UNIVERSITÀ
DI SIENA
1240

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
E SCIENZE MATEMATICHE

Corso di Laurea in
Artificial Intelligence and Automation Engineering

Organic Molecule Generation using Latent Diffusion

Relatore:
Prof. Franco Scarselli

Candidato:
Duccio Meconcelli

Correlatori:
Niccolò Pancino
Pietro Bongini

Anno Accademico 2023-2024

Caro m'è 'l sonno, e più l'esser di sasso,
mentre che 'l danno e la vergogna dura:
non veder, non sentir m'è gran ventura;
però non mi destar, deh, parla basso.

— Michelangelo Buonarroti

Abstract

This thesis explores recent advancements in graph generation using latent diffusion models. We present a novel framework that employs a two-stage approach: first mapping graphs to a low-dimensional latent space via a graph variational autoencoder, then applying diffusion models in this latent space. This enables more efficient and expressive graph generation compared to existing methods that operate directly in discrete graph space. The methodology is applied specifically to molecule generation tasks, aiming to capitalize on the strengths of both techniques: VAEs' ability to learn complex distributions over molecular structures and Latent Diffusion's efficiency in exploring latent spaces. While our results do not consistently surpass the state-of-the-art across all metrics, our model demonstrates promising performance, consistently outperforming GraphVAE, the model from which we began our research. This research contributes to the advancement of more effective generative models for organic molecule synthesis, paving the way for further exploration in areas such as conditional generation, iterative refinement, and self-supervised learning.

All the code of this project can be found on this GitHub Page [1].

Contents

1 Introduction

The ability to generate new, synthetic data that resembles real-world samples is a powerful capability of modern artificial neural networks. This approach, known as generative modeling, has seen remarkable progress in recent years across various domains including images, text, audio, and graphs.

Generative models aim to learn the underlying probability distribution of a given dataset, allowing them to generate new samples that are similar to, but distinct from, the training data. This is in contrast to discriminative models, which learn to map inputs to specific outputs or classifications.

1.1 Graph generation

Graph generation is a fundamental and challenging problem in machine learning with wide-ranging applications across fields such as chemistry, biology, social network analysis, and materials science. The task involves producing novel graph structures that exhibit properties similar to those found in a given dataset of real-world graphs. This is particularly challenging due to the discrete nature of graphs and the complex dependencies between nodes and edges.

Graph generation poses several key challenges:

❶ Graphs have variable sizes and topologies, unlike fixed-dimensional data like images.

❷ There are complex dependencies between nodes and edges that must be captured.

❸ Graphs are permutation invariant - the node ordering should not affect the overall structure.

❹ For many applications, the generated graphs must satisfy domain-specific constraints.

❺ Evaluating the quality of generated graphs is non-trivial and often domain-specific.

In recent years, deep learning approaches have shown great promise for graph generation tasks. Neural network architectures like Graph Convolutional Networks [2] and Graph Attention Networks [3] have enabled powerful representations of graph-structured data. Building on these advances, various deep generative models have been adapted for graph generation with impressive results. For example AlphaFold 3, developed by DeepMind, is indeed an excellent example of graph-based deep learning applied to a critical problem in biology. While it's not primarily a graph generation model, it uses graph neural networks to predict protein structures; use attention-based neural networks to process protein sequences and predict their 3D structures. [4]

1.2 Molecule generation

This thesis focuses on the application of graph generation to the bioinformatics domain, specifically for *de novo* molecular design. The ability to computationally generate novel molecular structures with desired properties has immense potential to accelerate drug discovery and materials science. However, modeling the complex chemical and structural constraints of molecules presents unique challenges for graph generation approaches.

Molecules can be naturally represented as graphs, with atoms as nodes and chemical bonds as edges. Generated molecular graphs must satisfy chemical validity constraints such as valency rules. Beyond just validity, generated molecules should ideally be synthesizable. This makes molecular graph generation a rich test bed for developing expressive yet constrained generative models.

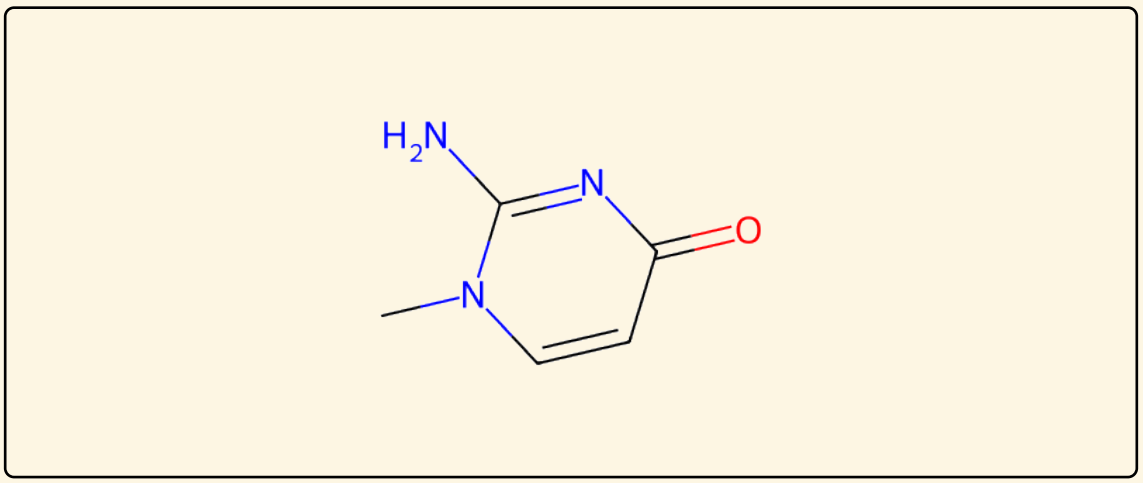


Figure 1: An example of Molecule from QM9 Dataset [5]

1.3 Our Model

To address these challenges, we propose a novel graph generation framework that combines a Graph Variational Autoencoder (GraphVAE) [6] with a Latent Diffusion Model (LDM). The GraphVAE component learns to encode molecular graphs into a continuous latent space, while preserving important structural properties. Then,the LDM models the distribution of these latent representations and enables high-quality sampling of novel molecular structures.

Our approach draws inspiration from recent breakthroughs in image generation, such as DALL-E [7] and Stable Diffusion [8], which have demonstrated the power of combining VAEs with diffusion models in latent space. By adapting these ideas to the graph domain, we aim to push the boundaries of molecular generation capabilities. The latent diffusion approach allows us to leverage the strengths of both VAEs and diffusion models - the VAE provides a compact and meaningful latent space, while the diffusion model enables high-quality and diverse sampling.

Some key advantages of our proposed approach include:

1. The ability to generate valid molecular graphs with high probability
2. Fast sampling compared to autoregressive approaches
3. The potential to leverage pretrained VAE models for transfer learning

Beyond molecular design, this latent graph diffusion framework could potentially be applied to other graph generation domains like synthesizing protein structures, generating synthetic social networks, or designing novel materials.

1.4 Summary of the next chapters

This thesis is organized as follows:

- **Chapter 2** surveys related work in graph generation, molecular design, and diffusion models, providing important context and baselines for our approach.
- **Chapter 3** presents the technical details of our proposed GraphVAE + LDM framework, including the model architectures, training procedure, and generation algorithm.
- **Chapter 4** describes the experimental setup used to evaluate our method, including datasets, model configurations, and evaluation metrics.
- **Chapter 5** provides a comprehensive analysis of the results, comparing our approach to state-of-the-art baselines.
- Finally, **Chapter 6** concludes with a summary of our key findings and contributions, as well as a discussion of future research directions.

Through this work, we aim to demonstrate the potential of latent diffusion models for graph generation. By combining the strengths of GraphVAEs and LDMs, we hope to enable more efficient and effective generation of novel molecular structures to accelerate scientific discovery. The proposed framework provides a flexible foundation that can potentially be extended to other graph generation domains beyond chemistry in future work.

2 Related Works

In recent years, the field of graph generation has seen significant advancements, largely driven by the application of deep learning techniques. This chapter provides an overview of the current state of the art in graph generation using neural networks.

2.1 Graph Rappresentation

Graphs are mathematical structures used to represent relationships between objects. A graph G is defined by a set of nodes V (vertices) and a set of edges E that connect pairs of nodes. Formally, $G = (V, E)$.

A common representation of a graph structure is using the adjacency matrix (A). If the graph has N nodes, A is an $N \times N$ matrix where the element $A_{i,j}$ is 1 if there is an edge between node i and node j, otherwise it is 0. In addition to the connectivity structure, nodes and edges can have attributes or characteristics associated with them. For example, in a molecule, atoms are represented as nodes and chemical bonds are represented as edges. Node characteristics might include atom type and formal charge, while edge characteristics might include bond type and bond distance.

2.2 Traditional Methods for Graph Generation

Graph generation has been a topic of interest long before the advent of neural networks. Traditional methods for generating graphs typically involve specifying a generative process that defines how edges between nodes are created. These methods often rely on statistical properties and probabilistic models to generate graphs that mimic real-world structures.

One of the earliest and most well-known models is the Erdős-Rényi (ER) model [9]. Introduced by Paul Erdős and Alfréd Rényi in 1960. This model generates graphs by connecting nodes randomly. Specifically, the probability of an edge existing between any two nodes is defined by a parameter (r), which controls the density of the graph. Despite its simplicity, the ER model has been fundamental in understanding random graph theory and has laid the groundwork for more complex models.

Another significant traditional method is the Barabási-Albert (BA) model [9]. This model, introduced by Albert-László Barabási and Réka Albert in 1999, generates scale-free networks using a preferential attachment mechanism. In this model, new nodes are more likely to connect to existing nodes with a higher degree, leading to the emergence of hubs. This method has been particularly useful in modeling real-world networks such as social networks, the internet, and biological systems.

The Watts-Strogatz (WS) model [9], proposed by Duncan J. Watts and Steven Strogatz in 1998, is another traditional approach that generates small-world networks. This model starts with a regular lattice and then randomly rewires edges with a certain probability. The resulting graphs exhibit high clustering coefficients and short average path lengths, properties observed in many real-world networks.

These traditional methods have provided valuable insights into the structural properties of graphs and have served as benchmarks for evaluating more advanced graph generation techniques. However, they often rely on hand-crafted statistical features and may oversimplify the underlying distributions of real-world graphs [10] As a result, they may not capture the full complexity and diversity of graph structures observed in practice.

2.3 Generative Neural Networks

Generative Neural Networks have revolutionized the field of graph generation by leveraging the power of deep learning to model complex data distributions. Among the various types Variational Autoencoders (VAEs) and Diffusion Models have emerged as prominent approaches due to their ability to generate high-quality and diverse graph structures.

2.3.1 Variational Autoencoders

Variational Autoencoders (VAEs) are a class of generative models introduced by Diederik P. Kingma and Max Welling in 2013 [11]. VAEs combine principles from probabilistic graphical models and neural networks to learn a latent space representation of the input data. The key idea behind VAEs is to encode the input data into a probabilistic latent space and then decode it back to the original space, allowing for the generation of new data samples.

The architecture of a VAE consists of an encoder, which maps the input data to a latent space, and a decoder, which reconstructs the data from the latent space. The encoder outputs a distribution over the latent space, typically modeled as a multivariate Gaussian distribution. During training, the model optimizes the Evidence Lower Bound (ELBO), which balances the reconstruction loss and the Kullback-Leibler (KL) divergence between the learned latent distribution and a prior distribution [12].

2.3.2 Diffusion Models

Diffusion Models, also known as Diffusion Probabilistic Models or Score-Based Generative Models, are a class of generative models that learn to generate data by reversing a diffusion process. These models were introduced by Sohl-Dickstein et al. in 2015 [13]. The core idea is to model the data generation process as a sequence of steps that gradually add noise to the data, and then learn to reverse this process to generate new samples.

A Diffusion Model consists of three main components: the forward process, the reverse process, and the sampling procedure. The forward process gradually adds Gaussian noise to the data, transforming it into a noise distribution. The reverse process, typically modeled using a neural network, learns to denoise the data step by step, effectively reversing the forward process.

Diffusion Models have shown remarkable success in generating high-quality data, particularly in the field of computer vision. For example, the Denoising Diffusion Probabilistic Model (DDPM) introduced by Ho et al. (2020) has been widely adopted for image generation tasks [14].

2.4 Generative Neural Networks for Graphs

Generative Neural Networks have significantly advanced the field of graph generation by leveraging deep learning techniques to model complex graph structures. This section explores various GNN-based approaches, including Graph Variational Autoencoders, Latent Diffusion Models, and Auto-Regressive Models.

