Molecular Property Predictors for Downstream De Novo Generation

Molecular Property Predictors for Downstream *De Novo* Generation

a dissertation presented
by
Varun Tekur
to
The Department of Computer Science

in partial fulfillment of the requirements
for the degree of
Bachelor of Arts with Honors
in the subject of
Computer Science

Harvard University
Cambridge, Massachusetts
May 2022

Thesis advisors: Professors Marinka Zitnik and David C. Parkes Varun Tekur

Molecular Property Predictors for Downstream *De Novo* Generation

Abstract

The task of *de novo* molecular generation involves creating new molecular structures in order to optimize a certain objective, and it is extremely useful in the context of drug discovery. A key challenge in this task is that in many cases, it is not possible to frequently perform ground-truth evaluation of molecules because this evaluation must be done in a wet lab and is therefore resource intensive. In this thesis, I propose a *de novo* generation method that first trains a Graph Neural Network predictor for the property to optimize and then uses this predictor as a scoring function (instead of ground truth scoring) in a Graph-Based Genetic Algorithm generation method. I use batch Bayesian Optimization to create the training dataset for the predictor, and do so iteratively and with realistically sized batches. This training dataset also provides high quality molecules to use as a starting point for the Genetic Algorithm. I evaluate this method on the task of generating molecules that dock well to the human Dopamine Receptor D3 and observe that using Bayesian Optimization to create the training dataset frequently leads to better top molecules generated by the Genetic Algorithm on average compared to using random dataset creation. Additionally, the top molecules generated by the Genetic Algorithm are on average better than the top molecules in the training dataset of the predictor. This highlights the utility of the Genetic Algorithm as an additional optimization

step after Bayesian Optimization.

Contents

Acknowledgments

First of all, I would like to thank my thesis advisors, Professors Marinka Zitnik and David C. Parkes. Prof. Zitnik - I genuinely appreciate the freedom that you gave me to explore a topic that I was interested in while simultaneously pointing me in the right direction with suggestions of papers to read and ideas to explore. The community that you have built in the lab and through the Therapeutics Data Commons research project is truly amazing and helped me easily find other researchers to talk to. Prof. Parkes - you are one of the most understanding and supportive teachers I have ever had. I am constantly amazed by your ability to quickly understand new ideas that I am talking about and give extremely useful advice, and I am grateful for all of your thesis draft feedback and advice regarding setting deadlines.

Additionally, I'd like to thank the graduate students who helped me throughout this project: Yasha Ektefaie, Kexin Huang, Wenhao Gao, and Tianfan Fu. Your guidance was very helpful in deciding what to focus on and moving the progress of my project along.

I would also like to thank Juhee Goyal, Nikhil Behari, and Drew Kelner for all of the memories throughout these past four years. College would not be the same without you all.

Finally, I'd like to thank my parents for their endless support and love. I would not be anywhere near where I am without you, and I know that I can rely on you to talk about anything and guide me in the right direction whenever I need.

0

Introduction

The task of *de novo* molecular generation involves generating new molecules that maximize certain properties and/or meet certain constraints. *De novo* generation is a central task in the drug discovery process, and the key property that researchers often care about is how well a generated drug candidate molecule *docks*, or binds, to a target protein [5]. This a necessary property to optimize for because binding to the target can disrupt its activity and therefore potentially cure or reduce the

severity of a disease associated with the target.

In recent years, there has been lots of work published on using machine learning and other algorithmic methods to perform the task of *de novo* molecular generation for drug discovery. These methods have the potential to provide value because they can explore the molecular space quickly and effectively in order to find a small number of high-quality drug candidates.

One of the most challenging aspects of developing these methods is getting them to work with limited amounts of labeled data (pairs of molecules and their scores for the properties being optimized) and realistic procedures for collecting this data. This is because many useful molecular properties, such as how well a molecule docks to a target protein, can only be determined in a wet lab experiment. Wet lab experiments are resource-intensive, which limits the number of data points that can be labeled. These experiments are also conducted on batches of many molecules at a time. This latter point is particularly relevant to many recently published methods for *de novo* generation (e.g., based on reinforcement learning [30], MCMC sampling [27], and genetic algorithms [13]), which rely on iteratively querying ground-truth property scores for one or small numbers of molecules at a time.

An approach for *de novo* generation under these constraints is to first train a machine learning model to predict the property that needs to be optimized using a realistic amount of labeled data and then query this model as a proxy for ground-truth scores in an iterative generation algorithm [9] [19]. A straightforward way to do this would be to randomly query n data points from an existing library of drug-like molecules for wet-lab labeling and then train a model once using this data.

I instead explore the idea of using *Bayesian Optimization* for iterative training dataset generation.

For a training dataset size of n, this method involves b rounds of 1) selecting n/b molecules to label and add to the training set and 2) re-training the model on the new, larger training dataset. The n/b molecules selected for each round are chosen from a library of drug-like molecules to maximize of a function of a) predicted score and b) epistemic uncertainty (uncertainty due to lack of training data) as estimated by the model. The motivation behind adding molecules with high predicted property scores to the training dataset is that many of these molecules will have high ground-truth property scores. This can allow the model to have finer-grained accuracy in high-property score regions of the input space, which may be useful to the downstream *de novo* generation algorithm that uses it. This can also allow the generation algorithm to use high property score molecules as starting points that can be improved upon. The motivation behind adding molecules with high epistemic uncertainty to the training dataset is to improve the model's accuracy on regions of the input space for which it has less data in, which can improve its overall accuracy. This could improve the overall performance of the *de novo* generation algorithm that uses it and also improve the model's ability to identify high property score molecules during subsequent rounds of data collection.

In this thesis, I present a method for *de novo* molecule generation using a realistic data-labeling scheme that first trains a Graph Neural Network property prediction model [29] using Bayesian Optimization for iterative training dataset creation and then uses this predictor as a scoring function in a Graph-Based Genetic Algorithm [13] for molecule generation. I evaluate the method's effectiveness on the task of generating molecules that dock effectively to the human Dopamine Receptor D3 (DRD3). In order to simulate data labeling using wet lab experiments, I use AutoDock Vina [25], a computational docking software that generates a chemistry-based estimate (docking score) of

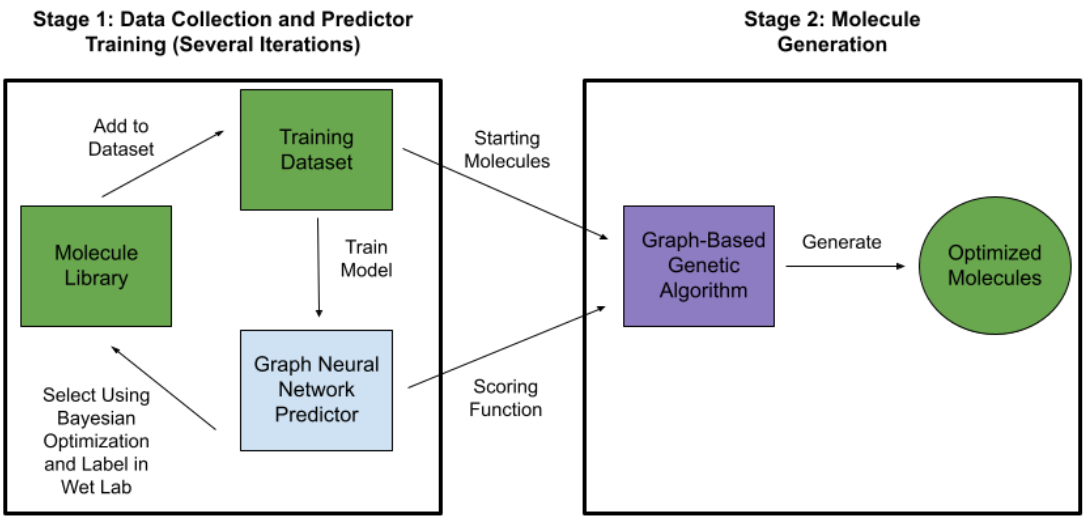


Figure 1: A high level illustration of the method presented in this thesis

how well an arbitrary input molecule binds to a target protein with a known structure. I find that the method is able to generate molecules that have higher docking scores than those in the training dataset and also show the benefit of using Bayesian Optimization for training dataset creation. Additionally, I find that the method works best when the molecules with the highest docking scores in the collected training dataset are used as the starting population for the Genetic Algorithm. Figure 1 illustrated a high level diagram of this method.

