Deep Generative Models for Molecular Optimization

Guy Barshatski
Deep Generative Models for Molecular Optimization

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Guy Barshatski

Submitted to the Senate of the Technion — Israel Institute of Technology
Elul 5781 Haifa August 2021
This research was carried out under the supervision of Dr. Kira Radinsky, in the Faculty of Computer Science.

Some results in this thesis have been published as articles by the author and research collaborators in conferences during the course of the author’s research period, the most up-to-date versions of which being:


Acknowledgements

I would first like to thank my supervisor, Dr. Kira Radinsky, for the continued guidance, support and the journey we have taken together. You taught me with endless tolerance how to research, which questions should be asked, and above all, you were, and continue to be, my role model. I am beyond grateful to you for everything you have taught me.

I would like to thank my beloved family for the help, support and education I have received all these years. To my lovely fiancee, Shani, for her continuous support, advice, acceptance and for always being by my side, unconditionally. Last but not least, my dear parents, Lili and Vladimir, for the endless support and encouragement, for instilling in me the drive for excellence and inspiring me with an endless thirst for knowledge which led me to this point.

This thesis is dedicated to you.

The generous financial help of the Technion is gratefully acknowledged.
Contents

List of Figures

Abstract 1

1 Introduction 3

2 Related Work and Background 7
  2.1 Molecule Generation ........................................... 7
  2.2 Molecule Single-Property Optimization ........................... 7
  2.3 Molecule Multi-Property Optimization ............................ 8
  2.4 Drug Development ............................................ 9
  2.5 Drug Discovery ............................................. 9
  2.6 Molecular Representation - SMILES and Fingerprints .......... 10

3 Unpaired Generative Molecule-to-Molecule Translation for Lead Optimization 13
  3.1 Methods ......................................................... 13
    3.1.1 Molecule-Embedding Translation Network (METN) ........ 14
    3.1.2 Embedding-Embedding Translation Network (EETN) ....... 15
    3.1.3 End-to-End Architecture ................................ 16
  3.2 Experimental Settings ........................................ 19
    3.2.1 Implementation Details .................................. 19
    3.2.2 Datasets .................................................. 20
    3.2.3 Baseline Methods ......................................... 21
  3.3 Experiments and Results ...................................... 22
    3.3.1 Main Result: Molecule Optimization ....................... 22
    3.3.2 Ablation Experiments .................................... 24
    3.3.3 Optimized Drug Generation ............................... 26
  3.4 Appendices ................................................... 29
    3.4.1 Full Datasets Details .................................... 29
    3.4.2 Add Adversarial Ablation Experiment Details ............. 30
4 Multi-Property Molecular Optimization using an Integrated Poly-Cycle Architecture  33

4.1 Methods  ......................................................... 33
  4.1.1 Molecular Optimization Path  ................................ 34
  4.1.2 Multi-Property Optimization  ................................. 35
  4.1.3 Adaptive Loss  ................................................. 38
  4.1.4 METN for Molecule-Embedding Translation  ................... 38

4.2 Experimental Settings  ............................................. 39
  4.2.1 Datasets  ......................................................... 39
  4.2.2 Metrics  ........................................................... 39
  4.2.3 Empirical Methodology  ........................................ 39
  4.2.4 Baselines  ......................................................... 40
  4.2.5 Implementation Details for IPCA  ............................. 40

4.3 Experiments and Results  ............................................ 42
  4.3.1 Main Result  ....................................................... 42
  4.3.2 Qualitative Examples  .......................................... 44
  4.3.3 Ablation Tests  .................................................. 46

5 Conclusions and Future Directions  49

Hebrew Abstract  i
## List of Figures

1.1 FDA approved drugs generated using UGMMT. ................................................. 4

2.1 Drug Discovery process. ......................................................................................... 10

2.2 SMILES representation of Aspirin. Numbers and arrows represent the order of graph traversal, where 1 is the initial node. ................................. 11

2.3 Molecular Fingerprint. ......................................................................................... 11

3.1 METN. Left: Discrete molecule domain $Y$ represented by SMILES strings. Right: Molecule’s continuous representation space $\langle Y \rangle$, i.e. embedding. 14

3.2 Molecule-to-Molecule end-to-end architecture. Consists of two METNs (two sets of $En + De$) and an EETN (two $Ts$). The two training paths are marked with dotted arrows while the inference paths are marked with thick solid arrows. .................................................. 16

4.1 IPCA end-to-end architecture. The training paths are marked with dotted arrows while the inference paths are marked with thick solid arrows. 34

4.2 IPCA end-to-end architecture with additional property $prop_3$. Encoder $En_{B_3}$ and translator $T_{B_3,A}$ are added to support the additional property. The same approach may be used to support more properties. .......... 35

4.3 Success rate as a function of the number of training examples confirming to all properties. ................................................................. 42

4.4 Sample of molecules generated using IPCA $70,30$. The generated molecules preserve high similarity to the lead molecules. ............................... 45

4.5 Sample of molecules generated using IPCA $70,30$. The lead molecules have low QED and DRD2 properties. ................................................. 45

4.6 Sample of molecules generated using IPCA $70,30$. The generated molecules have high QED and DRD2 properties. ....................................... 46

4.7 Unified target domains ablation architecture. Marked in yellow is the unified component. .............................................................. 47
Abstract

Molecular lead optimization is an important task of drug discovery focusing on generating novel molecules similar to a drug candidate but with enhanced properties. Prior works focused on supervised models requiring datasets of pairs of a molecule and an enhanced molecule. These approaches require large amounts of data and are limited by the bias of the specific examples of enhanced molecules. In this thesis, we first tackle the single property molecule optimization problem and present an unsupervised generative approach with a molecule-embedding component that maps a discrete representation of a molecule to a continuous space. The components are then coupled with a unique training architecture leveraging molecule fingerprints and applying double cycle constraints to enable both chemical resemblance to the original molecular lead while generating novel molecules with enhanced properties. We evaluate our method on multiple common molecular optimization tasks, including dopamine receptor (DRD2) and drug likeness (QED), and show our method outperforms previous state-of-the-art baselines. Moreover, we conduct thorough ablation experiments to show the effect and necessity of important components in our model. Furthermore, we demonstrate our method’s ability to generate FDA-approved drugs it has never encountered before, such as Perazine and Clozapine, which are used to treat psychotic disorders, like Schizophrenia. The system is currently being deployed for use in the Targeted Drug Delivery and Personalized Medicine laboratories generating treatments using nanoparticle-based technology.

Next, since often molecules that satisfy multiple constraints are needed, e.g. DRD2 and QED, we focus on multi-property optimization. Simultaneously optimizing these constraints was shown to be difficult, mostly due to the lack of training examples satisfying all constraints. In this thesis, we present a novel unpaired approach for multi-property optimization. Our architecture learns a transformation for each property optimization separately, while constraining the latent embedding space between all transformations. This allows generating a molecule which optimizes multiple properties simultaneously. We present a novel adaptive loss which balances the separate transformations and stabilizes the optimization process. We evaluate our method on optimizing for two properties: dopamine receptor (DRD2) and drug likeness (QED), and show our method outperforms previous state-of-the-art, especially when training examples satisfying all constraints are sparse.

This thesis is the first work to demonstrate a unique dual learning style training
leveraging shared translation components and molecules’ fingerprints for molecular optimization.
Chapter 1

Introduction

Single-Property Molecular Optimization

Drug development is highly resource intensive, it might take over 10–15 years and more than 2 billion dollars. Potential drug leads are located in a process known as drug discovery. A common approach is to screen enormous number of chemical compounds in high throughput screening (HTS) assays to identify a modification in the target activity. This task is extremely challenging as the number of synthetically valid chemicals which are potentially drug-like molecules is estimated to be between $10^{23} - 10^{60}$ [PMV13]. To overcome this challenge, a lot of focus was recently aimed at the task of lead optimization [JBJ18, JYBJ19, FXS20, MPK+20]. In this task a lead compound is first identified by HTS or using chemistry expertise to have some of the desired properties. Then, it is improved to create an optimal substance which will become a potential drug candidate. The goal is to explore similar molecules with better properties than the original candidate.

Initial machine-learning modeling of the problem were unsupervised approaches [JBJ18, MPK+20], that showed low-performance results. The current state-of-the-art (SOTA) approaches are supervised requiring the acquisition of a set of paired molecules – the original molecule and an enhanced molecule with more desirable properties. The goal is to learn a generative model to produce novel molecules with similar chemical properties to the input molecule but with enhanced properties. Specifically, SOTA approaches for this task formulate the problem as a graph-to-graph translation problem [FXS20, JYBJ19].

One of the drawbacks of the paired approaches is the need for paired data for training, thus forcing a specific mapping between source and destination molecules although there may be various ways to translate an input molecule to other molecules with improved property. Furthermore, sometimes paired data in sufficient amounts is not available. In this work, we present an architecture that does not require any paired data reaching significant performance gains over graph-to-graph translation models.

We present the Unpaired Generative Molecule-to-Molecule (UGMMT) model – an
end-to-end generative model trained to generate novel discrete molecules with desired properties. The architecture is composed of three components: (1) Given a discrete string representation of a molecule, it generates the molecule’s continuous embedding. In this work, we leverage SMILES [Wei88] for the molecule discrete representation – a popular ASCII string molecular representation that preserves the molecule’s chemical information. (2) Using a translation component trained under double cycle constraints, it converts the continuous representation to an embedded molecule with desired properties. A novel molecular attention mechanism is added to drive the molecule generation to keep similarity to the original molecule’s chemical fingerprints (fp). (3) Converting the embedded molecule back to a discrete SMILES string representing a molecule with the desired properties.

We perform empirical evaluation of our model on several common lead optimization tasks and present significant performance gains compared to various SOTA. In addition, we conduct comprehensive ablation experiments to further analyse our method. We also present the system ability to generate FDA-approved drugs, although it has never seen a drug before. Figure 1.1 presents several such examples. D2 is the main receptor for most antipsychotic drugs [WCL+18]. Any disorder in equilibration of the D2 receptor’s states may lead to diverse serious disorders, such as Schizophrenia, autism and Parkinson’s disease. DRD2 score is a measure of a molecule’s biological activity against a biological target named the dopamine type 2 receptor. As input, our algorithm received a molecule with low DRD2 score and the goal was to generate a molecule with higher DRD2 score. All drugs have never been seen by our model during training, nor validation. In Fig. 1.1A, given Alimemazine (DRD2 score 0.3749), the algorithm generated Perazine (DRD2 score 0.6184) reaching 40% fingerprint similarity to Alimemazine. Both drugs are phenothiazine derivatives and structurally related to Chlorpromazine, an older remedy for Schizophrenia. In Fig. 1.1B, Clozapine was generated using the
prototype drug Anagrelide with 20.96% similarity and DRD2 score improvement from 0.0009 to 0.8412. Clozapine is used for treatment-resistant Schizophrenia and is on the World Health Organization’s list of "essential, safest and most effective medicines” [Org19]. Interestingly, all drugs generated by our model to improve DRD2 score are indeed antipsychotic drugs.

The contributions of this work are threefold:

- We design a novel unsupervised end-to-end generative model and a unique double-cycle training scheme for molecule translation preserving molecule similarity using molecular fp utilization, without the need of a large paired dataset.

- We present empirical results on the DRD2 (dopamine receptor) and QED (drug likeness) properties and demonstrate superior performance over SOTA baselines in generating novel, chemically similar molecules with higher property scores and success rates.

- We perform retrospective experiments to demonstrate our model’s ability to produce FDA approved drugs, it has never seen before. Additionally, in collaboration with the Technion laboratory for targeted drug delivery and personalized medicine technologies, additional generated molecules are being tested today for impact for personalized treatments.

We believe our method lays the foundations to an automatic-algorithmic HTS process to enable lead optimizations. All code and data in this work are published on our GitHub1 for further research.

Multi-Property Molecular Optimization

Following the development of the single property optimization system, we noticed that although prior work mostly focused on optimizing a single property while keeping similarity to the original lead molecule, in real applications, there is often a need to generate molecules that satisfy multiple constraints, e.g., potency and safety.

This task was shown to be challenging [JBJ19], as those approaches not only require large training sets, but also significant number of training instances of molecules conforming to all the constraints simultaneously. The latter is highly complicated to obtain in nature for most molecular properties. For example, the authors of [JBJ19] report that only 1.6% of the training pairs are both drug-like and DRD2-active – the main properties molecular optimizations focus on due to their significance to the drug discovery industry [JBJ19, JYBJ19].

In this work, we present a novel Integrated Poly-Cycle Architecture (IPCA) for multi-property optimization. The architecture learns parallel transformations for each optimization property while constraining the transformations to maintain a single

1https://github.com/guy-ba/UGMMT
multi-property optimized molecule rather than an optimized molecule for each property. Our approach does not require any paired data and is less sensitive to lack of training instances that satisfy all target properties. Similarly to the single property optimization case, our model belongs to the family of models that optimize molecules given their SMILES representation. For each target property an encoder is trained to translate a discrete molecule to a continuous representation and a translator to translate this embedding to an embedded source domain. Then, we concatenate these components to a central translator, shared by all properties, forcing a common optimized embedding space, and a central decoder to translate it back to the target domain. Thus a cycle is formed for each property. We use these multiple cycles to train our model. At test time, given a discrete source molecule, we first embed it using the encoder, then use the shared translator to map it to a continuous representation of a molecule which has the enhanced properties. Finally, we use the shared decoder to generate an optimized molecule. Following UGMMT, molecule fingerprint is supplied to the translator in order to encourage a molecular similarity preservation to the source molecule. As different properties induce different optimization paths, with different training complexity, we present a novel adaptive loss, which adjusts components’ relative importance by adapting their coefficients during learning.

The contributions of this work are threefold:

- We design a novel unpaired end-to-end generative model and a unique poly-cycle training scheme for molecule translation preserving molecule similarity while optimizing multiple molecular properties. To the best of our knowledge, our work is the first in the family of models considering the SMILES molecular representation to tackle the multi-property optimization problem.

- We present an adaptive loss function that allows to balance the multi-property optimization during the training, thus yielding superior results.

