is reproduced L times. Now let us introduce their corresponding hidden states: each l'th arguments sublayer defines a hidden state vector $\widetilde{\mathbf{h}}^{(l)}$, and

each l'th image sublayer defines a hidden state $\mathbf{h}^{(l)}$, as follows:

$$\widetilde{\mathbf{h}}^{(l)} = \left[\widetilde{h}_1^{(l)}, \ldots, \widetilde{h}_M^{(l)}\right], \quad \mathbf{h}^{(l)} = \left[h_1^{(l)}, \ldots, h_N^{(l)}\right], \tag{4.4}$$

where

$$M = \sum_{0 \leq k \leq N} \alpha(\phi_k)$$

and $\alpha(\phi)$ is the arity of function $\phi(\cdot, \ldots, \cdot)$. These vectors are related through the basis functions

$$h_i^{(l)} = \phi_i\left(\widetilde{h}_{j+1}^{(l)}, \ldots, \widetilde{h}_{j+\alpha(\phi_i)}^{(l)}\right), \quad j = \sum_{0 \leq k < i} \alpha(\phi_k). \tag{4.5}$$

This formally expresses how the arguments connect to the images in any given layer, visualized as the bold edges between sublayers in Figure 1 in the main paper. To complete the architecture and connect the images from layer l to the arguments of layer $(l+1)$, we use the described softmax transformation: [5]

$$\begin{aligned} \mathbf{W}(T) \cdot \mathbf{h}^{(l)} &= \mathsf{sample}\left(\begin{bmatrix} \mathsf{softmax}(\mathbf{w}_1; T)^\top \\ \vdots \\ \mathsf{softmax}(\mathbf{w}_{M_{l+1}}; T)^\top \end{bmatrix}\right) \begin{bmatrix} h_1^{(l)} \\ \vdots \\ h_{N_l}^{(l)} \end{bmatrix} \\ &\equiv \begin{bmatrix} \widetilde{h}_1^{(l+1)} \\ \vdots \\ \widetilde{h}_{M_{l+1}}^{(l+1)} \end{bmatrix} = \widetilde{\mathbf{h}}^{(l+1)}, \end{aligned} \tag{4.6}$$

where the hidden states $\boldsymbol{b}h^{(l)}$ and $\widetilde{\boldsymbol{b}h}^{(l+1)}$ have N_l and M_{l+1} coordinates, respectively, and where the sample function samples a one-hot row vector for each row based on the categorical probability distribution defined by $\mathsf{softmax}(\mathbf{w}; T)^\top$. In practice, we set T to a fixed, typically small, number. The last layer is usually set to a higher temperature to allow more compositionality. These sampled edges are encoded as

[5] as before, we define for any $\mathbf{z} = [z_1, \ldots, z_{N_l}]$ the softmax function as follows $\mathsf{softmax}(\mathbf{z}; T) := \left[\frac{\exp(z_1/T)}{\sum_{i=1}^{N_l} \exp(z_i/T)}, \ldots, \frac{\exp(z_{N_l}/T)}{\sum_{i=1}^{N_l} \exp(z_i/T)}\right]^\top$

sparse matrices, through which a forward pass evaluates $\vec{f}$.

It is also possible to implement OccamNet without the sampling part of the propagation. In this case, the softmax of the weight matrices is treated as the weights of linear layers, and we minimize the MSE loss between the outputs and the desired outputs. In practice, however, we find that this approach is less interpretable and often converges to suboptimal local minima.

As shown in Figure 4-9, we use skip connections similar to those in DenseNet [Huang et al., 2017] and ResNet [He et al., 2016b], concatenating image states with those of subsequent layers. Skip connections yield several desirable properties: (i) The depth of equations is not fixed, lifting the requirement that the number of layers of the solution be known in advance. (ii) The network can find compact solutions as it considers all levels of composition. This promotes solution sparsity and interpretability. (iii) Primitives in shallow layers can be reused, analogous to feature reuse in DenseNet. (iv) Subsequent layers may behave as higher-order corrections to the solutions found in early layers. Additionally, if we implement OccamNet without sampling, shallow layers are trained before or alongside the subsequent layers due to more direct supervision because gradients can propagate to shallow layers more easily to avoid exploding or vanishing gradients.

From Equation equation 4.5, we see that $M_{l+1} = M = \sum_{0 \leq k \leq N} \alpha(\phi_k)$. If no skip connections are used, $N_l = N = |\mathbf{\Phi}|$. If skip connections are used, however, N_l grows as l increases. We demonstrate how the scaling grows as follows. Let u be the number of inputs and v be the number of outputs. When learning connections from images to arguments at layer l $(1 \leq l \leq L)$, there will be skip connections from the images of the previous $l-1$ layers $1, \ldots, l-1$. Hence the ith layer has an image size of $u + (i+1)N$, as shown in Figure 4-9. We learn linear layers from these images to arguments, and the number of arguments is always M. Thus, in total, we have the following number of parameters:

$$v(u + (L+1)N) + M \sum_{i=0}^{L-1} (u + (i+1)N) \in O(NML^2).$$

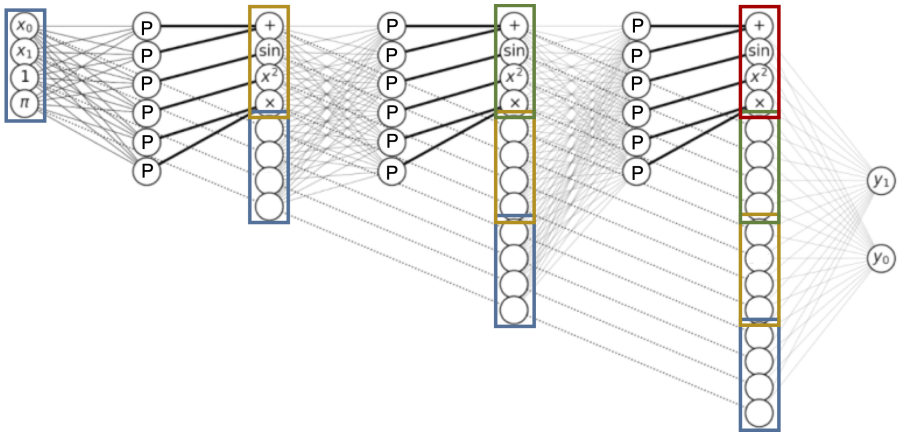

Figure 4-9: Skip connections. Dotted lines and color: the origin of the reused neurons.

Note that in the above discussion we assume that M remains constant. However, to be able to represent all functions up to a particular depth, we must repeat bases in earlier layers, causing M to grow exponentially. For small numbers of layers, this is not problematic, and one can always increase the number of layers to generate more functions.

Along with the added inputs and constants, this description fully specifies the mathematical structure of our architecture.

Initialization We originally initialized all model weights to 1. However, this initializes complex functions, which have DAGs with many more edges than simple functions, to low probabilities. As a result, we found in practice that the network sometimes struggled to converge to complex functions with high fitness $K(\mathcal{M}, f)$ because their initial low probabilities meant that they were sampled far less often than simple functions. This is because even if complex functions have a higher probability increase than simple functions when they are sampled, the initial low probabilities caused the complex functions to be sampled far less and to have an overall lower expected probability increase.

To address this issue, we use a second initialization algorithm, which initializes all functions to equal probability.

This initialization algorithm iterates through the layers of the network. It estab-

lishes as an invariant that, after assigning the weights up to the lth layer, all paths leading to a given node in the lth argument layer have equal probabilities. Then, each argument layer node has a unique corresponding probability, the probability of all paths up to that node. We denote the probability of the ith node in the lth argument sublayer as $\widetilde{p}_i^{(l)}$. Because each argument layer node has a corresponding probability, each image layer node must also have a unique corresponding probability, which, for the ith node in the lth image sublayer, we denote as $p_i^{(l)}$. These image layer probabilities are given by

$$p_i^{(l)} = \prod_{k=n+1}^{n+\alpha(\phi_i)} \widetilde{p}_k^{(l)}, \quad n = \sum_{j=1}^{i-1} \alpha(\phi_j). \tag{4.7}$$

Our algorithm initializes the input image layer's nodes to probability 1. As the algorithm iterates through all subsequent T-Softmax layers, the invariant established above provides a system of linear equations involving the desired connection probabilities, which the algorithm solves. The algorithm groups the previous image layer according to the node probabilities, obtaining a set of ordered pairs $\{(p'^{(l)}_i, n_i^{(l)})\}_{i=1}^k$ representing $n_i^{(l)}$ nodes with probability $p'^{(l)}_i$ in the lth layer. Note that if two image nodes have the same probability, for each P-node in the arguments sublayer, the edges between the image nodes and the P-node must have the same probability in order to satisfy the algorithm's invariant. Then, we define $p'^{(l,j)}_i$ as the probability of the edges between the image nodes with probability $p'^{(l)}_i$ and the jth argument P-node of the lth layer. The probabilities of the edges to a given P-node sum to 1, so for each j, we must have $\sum_i n_i p'^{(l,j)}_i = 1$. Further, the algorithm requires that the probability of a path to a P-node through a given connection is the same as the probability of a path to that P-node through any other connection. The probability of a path to the jth P-node through a connection with probability $p'^{(l,j)}_i$ is $p'^{(l)}_i p'^{(l,j)}_i$, so we obtain the equations $p'^{(l)}_0 p'^{(l,j)}_0 = p'^{(l)}_i p'^{(l,j)}_i$, for all i and j. These two constraints give the vector

equation

$$\begin{bmatrix} n_0^{(l)} & n_1^{(l)} & n_2^{(l)} & \cdots & n_k^{(l)} \\ p'^{(l)}_0 & -p'^{(l)}_1 & 0 & \cdots & 0 \\ p'^{(l)}_0 & 0 & -p'^{(l)}_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p'^{(l)}_0 & 0 & 0 & \cdots & -p'^{(l)}_k \end{bmatrix} \begin{bmatrix} p'^{(l,j)}_0 \\ p'^{(l,j)}_1 \\ p'^{(l,j)}_2 \\ \vdots \\ p'^{(l,j)}_k \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

for all $1 \leq j \leq M$. The algorithm then solves for each $p'^{(l,j)}_i$.

After determining the desired probability of each connection of the lth layer, the algorithm computes the SPL weights $\mathbf{w}'^{(l,j)}$ that produce the probabilities $p'^{(l,j)}_i$. Since there are infinitely many possible weights that produce the correct probabilities, the algorithm sets $w'^{(l,j)}_0 = 0$. Then, the algorithm uses the softmax definition of the edge probabilities to determine the required value of $\sum_{m=1}^{k} \exp(w'^{(l,j)}_i / T^{(l)})$:

$$\begin{aligned} p'^{(l,j)}_0 &= \frac{\exp(w'^{(l,j)}_0 / T^{(l)})}{\sum_{m=1}^{k} \exp(w'^{(l,j)}_m / T^{(l)})} \\ &= \frac{1}{\sum_{m=1}^{k} \exp(w'^{(l,j)}_m / T^{(l)})} \end{aligned}$$

so

$$\sum_{m=1}^{k} \exp(w'^{(l,j)}_m / T) = 1 / p'^{(l,j)}_0.$$

Substituting this equation into the expression for the other probabilities gives

$$\begin{aligned} p'^{(l,j)}_i &= \exp(w'^{(l,j)}_i / T^{(l)}) / \left(\sum_{m=1}^{k} \exp(w_i / T^{(l)}) \right) \\ &= p'^{(l,j)}_0 \exp(w'^{(l,j)}_i / T^{(l)}). \end{aligned}$$

Solving for $w'^{(l,j)}_i$ gives

$$w'^{(l,j)}_i = T^{(l)} \log \left(p'^{(l,j)}_i / p'^{(l,j)}_0 \right),$$

which the algorithm uses to compute $w'^{(l,j)}_i$.

After determining the weights $w'^{(l,j)}_i$ the algorithm assigns them to the correspond-

ing $w_i^{(l,j)}$. In particular, if the ith image node has probability $p'^{(l)}_k$, the weights of edges to the ith node are given by $w_i^{(l,j)} = w'^{(l,j)}_k$, for all j. The algorithm then determines the values of $\widetilde{p}_i^{(l+1)}$, given by $\widetilde{p}_i^{(l+1)} = p_1^{(l)} p_1^{(l,i)}$. Finally, the algorithm determines $p_i^{(l+1)}$ using Equation 4.7 and repeats the above process for subsequent layers until it reaches the end of the network.

This algorithm efficiently equalizes the probabilities of all functions in the network. In practice, however, we find that perfect equalization of functions causes activation functions with two inputs to be highly explored. This is because there are many more possible functions containing activation functions with two inputs than with one input. Additionally, as mentioned in Section 4.2.6, in this section we have implicitly been using the approximate probability $\tilde{q}$. This probability underestimates many functions that include activation functions with two or more inputs because these functions are those which can use a node multiple times in their DAG. As a result, although all functions will have an equal $\tilde{q}$, some functions with multiple inputs will have larger q than other functions, and q is what determines the probability of being sampled. In practice, therefore, we find that a balance between initializing all weights to one and initializing all functions to equal probability is most effective for exploring all types of functions.

To implement this balance, we create an equalization hyperparameter, E. If $E = 0$, we initialize all weights to 1 as in the original OccamNet architecture. If $E \neq 0$, we use the algorithm presented above to initialize the weights and then divide all of the weights by E. For $E > 1$, this has the effect of initializing weights between the two initialization approaches. In practice, we find that values of $E = 1$ and $E = 5$ are most effective for exploring all types of functions (See Section 4.2.6).

Function Selection As discussed in the main text, after training using a sampling strategy, the network selects the function $\vec{f}$ with the highest probability $q(\vec{f}|\mathbf{W})$.

We develop a dynamic programming algorithm that determines the DAG with the highest probability. The algorithm steps sequentially through each argument layer, and at each argument layer it determines the maximum probability path to each

argument node. Knowing the maximum probability paths to the previous argument layer nodes allows the algorithm to easily determine the maximum probability paths to the next argument layer.

As with the network initialization algorithm, the function selection algorithm associates the ith P-node of the lth argument sublayer with a probability, $\widetilde{p}_i^{(l)}$, which represents the highest probability path to that node. Similarly, we let $p_i^{(l)}$ represent the assigned probability of the ith node of the lth image sublayer, defined as the highest probability path to a given image node. $p_i^{(l)}$ can once again be determined from $\widetilde{p}_i^{(l)}$ using Equation 4.7. Further, the algorithm associates each node with a function, $\widetilde{f}_i^{(l)}$ for argument nodes and $f_i^{(l)}$ for image nodes, which represents the highest probability function to the corresponding node. Thus, $\widetilde{f}_i^{(l)}$ has probability $\widetilde{p}_i^{(l)}$, and $f_i^{(l)}$ has probability $p_i^{(l)}$. Further, $f_i^{(l)}$ is determined from $\widetilde{f}_i^{(l)}$ using

$$f_i^{(l)}(\vec{x}) = \phi_i\left(\widetilde{f}_{n+1}^{(l)}(\vec{x}), \ldots, \widetilde{f}_{n+\alpha(\phi_j)}^{(l)}(\vec{x})\right), \quad n = \sum_{j=1}^{i-1} \alpha(\phi_j). \tag{4.8}$$

The algorithm iterates through the networks layers. At the lth layer, it determines the maximum probability path to each argument node, computing

$$\widetilde{p}_i^{(l+1)} = \text{MAX}\left(p_0^{(l)} p_0^{(l,i)}, \ldots, p_N^{(l)} p_N^{(l,i)}\right)$$

$$\widetilde{f}_i^{(l+1)} = \begin{cases} f_0^{(l)} & \text{if } \widetilde{p}_i^{(l+1)} = p_0^{(l)} p_0^{(l,i)} \\ f_1^{(l)} & \text{if } \widetilde{p}_i^{(l+1)} = p_1^{(l)} p_1^{(l,i)} \\ \vdots & \vdots \\ f_N^{(l)} & \text{if } \widetilde{p}_i^{(l+1)} = p_N^{(l)} p_N^{(l,i)} \end{cases}.$$

Next, it determines the maximum probability path up to each image node, computing $p_i^{(l+1)}$ and $f_i^{(l+1)}$ using Equations 4.7 and 4.8, respectively. The algorithm repeats this process until it reaches the output layer, at which point it returns $\vec{f}_{\max} = [\widetilde{f}_1^{(L)}, \ldots, \widetilde{f}_N^{(L)}]^\top$ and $p_{\max} = \prod_{i=1}^N \widetilde{p}_i^{(L)}$.

An advantage of this process is that identifying the highest probability function

is just as efficient as sampling functions. In particular, the complexity at each layer is $O(MN_i)$, leading to an overall complexity of $O(NML^2)$ if skip connections are included.

Loss We train our network on mini-batches of data to provide flexibility for devices with various memory constraints. Consider a mini-batch $\mathcal{M} = (X, Y)$, and a sampled function from the network $\vec{f}(\cdot) \sim q(\cdot|\mathbf{W})$. We compute the *fitness* of each $f_{(i)}(\cdot)$ with respect to a training pair $(\vec{x}, \vec{y})$ by evaluating the likelihood

$$k_i\left(f_{(i)}(\vec{x}), \vec{y}\right) = (2\pi\sigma^2)^{-1/2} \exp(-\left[f_{(i)}(\vec{x}) - (\vec{y})_i\right]^2 / (2\sigma^2)),$$

which is a Normal distribution with mean y and variance σ^2, and measures how close $f_{(i)}(\vec{x})$ is to the target $(\vec{y})_i$. The likelihood can be also viewed as a Bayesian posterior with a noninformative prior. The total fitness is determined by summing over the entire mini-batch: $K_i\left(\mathcal{M}, f_{(i)}\right) = \sum_{(\vec{x},\vec{y})\in\mathcal{M}} k_i\left(f_{(i)}(\vec{x}), \vec{y}\right)$.

The variance of $k_i\left(f_{(i)}(\vec{x}), \vec{y}\right)$ characterizes the fitness function's smoothness. As $\sigma^2 \to 0$, the likelihood is a delta function with nonzero fitness for some $(\vec{x}, \vec{y})$ only if $f_{(i)}(\vec{x}) = (\vec{y})_i$. Similarly, a large variance characterizes a fitness in which potentially many solutions provide accurate approximations, increasing the risk of convergence to local minima. In the former case, learning becomes harder as few $f_{(i)}(\cdot)$ out of exponentially many sampleable functions result in any signal, whereas in the latter case learning might not converge to the optimal solution. We let σ^2 be a network hyperparameter, tuned for the tradeoff between ease of learning and solution optimality for different tasks.

Similar to Petersen et al. [2021], we use a loss function for backpropagating on the weights of $q(\cdot|\mathbf{W})$:

$$H_{q_i}[f_{(i)}, \mathbf{W}, \mathcal{M}] = -K_i\left(\mathcal{M}, f_{(i)}\right) \cdot \log\left[q_i(f_{(i)}|\mathbf{W})\right]. \tag{4.9}$$

We can interpret equation 4.9 as the cross-entropy of the posterior for the target and the probability of the sampled function $f_{(i)}$. If the sampled function $f_{(i)}$ is close to

$f^*_{(i)}$, then $K_i(\mathcal{M}, f_{(i)})$ will be large, and the gradient update below will also be large:

$$\nabla_{\mathbf{W}} H_{q_i}\left[f_{(i)}, \mathbf{W}, \mathcal{M}\right] = -\frac{\nabla_{\mathbf{W}} q_i(f_{(i)}|\mathbf{W})}{q_i(f_{(i)}|\mathbf{W})} K_i\left(\mathcal{M}, f_{(i)}\right). \tag{4.10}$$

The first term on the right-hand side (RHS) of update equation 4.10 increases the likelihood of the function $f_{(i)}$. The second term on the RHS is maximal when $f_{(i)} \equiv f^*_{(i)}$. Importantly, the second term approaches zero as $f_{(i)}$ deviates from $f^*_{(i)}$. If the sampled function is far from the target, then the likelihood update is suppressed by $K_i(\mathcal{M}, f_{(i)})$. Therefore, we only optimize the likelihood for functions close to the target. Note that in equation 4.10 we backpropagate only through the probability of the function $f_{(i)}$ given by $q_i\left(f_{(i)}|\mathbf{W}\right)$, whose value *does not* depend on the bases in $\mathbf{\Phi}$, implying that the bases can be non-differentiable. This is particularly useful for applications requiring non-differentiable basis functions. Furthermore, this loss function allows non-differentiable regularization terms, which greatly expands the regularization possibilities.

Sample-based Training We use a sampling-based strategy to update our model, explained below. This training procedure was first proposed in Risk-Seeking Policy Gradients [Petersen et al., 2021]. We denote $\mathbf{W}^{(t)}$ as the set of weights at training step t, and we fix two hyperparameters: R, the number of functions to sample at each training step, and λ, or the *truncation parameter*, which defines the number of the R paths chosen for optimization via equation 4.10. We initialize $\mathbf{W}^{(0)}$ as described in Section 4.2.2. We then proceed as follows:

1. Sample R functions $\vec{f}_1, \ldots, \vec{f}_R \sim q(\cdot|\mathbf{W}^{(t)})$. We denote the jth output of $\vec{f}_i$ as $f_{i(j)}$.

2. For each output j, sort $f_{i(j)}$ from greatest to least value of $K_j\left(\mathcal{M}, f_{i(j)}\right)$ and select the top λ functions, yielding a total of $v\lambda$ selected functions $g_{1,j}, \ldots, g_{\lambda,j}$. The total loss is then given by $\sum_{i=1}^{\lambda}\sum_{j=0}^{v-1} H_{q_j}[g_{i,j}, \mathbf{W}, \mathcal{M}]$, which yields the training step gradient update:

$$-\sum_{i=1}^{\lambda}\sum_{j=0}^{v-1} \frac{\nabla_{\mathbf{W}} q_j(g_{i,j}|\mathbf{W})}{q_j(g_{i,j}|\mathbf{W})}\, K_j(\mathcal{M}, g_{i,j}). \tag{4.11}$$

Notice that through equation 4.11 we have arrived at a modified REINFORCE update [Williams, 1992], where the policy is $q_i(\cdot|\cdot)$ and the regret is the fitness $K_i(\cdot,\cdot)$.

3. Perform the gradient step equation 4.11 on $\mathbf{W}^{(t)}$ for all selected paths to obtain $\mathbf{W}^{(t+1)}$. In practice, we find that the Adam algorithm [Kingma and Ba, 2015] works well.

