ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Integrating Latent Variable and Auto-Regressive Models for Enhanced Goal Directed Molecule Generation

Master Thesis in Data Science

Arthur-Louis Heath

July 24, 2024

Supervisors:
Dr. Amina Mollaysa
Prof. Dr. Ender Konukoglu
D-INFK, ETH Zürich

Acknowledgements

I am grateful to my supervisor, Dr. Amina Mollaysa, for her patience, guidance and co-authorship of the workshop paper on which this manuscript was based. Additionally, I would like to acknowledge the assistance of OpenAI's ChatGPT for providing spell checking and editorial suggestions during the preparation of this manuscript.

Contents

Chapter 1

Introduction

In recent years, the use of generative machine learning for molecule generation and drug design has become increasingly popular. These techniques promise to revolutionize the process of discovering new drugs by significantly reducing the time and cost required to bring them to market. By increasing the rate of development for new treatments, these technologies stand to improve the quality of life of many people.

As research in molecule generation has progressed, its focus has shifted to increasingly sophisticated molecular representations and model architectures. Through these advancements, researchers have achieved increasingly fine-grained control over the molecule generation process. State-of-the-art approaches can now control the precise 3D positions of atoms and generate candidate drugs that target specific diseases. However, such techniques bring issues related to computational cost, complexity, and data scarcity.

An early approach, which endures to this day, involves representing molecular graphs as sequences of characters, allowing them to be processed as text strings. Recent advances in natural language processing, particularly the success of large transformer-based models, have reinvigorated interest in text-based molecular representations. However, unanswered questions remain regarding the performance of older techniques, with many of their limitations remaining unaddressed. In light of this, we believe the time is right to revisit these methods and examine whether they can be brought up to modern performance standards.

In this work, we revisited two classical modeling approaches: auto-regressive (Section 3.2) and latent variable models (Section 3.3). Each of these model classes has its own distinct advantages. Autoregressive models are able to accurately model the distribution of molecular sequences and control generation well. In comparison, latent variable models offer increased flexibility, allowing them to perform multiple molecular design tasks jointly. However,

they also struggle to generate valid molecular sequences and have trouble controlling generation (Section 3.3.1).

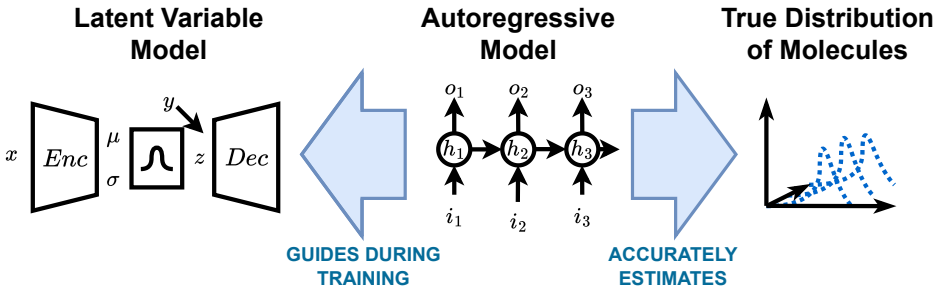

Figure 1.1: A high-level overview of our method.

In this report, we will present our method for combining the strengths of both approaches. We propose to use a hybrid training approach (Section 3.4), where a pre-trained autoregressive model is used as a surrogate model for the true distribution of molecules. It is then used to guide the training of the latent variable model to improve the validity and accuracy of its generations (Figure 1.1). Additionally, we aim to achieve this performance increase while preserving its flexibility. This approach offers an attractive alternative to previous methods for improving such models' performance, which rely on more complex and computationally expensive molecular representations.

To validate our methodology, we conducted an experimental study on two commonly used benchmark datasets. Our results demonstrated that our proposed hybrid approach effectively combines the strengths of the two model classes and greatly improves the performance of latent variable models. Furthermore, we achieve performance comparable to more modern methods that rely on complex molecular representations. Importantly, our findings also challenge the prevailing assumption in the literature about the detrimental effects of strong decoder models on the learned latent representations.

The remainder of this report is structured as follows:

1. **Chapter 2. Background and Literature Review:** This chapter introduces the motivation for the development of molecule generation methods, explain the key design decisions and tasks that delineate the field and examine prior work relating to our method.
2. **Chapter 3. Methodology:** Formally introduces both model classes and explain what molecule generation tasks they can solve. Introduces the methodology and explains how it can be integrated into model training.
3. **Chapter 4. Experimental Setup:** Presents the details of our experimental study, including evaluation methodology, datasets used and implementation details.

4. **Chapter 5. Results and Discussion:** Discusses the results of our study, analyses models' behavior and discusses the benefits/drawbacks of our method.
5. **Chapter 6. Conclusions and Further Work:** Contextualizes our results within the goals of this project and discusses promising avenues for future work.

Chapter 2

Background and Literature Review

Before formally introducing our methodology, it is important to understand the broader context of molecule generation, its place in the drug discovery process and the positioning of our work in the context of prior research. In this chapter, we will present the following:

- Introduce drug development and the use of generative machine learning for discovering candidate compounds.
- The major design decisions that determine what molecule generation techniques can be used, namely the choice of molecular representation and specific sub-task to be solved.
- Relevant prior work on the generation of molecules represented by molecular sequences.
- Provide a summary of the scope, methodology and objectives of this study.

2.1 Molecule Generation with Generative ML

The modern process of drug development is notoriously expensive, complicated and lengthy. Before being brought to market, candidate molecules must pass multiple screening steps, preclinical/clinical trials and regulatory review. The cost of discovering an individual drug has been estimated to be as high as 2.6 billion dollars (Zhang et al., 2023b), with an average time to market of twelve years (Agrawal et al., 2023). Furthermore, the success rate of an individual candidate successfully passing the various screening stages is notoriously low (Jiang et al., 2023), and many compounds must be abandoned after significant investment.

The discovery stage is an important first step in the drug development process. The aim at this stage is to identify candidates that possess qualities

desirable for drugs. For example, we may wish to design candidates that are likely to bind to a specific protein, inhibiting a target disease pathway. Alternatively, we may try to find molecules with favourable general physiochemical properties to further screen for useful drugs. For example, we may wish to find molecules with appropriate solubility (that can be absorbed through the digestive tract) or molecules easily synthesized from known compounds in a laboratory. Strong candidates are more likely to make it to the end of the pharmaceutical research pipeline and are less likely to be abandoned after multiple lengthy and expensive screening steps. Improvements at the discovery stage can, therefore, greatly reduce the cost and length of the drug design process.

Conventional approaches to identifying promising candidates usually involve computer-aided drug design (CADD) techniques such as virtual screening, scaffold hopping and molecular docking (Pang et al., 2023a). Researchers typically start from a large library of candidate compounds, often generated using chemo-informatics software. These will then pass through multiple stages of *in-silico* screening to filter out unsuitable compounds before moving to *in-vitro* experiments. However, these techniques can only explore a small portion of the space of possible drug-like molecules. It has been estimated that only 10^8 out of 10^{60} synthetically accessible drug-like molecules have been discovered to date (Mokaya et al., 2023). Recent advances in deep learning represent an attractive alternative. Compared to CADD techniques, generative machine learning (ML) models can be trained to generate candidates with specific properties and behaviours directly. Importantly, they also allow us to explore new regions of the chemical space, uncovering completely new classes of compounds.

Although relatively young, these techniques are quickly growing in popularity with both researchers and industry. The state-of-the-art has progressed from generating only small molecules and controlling only relatively simple molecular properties, to generating candidate molecules that target the 3D structure of binding sites on specific proteins. Furthermore, these techniques are already seeing real-world adoption. There are several small-to-medium enterprises operating in the space, as well as significant investment from corporate and state actors (Qureshi et al., 2023). Although not yet the industry standard, several compounds discovered with generative ML have already progressed to advanced stages of the pharmaceutical research and development pipeline (Arnold, 2023).

As a whole, ML-based drug discovery has become an enormous field of research, encompassing many varied approaches, techniques and sub-tasks. Most breakthroughs in generative ML are quickly adapted for molecule design (Pang et al., 2023b). Furthermore, researchers have designed many tailored and combined approaches that leverage prior domain knowledge

from chemistry and physics (Zhang et al., 2023a; Zhang and Liu, 2023; Zhang et al., 2023c). To be able to discuss relevant works, we must, narrow the scope of our examination. Fortunately, the specific techniques that constitute the state-of-the-art depend greatly on several key design decisions. Chief among these are the choice of molecular representation (how a molecule is represented in the computer) and the specific sub-task or problem within molecule generation we wish to solve. In the following sections, we will discuss each of these design decisions, identify what is most relevant to our work, and examine the trade-offs involved.

2.2 Molecular Representations

A critical design decision when modeling molecules is how to represent them in a computer. Broadly, the choice of representation determines a trade-off between fidelity/accuracy and efficiency. For example, a simple representation that only considers atom types and connectivity (a molecule's 2D graph) will be fast, easy to work with, and will have large quantities of data readily available. However, it might ignore important information about the relative 3D positioning of the atoms that might be relevant for determining a molecule's tendency to bind with a desired protein. The trend in the literature over time runs from simple to complex (Mokaya et al., 2023). Early research typically focused on simple string-based (1D) and graph-based (2D) representations. As the field matured, the focus switched to more information-rich but higher-fidelity representations, particularly ones that consider the 3D structure of a molecule.

Crucially, the choice of representation largely determines what machine learning techniques can be applied to a problem. For example, modeling the spacial conformation of a molecule typically requires the use of techniques that mode rotational and transnational invariances (Isert et al., 2022). The following section presents several molecular representations that are particularly popular for ML-based molecule generation.

2.2.1 Common Molecular Representations

Molecular Embeddings and Fingerprints: Embeddings learned by ML models can be considered a form of molecular representation. Encoding molecules as continuous vectors can facilitate various downstream tasks, including further processing with more specialized models, information fusion (Zhang and Liu, 2023), and the use of continuous optimization techniques (Gómez-Bombarelli et al., 2016). The field of molecular representation learning focuses on developing frameworks that produce general-purpose embeddings of molecules and remains an active area of research (Xia et al., 2023; Liu et al., 2023). A similar concept is molecular fingerprinting. A molecular

fingerprint is a vector encoding a molecule's structure and properties. They are commonly used in bio-informatics as such representations are easy and efficient to process with a computer. The most widely used of these is the Morgan fingerprint (Capecchi et al., 2020). A molecule's Morgan fingerprint is computed by encoding the substructure within a specified radius around each atom and hashing these values into a fixed-length vector, with each part indicating the presence or absence of a specific substructure.

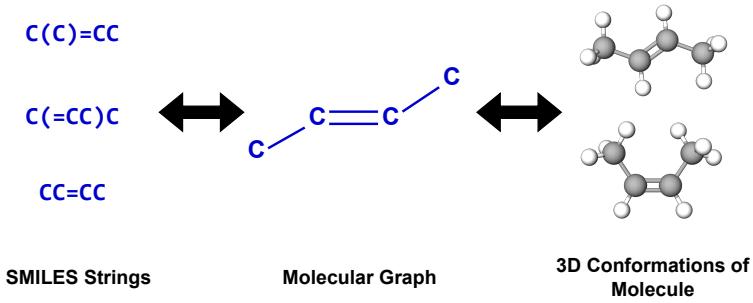

Figure 2.1: Illustration of many-to-many relationship between SMILES strings and 3D molecular conformations for but-2-ene.

SMILES: A historically popular method for molecular representation is as sequences of characters. The most popular of these, and the representation considered in this project, is the simplified molecular line entry system or SMILES strings (Weininger, 1988). A SMILES string encodes a molecule's constituent atoms and connectivity graph as a sequence of characters. Individual symbols represent atom types, chemical bonds, rings, aromaticity and branching information. SMILES strings are lightweight and easy to process. Additionally, they facilitate using powerful techniques and architectures from the field of natural language processing.

However, despite these attractive characteristics, SMILES strings possess several drawbacks. Firstly, they are non-unique, as one molecule can be represented as several valid SMILES strings. This issue can be addressed using one of several available canonicalization algorithms (O'Boyle, 2012) or data augmentation techniques, such as randomization (Arús-Pous et al., 2019). A further issue of this representation is its sensitivity, as small perturbations to a SMILES string can easily result in a string that no longer maps to a valid molecule; for example it may violate valance constraints. The result is that models can generate strings that do not map to valid molecules, one of the central issues tackled in this project. Finally, SMILES strings do not encode the relative 3D positioning of atoms. Due to aspects of molecular structure such as chirality and rotatable bonds, a single SMILES string can correspond

to several different conformations of a molecule in 3D space. Since the conformation of a molecule directly affects its interactions and properties, this represents an important loss of information. Figure 2.1 illustrates this on the example of but-2-ene. It shows the three possible SMILES strings encoding its molecular graph and the two possible conformations of this molecule in 3D space (its cis and trans isomers).

Various extensions of SMILES have been proposed to address these and other drawbacks, of which we list a few notable examples. DeepSMILES introduced by O'Boyle and Dalke (2018) addresses the issue of syntactically invalid generations due to parenthesis mismatch and incorrect ring closure. A recent work by Wang et al. (2023a) uses a variant of smiles with enhanced ring encodings called FSMILES in a 3D context. They are used in conjunction with coordinates as an intermediate representation before being fully decoded into a 3D structure. However, the authors of Arús-Pous et al. (2019) and Bjerrum (2017) observe that using complicated SMILES variants can negatively impact performance.

Due to these drawbacks, molecule design research has largely moved to more complex representations. The authors of Mokaya et al. (2023) conclude that the future of *de novo* drug design likely lies in higher dimensional representations that consider the 3D structure of molecules. Despite this, SMILES strings remain popular due to their simplicity and ease of processing. Additionally, the recent success of large transformer-based models for natural language processing has brought them back into focus. Many recent works choose to use SMILES, both in the context of large language models such as Sakhinana and Runkana (2023); Wang et al. (2023a); Brown et al. (2020) and the improved performance of transformer architectures as in the case of (Born and Manica, 2023; Izdebski et al., 2023; Wang et al., 2023b; Feng et al., 2024).

Molecular Graphs: Many methods can operate on graphs directly without serialisation. In this case, the graph information can be encoded as a set of matrices. For example, as a pair of matrices. One connectivity matrix indicates the presence and type of bonds and one atom species matrix encodes atom types (an example is provided in Figure 2.2). Similarly to SMILES, there is a large body of research on generating graphs in this form. Additionally, since the edge matrices of molecular graphs are typically sparse, the overall representation is still more lightweight than a volumetric one (Pang et al., 2023b). Finally, graph-based representations can be easily extended to a 3D representation by storing a set of coordinates for each atom.

A drawback shared with SMILES is that of validity. A generated graph must still obey structural constraints such as atom valency. Additionally, the permutation invariance of a graph's nodes must be considered (Guo and

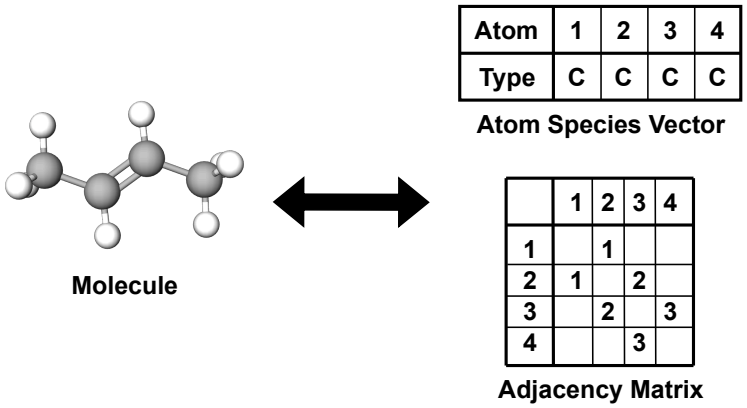

Figure 2.2: Graph-based representation of but-2-ene with matrices. In this case, the atom species vector encodes atom types and the adjacency matrix their connections (the number of covalent bonds).

Zhao, 2022). Another shared downside with SMILES is the loss of structural information when using a simple graph to represent a molecule. If we wish to encode the 3D position of atoms, we must also account for translational and rotational invariances in 3D space.

Volumetric Techniques: In addition to 3D molecular graphs, it is possible to represent molecules directly in 3D space. A historically popular example of this are so-called "Density Grids". These involve discretizing 3D space into a grid of voxels, which contain probability densities for individual atoms. These allow for the use of approaches from computer vision, such as convolutional neural network (CNN) based variational auto-encoders (kali et al., 2019) and adversarial auto-encoders (Ragoza et al., 2020, 2021).

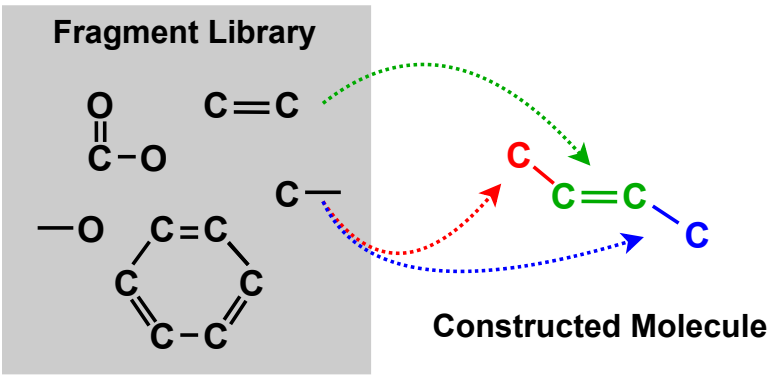

Figure 2.3: Illustration of how a molecule can be constructed from a set of motifs/fragments. Note that constructing molecules in this manner requires both a choice of representation for the fragments and defining a process for their combination.

Combined Representations: Many sophisticated models take a multi-modal approach to representation. Combining multiple representations can help mitigate the downsides and accentuate the benefits of each approach at the expense of modeling and computational complexity. An example of a 1D intermediate representation in a 3D pipeline is Lingo3DMol developed by Feng et al. (2024). An additional example is SQUID (Adams and Coley, 2022), where a point cloud and a graph representation are jointly encoded.

