Graph Representation Learning for Molecule Screening and Generation

by

Wengong Jin

B.Eng., Shanghai Jiao Tong University (2016)
S.M., Massachusetts Institute of Technology (2018)

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

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2021

© Massachusetts Institute of Technology 2021. All rights reserved.
Graph Representation Learning for Molecule Screening and Generation

by

Wengong Jin

Submitted to the Department of Electrical Engineering and Computer Science on August 27, 2021, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

Drug discovery is an expensive and labor-intensive process, typically taking an average of 10–15 years. The goal of this thesis is to substantially accelerate this process by developing machine learning (ML) algorithms for three key steps in drug discovery pipeline. First, we develop better property predictors that enable us to effectively navigate known chemical space. The main challenge is to learn a predictor based on a small, biased assay and generalize to a much broader chemical space. We address this challenge by a new domain generalization method called counterfactual consistency regularization, which seeks to eliminate spurious correlations in biological assays. Second, we extend property prediction capabilities to combinations of molecules, enabling us to screen and discover synergistic drug therapies. Direct experimental data about combinations are extremely limited. To counter this limitation, we build more biological structure (drug-target interaction) into the models in order to leverage heterogeneous single-compound assays as well as to provide a mechanism to assess drug combinations through competitive binding to such targets. Third, we extend the search for new drugs beyond known chemical matter by developing deep generative models that can realize novel compounds with better characteristics. To this end, we propose hierarchical graph generative models that make use of larger structural building blocks derived from either tree decomposition of molecular graphs or molecular rationales explaining the outcome of property predictors. Lastly, we demonstrate how these techniques managed to discover novel antibiotics and COVID-19 antiviral drug combinations. These discoveries highlight the significant impact that deep learning can have on drug discovery by decreasing its time and cost.

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

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

First of all, I would like to thank my thesis supervisor Prof. Regina Barzilay and Prof. Tommi Jaakkola. They have helped me so much and I cannot finish this thesis without their help. I would like to thank Prof. Klavs Jensen for being my thesis committee member. I would like to thank Prof. William Green, Prof. Phillip A. Sharp, Prof. James Collins, Prof. Rafael Gomez-Bombarelli, and Prof. Connor Coley. They have been a wonderful collaborator for many chemistry projects. I also want to thank Jonathan Stokes, Kevin Yang, and Kyle Swanson for their help in the antibiotic project. I want to thank Alexey Zakharov from NCATS for running COVID-19 assays. I want to thank all the colleagues in Regina’s group, Tommi’s group, and Klavs’ group for helping me on various chemistry experiments. This thesis is sponsored by the MIT MLPDS Consortium, DARPA AMD Program, DARPA Make-it Program, MIT J-Clinic, and NCATS.

Lastly, I am grateful to my mother for her sacrificial love and her support of my study. Her support and love has always been my motivation to overcome many difficulties in my life. I also want to thank God and my church family including P. Joseph Han, P. John Peng, Jonghoon Jeong, Yue Gao, Zhenghua Wu, Ce Liu, Hu Huang, Jiaxing Zhang, Donglai Wei, Wenjie Lu, Peilun Dai, Xun Peng, and Jihang Chen for their love and caring for me.
Contents

1 Introduction 19

2 Molecular Property Prediction 25
   2.1 Chemprop Model Architecture ................................. 26
   2.2 Generalization challenge of property predictors .................. 27
   2.3 Counterfactual Consistency Regularization ....................... 28
      2.3.1 Implementing the regularizer ............................... 30
   2.4 Experiments .................................................. 35
      2.4.1 QM9 ........................................................ 36
      2.4.2 Merck molecular activity challenge ......................... 36
      2.4.3 Ablation studies and analysis .............................. 38
   2.5 Summary ...................................................... 40

3 Molecular Graph Generation 41
   3.1 Hierarchical Graph Generation .................................. 43
      3.1.1 Motif Extraction ......................................... 45
      3.1.2 Hierarchical Graph Encoder ............................... 46
      3.1.3 Hierarchical Graph Decoder ............................... 47
      3.1.4 Extension to Graph-to-Graph Translation .................... 50
   3.2 Experiments .................................................. 51
      3.2.1 Polymer Generative Modeling ............................... 51
      3.2.2 Graph-to-Graph Translation ............................... 54
   3.3 Summary ...................................................... 58
4 Multi-objective Molecule Generation

4.1 Proposed Approach: RationaleRL

4.1.1 Rationale Extraction from Predictive Models

4.1.2 Graph Completion

4.1.3 Training Procedure

4.2 Experiments

4.2.1 Results

4.2.2 Property Predictor Applicability

4.2.3 Faithfulness of Rationales

4.3 Summary

5 Application: Antibiotic Discovery

5.1 Identification of halicin via Chemprop

5.1.1 Halicin displays efficacy in murine models of infection

5.2 Summary

6 Application: COVID Drug Discovery

6.1 ComboNet Architecture

6.2 Model evaluation

6.3 Screening predicted drug combinations

6.4 Summary

7 Conclusion
List of Figures

1-1  My contribution to ML-based drug discovery includes both algorithmic innovations and scientific discoveries. On the algorithmic side, I have developed (1) novel training algorithms for learning generalizable property predictors; (2) new generative models for molecular graph generation. On the scientific side, my work has discovered (3) new antibiotics for a wide range of pathogens and (4) novel synergistic drug combinations for COVID-19. ......................................................... 21

2-1  Spurious correlations observed in a HIV protease inhibition assay from Merck [64]. \( E = 0 \) (or \( E = 1 \)) means compounds assayed before (or after) a certain date. (a) A t-SNE plot of molecule distribution \( P(X) \) under different \( E \). (b) The correlation between a molecular feature and HIV inhibition \( Y \) under different \( E \). (c) Our assumption of the data generation process. ................................................................. 28

2-2  Illustration of the forward pass with CFC regularization, which contains four components: 1) standard ERM objective \( \mathcal{L}(f \circ \phi) \); 2) VAE training loss \( \mathcal{L}_{\text{VAE}}(\psi_e \circ \xi_e) \); 3) factual loss \( \mathcal{L}^e(f_e \circ \phi) \); 4) counterfactual loss \( \mathcal{L}^e(f_{e'} \circ g_{e,e'} \circ \phi) \). During back-propagation, the gradient from the VAE loss is not propagated to \( \phi \). The gradient from \( \mathcal{L}^e(f_e \circ \phi) \) goes through a gradient reversal layer. The gradient from \( \mathcal{L}^e(f_{e'} \circ g_{e,e'} \circ \phi) \) is only used to update \( \phi \). ................................................................. 33
2-3 (a) Comparison between CFC and CFC\textsubscript{cyc} using VAE and CycleGAN respectively. (b) Model performance under different CFC regularization weight $\lambda$ on the QM9 dataset. ........................................ 39

2-4 T-SNE visualization of learned representations in the Merck dataset. Blue dots are the factual representations $\phi(x)$ of the training molecules in the first environment from earlier assays. Orange dots correspond to their counterfactual representations $g_{1,2}(\phi(x))$ after translation. Green dots are the representations $\phi(x)$ of the test molecules from later assays. 40

3-1 Two almost identical molecules with markedly different canonical SMILES in RDKit. The edit distance between two strings is 22 (50.5\% of the whole sequence). .................................................. 42

3-2 Left: Illustration of structural motifs in polymers. Right: Reconstruction accuracy for polymers with various sizes (number of atoms). Notably, the atom-based generative model CG-VAE [61] fails to reconstruct molecules over 80 atoms. In contrast, the proposed model maintains high accuracy for large molecules by utilizing motifs as building blocks for generation (red curve). ............................... 43

3-3 Hierarchical graph encoder. Dashed arrows connect each atom to the motifs it belongs. In the attachment layer, each node $\mathcal{A}_i$ is a particular attachment configuration of motif $\mathcal{S}_i$. The atoms in the intersection between each motif and its neighbors are highlighted in faded block. .......................... 46

3-4 Hierarchical graph decoder. In each step, the decoder first runs hierarchical message passing to compute motif, attachment and atom vectors. Then it performs motif and attachment prediction for the next motif node. Finally, it decides how the new motif should be attached to the current graph via graph prediction. ........................................ 48

3-5 .................................................. 53
4-1 Illustration of RationaleRL. Left: To generate a dual inhibitor against biological targets GSK3/β and JNK3, our model first identifies rationale substructures $S$ for each property. Note that rationales are not provided as domain knowledge. Middle: The model learns to compose multiple rationales $S$ into a complete molecule $G$. Right: Our method achieves much higher success rate than the current state-of-the-art molecule design method REINVENT [71]) under four property constraints.

4-2 Overview of our approach. We first construct rationales for each individual property and then combine them as multi-property rationales. The method learns a graph completion model $P(G|S)$ and rationale distribution $P(S)$ in order to generate positive molecules.

4-3 Illustration of Monte Carlo tree search for molecules. Peripheral bonds and rings are highlighted in red. In the forward pass, the model deletes a peripheral bond or ring from each state which has maximum $Q + U$ value (see Eq.(4.6)). In the backward pass, the model updates the statistics of each state.

4-4 Illustration of multi-property rationale construction. Given two single-property rationales, we first find their maximum common substructure (MCS). If their MCS is not empty, we superpose one rationale on another so that their MCS coincides.

4-5 Sample rationales of GSK3/β (top) and JNK3 (bottom).

4-6 Left & middle: t-SNE plot of the extracted rationales for GSK3/β and JNK3. For both properties, rationales mostly covers the chemical space populated by existing positive molecules. Right: t-SNE plot of generated GSK3/β+JNK3 dual inhibitors.
4-7  **Left:** Examples of molecules generated in the GSK3β+JNK3+QED+SA task. The model learns to combine two disjoint rationale graphs into a complete molecule.  **Right:** Example structural alerts in the toxicity dataset. The ground truth rationale (Azobenzene) is highlighted in red. Our learned rationale almost matches the ground truth (error highlighted in dashed circle).

5-1  Machine learning in antibiotic discovery. We first train our Chemprop model using a collection of a few thousand diverse molecules for those that inhibited the growth of *E. coli*. Next, we apply the resulting model to multiple discrete chemical libraries, including a collection of >107 million molecules, to identify potential lead compounds with activity against *E. coli*. After ranking the candidates according to the model’s predicted score, we select a list of promising candidates based on a pre-defined threshold.
5-2 Initial model training and the identification of halicin. (A) Primary screening data for growth inhibition of *E. coli* by 2,560 molecules. (B) ROC-AUC plot evaluating model performance after training. Dark blue is the mean of six individual trials (cyan). (C) Rank-ordered prediction scores of Broad Repurposing Hub molecules that were not present in the training dataset. (D) The top 99 predictions from the data shown in (C) were curated for empirical testing for growth inhibition of *E. coli*. Fifty-one of the 99 molecules were validated as true positives. (E) For all molecules shown in (D), ratios of OD600 to prediction score were calculated and these values were plotted based on prediction score for each corresponding molecule. This shows that a higher prediction score generally correlates with a greater probability of growth inhibition. (F) The bottom 63 predictions from the data shown in (C) were curated for empirical testing for growth inhibition of *E. coli*. Two of the 63 molecules tested as false negatives. Red are growth inhibitory molecules; blue are non-growth inhibitory molecules. (G) t-SNE of all molecules from the training dataset (blue) and the Broad Repurposing Hub (red). (H) Tanimoto similarity between halicin and each molecule in the de-duplicated training dataset. (I) Growth inhibition of *E. coli* by halicin. Shown is the mean of two biological replicates. Bars denote absolute error. . . . . . . . . . . . . 82
5-3 Halicin displays efficacy in murine models of infection. (A) Growth inhibition of pan-resistant *A. baumannii* CDC288 by halicin. Shown is the mean of two biological replicates. (B) Killing of *A. baumannii* CDC288 in the presence of varying concentrations of halicin after 2 hr (blue), 4 hr (cyan), 6 hr (green), and 8 hr (red). (C) In a wound infection model, mice were infected with *A. baumannii* CDC288 for 1 hr and treated with either vehicle or halicin periodically over 24 hr. Black lines represent the geometric mean of the bacterial load for each treatment group. (D) Growth inhibition of *C. difficile* 630 by halicin. (E) Experimental design for *C. difficile* infection and treatment. (F) Bacterial load of *C. difficile* 630 in feces of infected mice. Metronidazole did not result in enhanced rates of clearance relative to vehicle controls. Halicin-treated mice displayed sterilization beginning at 72 hr after treatment, with 100% of mice being free of infection at 96 hr.

6-1 **ComboNet for synergistic drug combination discovery**

**a)** ComboNet is composed of two networks: a drug-target interaction (DTI) and target-disease association network. The antiviral effect of a single drug $p_A$ is predicted from its representation $z_A$. The vector $z_A$ characterizes the drug-target interaction features of drug A. **b)** The antiviral effect of a combination is predicted from its representation $z_{AB}$, which is computed from the molecular representations of each individual drug $z_A, z_B$. ComboNet is trained on drug combination synergy, single-drug antiviral activity and drug-target interaction data.
6-2  **In silico evaluation of ComboNet**  

**a)** The training, validation, and test set composition for SARS-CoV-2.  

**b)** Results on SARS-CoV-2 drug combination test set. Our full ComboNet model outperforms all other baselines.  

**c)** ROC-AUC plot of ComboNet ensemble on the entire test set.  

**d)** ROC-AUC plot of ComboNet ensemble on the hard drug combinations with at least one new drug.  

**e)** Statistical characteristics of ComboNet ensemble for all the datasets, where “screen” refers to the top 30 candidates we experimentally tested.  

6-3  **Discovery of novel synergistic drug combinations.**  

**a)** Two new drug combinations are discovered by our model: Remdesivir + Reserpine and Remdesivir + IQ-1S.  

**b)** Host cell viability matrices show the two drug combinations have low cytotoxicity.  

**c)** Dose response and bliss synergy matrices of Remdesivir + Reserpine. Numbers in the dose response matrix stands for viral infection rate. Numbers in the bliss synergy matrix stands for synergy score. Both are the lower the better.  

**d)** Dose response and bliss synergy matrices of Remdesivir + IQ-1S.  

**e)** The correlation between predicted ranking and DBSumNeg score (lower DBSumNeg means more synergistic).  

**f)** t-SNE visualization [65] of the chemical space explored across the training set, test set, and experimentally validated combinations.
# List of Tables

2.1 Results on the QM9 dataset. Each cell reports the MAE metric ($\downarrow$). We normalize each property by its standard deviation before MAE calculation since their scales are very different. The standard deviation of the final MAE is calculated across two independent runs with different random seeds.

37

2.2 Results on the Merck dataset. We evaluate each model under a temporal split. Each cell reports the $R^2$ metric (coefficient of determination) and it is the higher the better ($\uparrow$).

38

3.1 Results on polymer generative modeling. The first row reports the oracle performance using real data as generated samples. The last row (small motif) is a variant of our model where we restrict the motif vocabulary to contain only single rings and bonds (similar to JT-VAE). SNN means nearest neighbor similarity; “Frag / Scaf” means fragment and scaffold similarity. Except property statistics, all metrics are the higher the better.

52

3.2 Results on graph translation tasks from Jin et al. [42]. We report average improvement for continuous properties (logP), and success rate for binary properties (e.g., DRD2).

55

3.3 Ablation study: the importance of hierarchical graph encoding, LSTM MPN architecture and structure-based decoding.

56

4.1 Results on molecule design with one property constraint.

72
4.2 Molecule design with multiple property constraints. The novelty and diversity of JT-VAE, GVAE-RL and GCPN are not reported due to their low success rate.

4.3 Frechet ChemNet Distance (FCD) between generated compounds and true positive molecules in the training set.

4.4 Rationale accuracy on the toxicity dataset. Our rationales are more faithful to the property of interest.
Chapter 1

Introduction

The process of finding a novel drug candidate typically consists of two stages. The first step is called molecule screening, where chemists put a library of existing compounds into a biological assay to measure their biological properties, such as potency, toxicity, and solubility. Compounds with desired properties are selected as “lead compounds” after screening is finished. The second step is called lead optimization. The goal is to modify the structure of a lead compound to optimize its property profiles. This step is necessary because a lead compound is usually imperfect and does not satisfy all the desired property constraints. After a lead compound is optimized, a chemist will test it in animal models or human trials to validate its biological properties. The overall goal is to maximize the success rate in this final experimental validation step.

Drug discovery is a costly and time-consuming process that takes 10-15 years and $1.3 billion on average. For molecule screening, the major bottleneck is the large search space of drug candidates. The size of the chemical space is estimated to be at least $10^{60}$ [51]. In contrast, standard high-throughput screening facilities in the pharmaceutical industry can only test around $10^5$ compounds per day. As a result, we have to restrict the size of the compound library to make the screening time and cost feasible. On the other hand, medicinal chemists currently perform lead optimization manually based on trial and error. However, manual exploration of the chemical space is very inefficient because drug discovery involves many property constraints. Moreover, the relationship between molecular structures and biological properties is
much more complex than simple rules used by medicinal chemists [64].

We seek to accelerate and automate drug discovery using machine learning (ML). Previous molecule screening efforts in the pharmaceutical industry have generated many datasets of molecules with labeled properties. This allows us to build **molecular property prediction** models to predict the properties of a compound without testing it in a wet lab. We can then use these models to *virtually* screen a much larger collection of molecules at a much faster speed ($10^8$ compounds/day) than current high-throughput screening facilities in a wet lab. The same dataset also enables us to learn **molecular graph generation** models to inverse design a molecular structure that satisfies property constraints (e.g., high potency and low toxicity). These generative models can be used to automate lead optimization currently done manually in the pharmaceutical industry.

Indeed, ML-based drug discovery involves a lot of challenges. Predicting molecular properties is difficult because biological datasets are small (1000-10000 compounds), and molecules in these biological assays are often biased toward certain parts of the chemical space. Therefore, models trained under standard empirical risk minimization (ERM) cannot generalize to new chemical spaces at test time. This challenge is further magnified when predicting the synergy of drug combinations, where the dataset is even smaller (100-200 combinations) as combination assays are much more complicated and time-consuming. On the other hand, the generation of molecules is challenging due to the combinatorial nature of compounds. Molecules are represented as graphs, whose nodes and edges correspond to atoms and bonds. While standard deep learning architectures are good at generating regular data structures like images and text, generating graphs is more difficult due to their irregular and heterogeneous structure. Drug discovery thus requires new deep learning architectures tailored for graph generation.

Over the past few years, there has been a surge of interest in the development and application of machine learning methods to problems in medicinal chemistry, from molecular property prediction [24, 48, 28, 55, 104], chemical reaction modeling [19, 39, 85], to molecule generation [83, 52, 33, 30, 47, 74, 112]. Most of the existing approaches
are direct application of standard deep learning architectures used in computer vision and natural language processing. For example, prior approaches formulated molecule generation as a string generation problem [29, 52]. These models cast molecules as SMILES strings [98] and used standard recurrent neural networks (RNNs) to generate molecules. However, SMILES string grammar is complex, and standard RNNs fail to generate valid SMILES strings [52]. For molecular property prediction, property predictors are de facto trained by ERM, which fail to generalize to new chemical spaces [64].

In this thesis, I present novel deep learning architectures that address the limitations of previous works by leveraging the inductive bias of molecular structures and modeling the spurious correlations in biological assays. My contribution to ML-based drug discovery involves both algorithmic innovations and scientific discoveries (Figure 1-1), including new training algorithms for generalizable property predictors (Chapter 2), new generative models for molecular graphs (Chapter 3 and 4), the
discovery of novel antibiotics (Chapter 5), and discovery of novel synergistic drug combinations for COVID-19 (Chapter 6).

Chapter 2 proposes a new domain generalization method for property prediction called counterfactual consistency regularization [45]. It seeks to learn a model whose predictions are not affected by interventions on the confounding variables, such as the choice of chemical libraries, batch effects, and measurement biases. We quantify consistency as the difference between two predictions, one from the original compound combined with an environment-specific predictor, and the other based on its counterfactual representation that effectively transports the compound to a different environment. The proposed method significantly outperforms state-of-the-art baselines across multiple property prediction benchmarks.