4. Set $t = t + 1$ and repeat from Step 1 until a stop criterion is met.

Note that Equations equation 4.10 and equation 4.11 represent different objective functions. The benefit of using Equation equation 4.11 is that accumulating over the top $v\lambda$ best fits to the target allows for explorations of function compositions that contain desired components but are not fully developed. In practice, we find that reweighting the importance of the top-$v\lambda$ routes, substituting $K'_j(\mathcal{M}, g_{i,j}) = K_j(\mathcal{M}, g_{i,j})/i$, improves convergence speed by biasing updates towards the best routes.

Constant Fitting In some cases, we may wish to fit functions that involve constants that are not known *a priori*. To fit such undetermined constants, we use activation functions with unspecified constants, such as x^c and $c \cdot x$ (c is undefined). We then combine the training process described in Section 4.2.6 with a constant fitting training process.

The two-step training process works as follows: We first sample a batch $\mathcal{M}$ and a function batch $(\vec{f}_1, \ldots, \vec{f}_R)$. Next, for each function $\vec{f}_i$, we fit the unspecified constants to $\mathcal{M}$ in $\vec{f}_i$ using gradient descent. Any other constant optimization method would also work. Finally, we update the network weights according to Section 4.2.6, using the fitness K of the constant-fitted function batch. To increase training speed, we store each function's fitted constants for reuse.

Recurrence OccamNet can also be trained to find recurrence relations, which is of particular interest for programs that rely on FOR or WHILE loops. To find such

recurrence relations, we assume a *maximal* recursion D. We use the following notation for recurring functions: $f^{\circ(n+1)}(x) \equiv f^{\circ n}(f(x))$, with base case $f^{\circ 1}(x) \equiv f(x)$.

To augment the training algorithm, we first sample $(\vec{f}_1, \ldots, \vec{f}_R) \sim q(\cdot|\mathbf{W}^{(t)})$. For each $\vec{f}_i$, we compute its recurrence to depth D as follows $\left(\vec{f}_i^{\circ 1}, \vec{f}_i^{\circ 2}, \ldots, \vec{f}_i^{\circ D}\right)$, obtaining a collection of RD functions. Training then continues as usual; we compute the corresponding $K_j(\mathcal{M}, \vec{f}_{i(j)}^{\circ n})$, select the best $v\lambda$, and update the weights. It is important to note that we consider all depths up to D since our maximal recurrence depth might be larger than the one for the target function.

Note that we do not change the network architecture to accommodate for recurrence depth $D > 1$. As described in the main text, we can efficiently use the network architecture to evaluate a sampled function $\vec{f}(\vec{x})$ for a given batch of $\vec{x}$. To incorporate recurrence, we take the output of this forward pass and feed it again to the network D times, similar to a recurrent neural network. The resulting outputs are evaluations $\left(\vec{f}_i^{\circ 1}(\vec{x}), \vec{f}_i^{\circ 2}(\vec{x}), \ldots, \vec{f}_i^{\circ D}(\vec{x})\right)$ for a given batch of $\vec{x}$.

Regularization As discussed in the main text, to improve implicit function fitting, we implement a regularized loss function,

$$K_i'(\mathcal{M}, f) = K_i(\mathcal{M}, f) - s \cdot r[f],$$

for some regularization function r, where $s = n(\mathcal{M})/\sqrt{2\pi\sigma^2}$ is the maximum possible value of $K_i(\mathcal{M}, f)$. We define

$$r[f] = w_\phi \cdot \phi[f] + w_\psi \cdot \psi[f] + w_\xi \cdot \xi[f] + w_\gamma \cdot \gamma[f],$$

where $\phi[f]$ measures trivial operations, $\psi[f]$ measures trivial approximations, $\xi[f]$ measures the number of constants in f, $\gamma[f]$ measures the number of activation functions in f, and w_ϕ, w_ψ, w_ξ, and w_γ, are weights for their respective regularization terms. We now discuss each of these regularization terms in more detail.

The Phi Term The $\phi[f]$ term measures whether the unsimplified form of f contains trivial operations, operations that cause an expression to simplify. For example, division is a trivial operation in x/x, because the expression simplifies to 1. Similarly, $1 \cdot x$, x^1, and x^0 are all trivial operations. We punish these trivial operations because they often produce constant outputs without ever adding meaning to an expression.

To detect trivial operations, we employ two procedures. The first uses the `SymPy` package [Meurer et al., 2017] to simplify f. If the simplified expression is different from the original expression, then there are trivial operations in f, and this procedure returns 1. Otherwise the first procedure returns 0. Unfortunately, the `SymPy` `==` function to test if functions are equal often incorrectly indicates that nontrivial functions are trivial. For example, `SymPy`'s `simplify` function, which we use to test if a function can be simplified, converts $x + x$ to $2 \cdot x$, and the `==` function states that $x + x \neq 2 \cdot x$. To combat this, we develop a new function, `sympyEquals` which corrects for these issues with `==`. The `sympyEquals` is equivalent to `==`, except that it does not take the order of terms into account, and it does not mark expressions such as $x + x$ and $x \cdot x$ as unsimplified. We find that this greatly improves function fitting.

The constant fitting procedure often produces functions that only differ from a trivial operation because of imperfect constant fitting, such as $f(x_0) = x_0^{0.0001}$, which is likely meant to represent x_0^0. `SymPy`, however, will not mark this function as trivial. The second procedure addresses this issue by counting the constant activations, such as $x_0^{0.0001}$, $1.001 \cdot x_0$, and $x_0 + 0.001$, which approximate trivial operations. For the activation function $f(x) = x + c$, if the fitted c satisfies $-0.1 < c < 0.1$, the procedure adds 1 to its counter. Similarly, for the activation functions $f(x) = cx$ and $f(x) = x^c$, if the fitted c satisfies $-0.1 < c < 0.1$ or $0.9 < c < 1.1$, the procedure adds 1 to its counter. We select these ranges to capture instances of imperfect constant fitting without labeling legitimate solutions as trivial. After checking all activation functions used, the procedure returns the counter.

The $\phi[f]$ term returns the sum of the outputs of the first and second procedures. We find that a weight of $w_\phi \approx 0.7$ for $\phi[f]$ is most effective in our loss function. This value of w_ϕ ensures that most trivial f have $K_i(\mathcal{M}, f) - s \cdot w_\phi \cdot \phi[f] < 0$, thus actively

reducing the weights corresponding to functions with trivial operations, without over punishing functions and hindering learning.

The Psi Term When punishing trivial operations using the ϕ term, we find that the network discovers many nontrivial operations which very closely approximate trivial functions by exploiting portions of functions with near-zero derivatives, which can be used to artificially compress data. For example, $\cos(x/2)$ closely approximates 1 if $-1 < x < 1$. Unfortunately, it is often difficult to determine if a function approximates a trivial function simply from its symbolic representation. This issue is also identified in [Schmidt and Lipson, 2010].

To detect these trivial function approximations, we develop an approach that analyzes the activation functions' outputs *during* the forward pass. The $\psi[f]$ term counts the number and severity of activation functions which, during a forward pass, the network identifies as possibly approximating trivial solutions. For each basis function, the network stores values around which outputs of that function often cluster artificially. Table 4.4 lists the bases which the network tests for clustering.

The procedure for determining ψ is as follows. The algorithm begins with a counter of 0. During the forward pass, if the network reaches a basis function ϕ listed in Table 4.4, the algorithm tests each ordered tuple (ϕ, a, δ) from Table 4.4, where a is the point tested for clustering and δ is the clustering tolerance. If the mean of all the outputs of the basis function, $\bar{y}$, for a given batch satisfies $|\bar{y} - a| < \delta$, the algorithm adds $\min(5, 0.1/|\bar{y} - a|)$ to the counter. These expressions increase with the severity of clustered data; the more closely the outputs are clustered, the higher the punishment term. The minimum term ensures that $\psi[f]$ is never infinite.

We also test for the approximation $\sin(x) \approx x$ by testing the inputs and outputs of the sine basis function. If the inputs and outputs x and y to the sine basis satisfy $\overline{|y - x|} < 0.1$, the algorithm adds $\min(5, 0.05/\overline{|y - x|})$ to the counter. In the future, we plan to consider more approximations similar to the small angle approximation.

$\psi[f]$ should not artificially punish functions involving the bases listed in Table 4.4 that are not trivial approximations because no proper use of these basis functions

Table 4.4: Basis functions tested for clustering

Basis Functions	Cluster Points	Cluster Tolerance
$(\cdot)^2$	$\{0\}$	0.25
$(\cdot)^3$	$\{0\}$	0.25
$\sin(\cdot)$	$\{1, -1\}$	0.25
$\cos(\cdot)$	$\{1, -1\}$	0.25
$(\cdot)^c$	$\{1\}$	0.5

will always produce outputs very close to the clustering points. Because $\psi[f]$ flags functions based on their batch outputs, each batch will likely give different outcomes. This allows $\psi[f]$ to better discriminate between trivial function approximations and nontrivial operations: $\psi[f]$ should flag trivial function approximations often, but it should only flag nontrivial operations rarely when the inputs statistically fluctuate to produce clustered outputs. In practice, we find that a weight of $w_\psi \approx 0.3$ for $\psi[f]$ is most effective in our loss function.

The Xi Term When our network converges to the correct solution, it may converge to a more complicated expression equivalent to the desired expression. To promote simpler expressions, we slightly punish functions based on their complexity. The $\xi[f]$ term counts the number of activation functions used to produce f, which serves as a measure of f's complexity. We find that a small weight of $w_\xi \approx 0.1$ for $\xi[f]$ is most effective in our loss function. This small value has little significance when distinguishing between a function that fits a dataset well and a function that does not, but it is enough to promote simpler functions over complex functions when they are otherwise equivalent.

The Gamma Term The $\gamma[f]$ term also punishes functions for their complexity. The $\gamma[f]$ term counts the number of constants in f, which, like the number of activation functions, serves as a metric for f's complexity. We find that a weight of $w_\gamma \approx 0.15$ for $\gamma[f]$ is most effective in our loss function. Just as with $\xi[f]$, this small value has little significance when distinguishing between a function that fits a dataset well and a function that does not, but it is enough to slightly promote simpler func-

tions over complex functions when they are otherwise equivalent. We weight $\gamma[f]$ slightly higher than $\xi[f]$ because many functions with constants can be simplified.

OccamNet with Units

Although we do not use this feature in our experiments, we also allow users to provide units for inputs and outputs. OccamNet will then regularize its functions so that they preserve the desired units.

To determine if a function f preserves units, we first encode the units of each input and output. We encode an input parameter's units as a NumPy array in which each entry represents the power of a given base unit. For example, if for a problem the relevant units are kg, m, and s, and we have an input F with units $\mathrm{kg} \cdot \mathrm{m/s^2}$, we would represent F's units as $[1, 1, -2]$.

We then feed these units through the sampled function. Each basis function receives a set of variables with units, may have requirements on those units for them to be consistent, and returns a new set of units. For example, $\sin(\cdot)$ receives one variable which it requires to have units $[0, \ldots, 0]$ and returns the units $[0, \ldots, 0]$. Similarly, $+(\cdot, \cdot)$ takes two variables with units that it requires to be equal and returns the same units. Using these rules, we propagate units through the function until we obtain units for the output. If at any point the input units for a basis function do not meet that basis's requirements, that basis returns $[\infty, \ldots, \infty]$. Any basis functions that receive $[\infty, \ldots, \infty]$ also return $[\infty, \ldots, \infty]$. Finally, if the output units of f do not match the desired output units (including if f outputs $[\infty, \ldots, \infty]$), we mark f as not preserving units.

For the multiplication by a constant basis function, $\cdot c$, we have to be careful. Because the units of c are unspecified, this basis function can produce any output units. As such, it returns $[\mathrm{NaN}, \ldots, \mathrm{NaN}]$. If any basis function receives $[\mathrm{NaN}, \ldots, \mathrm{NaN}]$, it will either return $[\mathrm{NaN}, \ldots, \mathrm{NaN}]$ if it has no constraints on the input units, or it will treat the $[\mathrm{NaN}, \ldots, \mathrm{NaN}]$ as being the units required to meet the basis function's consistency conditions. For example, if the $\sin(\cdot)$ function receives $[\mathrm{NaN}, \ldots, \mathrm{NaN}]$, it will treat the input as $[0, \ldots, 0]$, and if the $+(\cdot, \cdot)$ function receives $[1, 2, 3]$ and

$[\text{NaN}, \text{NaN}, \text{NaN}]$, it will return $[1, 2, 3]$.

After sampling functions from OccamNet, we determine which functions do not preserve units. Because we wish to avoid these functions entirely, we bypass evaluating their normal fitness (thereby saving compute time) and instead assign a fitness of $K_i'(\mathcal{M}, f) = -w_{\text{units}} s$, where $s = n(\mathcal{M})/\sqrt{2\pi\sigma^2}$ is the maximum possible value of $K_i(\mathcal{M}, f)$ and w_{units} is a hyperparameter that can be tuned (set to 1 by default).

Functions with Undefined Outputs One difficulty that may arise when training OccamNet is that many sampled functions are undefined on the input data range. Two cases of undefined functions are: 1) the function is undefined on part of the input data range for all values of a set of constants, or 2) the function is only undefined when the function's constants take on certain values. An example function satisfying case 1 is $f_1(x_0) = c_0/(x_0 - x_0)$, which divides by 0 regardless of the value of c_0. An example function satisfying case 2 is $f_2(x_0) = x_0^{c_0}$, which is undefined whenever x_0 is negative and c_0 is not an integer.

In the first case, the network should abandon the function. In the second case, the network should try other values for the constants. However, the network cannot easily determine which case an undefined function satisfies. To balance both cases, if the network obtains an undefined result, such as `NaN` or `inf`, for the forward pass, the network tests up to 100 more randomized sets of constants. If none of these attempts produce defined results, the network returns the array of undefined outputs. For example, with $c_0/(x_0 - x_0)$, the network tests a first set of constants, determines that they produce an undefined output, and tests 100 more constants. None of these functions are defined on all inputs, so the network returns the undefined outputs.

In contrast, with $x_0^{c_0}$, the network might find that the first set of constants produces undefined outputs, but after 20 retries, the network might discover that $c_0 = 2$ produces a function defined on all inputs. The network will then perform gradient descent and return the fitted value of c_0. Further, if at any point in the gradient descent, the forward pass yields undefined results, the network returns the well-defined constants and associated output from the previous forward pass. For example, for

$x_0^{c_0}$, after the network discovers that $c_0 = 2$ works, the gradient for the constants will be undefined because c_0 can only be an integer. Thus, the network will return the outputs of $x_0^{c_0}$, for $c_0 = 2$, before the undefined gradients.

We find that if the network simply ignores functions with undefined outputs, these functions will increase in probability because our network regularization punishes many other functions. Since these punished functions decrease in probability during training, the functions with undefined outputs begin to increase in probability. To combat this, instead of ignoring undefined functions, we use a modified fitness for undefined functions, $K_i'(\mathcal{M}, f) = -w_{\text{undef}} s$, where w_{undef} is a hyperparameter that can be tuned. This punishes undefined functions, causing their weights to decrease. In practice, we find that a value of w_{undef} between 0 and 1 is most effective, depending on the application.

Experimental Setup

We divide this section as follows:

1. Section 4.2.6 describes the hyperparameters used for the experiments described in the main text testing OccamNet on Analytic Functions, Non-Analytic Functions, Implicit Functions, and Image Recognition.

2. Section 4.2.6 describes the experimental setup for our tests with the PMLB datasets.

Experimental Setup and Hyperparameters for Non-PMLB Experiments

For the non-PMLB experiments, we terminate learning when the top-$v\lambda$ sampled functions all return the same fitness $K(\cdot, f)$ for 30 consecutive epochs. If this happens, these samples are equivalent function expressions.

Computing the most likely DAG allows retrieval of the final expression. If this final expression matches the correct function, we determine that the network has converged. For pattern recognition, there is no correct target composition, so we measure the accuracy of the classification rule on a test split, as is conventional. Note

that in the experiments where $E = 0$, we instead take an approximate of the highest probability function by taking the argmax of the weights into each argument node.

In all experiments, if termination is not met in a set number of steps, we consider it as not converged. We also keep a constant temperature for all the layers except for the last one. An increased last layer temperature allows the network to explore higher function compositionality, as shallow layers can be further trained before the last layer probabilities become concentrated; this is particularly useful for learning functions with high degrees of nesting. More details on hyperparameters for experiments are in the SM. Our network converges rapidly, often in only a few seconds and at most a few minutes.

In Tables 4.5 and 4.6, we present and detail the hyperparameters we used for our experiments in the main paper. Note that detail about the setup for each experiment is provided in the attached code.

In Tables 4.5 and 4.6, $+$ is addition (2 arguments); $-$ is subtraction (2 arguments) $\cdot$ is multiplication (2 arguments); / is division (2 arguments); $\sin(\cdot)$ is sine, $+c$ is addition of a constant, $\cdot c$ is multiplication of a constant, $(\cdot)^c$ is raising to the power of a constant, $\leq$ is an if-statement (4 arguments: comparing two numbers, one return for a true statement, and one for a false statement); $-(\cdot)$ is negation. MIN, MAX, and XOR all have two arguments. Here, SIGMOID$'$ is a sigmoid layer, and $\tanh'$ is a tanh layer where the inputs to both functions are scaled by a factor of 10, $+_4$, and $+_9$ are the operations of adding 4 and 9 numbers respectively, and MAX_4, MIN_4, MAX_9 and MIN_9 are defined likewise. The bases for pattern recognition experiments are given as follows: $\mathbf{\Phi}_A$ consists of SIGMOID$'$, SIGMOID$'$, $\tanh'$, $\tanh'$, $+_4$, $+_4$, $+_9$, $+_9$, $+$, $+$, MIN, MIN, MAX and MAX; $\mathbf{\Phi}_B$ consists of id, id, id, id, $+$, $+$, $+$, $+_4$, $+_4$, $+_9$, $+_9$, $+_9$, tanh,, tanh, SIGMOID, and SIGMOID. Additionally, the constants used for pattern recognition are $\mathbf{C} = \{-1, -1, 0, 0, 1, 1, 1\}$.

In Tables 4.5 and 4.6, L is the depth, T is the temperature, T_{last} is the temperature of the final layer, σ is the variance, R is the sample size, λ is the fraction of best fits, α is the learning rate, E is the initialization parameter described in Section 4.2.6, and w_ϕ, w_ψ, w_ξ, and w_γ are as defined in Appendix 4.2.6. Table 4.5 does not include E as

a listed hyperparameter because for all experiments listed, $E = 0$. With * we denote the experiments for which the best model is without skip connections. We do not regularize for any experiments in Table 4.5. NA entries mean that the corresponding hyperparameter is not present in the experiment. Note that the first three equations in Table 4.6 are not discussed in the main text. Instead, they are smaller experiments that we performed and which we discuss in the SM.

For all experiments in Table 4.6, we use a learning rate of 0.01 and, when applicable, a constant-learning rate of 0.05. We also set the temperature to 1 and the final layer temperature to 10 for all experiments in the table. For the equation $m_1 v_1 - m_2 v_2 = 0$, we sample m_1, v_1, and m_2 from $[-10, 10]$ and compute v_2 using the implicit function.

All our experiments in Table 4.5 use a batch size of 1000, except for *Backprop OccamNet* and *Finetune ResNet*, for which we use batch size 128. All our experiments in Table 4.6 use a batch size of 200. For each of our pattern recognition experiments, we use a 90%/10% train/test random split for the corresponding datasets. The input pixels are normalized to be in the range [0, 1]. During validation, for *MNIST Binary*, *MNIST Trinary* and *ImageNet Binary* the outputs of OccamNet are thresholded at 0.5. If the output matches the one-hot label, then the prediction is accurate; otherwise, it is inaccurate. For *Backprop OccamNet* and *Finetune ResNet* the outputs of OccamNet are viewed as the logits of a negative log likelihood loss function, so the prediction is the argmax of the logits. Backprop OccamNet and Finetune ResNet use an exponential decay of the learning rate with decay factor 0.999.

PMLB Experiments Setup As described in the main text, we test OccamNet on 15 datasets from the Penn Machine Learning Benchmarks (PMLB) repository [Olson et al., 2017]. The 15 datasets chosen and the corresponding numbers we use to reference them, are shown in Table 4.7. We chose these datasets by selecting the first 15 regression datasets with fewer than 1667 datapoints. These 15 datasets are the only datasets from PMLB we examine.