1

Problem: *De Novo* Generation of Molecules for Drug Discovery

The task of *de novo* molecule generation involves designing new molecules in order to meet a particular objective. In the context of drug discovery, this objective is to cure or weaken the severity of a particular disease. Since the space of all possible drug candidate molecules is extremely large and

wet-lab experiments to determine the efficacy of candidates are resource intensive, it is not possible to use a brute-force search approach to find ideal candidates. Therefore, computational methods that generate a reasonable number of high-quality drug candidates whose efficacy can then be tested using wet lab experiments are very useful to the drug discovery process [12]. In this chapter, we describe the concept of a molecule docking to a target protein, which is an important property for these computational generation methods to optimize and is the property that we focus on in this thesis. We also discuss computational docking software, which allows us to noisily estimate how well molecules dock to certain targets. Finally, we briefly describe some other properties that are important to optimize in the drug discovery process that are not covered by our experiments.

1.1 Property to Optimize: Docking to a Target

Many diseases are caused by the activity of particular proteins in an individual's body. Therefore, a way to cure or lessen the severity of these diseases is to disrupt the activity of these proteins by using drug molecules that are smaller than the target proteins to physically bind to them. For example, enzymes are proteins that have pockets called active sites. These pockets can physically take in molecules involved in a chemical reaction in order to help the reaction occur successfully. If the reaction that a particular enzyme is involved with contributes to a disease, binding a small molecule to that active site can potentially lessen the severity of the disease because it can disrupt the occurrence of this reaction [4]. This means that a property that *de novo* methods for drug discovery need to be able to optimize is the effectiveness with which a generated molecule docks to a target protein of in-

terest [5] [25], and we focus on this propety in this thesis. An illustration of docking can be seen in Figure 1.1.

In particular, the experiments in this thesis evaluate docking to the human Dopamine Receptor D3 (DRD3). Dopamine receptors are common drug targets for neuropsychiatric (brain-related) diseases [28]. DRD3 was chosen because its structure is known, so the ability of molecules to dock to it can be estimated by computational docking software (explained in the next subsection), and this software for DRD3 was easily available through the Therapeutics Data Commons library [12].

It is worth noting that many recent papers on machine learning methods for *de novo* generation evaluate these methods using different metics, which commonly include LogP (octanol-water partition coefficient), QED (a drug-likeness metric), and the inhibition of kinases GSK3β and JNK3 as predicted by a random forest model. The first two metrics have been criticised for not being similar to objectives optimized in real drug discovery processes [5] [6]. The latter two metrics are drug-discovery objectives in nature, but they use a machine learning model for evaluation, which may fail to generalize well beyond its training distribution and therefore may not be an accurate reflection of ground truth evaluation.

1.2 Measuring this Property: Computational Docking Software

In an ideal world, we would only evaluate the docking ability of molecules to target proteins using wet lab experiments. However, this would make the development of methods extremely slow because it would require a wet lab experiment for every round of method evaluation.

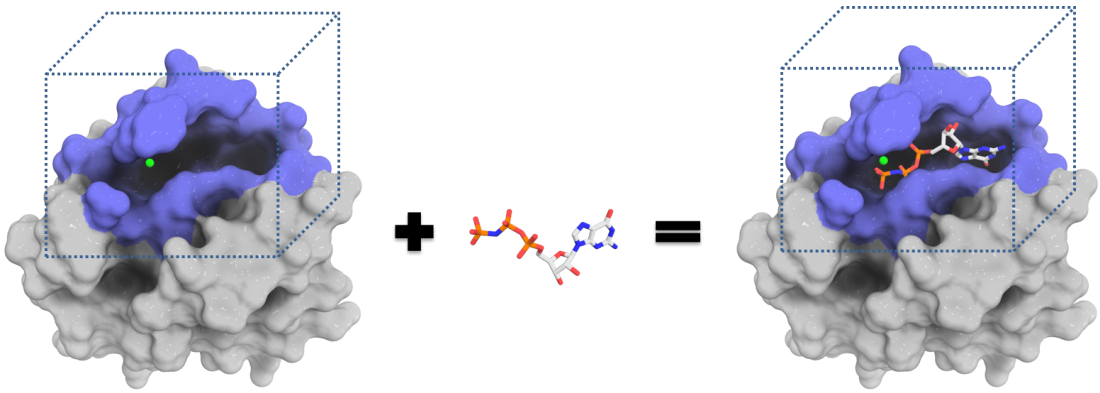


Figure 1.1: An illustration of a small molecule docking to a target molecule [11]

A viable alternative is to use computational docking software as a proxy for wet lab experiments during initial method development, and this is the approach that we take in this thesis. This software estimates the strength with which a small molecule (ligand) binds to a target molecule (receptor) and structure of the receptor/ligand pair after binding using the structures of the ligand and receptor and properties of chemistry. For each input ligand, it outputs a single number (docking score), and a lower (more negative) docking score indicates that a ligand binds better to a receptor.

The docking software that we use is AutoDock Vina [25], which is accessible through the pyscreener python wrapper [10] (which is called using another wrapper, the Therapeutics Data Commons [12] python library). We also note that docking software only provides an *estimate* of how well a ligand binds to a receptor (and can therefore be noisy), and it is also computationally intensive (for example, it can take about an average of 10 seconds per molecule to find docking scores for molecules in the ChEMBL dataset [20]). However, it is useful as a first step in method development, and methods that evaluate well on it can then be further evaluated using wet lab experiments.

1.3 Other Drug Development Considerations

Despite its utility for finding drug candidates, finding docking scores is not the sole focus of drug discovery [25]. Researchers also need to determine if drug candidates affect the activity of targets that they bind to in specific ways in order to reduce the severity of a disease [4], and the candidates eventually need to be tested in animals/humans for both efficacy and safety.

Some additional properties that can be optimized or required to meet a certain threshold by generation methods are synthesizability and side effects to molecules/processes in the body other than the target molecule [12]. In the future, it would be interesting to test this thesis's method and related methods on the task of multi-objective optimization of these properties and docking scores.

2

Methods: Molecular Property Prediction and Bayesian Optimization

We now discuss the first stage of our method for *de novo* molecule generation: training a property predictor that can be used as a scoring function in a generation method. While we train a predictor of docking scores to human Dopamine Receptor D3, this training procedure can be used for a

model that needs to predict any numerical property of molecules.

2.1 Loss Functions for Regression and Uncertainty Quantification

2.1.1 Mean Squared Error

For models that only need to predict a molecular property, we use the mean squared error (MSE) loss function, a standard regression loss function,

$$\frac{1}{n}\sum_{i=1}^{N}(y_i - F(x_i))^2,$$

defined here for a batch of N molecules $\{x_i\}_{i=1}^{N}$, true docking scores $\{y_i\}_{i=1}^{N}$, and regression model F.

2.1.2 Evidential Regression

For Bayesian Optimization (described in a later section), we need models to be able to perform property prediction and quantify the epistemic uncertainty for each input molecule. Epistemic uncertainty is a measure of model uncertainty due to lack of training data. For an input x, epistemic uncertainty should be high if the model has not been trained on many data points that are similar to x. The model is uncertain, or has low confidence, for input x in this case because its predictions are unlikely to be very accurate on regions of the input space that it has not been trained well on.