Molecular Motifs: A design decision related to representation is whether to construct molecules from molecular fragments. Typically, this involves constructing a vocabulary of substructures, called fragments or motifs, either from the molecules in the training dataset or a well-defined splitting process (Yang et al., 2022). The final molecule is then generated by combining these substructures (an illustration of this can be found in Figure 2.3). Treating molecules as motifs enforces a chemical prior on a model, effectively acting as a strong regularizing factor (Zhang et al., 2023c). Ensuring individual motifs are chemically valid and constraining how they can be combined can help ensure the chemical validity of the final structure. Note that motifs are not a complete representation, as the individual substructures still need to be represented in some way.

2.3 Tasks in Molecular Design

Another facet of molecule generation that determines how the problem must be approached is the specific task we aim to solve. Depending on the context, we may wish to generate a set of new molecules entirely from scratch or prefer to optimize an existing molecule to improve its properties. We must also consider what properties of a molecule we wish to control, and how we wish to control them. For example, when considering a measure of a molecule's toxicity, we may wish to decrease it as much as possible, but when considering its solubility in water, we may wish to keep its value within a specific range. In the following sections, we will discuss three major tasks in drug design, the first two of which can be directly tackled by the model classes presented in this report.

2.3.1 Targeted Molecule Generation

Targeted molecule generation is the task of generating molecules with desired properties from scratch (*de novo*). In this task, the aim is to generate a diverse set of novel molecules with a set of desirable properties. More precisely, we consider a set of numerical scores or metrics derived from a molecular representation that act as stand-ins for a molecule's physiochemical properties. Typically, we want the generated molecules to possess

properties that make them "drug-like". One example could be a measure of a molecule's hydro/lipo-philicity, such as its LogP score. Another are complex aggregate metrics aiming to quantify an abstract concept, such as the "drug-likeness" of a molecule. Examples of commonly used complex metrics are the quantitative estimation of drug-likeness (QED score) (Bickerton et al., 2012), Lipinski's rule of five (another estimate of drug-likeness) (Lipinski, 2004) and a molecule's synthetic accessibility score (i.e., an estimate of the ease of synthesizing a compound) (Ertl and Schuffenhauer, 2009).

The real-world requirements for suitable drug candidates are both complicated and multifaceted, which means that realistically, multiple properties need to be controlled at once. In a practical setting, we are typically interested in multi-property control. Additionally, since these scores are often approximations, their accuracy can vary. This fact interacts with the choice of representation, as estimations of molecular properties for low-fidelity representations are necessarily less accurate than high-fidelity ones.

Depending on requirements, there are many different approaches to implementing property control. For example, models can be provided with a reward computed from molecules property scores in a reinforcement learning setting (Olivecrona et al., 2017; Tang et al., 2024). Another approach (the one considered in this study) is to provide property scores as conditioning information to a model during training. The model's generation can be guided by setting the conditioning information at inference time. Finally, if we can train a model to map discrete representations of molecules to a continuous latent space, we can search it for desirable molecules using Bayesian optimization techniques.

2.3.2 Molecular Optimization

Instead of generating new molecules from scratch, we may wish to make small alterations to a known prototype molecule. The task of adjusting a molecule to improve its properties is called molecular optimization. When searching for candidate drugs, it is often easier to start with a promising lead molecule and improve it than to generate new molecules completely from scratch. The rationale behind this is, that if two molecules are structurally similar, they will likely possess similar properties, and it should be simple to synthesize one from the other. This mirrors how candidate compound identification is traditionally performed using an expert's intuition to improve a lead molecule (He et al., 2020; Xia et al., 2024).

As with targeted molecule generation, molecular optimization can be achieved with ML in diverse ways. One example are latent search based methods, which first encode a molecule into a low-dimensional latent space. The neighbourhood of the encoding can then be searched for improved versions of the source molecule (Jin et al., 2018a). Alternatively, it is possible to di-

rectly search the high-dimensional and discrete space of molecules using reinforcement learning (Popova et al., 2017; Olivecrona et al., 2017; Jeon and Kim, 2020). A final example involves re-framing the problem as a translation task. For example, as translation between SMILES sequences (He et al., 2020) or molecular graphs (Jin et al., 2018b). In this case a model learns to modify a molecule from pairs of molecules with similar structures and significantly different properties.

2.3.3 Structure-Based Drug Design

So far, we have considered optimizing a set of general properties (typically ones that make a molecule "drug-like"). However, what ultimately determines the usefulness of a drug, is its efficacy in treating a specific disease. If a disease's pathway (the series of interactions that cause it) in the body is sufficiently understood, it is possible to identify a specific protein that can be inhibited or activated to treat it. We can then design a drug that binds to a specific protein binding site (or pocket) to treat a specific disease; this approach is called structure-conditioned drug design.

The ability of a compound to bind to a protein pocket can be modelled with a stand-in score (called a docking score). This allows us to treat structure-based drug like targeted generation with other scores, and use the methods described in Section 2.3.1. However, these can often be inaccurate and difficult to operate with (Zhou et al., 2018). What generally sets structure-based drug design apart is the ability to leverage knowledge of the detailed structure of the target protein pocket. Many contemporary techniques consider the full 3D arrangement of atoms in the target pocket and their interaction with the detailed 3D conformation of the target drug (Zhang et al., 2023c,a; Zhang and Liu, 2023; Adams and Coley, 2022).

Although it may always seem preferable to target generation in this way, it requires a deep understanding of the disease and detailed information about the target protein pocket. Additionally, the availability of training datasets with drug-pocket pairs is limited (Zhang and Liu, 2023).

2.4 Prior Work on Molecule Generation with SMILES

As mentioned in Chapter 1, our goal with this project is to examine classical methods using simpler string-like representations to see whether they can be brought up to scratch. Specifically, we are interested in latent variable and auto-regressive approaches to generating molecules represented as SMILES. In terms of tasks, latent variable models (and our approach presented in Chapter 3) can be jointly used to tackle both conditional generation and molecular optimization by manipulating latent encodings. We will limit the remainder of our discussion to methods within this scope. The following

sections will discuss prior work using auto-regressive and latent variable architectures to generate SMILES, compare both approaches, and identify previous attempts to combine them.

2.4.1 Auto-Regressive Models for SMILES Generation

Early auto-regressive approaches to molecule generation (Segler et al., 2017; Bjerrum and Threlfall, 2017) showed that auto-regressive models can successfully learn a distribution of SMILES strings and generate realistic candidates with high accuracy. Furthermore, Segler et al. (2017) proposed that targeted generation can be achieved by generating a large set of candidates to filter or by further fine-tuning the model on a curated dataset of desirable molecules. Methods have also been developed to control the properties of generated molecules directly, avoiding further fine-tuning. For example, concatenating conditional information to the model's inputs during generation or encoding conditional information into the model's initial hidden state (Kotsias et al., 2020).

A parallel approach to guiding auto-regressive models for conditional generation uses policy optimization and a set of objective functions to fine-tune auto autogressive models. In this case, the model is trained by providing a reward computed from previous generation's property scores. The seminal works implementing this method are REINVENT by Olivecrona et al. (2017) and ReLeaSe by Popova et al. (2017). Recent works have built on this by incorporating various innovations from the field of reinforcement learning, such as curiosity (Thiede et al., 2020) and curriculum learning (Mokaya et al., 2023).

Following their success in natural language processing, many recent publications have adopted transformer-based architectures for molecule generation. These can be used as auto-regressive models or to generate sequences in one shot. MolGPT (Bagal et al., 2021) is a notable example, where the authors applied the Generative Pre-Trained Transformer (GPT) architecture to molecule generation. Other approaches have expanded beyond simple conditional generation, such as generating molecules from natural language descriptions (Sakhinana and Runkana, 2023), knowledge augmentation to inform molecule design (Wang et al., 2023b) and exploring the potential of large language models trained on natural language for conditioned design of simple molecules (Wang et al., 2023a; Sakhinana and Runkana, 2023).

Additionally, Wang et al. (2019) and Grechishnikova (2019) propose transformer-based approaches for structure based drug design. Moreover, the authors of Wang et al. (2021) combine several prior approaches into a multi-step pipeline, distilling knowladge from a transformer model to a simpler recurrent neural network (RNN), which is then further fine-tuned using reinforcement learning.

Although transformer-based approaches can outperform older architectures such as RNNs and long short-term memory cells (LSTMs) for conditional generation, the difference is minimal in simpler settings Bagal et al. (2021). For this reason, and to maintain a similar architecture between the combined models, an LSTM-based architecture was used in this study.

2.4.2 Latent Variable Models for SMILES Generation

The seminal application of latent variable models is the work by Gómez-Bombarelli et al. (2016). In their work, they train a character VAE model that encodes SMILES as continuous latent vectors. They train an additional network to predict molecular properties from latent points, with which they can perform gradient-based optimization. The authors of Lim et al. (2018) propose an approach based on a conditional character VAE to perform molecular optimization through style transfer.

However, both these works struggle to generate a high proportion of valid SMILES. Researchers have addressed these issues through representational modifications. In particular, grammar VAE (GVAE) by Kusner et al. (2017) is trained to generate sequences of derivations of the SMILES grammar instead of sequences of tokens. By only allowing valid derivations of the SMILES grammar, it can significantly increase the proportion of valid SMILES. This approach was later refined by syntax-directed VAE (SD-VAE) (Dai et al., 2018), which considers an augmented (attribute) grammar to increase validity further. Considering attribute grammars allows the authors to encode additional semantic constraints not modelled by the SMILES context free grammar.

Later works have tried to overcome the issue of invalid SMILES by avoiding string-based representations altogether. Popular alternatives include graph-based representations (You et al., 2018; Liu et al., 2018; Simonovsky and Komodakis, 2018) and fragment-based junction tree methods (Jin et al., 2018a).

A parallel class of latent variable model applied to drug design are adversarial auto-encoders (AAEs). The AAE framework was first introduced by Makhzani et al. (2015) and combines the VAE and generative adversarial network (GAN) architectures by training a GAN on the continuous latent representations of molecules. For instance, Hong et al. (2019) applied AAEs to generate drug-like molecules, showing that adversarial training can help learn a more robust latent space. Additionally, Kadurin et al. (2017) utilized adversarial networks to refine molecular generation further, demonstrating improved property optimization and diversity of generated compounds. Although AAEs can generate more realistic and desirable compounds, they suffer from issues such as training instability and mode collapse, requiring careful tuning to train the network effectively.

2.4.3 Comparison of Auto-Regressive and Latent Variable Models

As mentioned in the previous section, the main challenge in training latent variable models (specifically VAEs) on SMILES strings has been ensuring the validity of generated sequences. We present an analysis of the reasons for these struggles in Section 3.3.1. The solutions previously discussed Kusner et al. (2017); Dai et al. (2018) rely on more complex representations that are both heavier and more complex. Parsing attribute grammars, in particular, is a complicated and chemically involved process. Conversely, auto-regressive models can achieve a high level of validity on non-trivial datasets (Bagal et al., 2021) by simply predicting each consecutive SMILES character. Despite this, latent variable models are still attractive, as they allow powerful latent manipulations, such as gradient-based optimization (Gómez-Bombarelli et al., 2016) and molecular optimization through style transfer (Lim et al., 2018) to be performed.

2.4.4 Combining Latent Variable and Autoregressive Models

Prior attempts at combining these two model classes have focused on introducing latent variables into auto-regressive models. Early examples include Chung et al. (2015) who developed VRNN and others (Bayer and Osendorfer, 2014; Boulanger-Lewandowski et al., 2012; Fabius and van Amersfoort, 2014). However, the goal of these approaches is to introduce more stochasticity into the model, as a standard RNN has limited power to model the variability observed in highly-structured data. Integrating latent variables into the hidden state of the RNN allows it to draw on an additional source of variability and consequently increase sample diversity. Unlike latent variables in VAEs, this type of latent variable cannot control aspects of the generated molecules' structure.

Recently, Bird and Williams (2019) introduced an approach similar to VRNN, incorporating a latent variable into an RNN model. In addition to adding variability, this work aims to capture sequence-specific features through the latent variable and use it to perform style transfer. The parameters of the RNN is made dependent on a latent variable that is inferred from input-output sequence pairs. This approach enables each sequence to be modelled as a single dynamical system and capture instance-specific variation. Due to auto-regressive models' strong conditional generation performance, an interesting problem for future research might be performing style transfer in an unsupervised manner, without needing to construct a paired dataset.

2.5 Research Objectives and Scope

Recently, there has been a noticeable shift towards more complex and heavy representations, sophisticated architectures and fine-grained structural de-

sign tasks. Instead of continuing this trend, this work will revisit classical approaches using the SMILES representation. It will re-evaluate their challenges and attempt to enhance their performance to meet modern standards. Concretely, we will consider latent variable models (specifically modifications of character VAEs) and auto-regressive models. We will also critically re-examine overlooked assumptions in the literature.

The project can be summarized as follows:

1. **Scope:** We will focus on simpler SMILES-based representations and classical approaches (autoregressive models and latent variable models).
2. **Methodology:** We present a hybrid approach between the two model classes that leverages the strong conditional generation performance of autoregressive models to improve the performance of a latent variable model.
3. **Goals:** The goal of this work is to improve the performance of latent variable models to be able to successfully tackle conditional generation and molecular optimization jointly, without the need for complex representations or paired datasets.

Chapter 3

Methodology

In the previous chapter, we examined the literature surrounding latent variable and auto-regressive models for generating SMILES sequences. We also discussed the drawbacks of each model class and outlined the goal of this project: to combine the two. In the following sections, we will formally introduce our methodology for achieving this. Specifically, we will:

- Discuss the conditional generation and molecular optimization tasks.
- Formally introduce latent variable models and auto-regressive models and explain how they can be used for molecule design.
- Outline the central research question and project goals.
- Present our methodology for solving the allotted task.

3.1 Formalization of Conditional Generation and Molecular Optimization Tasks

Suppose we are given a training set of pairs $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}; i = 1, \ldots, N$, where each $\mathbf{x}_i$ corresponds to a molecule (in our case represented as a SMILES sequence) and $\mathbf{y}_i$ is a tuple of one or more of its corresponding molecular property scores. **Conditional generation** of molecules can be framed as the task of learning the conditional distribution $p(\mathbf{x}|\mathbf{y})$. With this distribution, we can generate new molecules with desired properties $\mathbf{y}'$ by sampling $\mathbf{x}' \sim p(\mathbf{x}|\mathbf{y})$. Sections 3.2 and 3.3 explain how this task can be tackled both with autoregressive architectures and with the decoder of a VAE-like latent variable model.

On the other hand, molecular optimization involves changing a molecule's structure to obtain a new molecule with more favorable properties. In this setting, we aim to perform minor edits to molecule $\mathbf{x}$ to obtain molecule $\mathbf{x}'$. We want our new molecule $\mathbf{x}'$ to have properties close to our target $\mathbf{y}'$ while

staying as structurally similar to **x** as possible. Note that the properties of a molecule are a direct result of its structure, making it impossible to change a molecules properties without also changing its structure. Furthermore, the degree to which the two can be affected independently depends on the specific molecule and the properties under consideration. In Section 3.3, we will explain how this can be achieved through style transfer of molecules (Mollaysa et al., 2019).

3.2 Autoregressive Models for Conditional Generation

When working with SMILES strings, a molecule **x** is represented as a sequence of characters $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ from a finite set of symbols. As with other sequence generation tasks, we can train an autoregressive model to model the next-character conditional probabilities for each step in a sequence $p(\mathbf{x}_i|\mathbf{x}_{<i})$. This can be used to generate new sequences one step at a time.

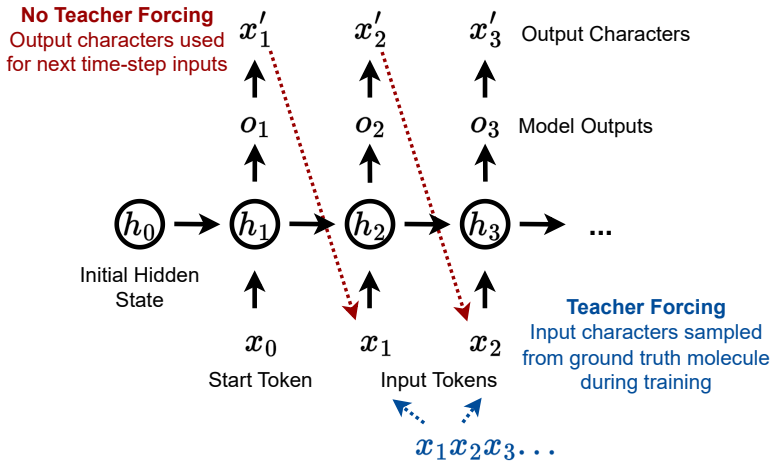

Figure 3.1: Diagram of an autoregressive (RNN) architecture unrolled across time-steps.

One method of adapting such models for conditional generation is by providing a molecule's properties as conditional information during training. This can be achieved by concatenating the conditional information to the previous token/s provided to the model as an input and results in next-character conditional probabilities $p(\mathbf{x}_i|\mathbf{y}, \mathbf{x}_{<i})$. Throughout training, the model learns to rely on this information to inform generation, and can later be used to guide generation during inference. Specifically, sequentially sampling characters while providing target properties $\mathbf{y}'$ as conditioning information allows us to draw samples from $p(\mathbf{x}|\mathbf{y})$ and perform conditional generation. A diagram presenting how this architecture can be implemented using a recurrent neural network (RNN) is presented in Figure 3.1.

A design decision related to autoregressive models relevant to this work is weather or not to use teacher forcing during training. For an input sequence $x_1 x_2 \dots x_n$, teacher forcing involves providing the token x_{t-1} from the input sequence for training step t. Alternatively, when teacher forcing is not used, the model's outputs at time-step $t-1$ determine the input character at t. Teacher forcing can improve model performance and training efficiency, but is not always an appropriate design choice for reasons discussed in Section 3.3.1.