- We present empirical results on numerous datasets and optimize multiple levels of DRD2 (dopamine receptor) and QED (drug likeness) properties. We demonstrate superior performance over SOTA baselines in generating molecules with higher property scores and success rates, especially notable when few examples confirming to multiple properties exist in the data. We perform ablation tests studying the behavior of IPCA in different settings.

All code and data in this work are published on our GitHub for further research.

---

2https://github.com/guy-ba/IPCA
Chapter 2

Related Work and Background

2.1 Molecule Generation

Numerous previous works focused on generating molecules. The models can be characterized by the molecular representation they generate. Some of these formulated molecule generation as a sequence generation problem, representing a molecule as a sequence of characters. Most commonly, SMILES notation was used and showed merit for this task [DTD+18, KPHL17, GBWD+18, HR18, PIT17, OBEC17], others used molecular graphs [SK18, JBJ20].

The generation process was formulated either by leveraging variational autoencoders [DTD+18, KPHL17, GBWD+18, HR18, SK18] or reinforcement learning [OBEC17, PIT17, JBJ20].

Most of the aforementioned models mainly attempt to generate valid molecules without the constraint of similarity to an initial molecular lead. These models overcome the initial task of generating valid SMILES sequences or molecule graphs, but without specifically addressing the optimization task and similarity to the lead, their performance is poor [FXS20].

2.2 Molecule Single-Property Optimization

Is a separate task from molecule generation as it requires to not only generate an enhanced molecule but also to keep similarity to the original molecule lead. Molecular optimization methods mainly focused on graph-to-graph translation methods. Those represent molecules as graphs and try to translate input molecular graphs into improved graphs.

JTVAE [JBJ18] presents an encoder-decoder architecture for graph-to-graph translation and molecule optimization. Given input molecule’s graph, the encoder generates a tree where each node represents molecule’s substructure, then embeds both to get the latent representation. The decoder reproduces the tree and uses it to predict the output molecular graph. Optimization is done by first training a property score predic-
tor on top of JTVAE’s latent space, then gradient ascent is applied on input molecule’s embedding to improve its score. [MPK’20] suggest Mol-CG as an extension of JTVAE. Instead of using gradient ascent, the model splits the embedded molecules to two distinct sets – with and without the enhanced property. Two sets of generators and discriminators are applied in an adversarial manner introduced by [ZPIE17] to perform the optimization. At test time, the generated embeddings are decoded back to molecules using the same JTVAE latent space decoder. Although showing initial success, the results were not sufficient for practical molecular optimization.

To improve the performance, numerous supervised methods have been suggested requiring substantial amount of training data. [JYBJ19] (Graph-to-Graph) extended JTVAE by presenting attention mechanism during the tree decoding procedure and integrating adversarial training to further improve performance and thus generate molecules with improved desired property. [FXS20] (CORE) extended the above by introducing the copy&refine technique where the generator at each step decides whether to copy the next substructure from the input molecule or to generate a new substructure. This relaxes the problem of challenging substructure prediction from a large set in every iteration, which is especially problematic for infrequent substructures. These approaches require paired data for training, thus forcing a specific mapping between source and destination molecules although there may be various ways to translate an input molecule to other molecule with an improved property. Furthermore, sometimes paired data in sufficient amounts is not available.

In the first part of this thesis, we present an unsupervised approach (UGMMT), requiring no paired-data, which reaches SOTA results outperforming the supervised paired-approaches. We present a novel architecture which does not rely on molecular graph structures, but rather leverages raw SMILES representations, embeds them and learns to efficiently translate them using a novel molecular attention driven by molecular fp to an optimized SMILES molecule.

### 2.3 Molecule Multi-Property Optimization

The aforementioned methods do not address the task of optimizing multiple properties simultaneously. Jin el al. [JBJ19] was the first to address the problem of multi-property optimization. Their work extends the graph-to-graph translation model incorporating conditional translation. The translation condition is a vector describing the required property values after translation (e.g., 11 for high QED and high DRD2, 01 for low QED and high DRD2, etc.) However, these models require large training sets and fail to perform well when the datasets contain small number of molecules that comply to all constrains. In the second part of this thesis, we present a novel architecture in which the property optimization translators are learned in parallel while being constrained by two factors: (1) a cyclic constraint on each translator maintaining similarity between the source molecule and its optimization, (2) the latent space from encoder to decoder...
is shared between translators thus creating one molecule that is optimized in multiple aspects. We show our model reaches SOTA results and is not as sensitive to the existence of molecules that comply to all constrains in the dataset. To the best of our knowledge, our work is the first to tackle the multi-property molecular optimization in the family of models considering the SMILES representation.

2.4 Drug Development

Drug development is designed to ensure that innovative new medicines are effective, safe and available for patients in the shortest possible time. Discovering and bringing one new drug to the market typically takes an average of 14 years of research and clinical development efforts and cost around 2 billion U.S. dollars.

There are 3 main steps in Drug development:

- Stage 1 - Drug Discovery in which we identify the lead drug candidate.
- Stage 2 – Pre-Clinical development in which the drug candidate is tested for safety and desired properties in the laboratory and on animals.
- Stage 3 – Clinical Development in which safety and effectiveness are being tested on people.

In this thesis we focus on Drug Discovery.

2.5 Drug Discovery

Drug discovery is a long and winding road starting from a disease we want to treat. The steps are described in Figure 2.1 [HRKP10].

The first step is to identify the best target for treating or preventing the disease. Targets are usually proteins in a patient’s body that are associated with a disease or causing a disease. The challenge is to identify which targets are relevant and to confirm and demonstrate their role in a disease process. That is done in the target validation step. Having identified and validated the target, one then goes through a process known as hit identification. The role of hit identification is to find compounds that actually engage the identified target, i.e., compounds which bind to the protein of interest. These are called hits. A common approach is to screen enormous number of chemical compounds in high throughput screening (HTS) assays to identify a modification in the target activity. This task is extremely challenging as the number of synthetically valid chemicals which are potentially drug-like molecules is practically infinite. To overcome this challenge, a prototype based technique may be applied. In this approach, we start from a molecule which already has some of the desired properties identified previously by HTS or by experts. Then in the hit to lead step, the hits are tested for having various
properties such as efficacy, potency, water solubility, chemical stability, Lipinski’s “rule-of-five”, toxicity and many more. The most promising hits are selected to advance into the lead optimization stage where the leads are chemically modified in order to improve their biological activity and other critical properties such as efficacy or reducing adverse effects. The optimized leads eventually become drug candidates. In this thesis we focus on the lead optimization step. The overall goal of lead optimization is to explore and generate similar molecules with better properties than the original lead molecule. Note that generally, lead optimization is extremely challenging since there is no certain way to modify a molecule to improve its properties. As a result, its highly time consuming, costly and often it is even the tightest bottleneck in drug discovery.

2.6 Molecular Representation - SMILES and Fingerprints

There are two common approaches to represent molecular data. The first is character based, it can be seen as a chemical language using a sequence of characters to represent a molecule. The most popular is SMILES notation [Wei88, DTME20]. Another well known way to represent molecules is by using mathematical graphs where vertices represent atoms and edges represent bonds.
There is a similarity between SMILES and natural language. It strengthen our belief that advances in natural language processing domain may be successfully harnessed to manipulate SMILES for molecular optimization. Hence, in this work we focus on SMILES notation for molecular representation.

SMILES stands for Simplified Molecular-Input Line-Entry System. It is a highly informative, easy to manipulate, compact language which can represent atoms, bonds, rings, aromaticity and much more. For a given molecular graph, its SMILES string is obtained by printing the symbol nodes encountered in depth-first traversal (DFS) of the graph. This is demonstrated in Figure 2.2, where the molecular graph of Aspirin and its SMILES representation are shown. The numbers on the graph and the green arrows indicate how the molecular graph is traversed. 1 is the initial node representing a Carbon atom contributing the first ‘C’ character in the SMILES string, from there we continue to another Carbon atom contributing the next ‘C’ in the SMILES, enter to a branch containing Oxygen atom (next ‘O’ in the SMILES) and so on. In the SMILES string, brackets indicate branches and numbers represent the entry and exit points of rings.

Another tool we use in our work is molecular fingerprints [CMOV\textsuperscript{15}, RH10]. Given
a molecule, we use its fingerprint to encode its structural fragments as a vector. Most commonly – a series of binary digits that represent the presence or absence of particular substructures in the molecule. This is demonstrated in Figure 2.3. According to the similar property principle, similar molecules tend to exhibit similar properties [JM90]. In addition, molecular fingerprints play a major role in many tasks like mapping chemical space, chemical property prediction and more. Hence we hypothesize that leveraging molecule’s fingerprints during its mapping to another domain will help preserving its original chemical and structural properties while improving the desired property.

Molecular fingerprints are also commonly used for measuring molecular similarity. If two molecular fingerprints have 1’s at the same position, then both molecules have the same fragment, and the more fragments they share, the more similar they are considered. The most popular way to measure the similarity of molecular fingerprints is by computing the Tanimoto coefficient [BRH15]. The Tanimoto coefficient measures the ratio of the number of fragment positions shared by the molecules divided by the fragment positions set by either molecule. Ranges from 0 for molecules with no shared fragments to 1 for molecules that share all fragments.
Chapter 3

Unpaired Generative Molecule-to-Molecule Translation for Lead Optimization

In the first part of this thesis, we tackle the single property molecule optimization problem and present the Unpaired Generative Molecule-to-Molecule (UGMMT) model – an end-to-end generative model trained to generate novel discrete molecules with a desired property.

3.1 Methods

We denote a domain of molecules by a capital Latin letter, e.g. $X$, their distribution by $p(X)$, and a molecule taken from this domain by a small Latin letter, $x$. That is, $x \in X$ denotes $x$ is a sample of molecules taken from domain $X$ and consequently $x \sim p(X)$. Similarly, we denote the embedding vector of a molecule $x$ by $\langle x \rangle$. It belongs to the domain of all the embedded molecules of $X$, which we denote by $\langle X \rangle$ with a distribution $p(\langle X \rangle)$. Given a domain $A$ of input molecules with some property and distribution $p(A)$, and a domain of the enhanced molecules, $B$, with distribution $p(B)$ our goal is to learn a mapping $M : A \rightarrow B'$, s.t. the distribution of molecules $B' = M(A)$, $p(B')$, is indistinguishable from the distribution $p(B)$. To achieve it and guarantee that the input molecule $a \sim p(A)$ and output $b' = M(a) \sim p(B')$ carry chemical molecular similarity, we enforce double cycle constraints, that share common translation components strongly relying on molecular fp. This procedure can be applied if molecules are represented in a continuous space. We therefore first present a Molecule-Embedding translation, where a discrete molecule’s SMILES representation is transformed to an embedding and vice-versa (Section 3.1.1). We then present the Embedding-Embedding translation where embedded prototype molecule is translated to an embedded desired molecule and vice-versa (Section 3.1.2). Lastly, we provide the overall end-to-end architecture of our model (Section 3.1.3).
3.1.1 Molecule-Embedding Translation Network (METN)

We propose METN to translate between molecule’s discrete SMILES representation and its continuous representation, i.e., embedding. The general structure is depicted in Figure 3.1. Given a molecule’s SMILES representation (denoted by \( y \in Y \)), an encoder generates its embedding (denoted by \( \langle y \rangle \in \langle Y \rangle \)) and a decoder reverts the process – generating molecule’s SMILES string given its embedding. This network is based on Variational Auto Encoder (VAE) [KW14] structure with a bidirectional GRU-based encoder and a multilayered GRU-based decoder [CvMG+14] creating a sequence-to-sequence translation trained using the teacher-forcing method [WZ89] in order to increase validity. VAEs add stochasticity to the generation process adding variation to the latent representation learning and thus forcing the decoder to learn how to decode a wide range of latent points better suited for embeddings. In addition, to make the latent space dense, a KL divergence term is added to the loss function, encouraging the encoder to distribute molecule encodings according to a known prior distribution. METN’s training algorithm is shown in Algorithm 3.1. \( \lambda_0 \) is a regularization parameter, CE is the cross-entropy loss (reconstruction loss), KL is the KL-divergence (has close form in this case of normal distribution) and the loss is averaged over the number of samples in the mini-batch. In our system, we have two input molecule domains, \( A \) and \( B \). Embedding all the molecules using only one METN would make the overall model training and convergence more difficult yielding worse results since the latent embedding space would not only have to represent molecular similarity but also molecule property. Hence, we enable “domain embedding specialization” by training two METNs, one for each domain as depicted in Figure 3.2. We denote domain \( A \) encoder as \( E_{n_A} \). The encoder, given a molecule \( a \in A \), generates its embedding \( \langle a \rangle \in \langle A \rangle \) (a continuous vector), while the decoder, \( D_{e_A} \), converts molecule continuous

![Figure 3.1: METN. Left: Discrete molecule domain \( Y \) represented by SMILES strings. Right: Molecule’s continuous representation space \( \langle Y \rangle \), i.e. embedding.](image-url)
embedding \( \langle a' \rangle \in \langle A' \rangle \) to its SMILES \( a' \in A' \). We expect that \( a' \sim p(A) \), i.e., \( a' \) has domain’s \( A \) property or alternatively, the distribution of the decoded molecules domain \( A' \), \( p(A') \), is indistinguishable from \( p(A) \). Similar notations are used for domain \( B \).