While there are many methods for quantifying the epistemic uncertainty of a deep regression model, one method that has been shown to work well [22] for molecular property predictors (and

in particular the D-MPNN archiecture that we used) is deep evidential regression [1]. In evidential regression, we assume that each regression target y is sampled from a normal distribution with mean μ and variance σ^2, as in traditional MSE regression. However, unlike MSE regression, we assume that μ is sampled from a normal distribution and σ^2 is sampled from an invese gamma distribution, the parameters for which are a function of the input x:

$$\mu \sim \mathcal{N}\left(\gamma, \sigma^2 v^{-1}\right) \quad \sigma^2 \sim \Gamma^{-1}(\alpha, \beta)$$

And $F(x) = (\gamma, v, \alpha, \beta)$ for regression model F. The loss function that used is the sum of the negative logarithm of the marginal likelihood, $p(y|F(x))$, and a regularization term. The goal of minimizing this loss function is to find the model parameters that best explain the observed data (assuming the data generation process described above) and associate uncertainty with error.

We fill first explain the marginal likelihood loss term. It is known that when the assumed distributions on μ and σ^2 are normal and inverse-gamma, respectively, the distribution of the marginal likelihood is the Student-t distribution:

$$p\left(y \mid F(x)\right) = \mathrm{St}\left(y; \gamma, \frac{\beta(1+v)}{v\alpha}, 2\alpha\right).$$

So, we can find the negative log marginal likelihood using the Student-t PDF,

$$\frac{1}{2}\log\left(\frac{\pi}{v}\right) - \alpha\log(\Omega) + \left(\alpha + \frac{1}{2}\right)\log\left(\left(y-\gamma\right)^2 v + \Omega\right) + \log\left(\frac{\Gamma(\alpha)}{\Gamma\left(\alpha + \frac{1}{2}\right)}\right),$$

where $\Omega = 2\beta(1+v)$.

The following regularization term is also used:

$$|y-\gamma|(2v+\alpha).$$

When v increases, then the variance of μ decreases, and when α increases the mean and variance of σ^2 (based on the inverse gamma distribution) both decrease. Intuitively, this indicates when $(2v+\alpha)$ increases, then we have parameters that indicate more total evidence for the prediction $E[\mu] = \gamma$. So, the regularization term causes high loss values when $(2v+\alpha)$ is high and the absolute error is high because we do not want parameters that indicate high confidence when the model is incorrect.

The total loss is the sum of the negative log marigial likelihood term and λ times the regularization term. More detailed derivations of these loss terms can be found in the deep evidential regression paper [1].

Training a model using this loss function allows us to use the predicted parameters to estimate y as $E[\mu] = \gamma$ and the epistemic uncertainty of this prediction as $Var[\mu] = \frac{\beta}{v(\alpha-1)}$. The intuition behind this variance term as epistemic uncertainy is that if we haven't seen many data points near some input x, then we are likely to not be confident in the distribution of its regression target y, which is parameterized with mean μ.

2.2 D-MPNN Model Architecture

We now describe the architecture of the models that we train with the above loss functions. The ideas of the model architecture that we use, D-MPNN, were initially proposed by Dai et al. [7], and D-MPNN was shown to work well for the task of molecular property prediction by Yang et al [29]. In our experiments, we trained our models using the chemprop python package, which implements this arhitecture for the standard MSE regression case, as well as the codebase of Amini et al. [1], which implements it for the evidential regression case. D-MPNN is an example of a *Graph Neural Network*, which applies the ideas of neural networks to graph structures. For molecular property prediction, the nodes of the input graph are atoms and the edges are chemical bonds. We note that the model is trained using minibatch gradient descent/backpropogration, the learning rate is updated using a Noam learning rate scheduler (the idea for this is explained in Vaswani et al. [26]), and the Adam optimizer is used [17]. The architecture consists of the following four steps (as explained in Yang et al.).

2.2.1 Initial Directed Edge Hidden States

Nodes in the graph are connected by directed edges going in both directions, and each directed edge has a hidden state, or vector, associated with it. On each forward pass of the model, the starting hidden state for the directed edge from node a to b is computed as:

$$h_{ab}^{0} = relu(W_i cat(x_a, e_{ab})).$$

In this formula, x_a and e_{ab} are chemistry-informed features for atom a (e.g., atomic number, atomic mass) and the bond from atom a to atom b (e.g., bond type, if the bond is in a ring or not), *cat* is vector concatenation, *relu* is the standard relu activation function ($relu(x) = x$ if $x \geq 0$ and 0 otherwise), and W_i is a parameter matrix (updated on each model backward pass) of size $s \times |x_a| + |e_{ab}|$ (s is the size of hidden states, a hyperparameter).

2.2.2 Directed Edge Message Passing

The model then performs t (a hyperparameter) rounds of message passing in order to iteratively update the hidden states for each directed edge. For round $i + 1$, this means that the hidden state for the edge from a to b gets updated as:

$$h_{ab}^{i+1} = relu(h_{ab}^0 + W_m \sum_{c \in N(a) \setminus b} h_{ca}^i).$$

In this formula, $N(a)$ are the neighbors of a and W_m is a parameter matrix of size $s \times s$. The intuition behind performing n rounds of message passing is that the representation for each directed bond edge should be informed by its *context*, or the structure of the graph around it. If we iteratively set its the hidden representation to be a function of the bonds connected to it, then information from bonds multiple edges away from it will eventually be passed to it.

2.2.3 Node Hidden States

After t rounds of message passing, we can then compute hidden states for each node:

$$h_a = relu(W_n cat(x_a, \sum_{b \in N(a)} h^t_{ba}))$$

In this formula, W_n is a square parameter matrix.

2.2.4 Aggregation and Property Prediction

Finally, we can compute a full graph representation by adding the hidden states for all nodes together and then passing this sum into a feed forward neural network that uses the *relu* activation function to obtain an output value. For standard regression, this output value is a single number, and for evidential regression, this output value is a vector of size 4.

An illustration of the model can be seen in Figure 2.1.

2.3 Bayesian Optimization for Training Data Collection

We now describe the process used to select molecules to be labeled with a property score and added to the training dataset. The most direct way to construct a training dataset of size n from a pool of unlabeled molecules is to randomly sample n of the molecules and then label them (ideally using a wet-lab experiment, but in our experiments using computational docking software as described in Chapter 1). We instead make use of batch Bayesian Optimization [8] for dataset creation. Batch Bayesian Optimization with batch size b involves n/b iterations of selecting the b data points from a pool of unlabeled data points that maximize an *acquisition function*. This acquisition function is generally a function of a property score and an uncertainty measure, each of which are estimated