We test four methods on these datasets. OccamNet, V100, Eplex, AIF, and Ex-

Table 4.5: Hyperparameters for Experiments Where $E = 0$

Target	Bases	Constants	Range	L/ T/ T_{last}/ σ	R/ λ/ α
		Analytic Functions			
$2x^2 + 3x$	$(\cdot, \cdot, +, +)$	$\emptyset$	$[-10, 10]$	2/1/1/0.01	50/5/0.05
$\sin(3x + 2)$	$(\cdot, \sin, \sin, +, +)$	1, 2	$[-10, 10]$	3/1/1/0.001	50/5/0.005
$\sum_{n=1}^{3} \sin(nx)$	$(\sin, \sin, +, +, +)$	1, 2	$[-20, 20]$	5/1/1/0.001	50/5/0.005
$(x^2 + x)/(x + 2)$	$(\cdot, \cdot, +, +, /, /)$	1	$[-6, 6]$	2/1/2/0.0001	100/5/0.005
$x_0^2(x_0 + 1)/x_1^5$	$(\cdot, \cdot, +, +, /, /)$	1	$[[-10, 10], [0.1, 3]]$	4/1/3/0.0001	100/10/0.002
$x_0^2/2 + (x_1 + 1)^2/2$	$(\cdot, \cdot, +, +, /)$	1, 2	$[[-20, -2], [2, 20]]$	3/1/2/0.1	150/5/0.005
		Program Functions			
$3x$ if $x > 0$, else x	$(\leq, \leq, \cdot, +, +, /)$	1	$[-20, 20]$	2/1/1.5/0.1	100/5/0.005
x^2 if $x > 0$, else $-x$	$(\leq, \leq, -(\cdot), +, +, -, \cdot)$	1	$[-20, 20]$	2/1/1.5/0.1	100/5/0.005
x if $x > 0$, else $\sin(x)$	$(\leq, \leq, +, +, \sin, \sin)$	1	$[-20, 20]$	3/1/1.5/0.01	100/5/0.005
$\mathsf{SORT}(x_0, x_1, x_2)$	$(\leq, +, \mathsf{MIN}, \mathsf{MAX}, \mathsf{MAX}/, \cdot, -)$	1, 2	$[-50, 50]^4$	3/1/4/0.01	100/5/0.004
$\mathsf{4LFSR}(x_0, x_1, x_2, x_3)$	$(+, +, \mathsf{XOR}, \mathsf{XOR})$	$\emptyset$	$\{0, 1\}^4$	2/1/1/0.1	100/5/0.005
$y_0(\vec{x}) = x_1$ if $x_0 < 2$, else $-x_1$ $y_1(\vec{x}) = x_0$ if $x_1 < 0$, else x_1^2	$(\leq, \leq, -(\cdot), \cdot)$	1, 2	$[-5, 5]^2$	3/1/3/0.01	100/5/0.002
$g(x) = x^2$ if $x < 2$, else $x/2$ $y(x) = g^{\circ 4}(x)$	$(\leq, \leq, +, \cdot, \cdot, /, /)$	1, 2	$[-8, 8]$	2/1/2/0.01	100/5/0.005
$g(x) = x + 2$ if $x < 2$, else $x - 1$ $y(x) = g^{\circ 2}(x)$	$(\leq, \leq, +, +, +, -, -)$	1, 2	$[-3, 6]$	2/1/1.5/0.01	100/5/0.005
		Pattern Recognition			
MNIST Binary	$\mathbf{\Phi}_A$	$\mathbf{C}$	$[0, 1]^{784}$	2/1/10/0.01	150/ 10/0.05
MNIST Trinary	$\mathbf{\Phi}_A$	$\mathbf{C}$	$[0, 1]^{784}$	2/1/10/0.01	150/ 10/0.05
ImageNet Binary*	$\mathbf{\Phi}_A$	$\mathbf{C}$	$[0, 1]^{2048}$	4/1/10/10	150/10/0.0005
Backprop OccamNet*	$\mathbf{\Phi}_B$	$\mathbf{C}$	$[0, 1]^{2048}$	4/1/10/NA	NA/NA/0.1
Finetune ResNet*	$\mathbf{\Phi}_B$	$\mathbf{C}$	$[0, 1]^{3\times 224\times 224}$	4/1/10/NA	NA/NA/0.1

treme Gradient Boosting (XGB) [Chen and Guestrin, 2016]. We have described all of these methods except for XGB in the main text. XGB is a tree-based method that was identified by [Orzechowski et al., 2018] as the best machine learning method based on validation MSE for modeling the PMLB datasets. However, XGB is not interpretable and thus cannot be used as a one-to-one comparison with OccamNet. Hence, although we provide the raw data for XGB's performance, we do not analyze it further. We train all methods except "V100" on a single core of an

Table 4.6: Hyperparameters for Experiments Where $E = 1$

Target	Bases	Constants	Range	L	σ	R	λ	$w_\phi/w_\psi/w_\xi/w_\gamma$
			Analytic Functions					
$10.5x^3.1$	$(+, -, \cdot, /, \sin, \cos, +c, \cdot c, (\cdot)^c)$	$\emptyset$	$[0, 1]$	2	0.0005	200	10	0/0/0/0
$\cos(x)$	$(+, /, \sin)$	$2, \pi$	$[-100, 100]$	3	0.01	400	50	0/0/0/0
e^x	$(+, \cdot c, (\cdot)^c)$	10	$[0, 1]$	3	0.05	200	1	0.7/0.3/0.05/0.03
			Implicit Functions					
$x_0 x_1 = 1$	$(+, -, \cdot, /, \sin, \cos)$	$\emptyset$	$[-1, 1]$	2	0.01	400	1	0.7/0.3/0.15/0.1
$x_0/x_1 = 1$	$(+, -, \cdot, /, \sin, \cos)$	$\emptyset$	$[-1, 1]$	2	0.01	400	1	0.7/0.3/0.15/0.1
$x_0^2 + x_1^2 = 1$	$(+, -, \cdot, /, \sin, \cos)$	$\emptyset$	$[-1, 1]$	2	0.01	200	10	0.7/0.3/0.15/0.1
$x_0/\cos(x_1) = 1$	$(+, -, \cdot, /, \sin, \cos)$	$\emptyset$	$[-1, 1]$	2	0.01	200	10	0.7/0.3/0.15/0.1
$m_1 v_1 - m_2 v_2 = 0$	$(+, -, \cdot, /, \sin, \cos)$	$\emptyset$	$[-10, 10]^3$	2	0.01	200	10	0.7/0.3/0.15/0.1

Intel Xeon E5-2603 v4 @ 1.70GHz. For all methods, we use the basis set $\Phi = (+(\cdot,\cdot), -(\cdot,\cdot), \times(\cdot,\cdot), \div(\cdot,\cdot), \sin(\cdot), \cos(\cdot), \exp(\cdot), \log|\cdot|)$.

Table 4.7: Datasets Tested

#	Dataset	Size	# Features
1	1027_ESL	488	4
2	1028_SWD	1000	10
3	1029_LEV	1000	4
4	1030_ERA	1000	4
5	1089_USCrime	47	13
6	1096_FacultySalaries	50	4
7	192_vineyard	52	2
8	195_auto_price	159	15
9	207_autoPrice	159	15
10	210_cloud	108	5
11	228_elusage	55	2
12	229_pwLinear	200	10
13	230_machine_cpu	209	6
14	4544_GeographicalOriginalofMusic	1059	117
15	485_analcatdata_vehicle	48	4

For each dataset, we perform grid search to identify the best hyperparameters. The hyperparameters searched for the two OccamNet runs are shown in Table 4.8. The other hyperparameters not used in the grid search are set as follows: $T = 10$, $T_{\text{last}} = 10$, $w_\phi = w_\psi = w_\xi = w_\gamma = 0$, and the dataset batch size is the size of the training data. For OccamNet V100, we set R to be approximately as large as can fit on the V100 GPU, which varies between datasets. See Table 4.9 for the exact number of functions tested for each dataset for OccamNet V100. For XGBoost, we use exactly the same hyperparameter grid as used in Orzechowski et al. [2018]. For Eplex, we use the same hyperparameter grid as used in Orzechowski et al. [2018], with the exception that we use a depth of 4 to match that of OccamNet.

Table 4.8: OccamNet Hyperparameters

Hyperparameter	OccamNet	OccamNet V100
α	$\{0.5, 1\}$	$\{0.5, 1\}$
σ	$\{0.5, 1\}$	$\{0.1, 0.5, 1\}$
E	$\{1, 5\}$	$\{0, 1, 5\}$
λ/R	$\{0.1, 0.5, 0.9\}$	$\{0.1, 0.5, 0.9\}$
R	$\{500, 1000, 2000\}$	max
N	$1000000/R$	1000

Table 4.9: Number of Functions Sampled Per Epoch

#	R
1	17123
2	8333
3	8333
4	8333
5	178571
6	166666
7	161290
8	52631
9	52631
10	78125
11	151515
12	41666
13	40000
14	7874
15	178571

We select the best run from the grid search as follows. For each hyperparameter combination, we first identify the models with the lowest training MSE and the lowest validation MSE:

- For OccamNet, we examine the highest probability function after each epoch. From these functions, we select the function with the lowest testing MSE and the function with the lowest validation MSE.
- For Eplex, we examine the highest-fitness individual from each generation. From these individuals, we select the individual with the lowest training MSE and the individual with the lowest validation MSE.
- For XGBoost, we train the model until the validation loss has not decreased for 100 epochs. We then return this model as the model with the best training MSE and validation MSE.

Once we have the models with the lowest training and validation MSE for each hyperparameter combination, we identify the overall model with the lowest training MSE from the set of lowest training MSE models, and we identify the overall model with the lowest validation MSE from the set of lowest validation MSE models. We then record these models' training MSE and validation MSE as the best training MSE

and validation MSE, respectively. Finally, we test the model with the overall lowest validation MSE on the testing dataset and record the result as the grid search testing MSE.

For our test of OccamNet and Eplex's scalability on the PMLB datasets, we use the same hyperparameter combinations as those listed described above, except that, as described in the main text, we run OccamNet with 250, 1000, 4000, 16000, and 64000 functions sampled per epoch and Eplex with 250, 500, 1000, 2000 and 4000 functions sampled per epoch. Our evaluation of training, validation, and testing loss is exactly the same as described above, except that we evaluate the lowest losses for each value of N instead of grouping N with all of the other hyperparameters.

Supplemental Material

Therefore, we leave tackling adversarial robustness of neural models for symbolic regression as an exciting direction for future work.

We have organized the Supplemental Material as follows:

- In Section G we provide the results of our experiments on the PMLB datasets.
- In Section H we examine the fits each method provides for the PMLB Datasets.
- In Section I analyze the results of the experiment scaling OccamNet on the PMLB datasets.
- In Section J we present a series of ablation studies.
- In Section K we discuss neural models for sorting and pattern recognition.
- In Section L we discuss a few small experiments we tested.
- In Section M we discuss research related to the various applications of OccamNet.
- In Section N we discuss the evolutionary strategies for fitting functions and programs that we use as benchmarks.

- In Section O, we catalog our code and video files.

G PMLB Experiment Results

We compare OccamNet, Eplex, and AIF, marking the method with the lowest MSE or training time in red. We also compare V100, Eplex, and AIF. Plots of the full results for the PMLB scaling experiment are shown in Figures 4-10 and 4-11. As discussed in Section I, Figure 4-11 shows OccamNet's performance when only considering a restricted set of hyperparameters.

H Analysis of Fits to PMLB Datasets

In this section, we analyze the fits that the methods discussed in Section 4.2.6 provide for the PMLB dataset.

OccamNet's solutions are all short, easy to comprehend fits to the data. We find that OccamNet uses addition, subtraction, multiplication, and division most extensively, exploiting $\sin(\cdot)$ and $\cos(\cdot)$ for more nonlinearity. Interestingly, OccamNet uses $\exp(\cdot)$ and $\log|\cdot|$ less frequently, perhaps because both functions can vary widely with small changes in input, making functions with these bases more likely to represent poor fits.

OccamNet's solutions demonstrate its ability to exploit modularity and reuse components. These solutions often have repeated components, for example in dataset #1, 1027_ESL, where the best fit to the training data is

$$y_0 = \frac{(\sin(x_2) + x_3 + x_1) \cdot (\sin(x_2) + x_3 + x_1)}{(\sin(x_2) + x_3 + x_1) + (x_3 + x_1) + (x_1 + x_3)}.$$

In this fit, OccamNet builds $\sin(x_2)+x_3+x_1$ in the first two layers of the network and then reuses it three times. Solutions like the above demonstrate OccamNet's ability to identify successful subcomponents of a solution and then to rearrange the subcomponents into a more useful form. Examples like the above, however, also demonstrate that OccamNet often overuses modularity, potentially restricting the domain of functions it can search. We suspect that the main reason that OccamNet may rely too

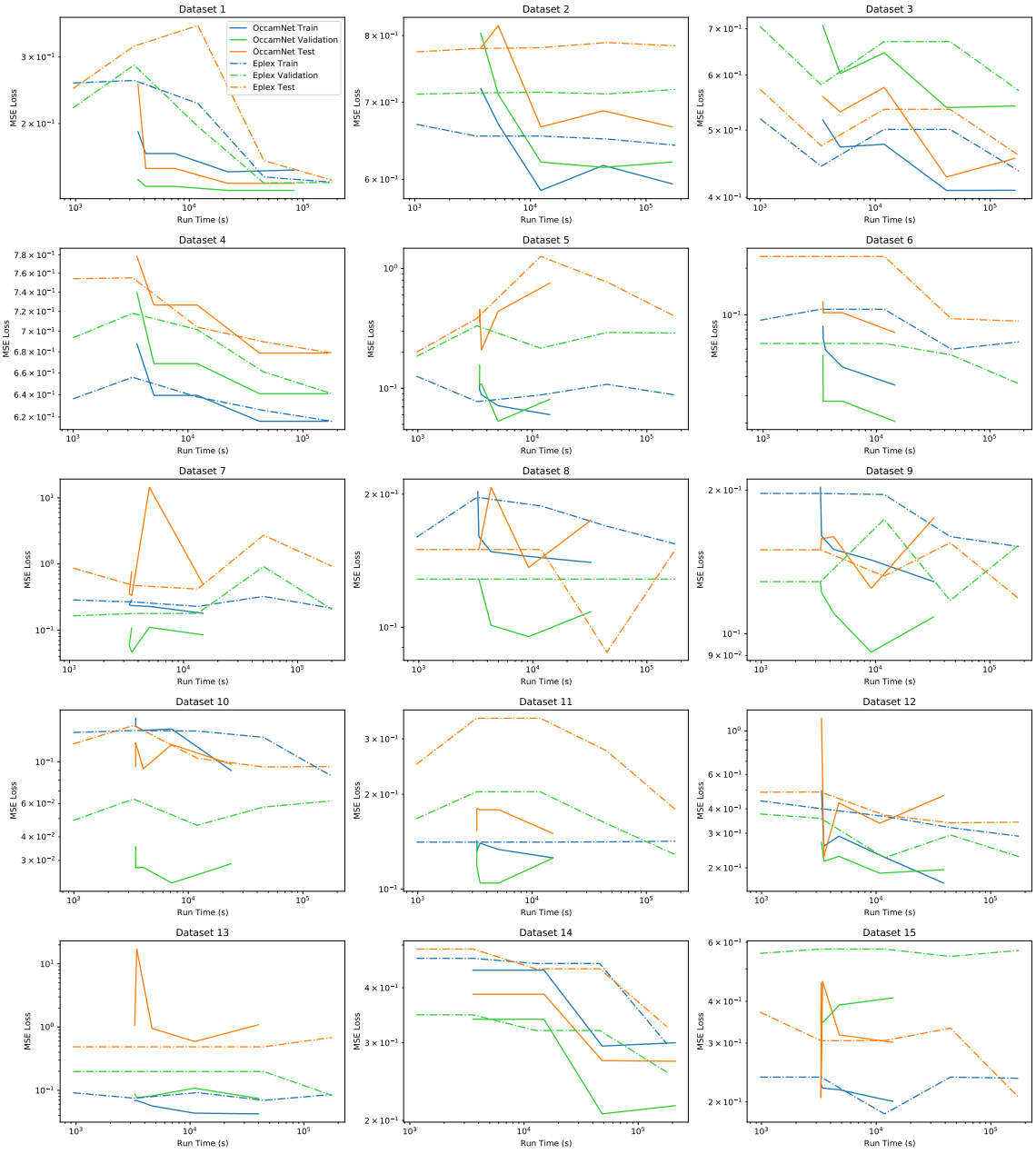

Figure 4-10: OccamNet V100 and Eplex's Training, Validation, and Testing MSE as a function of run time for the 15 PMLB datasets discussed above.

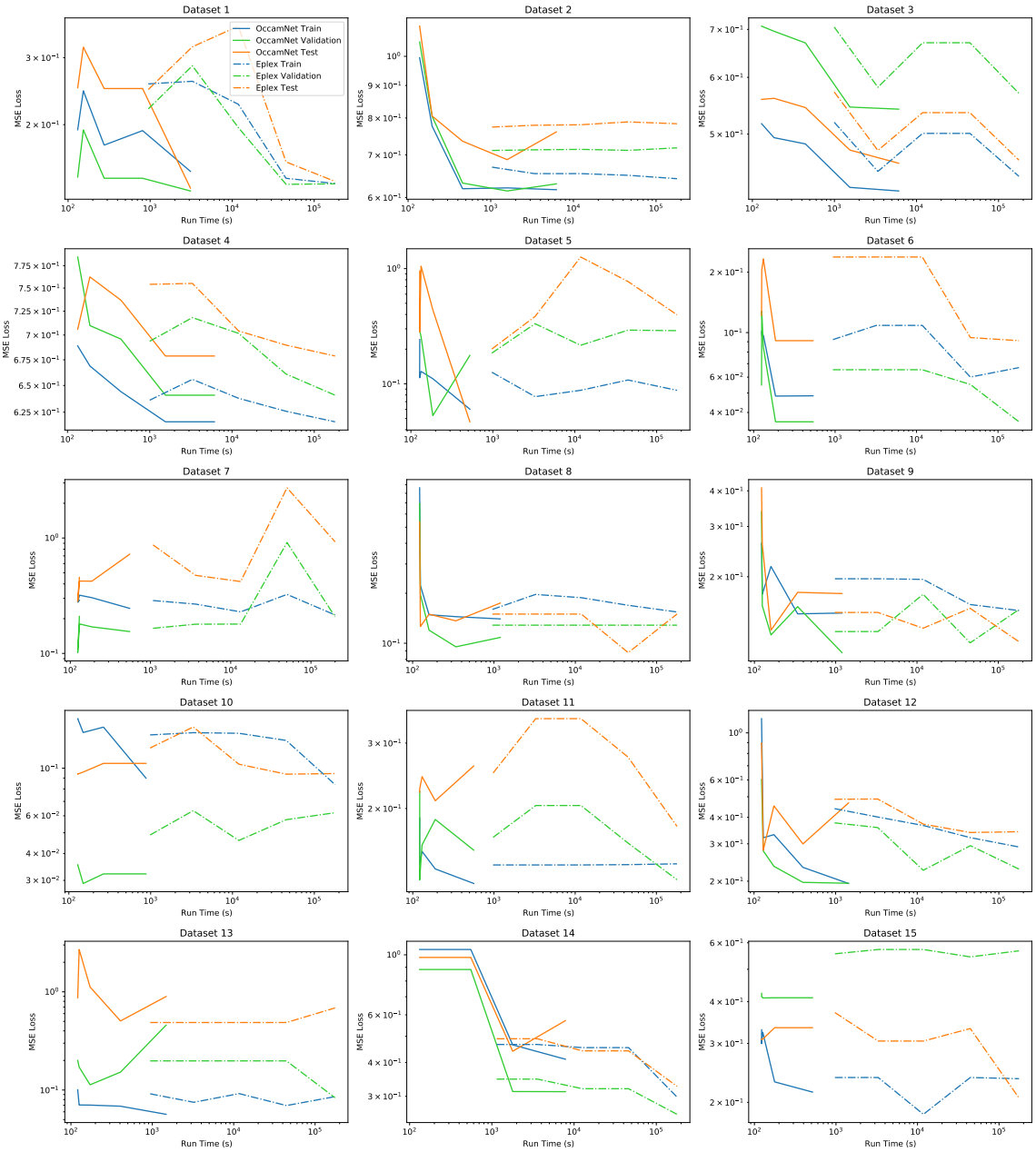

Figure 4-11: OccamNet V100 and Eplex's Training, Validation, and Testing MSE as a function of run time for the 15 PMLB datasets discussed above. For this figure, we only consider the losses for a restricted subset of hyperparameter combinations.

heavily on modularity in some fits is that OccamNet uses an extremely high learning rate of 1 for its training. We used such a large learning rate to allow OccamNet to converge even when faced with 10^{30} or more functions. However, we suspect that this may also cause OccamNet to converge to certain paths before exploring sufficiently. For example, with the function above, OccamNet may have identified that $\sin(x_2) + x_3 + x_1$ is a useful component and, because of its high learning rate, used this pattern several times instead of the one time needed. This hypothesis is supported by the fact that OccamNet V100, which samples many more functions before taking a training step, repeats patterns less frequently than OccamNet. For example OccamNet V100's best-fit solution for the training dataset of dataset #1 is

$$y_0 = \cos(x_1/x_1) \cdot \cos(x_1/x_1) \cdot (x_2 + x_3 + \sin(x_0) + x_3 + x_1 - \sin(x_3)),$$

which contains almost no repetition.

Remarkably, for dataset #4, 1030_ERA, both Eplex and OccamNet V100 discover equivalent functions for both training and validation: OccamNet V100 discovers

$$y_0 = \cos(\sin(x_1 - x_2)) \cdot (\sin(x_2) + x_0 + x_1) \cdot \cos(x_2/x_2),$$

and Eplex discovers

$$y_0 = \cos(x_1/x_1) \cdot (\sin(x_2) + x_0 + x_1) \cdot \cos(\sin(x_2 - x_1)).$$

As a result, the two methods' losses are identical up to 7 decimal places. Still, we mark Eplex as performing better on this dataset because after the seventh decimal place it has a slightly lower loss, likely due to differences in rounding or precision between the two approaches. Two different methods identifying the same function is extremely unlikely; OccamNet's search space includes $2 \cdot 10^{30}$ paths for this dataset, meaning that the probability of both methods identifying this function purely by chance is minuscule. In combination with the fact that this function was the best fit to both the training and validation datasets for both methods, this suggests that the identified

function is a nearly optimal fit to the data for the given search space. Given the size of the search space, this result thus provides further evidence that OccamNet and Eplex perform far better than brute-force search. Interestingly, although OccamNet did not discover this function, it's best fit for the validation,

$$y_0 = \sin(x_2/x_2) \cdot (\sin(x_2) + x_0 + x_1) \cdot \cos(\cos(x_3)) \cdot \cos(\sin(x_3)),$$

does include several features present in the fits found by V100 and Eplex, such as the $\sin(x_2) + x_0 + x_1$ term, the $\cos(\sin(\cdot))$ term, and the x_2/x_2 inside of the trigonometric function. This suggests that OccamNet may also have been close to converging to the function discovered by Eplex and OccamNet V100. OccamNet's loss was also always within 5% of Eplex's loss on this dataset, again suggesting that OccamNet had identified a function close to that of Eplex and V100.

Interestingly, AI Feynman 2.0's fits generally tend to be very simple compared to those of OccamNet. For example, AIF's fit for the training dataset #11 is

$$y_0 = -0.050638447726 + \log(x_0/\sin(x_0)) - x_0,$$

whereas OccamNet's fit is

$$y_0 = \sin(x_0) \cdot x_1 \cdot x_0 \cdot \sin(x_0) \cdot \log|x_0| - \cos(x_1 \cdot x_0 - x_1).$$

AI Feynman's fit is slightly simpler and easier to interpret, but it comes at the cost of having a 35% higher loss. We suspect that because the PMLB datasets likely do not have modular representations, AI Feynman must rely mainly on its brute-force search, which ultimately produces shorter expressions. AI Feynman can also produce constants because of its polynomial fits, and it uses constants in nearly every solution it proposes. We did not allow the other symbolic methods to fit constants, but they still consistently performed better than AI Feynman, suggesting that fitting constants may not be essential to accurately modeling the PMLB datasets.