2.4.1 Graph Variational Autoencoders

The Graph Variational Autoencoders (GraphVAE) is a generative model that uses variational autoencoders (VAEs) to generate small graphs. It was introduced in a research paper that proposes to bypass the challenges associated with the linearization of such discrete structures by having a decoder that directly produces a fully connected probabilistic graph of a predefined maximum size all at once [6].

The model takes a graph as input, such as a molecule represented by atoms (nodes) and bonds (edges). The model's encoder, defined by a variational posterior, embeds the graph into a continuous representation, essentially compressing it into a vector in a latent code space. The dimensionality of this latent space is typically quite small, encouraging the autoencoder to learn a high-level compression of the input instead of simply copying any given input.

The decoder, defined by a generative distribution, takes this vector from the latent code space and generates a fully connected probabilistic graph.

A standard graph matching algorithm is then used to align the generated graph with the ground truth graph (graph matching).

The entire model is trained by minimizing an upper bound to the negative log-likelihood (ELBO). This involves using a reconstruction loss, which ensures a high similarity between the sampled generated graphs and the input graph, and a KL divergence, which regularizes the code space to allow for sampling.

GraphVAE has been evaluated on the challenging task of molecule generation, and while its performance leaves room for improvement, it represents an important first step toward powerful and efficient graph decoders.

2.4.2 Generating Molecules Based on Latent Diffusion Model for Hierarchical Graph

The Hierarchical Graph Latent Diffusion Model (HGLDM) represents a novel approach to molecule generation by leveraging latent diffusion processes. This model, introduced by Bian et al. (2023) [15], addresses the challenge of generating hierarchical graph structures by incorporating node-level, subgraph-level, and graph-level

embeddings. The HGLDM framework overcomes the limitations of traditional diffusion models by ensuring that the generated graphs preserve the structural properties of the original data.

The HGLDM has shown superior performance in generating complex molecular structures, significantly reducing computation time compared to existing graph diffusion models. This approach has been evaluated on multiple benchmarks, demonstrating its effectiveness in both unconditional and conditional generation tasks.

2.4.3 Hyperbolic Geometric Latent Diffusion Model for Graph Generation

The Hyperbolic Geometric Latent Diffusion Model (HypDiff) is another innovative approach that combines the strengths of diffusion models and hyperbolic embeddings. Proposed by Fu et al. (2024) [16], HypDiff aims to capture the non-Euclidean structure of graphs by diffusing them within a hyperbolic latent space. This method ensures that the generated graphs maintain their topological properties, leveraging the geometric properties of hyperbolic spaces.

HypDiff has been shown to be effective in generating graphs with various topologies, providing a robust framework for graph generation tasks. The model's ability to preserve the original topological properties makes it a valuable tool for applications requiring high-fidelity graph generation.

2.4.4 Graphusion

Graphusion is a model that leverages large language models (LLMs) for scientific knowledge graph fusion and construction. Introduced by Yang et al. (2024) [17], Graphusion employs a variational graph autoencoder to map raw graphs to a low-dimensional latent space, followed by latent diffusion models to generate new graphs. This approach provides a smoother, faster, and more expressive graph generation process.

Graphusion has demonstrated its effectiveness in constructing knowledge graphs from free text, surpassing traditional methods in accuracy and efficiency. The model's ability to integrate entity merging, conflict resolution, and novel triplet discovery makes it a powerful tool for knowledge graph construction.

2.4.5 Auto-Regressive Generative Models for Graph Generation

Auto-regressive generative models represent a class of models that generate graphs by sequentially adding nodes and edges. One notable example is the GraphRNN model introduced by You et al. (2018) [18]. GraphRNN decomposes the graph generation process into a sequence of node and edge generations, conditioned on the graph structure generated so far. This approach allows the model to learn complex graph distributions and generate realistic graph structures.

Another recent development is the Autoregressive Diffusion Model for Graph Generation proposed by Kong et al. (2023) [19]. This model defines a node-

absorbing diffusion process that operates directly in the discrete graph space, improving training efficiency and sampling speed. The autoregressive nature of the model allows it to incorporate constraints and generate high-quality graphs.

In summary, Generative Neural Networks for graphs encompass a wide range of approaches, each with its unique strengths and applications. From Graph VAEs to diffusion models and auto-regressive models, these techniques have significantly advanced the field of graph generation, providing powerful tools for various applications.

3 The Model

The proposed architecture consists of two main components: a Graph Variational Autoencoder (GraphVAE) and a Latent Diffusion Model (LDM). The key idea is to leverage the GraphVAE to learn a compact latent representation of molecular graphs, and then apply diffusion modeling in this latent space to improve graph generation. Molecular graphs are typically represented by multiple matrices of varying dimensions - an adjacency matrix encoding connectivity, a node feature matrix, and an edge feature matrix. The GraphVAE is designed to encode these high-dimensional, structured representations into a low-dimensional latent vector. Once trained, the GraphVAE's encoder maps input graphs to latent codes, while its decoder reconstructs graphs from latent codes. After training the GraphVAE, a Latent Diffusion Model is trained on the learned latent space. The LDM applies iterative forward diffusion and reverse denoising processes to refine the generation of latent codes, which can then be decoded into molecular graphs using the GraphVAE decoder. This two-stage approach allows us to leverage the strengths of both VAEs for learning meaningful representations and diffusion models for high-quality generation.

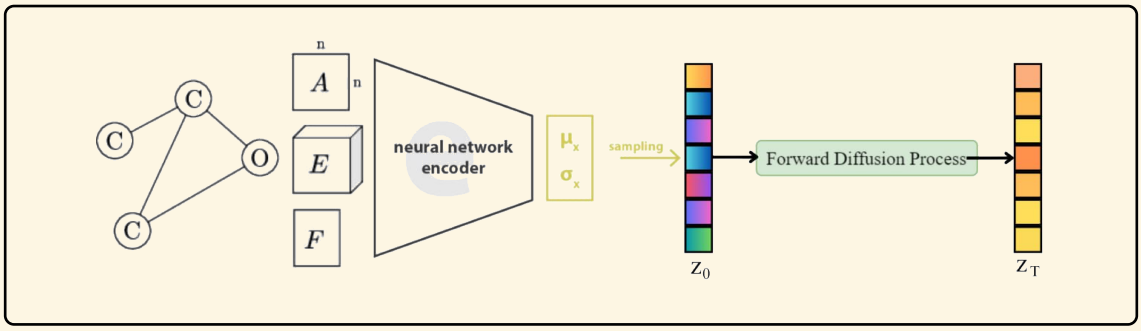


Figure 2: Latent Diffusion Process

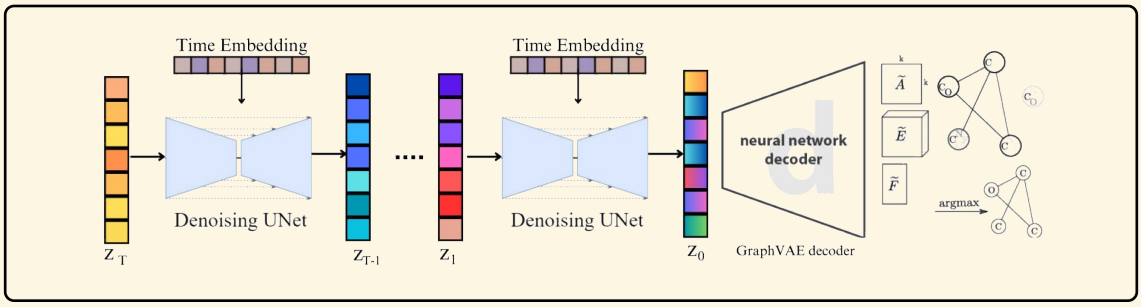


Figure 3: Latent Denoising Process

3.1 Variational AutoEncoder

Variational Autoencoders (VAEs) are a class of generative models that combine the principles of autoencoders with variational inference, enabling the modeling of

complex data distributions. This document provides an in-depth explanation of how VAEs function, including their architecture, mathematical foundations, and training process. Unlike traditional autoencoders, VAEs learn a probabilistic mapping between the input space and a latent space, enabling generative capabilities. The key difference between VAEs and standard autoencoders lies in the nature of the latent space. While standard autoencoders learn a deterministic encoding, VAEs learn a probabilistic encoding, typically modeled as a multivariate Gaussian distribution.

A VAE is composed of two main parts: an **encoder** and a **decoder**. The encoder maps the input data into a latent space, while the decoder reconstructs the input data from this latent space. The encoder and decoder are trained together to minimize the difference between the input data and the reconstructed data.

VAEs are often employed for **data generation**. For instance, a VAE trained on an image dataset can be used to generate new images.

Here are some **advantages** of using VAEs:

- ➡ VAEs can be used to generate new and realistic data.
- ➡ VAEs can learn meaningful representations of data.
- ➡ VAEs can be applied to various tasks, including data generation, data imputation, and dimensionality reduction.

However, there are also some **disadvantages** associated with VAEs:

- ✖ VAEs can be difficult to train.
- ✖ The quality of the data generated by VAEs can vary depending on the task and the dataset.

3.1.1 Encoder

The encoder, denoted as $q_\phi(z|x)$, maps the input data x to a distribution over latent variables z. Following common practice, the distribution is assumed to be normal:

$$q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \sigma^2_\phi(x)I) \tag{1}$$

where $\mu_\phi(x)$ and $\sigma^2_\phi(x)$ are neural networks parameterized by ϕ. The encoder outputs the parameters of this distribution for each input x.

3.1.2 Decoder

The decoder, denoted as $p_\theta(x|z)$, maps latent variables z back to the data space. It defines a probability distribution over the possible corresponding x values:

$$p_\theta(x|z) = f(x; g_\theta(z)) \tag{2}$$

where $g_\theta(z)$ is a neural network parameterized by θ, and f is a Gaussian probability distribution

3.1.3 Loss Function

In line with prevailing academic practices the VAE is trained by maximizing the **evidence lower bound (ELBO)** on the log likelihood of the observed data:

$$\log p(x) \geq \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)||p(z))$$

Here:

- ▲ The first term, $\mathbb{E}_{q(z|x)}[\log p(x|z)]$, represents the **reconstruction loss**, which measures how well the decoder reconstructs x from z.
- ▼ The second term, $D_{KL}(q(z|x)||p(z))$, is the **Kullback-Leibler divergence** between the learned distribution $q(z|x)$ and a prior distribution $p(z)$, often chosen to be a standard normal distribution $\mathcal{N}(0, I)$.

3.1.4 Reparameterization Trick

To enable backpropagation through stochastic nodes (i.e., sampling from distributions),it was used the **reparameterization trick**. Instead of sampling directly from $q(z|x)$, z was expressed as:

$$z = \mu(x) + \sigma(x) \cdot \epsilon$$

where $\epsilon \sim \mathcal{N}(0, I)$. This reformulation allows gradients to propagate through $\mu(x)$ and $\sigma(x)$, allowing the training of encoder.

3.1.5 Training Process

Training involves:

1. **Forward Pass**:
   encoding the input to obtain latent distribution parameters.
   For each input x_i:
   - Compute parameters $(\mu_i, \sigma_i^2) = f_{\text{encoder}}(x_i)$.
   - Sample $z_i = \mu_i + \sigma_i^2 * \epsilon_i$ from a distribution using the reparameterization trick
2. **Reconstruction**: Pass z_i through the decoder to obtain reconstructed output $x_i' = g_{\text{decoder}}(z_i)$.
3. **Loss Calculation**: Compute loss using ELBO and its components, namely
   - For continuous data: Mean Squared Error (MSE)
   - For binary data: Binary Cross-Entropy
   - Calculate KL divergence between distributions.

4. **Backpropagation**: Update weights of both encoder and decoder networks using gradient descent based on total loss.

During inference, new data can be encoded into the latent space by computing $q_\phi(z|x)$. For generation, sample $z \sim p(z)$ and pass it through the decoder to obtain new data samples. The learned latent space captures meaningful features of the data, allowing for smooth interpolation between data points and controlled generation by manipulating the latent variables.

3.2 Graph Variational Autoencoder

In this thesis was used GraphVAE, which extends the standard VAE framework to handle graph-structured data. While a standard VAE operates on fixed-size vectors, the GraphVAE can process and generate variable-sized graphs with complex topologies. Our GraphVAE encoder consists of multiple SAGEConv layers [20], each followed by LayerNorm [21] for stable training. A Set2Set [22] pooling layer aggregates node-level features into a graph-level representation. Finally, linear layers map this representation to the mean and variance parameters of the latent distribution. The decoder comprises several linear layers with LayerNorm, culminating in the reconstruction of three matrices: the adjacency matrix, node feature matrix, and edge feature matrix. These matrices are then used to reconstruct the molecular graph with assistance from the RDKit library.

