Molecular Graph Representation Learning and Generation for Drug Discovery

by

Benson Chen

B.S., University of Pennsylvania (2016)
S.M. Massachusetts Institute of Technology (2019)

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

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Author . . . .
Department of Electrical Engineering and Computer Science
January 26, 2022

Certified by . . . .
Regina Barzilay
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by . . . .
Tommi Jaakkola
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

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

Molecular Graph Representation Learning and Generation for Drug Discovery

by

Benson Chen

Submitted to the Department of Electrical Engineering and Computer Science
on January 26, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Machine learning methods have been widely pervasive in the domain of drug discovery, enabling more powerful and efficient models. Before deep models, modeling molecules was largely driven by expert knowledge; and to represent the complexities of the molecular landscape, these hand-engineered rules prove insufficient. Deep learning models are powerful because they learn the important statistical features of the problem–but only with the correct inductive biases. We tackle this important problem in the context of two molecular problems: representation and generation. Canonical success of deep learning is deeply rooted in its ability to map the input domain into a meaningful representation space. This is especially poignant for molecular problems, where the "right" relations between molecules is nuanced and complex.

The first part of this thesis will focus on molecular representation, in particular, property and reaction prediction. Here, we explore a transformer-style architecture for molecular representation, providing new tools to apply these models to graph-structured objects. Moving away from the traditional graph neural network paradigm, we demonstrate the efficacy of prototype networks for molecular representation, which allows us to reason over learned property prototypes of molecules. Lastly, we look at the molecular representations in the context of improving reaction predictions.

The second part of this thesis will focus on molecular generation, which is crucial in drug discovery as a means to propose promising drug candidates. Here we develop a new method for multi-property molecule generation, by first learning a distributional vocabulary over molecular fragments. Then, using this vocabulary, we survey efficient exploration methods over the chemical space.

Thesis Supervisor: Regina Barzilay
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Tommi Jaakkola
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First and foremost, I would like to thank both of my advisors Regina Barzilay and Tommi Jaakkola for all their guidance over my many years at MIT. They have always given the support I needed, especially during the difficult times of the last two years of my PhD, when covid and quarantine greatly impacted how we work. They have always inspired me to think bigger, and more creatively–to really think critically and work on problems that can have a real impact. Moreover, I really appreciate their compassion that they've shown me during my difficult times. Next, I want to thank Klavs Jensen for being in my thesis committee member, as well as being a great resource for the chemistry projects I've worked on.

I want to thank all my wonderful labmates who have been with me throughout my journey here at MIT–one's experience is very much shaped by the people that one is surrounded by. I really appreciate all the time that we've spent together, from talking about research to bantering about everyday happenstance in our daily lives. I feel very honored to be surrounded by so many smart, driven individuals–thank you everyone.

Lastly, I want to thank my friends new and old, whether we met in-person or virtually. Thank you for encouraging me when I felt down, and supporting me through each part of my journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning has rapidly transformed the traditional pipeline of drug discovery, providing new tools for each step of the process. Many problems, which traditionally require extensive, expert domain knowledge, have been tackled through deep learning tools, making them more efficient and cheaper. Prior cheminformatics methods use many hand-engineered rules to model small molecules. These techniques are utilized to address problems like property prediction, where the the task to predict the properties of a molecule such as potency. However, traditional methods that attempt to tackle these representation problems lack the ability to generalize well due to their inflexible nature. The transformative facet of deep learning models lies within the models' abilities to learn and extract the important features directly from the data. However, this is only possible with the correct structural biases and modeling assumptions underlying the models. Naively applying deep methods on molecular problems can and will limit the capacity or usefulness of the models, hampering their ability to generalize and their usefulness in practice. Therefore, the importance of utilizing the correct inductive biases cannot be understated.

Before the advent of deep learning methods, molecular modeling required heavy engineering and fixed representations commonly known as quantitative structure-activity relationship (QSAR) methods. Within these methods, fingerprint techniques are widely popular, and can be broadly categorized into several types including structure-based [30], topological [1], circular [8] and pharmacophore fingerprints [91].

Some of these fingerprints like the structure-based MACCS [30] fingerprints, are highly specific representations, consisting of an indicator function for a fixed set of predefined structures. Other fingerprints, topological and circular, which include Morgan fingerprints are more flexible. These fingerprints capture local topology by enumerating paths or circular neighborhoods. However, the problem still lies within the deterministic nature of the generating method: if these predefined rules do not capture the right representation for the task, they will not work well. For instance, property cliffs, a phenomenon in which similar molecules exhibit different properties, remain a challenging problem for many small molecule problems. This problem is especially poignant for molecular fingerprints, because the featurization is fixed. However, using deep models is not a bandage for this problem either, as deep models can easily overfit to the data, and offer poor generalization.

Therefore, it is crucial that our deep learning models incorporate the right kinds of structural biases. Graph neural networks operate through an iterative aggregation scheme, wherein at each step, nodes aggregate information from its neighbors. Successively, a node should incorporate more and more information about a larger neighborhood. The node representations are eventually aggregated into a single vector representing the graph. While sometimes effective, this simple paradigm may not always incorporate the right kind of biases for molecular tasks. For instance, this local neighborhood aggregation can fail to capture long-range dependencies that are important when considering properties of molecules. Even more so, perhaps aggregation on the 2-D molecular graph is not suitable for the ideal molecular representation, and we should look at 3-D structures instead. There are many considerations for the development of deep models for molecules, but they require the correct structure to be effective. Fingerprint representations are simple, but they are inflexible and often involve a lot of human-engineered rules. Deep models, on the other hand, can easily overfit and fail to capture the correct structural representations.

1.1 Machine Learning Applications for Drug Discovery

With that in mind, I will next explain how machine learning has been used in drug discovery, and my efforts to tackle these problems using more chemically-inspired models. Drug discovery is a process consisting of several stages, and machine learning is particularly suitable for early-stage drug discovery, searching for new drug molecules. During the discovery phase, high throughput screening (HTS) is conducted on large libraries of molecules, which yields candidate molecules, known as *hits*. These *hit* molecules then undergo more screening and optimization to generate a smaller set of *lead* molecules. Usually, the HTS filters are quite relaxed, only testing for general applicability, while generating *lead* compounds require much more scrutinized evaluation and optimization. Once the *lead* compounds have been further optimized, promising candidates are then tested on animals and later human trials.

The selection of *hit* and *lead* compounds is the ideal frontier for machine learning methods to pave new improvements. In order to search for *hit* compounds, biological assays are conducted to assess the properties for a large library of molecules. Using machine learning, we can perform virtual screening (VS), which is to use computational tools to predict the properties instead of performing actual assays. This allows us to speed up the screening process as well as being able to screen a much larger set of molecules. Prior to machine learning, QSAR methods were broadly applied to virtual screening. In its most basic form, QSAR methods use a variety of hand-engineered descriptors, such as simple features including atom and bond counts, molecular weight and ring information; more complex descriptors include higher-order topological features and physicochemical properties. However, as mentioned earlier, these methods that rely on a lot of hand-engineered features can underfit by not capturing the correct representations, or overfit by introducing many noisy features.

Once *hits* are discovered, they usually need to be optimized. This is a challenging problem because drugs need to satisfy multiple constraints in addition to their biological activity: they have to have exhibit certain properties like water-solubility or non-toxicity. When optimizing for one property, other properties may change, making this a difficult combinatorial problem to solve. Often, chemists test structural analogs [59] of *hits* in order to optimize these molecules. These analogs are obtained through methods like QSAR, and usually require a lot of expert knowledge.

Machine learning tools can greatly enhance the performance of models on these drug discovery tasks. For instance, graph neural networks have already been demonstrated to attain state-of-the-art performances on many property prediction tasks [144]. Perhaps on a more exciting level, machine learning methods can also generate new molecules, instead of relying on the expert knowledge of human chemists. For instance, machine learning methods can pool information from a much larger database of drug molecules, and make more informative optimizations of molecules. But, as these chemical problems are extremely challenging in nature, coupled with the often noisy training data, deep models require careful design to effectively learn and generalize to these domains. This is the focus of my thesis, where I will talk about how we build models that are cognizant of chemical knowledge, and better adapt themselves to how chemists think about these problems in practice.

1.2 Thesis Overview

Next I will overview the layout of this thesis, which will go into detail about my technical contributions to the field. In Chapter 2, I'll introduce the different representations of molecules, and new models for their improvement. In the following chapter (Chapter 3), I will talk about another new graph neural network paradigm that borrows ideas from prototype learning. Chapter 4 will talk about retrosynthesis, and how we can produce accurate and diverse synthesis suggestions. Lastly, Chapter 5 will introduce a new method for molecular optimization.

Chapter 2 delves into one of the most fundamental problems of using deep learning for drug discovery, which is representational learning of graphs. First, I will talk about how molecules are traditionally represented, using structural hashes known as fingerprints. Then, I will proceed to introduce how molecules can be learned in a deep learning framework. To reason across the complex, discrete structure of molecular graphs, we have to impose the right inductive biases. Here, I introduce how we augment typical message-passing neural networks to improve property prediction. I developed a new method that better captures the connectivity of the molecular graph, where local operators commonly used in typical MPNN often prove insufficient.

In Chapter 3, I will discuss a completely different way of parameterizing the learning problem that leverages ideas from prototype learning: the key representational step consists of comparing each input graph to a set of abstract prototypes. Typical MPNN compute node embeddings and aggregate them using simple sum or mean aggregation, which can potentially lose structural or semantic information. I will introduce a new model that learns prototype point clouds, and computes optimal transport (OT) distances to these point clouds in order to reason about the input molecular graph. This is a general framework that can be flexibly applied to any graph neural network, and shows good empirical performance improvements. Moreover, I show that this model has nice interpretability through the learned point clouds.

Next in Chapter 4, I will talk about a different problem in the drug discovery pipeline, which is retrosynthesis, which is a crucial task that aims to predict which reactants are needed to generate a given target molecule. Traditionally, this task has been solved using template-based approaches, which encode transformation rules as regular expressions operating on SMILES strings and are typically extracted directly from the available training reactions. However, these approaches greatly limits the generalizability of the algorithms. I introduce a new method to tackle the retrosynthesis problem, using novel pretraining ideas and introducing a mixture distributional prior. This model shows better generalization than previous template-based and template-free models, and can generate more diverse predictions, as evaluated by human chemists.

Finally in Chapter 5, I will talk about generative modeling for molecular graphs, which is exciting because they provide a medium for machines to do what is traditionally limited to only expert human chemists. Generation of molecules has traditionally been approached by generating atom-by-atom or using a fixed fragment vocabulary. Shifting away from previous frameworks, my model learns a distribution of molecular fragments, and builds molecular graphs through the addition and deletion of molecular fragments from the learned distributional fragment vocabulary. This enables the generative model to span a much larger chemical space than models with a fixed fragment vocabulary. I further introduce a novel generation scheme for molecular optimization which starts the search by translating from known active molecules and store the discovered molecules as new potential initialization states for subsequent searches.

Machine learning has greatly impacted the computational aspects of drug discovery: tackling problems under-explored through traditional statistical methods. However, bland application of deep learning is often unimpressive: the models require the correct domain-knowledge and inductive biases to work well. I show that, with the proper modeling choices, we can greatly improve the performance and usefulness of these models in practice.

Chapter 2

Graph Representation

Property prediction relies on having a robust representation of the graph. The typical models used are named Graph Neural Networks (GNN), which use iterative aggregation to learn the important features from the input molecular graphs. The input featurization often only include the most primitive features such as atom and bond types. This is a stark departure to prior QSAR methods that utilizes hand-engineered features. However, the typical GNN architecture that relies on local aggregation operations can often miss higher-order graph information. In the context of molecular problems, locality can prove myopic, as longer-range dependencies are important for many biological prediction targets. To remedy this, we propose Path-Augmented Graph Transformer Networks (PAGTN) that are explicitly built on longer-range dependencies in graph-structured data. Our model leverages the widely-successful transformer architecture to augment traditional graph neural networks.

2.1 Graph Neural Networks

Graph Convolution Networks (GCN) have successfully been applied to molecular graph datasets [31, 71, 95, 65]. These “message-passing" algorithms exploit the feature locality of graphs through the usage of convolution operations [44]. However, the convolution operator aggregates only local information, so long-range dependencies are naturally difficult for these models to learn. In molecular graphs, many informative structures are characterized by the paths between nodes. We propose the Path-Augmented Graph Transformer Network (PAGTN) model that utilizes these path features in global attention layers, resulting in a richer, more expressive model. Specifically, our model learns a better representation of the graph in the following ways:

Long-range dependencies In GCNs, long-range dependencies take many convolution layers to learn, because feature aggregation happen only within the immediate neighborhoods of each node. For large enough graphs, GCNs may fail to capture these long-range dependencies entirely. Our PAGTN model can more easily capture these dependencies because every node attends to all other nodes in the graph.

Substructures In graph problems, it is imperative for a model to pick up the important substructures in the graph. GCN models necessitate several layers to propagate information and learn these substructures. The advantage of our model is that this interaction can be learned within a single layer.

Transformer architectures have triumphed over traditional recurrent and convolution models in many natural language tasks such as machine translation [131]. While recurrent and convolution models often incorporate a single attention layer at the top [88], it has been shown that using only these globally-connected self-attention layers learns a much more powerful model.

Attention models on graphs have been explored in previous works. Primarily, the Graph Attention Network [133] and its variants [48, 148, 93] aggregates information within local neighborhoods by using attention. We emphasize that our model focuses on the global connectivity of the nodes. Moreover, our model does not use any complex attention mechanism across layers, but rather provides a simple framework using the path features that works well empirically. Another proposed model, Graph Transformer [80], uses global attention layers, but that model does not extend to graphs in which edge and path features are important.

We test our PAGTN model against the GCN model on 7 benchmark moelcular property prediction tasks ranging from quantum chemistry (QM7, QM8, QM9), physical chemistry (ESOL, Lipophilictiy) and biochemistry (BACE, BBBP) [139]. Each dataset focuses on a different property of the molecule, making composition of these datasets highly variable. Nevertheless, our model consistently shows improved performance against the GCN baseline, demonstrating that our model can learn more powerful representations.

2.2 Path-Augmented Transformer Network

In this section, we first briefly overview the Transformer model. Then, we will go over our contributions, describing our variant of the Transformer model that uses path features to learn expressive representations of graphs.

The Transformer model [131], in contrast to traditional recurrent or convolution architectures, consists of fully-connected attention layers. These models use multi-head self-attention, which confers more flexibility for the attention module. The attention layers are connected by position-wise feed-forward layers, with residual links and layer normalization present at each layer.

The transformer model itself has no direct notion of relative position, so it uses positional encodings in the form of sinusoidal functions. However, this form of positional encoding is not possible in graphs, because there is no longer a natural sequential ordering of the nodes. We introduce path features, which represent how two nodes are connected. These path features influence the attention module in the network, so that the node embeddings are globally aware. We first explain how we construct these path features, then how they are incorporated into the attention framework.

We compute the path features between each node pair by taking the shortest path between them. Due to cycles on graphs, these shortest paths may not be unique. For molecular graphs, these cycles arise due to ring substructures on the graphs. Because the edge features are consistent within a single ring or cycle, multiple paths are almost always equivalent feature-wise; therefore, this approach is sensible for our model.

For efficiency, we truncate the path features between nodes up to a distance d apart. We make the assumption that as the distance between two nodes increases, the connectivity between the two nodes matter less. Therefore, this constraint puts

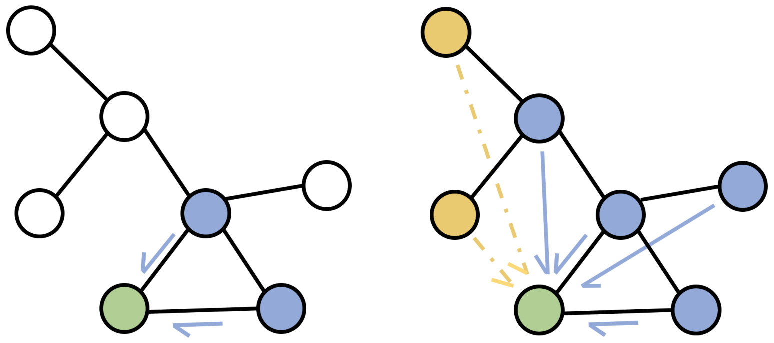


Figure 2-1: Illustration of graph propagation properties for GCN (left) and our PAGTN model (right). For the GCN, the source attention node (green) only attends to its immediate neighbors (blue). In the PAGTN, the source attention node (green) has connectivity information in the form of path features for its local neighborhood, $d = 2$, (blue), but also attends to all other nodes (yellow).

a natural regularizer on the model. So while each node attend to all other nodes in the graph, that node only has rich edge features for a local neighborhood.

The path features between two nodes $i \to j$ is a concantenation of the following three components:

Edge features: are constructed by concatenating the individual bond features of the shortest path between $i \to j$. Let b_k be the bond features of the kth bond along the path, which includes the bond type, conjugacy and ring membership (whether or not that bond is in a ring) features. Then, the edge features are just the concatenation of the features: $[b_1; b_2; ...; b_n]$. Note that if $n > d$, we zero out these features, and if $n < d$, we pad the feature vector with zeros.

Distance: is a one-hot feature of the distance between two nodes $i \to j$, truncated by d.

Ring Membership: is a one-hot feature denoting whether the node i and node j are in the same ring. For molecular graphs, we find that it's also helpful to include one-hot features for specific rings such as five/six-membered aromatic rings. Note that this is distinct from the bond ring membership features which indicates whether a particular bond is part of a ring.

A comparison of the information propagation properties of the network layers is illustrated in Figure fig. 2-1. In regular GCNs, only the direct neighborhood is impacted–which can require many layers of computation to learn from the graph. In our PAGTN model, every node is globally connected, which makes learning complex dependencies easier.

Although transformer models normally use scaled dot-product attention, we found in our experiments that an additive form of attention was easier to train and resulted in better performance. One way we deviate from standard self-attention modules is that we exclude the source node when computing attention for that node. The residual links at each layer grounds the learned embedding at each layer to be representative of the original input node.

Define $\mathbf{x} = (x_1, x_2, ..., x_n) \in \mathbb{R}^{n \times F_n}$ as a matrix of the input node features, where n is the number of nodes and F_n is the number of node features. Similarly, let $\mathbf{p} = (p_{1,1}, p_{1,2} ... p_{n,n}) \in \mathbb{R}^{n \times n \times F_p}$ be a matrix of the input pairwise path features where F_p is the number of path features.

At each layer, we update the node features by computing a weighted average using learned attention weights. Let $\mathbf{h}^l = (h_1^l, h_2^l, ..., h_n^l) \in \mathbb{R}^{n \times F_m}$ represent the node features at layer l, where F_m is the number of model features. Note that the elements of $\mathbf{h}^0$ are the linearly transformed input features ($\mathbf{h^0} = W\mathbf{x^T}$). We compute $s_{i,j}^l$, the attention score of node $i \to j$, as:

$$s_{i,j}^l = W^{S_2}\Big[\text{LeakyReLU}\Big(W^{S_1}[h_i^{l-1}; h_j^{l-1}; p_{i,j}]\Big)\Big] \tag{2.1}$$

The attention probabilities $a_{i,j}$ are calculated as a softmax over the attention scores. As mentioned earlier, we exclude the source node itself when computing the attention probabilities.

$$\alpha_{i,j}^l = \text{softmax}(s_{i,j}^l) = \frac{\exp(s_{i,j}^l)}{\sum_{j' \neq i} \exp(s_{i,j'}^l)} \tag{2.2}$$

Using attention probabilities, we can compute a weighted average over the node features. Since we note the importance of path features in graphs, we define the output features to be a function of both node and path features. Here, σ is some non-linear function (we use ReLU for our experiments).

$$h_i^l = \sigma\big(W^{H_2} h_i^{l-1} + \sum_{j \neq i} \alpha_{i,j}^l W^{H_1} [h_j^{l-1}; p_{i,j}]\big) \tag{2.3}$$

As introduced in [131], multi-head attention can often benefit the model by allowing it more easily to attend to different aspects of the input data. If we split the attention into K heads, we can define the update rule for h_i^l as a function of the embeddings associated with individual heads $h_i^{l,k}$:

$$h_i^l = \Big\|_k \sigma\big(W^{H_2,k} h_i^{l-1,k} + \sum_{j \neq i} \alpha_{i,j}^{l,k} W^{H_1,k} [h_j^{l-1,k}; p_{i,j}]\big) \tag{2.4}$$

Here, $\|$ is the concatenation operator. Empirically, we find that using multi-head attention helps on some tasks, but not on all tasks.