### 3.1.2 Embedding-Embedding Translation Network (EETN)

We propose EETN to translate between embedded molecules that belong to different domains, e.g. low and high DRD2 score. Following our notations, the EETN is composed of two translation networks, \( T_{AB} \) for \( \langle A \rangle \rightarrow \langle B' \rangle \) translation and \( T_{BA} \) for \( \langle B \rangle \rightarrow \langle A' \rangle \), the opposite direction. Figure 3.2 illustrates the architecture. We mark in dotted lines the training paths and in thick lines the inference paths. Our design for the EETN allows it to meet two main goals – property enhancement and preservation of the input-output molecular similarity. Given \( \langle a \rangle \), the EETN attempts to produce an embedded molecule \( \langle b' \rangle \), s.t. \( b' = De_B(\langle b' \rangle) \sim p(B) \), i.e., \( b' \) has domain’s \( B \) enhanced property (e.g. high DRD2 score). \( De_B \) must be able to successfully decode \( \langle b' \rangle \) and \( T_{AB} \) must successfully translate \( \langle a \rangle \in \langle A \rangle \) into \( \langle b' \rangle \). During training, naively applying on \( a \) the sequence of components on the thick lines \( (En_A \rightarrow T_{AB} \rightarrow De_B) \) would not be successful. \( De_B \) is not necessarily producing valid molecules, therefore calculating their property score is not feasible. Furthermore, molecule’s property calculation is not differentiable, hence penalizing molecule’s property is not feasible for valid molecules as well. Therefore during training, we apply the sequence \( En_B \rightarrow T_{BA} \rightarrow T_{AB} \rightarrow De_B \) (purple dotted path) on \( b \). Requiring \( b' \approx b \) might successfully train \( De_B \), however \( T_{BA} \) would produce embedded molecules in domain \( \langle A' \rangle \), which may be distributed differently from \( \langle A \rangle \). Hence, we train two translation components simultaneously. In each iteration we train by applying two semi-shared sequences of components on the inputs \( a \) and \( b \). We apply the sequence \( En_B \rightarrow T_{BA} \rightarrow T_{AB} \rightarrow De_B \) (purple dotted path) on \( b \) and \( En_A \rightarrow T_{AB} \rightarrow T_{BA} \rightarrow De_A \) (red dotted path) on \( a \) and require \( b' \approx b \), and \( a' \approx a \). This coupling between the two translation sequences sharing \( T_{AB} \) and \( T_{BA} \), where one’s output is the other’s input, encourages proximity of the distribution of \( \langle A \rangle \) and \( \langle A' \rangle \). Intuitively, if \( T_{BA} \) translates \( \langle b \rangle \in \langle B \rangle \) to \( \langle a' \rangle \notin \langle A \rangle \) then \( T_{AB} \) (trained to translate \( \langle a \rangle \in \langle A \rangle \)) will perform poorly on the input \( \langle a' \rangle \) and produce \( b' \) notably different from \( b \), which will be penalized by the loss function. Applying these two sequences on the
inputs and demanding input-output proximity constitute our double cycle constraints, which we refer to as double-cycle training scheme. The technique can be considered a dual learning method \([\text{HXQ}^{+16}]\), which leverages primal (e.g. \(A \rightarrow B\)) and dual (e.g. \(B \rightarrow A\)) tasks to create an informative feedback loop.

To encourage similarity to the original molecule, we wish the translation to keep its chemical characteristics. We leverage Morgan fp \([\text{RH10}]\) to represent these characteristics. We introduce the input molecule fp to the translators (concatenated with the input embedding) during the training and the inference. As a result, their output latent embedding spaces become fp dependant, encouraging molecules with similar fp to be closely embedded. The translators \(T_{AB}\) and \(T_{BA}\) are designed to create a bottleneck by downsampling followed by upsampling architecture in order to extract vital information while dropping the redundant data. To make this process more efficient we pass the input molecule fp through an attention mechanism of a fully-connected (FC) layer followed by a softmax layer producing a weight vector, which multiplies the fp vector elementwise, highlighting the “important” information inside the fp vector. We refer to this component as molecular attention.

3.1.3 End-to-End Architecture

We introduce the UGMMT– an end-to-end unsupervised generative deep neural network architecture. The architecture is illustrated in Figure 3.2. The architecture consists of two METNs, one for domain \(A\) (blue) and one for domain \(B\) (green), and
an EETN (red) for molecular translation. The input molecules are given as discrete SMILES strings, however, EETN’s functionality described in Section 3.1.2 relies heavily on embedded molecule domains. Therefore, METNs enable converting discrete SMILES strings to continuous representations (embedding) and vise-versa.

Training:

We pre-train the METNs before training the end-to-end model to enable the preparation of the latent embeddings of domain A (e.g., low DRD2 scored molecules) and domain B (e.g., high DRD2 scored molecules) separately. We conjecture this allows the encoders (En_A and En_B) to produce better embeddings and the decoders (De_A and De_B) in turn would produce more valid molecules with the enhanced property. Our experiments (Section 3.3.2) support this hypothesis. We then perform end-to-end training, where the METNs and the EETN are trained together. Intuitively, this allows the embeddings generation and the translation between them to evolve simultaneously to decrease the overall loss. This enables generating a similar molecule with the desired

Algorithm 3.2 UGMMT Training algorithm. Corresponds to both dotted paths in Figure 3.2, the red path along En_A \(\rightarrow\) T_{AB} \(\rightarrow\) T_{BA} \(\rightarrow\) De_A and the purple path along En_B \(\rightarrow\) T_{BA} \(\rightarrow\) T_{AB} \(\rightarrow\) De_B.

**Input:** A_t, B_t molecule training sets. A_v molecule validation set.

1: Train domain’s A METN on A_t for E_{METN} epochs (Algorithm 3.1)
2: Train domain’s B METN on B_t for E_{METN} epochs (Algorithm 3.1)
3: for epoch = 1, 2, \ldots, E_{max} do
4: \hspace{1em} Sample mini-batches a \(\in\) A_t, b \(\in\) B_t
5: \hspace{1em} (a) = En_A(a), (b) = En_B(b)
6: \hspace{1em} Calculate f_{pa}, f_{pb}
7: \hspace{1em} (a') = T_{AB}((a), f_{pa})
8: \hspace{1em} (a') = T_{BA}((b'), f_{pa})
9: \hspace{1em} a' = De_A((a'))
10: \hspace{1em} L_1 = CE(a', a)
11: \hspace{1em} (a') = T_{BA}((b'), f_{pb})
12: \hspace{1em} (b') = T_{AB}((a'), f_{pb})
13: \hspace{1em} b' = De_B((b'))
14: \hspace{1em} L_2 = CE(b', b)
15: \hspace{1em} L = \lambda_1 \cdot L_1 + L_2
16: \hspace{1em} Minimize L using Adam optimizer
17: \hspace{1em} Evaluate the model every V_f epochs on A_v
18: \hspace{1em} if evaluation criterion improves then
19: \hspace{2em} save model
20: \hspace{1em} end if
21: \hspace{1em} if no improvement for P evaluations then
22: \hspace{2em} stop training
23: \hspace{1em} end if
24: end for
property. The detailed algorithm is presented in Algorithm 3.2. CE is the cross-entropy loss, $L_1$ and $L_2$ are the two cycle constraints (Section 3.1.2) and $\lambda_1$ is a regularization parameter that controls their relative importance. The loss is averaged over the number of samples in the mini-batch.

**Inference:** Given a SMILES-represented molecule $a$ to be enhanced from domain $A$, the $E_{n A}$ converts it to an embedding. The translator $T_{AB}$ then translates the embedding along with the molecule $fp$ (extracted from original molecule $a$ using RDkit) to an embedded molecule of domain $B'$. Finally, $D_{EB}$ converts it to a molecule in domain $B'$, represented by a SMILES string. We draw the reader attention that the double cycle constraints enforced during the training process enables bidirectional optimization, i.e., inference from $B$ to $A$ as well. Both optimization paths are shown in Figure 3.2, $A \rightarrow B$ in red thick arrows and $B \rightarrow A$ in purple thick arrows. Algorithm 3.3 presents the inference algorithm for $A \rightarrow B$ direction ($B \rightarrow A$ is symmetric).

**Algorithm 3.3 UGMT $A \rightarrow B$ Inference algorithm.** Corresponds to the red thick path along $E_{n A} \rightarrow T_{AB} \rightarrow D_{EB}$ in Figure 3.2.

**Input:** $a \in A$ mini-batch of molecules to optimize.

1: $\langle a \rangle = E_{n A}(a)$
2: Calculate $fp_a$
3: $\langle b' \rangle = T_{AB}(\langle a \rangle, fp_a)$
4: $b' = D_{EB}(\langle b' \rangle)$
3.2 Experimental Settings

We provide implementation and hyperparameter details for reproducibility, the datasets we use and the baselines we compare to.

3.2.1 Implementation Details

We run all training and experiments on Ubuntu 18.04.5 using one Nvidia GeForce RTX 2080 Ti 11GB GPU, two Intel Xeon Gold 6230 2.10GHZ CPUs and 64GB RAM memory. We use Pytorch 1.4.0 API [PGM+19], Python 3.6.12 for UGMMT model training and experiments. Other baselines are trained using their publicly available implementation with the setting they set and reported in their papers. Our code and data are publicly available on our GitHub. See file README.md for installation, training, ablation experiments and inference instructions. Our model depends on randomness, we set seed 50 for reproducibility, other seeds lead to close results as well. However, small variations in the results might occur following different training sessions. The source of variation is in the EETN networks, probably due to PyTorch functions that use CUDA functions that can be a source of non-determinism (e.g. atomic operations), it is a known problem in the community and we are trying to handle it. Hyper-parameters are set following evaluation on the validation set, where the optimization metric is the optimization Success criterion defined in Section 3.3.1. Values we experimented with are specified in the brackets beside each hyper-parameter.

**METN:** Latent space dimension (embedding dimension) 256 (experimented with 128, 256, 512), \( \lambda_0=10 \) (experimented with 1, 10, 20), max molecule length during inference 90 (experimented with 80-120). Encoder – bidirectional GRU with 1 layer, hidden dimension 256 \((En, h_d)\), followed by 2 fully-connected layers with 256 neurons, one for \( \mu \) and one for \( \sigma \). Decoder – fully-connected layer with 512 neurons, followed by a unidirectional GRU with 3 layers (experimented with 1-5), dropout 0.5 between layers, hidden dimension 512 \((De, h_d)\) and a fully-connected layer with vocabulary length number of neurons.

METN’s training: Optimizer – Adam algorithm [KB15]. Learning rate – Cosine annealing scheduler with restart [LH19] after 10 epochs, initial learning rate \(3 \cdot 10^{-3}\) (experimented with \(1 \cdot 10^{-3}-5 \cdot 10^{-3}\)), final learning rate \(3 \cdot 10^{-4}\). Weights initialization – GRU’s weights uniformly distributed ranging from \(-h_d^{-1}\) to \(h_d^{-1}\), linear layer’s weights uniformly distributed ranging from \(-f_m^{-1}\) to \(f_m^{-1}\), where \(f_m\) is the number of input features. Mini-batch size 32 and pre-trained for \(E_{METN} = 1\) (QED) (experimented with 0-10) or \(E_{METN} = 12\) (DRD2) (experimented with 0-20) epochs before training the whole model together. Trained using the teacher-forcing method [WZ89] in order to increase validity.

**EETN:** Translator – Bottleneck structure, i.e. downsampling followed by upsampling. Contains initial convolution block (filter size 7), stride-2 convolution block for downsampling (filter size 3), 4 residual blocks (experimented with 4-8), stride-2 transposed
convolution block for upsampling (filter size 3), final convolution layer (filter size 7) followed by Tanh layer and 2 fully connected layers with 1,152 and 256 neurons separated by batch normalization, LeakyReLU and Dropout (0.2) layers (all convolution blocks contain instance normalization and ReLU layers). These types of structures showed outstanding results in image translation and style transfer tasks [JAFF16, ZPIE17].

We pass molecule’s fingerprints through a basic attention mechanism consists of a fully-connected layer followed by a softmax layer to generate a weights vector, which multiplies the fingerprints vector elementwise, highlighting the “important” information inside the fingerprints vector. The embedded molecule and molecule’s weighted fingerprints vector are concatenated before applying the Translator.

**Whole model training:** Optimizer – Adam algorithm. Learning rate - LambdaLR scheduler with initial learning rate $3 \cdot 10^{-4}$ (experimented with $1 \cdot 10^{-4}$-$5 \cdot 10^{-4}$) and linear decay towards 0 from epoch 90 (experimented with 70-100). Weights initialization – from zero-centered normal distribution with a standard deviation of 0.02 [RMC16]. Mini-batch size 32. Maximal epochs for training $E_{max} = 120$ (experimented with 80-120). Training data is shuffled during training. In addition, to further reduce overfitting and improve generalization, we evaluate our model every $V_f = 3$ epochs on a validation set and save a checkpoint only if improvement in the criterion is achieved. Furthermore, we add early stopping mechanism, i.e. if this criterion does not improve for $P = 15$ evaluations, we stop the training. Regularization hyper-parameter $\lambda_1 = 2$ (experimented with 1,2,4,8,16).

### 3.2.2 Datasets

Following [JYBJ19, FXS20], we focus on two common properties which are vital for generated drug’s effectiveness evaluation:

1. **Dopamine Receptor D2 (DRD2):** DRD2 score measures molecule’s biological activity against a biological target named the dopamine type 2 receptor.

2. **Drug likeness (QED):** QED score [BPB+12] measures, intuitively, how “drug-like” a molecule is.

We use RDkit [Lan16] library to calculate these scores as done by [FXS20] in order to preserve comparability. We also use their datasets, which were adapted from [JYBJ19]. For UGMMT and Mol-CG we split the train pairs and randomly sample molecules with low property score (DRD2, QED) (lower decimal) for domain A set and high (high decimal) for B set. For CDN and JTVAE we merge these two sets to get the final train set. UGMMT’s validation set was also randomly sampled from the remaining low score molecules, for other baselines the original validation sets from [FXS20] are used. All models are tested on the same test set taken from [FXS20]. Publicly available DrugBank dataset [WKG+06], which contains a list of FDA approved drugs, is used for drug
generation experiments. We share the datasets publicly, please refer to Appendix 3.4.1 for more details.

3.2.3 Baseline Methods

We compare to SOTA methods for lead optimization. **CORE [FXS20]** is currently regarded as the SOTA, followed by **G2G [JYBJ19]**. Both methods are supervised, using paired train sets. UGMMT method is unsupervised, hence we include the leading unsupervised **JTVAE [JBJ18]** and **Mol-CG [MPK+20]** methods. Following [FXS20], we evaluate the checkpoints on the validation set for highest optimization success (Success), then evaluate on the test set and report the results. All models use datasets which are derived from CORE’s datasets with the same test set.
3.3 Experiments and Results

We conduct three main experiments. First, we evaluate our model’s ability to generate valid, novel, diverse molecules which have a desired property and share similarity with a prototype and compare it to the current SOTA methods (Section 3.3.1). Then, we conduct thorough ablation experiments to show the effect and necessity of important components in our model (Section 3.3.2). Afterwards, we investigate and demonstrate our model’s capability to generate FDA approved drugs with enhanced properties (Section 3.3.3).