In Chapter 3, I develop a hierarchical graph variational autoencoder (VAE) for molecular graph generation [43]. The proposed generative model learns to construct molecules motif by motif in an autoregressive manner. The approach is motivated by the low tree-width structure of molecular graphs, analogous to the junction tree algorithm in graphic model inference. By representing molecular graphs in terms of motifs, we reduce the graph generation problem to a simpler tree generation problem. As a result, the method can generate 100% valid molecules, which significantly outperforms prior SMILES-based methods.

Chapter 4 introduces a hierarchical reinforcement learning algorithm to generate molecules under multiple objectives [44]. Compared to hierarchical graph VAE, which relies on a pre-defined set of motifs, this method automatically learns to discover motifs tailored for different properties. These motifs are found by interpretation of molecular property predictors using a novel graph attribution method. These biology-aware motifs are crucial to multi-objective drug design since the solution space decreases exponentially as the number of constraints increases. Indeed, our method outperforms previous state-of-the-art by more than 30% on a multi-objective drug design benchmark.

In Chapter 5, I demonstrate how deep learning can automatically discover novel antibiotics [90]. Specifically, we discovered a novel compound called Halicin that
showed strong efficacy against multiple resistant bacteria in mice. As a result, Halicin is now listed as an investigational new drug by the U.S. Food and Drug Administration (FDA) [99]. This discovery is significant because no clinical antibiotics have been discovered using traditional high-throughput screening since 1987 [88], and there are many bacteria species resistant to all clinical antibiotics.

In Chapter 6, we develop a new property prediction architecture called ComboNet and discover novel drug combinations with strong synergy in vitro: Reserpine + Remdesivir and Remdesivir + IQ-1S. ComboNet jointly models molecular structure and biological targets to predict synergistic drug combinations. We hypothesize that we can significantly decrease the dependence on combination synergy data by explicitly modeling interactions between drugs and biological targets, which is rare for novel pathogens in practice.

In summary, my thesis highlights the significant impact that deep learning can have on drug discovery by reducing the time and cost. For example, my property prediction tools can screen more than $10^8$ compounds per day, which is 1000 times faster than standard high-throughput screening in the industry. The molecular property prediction tools and hierarchical graph generative models are being used in many pharmaceutical companies, including Amgen, AstraZeneca, BASF, Bayer, GSK, Janssen, Novartis, Merck, and Pfizer.
Chapter 2

Molecular Property Prediction

Molecular property prediction, one of the oldest cheminformatics tasks, has received new attention in light of recent advancements in deep neural networks. These architectures either operate over fixed molecular fingerprints common in traditional QSAR models, or they learn their own task-specific representations using graph convolutions [24, 103, 48, 28, 59, 50, 23, 13, 18, 82, 7]. Both approaches are reported to yield substantial performance gains, raising state-of-the-art accuracy in property prediction.

Despite these successes, many questions remain unanswered. The first question concerns the evaluation setup. Wu et al. [103] show that neural models typically outperform traditional methods, while experiments reported in Mayr et al. [66] report the opposite. Part of these discrepancies can be attributed to differences in evaluation setup, including the way datasets are constructed. We need proper evaluation protocols to measure the generalization power of a method when applied to a new chemical space, as is common in drug discovery. This leads us to the second question on learning generalizable property predictors. In molecular property prediction, training data is usually limited or otherwise biased. Molecular assays used for learning property predictors involve many sources of spurious correlations, from the choice of chemical libraries or batch effects, to measurement biases. Effective molecular property prediction therefore requires that models generalize beyond the chemical space of training examples and avoid learning spurious correlations introduced by these biases.
In this chapter, we aim to answer both of these questions by designing a comprehensive evaluation setup and a novel domain generalization algorithm for property prediction. We first develop a software package called Chemprop [107] that achieves new state-of-the-art performance across a range of datasets. Based on the Chemprop software, we propose counterfactual consistency regularization to help models extrapolate to new chemical spaces. It seeks to learn a model whose predictions are not affected by interventions on the confounding variables, such as the choice of chemical libraries, batch effects, and measurement biases. Specifically, we quantify consistency as the difference between two predictions, one from the original compound combined with an environment-specific predictor, and the other based on its counterfactual representation that effectively transports the compound to a different environment. The proposed method significantly outperforms state-of-the-art baselines across multiple property prediction benchmarks.

2.1 Chemprop Model Architecture

A molecule $x$ is be represented as a labeled graph $G_x$ whose nodes are the atoms in the molecule and edges are the bonds between the atoms. Each node $v$ has a feature vector $f_v$ including its atom type, valence, and other atomic properties. Each edge $(u, v)$ is also associated with a feature vector $f_{uv}$ indicating its bond type.

Chemprop builds on the message passing neural network (MPNN) framework presented in Gilmer et al. [28]. A MPNN $\phi$ learns to embed a graph $G_x$ into a continuous vector $\phi(x)$. Chemprop adopt the MPNN architecture from Dai et al. [20], which associates hidden states $h_v$ with each node $v$ and updates these states by passing messages $m_{uv}$ over edges $(u, v)$. Each message $m_{uv}^{(0)}$ is initialized at zero. At time step $t$, the messages are updated as follows:

$$m_{uv}^{(t+1)} = \text{MLP}(f_u, f_{uv}, \sum_{w \in N(u), w \neq v} m_{wu}^{(t)}) \quad \forall (u, v) \in G_x$$

where $N(u)$ is the set of neighbor nodes of $u$ and MLP stands for a multilayer per-
ceptron. After $T$ message passing steps, we compute hidden states $h_u$ as,

$$h_u = \text{MLP}(f_u, \sum_{v \in N(u)} m^{(T)}_{uv}),$$

(2.2)

and we compute the representation $\phi(x) = \sum_{u \in G_x} h_u$. Finally, we predict the property $y$ through another MLP $f$:

$$f(G_x) = \text{MLP}(\phi(x)) = \text{MLP}(\sum_{u \in G_x} h_u).$$

(2.3)

### 2.2 Generalization challenge of property predictors

In pharmaceutical industry, property predictions models are de facto evaluated under a temporal split, where the training and test set contain molecules assayed before and after a certain year. Temporal split seeks to simulate real-world property prediction setting, where one trains a model on past data to make predictions on future data. In quantum chemistry, models are required to extrapolate to new chemical spaces, e.g., molecules with more heavy atoms [16, 95]. Formally, we assume our training set is a list of triplets $\{(X, Y, E)\}$, where $X$ is a molecule, $Y$ represents its property, and $E$ is the environment from which $X$ is constructed. For instance, $E$ can be the year when a compound $X$ is assayed or the number of heavy atoms of $X$.

In both cases, the major challenge is that each environment is biased towards certain types of molecules and the training and test distribution can be quite different [64]. For example, the assay date $E$ is an indicator of many underlying factors that cause spurious correlations between $X$ and $Y$. Specifically, $E$ can influence the distribution of $X$ due to typical selection biases such as a chemist favoring larger molecules. $E$ may also correlate with $Y$ if known inactive compounds are discarded from later assays. The confounding effect of $E$ can easily misdirect models to adopt wrong conclusions such as “larger molecules are more bioactive”. One can find similar confounders in many datasets. For example, in a HIV protease inhibition assay from Merck [64], both $P(X)$ and $P(Y|X)$ vary as a function of $E$ (Figure 2-1a-b). The problem is challenging since confounders violate simplifying assumptions such as
Figure 2-1: Spurious correlations observed in a HIV protease inhibition assay from Merck [64]. \( E = 0 \) (or \( E = 1 \)) means compounds assayed before (or after) a certain date. (a) A t-SNE plot of molecule distribution \( P(X) \) under different \( E \). (b) The correlation between a molecular feature and HIV inhibition \( Y \) under different \( E \). (c) Our assumption of the data generation process.

covariate shift typically adopted in domain adaptation [8, 26].

To generalize to new distributions, a predictor should pay no attention to the spurious correlations caused by the confounder \( E \). However, standard empirical risk minimization (ERM) is prone to learning spurious correlations in the data rather than true causal relationship between a molecular structure and its property. To address this issue, we propose counterfactual consistency regularization, which contains a predictor \( \hat{Y} \) in a manner that removes the impact from the confounder \( E \).

### 2.3 Counterfactual Consistency Regularization

**Causal graph.** To introduce our method, we first state our assumption on the data generation process. We assume all the distributions that may be observed over \((X,Y)\) arise from the causal graph in Figure 2-1c. The underlying structural equations are

\[
X = F_x(U, E), \quad Z = F_z(X), \quad Y = F_y(Z, E)
\]  

(2.4)

\( U \) is a latent exogenous variable which renders the molecular structure. \( Z \) is a feature representation of \( X \) that is either manually defined by a chemist or learned automatically by a model. Confounder \( E \) may influence \( X \) through a mechanism involving a chemist (e.g., selection of a particular chemical library). \( Y \) may also depend on \( E \), for
example, due to \textit{batch effects} [100] — laboratory conditions may change from one year to another and therefore affect measured properties. Structural equations $F_x, F_z, F_y$ are usually unknown and need to be inferred from data. In practice, we realize $F_z, F_y$ using neural networks and leave $F_x$ unspecified since we can only observe $X$, not the associated $U$.

\textbf{A motivating example.} To motivate our method, let us consider the following example. Suppose $Y$ represents the binding affinity of compound $X$ to a target protein. In this case, we can interpret $U$ as the pharmacophore of $X$, i.e., a substructure of $X$ responsible for binding [101]. The environment $E$ specifies a particular binding assay. Thus, $E$ influences how $X$ with a pharmacophore $U$ is realized based on the chosen chemical library. $E$ may also impact $Y$ directly based on assay conditions. Now, consider the following counterfactual question: \textit{“Given an outcome $Y$ in response to $X$ under assay $E = e$, what would the outcome be if $E$ had been set differently $E = e'$?”} To answer this question, we envision a new molecule $X'$ that contains the same pharmacophore $U$, modified in a manner suitable for inclusion in assay $e'$, and measurement performed under the conditions set by $E = e'$. In order to guide the model to rely primarily on the underlying pharmacophores, we require that the answer to the counterfactual question remains the same.

Formally, we define counterfactual outcome using notation $Y_{E \leftarrow e}(U)$. Intervention $E \leftarrow e$ means that we replace a random variable $E$ with a value $e$ in context $U$. The remaining endogenous variables $X, Z$ and $Y$ will be affected by changes in $e$ while $U$ remains as inferred from the factual observation. In other words, the intervened structural equations become

$$
X_{E \leftarrow e} = F_x(U,e), \quad Z_{E \leftarrow e} = F_z(X_{E \leftarrow e}), \quad Y_{E \leftarrow e} = F_y(Z_{E \leftarrow e}, e) \quad (2.5)
$$

We keep the latent variable $U$ explicit in the notation $Y_{E \leftarrow e}(U)$ to illustrate that $U$ remains fixed during the intervention. The inductive bias we introduce is the requirement that in our estimated model $\forall e, e' \in \mathcal{E} : \hat{Y}_{E \leftarrow e}(U) = \hat{Y}_{E \leftarrow e'}(U)$ (\mathcal{E} is the
set of all training environments). It emphasizes the causative role of the underlying pharmacophore $U$. This intuition leads us to the following consistency constraint.

**Counterfactual consistency (CFC).** For every training example $(x, y, e)$ and $e' \neq e$, we impose the following condition denoted as $\text{CFC}(x, y, e, e')$

$$P(\hat{Y}_{E\leftarrow e}(U) = y | X = x, E = e) = P(\hat{Y}_{E\leftarrow e'}(U) = y | X = x, E = e)$$ \hspace{1cm} (2.6)

where $\hat{Y}_{E\leftarrow e}(U)$ is the prediction of the counterfactual outcome $Y_{E\leftarrow e}(U)$ realized by our model. In other words, a predictor $\hat{Y}$ is counterfactually consistent when $E$ no longer has a causal effect on $\hat{Y}$. We note that counterfactual consistency is imposed per observation and depends on the causal model.

**CFC regularization.** In practice, the representation $Z = \phi(X)$ and output $\hat{Y} = f(Z)$ are parameterized by neural networks $f$ and $\phi$. $f$ is trained on all environments since it will be used to make predictions for compounds in new test environments. Given a training set $\mathcal{D}$, we want to find a counterfactually consistent model by solving the following constrained problem:

$$\min_{f, \phi} \mathcal{L}(f \circ \phi) \quad \text{s.t.} \ \forall (x, y, e) \in \mathcal{D}, \forall e' \neq e : \text{CFC}(x, y, e, e') = \text{true}$$ \hspace{1cm} (2.7)

where $\mathcal{L}(f \circ \phi) = \sum_{(x, y, e) \in \mathcal{D}} \ell(y, f(\phi(x)))$ is the empirical loss of the training data. To implement the regularizer, we need to estimate $\hat{Y}_{E\leftarrow e}(U)$ and $\hat{Y}_{E\leftarrow e'}(U)$ properly so that they are consistent with the causal graph.

### 2.3.1 Implementing the regularizer

In theory, a model is counterfactually consistent if $\hat{Y}$ is only a function of $U$. However, implementing CFC is challenging in practice because $U$ is a hidden variable. We cannot infer $U$ from $X$ directly without knowing the structural equation $F_x$. Though we can parameterize $F_x$ as a generative model of molecules [30, 84, 44], it introduces too much overhead for a prediction problem because molecule generation is compu-
tationally expensive. The key challenge is how to efficiently estimate the factual and counterfactual losses for $\hat{Y}_{E \leftarrow e}(U)$ and $\hat{Y}_{E \leftarrow e'}(U)$ without estimating $F_x$ explicitly.

**Factual loss for $\hat{Y}_{E \leftarrow e}(U)$.** As shown in Eq. (2.5), the intervention $E \leftarrow e$ does not change the value of $X$ because it is constructed from environment $e$, i.e. $X_{E \leftarrow e} = F_x(U, e) = X$. This means we can simply model $Z_{E \leftarrow e} = \phi(X)$. In contrast, the output $Y$ is affected by the intervention because $E$ is a causal parent of $Y$. We cannot model $F_y(Z_{E \leftarrow e}, e) = f(\phi(X))$ because $f$ is shared across all the environments and the intervention would have no effect on $Y$. Given these observations, we introduce an environment specific predictor $f_e$ and model the factual loss as

$$-\log P(\hat{Y}_{E \leftarrow e}(U) = y | X = x, E = e) = \ell(y, f_e(\phi(x)))$$  \hspace{1cm} (2.8)$$

Here $f_e$ is defined as the minimizer of the training loss $L^e(h \circ \phi)$ of environment $e$:

$$f_e = \arg \min_h L^e(h \circ \phi) = \arg \min_h \sum_{(x,y) \in D_e} \ell(y, h(\phi(x)))$$  \hspace{1cm} (2.9)$$

where $D_e$ is the set of all training examples $(x,y) \in D$ from environment $e$.

**Counterfactual loss for $\hat{Y}_{E \leftarrow e'}(U)$.** In contrast to the factual term, the intervention $E \leftarrow e'$ affects the value of $X$ since $F_x(U, e) \neq F_x(U, e')$ in general. We cannot directly construct a counterfactual example $X_{E \leftarrow e'} = F_x(U, e')$ without knowing the structural equation $F_x$. Although it is possible to learn a function that translates $X$ to its counterfactual molecule $X_{E \leftarrow e'}$ without knowing $F_x$, it still requires the model to generate discrete molecules [42, 43], which is quite time-consuming.

For computational efficiency, we propose to modify the molecule in the representation space instead. Specifically, we learn a translation model $g_{e,e'}$ to map the original representation $z = \phi(x)$ into its counterfactual representation $z_{E \leftarrow e'} = g_{e,e'}(z)$, which corresponds to $F_z(F_x(U, e'))$ in the intervened causal model. The counterfactual loss
is then modeled as

$$-\log P(\hat{Y}_{E\leftarrow e'}(U) = y|X = x, E = e) = \ell(y, f_{e'}(z_{E\leftarrow e'})) = \ell(y, f_{e'}(g_{e,e'}(z))) \quad (2.10)$$

The impact of the intervention is modeled by changing the predictor to $f_{e'}$ and the representation $z = \phi(x)$ into its counterfactual version $z_{E\leftarrow e'}$, depending on the choice of $g_{e,e'}$ (Figure 2-2b).

The form that $g_{e,e'}$ should take depends on the information captured in the representation $Z$. If $Z$ happens to fully recover only the latent variable $U$, then we could set $z_{E\leftarrow e'} = z$ because $U$ is treated as an environment-independent pharmacophore. In other words, we could in this case set $g_{e,e'}$ to be an identity function $I(z) = z$. However, we find that using the identity translation $g_{e,e'} = I$ yields suboptimal performance in our experiments. Indeed, recent work [80] has shown that when the model is highly nonlinear, it is hard to directly learn an environment-invariant representation $Z = U$ without additional guidance.

When $Z$ contains environment-specific information, a translation function $g_{e,e'}$ needs to properly capture the changes when transporting $z$ from environment $e$ to $e'$. Motivated by recent work in multi-domain translation [21], we can learn $g_{e,e'}$ in an unsupervised way. For each environment $e$, we first train a VAE $\psi_e \circ \xi_e$ which minimizes the loss (Figure 2-2a)

$$\mathcal{L}_{\text{VAE}}(\psi_e \circ \xi_e) = \mathbb{E}_{(x,y) \in \mathcal{D}_e} [||\phi(x) - \psi_e(\xi_e(\phi(x)))||^2 + \beta \ell_{\text{KL}}(\xi_e(\phi(x))))] \quad (2.11)$$

where $\xi_e$ and $\psi_e$ are the encoder and decoder. $\beta$ is a hyperparameter controlling the KL divergence (the KL divergence omits an explicit reference to the prior for simplicity). The reconstruction loss enforces $\psi_e(\xi_e(z)) \approx z$ for every $z$. We then parameterize the translation model as $g_{e,e'} = \psi_{e'} \circ \xi_e$. It learns to translate a representation $z$ from environment $e$ to $e'$ in a manner that preserves the content $U$ while still accounting for the remaining environment dependence.
Given the above formulation, one can naturally define the CFC regularizer as \( \ell(y, f_{e'}(g_{e,e'}(z))) - \ell(y, f_e(z)) \) (cf. Eq. (2.8) & (2.10)). Note that in the factual loss \( \ell(y, f_e(z)) \), predictor \( f_e \) has already been trained on the true label for \( z \). In contrast, \( f_{e'} \) and \( g_{e,e'} \) in the counterfactual loss have not seen this \((z,y)\), and \( f_{e'} \) has been adjusted towards a different environment \( e' \). Therefore, the factual loss is typically lower than the counterfactual loss in practice. To this end, we propose the following simpler objective involving a CFC regularizer that is easier to optimize:

\[
\mathcal{L}_{CFC} = \mathcal{L}(f \circ \phi) + \lambda \sum_{(x,y,e) \in \mathcal{D}} \sum_{e' \neq e} \ell(y, f_{e'}(g_{e,e'}(z))) - \ell(y, f_e(z))
\]

\[
= \mathcal{L}(f \circ \phi) + \lambda \sum_{e} \sum_{e' \neq e} \sum_{(x,y) \in \mathcal{D}_e} \ell(y, f_{e'}(g_{e,e'}(z))) - \ell(y, f_e(z))
\]

\[
= \mathcal{L}(f \circ \phi) + \lambda \sum_{e} \sum_{e' \neq e} \mathcal{L}^e(f_{e'} \circ g_{e,e'} \circ \phi) - \mathcal{L}^e(f_e \circ \phi) \tag{2.14}
\]

The regularizer \( \mathcal{R}_{e,e'}(\phi) \) measures the performance gap between \( f_{e'} \circ g_{e,e'} \) and the environment-specific predictor \( f_e \). Moreover, we find it helpful to use multiple translation functions \( g \in \mathcal{G}_{e,e'} = \{g_{e,e'}, I\} \) in the regularizer to take into account the two
aforementioned types of \(Z\), i.e.

\[
R_{e,e'}(\phi) = \sum_{g \in G_{e,e'}} \mathcal{L}^e(f_{e'} \circ g \circ \phi) - \mathcal{L}^e(f_e \circ \phi)
\]  

(2.15)

Including the identity function encourages the feature extractor \(\phi\) to reduce environment-specific information in \(z\). The VAE translation function \(g_{e,e'}\) then highlights the remaining environment-specific information through the counterfactual representation.