Since we are interested in property prediction tasks for the molecule as a whole, we compute a molecule embedding h_M by aggregating the individual node embeddings. Here, we add a residual link to the input features, $\mathbf{x}$, of the network.

$$h_M = \sum_i \sigma\Big(W^M [h_i^L; x_i]\Big) \tag{2.5}$$

We choose the sum operator to aggregate the feature embeddings, which has higher expressive power than other classic operators [142]. The target property is predicted using a 1-layer MLP with h_M as input.

2.3 Property Prediction Experiments

We test our model on 7 benchmark property prediction tasks, including quantum mechanics (QM7, QM8, QM9), physical chemistry (ESOL, Lipophilicity) and biochemistry (BACE, BBBP) [139].

We split each dataset into 10 different folds of 80:10:10 (train:validation:test) splits, and record the average performance over the folds using the appropriate measure for each dataset. Since these datasets feature markedly different properties, we

Table 2.1: Results comparing our PAGTN model to various baselines. The metrics used were MAE for the quantum mechanics datasets (QM7, QM8, QM9), RMSE for the physical chemistry datasets (ESOL, Lipophilicity), and AUC for the biochemistry datasets (BACE, BBBP). The bold numbers represent the model with the best performance.

Data set	# Data	Metric	MolNet	GCN	PAGTN (Local)	PAGTN (Global)
QM7	6,830	MAE	–[1]	52.4 ± 2.8	48.9 ± 3.4	**47.8 ± 3.0**
QM8	21,786	MAE	.0143	.0105 ± .0003	.0108 ± .0003	**.0102 ± .0003**
QM9	133,885	MAE	2.35	2.20 ± .03	2.10 ± .04	**2.07 ± .05**
ESOL	1,128	RMSE	.580	.587 ± .05	.592 ± .06	**.554 ± .06**
Lipo	4,200	RMSE	.655	.578 ± .05	.592 ± .05	**.572 ± .04**
BACE	1,513	AUC	.867	.878 ± .02	.876 ± .02	**.880 ± .01**
BBBP	2,039	AUC	.729	.907 ± .03	.898 ± .04	**.913 ± .03**

tune the hyperparameters of the model for individual datasets. We compare our transformer to several baselines, which we explain as follows.

MolNet Molecule Net [139] tested many graph-based deep learning methods as well as more conventional methods on these property prediction datasets. We use their top performing model for each dataset.

GCN This is a traditional graph convolution model, and here we use a similar model to [65]. We find that this model achieves very competitive results compared MolNet (which itself uses many different graph-based convolution models), and therefore is a fair baseline. GCN models can have a self-attention layer at the top, but we find empirically that this often hurts performance so we do not include this attention layer in our baseline.

PAGTN (Local) We include a variant of our PAGTN model, which does not attend to nodes for which there are no path features. That is, the model masks out nodes that are further than d from the source attention node. We include this baseline to show that global attention does indeed improve performance.

Our proposed model is dubbed the **PAGTN (Global)**, which attends globally to all nodes.

The results of the property prediction tasks can be seen from Table 2.1. We first

Table 2.2: Results comparing the GCN and the PAGTN (Global) models on a synthetic ring membership prediction task, which is to test whether or not two nodes are in the same ring on the graph. GCN does cannot always predict this property well, while the PAGTN can easily incorporate these features into the model.

Model	Accuracy	AUC
GCN	91.6	96.5
PAGTN (Global)	97.8	99.8

see that the GCN model is very comparable to those of MolNet [139]. And compared to the GCN model, our PAGTN model achieves surperior performance in all 7 of these property prediction tasks, illustrating the broad representational power of the model. Furthermore, we see from the local PAGTN model that by attending globally rather than restricting to the local neighborhood, we always see an improvement in performance. This reveals that the global attention does indeed help the model.

To help elucidate why the PAGTN formulation is better than that of GCN, we turn to a synthetic task. We note that certain properties such as ring membership can prove difficult for regular graph convolution networks. To test this observation, and to demonstrate the effectiveness of our PAGTN model, we create a synthetic dataset by choosing a subset of 5,769 molecules from the property prediction datasets that have at least 2 rings. For each molecule, we randomly choose 5 pairs of atoms that are in the same ring, and 5 pairs of atoms that are in different rings. For atoms in fused ring systems, we count two atoms in the same ring if they are in the smallest possible ring system.

From Table 2.2, we see that the GCN fails to perfectly predict ring membership. This is not surprising as the convolution operation has to learn to disambiguate features of nodes in same and different rings. These subtle but important graph features are imperative for models to fully capture the representation of the graph. Our PAGTN naturally solves this issue, since we can incorporate these features as a part of the network, whereas it is a lot more difficult to incorporate these features in the local convolution model. Note that the PAGTN still does not solve the problem perfectly, and this is due to the fact that in highly symmetrical graphs, multiple nodes are equivalent which leads to ambiguous ring membership as see from Figure 2-2.

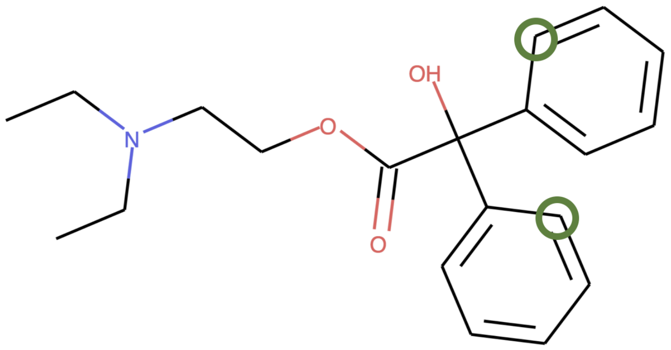


Figure 2-2: The two green-circled atoms are completely symmetric, so their output feature embeddings are equivalent. Since the ring membership prediction is made by aggregating pairwise node features, it is impossible to tell whether any other atom is in the same or different ring from these two atoms.

2.4 Summary

Here, I introduced our the PAGTN model that exploits the connectivity structure of the data in its global attention mechanisms. Through the path features that we engineer into model's attention layers, our model better captures the complex structures of graphs compared to GCNs. On 7 different chemical property prediction tasks, we have shown that our PAGTN model can outperform traditional GCNs. This is one demonstration that we need the correct inductive biases in order to achieve better performance for molecular problems.

Chapter 3

Prototype Learning using Optimal Transport

When chemists reason over molecules, they are rarely doing so in isolation. In fact, chemists often reason about molecules in the context of their repository of knowledge about other familiar molecules. Here, we want to diverge from the traditional paradigm of molecular representation, and instead suggest a new modeling perspective that better captures how we think about molecules in practice.

In fact, current graph neural network (GNN) architectures naively average or sum node embeddings into an aggregated graph representation—potentially losing structural or semantic information. We here introduce OT-GNN, a model that computes graph embeddings using parametric prototypes that highlight key facets of different graph aspects. Towards this goal, we successfully combine optimal transport (OT) with parametric graph models. Graph representations are obtained from Wasserstein distances between the set of GNN node embeddings and "prototype" point clouds as free parameters. Empirically, we address an inherent collapse optimization issue by proposing a noise contrastive regularizer to steer the model towards truly exploiting the OT geometry. Finally, we outperform popular methods on several molecular property prediction tasks, while exhibiting smoother graph representations, and we show that our new way of reasoning about molecules is both well-motivated and interpretable.

3.1 Prototype Learning and Optimal Transport

Recently, there has been considerable interest in developing learning algorithms for structured data such as graphs. For example, molecular property prediction has many applications in chemistry and drug discovery [144, 129]. Historically, graphs were decomposed into features such as molecular fingerprints, or turned into non-parametric graph kernels [136, 120]. More recently, learned representations via graph neural networks (GNNs) have achieved state-of-the-art on graph prediction tasks [32, 27, 72, 144].

Despite these successes, GNNs are often underutilized in whole graph prediction tasks such as molecule property prediction. Specifically, GNN node embeddings are typically aggregated via simple operations such as a sum or average, turning the molecule into a single vector prior to classification or regression. As a result, some of the information naturally extracted by node embeddings may be lost.

Departing from this simple aggregation step, [128] proposed a kernel function over graphs by directly comparing non-parametric node embeddings as point clouds through optimal transport (Wasserstein distance). Their *non-parametric* model yields better empirical performance over popular graph kernels, but this idea hasn't been extended to the more challenging parametric case where optimization difficulties have to be reconciled with the combinatorial aspects of OT solvers.

Motivated by these observations and drawing inspiration from prior work on prototype learning, we introduce a new class of GNNs where the key representational step consists of comparing each input graph to a set of abstract prototypes (fig. 3-1). Our desire is to learn prototypical graphs and represent data by some form of distance (OT based) to these prototypical graphs; however, for the OT distance computation it suffices to directly learn the point cloud that represents each prototype, so learning a graph structure (which would be difficult) is not necessary. In short, these prototypes play the role of basis functions and are stored as point clouds as if they were encoded from real graphs. Each input graph is first encoded into a set of node embeddings using any existing GNN architecture. The resulting embedding point cloud

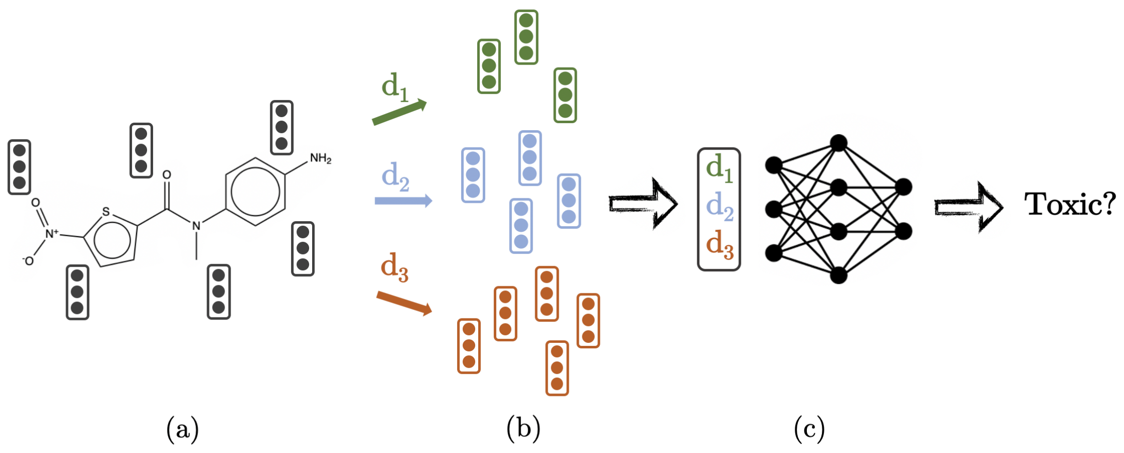


Figure 3-1: Our OT-GNN prototype model computes graph embeddings from Wasserstein distances between (a) the set of GNN node embeddings and (b) prototype embedding sets. These distances are then used as the molecular representation (c) for supervised tasks, e.g. property prediction. We assume that a few prototypes, e.g. some functional groups, highlight key facets or structural features of graphs relevant to a particular downstream task at hand. We express graphs by relating them to these abstract prototypes represented as free point cloud parameters.

is then compared to the prototype embedding sets, where the distance between two point clouds is measured by their Wasserstein distance. The prototypes as abstract basis functions can be understood as keys that highlight property values associated with different graph structural features. In contrast to previous kernel methods, the prototypes are learned together with the GNN parameters in an end-to-end manner.

Our notion of prototypes is inspired from the vast prior work on prototype learning. In our case, prototypes are not required to be the mean of a cluster of data, but instead they are entities living in the data embedding space that capture helpful information for the task under consideration. The closest analogy are the centers of radial basis function networks [18, 106], but we also inspire from learning vector quantization approaches [74] and prototypical networks [124].

Our model improves upon traditional aggregation by explicitly tapping into the full set of node embeddings without collapsing them first to a single vector. We theoretically prove that, unlike standard GNN aggregation, our model defines a class of set functions that is a universal approximator.

Introducing prototype points clouds as free parameters trained using combinatorial optimal transport solvers creates a challenging optimization problem. Indeed, as the models are trained end-to-end, the primary signal is initially available only in aggregate form. If trained as is, the prototypes often collapse to single points, reducing the Wasserstein distance between point clouds to Euclidean comparisons of their means. To counter this effect, we introduce a contrastive regularizer which effectively prevents the model from collapsing, and we demonstrate its merits empirically.

Our contributions. First, we introduce an efficiently trainable class of graph neural networks enhanced with OT primitives for computing graph representations based on relations with abstract prototypes. Second, we train parametric graph models together with combinatorial OT distances, despite optimization difficulties. A key element is our noise contrastive regularizer that prevents the model from collapsing back to standard summation, thus fully exploiting the OT geometry. Third, we provide a theoretical justification of the increased representational power compared to the standard GNN aggregation method. Finally, our model shows consistent empirical improvements over previous state-of-the-art on molecular datasets, yielding also smoother graph embedding spaces.

3.2 Preliminaries

First we go over some related work, followed by some preliminaries on basic graph neural network architecture applicable to our model and optimal transport ideas.

Related Work on Graph Neural Networks. Graph Neural Networks were introduced by [49] and [112] as a form of recurrent neural networks. Graph convolutional networks (GCN) appeared later on in various forms. [32, 4] proposed a propagation rule inspired from convolution and diffusion, but these methods do not scale to graphs with either large degree distribution or node cardinality. [96] defined a GCN as a 1D convolution on a chosen node ordering. [71] also used graph convolutions to generate high quality molecular fingerprints. Efficient spectral methods were proposed by [12, 27]. [72] simplified their propagation rule, motivated from spectral graph theory [54]. Different such architectures were later unified into the message passing neural networks (MPNN) framework by [45]. A directed MPNN variant was later used to improve state-of-the-art in molecular property prediction on a wide variety of datasets by [144]. Inspired by DeepSets [147], [143] propose a simplified, theoretically powerful, GCN architecture. Other recent approaches modify the sum-aggregation of node embeddings in the GCN architecture to preserve more information [75, 103]. In this category there is also the recently growing class of hierarchical graph pooling methods which typically either use deterministic and non-differentiable node clustering [27, 61], or differentiable pooling [145, 100, 40]. However, these methods are still strugling with small labelled graphs such as molecules where global and local node interconnections cannot be easily cast as a hierarchical interaction. Other recent geometry-inspired GNNs include adaptations to non-Euclidean spaces [86, 17, 5, 36], and different metric learning on graphs [107, 6, 80], but we emphasize our novel direction in learning prototype point clouds.

Related Work on Prototype Learning. Learning prototypes to solve machine learning tasks started to become popular with the introducton of generalized learning vector quantization (GLVQ) methods [74, 110]. These approaches perform classification by assigning the class of the closest neighbor prototype to each data point, where Euclidean distance function was the typical choice. Each class has a prototype set that is jointly optimized such that the closest wrong prototype is moved away, while the correct prototype is brought closer. Several extensions [53, 114, 13] introduce feature weights and parameterized input transformations to leverage more flexible and adaptive metric spaces. Nevertheless, such models are limited to classification tasks and might suffer from extreme gradient sparsity.

Closer to our work are the radial basis function (RBF) networks [18] that perform classification/regression based on RBF kernel similarities to prototypes. One such similarity vector is used with a shared linear output layer to obtain the final prediction per each data point. Prototypes are typically set in an unsupervised fashion, e.g. via k-means clustering, or using the Orthogonal Least Square Learning algorithm, unlike being learned using backpropagation as in our case.

Combining non-parametric kernel methods with the flexibility of deep learning models have resulted in more expressive and scalable similarity functions, conveniently trained with backpropagation and Gaussian processes [138]. Learning parametric data embeddings and prototypes was also investigated for few-shot and zero-shot classification scenarios [124]. Last, [29] use distances to prototypes as opposed to p.d. kernels.

In contrast with the above line of work, our research focuses on learning parametric prototypes for graphs trained jointly with graph embedding functions for both graph classification and regression problems. Prototypes are modeled as sets (point clouds) of embeddings, while graphs are represented by sets of unaggregated node embeddings obtained using graph neural network models. Disimilarities between prototypes and graph embeddings are then quantified via set distances computed using optimal transport. Additional challenges arise due to the combinatorial nature of the Wasserstein distances between sets, hence our discussion on introducing the noise contrastive regularizer.

Directed Message Passing Neural Networks (D-MPNN) We briefly remind here of the simplified D-MPNN [25] architecture which was adapted for state-of-the-art molecular property prediction by [144]. This model takes as input a directed graph $G = (V, E)$, with node and edge features denoted by $\mathbf{x}_v$ and $\mathbf{e}_{vw}$ respectively, for v, w in the vertex set V and $v \to w$ in the edge set E. The parameters of D-MPNN are the matrices $\{\mathbf{W}_i, \mathbf{W}_m, \mathbf{W}_o\}$. It keeps track of *messages* $\mathbf{m}_{vw}^t$ and *hidden states* $\mathbf{h}_{vw}^t$ for each step t, defined as follows. An initial hidden state is set to $\mathbf{h}_{vw}^0 := ReLU(\mathbf{W}_i \mathrm{cat}(\mathbf{x}_v, \mathbf{e}_{vw}))$ where "cat" denotes concatenation. Then, the updates are:

$$\mathbf{m}_{vw}^{t+1} = \sum_{k \in N(v) \setminus \{w\}} \mathbf{h}_{kv}^t, \qquad \mathbf{h}_{vw}^{t+1} = ReLU(\mathbf{h}_{vw}^0 + \mathbf{W}_m \mathbf{m}_{vw}^{t+1}) \tag{3.1}$$

where $N(v) = \{k \in V | (k, v) \in E\}$ denotes v's incoming neighbors. After T steps of message passing, node embeddings are obtained by summing edge embeddings:

$$\mathbf{m}_v = \sum_{w \in N(v)} \mathbf{h}_{vw}^T, \quad \mathbf{h}_v = ReLU(\mathbf{W}_o \text{cat}(\mathbf{x}_v, \mathbf{m}_v)). \tag{3.2}$$

A final graph embedding is then obtained as $\mathbf{h} = \sum_{v \in V} \mathbf{h}_v$, which is usually fed to a multilayer perceptron (MLP) for classification or regression.

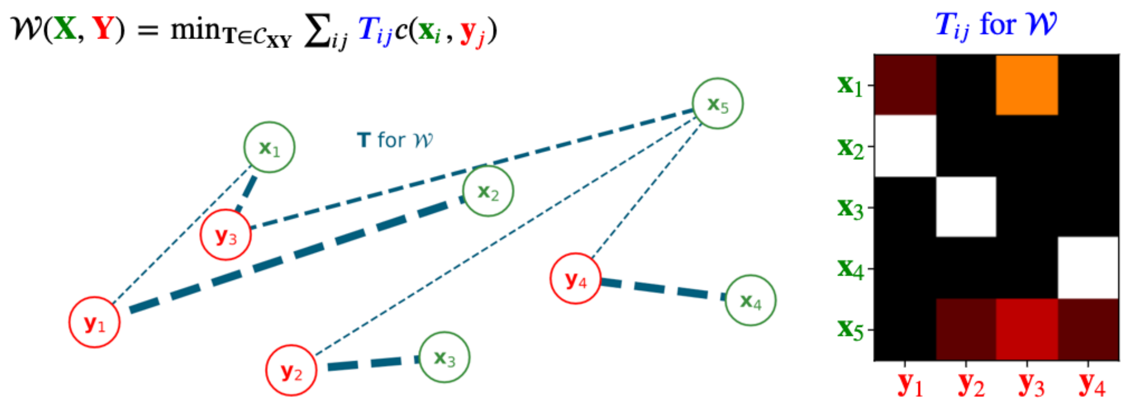

$\mathcal{W}(\mathbf{X}, \mathbf{Y}) = \min_{\mathbf{T} \in \mathcal{C}_{\mathbf{XY}}} \sum_{ij} T_{ij} c(\mathbf{x}_i, \mathbf{y}_j)$
T for $\mathcal{W}$
T_{ij} for $\mathcal{W}$
x1 x2 x3 x4 x5
y1 y2 y3 y4

Figure 3-2: We illustrate, for a given 2D point cloud, the optimal transport plan obtained from minimizing the Wasserstein costs; $c(\cdot, \cdot)$ denotes the Euclidean distance. A higher dotted-line thickness illustrates a greater mass transport.

Optimal Transport Geometry Optimal Transport [104] is a mathematical framework that defines distances or similarities between objects such as probability distributions, either discrete or continuous, as the cost of an optimal transport plan from one to the other.