3.3 Conditional Generation and Molecular Optimization With Latent Variable Models

Latent variable models can be used to jointly perform both conditional generation and molecular optimization through style transfer. We can perform both these tasks by modelling the conditional distribution $p(\mathbf{x}|\mathbf{y},\mathbf{z})$, where $\mathbf{z}$ is a learned latent variable (with a known form) corresponding to aspects of a molecule's structure $\mathbf{x}$ independent of its properties. Disentangling a molecules property-dependent and property-independent structure in this manner allows us to control each independently, and as we will later see, to perform style transfer of molecules.

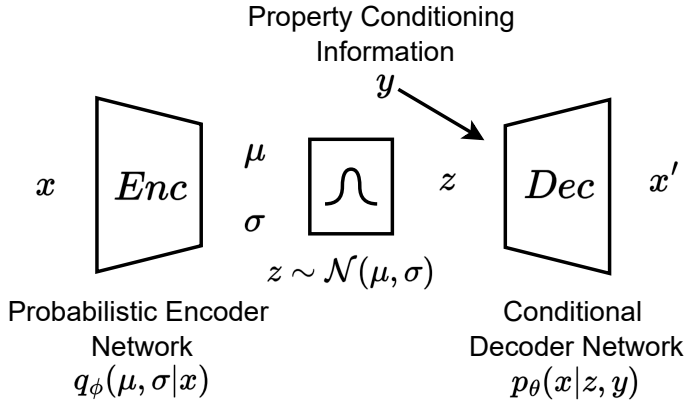

Figure 3.2: The (conditional) VAE architecture. Input instances are encoded as distributions in a latent space. Output instances are constructed from latent points and the conditioning information.

We can model this distribution using a joint generative model of the form $p_\theta(\mathbf{x},\mathbf{y},\mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})p(\mathbf{y})p(\mathbf{z})$, where $p(\mathbf{y})$ and $p(\mathbf{z})$ are assumed to be independent, and $p(\mathbf{y})$ is of a known form. Since we cannot directly access a molecule's property-independent information $\mathbf{z}$, we must infer it by fitting an additional model $q_\varphi(\mathbf{z}|\mathbf{x})$. This model can be trained using the Evidence

Lower Bound (ELBO) objective function of the form:

$$\mathcal{L}_{ELBO}(\theta, \varphi) = \sum_{i=1}^{N} \Big\{ \mathbb{E}_{q_\varphi(\mathbf{z}_i|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i|\mathbf{y}_i, \mathbf{z}_i)] - D_{KL}(q_\varphi(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i)) \Big\} \tag{3.1}$$

The first component (the reconstruction loss) ensures the model can reconstruct an encoded input sequence from its latent representation $\mathbf{z}$ and conditional information. The second loss component measures the error between the learned distribution of the latent variable and the prior $p(z)$. It does so by estimating the Kullback-Liebler (KL) divergence (Kullback and Leibler, 1951), an information-based measure of the difference between two probability distributions. When trained with this objective, this model corresponds to a conditional version of the Variational Auto-Encoder proposed by Kingma and Welling (2013). An architectural diagram of the model is presented in Figure 3.2.

By conditioning this model, the aim is for the decoder distribution/network $p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})$ to rely on the property information $\mathbf{y}$ to inform property-dependent and $\mathbf{z}$ to inform property-independent aspects of a sampled molecule $\mathbf{x}'$. However, it can be difficult to achieve this in practice for the reasons discussed in 3.3.1. A final point is that we are working with molecules encoded as sequences. We must, therefore, implement both the decoder and encoder $q_\varphi(z|x)$ as networks that can handle sequential data. In this case, we model the encoder distribution with a convolutional neural network (CNN) and the decoder distribution as an autoregressive model.

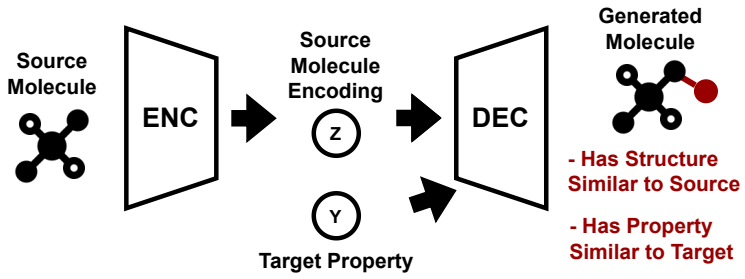

Figure 3.3: Diagram of style transfer of molecules. The source molecule encoding informs the property-independent structure and the target property information the property-dependent structure of the generated molecule.

With this model, we can perform conditional generation by selecting a random latent point $\mathbf{z}$ and sampling a new molecule from the decoder

$\mathbf{x}' \sim p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})$. The model can also be used to perform style transfer of molecules, as presented by Mollaysa et al. (2019). A source molecule $\mathbf{x}$ can be encoded to obtain a latent point $\mathbf{z}$. This point can be decoded with the target properties $\mathbf{y}'$ to obtain a generated molecule $\mathbf{x}'$. If the decoder has adequately disentangled $\mathbf{x}$ and $\mathbf{z}$, this should result in a molecule with a structure similar to $\mathbf{x}$ (due to the shared $\mathbf{z}$) and property values similar to the target $\mathbf{y}$ (Figure 3.3).

3.3.1 Training Difficulties of Latent Variable Models

In practice, such models can suffer from several issues. The standard ELBO objective only focuses on successfully reconstructing molecules. If the conditional information $\mathbf{y}$ only weakly informs reconstruction, the decoder will learn to rely entirely on $\mathbf{z}$ to reconstruct the molecule. In this case, we will lose the ability to control generation through conditioning information negatively impacting style transfer.

The converse scenario is also possible. If the decoder relies too heavily on $\mathbf{y}$ for reconstruction, the model may disregard $\mathbf{z}$, resulting in a degraded latent space. In this case, the decoder collapses to an autoregressive model $p(\mathbf{x}|\mathbf{y})$. This collapse was observed to occur by Gómez-Bombarelli et al. (2016) when using teacher forcing for training the autoregressive decoder network, and we expect it to harm performance for the style transfer task. This issue is called the strong decoder problem or "posterior collapse".

A final concern in this study is the degree to which style transfer of molecules is possible at all. As mentioned in Section 3.3, we rely on the decoder to disentangle the aspects of a molecule's structure that affect its property score from the ones that do not. Due to the interdependence between structure and properties, it is not immediately obvious to what degree this is possible. Additionally, the degree to which the model can theoretically learn this may vary significantly depending on which specific property we consider.

3.3.2 Effect of Decoder Structure

As mentioned in the previous section, prior work Gómez-Bombarelli et al. (2016) claim that teacher forcing leads to the strong decoder problem/posterior collapse and consequently destroys the learned latent representation. Some works (Dai et al., 2018; Kusner et al., 2017) have opted to use a "weaker" decoder structure, where the decoder does not accept previous characters as input. Since it does not use teacher forcing during training, its outputs are determined only by $\mathbf{y}$, $\mathbf{z}$ and its hidden state (which propagates sequential information). Sequences can be sampled from this decoder by collecting model outputs for each time-step and independently sampling characters. In light of this, we have termed it a **One-Shot** decoder structure. Such a structure

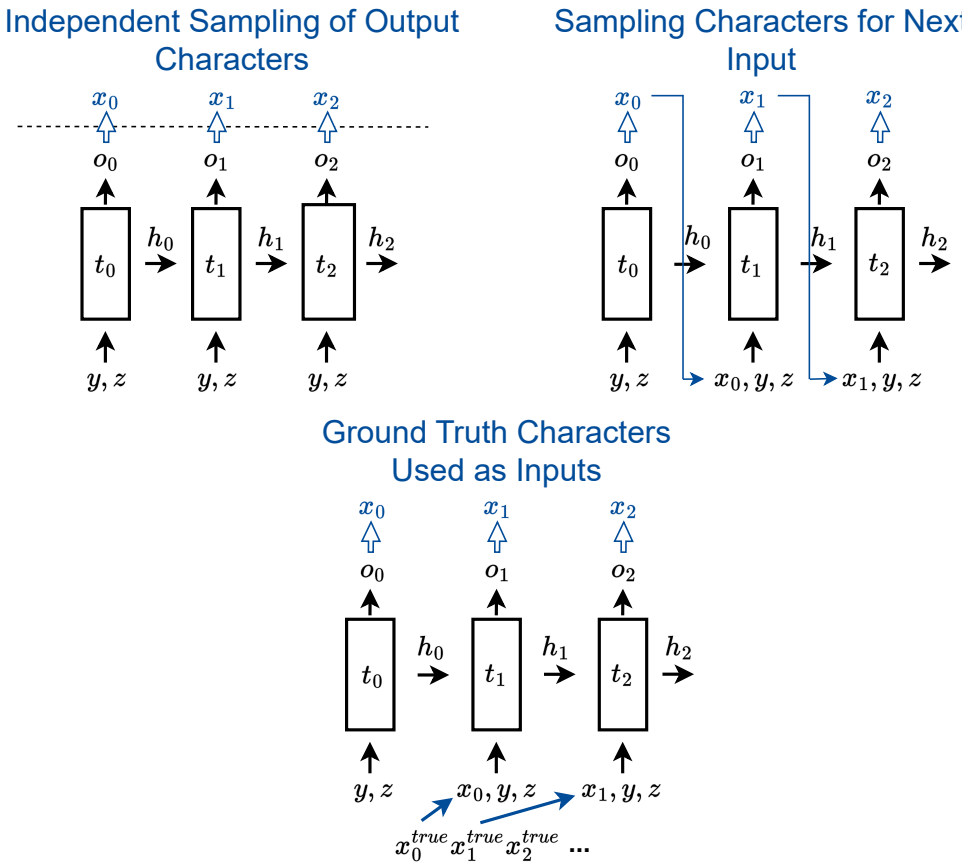

Figure 3.4: The three decoder types are one-shot (top left), explicit autoregressive (top right) and autoregressive with teacher forcing (bottom).

remedies the issue of the strong decoder, as the decoder must fully recover the structure of **x** from **z** during training. However, it can struggle to generate a high proportion of valid SMILES strings. In particular, SD-VAE and GVAE (introduced in Section 2.4.2) rely on grammar-based representations to ensure validity despite using this type of decoder.

As discussed in Chapter 4, we ran into issues when deploying our methodology using a one-shot decoder. We, therefore, considered a third option for the decoder structure we have termed an **explicit** autoregressive decoder. At each time step, the character generated at the previous step is explicitly provided as input to the decoder; however, during training, it is sampled from previous outputs instead of the input sequence (unlike with teacher forcing). This structure still allows us to condition generation on a specific molecule when using the regularizers discussed in Section 3.5, while avoiding the use of teacher forcing when computing the reconstruction objective. However, as it relies on a sampling operation between generation steps, it significantly

slows down training compared to the other two.

An overview of all three decoder architectures considered in this work is presented in Figure 3.4.

3.4 Research Objective

As discussed in Chapter 2, autoregressive models can effectively model the conditional distribution of molecules. However, they lack the flexibility of a latent space and, in particular, cannot be used for molecular optimization through style transfer. On the other hand, VAE-based latent variable models trained on SMILES can struggle to generate a high proportion of valid SMILES and respect conditioning information.

The research question of this work follows from these observations:

- *Can we combine the strengths of auto-regressive and latent variable models to enhance the conditional generation performance of VAEs when applied to SMILES strings?*

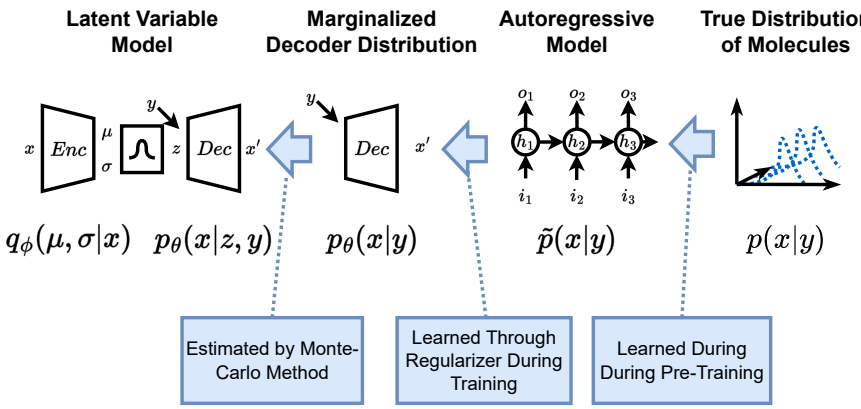

Figure 3.5: Overview of our proposed methodology. A pre-trained autoregressive model guides the generative (marginalized over z) decoder distribution.

A high-level overview of our approach can be found in Figure 3.5. We propose to use a pre-trained autoregressive model as a surrogate of the true conditional distribution of molecules. We then use this model to guide the decoder distribution marginalized over $\mathbf{z}$ (the distribution used for conditional generation) to approximate the true distribution of molecules better. The goal is to improve the proportion of valid smiles and the conditioning control during generation. We aim to achieve a high level of performance without relying on teacher forcing (to avoid the problem of the strong decoder). Additionally, we hope to be able to compete with grammar-based

methods such as GVAE and SD-VAE in terms of the validity of generated SMILES while avoiding the need for computationally intensive and complex representational modifications.

The following sections present three alternative regularizer formulations that aim to guide the decoder distribution and explain how they can be integrated into training.

3.5 Regularizer Formulation

As previously mentioned, our specific aim is to guide the marginalized (over $\mathbf{z}$) decoder distribution to better approximate the true conditional distribution of molecules $p(\mathbf{x}|\mathbf{y})$. The former can be expressed as:

$$p_\theta(\mathbf{x}|\mathbf{y}_i) = \int p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z}) p(\mathbf{z})\, d\mathbf{z} = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z})$$

However, we do not have direct access to this distribution through our model. Fortunately, since the prior distribution $p(\mathbf{z})$ is known, we can use (a sufficiently large) sample of latent points $\mathbf{z}$ to estimate it using the Monte Carlo (MC) method:

$$p_\theta(\mathbf{x}|\mathbf{y}_i) \approx \hat{p}_\theta(\mathbf{x}|\mathbf{y}_i) = \frac{1}{N} \sum_{n=1}^{N} p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z}_n) \tag{3.2}$$

We can also easily and effectively model the true conditional distribution using a surrogate autoregressive model $\tilde{p}(\mathbf{x}|\mathbf{y})$. Therefore, what remains to be determined is the exact method through which we can align $\hat{p}_\theta(x|y_i)$ and $\tilde{p}(\mathbf{x}|\mathbf{y})$. We propose to achieve this by using a regularizer that can be added as a loss component to the standard ELBO loss for VAEs. Due to interactions with the decoder structure (explored in Chapter 5), we developed three alternative formulations for the regularizer (presented in Sections 3.5.1, 3.5.2 and 3.5.3). Finally, in Section 3.6, we will present how they can be incorporated into training.

3.5.1 Calibration Regularizer

When presenting VAEs in 3.3, we examined how $q_\varphi(\mathbf{z}|\mathbf{x})$ and $p(z)$ are aligned by minimizing a loss component and estimating their KL divergence. We can apply a similar approach to aligning $\hat{p}_\theta(\mathbf{x}|\mathbf{y})$ and $\tilde{p}(\mathbf{x}|\mathbf{y})$. We define the **calibration regularizer** as follows:

$$\mathcal{R}_1(\theta) = \min_\theta D_{KL}(\hat{p}_\theta(\mathbf{x}|\mathbf{y}) || \tilde{p}(\mathbf{x}|\mathbf{y})) \tag{3.3}$$

To optimize it, we must estimate the value of the KL divergence. For a given $\mathbf{y}_i$, we can express the KL divergence in terms of the probability of each possible molecule $\mathbf{x}$ as so:

$$D_{KL}(\hat{p}_\theta(\mathbf{x}|\mathbf{y}_i) \parallel \tilde{p}(\mathbf{x}|\mathbf{y}_i)) = \sum_{\mathbf{x}} \hat{p}_\theta(\mathbf{x}|\mathbf{y}_i) \log \frac{\hat{p}_\theta(\mathbf{x}|\mathbf{y}_i)}{\tilde{p}(\mathbf{x}|\mathbf{y}_i)} \tag{3.4}$$

Computing this for every $\mathbf{x}$ is infeasible. However, we can estimate it using a sample of M molecules $(\mathbf{x}_j)_{j=1}^{M}$ (drawn from $\tilde{p}$) as:

$$\hat{D}_{KL}(\hat{p}_\theta(\mathbf{x}|\mathbf{y}_i) \parallel \tilde{p}(\mathbf{x}|\mathbf{y}_i)) = \frac{1}{M} \sum_{j=1}^{M} \log \frac{\hat{p}_\theta(\mathbf{x}_j|\mathbf{y}_i)}{\tilde{p}(\mathbf{x}_j|\mathbf{y}_i)} \tag{3.5}$$

Combining this with the MC estimation of p_θ allows us to define the following loss component (added to the ELBO):

$$\mathcal{L}(\theta, \varphi) = \mathcal{L}_{ELBO}(\theta, \varphi) + \frac{\gamma}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \hat{D}_{KL}(p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z}_j) \parallel \tilde{p}(\mathbf{x}|\mathbf{y}_i)) \tag{3.6}$$

Here γ is an added parameter that allows us to tune the strength of the regularizer loss relative to the ELBO loss during training.