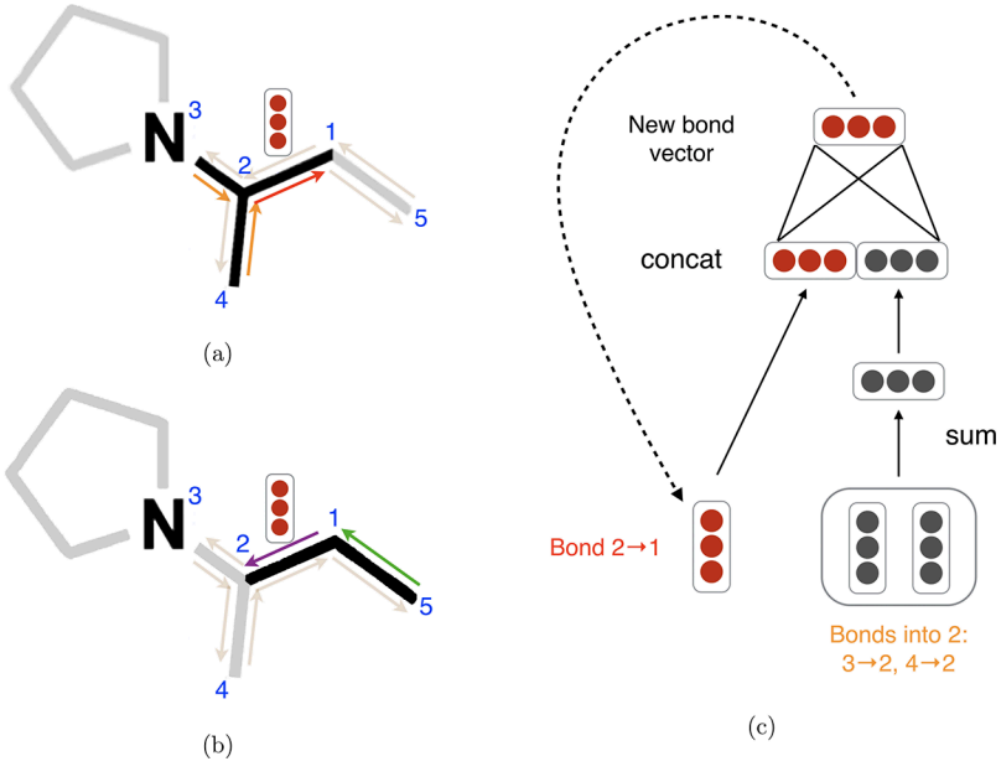


Figure 2.1: Figure 1 from Yang et al.[29], with caption: "Illustration of bond-level message passing in our proposed D-MPNN. (a) Messages from the orange directed bonds are used to inform the update to the hidden state of the red directed bond. By contrast, in a traditional MPNN, messages are passed from atoms to atoms (for example, atoms 1, 3, and 4 to atom 2) rather than from bonds to bonds. (b) Similarly, a message from the green bond informs the update to the hidden state of the purple directed bond. (c) Illustration of the update function to the hidden representation of the red directed bond from diagram (a)."

by a model that has been trained on the previously labeled data points. In the case of molecular properties with ground truth values determined using wet lab experiments, b must be a number of molecules that researchers can reasonably run an experiment with. So, it cannot be too large because researchers may not have the resources to run an experiment of this size. It also cannot be too small (e.g., it doesn't make sense to run a wet lab experiment using only one or a handful of molecules). Similarly, the size of n (total molecules) is constrained by researcher resources. Below, we describe the acquisition function that we use in our experiments, molecule libraries that we select from, and motivation behind this method of data collection.

2.3.1 Upper Confidence Bound Batch Bayesian Optimization

As in Bertin et al. [2] and Soleimany et al. [22], we use the upper confidence bound (UCB) acquisition function, which follows the form

$$a * score + b * uncertainty.$$

In our experiments, *score* and *uncertainty* are the negative (since lower docking scores are better) of the predicted docking score and the epistemic uncertainty, respectively, output by a D-MPNN trained using an evidential regression loss function. Hyperparameter a is generally 1, although we also tested the case when $a = 0$ (also known as *Active Learning*).

On the first iteration of data collection, n/b points are selected randomly from a molecule library. On each subsequent iteration, n/b points are selected from a *selection pool* to maximize the UCB

function. This selection pool is a random sample (without replacement) from the library or the whole library itself. Once these n/b points are selected, they are removed from the library (so that they cannot be selected again).

The hyperparameters in this process are a, b and the size of the selection pool.

2.3.2 ZINC and ChEMBL Libraries

We run experiments using both the ZINC 250K [24], [18] and the GuacaMol ChEMBL [20], [3] molecule libraries, which are meant for searching for molecules for drug discovery purposes. The GuacaMol ChEMBL library is larger (about 1.5 million molecules vs 250K) and is also known to have other potential advantages like having larger and exclusively previously synthesized molecules. ZINC 250K can be downloaded using the Thereapautics Data Commons python package [12] and GuacaMol ChEMBL can be downloaded from a link in the GuacaMol paper [3]. Not all molecules in the libraries were used. For example, molecules that caused the docking software to raise errors were discarded, and a random set of 3k molecules was removed from ZINC 250K for predictor validation/testing during the initial stages of method development.

2.3.3 Motivation Behind Approach

Bayesian Optimization for the collection of training data can offer several possible advantages over random training data collection in the pipeline of training a model and then using it as a scoring function for a generation method:

1. If data points with high predicted values for a property are selected, then many of these

points may have high ground-truth values for that property. If the training dataset has many high-docking score molecules, then the model uses more data points to learn a regression function in high docking score than low docking score regions of the input space, potentially leading to higher accuracy in the former. This may lead to better performance by the generation method because it can more accurately differentiate between which molecules are "good" vs. "very good". These fine-grained predictions are likely to matter less in "bad" regions of the input space as long as points here are labeled as "bad".

2. If top molecules in the training dataset have higher docking scores, then these better molecules can provide a better starting point for a Genetic Algorithm molecule generation method, which we use. The Genetic Algorithm will be explained in more detail in chapter 3, but this advantage can be thought of as a generation method being given a better inductive bias.

3. The uncertainty term in the UCB acquisition function as well as sampling a fraction of the full library to be used as a selection pool can allow for selection of more *diverse* data points. The former can allow for this because data points with a higher epistemic uncertainty quantification will be dissimilar to existing points in the training dataset. So, adding these points to the training dataset can allow the model to improve its predictions on input areas about which it has little knowledge. The latter can help with this by helping Bayesian Optimization data collection not over-rely on the knowledge of the model. If points are selected from the entire molecule library to maximize the acquisition function, then these points are likely to correspond to the model's current idea of a high acquisition function molecule. If points are instead selected from a random sample of the library, then points which the model does not believe are optimal will be selected. If some of these points are optimal (and the model was wrong about them), the training dataset will gain molecules about which the model had less knowledge.

   A more diverse set of data points is important because it can allow the model to cover more area in the input space for a given training set size, potentially increasing its performance. This can be useful for both the downstream generation method as well as subsequent iterations of Bayesian Optimization (due to higher accuracy acquisition function predictions).

3

Methods: *De Novo* Molecular Generation

We will now describe the method used for *de novo* generation, a Graph-Based Genetic Algorithm (Graph GA) that was presented by Jensen et al. [13]. We use this algorithm with a D-MPNN molecular property predictor output that is trained as described in Chapter 2 as the scoring function. In our experiments, we used the version of this algorithm implemented by the GuacaMol team [3] and adapted it to work with a D-MPNN scoring function. We also briefly highlight other recently pub-

lished *de novo* molecule generation methods, which can similarly make use of a property predictor as a scoring function.

3.1 Graph-Based Genetic Algorithm

A Genetic Algorithm is a technique to find data points in a space that maximize a function [23], and it generally follows the following pseudocode :

Algorithm 1 Genetic Algorithm

```
Set starting population
Evaluate scores of starting population
while not converged do
    Sample molecules weighted by score
    Offspring ← Crossover and Mutation of sampled molecules
    Evaluate scores of offspring
    Population ← top scoring molecules between offspring and old population
end while
```

Once the algorithm terminates, the population, and particularly the highest-scoring member, is the algorithm's solution to the optimization. We break down these different steps in the context of Jensen et al.'s Graph GA [13] and using a D-MPNN property predictor for scoring.

3.1.1 Starting population and score evaluation

The starting population of molecules is either a random sample of size *population_size* from the molecule library used for training the D-MPNN property predictor (either ZINC 250K or GuacaMol ChEMBL), or it is the top *population_size* scoring molecules based on grouth-truth collected scores in the training dataset of the property predictor. Score evaluation involves assigning the

property predictor's estimated score to each member of the population. For docking scores, since a smaller docking score is better, we use -1 times the output of the property predictor for score evaluation.

3.1.2 Sample molecules weighted by score

For the current population, we subtract the minimum score in the population from each score (let us call this new score *sampling_score*), and then sample *offspring_size* molecules from the population with replacement where molecule i is sampled with probability $\frac{sampling\_score_i}{\sum_{k \in pop} sampling\_score_k}$.

3.1.3 Crossover/Mutation and new population creation

Crossover involves randomly choosing two members of the sampled molecules in the previous step, randomly splitting each molecule into two pieces, exchanging these pieces between the two molecules, and then putting the new pieces for each molecule together. One of the newly created molecules is added to a candidate list if it has certain chemical properties (e.g., fewer than five heavy atoms, see Jensen et al. for the complete list [13]).