Wasserstein distance for point clouds. Let a *point cloud* $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ *of size* n be a set of n points $\mathbf{x}_i \in \mathbb{R}^d$. Given point clouds $\mathbf{X}, \mathbf{Y}$ of respective sizes n, m, a ***transport plan*** (or ***coupling***) is a matrix $\mathbf{T}$ of size $n \times m$ with entries in $[0, 1]$, satisfying the two following *marginal constraints*: $\mathbf{T}\mathbf{1}_m = \frac{1}{n}\mathbf{1}_n$ and $\mathbf{T}^T\mathbf{1}_n = \frac{1}{m}\mathbf{1}_m$. Intuitively, the marginal constraints mean that $\mathbf{T}$ preserves the mass from $\mathbf{X}$ to $\mathbf{Y}$. We denote the set of such couplings as $\mathcal{C}_{\mathbf{XY}}$.

Given a cost function c on $\mathbb{R}^d \times \mathbb{R}^d$, its associated ***Wasserstein discrepancy*** is defined as

$$\mathcal{W}(\mathbf{X}, \mathbf{Y}) = \min_{\mathbf{T} \in \mathcal{C}_{\mathbf{XY}}} \sum_{ij} T_{ij} c(\mathbf{x}_i, \mathbf{y}_j). \tag{3.3}$$

3.3 Optimal Transport Neural Networks

Reformulating standard architectures. The final graph embedding $\mathbf{h} = \sum_{v \in V} \mathbf{h}_v$ obtained by aggregating node embeddings is usually fed to a multilayer perceptron (MLP) performing a matrix-multiplication whose i-th component is $(\mathbf{Rh})_i = \langle \mathbf{r}_i, \mathbf{h} \rangle$, where $\mathbf{r}_i$ is the i-th row of matrix $\mathbf{R}$. Replacing $\langle \cdot, \cdot \rangle$ by a distance/kernel $k(\cdot, \cdot)$ allows the processing of more general graph representations than just vectors in $\mathbb{R}^d$, such as point clouds or adjacency tensors.

From a single point to a point cloud. We propose to replace the aggregated graph embedding $\mathbf{h} = \sum_{v \in V} \mathbf{h}_v$ by the point cloud (of unaggregated node embeddings) $\mathbf{H} = \{\mathbf{h}_v\}_{v \in V}$, and the inner-products $\langle \mathbf{h}, \mathbf{r}_i \rangle$ by the below written ***Wasserstein discrepancy***:

$$\mathcal{W}(\mathbf{H}, \mathbf{Q}_i) := \min_{\mathbf{T} \in \mathcal{C}_{\mathbf{HQ}_i}} \sum_{vj} T_{vj} c(\mathbf{h}_v, \mathbf{q}_i^j), \tag{3.4}$$

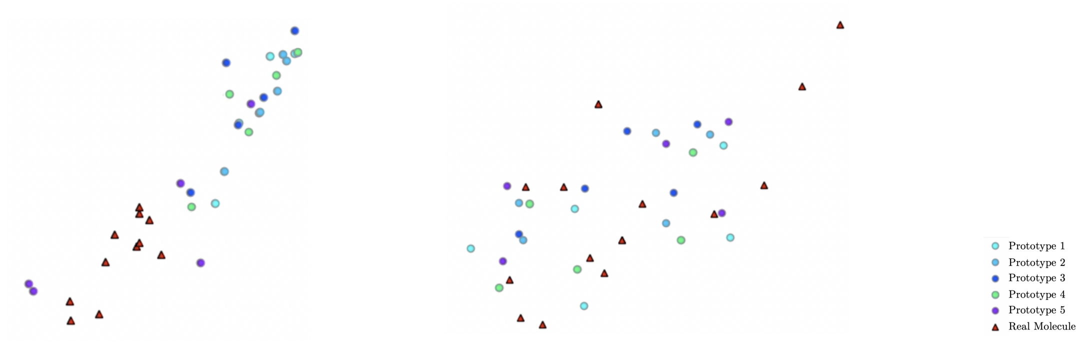
where $\mathbf{Q}_i = \{\mathbf{q}_i^j\}_{j \in \{1,\ldots,N\}}, \forall i \in \{1, \ldots, M\}$ represent M prototype point clouds each being represented as a set of N embeddings as free trainable parameters, and the cost is chosen as $c = \| \cdot - \cdot \|_2^2$ or $c = -\langle \cdot, \cdot \rangle$. Note that both options yield identical optimal transport plans.

Greater representational power. We formulate mathematically that this kernel has a strictly greater representational power than the kernel corresponding to standard inner-product on top of a sum aggregation, to distinguish between different point clouds.

Final architecture. Finally, the vector of all Wasserstein distances in eq. (3.4) becomes the input to a final MLP with a single scalar as output. This can then be used as the prediction for various downstream tasks, depicted in **??**.

Contrastive Regularization What would happen to $\mathcal{W}(\mathbf{H}, \mathbf{Q}_i)$ if all points $\mathbf{q}_i^j$ belonging to point cloud $\mathbf{Q}_i$ would collapse to the same point $\mathbf{q}_i$? All transport plans would yield the same cost, giving for $c = -\langle \cdot, \cdot \rangle$:

$$\mathcal{W}(\mathbf{H}, \mathbf{Q}_i) = -\sum_{vj} T_{vj} \langle \mathbf{h}_v, \mathbf{q}_i^j \rangle = -\langle \mathbf{h}, \mathbf{q}_i / |V| \rangle. \tag{3.5}$$



(a) No regularization (b) Using regularization (0.1)

Figure 3-3: 2D embeddings of prototypes and of a real molecule with and without contrastive regularization for same random seed runs on the ESOL dataset. Both prototypes and real molecule point clouds tend to cluster when no regularization is used (left). For instance, the real molecule point cloud (red triangle) is much more dispersed when regularization is applied (right) which is desirable in order to interact with as many embeddings of each prototype as possible.

In this scenario, our proposition would simply over-parametrize the standard Euclidean model.

Empirically, OT-enhanced GNNs with only the Wasserstein component sometimes perform similarly to the Euclidean baseline in both train and validation settings, in spite of its greater representational power. Further investigation revealed that the Wasserstein model would naturally displace the points in each of its prototype point clouds in such a way that the optimal transport plan $\mathbf{T}$ obtained by maximizing $\sum_{vj} T_{vj}\langle \mathbf{h}_v, \mathbf{q}_i^j \rangle$ was not *discriminative*, *i.e.* many other transports would yield a similar Wasserstein cost. Indeed, as shown in eq. (3.5), if each point cloud collapses to its mean, then the Wasserstein geometry collaspses to Euclidean geometry. In this scenario, any transport plan yields the same Wasserstein cost. However, partial or local collapses are also possible and would still result in non-discriminative transport plans, also being undesirable.

Intuitively, the existence of multiple optimal transport plans implies that the same prototype can be representative for distinct parts of the molecule. However, we desire that different prototypes disentangle different factors of variation such as different functional groups.

Contrastive regularization. To address this difficulty, we add a regularizer which encourages the model to displace its prototype point clouds such that the optimal transport plans would be discriminative against chosen *contrastive transport plans*. Namely, consider a point cloud $\mathbf{Y}$ of node embeddings and let $\mathbf{T}^i$ be an optimal transport plan obtained in the computation of $\mathcal{W}(\mathbf{Y}, \mathbf{Q}_i)$. For each $\mathbf{T}^i$, we then build a set $Neg(\mathbf{T}^i) \subset \mathcal{C}_{\mathbf{Y}\mathbf{Q}_i}$ of *noisy/contrastive* transports. If we denote by $\mathcal{W}_{\mathbf{T}}(\mathbf{X}, \mathbf{Y}) := \sum_{kl} T_{kl} c(\mathbf{x}_k, \mathbf{y}_l)$ the Wasserstein cost obtained for the particular transport $\mathbf{T}$, then our contrastive regularization consists in maximizing the term:

$$\sum_i \log \left(\frac{e^{-\mathcal{W}_{\mathbf{T}^i}(\mathbf{Y}, \mathbf{Q}_i)}}{e^{-\mathcal{W}_{\mathbf{T}^i}(\mathbf{Y}, \mathbf{Q}_i)} + \sum_{\mathbf{T} \in Neg(\mathbf{T}^i)} e^{-\mathcal{W}_{\mathbf{T}}(\mathbf{Y}, \mathbf{Q}_i)}} \right), \tag{3.6}$$

which can be interpreted as the log-likelihood that the correct transport $\mathbf{T}_i$ be (as it should) a better minimizer of $\mathcal{W}_{\mathbf{T}}(\mathbf{Y}, \mathbf{Q}_i)$ than its negative samples. This can be considered as an approximation of $\log(\Pr(\mathbf{T}_i \mid \mathbf{Y}, \mathbf{Q}_i))$, where the partition function is approximated by our selection of negative examples, as done e.g. by [94]. Its effect is shown in fig. 3-3.

The selection of negative examples should reflect the trade-off: *(i)* not be too large, for computational efficiency while *(ii)* containing sufficiently meaningful and challenging contrastive samples. Details about our choice of contrastive samples are in the experiments section. Note that replacing the set $Neg(\mathbf{T}^i)$ with a singleton $\{\mathbf{T}\}$ for a contrastive random variable $\mathbf{T}$ lets us rewrite (eq. (3.6)) as[1] $\sum_i \log \sigma(\mathcal{W}_{\mathbf{T}} - \mathcal{W}_{\mathbf{T}^i})$, reminiscent of noise contrastive estimation [52].

One may speculate that it was locally easier for the model to extract valuable information if it would behave like the Euclidean component, preventing it from exploring other roads of the optimization landscape. To better understand this situation, consider the scenario in which a subset of points in a prototype point cloud "collapses", i.e. points become close to each other (see fig. 3-3), thus sharing similar distances to all the node embeddings of real input graphs. The submatrix of the optimal transport matrix corresponding to these collapsed points can be equally replaced by any other

[1]where $\sigma(\cdot)$ is the sigmoid function.

submatrix with the same marginals (*i.e.* same two vectors obtained by summing rows or columns), meaning that the optimal transport matrix is not discriminative. In general, we want to avoid any two rows or columns in the Wasserstein cost matrix being proportional.

An optimization difficulty. An additional problem of point collapsing is that it is a non-escaping situation when using gradient-based learning methods. The reason is that gradients of all these collapsed points would become and remain identical, thus nothing will encourage them to “separate" in the future.

Total versus local collapse. Total collapse of all points in a point cloud to its mean is not the only possible collapse case. We note that the collapses are, in practice, mostly local, i.e. some clusters of the point cloud collapse, not the entire point cloud. We argue that this is still a weakness compared to fully uncollapsed point clouds due to the resulting non-discriminative transport plans and due to optimization difficulties discussed above.

Our experiments were conducted with ten negative samples for each correct transport plan. Five of them were obtained by initializing a matrix with uniform *i.i.d* entries from $[0, 10)$ and performing around five Sinkhorn iterations [24] in order to make the matrix satisfy the marginal constraints. The other five were obtained by randomly permuting the columns of the correct transport plan. The latter choice has the desirable effect of penalizing the points of a prototype point cloud $\mathbf{Q}_i$ to collapse onto the same point. Indeed, the rows of $\mathbf{T}^i \in \mathcal{C}_{\mathbf{HQ}_i}$ index points in $\mathbf{H}$, while its columns index points in $\mathbf{Q}_i$.

Complexity Backpropagating gradients through optimal transport (OT) has been the subject of recent research investigations: [43] explain how to unroll and differentiate through the Sinkhorn procedure solving OT, which was extended by [113] to Wasserstein barycenters. However, more recently, [141] proposed to simply invoke the envelop theorem [2] to support the idea of keeping the optimal transport plan fixed during the back-propagation of gradients through Wasserstein distances.

For the sake of simplicity and training stability, we resort to the latter procedure: keeping **T** *fixed during back-propagation.*

In our experiments, we used the Python Optimal Transport (POT) library [37]. We noticed empirically that the Earth Mover Distance (EMD) solver yielded faster and more accurate solutions than Sinkhorn for our datasets, because the graphs and point clouds were small enough (< 30 elements). As such, we our final models use EMD. Significant speed up could potentially be obtained by rewritting the POT library for it to solve OT in batches over GPUs. In our experiments, we ran all jobs on CPUs.

3.4 Experiments

We experiment on 4 benchmark molecular property prediction datasets [144] including both regression (ESOL, Lipophilicity) and classification (BACE, BBBP) tasks. These datasets cover different complex chemical properties (e.g. ESOL - water solubility, LIPO - octanol/water distribution coefficient, BACE - inhibition of human β-secretase, BBBP - blood-brain barrier penetration).

Each dataset is split randomly 5 times into 80%:10%:10% train, validation and test sets. For each of the 5 splits, we run each model 5 times to reduce the variance in particular data splits (resulting in each model being run 25 times). We search hyperparameters for each split of the data, and then take the average performance over all the splits. The hyperparameters are separately searched for each data split, so that the model performance is based on a completely unseen test set, and that there is no data leakage across data splits. The models are trained for 150 epochs with early stopping if the validation error has not improved in 50 epochs and a batch size of 16. We train the models using the Adam optimizer with a learning rate of 5e-4. For the prototype models, we use different learning rates for the GNN and the point clouds (5e-4 and 5e-3 respectively), because empirically we find that the gradients are much smaller for the point clouds. The molecular datasets used for experiments here are small in size (varying from 1-4k data points), so this is a fair method of comparison, and is indeed what is done in other works on molecular property prediction

Fingerprint + MLP applies a MLP over the input features which are hashed graph structures (called a molecular fingerprint). **GIN** is the Graph Isomorphism Network from [143], which is a variant of a GNN. The original GIN does not account for edge features, so we adapt their algorithm to our setting. Next, **GAT** is the Graph Attention Network from [134], which uses multi-head attention layers to propagate information. The original GAT model does not account for edge features, so we adapt their algorithm to our setting.

Chemprop - D-MPNN [144] is a graph network that exhibits state-of-the-art performance for molecular representation learning across multiple classification and regression datasets. Empirically we find that this baseline is indeed the best performing, and so is used as to obtain node embeddings in all our prototype models. Additionally, for comparison to our methods, we also add several graph pooling baselines. We apply the graph pooling methods, **SAG pooling** and **TopK pooling** [40], on top of the D-MPNN for fair comparison.

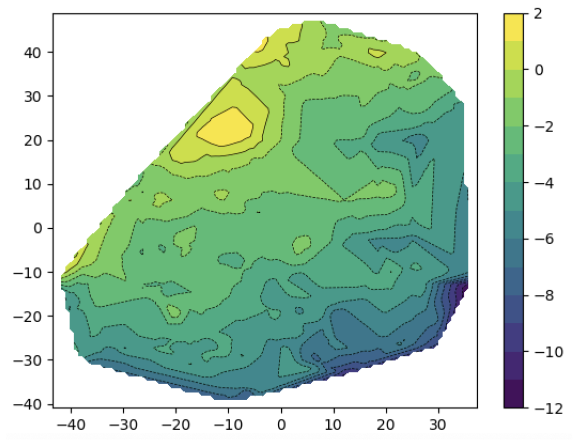
Different variants of our OT-GNN prototype model are described below:

ProtoW-L2/Dot is the model that treats point clouds as point sets, and computes the Wasserstein distances to each point cloud (using either L2 distance or (minus) dot product cost functions) as the molecular embedding. **ProtoS-L2** is a special case of **ProtoW-L2**, in which the point clouds have a *single* point and instead of using Wasserstein distances, we just compute simple Euclidean distances between the aggregated graph embedding and point clouds. Here, we omit using dot product distances since such a model is mathematically equivalent to the GNN model.

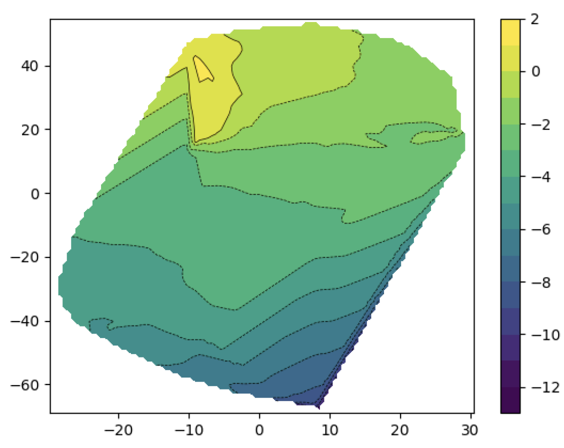
We use the the POT library [37] to compute Wasserstein distances using the Earth Movers Distance algorithm. We define the cost matrix by taking the pairwise L2 or negative dot product distances. We fix the transport plan, and only back-propagate through the cost matrix for computational efficiency. Additionally, to account for the variable size of each input graph, we multiply the OT distance between two point clouds by their respective sizes. We next delve into further discussions of our experimental results.

	Models	ESOL (RMSE)	Lipo (RMSE)	BACE (AUC)	BBBP (AUC)
Baselines	Fingerprint+MLP	.922 ± .017	.885 ± .017	.870 ± .007	.911 ± .005
	GIN	.665 ± .026	.658 ± .019	.861 ± .013	.900 ± .014
	GAT	.654 ± .028	.808 ± .047	.860 ± .011	.888 ± .015
	D-MPNN	.635 ± .027	.646 ± .041	.865 ± .013	.915 ± .010
	D-MPNN+SAG Pool	.674 ± .034	.720 ± .039	.855 ± .015	.901 ± .034
	D-MPNN+TopK Pool	.673 ± .087	.675 ± .080	.860 ± .033	.912 ± .032
Ours	ProtoS-L2	.611 ± .034	**.580 ± .016**	.865 ± .010	.918 ± .009
	ProtoW-Dot *(no reg.)*	.608 ± .029	.637 ± .018	.867 ± .014	.919 ± .009
	ProtoW-Dot	**.594 ± .031**	.629 ± .015	.871 ± .014	.919 ± .009
	ProtoW-L2 *(no reg.)*	.616 ± .028	.615 ± .025	<u>.870 ± .012</u>	<u>.920 ± .010</u>
	ProtoW-L2	<u>.605 ± .029</u>	<u>.604 ± .014</u>	**.873 ± .015**	**.920 ± .010**

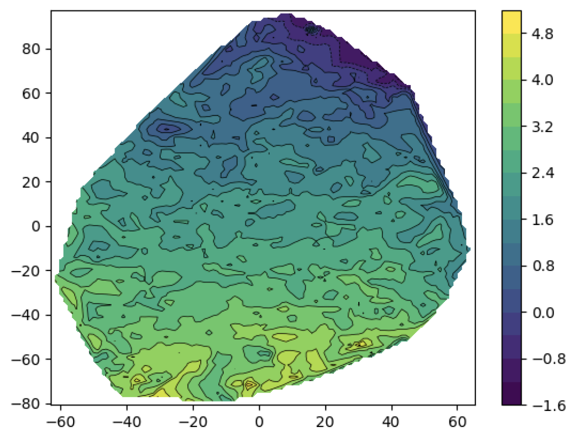
Table 3.1: Results on the property prediction datasets. **Best** model is in bold, <u>second</u> best is underlined. Lower RMSE and higher AUC are better. Wasserstein models are by default trained with contrastive regularization. GIN, GAT and D-MPNN use summation pooling which outperforms max and mean pooling. SAG and TopK graph pooling methods are also used with D-MPNN.



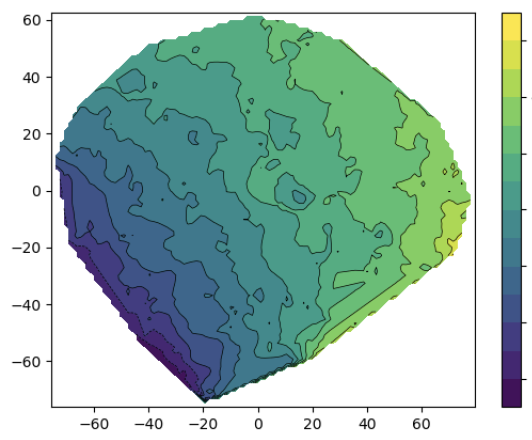
(a) ESOL D-MPNN



(b) ESOL ProtoW-L2



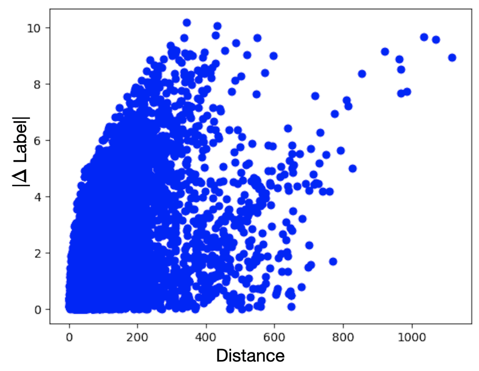
(c) LIPO D-MPNN



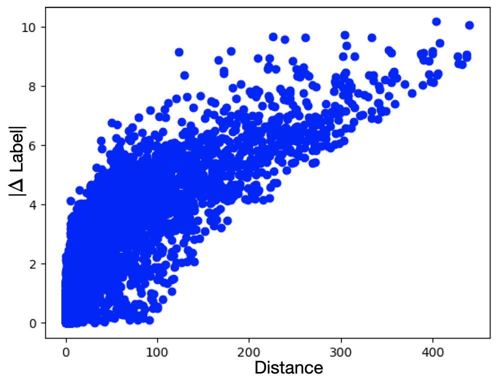
(d) LIPO ProtoW-L2

Figure 3-4: 2D heatmaps of T-SNE projections of molecular embeddings (before the last linear layer) w.r.t. their associated predicted labels on test molecules. Comparing (a) vs (b) and (c) vs (d), we can observe a smoother space of our model compared to the D-MPNN baseline.