3.3.1 Main Result: Molecule Optimization

Focusing on the $A \rightarrow B$ optimization direction, for each input test molecule we first generate $K$ output molecules using $K$ random seeds (following the practice of CORE [FXS20], in our experiments $K = 20$). We calculate and report:

• **Validity**: The proportion of the input test molecules which have at least one valid optimized molecule. The validity of a molecule is determined based on [Lan16].

After that, we randomly select one valid output molecule $b'$ for each input test molecule $a_{te} \in A_{te}$, if exists. Note, following [JYBJ19], we exclude the input molecules which do not have valid output molecules from the other metrics calculations in order to isolate these from the validity measure. Otherwise, models with lower validity will always have lower metrics. We repeat the random molecule selection and the following metrics calculations 10 times and report their mean and standard deviation values:

• **Property**: The average desired property score (QED or DRD2) of all the optimized $b'$ molecules. The property score of each $b'$ is denoted as $Prop(b') \in [0, 1]$.

• **Similarity**: The average similarity of all $(a, b')$ pairs. The similarity of each molecule pair is measured using Tanimoto similarity [BRH15] over their Morgan fingerprints and denoted as $Sim(a, b') \in [0, 1]$.

• **Novelty**: The proportion of all the optimized $b'$ molecules which are novel. $b'$ is novel if it has not appeared in the train set and $b' \neq a_{te}$.

• **Optimization Success (Success)**: The proportion of all the optimized $b'$ molecules which are successful. $b'$ is successful if it holds similarity, property and novelty simultaneously, i.e., $Sim(a, b') > \lambda_s$ and $Prop(b') > \lambda_p$ and $b'$ is novel. Following the practice of CORE [FXS20], we use their “success rate 1” $\lambda_s$ with their challenging “success rate 2” $\lambda_p$, yielding $(\lambda_s, \lambda_p) : (0.3, 0.8)$.

• **Diversity**: The proportion of all the optimized $b'$ molecules which are unique, i.e., the number of unique $b'$ molecules divided by the number of all $b'$ molecules.
Tables 3.1 and 3.2 present these results for QED and DRD2 properties, respectively. We observe that for QED, UGMMT has slightly lower validity in QED compared to the other graph-based methods, however, since invalid molecules are easily and automatically disqualified without a human intervention (using RDkit), we mainly focus on the other metrics. For DRD2, UGMMT’s validity is excellent. Our method outperforms the SOTA supervised methods in the main optimization success metric, i.e., it generates the highest number of successful molecules. For QED, the main contributors, according to table 3.1, are UGMMT’s better similarity and novelty. Whereas, for DRD2, according to table 3.2, these are the property and the novelty. For DRD2, UGMMT’s improved novelty stands out compared to the supervised methods. Although the supervised methods are able to generate high quality molecules, most of these have already been seen during the training so these models mainly memorize them. UGMMT also has better diversity, meaning that it generates more unique molecules for different input molecules and not generates the same molecule for many leads. The results are surprising as these baselines are trained in a supervised paired manner, where pairs of molecules are carefully selected s.t. their similarity and property improvements are extremely high. On the contrary, our method is unsupervised and only requires two sets of molecules with no similarity requirements. All methods are evaluated on the same test set. Our method also shows significant performance gains compared to

Table 3.1: Evaluation results of our method and other baselines for various metrics for QED property. In bold: best optimization Success result, statistically significant with huge effect size ($P-value < 0.05$ and Cohen’s $d > 2$).

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Property</th>
<th>Similarity</th>
<th>Novelty</th>
<th>Success</th>
<th>Validity</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paired</td>
<td>G2G</td>
<td>0.895 ± 0.001</td>
<td>0.320 ± 0.003</td>
<td>0.953 ± 0.004</td>
<td>0.452 ± 0.010</td>
<td>1.000</td>
<td>0.993 ± 0.002</td>
</tr>
<tr>
<td></td>
<td>CORE</td>
<td>0.882 ± 0.002</td>
<td>0.339 ± 0.005</td>
<td>0.963 ± 0.006</td>
<td>0.472 ± 0.012</td>
<td>1.000</td>
<td>0.997 ± 0.002</td>
</tr>
<tr>
<td>Unpaired</td>
<td>JTVAE</td>
<td>0.816 ± 0.000</td>
<td>0.304 ± 0.000</td>
<td>0.977 ± 0.000</td>
<td>0.237 ± 0.000</td>
<td>1.000</td>
<td>0.996 ± 0.000</td>
</tr>
<tr>
<td></td>
<td>Mol-CG</td>
<td>0.783 ± 0.000</td>
<td>0.302 ± 0.000</td>
<td>0.980 ± 0.000</td>
<td>0.170 ± 0.000</td>
<td>0.998</td>
<td>1.000 ± 0.000</td>
</tr>
<tr>
<td></td>
<td>UGMMT</td>
<td>0.855 ± 0.001</td>
<td>0.365 ± 0.003</td>
<td>0.997 ± 0.001</td>
<td><strong>0.513 ± 0.009</strong></td>
<td>0.971</td>
<td>1.000 ± 0.000</td>
</tr>
</tbody>
</table>

Table 3.2: Evaluation results of our method and other baselines for various metrics for DRD2 property. In bold: best optimization Success result, statistically significant with huge effect size ($P-value < 0.05$ and Cohen’s $d > 2$).

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Property</th>
<th>Similarity</th>
<th>Novelty</th>
<th>Success</th>
<th>Validity</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paired</td>
<td>G2G</td>
<td>0.797 ± 0.005</td>
<td>0.330 ± 0.001</td>
<td>0.379 ± 0.006</td>
<td>0.075 ± 0.005</td>
<td>0.999</td>
<td>0.657 ± 0.009</td>
</tr>
<tr>
<td></td>
<td>CORE</td>
<td>0.758 ± 0.006</td>
<td>0.343 ± 0.002</td>
<td>0.400 ± 0.010</td>
<td>0.087 ± 0.005</td>
<td>1.000</td>
<td>0.648 ± 0.010</td>
</tr>
<tr>
<td>Unpaired</td>
<td>JTVAE</td>
<td>0.340 ± 0.000</td>
<td>0.239 ± 0.000</td>
<td>0.991 ± 0.000</td>
<td>0.033 ± 0.000</td>
<td>1.000</td>
<td>0.989 ± 0.000</td>
</tr>
<tr>
<td></td>
<td>Mol-CG</td>
<td>0.382 ± 0.000</td>
<td>0.190 ± 0.000</td>
<td>0.992 ± 0.000</td>
<td>0.013 ± 0.000</td>
<td>1.000</td>
<td>0.775 ± 0.000</td>
</tr>
<tr>
<td></td>
<td>UGMMT</td>
<td>0.826 ± 0.009</td>
<td>0.284 ± 0.001</td>
<td>0.799 ± 0.009</td>
<td><strong>0.192 ± 0.010</strong></td>
<td>1.000</td>
<td>0.914 ± 0.007</td>
</tr>
</tbody>
</table>
the unsupervised methods. Not only the main optimization success metric, which is much higher, but also the property and the similarity show impressive improvements. In addition, these methods unable to generate different molecules given different seeds hence the zero standard deviation, whereas our method present diverse outputs. Note that in nature, high DRD2 molecules are more rare compared to high QED molecules. Hence, the main challenges are to generate high DRD2 novel molecules and highly similar novel QED molecules. According to the results, UGMMT manages to successfully meet both challenges. In order to fully verify the results, it is not enough to calculate the P-value. While it indicates whether an effect (statistical significant difference) exists, it does not reveal the size of the effect [SF12]. Furthermore, with a sufficient large sample, a statistical test will almost always demonstrate a significant difference, unless there is no effect whatsoever. Hence, in addition to the P-value, we calculate Cohen’s d effect size [Coh88] and verify that our model’s Success is statistically significant ($P\text{-}value < 0.05$) and has huge effect size (Cohen’s $d > 2$) [Saw09] as compared to the other baselines.

3.3.2 Ablation Experiments

We perform various ablation experiments on UGMMT model for each dataset (DRD2 and QED). Results are presented in Table 3.3.

(1) No Pre-train experiment, we evaluate the contribution of pre-training the METNs before training the end-to-end model. Pre-training the MENTs enables the preparation of the latent embedding space of domain $A$ and domain $B$ separately. Thus,

Table 3.3: Ablation experiment evaluation results of our method for $P \in \{\text{QED, DRD2}\}$ properties. In bold: best optimization Success result, statistically significant with huge effect size ($P\text{-}value < 0.05$ and Cohen’s $d > 2$).

<table>
<thead>
<tr>
<th>P Method</th>
<th>Property</th>
<th>Similarity</th>
<th>Novelty</th>
<th>Success</th>
<th>Validity</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>UGMMT</td>
<td>0.855 $\pm$ 0.001</td>
<td>0.365 $\pm$ 0.003</td>
<td>0.997 $\pm$ 0.001</td>
<td><strong>0.513</strong> $\pm$ 0.009</td>
<td>0.971</td>
<td>1.000 $\pm$ 0.000</td>
</tr>
<tr>
<td>No Pre-train</td>
<td>0.843 $\pm$ 0.002</td>
<td>0.332 $\pm$ 0.002</td>
<td>0.999 $\pm$ 0.001</td>
<td>0.431 $\pm$ 0.009</td>
<td>0.879</td>
<td>1.000 $\pm$ 0.000</td>
</tr>
<tr>
<td>No EETN</td>
<td>0.824 $\pm$ 0.002</td>
<td>0.126 $\pm$ 0.001</td>
<td>1.000 $\pm$ 0.000</td>
<td>0.001 $\pm$ 0.001</td>
<td>0.925</td>
<td>0.936 $\pm$ 0.009</td>
</tr>
<tr>
<td>No fp</td>
<td>0.903 $\pm$ 0.001</td>
<td>0.127 $\pm$ 0.001</td>
<td>0.977 $\pm$ 0.005</td>
<td>0.002 $\pm$ 0.002</td>
<td>1.000</td>
<td>0.995 $\pm$ 0.002</td>
</tr>
<tr>
<td>Only fp</td>
<td>0.808 $\pm$ 0.002</td>
<td>0.391 $\pm$ 0.003</td>
<td>0.989 $\pm$ 0.002</td>
<td>0.411 $\pm$ 0.009</td>
<td>0.782</td>
<td>1.000 $\pm$ 0.000</td>
</tr>
<tr>
<td>Swap Cycle fp</td>
<td>0.872 $\pm$ 0.002</td>
<td>0.124 $\pm$ 0.001</td>
<td>1.000 $\pm$ 0.000</td>
<td>0.001 $\pm$ 0.001</td>
<td>0.962</td>
<td>0.996 $\pm$ 0.003</td>
</tr>
<tr>
<td>Add Adversarial</td>
<td>0.857 $\pm$ 0.002</td>
<td>0.299 $\pm$ 0.002</td>
<td>0.998 $\pm$ 0.001</td>
<td>0.344 $\pm$ 0.008</td>
<td>0.980</td>
<td>1.000 $\pm$ 0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P Method</th>
<th>Property</th>
<th>Similarity</th>
<th>Novelty</th>
<th>Success</th>
<th>Validity</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>UGMMT</td>
<td>0.826 $\pm$ 0.009</td>
<td>0.284 $\pm$ 0.001</td>
<td>0.799 $\pm$ 0.009</td>
<td><strong>0.192</strong> $\pm$ 0.010</td>
<td>1.000</td>
<td>0.914 $\pm$ 0.007</td>
</tr>
<tr>
<td>No Pre-train</td>
<td>0.505 $\pm$ 0.006</td>
<td>0.283 $\pm$ 0.001</td>
<td>0.988 $\pm$ 0.003</td>
<td>0.163 $\pm$ 0.005</td>
<td>0.582</td>
<td>0.992 $\pm$ 0.002</td>
</tr>
<tr>
<td>No EETN</td>
<td>0.836 $\pm$ 0.008</td>
<td>0.158 $\pm$ 0.002</td>
<td>0.812 $\pm$ 0.013</td>
<td>0.011 $\pm$ 0.003</td>
<td>1.000</td>
<td>0.508 $\pm$ 0.006</td>
</tr>
<tr>
<td>No fp</td>
<td>0.889 $\pm$ 0.006</td>
<td>0.159 $\pm$ 0.001</td>
<td>0.664 $\pm$ 0.013</td>
<td>0.010 $\pm$ 0.003</td>
<td>1.000</td>
<td>0.718 $\pm$ 0.012</td>
</tr>
<tr>
<td>Only fp</td>
<td>0.302 $\pm$ 0.006</td>
<td>0.279 $\pm$ 0.001</td>
<td>0.994 $\pm$ 0.004</td>
<td>0.085 $\pm$ 0.007</td>
<td>0.560</td>
<td>0.996 $\pm$ 0.002</td>
</tr>
<tr>
<td>Swap Cycle fp</td>
<td>0.809 $\pm$ 0.009</td>
<td>0.157 $\pm$ 0.001</td>
<td>0.883 $\pm$ 0.011</td>
<td>0.010 $\pm$ 0.003</td>
<td>0.997</td>
<td>0.408 $\pm$ 0.006</td>
</tr>
<tr>
<td>Add Adversarial</td>
<td>0.796 $\pm$ 0.009</td>
<td>0.253 $\pm$ 0.002</td>
<td>0.889 $\pm$ 0.006</td>
<td>0.156 $\pm$ 0.005</td>
<td>0.998</td>
<td>0.939 $\pm$ 0.008</td>
</tr>
</tbody>
</table>
we hypothesize that the encoders would produce better embeddings and the decoders in turn would produce more valid molecules with the enhanced property. In our ablation we validate this hypothesis and observe that the pre-training has higher effect in the DRD2 model. We conjecture this is due to the fact we pre-trained the DRD2 model for more epochs compared to the QED model. Additionally, we observe that although pre-training harms the novelty, it greatly improves the validity, the property and even has a positive effect on the similarity which leads to higher optimization success.

(2) No EETN experiment, we evaluate the contribution of the EETN component. The architecture without EETN is composed of separately pre-trained METNs using the reconstruction and the KL loss. We then train the models together optimizing the reconstruction loss only. The inference path is $En_A \rightarrow De_B$. During training, each domain’s decoder is specialized in generating molecules that hold the domain’s property, thus the property is relatively high. However, nothing preserves the input-output chemical connection so their similarity and hence optimization success are low. This experiment shows that having two METNs, one for each domain, is extremely important, enabling excellent initial opening point for property enhancement. This principal of METNs specialization partially explains UGMMT’s success comparing to other baselines.