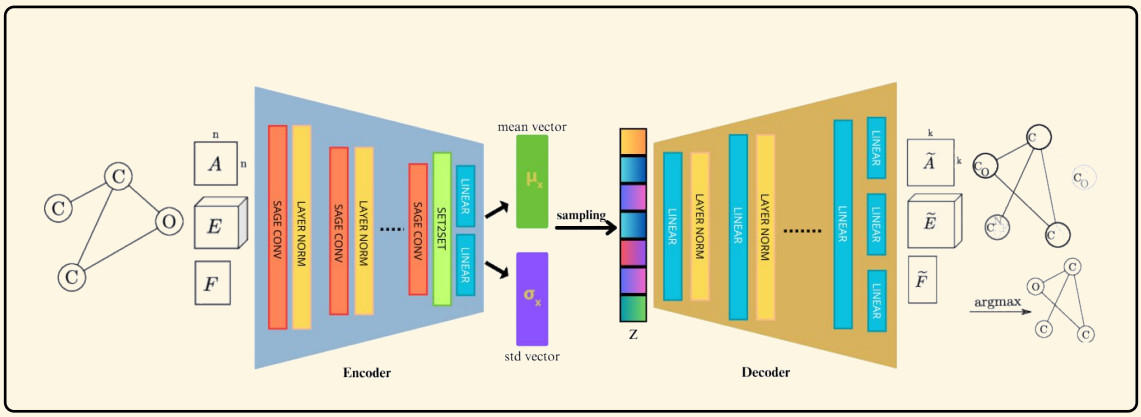


Figure 4: GraphVAE Architecture

The training algorithm for the GraphVAE is as follows:

Algorithm 1 GraphVAE Training

for each epoch **do**
  for each batch of graphs $G_1, \ldots, G_B$ **do**
    Encode graphs to latent distributions: $q_\phi(z|G_i)$
    Sample latent vectors: $z_i \sim q_\phi(z|G_i)$
    Decode latent vectors: $\hat{G}i = f\theta(z_i)$
    Compute reconstruction loss: $L_{recon} = \sum_{i=1}^{B} d(G_i, \hat{G}i)$
    Compute KL divergence: $L_{KL} = \sum_{i=1}^{B} KL(q_\phi(z|G_i)||p(z))$
    Compute total loss: $L = L_{recon} + \beta L_{KL}$
    Update encoder and decoder parameters
  end for
end for

Here, $d(G_i, \hat{G}_i)$ is a graph distance metric, $p(z)$ is the prior distribution (typically standard normal), and β is a hyperparameter balancing reconstruction and regularization. To generate new molecules using the trained GraphVAE:

Algorithm 2 GraphVAE Inference

Sample latent vector $z \sim p(z)$
Decode latent vector: $\hat{G} = f_\theta(z)$
Convert $\hat{G}$ to a valid molecular graph using RDKit
(Optional) Apply post-processing or filtering

This approach allows for efficient sampling of diverse molecular graphs from the learned latent space.

3.3 GraphVAE Loss

The loss function in the original GraphVAE paper is composed of several components to capture different aspects of graph structure. We first describe this original formulation and then present our modifications and alternative approaches.

The original GraphVAE paper proposed a loss function that incorporated graph matching to address the graph isomorphism problem. However, this approach proved computationally expensive and difficult to parallelize. Here, we discuss the original formulation and present alternative loss functions that maintain graph structure while improving efficiency.

3.3.1 Original Reconstruction GraphVAE Loss

Our starting approach to compute the loss was using the reconstruction loss for a graph G and its reconstruction $\tilde{G}$ following the original GraphVAE paper, and was calculated as a combination of node, edge, and graph-level losses:

$$L_{recon} = \lambda_V L_V + \lambda_E L_E + \lambda_F L_F \tag{3}$$

where:

- L_V is the node feature reconstruction loss
- L_E is the edge feature reconstruction loss
- L_F is the graph-level feature reconstruction loss
- λ_V, λ_E, and λ_F are weighting factors

The node and edge losses are computed as:

$$L_V = \frac{1}{n} \sum_i \log F_{i,:}^T \tilde{F}'_{i,:} \tag{4}$$

$$L_E = \frac{1}{(||A||_1 - n)} \sum_{i \neq j} \log E_{i,j,:}^T \tilde{E}'_{i,j,:} \tag{5}$$

where n is the number of nodes and A is the real adjacency matrix. E is the edge feature matrix and F is the node feature matrix. $\tilde{E}'$ and $\tilde{F}'$ are defined exploiting graph matching:

1. First of all we can obtain a binary assignment matrix $X \in 0, 1^{k \times n}$ using a graph matching algorithm, where $X_{a,i} = 1$ only if node $a \in \tilde{G}$ is assigned to $i \in G$ and $X_{a,i} = 0$ otherwise.
2. Than the input adjacency matrix is mapped to the predicted graph as $A' = XAX^T$, whereas the predicted node attribute matrix and slices of edge attribute matrix are transferred to the input graph as:
   - $\tilde{F}' = X^T \tilde{F}$
   - $\tilde{E}'_{:,:,l} = X^T \tilde{E}_{:,:,l} X$

   The graph-level feature reconstruction loss can be computed as:

$$\begin{aligned} L_F = \frac{1}{k} \sum_a A'a, a \log \tilde{A}a, a + (1 - A'a, a) \log(1 - \tilde{A}a, a) + \\ + \frac{1}{k(k-1)} \sum_{a \neq b} A'a, b \log \tilde{A}a, b + (1 - A'a, b) \log(1 - \tilde{A}a, b) \end{aligned} \tag{6}$$

3.3.2 Our Implementation and Modifications

In our implementation, we have made several changes to the loss function:

❶ **Adjacency Loss** L_{adj}: We added an adjacency loss term computed as the Binary Cross Entropy (BCE) between the upper triangular parts of the real and generated adjacency matrices:

$$L_{adj} = \text{BCE}(\text{triu}(A), \text{triu}(\tilde{A})) \tag{7}$$

where A and $\tilde{A}$ are the real and generated adjacency matrices, respectively, and triu$(\cdot)$ denotes the upper triangular part of a matrix.

❷ **Approximate graph-level reconstruction Loss (L_F)**: To address the computational issues of exact graph matching, we developed two types of loss (L_F) to replace the original graph matching algorithm and to ensure that the generated graphs have the same structure as the real ones. We will describe in detail in the next chapters which loss we finally chose as the graph matching approximator.

❸ **Approximate Node and Edge Feature Reconstruction Loss**: In the original paper, graph matching was used to compute $\tilde{F}'$ and $\tilde{E}'$. In our implementation, we relaxed this constraint using $\tilde{F}$ and $\tilde{E}$ and computing the Mean Squared Error (MSE):

$$L_V = \text{MSE}(F, \tilde{F}) \tag{8}$$

$$L_E = \text{MSE}(E, \tilde{E}) \tag{9}$$

The total reconstruction loss in our implementation is a weighted sum of these components:

$$L_{recon} = \alpha L_V + \beta L_E + \gamma L_{adj} + \epsilon L_F \tag{10}$$

where α, β, γ, δ, and ϵ are hyperparameters controlling the contribution of each loss term. This combined loss function aims to capture various aspects of molecular structure, from local node and edge features to global graph properties, while maintaining computational efficiency. The inclusion of the adjacency loss and the approximate graph-level reconstruction loss, in particular, helps to ensure that the generated molecules closely resemble the structural properties of the training data.

3.4 Approximate Graph-level Reconstruction Loss for GraphVAE in Details

3.4.1 Original Graph Matching Loss

In the original GraphVAE paper, the reconstruction loss for a graph G and its reconstruction $\tilde{G}$ was calculated as:

$$L_{recon} = \mathbb{E}_{z \sim q\phi(z|G,y)}[-\log p_\theta(G|z, y)] \tag{11}$$

This loss required finding the best possible matching between G and $\tilde{G}$, which was computationally expensive. The matching procedure had a complexity of $O(m^4)$ for graphs with m nodes.

3.4.2 Approximate Graph Matching Loss

To address the computational issues, an approximate graph matching loss was proposed [23] and used to compute the graph-level feature reconstruction loss (L_F:

$$\begin{aligned} L_F = & \left| \sum_i v_i - \sum_i \tilde{v}i \right|^2 + \left| \sum_{i,j} e_{i,j} - \sum_{i,j} \tilde{e}_{i,j} \right|^2 \\ & + \sum_{b \in single,double,triple} \left| \sum_{i,j} e_{i,j(b)} v_i - \sum_{i,j} \tilde{e}_{i,j(b)} \tilde{v}_i \right|^2 \\ & + \sum_{b \in single,double,triple} \left| \sum_{i,j} e_{i,j(b)} v_i v_j^T - \sum_{i,j} \tilde{e}_{i,j(b)} \tilde{v}_i \tilde{v}_j^T \right|^2 \end{aligned} \tag{12}$$

Where:

- ✳ $v_i \in V$ represents features of the real nodes; while $\tilde{v}_i \in \tilde{V}$ the reconstructed nodes.
- ✳ $e_{i,j} \in E$ represents features of real edge from node i to j; while $\tilde{e}_{i,j} \in \tilde{E}$ the reconstructed edge.

This loss compares the numbers of atom types, bond types, atom-bond pair types, and atom-bond-atom pair types between the original and reconstructed graphs.

3.4.3 Fingerprint-based Loss

The approximate graph matching loss did not yield satisfactory results, as we will see in the next chapters. Therefore we developed a novel approach using molecular fingerprints. This method involves training a neural network to generate fingerprints, inspired by the paper "*Convolutional Networks on Graphs for Learning Molecular Fingerprints*" by Duvenaud et al [24]. The fingerprint network was trained on the QM9 dataset [5], using various types of fingerprints from the RDKit library (e.g., Morgan fingerprints)[25] as reference. Once trained, this network generates fingerprints for both the original and reconstructed molecules. The reconstruction loss is then computed as the Mean Squared Error (MSE) between these fingerprints:

$$L_F = \text{MSE}(FP(G), FP(\tilde{G})) \tag{13}$$

where $FP(\cdot)$ denotes the fingerprint generation function. This fingerprint-based loss offers several advantages:

- ➡ It captures important structural and chemical information about the molecules.
- ➡ It is differentiable, allowing for end-to-end training of the GraphVAE.
- ➡ It is more computationally efficient than graph matching approaches.

By incorporating this fingerprint-based loss, we aim to improve the GraphVAE's ability to learn and preserve important molecular features during the encoding and decoding processes.

3.5 Diffusion Model

Diffusion models (DM) [26] are a class of generative models that have recently gained popularity for their ability to synthesize complex and realistic data. They work through a two-step process:

1. Forward diffusion: The forward diffusion process gradually adds Gaussian noise to the input data in a series of steps, eventually transforming the data into a quasi-Gaussian noise distribution.

2. Reverse diffusion: The reverse diffusion process learns to reverse this noise-adding process, starting with pure noise and gradually denoising it to generate new data similar to the training data distribution.

3.5.1 Latent Diffusion Models

Applying diffusion models directly to the discrete and high-dimensional graph space can be computationally expensive and lead to poor distribution coverage. To address this problem, latent diffusion models have been introduced to perform the diffusion process in a continuous and low-dimensional latent space.

Latent Diffusion Models (LDMs) differ from standard diffusion models by operating in a learned latent space rather than the original data space. This approach offers several benefits:

- Reduced computational complexity: By working in a lower-dimensional latent space, LDMs require fewer resources for training and inference.

- Improved sample quality: The latent space can capture more meaningful representations, leading to higher-quality generated samples.

- Flexibility: LDMs can be combined with various types of autoencoders, allowing for task-specific optimizations.

3.5.2 Architecture

Our LDM architecture is based on a U-Net structure [27], modified to work with latent representations:

- The U-Net consists of linear layers instead of convolutional layers, as we're working with vector representations rather than images.

- Time embedding is incorporated to condition the model on the noise level at each step.

- Residual connections are used to facilitate gradient flow through the network.