**Optimization** The CFC objective \(\mathcal{L}_{\text{CFC}}\) can be viewed as finding a stationary point of a multiplayer game between \(f, \phi\), the auxiliary predictors \(\{f_e\}\), and the translation models \(\{g_{e,e'}\}\). The label predictor \(f\) and feature extractor \(\phi\) minimize

\[
\min_{f,\phi} \mathcal{L}(f \circ \phi) + \lambda \sum_{e} \sum_{e' \neq e} R_{e,e'}(\phi)
\]

(2.16)

while the auxiliary predictors minimize

\[
\min_{f_e} \mathcal{L}^e(f_e \circ \phi) \quad \text{and} \quad \min_{\xi_e, \psi_e} \mathcal{L}_{\text{VAE}}(\psi_e \circ \xi_e)
\]

(2.17)

This multiplayer game can be optimized by stochastic gradient descent. To be specific, we simultaneously update all players with learning rate \(\eta\):

\[
\begin{align*}
f &\leftarrow f - \eta \nabla_f \mathcal{L}(f \circ \phi) \\
\phi &\leftarrow \phi - \eta \nabla_{\phi} \mathcal{L}(f \circ \phi) - \eta \lambda \sum_{e,e'} \nabla_{\phi} R_{e,e'}(\phi) \\
f_e &\leftarrow f_e - \eta \nabla_{f_e} \mathcal{L}^e(f_e \circ \phi) \\
(\xi_e, \psi_e) &\leftarrow (\xi_e, \psi_e) - \eta \nabla_{\xi_e, \psi_e} \mathcal{L}_{\text{VAE}}(\psi_e \circ \xi_e)
\end{align*}
\]

In each step, we sample a minibatch \(B\) from the training set \(\mathcal{D}\) to estimate \(\mathcal{L}(f \circ \phi)\). We also sample \(n\) minibatches \(B_1, \cdots, B_n\) from each environment \(\mathcal{D}_1, \cdots, \mathcal{D}_n\) to estimate the regularizer \(R_{e,e'}(\phi)\) and the VAE losses. The training procedure is detailed in Algorithm 1. Lastly, we would like to give two important remarks about the optimization procedure.

- This multiplayer game is inherently asymmetric. Given our formulation, the feature extractor \(\phi\) is not updated to minimize \(\mathcal{L}_{\text{VAE}}(\psi_e \circ \xi_e)\). In practice, this is
Algorithm 1 Counterfactual consistency: forward pass

1: Sample a minibatch $B$ from the training set $\mathcal{D}$.
2: Calculate the empirical loss $\mathcal{L}(f \circ \phi)$ on minibatch $B$.
3: Sample $n$ minibatches $B_1, \ldots, B_n$ from each environment $\mathcal{D}_1, \ldots, \mathcal{D}_n$.
4: for each environment $e$ do
5: Calculate the environment-specific loss $\mathcal{L}_e(f_e \circ \phi)$ on minibatch $B_e$.
6: Calculate the VAE loss $\mathcal{L}_{\text{VAE}}(\psi_e \circ \xi_e)$ on minibatch $B_e$.
7: for each environment $e' \neq e$ do
8: Compute the counterfactual representation $g_{e,e'}(\phi(x))$ for every $x \in B_e$.
9: Calculate the regularizer $\mathcal{R}_{e,e'}(\phi)$.
10: end for
11: end for

implemented through a stop-gradient operation between $\phi$ and $\{\xi_e, \psi_e\}$ (Figure 2-2a). Similarly, $f_{e'}$ and $g_{e,e'}$ are not updated to minimize the regularizer $\mathcal{R}_{e,e'}(\phi)$ (Figure 2-2b).

- $f_e$ and $\phi$ optimize $\mathcal{L}_e(f_e \circ \phi)$ in opposite directions, i.e. $f_e$ minimizes the loss while $\phi$ maximizes it in Eq.(2.14). In practice, we insert a gradient reversal layer [26] between $\phi$ and $f_e$ so that all the players can be updated in a single forward-backward pass (Figure 2-2b).

2.4 Experiments

Our method is evaluated on two real-world molecular property prediction datasets: Merck molecular activity challenge [64] and QM9 [77]. Our baselines include:

- Standard empirical risk minimization (ERM) trained on all the environments;
- Domain adversarial training methods (e.g. DANN [26] and CDAN [62]) which seek to align the distribution of training and test examples in the representation space;
- IRM [4] requiring the model to be simultaneously optimal in all environments;
- MLDG [56], a meta-learning method that simulates domain shift by dividing training environments into meta-training and meta-testing.
2.4.1 QM9

Setup. QM9 is a regression dataset with 12 properties related to quantum chemistry. The dataset has 134K molecules up to 9 heavy atoms. To compare the extrapolation ability of different models, prior work [16, 95] has proposed to split the dataset based on molecular size. Specifically, the training set contains molecules up to 8 heavy atoms and the test set consists of molecules with 9 heavy atoms. As most drug-like molecules considered in the industry are much larger than the QM9 molecules, this setup allows us to measure how well a model trained on QM9 can extrapolate to normal drug-like molecules.

The training set is further divided into two environments: one for molecules with no more than 7 atoms and the other for molecules with 8 atoms. In this case, the environment $E$ (molecular size) is a confounder because it affects both $X$ and $Y$. Therefore, this task agrees with our causal assumption.

Model. The molecule encoder $\phi$ is a message passing network based on the Chemprop architecture [107]. The predictor $f$ and $\{f_e\}$ are two-layer feed-forward networks (FFN) with hidden dimension $|z| = 300$. The VAE encoder and decoder are also FFNs with two hidden layers. We set $\lambda = 0.1$ and KL regularization weight $\beta = 0.1$. All the methods are trained for 30 epochs by an Adam optimizer with a learning rate of 0.001.

Results. Following Wu et al. [104], we report the mean absolute error (MAE) for each method. The results are shown in Table 2.1. Our method achieves the lowest MAE among all the methods (0.516 v.s. 0.637), with a 17.5% relative error reduction. Results are averaged across two independent runs.

2.4.2 Merck molecular activity challenge

Data. The Merck dataset consists of 15 properties related to medicinal chemistry. It is a regression dataset since the property $Y$ is a continuous variable. Given that the data is proprietary, the compound structure $X$ is anonymized and represented
Table 2.1: Results on the QM9 dataset. Each cell reports the MAE metric (↓). We normalize each property by its standard deviation before MAE calculation since their scales are very different. The standard deviation of the final MAE is calculated across two independent runs with different random seeds.

<table>
<thead>
<tr>
<th>Property</th>
<th>ERM</th>
<th>DANN</th>
<th>CDAN</th>
<th>IRM</th>
<th>MLDG</th>
<th>CFC</th>
<th>CFC_{id}</th>
<th>CFC_{vae}</th>
</tr>
</thead>
<tbody>
<tr>
<td>mu</td>
<td>0.368</td>
<td>0.376</td>
<td>0.369</td>
<td>0.418</td>
<td>0.375</td>
<td>0.462</td>
<td>0.431</td>
<td>0.450</td>
</tr>
<tr>
<td>alpha</td>
<td>1.123</td>
<td>1.228</td>
<td>1.130</td>
<td>0.938</td>
<td>0.936</td>
<td>0.711</td>
<td>0.810</td>
<td>0.844</td>
</tr>
<tr>
<td>HUMO</td>
<td>0.244</td>
<td>0.258</td>
<td>0.244</td>
<td>0.295</td>
<td>0.248</td>
<td>0.357</td>
<td>0.319</td>
<td>0.343</td>
</tr>
<tr>
<td>LUMO</td>
<td>0.132</td>
<td>0.149</td>
<td>0.133</td>
<td>0.162</td>
<td>0.132</td>
<td>0.203</td>
<td>0.197</td>
<td>0.199</td>
</tr>
<tr>
<td>gap</td>
<td>0.175</td>
<td>0.195</td>
<td>0.173</td>
<td>0.207</td>
<td>0.173</td>
<td>0.255</td>
<td>0.237</td>
<td>0.254</td>
</tr>
<tr>
<td>H2</td>
<td>0.793</td>
<td>0.646</td>
<td>0.792</td>
<td>0.631</td>
<td>0.660</td>
<td>0.516</td>
<td>0.550</td>
<td>0.592</td>
</tr>
<tr>
<td>ZPVE</td>
<td>0.438</td>
<td>0.343</td>
<td>0.433</td>
<td>0.419</td>
<td>0.392</td>
<td>0.273</td>
<td>0.309</td>
<td>0.311</td>
</tr>
<tr>
<td>Cv</td>
<td>0.889</td>
<td>0.696</td>
<td>0.860</td>
<td>0.751</td>
<td>0.720</td>
<td>0.495</td>
<td>0.567</td>
<td>0.601</td>
</tr>
<tr>
<td>U0</td>
<td>1.266</td>
<td>1.035</td>
<td>1.239</td>
<td>0.971</td>
<td>1.004</td>
<td>0.699</td>
<td>0.858</td>
<td>0.816</td>
</tr>
<tr>
<td>U</td>
<td>1.267</td>
<td>1.034</td>
<td>1.237</td>
<td>0.971</td>
<td>1.009</td>
<td>0.735</td>
<td>0.829</td>
<td>0.810</td>
</tr>
<tr>
<td>H</td>
<td>1.263</td>
<td>1.031</td>
<td>1.245</td>
<td>0.952</td>
<td>1.003</td>
<td>0.724</td>
<td>0.807</td>
<td>0.851</td>
</tr>
<tr>
<td>G</td>
<td>1.265</td>
<td>1.040</td>
<td>1.245</td>
<td>0.971</td>
<td>0.988</td>
<td>0.761</td>
<td>0.831</td>
<td>0.851</td>
</tr>
<tr>
<td>Average</td>
<td>0.768</td>
<td>0.669</td>
<td>0.758</td>
<td>0.640</td>
<td>0.637</td>
<td>0.516</td>
<td>0.562</td>
<td>0.577</td>
</tr>
<tr>
<td>Std</td>
<td>±.007</td>
<td>±.000</td>
<td>±.007</td>
<td>±.013</td>
<td>±.002</td>
<td>±.006</td>
<td>±.002</td>
<td>±.051</td>
</tr>
</tbody>
</table>

by a vector of anonymous features. The original dataset contains 153K molecules for training and 50K molecules for testing.

**Setup.** The original training and test sets are constructed based on temporal split. All the training compounds are assayed before a certain date, but their temporal order is unknown. As a result, the original training set has only one environment and it is impossible to apply any domain generalization methods. To construct a training set with multiple temporal-split environments, we sample a subset of test compounds and move them to the training set as the second environment. Specifically, we first partition the test set into multiple clusters based on the Jaccard distance between the compounds. We then select 20% of the clusters into the training set. This ensures the second training environment has a different distribution from the test set.

**Model.** The feature extractor $\phi$ is an FFN with two hidden layers. The predictors $f$ and $\{f_e\}$ are FFNs with one hidden layer. All hidden layers have 500 neurons with
Table 2.2: Results on the Merck dataset. We evaluate each model under a temporal split. Each cell reports the $R^2$ metric (coefficient of determination) and it is the higher the better ($\uparrow$).

<table>
<thead>
<tr>
<th>Property</th>
<th>ERM</th>
<th>DANN</th>
<th>CDAN</th>
<th>IRM</th>
<th>MLDG</th>
<th>CFC</th>
<th>CFC$_{id}$</th>
<th>CFC$_{vae}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A4</td>
<td>0.552</td>
<td>0.536</td>
<td>0.546</td>
<td>0.540</td>
<td>0.541</td>
<td>0.542</td>
<td>0.547</td>
<td>0.554</td>
</tr>
<tr>
<td>CB1</td>
<td>0.457</td>
<td>0.450</td>
<td>0.447</td>
<td>0.446</td>
<td>0.448</td>
<td>0.450</td>
<td>0.459</td>
<td>0.444</td>
</tr>
<tr>
<td>DPP4</td>
<td><strong>0.217</strong></td>
<td>0.177</td>
<td>0.208</td>
<td>0.176</td>
<td>0.173</td>
<td>0.200</td>
<td>0.197</td>
<td>0.194</td>
</tr>
<tr>
<td>HIV$_{INT}$</td>
<td>0.245</td>
<td><strong>0.299</strong></td>
<td>0.246</td>
<td>0.294</td>
<td>0.279</td>
<td>0.276</td>
<td>0.279</td>
<td>0.266</td>
</tr>
<tr>
<td>HIV$_{PROT}$</td>
<td>0.576</td>
<td>0.599</td>
<td>0.592</td>
<td>0.600</td>
<td>0.608</td>
<td>0.638</td>
<td>0.610</td>
<td><strong>0.642</strong></td>
</tr>
<tr>
<td>LOGD</td>
<td>0.827</td>
<td>0.825</td>
<td>0.825</td>
<td>0.825</td>
<td>0.825</td>
<td>0.829</td>
<td>0.826</td>
<td><strong>0.831</strong></td>
</tr>
<tr>
<td>METAB</td>
<td>0.654</td>
<td>0.666</td>
<td>0.672</td>
<td>0.673</td>
<td>0.658</td>
<td>0.677</td>
<td>0.680</td>
<td>0.675</td>
</tr>
<tr>
<td>NK1</td>
<td>0.386</td>
<td>0.360</td>
<td>0.380</td>
<td>0.403</td>
<td>0.384</td>
<td><strong>0.441</strong></td>
<td>0.384</td>
<td>0.399</td>
</tr>
<tr>
<td>OX1</td>
<td>0.589</td>
<td>0.605</td>
<td>0.593</td>
<td><strong>0.615</strong></td>
<td>0.592</td>
<td>0.610</td>
<td>0.600</td>
<td>0.592</td>
</tr>
<tr>
<td>OX2</td>
<td><strong>0.615</strong></td>
<td>0.602</td>
<td>0.602</td>
<td>0.596</td>
<td>0.600</td>
<td>0.606</td>
<td>0.599</td>
<td>0.599</td>
</tr>
<tr>
<td>PGP</td>
<td>0.565</td>
<td>0.558</td>
<td>0.564</td>
<td>0.565</td>
<td>0.560</td>
<td><strong>0.577</strong></td>
<td>0.574</td>
<td>0.577</td>
</tr>
<tr>
<td>PPB</td>
<td>0.610</td>
<td>0.609</td>
<td>0.607</td>
<td>0.603</td>
<td>0.613</td>
<td>0.604</td>
<td>0.609</td>
<td><strong>0.613</strong></td>
</tr>
<tr>
<td>RAT_F</td>
<td>0.092</td>
<td>0.155</td>
<td>0.158</td>
<td>0.137</td>
<td><strong>0.170</strong></td>
<td><strong>0.170</strong></td>
<td>0.113</td>
<td>0.126</td>
</tr>
<tr>
<td>TDI</td>
<td>0.298</td>
<td>0.262</td>
<td>0.290</td>
<td>0.256</td>
<td>0.269</td>
<td>0.265</td>
<td>0.265</td>
<td>0.275</td>
</tr>
<tr>
<td>THR</td>
<td>0.516</td>
<td><strong>0.540</strong></td>
<td>0.515</td>
<td>0.530</td>
<td>0.489</td>
<td>0.521</td>
<td>0.526</td>
<td>0.510</td>
</tr>
<tr>
<td>Average</td>
<td>0.480</td>
<td>0.483</td>
<td>0.483</td>
<td>0.484</td>
<td>0.484</td>
<td><strong>0.493</strong></td>
<td>0.484</td>
<td>0.487</td>
</tr>
<tr>
<td>Std</td>
<td>±.008</td>
<td>±.006</td>
<td>±.004</td>
<td>±.008</td>
<td>±.018</td>
<td>±.003</td>
<td>±.005</td>
<td>±.003</td>
</tr>
</tbody>
</table>

ReLU activation. The VAE architecture and the training hyperparameters are the same as in the previous experiment.

**Results.** Following Ma et al. [64], we report $R^2$ (coefficient of determination) for each method. Results are shown in Table 2.2. Our method outperforms all baselines in terms of average $R^2$, with the highest improvement on NK1 (0.441 v.s. 0.403). All results are averaged across two independent runs with different random seeds.

### 2.4.3 Ablation studies and analysis

In this section, we provide additional experiments to justify our modeling choices. We include three ablation studies:

- **CFC$_{id}$** uses only the identity function in the CFC regularizer: $R_{e,e'}(\phi) = \mathcal{L}(f_{e'} \circ \phi) - \mathcal{L}(f_e \circ \phi)$. It assumes that the representation $z$ does not contain any environment-
specific information.

- $\text{CFC}_{\text{vae}}$ uses only the VAE translation model in the CFC regularizer by setting $\mathcal{G}_{e,e'} = \{g_{e,e'}\}$ and $\mathcal{R}_{e,e'}(\phi) = \mathcal{L}^e(f_{e'} \circ g_{e,e'} \circ \phi) - \mathcal{L}^e(f_e \circ \phi)$.

- $\text{CFC}_{\text{cyc}}$ uses CycleGAN [113] as the translation model $g_{e,e'}$ instead of VAE. It enforces the counterfactual representations $z_{E \leftarrow e'} = g_{e,e'}(z)$ to have the same distribution as representations $z = \phi(x)$ from environment $e'$. Moreover, it requires $g_{e,e'}$ to be cycle consistent, i.e. $g_{e,e'}(g_{e',e}(z)) = z$ for all examples in the training set.

As shown in Table 2.1 and 2.2, $\text{CFC}_{\text{id}}$ and $\text{CFC}_{\text{vae}}$ perform worse than our full model. This shows the importance of including multiple translation functions in the CFC regularizer. Nonetheless, we find that $\text{CFC}_{\text{id}}$ and $\text{CFC}_{\text{vae}}$ generally outperform other baselines, which demonstrates the advantage of our framework.

As shown in Figure 2-3a, we find that $\text{CFC}_{\text{cyc}}$ performs much worse than CFC, even though CycleGAN puts more constraints on the translation model than VAE. We hypothesize that it is hard for a GAN to align different distributions when the true density of $\phi(x)$ is changing over time, since $\phi$ is constantly being updated during training. On the QM9 dataset, we also report the model performance under different CFC regularization weight $\lambda$ (Figure 2-3b).

**Visualization.** Lastly, we analyze the learned factual and counterfactual representations in the Merck dataset to understand the role of translation functions $g_{e,e'}$. In Figure 2-4, we provide a t-SNE visualization [96] of learned representations for three
properties (THROMBIN, DPP4, and METAB). We observe that there is a clear distributional difference between the factual and counterfactual representations (blue and orange dots). For THROMBIN and DPP4, we also find that the counterfactual representations are moved towards test molecules (green dots) far away from the training molecules. This is a desired behavior because it encourages the model to generalize to a new chemical space.

2.5 Summary

In this chapter, we have developed Chemprop software for molecular property prediction and CFC regularization for learning generalizable models. Despite its empirical improvement, CFC still has several limitations. For instance, it does not guarantee that the underlying content $U$ is preserved during counterfactual translation. A future direction is to leverage domain knowledge about $U$ (e.g., known pharmacophores) to guide the translation model $g_{e,e'}$. 
Chapter 3

Molecular Graph Generation

The key challenge of drug discovery is to find target molecules with desired chemical properties. Currently, this task takes years of development and exploration by expert chemists and pharmacologists. Our ultimate goal is to automate this process. From a computational perspective, we decompose the challenge into two complementary subtasks: learning to represent molecules in a continuous manner that facilitates the prediction and optimization of their properties (encoding); and learning to map an optimized continuous representation back into a molecular graph with improved properties (decoding). While deep learning has been extensively investigated for molecular graph encoding [24, 48, 28], the harder combinatorial task of molecular graph generation from latent representation remains under-explored.