3.5.2 Reward-based regularizer

As discussed in section 5.1.1, we encountered problems training the calibration regularizer with the one-shot decoder structure. To remedy these, we developed an alternate regularizer by re-framing the decoder network as an agent in a policy optimization setting. In this setting, the decoder learns a policy to generate sequences $\mathbf{x}$ that maximize a reward function. By rewarding sequences proportionately to their log-likelihood under the surrogate $\tilde{p}(\mathbf{x}|\mathbf{y})$, we can guide $p_\theta(\mathbf{x}|\mathbf{y})$ to better approximate $\tilde{p}(\mathbf{x}|\mathbf{y})$. We named this regularizer the **reward regularizer**. For a given property $\mathbf{y}_i$, the reward can be computed as:

$$\mathbb{E}_{x \sim p_\theta(\mathbf{x}|\mathbf{y}_i)} \tilde{p}(\mathbf{x}|\mathbf{y}_i) \tag{3.7}$$

The computed probability of $\mathbf{x}$ under $\tilde{p}$ can be very small, which causes issues when working with fixed-precision floating point numbers, such as

a loss of precision and underflow (when probabilities are rounded to zero). A standard way of addressing this is by considering the logarithm of the likelihood instead. Using this trick, we can reformulate the reward as:

$$\mathbb{E}_{x\sim p_\theta(\mathbf{x}|\mathbf{y}_i)} \log \tilde{p}(\mathbf{x}|\mathbf{y}_i) \tag{3.8}$$

Note that we are drawing samples $\mathbf{x}$ from the marginalized distribution $p_\theta(\mathbf{x}|\mathbf{y}_i)$. Instead, we can state this in terms of the decoder $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{y}_i, \mathbf{z})$ to get:

$$\mathbb{E}_{z\sim p(\mathbf{z})}\mathbb{E}_{x\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z})} \log \tilde{p}(\mathbf{x}|\mathbf{y}_i) \tag{3.9}$$

For a sample of properties $(\mathbf{y}_i)_{i=1}^N$ and a sample of latent points $(\mathbf{z}_j)_{j=1}^M \sim p(z)$, we can compute the reward as follows:

$$\mathcal{R}_2(\theta) = \max_\theta \sum_{i=1}^{N}\sum_{j=1}^{M} \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z}_j)} \log \tilde{p}(\mathbf{x}|\mathbf{y}_i) \tag{3.10}$$

Optimizing the Reward Regularizer: Directly optimizing this would require us to estimate the following gradient:

$$\nabla_\theta \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z}_j)} \log \tilde{p}(\mathbf{x}|\mathbf{y}_i) \tag{3.11}$$

However, this is problematic as the parameters θ affect both the distribution and the samples drawn from it. Instead, we can estimate the gradient using the score function gradient estimator (or log-gradient) trick. In general:

$$\nabla_\gamma \mathbb{E}_{x\sim g_\gamma(x)}[f(x)] = \mathbb{E}_{x\sim g_\gamma(x)}[f(x)\nabla_\gamma \log g_\gamma(x)] \tag{3.12}$$

Using this formula in our case yields gives a formula for the gradient that can be easily estimated by sampling from the decoder:

$$\nabla_\theta \mathbb{E}_{x\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z})} \log \tilde{p}(x|y_i) = \mathbb{E}_{x\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z})} \left[\log \tilde{p}(x|y_i)\nabla_\theta \log p_\theta(x|y_i, z)\right] \tag{3.13}$$

The final problem is that the reward function is framed as a maximization problem with reward values in the range $(-\inf, 0)$. However, we wish to add it as a component to the ELBO loss (which is being minimized). Therefore, we must consider its negation during implementation.

3.5.3 Alternate Reward Regularizer Formulation

The score function gradient estimator is known to have high variance and sparse rewards leading to instability during training (Mollaysa et al., 2020). We attempted to mitigate this by using large samples for the MC estimations and pre-training the decoder model before introducing the regularization. However, we still encountered issues (discussed in Section 5.1.5).

Instead, we can avoid using the score function gradient estimate altogether. Consider the integral form of the expectation from Equation 3.10:

$$\mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z})}\tilde{p}(\mathbf{x}|\mathbf{y}_i) = \int p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z})\tilde{p}(\mathbf{x}|\mathbf{y}_i)\,d\mathbf{x} \tag{3.14}$$

This can be equivalently expressed as the expected likelihood of sequences sampled from $\tilde{p}$ under p_θ:

$$\int p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z})\tilde{p}(\mathbf{x}|\mathbf{y}_i)\,d\mathbf{x} = \mathbb{E}_{\mathbf{x}\sim\tilde{p}(\mathbf{x}|\mathbf{y}_i)}p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z}) \tag{3.15}$$

We can perform this substitution to equation 3.10 to obtain the **alternate reward regularizer**:

$$\mathcal{R}_3(\theta) = \max_\theta \sum_{i=1}^{N}\sum_{j=1}^{M}\mathbb{E}_{\mathbf{x}\sim\tilde{p}(\mathbf{x}|\mathbf{y}_i)}\log p_\theta(\mathbf{x}|\mathbf{y}_i,\mathbf{z}_j) \tag{3.16}$$

In particular, we now have an expectation over samples drawn from $\tilde{p}(\mathbf{x}|\mathbf{y})$, which is independent of the decoder parameters θ. We can normally estimate this expectation and compute its gradient using the MC method, avoiding the need for the log-gradient trick. As with the previous regularizer, we used the negative of this reward to convert it from a maximization to a minimization problem.

3.6 Incorporating the Regularizers

The final training objective consists of the standard ELBO (Eq. 3.1) and one of the three regularizer formulations (either $\mathcal{R}_1(\theta)$ or the negative of $\mathcal{R}_2$ and $\mathcal{R}_3(\theta)$):

$$\mathcal{L}(\theta,\varphi) = \mathcal{L}_{ELBO}(\theta,\varphi) + \lambda\mathcal{R}(\theta) \tag{3.17}$$

The hyper-parameter λ was used to scale the magnitude of the regularizer and prevent the model from prioritizing it over the ELBO components. Note that the parameters of the surrogate model are frozen during VAE training, so that it remains a faithful approximation of $\tilde{p}(\mathbf{x}|\mathbf{y})$.

Chapter 4

Experimental Setup

In this chapter, we will describe the experiments conducted to evaluate the efficacy of the regularizers presented in Section 3.5. The following topics are covered:

- The molecular representation and molecular properties used.
- The datasets and pre-processing pipeline.
- The methodology for evaluating the models' performance and defining the performance metrics used.
- Model architecture and implementation details.
- The experiments, together with what models were trained and evaluated.

4.1 Molecule Representation and Properties

To avoid issues relating to the non-uniqueness of standard SMILES (Section 2.2), we used the canonical SMILES representation obtained with the canonicalization algorithm of the RDKit chemo-informatics package (Landrum, 2016).

Due to the limited scope of this project, we examined a simple setting with a single molecular property. We opted for the commonly used and (comparatively) simple logarithm of the octanol-water partition coefficient (LogP) property. The LogP of a molecule is a measure of the relationship between the compound's lipophilicity (fat solubility) and hydrophilicity (water solubility). A lower LogP value indicates that a compound is more water soluble and a higher more fat soluble. All LogP values were computed from canonical SMILES using the RDKit, and normalized using z-score standardization during training and inference.

4.2 Datasets and Preprocessing

Our models were trained on two datasets: Quantum Machine 9 (QM9) (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014) and the ZINC250k, a subset of the "Zinc is not Commercial" (ZINC) (Irwin et al., 2012) datasets.

4.2.1 The "Quantum Machine 9" Dataset

The QM9 dataset consists of 134.000 small organic molecules. The dataset was generated by enumerating all stable organic compounds (composed of a limited set of atoms) with nine or fewer non-hydrogen atoms. The QM9 dataset is relatively small and contains small molecules (resulting in short SMILES), allowing for faster iterations during development/debugging. Accordingly, it is used for prototyping and proof-of-concept.

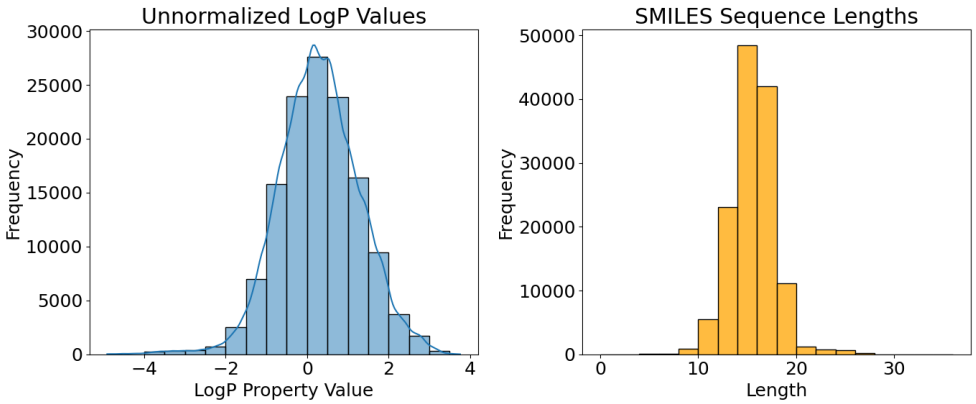

Figure 4.1: Distribution of LogP values and SMILES lengths in QM9 dataset.

Symbol	Freq.	Norm. Freq.	Symbol	Freq.	Norm. Freq.
C	767830	0.38	3	35634	0.02
1	259368	0.13	[	12523	0.01
O	177823	0.09	]	12523	0.01
2	131751	0.07	H	10568	0.01
(	118216	0.06	o	10174	0.01
)	118216	0.06	4	4952	0.00
N	98355	0.05	F	3314	0.00
=	94597	0.05	-	1974	0.00
c	78726	0.04	+	1847	0.00
n	41409	0.02	5	174	0.00
#	37027	0.02			

Table 4.1: Symbol frequencies for all canonical SMILES obtained from the QM9 dataset.

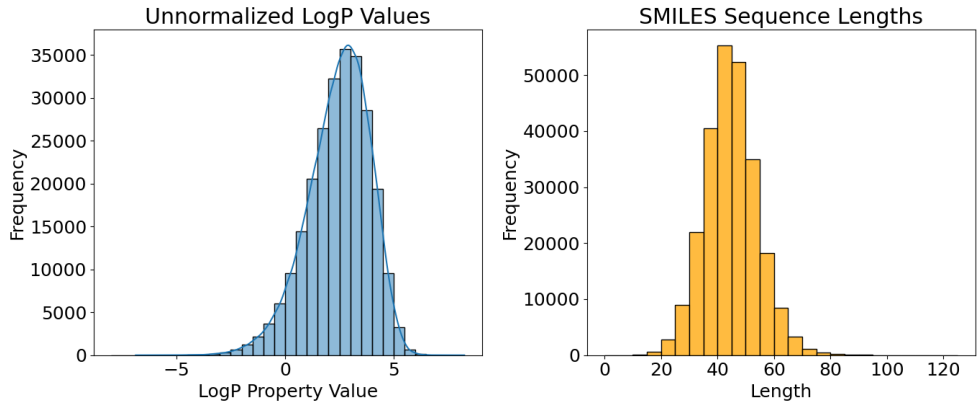

Figure 4.2: Distribution of LogP values and SMILES lengths in the ZINC250k dataset.

Pre-Processing: RDKit was used to extract the canonical SMILES representation and compute LogP values. Each SMILES character was encoded with an integer encoding scheme, equipped with START and END tokens and zero-padded to a final sequence length of 38. The distributions of LogP values and un-padded SMILES string lengths are presented in Figure 4.1. The frequencies of SMILES characters are presented in Table 4.1.

4.2.2 The "ZINC is not Commercial" Dataset

The simplicity of molecules in QM9, their unrealistic structure, and their synthetic nature limit its usefulness as an analogue for real-world conditions. Additionally, we found that autoregressive models trained on the dataset had trouble generating novel molecules, which does not occur in more complex settings (Mollaysa et al., 2020).

Given these drawbacks, we repeated our experiments on a larger, more realistic dataset. To maintain comparability with prior work (Gómez-Bombarelli et al., 2016; Mollaysa et al., 2019; Dai et al., 2018), we opted for the ZINC-250k dataset. This dataset contains a subset of 250.000 molecules from the "ZINC is not Commercial" dataset. The compounds in ZINC are larger and more complex than in QM9. Additionally, they are all commercially available and drug-like compounds.

Pre-Processing: The ZINC250k compounds were pre-processed in the same manner as for QM9. Sequences were padded to a final length of 120 characters. The distributions of sequence lengths, property values and SMILES symbols for the dataset are provided in Figure 4.2 and Table 4.2.

Char.	Freq.	Norm. Freq.	Char.	Freq.	Norm. Freq.
c	2208591	0.20	S	63791	0.01
C	2089152	0.19	-	63474	0.01
(	963303	0.09	l	42872	0.00
)	963303	0.09	s	39088	0.00
1	724766	0.07	o	30987	0.00
O	545244	0.05	4	30588	0.00
2	471857	0.04	/	28557	0.00
N	437084	0.04	#	15587	0.00
=	395693	0.04	r	12722	0.00
n	268119	0.02	B	12722	0.00
3	166892	0.02	\	5839	0.00
[	344325	0.03	5	2592	0.00
]	344325	0.03	I	888	0.00
@	335552	0.03	P	127	0.00
H	290126	0.03	6	88	0.00
F	79430	0.01	7	8	0.00
+	76813	0.01	8	2	0.00

Table 4.2: Symbol frequencies for all canonical SMILES obtained from the ZINC250k dataset.

4.3 Evaluation

We evaluated the performance of VAE-based models for the conditional generation, molecular optimization and reconstruction tasks. Since both conditional generation and molecular optimization are multi-faceted problems, we computed several metrics for each that we will present in Sections 4.3.1 and 4.3.2.

Models' reconstruction accuracy was computed by probabilistically encoding 1000 test set molecules. Each was decoded five times with their original LogP values as the conditional information. The final reconstruction performance was evaluated as the proportion of perfectly reconstructed molecules.

4.3.1 Conditional Generation Performance Metrics

The major use-case for molecule design methods is drug design (Section 2.1). In this setting, we wish to generate a set of candidate molecules for downstream screening. Therefore, the aim is to generate a diverse set of novel (previously unknown) candidate molecules. Additionally, we want these to be chemically valid (i.e. valid SMILES) and to respect the provided conditioning information. A good model needs to balance all of these requirements, and a model incapable of generating novel compounds is as

Metric	Abbreviation	Description
Validity	Valid ↑	*Proportion of generated SMILES sequences that correspond to valid molecules. We considered SMILES that can be be successfully parsed by as valid.*
Uniqueness	Uniq ↑	*The number of unique (canonical forms of) valid SMILES as a proportion of all generated valid SMILES.*
Novelty	Novel ↑	*The proportion of valid SMILES whose canonical form absent in the training set.*
Property Error	MAE ↓	*The mean absolute error (MAE) between the target property y and the property y' of each corresponding generated molecule x' among valid generated SMILES.*

Table 4.3: Details of performance metrics computation for conditional generation.

Dataset	QM9	ZINC250k
Value	1.1213	1.6067

Table 4.4: Property error upper bounds values.

useless as one that cannot generate valid molecules. Comparative evaluation of two models requires us to consider the trade-off between these different aspects of performance, and one model cannot be definitively said to be better than another unless it is Pareto dominant.

To facilitate a multi-faceted evaluation of models' conditional generation performance, we selected a subset of the distribution learning performance metrics from the GuacaMol (Brown et al., 2019) drug design benchmark. We also evaluated the conditional performance, with the error between the target and the achieved LogP scores. Details on the computation of each conditional generation metric are presented in Table 4.3. All metrics were computed based on a set of $1,000$ SMILES, sampled $x \sim p_\theta(y, z)$, for $z \sim p(z)$ and distinct y values from the test set.

Interpreting Property Errors: The magnitudes of property errors depend on the model and the distribution of property values in the dataset. We computed a data-informed upper bound for the property error by computing the mean error between randomly sampled pairs of properties. The results for both datasets are presented in Table 4.4.

Metric	Abbreviation	Description
Style Tr. Validity	Valid ↑	*The proportion of valid generated SMILES.*
Style Tr. Failure	Fail% ↓	*The proportion of generated SMILES who's canonical form is equivalent to the source molecule.*
Structural Similarity Between Source and the Generated	Tan ↑	*Computed as the average Tanimoto similarity between the Morgan fingerprints (computed with radius 3) of the source and the generated molecules.*
Property Error Between Target and Generation	MAE ↓	*Computed as the mean absolute error between the target property and the generated molecule's property*

Table 4.5: Details of performance metrics for molecular optimization.

4.3.2 Molecular Optimization Performance Metrics

In Section 3.3, we presented how VAE-based latent variable models can be used for molecular optimization through style transfer. Recall that our goal when performing style transfer is to endow a source molecule $\mathbf{x}$ with target properties $\mathbf{y}$, resulting in a newly generated molecule $\mathbf{x}'$ that retains (as much as possible) the structure of $\mathbf{x}$. The success of this process can be evaluated by measuring the structural similarity to the source molecule and the error between the generated and target properties. Since this process may still result in invalid SMILES, we also measured the validity of generated sequences. Finally, as discussed in 3.3.1, the model may fail to attend to $\mathbf{y}$ during reconstruction. In this case, the source molecule will remain unaltered, and style transfer will fail. We also measured the proportion of such cases to gauge the likelihood of this effect. Details on the metrics used to measure these performance aspects are presented in Table 4.5.

Molecular optimization metrics were computed based on a set of $2,500$ SMILES. These were obtained by probabilistically encoding $2,500$ test set molecules $\mathbf{x}$ to obtain latent points $\mathbf{z}$. We then independently sampled a set of test set target properties $\mathbf{y}$ and decoded them to obtain the final generated molecules $\mathbf{x}' \sim p_\theta(x|y,z)$.

Interpreting Structural Similarity: The Tanimoto similarity between Morgan fingerprints (presented in Section 2.2.1) is a common way to measure structural similarity between two molecular graphs. The Tanimoto similarity is the proportion of shared features (i.e. the binary presence of structures constituting the Morgan fingerprint) common to both molecules. The values of the Tanimoto similarity range from 0 to 1, with higher values indicating a

higher structural similarity.

The expected absolute magnitude of structural similarity scores differs depending on the length of the molecules being considered. The effect of this can be seen in the examples of similar molecules obtained during style transfer on the QM9 dataset (Section 5.1.3). Even a small change to a molecule can result in a molecule with structural similarity to the original much smaller than 1. Additionally, interpreting the degree to which the structure of a molecule has been preserved during style transfer requires a notion of what the structural similarity would be if none of the structure was preserved. We computed two lower bounds for the structural similarity for each set of style transferred molecules, the definitions of which are presented in Table 4.6.