(3) No fingerprints (fp) experiment, we evaluate the contribution of the fp component during the translation. We remove the usage of molecules’s fingerprints so the translators inside the EETN receive only the embedding without the fp. We observe the fp-dependant translation reaches high similarity since it encourages the translators to embed chemically similar molecules next to each other preserving input-output similarity. Without the fp, the similarity is extremely low and as a result so is the optimization success. We notice the high property compared to the baseline model which shows that the model without the fp is highly focused on property optimization disregarding similarity to the original molecule. The high property and validity imply that the double cycle constraints indeed train the translators to produce better embeddings for the following components, e.g. the decoders. We hypothesize that the double cycle constraints during training and the fp usage are the key for UGMMT’s success comparing to other baselines.

(4) Only fingerprints (fp) experiment, we wish to evaluate the necessity of the translators (EETN) and the embeddings that are jointly trained under the double cycle constraints. Given the results of the previous ablation test (“No fingerprints experiment”), one might claim that the only signal which is needed for molecule optimization is the source molecule fp. Hence, we designed a baseline which uses merely the molecule fp as the molecule embedding. We discard the training cycle that originates in $A$ together with its encoder, decoder and the translators (EETN). For domain $B$, we remove the encoder and leave the decoder. The training is performed by training $B$’s decoder to generate a molecule from its fp. Then, during inference, given a source molecule, its fp is calculated and passed to $B$’s decoder to generate the output molecule. This design
makes sense since $B$’s decoder is trained to produce molecules from domain $B$ with the same fp. Meaning, a similar molecule but with the enhanced property. Observing the results, indeed the similarity is higher compared to the baseline, however the property is lower so the overall optimization success is significantly lower. Especially, we notice the DRD2 property which is extremely low and unacceptable for molecule optimization process. Note that the validity is much lower compared to the baseline as well. As a result, we conclude that although molecule’s fp is a key signal in our model, it is not enough. The METNs (encoders and decoders) and EETN trained under the double cycle constraints are essential.

(5) **Swap Cycle-fp experiment**, we change the architecture s.t. $T_{AB}$ always gets $f_{Pa}$ and $T_{BA}$ always gets $f_{Pb}$ (changes lines 8 and 12 in Algorithm 3.2). Although it might seem intuitive to insert the fp according to the translation direction of the translator, preserving the fp of the original molecule helps optimizing similarity to the input molecule. Swapping the fp, as we see in the results, has a destructive effect on the similarity and the optimization success. The property is relatively high as we already deduced from the “No fingerprints experiment” (ablation experiment number 3) that it is mostly connected to the none fp parts of the model, which we kept intact in this experiment.

(6) **Add adversarial experiment**, we wish to study the impact of adversarial training on our model, similar to Mol-CG. We train our model in an adversarial manner by treating the translators as generators and adding a discriminator for each domain. Each discriminator, e.g. $B$’s discriminator $D_B$, gets an embedded molecule and decides whether it is an original embedded molecule, $⟨b⟩ \in ⟨B⟩$, or an embedding of a translated molecule, $⟨b'⟩ \in ⟨B'⟩$, classifies it as “real” or “fake”, respectively. Furthermore, KL loss terms are added to encourage dense and continuous embedding spaces for training stability and easier convergence. please refer to Appendix 3.4.2 for more details. This model is an improved version of Mol-CG model. Similar to Mol-CG, this baseline also uses cycle constraints and adversarial training, but additionally we train METN for each domain and use molecule fp, both of which we found to have high importance. Furthermore, we apply the cycle constrains directly on the molecules and not the embeddings which is more effective and we also train the METNs together with the EETN which is more target optimized. Nevertheless, taking into account all these improvement over Mol-CG, we see that this ablation model has still worse performance compared to the baseline UGMMT model we propose. Particularly, the similarity and the overall optimization success are lower. This explains the significant performance gap over Mol-CG in Section 3.3.1.

### 3.3.3 Optimized Drug Generation

Drug repositioning involves the exploration and improvement of existing drugs for new therapeutic purposes. We explore the use of our system for this purpose, where the
input to the system is an existing drug and the goal is to produce a similar yet enhanced drug. CDN [HR18], which is a sequence-based method for generating molecules, was the only algorithm attempting a similar task in the past. However, their task did not include the requirement of an enhanced property. We conduct a retrospective experiment using 1,897 FDA approved drugs. We ensured none of them was observed in the training or validation data. As we focus on existing drugs the enhancement of drug likeliness is less relevant and therefore we focus on DRD2 enhancement. We apply our DRD2 trained model (Sec. 3.3.1) on every drug in the approved drugs set as prototype, and generate 100 drugs for each drug using 100 different random seeds.

We observe that although the chance of generating a drug using exhaustive search without constrains, e.g. HTS, is negligible, our method generates approximately one approved drug for every \( \sim 3260 \) valid molecules generated. CDN for comparison reaches one drug per 14,596 generated molecules. Furthermore, the chance of discovering a drug-drug pair when going over all the drug-valid molecule pairs is 0.05971% which is relatively significant compared to CDN, 0.01185%. We observe that the average generated drugs’ DRD2 score is 0.879. For CDN, the average score is 0.124. We evaluate the average DRD2 score improvement on all the unique drug-drug pairs reaching an average property improvement of 0.627. The generated drugs show high similarity to their prototypes with an average of Tanimoto similarity of 0.3875.

We investigate the generated drugs and observe that many of the generated drugs are indeed used to treat psychotic disorders, like Schizophrenia. We remind the reader that D2 is the main receptor for most antipsychotic drugs [WCL\textsuperscript{+18}]. Table 3.4 presents a sample of the generated drugs.

We demonstrate our model’s capability of generating FDA-approved drugs. Today, additional laboratory experiments are carried on novel molecules. The system is currently deployed for use in a personalized medicine and nanotechnology research laboratory, which is currently focusing on RNA based therapeutics [SRH19]. We leverage RNA molecules to turn on and off pathogenic genes. Delivering these molecules to their

<table>
<thead>
<tr>
<th>Drug Name</th>
<th>DRD2</th>
<th>Generated Drug Name</th>
<th>DRD2</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pitavastatin</td>
<td>0.0077</td>
<td>Benperidol</td>
<td>1.0000</td>
<td>0.2651</td>
</tr>
<tr>
<td>Anagrelide</td>
<td>0.0009</td>
<td>Clozapine</td>
<td>0.8412</td>
<td>0.2096</td>
</tr>
<tr>
<td>Dapiprazole</td>
<td>0.1885</td>
<td>Aripiprazole</td>
<td>1.0000</td>
<td>0.2278</td>
</tr>
<tr>
<td>Alimemazine</td>
<td>0.3749</td>
<td>Perazine</td>
<td>0.6184</td>
<td>0.4000</td>
</tr>
<tr>
<td>Terazosin</td>
<td>0.0028</td>
<td>Iloperidone</td>
<td>1.0000</td>
<td>0.2556</td>
</tr>
<tr>
<td>Doxepin</td>
<td>0.1607</td>
<td>Amitriptyline</td>
<td>0.4743</td>
<td>0.6842</td>
</tr>
<tr>
<td>Promazine</td>
<td>0.5579</td>
<td>Triflupromazine</td>
<td>0.9908</td>
<td>0.7073</td>
</tr>
</tbody>
</table>
destination is challenging and requires specific carrier molecules. Since a certain family of molecules already has some of the properties required for this task, our method is used to optimize them and generate novel carrier molecules. The generated molecules are currently being evaluated by the chemists. The most promising molecules would be synthesized and pass to advanced chemical trials.
3.4 Appendices

3.4.1 Full Datasets Details

We provide experiments demonstrating our model’s capability of molecule and drug optimization. Hence we use two different datasets.

1. **Molecule Dataset**: The current SOTA method in molecule optimization is CORE [FXS20]. Therefore we use their datasets, which were adapted from [JYBJ19] and are publicly available on their GitHub\(^1\).

**Train set**: These datasets are paired, designed for supervised models, however UGMMT, CDN, JTVAE and Mol-CG are unsupervised. Hence we use the paired train set to construct unpaired train set. UGMMT method requires 2 input sets, set for domain \(A\) which contains low property molecules and set for domain \(B\) which contains high property molecules. We construct these by splitting the pairs, removing duplicates and then for each molecule we calculate its property score \{DRD2, QED\} and add it to the relevant set if the score exceeds a certain threshold. i.e., the molecule is added to set \(A\) if its property score is lower than domain’s \(A\) threshold and it is added to set \(B\) if its property score is higher than domain’s \(B\) threshold. In order to avoid unbalanced domains issues we randomly sample equal number of molecules from each set and thus obtain the final train set for \(A\) and \(B\). Since JTVAE and CDN require only one training set, we merge \(A\)’s and \(B\)’s train sets to one set and use it to train them. We set empirically the following domain thresholds: DRD2 – \(A\) 0.02 (experimented with 0.01-0.2) ; \(B\) 0.85 (experimented with 0.75-0.9) / QED – \(A\) 0.78 (experimented with 0.75-0.8) ; \(B\) 0.91 (experimented with 0.88-0.93), other thresholds yield similar results.

All dataset files are located inside `dataset/DRD2` and `dataset/QED` folders on our GitHub. Details:

UGMNNMT and Mol-CG– The train sets contain 2,097 molecules for DRD2 and 8,968 molecules for QED (in each set among \{\(A, B\}\)).

Files names: `A_train.txt` and `B_train.txt`.

CDN and JTVAE– The train sets contain 4,194 molecules for DRD2 and 17,936 molecules for QED.

Files names: `DRD2_mergedAB_specific_train.txt` and `QED_mergedAB_specific_train.txt`.

G2G and CORE– Taken from their GitHub, contains 34,404 molecule pairs for DRD2 and 88,306 molecule pairs for QED.

Files names: `DRD2_DATASET.txt` and `QED_DATASET.txt`.

\(^1\)https://github.com/wengong-jin/iclr19-graph2graph
**Validation set:** For UGMMT, after sampling the final train set from set $A$, we randomly sample the validation set from set $A$ remaining molecules. For the other models we use the original validation sets from CORE [FXS20].

UGMMT– The validation sets contain 800 molecules for DRD2 and 800 molecules for QED.
File name: `A_validation.txt`.

Other models– The validation sets contain 500 molecules for DRD2 and 360 molecules for QED.
File name: `g2g_validation.txt`.

**Test set:** The test set is taken as is from CORE [FXS20], thus all methods are evaluated on exactly the same data. The test sets contain 1,000 molecules for DRD2 and 800 molecules for QED.
File name: `A_test.txt`.

2. **Drug Dataset:** We use DrugBank dataset [WKG+06], which contains a list of FDA approved drugs, to conduct retrospective experiments and confirm our model’s capability of potential drug optimization. We extract a set of 1,897 drugs and ensure none of them appeared during training or validation. The drug dataset is located in the `FDA_approved_canon_clean.csv` file inside the `dataset/FDA_approved_drugs_drugbank` folder on our GitHub.

### 3.4.2 Add Adversarial Ablation Experiment Details

We wish to study the impact of adversarial training on our model, similar to Mol-CG. In addition to our cycle constraints with fp attention we add adversarial training. In this experiment, similarly to our baseline UGMMT model, we pre-train the METNs before training the whole model.

We treat the translators as embedding generators and add two embedding discriminators, $D_A$ and $D_B$, one for each domain. Each discriminator, e.g. $D_B$, gets an embedded molecule and decides whether it is an original embedded molecule, $\langle b \rangle \in \langle B \rangle$, or an embedding of a translated molecule, $\langle b' \rangle \in \langle B' \rangle$, classifies it as “real” or “fake”, respectively. Hence, generator’s goal is to create embeddings indistinguishable from the real data distribution while discriminator’s goal is to distinguish candidates artificially created from the real data distribution. This process is expressed in the additional adversarial loss terms for translators training:

$$MSE(D_B(T_{AB}(E_{NA}(a), fp_a)), 1_v) + MSE(D_A(T_{BA}(E_{NB}(b), fp_b)), 1_v)$$

Where MSE is the mean square error loss, $1_v$ is a vector of ones. We also add KL loss
terms to encourage dense and continuous embedding spaces for training stability:

$$0.2 \cdot (KL(En_A(a)||N(0, I)) + KL(En_B(b)||N(0, I)))$$

Where 0.2 is a small scalar. Its purpose is, on the one hand, to make these terms significant enough for improving training stability, and on the other hand, to be negligible enough compared to the other terms to not interrupt the adversarial training.

After training the translators, in the same iteration, discriminator A is trained by minimizing:

$$0.5 \cdot MSE(D_A(En_A(a)), 1_v) + 0.5 \cdot MSE(D_A(T_{BA}(En_B(b), fp_b)), 0_v)$$

Similarly, discriminator B is trained by minimizing:

$$0.5 \cdot MSE(D_B(En_B(b)), 1_v) + 0.5 \cdot MSE(D_B(T_{AB}(En_A(a), fp_a)), 0_v)$$

Where MSE is the mean square error loss, $1_v$ is a vector of ones and $0_v$ is a vector of zeros. The first term in each objective function encourages the discriminator to identify “real” molecules and the second – “fake” molecules.

Each Discriminator contains 3 convolution blocks, each one consists of convolution (filter sizes 3, 3, 4), instance normalization and LeakyReLU (0.2) layers. Then convolution (filter size 4) and average pooling. First two convolutions are stride-2.

One might consider this model as an improved version of Mol-CG model as in addition to adversarial training introduced by the latter, we also train METN for each domain and use molecule fp. Furthermore, we apply the cycle constraints directly on the molecules and not the embeddings which is more effective and we also train the METNs together with the EETN which is more target optimized. However, taking into account all these improvements over Mol-CG, we see that this ablation model has still worse performance compared to the baseline UGMMT model we propose. This explains the significant performance gap of UGMMT over Mol-CG in Section 3.3.1.
Chapter 4

Multi-Property Molecular Optimization using an Integrated Poly-Cycle Architecture

The first chapter of this thesis focused on the development of the single property optimization system. In this chapter, we discuss the extension of our methods to multi-property molecular optimization. Although prior work mostly focused on optimizing a single property, in real applications, there is often a need to generate molecules that satisfy multiple constraints simultaneously. In this chapter, we focus on multi-property molecular optimization and present the Integrated Poly-Cycle Architecture (IPCA) – an end-to-end generative model trained to generate novel discrete molecules with multiple desired properties.