Prior work on drug design formulated the graph generation task as a string generation problem [29, 52] in an attempt to side-step direct generation of graphs. Specifically, these models start by generating SMILES [98], a linear string notation used in chemistry to describe molecular structures. SMILES strings can be translated into graphs via deterministic mappings (e.g., using RDKit [54]). However, this design has two critical limitations. First, the SMILES representation is not designed to capture molecular similarity. For instance, two molecules with similar chemical structures may be encoded into markedly different SMILES strings (e.g., Figure 3-1). This prevents generative models like variational autoencoders from learning smooth molecular embeddings. Second, essential chemical properties such as molecule valid-
Figure 3-1: Two almost identical molecules with markedly different canonical SMILES in RDKit. The edit distance between two strings is 22 (50.5% of the whole sequence).

ity are easier to express on graphs rather than linear SMILES representations. We hypothesize that operating directly on graphs improves generative modeling of valid chemical structures.

Graphs are challenging objects to generate, especially for larger molecules such as polymers. For the polymer dataset used in our experiment, there are thousands of molecules with more than 80 atoms. To illustrate the challenge, we tested two state-of-the-art variational autoencoders [61, 40] on this dataset and found these models often fail to reconstruct molecules from their latent embedding (see Figure 4-1). The reason of this failure is that these methods generate molecules based on small building blocks. Previous methods [56, 109, 61] have sought to generate molecules atom by atom. As the building blocks are typically small, it requires many decoding steps for current models to reconstruct polymers. Therefore they are prone to make errors when generating large molecules. On the other hand, many of these molecules consist of structural motifs beyond simple atoms. The number of decoding steps can be significantly reduced if graphs are generated motif by motif. As shown in Figure 4-1, motif-based generation achieves a much higher reconstruction accuracy.

In this chapter, we propose a motif-based hierarchical encoder-decoder for graph generation. The motifs themselves are extracted separately at the outset from frequently occurring substructures, regardless of size. During generation, molecules are built step by step by attaching motifs, large or small, to the emerging molecule. The decoder operates hierarchically, in a coarse-to-fine manner, and makes three key consecutive predictions in each pass: new motif selection, which part of it attaches, and the points of contact with the current molecule. These decisions are highly coupled...
and naturally modeled auto-regressively. Moreover, each decision is directly guided by the information explicated in the associated layer of the mirroring hierarchical encoder. The feed-forward fine-to-coarse encoding performs iterative graph convolutions at each level, conditioned on the results from layer below.

The proposed model is evaluated on various tasks ranging from polymer generative modeling to graph translation for molecule property optimization. Our baselines include state-of-the-art graph generation methods [109, 61, 42]. On polymer generation, our model achieved state-of-the-art results under various metrics, outperforming the best baselines with 20% absolute improvement in reconstruction accuracy. On graph translation tasks, our model outperformed all the baselines, yielding 3.3% and 8.1% improvement on QED and DRD2 optimization tasks. We further conduct ablation studies to validate the advantage of using larger motifs and the proposed architecture.

### 3.1 Hierarchical Graph Generation

Molecules are represented as graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with atoms $\mathcal{V}$ as nodes and bonds $\mathcal{E}$ as edges. Our approach extends the variational autoencoder [49] to molecular graphs by introducing a hierarchical decoder and a matching encoder. In our framework, the probability of a graph $\mathcal{G}$ is modeled as a joint distribution over structural motifs $\mathcal{S}_1, \ldots, \mathcal{S}_n$ constituting $\mathcal{G}$, together with their attachments $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Each attach-
ment \( \mathcal{A}_i = \{ v_j \mid v_j \in \bigcup_k \mathcal{S}_i \cap \mathcal{S}_k \} \) indicates the intersecting atoms between \( \mathcal{S}_i \) and its neighbor motifs. To capture complex dependencies involved in the joint distribution of motifs and their attachments, we propose an auto-regressive factorization of \( P(\mathcal{G}) \):

\[
P(\mathcal{G}) = \int_{z} P(z) \prod_k P(\mathcal{S}_k, \mathcal{A}_k \mid \mathcal{S}_{<k}, \mathcal{A}_{<k}, z) dz
\]

(3.1)

As illustrated in Figure 3-4, in each generation step, our decoder adds a new motif \( \mathcal{S}_k \) (motif prediction) and its attachment configuration \( \mathcal{A}_k \) (attachment prediction). Then it decides how the new motif should be attached to the current graph (graph prediction).

To support the above hierarchical generation, we need to design a matching encoder representing molecules at multiple resolutions in order to provide necessary information for each decoding step. Therefore, we propose to represent a molecule \( \mathcal{G} \) by a hierarchical graph \( \mathcal{H}_\mathcal{G} \) with three layers (see Figure 3-3):

1. **Motif layer**: This layer represents how the motifs are coarsely connected in the graph. This layer provides essential information for the motif prediction in the decoding process. Specifically, this layer contains \( n \) nodes \( \mathcal{S}_1, \ldots, \mathcal{S}_n \) and \( m \) edges \( \{(\mathcal{S}_i, \mathcal{S}_j) \mid \mathcal{S}_i \cap \mathcal{S}_j \neq \emptyset \} \) for all intersecting motifs \( \mathcal{S}_i, \mathcal{S}_j \). This layer is tree-structured due to our way of constructing motifs.

2. **Attachment layer**: This layer encodes the connectivity between motifs at a fine-grained level. Each node \( \mathcal{A}_i = (\mathcal{S}_i, \{v_j\}) \) in this layer represents a particular attachment configuration of motif \( \mathcal{S}_i \), where \( \{v_j\} \) are atoms in the intersection between \( \mathcal{S}_i \) and one of its neighbor motifs (see Figure 3-3). This layer provides crucial information for the attachment prediction step during decoding, which helps reducing the space of candidate attachments between \( \mathcal{S}_i \) and its neighbor motifs. Just like the motif vocabulary \( \mathcal{V}_\mathcal{S} \), all the attachment configurations of \( \mathcal{S}_i \) form a motif-specific vocabulary \( \mathcal{V}_\mathcal{A}(\mathcal{S}_i) \), which is computed from the training set.\(^1\)

3. **Atom layer**: The atom layer is the molecular graph \( \mathcal{G} \) representing how its atoms are connected. Each atom node \( v \) is associated with a label \( a_v \) indicating its atom

\(^1\)In our experiments, the average size of attachment vocabulary \( | \mathcal{V}_\mathcal{A}(\mathcal{S}_i) | \leq 10 \) and the size of motif vocabulary \( | \mathcal{V}_\mathcal{S} | < 500 \).
type and charge. Each edge \((u, v)\) in the atom layer is labeled with \(b_{uv}\) indicating its bond type. This layer provides necessary information for the graph prediction step during decoding.

We further introduce edges that connect the atoms and motifs between different layers in order to propagate information in between. In particular, we draw a directed edge from atom \(v\) in the atom layer to node \(\mathcal{A}_i\) in the attachment layer if \(v \in \mathcal{S}_i\). We also draw edges from \(\mathcal{A}_i\) to \(\mathcal{S}_i\) in the motif layer. This gives us the hierarchical graph \(\mathcal{H}_G\) for molecule \(\mathcal{G}\), which will be encoded by a hierarchical message passing network (MPN). During encoding, each node \(\mathcal{S}_i\) is represented as a one-hot encoding in the motif vocabulary \(V_{\mathcal{S}}\). Likewise, each node \(\mathcal{A}_i\) is represented as a one-hot encoding in the attachment vocabulary \(V_{\mathcal{A}(\mathcal{S}_i)}\).

### 3.1.1 Motif Extraction

We define a motif \(\mathcal{S}_i = (\mathcal{V}_i, \mathcal{E}_i)\) as a subgraph of molecule \(\mathcal{G}\) induced by atoms in \(\mathcal{V}_i\) and bonds in \(\mathcal{E}_i\). Given a molecule, we extract its motifs \(\mathcal{S}_1, \ldots, \mathcal{S}_n\) such that their union covers the entire molecular graph: \(\mathcal{V} = \bigcup_i \mathcal{V}_i\) and \(\mathcal{E} = \bigcup_i \mathcal{E}_i\). To extract motifs, we decompose a molecule \(\mathcal{G}\) into disconnected fragments by breaking all the bridge bonds that will not violate chemical validity (illustrations in the appendix).

1. Find all the bridge bonds \((u, v)\) \(\in \mathcal{E}\), where both \(u\) and \(v\) have degree \(\Delta_u, \Delta_v \geq 2\) and either \(u\) or \(v\) is part of a ring. Detach all the bridge bonds from its neighbors.

2. Now the graph \(\mathcal{G}\) becomes a set of disconnected subgraphs \(\mathcal{G}_1, \ldots, \mathcal{G}_N\). Select \(\mathcal{G}_i\) as motif in \(\mathcal{G}\) if its occurrence in the training set is more than \(\Delta = 100\).

3. If \(\mathcal{G}_i\) is not selected as motif, further decompose it into rings and bonds and select them as motif in \(\mathcal{G}\).

We apply the above procedure to all the molecules in the training set and construct a vocabulary of motifs \(V_{\mathcal{S}}\). In the following section, we will describe how we encode and decode molecules using the extracted motifs.
3.1.2 Hierarchical Graph Encoder

Our encoder contains three MPNs that encode each of the three layers in the hierarchical graph. For simplicity, we denote the MPN encoding process as $\text{MPN}_\psi(\cdot)$ with parameter $\psi$, and denote $\text{MLP}(x,y)$ as a multi-layer neural network whose input is the concatenation of $x$ and $y$. The details of MPN architecture is listed in the appendix.

**Atom Layer MPN** We first encode the atom layer of $\mathcal{H}_G$ (denoted as $\mathcal{H}^a_G$). The inputs to this MPN are the embedding vectors $\{e(a_u)\}, \{e(b_{uv})\}$ of all the atoms and bonds in $G$. During encoding, the network propagates the message vectors between different atoms for $T$ iterations and then outputs the atom representation $h_v$ for each atom $v$:

$$c^a_G = \{h_v\} = \text{MPN}_{\psi_1}(\mathcal{H}^a_G, \{e(a_u)\}, \{e(b_{uv})\})$$  \hspace{1cm} (3.2)
**Attachment Layer MPN**  The input feature of each node $A_i$ in the attachment layer $H^a_{\mathcal{G}}$ is an concatenation of the embedding $e(A_i)$ and the sum of its atom vectors $\{h_v \mid v \in S_i\}$:

$$f_{A_i} = \text{MLP} \left( e(A_i), \sum_{v \in S_i} h_v \right)$$  \hspace{1cm} (3.3)

The input feature for each edge $(A_i, A_j)$ in this layer is an embedding vector $e(d_{ij})$, where $d_{ij}$ describes the relative ordering between node $A_i$ and $A_j$ during decoding. Specifically, we set $d_{ij} = k$ if node $A_i$ is the $k$-th child of node $A_j$ and $d_{ij} = 0$ if $A_i$ is the parent. We then run $T$ iterations of message passing over $H^a_{\mathcal{G}}$ to compute the motif representations:

$$e^a_{\mathcal{G}} = \{h_{A_i}\} = \text{MPN}_{\psi_2} \left( H^a_{\mathcal{G}}, \{f_{A_i}\}, \{e(d_{ij})\} \right)$$  \hspace{1cm} (3.4)

**Motif Layer MPN**  Similarly, the input feature of node $S_i$ in this layer is computed as the concatenation of embedding $e(S_i)$ and the node vector $h_{A_i}$ from the previous layer. Finally, we run message passing over the motif layer $H^s_{\mathcal{G}}$ to obtain the motif representations:

$$f_{S_i} = \text{MLP} \left( e(S_i), h_{A_i} \right)$$  \hspace{1cm} (3.5)$$c^s_{\mathcal{G}} = \{h_{S_i}\} = \text{MPN}_{\psi_3} \left( H^s_{\mathcal{G}}, \{f_{S_i}\}, \{e(d_{ij})\} \right)$$  \hspace{1cm} (3.6)

Finally, we represent a molecule $\mathcal{G}$ by a latent vector $z_{\mathcal{G}}$ sampled through reparameterization trick with mean $\mu(h_{S_1})$ and log variance $\Sigma(h_{S_1})$:

$$z_{\mathcal{G}} = \mu(h_{S_1}) + \exp(\Sigma(h_{S_1})) \cdot \epsilon; \quad \epsilon \sim \mathcal{N}(0, I)$$  \hspace{1cm} (3.7)

where $S_1$ is the root motif (i.e., the first motif to be generated during reconstruction).

### 3.1.3 Hierarchical Graph Decoder

As illustrated in Figure 3-4, our graph decoder generates a molecule $\mathcal{G}$ by incrementally expanding its hierarchical graph. In $t^{th}$ generation step, we first use the same
Figure 3-4: Hierarchical graph decoder. In each step, the decoder first runs hierarchical message passing to compute motif, attachment and atom vectors. Then it performs motif and attachment prediction for the next motif node. Finally, it decides how the new motif should be attached to the current graph via graph prediction.

Hierarchical MPN architecture to encode all the motifs and atoms in $\mathcal{H}_G^{(t)}$, the (partial) hierarchical graph generated till step $t$. This gives us motif vectors $h_{S_k}$ and atom vectors $h_{v_j}$ for the existing motifs and atoms.

During decoding, the model maintains a set of frontier nodes $\mathcal{F}$ where each node $S_k \in \mathcal{F}$ is a motif that still has neighbors to be generated. $\mathcal{F}$ is implemented as a stack because motifs are generated in their depth-first order. Suppose $S_k$ is at the top of stack $\mathcal{F}$ in step $t$, the model makes the following predictions conditioned on latent representation $z_G$:

1. **Motif Prediction**: The model predicts the next motif $S_t$ to be attached to $S_k$.

   This is cast as a classification task over the motif vocabulary $V_S$:

   $$p_{S_t} = \text{softmax}(\text{MLP}(h_{S_k}, z_G))$$  \hspace{1cm} (3.8)
2. **Attachment Prediction**: Now the model needs to predict the attachment configuration $A_t$ of motif $S_t$ (i.e., what atoms $v_j \in S_t$ belong to the intersection of $S_t$ and its neighbor motifs). This is also cast as a classification task over the attachment vocabulary $V_A(S_t)$:

$$p_{A_t} = \text{softmax}(\text{MLP}(h_{S_k}, z_G))$$

(3.9)

This prediction step is crucial because it significantly reduces the space of possible attachments between $S_t$ and its neighbor motifs.

3. **Graph Prediction**: Finally, the model must decide how $S_t$ should be attached to $S_k$. The attachment between $S_t$ and $S_k$ is defined as atom pairs $M_{tk} = \{(u_j, v_j) \mid u_j \in A_k, v_j \in A_t\}$ where atom $u_j$ and $v_j$ are attached together. The probability of a candidate attachment $M$ is computed based on the atom vectors $h_{u_j}$ and $h_{v_j}$:

$$p_M = \text{softmax} (h_M \cdot z_G)$$

(3.10)

$$h_M = \sum_j \text{MLP}(h_{u_j}, h_{v_j})$$

(3.11)

The number of possible attachments are limited because the number of attaching atoms between two motifs is small and the attaching points must be consecutive.\(^2\)

The above three predictions together give an autoregressive factorization of the distribution over the next motif and its attachment. Each of the three decoding steps depends on the outcome of previous step, and predicted attachments will in turn affect the prediction of subsequent motifs.

**Training** During training, we apply teacher forcing to the above generation process, where the generation order is determined by a depth-first traversal over the ground truth molecule. Given a training set of molecules, we seek to minimize the negative ELBO:

$$-\mathbb{E}_{z \sim Q}[\log P(G|z)] + \lambda_{\text{KL}} \mathcal{D}_{\text{KL}}[Q(z|G)||P(z)]$$

(3.12)

\(^2\)In our experiments, the number of possible attachments are usually less than 20 for polymers and small molecules.
3.1.4 Extension to Graph-to-Graph Translation

The proposed architecture can be naturally extended to graph-to-graph translation [42] for molecular optimization, which seeks to modify compounds in order to improve their biochemical properties. Given a corpus of molecular pairs \( \{(X, Y)\} \), where \( Y \) is a structural analog of \( X \) with better chemical properties, the model is trained to translate an input molecular graph into its better form. In this case, we seek to learn a translation model \( P(Y|X) \) parameterized by our encoder-decoder architecture. We also introduce attention layers into our model, which is crucial for translation performance [6].

**Training** In graph translation, a compound \( X \) can be associated with multiple outputs \( Y \) since there are many ways to modify \( X \) to improve its properties. In order to generate diverse outputs, we follow previous work [114, 42] and incorporate latent variables \( z \) to the translation model:

\[
P(Y|X) = \int_z P(Y|X, z)P(z)dz
\]  

(3.13)

where the latent vector \( z \) indicates the intended mode of translation, sampled from a prior \( P(z) \) during testing.

The model is trained as a conditional variational autoencoder. Given a training example \( (X, Y) \), we sample \( z \) from the approximate posterior \( Q(z|X, Y) = \mathcal{N}(\mu_{X,Y}, \sigma_{X,Y}) \). To compute \( Q(z|X, Y) \), we first encode \( X \) and \( Y \) into their representations \( c_X \) and \( c_Y \) and then compute difference vector \( \delta_{X,Y} \) that summarizes the structural changes from molecule \( X \) to \( Y \) at both atom and motif level:

\[
\delta_{X,Y}^s = \sum c_{Y}^s - \sum c_{X}^s \quad \delta_{X,Y}^g = \sum c_{Y}^g - \sum c_{X}^g
\]

Finally, we compute \( [\mu_{X,Y}, \sigma_{X,Y}] = MLP(\delta_{X,Y}^s, \delta_{X,Y}^g) \) and sample \( z \) using reparameterization trick. The latent code \( z \) is passed to the decoder along with the input representation \( c_X \) to reconstruct output \( Y \). The training objective is to minimize
negative ELBO similar to Eq.(3.12).

**Attention** For graph translation, the input molecule $X$ is embedded by our hierarchical encoder into a set of vectors $c_X = c^s_X \cup c^a_X \cup c^g_X$, representing the molecule at multiple resolutions. These vectors are fed into the decoder through attention mechanisms [63]. Specifically, we modify the motif prediction (Eq. 3.8) into

$$p_{S_t} = \text{softmax}(\text{MLP}(h_{S_t}, \alpha^s_t, z)) \quad (3.14)$$

$$\alpha^s_t = \text{attention}(h_{S_t}, c^s_X) \quad (3.15)$$

where $\text{attention}(h, c^s_X)$ is a bilinear attention over vectors $c^s_X$ with query vector $h_{S_t}$. The attachment prediction (Eq. 3.9) is modified similarly with its attention over $c^a_X$. The graph prediction (Eq. 3.10) is modified into

$$p_M = \text{softmax}(h_M \cdot \text{attention}(h_M, c^g_X)) \quad (3.16)$$

$$h_M = \sum_j \text{MLP}(h_{u_j}, h_{v_j}, z) \quad (3.17)$$

### 3.2 Experiments

We evaluate our method on two application tasks. The first task is polymer generative modeling. This experiment validates our argument in section ?? that our model is advantageous when the molecules have large sizes. The second task is graph-to-graph translation for small molecules. Here we show the proposed architecture also brings benefits to small molecules compared to previous state-of-the-art graph generation methods.