Metric	Abbreviation	Description
Tanimoto Baseline 1	Tan1	*Similarity between generated molecules and a complimentary set decoded with with random latent points* $z \sim p(z)$.
Tanimoto Baseline 2	Tan2	*Similarity between generated molecules and a complimentary set generated from random encoding or independently sampled molecules* $z \sim q_\varphi(z\|x')$.

Table 4.6: Baseline values for structural similarity.

4.4 Models and Experiments

4.4.1 Surrogate Models

Dataset	Validity	Uniqueness	Novelty	Property MAE
QM9	0.9700	1.0000	0.3024	0.3210
ZINC	0.9333	1.0000	0.9964	0.1714

Table 4.7: Performance of surrogate models.

4.5 Models and Experiments

4.5.1 Surrogate Models

We used the same auto-regressive model for both the QM9 and ZINC250k settings. Both were based on a stacked long-short term memory cell (LSTM) architecture (Hochreiter and Schmidhuber, 1997) and achieved targeted generation by concatenating the conditional (property) information to the

input at each time step. The conditional generation performance of both models is provided in Table 4.7. Both models managed to accurately model the true conditional distribution $p(\mathbf{x}|\mathbf{y})$ and were able to generate a high proportion of valid SMILES with LogP close to the targets. The surrogate model trained on the QM9 dataset exhibits a low novelty score; however, this does not carry over to the more realistic ZINC250k setting.

4.5.2 Model Architecture

Surrogate Model Architecture: The surrogate models were trained using teacher forcing and a maximum likelihood objective until convergence. Both models consisted of an initial embedding layer (47x512), three stacked LSTM layers with input feature size 513 and output feature size 512, and a final linear layer with input size 512 and output size 47 (the number of characters in the SMILES language). LSTM layers were trained with a dropout probability of 0.2 and were initialized using a Xavier uniform initialization strategy.

Latent Variable Model Architecture: We based our model architecture on the architecture used in (Mollaysa et al., 2019). The dimensionality of the latent space (LS) was set to 159 for QM9 and 196 for ZINC. Additionally, the QM9 models' decoder layers were implemented as RNNs, while the ZINC models' were implemented as GRUs. The rest of the model architecture was fixed for both datasets. The VAE model encoder consisted of an initial embedding layer followed by three 1 dimensional convolutional layers of dimensions 47x9x9, 9x9x9 and 9x10x11, each followed by a ReLu activation function. These fed into two linear layers, one for predicting the embedding mean (435xLS) and one for the log variance (435xLS). The state decoder consisted of an initial embedding layer (47xLS) followed by three stacked layers (113x501, 501x501, 501x501) each followed by a final output layer (501x47). The models were trained with gradient clipping above set to magnitude 0.2 and a learning rate scheduler set to reduce the learning rate on plateaus to a minimum of 1e-6.

Parameters: The ELBO loss' KL divergence component was scaled according to a logistic schedule starting at 0.01 and ending at 1.0, centered at epoch 29, in line with the approach used in Gómez-Bombarelli et al. (2016). The weight of the γ parameter for each regularizer was determined by inspection and a manual search of the parameter space, focusing on keeping the regularizer loss component lower than the ELBO components. We settled on a γ of 0.1 for the calibration regularizer and 0.01 for both reward regularizers for the final results. The remaining regularizer parameters relating to the Monte-Carlo estimation strategies were determined to maximize sample size without exceeding available GPU memory. An overview of the training parameters is presented in Table 4.8.

Parameter	Value
QM9 Latent Dim	156
ZINC Latent Dim	192
Optimizer	Adam
Batch Size	128
Learning Rate	1e-4
ELBO KL	0.001 to 1.0
Calib. γ	0.1
Rew. 1 γ	0.01
Rew. 1 MC Sample Size for Latent Points	4
Rew. 1 MC Sample Size for Decoder	20
Rew. 2 γ	0.01
Rew. 2 MC Sample Size	60

Table 4.8: Training parameters for VAE models.

Training Parameters: For each dataset, we created validation and test sets by selecting $10,000$ random SMILES-property pairs. Models were trained for a maximum of 500 epochs with early stopping after 50 consecutive epochs with no improvement in the total loss on the validation set. All performance metrics were computed using properties and input instances from the test set.

Environment Details: Models were implemented using the PyTorch library (version 2.2.1) and trained on an NVIDIA GeForce RTX 4090 graphics card using CUDA version 12.3.

Code Availability: The codebase for this project has been made available on GitHub.

4.6 Model Versions

We will present a detailed discussion of our experiments and the success/-failure of the regularizers in the following chapter. For the QM9 dataset, we trained one model for each combination of decoder structure (one-shot, explicit and teacher forcing) and regularizer (no regularizer, calibration, initial and alternate reward regularizer). As discussed in Chapter 5, not every combination of regularizer and decoder structure was successful, with the regularized one-shot models and models trained with the initial version of the policy regularizer exhibiting pathological behaviours. For the remaining models, we trained an additional version on the ZINC250k dataset. In total, we examined the performance of 12 models for QM9 and seven models for ZINC250k.

Chapter 5

Results & Discussion

This chapter discusses the experimental results of our study. The results for the QM9 dataset are detailed in Section 5.1.1, and for the ZINC250k dataset, they are in Section 5.2. Samples of the generated molecules are provided in Appendix A for further reference.

Terminology: For ease of reference, we will use abbreviations to denote the different versions of our models in tables and figures. Each abbreviation comprises two parts: the first part identifies the decoder structure: **cVAE** for one-shot, **EXP** for explicit autoregressive, and **TF** for autoregressive with teacher forcing. The second part indicates the type of regularizer used during training, if any. **KLD** denotes the calibration regularizer, **Pol1** the policy regularizer, and **Pol2** the alternate policy regularizer. For example, the model cVAE-KLD indicates a one-shot decoder model trained with the calibration regularizer, while TF denotes a model with the teacher forcing decoder and no regularizer.

5.1 Model Performance on the QM9 Dataset

The performance of models trained on the QM9 dataset is presented in Tables 5.1 - 5.4. The following sections describe the performance of models trained with the one-shot decoder (Section 5.1.1), the conditional generation performance (Section 5.1.2), and style transfer performance (5.1.3) of models trained with both autoregressive decoder structures. Finally, they discuss the effect of teacher forcing (Section 5.1.4) and issues with the initial reward regularizer (Section 5.1.5).

5.1.1 Performance of One-Shot Decoder Models

The conditional generation performance of one-shot decoder models is presented in Table 5.1. We see that the baseline model achieves a low validity

Model	Recon ↑	Valid ↑	Uniq ↑	Novelty ↑	Prop. MAE ↓
cVAE	0.8872	0.2210	1.0000	0.8959	0.7976
cVAE-KLD	0.8796	0.0110	1.0000	0.9090	0.7919
cVAE-Pol1	0.9436	1.0000	0.0010	0.0000	1.0121
cVAE-Pol2	0.8769	0.0050	1.0000	1.0000	1.4675

Table 5.1: Conditional generation performance of one-shot decoder and surrogate models.

Model	Valid ↑	Fail% ↓	MAE ↓	Tan ↑	Tan-B1	Tan-B2
cVAE	0.9260	0.7482	0.8638	0.4194	0.0622	0.0656
cVAE-KLD	0.9072	0.7579	0.8395	0.3894	0.0490	0.0677
cVAE-Pol1	0.9468	0.8906	0.9909	0.4451	0.0296	0.0652
cVAE-Pol2	0.9148	0.7420	0.8497	0.4111	0.0536	0.0659

Table 5.2: Style transfer performance of one-shot decoder models.

relative to the teacher forcing baseline (Table 5.6). Although our models performed better in absolute terms, this relative gap is consistent with the observations of Gómez-Bombarelli et al. (2016).

However, the introduction of all three regularizers not only failed to improve performance but also resulted in pathological model behaviour. As mentioned in Section 3.4, this issue is what initially motivated our use of the explicit decoder architecture.

We hypothesized why this is the case based on the models' behaviour. Recall that the one-shot decoder fully determines what characters to generate based on $\mathbf{z}$ and $\mathbf{y}$. Suppose we wish to compute the probabilities of a molecule $\mathbf{x}^1$ under the decoder and surrogate. Under the surrogate, we can simply examine the probability of each character $\mathbf{x}_t^1$ under $\tilde{p}(\mathbf{x}_t|\mathbf{x}_{t-1}^1, \mathbf{y}^{t-1})$. However, computing the probability under the decoder risks that the combination of some $\mathbf{z}^2$ and $\mathbf{y}^1$ corresponds to a different molecule ($\mathbf{x}^2$) than $\mathbf{x}^1$. In this case, the regularizer will have a destructive effect on the decoder, instead of a positive one.

Furthermore, this effect should be magnified early in training when the decoder has not sufficiently learned to generate meaningful sequences. Note that in the case of the one-shot decoder models, even when fully trained, the decoder's performance is low. As we will see in the following sections, switching to an explicit autoregressive decoder structure $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}, \mathbf{z})$ fixes this issue for the KLD and Pol2 regularizers. In this case, we use the probabilities $p_\theta(\mathbf{x}_t^1|\mathbf{z}^2, \mathbf{y}^1, \mathbf{x}_t^1)$. Although still incorrect, the additional qualification offered by the input tokens means that only these conditional probabilities are being "updated" and during inference, sampling $p_\theta(\mathbf{x}_t|\mathbf{z}^2, \mathbf{y}^1, \mathbf{x}_t^2)$ will not be affected.

Disregarding this explanation, we see that the failure behaviour for the

Pol1 regularizer is different from that of the other two, as instead of only generating invalid molecules, it experiences mode collapse, generating a low number of distinct molecules. As we will see in subsequent sections, the Pol1 regularizer fails regardless of the decoder structure used. A discussion of why this may happen is presented in Section 5.1.5.

The one-shot decoder models perform better in terms of style transfer, managing to generate a high proportion of valid molecules. However, we see that the MAE is only slightly decreased from the property error upper bound of 1.1213 (Table 4.4) implying weak control over the molecular property. Additionally, they have a high rate of style transfer failure (when the model reproduces the source molecule). Together, these observations demonstrate the first issue discussed in Section 3.3.1, where a weak decoder model is over-reliant on the structural information $\mathbf{z}$ relative to the conditioning information $\mathbf{y}$.

5.1.2 Conditional Generation Performance with Autoregressive Decoder Models

The performance of models trained with explicit decoders is presented in Tables 5.6 and 5.4, and teacher forcing decoders in Tables 5.3 and 5.5. Unlike in the one-shot case, both the Pol2 and KLD regularizers have worked well. In particular, comparing their conditional generation performance (Tables 5.3 and 5.6) to the surrogate model's performance (Table 4.7) shows that we have successfully aligned the two. In particular, this has allowed us to greatly improve the validity and property error of both the explicit and teacher forcing models. However, it has also resulted in our models' aligning with the poor novelty of the surrogate. As explained in Section 4.5.1, this is an idiosyncrasy of the QM9 dataset, and does not carry over to the more realistic ZINC setting (Table 5.7). Although both regularizers achieve a similar positive effect, the Pol2 regularizer consistently outperforms the KLD regularizer across decoder structures and datasets.

5.1.3 Style Transfer Performance with Autoregressive Decoder

In terms of style transfer performance, we see that in both cases (Tables 5.5 and 5.4) the property error and probability of successfully changing the source molecule have improved. This is consistent with a greater focus of the decoder on $\mathbf{y}$ relative to $\mathbf{z}$ and is a result of the regularizer guiding the marginalized distribution $p_\theta(x|y)$ (disregarding z). An unfavourable consequence of this shift in focus is a decreased similarity to the source molecule (as the source molecule structure is encoded in z). In the explicit decoder case, this decrease is, on average, 0.055 and in the teacher forcing case 0.0791. This is a relatively small difference, especially considering that in the QM9 setting, changes to a single atom or bond can drastically affect

Model	Recon ↑	Valid ↑	Uniq ↑	Novelty ↑	Prop MAE ↓
EXP	0.9509	0.2860	1.0000	0.8287	0.7058
EXP-KLD	0.6783	0.7460	1.0000	0.5161	0.2831
EXP-Pol1	0.9363	0.0010	1.0000	1.0000	0.1987
EXP-Pol2	0.6711	0.8250	0.9976	0.4642	0.2300

Table 5.3: Conditional generation performance of explicit decoder models.

Model	Valid ↑	Fail % ↓	MAE ↓	Tan ↑	Tan B1	Tan B2
TF	0.9176	0.3391	0.3955	0.3071	0.0667	0.0668
TF-KLD	0.9668	0.0852	0.1493	0.2328	0.0651	0.0658
TF-Pol1	0.4312	0.6790	0.2935	0.3614	0.0532	0.0652
TF-Pol2	0.9656	0.0849	0.1486	0.2280	0.0652	0.0651
Lim	-	0.0210	0.3439	0.1424	0.0816	0.1145

Table 5.4: Style transfer performance of teacher forcing decoder and baseline models.

Model	Valid ↑	Fail% ↓	MAE ↓	Tan ↑	Tan-B1	Tan-B2
EXP	0.8968	0.7243	0.7514	0.3711	0.0653	0.0646
EXP-KLD	0.8260	0.4459	0.5377	0.3507	0.0680	0.0667
EXP-Pol1	0.8888	0.7115	1.5314	0.3206	0.0209	0.0666
EXP-Pol2	0.8404	0.4131	0.5493	0.3481	0.0653	0.0662

Table 5.5: Style transfer performance of explicit decoder models.

Model	Recon ↑	Valid ↑	Uniq ↑	Novelty ↑	Prop MAE ↓
TF	0.9481	0.8340	1.0000	0.7014	0.3636
TF-KLD	0.6857	0.9580	0.9958	0.3862	0.1292
TF-Pol1	0.9575	0.0010	1.0000	0.0000	0.0816
TF-Pol2	0.6799	0.9710	0.9938	0.3326	0.1259
Lim	0.4381	0.5700	0.9863	0.9520	0.4510
GB	0.0361	0.1030	-	0.9000	-

Table 5.6: Conditional generation performance of teacher forcing decoder and baseline models.

the Tanimoto similarity (this can be seen nicely in Figure 5.1). Additionally, the decreased similarity is still significantly higher than both Tanimoto lower bound estimates, indicating that significant structural information is retained from the source molecule. We conclude that, although the regularizer does not improve style transfer across the board, it creates a desirable trade-off, since it also greatly improves conditional generation performance.

Figure 5.1 shows five examples of successful style transfer for a model trained with the calibration regularizer and explicit decoder (without teacher forcing). The model can find relatively simple and effective solutions to alter the LogP value. Additionally, we see that due to the short length of molecules in the

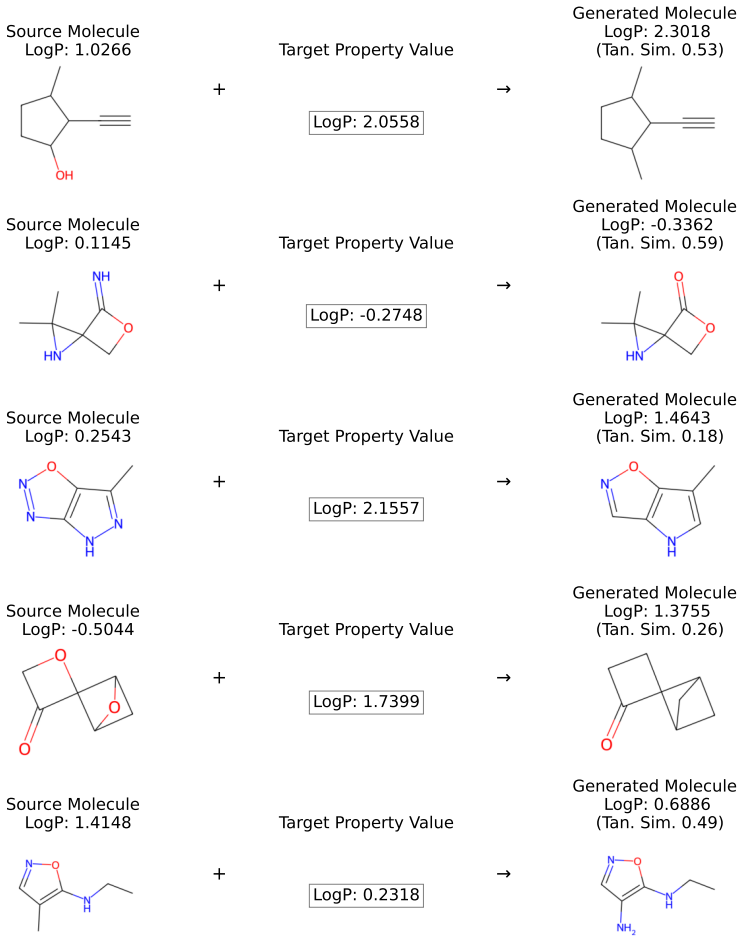

Figure 5.1: Examples of successful style transfer with TF-KLD model.

QM9 dataset, even relatively simple modifications lead to a relatively low structural similarity.

To further examine the flexibility/practicality of using the models for style transfer, we also examined their ability to edit a molecular structure for a range of target property scores. To show this, we attempted to change molecules' properties for various targets. The results are plotted in Figures 5.2-5.7, with the target property on the horizontal axis and the property of the transformed molecule on the vertical axis. For each model, we examined its ability to change the properties of the molecules with the lowest (left), median (center) and highest (right) LogP scores in the test set. These results are presented for the explicit decoder models with no regularizer (Figure 5.2, calibration regularizer (Figure 5.3) and alternate reward regularizer (Figure 5.4). Note that an ideal model would be able to perfectly match the target,

resulting in points along the diagonal. The introduction of the regularizers (bottom two plots) is significantly closer to this. In particular, they are more flexible and can find more distinct solution modes (these are visible in the horizontal striations). The uneven structure of these plots demonstrates the dependence on the ability of a model to perform style transfer on both the source molecule and target property. For example, the right hand side molecule appears to be easier to edit for all the models, and models generally struggle more when the target is far from their original property. Overall, the plots demonstrate that style transfer of molecules is possible in this setting.