Regression and Classification. Results are shown in table 3.1. Our prototype models outperform popular GNN/D-MPNN baselines on all 4 property prediction tasks. Moreover, the prototype models using Wasserstein distance (**ProtoW-L2/Dot**) achieve better performance on 3 out of 4 of the datasets compared to the prototype model that uses only Euclidean distances (**ProtoS-L2**). This indicates that Wasserstein distance confers greater discriminative power compared to traditional aggregation methods. We also find that the baseline pooling methods perform worse than the D-MPNN, and we attribute this to the fact that these models were originally created for large graphs networks without edge features, not for small molecular graphs.



(a) ESOL D-MPNN



(b) ESOL ProtoW-L2

Figure 3-5: Comparison of the correlation between graph embedding distances (X axis) and label distances (Y axis) on the ESOL dataset.

	Spearman ρ	Pearson r
D-MPNN	.424 ± .029	.393 ± .049
ProtoS-L2	.561 ± .087	.414 ± .141
ProtoW-Dot	.592 ± .150	.559 ± .216
ProtoW-L2	**.815 ± .026**	**.828 ± .020**

Figure 3-6: The Spearman and Pearson correlation coefficients on the ESOL dataset for the $\mathrm{D}-\mathrm{MPNN}$ and ProtoW-L2 model w.r.t. the pairwise difference in embedding vectors and labels.

Noise Contrastive Regularizer. Without any constraints, the Wasserstein prototype model will often collapse the set of points in a point cloud into a single point. As mentioned in earlier, we use a contrastive regularizer to force the model to meaningfully distribute point clouds in the embedding space. We show 2D embeddings in fig. 3-3, illustrating that without contrastive regularization, prototype point clouds are often displaced close to their mean, while regularization forces them to nicely scatter. Quantitative results in table 3.1 also highlight the benefit of this regularization.

Learned Embedding Space. We further examine the learned embedding space of the best baseline (i.e. D-MPNN) and our best Wasserstein model. We claim that our models learn smoother latent representations. We compute the pairwise difference in embedding vectors and the labels for each test data point on the ESOL dataset. Then, we compute two measures of rank correlation, Spearman correlation coefficient

(ρ) and Pearson correlation coefficient (r). This is reminiscent of evaluation tasks for the correlation of word embedding similarity with human labels [89].

Our ProtoW-L2 achieves better ρ and r scores compared to the D-MPNN model (fig. 3-6) indicating that our Wasserstein model constructs more meaningful embeddings with respect to the label distribution, which can be inferred also from fig. 3-5. Our ProtoW-L2 model, trained to optimize distances in the embedding space, produces more meaningful representations with respect to the label of interest.

Moreover, as qualitatively shown in fig. 3-4, our model provides more robust molecular embeddings compared to the baseline, in the following sense: we observe that a small perturbation of a molecular embedding corresponds to a small change in predicted property value – a desirable phenomenon that holds rarely for the baseline D-MPNN model. Our Proto-W-L2 model yields smoother heatmaps.

What types of molecules do prototypes capture ?

To better understand if the learned prototypes offer interpretability, we examined the ProtoW-Dot model trained with NC regularization (weight 0.1). For each of the 10 learned prototypes, we computed the set of molecules in the test set that are closer in terms of the corresponding Wasserstein distance to this prototype than to any other prototype. While we noticed that one prototype is closest to the majority of molecules, there are other prototypes that are more interpretable as shown in fig. 3-7.

3.5 Summary

We propose **OT-GNN**: one of the first parametric graph models that leverages optimal transport to learn graph representations. It learns abstract prototypes as free parametric point clouds that highlight different aspects of the graph. Empirically, we outperform popular baselines in different molecular property prediction tasks, while the learned representations also exhibit stronger correlation with the target labels.

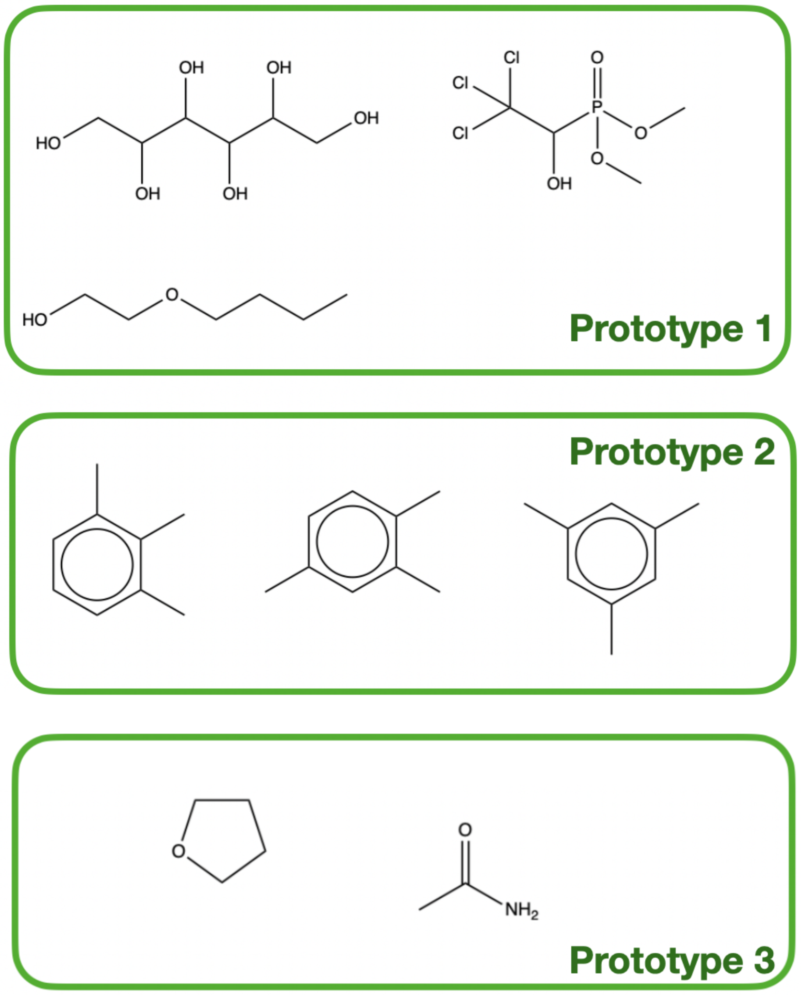


Figure 3-7: The closest molecules to some particular prototypes in terms of the corresponding Wasserstein distance. One can observe that some prototypes are closer to insoluble molecules containing rings (Prototype 2), while others prefer more soluble molecules (Prototype 1).

Chapter 4

Retrosynthesis

Deep learning techniques that are widely employed for traditional tasks like NLP and vision do not directly transfer to the more complex structure of chemical data. Here, we explore some more chemically-motivated methods for retrosynthesis prediction. Specifically, we propose a new model for making generalizable and diverse retrosynthetic reaction predictions. Given a target compound, the task is to predict the likely chemical reactants to produce the target. This generative task can be framed as a sequence-to-sequence problem by using the SMILES representations of the molecules. Building on top of the popular Transformer architecture, we propose two novel pre-training methods that construct relevant auxiliary tasks (plausible reactions) for our problem. Furthermore, we incorporate a discrete latent variable model into the architecture to encourage the model to produce a diverse set of alternative predictions. On the 50k subset of reaction examples from the United States patent literature (USPTO-50k) benchmark dataset, our model greatly improves performance over the baseline, while also generating predictions that are more diverse.

4.1 Introduction

Retrosynthesis is a task crucial for material and drug manufacturing [22, 23] and aims to predict which reactants are needed to generate a given target molecule as the main product. For instance, fig. 4-1 demonstrates that the input molecule "[N-

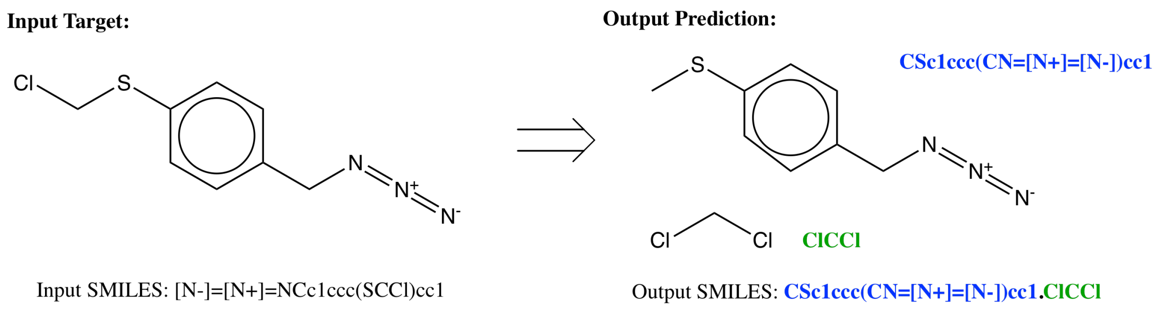

Input Target:
Output Prediction:
CSc1ccc(CN=[N+]=[N-])cc1
ClCCl
Input SMILES: [N-]=[N+]=NCc1ccc(SCCl)cc1
Output SMILES: CSc1ccc(CN=[N+]=[N-])cc1.ClCCl

Figure 4-1: An example prediction task: on the left is the input target SMILES, and on the right are the output reactants SMILES. The input is a single molecule, while the output is a set of molecules separated by a period (".").

]=[N+]=NCc1ccc(SCCl)cc1", expressed here as a SMILES string [137], can be generated using reactants "CSc1ccc(CN=[N+]=[N-])cc1" and "ClCCl". For decades, this task has been solved using template-based approaches [42, 111]. Templates encode transformation rules as regular expressions operating on SMILES strings and are typically extracted directly from the available training reactions. The primary limitation of such templates is coverage, i.e., it is possible that none of the templates applies to a test molecule. In order to better generalize to newer or broader chemical spaces, recently developed template-free approaches cast the problem as a sequence-to-sequence prediction task. These approaches were first explored by [84] using LSTM models; the current state-of-the-art performance on this task uses Transformer models [82, 70].

Out-of-the-box Transformers nevertheless do not effectively generalize to rare reactions. For instance, model accuracy drops by 25% on reactions with 10 or fewer representative instances in the training set. [2] Another key issue is diversity. Manufacturing processes involve a number of additional criteria — such as green chemistry (having low detrimental effects on the environment). It is therefore helpful to generate a diverse collection of alternative ways of synthesizing the given target molecule. However, predicted reactions are unlikely to encompass multiple reaction classes (see fig. 4-2) without additional guidance. This is because the training data only provides

[2]To compute a subset of the data with only rare reactions, we extracted all the templates from the entire USPTO-50k dataset, and selected the templates that occured at most 10 times. The reactions in the test set that have these templates constitute the rare reaction subset, which is around 400 examples.

a single reactant set for each input target, even if this is not the only valid reaction to synthesize the target.

We extend molecular Transformers to address both of these challenges. First, we propose a novel pre-training approach to drive molecular representations to better retain alternative reaction possibilities. Our approach is reminiscent of successful pre-training schemes in natural language processing (NLP) applications [28]. However, rather than using conventional token masking methods, we adopt chemically-relevant auxiliary tasks. Each training instance presents a single way to decompose a target molecule into its reactants. Here, we add alternative proxy decompositions for each target molecule by either 1) randomly removing bond types that can possibly break during reactions, or 2) transforming the target based on templates. While neither of these two auxiliary tasks are guaranteed to construct valid chemical reactions, they are closely related to the task of interest. Indeed, representations trained in this manner provide useful initializations for the actual retrosynthesis problem.

To improve the diversity of predicted reactions, we incorporate latent variables into the generation process. Specifically, we merge the Transformer architecture with a discrete mixture over reactions. The role of the latent variable is to encode distinct modes that can be related to underlying reaction classes. Even though the training data only presents one reaction for each target molecule, our model learns to associate

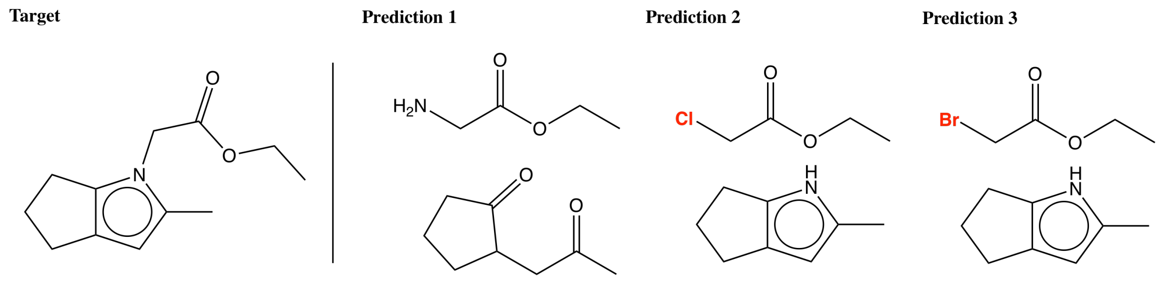


Figure 4-2: For the input target compound shown on the left, three possible reactant predictions are shown on the right. Prediction 1 suggestions a heterocycle formation reaction, while Predictions 2 and 3 both suggest substitution reactions. The only difference between the latter two is the halide functional group (Cl vs Br) highlighted in red. They share similar chemical properties and thus provide no additional insights for chemists.

each reaction with a latent class, and in the process covers multiple reaction classes across the training set. At test time, a diverse collection of reactions is then obtained by collecting together predictions resulting from conditioning on each latent class. Analogous mixture models have shown promise in generating diverse predictions in natural language translation tasks [56, 119]. We demonstrate similar gains in the chemical context.

We evaluate our model on the benchmark USPTO-50k dataset, and compare it against state-of-the-art template-free baselines using the Transformer model. We focus our evaluation on top-10 accuracy, because there are many equally valuable reaction transformations for each input target, though only one is presented in the data. Compared to the baseline, we achieve better performance overall, with over 13% increase in top-10 accuracy for our best model. When we create a split of the data based on different reaction templates (a task that any template-based model would fail on), we similarly observe a performance increase for our model. Additionally, we demonstrate that our model outputs exhibit significant diversity through both quantitative and human evaluations.

Template-based Models Traditional methods for retrosynthetic reaction prediction use template-based models. Templates, or rules, denote the exact atom and bond changes for a chemical reaction. [20] applies these templates for a given target compound based on similar reactions in the dataset. Going one step further, [117] learns the associations between molecules and templates through a neural network. [7] uses a hierarchical network to first predict the reaction group and then the correct template for that group. However, to have the flexibility to generalize beyond extracted rules, we explore template-free generative models.

Molecule Generation There are two different approaches to generative tasks for molecules, demonstrated through graph and SMILES representations. The graph-generation problem has been explored in [81] as a node-by-node generation algorithm, but this model does not guarantee the validity of the output chemical graph. [62, 68] improves upon this method using a junction-tree encoder-decoder that forces the outputs to be constrained in the valid chemical space; however, these models require complex, structured decoders. We focus on the generative task of SMILES string representations of the molecules, which has been explored in [78] and [47].

Pre-training Pre-training methods have been shown to vastly improve the performance of Transformer models in NLP tasks without additional data supervision. [28] use a masked language modeling objective to help their model learn effective representations for downstream tasks. Similar pre-training methods on molecules have been explored by [58], where they mask out atoms in molecular graphs. Meanwhile, our work does not use a masked objective, but instead creates pre-training tasks that are relevant to the retrosynthesis prediction problem.

4.2 Models for Retrosynthesis Prediction

Given an input target molecule, the task of retrosynthetic reaction prediction is to output likely reactants that can form the target product. Formally, we express a molecule as a text string via its SMILES representation, and cast our task into a sequence-to-sequence (seq2seq) prediction problem (example shown in fig. 4-1). For this task, the input target is always a single molecule, while the output predictions are usually a set of more than one molecule concatenated by separators ".".

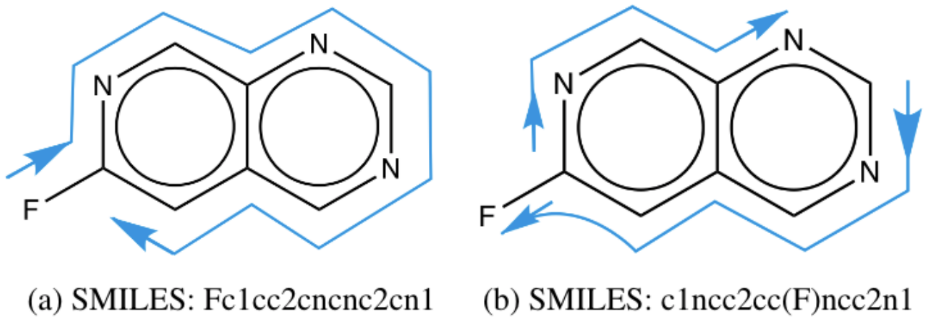


Figure 4-3: A single molecule has many different SMILES representations. On the left (a) is the canonical SMILES string, and on the right (b) is another SMILES string representing in the same molecule.

To provide more intuition for this generative task, we describe some properties of

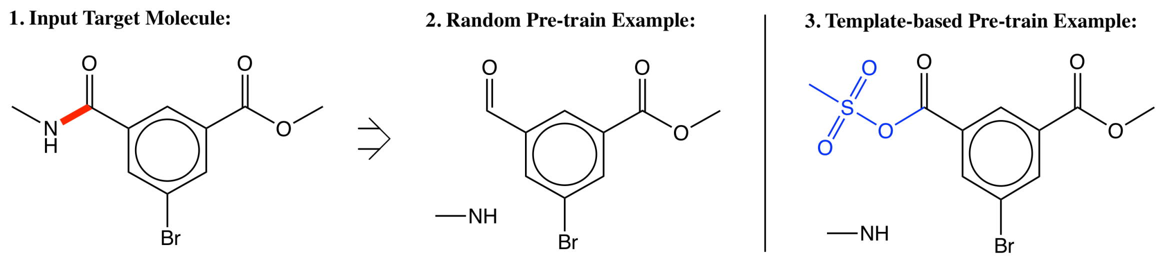

1. Input Target Molecule:
2. Random Pre-train Example:
3. Template-based Pre-train Example:
O
N
H
O
O
Br
NH
S

Figure 4-4: Input target molecule (1) with two automatically generated pre-training targets formed by breaking the bond highlighted in red. Examples (2) and (3) are generated from the random and template-based methods respectively. The only difference is that the template-based pre-training example (3) adds an additional function group to the molecule (blue).

SMILES strings. Each SMILES string is 1-D encoding of a 2-D molecular graph. If the predicted SMILES does not adhere to the SMILES grammar, then a valid molecular graph cannot be reconstructed. Moreover, each molecule has many equivalent SMILES representations, as a single instance of its SMILES is just a graph traversal starting at some arbitrary node. Therefore, two very different SMILES string can encode the same molecule, and the model needs to be robust to the given input. One method, proposed by [116], augments the input data with different SMILES strings of the same input target molecule.

For our model architecture, we apply a Transformer model for the seq2seq task, which has an encoder-decoder structure [132, 116]. The encoder maps an input sequence of tokens (from the SMILES string) to a sequence of continuous representations, which are then fed to the decoder to generate an output sequence of tokens one element at a time, autoregressively. Once the model is trained, a beam search procedure is used at inference time to find likely output sequences.

The main building block of the Transformer architecture lies in its global self-attention layers, which are well-suited for predictions of the latent graph structure of SMILES strings. For example, two tokens that are far apart in the SMILES string could be close together topologically in the corresponding molecular graph. The global connectivity of the Transformer model allows it to better leverage this information. Additionally, since SMILES follow a rigid grammar requiring long range-dependencies, these dependencies can be more easily learned through global attention layers.