Mutation involves, with some probability *mutation_rate*, changing the structure of the molecule created by crossover. Examples of mutations include adding and deleting atoms from the graph, and the type of mutation is chosen at random, with roughly equal probability for each type. See Jensen et al. for the complete list of mutation types [13]. Again, mutated molecules must have certain chemical properties (same as described above), and they are removed by the algorithm if they do not.

The new population is set to be the *population_size* molecules with the highest scores between the old population and the candidate list from crossover/mutation.

3.1.4 Convergence

The algorithm is considered to be converged if for *patience* (a hyperparameter) iterations, the scores of the population do not change. The algorithm also terminates after *generations* (a hyperparameter) iterations if it has not converged by then.

3.1.5 Ground Truth Evaluation

Once the algorithm terminates, we can compute an estimate of the ground-truth docking scores of the final population of molecules using docking software.

3.2 Related Work

There are also a wide variety of other recent approaches for *de novo* molecular generation. One category of these approaches is iterative methods such as the Graph GA. Some other examples in this category are based on Markov Chain Monte Carlo sampling, [27] Reinforcement Learning [30], identifying molecule subgraphs (rationales) that contribute to meeting property constraints and then generating molecules using these subgraphs [16], and using gradient-based optimization by creating scaffolding trees of molecules that are differentiable [9]. Another category involves learning a generative model of the molecular space. For example, Jin et al. [14] use Bayesian Optimization in the latent space of a variational autoencoder based model. Additionally, Segler et al. suggest the

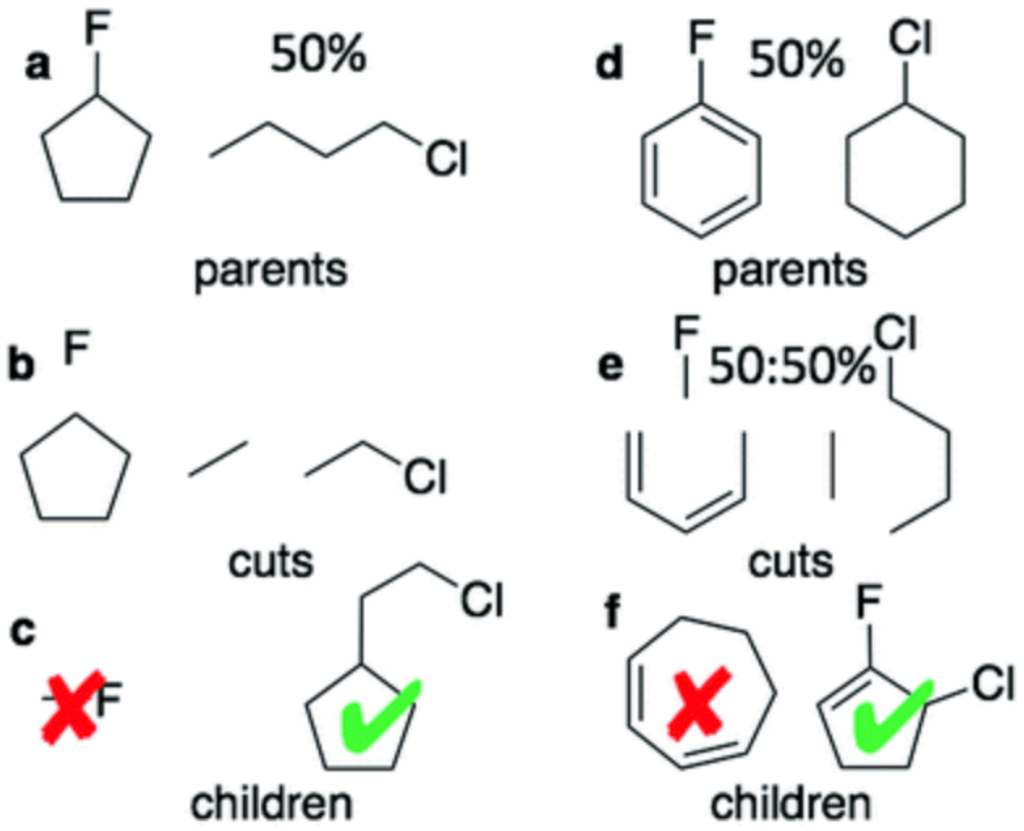


Figure 3.1: Figure 1 from Jensen et al. [13] illustrating the crossover process. The "cuts" section illustrates splitting the molecules into pieces, and the "children" are created as a result of the pieces being put back together after exchange. The molecules with a red cross do not meet the chemical criteria to be considered for the population while the molecules with a green check mark do.

strategy of pretraining an LSTM based model using a large set of molecules, fine-tuning it using molecules with ideal values for properties of interest, and then generating molecules with the hope that they will also have ideal values for these properties.

4

Results and Discussion

We now describe the experiments that we ran to test the performance of the *de novo* molecule generation pipeline that first trains a predictor using Bayesian Optimization and then uses it as a scoring function in a Graph-Based Genetic Algorithm (Graph GA) for generation. We compare this method to training a predictor using a randomly collected training dataset and using it as a scoring function during generation (in order to investigate the benefits of Bayesian Optimization for data

collection). Additionally, we investigate the performance of Graph GA in these pipelines when a randomly selected starting population is used compared to when the starting population is set to the best molecules in the training dataset of the predictor. The performance of the generation pipeline for each particular data collection method is evaluated by reporting the ground truth docking scores of top generated molecules averaged across several trials (three runs of data collection and predictor training and three runs of Graph GA for each trained predictor). In all experiments, we use a training dataset size of $n = 5000$, and in Bayesian Optimization, we use a batch size of $b = 1000$. As a reminder, we also emphasize that a lower (more negative) docking score is better.

We also credit the codebases used to perform these experiments. The chemprop python package (based on the architecture and training described in Yang et al.[29]) and codebase from Soleimany et al. [22] were used for training and inference of D-MPNN models using MSE and evidential regression loss functions, respectively. We also used the GuacaMol baselines [3] implementation of Graph GA and adapted it to use a D-MPNN for docking score prediction as the score function.

4.1 D-MPNN hyperparameter optimization

We optimized parameters using a 5,000 molecule training set and 1,000 molecule validation set (for testing) set that were both sampled from ZINC 250K and labeled using docking software. For each parameter setting, we ran 5 instances of training with an 80/20 training/validation split and averaged the root mean squared error (RMSE, measure of accuracy that we used) accross these instances to determine a score for that parameter setting.

For models trained using the MSE loss function, we ran a grid search over hidden state vector size = 300,600,1000, number of layers in the feed forward neural network (last stage of the model) = 2,3, and number of message passing steps = 3-5 and found that 300, 2, and 5 were the best choices for these parameters, respectively. Then, using this architecture, we searched over epochs = 35,50,75,100 and found 50 to be the best choice.

Models that are trained using the evidential loss function have some additional considerations since they are used for Bayesian Optimization: 1) they are trained on dataset sizes of 1k,2k,3k,4k,and 5k (each iteration of Bayesian Optimization), and 2) both accuracy (RMSE) and epistemic uncertainty calibration are important to optimize. Following Soleimany et al. [22], we quantified epistemic uncertainty calibration using the Spearman rank correlation coefficient between RMSE and epismetic uncertainty. This is motivated by the fact that for uncertainty quantification that is well calibrated, a model will be less accurate (high RMSE) on high uncertainty points (because it has not seen much training data like this) and more accurate (low RMSE) on low uncertainty points.

We first optimized architecture parammeters using the same search space as for D-MPNN MSE training, and we found that the same parameters as above (hidden size = 300, feed forward network layers = 2, and message passing steps = 5) had both very good accuracy (RMSE = 0.700) and uncertainty calibration (rank correlation coefficient = 0.238) relative to other parameter settings.