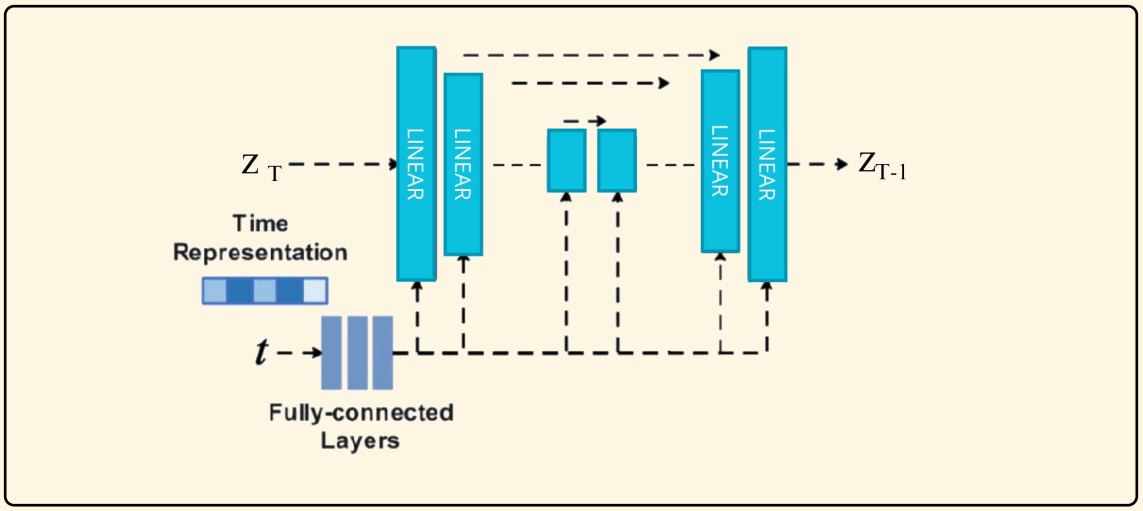


Figure 5: Latent Diffusion Architecture

The model can be described as:

$$\epsilon_\theta(z_t, t) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d \tag{14}$$

where z_t is the noisy latent vector at time step t, and ϵ_θ predicts the noise to be removed.

3.5.3 Training Algorithm

The training process for our LDM leverages the pre-trained GraphVAE:

Algorithm 3 Latent Diffusion Model Training

Require: ϵ_θ as the LDF network with θ parameters
Require: Pre-trained GraphVAE encoder E_ϕ
Require: Training dataset of molecules $\{G_1, \ldots, G_N\}$
for each epoch **do**
  for each batch of molecules $\{G_1, \ldots, G_B\}$ **do**
    Encode molecules to latent space: $z_i = E_\phi(G_i)$
    Sample timesteps $t \sim \text{Uniform}(\{1, \ldots, T\})$
    Sample noise $\epsilon \sim \mathcal{N}(0, I)$
    Compute noisy latents $z_t = \sqrt{\bar{\alpha}_t} z + \sqrt{1 - \bar{\alpha}_t}\epsilon$
    Predict noise $\hat{\epsilon} = \epsilon_\theta(z_t, t)$
    Compute loss $L = \|\epsilon - \hat{\epsilon}\|^2$
    Update θ to minimize L
  end for
end for

Here, $\bar{\alpha}_t$ represents the cumulative product of noise scale factors up to time t.

3.5.4 Inference

To generate new molecules using the trained LDM:

Algorithm 4 Latent Diffusion Model Inference

Require: Trained LDM ϵ_θ
Require: Pre-trained GraphVAE decoder D_ψ
1: Sample initial noise $z_T \sim \mathcal{N}(0, I)$
2: **for** $t = T$ to 0 **do**
3: Predict noise $\epsilon_t = \epsilon_\theta(z_t, t)$
4: Compute $z_{t-1} = \frac{1}{\sqrt{\alpha_t}}(z_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t) + \sigma_t \xi$
where $\xi \sim \mathcal{N}(0, I)$ and σ_t is a small noise term
5: **end for**
6: Decode latent to molecule: $G = D_\psi(z_0)$
7: (Optional) Apply post-processing or filtering

3.6 Data Generation

Generating new graphs with latent diffusion models involves sampling a point from the latent space and passing it through the decoder. The decoder then generates a graph that corresponds to this point in the latent space.

3.6.1 Sampling from Latent Space

The generation of new molecules begins with sampling from the learned latent space. To achieve acceptable results, we employ a progressive sampling strategy that explores the latent space more thoroughly.

Initially, we sample from a standard Gaussian distribution:

$$z_0 \sim \mathcal{N}(0, I) \tag{15}$$

For subsequent iterations, we incrementally shift the sampling space by adjusting the variance while keeping the mean constant:

$$z_i \sim \mathcal{N}(0, (1 + i\Delta)I) \tag{16}$$

where i is the iteration number and Δ is a small increment. This approach allows us to systematically explore a wider range of the latent space, potentially uncovering more diverse molecular structures.

3.6.2 Threshold-based Molecule Generation

In the process of generating new molecules, we introduce two critical thresholds that control different aspects of the molecular structure:

✳ Diagonal Threshold (τ_d): This threshold determines the activation of elements on the diagonal of the generated adjacency matrix. It effectively controls the number of atoms in the generated molecule. Elements on the diagonal exceeding τ_d are considered as active atoms.

✳ Adjacency Threshold (τ_a): This threshold regulates the off-diagonal elements of the adjacency matrix, determining the connections and bonds between atoms. Elements exceeding τ_a are interpreted as bonds between the corresponding atoms.

The generation process can be summarized as:

$$\tilde{A}'_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and } \tilde{A}_{ij} > \tau_d \\ 1 & \text{if } i \neq j \text{ and } \tilde{A}_{ij} > \tau_a \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where $\tilde{A}$ is the raw output from the decoder and $\tilde{A}'$ is the final adjacency matrix.

3.6.3 Threshold Selection and Impact

The values for τ_d and τ_a were determined experimentally to optimize the quality of generated molecules. These thresholds significantly influence the characteristics of the generated molecules:

✳ Diagonal Threshold (τ_d):

- Higher values lead to smaller molecules with fewer atoms.
- Lower values produce larger molecules with more atoms.
- Pros: Allows control over molecular size.
- Cons: Extreme values may lead to unrealistic molecular sizes.

✳ Adjacency Threshold (τ_a):

- Higher values result in sparser molecular graphs with fewer bonds.
- Lower values create denser graphs with more connections.
- Pros: Enables fine-tuning of molecular connectivity.
- Cons: Inappropriate values may lead to chemically invalid structures.

The interplay between these thresholds is crucial. For instance, a low τ_d combined with a high τ_a might generate large but sparsely connected molecules, while the opposite combination could produce small, densely connected structures.

Careful selection of these thresholds is essential to balance:

- Chemical validity: Ensuring the generated structures adhere to chemical rules.

- Diversity: Producing a wide range of molecular structures.
- Realism: Generating molecules that resemble those in the training set.
- Novelty: Creating new, previously unseen molecular structures.

The optimal values for τ_d and τ_a were determined through a combination of random search on a held-out validation set and manual tuning based on qualitative assessment of the generated molecules. Specifically, we evaluated various threshold combinations on the validation set and test set, measuring metrics such as chemical validity and uniqueness. This process allowed us to identify a range of promising threshold values.

It's important to note that while these thresholds provide valuable control over the generation process, they also introduce a level of human bias. The optimal threshold values may vary depending on the specific application or desired properties of the generated molecules. Therefore, it may be beneficial to treat these thresholds as hyperparameters that can be tuned based on the particular requirements of the task at hand.

4 Experiment Setup

4.1 Dataset

For our experiments, we used the QM9 dataset [5], a widely used benchmark dataset in molecular machine learning. QM9 consists of 130,828 small organic molecules with up to 9 heavy atoms (C, N, O, F). Each molecule is represented as a graph, where nodes correspond to atoms and edges to chemical bonds.

The dataset provides the following information for each molecule:

- Node features: Atomic number, atom type, aromaticity, formal charge, etc. (11 features total)
- Edge features: Bond type (single, double, triple, aromatic) (4 features total)

We performed several preprocessing steps to prepare the data for our latent diffusion model:

1. Adjacency matrix: We constructed an adjacency matrix from the provided edge index to explicitly represent the molecular graph structure.
2. Fixed-size padding: All matrices (adjacency, node features, edge features) were padded to a fixed size of 9x9, corresponding to the maximum number of heavy atoms. This allows for efficient batching during training and inference.
3. Edge feature padding: The edge feature tensor was padded to accommodate the maximum possible number of edges, calculated as $N(N-1)/2$ where $N = 9$.
4. One-hot encoding: Node and edge features were one-hot encoded. This is a common practice in machine learning as it removes any implied ordinality in categorical features and allows the model to learn the importance of each category independently.
5. Sanitization: We removed invalid molecules by verifying and sanitizing each structure using RDKit, a cheminformatics library [25]. This ensures that all molecules in our dataset are chemically valid.
6. Hydrogen removal: Hydrogen atoms were removed to reduce computational complexity. This is a common simplification in molecular modeling as the positions of hydrogen atoms can often be inferred from the heavy atom structure.

After preprocessing, our final dataset consisted of at least 100'000 valid molecules, each represented by fixed-size graph tensors and associated SMILES strings.

4.2 Evaluation Metrics

To assess the performance of our latent diffusion model for molecular graph generation, we employed several established metrics:

4.2.1 Validity, Uniqueness, and Novelty

- **Validity**: The fraction of generated molecules that are chemically valid. A molecule is considered valid if it can be parsed by RDKit and passes a series of chemical sanity checks.

$$\text{Validity} = \frac{\text{Number of valid molecules}}{\text{Total number of generated molecules}}$$

- **Uniqueness**: The fraction of unique molecules among the valid generated molecules. This metric ensures that the model is not simply memorizing and reproducing training examples.

$$\text{Uniqueness} = \frac{\text{Number of unique valid molecules}}{\text{Number of valid molecules}}$$

- **Novelty**: The fraction of valid generated molecules that are not present in the training set. This measures the model's ability to generate novel chemical structures.

$$\text{Novelty} = \frac{\text{Number of valid molecules not in training set}}{\text{Number of valid molecules}}$$

These metrics provide a comprehensive evaluation of the model's ability to generate diverse, valid, and novel molecular structures.

4.2.2 Molecular Property Distribution

We also compared the distribution of key molecular properties between the generated molecules and the real dataset:

- **QED (Quantitative Estimate of Drug-likeness)**: A measure between 0 and 1 that quantifies the likelihood of a compound being a viable drug candidate based on its structural properties.
- **LogP (Octanol-Water Partition Coefficient)**: A measure of a compound's lipophilicity, which is crucial for predicting its behavior in biological systems and its potential as a drug.
- **Molecular Weight**: The sum of the atomic weights of all atoms in a molecule. This property is important in drug design as it affects a compound's absorption and distribution in the body.
- **Ring Count**: The number of rings in a molecule. Ring structures are common in drugs and can significantly influence a compound's properties and behavior.

To quantitatively compare the distributions of these properties between the generated molecules and the real dataset, we employed the Fréchet distance. The Fréchet distance measures the similarity between two distributions, with lower values indicating more similar distributions. The Fréchet Distance is defined as:

$$FD(p, q) = \inf_{\alpha} \int_0^1 d(p(\alpha(t)), q(t))dt$$

where p and q are the two distributions being compared, α is a reparameterization of p, and d is a distance metric in the space of the distributions. We calculated this distance for each of the above properties (QED, LogP, Molecular Weight, and Ring Count) against to the real dataset (QM9), providing a comprehensive assessment of how well our model captures the distribution of these important molecular characteristics.

This approach allows us to quantify how well the generated molecules match the property distributions of real molecules, providing insight into the model's ability to capture the underlying chemical space.

The combination of these metrics offers a multi-faceted evaluation of our model's performance. While validity, uniqueness, and novelty assess the fundamental quality and diversity of the generated molecules, the property distribution comparisons provide a more clear view of how well the model captures the characteristics of real molecular structures.

It's worth noting that these metrics, while comprehensive, may not capture all aspects of molecular generation quality. For instance, they don't directly measure the synthesizability or stability of the generated molecules. Future work may explore additional metrics to address these aspects, potentially incorporating domain-specific knowledge from chemistry and pharmacology.

4.3 Experimental Comparisons

Our experimental process involved two main phases: initial model development and subsequent comparative analysis.

4.3.1 Initial Model Development

We began with an extensive study of network architectures and hyperparameters to establish a robust baseline model. This phase involved exploring various aspects of the model, including:

- Network structure (number and size of layers)
- Learning rate and optimization algorithms
- Batch size and training duration
- Regularization techniques (e.g., dropout)

- Encoder/decoder architecture: We experimented with different architectures for the encoder and decoder networks, including varying depths and widths.
- Diffusion process parameters: We explored variations in the number of diffusion steps and noise scheduling, which can affect the trade-off between generation quality and computational efficiency.

These parameters were tuned through a manual adjustment based on performance on a held-out validation set.

4.3.2 Comparative Analysis

After establishing a baseline model, we conducted a series of experiments to analyze the impact of various model components and design choices. These comparisons were chosen based on their potential to significantly influence molecular generation quality, for example comparing different molecular fingerprint types and varying the size of the latent space.

5 Results

In this section, we present a selection of the most significant experimental results obtained from our proposed model for molecular graph generation. While numerous experiments were conducted, we focus here on the key findings that demonstrate the efficacy and capabilities of our approach.