Despite the flexible architecture of Transformer models, we recognize that there are ways to improve model generalization. Additionally, there is no inductive bias for proposing diverse outputs. We propose two techniques to enhance the base molecular Transformer model, which we describe now.

4.3 Pretraining

In the data, each input target molecule is associated with a single reaction transformation — though there are many equally good chemical reactions. Therefore, for each input target, we construct several new prediction examples that are chemically meaningful, and pre-train the model on these auxiliary examples. We do so without requiring additionally data, or data supervision. The two variants of our method are described in detail below.

Random pre-training For each input target molecule, we generate new examples by selecting a random bond to break. The types of bonds that we consider are acyclic single bonds, because these are the bonds most commonly broken in chemical reactions. As we break an acyclic bond, the input molecule is necessarily broken up into two output molecules, each being a subgraph of the input molecule. Although the examples generated by this method do not cover the entire space of chemical reactions (for instance some reactions do not break any bonds at all), these examples are easy to generate and cover a diverse range of transformations.

Template-based pre-training Instead of randomly breaking bonds, we can also use the templates extracted from the training data to create reaction examples. Each template matches a specific pattern in the input molecule, and transforms that pattern according to the template specifications. When the matched pattern is a single acyclic bond, this method will generate similar outputs as the random pre-training method, except that templates usually add additional pieces (functional groups) to the output example.

As shown in **??**, both examples are derived from the same bond broken in the input

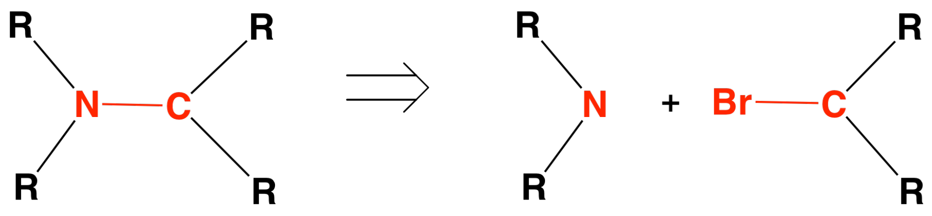


Figure 4-5: An example of a template, where the exact bond changes are described in red. The "C-N" bond (left) is broken and a "Br" atom is attached to the broken "C" atom (right).

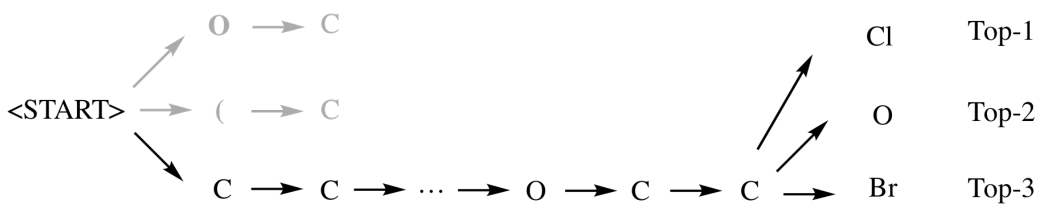


Figure 4-6: An example beam search; often times, the outputs of a beam search will be very similar to each other, here only differing in a single atom for the top 3 predictions. For SMILES strings in particular, this typically leads to predictions that belong to the same group (ie. only swapping a halide atom).

target molecule, but for the template-based example, an additional functional group was added, matching a more realistic reaction context. On average, for a random input molecule, there are 10 different possible examples that can be extracted from the random pre-training method, while there are over 200 different possible examples that can be extracted using the template-based pre-training method. However, many of these 200 examples represent similar chemical transformations, only differing in the type of functional group added.

More broadly speaking, we can say that the template pre-training method generates more chemically valid reactions compared to the random pre-training method. However, the advantage of the random pre-training method is that it can break bonds that are not represented within the templates, thereby perhaps conferring a higher ability to generalize. As routine, the model is pre-trained on these automatically constructed auxiliary tasks, and then used as initialization to be fine-tuned on the actual retrosynthesis data.

4.4 Mixture Model

Next, we tackle the problem of generating diverse predictions. As mentioned earlier, the retrosynthesis problem is a one-to-many mapping since a target molecule can be formed from different types of reactions. We would like the model to produce a diverse set of predictions so that chemists can choose the most feasible and economical one in practice. However, hypotheses generated by a vanilla seq2seq model with beam search typically exemplifies low diversity with only minor differences in the suffix [135]. To address this, we use a mixture seq2seq model that has shown sucess in generating diverse machine translations to generate diverse retrosynthesis reaction predictions [56, 119].

Specifically, given a target SMILES string x and reactants SMILES string y, a mixture model introduces a multinomial latent variable $z \in \{1, \cdots, K\}$ to capture different reaction types, and decomposes the marginal likelihood as:

$$p(y|x;\theta) = \sum_{z=1}^{K} p(y, z|x;\theta) = \sum_{z=1}^{K} p(z|x;\theta)p(y|z, x;\theta) \tag{4.1}$$

Here, the prior $p(z|x;\theta)$ and likelihood $p(y|z, x;\theta)$ parameterized by θ are functions to be learned.

We use a uniform prior $p(z|x;\theta) = 1/K$, which is easy to implement and works well in practice [119]. For $p(y|z, x;\theta)$, we share the encoder-decoder network among mixture components, and feed the embedding of z as an input to the decoder so that y is conditioned on it. The increase in the parameters of our model is negligible over the baseline model.

We train the mixture model with the online hard-EM algorithm. Taking a mini-batch of training examples $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, we enumerate all K values of z and compute their loss, $-\log p(y^{(i)}|z, x^{(i)};\theta)$. Then, for each $(x^{(i)}, y^{(i)})$, we select the value of z that yields the minimum loss: $z^{(i)} = \arg\min_z -\log p(y^{(i)}|z, x^{(i)};\theta)$, and back-propagate through it, so only one component receives gradients per example. An important detail for successfully training a mixture model is that dropout is turned off in the forward passes for latent variable selection, and turned back on at backpropagation time for gradient regularization. Otherwise even a small amount of dropout noise will corrupt the optimal value of z, making the selection random and the different latent components will fail to diversify [119].

The hard selection of the latent variable forces different components to specialize on different subsets of the data. As we shall later see in the experimental results, our mixture model can learn to represent different reaction types in the training data and show improved diversity over the baseline.

4.5 Experimental Setup

Data The benchmark dataset we use is a subset of the open source patent database of chemical reactions [87]. Specifically, we use the curated 50k subset (USPTO-50k) from [84], including the same data splits (80:10:10 for train, validation and test). Each example reaction in this dataset is labeled with one of ten reaction classes, which describes its transformation type, but we do not use this information in our experiments, similar to [70]. Since we are only interested in the retrosynthesis prediction problem, the examples are processed to remove any reagent molecules (molecules that do not contribute atoms to the reaction).

In addition, we create a separate split of the USPTO-50k data, in which the train and test sets are split by reaction templates. Specifically, we split the data so that no example in the test set can be solved correctly with any templates extracted from training data. We use the template extraction code from [21], which to the best of our knowledge, is the only publicly available template extraction code.

Metrics: Accuracy The evaluation of retrosynthesis is challenging, because each input target has many valid syntheses, but only one is given in the data. When the model output does not exactly match the single solution in the data, the model is not necessarily wrong, but simply giving a plausible alternative. Therefore, we focus on the top-10 accuracy for our evaluation, but present all results from our experiments. We compute the accuracies by matching the canonical SMILES strings of molecule sets. For the mixture model, we output the top 10 predictions for each latent class,

Table 4.1: Accuracy metrics on the USPTO-50K dataset without reaction labels. The variations we test are data augmentation, pre-training and number of latent classes. The highest accuracy model for each different latent model is bolded, and the highest accuracy model overall is parenthesized.

	Model	Top-1	Top-2	Top-3	Top-4	Top-5	Top-10
	Template [21]	37.3	-	54.7	-	63.3	74.1
	SCROP [149]	43.7	-	60.0	-	65.2	68.7
N Latent = 1	Base	42.0	52.8	57.0	59.9	61.9	65.7
	Aug	44.0	55.3	60.1	63.0	65.1	69.0
	Pre-train (R)	43.3	54.6	59.7	62.4	64.6	68.7
	Pre-train (T)	43.5	55.6	61.5	64.8	67.4	71.3
	Pre-train (R) + Aug	**(44.8)**	**57.1**	62.6	**65.7**	**67.7**	71.1
	Pre-train (T) + Aug	44.5	56.9	**62.7**	65.6	**67.7**	**71.7**
N Latent = 2	Base	42.1	54.4	60.0	63.1	64.9	70.3
	Aug	43.1	56.6	62.2	65.9	68.1	73.3
	Pre-train (R)	42.5	56.1	61.8	65.4	67.7	72.9
	Pre-train (T)	42.7	56.0	62.3	66.0	68.0	74.2
	Pre-train (R) + Aug	**43.6**	**(57.7)**	63.7	67.3	69.6	75.2
	Pre-train (T) + Aug	42.6	57.0	**64.0**	**68.6**	**71.3**	**76.6**
N Latent = 5	Base	39.1	55.4	62.5	66.5	69.1	74.5
	Aug	39.7	56.9	64.1	68.1	71.1	77.0
	Pre-train (R)	39.7	55.8	63.5	67.6	70.1	76.0
	Pre-train (T)	39.9	54.6	62.9	68.2	71.2	77.7
	Pre-train (R) + Aug	40.2	56.7	64.9	69.6	72.4	78.4
	Pre-train (T) + Aug	**40.5**	**56.8**	**(65.1)**	**(70.1)**	**(72.8)**	**(79.4)**

and then combine those results based on likelihoods to get top 10 predictions overall.

Metrics: Diversity To measure diversity, we provide both quantitative and human evaluations. For the former, we train a model to predict the reaction class given the input target molecule and the predicted output. We use a typical message-passing graph convolution network [66] to embed both the input and output molecules (using weight-sharing) and compute the reaction embedding as a difference of the two embeddings. This predictor is trained on the 10 reaction class labels in the USPTO-50k dataset, and achieves 99% accuracy on the test set, so we can be fairly confident in its ability to predict the reaction class in-domain.

Baselines Our main baseline is the SMILES transformer (**Base**), adapted from [116]. We run the same model as other recent works for this task [82, 70], and we build on top of the Transformer implementation from OpenNMT [73]. We run ablation experiments for pre-training and different mixture models to show the impact of each approach. Random pre-training is referred to as **Pre-train (R)**, while template-based pre-training is referred to as **Pre-train (T)**. For each example, we construct up to 10 new auxiliary examples, and pre-train the model on these examples. Additionally, following [116], we also augment the training data with variations of the input SMILES string, referred to as **Aug**. That is, for each training example, we add an extra example using a different input SMILES string, which is trained to predict the same output reactants. This helps the model learn representations robust to the permutation of the input SMILES string. In addition to our experiments, we include a template-based approach from [20], and a template-free approach from [149] that adds a syntax predictor on top of the transformer model.

Training Details We run all models with the same base configurations, to ensure a fair comparison. The reactions are tokenized in the same manner as in [116], with each token being a meaningful subunit of the molecule (ie. “Br" would be a single token denoting the Bromine atom). We train each model for 500k steps with a batch size of 4096 tokens, evaluating the performance on the validation set every 10k steps, with the final model being chosen based on the best performance on the validation set. We use the Adam optimizer with an initial learning rate of 2, 8k warmup steps and noam decay. The transformer uses 8 heads with embedding size of 256, and uses 2048 as the embedding size of the feedforward layer. We run all models with a dropout of 0.1. For the pre-training models, we pre-train the model for 100k steps on the pre-training task, and then switch to the actual dataset with the loaded pre-train model.

The accuracy results of our model is shown in table 4.1. We observe that both pre-training tasks improve over the baseline, and more so when combined with data augmentation. This shows that our pre-training tasks help the model learn the chemical reaction representational space, and are useful for the retrosynthesis prediction

Table 4.2: Prediction accuracies when tested on our template split of the USPTO dataset, for which any template-based model would get 0% accuracy on the test set. We see that our template-free methods can still generalize to this test set.

	Top-1	Top-2	Top-3	Top-4	Top-5	Top-10
Base Model	4.3	8.7	11.9	14.7	16.6	20.6
Best Mixture Model	5.5	9.2	12.6	15.4	17.6	26.6

Table 4.3: The left table shows the comparison of number of unique reactions predicted by the base model vs. the mixture model (holding other factors constant). The right table shows human evaluation metrics, in which a human was asked to rate whether the outputs of the base model or the mixture model was more diverse, or neither.

	Unique Reactions
Base Model	2.66
Mixture Model	**3.32**

	Human Diversity
Base Model	21
Mixture Model	**43**
Neither	36

problem. However, interestingly, there seem to be marginal differences between the two pre-training methods. We attribute this to the fact that both pre-training methods usually generate very similar sets of examples. Previously shown in **??**, one of the main differences of template-based pre-training is just that it adds additional functional groups. But since these generated examples are not always chemically valid, having this extra information may not prove to be very valuable.

We do note, however, that constructing additional decompositions of the input targets does actually matter for the pre-training task. We had also experimented with pre-training methods that only used variations of the input SMILES strings as pre-training output targets (because each molecule has many different SMILES representations). However, these methods did not result in the same performance gains, because these pre-training targets do not contain much useful information for the actual task.

Our original motivation for using a mixture model was to improve diversity, but we observe that it also leads to an increase in performance. We try $N = \{1, 2, 5\}$ for the number of discrete latent classes, and we see that more latent classes generally

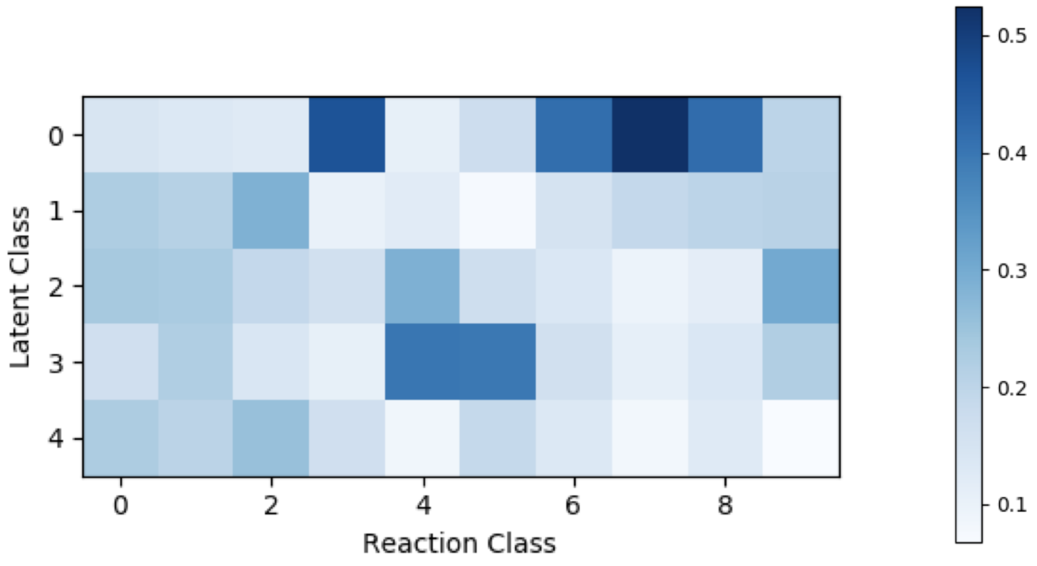


Figure 4-7: Heat map plotting the frequency that each latent class generates a specific reaction class. Here, we use a mixture model with 5 latent classes, and there are 10 reaction classes of interest for this data set.

leads to higher accuracies. The top-1 accuracy does decrease slightly as the number of latent classes increases, but we observe much higher accuracies at top-10 (increase of 7-8%). Importantly, we note that our method of combining outputs from different latent classes is not perfect, as the likelihoods from different latent classes are not totally comparable. That is likely the cause of the decrease in top-1 accuracy, but top-10 accuracies are significantly more meaningful for our problem.

Next, we show our results on a different split of the data, which is more challenging to generalize. Using the dataset split on templates, we explore the performance of our best mixture model with pre-training compared to the baseline with no pre-training. As mentioned earlier, template-free models confer advantages over template-based models, as template-based models lack the ability to generalize outside of extracted rules. For this dataset, any template-based model would necessarily achieve 0% accuracy based on construction. table 4.2 compares the performance of the different models and we see that, although the task is challenging, we can attain substantial accuracy of 26.6% at top-10 compared to 20.6% of the baseline. Even on this difficult task, we show that our model offers generalizability on the test set.

We now look at evaluations of diversity for our model. Using the reaction class model, we predict the reaction class for every output of our models. Then, we compute the average number of unique reaction classes, holding all other factors constant besides varying the number of latent classes (results shown in table 4.3). The number of unique reaction classes is 3.32 for the mixture model compared to the 2.66 for the base model, suggesting that the mixture model predicts a more diverse cast of outputs.

The diversity of the predictions can also be examined from an interpretability standpoint for the latent classes of the mixture model. Using the reaction class model, we take the 10 top predictions from each latent class, and count the number of occurrences for each reaction class. Normalizing across reaction classes, we can see from fig. 4-7 that each latent class learns to predict a different distribution of reaction classes.

We also supplement our diversity results with human evaluation. For this, we asked a senior (5+ years) PhD chemistry student to compare the outputs of the base model versus the mixture model. To make the problem tractable for a human chemist, we randomly select 100 different reactions from the test set and present the top 5 predicted outputs from both the base and mixture model, where the the task is to determine diversity based on number of different types of reactions.

The human is asked to judge diverse by comparing the number of reactions that differ in reaction type or location of reaction on molecule. Reactions that used slightly different precursors were considered identical, and therefore do not contribute to diversity (for example, protection reactions with different protection groups are considered as one type). Lastly, the evaluation was done with the correctness of the overall reaction in mind. For this task, the human chemist chose the mixture model more than twice as often as the base model (43 times vs 21), see table 4.3. Although not perfect, these results exemplify that our model does generate more diverse outputs than the baseline.

Lastly, to stress the importance of our chemically-relevant pre-training tasks, we also explore the performance of typical masked language model objectives. For instance, [58] masks out atoms in graphs, and try to predict the masked atoms using their contexts. Here, we try a similar experiment, masking out tokens in SMILES strings. Specifically, we run the following pre-training experiment: we mask out random tokens (with probability $p = 0.2$), and try to decode the target molecule.

The results of these pre-training experiments can be seen in table 4.4. The top-10 accuracy of this model is 61.6 which is worse than even the base model without pre-training, which achieves a top-10 accuracy of 65.7. This suggests that training on poor objectives can lead to negative transfer, in which the model is poorly initialized and has a more difficult time optimizing on the actual data.

Table 4.4: Prediction accuracies using a masked pre-training objective; numbers for other models taken from table 4.1 for easy comparison. The model trained with this masked pre-training object performs worse than the baseline model, suggesting the importance of having chemically-relevant pre-training tasks.

	Top-1	Top-5	Top-10
Base	42.0	57.0	65.7
Pre-train (Mask)	38.7	57.9	61.6
Pre-train (R)	43.3	60.1	69.0
Pre-train (T)	43.5	61.5	71.3

4.6 summary

We explored the problem of making one-step retrosynthesis reaction predictions, dealing with the issues of generalizability and making diverse predictions. Through pre-training and use of mixture models, we show that our model beats state-of-the-art methods in terms of accuracy and generates more diverse predictions. Even on a challenging task, for which any template-based models would fail, our model still is able to generalize to the test set.

Chapter 5

Molecular Generation

Searching for novel molecular compounds with desired properties is an important problem in drug discovery. Many existing frameworks generate molecules one atom at a time. We instead propose a flexible editing paradigm that generates molecules using learned molecular fragments—meaningful substructures of molecules. To do so, we train a variational autoencoder (VAE) to encode molecular fragments in a coherent latent space, which we then utilize as a vocabulary for editing molecules to explore the complex chemical property space. Equipped with the learned fragment vocabulary, we propose FaST, which learns a reinforcement learning (RL) policy to iteratively translate model-discovered molecules into increasingly novel molecules while satisfying desired properties. Empirical evaluation shows that FaST significantly improves over state-of-the-art methods on benchmark single/multi-objective molecular optimization tasks.

5.1 Introduction

Molecular optimization is a challenging task that is pivotal to drug discovery applications. Part of the challenge stems from the difficulty of exploration in the molecular space: not only are there physical constraints on molecules (molecular strings/graphs have to obey specific chemical principles), molecular property landscapes are also very complex and difficult to characterize: small changes in the molecular space can lead to large deviations in the property space.