#### 3.2.1 Polymer Generative Modeling

**Dataset** Our method is evaluated on the polymer dataset from St. John et al. [89], which contains 86K polymers in total (after removing duplicates). The dataset is divided into 76K, 5K and 5K for training, validation and testing. Using our motif
Table 3.1: Results on polymer generative modeling. The first row reports the oracle performance using real data as generated samples. The last row (small motif) is a variant of our model where we restrict the motif vocabulary to contain only single rings and bonds (similar to JT-VAE). SNN means nearest neighbor similarity; “Frag / Scaf” means fragment and scaffold similarity. Except property statistics, all metrics are the higher the better.

<table>
<thead>
<tr>
<th></th>
<th>Reconstruction</th>
<th>Validity</th>
<th>Uniqueness</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real data</td>
<td>-</td>
<td>100%</td>
<td>100%</td>
<td>0.823</td>
</tr>
<tr>
<td>SMILES</td>
<td>21.5%</td>
<td>93.1%</td>
<td>97.3%</td>
<td>0.821</td>
</tr>
<tr>
<td>CG-VAE</td>
<td>42.4%</td>
<td>100%</td>
<td>96.2%</td>
<td>0.879</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>58.5%</td>
<td>100%</td>
<td>94.1%</td>
<td>0.864</td>
</tr>
<tr>
<td>HierVAE</td>
<td>79.9%</td>
<td>100%</td>
<td>97.0%</td>
<td>0.817</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMILES</td>
<td>21.5%</td>
<td>93.1%</td>
<td>97.3%</td>
<td>0.821</td>
</tr>
<tr>
<td>CG-VAE</td>
<td>42.4%</td>
<td>100%</td>
<td>96.2%</td>
<td>0.879</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>58.5%</td>
<td>100%</td>
<td>94.1%</td>
<td>0.864</td>
</tr>
<tr>
<td>HierVAE</td>
<td>79.9%</td>
<td>100%</td>
<td>97.0%</td>
<td>0.817</td>
</tr>
<tr>
<td>Small motif</td>
<td>71.0%</td>
<td>100%</td>
<td>97.2%</td>
<td>0.835</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Property Statistics (↓)</th>
<th>Structural Statistics (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>logP</td>
<td>SA</td>
</tr>
<tr>
<td>Real data</td>
<td>0.094</td>
<td>6.7e-5</td>
</tr>
<tr>
<td>SMILES</td>
<td>1.471</td>
<td>0.011</td>
</tr>
<tr>
<td>CG-VAE</td>
<td>3.958</td>
<td>2.600</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>2.645</td>
<td>0.157</td>
</tr>
<tr>
<td>HierVAE</td>
<td>0.525</td>
<td>0.007</td>
</tr>
<tr>
<td>Small motif</td>
<td>0.872</td>
<td>0.042</td>
</tr>
</tbody>
</table>

extraction, we collected 436 different motifs (examples shown in Figure 3-5a). On average, each motif has 5.24 different attachment configurations. The distribution of motif size and their frequencies are reported in Figure 3-5b.

**Evaluation Metrics**  Our evaluation effort measures various aspects of molecule generation proposed in Kusner et al. [52], Polykovskiy et al. [73]. Besides basic metrics like chemical validity and diversity, we compare distributional statistics between generated and real compounds. A good generative model should generate molecules which present similar aggregate statistics to real compounds. Our metrics include (with details shown in the appendix):

- **Reconstruction accuracy**: We measure how often the model can completely reconstruct a given molecule from its latent embedding $z$. The reconstruction
accuracy is computed over 5K compounds in the test set.

- **Validity**: Percentage of chemically valid compounds.
- **Uniqueness**: Percentage of unique compounds.
- **Diversity**: We compute the pairwise molecular distance among generated compounds. The molecular distance \( \text{dist}(X, Y) \) is defined as the Tanimoto distance over Morgan fingerprints [79] of two molecules.
- **Property statistics**: We compare property statistics between generated molecules and real data. Our properties include *partition coefficient* (logP), *synthetic accessibility* (SA), *drug-likeness* (QED) and *molecular weight* (MW). To quantitatively evaluate the distance between two distributions, we compute Frechet distance between property distributions of molecules in the generated and test sets [73].
- **Structural statistics**: We also compute structural statistics between generated molecules and real data. *Nearest neighbor similarity* (SNN) is the average similarity of generated molecules to the nearest molecule from the test set. *Fragment similarity* (Frag) and *scaffold similarity* (Scaf) are cosine distances between vectors of fragment or scaffold frequencies of the generated and the test set.

**Baselines** We compare our method against three state-of-the-art variational autoencoders for molecular graphs. SMILES VAE [29] is a sequence to sequence VAE that generates molecules based on their SMILES strings [98]. CG-VAE [61] is a
graph-based VAE generating molecules atom by atom. JT-VAE [40] is also a graph-
based VAE generating molecules based on simple substructures restricted to rings
and bonds. Finally, we report the oracle performance of distributional statistics by
using real molecules in the training set as our generated samples.

Results The performance of different methods are summarized in Table 3.1. Our
method (HierVAE) significantly outperforms all previous methods in terms of recon-
struction accuracy (79.9% vs 58.5%). This validates the advantage of utilizing large
structural motifs, which reduces the number of generation steps. In terms of dis-
tributional statistics, our method achieves state-of-the-art results on logP (0.525 vs
1.471), molecular weight Frechet distance (1928 vs 4863) and all the structural simi-
ilarity metrics. Since our model requires fewer generation steps, our training speed is
much faster than other graph-based methods (see Figure 3-5b).

Ablation Study To validate the importance of utilizing large structural motifs,
we further experiment a variant of our model (small motif), which keeps the same
architecture but replaces the large structural motifs with basic substructures such as
rings and bonds (with less than ten atoms). As shown in Table 3.1, its performance is
significantly worse than our full model even though it builds on the same hierarchical
architecture.

3.2.2 Graph-to-Graph Translation

We follow the experimental design by Jin et al. [42] and evaluate our model on their
graph-to-graph translation tasks. Following their setup, we require the molecular
similarity between $X$ and output $Y$ to be above certain threshold $\text{sim}(X, Y) \geq \delta$ at
test time. This is to prevent the model from ignoring input $X$ and translating it
into arbitrary compound. Here the molecular similarity is defined as $\text{sim}(X, Y) =
1 - \text{dist}(X, Y)$. 
Table 3.2: Results on graph translation tasks from Jin et al. [42]. We report average improvement for continuous properties (logP), and success rate for binary properties (e.g., DRD2).

<table>
<thead>
<tr>
<th></th>
<th>logP (sim ≥ 0.6)</th>
<th>logP (sim ≥ 0.4)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Improvement</td>
<td>Diversity</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>0.28 ± 0.79</td>
<td>-</td>
</tr>
<tr>
<td>CG-VAE</td>
<td>0.25 ± 0.74</td>
<td>-</td>
</tr>
<tr>
<td>GCPN</td>
<td>0.79 ± 0.63</td>
<td>-</td>
</tr>
<tr>
<td>MMPA</td>
<td>1.65 ± 1.44</td>
<td>0.329</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>2.33 ± 1.17</td>
<td>0.331</td>
</tr>
<tr>
<td>JTNN</td>
<td>2.33 ± 1.24</td>
<td>0.333</td>
</tr>
<tr>
<td>AtomG2G</td>
<td>2.41 ± 1.19</td>
<td>0.379</td>
</tr>
<tr>
<td>HierG2G</td>
<td>2.49 ± 1.09</td>
<td>0.381</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Drug likeness</th>
<th>DRD2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Success</td>
<td>Diversity</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>8.8%</td>
<td>-</td>
</tr>
<tr>
<td>CG-VAE</td>
<td>4.8%</td>
<td>-</td>
</tr>
<tr>
<td>GCPN</td>
<td>9.4%</td>
<td>0.216</td>
</tr>
<tr>
<td>MMPA</td>
<td>32.9%</td>
<td>0.236</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>58.5%</td>
<td>0.331</td>
</tr>
<tr>
<td>JTNN</td>
<td>59.9%</td>
<td>0.373</td>
</tr>
<tr>
<td>AtomG2G</td>
<td>73.6%</td>
<td>0.421</td>
</tr>
<tr>
<td>HierG2G</td>
<td>76.9%</td>
<td>0.477</td>
</tr>
</tbody>
</table>

**Dataset**  
The dataset consists of four property optimization tasks. In each task, we train and evaluate our model on their provided training and test sets.

- **LogP**: The penalized logP score [52] measures the solubility and synthetic accessibility of a compound. In this task, the model needs to translate input $X$ into output $Y$ such that $\log P(Y) > \log P(X)$. We experiment with two similarity thresholds $\delta = \{0.4, 0.6\}$.

- **QED**: The QED score [9] quantifies a compound’s drug-likeness. In this task, the model needs to translate molecules with QED scores from the lower range $[0.7, 0.8]$ into the higher range $[0.9, 1.0]$. The similarity constraint is $\text{sim}(X, Y) \geq 0.4$.

- **DRD2**: This task involves the optimization of a compound’s biological activity
against dopamine type 2 receptor (DRD2). The model needs to translate inactive compounds ($p < 0.05$) into active compounds ($p \geq 0.5$), where the bioactivity is assessed by a property prediction model from Olivecrona et al. [71]. The similarity constraint is $\text{sim}(X, Y) \geq 0.4$.

**Evaluation Metrics** Our evaluation metrics include translation accuracy and diversity. Each test molecule $X_i$ is translated $K = 20$ times with different latent codes sampled from the prior distribution. On the logP optimization, we select compound $Y_i$ as the final translation of $X_i$ that gives the highest property improvement and satisfies $\text{sim}(X_i, Y_i) \geq \delta$. We then report the average property improvement $\frac{1}{|D|} \sum_i \text{logP}(Y_i) - \text{logP}(X_i)$ over test set $D$. For other tasks, we report the translation success rate. A compound is successfully translated if one of its $K$ translation candidates satisfies all the similarity and property constraints of the task. To measure the diversity, for each molecule we compute the average pairwise Tanimoto distance between all its successfully translated compounds.

**Baselines** We compare our method against the baselines including GCPN [109], MMPA [22] and translation based methods Seq2Seq and JTNN [42]. Seq2Seq is a sequence-to-sequence model that generates molecules by their SMILES strings. JTNN is a graph-to-graph architecture that generates molecules structure by structure, but its decoder is not fully autoregressive.

To make a direct comparison possible between our method and atom-based generation, we further developed an atom-based translation model (AtomG2G) as baseline. It makes three predictions in each generation step. First, it predicts whether the de-
coding process has completed (no more new atoms). If not, it creates a new atom \( a_t \) and predicts its atom type. Lastly, it predicts the bond type between \( a_t \) and other atoms autoregressively to fully capture edge dependencies [110]. The encoder of AtomG2G encodes only the atom-layer graph and the decoder attention only sees the atom vectors \( c_x^g \). All translation models are trained under the same variational objective. Details of baseline architectures are in the appendix.

**Results** As shown in Table 3.2, our model (HierG2G) achieves the new state-of-the-art on the four translation tasks. In particular, our model significantly outperforms JTNN in both translation accuracy (e.g., 76.9% versus 59.9% on the QED task) and output diversity (e.g., 0.564 versus 0.480 on the logP task). While both methods generate molecules by structures, our decoder is autoregressive which can learn more expressive mappings. In addition, our model runs 6.3 times faster than JTNN during decoding. Our model also outperforms AtomG2G on three datasets, with over 10% improvement on the DRD2 task. This shows the advantage of our hierarchical model.

**Ablation Study** To understand the importance of different architecture choices, we report ablation studies over the QED and DRD2 tasks in Table 3.3. We first replace our hierarchical decoder with the atom-based decoder of AtomG2G to see how much the motif-based decoding benefits us. We keep the same hierarchical encoder but modified the input of the decoder attention to include both atom and motif vectors. Using this setup, the model performance decreases by 0.8% and 10.9% on the two tasks. We suspect the DRD2 task benefits more from motif-based decoding because biological target binding often depends on the presence of specific functional groups.

Our second experiment reduces the number of hierarchies in our encoder and decoder MPN, while keeping the same hierarchical decoding process. When the top motif layer is removed, the translation accuracy drops slightly by 0.8% and 2.4%. When we further remove the attachment layer (one-layer encoder), the performance degrades significantly on both datasets. This is because all the motif information is lost and the model needs to infer what motifs are and how motif layers are constructed.
for each molecule. This shows the importance of the hierarchical representation.

### 3.3 Summary

In this chapter, we developed a hierarchical encoder-decoder architecture generating molecular graphs using structural motifs as building blocks. The experimental results show our model outperforms prior atom and substructure based methods in both small molecule and polymer domains.
Chapter 4

Multi-objective Molecule Generation

In the last chapter, we presented a generative model for single-property molecular design. However, in real-world drug discovery projects, a molecule needs to satisfy multiple constraints, from potency, safety, to desired metabolic profiles. Optimizing these constraints simultaneously is challenging for existing computational models. The primary difficulty lies in the lack of training instances of molecules that conform to all the constraints. For example, for this reason, Jin et al. [41] reports over 60% performance loss when moving beyond the single-constraint setting.

In this chapter, we propose a novel approach to multi-property molecular optimization. Our strategy is inspired by fragment-based drug discovery [70] often followed by medicinal chemists. The idea is to start with substructures (e.g., functional groups or later pieces) that drive specific properties of interest, and then combine these building blocks into a target molecule. To automate this process, our model has to learn two complementary tasks: (1) identification of the building blocks that we call rationales, and (2) assembling multiple rationales together into a fully formed target molecule. In contrast to competing methods, our generative model does not build molecules from scratch, but instead assembles them from automatically extracted rationales already implicated for specific properties (see Figure 4-1).

We implement this idea using a generative model of molecules where the rationale choices play the role of latent variables. Specifically, a molecular graph $G$ is generated
Figure 4-1: Illustration of RationaleRL. **Left:** To generate a dual inhibitor against biological targets GSK3\(\beta\) and JNK3, our model first identifies rationale substructures \(\mathcal{S}\) for each property. Note that rationales are not provided as domain knowledge. **Middle:** The model learns to compose multiple rationales \(\mathcal{S}\) into a complete molecule \(\mathcal{G}\). **Right:** Our method achieves much higher success rate than the current state-of-the-art molecule design method REINVENT [71]) under four property constraints.

\[
P(\mathcal{G}) = \sum_{\mathcal{S}} P(\mathcal{G}|\mathcal{S}) P(\mathcal{S})
\]  

(4.1)

As ground truth rationales (e.g., functional groups or subgraphs) are not provided, the model has to extract candidate rationales from molecules with the help of a property predictor. We formulate this task as a discrete optimization problem efficiently solved by Monte Carlo tree search. Our rationale conditioned graph generator, \(P(\mathcal{G}|\mathcal{S})\), is initially trained on a large collection of real molecules so that it is capable of expanding any subgraph into a full molecule. The mixture model is then fine-tuned using reinforcement learning to ensure that the generated molecules preserve all the properties of interest. This training paradigm enables us to realize molecules that satisfy multiple constraints without observing any such instances in the training set.

The proposed model is evaluated on molecule design tasks under different combinations of property constraints. Our baselines include state-of-the-art molecule generation methods [71, 109]. Across all tasks, our model achieve state-of-the art results in terms of accuracy, novelty and diversity of generated compounds. In particular, we outperform the best baseline with 38% absolute improvement in the task with three property constraints. Finally, we show that identified rationales are chemically meaningful in a toxicity prediction task [92].
Figure 4-2: Overview of our approach. We first construct rationales for each individual property and then combine them as multi-property rationales. The method learns a graph completion model \( P(G|S) \) and rationale distribution \( P(S) \) in order to generate positive molecules.

### 4.1 Proposed Approach: RationaleRL

Molecules are represented as graphs \( G = (V, E) \) with atoms \( V \) as nodes and bonds \( E \) as edges. The goal of drug discovery is to find novel compounds satisfying given property constraints (e.g., drug-likeness, binding affinity, etc.). Without loss of generality, we assume the property constraints to be of the following form:

\[
\text{Find molecules } G \\
\text{Subject to } r_i(G) \geq \delta_i; \quad i = 1, \ldots, M
\]  

(4.2)

For each property \( i \), the property score \( r_i(G) \in [0, 1] \) of molecule \( G \) must be higher than threshold \( \delta_i \in [0, 1] \). A molecule \( G \) is called positive to property \( i \) if \( r_i(G) \geq \delta_i \) and negative otherwise.

Following previous work [71, 74], \( r_i(G) \) is output of property prediction models (e.g., random forests) which effectively approximate empirical measurements. The prediction model is trained over a set of molecules with labeled properties gathered from real experimental data. The property predictor is then fixed throughout the rest of the training process.

**Overview**  Our model generates molecules by first sampling a rationale \( S \) from the vocabulary \( V_S^{[M]} \), and then completing it into a molecule \( G \). The generative model is
defined as

\[
P(\mathcal{G}) = \sum_{\mathcal{S} \in V_{S}^{[M]}} P(\mathcal{S})P(\mathcal{G}|\mathcal{S})
\]  \tag{4.3}

As shown in Figure 4-2, our model consists of three modules:

- **Rationale Extraction**: Construct rationale vocabulary \( V_{S}^{i} \) for each individual property \( i \) and combines these rationales for multiple properties \( V_{S}^{[M]} \) (see §4.1.1).

- **Graph Completion \( P(\mathcal{G}|\mathcal{S}) \)**: Generate molecules \( \mathcal{G} \) using multi-property rationales \( S^{[M]} \in V_{S}^{[M]} \). The model is first pre-trained on natural compounds and then fine-tuned to generate molecules satisfying multiple constraints (see §4.1.2 for its architecture and §4.1.3 for fine-tuning).

- **Rationale Distribution \( P(\mathcal{S}) \)**: The rationale distribution \( P(\mathcal{S}) \) is learned based on the properties of complete molecules \( \mathcal{G} \) generated from \( P(\mathcal{G}|\mathcal{S}) \). A rationale \( \mathcal{S} \) is sampled more frequently if it is more likely to be expanded into a positive molecule \( \mathcal{G} \) (see §4.1.3).

### 4.1.1 Rationale Extraction from Predictive Models

**Single-property Rationale** We define a rationale \( S^{i} \) for a single property \( i \) as a subgraph of some molecule \( \mathcal{G} \) which causes \( \mathcal{G} \) to be active (see Figure 4-1). To be specific, let \( V_{S}^{i} \) be the vocabulary of such rationales for property \( i \). Each rationale \( S^{i} \in V_{S}^{i} \) should satisfy the following two criteria to be considered as a rationale:

1. The size of \( S^{i} \) should be small (less than 20 atoms).
2. Its predicted property score \( r_{i}(S^{i}) \geq \delta_{i} \).

For a single property \( i \), we propose to extract its rationales from a set of *positive* molecules \( D_{i}^{pos} \) used to train the property predictor. For each molecule \( \mathcal{G}_{i}^{pos} \in D_{i}^{pos} \), we find a rationale subgraph with high predicted property and small size \((N_{s} = 20)\):

\[
\text{Find subgraph } S^{i} \subset \mathcal{G}_{i}^{pos} \\
\text{Subject to } r_{i}(S^{i}) \geq \delta_{i}, \quad |S^{i}| \leq N_{s} \text{ and } S^{i} \text{ is connected}
\]  \tag{4.4}
Solving the above problem is challenging because rationale $S^i$ is discrete and the potential number of subgraphs grows exponentially to the size of $G_{pos}^i$. To limit the search space, we have added an additional constraint that $S^i$ has to be a connected subgraph. This assumption is valid in many cases. For instance, rationales for toxicity (i.e., toxicophores) are connected subgraphs in most cases [92]. In this case, we can find a rationale $S^i$ by iteratively removing some peripheral bonds while maintaining its property. Therefore, the key is learning to prune the molecule.