Our experiments were carried out on two subsets of the QM9 dataset: one containing 50,000 examples and another with 100,000 examples. The experimental procedure involved first training a GraphVAE model, followed by the training of a Latent Diffusion Model using the encoder and decoder from the pre-trained GraphVAE.

5.1 Performance of the Baseline LDM

We began with a baseline model after an initial exploratory phase. This baseline model consists of a GraphVAE with a latent space dimension of 80 and a Latent Diffusion Model. It incorporates a fingerprint loss trained using Morgan fingerprints with a size of 128. Based on empirical testing and loss curve analysis, we selected 30 epochs for training the GraphVAE and 50 epochs for training the Latent Diffusion Models.

5.1.1 GraphVAE Training

The GraphVAE was trained for 30 epochs, as suggested in Figure 6. The training loss curve demonstrates a steady decrease, indicating effective learning. The total loss, which is a combination of reconstruction loss, KL divergence, edge loss, node loss, and adjacency loss, stabilizes around 0.2 by the end of training. An interesting thing is that the largest contribution is given by the loss regarding the node features while the adjacent and edge loss is practically constant throughout the training. The loss trend reported in the figure is similar for every other experiment. Notably, the validity of generated molecules, as measured on the validation set, shows significant improvement over the course of training, reaching approximately 0.65 by the final epoch. Conversely, uniqueness tends to decrease as training progresses.

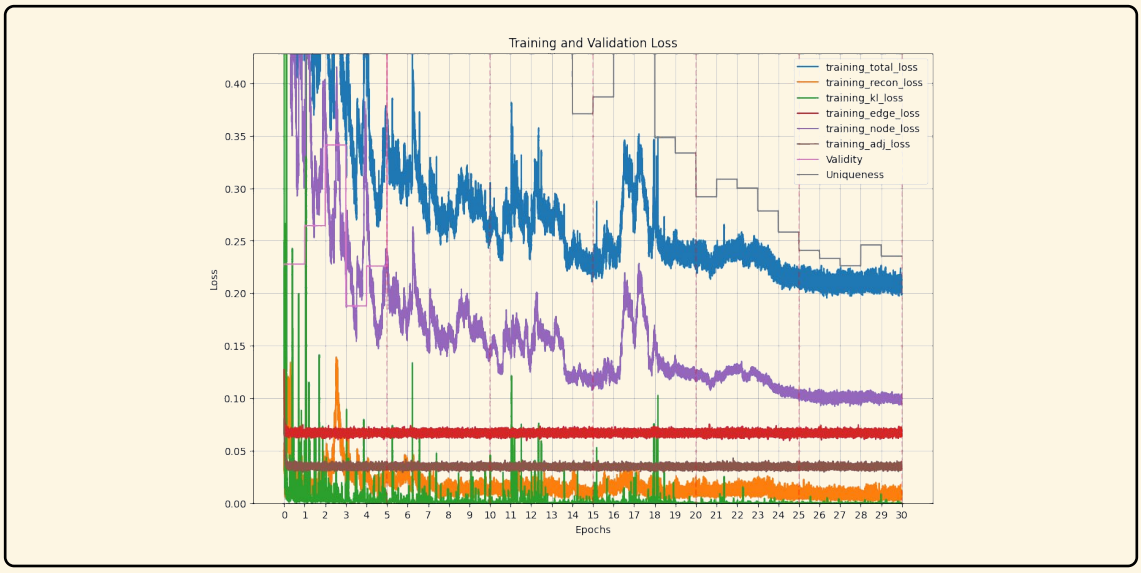


Figure 6: Training Trend GraphVAE Baseline with dataset size = 50'000

5.1.2 Latent Diffusion Model Training

The LDM, trained for 50 epochs using the pre-trained GraphVAE's encoder and decoder, shows a consistent decrease in total loss (Figure 7). The loss stabilizes around 0.6-0.7 by the end of training, suggesting convergence. The validity of the generated molecules slightly decreases throughout training, while uniqueness peaks around 25 epochs and then it also decreases.

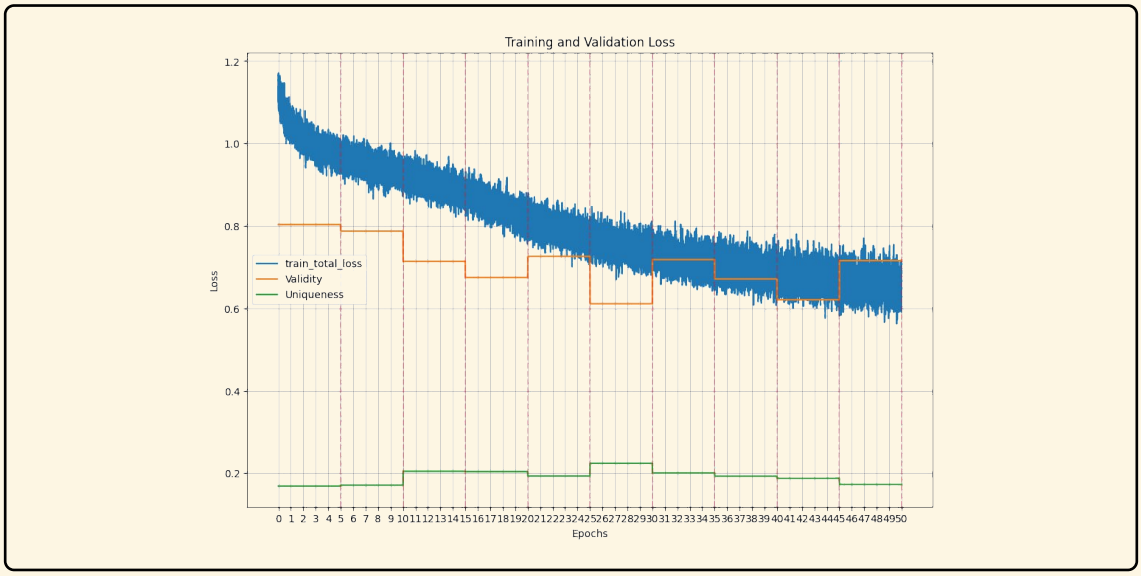


Figure 7: Training Trend Diffusion Baseline with dataset size = 50'000

5.1.3 Generation Performance

To evaluate the generation performance of the proposed model, we compared the proposed architecture (LDM), the GraphVAE of LDM (without latent diffusion) and the results published in (Baseline GraphVAE). We report also the results published when graphVAE has been proposed (original GraphVAE) [6]. Table 1 shows the results obtained with 10,000 molecules.

Metric	Baseline GraphVAE	LDM	Original GraphVAE
Validity	78.52%	77.57%	57.1%
Uniqueness	17.65%	18.81%	52.0%
Novellty	80.81%	80.05%	71.9%
QED (std)	0.484 (0.050)	0.487 (0.054)	-
LogP (std)	2.073 (1.125)	1.938 (1.123)	-
Mol. Weight (std)	106.285 (15.199)	105.833 (15.682)	-
Ring Count (std)	0.821 (0.683)	0.853 (0.803)	-

Table 1: Generation performance comparison between GraphVAE and LDM with 50k training samples

Both our models demonstrate good performance in generating valid molecules, with the GraphVAE slightly outperforming the LDM (78.52% vs 77.57%). However, the LDM generates a higher percentage of unique molecules (18.81% vs 17.65%) but the results of this metric are lower than the original implementation (52.0%). Both models show high novelty, with over 80% of the unique molecules being novel, outperforming original GraphVAE implementation (71.9%).

The generated molecules from our both models have similar properties, with slight variations. The LDM produces molecules with marginally higher QED values and ring counts, while the GraphVAE generates molecules with slightly higher LogP values and molecular weights.

5.1.4 Distribution Matching

To assess how well the generated molecules match the distribution of the training set, we calculated the Fréchet distance for various molecular properties where the smaller the distance, the more closely the model approaches the real distribution of the training dataset.:

Fréchet Distance	Baseline GraphVAE	LDM
QED	0.052	0.029
LogP	0.324	0.261
Molecular Weight	0.191	0.298
Ring Count	0.134	0.090

Table 2: Fréchet distances QM09 Dataset for various molecular properties, 50k training samples

The LDM shows better distribution matching for QED, LogP, and Ring Count, while the GraphVAE performs better for Molecular Weight. These results suggest that the LDM is generally more effective at capturing the distribution of the training set, particularly for key drug-like properties such as QED and LogP.

5.1.5 Comparison of GraphVAE and LDM with 100,000 Molecule Training Set

Metric	Baseline GraphVAE	LDM	Original GraphVAE
Validity	86.52%	82.89%	57.1%
Uniqueness	6.97%	8.51%	52.0%
Novellty	83.91%	84.54%	71.9%
QED (std)	0.542 (0.044)	0.538 (0.047)	-
LogP (std)	2.711 (0.745)	2.704 (0.784)	-
Mol. Weight (std)	124.006 (9.903)	123.516 (10.747)	-
Ring Count (std)	0.818 (0.601)	0.699 (0.656)	-

Table 3: Generation performance comparison between GraphVAE and LDM with 100k training samples

Comparison with Original GraphVAE Paper Comparing our results to the unconditional generation results from the original GraphVAE paper:

- Validity: Our implementation achieves a significantly higher validity rate (86.52% vs 57.1% in the original paper). This improvement could be attributed to the larger dataset and potential modifications in the implementation.
- Uniqueness: The original paper reported a uniqueness of 52.0% among valid molecules, which is higher than our result (6.97%). This suggests that our model might be overfitting to the training data more than the original implementation.
- Novelty: Our model achieves a higher novelty rate (83.91% vs 71.9% in the original paper), indicating better generalization to new molecular structures.

5.1.6 Impact of Increased Dataset Size

Comparing the results of models trained on 100,000 molecules to those trained on 50,000 molecules:

- Validity: Both GraphVAE and LDM show improved validity rates with the larger dataset.
- Uniqueness: Both models show a decrease in uniqueness, suggesting some level of overfitting with the larger dataset.

- Novelty: Novelty rates remain high for both models, with slight improvements for the larger dataset.
- Property Distributions: The larger dataset leads to generated molecules with properties more closely matching the training set, as evidenced by the generally lower Fréchet distances.

In conclusion, increasing the dataset size to 100,000 molecules has led to improvements in validity and novelty for both models, but at the cost of reduced uniqueness. The LDM shows better performance in generating diverse molecules, while the GraphVAE excels in validity. Both models show improvements over the original GraphVAE paper in terms of validity and novelty, but struggle with uniqueness, suggesting a need for strategies to combat overfitting in future work.

5.2 Ablation Study with Variants of the Baseline LDM

To explore the impact of different design choices and hyperparameters, we conducted a series of experiments based on our baseline model. For each variant, we trained both the GraphVAE and the Latent Diffusion Model (LDM) using 50,000 and 100,000 training molecules. The following subsections describe each variant and its impact on model performance.

- **Without Fingerprint Loss**: In this variant, we removed the fingerprint loss from the training objective. The fingerprint loss was initially included to encourage the model to capture important structural features of the molecules. Removing the fingerprint loss likely reduced the model's ability to capture fine-grained structural information. This could lead to less chemically realistic molecules being generated, but might also allow for more diverse outputs as the model is less constrained.
- **Increased Fingerprint Dimension**: We increased the fingerprint dimension from 128 to 2048 elements. This change provides a more detailed molecular representation, potentially capturing more structural information. The larger fingerprint dimension could lead to improved capture of molecular features, potentially resulting in more realistic and diverse molecule generation. However, it may also increase computational complexity and could lead to overfitting if not properly regularized.
- **RDKit Base Fingerprint**: Instead of using Morgan fingerprints, we employed the base RDKit fingerprint with a dimension of 128. This change alters the type of molecular features being captured by the fingerprint. More precisely, the RDKit base fingerprint captures different structural information compared to Morgan fingerprints. This could lead to changes in the types of molecules generated, potentially emphasizing different chemical properties or structural elements.

- **Approximate Graph Matching Loss**: We use the approximate graph matching loss instead of the fingerprint loss used in the baseline. This change aims to reduce computational complexity while still maintaining structural consistency. The approximate matching could speed up training and potentially allow for scaling to larger molecules. However, it might lead to less precise structural matching, which could affect the quality of generated molecules.

- **Reduced Latent Space Dimension**: We reduced the latent space dimension from 80 to 20. This change constrains the model's representational capacity, forcing it to learn a more compact encoding of molecular structures. A smaller latent space could lead to faster training and generation times, and might force the model to learn more efficient representations. However, it could also limit the diversity of generated molecules and make it harder to capture complex structural relationships.

- **Increased Latent Space Dimension**: We increased the latent space dimension from 80 to 120. This expansion gives the model more capacity to represent molecular structures and their relationships. The larger latent space could allow for more nuanced representations of molecules, potentially leading to higher quality and more diverse outputs. However, it might also increase the risk of overfitting and require more computational resources for training and generation.