Recent fragment-based molecular generative models have shown significant empirical advantages [63, 105, 140] over atom-by-atom generative models in molecular optimization. However, they generally operate over a fixed set of fragments which limit the generative capabilities of these models. Shifting away from previous frameworks, we learn a distribution of molecular fragments using vector-quantized variational autoencoders (VQ-VAE) [130]. Our method builds molecular graphs through the addition and deletion of molecular fragments from the learned distributional fragment vocabulary, enabling the generative model to span a much larger chemical space than models with a fixed fragment vocabulary. Considering atomic edits as primitive actions, the idea of using fragments can be thought of as *options* [126, 125] as a temporal abstraction to simplify the search problem.

We further introduce a novel *sequential translation* scheme designed for fragment-based molecular optimization. We start the molecular search by translating from known active molecules and store the discovered molecules as new potential initialization states for subsequent searches; we incorporate a delete action in our model, enabling our method to backtrack to good molecular states. Previous works optimize molecules either by generating from scratch or a single translation from known molecules, which is inefficient in finding high-quality molecules and often discovering molecules lacking novelty/diversity. Our proposed framework addresses these deficiencies since our method is (1) very efficient in finding molecules that satisfy property constraints as the model stay close to the high-property-score chemical manifold; and (2) able to produce highly novel molecules with our flexible learned fragment vocabulary and a sequence of fragment-based editing.

Combining the advantage of a distributional fragment vocabulary and the sequential translation scheme, we propose FaST, which is realized by an RL policy that proposes fragment addition/deletion to a given molecule. Our proposed method can generate molecules under various objectives such as property constraints, novelty constraints, and diversity constraints. The main contribution of this paper includes:

1. We demonstrate a way to learn distributional molecular fragment vocabulary

through a VQ-VAE and the effectiveness of the learned vocabulary in molecular optimization.

2. We propose a novel molecular search scheme of sequential translation, which gradually improves the quality and novelty of generation through backtracking and a stored frontier.

3. We implement a RL policy combining the fragment vocabulary and the sequential translation scheme that significantly outperforms state-of-the-art methods in benchmark single/multi-objective molecular optimization tasks.

Molecular generation and optimization. Early works on molecular optimization build on generative models on both SMILES/SELFIES string [46, 118, 69, 77, 99, 122], and molecular graphs [123, 85, 90, 26, 109, 92] and generate molecules character-by-character or node-by-node. [61] generates graphs as junction trees by considering the vocabulary as the set of atoms or predefined rings from the data; [64] use the same atom+ring vocabulary to generate molecules by augmenting extracted rationales of molecules.

Generating molecules with a fixed molecular fragment vocabulary is a well-established idea in traditional drug design [34], and has been recently explored through deep learning models [105, 140, 76, 38, 39, 79], outperforming previous atom-level models. Recent synthesizability-aware models can also generate single-step reaction [10] and molecule synthesis graphs [11] based on a fixed reactant pool. However, the fixed fragment vocabularies used by these models, which are typically small and predefined a priori, limit the chemical space spanned by the models. In our work, we utilize a learned molecular fragment vocabulary, which is obtained by training a VQ-VAE on a large set of fragments extracted from ChEMBL [41]. By sampling fragments from the learned distribution, our model can span a much larger chemical space than methods using a fixed vocabulary (visualized in fig. 5-3, fig. 5-4).

Sequential generation of molecules. [51, 101, 146, 150] frame the molecular optimization problem as a reinforcement learning problem, but they generate on the atom/character level and from scratch each time, reducing the efficiency of the search algorithm. [67] uses a graph-to-graph translation model for property optimization. However, it requires a large number of translation pairs to train, which often involves expert human knowledge and is expensive to obtain. Others have used genetic/evolutionary algorithms to tackle this problem [97, 98], which performs random mutations on chemical strings. Although these methods use learned discriminators to prune sub-optimal molecules, the random mutation process can become inefficient in searching for good molecules under complex property constraints. [140, 38] applies Markov Chain Monte Carlo (MCMC) sampling through editing molecules, while [76] uses Bayesian optimization on the latent space. While there are extensive studies in exploration strategies for RL [102, 14, 33], diversity/novelty driven molecular generation is under-explored. We train a novelty/diversity-aware RL policy to search for novel, diverse molecules that retain desired properties. Our method initializes searches from model-discovered molecules, which greatly improves the efficiency and diversity of the generated molecules.

5.2 Generative Models for Molecules

Message Passing Neural Networks (MPNN) Molecules are represented as directed graphs, where the atoms are the nodes and the bonds are the edges of the graph. More formally, let $x = (V, E)$ denote a directed graph where $v_i \in V$ are the atoms, and $e_{ij} \in E$ are the edges of the graph. The network maintains hidden states $h^t_{e_{ij}}$ for each directed edge, where t is the layer index. At each step, the hidden representations aggregate information from neighboring nodes and edges, and captures a larger neighborhood of atoms. Iteratively, the hidden states are updated as:

$$h^{t+1}_{e_{ij}} = f([h^0_{e_{ij}}; \sum_{k \in N(v_i) \setminus \{v_j\}} h^t_{e_{ki}}]) \tag{5.1}$$

Here, f is parameterized by RNN cells (e.g. LSTM cells [57] or GRU cells [19]), and $N(v_i)$ is the set of neighbors of v_i. After T steps of message-passing, the final node embeddings h_{v_i} are obtained by summing their respective incoming edge embeddings:

$$h_{v_i} = \text{ReLU}(W_o[h^0_{v_i}; \sum_{v_k \in N(v_i)} h^T_{e_{ki}}]) \tag{5.2}$$

The final node embeddings are then summed to get a graph embedding representation $h_x = \sum_{v_i} h_{v_i}$.

Vector-Quantised Variational Autoencoders (VQ-VAE) To learn useful representations of fragments, we employ the VQ-VAE architecture [130], which maps molecule fragment graphs to a discrete latent space through using categorical distributions for the prior and posterior. The VQ-VAE defines a dictionary of k embedding elements, $[s_1, s_2, ...s_k] \in \mathbb{R}^{k \times l}$. Given an input x (here the graph for a molecular fragment), let $z_e(x) \in \mathbb{R}^{d \times l}$ be the output of the encoder (a MPNN in our case). We define l to be the same dimension for both encoder output embeddings $z_e(x)$ and dictionary embeddings s_i, because input $z_q(x)$ is computed by finding the l_2 nearest neighbor dictionary elements for each row of $z_e(x)$:

$$z_q(x)_i = s_k, \text{where } k = \arg\min_j ||z_e(x)_i - s_j||_2 \text{ for } i = 1, \dots, d \tag{5.3}$$

This embedding scheme allows us to represent each molecular fragment using a length d vector, where each entry takes value from $\{1, \dots, k\}$ that corresponds to the dictionary embedding index for that row. The combinatorial vocabulary defined by the VQ-VAE has the capacity to represent $k^d 3a$ distinct molecular fragments, which lifts the constraints of a limited generative span under a fixed fragment vocabulary.

Since the discretization step does not allow for gradient flow, gradients are passed through the network through approximating the gradient from the dictionary embeddings to the encoder embeddings. Additionally, there is a *commitment loss* that encourages the encoder to output embeddings that are similar to those in the dictionary (hence commitment). The total loss of the VAE is the following:

$$\mathcal{L} = \log p(x|z_q(x)) + \sum_i ||\texttt{sg}[z_e(x)_i] - s_{ij}||^2_2 + \beta \sum_i ||z_e(x)_i - \texttt{sg}[s_{ij}]||^2_2 \tag{5.4}$$

Where s_{ij} is the closest dictionary element s_j for the $z_e(x)_i$. Additionally, β is a hyperparameter that controls for contribution of the commitment term, and `sg`

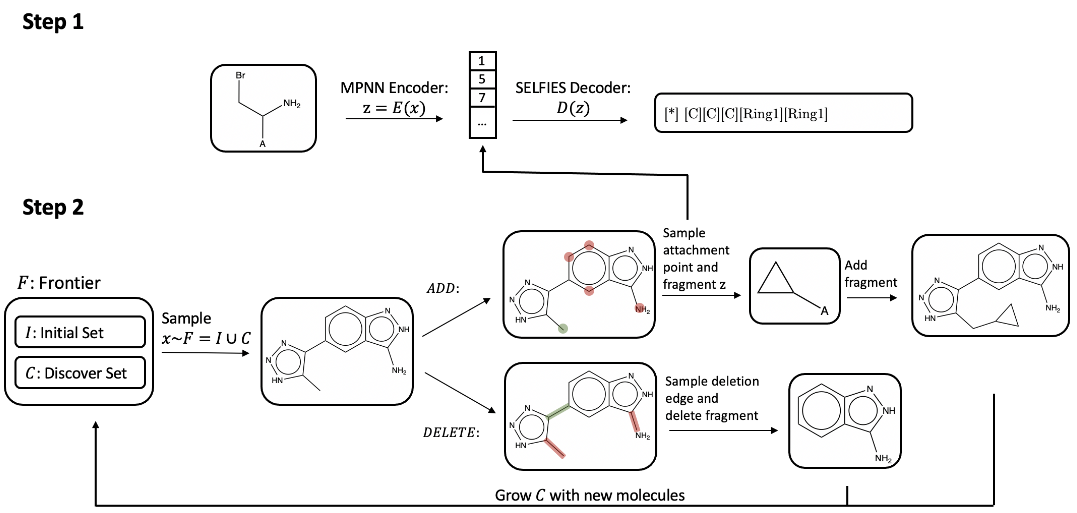

Step 1
MPNN Encoder: $z = E(x)$
1
5
7
...
SELFIES Decoder: $D(z)$
[*] [C][C][C][Ring1][Ring1]
Step 2
F: Frontier
I: Initial Set
C: Discover Set
Sample $x \sim F = I \cup C$
ADD:
DELETE:
Sample attachment point and fragment z
Add fragment
Sample deletion edge and delete fragment
Grow C with new molecules

Figure 5-1: Overview of (FaST). FaST is trained in a two-step fashion. In the first step, we train a VQ-VAE that embeds molecular fragments. In the second step, we train a search policy that uses the learned latent space as an action space. The search policy starts an episode by sampling a molecule from the *frontier* set F, which consists of an initial set of starting molecules ($\mathcal{I}$), and good molecules discovered by the policy ($\mathcal{C}$). The molecule is encoded by an MPNN, which is then used to predict either an *Add* or *Delete* action. When the *Add* action is selected, the model predicts and samples an atom as the attachment point and subsequently predicts a fragment to attach to that atom. When the *Delete* action is selected, the model samples a directed edge, indicating the molecular fragment to be deleted.

represents the stop-gradient operator.

Molecular Optimization. The goal of molecular optimization is to generate a set of high-quality molecules $\mathcal{C}$ ($\underline{\textbf{C}}$onstrained set) which satisfy or optimize a set of properties P. High novelty and diversity (detailed in section 5.4) are also desired for de novo generation applications. We model the molecular optimization problem as a Markov decision process (MDP), defined by the 5-tuple $\{\mathcal{S}, \mathcal{A}, p, r, \rho_0\}$, where the state space $\mathcal{S}$ is the set of all possible molecular graphs. As an overview, our method introduces novel designs over the action space $\mathcal{A}$ and the transition model p (**??**) by utilizing a distributional fragment vocabulary, learned by a VQ-VAE. We define the reward and initial state distribution, r and ρ_0 (section 5.3) accordingly for specified tasks and to implement the proposed sequential translation generation scheme. An illustration of our model is in fig. 5-1.

Molecular Fragments are extracted from molecules in the ChEMBL database [41]. For each molecule, we randomly sample fragments by extracting subgraphs that contain ten or fewer atoms that have a single bond attachment to the rest of the molecule. We then use a VQ-VAE to encode these fragments into a meaningful latent space. The use of molecular fragments simplifies the search problem, while the variable-sized fragment distribution maintains the reachability of most molecular compounds. Because our search algorithm ultimately uses the latent representation of the molecules as the action space, we find that using a VQ-VAE with a categorical prior instead of the typical Gaussian prior makes RL training stable and provides good performance gains [127, 50]. The training instability under a normal VAE with Gaussian prior and continuous latents causes failture of the RL training. Our ablation study also shows that the fragment samples from a VQ-VAE are more diverse than the samples from a continuous VAE.

Encoder/Decoder We use MPNN encoders for any graph inputs, which include both fragments for the VQ-VAE, as well as molecular states during policy learning. The graph models are especially suitable for describing actions on the molecular state, as they explicitly parametrize the representations of each atom and bond. Meanwhile, the decoder architecture is a recurrent network that decodes a SELFIES representation of a molecule. We choose a recurrent network for the decoder because we do not need the full complexity of a graph decoder. Due to the construction scheme, the fragments are rooted trees, and all have a single attachment point. As our fragments are small in molecular size (≤ 10 atoms), the string grammar is simple to learn, and we find the SELFIES decoder works well empirically.

Adding and deleting fragments as actions. At each step of the MDP, the policy network first takes the current molecular graph as input and produces a Bernoulli distribution on whether to add or delete a fragment. Equipped with the fragment VQ-VAE, we define the *Add* and *Delete* actions at the fragment-level:

- **Fragment Addition.** The addition action is characterized by (1) a probability distribution over the atoms of the molecule: $p_{add}(v_i) = \sigma[\text{MLP}(h_v)]$, where σ is the softmax operator. (2) Conditioned on the graph embedding h_x and the attachment point atom v_{add} sampled from p_{add}, we predict a d-channel categorical distribution $p_{fragment} = \sigma[\text{MLP}([h_{v_{add}}; h_x])] \in \mathbb{R}^{d \times k}$, where each row of $p_{fragment}$ sums to 1. We can then sample the discrete categorical latent $z_{add} \in \{1, ..., k\}^d$ from $p_{fragment}$. The fragment to add is then obtained by deocoding z_{add} through the learned frozen fragment decoder. We then assemble the decoded fragment with the current molecular graph by attaching the fragment to the predicted attachment point v_{add}. Note that the attachment point over the fragment is indicated through the generated SELFIES string.

- **Fragment Deletion.** The deletion action acts over the directed edges of the molecule. A probability distribution over deletable edges is computed with a MLP: $p_{del}(e_{ij}) = \sigma[\text{MLP}(h_{e_{ij}})]$. One edge is then sampled and deleted; since the edges are directed, the directionality specifies the the molecule to keep and the fragment to be deleted.

With the action space $\mathcal{A}$ defined as above, the transition model for the MDP is simply $p(s'|s, a) = 1$ if applying the addition/deletion action a to the molecule s results in the molecule s', and $p(s'|s, a) = 0$ otherwise. The fragment-based action space is powerful and suitable for policy learning as it (1) is powered by the enormous distributional vocabulary learned by the fragment VQ-VAE, thus spans a diverse set of editing operations over molecular graphs; (2) exploits the meaningful latent representation of fragments since the representation of similar fragments are grouped together. These advantages greatly simplify the molecular search problem. We terminate an episode when the molecule fails to satisfy the desired property or when the episode exceeds ten steps.

5.3 Discover Novel Molecules through Sequential Translation

We propose sequential translation that incrementally grows the set of discovered novel molecules and use the model-discovered molecules as starting points for further search episodes. This regime of starting exploration from states reached in previous episodes was also explored under the setting of RL from image inputs [33]. More concretely, we implement sequential translation with a reinforcement learning policy that operates under the fragment-based action space, while using a moving initial state distribution ρ_0, which is a distribution over molecules in the *frontier* set $\mathcal{F} = \mathcal{I} \cup \mathcal{C}$. By starting new search episodes from the frontier set – the union of the initial set and good molecules that are discovered by the RL policy, we achieve efficient search in the chemical space by staying close to the high-quality subspace and achieve novel molecule generation through a sequence of fragment-based editing operations to the known molecules. Our proposed search algorithm is detailed in section 5.3.

```
1: Input $N$ the desired number of discovered new molecules
2: Input $\mathcal{I}$ the initial set of molecules
3: Input $D$ the pretrained fragment decoder of VQ-VAE
4: Input $C_P : \mathcal{S} \to \{0,1\}$ returns 1 if the input $x$ satisfies desired properties    ▷ eq. (5.5)
5: Input $C_{ND} : \mathcal{S} \to \{0,1\}$ returns 1 if the input $x$ satisfies novelty/diversity criterion    ▷ eq. (5.6)
6: Let $\mathcal{C} = \emptyset$ be the discovered set of molecules
7: Let $\mathcal{F} = \mathcal{I} \cup \mathcal{C}$ be the frontier where search is initialized from
8: Let $t = 0$ be the number of episodes
9: while $|\mathcal{C}| \leq N$ do
10:     Let $t = t + 1$
11:     Update $UCB(x_0, t) \forall x_0 \in \mathcal{F}$ according to eq. (5.8)
12:     Sample initial molecule $x = (V, E)$ from $p_{init} = \sigma[UCB(x_0, t)] \forall x_0 \in \mathcal{F}$
13:     Let step $= 0$
14:     while $C_P(x) = 1$ & step $<$ T do    ▷ T $= 10$ in our experiments
15:         Encode $x$ with MPNN($x$) to get node representation $h_v, \forall v \in V$ and graph representation $h_x$
16:         Sample action type $a \in \{\text{ADD}, \text{DELETE}\}$ from $p_{action} = \sigma[\text{MLP}(h_x)]$
17:         if $a =$ ADD then
18:             Sample $v_{add}$ from $p_{add}(v) = \sigma[\text{MLP}(h_v)]\ \forall v \in V$
19:             Sample fragment encoding $z_{add}$ from $p_{fragment} = \text{MLP}([h_x; h_{v_{add}}])$
20:             Decode fragment $y = D(z_{add})$
21:             Add fragment $y$ to molecule: $x \leftarrow x + y$
22:         else
23:             Sample $e$ from $p_{del}(e) = \sigma[\text{MLP}(h_{e_{ij}})]\ \forall e \in E$
24:             Let $y$ be the fragment designated by $e$, delete fragment $x \leftarrow x - y$
25:         if $C_P(x) = 1$ & $C_{ND}(x) = 1$ then
26:             $\mathcal{C} \leftarrow \mathcal{C} \cup \{x\}$
27:             $\mathcal{F} \leftarrow \mathcal{I} \cup \mathcal{C}$
28:         Let step $\leftarrow$ step $+ 1$
```

Discover novel molecules and expand the frontier. Our method explores the chemical space with a property-aware and novelty/diversity-aware reinforcement learning policy that proposes addition/deletion modifications to the molecular state at every environment step to optimize for the reward r. We gradually expand the discovered set $\mathcal{C}$ by adding *qualified* molecules found in the RL exploration within the MDP. A molecule x is qualified if: (1) x satisfies the desired properties measured by property scores

$$C_P(x) = \prod_{p \in P} \mathbb{1}\{\texttt{score}_p(x) > \texttt{threshold}_p\} \tag{5.5}$$

where P is the set of desired properties and $\texttt{threshold}_p$ is the score threshold for satisfying property p. A molecule x satisfying all desired properties has $C_P(x) = 1$ and $C_P(x) = 0$ otherwise. (2) x is novel/diverse compared to molecules currently in the frontier F, measured by fingerprint similarity (detailed in section 5.4):

$$C_{ND}(x) = \mathbb{1}\{\max_{i \in I} \texttt{sim}(x, i) < \texttt{threshold}_{nov}\} \cdot \mathbb{1}\{\underset{g \in G}{\text{mean}}(\texttt{sim}(x, g)) < \texttt{threshold}_{div}\} \tag{5.6}$$

Where $\texttt{sim}$ denotes fingerprint similarity, $\texttt{threshold}_{nov}$ and $\texttt{threshold}_{div}$ are predefined similarity thresholds for novelty and diversity, $\mathcal{I}$ and $\mathcal{C}$ are the initial set of good molecules and model discovered molecules as defined in previous sections. A molecule that satisfies novelty/diversity criterion has $C_{ND}(x) = 1$ and $C_{ND}(x) = 0$ otherwise.

We use a reward of $+1$ for a transition that results in a molecule qualified for the set $\mathcal{C}$, and discourage the model from producing invalid molecules by adding a reward of -0.1 for a transition that produces an invalid molecular graph [1]:

$$r(x, a) = C_P([x \leftarrow a]) \cdot C_{ND}([x \leftarrow a]) - 0.1 \cdot \mathbb{1}([x \leftarrow a] \text{ invalid}) \tag{5.7}$$

where $[x \leftarrow a]$ denotes the molecule resulting from editing x with the fragment addition/deletion action a.