This search problem can be solved by Monte Carlo Tree Search (MCTS) [87]. The root of the search tree is $G_{pos}^i$ and each state $s$ in the search tree is a subgraph derived from a sequence of bond deletions. To ensure that each subgraph is chemically valid and stays connected, we only allow deletion of one peripheral non-aromatic bond or one peripheral ring from each state. As shown in Figure 4-3, a bond or a ring $a$ is called peripheral if $G_{pos}^i$ stays connected after deleting $a$.

During search process, each state $s$ in the search tree contains edges $(s, a)$ for all legal deletions $a$. Following Silver et al. [87], each edge $(s, a)$ stores the following:

- $Q + U$: The sum of the heuristic scores for all states at the current level of the tree.
- $max$: The maximum score at the next level of the tree.
- $r_i(s)$: The return property score of the current state $s$. 

**Figure 4-3:** Illustration of Monte Carlo tree search for molecules. Peripheral bonds and rings are highlighted in red. In the forward pass, the model deletes a peripheral bond or ring from each state which has maximum $Q + U$ value (see Eq. (4.6)). In the backward pass, the model updates the statistics of each state.
statistics:

- $N(s, a)$ is the visit count of deletion $a$, which is used for exploration-exploitation tradeoff in the search process.

- $W(s, a)$ is total action value which indicates how likely the deletion $a$ will lead to a good rationale.

- $Q(s, a)$ is the mean action value: $W(s, a) / N(s, a)$

- $R(s, a) = r_i(s')$ is the predicted property score of the new subgraph $s'$ derived from deleting $a$ from $s$.

Guided by these statistics, MCTS searches for rationales in multiple iterations. Each iteration consists of two phases:

1. **Forward pass**: Select a path $s_0, \ldots, s_L$ from the root $s_0$ to a leaf state $s_L$ with less than $N$ atoms and evaluate its property score $r_i(s_L)$. At each state $s_k$, an deletion $a_k$ is selected according to the statistics in the search tree:

   $$ a_k = \arg\max_a Q(s_k, a) + U(s_k, a) $$

   $$ U(s_k, a) = c_{puct} R(s_k, a) \sqrt{\sum_b N(s_k, b)} / (1 + N(s_k, a)) $$

   where $c_{puct}$ determines the level of exploration. This search strategy is a variant of the PUCT algorithm [81]. It initially prefers to explore deletions with high $R(s, a)$ and low visit count, but asymptotically prefers deletions that are likely to lead to good rationales.

2. **Backward pass**: The edge statistics are updated for each state $s_k$. Specifically,

   $$ N(s_k, a_k) \leftarrow N(s_k, a_k) + 1 \text{ and } W(s_k, a_k) \leftarrow W(s_k, a_k) + r_i(s_L). $$

   In the end, we collect all the leaf states $s$ with $r_i(s) \geq \delta_i$ and add them to the rationale vocabulary $V^i_S$. 

64
Multi-property Rationale  For a set of $M$ properties, we can similarly define its rationale $S^{[M]}$ by imposing $M$ property constraints at the same time, namely

$$\forall i : r_i(S^{[M]}) \geq \delta_i, i = 1, \cdots, M$$

In principle, we can apply MCTS to extract rationales from molecules that satisfy all the property constraints. However, in many cases there are no such molecules available. To this end, we propose to construct multi-property rationales from single-property rationales extracted by MCTS. Specifically, each multi-property rationale $S^{[M]}$ is merged from single-property rationales $S^1, \cdots, S^M$. We merge two rationales $S^i$ and $S^j$ by first finding their maximum common substructure (MCS) and then superposing $S^i$ on $S^j$ so that their MCS coincides (see Figure 4-4).\(^1\) This gives us a set of candidate rationales:

$$C^M_S = \bigcup_{(S^1, \cdots, S^M)} \text{MERGE}(S^1, \cdots, S^M)$$

(4.7)

where $(S^1, \cdots, S^M) \in V^1_S \times \cdots \times V^M_S$. Note that the output of MERGE is a set as there are multiple ways of superposing two rationales. Finally, the vocabulary of multi-property rationales is the subset of $C^M_S$ which satisfies all the property constraints:

$$V^{[M]}_S = \{ S \in C^M_S | r_i(S^{[M]}) \geq \delta_i, \forall i \}$$

(4.8)

For notational convenience, we will denote both single and multi-property rationales as $S$ from now on.

4.1.2 Graph Completion

This module is a variational autoencoder which completes a full molecule $G$ given a rationale $S$. Since each rationale $S$ can be realized into many different molecules, we

\(^1\)The MCS of two (or multiple) rationales is computed using RDKit [54].
introduce a latent variable $z$ to generate diverse outputs:

$$P(G|S) = \int_z P(G|S, z) P(z) dz$$

(4.9)

where $P(z)$ is the prior distribution. Different from standard graph generation, our graph decoder must generate graphs that contain subgraph $S$. Our VAE architecture is adapted from existing atom-by-atom generative models [110, 61] to incorporate the subgraph constraint. For completeness, we present our architecture here:

**Encoder** Our encoder is a message passing network (MPN) which learns the approximate posterior $Q(z|G, S)$ for variational inference. Let $e(a_u)$ be the embedding of atom $u$ with atom type $a_u$, and $e(b_{uv})$ be the embedding of bond $(u, v)$ with bond type $b_{uv}$. The MPN computes atom representations $\{h_v|v \in G\}$.

$$\{h_v\} = MPN_e (G, \{e(a_u)\}, \{e(b_{uv})\})$$

(4.10)

For simplicity, we denote the MPN encoding process as $MPN(\cdot)$, which is detailed in the appendix. The atom vectors are aggregated to represent $G$ as a single vector $h_G = \sum_v h_v$. Finally, we sample latent vector $z_G$ from $Q(z|G, S)$ with mean $\mu(h_G)$.
and log variance \( \Sigma(h_g) \):

\[
z_g = \mu(h_g) + \exp(\Sigma(h_g)) \cdot \epsilon; \quad \epsilon \sim \mathcal{N}(0, I)
\]  \hspace{1cm} (4.11)

**Decoder**   The decoder generates molecule \( G \) according to its breadth-first order. In each step, the model generates a new atom and all its connecting edges. During generation, we maintain a queue \( Q \) that contains frontier nodes in the graph who still have neighbors to be generated. Let \( G_t \) be the partial graph generated till step \( t \). To ensure \( G \) contains \( S \) as subgraph, we set the initial state of \( G_1 = S \) and put all the peripheral atoms of \( S \) to the queue \( Q \) (only peripheral atoms are needed due to the rationale extraction algorithm).

In \( t \text{th} \) generation step, the decoder first runs a MPN over current graph \( G_t \) to compute atom representations \( h_v^{(t)} \):

\[
\{h_v^{(t)}\} = \text{MPN}_d(G_t, \{e(a_u)\}, \{e(b_{uv})\})
\]  \hspace{1cm} (4.12)

The current graph \( G_t \) is represented as the sum of its atom vectors \( h_{G_t} = \sum_{v \in G_t} h_v^{(t)} \). Suppose the first atom in \( Q \) is \( v_t \). The decoder decides to expand \( G_t \) in three steps:

1. Predict whether there will be a new atom attached to \( v_t \):

\[
p_t = \text{sigmoid} (\text{MLP}(h_v^{(t)}, h_{G_t}, z_G))
\]  \hspace{1cm} (4.13)

where MLP(\( \cdot, \cdot, \cdot \)) is a ReLU network whose input is a concatenation of multiple vectors.

2. If \( p_t < 0.5 \), discard \( v_t \) and move on to the next node in \( Q \). Stop generation if \( Q \) is empty. Otherwise, create a new atom \( u_t \) and predict its atom type:

\[
p_{u_t} = \text{softmax} (\text{MLP}(h_v^{(t)}, h_{G_t}, z_G))
\]  \hspace{1cm} (4.14)

3. Predict the bond type between \( u_t \) and other frontier nodes in \( Q = \{q_1, \cdots, q_n\} \) (\( q_1 = v_t \)). Since atoms are generated in breadth-first order, there are no bonds
between \(u_t\) and atoms not in \(Q\).

To fully capture edge dependencies, we predict the bonds between \(u_t\) and atoms in \(Q\) sequentially and update the representation of \(u_t\) when new bonds are added to \(G_t\). In the \(k\)th step, we predict the bond type of \((u_t, q_k)\) as follows:

\[
b_{u_t, q_k} = \text{softmax} \left( \text{MLP} \left( g_{u_t}^{(k-1)}, h_{q_k}^{(t)}, h_{G_t}, z_G \right) \right)
\]

where \(g_{u_t}^{(k-1)}\) is the new representation of \(u_t\) after bonds \{\((u_t, q_1), \cdots, (u_t, q_{k-1})\)\} have been added to \(G_t\):

\[
g_{u_t}^{(k-1)} = \text{MLP} \left( e(a_{u_t}), \sum_{j=1}^{k-1} \text{MLP} \left( h_{q_j}^{(t)}, e(b_{q_j, u_t}) \right) \right)
\]

### 4.1.3 Training Procedure

Our training objective is to maximize the expected reward of generated molecules \(G\), where the reward is an indicator of \(r_i(G) \geq \delta_i\) for all properties \(1 \leq i \leq M\)

\[
\sum_G \mathbb{I} \left[ \bigwedge_i r_i(G) \geq \delta_i \right] P(G) + \lambda \mathbb{H}[P(S)]
\]

(4.16)

We incorporate an entropy regularization term \(\mathbb{H}[P(S)]\) to encourage the model to explore different types of rationales. The rationale distribution \(P(S)\) is a categorical distribution over the rationale vocabulary. Let \(\mathbb{I}[G] = \mathbb{I} \left[ \bigwedge_i r_i(G) \geq \delta_i \right]\). It is easy to show that the optimal \(P(S)\) has a closed form solution:

\[
P(S_k) \propto \exp \left( \frac{1}{\lambda} \sum_G \mathbb{I}[G] P(G|S_k) \right)
\]

(4.17)

The remaining question is how to train graph generator \(P(G|S)\). The generator seeks to produce molecules that are realistic and positive. However, Eq.(4.16) itself does not take into account whether generated molecules are realistic or not. To encourage the model to generate realistic compounds, we train the graph generator in two phases:

- **Pre-training** \(P(G|S)\) using real molecules.
Algorithm 2 Training method with \( n \) property constraints.

1: for \( i = 1 \) to \( M \) do
2: \( V^i_S \leftarrow \) rationales extracted from existing molecules \( \mathcal{D}^i_{\text{pos}} \) positive to property \( i \).
   (see §4.1.1)
3: end for
4: Construct multi-property rationales \( V^M_S \).
5: Pre-train \( P(\mathcal{G}|\mathcal{S}) \) on the pre-training dataset \( \mathcal{D}^{\text{pre}} \).
6: Fine-tune model \( P(\mathcal{G}|\mathcal{S}) \) on \( \mathcal{D}^f \) for \( L \) iterations using policy gradient.
7: Compute \( P(\mathcal{S}) \) based on Eq.(4.17) using fine-tuned model \( P(\mathcal{G}|\mathcal{S}) \).

- Fine-tuning \( P(\mathcal{G}|\mathcal{S}) \) using policy gradient with reward from property predictors.

The overall training algorithm is shown in Algorithm 2.

Pre-training In addition to satisfying all the property constraints, the output of the model should constitute a realistic molecule. For this purpose, we pre-train the graph generator on a large set of molecules from ChEMBL [27]. Each training example is a pair \((\mathcal{S}, \mathcal{G})\), where \( \mathcal{S} \) is a random connected subgraph of a molecule \( \mathcal{G} \) with up to \( N \) atoms. The task is to learn to expand a subgraph into a full molecule. In particular, we train the generative model \( P(\mathcal{G}|\mathcal{S}) \) to maximize the likelihood of the pre-training dataset \( \mathcal{D}^{\text{pre}} = \{(\mathcal{S}_i, \mathcal{G}_i)\}_{i=1}^n \).

Fine-tuning After pre-training, we further fine-tune the graph generator on property-specific rationales \( \mathcal{S} \in \mathcal{V}_S \) in order to maximize Eq.(4.16). The model is fine-tuned through multiple iterations using policy gradient [93]. Let \( P_{\theta^t}(\mathcal{G}|\mathcal{S}) \) be the model trained till \( t^{\text{th}} \) iteration. In each iteration, we perform the following two steps:
1. Initialize the fine-tuning set \( \mathcal{D}^f = \emptyset \). For each rationale \( \mathcal{S}_i \), use the current model to sample \( K \) molecules \( \{\mathcal{G}^1_i, \cdots, \mathcal{G}^K_i\} \sim P_{\theta^t}(\mathcal{G}|\mathcal{S}_i) \). Add \( (\mathcal{G}^k_i, \mathcal{S}_i) \) to set \( \mathcal{D}^f \) if \( \mathcal{G}^k_i \) is predicted to be positive.
2. Update the model \( P_\theta(\mathcal{G}|\mathcal{S}) \) on the fine-tuning set \( \mathcal{D}^f \) using policy gradient method.

After fine-tuning \( P(\mathcal{G}|\mathcal{S}) \), we compute the rationale distribution \( P(\mathcal{S}) \) based on Eq.(4.17).
4.2 Experiments

We evaluate our method (RationaleRL) on molecule design tasks under various combination of property constraints. In our experiments, we consider the following two properties:

- **GNK3β**: Inhibition against glycogen synthase kinase-3 beta. The GNK3β prediction model is trained on the dataset from Li et al. [60], which contains 2665 positives and 50K negative compounds.

- **JNK3**: Inhibition against c-Jun N-terminal kinase-3. The JNK3 prediction model is also trained on the dataset from Li et al. [60] with 740 positives and 50K negatives. Following Li et al. [60], the property prediction model is a random forest using Morgan fingerprint features [79]. We set the positive threshold $\delta_i = 0.5$. For each property, we split its property dataset into 80%, 10% and 10% for training, validation and testing. The test AUROC score is 0.86 for both GSK3β and JNK3.

**Multi-property Constraints** We also consider combinations of property constraints that are biologically relevant:

- **GNK3β + JNK3**: Jointly inhibiting JNK3 and GSK-3β may provide potential benefit for the treatment of Alzheimer’s disease [60]. There exist 316 dual inhibitors already available in the dataset.

- **GNK3β + JNK3 + QED + SA**: We further require the generated dual inhibitors to be drug like and synthetically accessible (i.e., easy to be synthesized). These two properties are quantified by QED [9] and synthetic accessibility (SA) score [25]. In particular, we require $\text{QED} \geq 0.6$ and $\text{SA} \leq 4.0$.

**Evaluation Metrics** Our evaluation measures various aspects of molecule design. For each method, we generate $n = 5000$ molecules and compute the following metrics:

- **Success**: The fraction of sampled molecules predicted to be positive (i.e., satisfying all property constraints). A good model should have a high success rate. Following previous work [71, 109], we only consider the success rate under property predictors,
as it is hard to obtain real property measurements for GSK3β and JNK3.

- **Diversity**: It is also important for a model generate diverse range of positive molecules. To this end, we measure the diversity of generated positive compounds by computing their pairwise molecular distance $\text{sim}(X,Y)$, which is defined as the Tanimoto distance over Morgan fingerprints of two molecules.

$$\text{Diversity} = 1 \frac{2}{n(n-1)} \sum_{X,Y} \text{sim}(X,Y)$$

- **Novelty**: Crucially, a good model should discover novel positive compounds. In this regard, for each generated positive compound $\mathcal{G}$, we find its nearest neighbor $\mathcal{G}_{SNN}$ from positive molecules in the training set. We define the novelty as the fraction of molecules with nearest neighbor similarity lower than 0.4 [71]:

$$\text{Novelty} = \frac{1}{n} \sum_{\mathcal{G}} 1[\text{sim}(\mathcal{G}, \mathcal{G}_{SNN}) < 0.4]$$

**Baselines** We compare our method against the following state-of-the-art generation methods for molecule design:

- **JT-VAE** [40] is a generative model which generate molecules based on substructures such as rings and bonds. The model contains auxiliary property predictors over the VAE latent space. During testing, we perform multi-objective optimization in the latent space using the gradient from the property predictors.

- **REINVENT** [71] is a RL model generating molecules based on their SMILES strings [98]. To generate realistic molecules, their model is pre-trained over one million molecules from ChEMBL and then finetuned under property reward.

- **GCPN** [109] is a RL model which generates molecular graphs atom by atom. It uses GAN [31] to help generate realistic molecules.

- **GVAE-RL** is a graph variational autoencoder which generates molecules atom by atom. The graph VAE architecture is the same as our model, but it generates molecules from scratch without using rationales. The model is pre-trained on the
Table 4.1: Results on molecule design with one property constraint.

<table>
<thead>
<tr>
<th></th>
<th>GSK3β</th>
<th>JNK3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Success</td>
<td>Novelty</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>32.2%</td>
<td>11.8%</td>
</tr>
<tr>
<td>GCPN</td>
<td>42.4%</td>
<td>11.6%</td>
</tr>
<tr>
<td>GVAE-RL</td>
<td>33.2%</td>
<td>76.4%</td>
</tr>
<tr>
<td>REINVENT</td>
<td>99.3%</td>
<td>61.0%</td>
</tr>
<tr>
<td>RationaleRL</td>
<td>100%</td>
<td>53.4%</td>
</tr>
</tbody>
</table>

same ChEMBL dataset and fine-tuned for each property using policy gradient. This is an ablation study to show the importance of using rationales.

Rationales Details of the rationales used in our model:
- GSK3β, JNK3: For single properties, the rationale size is required to be less than 20 atoms. For each positive molecule, we run 20 iteration of MCTS with \( c_{\text{pact}} = 10 \). There are 1234 and 234 rationales for GSK3β and JNK3.
- GSK3β + JNK3: We construct two-property rationales by merging the single-property rationales (see Figure 4-4). For computational efficiency, we only consider rationales \( S_k \) receiving the highest \( P(S_k) \) within each property. This gives us two rationale subsets with \( |V^{GSK3\beta}_S| = 238 \) and \( |V^{JNK3}_S| = 39 \). In total, there are 2394 two-property rationales merged from \( V^{GSK3\beta}_S \) and \( V^{JNK3}_S \).
- GSK3β + JNK3 + QED + SA: We construct four-property rationales by selecting the two-property rationales whose QED \( \geq 0.6 \) and SA \( \leq 4.0 \). There are in total 61 different four-property rationales.

Model Setup We pre-train all the models on the same ChEMBL dataset, which contains 1.02 million training examples. On the four-property generation task, our model is fine-tuned for \( L = 50 \) iterations, with each rationale being expanded for \( K = 200 \) times.
Table 4.2: Molecule design with multiple property constraints. The novelty and diversity of JT-VAE, GVAE-RL and GCPN are not reported due to their low success rate.

<table>
<thead>
<tr>
<th></th>
<th>GSK3β + JNK3</th>
<th>GSK3β + JNK3 + QED + SA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Success</td>
<td>Novelty</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>3.3%</td>
<td>7.9%</td>
</tr>
<tr>
<td>GCPN</td>
<td>3.5%</td>
<td>8.0%</td>
</tr>
<tr>
<td>GVAE-RL</td>
<td>40.7%</td>
<td>80.3%</td>
</tr>
<tr>
<td>REINVENT</td>
<td>97.4%</td>
<td>39.7%</td>
</tr>
<tr>
<td>RationaleRL</td>
<td><strong>100%</strong></td>
<td><strong>97.3%</strong></td>
</tr>
</tbody>
</table>

4.2.1 Results

The results are reported in Table 4.1 and 4.2. On the single-property generation task, both REINVENT and RationaleRL demonstrate nearly perfect success rate since there is only one constraint. On the two-property generation task, our model achieves 100% success rate while maintaining 97.3% novelty and 0.824 diversity score, which is much higher than REINVENT. Meanwhile, GCPN has a low success rate (3.5%) due to reward sparsity.

Table 4.2 shows the results on the four-property generation task, which is the most challenging. The difference between our model the baselines become significantly larger. In fact, GVAE-RL and GCPN completely fail in this task due to reward sparsity. Our model outperforms REINVENT with a wide margin (success rate: 74.8% versus 47.9%; diversity: 0.701 versus 0.621).