4.1 Methods

We denote a domain of molecules, e.g. high drug likeness, by a capital Latin letter, e.g. $X$, their distribution by $p(X)$, and a molecule taken from this domain by a small Latin letter, $x$. That is, $x \in X$ denotes $x$ is a sample of molecules taken from domain $X$ and consequently $x \sim p(X)$. Similarly, we denote the embedding vector of a molecule $x$ by $\langle x \rangle$. It belongs to the domain of all the embedded molecules of $X$, which we denote by $\langle X \rangle$ with a distribution $p(\langle X \rangle)$. We also denote a molecule property by $prop_i$. For example, if $prop_1$ is QED, $prop_1(x)$ is the QED value of molecule $x$. We wish to transform a molecule $m$ with degraded properties to a molecule $m'$ with optimized properties. Suppose $A$ is a domain of molecules with degraded $N$ properties, $prop_i$, $i \in [1 \ldots N]$. We wish to transform $m \in A$ into $m' \in B_i$ in which $prop_i(m') > prop_i(m)$ $\forall i \in [1 \ldots N]$. In addition, we wish that $m'$ remains similar to $m$. Figure 4.1 presents the poly-cyclic architecture for $N = 2$. Our model contains an optimization path for each property $prop_i$. The paths are then joined by a shared embedding space from which a joined optimization can be induced (solid red line). Thus, in the case of
double property optimization, one optimization path will transform molecules from \( A \) to \( B_1 \), the second optimization path will transform molecules from \( A \) to \( B_2 \) and the transformation to domain \( B_{1,2} \), in which both properties are optimal, will be achieved through a shared embedding space.

### 4.1.1 Molecular Optimization Path

In this section, we describe an end-to-end optimization path for a single property. For simplicity, we use the \( \text{prop}_1 \) notation for the property (the upper half of Figure 4.1). The dotted lines in the figure represent training paths and the solid lines represent inference paths.

During inference, following the translation along the path from \( A \) to \( B_1 \) (solid red line), an input molecule \( a \in A \), given in a discrete textual SMILES representation, is encoded by an encoder \( E_{nA} \) to its continuous representation \( \langle a \rangle \) (Section 4.1.4), then using a translator \( T_{AB1,2} \) it is mapped to a continuous space of molecules with the enhanced property \( \text{prop}_1 \). Lastly, it is decoded to its SMILES representation using the decoder \( D_{eB1,2} \). \( En \) and \( De \) are respectively an encoder and decoder GRU networks followed by fully connected layers, and \( T \) are convolutional bottleneck networks with resnet layers (see Section 4.2.5 for details).

The training for this path is described in the dotted lines. The upper red dotted line forms a counter-clockwise circle from \( A \) through \( E_{nA}, T_{AB1,2}, T_{B1A} \) and \( D_{eA} \)

![Figure 4.1: IPCA end-to-end architecture. The training paths are marked with dotted arrows while the inference paths are marked with thick solid arrows.](image-url)
back to $A$. $En_A$ encodes $a \in A$ to a latent embedded space $\langle A \rangle$, $T_{AB_1,2}$ and $T_{B_1,A}$ transform the encoding sequentially to $\langle B_{1,2}' \rangle$ and $\langle A' \rangle$, and $De_A$ decodes it back to $A'$. The distribution of $A'$ should be indistinguishable from the distribution of $A$. We encourage this by penalizing $a$ and $a'$ difference in the loss function. The upper purple dotted line describes an identical mirrored cycle starting (and ending) in domain $B_1$.

Similarly to [BR21], to encourage similarity between the source and the optimized molecules, we concatenate the molecule’s extended connectivity fingerprints ($fp_a$ and $fp_{b_1}$) [RH10] to its latent representation. This forces the embedding spaces to become fingerprint dependant, representing molecules with similar molecular structures with similar embeddings.

### 4.1.2 Multi-Property Optimization

For multi-property optimization, we combine several optimization paths. The paths are linked by a shared embedding space (in our example $\langle B_{1,2}' \rangle$) which is now constrained by both $B_1$ and $B_2$. Applying the decoder $De_{B_{1,2}}$ on this latent space will produce

![Figure 4.2: IPCA end-to-end architecture with additional property prop3. Encoder $En_{B_1}$ and translator $T_{B_1,A}$ are added to support the additional property. The same approach may be used to support more properties.](image-url)
molecules that are optimized both for prop$_1$ and prop$_2$. Adding another property optimization (prop$_3$) will entail adding an encoder ($En_{B_3}$), translator ($T_{B_3,A}$) and linking them to the main translator (would be $T_{AB_{1,2,3}}$) that generates the shared embedding space. See Figure 4.2. Some details are not shown for readability (e.g. $fp_{b_2}$), however it contains all the components that were presented in Figure 4.1. That will contribute two additional cycles to the training phase: $En_A \rightarrow T_{AB_{1,2,3}} \rightarrow T_{B_3,A} \rightarrow De_A$ (the additional red dotted path) and $En_{B_3} \rightarrow T_{B_3,A} \rightarrow T_{AB_{1,2,3}} \rightarrow De_{B_{1,2,3}}$ (the additional purple dotted path). In the same manner, additional properties for optimization may be simply added.

**Training:** Algorithm 4.1 describes the end-to-end training of the model. We first pre-train all encoders and decoders to generate valid continuous molecule representation for $A$, $B_1$ and $B_2$ molecule domains. This allows quality inputs to the translators, which, in turn, produce results with higher validity. We then simultaneously train four cycles: $En_A \rightarrow T_{AB_{1,2}} \rightarrow T_{B_1,A} \rightarrow De_A$ (lines 8-11), $En_A \rightarrow T_{AB_{1,2}} \rightarrow T_{B_2,A} \rightarrow De_A$ (lines 12-14), $En_{B_1} \rightarrow T_{B_1,A} \rightarrow T_{AB_{1,2}} \rightarrow De_{B_{1,2}}$ (lines 15-18) and $En_{B_2} \rightarrow T_{B_2,A} \rightarrow T_{AB_{1,2}} \rightarrow De_{B_{1,2}}$ (lines 19-22). The translators together with the unique training technique encourage similar distribution in domains $\langle A \rangle$ and $\langle A' \rangle$, $\langle B_1 \rangle$ and $\langle B'_{1,2} \rangle$, $\langle B_2 \rangle$ and $\langle B'_{1,2} \rangle$. $T_i$ is a translation neural network from one latent embedding domain to another. For example: $T_{AB_{1,2}}$ maps $\langle a \rangle \in \langle A \rangle$ to $\langle b'_{1,2} \rangle \in \langle B'_{1,2} \rangle$. During training, $T_{AB_{1,2}}$ is trained to transform $\langle a \rangle \in \langle A \rangle$ to $\langle b'_{1,2} \rangle \in \langle B'_{1,2} \rangle$ through two training cycles (upper left and lower left in Figure 4.1). During inference, $\langle b'_{1,2} \rangle \in \langle B'_{1,2} \rangle$ is then decoded into $b'_{1,2} \in B'_{1,2}$ which is a SMILES representation of the optimized molecule. Translation between embedding domains is further conditioned on the molecules fingerprints $fp_a$, $fp_{b_1}$ and $fp_{b_2}$ (lines 8, 9, 12, 15, 16, 19, 20) to maintain similarity between the source and the optimized molecules. Since SMILES notation is a discrete representation, we use multi-layer GRU cells in the decoder to predict the next character in the SMILES representation given the current state and the current input character. Hence, the correct loss for this classification task is the Cross-Entropy (CE), where $CE(a',a)$ means the mean CE loss between the original $a$ molecule (SMILES characters) and the reconstructed $a'$ molecule.

The overall loss is composed of the four cycles’ cross-entropy losses (CE). Note that each cycle may be easily formalized as an objective function to minimize, e.g., for the upper left cycle that starts in domain $A$:

$$CE(De_A(T_{B_1,A}(En_A(a), fp_a), fp_a), a)$$  \hfill (4.1)

To ease the readability and reproducibility of the thesis, we present an algorithmic formulation of the above (Algorithm 4.1).

We suggest a novel loss function that automatically weighs the cross entropy losses.
of the cycles: $En_{B_1} \to T_{B_1A} \to T_{AB_{1.2}} \to De_{B_{1.2}}$ and $En_{B_2} \to T_{B_2A} \to T_{AB_{1.2}} \to De_{B_{1.2}}$. The loss component coefficients are adapted during the training as described in Section 4.1.3.

**Inference:** During inference a lead molecule $m \in A$ is encoded by $En_A$ to the embedding domain $\langle A \rangle$, mapped by $T_{AB_{1.2}}$ to the embedding space $\langle B_{1.2}' \rangle$ and decoded by $De_{B_{1.2}}$. Since $T_{AB_{1.2}}$ is shared by all cycles during the training, the embedding space

Algorithm 4.1 IPCA Training algorithm. Corresponds to the four dotted paths in Figure 4.1, the red paths along $En_A \to T_{AB_{1.2}} \to T_{B_1A} \to De_A$ and $En_A \to T_{AB_{1.2}} \to T_{B_2A} \to De_A$ and the purple paths along $En_{B_1} \to T_{B_1A} \to T_{AB_{1.2}} \to De_{B_{1.2}}$ and $En_{B_2} \to T_{B_2A} \to T_{AB_{1.2}} \to De_{B_{1.2}}$.

**Input:** $A_t$, $B_{1,t}$, $B_{2,t}$ molecule training sets. $A_v$ molecule validation set.

1. Train domain’s $A$ METN on $A_t$ for $Ep_A$ epochs (Algorithm 3.1)
2. Train domain’s $B_1$ METN on $B_{1,t}$ for $Ep_{B_1}$ epochs (Algorithm 3.1)
3. Train domain’s $B_2$ METN on $B_{2,t}$ for $Ep_{B_2}$ epochs (Algorithm 3.1)
4. for epoch $= 1, 2, \ldots, E_{max}$ do
5. Sample mini-batches $a \in A_t$, $b_1 \in B_{1,t}$, $b_2 \in B_{2,t}$
6. $\langle a \rangle = En_A(a)$, $\langle b_1 \rangle = En_{B_1}(b)$, $\langle b_2 \rangle = En_{B_2}(b)$
7. Calculate $fp_a$, $fp_{b_1}$, $fp_{b_2}$
8. $\langle b_{1.2} \rangle = T_{AB_{1.2}}(\langle a \rangle, f_{p_a})$
9. $\langle a' \rangle = T_{B_1A}(\langle b_{1.2} \rangle, fp_a)$
10. $a' = De_A(\langle a' \rangle)$
11. $L_{AB_1} = CE(a', a)$
12. $\langle a' \rangle = T_{B_2A}(\langle b_{1.2} \rangle, fp_a)$
13. $a' = De_A(\langle a' \rangle)$
14. $L_{AB_2} = CE(a', a)$
15. $\langle a' \rangle = T_{B_1A}(\langle b_1 \rangle, fp_{b_1})$
16. $\langle b_{1.2} \rangle = T_{AB_{1.2}}(\langle a' \rangle, fp_{b_1})$
17. $b_{1.2}' = De_{B_{1.2}}(\langle b_{1.2} \rangle)$
18. $L_{B_1} = CE(b_{1.2}', b_1)$
19. $\langle a' \rangle = T_{B_2A}(\langle b_2 \rangle, fp_{b_2})$
20. $\langle b_{1.2}' \rangle = T_{AB_{1.2}}(\langle a' \rangle, fp_{b_2})$
21. $b_{1.2}' = De_{B_{1.2}}(\langle b_{1.2}' \rangle)$
22. $L_{B_2} = CE(b_{1.2}', b_2)$
23. $L = L_{AB_1} + L_{AB_2} + \lambda_{B_1} \cdot L_{B_1} + \lambda_{B_2} \cdot L_{B_2}$
24. Minimize $L$ using Adam optimizer
25. Evaluate the model every $V_f$ epochs on $A_v$
26. Update $\lambda_{B_1}$ and $\lambda_{B_2}$ according to the validation performance
27. if evaluation criterion improved then
28. save model
29. end if
30. if no improvement for $P$ evaluations then
31. stop training
32. end if
33. end for
\( \langle B'_1, B'_2 \rangle \) represents molecules with optimized values for both \( \text{prop}_1 \) and \( \text{prop}_2 \) and so is the consequent SMILES representation created by the decoder \( De_{B_1, B_2} \).

### 4.1.3 Adaptive Loss

Since our model’s goal is to generate molecules which have multiple enhanced properties, there might be properties that are more challenging to optimize than others. In this case, the optimization may be leaning towards the “easy to optimize” ones. Although setting constant coefficients in the loss might help, we notice that adjusting these coefficients during training yields smoother, more stable training process and consequently improved performance (see Section 4.3.3). We achieve this by dynamically adjusting the loss component coefficients during training (\( \lambda_{B_1} \) and \( \lambda_{B_2} \) in Algorithm 4.1).

We initialize \( \lambda_{B_1} = 1 \) and \( \lambda_{B_2} = 1 \) and following every validation step we update

\[
\lambda_{B_1} = \lambda_1 \times \frac{\text{wanted}_1}{\text{validation}_1} \tag{4.2}
\]

and

\[
\lambda_{B_2} = \lambda_2 \times \frac{\text{wanted}_2}{\text{validation}_2} \tag{4.3}
\]

where \( \lambda_1 \) and \( \lambda_2 \) are initial constant values, \( \text{wanted}_i \) is the desired \( \text{prop}_i \) property value and \( \text{validation}_i \) is the average \( \text{prop}_i \) property value of the molecules which are generated in the validation process.