Initialize search episodes from promising candidates. To bias the initial state distribution ρ_0 to favor molecules that can derive more novel high-quality molecules, we keep an upper-confidence-bound (UCB) score for each initial molecule in the frontier F. We record the number of times we initiate a search $N(x, t)$ from a

[1] validity is checked by the chemistry software RDKit.

molecule $x \in F$, and the number of molecules qualified for adding to $\mathcal{C}$ that is found in an episode strating from x: $R(x,t)$. Here $t = \sum_{x \in \rho_0} N(x)$ is the total number of search episodes. The UCB score of the initial molecule m is calculated by:

$$UCB(x,t) = \frac{R(x,t)}{N(x,t)} + \frac{\sqrt{\frac{3}{2}\log(t+1)}}{N(x,t)} \tag{5.8}$$

The probability of a molecule in the initialization set being sampled as the starting point of a new episode is then computed by a softmax over the UCB scores: $p_{init}(x, t+1) = \frac{\exp(UCB(x,t))}{\sum_{x \in I} \exp(UCB(x,t))}$. To summarize, FaST learns a policy that (1) choose good initial molecules to start search episodes; (2) choose to add a fragment to or delete a subgraph from a given state (a molecule in our case); (3) choose what to add through predicting a fragment latent embedding, or what to delete through predicting a directed edge, and remove part of the molecular graph accordingly.

Although we present our method in this section under the most realistic multi-objective optimization task settings (with experiments in section 5.4), our method is easily extendable to other problem settings by modifying the constraints C_P, C_{ND}, and the reward function r accordingly.

5.4 Experiments

Datasets. We use benchmark datasets for molecular optimization, which aims to generate ligand molecules for inhibition of two proteins: glycogen synthase kinase-3 beta (GSK3β) and c-Jun N-terminal kinase 3 (JNK3). Following previous work [64, 140, 98], we adopt the same strategy of using a random forest trained on these datasets as the oracle property predictor, and incorporate the additional factors, quantitative estimate of drug-likeliness (QED) [9] and synthetic accessibility (SA) [35] as our optimization objectives. Single property optimization is often a flawed task, because the generator can overfit to the pretrained predictor and generate unsynthesizable compounds.

Evaluation metrics. Following previous works, we evaluate our generative model on three target metrics, success, novelty and diversity. 5,000 molecules are generated by the model, and the metric scores are computed as follows: **Success rate (SR)** measures the proportion of generated molecules that fit the desired properties. **Novelty (Nov)** measures how different the generated molecules are compared to the set of actives in the dataset (range $[0, 1]$), and **Diversity (Div)** measures how different the generated molecules are compared to each other (range $[0, 1]$). **PM** is the product of the three metrics above ($\mathbf{PM} = \mathbf{SR} \cdot \mathbf{Nov} \cdot \mathbf{Div}$).

Implementation details. We construct the initial set of molecules for our search algorithm from the rationales extracted from [64]. These rationales are obtained through a sampling process on the active molecules that tries to minimize the size of the rationale subgraph, while maintaining their inhibitory properties. Rationales for multi-property tasks (GSK3β+JNK3) are extracted by combining the rationales for single-property tasks. Initializing generation with subgraphs is commonly done in molecular generative models such as [121] and [76]. We train the RL policy using the Proximal Policy Optimization (PPO, [115]) algorithm. We find the RL training robust despite both the reward function r and the initial state distribution ρ_0 are non-stationary (i.e., changing during RL training).

Baseline methods. Rationale-RL [64] extracts rationales of the active molecules and then uses RL to train a completion model that add atoms to the rationale in a sequential manner to generate molecules satisfying the desired properties. **GA+D & JANUS** [97, 98] are two genetic algorithms that use random mutations of SELFIES strings to generate promising molecular candidates; JANUS leverages a two-pronged approach, accounting for mutations towards both exploration and exploitation. **MARS** [140] uses Markov Chain Monte Carlo (MCMC) sampling to iterative build new molecules by adding or removing fragments, and the model is trained to fit the distribution of the active molecules. To provide a fair comparison against baselines that do not use rationales, we additionally include a baseline **MARS+Rationale** that initialize the MARS algorithm with the same starting initial rationale set used in Rationale-RL and our method. where possible, we use the numbers from the original corresponding paper.

Performance. The evaluation metrics are shown in table 5.1; FaST signifi-

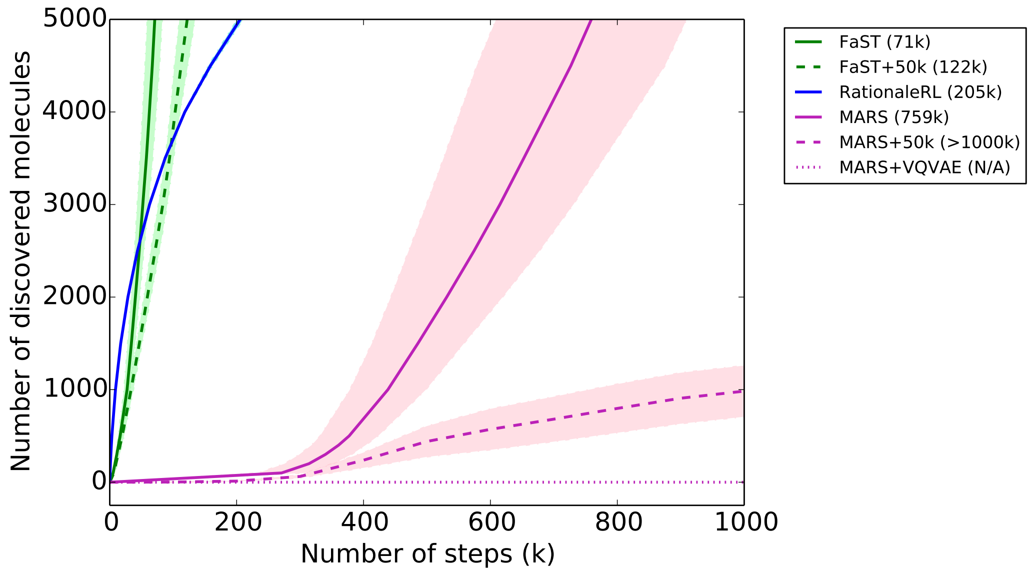


Figure 5-2: Comparison of the sample complexity of different methods. The x-axis is the number of molecules searched through and the y-axis is the number of discovered molecules, where the target is to obtain a set of 5,000 molecules that achieves **SR** = 1, **Nov** = 1 and **Div** = .7. Our method (FaST) achieves the best sample complexity with 71k molecules visited. Fast+50k and MARS+50k are their respective models trained with the same fixed fragment vocabulary extracted from ChEMBL.

Table 5.1: FaST outperforms all baselines on both single-property and multi-property optimization. Error bars indicates one standard deviation, obtained from averaging 5 random seeds.

Model	GSK3β				GSK3β+QED+SA			
	SR	Nov	Div	PM	SR	Nov	Div	PM
Rationale-RL	1.00	.534	.888	.474	.699	.402	.893	.251
GA+D	.846	1.00	.714	.600	.891	1.00	.628	.608
JANUS	1.00	.829	.884	.732	-	-	-	-
MARS	1.00	.840	.718	.600 (± .04)	.995	.950	.719	.680 (± .03)
MARS+Rationale	.995	.804	.746	.597 (± .07)	.981	.800	.807	.632 (± .07)
FaST	1.00	1.00	.905	**.905 (± .000)**	1.00	1.00	.861	**.861 (± .001)**

Model	JNK3				JNK3+QED+SA			
	SR	Nov	Div	PM	SR	Nov	Div	PM
Rationale-RL	1.00	.462	.862	.400	.623	.376	.865	.203
GA+D	.528	.983	.726	.380	.857	.998	.504	.431
JANUS	1.00	.426	.895	.381	-	-	-	-
MARS	.988	.889	.748	.660 (± .04)	.913	.948	.779	.674 (± .02)
MARS+Rationale	.976	.843	.780	.642 (± .04)	.634	.779	.787	.386 (± .08)
FaST	1.00	1.00	.905	**.905 (± .001)**	1.00	.866	.856	**.741 (± .001)**

Model	GSK3β+JNK3				GSK3β+JNK3+QED+SA			
	SR	Nov	Div	PM	SR	Nov	Div	PM
Rationale-RL	1.00	.973	.824	.800	.750	.555	.706	.294
GA+D	.847	1.00	.424	.360	.857	1.00	.363	.311
JANUS	1.00	.778	.875	.681	1.00	.326	.821	.268
MARS	.995	.753	.691	.520 (± .08)	.923	.824	.719	.547 (± .05)
MARS+Rationale	.976	.843	.780	.642 (± .04)	.654	.687	.724	.321 (± .09)
FaST	1.00	1.00	.863	**.863 (± .001)**	1.00	1.00	.716	**.716 (± .011)**

cantly outperforms all baselines on all tasks including both single-property and multi-property optimization. On the most challenging task, GSK3β+JNK3+QED+SA, FaST improves upon the previous best model by over 30% in the product of the three evaluation metrics. Our model is able to efficiently search for molecules that stay within the constrained property space, and discover novel and diverse molecules by sequentially translating known and discovered active molecules. The MARS+Rationale model, which uses the same rationales as the initialization for their search algorithm, does not perform well compared to the original implementation, which initializes each search with a simple "C-C" molecule.

Sample complexity comparison given performance thresholds. Another comparison scheme is to let a model keep generating molecules until it achieves a good candidate set under certain performance thresholds. Under this evaluation protocol, all models will have the same or very similar performance in SR, Nov, and Div. The metric of interest will then be the sample complexity of the algorithm: how many molecules it requires to visit/generate to obtain a good candidate set. This places every model under the same regime, allowing each model to generate molecules in a novelty/diversity-aware setting. We compare FaST to Rationale-RL and MARS under this setting in fig. 5-2, where we impose SR=1, Nov=1, Div=.7 for the candidate set. FaST on average searched through 71k molecules in total to gather the 5k proposal set, while Rational-RL and MARS need to search through 205k and 759k molecules to obtain their corresponding proposal sets. Being a pretrained generative model, Rationale-RL has a steeper slope initially but then slows down to find more good molecules. The flexibility of our learned vocabulary and the RL search strategy lead to the superior performance of FaST.

Optimize for different novelty/diversity metrics. The Morgan fingerprints used for similarity comparison contain certain inductive biases. Under different applications, different novelty/diversity metrics may be of interest. To demonstrate the viability of our model under any metrics, we train FaST using Atom Pairs (AP) fingerprints [16] on the GSK3β+JNK3+QED+SA task. The results, and discussion of the different fingerprint methods, are reported in section 5.4. We find that (1) FaST can still find high-quality molecules that are novel and diverse, while the baseline methods suffer a low novelty; (2) FaST trained with AP fingerprints still attains good performance when evaluated under Morgan fingerprints but the reverse is not true. This result shows that Novelty/diversity under AP fingerprints is a stricter criterion to satisfy and necessitates novelty/diversity-awareness during optimization.

Diversity of generation. In addition to the fingerprint diversity metrics presented in section 5.4, we also examine functional group diversity. We extract all unique molecular fragments of the 5,000 molecules generated for GSK3β+JNK3+QED+SA task for each model, and produce t-SNE visualization of these fragments in fig. 5-3 and fig. 5-4. In total, we extracted 1.7k unique fragments from our model outputs vs only 1.1k unique fragments for Rational-RL and 500 unique fragments from MARS. The visualization shows that the fragments in the molecules generated by our model spans a much larger chemical space. This confirms the advantages of using a learned vocabulary, compared to using a fixed set of fragments, as we are able to utilize a much more diverse set of chemical subgraphs.

Benefit of distributional vocabulary. To investigate the benefit of using a distributional vocabulary, instead of using the pretrained VQ-VAE, we also train our model using a fixed vocabulary of fragments, which consists of roughly 50k unique fragments (the same set used to pretrain the VQ-VAE). fig. 5-2 compares the performance of the two models. On average, the model with fixed fragments took 122k steps, while with VQ-VAE it only took 71k steps to find a set of 5,000 good molecules (72% improvement). We further analyze the benefit of using discrete latents with a VQ-VAE rather than continuous latents with a Gaussian prior VAE in the experiments.

Importance of RL search. We also demonstrate the importance of our RL search policy compared to previous sampling methods. To do so, we run MARS with both the 50k fixed fragments and our VQ-VAE. fig. 5-2 shows the performance of using the 50k fixed fragment vocabulary compared to the original MARS model which uses a small 1k vocabulary. When the vocabulary is large, MARS exhibits very poor sample complexity. Additionally, we also implemented our VQ-VAE with the sampling

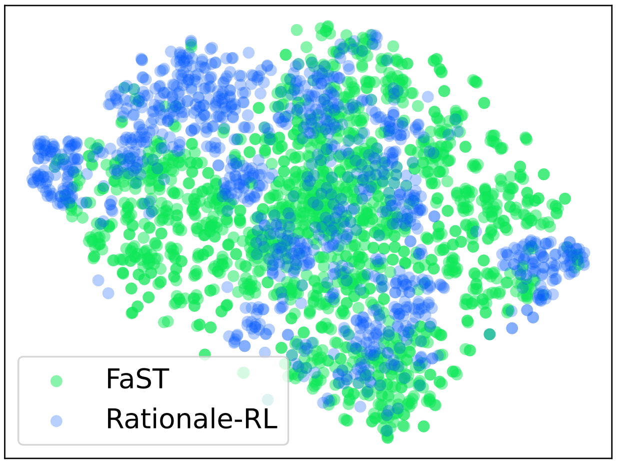


Figure 5-3: This plots the t-SNE embedding of fragments from generated compounds of our model vs. Rational-RL. The visualization shows that that our model produces a much more diverse set of fragments, which is a proxy for functional groups appearing in generated molecules.

strategy proposed in MARS, but this model was altogether unable to successfully generate good candidate molecules. Therefore, we see that when the vocabulary is more complex, we need a better search strategy, highlighting the importance of our RL algorithm.

Vocabulary learning through VQ-VAE To evaluate the benefits of VQ-VAE over a typical VAE trained with Gaussian priors, we train both models, and look at the distribution of fragments. fig. 5-5 compares the t-SNE distributions of the two models, where we sample 2,000 fragments from each model. The VAE model has tight clusters, while the VQ-VAE model exhibits a much more diverse set of fragments. We visualize random samples from VAE and VQ-VAE fig. 5-6, where we see that the samples from VAE are relatively simple and generic fragments, while samples from the VQ-VAE demonstrate diverse patterns. This is because the more generic fragments appear more frequently in real molecules, and a Gaussian prior over the fragment latent space would favor these fragments.

Constrained Penalized LogP To demonstrate the general applicability of our model for any molecular optimization task, we also run our model on another con-

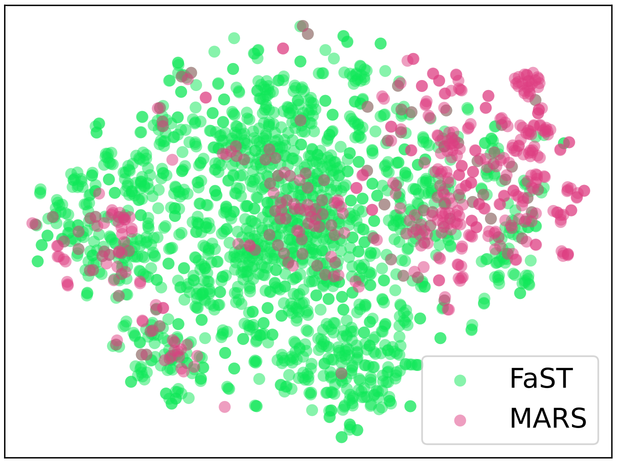


Figure 5-4: This plots the t-SNE embedding of fragments from generated compounds of our model vs. MARS. The visualization shows that that our model produces a much more diverse set of fragments, which is a proxy for functional groups appearing in generated molecules.

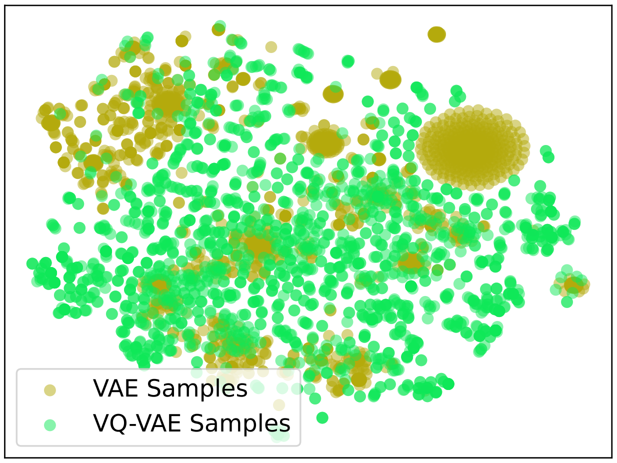


Figure 5-5: t-SNE of fragments sampled from a trained VAE and VQ-VAE. The fragments sampled from the VAE are tightly clustered, showing much less diversity compared to the fragments sampled from the VQ-VAE.

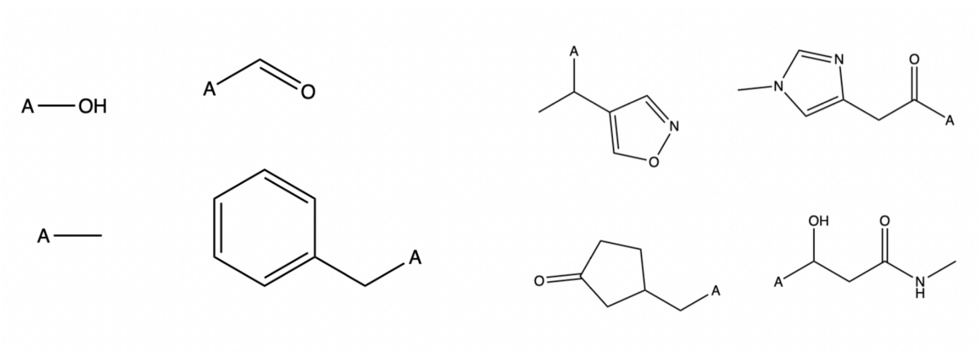


Figure 5-6: (a,b) Random samples from the VAE (a) and the VQ-VAE (b). “A" denotes the attachment point for the fragment. We see that the samples from the VAE are relatively simple. Meanwhile, the samples from the VQ-VAE are more diverse.

strained optimization task, here optimizing for penalized octanol-water partition coefficients (logP) scores of ZINC [60] molecules. The penalized logP score is the logP score penalized by synthetic accessibility and ring size. We use the exact computation in [146], where the components of the penalized logP score are normalized across the entire 250k ZINC training set. The generated molecules are constrained to have similar Morgan fingerprints [108] as the original molecules.

Following the same setup as previous work [67, 146, 121, 97, 76], we try to optimize the 800 test molecules from ZINC with the lowest penalized logP scores (the initial set $\mathcal{I}$). Specifically, the task is to translate these molecules into new molecules with the Tanimoto similarity of the fingerprints constrained within $\delta \in \{.4, .6\}$. This task aims for optimizing a certain quantity (instead of satisfying property constraints) and is a translation task (need to stay close to original molecules rather than finding novel ones). To run FaST on this task, we apply the following changes to the reward function, the qualification criterion, and the episode termination criterion, of FaST. We denote $\texttt{score}(x)$ to be the penalized logP scoring function, and $\texttt{sim}(\cdot, \cdot)$ to be the Tanimoto similarity between two molecules:

- reward $r = \texttt{score}(x_j) - \texttt{score}(x_i)$ for any transition from molecule $x_i \rightarrow x_j$
- $\mathcal{C}$, the discovered set contains all explored molecules that satisfy eq. (5.5), where

the threshold is given by the input parameter δ

- We terminate an episode when the number of steps exceeds 10.

For each molecule we add to G, we keep track of its original parent (the molecule from the 800 test molecules). After training, for each of the 800 test molecules, we take the set of translated molecules in G, and select the one with the highest property score.

Table 5.2: Results on the constrained penalized logP task. FaST significantly outperform all baselines.