Ablation study In all tasks, our method significantly outperforms GVAE-RL, which has the same generative architecture but does not utilize rationales and generates molecules from scratch. Thus we conclude the importance of rationales for multi-property molecular design.

Visualization We further provide visualizations to help understand our model. In Figure 4-6, we plotted a t-SNE [65] plot of the extracted rationales for GSK3β and JNK3. For both properties, rationales mostly cover the chemical space populated
Figure 4-5: Sample rationales of GSK3β (top) and JNK3 (bottom).

Figure 4-6: **Left & middle**: t-SNE plot of the extracted rationales for GSK3β and JNK3. For both properties, rationales mostly covers the chemical space populated by existing positive molecules. **Right**: t-SNE plot of generated GSK3β+JNK3 dual inhibitors.

by existing positive molecules. The generated GSK3β+JNK3 dual inhibitors are distributionally close to the true dual inhibitors in the training set. In Figure 4-5, we show samples of rationales extracted by MCTS. In Figure 4-7, we show examples of generated molecules that satisfy all the four constraints (GSK3β+JNK3+QED+SA).

### 4.2.2 Property Predictor Applicability

Since the reported success rate is based on property predictors, it is possible for the generated molecules to exploit flaws of the property predictors. In particular, the predicted properties may be unreliable when the distribution of generated compounds is very different from the distribution of molecules used to train the property predictors.

This issue is alleviated by our framework because our generated compounds are built from rationales extracted from true positive compounds used to train the property predictors. Therefore, our generated compounds are closer to the true compounds.
Figure 4-7: **Left**: Examples of molecules generated in the GSK3+JNK3+QED+SA task. The model learns to combine two disjoint rationale graphs into a complete molecule. **Right**: Example structural alerts in the toxicity dataset. The ground truth rationale (Azobenzene) is highlighted in red. Our learned rationale almost matches the ground truth (error highlighted in dashed circle).

Table 4.3: Frechet ChemNet Distance (FCD) between generated compounds and true positive molecules in the training set.

<table>
<thead>
<tr>
<th></th>
<th>GSK3+</th>
<th>JNK3</th>
<th>GSK3+JNK3</th>
</tr>
</thead>
<tbody>
<tr>
<td>REINVENT</td>
<td>28.63</td>
<td>25.16</td>
<td>36.75</td>
</tr>
<tr>
<td>RationaleRL</td>
<td><strong>6.88</strong></td>
<td><strong>9.68</strong></td>
<td><strong>25.52</strong></td>
</tr>
</tbody>
</table>

than compounds generated from scratch. To show this, we adopt Frechet ChemNet Distance (FCD) to measure distributional discrepancy between generated molecules and positive compounds in the training set [76]. As shown in Table 4.3, the FCD of RationaleRL is much lower than REINVENT, which means our generated compounds are distributionally closer to the true compounds. As a result, the predicted properties are more reliable for molecules generated by our model.

### 4.2.3 Faithfulness of Rationales

While our rationales are mainly extracted for generation, it is also important for them to be chemically relevant. In other words, the extracted rationales should accurately explain the property of interest. As there is no ground truth rationales for JNK3 and GSK3β, we turn to an auxiliary toxicity dataset for evaluating rationale quality.

**Data** The toxicity dataset contains 125K molecules randomly selected from ChEMBL. Each molecule is labeled as toxic if it contains structural alerts [92] — chemical sub-
Table 4.4: Rationale accuracy on the toxicity dataset. Our rationales are more faithful to the property of interest.

<table>
<thead>
<tr>
<th>Method</th>
<th>Partial Match</th>
<th>Exact Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Gradient</td>
<td>0.857</td>
<td>39.4%</td>
</tr>
<tr>
<td>MCTS Rationale</td>
<td><strong>0.861</strong></td>
<td><strong>46.0%</strong></td>
</tr>
</tbody>
</table>

Structures that is correlated with human or environmental hazards (see Figure 4-7). Structural alerts used in our paper are from surechembl.org/knowledgebase/169485-non-medchem-friendly-smarts. Under this setup, the structural alerts are ground truth rationales and we evaluate how often the extracted rationales match them. The dataset is split into 105K for training and 20K for testing. In total, there are 26.5K toxic molecules and 164 types of structural alerts. We train a graph convolutional network [107] to predict toxicity, which achieves 0.99 test AUROC.

**Results**  We compare our MCTS based rationale extraction with integrated gradient [91], which has been applied to explain property prediction models [67]. We report two metrics: partial match AUC (attribution AUC metric used in McCloskey et al. [67]) and exact match accuracy which measures how often a rationale graph exactly matches the true rationale in the molecule. As shown in Table 4.4, our method significantly outperforms the baseline in terms of exact matching. The extracted rationales has decent overlap with true rationales, with 0.86 partial match on average. Therefore, our model is capable of finding rationales faithful to the properties.

### 4.3 Summary

In this chapter, we developed a rationale based generative model for molecular design. Our model generates molecules in two phases: 1) identifying rationales whose presence indicate strong positive signals for each property; 2) expanding rationale graphs into molecules using graph generative models and fine-tuning it towards desired combination of properties. Our model demonstrates strong improvement over prior reinforcement learning methods in various tasks.
Chapter 5

Application: Antibiotic Discovery

Since the discovery of penicillin, antibiotics have become the cornerstone of modern medicine. However, the continued efficacy of these essential drugs – of which we have on the order of a couple hundred in clinical use – is uncertain due to the persistent global dissemination of antibiotic-resistance determinants. Moreover, the decreasing development of new antibiotics in the private sector that has resulted from a lack of economic incentives is exacerbating this already dire problem; only ten molecules, all from existing antibiotic classes, have been approved by the FDA since 2014. Indeed, without immediate action to discover and develop new antibiotics, it is projected that deaths attributable to resistant infections will reach 10 million per year by 2050.

Unfortunately, the discovery of new antibiotics is becoming increasingly difficult. Indeed, natural product discovery is now plagued by the dereplication problem, wherein the same molecules are being repeatedly discovered from discrete species that inhabit similar ecological niches. Moreover, given the rapid expansion of chemical spaces that are accessible by the derivatization of complex scaffolds, engineering next-generation versions of existing antibiotics can result in substantially more failures than leads. With these challenges, many contemporary antibiotic discovery programs have turned to screening large synthetic chemical libraries generated by high-throughput combinatorial synthesis. However, these libraries, which can contain hundreds of thousands to a few million molecules, are often prohibitively costly to curate, limited in chemical diversity, and fail to reflect the chemistry that is inherent
to antibiotic molecules. Since the implementation of high-throughput screening in the late 1980s, no new clinical antibiotics have been discovered using this approach.

Clearly, new approaches to antibiotic discovery are critically needed, in order to increase the rate at which new antibiotics are identified and simultaneously decrease the associated cost of early lead discovery. Given recent advancements in machine learning, the antibiotic discovery field is now ripe for the application of algorithmic solutions for molecular property prediction to identify novel structural classes of antibiotics, as well as new analogs of existing scaffolds. Indeed, adopting methodologies that allow early drug discovery to be performed largely in silico enables the exploration of vast chemical spaces that is beyond the reach of current experimental approaches due to prohibitive cost, labor, and time constraints.

The idea of analytical exploration in drug design is not new. Decades of prior work in chemoinformatics has developed models for molecular property prediction, including both bioactivity and ADME properties. However, the accuracy of these models has been insufficient to substantially change the traditional drug discovery pipeline. With recent algorithmic advancements in modelling neural network-based molecular representations, we now have the opportunity to change the paradigm of drug discovery. A significant development relates to how molecules are represented; traditionally, molecules were represented by their fingerprint vectors, which reflected the presence or absence of certain functional groups in the molecule, or by descriptors that include computable molecular properties and require expert knowledge to construct. Even though the mapping from these representations to properties was learned automatically, the fingerprints and descriptors themselves were designed manually. The innovation of neural network approaches lies in their ability to learn this representation automatically, mapping molecules into continuous vectors which are subsequently used to predict their properties. This design results in molecular representations that are highly attuned to the desired property, yielding significant gains in property prediction accuracy over manually crafted representations.

While neural network models narrowed the performance gap between analytical and experimental approaches, a difference still exists. In this chapter, we demonstrate
Modern approaches to antibiotic discovery often include screening large chemical libraries for those that elicit a phenotype of interest. These screens, which are limited in chemical diversity, and fail to reflect the chemistry that is beyond the reach of current experimental approaches. As a result, the rate at which new antibiotics are identified and simultaneously decrease the associated cost of early lead discovery has been insufficient to substantially change the traditional drug discovery pipeline. With recent algorithmic advancements in chemoinformatics, recent algorithmic advancements in chemoinformatics have developed novel approaches to antibiotic discovery that allow early drug discovery to be performed largely in silico. An important development relates to how machine learning approaches afford the opportunity to rapidly and inexpensively explore vast chemical spaces (Figure 5-1). Our approach consists of three stages. First, we trained a deep neural network model using a collection of 2,335 diverse molecules for those that inhibited the growth of \textit{E. coli}. After ranking the candidates according to the model’s predicted score, we select a list of promising candidates based on a pre-defined threshold.

Figure 5-1: Machine learning in antibiotic discovery. We first train our Chemprop model using a collection of a few thousand diverse molecules for those that inhibited the growth of \textit{E. coli}. Next, we apply the resulting model to multiple discrete chemical libraries, including a collection of 107,349,233 molecules, to identify potential lead compounds with activity against \textit{E. coli}. After ranking the candidates according to the model’s predicted score, we lastly selected a list of promising candidates based on a pre-specified prediction score threshold, structural diversity, and commercial/synthesis availability.

Through this approach, we identified the \textit{c-Jun} N-terminal kinase inhibitor SU3327 (renamed halicin herein), which is structurally divergent from conventional antibiotics, as a potent inhibitor of \textit{E. coli} growth. Importantly, halicin shows efficacy against \textit{Clostridioides difficile} and pan-resistant \textit{Acinetobacter baumannii} infections.
in murine models. Of note, the World Health Organization has designated *A. baumannii* as the highest priority pathogen against which new antibiotics are urgently required, due to its propensity to acquire antibiotic-resistance determinants at high frequency and the broad spectrum of diseases it can cause, particularly in wounded soldiers. Altogether, this work highlights the significant impact that machine learning can have on early antibiotic discovery efforts by increasing the accuracy rate of lead compound identification and decreasing the cost of preclinical screening efforts.

### 5.1 Identification of halicin via Chemprop

To collect training data, we screened for growth inhibition against *E. coli* using a widely-available FDA-approved drug library consisting of 1,760 molecules of diverse structure and function. To supplement these molecules and further increase chemical diversity, we included an additional 800 natural products isolated from plant, animal, and microbial sources, resulting in a primary training set of 2,335 molecules (Figure 5-2A). Using 80% growth inhibition as a hit cut-off, this primary screen resulted in the identification of 120 molecules with growth inhibitory activity against *E. coli*.

Next, all 2,335 compounds from the primary training dataset were binarized as hit or non-hit. After binarization, we used this data to train a Chemprop model that predicts the probability of whether a new compound will inhibit the growth of *E. coli* based on its molecular structure. The resulting model achieved an ROC-AUC of 0.896 on the test data (Figure 5-2B).

After model development and optimization using our training dataset of 2,335 molecules, we subsequently applied an ensemble of models trained on all twenty folds to identify potential antibacterial molecules from the Drug Repurposing Hub housed at the Broad Institute. Here, prediction scores for each compound were determined, molecules were ranked based on their probability of displaying growth inhibition against *E. coli*, and compounds with molecular graphs common between the training dataset and the Drug Repurposing Hub were removed (Figure 5-2C). Next, we curated the 99 molecules unique to the Drug Repurposing Hub that were most strongly
predicted to display antibacterial properties and empirically tested these for growth inhibition. We observed that 51 of the 99 predicted molecules (51.5% true positive rate) displayed growth inhibition against *E. coli* when empirically assayed (Figure 5-2D). Importantly, when considering these 99 molecules simultaneously, higher prediction scores correlated with a greater probability of growth inhibition (Figure 5-2E). Furthermore, empirically testing the lowest predicted 63 molecules that were unique to the Broad Repurposing Hub revealed that only two of these compounds displayed growth inhibitory activity (3.2% false negative rate; Figure 5-2F). Collectively, these data highlight the accuracy of our model in assigning high prediction scores to compounds more likely to display antibacterial properties, and low prediction scores to non-antibiotic molecules.

After identifying the 51 molecules that displayed growth inhibition against *E. coli*, we prioritized these based on clinical phase of investigation, structural similarity to molecules in the primary training dataset, and predicted toxicity using a deep neural network model trained on the ClinTox database. Specifically, we prioritized predicted compounds with unconventional biological functions; those in preclinical or phase 1, 2, and 3 studies; those with low structural similarity to training set molecules; and those with low predicted toxicity. The predicted compound that satisfied all of these criteria was the c-Jun N-terminal kinase inhibitor SU3327 (renamed halicin herein). Excitingly, halicin, which is structurally most similar to a family of nitro-containing antiparasitic compounds (Tanimoto similarity 0.37; Figure 5-2G, 5-2H) and the antibiotic metronidazole (Tanimoto similarity 0.21), displayed excellent growth inhibitory activity against *E. coli* when tested in dose, achieving a minimum inhibitory concentration (MIC) of 2 µg/ml in rich growth conditions (Figure 5-2I).

### 5.1.1 Halicin displays efficacy in murine models of infection

Given that halicin displays broad-spectrum bactericidal activity and is not highly susceptible to plasmid-borne, antibiotic-resistance elements or de novo resistance mutations at high frequency, we next asked whether this compound might have utility as a clinical antibiotic. To initially understand the potential clinical utility of this
molecule, we first tested the efficacy of halicin in a murine wound model of *A. baumannii* infection.
On the dorsal surface of neutropenic Balb/c mice, we established a 2 cm² wound and infected with 2.5x10⁵ CFU of A. baumannii strain 288 acquired from the Centers for Disease Control and Prevention (CDC). This strain is non-sensitive to any clinical antibiotics generally used for treatment of A. baumannii, and therefore represents a pan-resistant isolate. Importantly, halicin displayed potent growth inhibition against this strain in vitro (MIC = 1 µg/ml; Figure 5-3A) and was able to sterilize A. baumannii 288 cells residing in metabolically repressed, antibiotic-tolerant conditions (Figure 5-3B, Figure S4A, S4B). After one hr of infection establishment, mice were treated with Glaxal Base Moisturizing Cream supplemented with vehicle (0.5% DMSO) or halicin (0.5% w/v). Mice were then treated after 4 hr, 8 hr, 12 hr, 20 hr, and 24 hr of infection, and mice were sacrificed at 25 hr post-infection. We observed that wound-carrying capacity had reached 10⁸ CFU/g in the vehicle control group, whereas 5 of the 6 mice treated with halicin contained <10³ CFU/g (below the limit of detection) and one mouse contained 10⁵ CFU/g.

After showing that halicin displayed efficacy against A. baumannii in a murine wound model, we next sought to investigate whether this molecule may have utility against a phylogenetically discrete pathogen that is increasingly becoming burdensome to healthcare systems – namely, C. difficile. This spore-forming anaerobe causes pseudomembranous colitis, often as a result of dysbiosis following systemic antibiotic administration. Metronidazole or vancomycin are first-line treatments, with failure resulting from antibiotic resistance and/or the presence of metabolically dormant cells. In such cases of recurrent infection, fecal bacteriotherapy is required to re-establish the normal colonic microbiota to outcompete C. difficile cells, which is substantially more invasive than antibiotic therapy.

Towards understanding the efficacy of halicin against C. difficile infections, we assayed for the ability of this molecule to inhibit the growth of C. difficile strain 630 in vitro and observed an MIC of 0.5 µg/ml (Figure 5-3D). To establish the murine infection, C57BL/6 mice were administered intraperitoneal injections of ampicillin (200 mg/kg) every 24 hr for 72 hr. Mice were then given 24 hr to recover, and subsequently administered 5x10³ spores of C. difficile 630 via oral gavage. Beginning 24 hr after
Figure 5-3: Halicin displays efficacy in murine models of infection. (A) Growth inhibition of pan-resistant *A. baumannii* CDC288 by halicin. Shown is the mean of two biological replicates. (B) Killing of *A. baumannii* CDC288 in the presence of varying concentrations of halicin after 2 hr (blue), 4 hr (cyan), 6 hr (green), and 8 hr (red). (C) In a wound infection model, mice were infected with *A. baumannii* CDC288 for 1 hr and treated with either vehicle or halicin periodically over 24 hr. Black lines represent the geometric mean of the bacterial load for each treatment group. (D) Growth inhibition of *C. difficile* 630 by halicin. (E) Experimental design for *C. difficile* infection and treatment. (F) Bacterial load of *C. difficile* 630 in feces of infected mice. Metronidazole did not result in enhanced rates of clearance relative to vehicle controls. Halicin-treated mice displayed sterilization beginning at 72 hr after treatment, with 100% of mice being free of infection at 96 hr.

*C. difficile* gavage, mice were gavaged with antibiotics (50 mg/kg metronidazole or 15 mg/kg halicin) or vehicle (10% PEG 300) every 24 hr for five days, and fecal samples were collected to quantify *C. difficile* load (Figure 5-3E). Excitingly, we observed that halicin resulted in *C. difficile* clearance from feces at a greater rate than vehicle or the antibiotic metronidazole (Figure 5-3F), which is not only a first-line treatment for *C. difficile* infection, but also the antibiotic most similar to halicin based on Tanimoto score (Figure 1H). Indeed, halicin resulted in sterilization of 3 out of 4 mice after 72 hr of treatment, and 4 out of 4 mice after 96 hr of treatment, suggesting that this compound may represent a new structural class of antibiotics with efficacy against this notoriously difficult bacterial pathogen.
5.2 Summary

Since 1987, no new clinical antibiotics have been discovered using traditional high-throughput screening methods [88]. Finding new antibiotics is an urgent task since over 700,000 people die each year due to drug-resistant diseases. Together with biologists in the Broad Institute, we trained a GNN to predict compounds’ antibacterial activity and identified an antibacterial compound, Halicin, that was structurally distant from known antibiotics [90] and showed strong efficacy against multiple pathogens in mice. This work has been covered in MIT News, Nature, Science, Guardian, BBC News, GovTech, Innovators, Financial Times, CBS Boston, Wbur, and STAT. Halicin is now listed as an investigational new drug by the U.S. Food and Drug Administration (FDA) [link].
Chapter 6

Application: COVID Drug Discovery

COVID-19 has caused more than 2.5 million deaths worldwide. It is imperative that we develop therapies that can mitigate the effect of the disease. While searching for individual drugs for this purpose has been met with difficulties, synergistic drug combinations offer a promising alternative. Combination therapies have shown to be more effective than single drugs for multiple diseases such as HIV [94] and infections caused by Mycobacterium tuberculosis [108]. Synergistic combinations can improve both therapeutic potency and efficacy, either achieving stronger therapeutic effects and/or decreasing the required dose, thereby reducing side effects. To address the COVID-19 pandemic, finding useful combinations of approved molecules has an additional benefit over discovering and developing an entirely novel single-agent therapy: reduced time to clinical adoption. Approved drugs are readily available at scale, have well-studied toxicity profiles, and may be used off-label in extenuating circumstances. Collectively, these considerations highlight the benefits of discovering new synergistic drug combinations for treating COVID-19.

Exploring the space of combinations via high-throughput screening of even mid-sized chemical libraries is prohibitive due to the exceedingly large number of unique chemical combinations. Therefore, in silico screening based on various computational methods is an appealing alternative [14, 68]. For example, Bobrowski et al. [11] used knowledge-based methods to generate candidate drug combinations and experimentally validated their antiviral SARS-CoV-2 synergies. Cheng et al. [17] developed a
biological network proximity measure to predict drug synergy for hypertension and cancer. Prior work has applied various machine learning techniques for synergy prediction [58, 102, 57], including deep learning approaches [75, 105, 86]. Indeed, Preuer et al. [75] trained a deep neural network on a large oncology screen [72] and demonstrated the advantage of deep learning over standard machine learning models such as random forests and support vector machines.