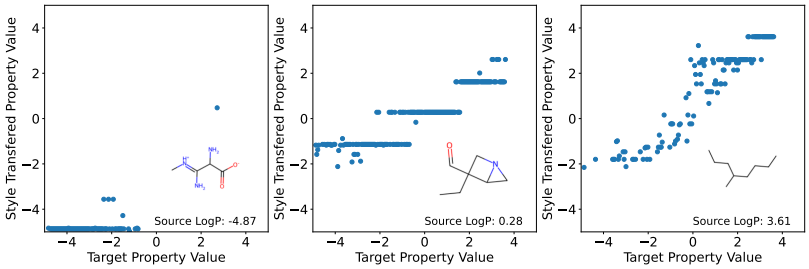

Figure 5.2: Style transfer generation property scores for a range of target properties sampled from the base EXP model.

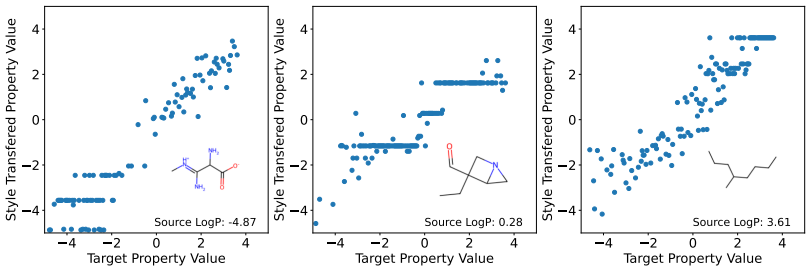

Figure 5.3: Style transfer generation property scores for a range of target properties sampled from the EXP-KLD model.

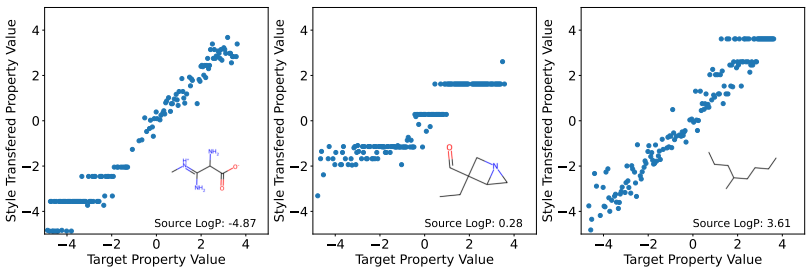

Figure 5.4: Style transfer generation property scores for a range of target properties sampled from the EXP-Pol2 model.

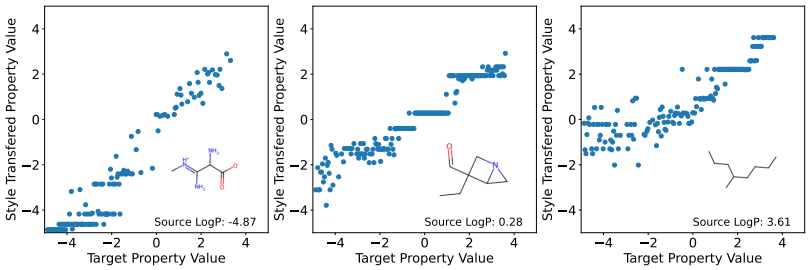

Figure 5.5: Style transfer generation property scores for a range of target properties sampled from the base TF model.

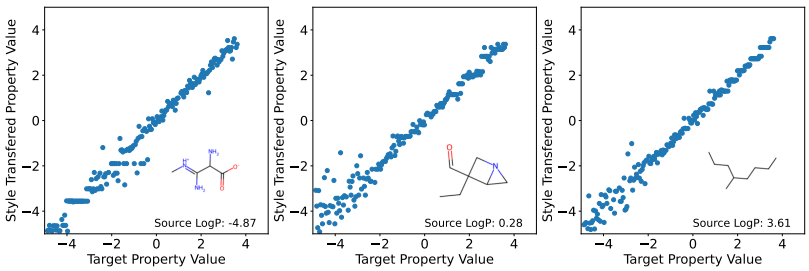

Figure 5.6: Style transfer generation property scores for a range of target properties sampled from the TF-KLD model.

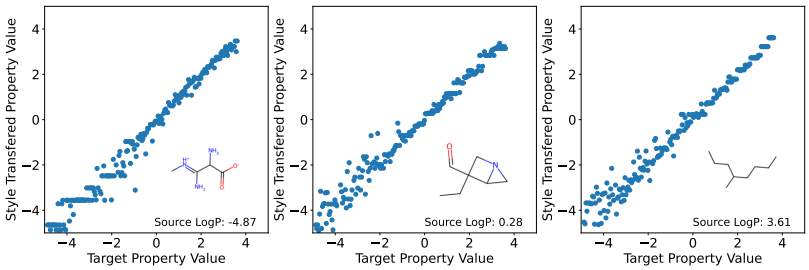

Figure 5.7: Style transfer generation property scores for a range of target properties sampled from the TF-Pol2 model.

5.1.4 Effect of Introduction of Teacher Forcing

The effect of the KLD and Pol2 regularizers is similar for the explicit and teacher forcing decoders. However, the absolute level of model performance differs significantly, and, therefore, deserves explicit comparison. In terms of conditional generation, teacher forcing significantly improves the validity and property error of molecules. In the QM9 case (Table 5.6), the improved baseline performance diminishes the marginal improvement introduced by the regularizer. However, as we will see in the ZINC250k results (Table 5.7), in the more complex setting, teacher forcing alone cannot achieve a high validity.

Expectedly, teacher forcing increases the decoder's focus on $\mathbf{y}$; In Table 5.4, it decreases Tanimoto similarity and improves property error. However the prevailing assumption in the literature (Gómez-Bombarelli et al., 2016) is that it destroys the latent space completely. Our findings contradict this, as we are still able to use the learned $\mathbf{z}$ to perform style transfer. In particular, the structural similarity is still much higher than both baselines, implying that a significant portion of the source structure is preserved. We conclude that we are not observing complete posterior collapse. We additionally confirmed this by examining the posterior empirical posterior distribution of the decoder. In the case of complete posterior collapse, sequences are typically encoded tightly around the origin, an effect not present in our case. This finding remains consistent in the ZINC250k setting (Section 5.2).

Notably, if we compare the baseline teacher forcing model with either EXP-cVAE-KLD or EXP-cVAE-Pol2, the latter achieve higher validity, lower property error, and higher similarity to the source molecule. In this case using a regularizer is a better way to improve conditional generation performance and maintain a useful latent space than using teacher forcing.

We repeated the experiment with style transfer for a range of targets with the teacher forcing decoder models (Figures 5.5, 5.6 and 5.7)). The base model performs much better, likely due to the increased focus on property conditioning information caused by teacher forcing. However, both regularizers still improve the range of achieved properties, diversity of solution modes and conditioning success.

5.1.5 Performance of Initial Reward Regularizer

We can see in Tables 5.1, 5.3 and 5.6 that the initial version of the reward regularizer destroys conditional generation performance regardless of the decoder structure. As mentioned in Section 3.5.3, a potential cause of these difficulties is the score function gradient estimate, which can be noisy and unstable (Mollaysa et al., 2020). This issue is what initially motivated the examination of the alternate reward regularizer. Another potential contributing factor is that the computation of the regularizer requires the probabilities of sequences sampled from the decoder. If the decoder cannot produce realistic sequences, both models compute probabilities for unusual sequences, leading to unexpected behaviour. In contrast, the Pol2 regularizer samples sequences from the surrogate model, almost always correspond to realistic molecules (even early during training).

We examined several strategies in an attempt to address this issue, including scheduling of the regularizer loss magnitude, pre-training of the decoder model, increasing the sample sizes and using latent points from the encoder posterior. However, none of these successfully addressed this issue.

5.2 Model Performance on the ZINC250k Dataset

The results for the retrained versions of the base models and all working decoder-regularizer configurations are presented in Tables 5.7 and 5.8. We see that the novelty issue of the surrogate trained on QM9 is not present for ZINC250k (Table 4.7), and that, as predicted, the introduction of either regularizer (Table 5.7) does not decrease novelty. Additionally, the one-shot decoder model struggled to generate a non-trivial proportion of valid SMILES. This result aligns with the experience of Gómez-Bombarelli et al. (2016) and is the motivation they originally cite for using teacher forcing.

Model	Recon ↑	Valid ↑	Uniq ↑	Novelty ↑	Prop MAE ↓
cVAE	0.1852	0.0010	1.0000	1.0000	1.0228
EXP	0.4301	0.0020	1.0000	1.0000	1.5033
EXP-KLD	0.1716	0.855	1.0000	1.0000	0.3690
EXP-Pol2	0.4321	0.936	1.0000	1.0000	0.2065
TF	0.5832	0.5680	1.0000	1.0000	0.6600
TF-KLD	0.3439	0.9110	1.0000	0.9989	0.3283
TF-Pol2	0.5369	0.9380	1.0000	0.9989	0.1998

Table 5.7: Conditional generation performance of ZINC250k models.

Model	Valid ↑	Fail% ↓	MAE ↓	Tan ↑	Tan-B1	Tan-B2
cVAE	0.3944	0.3765	1.5439	0.5755	0.0921	0.1186
EXP	0.6492	0.5699	1.3499	0.6227	0.07410	0.1164
EXP-KLD	0.4308	0.2953	1.3105	0.5700	0.1125	0.1149
EXP-Pol2	0.6692	0.5642	1.2572	0.61154	0.1138	0.1164
TF	0.8448	0.2656	0.7114	0.4311	0.1013	0.1136
TF-KLD	0.8456	0.09413	0.4119	0.4298	0.1129	0.1115
TF-Pol2	0.8724	0.1444	0.3759	0.4214	0.1145	0.1142

Table 5.8: Style transfer performance of ZINC250k models.

Our observations from the QM9 dataset carry over to the ZINC scenario, with conditional generation performance (Table 5.7) improving both validity and property error when either regularizer is introduced. These changes are particularly dramatic in the case of the explicit decoder, where the base model performs poorly.

The regularizers' effect on style transfer performance (Table 5.8) is similar in nature to the QM9 case. Introducing either form of the regularizer improves the target-generation MAE and decreases the proportion of style transfer failures. Additionally, it slightly decreases the source-generated structural similarity in the explicit case. As with QM9, the second reward regularizer appears to have a stronger effect on both tasks except for the proportion of

style transfer failures, which is surprisingly higher.

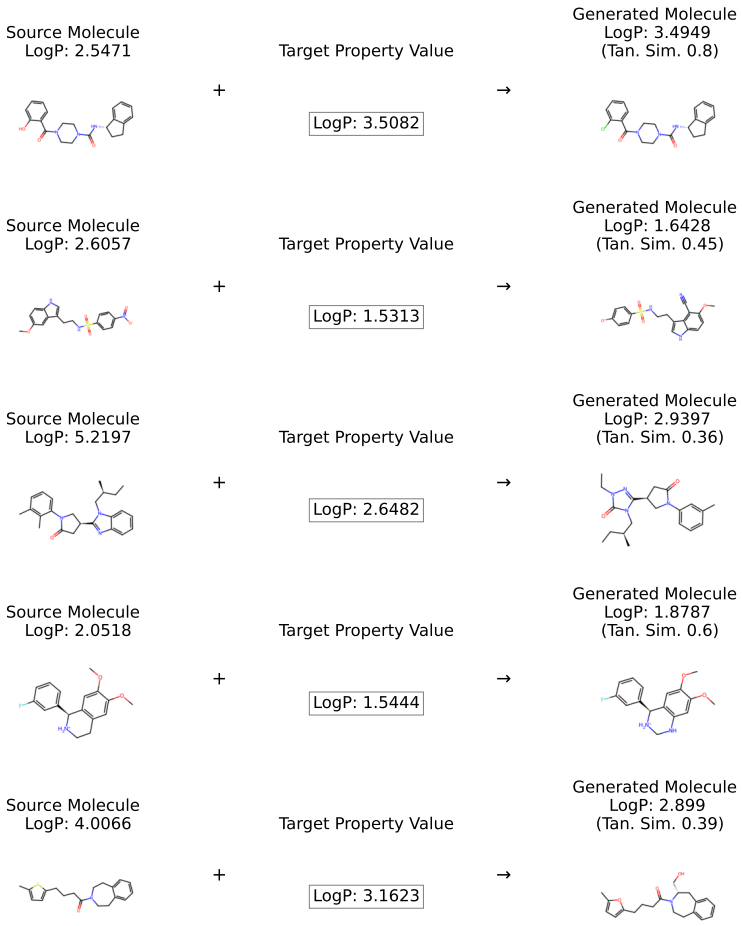

Figure 5.8: Examples of successful style transfer with TF-Pol2 model trained on ZINC250k.

Although the directionality/nature of the performance changes introduced by the regularizers is the same as for QM9, the absolute magnitudes of the performance metrics differ significantly. This leads to a different trade-off when considering whether to employ regularization. In contrast to the QM9 case, the base teacher forcing decoder model is less effective, only achieving a validity of 0.568 compared to 0.938 when trained with the Pol2 regularizer. Additionally, in all three explicit decoder cases, the property error is very high compared to the baseline value from Table 4.4. Even with the regularizer, the model is still overemphasizing $\mathbf{z}$ for reconstruction. To achieve a good degree of conditional control, both teacher forcing and the use of a regularizer are apparently necessary. An sample of successful style transfer attempts for the TF-Pol2 model is provided in Figure 5.8.

5.3 Comparison with Previous Work

We compared our work with cVAEs trained for SMILES strings from the literature. The work by (Gómez-Bombarelli et al., 2016) (denoted GB in the tables) and the model by Lim et al. (2018) (denoted Lim in the tables). The latter is significant as the authors also used it for style transfer of molecules, although they used a different terminology in their paper. As the latter paper's evaluation methodology was incompatible with ours, we recomputed their model's performance according to our metrics. Both employ teacher forcing and are included in Tables 5.6 and 5.4.

We see that our baseline model performs significantly higher than both. This can be partially explained by differences in the experimental setup and architecture. Our model features the stronger encoder from Mollaysa et al. (2019) and a stronger form of conditioning where conditional information is prepended to the input for each layer of the stacked decoder model, instead of only the last as in Gómez-Bombarelli et al. (2016). A larger factor is likely the training parameters described in Section 4.5.2, as we train our models for a large number of epochs (500) with relatively lax early stopping conditions (50 epochs without improvement).

Another separate axis of comparison is with other approaches that address the performance issues of molecular VAEs. In particular, we discussed models that utilise grammar-based representations to increase validity in Section 2.4.2. SD-VAE Dai et al. (2018) can guarantee validity by making it impossible for the model to generate invalid SMILES strings. Even in the ZINC250k setting, our model can achieve comparable performance, without relying on the slow and complex representation used by SD-VAE (93.8% when using the Pol2 regularizer). This also places the performance of our model significantly above that of SD-VAE's predecessor GVAE Kusner et al. (2017), which only achieves a validity of 7.2%. Although differences in experimental settings and model architecture complicate this comparison, these results still demonstrate that our approach is a promising alternative.

Chapter 6

Conclusions and Further Work

6.1 Conclusions

Before drawing any final conclusions, we will reiterate the goals of this study (Section 3.4):

- *Achieve a high level of conditional generation performance without relying on teacher forcing (to avoid the problem of the strong decoder).*
- *Compete with grammar-based methods while avoiding the need for computationally intensive and complex representational modifications.*

Our results showed that the proposed regularizer can significantly improve the validity and property conditioning performance of molecular VAE models (Section 5.1.2). Additionally, with the explicit decoder structure, we were able to achieve this without the use of style-transfer. Moreover, although using an explicit decoder means that we no longer have a clear computational advantage over grammar-based methods, we managed to maintain the simplicity of our representation.

Furthermore, the assumption underpinning our goals was that teacher forcing would destroy the latent space and, therefore, prevent us from performing molecular optimization. However, our results (Section 5.1.4) show that, in our case, this effect, although still present, appears to be mild. More importantly, we successfully performed molecular optimization through style transfer, even when employing both teacher forcing and our regularizer. Finally, our experiments on the ZINC250k dataset (Section 5.2) show, that teacher forcing and an added regularizer are both required to achieve near perfect validity (and compete with grammar based approaches).

The impact of this is that the use of teacher forcing in the decoder might be permissible or even desirable, which is an impactful observation potentially overlooked in the literature. Additionally, it makes our methodology much

more computationally efficient, as the teacher forcing decoder does not require random sampling during sequence generation (Section 3.3.2).

In light of these observations, we conclude that the study was a measured success. We successfully combined the two model classes and addressed the performance issues of molecular VAEs. Additionally, we found convincing evidence challenging a fundamental assumption about their limitations. Finally, we thoroughly evaluated their ability to perform molecular style transfer in our desired setting.

Naturally, our study also possesses limitations. In the following section, we will analyse these and explain how they can be overcome. Finally, in Section 6.2.2, will suggest future work.

6.2 Future Work

6.2.1 Strengthening the Conclusions of the Study

The performance of our regularizers may be influenced by numerous confounding variables, potentially affecting the robustness of our findings. Recognizing the importance of these factors, we present a list of limitations of our experimental approach and propose how they may be addressed in future work:

- **Adapting Regularizers for Use with One-Shot Decoders**: As discussed in Section 3.3.2, one-shot decoders are fast during training and inference and help avoid posterior collapse. It would therefore be helpful to improve or adapt the regularizers presented in this study to work with this architecture. To this end, future work could further experimentally validate our hypothesized reasons for their failure (Sections 5.1.1 5.1.5).
- **Deeper Examination of Latent Space**: Our conclusions regarding the latent space in Chapter 5 are based on observing the models style transfer performance. It could be beneficial to examine it directly. In particular, future work could examine the locality of the latent space by examining the structural similarity of a molecule's latent space neighbourhood or sample molecules along latent trajectories.
- **Evaluate Style Transfer for Different Properties**: As discussed in Section 3.3.1, the performance of molecular style transfer depends on the specific property score considered. Property scores susceptible to minor structural changes would likely complicate the disentanglement of property-dependent and independent structural information. Future research could explore style transfer efficacy for more complex molecular properties, such as the QED and SA scores introduced in Section 2.3.1.