### 4.1.4 METN for Molecule-Embedding Translation

One of the main challenges of models leveraging SMILES representations in deep generative settings is their discrete representations. We describe an architectural component that allows their transformation to a continuous representation, on which optimization can be performed. We follow the design of UGMMT \cite{BR21} and use the METN component (Section 3.1.1) which enables a sequence-to-sequence translation with a bidirectional GRU-based encoder and a multilayered GRU-based decoder \cite{CvMG14, KW14}. Following \cite{BR21}, in order to boost the validity of the molecules that are generated by the decoder, we leverage the teacher-forcing method \cite{WZ89} during training: we provide each GRU cell the correct input character even if the previous GRU cell predicted it incorrectly. We draw the reader attention that the encoder’s architecture can be replaced with stronger architectures (e.g. transformers). In this work, we present simple GRU architecture to emphasize that the key for IPCA’s success lies in the cycle constraints with molecular fingerprints and the adaptive loss components.
4.2 Experimental Settings

4.2.1 Datasets

A common multi-property optimization task is to optimize drug likeness (QED) [BPB+12] and Dopamine Receptor D2 (DRD2). We used the datasets by [JYBJ19] as reported by [JBJ19]. QED and DRD2 values for optimized molecules are calculated using the RDKit package [Lan16] and the trained model for DRD2 activity from [OBEC17]. The original training set contains 122,710 molecule pairs and the test set contains 780 molecules. Different application may require different thresholds of DRD2 and QED. In this work, we experiment with several such thresholds. We chose to test the above two attributes following the latest related work (Section 2, e.g. [JBJ19]). Note that QED, i.e. drug likeness, is a combination of molecular properties such as solubility, ligand efficiency, molecular weight, etc.. Hence, by optimizing QED we effectively optimize multiple molecular properties.

4.2.2 Metrics

We evaluate our task using several metrics:

1. Validity – the percentage of source molecules that have at least one valid optimized molecule.

2. Average Property Value – the average QED, DRD2 and similarity values of the optimized molecules.

3. Novelty – the percentage of the valid optimized molecules which were not seen in the training set.

4. Diversity – the percentage of unique generated molecules, i.e., the number of unique molecules divided by the number of valid molecules.

5. Success rate (SR) – the percentage of optimized molecule which have similarity larger than 0.3, and QED and DRD2 values above their respective thresholds.

4.2.3 Empirical Methodology

We create 100 optimization candidates for each test molecule, resulting in 78,000 optimized molecules for each model. We randomly select one candidate for each source molecule and compute the metrics on the 780 optimised molecules selected (if there exists a source molecule with no valid optimized molecule for one of the models, this source molecule is discarded from the comparison for all models). This process is repeated 10 times, and the average metrics are reported. For the 780 optimized molecules we calculate the average QED and DRD2 scores, the average similarity to the source molecule and the overall success rate.
4.2.4 Baselines

We compare IPCA both to the SOTA supervised and unpaired methods for multi-property lead optimization:

1. **SOTA Supervised Baseline**: HG2G [JBJ19] is a supervised method for multi-property lead optimization. It uses a paired training set of a lead molecule and an enhanced molecule.

2. **Leading Unpaired Baseline**: As IPCA method is unpaired, i.e., our training set does not contain pairs of source and destination molecules as opposed to HG2G for example, we also include the leading unpaired method JTVAE [JBJ18]. We adapt it to the multi-property lead optimization task. For single-property molecule optimization JTVAE is jointly trained with a property predictor $C$ (commonly a feed-forward network) to predict the property from the latent embedding [JBJ18]. For multi-property optimization we trained a predictor for each property. Gradient ascent is then applied in the latent space to improve the sum of the predicted scores of the classifiers. The molecule with the highest score that satisfies the similarity constraint is outputted. Note that this model produces one output molecule per source molecule. As a result, calculating the metrics on its outputs leads to zero standard deviation.

4.2.5 Implementation Details for IPCA

For the training of IPCA, molecule domain $A$ is created by selecting molecules with low QED and DRD2 values from the training set. Molecule domain $B_1$ is created by selecting molecules with high QED values ($QED \geq 0.92$) and domain $B_2$ is created by selecting molecules with high DRD2 values ($DRD2 \geq 0.65$).

**Hardware**: Nvidia GeForce RTX 2080 Ti 11GB GPU, 2 Intel Xeon Gold 6230 2.10GHZ CPUs, 64GB RAM.

**Software**: Ubuntu 18.04.5, Pytorch 1.4.0 [PGM+19], Python 3.6.

**Reproducibility**: We set seed 50, other seeds yield similar results. Hyper-parameters are set following evaluation on a validation set.

**METN**: Embedding dimension 625 (experimented with 500, 625, 750), $\lambda_0=10$ (experimented with 1, 10, 20), max molecule length during inference 90 (experimented with 80-120). Encoder – bidirectional GRU, hidden dimension 625, followed by 2 FC with 625 neurons, for $\mu$ and $\sigma$. Decoder – FC layer with 1250 neurons, followed by 3-layered GRU (experimented with 1-5), dropout 0.5 between layers, hidden dimension 1250 and a FC with vocabulary length neurons. Training– Adam optimizer [KB15]. Initial learning rate $3 \cdot 10^{-3}$ (experimented with $1 \cdot 10^{-3}$-$5 \cdot 10^{-3}$), final $3 \cdot 10^{-4}$ using cosine annealing scheduler with restart [LH19] after 10 epochs. Mini-batch size 32. Pre-trained for $Ep_A = 20$, $Ep_{B_1} = 5$ (QED), $Ep_{B_2} = 25$ (DRD2) epochs (experimented with 5, 10, 15, 20, 25, 30).
Translators: Conv block (filter size 7), stride-2 conv downsampling block (filter size 3), 4 residual blocks (experimented with 4-8), stride-2 transposed conv upsampling block (filter size 3), conv (filter size 7) followed by Tanh and 2 FC with 800 and 625 neurons separated by BN, LeakyReLU and Dropout (0.2). These types of structures presented outstanding results in image translation and style transfer tasks [JAFF16, ZPIE17].

Whole model training: Adam optimizer, initial learning rate $1.5 \cdot 10^{-4}$ (experimented with $1 \cdot 10^{-4}$-$5 \cdot 10^{-4}$) and a linear decay towards 0 from epoch 120 (experimented with 80-140). Mini-batch size 32. Max epochs $E_{max}=150$ (experimented with 120-170). Data is shuffled during training. Evaluation frequency $V_f=3$, early stopping with $P=15$ evaluations. $\lambda_1=30$, $\lambda_2=90$ (experimented with 1,30,60,90,120). $(wanted_1, wanted_2) \in \{(0.7, 0.3), (0.75, 0.35), ..., (0.9, 0.5)\}$ according to the specific experiment property threshold settings. Training time: 5 hours. Inference time: $\sim 10$ seconds on the whole test set. Our code and data are publicly available.
4.3 Experiments and Results

In this section, we present the comparison of our model with the baselines, show qualitative examples of molecule optimization and perform numerous ablation tests.

4.3.1 Main Result

We study the behavior of the algorithm on several settings with different QED and DRD2 success thresholds. Given a lead molecule, an optimization is successful if the generated molecule’s QED and DRD2 are above those success thresholds and it is similar enough to the lead molecule. Each pair of thresholds constitutes a separate dataset. For each pair of thresholds we perform an experiment studying the impact of the number of examples that satisfy all properties on the performance of the algorithm. Figure 4.3 presents the results for QED threshold of 0.7 and DRD2 threshold of 0.3. Similar results were obtained with other thresholds (we experimented with (0.7,0.3) – (0.9,0.5) with lags of 0.05). We observe that the performance of IPCA is stable, slightly decreases as the amount of training examples satisfying all conditions decreases, whereas HG2G performance is highly impacted. We hypothesize that while IPCA focuses on optimization of each property and its contribution to the shared optimized embedding space, HG2G tries to optimize all properties at once. As a result, IPCA’s performance is stable as long as molecules with even one high property exist, whereas HG2G quickly deteriorates as the number of molecules satisfying all properties decreases. Similar results of HG2G were reported by [JBJ19]. Similarly to IPCA, JTVAE presents stable results, but with significantly degraded SR. It suggests that “simple” gradient ascent optimizing both properties is insufficient for the task of multi-property optimization.

![Figure 4.3](image-url)

Figure 4.3: Success rate as a function of the number of training examples confirming to all properties.
We now dwell deeper into the common scenario in nature, where molecules satisfying all properties to be optimized are scarce. We remove training examples with both QED and DRD2 above the success thresholds. As this naturally affects the number of training examples, we repeat this process for several property thresholds. We start from the thresholds set in [JBJ19], according to which \( QED > 0.9 \) and \( DRD2 > 0.5 \). We then lower QED and DRD2 threshold values by 0.05 for each experiment. Table 4.1 summarizes the amount of examples removed from each training set. The results of IPCA and the other baselines over these datasets are reported in Table 4.2.

Note that JTVAE generates a single molecule, therefore the zero standard deviation. Observing the results summarized in Table 4.2, we notice that as the success thresholds increase, creating a molecule with such high DRD2 and QED values becomes an increasingly more challenging optimization task for all algorithms, even though the number of training examples increases. However, the success rate of the IPCA model is consistently superior to that of JTVAE and HG2G (regardless of the success thresholds and number of training examples). The difference between success rates grows as the thresholds are lowered, reaching a factor of 2 and over at the lowest thresholds. We draw the reader’s attention that although the SR might seem generally low, it is quite dramatic for the drug development industry. Even one successful molecule can progress us towards a drug. Note that we calculated the standard deviation and Cohen’s \( d \) effect size and ensured it is over 1, i.e. our model’s SR is substantially different with a large effect from the SOTA.

The IPCA model achieves higher average DRD2 while the HG2G model achieves higher QED. JTVAE struggles to optimize DRD2. This might stem from the fact that naturally there are more molecules with high QED, but high DRD2 is relatively rare making its optimization more challenging compared to QED.

Note that the average QED, DRD2 and similarity values alone do not provide the complete performance analysis since the multi-optimization task requires all properties to exist at once in the optimized molecule. The success rate is therefore a more appropriate metric by which to evaluate the overall performance.

As expected, the validity of the HG2G model is superior to that of the IPCA model. This is less of a concern as we can produce a large number of candidates for

<table>
<thead>
<tr>
<th>Thresholds (QED, DRD2)</th>
<th>Training Pairs Removed (Above Thresholds)</th>
<th>Training Pairs In Set (Below Thresholds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.90, 0.50)</td>
<td>1.6%</td>
<td>120,745</td>
</tr>
<tr>
<td>(0.85, 0.45)</td>
<td>4.5%</td>
<td>117,252</td>
</tr>
<tr>
<td>(0.80, 0.40)</td>
<td>9.4%</td>
<td>111,131</td>
</tr>
<tr>
<td>(0.75, 0.35)</td>
<td>12.96%</td>
<td>106,811</td>
</tr>
<tr>
<td>(0.70, 0.30)</td>
<td>17.07%</td>
<td>101,758</td>
</tr>
</tbody>
</table>

Table 4.1: Training set sizes.
<table>
<thead>
<tr>
<th>Type Method</th>
<th>DRD2</th>
<th>QED</th>
<th>Similarity</th>
<th>SR</th>
<th>Validity</th>
<th>Novelty</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>HG2G 90.50</td>
<td>0.115 ± 0.006</td>
<td>0.867 ± 0.003</td>
<td>0.232 ± 0.003</td>
<td><strong>0.003±0.002</strong></td>
<td>1.000</td>
<td>0.875 ± 0.008</td>
<td>0.960 ± 0.006</td>
</tr>
<tr>
<td>HG2G 85.45</td>
<td>0.178 ± 0.009</td>
<td>0.848 ± 0.004</td>
<td>0.221 ± 0.002</td>
<td>0.005 ± 0.002</td>
<td>1.000</td>
<td>0.888 ± 0.011</td>
<td>0.965 ± 0.006</td>
</tr>
<tr>
<td>HG2G 80.40</td>
<td>0.143 ± 0.009</td>
<td>0.853 ± 0.005</td>
<td>0.233 ± 0.004</td>
<td>0.006 ± 0.005</td>
<td>1.000</td>
<td>0.866 ± 0.015</td>
<td>0.960 ± 0.006</td>
</tr>
<tr>
<td>HG2G 75.35</td>
<td>0.206 ± 0.013</td>
<td>0.826 ± 0.004</td>
<td>0.230 ± 0.004</td>
<td>0.008 ± 0.003</td>
<td>1.000</td>
<td>0.871 ± 0.012</td>
<td>0.948 ± 0.008</td>
</tr>
<tr>
<td>HG2G 70.30</td>
<td>0.102 ± 0.009</td>
<td>0.850 ± 0.005</td>
<td>0.228 ± 0.003</td>
<td>0.007 ± 0.002</td>
<td>1.000</td>
<td>0.906 ± 0.012</td>
<td>0.962 ± 0.004</td>
</tr>
</tbody>
</table>

Table 4.2: Evaluation over multiple datasets of DRD2 and QED properties. In bold: best result for SR (cohen’s d effect size over 1).

Each molecule, screening out non-valid molecules automatically using RDKit.

Overall, the results show that our method outperforms both supervised paired methods, e.g. HG2G, the leading method for multi-property optimization, and unsupervised unpaired methods, e.g. JTVAE.

### 4.3.2 Qualitative Examples

Figures 4.4 – 4.6 present several examples of molecules generated during the testing of IPCA 70.30. Note that IPCA 70.30 model has never seen a molecule with QED > 0.7 and DRD2 > 0.3, nevertheless, it is able to generate novel molecules above these thresholds and similarity over 0.3.

The generated molecules depicted in Figure 4.4 preserve high fingerprint similarity (over 0.6) to the lead molecules. IPCA 70.30 manages to optimize the lead molecule not only when its QED property is already above the threshold (Figure 4.4B), but also when it is below the threshold (Figure 4.4A). In Figure 4.4A, the QED score improves from 0.6493 to 0.7209 and the DRD2 score improves from 0.0062 to 0.9901, while keeping a high similarity of 0.6875 during this optimization. In Figure 4.4B, the model keeps the already successful QED property (slight improvement from 0.7856 to 0.7862) and optimizes the DRD2 score from 0.0284 to 0.4733, while reaching 0.6363 similarity between the generated and the lead molecules.