Unfortunately, there are two primary challenges that prevent one from applying existing deep learning approaches to predict therapeutic chemical combinations for emerging pathogens such as SARS-CoV-2. First, deep neural networks require a large amount of training data with measured synergy scores. While such data are readily available for some diseases such as cancer [72] (more than 20,000 combinations), the amount of SARS-CoV-2 drug combination data [11] is very limited (less than 200 combinations). Second, even the largest combination screen for cancer [36] covers only around 100 different molecules, since the number of pairwise combinations grows quadratically. This significantly limits a model’s ability to generalize to new chemical spaces outside of the training set. Therefore, we posit that a model should incorporate additional information besides molecular structures in order to accurately predict new synergistic drug combinations.

The main contribution of this chapter is a new deep learning architecture, which we call ComboNet, that jointly models molecular structure, as well as biological targets, for the purpose of predicting synergistic drug combinations. Our hypothesis is that by explicitly modeling interactions between drugs and biological targets, we can significantly decrease the dependence on combination synergy data. Indeed, uniquely relative to previous approaches [53, 14, 57, 68, 37] using drug-target interaction (DTI) as fixed descriptors, ComboNet learns to predict DTI from molecular structures, which is advantageous since a large proportion of compounds in our training dataset have incomplete DTI information.

The ComboNet architecture consists of two components. The first component is a graph convolutional network (GCN) [107] that learns a continuous representation of a molecule. This representation contains both structural features of the molecule
and predicted targets (i.e., what biological targets may interact with the molecule). Specifically, the biological targets in our training dataset include SARS-CoV-2 3CL protease, angiotensin-converting enzyme 2 (ACE2), and 31 host targets that physically interact with viral proteins [32]. The GCN learns to predict the most likely targets using data collected from ChEMBL [27] and NCATS OpenData portal [12]. The 31 host targets included in ComboNet are only a subset of the 332 targets that physically interact with SARS-CoV-2 [32]. Other targets were excluded because they lack available DTI data.

The second component of ComboNet models target-disease association. It is a linear function that learns how biological targets and structural features of molecules are related to antiviral activity and synergy. It is trained on NCATS single-agent SARS-CoV-2 cytopathic effect (CPE) assay data [15] and available drug combination assays [1]. In short, ComboNet predicts drug combination synergy by modeling structural features of both compounds and biological targets.

Herein, we evaluated ComboNet on a hold-out test set [11] of 71 drug combinations with measured anti-SARS-CoV-2 synergy in vitro. Our model achieves 0.82 receiver operating characteristic-area under the curve (ROC-AUC) using approximately 200 drug combination data for training, with specificity = 0.75 and sensitivity = 0.80, respectively. We additionally applied ComboNet to in silico repurposing of existing drugs and experimentally tested 30 drug combinations. From this empirical set of 30 tested combinations, we discovered two novel drug combinations (remdesivir and reserpine; remdesivir and IQ-1S) with strong synergy in vitro. In general, ComboNet represents an advance toward predicting novel chemical-chemical synergy for instances where minimal combination training data exist.

### 6.1 ComboNet Architecture

Figure 6-1 provides an overview of the network architecture. It is composed of a DTI network and target-disease association network. These are trained to accomplish three tasks: 1) predict the interaction between a drug and a set of $K$ biological
targets \{t_1, \cdots, t_K\} related to the disease of interest; 2) predict a drug’s intrinsic antiviral activity; 3) predict the synergy of two drugs. The latter two tasks depend on both the predicted biological targets and structural features of input molecules.

**Drug-target interaction prediction** We first train the DTI network to predict whether a drug binds to a biological target. The DTI training data is compiled from ChEMBL, including \(K\) biological targets related to the the indication or pathogen of interest – in our case SARS-CoV-2. Each DTI dataset consists of a list of molecules
and their binary DTI labels (positive/negative). A positive label means the binding affinity (e.g. EC$_{50}$) of a molecule to a target is below a certain threshold. In terms of SARS-CoV-2 biological targets, we consider both viral proteases and host proteins involved in viral infection. The replication of SARS-CoV-2 requires the processing of a chymotrypsin-like protease (3CLpro). It is known that SARS-CoV-2 entry into host cells depends on ACE2 and TMPRSS2 [35]. Furthermore, Gordon et al. [32] identified 332 human proteins that physically interact with SARS-CoV-2.

The DTI training data for these targets are collected from various sources. NCATS conducted a high-throughput screen of 10,442 compounds with measured 3CLpro enzymatic activity [115]. NCATS also released two high-throughput screens of 3,285 molecules with measured ACE2 enzymatic activity [3] and inhibition against Spike-ACE2 protein-protein interaction [34]. Among the 332 human proteins, we selected 31 targets based on their DTI data availability in ChEMBL [27]. Other targets were excluded due to a lack of existing DTI data.

We parameterize the DTI network as a directional MPNN (DMPNN) using the Chemprop software [107]. Each compound is characterized as a graph, whose nodes and edges correspond to its atoms and bonds. The DMPNN applies a series of message passing steps to aggregate information from neighboring atoms and bonds to build a continuous vector representation $z_A$ of drug $A$. We divide $z_A$ into two vectors $z_A = z_{covid}^A \oplus z_{struct}^A$ (\oplus represents vector concatenation). $z_{covid}^A$ represents the predicted interaction between drug $A$ and biological targets related to SARS-CoV-2. Each element $z_{covid}^{A,i} \in [0, 1]$ indicates the probability of drug $A$ interacting with a target $t_i$. $z_{struct}^A$ represents the structural features of drug $A$ learned automatically from its molecular structure. Each element $z_{struct}^{A,i} \in [0, 1]$ is output from a sigmoid activation function.

We propose to include these structural features to increase the modeling power when target information is incomplete. Among the 332 human proteins, only 31 of them have associated DTI data and the other 300 targets cannot be included in the model. Moreover, our biological understanding of emerging pathogens is continuously involving. Including these structural features allows the model to complement any
missing biological information needed for antiviral activity and synergy prediction. Indeed, we observe a decrease in synergy prediction accuracy when these structural features are removed (Figure 6-2).

**Single-agent activity prediction** We then train the entire ComboNet to predict the antiviral activity of single drugs. The single-agent training set is a collection of molecules with their antiviral activity labels (positive/negative). A positive label indicates that a drug inhibits the viral replication. The DTI network is trained to extract useful features from molecular structures for antiviral activity prediction. The target-disease association network $f$ learns how to associate the biological targets and learned structural features of molecules to antiviral activity. It is parameterized as a simple linear layer with sigmoid activation function $\sigma(\cdot)$. The antiviral activity of a single drug $A$ is predicted as

$$p_A = f(z_A) = \sigma(w^\top z_A + b) \quad (6.1)$$

The model is trained on SARS-CoV-2 single-agent antiviral activity data using a CPE assay [15] in VeroE6 cells. It contains approximately 8,800 compounds with 320 hits ($EC_{50} \leq 10\mu M$).

**Synergy prediction** Lastly, we train the entire ComboNet to predict drug-drug synergy. The training set for this task is a list of pairwise drug combinations and their synergy labels (synergistic/non-synergistic). Different from the previous two tasks, inputs to the model become two molecules instead of one. Given a pairwise drug combination $(A, B)$, the DTI network outputs a continuous vector representation $z_{AB}$ by combining their individual representations $z_A, z_B$. The combined vector characterizes how the two drugs interact via their individual biological targets. It is then fed into the target-disease association network to predict its synergy based on Bliss scores [10].

We adopt the Bliss score [10] to predict synergy of a drug combination (Figure 6-1b). Suppose the individual antiviral effect of drugs $A$ and $B$ are $p_A, p_B$. The
expected activity of a combination \((A, B)\) is defined as \(e_{AB} = p_A + p_B - p_Ap_B\). A drug combination is determined to be synergistic if its actual activity \(p_{AB} > e_{AB}\). Thus, we define its synergy score as

\[
 s_{AB} = p_{AB} - e_{AB} = p_{AB} - p_A - p_B + p_Ap_B
\]

where the antiviral activity \(p_{AB}\) of a drug combination \((A, B)\) is predicted as

\[
 p_{AB} = f(z_{AB}) = \sigma(w^T z_{AB} + b)
\]

The remaining question is how to compute the molecular representation \(z_{AB}\) for a drug combination. Since we model drug synergy using Bliss scores, we introduce a Bliss layer to compute the representation of a drug combination. Let \(z_A, z_B\) be the learned features of drug \(A\) and \(B\). The representation \(z_{AB}\) of a combination \((A, B)\) is defined as

\[
 z_{AB} = z_A + z_B - z_A \odot z_B
\]

where \(\odot\) stands for element-wise multiplication. With this aggregation scheme, a drug combination benefits the most when they interact with different targets. For instance, suppose only drug \(A\) interacts with target \(t_i\) (e.g. \(z_{A,i} = 0.9, z_{B,i} = 0\)), the combination still interacts with target \(t_i\) as \(z_{AB,i} = 0.9\).

The SARS-CoV-2 drug combination training data came from three data sources. NCATS performed two combination assays [11, 1] in VeroE6 cells, which contained 160 two-drug combinations in total. Riva et al. [78] also analyzed synergy between remdesivir and 20 hit molecules identified from their high-throughput screen in VeroE6 cells.

**Multi-disease training** The drug combination data of emerging pathogens are inherently limited. To address the low-resource challenge, it is helpful to utilize data from multiple diseases as a source of supervision. For example, we can utilize existing HIV drug combination data to improve the model performance. Indeed, prior work [32] has shown significant interactome similarity between HIV and SARS-CoV-
2. With multi-disease training, the molecular representation \( z_A \) contains three parts \( z_A^{\text{covid}}, z_A^{\text{hiv}}, z_A^{\text{struct}} \). Features in \( z_A^{\text{hiv}} \) correspond to the interaction between drug \( A \) and HIV-relevant biological targets. Since each disease operates on different targets, we create two target-disease association networks \( f^{\text{covid}} \) and \( f^{\text{hiv}} \). The SARS-CoV-2 and HIV antiviral activity is computed as 
\[
  f^{\text{covid}}(z_A^{\text{covid}} \oplus z_A^{\text{struct}}) \quad \text{and} \quad f^{\text{hiv}}(z_A^{\text{hiv}} \oplus z_A^{\text{struct}}),
\]
respectively (\( \oplus \) represents vector concatenation).

In terms of HIV targets, we consider three viral proteases (HIV-1 protease, integrase, and reverse transcriptase) and three host proteins involved in viral entry (CCR5, CXCR4, and CD4) [5]. We compiled DTI data for these six targets from ChEMBL. The HIV single-agent activity data came from an NCI anti-HIV assay [2]. It includes approximately 35,000 compounds with 309 active hits (EC\(_{50} \leq 1 \mu\text{M})\). The HIV combination data [94] contains 114 drug combinations with measured synergy outcomes against HIV.

**Training objective** The ComboNet is trained to minimize a weighted average of three losses \( L = \lambda_{\text{DTI}} \ell_{\text{DTI}} + \lambda_S \ell_S + \ell_C \), where \( \lambda_{\text{DTI}}, \lambda_S \) are hyperparameters and \( \ell_{\text{DTI}}, \ell_S, \ell_C \) are the training losses on the DTI, single-agent and drug combination data. The weighted loss allows us to optimize the entire model with a single forward-backward pass in each gradient update.

6.2 Model evaluation

We evaluate our model’s performance at predicting SARS-CoV-2 chemical synergy. Our training, validation, and test sets are summarized in Figure 6-2a. Specifically, our validation set contains 20 drug combinations from Riva et al. [78] and our test set contains 71 drug combinations from Bobrowski et al. [11]. The training set contains 88 SARS-CoV-2 drug combinations from NCATS [1] as well as the DTI and single-agent antiviral activity data for SARS-CoV-2 and HIV. We note that 63.4% (45/71) of the drug combinations in the test set involved at least one new drug that did not appear in the training set.
Figure 6-2: *In silico* evaluation of ComboNet

**a)** The training, validation, and test set composition for SARS-CoV-2. **b)** Results on SARS-CoV-2 drug combination test set. Our full ComboNet model outperforms all other baselines. **c)** ROC-AUC plot of ComboNet ensemble on the entire test set. **d)** ROC-AUC plot of ComboNet ensemble on the hard drug combinations with at least one new drug. **e)** Statistical characteristics of ComboNet ensemble for all the datasets, where “screen” refers to the top 30 candidates we experimentally tested.

**Baselines** To test the effectiveness of ComboNet, we compare our approach with seven baselines: a random forest (RF), support vector machine (SVM), feed-forward neural network (DNN), and four state-of-the-art graph neural network architectures including message-passing neural network (MPNN) [28], DMPNN [107], graph attention network (GAT) [97], and AttentiveFP [106]. All baselines are trained on SARS-CoV-2 combination data only, while ComboNet is trained on additional HIV, DTI, and single-agent data.

The input to RF and SVM is the sum of ECFP4 fingerprints of the two drugs so that the model is permutation invariant, i.e. it outputs the same value for drug pairs \((A, B)\) and \((B, A)\). The DNN and graph neural network baselines predicts the synergy of drug \(A\) and \(B\) as \(p_{AB} = g(\phi(A) + \phi(B))\), where \(g\) is a feed-forward network with one hidden layer. For DNN, the input to \(\phi\) the ECFP4 fingerprint of drug \(A\) and \(B\). For MPNN, DMPNN, GAT, and AttentiveFP, the input to \(\phi\) is the molecular
graph of $A$ and $B$. We sum the two vectors $\phi(A) + \phi(B)$ instead of concatenating them so that the model is permutation invariant. We also use the same $\phi$ to encode drug $A$ and $B$ to ensure permutation invariance.

Moreover, we evaluate the following ComboNet variants to study the importance of different training data:

- ComboNet (no HIV): A model trained without HIV data.
- ComboNet (no DTI): A model trained on all training data except the DTI data.
- ComboNet (no struct): A model trained on all training data but the structural features are disabled.
- ComboNet: A model trained on all the training data.

**Synergy prediction accuracy**  The results of synergy prediction are shown in Figure 6-2b. We compute the ROC-AUC of each method averaged across five independent runs. The test ROC-AUC of ComboNet is $0.773\pm0.064$, which is significantly higher than the RF, SVM, DNN, and DMPNN baselines. Among all baseline methods, AttentiveFP achieves the best ROC-AUC of $0.621\pm0.050$. The Wilcoxon p-value between ComboNet and AttentiveFP is 0.028.

We then took five independently trained ComboNet models as an ensemble model. Ensembling is a standard machine learning technique to improve model performance, where we train five copies of ComboNet with different random initialization and average their predictions. The ensemble model achieves 0.821 ROC-AUC on the test set (Figure 6-2c), which is higher than a single ComboNet model.

We further adopt a “compounds out” strategy [111] to evaluate the model in terms of novel combination prediction. Specifically, we select 45 combinations from the test set that involve at least one new drug that has not appeared in the training set. The average Tanimoto similarity between these 45 combinations and the training set is low – approximately 0.22. Thus, these instances are significantly harder to predict and require the model to extrapolate beyond drugs in the training set. Remarkably, the ensemble model achieves similar performance on these difficult instances, with
ROC-AUC = 0.815 (Figure 6-2d). This result shows that ComboNet generalizes well to novel drug combinations.

We further conduct ablation studies to understand the importance of different model design choices (Figure 6-2b). We find the test ROC-AUC decreases to $0.658 \pm 0.079$ if the HIV data are removed (ComboNet, no HIV). Likewise, the test ROC-AUC drops to $0.706 \pm 0.088$ when we remove the DTI data (ComboNet, no DTI). This confirms the advantage of training with DTI data and additional viral diseases. We also find the test ROC-AUC decreases to $0.701 \pm 0.017$ if we remove the structural features (ComboNet, no struct) (Figure 6-2b). This highlights the advantage of using structural features to complement missing biological targets.

### 6.3 Screening predicted drug combinations

We applied the ComboNet ensemble to predict the synergy of novel drug combinations in the NCATS compound library. We considered pairwise combinations between 153 relatively potent drugs with IC$_{50}$ less than 30$\mu$M. This resulted in approximately 11,600 combinations, which were ranked according to predicted synergy scores. We selected the top 30 candidates and experimentally tested them in a SARS-CoV-2 CPE assay, which measures the ability of compounds to reverse the viral induced CPE in Vero E6 host cells. In this assay, viral infection and replication lead to a loss of host cell viability. Compounds with antiviral activity protect cells from the virus, thereby maintaining viability.

The synergy of these combinations was assessed based on the Delta Bliss Sum Negative (DBSumNeg) score [69]. Excitingly, from this set of 30 empirically tested predicted combinations, we identified two novel drug combinations – remdesivir and reserpine, as well as remdesivir and IQ-1S (Figure 6-3a) – with strong synergy in vitro (DBSumNeg $\leq -5$). Importantly, we also verified that these two drug combinations have low cytotoxicity (Figure 3b). Their dose response and bliss synergy matrices are reported in Figure 6-3c-d. The ComboNet ranking of the 30 drug combinations is reported in Figure 6-3e. As visualized in Figure 6-3f, the chemical space explored
**Figure 6-3: Discovery of novel synergistic drug combinations.**

a) Two new drug combinations are discovered by our model: Remdesivir + Reserpine and Remdesivir + IQ-1S.

b) Host cell viability matrices show the two drug combinations have low cytotoxicity.

c) Dose response and bliss synergy matrices of Remdesivir + Reserpine. Numbers in the dose response matrix stands for viral infection rate. Numbers in the bliss synergy matrix stands for synergy score. Both are the lower the better.

d) Dose response and bliss synergy matrices of Remdesivir + IQ-1S.

e) The correlation between predicted ranking and DBSumNeg score (lower DBSumNeg means more synergistic).

f) t-SNE visualization [65] of the chemical space explored across the training set, test set, and experimentally validated combinations. Across the training/test sets and experimentally validated combinations are similar.

Reserpine is an FDA-approved drug primarily used as an anti-hypertensive. It has a moderate potency against SARS-CoV-2, with IC$_{50}$=11.2µM in Vero E6 cells [15] and IC$_{50}$=6.4µM in HeLa-ACE2 cells [38]. IQ-1S is a JNK inhibitor with Kd=87nM, 360nM, and 390nM for JNK3, JNK2, and JNK1 respectively. It demonstrated an IC$_{50}$=6.3µM against SARS-CoV-2 in a Vero E6 cell CPE assay.
6.4 Summary

In this chapter, we developed ComboNet for chemical synergy prediction against SARS-CoV-2. ComboNet has two components: a drug-target interaction (DTI) network and a target-disease association network. The model architecture is designed to utilize additional drug-target interaction data and single-agent antiviral activity data. Although our synergy training set contains only 88 drug combinations, ComboNet achieves 0.82 test ROC-AUC while standard deep learning methods struggle to reach 0.6 ROC-AUC. We then performed virtual screening on 11,600 candidate drug combinations using ComboNet, empirically tested 30 of these predictions, and identified and validated two novel drug combinations with strong synergy in vitro.
Chapter 7

Conclusion

In this thesis, I have presented novel deep learning techniques to address the challenge of generalizable property prediction and molecular graph generation. I developed counterfactual consistency regularization that highlights the true causal relationship between molecules and properties through multiple training environments. Furthermore, I designed hierarchical graph VAE for molecular graph generation and rationale-based generative models for multi-objective molecule generation. Lastly, I demonstrated that deep learning can automatically discover novel antibiotics against multiple resistant bacteria species [90] and new antiviral drug combinations against COVID-19 [46]. In summary, my thesis work highlights the significant impact that deep learning can have on drug discovery.
Bibliography