- **Evaluate Model in a Multi-Property Setting**: As mentioned in Section 2.3.1 a drug design setting is typically concerned with multi-property control. Future work could examine the ability of the regularizers to improve conditional generation and be able to perform style transfer in a multi-property setting.

6.2.2 Further Impactful Applications of the Methodology

The regularizers presented in this report help address two significant issues of VAEs for conditional generation: validity and property conditioning performance. Grammar-based models, such as SD-VAE and GVAE, presented in Section 2.4.2, target the former by modifying the molecular representation. This addresses the issue by altering the "language" of the model so that it can only generate valid SMILES. It, therefore, seems plausible that this would not improve issues related to the poor conditional performance of one-shot decoder VAEs, which could be addressed with our methodology. This would show that our methodology brings value to a newer and more performant model class, improving the impact of this work. Additionally, this would represent a novel and potentially effective way to guide generation with an SD-VAE model. Although we have developed code for a conditional SD-VAE and a surrogate attribute grammar LSTM, time constraints prevented us from completing this line of inquiry.

Appendix A

Sample Molecules

Figures A.1-A.7 contain plots of samples of molecules generated with each of the successful regularizer-decoder combinations trained with the QM9 dataset. Figures A.8-A.12 contain samples of molecules from models trained on the ZINC250k dataset that generated a significant proportion of valid SMILES. They are provided to facilitate a qualitative assessment of conditional generation performance. Each plot was created by randomly selecting 25 valid molecules from the decoder test sample.

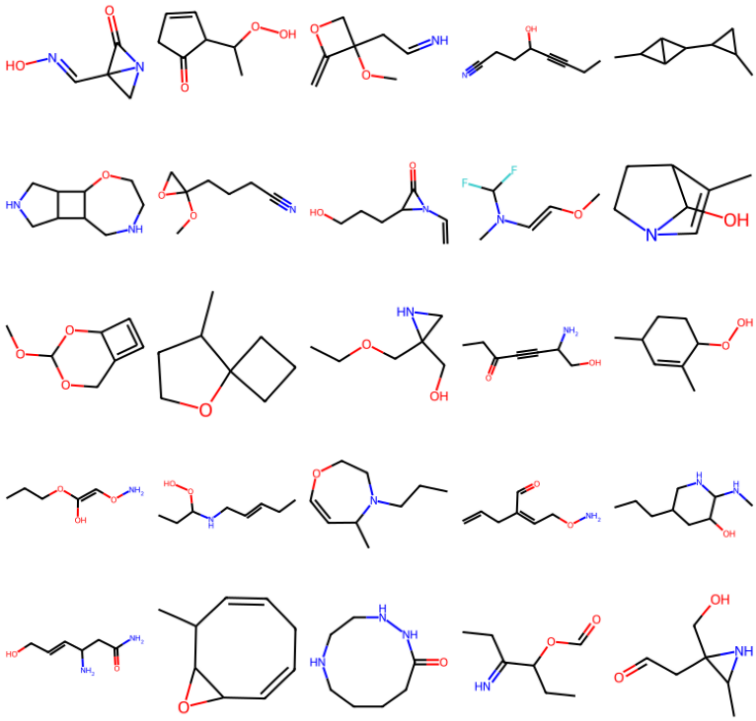

Figure A.1: Sample of valid molecules from base cVAE model trained on QM9.

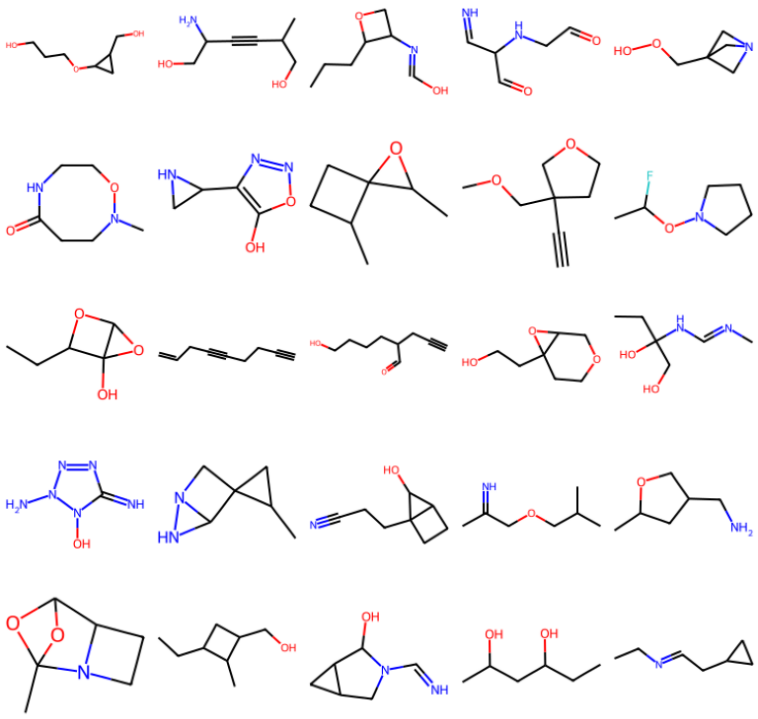

Figure A.2: Sample of valid molecules from the base EXP-cVAE model.

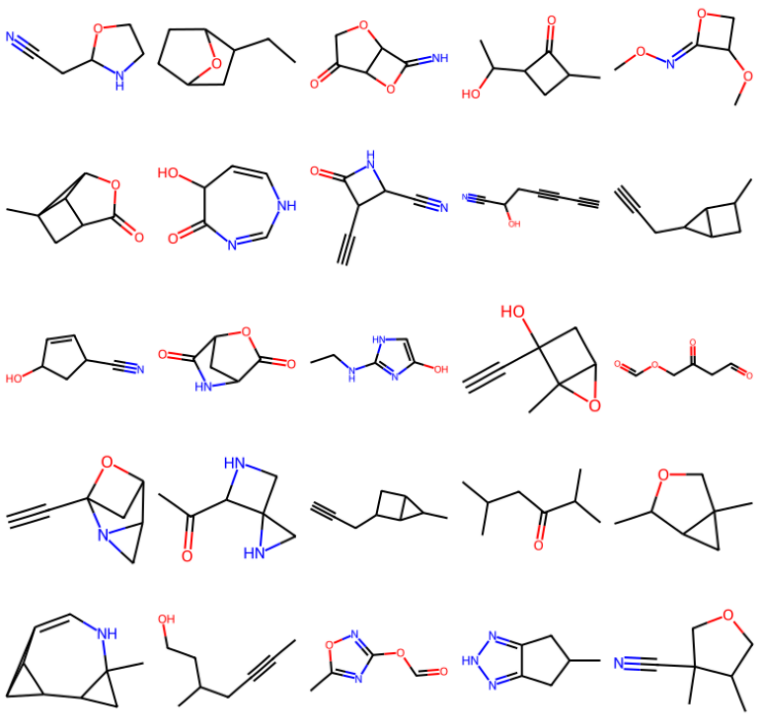

Figure A.3: Sample of valid molecules from the EXP-cVAE-KLD model trained on QM9.

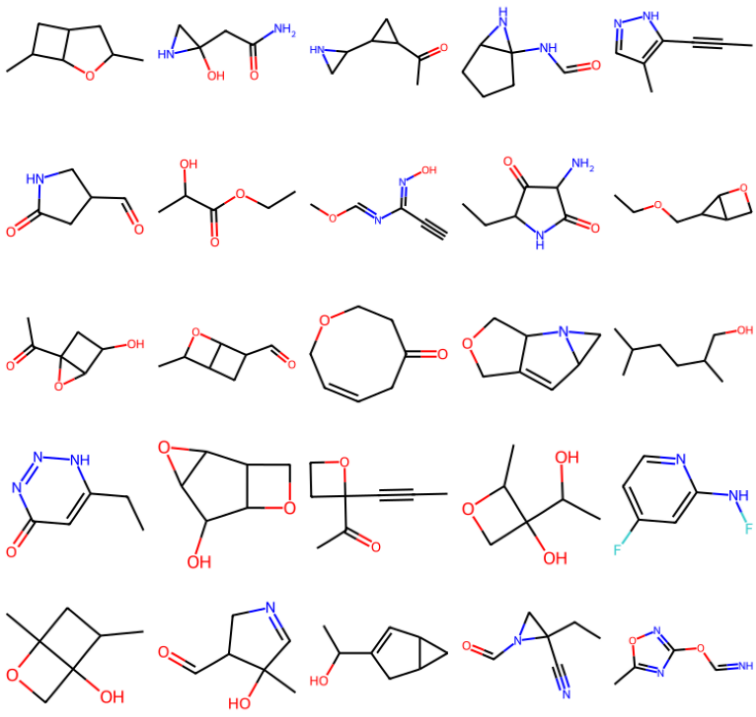

Figure A.4: Sample of valid molecules from the EXP-cVAE-Pol2 model trained on QM9.

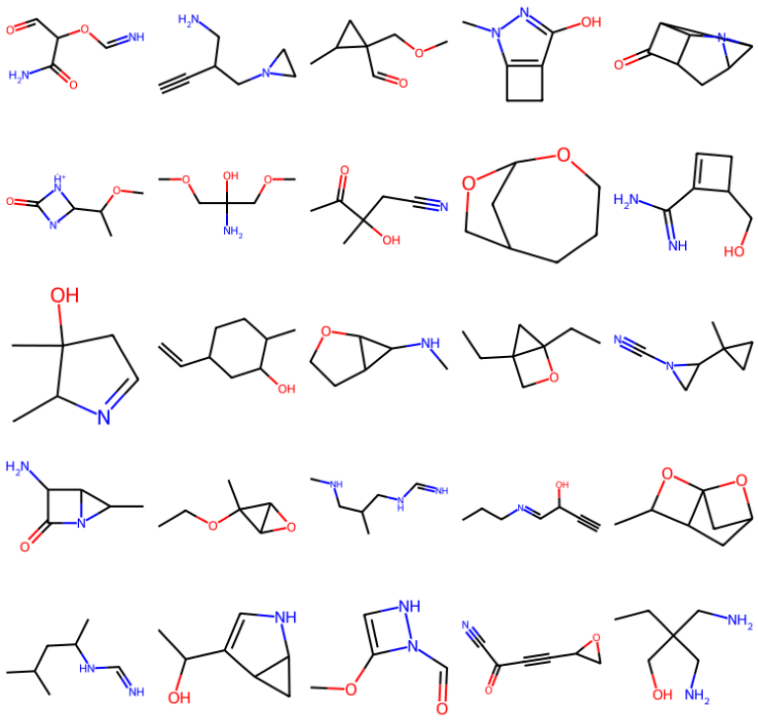

Figure A.5: Sample of valid molecules from the base TF-cVAE model trained on QM9.

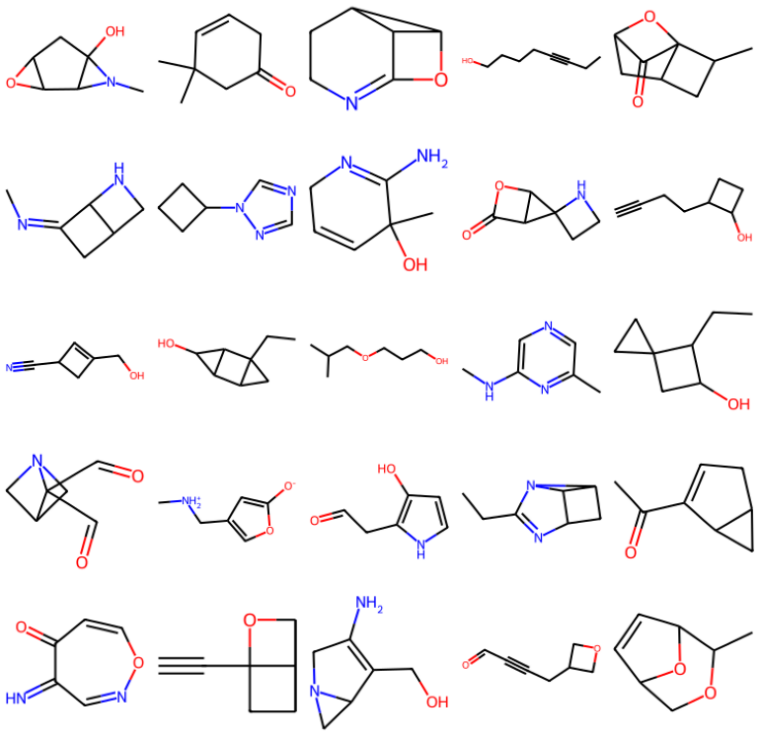

Figure A.6: Sample of valid molecules from the TF-cVAE-KLD model trained on QM9.

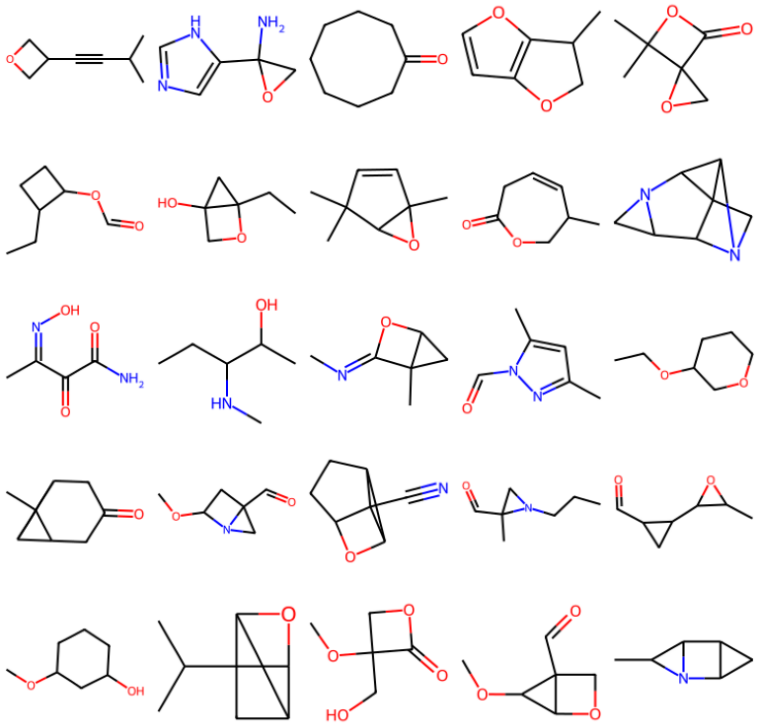

Figure A.7: Sample of valid molecules from the TF-cVAE-Pol2 model trained on QM9.

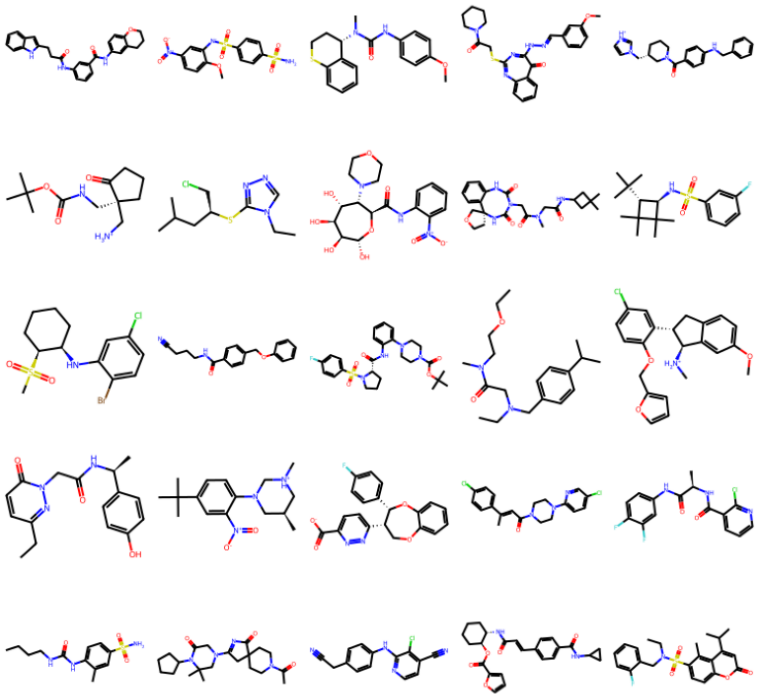

Figure A.8: Sample of valid molecules from the EXP-cVAE-KLD model trained on ZINC250k.

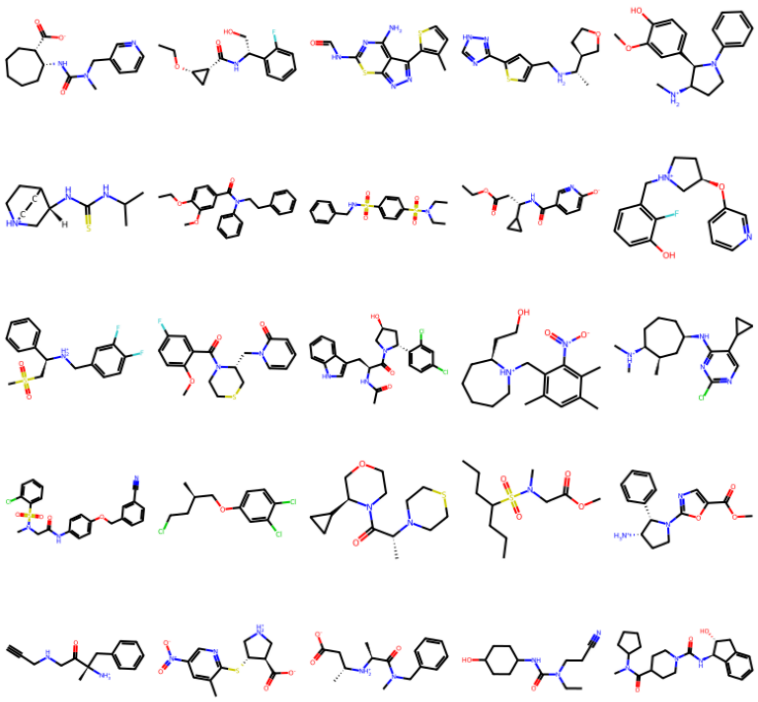

Figure A.9: Sample of valid molecules from the EXP-cVAE-Pol2 model trained on ZINC250k.