This comprehensive approach allows us to understand not only the impact of each modification but also how these impacts scale with dataset size.

5.3 Experimental Results on the Variants of LDM

We conducted a series of experiments to evaluate the performance of our GraphVAE and Latent Diffusion Model (LDM) under various configurations. Table 4 summarizes the performance of the best models for each version, compared to the original GraphVAE paper results.

Across all experiments, we observed the following main patterns:

- Increasing the dataset size from 50,000 to 100,000 molecules generally improved validity but often reduced uniqueness.
- The LDM showed more robustness to changes in model architecture and hyperparameters compared to the GraphVAE.
- Both models struggled with the approximate graph matching loss, highlighting the importance of precise structural representation.
- Larger fingerprint and latent space dimensions tended to improve validity at the cost of reduced uniqueness and novelty.

Model	Validity (%)	Uniqueness (%)	Novelty (%)	Dataset Size
Original GraphVAE	57.1	52.0	71.9	130k
Base Model GraphVAE	78.52	17.65	80.81	50k
Base Model LDM	77.57	18.80	80.05	50k
No Fingerprint GraphVAE	81.03	18.80	87.66	50k
No Fingerprint LDM	76.93	21.00	87.56	50k
Fingerprint 2048 GraphVAE	75.4	9.46	78.01	50k
Fingerprint 2048 LDM	72.98	9.87	76.83	50k
RDKit Fingerprint GraphVAE	71.77	23.14	90.84	50k
RDKit Fingerprint LDM	68.06	27.12	91.22	50k
Approx. Loss GraphVAE	6.62	69.03	95.84	50k
Approx. Loss LDM	6.62	69.03	95.84	50k
Latent Space 20 GraphVAE	66.42	24.63	92.23	50k
Latent Space 20 LDM	84.28	10.89	86.27	50k
Latent Space 120 GraphVAE	53.51	28.01	81.45	50k
Latent Space 120 LDM	40.05	41.39	82.14	50k

Table 4: GraphVAE vs. LDM performance comparison

These results underscore the delicate balance between generating valid molecules and maintaining diversity in the output. They also highlight the importance of carefully tuning model components and hyperparameters for optimal performance in molecular graph generation tasks.

5.4 Detailed Analysis of the ablation results

Our extensive experiments with various model configurations revealed several intriguing results that warrant further discussion. These findings not only confirm some of our initial hypotheses but also provide new insights into the behavior of graph generation models.

5.4.1 Approximate Loss Function

One of the most striking results came from the experiment using the approximate graph matching loss. This modification led to a dramatic decrease in the validity of generated molecules for both GraphVAE and LDM:

- **GraphVAE**: Validity dropped to 1.14% (100k dataset) and 6.32% (50k dataset)
- **LDM**: Validity plummeted to 2.07% (100k dataset) and 6.62% (50k dataset)

These results emphatically confirm our earlier assertions about the unsuitability of approximate loss functions for molecular graph generation. The extreme drop in validity suggests that precise graph matching is crucial for maintaining the structural integrity of generated molecules. Interestingly, while validity suffered, uniqueness and novelty scores were exceptionally high:

- **GraphVAE (100k)**: 50% uniqueness, 92.98% novelty

- **LDM (100k)**: 32.37% uniqueness, 95.52% novelty

This indicates that the approximate loss allows for more diverse outputs, but at the cost of chemical validity. Such a trade-off is unacceptable for practical applications in molecular design, reinforcing the need for precise structural representation in graph generation models. But these results are probably conditioned by the fact that since few valid molecules are generated this means that the total number of unique and novel molecules is very low in proportion, distorting the results.

5.4.2 Impact of Latent Space Dimension

Our experiments with varying latent space dimensions provided valuable insights into the role of representational capacity in molecular graph generation:

1. **Validity vs. Diversity Trade-off**: Increasing the latent space dimension generally improved validity but at the cost of reduced uniqueness. This suggests that larger latent spaces may lead to overfitting, where the model becomes proficient at reproducing common structures but loses some ability to generate diverse molecules.

2. **Model-Specific Effects**: The GraphVAE showed a more pronounced improvement in validity with increased latent space, while the LDM demonstrated more resilience to changes in latent dimension.

3. **Optimal Dimension**: The results suggest that the original 80-dimensional latent space may be near-optimal, balancing validity and diversity. Further increases in dimension yield diminishing returns and may even be detrimental to overall performance.

4. **Compression vs. Expressiveness**: The reduced latent space (20 dimensions) still maintained reasonable performance, particularly for the LDM. This indicates that even a compact latent representation can capture essential molecular features, though with some loss in generative capability.

5.4.3 Overfitting in Larger Datasets

The phenomenon of reduced uniqueness in models trained on larger datasets suggests a potential overfitting issue. This is a crucial finding that warrants further discussion:

- **Validity-Diversity Trade-off:** The increase in validity coupled with a decrease in uniqueness for larger datasets indicates that the models are becoming more proficient at reproducing common molecular structures from the training set. While this improves the chemical validity of outputs, it comes at the cost of reduced structural diversity.

- **Memorization vs. Generalization:** The decrease in uniqueness suggests that models trained on larger datasets may be memorizing specific molecular structures rather than learning generalizable rules for molecule generation. This is a classic sign of overfitting, where the model performs well on data similar to its training set but struggles to generate novel structures.
- **Impact on Novelty:** Interestingly, the novelty scores remained relatively high even for models trained on larger datasets. This suggests that while the models may be overfitting to certain structural patterns, they are still capable of combining these patterns in ways not seen in the training data.
- **Implications for Model Design:** These findings highlight the need for careful consideration of dataset size in molecular generation tasks. Larger datasets do not necessarily lead to better generative models, especially when diversity is a key objective.

5.5 The Best LDM Model

After extensive experimentation with various model configurations, we identified the best performing models for both GraphVAE and Latent Diffusion Model (LDM). These models use the fingerprint loss trained with the RDKit base fingerprint instead of Morgan fingerprint and were trained on a dataset of 50,000 molecules.

Table 5 summarizes the performance of the best-balanced models, compared to the baseline models and the original GraphVAE results.

Model	**Dataset Size**	**Validity** (%)	**Uniqueness**(%)	**Novelty**(%)
Original GraphVAE	130k	57.1	52.0	71.9
GraphVAE Baseline	50k	78.52	17.65	80.81
GraphVAE Best	50k	71.77	**23.14**	**90.85**
LDM Baseline	50k	77.57	18.81	80.05
LDM Best	50k	68.06	**27.12**	**91.22**

Table 5: Performance comparison of best-balanced models with baselines and original GraphVAE

5.5.1 Key Findings

- ➡ **Validity:** While our best-balanced models show slightly lower validity compared to the baselines, they still significantly outperform the original GraphVAE (71.77% and 68.06% vs 57.1%).
- ➡ **Uniqueness:** The best-balanced models demonstrate notably higher uniqueness compared to both the baselines and the original GraphVAE, especially when considering the Unique/Total metric.

➡ **Novelty:** Our models achieve substantially higher novelty rates than the original GraphVAE and the baselines, indicating better generalization to new molecular structures.

➡ **Optimal Dataset Size**: The 50,000 molecule dataset strikes a balance between providing sufficient data for learning and avoiding overfitting. This is evident in the higher uniqueness scores compared to models trained on 100,000 molecules.

1. **Overall Balance:** The best-balanced models show a more even distribution across all three metrics, suggesting a better trade-off between generating valid, unique, and novel molecules.

5.5.2 Impact of RDKit Fingerprints

Both the GraphVAE and LDM models achieved their best-balanced performance using RDKit fingerprints with 128 dimensions. This modification appears to strike a good balance between structural representation and model flexibility, leading to:

❖ A slight decrease in validity, likely due to less constrained generation.

❖ A substantial improvement in novelty, indicating better generalization beyond the training set.

5.5.3 Comparison with 100,000 Molecule Models

We will now analyze in detail the performance of GraphVAE and Latent Diffusion models using RDKit fingerprints trained on 50,000 and 100,000 molecules.

Metric	**GraphVAE** 50k	**GraphVAE 100k**	**LDM 50k**	**LDM 100k**
Valid Fraction	0.7177	0.8681	0.6806	0.8371
Unique Fraction	0.2314	0.0814	0.2712	0.0949
Novel Fraction	0.9085	0.8034	0.9122	0.8174
Valid/Total	0.7177	0.8681	0.6806	0.8371
Unique/Total	0.1661	0.0707	0.1846	0.0794
Novel/Total	0.1509	0.0568	0.1684	0.0649
Avg QED (stdev)	0.4786 (0.0499)	0.5467 (0.0492)	0.4774 (0.0575)	0.5295 (0.0494)
Avg logP (stdev)	2.2084 (1.0382)	2.5415 (0.7624)	2.0388 (1.1215)	2.6147 (0.8411)
Avg Mol.Wt. (stdev)	106.44 (14.86)	122.15 (12.01)	105.54 (16.28)	119.54 (13.76)
Avg Ring C. (stdev)	0.9192 (0.7117)	0.7803 (0.5710)	0.9356 (0.8658)	0.4948 (0.6595)
Fréchet Dist. QM9	0.0567	0.0687	0.0379	0.0565
Fréchet Dist. LogP	0.3801	0.3330	0.2602	0.1997
Fréchet Dist. Mol.Wt.	0.3856	0.7280	0.2702	0.3925
Fréchet Dist. Ring	0.1941	0.2748	0.0959	0.1443

Table 6: Detailed comparison of Best GraphVAE and Latent Diffusion models

Observations:

- **Validity:** Both models show improved validity when trained on 100k molecules (GraphVAE: 71.77% to 86.81%, LDM: 68.06% to 83.71%).
- **Uniqueness:** There's a significant drop in uniqueness for both models with the larger dataset (GraphVAE: 23.14% to 8.14%, LDM: 27.12% to 9.49%).
- **Novelty:** Novelty decreases for both models with the larger dataset, but the drop is less severe than for uniqueness (GraphVAE: 90.85% to 80.34%, LDM: 91.22% to 81.74%).
- **QED and LogP:** Both models generate molecules with higher QED and LogP values when trained on 100k molecules, indicating a shift towards more drug-like and lipophilic compounds.
- **Molecular Weight:** The average molecular weight increases with the larger dataset for both models, suggesting a tendency to generate larger molecules.
- **Ring Count:** Interestingly, the average ring count decreases for both models with the larger dataset, more significantly for LDM.
- **Fréchet Distances:** The results are mixed. For GraphVAE, distances generally increase with the larger dataset, indicating less similarity to the training set. For LDM, LogP and QM9 distances decrease, while molecular weight and ring count distances increase.

Reasons for Metric Deterioration with Larger Training Set:

- ➪ **Overfitting:** The significant drop in uniqueness and novelty suggests that the models might be overfitting to the larger dataset. With more examples, the models may be learning to reproduce specific molecules from the training set rather than generalizing the underlying molecular structure principles.
- ➪ **Mode Collapse:** The decrease in uniqueness could indicate a form of mode collapse, where the models focus on generating a smaller subset of "safe" molecules that are guaranteed to be valid, at the expense of diversity.
- ➪ **Bias Towards Common Structures:** The larger dataset may contain more repetitions of common molecular structures. This could lead the models to focus on these frequent patterns, resulting in less diverse outputs.
- ➪ **Reduced Exploration:** With a larger dataset, the models might not need to "explore" as much during training to produce valid molecules. This could lead to a narrower learned distribution, resulting in less unique and novel outputs.
- ➪ **Optimization for Validity:** The increase in validity suggests that the models are getting better at producing chemically valid structures. However, this improvement seems to come at the cost of diversity and novelty.

- **Dataset Characteristics:** The shift in molecular properties (higher QED, LogP, molecular weight) with the larger dataset suggests that the additional 50k molecules might have different characteristics. The models could be adapting to these characteristics, leading to changes in the generated molecules.
- **Model Capacity:** It's possible that the model architectures don't have sufficient capacity to fully capture the complexity of the larger dataset while maintaining generalization ability.

5.5.4 PCA Visualization Analysis

To further investigate the impact of dataset size on model behavior, we analyzed Principal Component Analysis (PCA) visualizations of the latent space during GraphVAE training for both 50,000 and 100,000 molecule datasets.