Method	$\delta = 0.4$		$\delta = 0.6$	
	Improvement	Success	Improvement	Success
JT-VAE [61]	0.84 ± 1.45	83.6 %	0.21 ± 0.71	46.4 %
GCPN [146]	2.49 ± 1.30	100.0 %	0.79 ± 0.63	100.0 %
DEFactor [3]	3.41 ± 1.67	85.9 %	1.55 ± 1.19	72.6 %
MolDQN [150]	3.37 ± 1.62	100 %	1.86 ± 1.21	100 %
GraphAF [121]	3.74 ± 1.25	100 %	1.95 ± 0.99	100 %
GP-VAE [76]	4.19 ± 1.30	98.9 %	2.25 ± 1.12	90.3 %
VJTNN [67]	3.55 ± 1.67	-	2.33 ± 1.17	-
GA+D [97]	5.93 ± 1.14	100 %	3.44 ± 1.09	99.8 %
FaST (Ours)	18.09 ± 8.72	100 %	8.98 ± 6.31	96.9 %

Results are shown in table 5.2; our method greatly outperforms the other baselines, but we point out a few flaws intrinsic to the task. Because the similarity is computed through Morgan fingerprint, which are hashes of substructures, repeatedly adding aromatic rings can often not change the fingerprint by a lot. Nevertheless, adding aromatic rings will linearly increase penalized logP score, which allows trivial solutions to produce high scores for this task (see fig. 5-7). This phenomenon is noted by [97], but they add a regularizer to constrain the generated compounds to look similar to the reference molecules. Due to the mentioned issues, we believe this task can be reformulated. For instance, one could use a different fingerprint method so that the fingerprint similarity is not so easily exploited (see AP [16], MACCS [30], or ROCS

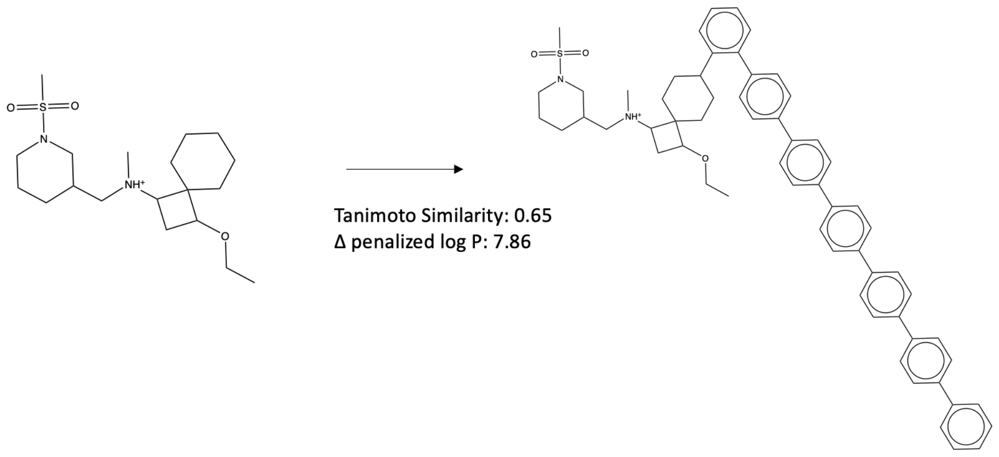


Figure 5-7: Sample translation of our model for the constrained penalized logP task ($\delta = 0.6$). The model generates a molecule with repeating aromatic rings; though not realistic, this molecule achieves a high score, while having close Tanimoto similarity using Morgan fingerprints.

[55]), or size constraints should be incorporated. Nevertheless, we provide our results for comparison to other molecular generation methods.

In general, the task of optimizing (increasing) the penalized logP scores is not entirely meaningful. According to Lipinski's rule of five [83], which are widely established rules to evaluate the druglikeness of molecules, the logP score should be lower than 5. So an unbounded optimization of logP has little practical usability. Perhaps a better task would be to optimize for all 5 rules in Linpinski's rule of five which includes constraints involving the number of hydrogen bond donors/acceptors and molecular mass.

Different Novelty/Diversity Metrics FaST is capable of optimizing for different novelty/diversity metrics. In this section, we compute the novelty/diversity metrics using atom-pair (AP) fingerprints [16]. While Morgan fingerprints have successfully been applied to many molecular tasks such as drug screening, it has some failure modes [15]. Namely, Morgan fingerprints is often not informative about the size or the shape of the molecules. These properties are better captured in AP fingerprints, as AP fingerprints account for all atom pairs, including their pairwise distances. We run the same experiment on the GSK3β+JNK3+QED+SA task described in section 5.4, but change the fingerprint from Morgan to AP for the novelty/diversity metrics. The results are shown in table 5.3 with comparison to baselines. We observe that our method outperform baselines by a greater margin, especially in the novelty metric. This is not surprising because our model can explicitly optimize for any similarity metric, while the baseline methods are not novelty/diversity-aware during training. Interestingly, we find that optimizing for AP fingerprints also yields molecules that score high under Morgan fingerprints for this task (but the converse is not true).

Table 5.3: Results on the GSK3β+JNK3+QED+SA task using AP fingerprints instead of Morgan fingerprints for novelty/diversity computation.

Method	Success (SR)		Novelty (Nov)		Diversity (Div)	
	Morgan	AP	Morgan	AP	Morgan	AP
Rationale-RL	.750	.750	.555	.023	.706	.630
MARS	.923	.733	.824	.077	.719	.644
FaST (Morgan)	1.00	1.00	1.00	.555	.716	.674
FaST (AP)	1.00	1.00	.987	.867	.675	.719

5.5 Summary

We propose a new framework for molecular optimization, which leverages a learned vocabulary of molecular fragments to search the chemical space efficiently. We demonstrate that FaST, which adaptively grows a set of promising molecular candidates, can generate high-quality, novel, and diverse molecules on single-property and multi-property optimization tasks. Ablation study shows that all components of our proposed method contribute to its superior performance. The learning of a flexible vocabulary is a complementary module to other research in fragment-based drug design. Incorporating FaST to more practical drug discovery pipelines while taking synthesis paths in mind is an exciting avenue for future work.

Chapter 6

Conclusion

In this thesis, I looked at applying machine learning to two core problems as they relate to drug discovery: molecule representation and generation. Without the correct modeling assumptions, these models fail to comprehensively capture the important facets of the molecular problem. On the representation side, I looked at two models that relate to property prediction. Firstly, I looked at applying attention-based transformer models for graph-structured molecular inputs. Next, I explored a new paradigm for reasoning about molecules in the form of prototype learning. In addition to property prediction, I also talked about our methods for improving reaction prediction, how to propose more generalizable and diverse reactions for retrosynthesis. For the generation task, I delve into our new generation model that first learns a generation vocabulary in the form of molecular fragments. Using this learned vocabulary, we explore different search strategies to efficient find molecules satisfying multi-criteria constraints.

Bibliography

[1] Daylight chemical information systems, daylight.

[2] SN Afriat. Theory of maxima and the method of lagrange. *SIAM Journal on Applied Mathematics*, 20(3):343–357, 1971.

[3] Rim Assouel, Mohamed Ahmed, Marwin H Segler, Amir Saffari, and Yoshua Bengio. Defactor: Differentiable edge factorization-based probabilistic graph generation. *arXiv preprint arXiv:1811.09766*, 2018.

[4] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, pages 1993–2001, 2016.

[5] Gregor Bachmann, Gary Bécigneul, and Octavian-Eugen Ganea. Constant curvature graph convolutional networks. *arXiv preprint arXiv:1911.05076*, 2019.

[6] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 384–392, 2019.

[7] Javier L Baylon, Nicholas A Cilfone, Jeffrey R Gulcher, and Thomas W Chittenden. Enhancing retrosynthetic reaction prediction with deep learning using multiscale reaction classification. *Journal of chemical information and modeling*, 59(2):673–688, 2019.

[8] Andreas Bender, Hamse Y Mussa, Robert C Glen, and Stephan Reiling. Similarity searching of chemical databases using atom environment descriptors (molprint 2d): evaluation of performance. *Journal of chemical information and computer sciences*, 44(5):1708–1718, 2004.

[9] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

[10] John Bradshaw, Brooks Paige, Matt J. Kusner, Marwin H. S. Segler, and José Miguel Hernández-Lobato. A model to search for synthesizable molecules.

In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7935–7947, 2019.

[11] John Bradshaw, Brooks Paige, Matt J. Kusner, Marwin H. S. Segler, and José Miguel Hernández-Lobato. Barking up the right tree: an approach to search over molecule synthesis dags. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[12] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[13] Kerstin Bunte, Petra Schneider, Barbara Hammer, Frank-Michael Schleif, Thomas Villmann, and Michael Biehl. Limited rank matrix learning, discriminative dimension reduction and visualization. *Neural Networks*, 26:159–173, 2012.

[14] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[15] Alice Capecchi, Daniel Probst, and Jean-Louis Reymond. One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome. *Journal of cheminformatics*, 12(1):1–15, 2020.

[16] Raymond E Carhart, Dennis H Smith, and R Venkataraghavan. Atom pairs as molecular features in structure-activity studies: definition and applications. *Journal of Chemical Information and Computer Sciences*, 25(2):64–73, 1985.

[17] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 4869–4880, 2019.

[18] Sheng Chen, Colin FN Cowan, and Peter M Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on neural networks*, 2(2):302–309, 1991.

[19] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[20] Connor W. Coley, William H. Green, and Klavs F. Jensen. Machine learning in computer-aided synthesis planning. *Accounts of Chemical Research*, 51(5):1281–1289, 05 2018.

[21] Connor W. Coley, Luke Rogers, William H. Green, and Klavs F. Jensen. Computer-assisted retrosynthesis based on molecular similarity. *ACS Central Science*, 3(12):1237–1245, 12 2017.

[22] E. J. Corey and W. Todd Wipke. Computer-assisted design of complex organic syntheses. *Science*, 166(3902):178–192, 1969.

[23] Elias James Corey. The logic of chemical synthesis: multistep synthesis of complex carbogenic molecules (nobel lecture). *Angewandte Chemie International Edition in English*, 30(5):455–465, 1991.

[24] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.

[25] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711, 2016.

[26] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[27] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc., 2016.

[28] 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.

[29] Robert PW Duin and Elżbieta Pękalska. The dissimilarity space: Bridging structural and statistical pattern recognition. *Pattern Recognition Letters*, 33(7):826–832, 2012.

[30] Joseph L Durant, Burton A Leland, Douglas R Henry, and James G Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of chemical information and computer sciences*, 42(6):1273–1280, 2002.

[31] David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. *CoRR*, abs/1509.09292, 2015.

[32] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[33] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[34] Daniel A Erlanson. Introduction to fragment-based drug discovery. *Fragment-based drug discovery and X-ray crystallography*, pages 1–32, 2011.

[35] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):1–11, 2009.

[36] Matthias Fey, Jan E Lenssen, Christopher Morris, Jonathan Masci, and Nils M Kriege. Deep graph matching consensus. *arXiv preprint arXiv:2001.09621*, 2020.

[37] Rémi Flamary and Nicolas Courty. Pot python optimal transport library. *GitHub: https://github. com/rflamary/POT*, 2017.

[38] Tianfan Fu, Cao Xiao, Xinhao Li, Lucas M. Glass, and Jimeng Sun. MIMOSA: multi-constraint molecule sampling for molecule optimization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 125–133. AAAI Press, 2021.

[39] Tianfan Fu, Cao Xiao, and Jimeng Sun. Core: Automatic molecule optimization using copy & refine strategy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 638–645, 2020.

[40] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *International Conference on Machine Learning*, pages 2083–2092, 2019.

[41] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

[42] Herbert Gelernter, J Royce Rose, and Chyouhwa Chen. Building and refining a knowledge base for synthetic organic chemistry via the methodology of inductive and deductive machine learning. *Journal of chemical information and computer sciences*, 30(4):492–504, 1990.

[43] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. *arXiv preprint arXiv:1706.00292*, 2017.

[44] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017.

[45] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[46] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[47] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, JoséMiguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 02 2018.

[48] Liyu Gong and Qiang Cheng. Adaptive edge features guided graph attention networks. *arXiv preprint arXiv:1809.02709*, 2018.

[49] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[50] Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Rémi Munos. Monte-carlo tree search as regularized policy optimization. In *International Conference on Machine Learning*, pages 3769–3778. PMLR, 2020.

[51] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. 2017.

[52] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[53] Barbara Hammer and Thomas Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.

[54] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[55] Paul CD Hawkins, A Geoffrey Skillman, Gregory L Warren, Benjamin A Ellingson, and Matthew T Stahl. Conformer generation with omega: algorithm and validation using high quality structures from the protein databank and cambridge structural database. *Journal of chemical information and modeling*, 50(4):572–584, 2010.

[56] Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. Sequence to sequence mixture model for diverse machine translation. *arXiv preprint arXiv:1810.07391*, 2018.

[57] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[58] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[59] James P Hughes, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011.

[60] John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

[61] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

[62] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. 02 2018.

[63] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical graph-to-graph translation for molecules. *arXiv preprint arXiv:1907.11223*, 2019.

[64] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International Conference on Machine Learning*, pages 4849–4859. PMLR, 2020.

[65] Wengong Jin, Connor W. Coley, Regina Barzilay, and Tommi S. Jaakkola. Predicting organic reaction outcomes with weisfeiler-lehman network. *CoRR*, abs/1709.04555, 2017.

[66] Wengong Jin, Connor W. Coley, Regina Barzilay, and Tommi S. Jaakkola. Predicting organic reaction outcomes with weisfeiler-lehman network. *CoRR*, abs/1709.04555, 2017.

[67] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecule optimization. In *International Conference on Learning Representations*, 2019.

[68] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi S. Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. *CoRR*, abs/1812.01070, 2018.

[69] Seokho Kang and Kyunghyun Cho. Conditional molecular design with deep generative models. *Journal of chemical information and modeling*, 59(1):43–52, 2018.

[70] Pavel Karpov, Guillaume Godin, and Igor V Tetko. A transformer model for retrosynthesis. In *International Conference on Artificial Neural Networks*, pages 817–830. Springer, 2019.

[71] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[72] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017.

[73] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.

[74] Teuvo Kohonen. Learning vector quantization. In *Self-organizing maps*, pages 175–189. Springer, 1995.

[75] Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. In *International conference on machine learning*, 2018.

[76] Xiangzhe Kong, Zhixing Tan, and Yang Liu. Graphpiece: Efficiently generating high-quality molecular graph with substructures. *arXiv preprint arXiv:2106.15098*, 2021.

[77] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

[78] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1945–1954. JMLR. org, 2017.

[79] Jules Leguy, Thomas Cauchy, Marta Glavatskikh, Béatrice Duval, and Benoit Da Mota. Evomol: a flexible and interpretable evolutionary algorithm for unbiased de novo molecular generation. *Journal of cheminformatics*, 12(1):1–19, 2020.

[80] Yuan Li, Xiaodan Liang, Zhiting Hu, Yinbo Chen, and Eric P. Xing. Graph transformer, 2019.

[81] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter W. Battaglia. Learning deep generative models of graphs. *CoRR*, abs/1803.03324, 2018.

[82] Kangjie Lin, Youjun Xu, Jianfeng Pei, and Luhua Lai. Automatic retrosynthetic pathway planning using template-free models. *arXiv preprint arXiv:1906.02308*, 2019.

[83] Christopher A Lipinski, Franco Lombardo, Beryl W Dominy, and Paul J Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced drug delivery reviews*, 23(1-3):3–25, 1997.

[84] Bowen Liu, Bharath Ramsundar, Prasad Kawthekar, Jade Shi, Joseph Gomes, Quang Luu Nguyen, Stephen Ho, Jack Sloane, Paul Wender, and Vijay Pande. Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS Central Science*, 3(10):1103–1113, 2017. PMID: 29104927.

[85] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L Gaunt. Constrained graph variational autoencoders for molecule design. *arXiv preprint arXiv:1805.09076*, 2018.

[86] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. In *Advances in Neural Information Processing Systems*, pages 8228–8239, 2019.

[87] Daniel Mark Lowe. *Extraction of chemical structures and reactions from the literature*. PhD thesis, University of Cambridge, 2012.

[88] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[89] Minh-Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, 2013.

[90] Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. *arXiv preprint arXiv:1809.02630*, 2018.

[91] Malcolm J McGregor and Steven M Muskal. Pharmacophore fingerprinting. 2. application to primary library design. *Journal of chemical information and computer sciences*, 40(1):117–125, 2000.

[92] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design. *Machine Learning: Science and Technology*, 2(2):025023, 2021.

[93] Federico Monti, Oleksandr Shchur, Aleksandar Bojchevski, Or Litany, Stephan Günnemann, and Michael M. Bronstein. Dual-primal graph convolutional networks. *CoRR*, abs/1806.00770, 2018.

[94] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347, 2017.

[95] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. *CoRR*, abs/1605.05273, 2016.

[96] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[97] AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alán Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[98] AkshatKumar Nigam, Robert Pollice, and Alan Aspuru-Guzik. Janus: Parallel tempered genetic algorithm guided by deep neural networks for inverse molecular design. 2021.

[99] AkshatKumar Nigam, Robert Pollice, Mario Krenn, Gabriel dos Passos Gomes, and Alan Aspuru-Guzik. Beyond generative models: superfast traversal, optimization, novelty, exploration and discovery (stoned) algorithm for molecules using selfies. *Chemical science*, 2021.

[100] Emmanuel Noutahi, Dominique Beaini, Julien Horwood, Sébastien Giguère, and Prudencio Tossou. Towards interpretable sparse graph representation learning with laplacian pooling. *arXiv preprint arXiv:1905.11577*, 2019.

[101] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.

[102] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[103] Hongbin Pei, Bingzhe Wei, Chen-Chuan Chang Kevin, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International conference on machine learning*, 2020.

[104] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[105] Marco Podda, Davide Bacciu, and Alessio Micheli. A deep generative model for fragment-based molecule generation. In *International Conference on Artificial Intelligence and Statistics*, pages 2240–2250. PMLR, 2020.

[106] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[107] Pau Riba, Andreas Fischer, Josep Lladós, and Alicia Fornés. Learning graph distances with message passing neural networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2239–2244. IEEE, 2018.

[108] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[109] Bidisha Samanta, Abir De, Gourhari Jana, Vicenç Gómez, Pratim Kumar Chattaraj, Niloy Ganguly, and Manuel Gomez-Rodriguez. Nevae: A deep generative model for molecular graphs. *Journal of machine learning research. 2020 Apr; 21 (114): 1-33*, 2020.

[110] Atsushi Sato and Keiji Yamada. Generalized learning vector quantization. *Advances in neural information processing systems*, 8:423–429, 1995.

[111] Koji Satoh and Kimito Funatsu. A novel approach to retrosynthetic analysis using knowledge bases derived from reaction databases. *Journal of chemical information and computer sciences*, 39(2):316–325, 1999.

[112] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[113] Morgan A Schmitz, Matthieu Heitz, Nicolas Bonneel, Fred Ngole, David Coeurjolly, Marco Cuturi, Gabriel Peyré, and Jean-Luc Starck. Wasserstein dictionary learning: Optimal transport-based unsupervised nonlinear dictionary learning. *SIAM Journal on Imaging Sciences*, 11(1):643–678, 2018.

[114] Petra Schneider, Michael Biehl, and Barbara Hammer. Adaptive relevance matrices in learning vector quantization. *Neural computation*, 21(12):3532–3561, 2009.

[115] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017.

[116] Philippe Schwaller, Teodoro Laino, Theophile Gaudin, Peter Bolgar, Costas Bekas, and Alpha A. Lee. Molecular Transformer – A Model for Uncertainty-Calibrated Chemical Reaction Prediction. 5 2019.

[117] Marwin H. S. Segler and Mark P. Waller. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chemistry – A European Journal*, 23(25):5966–5971, 2017.

[118] Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131, 2018.

[119] Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. Mixture models for diverse machine translation: Tricks of the trade. *arXiv preprint arXiv:1902.07816*, 2019.

[120] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

[121] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

[122] Bonggun Shin, Sungsoo Park, JinYeong Bak, and Joyce C Ho. Controlled molecule generator for optimizing multiple chemical properties. In *Proceedings of the Conference on Health, Inference, and Learning*, pages 146–153, 2021.

[123] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pages 412–422. Springer, 2018.

[124] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[125] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002.

[126] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.

[127] Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5981–5988, 2020.

[128] Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. In *Advances in Neural Information Processing Systems*, pages 6436–6446, 2019.

[129] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery*, 18(6):463–477, 2019.

[130] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6306–6315, 2017.

[131] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[132] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[133] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[134] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[135] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

[136] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010.

[137] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[138] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378, 2016.

[139] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[140] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. {MARS}: Markov molecular sampling for multi-objective drug discovery. In *International Conference on Learning Representations*, 2021.

[141] Hongteng Xu. Gromov-wasserstein factorization models for graph clustering. *arXiv preprint arXiv:1911.08530*, 2019.

[142] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *CoRR*, abs/1810.00826, 2018.

[143] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[144] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

[145] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, pages 4800–4810, 2018.

[146] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay S. Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6412–6422, 2018.

[147] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[148] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *CoRR*, abs/1803.07294, 2018.

[149] Shuangjia Zheng, Jiahua Rao, Zhongyue Zhang, Jun Xu, and Yuedong Yang. Predicting retrosynthetic reaction using self-corrected transformer neural networks, 2019.

[150] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.