As discussed in the main text, OccamNet is considerably faster than Eplex, often

running faster by more than an order of magnitude. This may be in part because we train Eplex with the DEAP evolutionary computation framework Fortin et al. [2012], which is implemented in Python and utilizes NumPy arrays for computation. Thus, our implementation of Eplex may be somewhat slower than an implementation written in C. However, because of its selection based on many fitness cases, Eplex is also by nature considerably slower than many other genetic algorithms, running in $O(TN^2)$, where T is the number of fitness cases and N is the population size Melo et al. [2019]. This suggests that even a pure C implementation of Eplex may not be as fast as OccamNet. More recent selection algorithms perform comparably to Eplex but run significantly faster, for example Batch Tournament Selection Melo et al. [2019]. However, because these methods did not exist at the time of Orzechowski et al. [2018], they have not been compared to other methods on the PMLB datasets. Thus, we have not tested these methods here. On the other hand, our current implementation of sampling and the forward pass work with DAGs in which an edge leads to each argument node, regardless of whether the argument node is connected to the outputs. The result is that our implementation of OccamNet evaluates more than $|\Phi|$ times more basis functions than is necessary, where $|\Phi|$ is the number of basis functions. In the case of these experiments, this amounts to more than eight times the number of calculations necessary. It may be possible to optimize OccamNet by not evaluating such unused connections, thereby obtaining a much faster runtime.

I Analysis of PMLB Scaling Tests

As can be seen in Figure 4-10, OccamNet's training loss decreases with increasing sample size for every dataset – the training loss for 64000 functions sampled is always less than the training loss for 250 functions sampled. For some datasets, OccamNet's training loss is not monotonically decreasing, but this is to be expected given OccamNet's inherent randomness and the size of the search space.

For datasets 1, 2, 3, 4, 13, and 14, the training loss does not drop noticeably when increasing the sample size beyond a certain point. There are two possible explanations for this. (i) For all of these datasets, OccamNet's training loss is very close to or lower

than Eplex's best training loss, suggesting that OccamNet may be approaching an optimal fit and that there is little room to further decrease the loss. (*ii*) There may be critical sample sizes before which OccamNet's training loss is stagnant and beyond which its training loss begins to decrease. This is apparent in datasets 1, 3, 4, 10, 12, and 14, where the OccamNet training loss temporarily stops decreasing at 1000 or 4000 functions sampled. It is possible that for some datasets another such critical sample size exists beyond 64000 functions.

For datasets 1, 2, 3, 4, 6, 12, and 14, OccamNet's validation loss also decreases with the number of functions sampled. However, for datasets 5, 7, 8, 9, 10, and 11, it initially decreases and then begins to increase, and for datasets 13 and 15 it does not decrease. Interestingly, the datasets for which OccamNet's testing loss does not decrease are generally the same as the datasets for which OccamNet's validation loss does not decrease. The datasets where the validation and testing loss do not decrease are generally very small, with around 200 or fewer datapoints. This suggests that OccamNet is overfitting. Such overfitting is to be expected given the small number of samples in the PMLB datasets. On the other hand, Eplex only seems to overfit in datasets 5 and 9. Because overfitting results from fitting a training dataset too well, this is further evidence that OccamNet is fitting the training datasets better than Eplex.

Note that for each number of functions sampled, we tested 81 different hyperparameter combinations for OccamNet and only 3 for Eplex. This is largely because, as a new architecture, OccamNet's optimal hyperparameters are not known. For a fair comparison, the runtimes we report in Figure 4-10 are the times required to run all hyperparameter combinations. Thus, because OccamNet uses 27 times more hyperparameter combinations, its speed advantage is lessened, although still significant.

After examining the results for all hyperparameter combinations, however, we noted that all hyperparameters but the learning rate had an "optimal" value, listed in Appendix 4.2.6. Restricting to only the remaining three hyperparameter combinations produces a best training loss that is often the same as, and is never more than 40% greater than, the lowest loss among all 81 hyperparameters. Figure 4-

11 shows the results when considering only OccamNet's restricted hyperparameter combinations for three datasets.

When OccamNet and Eplex are restricted to the same number of hyperparameter combinations, OccamNet always runs faster when sampling 64000 functions per epoch than Eplex does when sampling 1000 functions per epoch. OccamNet's training loss consistently decreases with increasing sample size, although its validation and testing losses do not always follow such a clear trend. Further, OccamNet almost always converges to training and validation losses that are close to or less than Eplex's training and validation losses. OccamNet's best training loss is less than or approximately the same as Eplex's best training loss for all but datasets 1, 14, and 15.

Dataset 14 consists of over 1000 datapoints with 117 features, so it is likely one of the most difficult datasets which we test. The fact that OccamNet does perform comparatively to Eplex when it tests additional hyperparameter combinations suggests that for such difficult problems OccamNet benefits from additional hyperparameter exploration, particularly involving weight initialization.

For dataset 15, because Eplex only identifies a better function than OccamNet when it samples 1000 functions and not when it samples 2000 or 4000 functions, Eplex's better performance appears to be somewhat of an outlier.

With the restricted set of hyperparameters, OccamNet still overfits on every dataset it overfitted on when using the full set of hyperparameters, suggesting that the overfitting is not due to the large number of hyperparameters. Interestingly, for both the full and restricted hyperparameter versions of OccamNet, OccamNet and Eplex again identify the same fit for Dataset 4,

$$y_0 = \cos(\sin(x_1 - x_2)) \cdot (\sin(x_2) + x_0 + x_1) \cdot \cos(x_3/x_3).$$

J Ablation Studies

We test the performance of various hyperparameters in a collection of ablation studies, as shown in Table 4.10. Here, we focus on what our experiments demonstrate to be the most critical parameters to be tuned: the collection of bases and constants,

the network depth, the variance of our interpolating function, the overall network temperature (as well as the last layer temperature), and, finally, the learning rate of our optimizer. As before, we set the stop criterion and terminate learning when the top-λ sampled functions all return the same fitness $K(\cdot, f)$ for 30 consecutive epochs. If this does not occur in a predefined, fixed number of iterations, or if the network training terminates and the final expression does not match the correct function we aim to fit, we say that the network has not converged. All hyperparameters for baselines are specified in Section 4.2.6, except for the sampling size, which is set to $R = 100$.

Table 4.10: Ablation studies on representative experiments

Modification	Convergence fraction η	Convergence epochs T_c
Experiment $\sin(3x + 2)$		
baseline	10/10	390
added constants (2) and bases $(\cdot, (\cdot)^2, -(\cdot))$	10/10	710
lower last layer temperature (0.5)	10/10	300
higher last layer temperature (3)	10/10	450
lower learning rate (0.001)	10/10	2500
higher learning rate (0.01)	10/10	170
deeper network (6)	8/10	3100
lower variance (0.0001)	10/10	390
higher variance (0.1)	10/10	450
lower sampling (50)	10/10	680
higher sampling (250)	10/10	200
Experiment x^2 if $x > 0$, else $-x$		
baseline	10/10	100
added constants (1, 2) and bases $(-, -(\cdot))$	10/10	290
lower last layer temperature (0.5)	10/10	160
higher last layer temperature (3)	10/10	150
lower learning rate (0.001)	10/10	780
higher learning rate (0.01)	10/10	90
deeper network (6)	10/10	180
shallower network (2)	10/10	160
lower variance (0.001)	10/10	160
higher variance (1)	10/10	180
lower sampling (50)	10/10	290
higher sampling (250)	10/10	140

Our benchmarks use a sampling size large enough for convergence in most experiments. It is worth noting, however, that deeper networks failed to always converge (with a convergence fraction of $\eta = 8/10$) for the analytic function we tested. Deeper networks allow for more function composition and let approximations emerge as local minima: in practice, we find that increasing the last layer temperature or reducing the variance is often needed to allow for a larger depth L. For pattern recognition, we found that *MNIST Binary* and *Trinary* require depth 2 for successful convergence, while the rest of the experiments require depth 4. Shallower or deeper networks either yield subpar accuracy or fail to converge. We also find that for OccamNet

without skip connections, larger learning rates usually work best, i.e., 0.05 works best, while OccamNet with skip connections requires a smaller learning rate, usually around 0.0005. We also tested different temperature and variance schedulers in the spirit of simulated annealing. In particular, we tested increasing or decreasing these parameters over training epochs, as well as sinusoidally varying them with different frequencies. Despite the increased convergence time, however, we did not find any additional benefits of using schedulers. As we test OccamNet in larger problem spaces, we will revisit these early scheduling studies and investigate their effects in those domains.

K Neural Approaches to Benchmarks

Since OccamNet is a neural model that is constructed on top of a fully connected neural architecture, below we consider a limitation of the standard fully connected architectures for sorting and then a simple application of our temperature-controlled connectivity.

Exploring the limits of fully connected neural architectures for sorting

We made a fully connected neural network with residual connections. We used the mean squared error (MSE) as the loss function. The output size was equal to the input size and represented the original numbers in sorted order. We used L_2 regularization along with Adam optimization. We tested weight decay ranging from 1e-2 to 1e-6 and found that 1e-5 provided the best training and testing accuracy. Finally, we found that the optimal learning rate was around 1e-3. We used $30,000$ data points to train the model with batch size of 200. Each of the data points is a list of numbers between 0 and 100. For a particular value of input size x (representing the number of points to be sorted), we varied the number of hidden units from 2 to 20 and the number of hidden layers from 2 (just an input and an output layer) to $x! + 2$. Then, the test loss was calculated on 20,000 points, chosen from same distribution. Finally, for each input size, Table 4.11 records the combination (hidden_layer, hidden_unit)

for which the loss is less than 5 and (hidden_ layer * hidden_units) is minimized. As seen from the table, the system failed to find any optimal combination for any input size greater than or equal to 5. For example, for input size 5, the hidden units were upper capped at 20 and hidden layers at 120 and thus 2400 parameters were insufficient to sort 5 numbers.

Table 4.11: Minimal configurations to sort list of length "input size."

Input Size	Hidden units	Hidden Layers	Parameters
2	6	2	12
3	8	4	32
4	18	4	72
5	-	-	-

Generalization The model developed above generalizes poorly on data outside the training domain. For example, consider the model with 18 hidden units and four hidden layers, which is successfully trained to sort four numbers chosen from the range 0 to 100. It was first tested on numbers from 0 to 100 and then on 100 to 200. The error in the first case was around 2 while the average error in the second case was between 6 and 8 (which is $(200/100)^2 = 4$ times the former loss). Finally, when tested on larger ranges such as $(9900, 10000)$, the error exploded to around 0.1 million (which is an order greater than $(10000/100)^2 = 10000$ times the original loss). This gives a hint that the error might be scaling proportionally to the square of the test domain with respect to the train domain. A possible explanation for this comes from the use of the MSE loss function. Scaling test data by ρ scales the absolute error by approximately the same factor and then taking a square of the error to calculate the MSE scales the total loss by the square of that factor, i.e., ρ^2.

Applying temperature-controlled connectivity to standard neural networks for MNIST classification

We would like to demonstrate the promise of temperature-controlled connectivity as a regularization method for the classification heads of models with a very simple

experiment. We used the ResNet50 model to train on the standard MNIST image classification benchmark. We studied two variants of the model: the standard ResNet model and ResNet augmented with our temperature-controlled connectivity (with $T = 1$) between the flattened layer and the last fully connected layer (on the lines discussed in the main paper). Then we trained both models with a learning rate fixed at 0.05 and a batch size of 64 and ran it for 10 epochs. The model with regularization performed slightly better than the one without it. The regularized model achieved the maximum accuracy of 99.18%, while the same figure for the standard one was 98.43%. Another interesting observation is that the regularized model produces much more stable and consistent results across iterations than the unregularized model. These results encourage us to study the above regularization method in larger experiments.

L Small Experiments

To demonstrate its ability to fit functions with constants, we also tested OccamNet on the function $10.5x^{3.1}$ without providing either 10.5 or 3.1 beforehand. OccamNet identified the correct function 10 times out of 10, taking an average of 553s.

We also investigated whether OccamNet could discover a formula for cosine using only the bases $\sin(\cdot)$, $+(\cdot,\cdot)$, and $\div(\cdot,\cdot)$ and the constants 2 and π. We expected OccamNet to discover $\cos(x) = \sin(x + \pi/2)$, but, interestingly, it instead always identified the double angle identity $\cos(x) = \sin(2x)/(2\sin(x))$. OccamNet successfully identified an identified an identity for cosine 8 out of 10 times and in an average of 410s. A more optimized implementation of OccamNet takes only 7s for the same task, although its accuracy is somewhat lower, fluctuating between 2 and 6 out of 10 correct identifications.

Similarly, we tested whether OccamNet could discover Taylor polynomials of e^x. OccamNet identified $e^x \approx 1 + x + x^2/2$, but was unable to discover the subsequent $x^3/6$ term.

M Related Work

Symbolic regression

OccamNet was partially inspired by the EQL network [Martius and Lampert, 2016, Sahoo et al., 2018a, Kim et al., 2020b], a neural network-based symbolic regression system that successfully finds simple analytic functions. Neural Arithmetic Logic Units (NALU) and related models [Trask et al., 2018a, Madsen and Johansen, 2020] provide a neural inductive bias for arithmetic in neural networks that can in principle fit some benchmarks in Table 4.1. NALU updates the weights by backpropagating through the activations, shaping the neural network towards a gating interpretation of the linear layers. However, generalizing those models to a diverse set of function bases might be a formidable task: from our experiments, backpropagation through some activation functions (such as division or sine) makes training considerably harder. In a different computational paradigm, genetic programming (GP) has performed exceptionally well at symbolic regression [Schmidt and Lipson, 2009, Udrescu and Tegmark, 2020], and a number of evolution-inspired, probability-based models have been explored for this goal Mckay et al. [2010].

A concurrent work [Petersen et al., 2021] explores deep symbolic regression by using an RNN to search the space of expressions using autoregressive expression generation. Interestingly, the authors observed that a risk-aware reinforcement learning approach is a necessary component in their optimization, which is similar to our approach of selecting the top λ function for optimization in Step 2 of our algorithm. A notable difference is that OccamNet does not generate the expressions autoregressively, although it still exhibits a gradual increase in modularity during training, as discussed in Section 4.2.5. This is also a benefit both for speed and scalability. Moreover, their entropy regularization is a potentially useful addition to our training algorithm. Marrying our approach with theirs is a promising direction for future work.

Transformer-based models can quickly and accurately identify functions given data by leveraging their extensive pretraining. However, these approaches are limited in

that they are restricted to a set of basis functions specified at training time. It may thus be fruitful to investigate combining OccamNet and such approaches in a way that increases convergence speed while maintaining OccamNet's flexibility.

Program synthesis

A field related to symbolic regression is program synthesis. For programs, one option to fit programs is to use EQL-based models with logic activations (step functions, MIN, MAX, etc.) approximated by sigmoid activations. Another is probabilistic program induction using domain-specific languages [Ellis et al., 2018b,a, 2019]. Neural Turing Machines [Graves et al., 2014, 2016] and their stable versions [Collier and Beel, 2018] are also able to discover interpretable programs based on observations of input-output pairs. They do so by simulating programs using neural networks connected to an external memory. Balog et al. [2016] first train a machine learning model to predict a DSL based on input-output pairs and then use methods from satisfiability modulo theory [Solar Lezama, 2008] to search the space of programs built using the predicted DSL. In contrast, our DSL is lower level and can fit components like "sort" instead of including them in the DSL directly. Kurach et al. [2016] develop a neural model for simple algorithmic tasks by utilizing memory access for pointer manipulation and dereferencing. However, here we achieve similar results (for example, sorting) without external memory and in only minutes on a CPU.

Integration with deep learning

We are not aware of classifiers that predict MNIST or ImageNet labels using symbolic rules. The closest baseline we found is using GP [Montana and Davis, 1989], which performs comparably well to our neural method, but cannot easily integrate with deep learning. In the reinforcement learning (RL) domain, Such et al. [2017] and Salimans et al. [2017] propose training deep models of millions of parameters on standard RL tasks using a gradient-free GP, which is competitive to gradient-based RL algorithms.

SCGs and pruning

Treating the problem of finding the correct function or program as a stochastic computational graph is appealing due to efficient soft approximations to discrete distributions [Maddison et al., 2017, Jang et al., 2017, Tucker et al., 2017]. Our T-softmax layers offer such an approximation and could further benefit from an adaptive softmax methodology [Grave et al., 2017], which we leave for future work. Furthermore, the sparsity induced by T-softmax layers parallels the abundant work on pruning connections and weights in neural networks [Han et al., 2015, Li et al., 2017] or using regularizations, encouraging sparse connectivity [Molchanov et al., 2017, Louizos et al., 2018].

N Symbolic Regression Benchmarks

Eureqa

Eureqa is a software package for symbolic regression where one can specify different target expressions, building block functions (analogous to the bases in OccamNet), and loss functions [Schmidt and Lipson, 2009]. For most functions, we use the absolute error as the optimization metric. We choose formula building blocks in Eureqa to match the basis functions used in OccamNet.

For implicit functions, we use the implicit derivative error. We also order the data to improve the performance. For the implicit functions in lines 1, 3, and 4 in Table 2 of the main text, the data is ordered by x_0. For the equation $x_0^2 + x_1^2 = 1$, the data is generated by sampling $\theta \in [0, 2\pi)$ and calculating $x_0 = \sin(x)$ and $x_1 = \cos(x)$, and is ordered by θ. When the data is not ordered, the value of the implicit derivative error is much higher, resulting in the algorithm favoring incorrect equations. For equation $m_1 v_1 - m_2 v_2$, the ordering is more ambiguous because of the higher dimensionality. We tried ordering by both m_1 and the product $m_1 v_1$ without success.

HeuristicLab

Due to limits on the number of data points and feature columns in Eureqa, we instead use HeuristicLab for the image recognition tasks described in Section 5.4 of the main text. HeuristicLab is a software package for optimization and evolutionary algorithms, including symbolic regression and symbolic classification. We use the Island Genetic Algorithm with default settings.

Similar to the building block functions in Eureqa, HeuristicLab can specify the basis symbols for each task. However, HeuristicLab does not have the bases MAX, SIGMOID, or tanh. Instead, we use the symbols IfThenElse, GreaterThan, LessThan, And, Or, and Not.

Eplex

As discussed in Section 4.2.6, Eplex [La Cava et al., 2016], short for Epsilon-Lexicase selection, is a genetic programming population selection technique that we use as a symbolic regression benchmark in our experiments with PMLB datasets. We implement a genetic algorithm using Eplex with the DEAP [Fortin et al., 2012] evolutionary framework, using Numpy arrays [Harris et al., 2020] for computation to increase speed.

Eplex selects individuals from a population by evaluating the individuals on subsets, or fitness cases, of the full data. For each fitness case, Eplex selects the top-performing individuals and proceeds to the next fitness case. This process is repeated until only one individual remains. This individual is then used as the parent for the next generation.

AI Feynman 2.0

We also benchmark OccamNet against AI Feynman 2.0 [Udrescu et al., 2020]. AI Feynman 2.0 is a mixed approach that combines brute-force symbolic regression, polynomial fits, and identification for modularity in the data using neural networks. To identify modularities in the data, AI Feynman first trains a neural network on it.

This serves as an interpolating function for the true data and allows the network to search for symmetries and other forms of modularities.

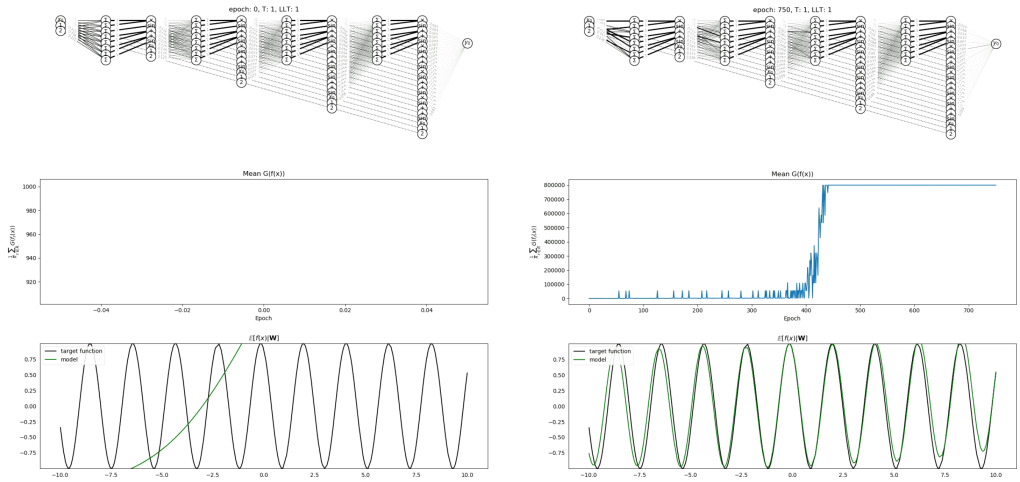

Figure 4-12: In this figure, we present two video frames for the target $\sin(3x + 2)$, which could be accessed via `videos/sin(3x + 2).mp4` in our code files. We show the beginning of the fitting (left) and the end, where OccamNet has almost converged (right).

O Code

We will make our code public. We have grouped our code into five main folders. `analytic-and-programs` stores our network and experiments for fitting analytic functions and programs. `implicit` stores our network and experiments for implicit functions, although it also includes the three analytic functions listed in Table 4.6. `constant-fitting` stores code very similar to `implicit` but optimized for constant fitting. `image-recognition` stores our network and experiments for image classification. `pmlb-experiment` stores our code for benchmarking against the PMLB regression datasets. Finally, `videos` stores several videos of our model converging to various functions. In Figures 4-12 and 4-13, we present snapshots of the videos.

Currently, our method is not explicitly designed against adversarial attacks. Thus, malicious stakeholders could exploit our method and manipulate the symbolic fits that OccamNet produces. A potential direction towards alleviating the problem would

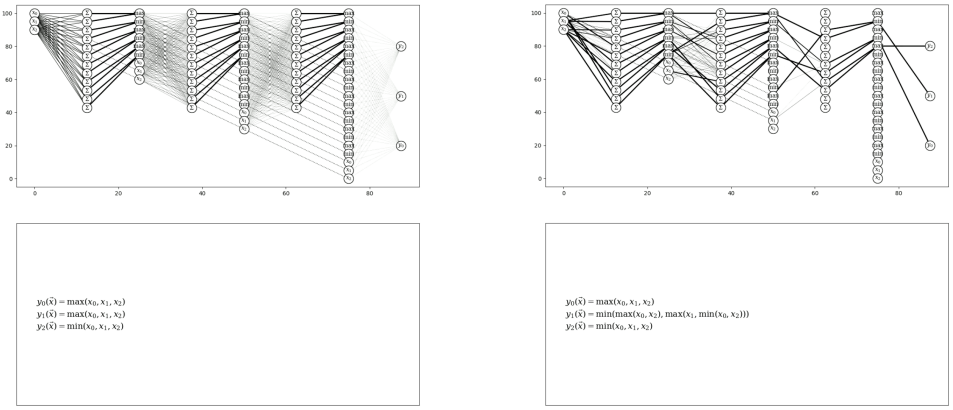

Figure 4-13: In this figure we present two video frames for the target $\mathsf{SORT}(x_0, x_1, x_2)$, which could be accessed via `videos/sorting.mp4` in our code files. We show the beginning of the fitting (left) and the end, where OccamNet has almost converged (right).

be to explore ways to robustify OccamNet by training it against an adversary. In the meantime, we ask that users of our code remain responsible and consider the repercussions of their use cases.