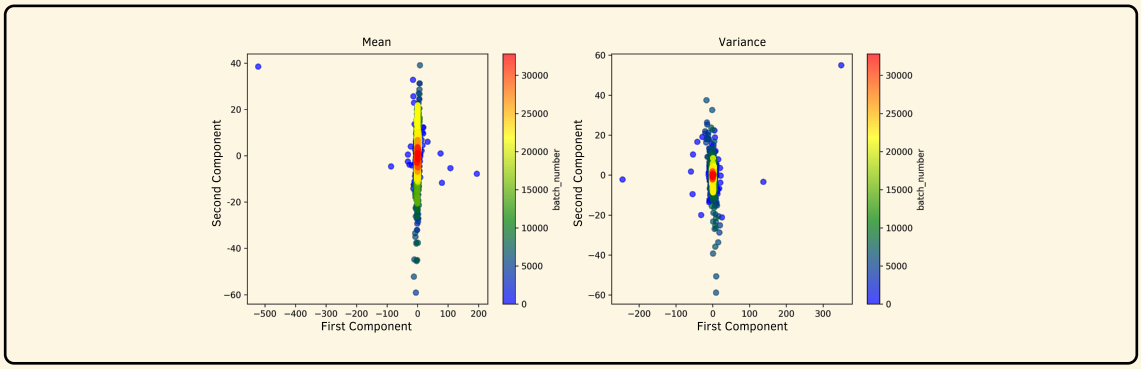


Figure 8: PCA of latent space from Best GraphVAE trained on 50,000 molecules

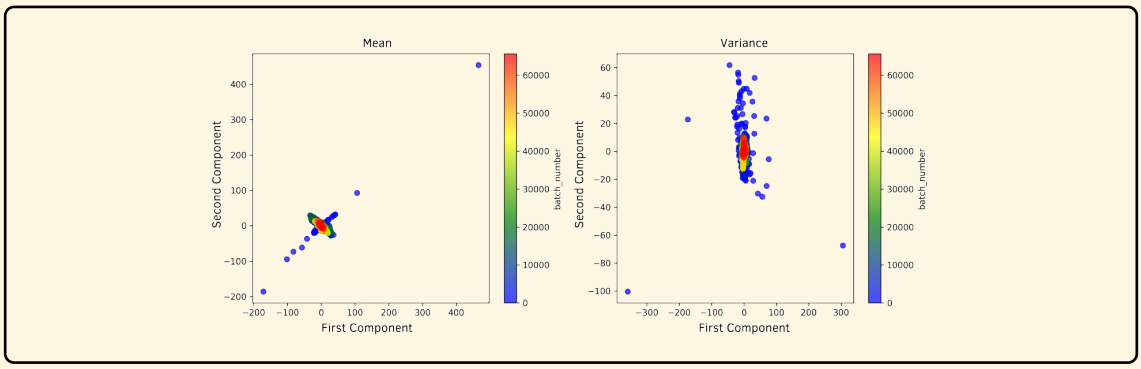


Figure 9: PCA of latent space from Best GraphVAE trained on 100,000 molecules

Figure 8 shows the PCA for the 50,000 molecule dataset, while Figure 9 represents the 100,000 molecule dataset. Key observations include:

- **Cluster Density:** The 100k model shows a denser, more compact clustering in the latent space, particularly evident in the mean distribution plot. This

suggests that the model is learning a more concentrated representation of molecular structures, which may explain the higher validity but lower uniqueness.

- **Variance Distribution:** The variance plots show a more spread-out distribution for the 50k model, especially along the second principal component. This increased variance might contribute to the higher uniqueness and novelty observed in the 50k model.

- **Outliers:** Both models show some outliers, but the 50k model appears to have more widely distributed outliers. This could indicate a greater capacity for generating diverse structures, albeit with a slightly lower validity rate.

These visualizations support our quantitative findings, illustrating how the larger dataset leads to a more constrained latent space representation. While this results in higher validity, it also limits the model's ability to generate diverse and novel structures.

5.6 GraphVAE vs. Latent Diffusion Model

Our experiments revealed that both GraphVAE and Latent Diffusion Model (LDM) achieved their best performance when trained on 50,000 molecules using the RDKit base fingerprint. Here, we compare these top-performing models, discussing their relative strengths and weaknesses, and analyze their ability to capture key molecular properties.

5.6.1 Analysis of Molecular Property Distributions

To further compare the models, we analyzed the distributions of two key molecular properties: LogP and QED.

LogP Distribution Analysis:

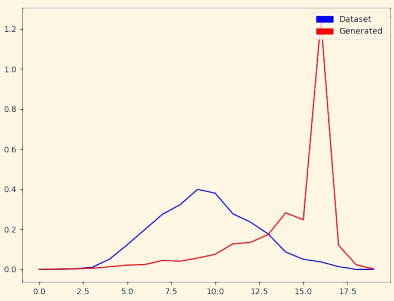
- Both models show a shift towards higher LogP values compared to the QM9 dataset, indicating a tendency to generate more lipophilic molecules.

- The GraphVAE (Figure 10a) shows a more pronounced peak, suggesting it may be overfitting to a particular LogP range.

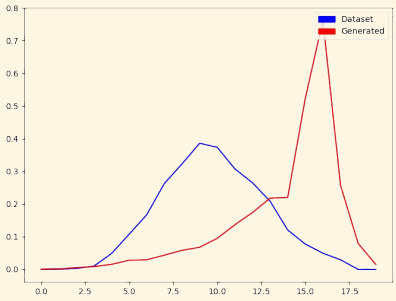
- The LDM (Figure 10b) exhibits a broader distribution, more closely resembling the dataset distribution, which aligns with its higher uniqueness scores.

QED Distribution Analysis:

- Both models show a shift towards higher QED values, suggesting a bias towards generating more drug-like molecules.



(a) LogP Distribution - GraphVAE



(b) LogP Distribution - LDM

Figure 10: LogP Distributions for the Best GraphVAE and LDM compared to QM9 dataset

- The GraphVAE (Figure 11a) demonstrates a sharper peak, indicating a stronger bias towards a specific QED range.
- The LDM (Figure 11b) shows a broader distribution, more closely matching the dataset distribution, particularly in the lower QED ranges.

5.6.2 Pros and Cons

GraphVAE Pros:

➡ Higher validity rate, generating a larger proportion of chemically valid molecules.

➡ Slightly faster training and generation times due to simpler architecture.

GraphVAE Cons:

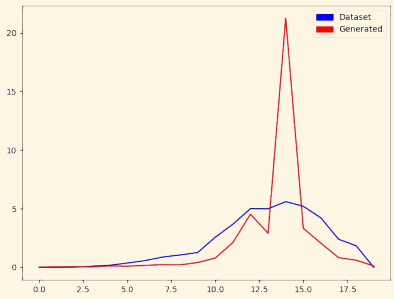
✖ Lower uniqueness, indicating a tendency to generate more repetitive structures.

✖ Slightly lower novelty, suggesting it may be more constrained by the training data.

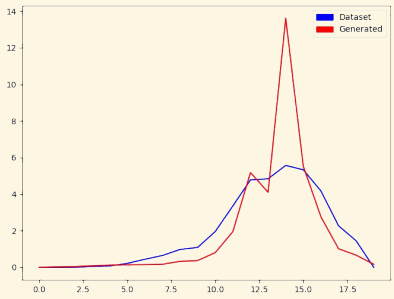
LDM Pros:

➡ Higher uniqueness, demonstrating a better ability to generate diverse molecular structures.

➡ Slightly higher novelty, indicating a stronger capacity for generating structures not seen in the training data.



(a) QED Distribution - GraphVAE



(b) QED Distribution - LDM

Figure 11: QED Distributions for the Best GraphVAE and LDM compared to QM9 dataset

LDM Cons:

- ✖ Lower validity rate, producing a smaller proportion of chemically valid molecules.
- ✖ More complex architecture, potentially leading to longer training and generation times.

5.6.3 Implications and Trade-offs

- **Validity vs. Diversity:** The GraphVAE achieves higher validity at the cost of diversity, while the LDM prioritizes diversity at the expense of some validity. This trade-off is reflected in the property distributions, where the LDM shows broader, more dataset-like distributions.
- **Property Biases:** Both models exhibit biases towards higher LogP and QED values, which may be beneficial for drug discovery applications but could limit their ability to explore the full chemical space represented in the dataset.
- **Generalization:** The LDM's broader property distributions and higher uniqueness/novelty scores suggest better generalization, potentially making it more suitable for discovering novel molecular structures.

In conclusion, while both models show promising performance, they exhibit different strengths and weaknesses. The GraphVAE excels in generating valid molecules but with a narrower range of properties, while the LDM produces more diverse and novel molecules at the cost of some validity. Future work could focus on combining the strengths of both approaches, perhaps through ensemble methods or by incorporating the LDM's diversity-promoting features into the GraphVAE architecture.

5.7 Comparison with State-of-the-Art Methods

To contextualize our results, we compare our best-performing GraphVAE and Latent Diffusion Model (LDM) with state-of-the-art methods, particularly focusing on Graphusion, which has shown exceptional performance on the QM9 dataset. We also include a comparison with the original GraphVAE paper to highlight the advancements in the field.

Method	Validity (%)	Uniqueness (%)	Novelty (%)
GraphVAE (Ours)	**71.77**	**23.14**	**90.85**
LDM (Ours)	**68.06**	**27.12**	**91.22**
Graphusion	97.35 ± 1.21	99.41 ± 0.11	98.49 ± 1.32
Original GraphVAE	57.1	52.0	71.9
GDSS	95.72 ± 1.94	98.46 ± 0.61	86.27 ± 2.29
DiGress	96.03 ± 2.08	98.17 ± 0.53	91.32 ± 1.98

Table 7: Comparison of our models with state-of-the-art methods on QM9 dataset

5.7.1 Analysis of Results

1. **Validity:** Graphusion significantly outperforms our models and even slightly edges out other state-of-the-art methods like GDSS and DiGress. Our GraphVAE shows improvement over the original GraphVAE but falls short of current standards.

2. **Uniqueness:** Graphusion excels in this metric, achieving near-perfect uniqueness. Our models, while showing decent uniqueness, are considerably behind. Interestingly, our GraphVAE's uniqueness is lower than the original, suggesting a potential trade-off made for improved validity.

3. **Novelty:** Graphusion again leads, with our models showing competitive performance. Both our GraphVAE and LDM significantly outperform the original GraphVAE in novelty, indicating substantial progress in generating

5.7.2 Discussion of Differences

- **Graphusion vs. Our Models:** Graphusion's superior performance across all metrics suggests a significant advancement in graph generation techniques. Its ability to maintain high validity while achieving near-perfect uniqueness and novelty is particularly noteworthy. This indicates that Graphusion has successfully addressed the trade-off between validity and diversity that our models still struggle with.

- **Evolution from Original GraphVAE:** Comparing our GraphVAE to the original, we see improvements in validity and novelty but a decrease in uniqueness. This shift suggests that recent advancements have focused on generating

more valid and novel molecules, possibly at the expense of diversity within the valid set.

- **LDM Performance:** Our LDM shows promising results, particularly in novelty, indicating its potential for generating diverse and distribution-matching molecules. However, it lacks validity compared to state-of-the-art methods.
- **Validity-Diversity Trade-off:** While Graphusion seems to have overcome this challenge, our models and even some other state-of-the-art methods like GDSS and DiGress still show signs of this trade-off. Graphusion's approach to balancing these aspects could provide valuable insights for improving our models.

6 Conclusion

In this work we presented a novel graph generation framework that combines variational graph autoencoders with latent diffusion models. Our approach aims to address the limitations of existing graph generation methods by moving the diffusion process from discrete graph space to a continuous latent space. The use of a latent diffusion model allowed for faster training and sampling compared to pixel-based diffusion models.

Our key contributions include:

❶ A graph variational autoencoder to map molecular graphs to a latent space

❷ A latent diffusion model operating in this learned latent space

❸ A Fingerprint loss to enable efficient training and to capture complex information about molecules

Our experiments demonstrate significant improvements over the original GraphVAE, particularly in terms of validity and novelty. The trade-off between these metrics and uniqueness highlights the challenge of balancing the generation of valid, novel, and diverse molecules.

The choice of fingerprint type and size, latent space dimension, and loss function all play crucial roles in model performance. The approximate loss stands out for its high uniqueness and novelty but struggles with validity, suggesting potential for future research in balancing these aspects.

The Latent Diffusion Model shows promise, especially in capturing molecular diversity with smaller datasets. This indicates that the LDM approach may be particularly valuable when working with limited data.

Our experiments demonstrate that using RDKit fingerprints with 128 dimensions significantly improves the balance between validity, uniqueness, and novelty in molecular generation. While there is a small trade-off in validity, the gains in uniqueness and novelty are substantial, resulting in models that generate a more diverse and novel set of valid molecules.

The LDM model shows slightly lower validity but higher uniqueness and novelty compared to the GraphVAE, suggesting it might be better at exploring the chemical space at the cost of some validity.

Interestingly, the best results for both architectures were achieved with the smaller dataset (50,000 molecules), indicating that larger datasets might lead to overfitting or less diverse generation in this context.