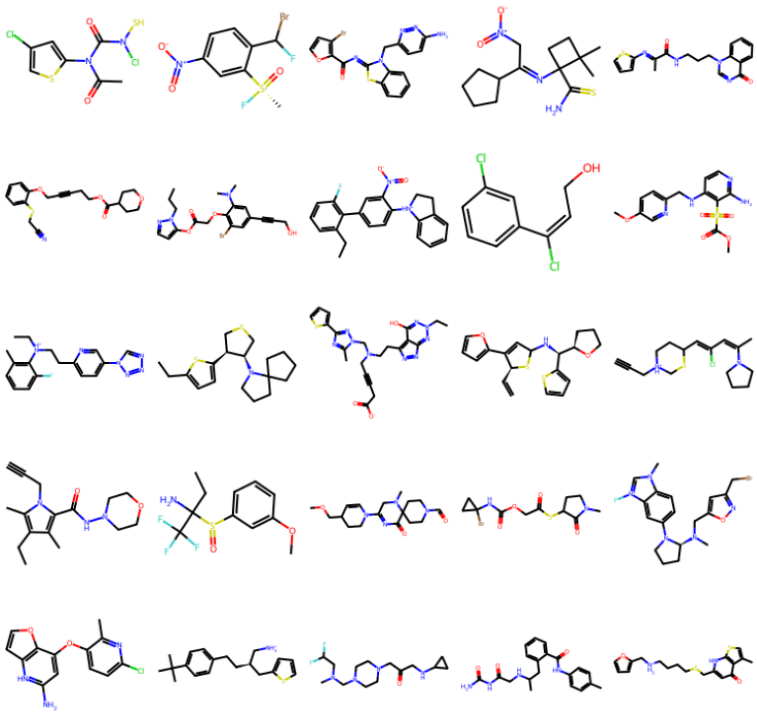

Figure A.10: Sample of valid molecules from the base TF-cVAE model trained on ZINC250k.

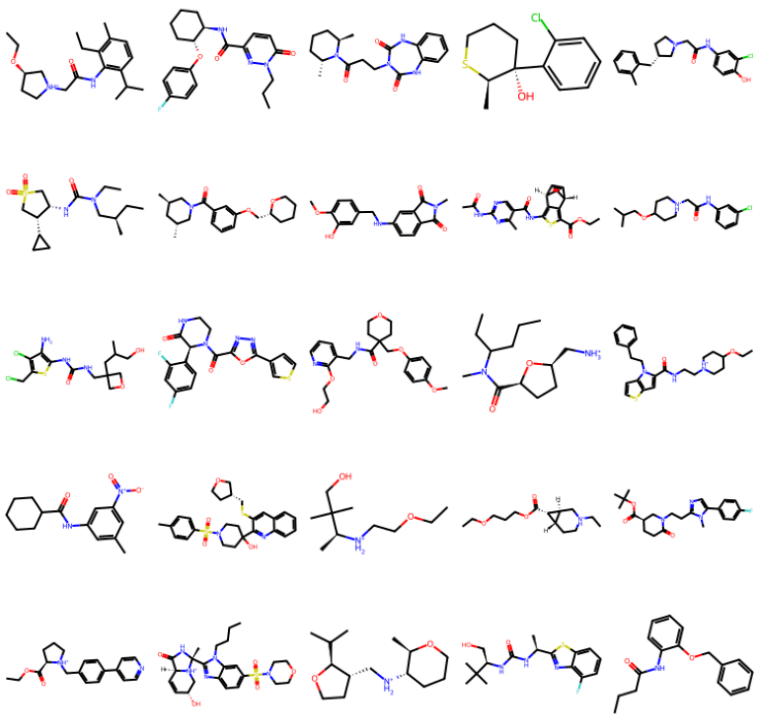

Figure A.11: Sample of valid molecules from the TF-cVAE-KLD model trained on ZINC250k.

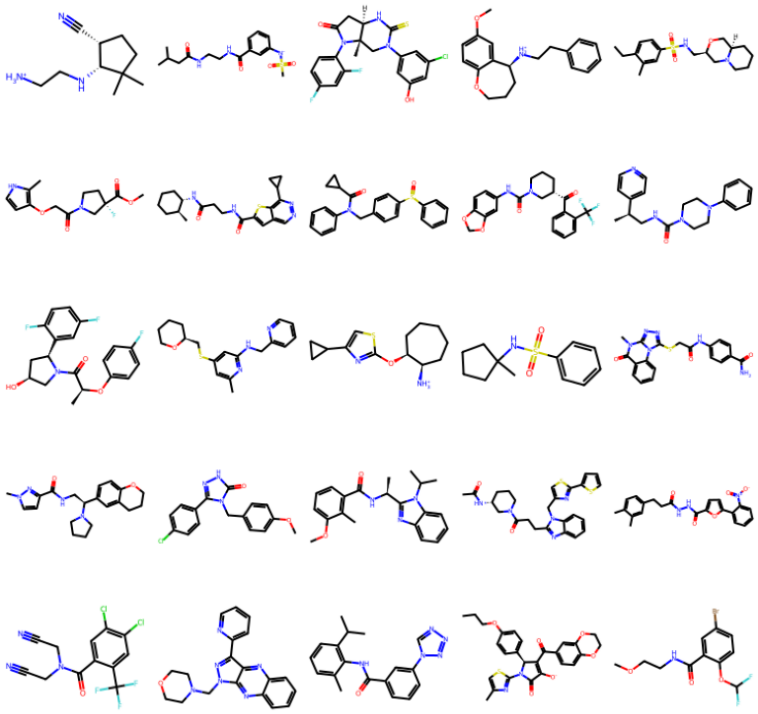

Figure A.12: Sample of valid molecules from the TF-cVAE-Pol2 model trained on ZINC250k.

Bibliography

Adams, K. and Coley, C. W. (2022). Equivariant shape-conditioned generation of 3d molecules for ligand-based drug design. *ArXiv*, abs/2210.04893.

Agrawal, G., Bader, F., Günther, J., and Wurzer, S. (2023). Fast to first-in-human: Getting new medicines to patients more quickly.

Arnold, C. (2023). Inside the nascent industry of AI-designed drugs. *Nature Medicine*, 29:1292–1295.

Arús-Pous, J., Johansson, S., Prykhodko, O., Bjerrum, E. J., Tyrchan, C., Reymond, J., Chen, H., and Engkvist, O. (2019). Randomized smiles strings improve the quality of molecular generative models. *Journal of Cheminformatics*, 11.

Bagal, V., Aggarwal, R., Vinod, P. K., and Priyakumar, U. D. (2021). Molgpt: Molecular generation using a transformer-decoder model. *Journal of chemical information and modeling*.

Bayer, J. and Osendorfer, C. (2014). Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*.

Bickerton, G. R. J., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. (2012). Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4 2:90–8.

Bird, A. and Williams, C. K. (2019). Customizing sequence generation with multi-task dynamical systems. *arXiv preprint arXiv:1910.05026*.

Bjerrum, E. J. (2017). Smiles enumeration as data augmentation for neural network modeling of molecules. *ArXiv*, abs/1703.07076.

Bjerrum, E. J. and Threlfall, R. (2017). Molecular generation with recurrent neural networks (rnns). *ArXiv*, abs/1705.04612.

Born, J. and Manica, M. (2023). Regression transformer enables concurrent sequence regression and generation for molecular language modelling. *Nature Machine Intelligence*, 5(4):432–444.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*.

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

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Capecchi, A., Probst, D., and Reymond, J.-L. (2020). One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome. *Journal of cheminformatics*, 12:1–15.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, pages 2980–2988.

Dai, H., Tian, Y., Dai, B., Skiena, S., and Song, L. (2018). Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*.

Ertl, P. and Schuffenhauer, A. (2009). Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1:8 – 8.

Fabius, O. and van Amersfoort, J. R. (2014). Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*.

Feng, W., Wang, L., Lin, Z., Zhu, Y., Wang, H., Dong, J., Bai, R., Wang, H., Zhou, J., Peng, W., et al. (2024). Generation of 3d molecules in pockets via a language model. *Nature Machine Intelligence*, pages 1–12.

Gómez-Bombarelli, R., Duvenaud, D. K., Hernández-Lobato, J. M., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2016). Automatic chemical design using a data-driven continuous representation of molecules. *CoRR*, abs/1610.02415.

Grechishnikova, D. (2019). Transformer neural network for protein-specific de novo drug generation as a machine translation problem. *Scientific Reports*, 11.

Guo, X. and Zhao, L. (2022). A systematic survey on deep generative models for graph generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5370–5390.

He, J., You, H., Sandström, E., Nittinger, E., Bjerrum, E. J., Tyrchan, C., Czechtizky, W., and Engkvist, O. (2020). Molecular optimization by capturing chemist's intuition using deep neural networks. *Journal of Cheminformatics*, 13.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hong, S. H., Lim, J., Ryu, S., and Kim, W. Y. (2019). Molecular generative model based on adversarially regularized autoencoder. *Journal of chemical information and modeling*.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. (2012). Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768.

Isert, C., Atz, K., and Schneider, G. (2022). Structure-based drug design with geometric deep learning. *Current opinion in structural biology*, 79:102548.

Izdebski, A., Weglarz-Tomczak, E., Szczurek, E., and Tomczak, J. M. (2023). De novo drug design with joint transformers. *ArXiv*, abs/2310.02066.

Jeon, W. and Kim, D. (2020). Autonomous molecule generation using reinforcement learning and docking to develop potential novel inhibitors. *Scientific Reports*, 10.

Jiang, Z., Wang, Z., Zhang, J., Wub, M., Li, C., and Yamanishi, Y. (2023). Mode collapse alleviation of reinforcement learning-based gans in drug design. *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 3045–3052.

Jin, W., Barzilay, R., and Jaakkola, T. (2018a). Junction tree variational autoencoder for molecular graph generation. *ArXiv*, abs/1802.04364.

Jin, W., Yang, K., Barzilay, R., and Jaakkola, T. (2018b). Learning multimodal graph-to-graph translation for molecular optimization. *ArXiv*, abs/1812.01070.

Kadurin, A., Nikolenko, S. I., Khrabrov, K., Aliper, A., and Zhavoronkov, A. (2017). druGAN: An advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular pharmaceutics*, 14 9:3098–3104.

kali, M., Jiménez, J., Sabbadin, D., and Fabritiis, G. D. (2019). Shape-based generative modeling for de novo drug design. *Journal of chemical information and modeling*, 59 3:1205–1214.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kotsias, P.-C., Arús-Pous, J., Chen, H., Engkvist, O., Tyrchan, C., and Bjerrum, E. J. (2020). Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks. *Nature Machine Intelligence*, 2:254 – 265.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. (2017). Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925*.

Landrum, G. (2016). Rdkit: Open-source cheminformatics software.

Lim, J., Ryu, S., Kim, J. W., and Kim, W. Y. (2018). Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of Cheminformatics*, 10.

Lipinski, C. A. (2004). Lead- and drug-like compounds: the rule-of-five revolution. *Drug discovery today. Technologies*, 1 4:337–41.

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. (2018). Constrained graph variational autoencoders for molecule design. *Advances in Neural Information Processing Systems*, 31.

Liu, Z., Shi, Y., Zhang, A., Zhang, E., Kawaguchi, K., Wang, X., and Chua, T.-S. (2023). Rethinking tokenizer and decoder in masked graph modeling for molecules. *ArXiv*, abs/2310.14753.

Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I. J. (2015). Adversarial autoencoders. *ArXiv*, abs/1511.05644.

Mokaya, M., Imrie, F., van Hoorn, W. P., Kalisz, A., Bradley, A. R., and Deane, C. M. (2023). Testing the limits of smiles-based de novo molecular generation with curriculum and deep reinforcement learning. *Nature Machine Intelligence*, 5(4):386–394.

Mollaysa, A., Paige, B., and Kalousis, A. (2019). Conditional generation of molecules from disentangled representations.

Mollaysa, A., Paige, B., and Kalousis, A. (2020). Goal-directed generation of discrete structures with conditional generative models. *Advances in Neural Information Processing Systems*, 33:21923–21933.

O'Boyle, N. M. (2012). Towards a universal smiles representation - a standard method to generate canonical smiles based on the inchi. *Journal of Cheminformatics*, 4:22 – 22.

O'Boyle, N. M. and Dalke, A. (2018). Deepsmiles: An adaptation of smiles for use in machine-learning of chemical structures. *ChemRxiv*.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. (2017). Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9.

Pang, C., Qiao, J., Zeng, X., Zou, Q., and Wei, L. (2023a). Deep generative models in de novo drug molecule generation. *Journal of Chemical Information and Modeling*.

Pang, C., Qiao, J., Zeng, X., Zou, Q., and Wei, L. (2023b). Deep generative models in de novo drug molecule generation. *Journal of Chemical Information and Modeling*.

Popova, M., Isayev, O., and Tropsha, A. (2017). Deep reinforcement learning for de novo drug design. *Science Advances*, 4.

Qureshi, R., Irfan, M., Gondal, T. M., Khan, S., Wu, J., Hadi, M. U., Heymach, J. V., Le, X., Yan, H., and Alam, T. (2023). Ai in drug discovery and its clinical relevance. *Heliyon*, 9.

Ragoza, M., Masuda, T., and Koes, D. R. (2020). Learning a continuous representation of 3d molecular structures with deep generative models. *ArXiv*, abs/2010.08687.

Ragoza, M., Masuda, T., and Koes, D. R. (2021). Generating 3d molecules conditional on receptor binding sites with deep generative models. *Chemical Science*, 13:2701 – 2713.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7.

Ruddigkeit, L., Van Deursen, R., Blum, L. C., and Reymond, J.-L. (2012). Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875.

Sakhinana, S. and Runkana, V. (2023). Crossing new frontiers: Knowledge-augmented large language model prompting for zero-shot text-based de novo molecule design. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. (2017). Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 4(1):120–131.

Simonovsky, M. and Komodakis, N. (2018). Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*.

Tang, H., Li, C., Kamei, S., Yamanishi, Y., and Morimoto, Y. (2024). Molecular generative adversarial network with multi-property optimization. *arXiv preprint arXiv:2404.00081*.

Thiede, L. A., Krenn, M., Nigam, A., and Aspuru-Guzik, A. (2020). Curiosity in exploring chemical spaces: intrinsic rewards for molecular reinforcement learning. *Machine Learning: Science and Technology*, 3.

Wang, B., Wang, Z., Wang, X., Cao, Y., Saurous, R. A., and Kim, Y. (2023a). Grammar prompting for domain-specific language generation with large language models. In *Neural Information Processing Systems*.

Wang, J., Hsieh, C.-Y., Wang, M., Wang, X., Wu, Z., Jiang, D., Liao, B., Zhang, X., Yang, B., He, Q., Cao, D., Chen, X., and Hou, T. (2021). Multi-constraint molecular generation based on conditional transformer, knowledge distillation and reinforcement learning. *Nature Machine Intelligence*, 3:914 – 922.

Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D. F., and Chao, L. S. (2019). Learning deep transformer models for machine translation. In *Annual Meeting of the Association for Computational Linguistics*.

Wang, Z., Nie, W., Qiao, Z., Xiao, C., Baraniuk, R., and Anandkumar, A. (2023b). Retrieval-based controllable molecule generation. In *The Eleventh International Conference on Learning Representations*.

Weininger, D. (1988). Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28:31–36.

Xia, J., Zhao, C., Hu, B., Gao, Z., Tan, C., Liu, Y., Li, S., and Li, S. Z. (2023). Mole-bert: Rethinking pre-training graph neural networks for molecules. In *International Conference on Learning Representations*.

Xia, Y., Wang, Y., Wang, Z., and Zhang, W. (2024). A comprehensive review of molecular optimization in artificial intelligence-based drug discovery. *Quant. Biol.*, 12:15–29.

Yang, N., Wu, H., Yan, J., Pan, X., Yuan, Y., and Song, L. (2022). Molecule generation for drug design: a graph learning perspective. *ArXiv*, abs/2202.09212.

You, J., Liu, B., Ying, Z., Pande, V., and Leskovec, J. (2018). Graph convolutional policy network for goal-directed molecular graph generation. *Advances in Neural Information Processing Systems*, 31.

Zhang, O., Zhang, J., Jin, J., Zhang, X., Hu, R., Shen, C., Cao, H., Du, H., Kang, Y., Deng, Y., Liu, F., Chen, G., Hsieh, C.-Y., and Hou, T. (2023a). Resgen is a pocket-aware 3d molecular generation model based on parallel multiscale modelling. *Nature Machine Intelligence*, 5:1020 – 1030.

Zhang, Y., Li, S., Xing, M., Yuan, Q., He, H., and Sun, S. (2023b). Universal approach to de novo drug design for target proteins using deep reinforcement learning. *ACS Omega*, 8:5464 – 5474.

Zhang, Z. and Liu, Q. (2023). Learning subpocket prototypes for generalizable structure-based drug design. In *International Conference on Machine Learning*.

Zhang, Z., Min, Y., Zheng, S., and Liu, Q. (2023c). Molecule generation for target protein binding with structural motifs. In *International Conference on Learning Representations*.

Zhou, Z., Kearnes, S. M., Li, L., Zare, R. N., and Riley, P. F. (2018). Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. In consultation with the supervisor, one of the following three options must be selected:

○ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].

◉ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies[2].

○ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[3]. In consultation with the supervisor, I did not cite them.

Title of paper or thesis:

Integrating Latent Variable and Auto-Regressive Models for Enhanced Goal Directed Molecule Generation

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	**First name(s):**
Heath	Arthur-Louis

With my signature I confirm the following:

- I have adhered to the rules set out in the Citation Guide.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	**Signature(s)**
Zurich, 24.7.2024	

If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] E.g. ChatGPT, DALL E 2, Google Bard
[2] E.g. ChatGPT, DALL E 2, Google Bard
[3] E.g. ChatGPT, DALL E 2, Google Bard