Outlook. One of the biggest challenges in symbolic regression is the lack of well-defined real-work benchmarks for discoveries. This is a notable limitation of the work in this section too, hence we focus on providing an application of OccamNet on data in the real world in the following section.

4.3 AI-Assisted Discovery of Quantitative and Formal Models in Social Science

In social science, formal and quantitative models, such as ones describing economic growth and collective action, are used to formulate mechanistic explanations, provide predictions, and uncover questions about observed phenomena. Here, we demonstrate the use of a machine learning system to aid the discovery of symbolic models that capture nonlinear and dynamical relationships in social science datasets. By extend-

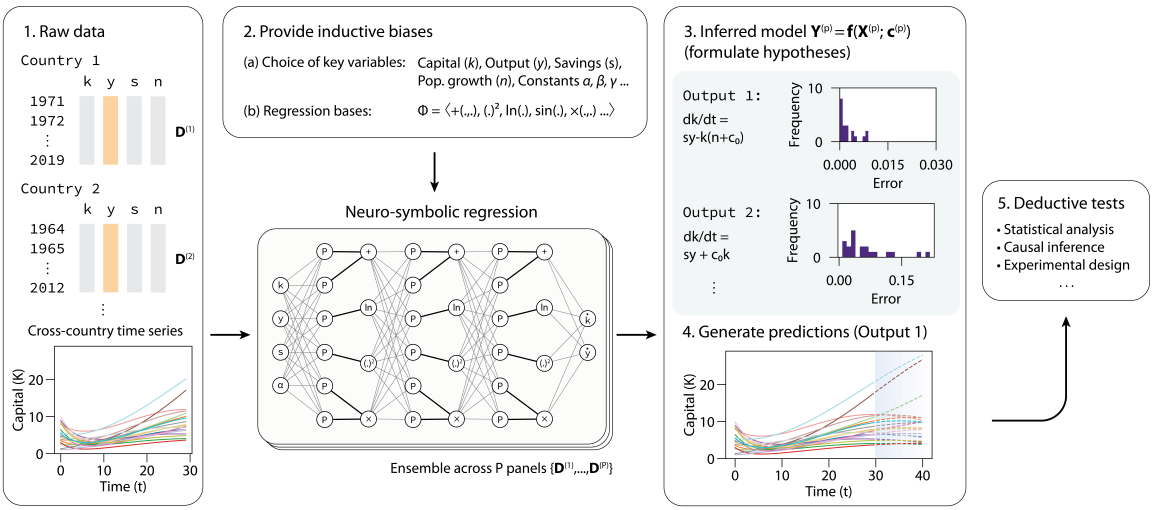

Figure 4-14: Using neuro-symbolic regression to systematically guide model discovery in social science. Analogous to the inductive-deductive reasoning process, a dataset of interest (1) – which may be time-series, cross-sectional, or longitudinal – is supplied to OccamNet. The user can provide inductive biases (2), such as the choice of key variables, known constants, or specific functional forms to constrain the search space. OccamNet finds interpretable and compact solutions that model the input data by sampling functions from an internal probability distribution represented using *P-nodes* [Costa et al., 2021]. In this example, OccamNet recovers the governing equation of the Solow-Swan model of economic growth [Solow, 1956] from a synthetic dataset. Each formal model is characterized by its error distribution in the training set (3), allowing the user to identify outliers and interrogate its internal validity. The symbolic model is then used to generate predictions (4) to perform deductive tests across unseen data, either by partitioning a test set or informing experimental designs (5).

ing neuro-symbolic methods to find compact functions and differential equations in noisy and longitudinal data, we show that our system can be used to discover interpretable models from real-world data in economics and sociology. Augmenting existing workflows with symbolic regression can help uncover novel relationships and explore counterfactual models during the scientific process. We propose that this AI-assisted framework can bridge parametric and non-parametric models commonly employed in social science research by systematically exploring the space of nonlinear models and enabling fine-grained control over expressivity and interpretability.

Our proposed workflow. Figure 4-14 outlines our workflow for AI-assisted model discovery in social science. Symbolic regression allows us to uncover interpretable expressions as opposed to black-box or fully non-parametric results. These expressions, which may take the form of differential equations, scaling laws, or functional relationships, can shed light on the interactions between variables and help social scientists generalize beyond linear relationships. Furthermore, because symbolic laws can have system-specific constants, the inferred models also describe precise structural relationships that can be used to generate testable predictions across populations or timeframes. We employ a neural network architecture built on OccamNet to find sparse, interpretable formal models that fit the desired input data. In particular, we leverage OccamNet's architecture to implement two key features to systematically explore the space of candidate models in social science: complexity regularization and data ensemble learning.

Predator-prey relationships. Ecologists have long relied on mathematical models of population dynamics for studying ecological processes such as predator-prey interactions. While these models – often taking the form of differential equations – were historically derived from first principles and rigorous theory-building, symbolic regression offers a promising tool for reverse-engineering ecological models from data Martin et al. [2018].

A simple yet paramount model of predator-prey interactions between two species is given by the Lotka-Volterra equations Lotka [1910],

$$\begin{aligned} \frac{dH}{dt} &= \alpha H - \beta HL \\ \frac{dL}{dt} &= \delta HL - \gamma L, \end{aligned} \tag{4.12}$$

where H and L are the populations of the prey and predator, respectively. The constant parameter α is the rate of exponential population growth for the prey, γ is the rate of exponential death for the predator, β is the rate at which the predators destroy the prey, and δ is the rate of predator population growth due to the consumption of prey. Inspired by the chemical law of mass action Berryman [1992], the Lotka-Volterra

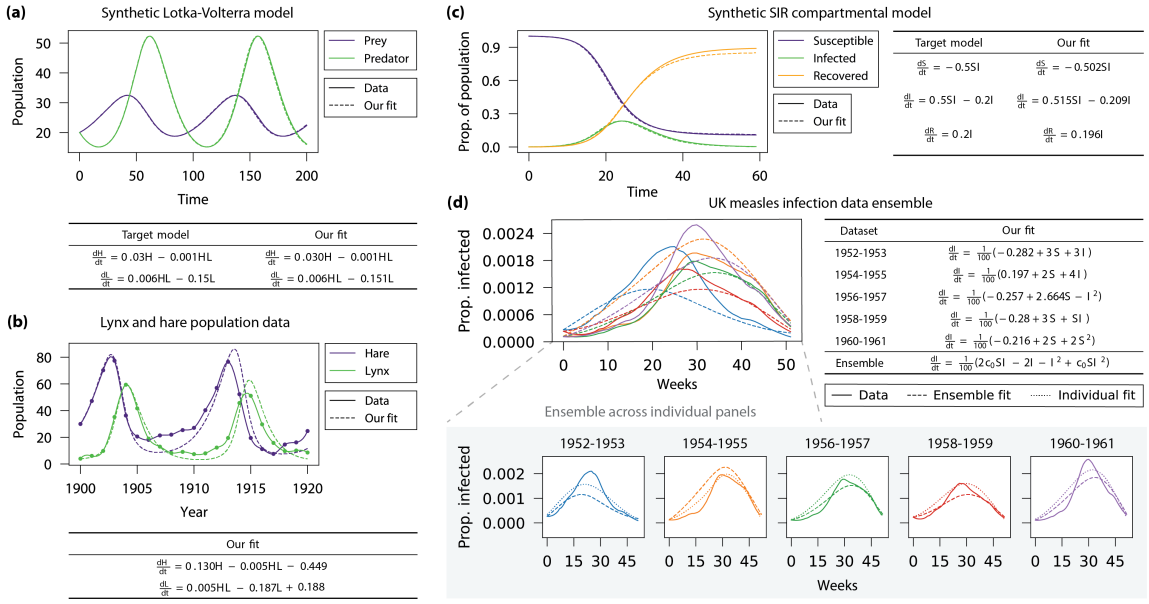

Figure 4-15: Regression of coupled dynamical models using noisy real-world data. **(a)** Time-series plot of a simulated Lotka-Volterra predator-prey system. OccamNet was able to correctly reconstruct the functional form and constants with high accuracy. **(b)** Using cubic spline interpolation, our system was able to learn the two differential equations from noisy, real-world data of lynx and hare populations with just 21 data points each. The inferred non-linear model can then be used to extend predictions of future populations. **(c)** The symbolic regression system is used to infer the SIR model of pandemic spread in synthetic data and **(d)** an ensemble of real-world measles infection data in the UK.

equations state that the rate of predation is directly proportional to the product of the populations of the prey and predator.

The Lotka-Volterra predator-prey model has significant implications beyond ecology. Richard Goodwin was one of the first to adopt the model to describe the cyclical relationship between employment and wages in economics, otherwise known as Goodwin's class struggle model Gandolfo [2007]. Other applications of generalized Lotka-Volterra systems in economics include the modeling of wealth distributions in society and values of firms in the stock market Malcai et al. [2002].

In Figure 4-15 (a) we simulate a system of Lotka-Volterra equations as defined in equation 4.12 with synthetic parameters α, β, γ, and δ. We then use OccamNet to fit $\frac{dH}{dt}$ and $\frac{dL}{dt}$ and simulate our generated differential equations using the same initial conditions. Our model is able to rediscover the exact functional form and constant parameters.

We then apply OccamNet to recover the Lotka-Volterra model from the Hudson Bay Company's data on the lynx and hare population in Canada from 1845-1935 [MacLulich, 1937]. As shown in Fig. 4-15 (b), we use the subset of records from 1900-1920 as it contains two population cycles with the least apparent noise.

When fitting the interpolated data using OccamNet, we recover a system of ODEs that closely resembles the Lotka-Volterra equations. In particular, our model is able to learn the exponential population growth and decay terms αH and $-\gamma L$, as well as the critical interaction terms of the form $-\beta HL$ and δHL. Additionally, the best OccamNet fit includes small constant correction terms of -0.449 and 0.188. These constants have much smaller magnitudes relative to the other terms and may be due to over-fitting. A researcher may choose to either ignore these kinds of correction terms, or instead increase the level of constant regularization in OccamNet to obtain a fit with higher loss but potentially better generalizability.

Epidemic spread. Next, we use OccamNet to discover compartmental models of epidemics. In particular, we use synthetic and real datasets for the well-known Susceptible-Infected-Recovered (SIR) [Kermack and McKendrick, 1927] model for

fixed population sizes, which is given by the ODE system

$$\frac{ds}{dt} = -\beta i, \quad \frac{di}{dt} = \beta is - \gamma i, \quad \frac{dr}{dt} = \gamma i, \tag{4.13}$$

where $s(t)$, $i(t)$, and $r(t)$ represent the proportions of the susceptible, infected, and recovered populations at time t respectively, and where $s + i + r = 1$.

The SIR model and numerous variants are often used to describe the spread of infectious diseases such as measles [Bjørnstad et al., 2002] or COVID-19 [Cooper et al., 2020]. Such models are valuable for predicting the magnitude or duration of an epidemic, estimating epidemiological parameters such as the basic reproduction number [van den Driessche, 2017] (given by $R_0 = \frac{\beta}{\gamma}$ in the SIR model), and for forecasting the impact of public health interventions. Beyond modeling disease spread, SIR variants have also been used to study the diffusion of information on social networks [Woo and Chen, 2016, Kumar and Sinha, 2021], and thus carry substantial relevance to social science.

In Figure 4-15 (c), we simulate a synthetic time-series using the SIR model. OccamNet discovers the correct functional form of the SIR model along with approximately correct constant parameters up to rounding. The deviations of the learned constant parameters likely stem from higher-order errors in the numerical derivative estimate which are not addressed by the central difference formula. One could opt to use higher-order derivative approximations to account for such errors if necessary.

Next, we demonstrate the data ensemble learning functionality of OccamNet on a panel dataset for measles infection data in the UK. Horrocks and Bauch used the SINDy (Sparse Identification of Nonlinear Dynamics) algorithm [Brunton et al., 2016] to fit a variant of the SIR model with time-varying contact rates to this measles dataset as well as to chickenpox and rubella data [Horrocks and Bauch, 2020]. We note that SINDy requires the specification of a pre-set library of possible terms for which the algorithm learns sparse vectors of coefficients. OccamNet instead uses a specified library of simple *bases* to compose more complex terms with no additional inductive bias necessary. Our method requires less prior knowledge about the expres-

sion structure and is thus better suited to deriving new functional forms.

Horrocks and Bauch fit the entire measles time-series to a singular equation for $i(t)$ with time-varying $\beta(t)$ [Horrocks and Bauch, 2020]. We instead demonstrate OccamNet's ability to discover the SIR model for each cycle of the epidemic with different β and γ parameters. Using a subset of the data from 1948-1958, we generate an ensemble of five measles incidence cycles. We then apply the denoising techniques as in [Brunton et al., 2016] and fit each dataset both individually and as an ensemble in which we learn the same functional form for all five datasets with varying constant parameters.

Figure 4-15 (d) highlights the advantage of ensemble learning over individual fits. When each 2-year cycle is fit independently, OccamNet struggles to learn expressions with the SIR-like form of $\frac{di}{dt} = \beta is - \gamma i$. Due to the sparsity and noisiness of each individual dataset, it only extracts the interaction term βis from one of the five periods. The ensemble fit, however, discovers a function that included the key form of $\beta is - \gamma i$, as shown in the last row of the table in Fig. 4-15 (d). The parameter c_0 is a placeholder for a constant that varies for each cycle. Ensemble learning can therefore help avoid over-fitting to individual datasets and improve generalization. While our model also learns higher order terms such as i^2 and $c_0 si^2$ for the ensemble, these terms are of much smaller magnitude compared to the leading terms and are thus of less importance to the correct fit. This is another example in which OccamNet's custom regularization capabilities could be applied to eliminate higher-order terms.

Long-run economic growth. The final dynamical system we consider is the neoclassical Solow-Swan model of economic growth Solow [1956]. The model postulates that long-run economic growth can be explained by capital accumulation, labor growth, and technological progress. The model typically assumes Cobb-Douglas-type aggregate production with constant returns to scale, given by

$$y(t) = k(t)^{\alpha} \tag{4.14}$$

where $y(t)$ is the output per unit of effective labor, $k(t)$ is the capital intensity (capital stock per unit of effective labor), and α is the elasticity of output with respect to capital. The central differential equation in the Solow model describes the dynamics of $k(t)$,

$$\frac{dk}{dt} = sy - (n + g + \delta)k, \tag{4.15}$$

where s is the savings rate, n is the population growth rate, g is the technological growth rate, and δ is the capital stock depreciation rate. This equation states that the rate of change of capital stock is equal to the difference between the rate of investment and the rate of depreciation. The key result of the Solow growth model is that a greater amount of savings and investments do not affect the rate of economic growth in the long run.

While the Solow model was originally derived to describe U.S. economic growth, it has also been applied to other countries. If the model were perfectly universal, we would expect to see every country's capital grow according to equation equation 4.15, with varying hidden parameters. We thus generate a synthetic example of country panel data for economic growth by simulating the Solow equations for $k(t)$ and $y(t)$, with the goal of rediscovering equation equation 4.15. As an additional level of complexity, we model the savings rate s and population growth rate n as time-dependent variables that grow according to the equations $\frac{ds}{dt} = 0.05$ and $\frac{dn}{dt} = 0.05n$. Each panel dataset is generated by randomly sampling initial conditions and parameters g and δ from uniform distributions. The resulting cross-country time series is displayed in Fig. 4-14. As demonstrated in the figure, OccamNet is able to recover the exact equation for $\frac{dk}{dt}$ (denoted as Output 1) with varying parameter c_0 for each of the 20 synthetic panels.

We then attempt to fit the Solow model of capital growth to real-world, noisy country data. In particular, we select macro-economics data from 18 of the original 20 countries in the Organisation for Economic Co-operation and Development (OECD) for which data was available. The input to our symbolic regression includes data on capital per capita, income per capita, savings rate, and population growth

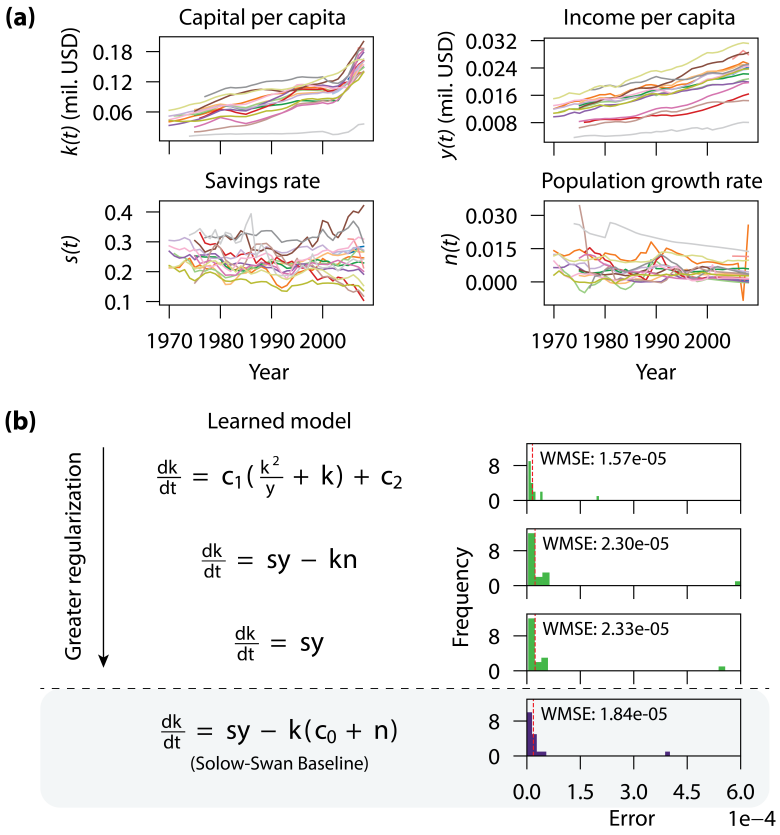

Figure 4-16: Ensemble learning of longitudinal (panel) macroeconomic data. **(a)** Country-level macroeconomic data on capital and income per capita, savings rates, and population growth for 18 OECD member countries. **(b)** Ensemble learning of the Solow economic growth model. The error distribution of the differential equation, applied to each country, is shown for three expressions generated with increasing levels of complexity regularization. The identification of outliers may inform alternative explanations, hidden parameters, or higher-order corrections to the economic model.

compiled by Khoo et al. [Khoo et al., 2021], where it was utilized for regression under the Green Solow model – a variant of Solow-Swan for modeling sustainable growth [Brock and Taylor, 2004]. The $k(t)$ data is originally sourced from the Penn World Tables [Feenstra et al., 2015], $y(t)$ and $n(t)$ are from the Maddison Project database [Maddison, 2017], and $s(t)$ is from the World Development Indicators [wor]. There is no available data for the remaining parameters g and δ, so they are instead treated as learnable constants. We also apply Savitzky-Golay filtering to smooth the data before running the regression.

In Fig. 4-16 (b), we compare the Solow-Swan model baseline to three expressions produced by OccamNet under increasing levels of complexity regularization. The Solow baseline is generated by finding the best-fit parameter $c_0 = g + \delta$ in a least-squares of fit of equation equation 4.15. The expression with no regularization is given by $\frac{dk}{dt} = c_1 \left(\frac{k^2}{y} + k \right) - c_2$ with constants c_1 and c_2 that vary across countries. While this expression has a lower weighted mean-squared error (WMSE) than the baseline, its functional form does not carry immediate economic intuition like the Solow model.

We then add *Constant Regularization* as described in 4.2.6, which results in the equation $\frac{dk}{dt} = sy - kn$ closely matching the functional form of equation 4.15. This suggests that the Solow model has strong external validity as it can be discovered without any strong human priors. Finally, we apply *Activation Regularization* in addition to *Constant Regularization*, resulting in the output $\frac{dk}{dt} = sy$. The learned expression contains only one term from the Solow model and has the highest WMSE. The example in Fig. 4-16 (b) concretely demonstrates the tradeoff between accuracy and simplicity in the discovery of symbolic models. A researcher would thus benefit from running OccamNet with several specifications of regularization to select a result with the desired level of precision and complexity.

Outlook. The fundamental limitation of the discoveries in this section is that we are focused on reconstructing the dynamics that govern the system. Reconstructing the exact dynamics in the real world is a challenging task due to the presence of noise and the lack of understanding of the ground truth. Given that it is hard to

hypothesize and reconstruct the exact dynamics of a system without knowing them, we could still infer useful properties about the dynamics and thus understand the system at a more fundamental level. This knowledge can comprise understanding the invariants of a system or the states of the system that are related to distinguishable properties. This direction of research is the focus of the following work.

Outlook. In this section we explored how we can use OccamNet to uncover the symbolic expression of dynamics in the observed data. However, in many real-world data the dynamics is unknown, and even unnecessary, if we want to disover conserved quantities, as we demonstrate in our work by using optimal transport and manifold learning [Lu et al., 2022]. Contrastive learning is a manifold learning method that attempts to extract the manifold of the original data in a latent space, similarly to [Lu et al., 2022]. It is natural to attempt to understand the latent-space dynamics of contrastive learning, which is the focus of the following section.

4.4 Phase Transitions and Representation Geometry in Contrastive Learning

One of the key problems in modern machine learning is crafting effective representations of data without human-generated labels. Contrastive learning and other self-supervised learning methods are among the most popular and effective methods to date for tackling this problem [Chen et al., 2020b, Chen and He, 2021a, Grill et al., 2020a, Bardes et al., 2022]. In contrastive learning, positive examples, consisting of two augmented version of a single image, have their representations pushed together in the output space; negative examples, which are augmented versions of different images, have their representations pushed apart.

The training dynamics behind contrastive learning and other self-supervised learning methods remain mysterious. We focus on studying the augmentations necessary for contrastive learning. The exact choice and strength of augmentations is critical for contrastive learning [Chen et al., 2020b, Tian et al., 2020a, Luo et al., 2023].