The challenges we encountered, particularly in maintaining at the same time chemical validity, uniqueness and novelty in the generated molecules, highlight the inherent difficulties in compressing and reconstructing graph-structured data. However, these challenges also point to exciting avenues for future research, from exploring more sophisticated neural architectures to incorporating domain-specific knowledge.

6.1 Future matters of research

The good performance of Graphusion suggests several areas for improvement in our models:

- **Improved Latent Space Modeling:** Graphusion's ability to maintain high validity, uniqueness, and novelty simultaneously indicates a more effective latent space representation. Investigating its latent space modeling techniques could provide insights for enhancing our GraphVAE and LDM.
- **Advanced Training Strategies:** Graphusion likely employs sophisticated training strategies to achieve its balance of metrics. Incorporating similar strategies, such as advanced regularization techniques or multi-objective optimization, could significantly improve our models.
- **Architectural Enhancements:** The superior performance of Graphusion suggests that its architectural choices are more suited to the graph generation task. Analyzing and potentially adopting some of these architectural elements could boost the performance of our models.

More generally, other tasks and models can be considered starting from the proposed model.

6.1.1 Conditional Graph Generation

An important direction for future work is to explore conditional graph generation. Both the variational graph autoencoder and the latent diffusion model components of our model support conditional generation, but this capability was not tested in the current study. Implementing and evaluating conditional generation could greatly enhance the model's utility for targeted graph design tasks.

6.1.2 Expanded Dataset Exploration

Future work should involve testing on a wider range of datasets, particularly in the field of bioinformatics. For example:

- Protein-protein interaction networks
- Gene regulatory networks
- Metabolic pathway graphs

These datasets could provide valuable insights into the model's performance on more complex biological graph structures.

6.1.3 Application to Other Domains

While this work focused on molecular graphs, it would be interesting to apply these types of model architectures to other domains. One particularly promising area is the use of graph generation for action models in reinforcement learning and planning. This is a relatively unexplored terrain where the ability of the above models to generate diverse, valid graphs could be particularly valuable. For instance, the model could be adapted to generate:

- State transition graphs for complex environments
- Causal models for decision-making processes
- User interface Interaction

6.1.4 Architecture Enhancements

Future work could explore more sophisticated architectures for both the encoder and decoder components of the variational graph autoencoder. This might include:

- Incorporating attention mechanisms to better capture long-range dependencies in graphs
- Exploring hierarchical encoding/decoding strategies for larger graphs
- Investigating graph transformer architectures for improved representational power

6.1.5 Scalability Studies

An important direction for future research is to investigate the scalability to larger and more complex graphs. This could involve:

- Benchmarking performance on graphs with thousands or tens of thousands of nodes
- Developing optimization techniques to handle larger graphs efficiently
- Exploring distributed training approaches for very large graph datasets
- Finding optimizations in the face of power-law that especially affects graphs of real datasets

By pursuing these directions, future work can build upon the foundation laid by this thesis to develop more powerful, flexible, and widely applicable graph generation models.

As the field of AI-driven molecular design continues to evolve, approaches like GraphVAE + LDM that balance efficiency with generative power will become increasingly important. The ability to quickly generate and iterate on molecular designs could accelerate drug discovery, materials science, and other fields relying on molecular innovation.

Our work contributes to the ongoing dialogue in the field of graph machine learning, offering new insights into the use of latent space methods for graph generation. The efficiency gains and the ability to generate diverse, valid graphs are promising steps forward. However, the fact that our method does not consistently outperform state-of-the-art approaches underscores the need for continued research and innovation in this area.

As we conclude, it's important to recognize that the field of graph generation is still in its early stages, with many open questions and challenges. The limitations of our approach serve not as endpoints, but as starting points for future research. They highlight the areas where our understanding and methods need refinement, guiding the way for the next generation of graph generation models.

In the broader context of artificial intelligence and machine learning, the ability to generate complex, structured data like graphs is becoming increasingly important. As we continue to push the boundaries of what's possible in this field, we move closer to systems that can understand and interact with the world in more sophisticated, human-like ways.

We hope that this work will inspire further research in graph generation, latent diffusion models, and their applications across various domains. The journey towards more powerful and flexible graph generation methods is far from over, and we look forward to the innovations and discoveries that lie ahead.

References

[1] D. Meconcelli, "Duccioo/organic-molecule-generation-using-latent-diffusion: Tesi magistrale," GitHub, 2024. [Online]. Available: https://github.com/Duccioo/Organic-Molecule-Generation-using-Latent-Diffusion

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[4] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick *et al.*, "Accurate structure prediction of biomolecular interactions with alphafold 3," *Nature*, pp. 1–3, 2024.

[5] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: A benchmark for molecular machine learning," 2018. [Online]. Available: https://arxiv.org/abs/1703.00564

[6] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*. Springer, 2018, pp. 412–422.

[7] J. Betker, G. Goh, L. Jing, T. Brooks, J. Wang, L. Li, L. Ouyang, J. Zhuang, J. Lee, Y. Guo *et al.*, "Improving image generation with better captions," *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, vol. 2, no. 3, p. 8, 2023.

[8] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022. [Online]. Available: https://arxiv.org/abs/2112.10752

[9] Y. Zhu, Y. Du, Y. Wang, Y. Xu, J. Zhang, Q. Liu, and S. Wu, "A survey on deep graph generation: Methods and applications," in *Learning on Graphs Conference*. PMLR, 2022, pp. 47–1.

[10] "Grl_book," https://www.cs.mcgill.ca/~wlh/grl_book/files/GRL_Book-Chapter_8-Traditional_Graph_Generation.pdf.

[11] D. P. Kingma, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[12] "Variational autoencoder - wikipedia," https://en.wikipedia.org/wiki/Variational_autoencoder.

[13] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[15] T. Bian, Y. Niu, H. Chang, D. Yan, T. Xu, Y. Rong, J. Li, and H. Cheng, "Hierarchical graph latent diffusion model for molecule generation."

[16] X. Fu, Y. Gao, Y. Wei, Q. Sun, H. Peng, J. Li, and X. Li, "Hyperbolic geometric latent diffusion model for graph generation," *arXiv preprint arXiv:2405.03188*, 2024.

[17] R. Yang, B. Yang, S. Ouyang, T. She, A. Feng, Y. Jiang, F. Lecue, J. Lu, and I. Li, "Graphusion: Leveraging large language models for scientific knowledge graph fusion and construction in nlp education," *arXiv preprint arXiv:2407.10794*, 2024.

[18] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "Graphrnn: Generating realistic graphs with deep auto-regressive models," in *International conference on machine learning*. PMLR, 2018, pp. 5708–5717.

[19] L. Kong, J. Cui, H. Sun, Y. Zhuang, B. A. Prakash, and C. Zhang, "Autoregressive diffusion model for graph generation," in *International conference on machine learning*. PMLR, 2023, pp. 17 391–17 408.

[20] "torch_geometric.nn.conv.sageconv — pytorch_geometric documentation," https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.SAGEConv.html, (Accessed on 10/12/2024).

[21] "torch_geometric.nn.norm.layernorm — pytorch_geometric documentation," https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.norm.LayerNorm.html.

[22] "torch_geometric.nn.aggr.set2set — pytorch_geometric documentation," https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.aggr.Set2Set.html.

[23] Y. Kwon, J. Yoo, Y. Choi, W. Son, D. Lee, and S. Kang, "Efficient learning of non-autoregressive graph variational autoencoders for molecular graph generation. j cheminf 11 (1): 1–10," 2019.

[24] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," 2015. [Online]. Available: https://arxiv.org/abs/1509.09292

[25] "Rdkit," https://www.rdkit.org/, (Accessed on 10/11/2024).

[26] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[27] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

7 Acknowledgements (ringraziamenti)

Ringraziamenti in italiano. Questa scelta è stata fatta in considerazione del fatto che i destinatari principali di questi ringraziamenti sono italiani. Spero che queste parole possano esprimere adeguatamente la mia gratitudine e il mio affetto per tutti coloro che hanno contribuito a questo percorso. Quindi ringraziamenti in italiano, perchè sì.

Procrastinazione.
Vorrei iniziare i ringraziamenti ringraziando proprio la *Procrastinazione*. Un mondo dove tutto corre veloce, dove il "fare presto" è diventato la norma, ci invita a riflettere sulla nostra necessità di fermarci. La *Procrastinazione* non è un nemico da sconfiggere, ma un richiamo della nostra mente a prendersi una pausa. In un'epoca dove ogni istante sembra dover essere sfruttato al massimo e lasciare qualcosa a metà è percepito come un fallimento, ricordiamo che non dobbiamo sempre correre. Anche se il tempo è prezioso, non siamo obbligati a concludere tutto in fretta. A volte, lasciare le cose in sospeso è un atto di gentilezza verso noi stessi. Questa tesi rappresenta non solo il risultato di un lungo percorso di studio, ma anche un viaggio personale fatto di pause, riflessioni e momenti in cui lasciare qualcosa in sospeso è stato necessario per assaporare la vita. Non c'è nulla di sbagliato nel ritagliarsi del tempo per sé stessi, nel godersi ogni passo senza fretta. La vita non è una corsa, ed è perfettamente ok lasciare le cose a metà.
Perché, a volte, è proprio nelle pause che troviamo il vero senso di tutto.

Un ringraziamento speciale va anche all'*Ansia*. Spesso vista come una nemica, in realtà ha giocato un ruolo cruciale nel mio percorso. In un mondo dominato dai social media, dove ci confrontiamo costantemente con persone e oggetti irrealistici, l'*Ansia* ci spinge a cercare di essere migliori di quanto realmente siamo. Forse è proprio questo che alimenta la mia ansia: il continuo confronto con immagini perfette e vite idealizzate. Ma è anche grazie a questa *Ansia* che ho trovato la forza di andare avanti, di migliorare e di cercare sempre nuove sfide. Mi ha insegnato a essere resiliente, a non arrendermi di fronte alle difficoltà e a trovare un equilibrio tra ciò che sono e ciò che voglio diventare.
Quindi, grazie Ansia.

E ora, un sincero ringraziamento alle persone che hanno reso possibile questo viaggio accademico.
Un grazie di cuore al professore Franco Scarselli, per l'immensa pazienza e per avermi sempre aiutato a qualsiasi ora e giorno. La sua guida e disponibilità sono state fondamentali per il completamento di questa tesi.
Un ringraziamento speciale ai correlatori Niccolò Pancino e Pietro Bongini, per il loro prezioso supporto e aiuto durante questo percorso. La loro collaborazione mi ha permesso di portare avanti la tesi con determinazione e serenità.

Un gigantesco Grazie agli amici di sempre di Buonconvento, Siena, Montalcino e qualche disgraziato di Ponte d'Arbia
(non mi metto a ringraziarvi singolarmente perchè sennò si fa notte) senza di voi questi anni di studi sarebbero stati mooooolto più lunghi.
Grazie per esserci sempre o quasi...
Grazie agli amici dispersi nel mondo, grazie a quelli che non stanno mai a casa. Grazie ai MammaAmore.
Grazie agli amici persi e ritrovati. A quelli che ci sono sempre stati.

Grazie ai Nonni,
Grazie Nonna Silvana, Nonna Lida, Nonno Giovanni per avermi cresciuto.
Vi sono eternamente grato.

Grazie Mamma
Grazie Babbo
Grazie Guido
Siete sempre stati presenti in ogni occasione, senza mai farmi mancare nulla. Avete sempre creduto in me, anche quando io stesso avevo dei dubbi. Grazie per avermi supportato nei momenti difficili e per aver celebrato con me ogni piccolo successo. Senza di voi, nulla di questo sarebbe stato possibile. *Vi voglio bene, davvero.*

Per finire mi voglio spostare i ringraziamenti nuovamente ad un concetto:
Gli Errori.
Gli errori che ho commesso lungo il cammino. Ogni errore, ogni passo falso, è stato un insegnante prezioso. Senza di loro, non avrei potuto crescere e imparare. Gli errori ci modellano, ci spingono a riflettere, a migliorare, a reinventarci.
Il cambiamento è una costante della vita. Continuiamo a evolverci, a cambiare prospettiva, a scoprire nuovi aspetti di noi stessi. Questo continuo cambiamento rende la vita interessante e ci evita la monotonia. Siamo come argilla che si modella col tempo, e ogni errore è un colpo di scalpello che ci rende più definiti, più veri.

Grazie agli errori per avermi mostrato la strada verso la crescita personale e il miglioramento continuo. È attraverso le cadute che impariamo a rialzarci, e ogni cambiamento, per quanto piccolo, contribuisce a renderci chi siamo oggi.

(E Grazie anche a Zoe e al Gatto.)

SEE YOU SPACE COWBOY...