Then, we jointly found values for the evidential regularization coefficient λ (search space = 0.01, 0.05, 0.1, 0.2) and epochs (search space = 35,50,75,100) for each data size. We used MinMax scaling to re-scale RMSE and uncertainty rank correlation coefficient and created a "score" for each parameter setting as the re-scaled rank correlation coefficient - re-scaled RMSE. For data size = 1k-4k,

we selected the parameters with the largest score (data size: (λ, epochs), = 1k: (0.01, 50), 2k: (0.05, 35), 3k: (0.05, 50),4: (0.05, 35)). For data size = 5k, we set $\lambda = 0$ and trained for 50 epochs (same as the MSE trained models) because uncertainty quantification isn't necessary for the final model that is used as a scoring function in Graph-GA (it is only used during Bayesian Optimization data collection).

4.2 ZINC 250K Bayesian Optimization + Graph GA

4.2.1 Initial hyperparameter filtering

We first searched over a large number of parameters for Bayesian Optimization in order to identify some sets of parameters to execute multiple runs for. We did this because the Bayesian optimization experiments take a very long time to run (about 15 hours for ZINC 250K as the dataset), so we wanted to select only the most promising parameter sets to run multiple trials and report results for.

Recall that we use the upper confidence bound (UCB) acquisition function during Bayesian optimization, which is of the form:

$$a * score + b * uncertainty$$

Additionally, recall that during each iteration of Bayesian Optimization, we select points that maximize the UCB acqisition function from a random sample of the molecule library being used (in this case, ZINC 250K). For batch size b, we define hyperparameter d as b/d = size of the random sample that is used. This means that d represents the percent of the sample that we are selecting to

add to the training set during each iteration of Bayesian Optimization. For convenience, we also define $d = -1$ to mean that we did not randomly sample at all and instead computed the acquisition function on the all of the molecules in the library that remain.

We first trained one model for each combination of $d \in -1, 0.04, 0.2$ and $(a, b) \in \{(1, 0), (1, 2), (1, 5), (0, 1)\}$ and then used that model as the scoring function in three runs of Graph GA for each starting population type (random molecules or the best molecules in the training dataset) with the following parameters: $\textit{population\_size} = 100$, $\textit{offspring\_size} = 200$, $\textit{generations} = 300$, $\textit{mutation\_rate} = 0.1$, $\textit{patience} = 5$ (all runs of Graph GA reported in the paper use these parameters). Then, for each starting population type, we selected the six parameter settings that resulted in the largest average docking score for top-ten generated molecules (averaged across the three runs of Graph GA). We then ran two more instances of Bayesian Optimization training with each of these parameter settings in order to be able to report results averaged across both runs of Bayesian Optimization training and runs of Graph GA (since these algorithms are both sources of variation). We report results for only these parameter settings for which we performed multiple instances of Bayesian Optimization training.

As mentioned above, for each predictor data collection method (a specific Bayesian Optimization parameter setting or random selection of the whole training dataset at once), we report results averaged across three trained docking score predictors and three runs of Graph GA for each predictor. We evaluate the molecules output by Graph GA by computing the Top docking score, the average of the Top 10 docking scores, and the average of the Top 100 docking scores (which are all of the docking scores generated, since the we use a population size of 100). When computing these metrics

for each trial, we drop molecules that the docking software did not work for from the average (the number of molecules that fell into this category was very small, and was at most an average of 0.556 across all reported parameter settings in the ZINC 250K experiments).

4.2.2 Graph GA Starting Population: Random vs Best in Training Data

We first analyze the difference between setting the starting population of Graph GA to random molecules sampled from ZINC 250K and setting it to the molecules in the D-MPNN predictor's training dataset with the highest ground truth docking scores (Table 4.1). For each start type, metrics are averaged across all Bayesian Optimization parameter settings. By examining Table 4.1, we can see that using the best molecules in the training dataset as the starting population leads to the best results. This is also true when random data collection is used for training the predictor. So, we report further metrics for only this starting population type.

4.2.3 Bayesian Optimization vs. Random Data Collection

We then analyze the advantage of using Bayesian Optimization over random selection for training data collection. Table 4.2 displays the docking score metrics of molecules generated by Graph GA for all Bayesian Optimization parameter settings as well as for random training data collection of the predictor used by Graph GA. Bayesian Optimization appears to have an advantage in generating high-quality candidate molecules: the Top 10 average and Top 100 average docking score is better for all Bayesian Optimization parameter settings except one, and the Top 1 docking score is better for all parameter settings except two.

4.2.4 Graph GA Molecules vs. Best Molecules in Training Dataset

Finally, we illustrate the advantages of using Graph GA as an extra optimizing step after initial data collection from ZINC 250K. Table 4.3 displays docking score metrics for molecules in D-MPNN predictor training datasets. By comparing this to Table 4.2, we can see that for each training data collection type, the Top 1 and Top 10 average docking score of molecules generated by Graph GA are always better than the Top 1 and Top 10 average docking score of molecules in the training dataset. However, the Top 100 average docking score of Graph GA generated molecules is always lower than the Top 100 average docking score of the training dataset molecules. This indicates that Graph GA is able to improve upon the 100 molecules it receives as input by generating some molecules that are better than all molecules in this input, but on average the molecules that it generates are worse than those input to it.

Comparing Table 4.4 to Table 4.3 can shed some insight into why this may be happening. We can see that the *predicted* (by the D-MPNN predictor model) docking metrics of molecules generated by Graph GA are similar to the docking metrics for the best molecules in the training dataset of that predictor. This means that Graph GA may be generating molecules that have similar structure and chemical properties to the best molecules in the training dataset of the D-MPNN predictor it uses. This seems plausible because the D-MPNN predictor (scoring function) only has knowledge of the training data, so it is likely to most consistently indicate that molecules similar to the best ones in the training data have high docking scores (in addition to indicating this for some molecules that are far from the training data due to model error). We hypothesize that this is occurring and is analogous

to slightly changing properties of the best molecules in the training dataset (and also incorrectly labeling some molecules far from the training data as having high docking scores), leading to some generated molecules that are better than the best in the training dataset but also many molecules that are worse.

Overall, since the goal of drug discovery is to eventually find a single or a few molecules to use as a drug, we believe that Graph GA provides an advantage on top of Bayesian Optimization because it seems to be capable of generating several top molecules that are improvements upon the best molecules found during data collection.

Starting Population Type	Top 1	Top 10 Avg	Top 100 Avg
Best in Collected Training Data	-13.05	-12.52	-10.60
Random Sample from ZINC 250K	-12.05	-11.46	-9.58

Table 4.1: Top 1, Top 10 Average, and Top 100 Average ground truth docking scores of molecules generated by Graph GA for each starting population type (averaged across all Bayesian Optimization parameter settings) using ZINC 250K molecule library. Note that a lower docking score is better.

4.3 GuacaMol ChEMBL Bayesian Optimization + Graph GA

We also ran experiments to test this method using the GuacaMol ChEMBL molecule library. We performed these experiments in order to see if the advantages of our method would be apparent when using a larger molecule library. We also wanted to see if the best results using GuacaMol ChEMBL are better than the best results using ZINC 250K (since this library is considered to have advantages over ZINC [3]).

Since $d = 0.2$ seemed to preform worse than the other settings of d in the previous experiments,

Bayes Opt Parameters (d,a,b) or Random	Top 1	Top 10 Avg	Top 100 Avg
(0.04,1,2)	-13.34	-12.75	-11.05
(-1,1,0)	-13.17	-12.62	-11.13
(-1,1,5)	-13.16	-12.61	-10.55
(-1,1,2)	-13.03	-12.59	-11.02
(0.04,1,0)	-13.14	-12.55	-10.89
(0.04,0,1)	-12.97	-12.49	-10.30
(0.2,0,1)	-13.00	-12.43	-9.49
(0.2,1,2)	-12.91	-12.37	-10.55
Random Train Data Collection	-12.94	-12.28	-9.91
(0.2,1,5)	-12.73	-12.26	-10.40

Table 4.2: Top 1, Top 10 Average, and Top 100 Average ground truth docking scores of molecules generated by Graph GA using best molecules in training data as starting population and ZINC 250K molecule library for different data collection methods. Data collection methods are ranked by Top 10 Avg. Note that a lower docking score is better.