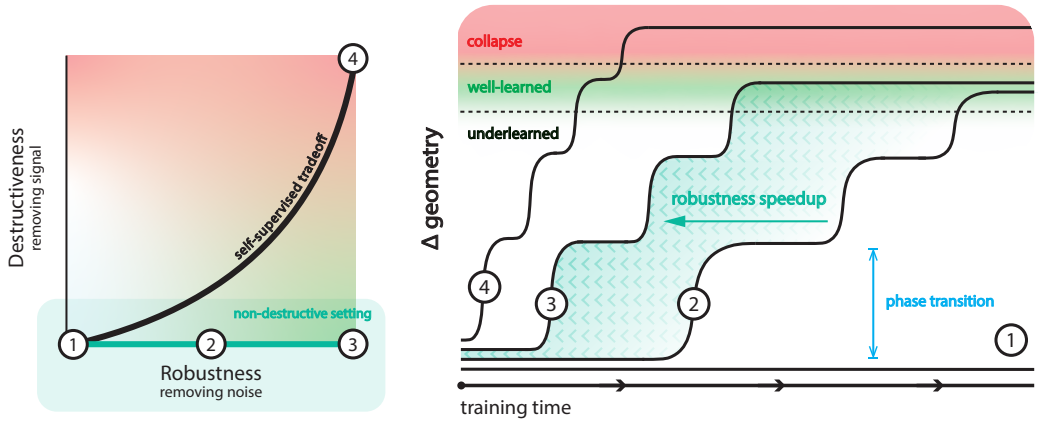

Figure 4-17: Main result. Augmentations used in contrastive learning typically exhibit a tradeoff between robustness (how much noise is removed) and destructiveness (how much signal is removed). In order to explore the effect of robustness alone, we study non-destructive augmentations which preserve all relevant data. We observe in multiple settings that increased robustness speeds up training by speeding up the occurrence of phase transitions, points in training at which representation geometry changes suddenly and significantly. Each numbered point in the first plot corresponds to the augmentations of the idealized training curve on the right.

Augmentations that are too strong destroy meaningful information and encourage *collapse*, where all data points are pushed together. On the other hand, it is commonly understood that augmentations that are too weak preserve too much extraneous information, which lowers the quality of the representations [Tian et al., 2020a].

With an ideal representation in mind, we propose refining our understanding of data augmentations based on the kind of information that augmentations remove. If they are destroying noise or meaningless information, they are *robust*; if they are destroying signal or meaningful information, they are *destructive*. We stress that robustness and destructiveness are both defined in relation to an ideal representation. Typically, strong augmentations are both robust and destructive, while weak augmentations are neither. Generally, we seek to find augmentations that are maximally robust but minimally destructive. The tradeoff between robustness and destructiveness was explored extensively in [Tian et al., 2020a].

Can we study the robustness of augmentations—and their effect on training dynamics—disentangled from their destructiveness? Moreover, can we understand how improving augmentations past the current state-of-the-art will impact training dynamics? We

explore these questions by varying the robustness of non-destructive augmentations, that is, augmentations that completely preserve meaningful information. We do this in two settings: first, we generate artificial datasets from physics simulations and introduce a non-destructive temporal augmentation; and second, we perform supervised contrastive learning [Khosla et al., 2020] on common computer vision benchmarks including CIFAR-10 [Krizhevsky and Hinton, 2009] and ImageNet [Deng et al., 2009b]. In the supervised contrastive learning setting, positive examples are two different examples of the same class, so augmentations are non-destructive.

The self-supervised nature of contrastive models, where representations are learned without a source of ground truth, means that there possibly are multiple stable representations corresponding to alternate interpretations of the input data. As contrastive models typically need long training times, when training they may be converting between these local optima. Understanding the timing and nature of such *phase transitions* between local optima can help further our understanding of contrastive learning dynamics. Phase transitions have been previously observed in simple contrastive learning experiments (e.g. those by [Tian, 2022]), but have not yet been documented in detail.

We observe that the timing of phase transitions is highly dependent on the robustness of augmentations. That is, strongly robust augmentations lead to phase transitions occurring early in training, and as augmentations are made less robust, phase transitions occur later and later in training before not occurring at all. As well as quantitative metrics, we use direct, low-dimensional embeddings to directly observe phase transitions. In particular, we aim to study the *geometric features* of representations as models train, and aim to design metrics that describe these geometric features in order to track phase transitions, as opposed to performance metrics such as loss or downstream accuracy.

Contributions. By utilizing an empiral perspective, our study reveals important insights about contrastive learning. As illustrated in Figure 4-17, our contributions include the following:

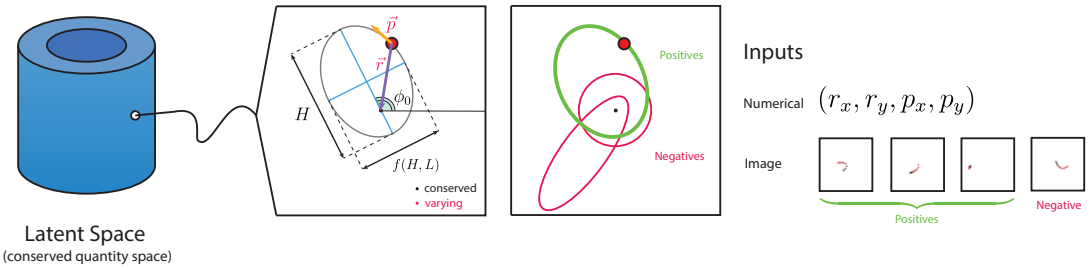

Figure 4-18: How the data generation for our Kepler dataset works. Images on bottom right are real examples sourced from the dataset.

- We emphasize that augmentations have both *robustness*, the amount they are able to remove meaningless information, and *destructiveness*, the amount they remove meaningful information. We study robustness disentangled from destructiveness.
- We provide novel observations of *phase transitions* wherein representation geometry drastically transforms between locally metastable conformations in a relatively short period of training time on synthetic physics datasets (Kepler and double pendulum) as well as on standard vision benchmarks (CIFAR-10 and ImageNet). The double pendulum and CIFAR-10 results will be included in the supplemental materials.
- We identify that a large factor in the timing of phase transitions is augmentation robustness. In particular, robust augmentations speed up phase transitions, and thus training.

Next, we demonstrate how we understand the contrastive learning dynamics for the Kepler problem. We limit the discussion for the Kepler problem, for brevity.

Kepler problem. We demonstrate the dataset in Figure 4-18. The trajectory of a planet orbiting around a central star constrained in two dimensions can be rigorously parameterized by Kepler's equations. The resulting elliptical orbit shape can be uniquely defined by fixing 3 conserved scalar quantities in the system's equations; for example, energy H, angular momentum $\|\mathbf{L}\|$, and the angle of the Laplace-Runge-Lenz vector ϕ_0. These conserved quantites can be calculated from the position $\mathbf{r}$ and

momentum $\mathbf{p}$ with

$$H = \frac{\|\mathbf{p}\|^2}{2} - \frac{1}{\|\mathbf{r}\|} \quad \mathbf{L} = \mathbf{r} \times \mathbf{p} \quad \phi_0 = \arg(\mathbf{p} \times \mathbf{L} - \hat{\mathbf{r}})$$

where $\arg(\mathbf{v})$ is the angle of vector $\mathbf{v}$ with respect to the positive x-axis.

Generating partial trajectories The amount of a trajectory that is sampled to produce positive examples is directly related to augmentation robustness. Two points sampled from distant points within the trajectory have much greater differences in position and momentum compared to points in close proximity within the trajectory. Sampling from a *partial* trajectory corresponds to a less robust augmentation.

Similar to the supervised contrastive learning setup, we use a hyperparameter α to quantify the augmentation robustness. In this case, we sample positives from a proportion of α of the trajectory. If T be the period of the trajectory, we randomly sample a starting time t from $[0, T]$ uniformly where T is the period of the orbit. Then we sample positive examples uniformly from the range $[t, t + \alpha T]$. Note that $\alpha = 1$ corresponds to full trajectories.

R^2 metric for phase transitions in non-categorical datasets All of our physics datasets are non-categorical. Theoretically, the optimal representation is an affine transform of the latent space [Zimmermann et al., 2021]. Thus, given that we know the latent space and the parameterization used to generate our data, the quality of a linear regression from our learned representations to the latent variable(s) reflects the quality of our representation. We use the R^2 of this linear regression to track phase transitions; typically, we calculate the R^2 of a linear regression from the representation to the last-learned linear latent variable, as the order of learning for latents tends to be stable across initializations.

Phase Transitions in the Kepler problem: "Twisted Disk" to "The Bowl." When $\alpha = 1.0$ on the Kepler dataset, all three conserved quantities are present in the representation, and moreover, the representation correctly represents the geometry

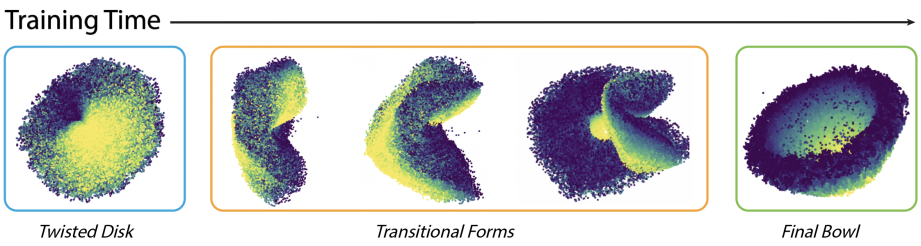

Figure 4-19: Phase transition from Twisted Disk to Bowl showing transitory intermediate embedding geometry.

of the latent space (ϕ_0, H, L) in nontrivial ways. We have nicknamed this particular representation *The Bowl* because of its shape. A more detailed discussion of this representation is the Supplemental Materials. We can weaken the robustness α down to around 0.02 and the representation, other than degrading somewhat in quality, remains roughly the same.

However, when we set $\alpha \leq 0.01$, the shape of the representations looks radically different. The representations remain stuck at a shape that we call the *Twisted Disk*, which embeds position directly and has a twist at the center at which orbits with $L \approx 0, 1$ are embedded close together (Figure 4-20). Despite being able to achieve low loss with partial trajectories, the disk is misleading about the global structure of the data, and tends to have worse than random loss when re-evaluated on full trajectories. Visually, orbits maintain their elliptical shape; the disk is essentially a direct embedding of the input space.

When training with α between 0.02 and 0.15, we can directly observe a phase transition between the twisted disk, which is reached first, and the bowl, which is reached after some number of epochs. Visually, transitional forms between the two phases can be seen in Figure 4-19. By using the linear R^2 metric, we can track when the phase transition occurs and how this depends on the robustness α in Figure 4-21. Within an individual training run, we can observe how the R^2 metric suddenly and surprisingly increases rapidly at the phase transition time Figure 4-22, corresponding to the visual changes.

Kepler Image domain. We created a version of the Kepler dataset that has 56×56 image inputs where the planet's trajectory is displayed as a trail of dark

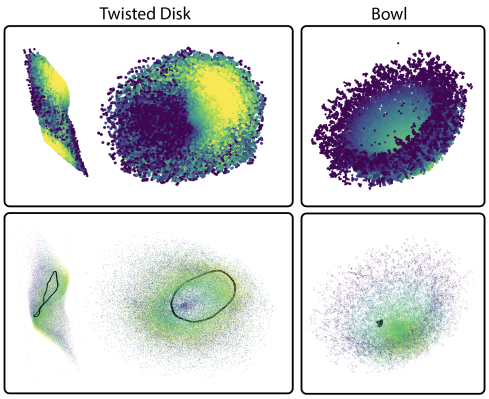

Figure 4-20: Visual comparison showing poor alignment in Twisted Disk, but not in the Bowl. Black outline is embedding of a single orbit.

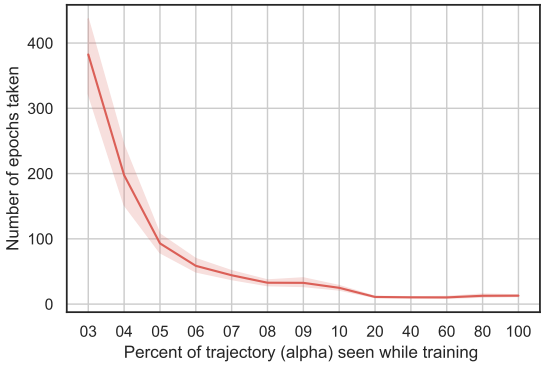

Figure 4-21: Number of epochs needed until the phase transition for Kepler charted against percentage of trajectory seen during training. 16 trials for each α plotted, with a 95% confidence interval shown.

points, and we use a ResNet-18 encoder to extract conserved quantities. The bowl is recovered with α as little as 0.05. The same phenomenon occurs in the image domain, except there are multiple phases and generally more complicated training dynamics. A detailed treatment of this dataset, as well as the impact of missing data, can be found in the Supplemental Materials.

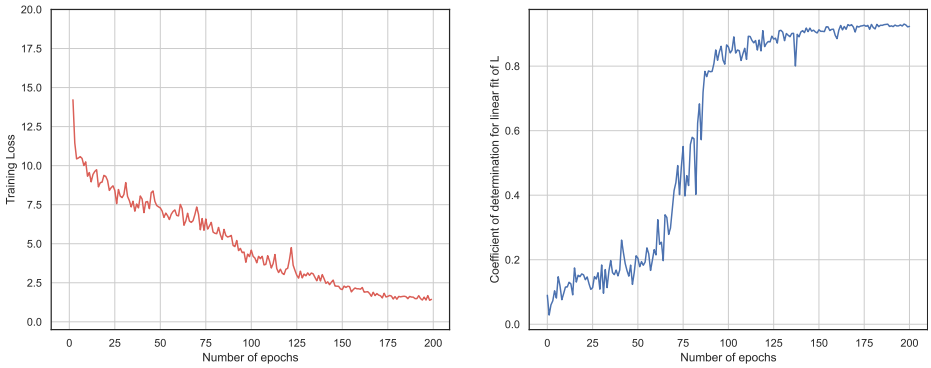

Figure 4-22: The phase transition in this run on the Kepler dataset can be observed from epochs 65-85 where there is a sudden and surprising increase in the R^2 metric. At the same time, loss does not sharply decrease.

4.5 Conclusion

Chronologically, in this chapter, we presented a novel method for symbolic regression based on the inductive bias of neural networks and Occam's razor [Costa et al., 2021]. We then presented an application of this method to the discovery of quantitative and formal models in the social sciences [Balla et al., 2022]. Then we shifted gears to discover phase transitions in the representations dynamics of contrastive learning [Cy et al., 2023]. Additionally, in a parallel study, we also make a contribution to understanding how representations from separate architectures can be connected, and we help interpret the compositionality of intermediate representations in neural network architectures [Hernandez et al., 2023]. Furthermore, in [Lu et al., 2022] we use manifold learning to obtain representations that reveal conserved quantities of a variety of dynamical systems, without requiring knowledge about the equations governing the system.

For future work, we are developing a series of modifications to OccamNet and are exploring the method for a wider range of real-world data and scenarios. We have focused on the domain of discovering learning algorithms in the brain, in the style of an "automated neuroscientist." The challenges of OccamNet are multiple: OccamNet needs to discover laws on stateful environments, has to handle discovering high-dimensional operations, and is able to evaluate likelihoods when working in probabilistic environments. We elaborate more on this topic in Chapter 5.

Chapter 5

Beyond the Limitations of *Representation Learning Through the Lens of Science*. Promising Directions of Future Work.

In this thesis, we explored representation learning, particularly the challenges posed by its conventional method: supervised learning. In Chapter 2, we tackled data limitations using the symmetry inductive bias. In Chapter 3, we discussed enhancing transferability through the language inductive bias. Chapter 4 delved into improving interpretability with the symbolic inductive bias. While our *Representation Learning Through the Lens of Science* methodology addresses key challenges, there remain unexplored areas and potential improvements. This chapter aims to outline and design extensions to our established approach.

In the following sections, we explore the three inductive biases central to our work: symmetry, language, and symbolic.

5.1 Future Work on the Symmetry Inductive Bias: *Multimodality*

Our work on symmetry explored how to embed the learned representations of data with structure so that the transformations in the input are accurately reflected in the representations. Invariant representations, representations that do not change when the transformations are varied, are useful for downstream tasks and overcome many challenges with supervised learning. On the Internet, and in the real world, we perceive the same phenomenon from many different modalities—image, text, audio, etc. The invariance of the representations to the modality becomes an important property and a useful training signal. The abundance of multimodal data makes it an obvious canditate for creating the next generation of representation learning.

Helen Keller was an American author, political activist, and lecturer who, despite being deaf and blind from a young age, became a symbol of courage and perseverance [Keller, 1903]. Helen Keller's understanding of the world was profoundly shaped by two key modalities. The first, *tactile sensation*, allowed her initial perceptions, while the second, *symbolic representation*, introduced by her teacher Anne Sullivan, connected these sensations to tangible objects and ideas. This multi-faceted approach to learning echoes the principles of multimodal learning in machine learning. In this field, algorithms process multiple modalities synchronously, akin to Keller's synthesis of touch and symbolism, to learn better representations.

Recent advances in multimodal representation learning embed visual, text, and audio in a common space, enabling state-of-the-art downstream performances and novel capabilities such as zero-shot learning. We propose a similar approach to learning a common embedding space for different modalities of scientific data. A popular approach (CLIP [Radford et al., 2021]; LiT [Zhai et al., 2022]) is based on contrastive learning, which trains neural network encoders for each modality and connects the embedding spaces of each modality by minimizing the distance between aligned embeddings and maximizing the distance between embeddings that are unaligned (here, an aligned pair is constructed by joining descriptions of a single material using multi-

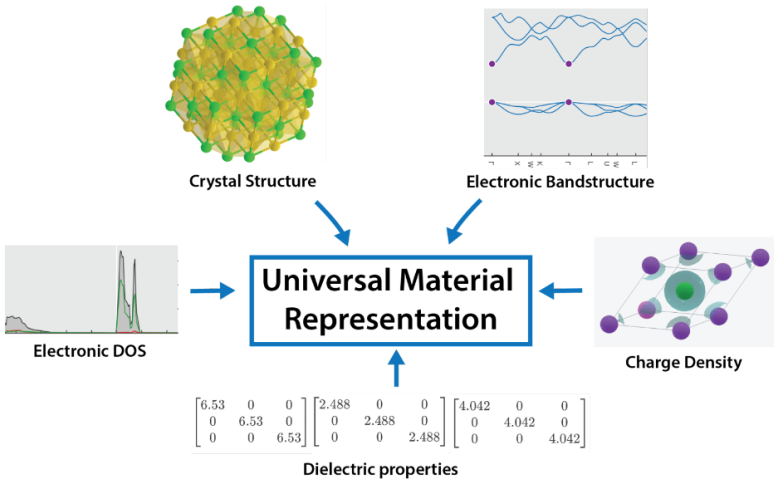

Figure 5-1: Multimodal representation learning for crystaline materials.

ple different modalities—for example, the crystal structure and dielectric properties of graphene). Unfortunately, the prevalent studies on multimodal learning are limited and unexplored in two notable ways: i) they are often conducted on a lot of aligned pairs (about 100's of M) and (ii) they are limited to connecting only a few modalities — about 2-3.

We argue that the field of materials science offers an ideal experimental testing ground for addressing the above limitations. Moreover, multimodal learning could lead to dramatically improved AI models for crystalline materials science, enabling improved analysis and discoveries of new crystalline materials, like new semiconductors (for enhanced computers, solar-cells, lasers), superconductors, and much more. In recent years, materials databases have witnessed unprecedented growth, and a key feature of these databases is that they contain a variety of properties - and thus modalities - for each material. These properties are labeled via either ab-initio computational tools (e.g. DFT) or through experimental data; some notable examples include crystal and electronic structures, thermodynamic and magnetic properties, and beyond. Many of these modalities are very rich in structure, i.e. they cannot be represented by single numerical values, and are thus amenable to representation learning. Furthermore, data for each modality appear in varying numbers and are often limited in size, making them a good fit to study (i) and (ii) above.

Proposed Outcomes

- **Universal Representations for Material Science**

  The main goal of this proposal is to build effective representations for materials, aided by the availability of a variety of properties or modalities (see Figure 5-1 for a sketch of the concept). This would enable accelerated property prediction for new scientific tasks of interest (e.g. predicting a new property not previously used for training), potentially using only a small number of labels. Representation learning for materials inherits challenges unique to the structures of materials. While there have been significant efforts towards representation learning for molecules, efforts towards crystalline materials have been relatively limited. Unlike molecules, many materials (ionic and metallic) lack well-defined bonds and have periodic structures which render graph neural networks, typically used for molecules, not well suitable. We can leverage our advances in contrastive learning for materials science [Loh et al., 2021, Dangovski et al., 2021a] to encourage periodicity into neural network encoders, and thus make them more suitable to the domain of materials. Furthermore, crystalline materials have modalities rich in structure; designing new architectures/ frameworks to capture the necessary inductive biases to effectively encode all of the modalities is an open problem and will be studied in this proposal. Our universal representations also resemble a Foundation Model for materials science, albeit foundation models typically use orders of magnitude more data than what is available to us.

- **Materials Discovery: Examples in High Critical-Temperature (High-Tc) Superconductivity**

  Combining these effective representations with a generative process would further enable rapid inverse design, where new materials of desired properties can be more rapidly discovered or optimized for. However, inverse design of materials is challenging because generative models trained to output crystal structures with certain properties need to satisfy several constraints; for example,

the AI-proposed structure needs to be thermally stable and experimentally viable. Progress toward generative modeling of crystalline materials (and not molecules) remains limited. While there are many properties of interest to machine learning for crystalline materials which we intend to explore, there is a particularly strong impetus towards the discovery of high-Tc superconductors (those that stay superconductive even at as high temperatures as possible). Discovering materials that are superconducting at room temperature has been one of the most important problems in physics and would potentially trigger an industrial revolution in fields as diverse as lossless electricity delivery over long distances, transportation (very efficient magnetically levitating trains), quantum computers, etc. Effective representations from materials can be used in conjunction (e.g. through further fine-tuning) with existing superconductivity datasets to accelerate superconductivity discovery. While discovering high-Tc superconductors may be an ambitious goal, we believe the tools we build in our work can contribute large strides toward solving this problem.