Figure 4.5 shows several examples where the lead molecules have relatively low properties (QED < 0.4 and DRD2 < 0.05). Nevertheless, IPCA 70.30 performs well and optimizes both properties while keeping similarity to the lead molecules. In Figure 4.5A, the model improves the QED score from a low value of 0.3940 to 0.9079 and the DRD2 score from 0.0425 to 0.7138 with a similarity of 0.3602 between the lead and the
Figure 4.4: Sample of molecules generated using IPCA $^{70,30}$. The generated molecules preserve high similarity to the lead molecules.

![Figure 4.4](image)

Figure 4.5: Sample of molecules generated using IPCA $^{70,30}$. The lead molecules have low QED and DRD2 properties.

![Figure 4.5](image)

In Figure 4.5B, the model improves the QED score from even a lower value of 0.3097 to 0.8209 and the DRD2 score from 0.0327 to 0.5324 preserving a similarity of 0.3559 between the lead and the generated molecules.

Figure 4.6 presents several examples where the generated molecules have relatively high properties. In Figure 4.6A, our model got a lead molecule with a QED score of 0.7493 and a DRD2 score of 0.0015 and was able to generate a molecule with a QED score of 0.9139 and a DRD2 score of 0.8973. The similarity is 0.4677. In Figure 4.6B, the QED score improved from 0.5041 to 0.8376, which is a meaningful. However the truly impressive improvement is the DRD2 score improvement from 0.0005 in the lead molecule to 0.9999 in the generated molecule.

In all the examples, investigating the SMILES strings and their corresponding chem-
Figure 4.6: Sample of molecules generated using IPCA $_{70,30}$. The generated molecules have high QED and DRD2 properties.

The chemical structure sketch reveals high resemblance between the lead and the generated molecules. It shows once more that IPCA not only improves molecule’s properties, but it does it while preserving its chemical features.

### 4.3.3 Ablation Tests

We perform several ablation experiments evaluating different aspects of our model. Table 4.3 describes the results of these experiments on IPCA $_{70,30}$.

#### Unified Target Domains

A plausible baseline for the multi-property optimization problem might be a reduction of the problem to a single property optimization problem. That is, instead of creating a target domain per property, a simplified solution might be to follow the single property optimization architecture UGMMT [BR21] (Chapter 3) and construct a single target domain for all the high property molecules. We then train the model to translate molecules between the source domain, which contains poor property molecules, and the target domain, which contains high property molecules. Furthermore, in that approach the number of training cycles is reduced from four to two and the adaptive loss is not used.

<table>
<thead>
<tr>
<th>Method</th>
<th>DRD2</th>
<th>QED</th>
<th>Similarity</th>
<th>SR</th>
<th>Validity</th>
<th>Novelty</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPCA $_{70,30}$</td>
<td>0.174 ± 0.009</td>
<td>0.762 ± 0.003</td>
<td>0.239 ± 0.002</td>
<td>0.025 ± 0.003</td>
<td>0.946</td>
<td>0.995 ± 0.002</td>
<td>0.997 ± 0.002</td>
</tr>
<tr>
<td>Unified target domains$_{70,30}$</td>
<td>0.071 ± 0.005</td>
<td>0.760 ± 0.003</td>
<td>0.224 ± 0.002</td>
<td>0.013 ± 0.002</td>
<td>0.849</td>
<td>0.999 ± 0.000</td>
<td>0.998 ± 0.002</td>
</tr>
<tr>
<td>Non-adaptive loss$_{70,30}$</td>
<td>0.129 ± 0.001</td>
<td>0.809 ± 0.003</td>
<td>0.257 ± 0.002</td>
<td>0.019 ± 0.002</td>
<td>0.980</td>
<td>0.999 ± 0.000</td>
<td>0.997 ± 0.002</td>
</tr>
<tr>
<td>No embedding$_{70,30}$</td>
<td>0.951 ± 0.006</td>
<td>0.600 ± 0.004</td>
<td>0.136 ± 0.000</td>
<td>0.004 ± 0.001</td>
<td>1.000</td>
<td>1.000 ± 0.000</td>
<td>0.175 ± 0.011</td>
</tr>
</tbody>
</table>

Table 4.3: Ablation experiment evaluation results for IPCA. In bold: best result for SR (cohen’s d effect size over 1).
Hence, we examine a simplified version of our architecture in which there is only one optimized domain $B$ which consists of the union of $B_1$ and $B_2$, i.e., $B$ contains molecules with high $\text{prop}_1$ or $\text{prop}_2$. Figure 4.7 illustrates this architecture. Marked in yellow is the unified component. In this architecture, there is only one optimization path with two parallel training cycles: 

$A \rightarrow \langle A \rangle \rightarrow \langle B' \rangle \rightarrow \langle A' \rangle \rightarrow A'$ and 

$B \rightarrow \langle B \rangle \rightarrow \langle A' \rangle \rightarrow \langle B' \rangle \rightarrow B'$.

We observe that the performance is degraded, likely due to the model’s inability to separate between the two optimization goals. It focuses on the “easy to optimize” property and neglects the other. We notice that QED remains stable while DRD2 is much lower. We deduce that having a target domain per property, together with its encoder, translator and training cycle is useful. It enables IPCA to perform high quality multi-property optimization by first focusing on each property individually and then constraining all these using a shared embedding space with the help of the adaptive loss. Additionally, we conclude that leveraging the strategies of single-property optimization, such as UGMMT [BR21], don’t translate directly to the multi-property optimization task.

**Non-adaptive Loss**

To examine the contribution of the adaptive loss function, we compare our model to a model with a fixed weighted loss: $L = L_{AB} + L_{AB_2} + \lambda_{B_1} \cdot L_{B_1} + \lambda_{B_2} \cdot L_{B_2}$ where $\lambda_{B_1} = \lambda_1 = 30$ and $\lambda_{B_2} = \lambda_2 = 90$. We experimented on several $\text{lambda}$ and report the best results. Although the weights should support the learning of “hard to optimize” properties, we observe that adapting the loss coefficients during training provides su-
No Embedding Constraint

To examine the contribution of the METN embedding (Section 4.1.4), we perform an experiment and discard the embedding $\langle a \rangle$ created from the SMILES encoding and rely only on the source molecule fingerprints ($fp_a$) during inference. It is evident by the results that this embedding is indeed an important factor to the diversity, similarity and the success rate. Intuitively, eliminating the embedding and relying on a subset of information – the source molecule’s fingerprints – limits the ability to generate different outputs.
Chapter 5

Conclusions and Future Directions

We have recently witnessed the urgent need for accelerating the drug discovery process, e.g., COVID-19 drug. Lead optimization is one of the most important steps in the drug discovery chain. First, initial set of hits is identified through the HTS or expert-driven process. Next, in the hit-to-lead step, the most promising hits are selected to advance into the lead optimization stage, where the leads are chemically modified in order to improve biological activity and other critical properties. However, this is a very lengthy and expensive process. The first computational models for this challenge addressed the molecule generation process as a sequence-to-sequence generation problem, representing molecules as sequences of characters, most commonly, SMILES notation was used [GBWD+18, KPHL17, HR18, DTD+18]. These models mainly focused on valid molecules generation, thus creating molecules unrelated to a target. To overcome this barrier, computational models were introduced for molecular lead optimization, which describes generating molecules similar to a drug candidate but with enhanced property. The SMILES notation was abandoned, molecules were seen as graphs and the lead optimization was reformulated as a graph-to-graph translation problem [JBJ18, MPK+20, JYBJ19, FXS20]. This approach is currently the SOTA, demonstrating impressive single-property optimization results.

In the first part of this thesis, we use the simple, yet powerful, SMILES notation and design a single-property end-to-end lead optimization model, combines molecule embedding, fp and unique training technique, which leverages SMILES notation to generate high quality novel molecules and even FDA-approved drugs. Our model outperforms the SOTA, generating more successful molecules with higher desired property score. Our work may revolutionize the field of computational molecule generation, bringing the focus back to the sequence generation methods.

Unlike prior works that required paired supervised data for training, we propose an unsupervised deep generative method for molecule translation, i.e., bidirectional optimization with no paired data required. The model is unique in its ability to si-
multaneously translate between discrete molecule representation and a continuous representation coupled with a double cycle constrained training technique with shared translation components that leverage molecule fp to gain both resemblance to the original molecular lead and generation of novel molecules with enhanced property. The algorithm shows significant performance gains in the success of generating novel optimized molecules that share similarity with a prototype. We conduct comprehensive ablation experiments supporting UGMMT’s analysis and architecture design. Finally, we demonstrated UGMMT’s ability of generating target-based approved drugs it has never encountered before. The system is currently being deployed for use in pharmaceutical laboratories to further analyze the additional generated molecules.

For drug development, the task of optimizing multiple properties of a molecule, while remaining similar to the source molecule, is of extreme importance. As the task is extremely challenging, most prior works in molecular optimization focused on single-property optimization. In the second part of this thesis, we tackled the problem of multi-property molecular lead optimization.

Unlike prior approaches, that required a large training set of pairs of a lead molecule and an enhanced molecule, our approach is unpaired and does not require large paired datasets which are hard to obtain. We introduced a novel integrated poly-cyclic architecture that consists of an optimization path for each property. Those are joined by a shared embedding space from which a multi-property optimization can be inferred. This architecture can be extended to a varying number of optimized properties by adding an optimization path for each additional property and joining it with the shared embedding space. Each optimization path is composed of two training cycles. The parallel independent cycles in our architecture allow it to scale up well for additional properties. For additional property only one encoder and translator should be added (Figure 4.2).

A significant advantage of IPCA is its ability to infer optimized molecules in cases where such examples are not available in the training set. We empirically show that our architecture outperforms current SOTA architectures in terms of the percentage of optimized molecules which meet all optimization goals. We show that the performance gap increases as the training set contains less examples which meet all property constraints. Additionally, we perform multiple ablation tests and identify that our adaptive loss function, that automatically learns to weigh different properties during training, shows notable performance boost. We also compare against other architectural designs and learn that optimizing properties separately with common constraints, rather than together, is not only scalable as the number of optimized properties grows, but it also allows our model to learn with a small number or even without any molecules satisfying all property constraints.

Due to the challenging nature of this task, only two properties datasets have been investigated in prior works. For future work, we plan to test our architecture for a larger number of properties and study the impact of using powerful known architec-
tures such as transformers to replace IPCA’s encoders-decoders, which might boost the performance even more.

We believe our methods lay strong foundations to an automatic-algorithmic drug discovery process and enable lead optimizations without the need for large paired training datasets.


The work is carried out by a group of researchers at Technion Computer Science Department. The research focuses on developing and optimizing small molecules for the treatment of various diseases, with a particular emphasis on the D2 receptor (DRD2).

In the previous study, the researchers highlighted the importance of using covalent bonds in small molecules to achieve high biological activity. They noted a 6% increase in activity for a particular covalent bond.

The model developed in this study is based on a covalent bond optimizer (CBO) and involves a covalent bond optimization loop (CBO-LOOP) to iteratively optimize the covalent bond.

The model is validated using SMILES (Simplified Molecular Input Line Entry System) to encode the small molecules. The optimization process involves a specific covalent bond, which is optimized iteratively until the desired biological activity is achieved.

In the current study, the researchers have applied this model to optimize small molecules for the D2 receptor, with promising results. They have shown that the model can improve the biological activity of the molecules by up to 10% compared to the initial design.

The researchers have also tested the model with a variety of small molecules, showing that it can be applied to different biological targets.

The study concludes with the development of a covalent bond optimizer (CBO) and the validation of its effectiveness in optimizing small molecules for the D2 receptor. The results are promising and suggest that this approach could lead to the development of new and effective treatments for diseases associated with the D2 receptor.
Aims and methods of the optimization process on the molecular structures

The goal of this study is to represent SMILES of the molecules for the purpose of further research.

We employed the SMILES representation of the molecules, which is a simple yet informative way to represent a molecule. The sequence of the characters represents the structure of the molecule, where atoms, bonds, rings, aromatic, and other properties are represented.

With this representation, the molecules are further optimized using the UGMMT tool.

The SMILES representation of the molecules is optimized using the UGMMT tool.

For each optimization, a new model is created and tested.

The process involves two main steps (1) and (2), where the model is converted to the optimized SMILES representation.

In the first model, the optimized function is applied to the molecular structure, and the optimized SMILES representation is created.

In the second model, the optimized function is applied to the molecular structure, and the optimized SMILES representation is created.

The optimized representations are used for further analysis and optimization.

The goal is to create optimized representations of the molecules, which can be used for further research.

For each representation, an optimization process is performed, where the optimized function is applied to the molecular structure, and the optimized SMILES representation is created.

The optimization process involves two main steps (1) and (2), where the model is converted to the optimized SMILES representation.

In the first model, the optimized function is applied to the molecular structure, and the optimized SMILES representation is created.

In the second model, the optimized function is applied to the molecular structure, and the optimized SMILES representation is created.

The optimized representations are used for further analysis and optimization.
תקציר

Billions of dollars and years of development are required to develop a drug, and the process often involves the formulation of the drug to achieve its therapeutic effect. This process is divided into two main stages: the discovery stage, which involves the identification of drug candidates, and the development stage, which involves the optimization of the drug candidates.

In the discovery stage, the goal is to identify potential drug candidates that can be used to treat a disease. This stage involves the identification of drug targets and the use of high-throughput screening (HTS) to identify potential drug candidates.

In the development stage, the goal is to optimize the drug candidates to improve their efficacy and selectivity. This stage involves the optimization of the drug candidates to improve their pharmacological properties and selectivity.

Overall, the development of a drug requires a significant investment of time and resources, and the process is often long and complex.
The work presented in this thesis is done under the supervision of Dr. Kira Radinsky, and is supported by her guidance, support, and encouragement.

Appendix A: References


Danita, Anir. The Choice. Tel Aviv: Dr. Kira Radinsky, Department of Computer Science, Technion, 2021.
מודלים גנרטיביים מבוססים למידה עמידה
עבורי אופטימייזציה של מחלקות

 giriş

لغיש מילוי חלקי של הדרישות לבלגת התואר
מגיש/ית למודיעי בمواد החשב

 ג'יא ברשם

הנהלת הטכניון — מכון טכנולוגי לישראל
אולק החשיפה'א חספ
אוגוסט 2021
מודלים גנרטיביים מבוססי למידהعمוקה עבור אופטימיזציה של מולקולות

גיא ברשניקי