Bayes Opt Parameters (d,a,b) or Random	Top 1	Top 10 Avg	Top 100 Avg
(0.04,1,2)	-12.13	-11.83	-11.26
(-1,1,0)	-12.37	-12.00	-11.53
(-1,1,5)	-12.20	-11.90	-11.27
(-1,1,2)	-12.33	-12.02	-11.45
(0.04,1,0)	-12.17	-11.84	-11.30
(0.04,0,1)	-12.17	-11.73	-10.96
(0.2,0,1)	-12.03	-11.62	-10.91
(0.2,1,2)	-12.23	-11.69	-10.96
Random Train Data Collection	-11.67	-11.23	-10.47
(0.2,1,5)	-11.90	-11.48	-10.90

Table 4.3: Top 1, Top 10 Average, and Top 100 Average ground truth docking scores of molecules present in the training dataset using ZINC 250K molecule library for different data collection methods. We emphasize that these are scores of molecules from the data collection phase of the generation pipeline, which occurs before the Graph GA generation phase. Note that a lower docking score is better.

we discarded this parameter value. Instead, we replaced it with $d = 0.006$. We chose this value because it leads to a selection pool that is approximately 1/10 of the training dataset size (roughly 1.5 Million), which is also the case for $d = 0.04$ in the previous experiments (for a data size of roughly

Bayes Opt Parameters (d,a,b) or Random	Top 1	Top 10 Avg	Top 100 Avg
(0.04,1,2)	-12.09	-11.91	-11.63
(-1,1,0)	-12.12	-11.99	-11.75
(-1,1,5)	-12.11	-11.84	-11.51
(-1,1,2)	-12.19	-11.98	-11.69
(0.04,1,0)	-12.03	-11.83	-11.52
(0.04,0,1)	-11.75	-11.56	-11.26
(0.2,0,1)	-11.74	-11.54	-11.20
(0.2,1,2)	-11.68	-11.55	-11.29
Random Train Data Collection	-11.68	-11.49	-11.16
(0.2,1,5)	-11.67	-11.42	-11.11

Table 4.4: Top 1, Top 10 Average, and Top 100 Average *predicted* (by the D-MPNN predictor) docking scores of molecules generated by Graph GA using best molecules in training data as starting population and ZINC 250K molecule library for different predictor training methods. Note that a lower docking score is better.

250K). Using this value as well as $d = 0.04$ allowed us to test two different interpretations of the d parameter: using it to dictate the size of $size(batch)/size(selection\_pool)$ and using it to dictate the size of $size(selection\_pool)/size(molecule\_library)$.

In these experiments, D-MPNN training and Graph GA used the same hyperparameters as in the ZINC 250K experiments. Also, results are reported for Graph GA with a starting population of the best molecules in the training data instead of a random starting population. This is because the average docking score metrics for both Bayesian Optimization (averaged across all parameter settings) data collection and random data collection were better when the best molecules in the training data were used as a starting population compared to when a random starting population was used.

Table 4.5 contains docking score metrics for molecules generated by Graph GA for different data collection methods and Table 4.6 contains docking score metrics for the best molecules in the

training dataset for these same data collection methods. Interestingly, when random data collection is used, the Graph GA generated molecules have lower Top 1 and Top 10 average docking scores than the molecules in the training dataset. However, when Bayesian Optimization data collection is used, the Graph GA generated molecules have higher values for these metrics than the training dataset molecules (as in the ZINC 250K experiments). Additionally, the Graph GA generated molecules have higher values of docking score metrics than when random data collection is used for all Bayesian Optimization parameter settings (as in the ZINC 250K experiments). Finally, the best docking score metrics for Graph GA generated molecules in these experiments are better than the best docking score metrics when ZINC 250K is used as the molecule library. These findings once again highlight the advantages of our method over using random data collection and also show improved top results due to a better library (GuacaMol ChEMBL) from which data is collected.

Bayes Opt Parameters (d,a,b) or Random	Top 1	Top 10 Avg	Top 100 Avg
(-1,1,0)	-13.64	-13.22	-11.60
(-1,1,2)	-13.54	-13.01	-11.28
(0.006,1,0)	-13.44	-12.98	-11.55
(0.004,1,0)	-13.19	-12.70	-11.31
(0.006,1,2)	-13.18	-12.56	-9.49
(0.004,1,2)	-13.17	-12.45	-10.06
Random Train Data Collection	-11.99	-11.19	-8.81

Table 4.5: Top 1, Top 10 Average, and Top 100 Average ground truth docking scores of molecules generated by Graph GA using best molecules in training data as starting population and GuacaMol ChEMBL molecule library for different data collection methods. Data collection methods are ranked by Top 10 Avg. Note that a lower docking score is better.

Bayes Opt Parameters (d,a,b) or Random	Top 1	Top 10 Avg	Top 100 Avg
(-1,1,0)	-13.33	-12.92	-12.28
(-1,1,2)	-13.27	-12.68	-11.62
(0.006,1,0)	-13.17	-12.69	-11.98
(0.004,1,0)	-13.13	-12.55	-11.67
(0.006,1,2)	-12.93	-12.20	-11.27
(0.004,1,2)	-12.70	-11.93	-10.96
Random Train Data Collection	-12.37	-11.59	-10.79

Table 4.6: Top 1, Top 10 Average, and Top 100 Average ground truth docking scores of molecules present in the training dataset using GuacaMol ChEMBL molecule library for different data collection methods. We emphasize that these are scores of molecules from the data collection phase of the generation pipeline, which occurs before the Graph GA generation phase. Note that a lower docking score is better.

4.4 Effect of Bayesian Optimization Hyperparameters

Finally, we briefly comment on the effect of Bayesian Optimization hyperparameters in both the ZINC 250K and GuacaMol ChEMBL experiments. Larger selection pool sizes seem to be better than smaller ones, as three of the four top results in the ZINC experiments and the top two results in the ChEMBL experiments used the full molecule library as the selection pool ($d = -1$). Additionally, it seems as though using no uncertainty $b = 0$ in the UCB acquisition function may potentially work better than using uncertainty because this works best for all selection pool sizes in GuacaMol ChEMBL experiments and for $d = -1$ in the ZINC experiments (although not for $d = 0.04$ in these experiments). However, more experiments should be done in order to validate these claims.

5

Conclusion

5.1 Method and Results Overview

In this thesis, we proposed a two-stage method for *de novo* molecular generation and evaluated it on the task of generating molecules that dock well to the target protein DRD3. This pipeline first trains a Graph Neural Network based model to predict docking scores and quantify uncertainty, using Bayesian Optimization for training data collection. This predictor was then used as a scoring

function in a Graph-Based Genetic Algorithm (Graph GA) for molecule generation, as it is not practical to query ground truth docking scores in this algorithm (due to the resource-intensive nature of running wet lat experiments). Additionally, the best molecules in the training dataset of the predictor were used as the starting population for Graph GA, which appeared to work better than using a randomly sampled starting population.

We observed that on average, this method with Bayesian Optimization training data collection resulted in better Top 1, Top 10 average, and Top 100 average docking scores of generated molecules than this same method with random data collection for most parameter settings across both molecule libraries tested (ZINC 250K and GuacaMol ChEMBL). Additionally, for all Bayesian Optimization parameter settings across both molecule libraries tested, molecules generated by Graph-GA had better Top 1 and Top 10 average docking scores than the collected training data. However, these generated molecules always had worse top 100 average docking scores than the collected training data.

Overall, these results indicate that the proposed method could be a way to realistically integrate Graph-GA, an iterative method that requires many calls to a scoring function for which ground truth values are determined in a wet lab, into a *de novo* generation pipeline. In particular, Graph GA seems to improve upon the performance of Bayesian Optimization and can be seen as an extra optimizing step on top of it. Once the ground truth docking scores of the molecules generated by Graph GA are determined in a wet lab, the best molecules in this set can serve as drug leads than can be further investigated and improved.