- **Advancing Multimodal Learning in Other Domains**

  Most existing work in multimodal learning has focused on visual, text, and audio modalities. Due to the growing ubiquity of such modalities on the Internet, advances in multimodal learning are promising avenues for developing training methods of deep neural network models. In handling multiple modalities, an alternative approach to contrastive learning (i.e. aligning embedding spaces of all modalities) is an "end-to-end" approach, which predicts one modality from another using an encoder-decoder neural network architecture. There is currently an insufficient understanding of which method - contrastive or end-to-end - is preferred. Most of our understanding is limited to empirical studies, e.g. (CLIP [Radford et al., 2021]; LiT [Zhai et al., 2022]), so it is imperative to deepen our understanding of multimodal learning. By leveraging our domain expertise in materials science and the availability of multimodal datasets in the domain, we aim to increase our understanding of the effectiveness of contrastive

vs. end-to-end training, as well as to address research directions (i) and (ii). We also plan to study multimodal learning beyond materials science by exploring medical imaging and radio reports, as well as various weather data in the domain of stormy events and tornado observations.

5.2 Future Work on the Language Inductive Bias: *Large Language Models for Science Education and Research.*

The power of language for transfer learning is manifested by the recent revolution of *Large Language Models* (LLMs) [Brown et al., 2020, OpenAI, 2023, Chowdhery et al., 2022, Anil et al., 2023]. LLMs are increasingly general in that they can respond to a wide range of tasks, formulated in language. Increasing the size of the model and training data yields an increasing number of capabilities, termed emergent, as they could hardly be predicted as a function of the described hyperparameters (data and model sizes) [Wei et al., 2022].

However, certain languages remain elusive for LLMs. While programming and natural language instructions are well-followed by LLMs, the language of natural sciences is less so. The development of LLMs for science is of primary importance for scaling scientific efforts to novel discoveries in the natural sciences.

Can we create our own LLM that is as general as the leading LLMs, yet fine-tuned on a well-curated curriculum from the natural sciences? Our goal is to investigate the capabilities of such custom LLMs. Could they become "tutor-helpers" in the classroom?

In Figure 5-2 we present the result of our preliminary experiments that research the above questions. We fine-tuned our own LLaMA [Touvron et al., 2023], an open-source LLM that is close to the state-of-the-art proprietary LLMs. The fine-tuning was conducted on a customized dataset including concept and technical questions that are publicly available on the Internet, and ones that are curated from the physics

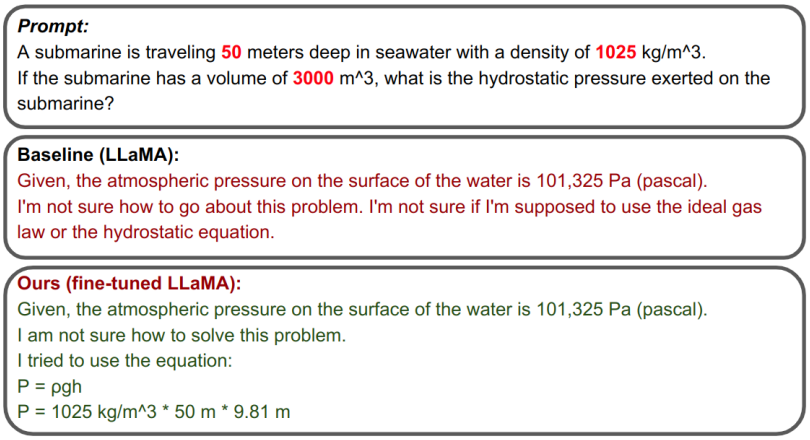

Figure 5-2: Our fine-tuning of LLaMA on an example question. The LLM can understand the question and remember the correct formula for the hydrostatic pressure exerted on the submarine.

curriculum at MIT. While the capabilities of the model are still low on Bloom's taxonomy, there is a lot of room for improvement, and we are currently scaling up our efforts.

The flexibility of the tutor-helpers that we develop will determine their usefulness in the classroom, and the student-tutor interactions will generate data that can further increase the usefulness of our custom-trained LLMs by continual training. Our proposal and experiments have the potential to rethink the way we approach transfer learning and education in the physical sciences. Furthermore, having a tutor-helper may potentially be extended to a "research advisor," which may revolutionize the way we approach scientific research.

5.3 Future Work on the Symbolic Inductive Bias: *OccamNet as Representation in Novel Domains*

In Chapter 4 we presented OccamNet [Costa et al., 2020] a neural network with symbolic representations that allows us to explore the combinatorially large space of

expressions that describe phenomena that can be expressed with symbolic laws. There is one major limitation of OccamNet that presents an opportunity for importing the method and its applications to novel real-world tasks of symbolic regression.

OccamNet is an *on-policy* method. More specifically, all of the equations that the model explores are generated by the model's own distribution, which may be suboptimal, leading the model to focus on the exploitation of its own distribution and get stuck in local optima. To address that, we propose two directions for future work:

1. Adding exploration to OccamNet. This could be as simple as having a small probably (a hyperparameter) to sample connections from a uniform distribution in order to sample paths from OccamNet. This exploration trick is widespread in the reinforcement learning literature [Sutton and Barto, 2018]. In our preliminary experiments with OccamNet, we found that exploration is crucial in being able to discover theories of how the brain works (in synthetic environments). That is, we were able to rediscover the classical Rescorla-Wagner [Rescorla and Wagner, 1972], TD- and Q-learning [Sutton and Barto, 2018] theories of the brain [Sutton and Barto, 2018].

2. Using an LLM to configure OccamNet. Our proposal is to have LLMs configure the weights of OccamNet by using a special token representation as the weights at each layer of the LLM. This will ensure that a (potentially) different OccamNet can be utilized at each layer of the LLM, and the outputs of the OccamNet can be used as additional representations to the LLM. OccamNet will provide a symbolic representation of the LLM that could be useful in solving problems that require mathematical manipulation of the input.

   In Figure 5-3 we present a design of how OccamNet can be used to address the issues that LLMs face with the parsing, using, and transforming of numerical representations. There are three main points of this design. In point a. we demonstrate that we preserve the representational power of Transformers. In particular, we could explore fine-tuning a pre-trained LLM. Point b. demon-

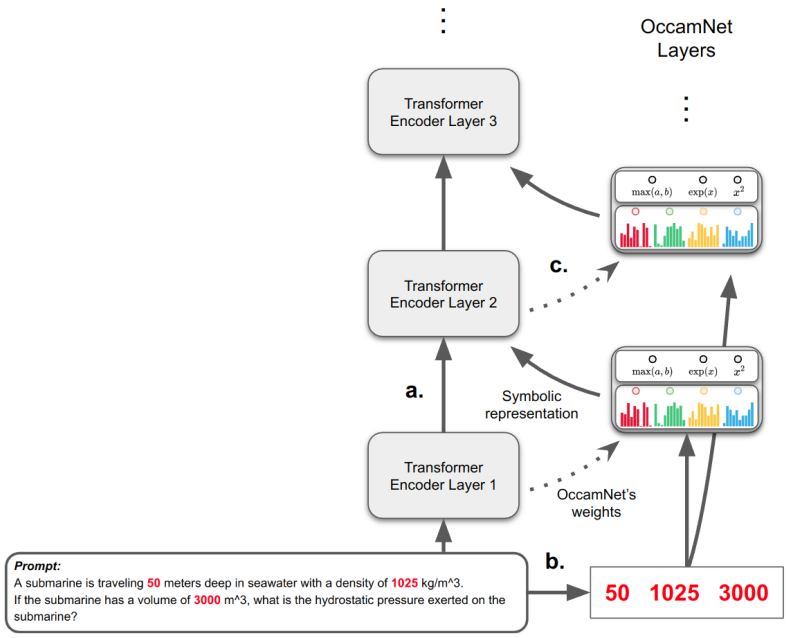

Figure 5-3: OccamNet as a representation for LLMs. **a.** The standard information flow for Transformers where the encoder layers updates the representations of each token in the input sequence. **b.** Extracting the numerical tokens from the tokenized input. **c.** A special token representation from the LLM is used to configure the weights of an OccamNet that receives the numerical tokens as input and produces a numerical output that is used by the following encoder layer. The process is repeated for all the layers of the Transformer.

strates how we can parse the input to extract the numerical inputs. Point c. demonstrates how the LLM can use special representations from its intermediate layer to configure the weights of OccamNet and thus have OccamNet to implement a symbolic expression. The output of that expression is used back in the intermediate representations of the LLM. Thus, not only may we be able to address the issues of exploitation in OccamNet, but also we may be able to help LLMs have more useful mathematical representations.

5.4 Final Remarks

We have identified three directions for future work on representation learning through the lens of science: *multimodal learning*, for the symmetry inductive bias; *LLMs for*

science, for the language inductive bias; *OccamNet representations for LLMs*, for the symbolic inductive bias. There are certain similarities between the three directions, which we will discuss next.

Firstly, all of the above benefit from or manifest multimodal learning. In the natural sciences, such as classical mechanics in physics, diagrammatic representations are crucial for procedural problem solving. Hence, multimodal learning can be very impactful in building LLMs for science. Likewise, the numerical tokens represented by OccamNet is a novel modality for the OccamNet representations for LLMs.

Secondly, experimentation on scientific environments and tasks is crucial to develop the novel representation learning methods that we proposed above. For example, without experimenting with the automating neuroscience task, we may not have identified the need for exploration (vs. exploitation) in OccamNet as clearly as our proposal suggests.

Finally, the language representation of the data may be a crucial component in the development of all proposals. Multimodal learning can be formulated as all modalities, expressed in language representation being processed by a single Transformer architecture. Furthermore, OccamNet's discrete symbolic representation may interface with the symbolic discrete representation of language, and help us develop better inference algrithms for LLMs by directly augmenting the input of the LLM with symbolic mathematical representations during the inference stage.

The core of our thesis and the proposals for future work in this chapter bridge developments in representation learning and science to establish fruitful areas of research in Representation Learning Through the Lens of Science.

Bibliography

World Development Indicators | DataBank. URL `https://databank.worldbank.org/source/world-development-indicators`.

Harold Abelson and Andrea DiSessa. *Turtle geometry: The computer as a medium for exploring mathematics*. MIT press, 1986.

Amjad Abu-Jbara and Dragomir Radev. Coherent citation-based summarization of scientific paper. In *ACL*, 2011.

David Adger. *Language unlimited: The science behind our most creative power*. Oxford University Press, USA, 2019.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1081. URL `https://aclanthology.org/S16-1081`.

Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015.

Roee Aharoni, Melvin Johnson, and Orhan Firat. Massively multilingual neural machine translation. In *NAACL-HLT*, 2019.

P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5 (1):54–65, 1994.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi

Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report, 2023.

Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

Dirk V. Arnold and Nikolaus Hansen. A (1+1)-CMA-ES for constrained optimisation. In *GECCO*, 2012.

Sébastien M R Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. learn2learn: A library for Meta-Learning research. August 2020. URL `http://arxiv.org/abs/2008.12284`.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *NeurIPS Deep Learning Symposium*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. 2023.

Julia Balla, Sihao Huang, Owen Dugan, Rumen Dangovski, and Marin Soljacic. Ai-assisted discovery of quantitative and formal models in social science. *arXiv preprint arXiv:2210.00563*, 2022.

Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. In *ICLR*, 2016.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *International Conference on Learning Representations*, 2022. URL `https://arxiv.org/abs/2105.04906`.

Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. In *EMNLP-IJCNLP*, 2019.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Alan A. Berryman. The Orgins and Evolution of Predator-Prey Theory. *Ecology*, 73(5):1530–1535, 1992. ISSN 0012-9658. doi: 10.2307/1940005. URL `https://www.jstor.org/stable/1940005`. Publisher: Ecological Society of America.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural Symbolic Regression that Scales. *arXiv e-prints*, art. arXiv:2106.06427, June 2021. doi: 10.48550/arXiv.2106.06427.

Ottar N. Bjørnstad, Bärbel F. Finkenstädt, and Bryan T. Grenfell. Dynamics of Measles Epidemics: Estimating Scaling of Transmission Rates Using a Time Series SIR Model. *Ecological Monographs*, 72(2):169–184, 2002. ISSN 0012-9615. doi: 10.2307/3100023. URL `https://www.jstor.org/stable/3100023`. Publisher: Ecological Society of America.

Armand Borel and Jacques Tits. Groupes réductifs. *Publications Mathématiques de l'IHÉS*, 27:55–151, 1965.

William A. Brock and M. Scott Taylor. The Green Solow Model. Working Paper 10557, National Bureau of Economic Research, June 2004. URL `https://www.nber.org/papers/w10557`. Series: Working Paper Series.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, April 2016. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1517384113. URL `https://pnas.org/doi/full/10.1073/pnas.1517384113`.

Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel S. Weld. Tldr: Extreme summarization of scientific documents. *arXiv preprint arXiv:2004.15011*, 2020.

Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

Fredrik Carlsson, Amaru Cuba Gyllensten, Evangelia Gogoulou, Erik Ylipää Hellqvist, and Magnus Sahlgren. Semantic re-tuning with contrastive tension. 2021. URL `https://openreview.net/forum?id=Ov_sMNau-PF`.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

Rich Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. Deep communicating agents for abstractive summarization. In *NAACL-HLT*, 2018.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001. URL `https://aclanthology.org/S17-2001`.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, 2018.

Muthu Kumar Chandrasekaran, Michihiro Yasunaga, Dragomir Radev, Dayne Freitag, and Min-Yen Kan. Overview and results: Cl-scisumm shared task 2019. In *BIRNDL 2019*, 2019.

Muthu Kumar Chandrasekaran, Guy Feigenblat, Eduard Hovy, Abhilasha Ravichander, Michal Shmueli-Scheuer, and Anita de Waard. Overview and insights from the shared tasks at scholarly document processing 2020: CL-SciSumm, LaySumm and LongSumm. In *Proceedings of the First Workshop on Scholarly Document Processing*, 2020.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-graber, and David M. Blei. Reading tea leaves: How humans interpret topic models. In *NeurIPS*. 2009.

Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. *arXiv e-prints*, art. arXiv:1603.02754, March 2016.

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, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*, 2020b. URL `https://arxiv.org/abs/2002.05709`.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *Conference on Computer Vision and Pattern Recognition*, 2021a. URL `https://arxiv.org/abs/2011.10566`.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021b.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised visual transformers. *arXiv e-prints*, pages arXiv–2104, 2021.

Yen-Chun Chen and Mohit Bansal. Fast abstractive summarization with reinforce-selected sentence rewriting. In *ACL*, 2018.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *AISTATS*, 2015.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Thomas Christensen, Charlotte Loh, Stjepan Picek, Domagoj Jakobović, Li Jing, Sophie Fisher, Vladimir Ceperic, John D. Joannopoulos, and Marin Soljačić. Predictive and generative machine learning models for photonic crystals. *Nanophotonics*, 9(13):4183–4192, October 2020. ISSN 2192-8614. doi: 10.1515/nanoph-2020-0197. URL `https://www.degruyter.com/document/doi/10.1515/nanoph-2020-0197/html`.

Thomas Christensen, Hoi Chun Po, John D. Joannopoulos, and Marin Soljačić. Location and topology of the fundamental gap in photonic crystals, 2021.

Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljačić, Shang-Wen Li, Wen-tau Yih, Yoon Kim, and James Glass. Diffcse: Difference-based contrastive learning for sentence embeddings. *arXiv preprint arXiv:2204.10298*, 2022.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *NAACL-HLT*, 2018.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

Mark Collier and Joeran Beel. Implementing neural turing machines. In *ICANN*, page 94–104, 2018.

Ed Collins, Isabelle Augenstein, and Sebastian Riedel. A supervised approach to extractive summarization of scientific papers. In *CoNLL*, 2017.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, 2017.

Ian Cooper, Argha Mondal, and Chris G. Antonopoulos. A SIR model assumption for the spread of COVID-19 in different communities. *Chaos, Solitons & Fractals*, 139:110057, October 2020. ISSN 09600779. doi: 10.1016/j.chaos.2020.110057. URL `https://linkinghub.elsevier.com/retrieve/pii/S0960077920304549`.

Allan Costa, Rumen Dangovski, Owen Dugan, Samuel Kim, Pawan Goyal, Marin Soljačić, and Joseph Jacobson. Fast neural models for symbolic regression at scale. *arXiv preprint arXiv:2007.10784*, 2020.

Allan Costa, Rumen Dangovski, Owen Dugan, Samuel Kim, Pawan Goyal, Marin Soljačić, and Joseph Jacobson. Fast Neural Models for Symbolic Regression at Scale. *arXiv:2007.10784 [cs, stat]*, July 2021. URL `http://arxiv.org/abs/2007.10784`. arXiv: 2007.10784.

John N Crossley, Anthony Wah-Cheung Lun, et al. *The nine chapters on the mathematical art: Companion and commentary*. Oxford University Press, USA, 1999.

Ali Cy, Anugrah Chemparathy, Michael Han, Rumen Dangovski, Peter Y. Lu, and Marin Soljacic. Studying phase transitions in contrastive learning with physics-inspired datasets. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023. URL `https://openreview.net/forum?id=djssHWljSA`.

Rumen Dangovski, Li Jing, Preslav Nakov, Mićo Tatalović, and Marin Soljačić. Rotational unit of memory: a novel representation unit for rnns with scalable applications. *Transactions of the Association for Computational Linguistics*, 7:121–138, 2019a.

Rumen Dangovski, Li Jing, Preslav Nakov, Mićo Tatalović, and Marin Soljačić. Rotational unit of memory: a novel representation unit for RNNs with scalable applications. *TACL*, 7:121–138, 2019b.

Rumen Dangovski, Li Jing, Charlotte Loh, Seungwook Han, Akash Srivastava, Brian Cheung, Pulkit Agrawal, and Marin Soljačić. Equivariant contrastive learning. *arXiv preprint arXiv:2111.00899*, 2021a.

Rumen Dangovski, Michelle Shen, Dawson Byrd, Li Jing, Desislava Tsvetkova, Preslav Nakov, and Marin Soljačić. We can explain your research in layman's terms: Towards automating science journalism at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12728–12737, 2021b.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *ICML*, 2017.

Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600 (7887):70–74, 2021.

Jacob Delvin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009a.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009b.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

Anna Divoli, Preslav Nakov, and Marti Hearst. Do peers see more in a paper than its authors? *Adv. Bioinformatics*, 2012:750214:1–750214:15, 2012.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Owen M Dugan, Peter Y Lu, Rumen Dangovski, Di Luo, and Marin Soljacic. Q-flow: Generative modeling for differential equations of open quantum dynamics with normalizing flows. In *International Conference on Machine Learning*, pages 8879–8901. PMLR, 2023.

Sergey Edunov, Alexei Baevski, and Michael Auli. Pre-trained language model representations for language generation. In *NAACL-HLT*, 2019.

Albert Einstein. *Die grundlage der allgemeinen relativitätstheorie*, volume 49. JA Barth, 1922.

Aaron Elkiss, Siwei Shen, Anthony Fader, Günes Erkan, David States, and Dragomir Radev. Blind men and elephants: What do citation summaries tell us about a research article? *J. Am. Soc. Inf. Sci. Technol.*, 59:51–62, 2008.

Kevin Ellis, Lucas Morales, Mathias Sablé-Meyer, Armando Solar-Lezama, and Josh Tenenbaum. Learning libraries of subroutines for neurally–guided bayesian program induction. In *NIPS*. 2018a.

Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. Learning to infer graphics programs from hand-drawn images. In *NIPS*. 2018b.

Kevin Ellis, Maxwell Nye, Yewen Pu, Felix Sosa, Josh Tenenbaum, and Armando Solar-Lezama. Write, execute, assess: Program synthesis with a REPL. In *NeurIPS*. 2019.

Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. Emergent communication in a multi-modal, multi-step referential game. *arXiv preprint arXiv:1705.10369*, 2017.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *ACL*, 2019.

Angela Fan, David Grangier, and Michael Auli. Controllable abstractive summarization. *NMT at ACL*, 2018a.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *ACL*, 2018b.

Angela Fan, Mike Lewis, and Yann Dauphin. Strategies for structuring story generation. In *ACL*, 2019.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

Robert C. Feenstra, Robert Inklaar, and Marcel P. Timmer. The next generation of the penn world table. *American Economic Review*, 105(10):3150–82, October 2015. doi: 10.1257/aer.20130954. URL `https://www.aeaweb.org/articles?id=10.1257/aer.20130954`.

Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross Girshick, and Kaiming He. A large-scale study on unsupervised spatiotemporal representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3299–3309, 2021.

Zeyu Feng, Chang Xu, and Dacheng Tao. Self-supervised representation learning by rotation feature decoupling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10364–10374, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, 2012.

Adam Foster, Rattana Pukdee, and Tom Rainforth. Improving transformation invariance in contrastive representation learning. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=NomEDgIEBwE`.

Martin Fowler. *Domain Specific Languages*. Addison-Wesley Professional, 1st edition, 2010.

Giancarlo Gandolfo. The Lotka-Volterra Equations in Economics: An Italian Precursor. *Economia Politica*, XXIV:343–348, December 2007.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *ICML*, 2017.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8059–8068, 2019.

Alexios Gidiotis and Grigorios Tsoumakas. A divide-and-conquer approach to the summarization of academic articles. *IEEE/ACM TASLP*, 28:3029–3050, 2020.

John M Giorgi, Osvald Nitski, Gary D Bader, and Bo Wang. Declutr: Deep contrastive learning for unsupervised textual representations. *arXiv preprint arXiv:2006.03659*, 2020.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for GPUs. In *ICML*, 2017.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen. King, C. Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *CoRR*, abs/2006.07733, 2020a. URL `https://arxiv.org/abs/2006.07733`.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020b.

Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *NAACL-HLT*, 2018.

Ligong Han, Seungwook Han, Shivchander Sudalairaj, Charlotte Loh, Rumen Dangovski, Fei Deng, Pulkit Agrawal, Dimitris Metaxas, Leonid Karlinsky, Tsui-Wei Weng, et al. Constructive assimilation: Boosting contrastive learning performance through view generation strategies. *arXiv preprint arXiv:2304.00601*, 2023.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. 2015.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016b.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

Danah Henriksen, Michael Henderson, Edwin Creely, Sona Ceretkova, Miroslava Černochová, Evgenia Sendova, Erkko T Sointu, and Christopher H Tienken. Creativity and technology in education: An international perspective. *Technology, Knowledge and Learning*, 23:409–424, 2018.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Adriano Hernandez, Rumen Dangovski, Peter Y Lu, and Marin Soljacic. Model stitching: Looking for functional similarity between representations. *arXiv preprint arXiv:2303.11277*, 2023.

Geoffrey E Hinton and Richard Zemel. Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Benjamin D Horne, William Dron, Sara Khedr, and Sibel Adali. Assessing the news landscape: A multi-module toolkit for evaluating the credibility of news. In *WWW*, 2018.