5.2 A Note on Evaluation

We note that using a property predictor as a scoring function is not the only way to realistically integrate an iterative generation method into a *de novo* generation pipeline. Another way to do this is to use computational docking software as a scoring function. This software is computationally intensive, but it can still serve as a good (but noisy) proxy for a wet lab experiment. However, it is not straightforward to evaluate this method compared to the method proposed in this thesis without using wet lab experiments. This is because in our experiments, we used the output of computational docking software *as if* it were the result of a wet lab experiment, meaning that we treated it as if it was not noisy. Therefore, simply running Graph GA with this same docking software as a scoring function and comparing it to our method is not an accurate comparison, as this comparison would treat the Graph GA as if it is able to arbitrarily query the results of a wet lab experiment (which is not realistic).

One way to potentially compare these methods without using any wet lab experiments is to assume some noise distribution of the docking software (e.g., a normal distribution with mean 0 and some standard deviation σ). Then, Graph GA can be run with a scoring function that is the output of the docking software plus a value sampled from the noise distribution, and its results could be compared to our method. However, the most accurate way to compare the two methods is to run the method proposed in this thesis using a wet lab for data labeling and compare it to using a docking software scoring function in Graph GA.

5.3 Extensions

It would be interesting to also investigate the performance of using Bayesian Optimization to find an ideal starting population for Graph GA, as in our pipeline, and then using docking software instead of a trained predictor as the scoring function in Graph GA. Additionally, it could be interesting to use samples from a generative model (e.g., [15], [21]) trained on a molecule library instead of the molecule library itself as a data source to select points from. The sampling from the generative model could perhaps result in better molecules than those present in the library being available for selection. Finally, it would be interesting to see how iterative *de novo* generation methods other than Graph GA perform when using a Bayesian Optimization trained property predictor as a scoring function.

References

[1] Amini, A., Schwarting, W., Soleimany, A., & Rus, D. (2020). Deep evidential regression. *Advances in Neural Information Processing Systems*, 33, 14927–14937.

[2] Bertin, P., Rector-Brooks, J., Sharma, D., Gaudelet, T., Anighoro, A., Gross, T., Martinez-Pena, F., Tang, E. L., Regep, C., Hayter, J., et al. (2022). Recover: sequential model optimization platform for combination drug repurposing identifies novel synergistic compounds in vitro. *arXiv preprint arXiv:2202.04202*.

[3] Brown, N., Fiscato, M., Segler, M. H., & Vaucher, A. C. (2019). Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3), 1096–1108.

[4] Chhabra, M. (2021). Chapter 6 - biological therapeutic modalities. In Y. Hasija (Ed.), *Translational Biotechnology* (pp. 137–164). Academic Press.

[5] Cieplinski, T., Danel, T., Podlewska, S., & Jastrzebski, S. (2020). We should at least be able to design molecules that dock well. *arXiv preprint arXiv:2006.16955*.

[6] Coley, C. W., Eyke, N. S., & Jensen, K. F. (2020). Autonomous discovery in the chemical sciences part ii: Outlook. *Angewandte Chemie International Edition*, 59(52), 23414–23436.

[7] Dai, H., Dai, B., & Song, L. (2016). Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning* (pp. 2702–2711).: PMLR.

[8] Frazier, P. I. (2018). A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*.

[9] Fu, T., Gao, W., Xiao, C., Yasonik, J., Coley, C. W., & Sun, J. (2021). Differentiable scaffolding tree for molecular optimization. *arXiv preprint arXiv:2109.10469*.

[10] Graff, D. E. & Coley, C. W. (2021). pyscreener: A python wrapper for computational docking software. *arXiv preprint arXiv:2112.10575*.

[11] Guedes, I. A., Krempser, E., & Dardenne, L. E. (2017). Dockthor 2.0: a free web server for protein-ligand virtual screening. *XIX SBQT–Simpósio Brasileiro de Química Teórica*.

[12] Huang, K., Fu, T., Gao, W., Zhao, Y., Roohani, Y., Leskovec, J., Coley, C. W., Xiao, C., Sun, J., & Zitnik, M. (2021). Therapeutics data commons: machine learning datasets and tasks for therapeutics. *arXiv preprint arXiv:2102.09548*.

[13] Jensen, J. H. (2019). A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12), 3567–3572.

[14] Jin, W., Barzilay, R., & Jaakkola, T. (2018). Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning* (pp. 2323–2332).: PMLR.

[15] Jin, W., Barzilay, R., & Jaakkola, T. (2020a). Hierarchical generation of molecular graphs using structural motifs. In *International Conference on Machine Learning* (pp. 4839–4848).: PMLR.

[16] Jin, W., Barzilay, R., & Jaakkola, T. (2020b). Multi-objective molecule generation using interpretable substructures. In *International Conference on Machine Learning* (pp. 4849–4859).: PMLR.

[17] Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[18] Kusner, M. J., Paige, B., & Hernández-Lobato, J. M. (2017). Grammar variational autoencoder. In *International conference on machine learning* (pp. 1945–1954).: PMLR.

[19] Liu, X., Ye, K., van Vlijmen, H. W., IJzerman, A. P., & van Westen, G. J. (2019). An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: a case for the adenosine a2a receptor. *Journal of cheminformatics*, 11(1), 1–16.

[20] Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M. P., Mosquera, J. F., Mutowo, P., Nowotka, M., et al. (2019). Chembl: towards direct deposition of bioassay data. *Nucleic acids research*, 47(D1), D930–D940.

[21] Segler, M. H., Kogej, T., Tyrchan, C., & Waller, M. P. (2018). Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1), 120–131.

[22] Soleimany, A. P., Amini, A., Goldman, S., Rus, D., Bhatia, S. N., & Coley, C. W. (2021). Evidential deep learning for guided molecular property prediction and discovery. *ACS central science*, 7(8), 1356–1367.

[23] Srinivas, M. & Patnaik, L. M. (1994). Genetic algorithms: A survey. *computer*, 27(6), 17–26.

[24] Sterling, T. & Irwin, J. J. (2015). Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11), 2324–2337.

[25] Trott, O. & Olson, A. J. (2010). Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2), 455–461.

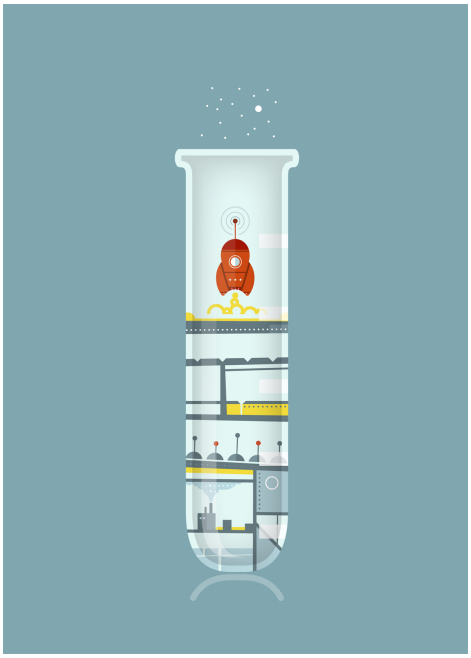
[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[27] Xie, Y., Shi, C., Zhou, H., Yang, Y., Zhang, W., Yu, Y., & Li, L. (2021). Mars: Markov molecular sampling for multi-objective drug discovery. *arXiv preprint arXiv:2103.10432*.

[28] Xu, P., Huang, S., Mao, C., Krumm, B. E., Zhou, X. E., Tan, Y., Huang, X.-P., Liu, Y., Shen, D.-D., Jiang, Y., et al. (2021). Structures of the human dopamine d3 receptor-gi complexes. *Molecular Cell*, 81(6), 1147–1159.

[29] Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., et al. (2019). Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8), 3370–3388.

[30] You, J., Liu, B., Ying, R., Pande, V., & Leskovec, J. (2018). Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*.



This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface. The above illustration, *Science Experiment 02*, was created by Ben Schlitter and released under CC BY-NC-ND 3.0. A template that can be used to format a PhD dissertation with this look & feel has been released under the permissive AGPL license, and can be found online at github.com/suchow/Dissertate or from its lead author, Jordan Suchow, at suchow@post.harvard.edu.