Jonathan Horrocks and Chris T. Bauch. Algorithmic discovery of dynamic models from infectious disease data. *Scientific Reports*, 10(1):7061, December 2020. ISSN 2045-2322. doi: 10.1038/s41598-020-63877-w. URL `http://www.nature.com/articles/s41598-020-63877-w`.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

Kokil Jaidka, Muthu Kumar Chandrasekaran, Sajal Rustagi, and Min-Yen Kan. Overview of the CL-SciSumm 2016 shared task. In *BIRNDL*, 2016.

Kokil Jaidka, Muthu Kumar Chandrasekaran, Devanshu Jain, and Min-Yen Kan. The CL-SciSumm shared task 2017: Results and key insights. In *BIRNDL*, 2017.

Kokil Jaidka, Michihiro Yasunaga, Muthu Kumar Chandrasekaran, Dragomir Radev, and Min-Yen Kan. The CL-SciSumm shared task 2018: Results and key insights. In *BIRNDL at SIGIR*, 2018.

Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1):011002, 2013.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić. Tunable efficient unitary neural networks (eunn) and their application to rnns. In *International Conference on Machine Learning*, pages 1733–1741. PMLR, 2017.

Li Jing, Rumen Dangovski, and Marin Soljacic. Waveletnet: Logarithmic scale efficient convolutional neural networks for edge devices. *arXiv preprint arXiv:1811.11644*, 2018.

J. D. Joannopoulos, S. G. Johnson, J. N. Winn, and R. D. Meade. *Photonic Crystals: Molding the Flow of Light*. Princeton University Press, 2 edition, 2008. URL `http://ab-initio.mit.edu/book/`.

Steven G. Johnson and J. D. Joannopoulos. Block-iterative frequency-domain methods for Maxwell's equations in a planewave basis. *Optics Express*, 8(3):173–190, January 2001. ISSN 1094-4087. doi: 10.1364/OE.8.000173. URL `https://www.osapublishing.org/oe/abstract.cfm?uri=oe-8-3-173`.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Helen Keller. *The Story of My Life*. Doubleday, Page & Company, 1903.

William Ogilvy Kermack and Anderson G McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

Zi-Yu Khoo, Kang Hao Lee, Zhibo Huang, and Stéphane Bressan. Neural Ordinary Differential Equations for the Regression of Macroeconomics Data Under the Green Solow Model. In Christine Strauss, Gabriele Kotsis, A. Min Tjoa, and Ismail Khalil, editors, *Database and Expert Systems Applications*, Lecture Notes in Computer Science, pages 78–90, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86472-9. doi: 10.1007/978-3-030-86472-9_7.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

Matthew Khoury, Rumen Dangovski, Longwu Ou, Preslav Nakov, Yichen Shen, and Li Jing. Vector-vector-matrix architecture: A novel hardware-aware framework for low-latency inference in nlp applications. *arXiv preprint arXiv:2010.08412*, 2020.

Samuel Kim, Peter Y Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Čeperić, and Marin Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE transactions on neural networks and learning systems*, 32(9):4166–4177, 2020a.

Samuel Kim, Peter Y. Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Čeperić, and Marin Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2020b.

Samuel Kim, Peter Y. Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljačić. Scalable and Flexible Deep Bayesian Optimization with Auxiliary Information for Scientific Problems. *arXiv:2104.11667*, April 2021a. URL `http://arxiv.org/abs/2104.11667`.

Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. Self-guided contrastive learning for bert sentence representations. *arXiv preprint arXiv:2106.07345*, 2021b.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. pages 3294–3302, 2015. URL `https://papers.nips.cc/paper/2015/hash/f442d33fa06832082290ad8544a8da27-Abstract.html`.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *technical report*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Pawan Kumar and Adwitiya Sinha. Information diffusion modeling and analysis for socially interacting networks. *Social Network Analysis and Mining*, 11(1):11, December 2021. ISSN 1869-5450, 1869-5469. doi: 10.1007/s13278-020-00719-7. URL `https://link.springer.com/10.1007/s13278-020-00719-7`.

Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *SIGIR*, 1995.

Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. Neural random-access machines. In *ICLR*, 2016.

William La Cava, Lee Spector, and Kourosh Danai. Epsilon-Lexicase Selection for Regression. In *GECCO*, 2016.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2016. URL `https://arxiv.org/abs/1612.01474`.

Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *ICLR*, 2020.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

Yann LeCun, L'eon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE 1998 computer society conference on computer vision and pattern recognition*, pages 586–591. IEEE, 1998a.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *IEEE*, 1998b.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015a.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015b.

Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 667–676, 2017.

Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-mix: A domain-agnostic strategy for contrastive representation learning. In *ICLR*, 2021.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

Guy Lev, Michal Shmueli-Scheuer, Jonathan Herzig, Achiya Jerbi, and David Konopnicki. Talksumm: A dataset and scalable annotation method for scientific paper summarization based on conference talks. In *ACL*, 2019.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, 2019.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.733. URL https://aclanthology.org/2020.emnlp-main.733.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017.

Isaac Liao, Rumen R Dangovski, Jakob N Foerster, and Marin Soljačić. Learning to optimize quasi-newton methods. *arXiv preprint arXiv:2210.06171*, 2022.

Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL*, 2003.

Hui Lin and Vincent Ng. Abstractive summarization: A survey of the state of the art. In *AAAI*, 2019.

Boyuan Liu, Steven G. Johnson, John D. Joannopoulos, and Ling Lu. Generalized Gilat-Raubenheimer method for density-of-states calculation in photonic crystals. *Journal of Optics*, 20(4):044005, April 2018a. ISSN 2040-8978, 2040-8986. doi: 10.1088/2040-8986/aaae52. URL http://arxiv.org/abs/1711.07993.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating Wikipedia by summarizing long sequences. In *ICLR*, 2018b.

Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *EMNLP-IJCNLP*, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. 2018. URL https://openreview.net/forum?id=rJvJXZb0W.

Charlotte Loh, Thomas Christensen, Rumen Dangovski, Samuel Kim, and Marin Soljacic. Surrogate- and invariance-boosted contrastive learning for data-scarce applications in science, 2021. URL https://arxiv.org/abs/2110.08406.

Charlotte Loh, Thomas Christensen, Rumen Dangovski, Samuel Kim, and Marin Soljačić. Surrogate-and invariance-boosted contrastive learning for data-scarce applications in science. *Nature Communications*, 13(1):4223, 2022a.

Charlotte Loh, Rumen Dangovski, Shivchander Sudalairaj, Seungwook Han, Ligong Han, Leonid Karlinsky, Marin Soljacic, and Akash Srivastava. On the importance of calibration in semi-supervised learning. *arXiv preprint arXiv:2210.04783*, 2022b.

Charlotte Loh, Seungwook Han, Shivchander Sudalairaj, Rumen Dangovski, Kai Xu, Florian Wenzel, Marin Soljacic, and Akash Srivastava. Multi-symmetry ensembles: Improving diversity and generalization via opposing symmetries. *arXiv preprint arXiv:2303.02484*, 2023.

Ilya Loshchilov and Frank Hutter. CMA-ES for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*, 2016.

Alfred J. Lotka. Contribution to the Theory of Periodic Reactions. *The Journal of Physical Chemistry*, 14(3):271–274, March 1910. ISSN 0092-7325, 1541-5740. doi: 10.1021/j150111a004. URL `https://pubs.acs.org/doi/abs/10.1021/j150111a004`.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through L_0 regularization. In *ICLR*, 2018.

Peter Y Lu, Rumen Dangovski, and Marin Soljačić. Discovering conservation laws using optimal transport and manifold learning. *arXiv preprint arXiv:2208.14995*, 2022.

Di Luo, Jiayu Shen, Rumen Dangovski, and Marin Soljačić. Koopman operator learning for accelerating quantum optimization and machine learning. *arXiv preprint arXiv:2211.01365*, 2022.

Rundong Luo, Yifei Wang, and Yisen Wang. Rethinking the effect of data augmentation in adversarial contrastive learning. In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=0qmwFNJyxCL`. under review.

Kelvin Luu, Rik Koncel-Kedziorski, Kyle Lo, Isabel Cachola, and Noah A. Smith. Citation text generation. *arXiv preprint arXiv:2002.00317*, 2020.

Andrew Ma, Yang Zhang, Thomas Christensen, Hoi Chun Po, Li Jing, Liang Fu, and Marin Soljacic. Topogivity: A machine-learned chemical rule for discovering topological materials. *Nano Letters*, 23(3):772–778, 2023.

D. A. MacLulich. *Fluctuations in the Numbers of the Varying Hare (Lepus Americanus)*. University of Toronto Press, 1937. ISBN 978-1-4875-8178-7. URL `http://www.jstor.org/stable/10.3138/j.ctvfrxkmj`.

Angus Maddison. Maddison Database 2010, November 2017. URL https://www.rug.nl/ggdc/historicaldevelopment/maddison/releases/maddison-database-2010.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

Andreas Madsen and Alexander Rosenberg Johansen. Neural arithmetic units. In *ICLR*, 2020.

Ofer Malcai, Ofer Biham, Peter Richmond, and Sorin Solomon. Theoretical analysis and simulations of the generalized Lotka-Volterra model. *Physical Review E*, 66(3):031102, September 2002. doi: 10.1103/PhysRevE.66.031102. URL https://link.aps.org/doi/10.1103/PhysRevE.66.031102. Publisher: American Physical Society.

Daniel J Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, et al. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023.

Simone Manti, Mark Kamper Svendsen, Nikolaj R Knøsgaard, Peder M Lyngby, and Kristian S Thygesen. Exploring and machine learning structural instabilities in 2d materials. *npj Computational Materials*, 9(1):33, 2023.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. A sick cure for the evaluation of compositional distributional semantic models. In *Lrec*, pages 216–223. Reykjavik, 2014.

Benjamin T. Martin, Stephan B. Munch, and Andrew M. Hein. Reverse-engineering ecological theory from data. *Proceedings of the Royal Society B: Biological Sciences*, 285(1878):20180422, May 2018. ISSN 0962-8452. doi: 10.1098/rspb.2018.0422. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5966606/.

Georg Martius and Christoph H. Lampert. Extrapolation and learning equations. *arXiv e-prints*, art. arXiv:1610.02995, October 2016.

Georg Martius and Christoph H Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016.

Robert I Mckay, Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O'neill. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11(3-4):365–396, 2010.

Qiaozhu Mei and ChengXiang Zhai. Generating impact-based summaries for scientific literature. In *COLING*, 2008.

Vinicius V. Melo, Danilo Vasconcellos Vargas, and Wolfgang Banzhaf. Batch Tournament Selection for Genetic Programming. In *GECCO*, 2019.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. SymPy: symbolic computing in python. *Peer J Computer Science*, 3:103, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

Tom M Mitchell. *The need for biases in learning generalizations*. Citeseer, 1980.

Saif Mohammad, Bonnie Dorr, Melissa Egan, Ahmed Hassan, Pradeep Muthukrishan, Vahed Qazvinian, and Dragomir Radev. Using citations to generate surveys of scientific paradigms. In *ACL-HLT*, 2009.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *ICML*, 2017.

David J. Montana and Lawrence Davis. Training feedforward neural networks using genetic algorithms. In *IJCAI*. Morgan Kaufmann Publishers Inc., 1989.

T. Nathan Mundhenk, Daniel Ho, and Barry Y. Chen. Improvements to context based self-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Preslav Nakov, Ariel Schwartz, and Marti Hearst. Citances: Citation sentences for semantic analysis of bioscience text. In *SIGIR workshop: Search and Discovery in Bioinformatics*, 2004.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *CoNLL*, 2016.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, 2017.

Hidetsugu Nanba, Noriko Kando, and Manabu Okumura. Classification of research papers using citation links and citation types: Towards automatic review article generation. In *ASIS SIG/CR*, 2000.

Isaac Newton. *Philosophiae Naturalis Principia Mathematica*. Jossu Societatis Regiae ac Typis Josephi Streater, 1687.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

Nikola Nikolov, Michael Pfeiffer, and Richard Hahnloser. Data-driven summarization of scientific articles. In *LREC*, 2018.

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.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, 10(1):36, 2017.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

OpenAI. Gpt-4 technical report, 2023.

Patryk Orzechowski, William La Cava, and Jason H. Moore. Where are we now? A large benchmark study of recent symbolic regression methods. In *GECCO*, 2018.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, and Nathan Ng. FAIRSEQ: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT*, 2019.

Seymour A Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic books, 2020.

Giorgio Parisi. Toward a mean field theory for spin glasses. *Physics Letters A*, 73(3): 203–205, 1979.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *ICLR*, 2018.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL-HLT*, 2018a.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 2018b.

Brenden K Petersen, Mikel Landajuela Larma, Terrell N. Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *ICLR*, 2021.

John Peurifoy, Yichen Shen, Li Jing, Yi Yang, Fidel Cano-Renteria, Brendan G DeLacy, John D Joannopoulos, Max Tegmark, and Marin Soljačić. Nanophotonic particle simulation and inverse design using artificial neural networks. *Science advances*, 4(6):eaar4206, 2018.

Riccardo Poli, William B Langdon, Nicholas F McPhee, and John R Koza. *A field guide to genetic programming*. lulu.com, 2008.

Vahed Qazvinian and Dragomir R. Radev. A supervised approach to extractive summarization of scientific papers. In *COLING*, 2008.

Yurui Qu, Li Jing, Yichen Shen, Min Qiu, and Marin Soljacic. Migrating knowledge between physical scenarios based on artificial neural networks. *ACS Photonics*, 6 (5):1168–1174, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

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 *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Guillem Ramírez, Rumen Dangovski, Preslav Nakov, and Marin Soljačić. On a novel application of wasserstein-procrustes for unsupervised cross-lingual learning. *arXiv e-prints*, pages arXiv–2007, 2020.

Colorado Reed, Sean Metzger, Aravind Srinivas, Trevor Darrell, and Kurt Keutzer. Evaluating self-supervised pretraining without using labels. In *CVPR*, 2021.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

Robert A Rescorla and Allan R Wagner. A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. *Classical conditioning II: Current research and theory*, 2:64–99, 1972.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

Sebastian Ruder. *Neural Transfer Learning for Natural Language Processing*. PhD thesis, National University of Ireland, Galway, 2019.

Ileana Rugina, Rumen Dangovski, Li Jing, Preslav Nakov, and Marin Soljačić. Data-informed global sparseness in attention mechanisms for deep neural networks. *arXiv preprint arXiv:2012.02030*, 2020.

Ileana Rugina, Rumen Dangovski, Mark Veillette, Pooya Khorrami, Brian Cheung, Olga Simek, and Marin Soljačić. Meta-learning and self-supervised pretraining for real world image translation. *arXiv preprint arXiv:2112.11929*, 2021.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

B Russell, Antonio Torralba, K Murphy, and W Freeman. Labelme: a database and web-based tool for image annotation. int. *Journal of Computer Vision*, 77, 2007.

Horacio Saggion, Ahmed AbuRa'ed, and Francesco Ronzano. Trainable citation-enhanced summarization of scientific articles. In *BIRNDL*, 2016.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings*

of Machine Learning Research, pages 4442–4450, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018a. PMLR. URL http://proceedings.mlr.press/v80/sahoo18a.html.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. PMLR, 2018b.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

Michael Schmidt and Hod Lipson. *Symbolic Regression of Implicit Equations*, pages 73–85. Springer US, 2010.

Timothy J. Schmit, Paul Griffith, Mathew M. Gunshor, Jaime M. Daniels, Steven J. Goodman, and William J. Lebair. A closer look at the abi on the goes-r series. *Bulletin of the American Meteorological Society*, 98(4):681 – 698, 2017. doi: 10.1175/BAMS-D-15-00230.1. URL https://journals.ametsoc.org/view/journals/bams/98/4/bams-d-15-00230.1.xml.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *ACL*, 2017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL*, 2016.

Jean-Pierre Serre et al. *Linear representations of finite groups*, volume 42. Springer, 1977.

Eva Sharma, Chen Li, and Lu Wang. BIGPATENT: A large-scale datasetfor abstractive and coherent summarization. In *ACL*, 2019.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138*, 2022.

Armando Solar Lezama. *Program Synthesis By Sketching*. PhD thesis, EECS Department, University of California, Berkeley, 2008.

Robert M. Solow. A Contribution to the Theory of Economic Growth. *The Quarterly Journal of Economics*, 70(1):65, February 1956. ISSN 00335533. doi: 10.2307/1884513. URL `https://academic.oup.com/qje/article-lookup/doi/10.2307/1884513`.

Gary Stager. Twenty things to do with a computer, forward 50: future visions of education inspired by seymour papert and cynthia solomon's seminal work. *(No Title)*, 2021.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint arXiv:2103.15316*, 2021.

Sandeep Subramanian, Raymond Li, Jonathan Pilault, and Christopher Pal. On extractive and abstractive neural document summarization with transformer language models. In *EMNLP*, 2020.

Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

Alex Tamkin, Mike Wu, and Noah Goodman. Viewmaker networks: Learning views for unsupervised representation learning. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=enoVQWLsfyL`.

Simone Teufel and Marc Moens. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Comput. Linguist.*, 28(4):409–445, December 2002.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. *Neural Information Processing Systems*, 2020a. URL `https://arxiv.org/abs/2005.10243`.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *arXiv preprint arXiv:2005.10243*, 2020b.

Yuandong Tian. Understanding deep contrastive learning via coordinate-wise optimization. *Neural Information Processing Systems*, 2022. URL `https://arxiv.org/abs/2201.12680`.

Antonio Torralba, Rob Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008. doi: 10.1109/TPAMI.2008.128.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. In *NIPS*. 2018a.

Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. *Advances in neural information processing systems*, 31, 2018b.

Vahe Tshitoya, John Dagdelen, and Leigh Weston. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571:95–98, 2019.

George Tucker, Andriy Mnih, Chris J. Maddison, and Jascha Sohl-Dickstein. REBAR: low-variance, unbiased gradient estimates for discrete latent variable models. In *NIPS*, 2017.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16), 2020.

Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. 2020.

Raghuram Vadapalli, Bakhtiyar Syed, Nishant Prabhu, Balaji Vasan Srinivasan, and Vasudeva Varma. When science journalism meets artificial intelligence: An interactive demonstration. In *EMNLP*, 2018.

Pauline van den Driessche. Reproduction numbers of infectious disease models. *Infectious Disease Modelling*, 2(3):288–303, August 2017. ISSN 24680427. doi: 10.1016/j.idm.2017.06.002. URL https://linkinghub.elsevier.com/retrieve/pii/S2468042717300209.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *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. *Advances in neural information processing systems*, 30, 2017b.

Mark Veillette, Siddharth Samsi, and Chris Mattioli. Sevir : A storm event imagery dataset for deep learning applications in radar and satellite meteorology. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22009–22019. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/fa78a16157fed00d7a80515818432169-Paper.pdf.

Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc Le. Towards domain-agnostic contrastive learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10530–10541. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/verma21a.html.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Evan Vogelbaum, Rumen Dangovski, Li Jing, and Marin Soljačić. Contextualizing enhances gradient based meta learning. *arXiv preprint arXiv:2007.10143*, 2020.

Stefan Wagner, Gabriel Kronberger, Andreas Beham, Michael Kommenda, Andreas Scheibenpflug, Erik Pitzer, Stefan Vonolfen, Monika Kofler, Stephan Winkler, Viktoria Dorfer, and Michael Affenzeller. *Advanced Methods and Applications in Computational Intelligence*, volume 6, chapter Architecture and Design of the HeuristicLab Optimization Environment, pages 197–261. Springer, 2014.

Qingyun Wang, Qi Zeng, Lifu Huang, Kevin Knight, Heng Ji, and Nazneen Fatema Rajani. Reviewrobot: Explainable paper review generation based on knowledge synthesis. In *INLG*, 2020.

Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *arXiv preprint arXiv:2104.07713*, 2021.

Yifei Wang, Zhengyang Geng, Feng Jiang, Chuming Li, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Residual relaxation for multi-view representation learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Maurice Weiler and Gabriele Cesa. General $E(2)$-Equivariant Steerable CNNs. *arXiv:1911.08251 [cs, eess]*, November 2019. URL `http://arxiv.org/abs/1911.08251`. arXiv: 1911.08251.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992.

Jiyoung Woo and Hsinchun Chen. Epidemic model for information diffusion in web forums: experiments in marketing exchange and political dialog. *SpringerPlus*, 5 (1):66, December 2016. ISSN 2193-1801. doi: 10.1186/s40064-016-1675-x. URL `http://www.springerplus.com/content/5/1/66`.

Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. In *CVPR*, 2018.

Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. *arXiv preprint arXiv:2008.05659*, 2020.

Eli Yablonovitch. Inhibited Spontaneous Emission in Solid-State Physics and Electronics. *Physical Review Letters*, 58(20):2059–2062, May 1987. doi: 10.1103/PhysRevLett.58.2059. URL `https://link.aps.org/doi/10.1103/PhysRevLett.58.2059`.

Shin'ya Yamaguchi, Sekitoshi Kanai, Tetsuya Shioda, and Shoichiro Takeda. Image enhanced rotation prediction for self-supervised learning. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 489–493. IEEE, 2021.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. Consert: A contrastive framework for self-supervised sentence representation transfer. *arXiv preprint arXiv:2105.11741*, 2021.

Ziyi Yang, Yinfei Yang, Daniel Cer, Jax Law, and Eric Darve. Universal sentence representation learning with conditional masked language model. *arXiv preprint arXiv:2012.14388*, 2020.

Michihiro Yasunaga, Jungo Kasai, Rui Zhang, Alexander R. Fabbri, Irene Li, Dan Friedman, and Dragomir R. Radev. ScisummNet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. In *AAAI*, 2019.

Weizhe Yuan, Pengfei Liu, and Graham Neubig. Can we automate scientific reviewing? *arXiv preprint arXiv:2102.00176*, 2021.

Amir R Zamir, Tilman Wekel, Pulkit Agrawal, Colin Wei, Jitendra Malik, and Silvio Savarese. Generic 3d representation via pose estimation and matching. In *European Conference on Computer Vision*, pages 535–553. Springer, 2016.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. In *NeurIPS*, 2019.

Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18123–18133, 2022.

Liheng Zhang. Equivariance and invariance for robust unsupervised and semi-supervised learning. 2020.

Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2547–2555, 2019.

Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

Yan Zhang, Ruidan He, Zuozhu Liu, Kwan Hui Lim, and Lidong Bing. An unsupervised sentence embedding method by mutual information maximization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1601–1610, 2020.

Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive learning inverts the data generating process. *International Conference on Machine Learning*, 2021. URL `https://arxiv.org/abs/2102.08850`.