Berta Reverté Àlvaro

Targeted Molecular Generation driven by Fingerprint Embeddings

Degree Final Project

Supervised by Dra Marta Sales Pardo and Dr Roger Guimerà Manrique

Bachelor's degree in Biomedical Engineering

UNIVERSITAT ROVIRA i VIRGILI

Tarragona

2024

Abstract

In the era of sustainability and technological innovation, computational chemistry and machine learning have emerged as revolutionary tools with the potential to transform sectors such as pharmaceuticals, biomedicine, and energy. This study explores targeted molecular generation driven by Mol2Vec embeddings within a diffusion generative model. Starting from an existing model, we modified the architecture to incorporate Mol2Vec embeddings after the node embeddings computed by the Graph Neural Network (GNN). Our analysis demonstrates that while Mol2Vec embeddings show potential in directing the reconstruction of molecules, the overall results are not yet optimal. We observed some variability in performance across different types of molecules, indicating a need for further optimization. Potential improvements include integrating Mol2Vec embeddings before the GNN and extending the training duration. These enhancements could significantly impact drug discovery, metabolomics, agriculture, materials science, and other fields by enabling the generation of molecules with specific desired properties, thus contributing to scientific advancements and industry innovations.

Keywords: Targeted Molecular Generation, Mol2Vec Embeddings, Diffusion Generative Model, Graph Neural Networks, Computational Chemistry, Drug Discovery.

Acknowledgments

First, I want to thank my supervisors, Dr. Marta Sales and Dr. Roger Guimerà, for their help and guidance. Their insights have been crucial in shaping this work.

I am also grateful to my colleagues from the SEES Lab, especially Manuel Ruiz, for his collaboration and for answering all my questions.

A special thanks to my university classmates, who have become friends, for their support and encouragement over the years.

Thanks to my partner, Joan, for his unwavering support and motivation. His interest in everything I do, as well as his patience and understanding during the most difficult moments, was very much appreciated.

Finally, I express my heartfelt gratitude to my parents and my sister for their unconditional love and support. Their confidence in me has been a constant source of strength and motivation.

Contents

1 Introduction

In the era of sustainability and technological innovation, computational chemistry and machine learning (ML)[1] have emerged as revolutionary tools with the potential to transform sectors such as pharmaceuticals, biomedicine, and energy. The relationship between medicine and artificial intelligence (AI) is increasingly close and promising. AI has become a key tool in the research, diagnosis, and treatment of various diseases, enabling more accurate diagnoses, improved management of chronic patients, and shorter vaccine development times [2], [3], [4]..

One of the main challenges in biomedical research has been the development of new drugs, which was evident during the COVID-19 global pandemic. Previously, chemists manually synthesized a broad spectrum of compounds and subjected them to bioactivity tests to ascertain their efficacy and safety—a laborious and costly process that could span over multiple years or even decades [5].

In recent years, biomedical research has focused on integrating AI to advance computational methods in drug development. By employing deep generative models (DGMs), researchers have been able to rapidly generate and optimize novel molecular structures, streamlining the development process. These AI-driven models predict molecular behaviors effectively, which significantly reduces the need for early-stage physical experiments, thereby saving both time and costs [6].

Similarly, in metabolomics, identifying unknown metabolites presents a significant hurdle due to database limitations. Computational techniques emerge as indispensable tools to address this gap, playing a pivotal role in driving advancements in personalized medicine and molecular biology. While traditional analytical methods such as mass spectrometry and nuclear magnetic resonance are crucial for identifying known metabolites in biological samples, computational approaches become crucial in uncovering unregistered metabolites, fostering innovative discoveries in understanding biological processes and diseases [7].

Additionally, in the realm of materials science, the ability to design molecules with specific properties is revolutionizing manufacturing, enabling the creation of stronger, lighter, and more flexible materials. This has applications across industries such as automotive, aerospace, electronics, and construction, as well as in the development of advanced electronic devices, sensors, and batteries, offering more economical and versatile alternatives to traditional materials [8], [9].

[1] Machine learning (ML) is a field of study in artificial intelligence (AI) concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions [1].

Moreover, the development of molecules with enhanced capabilities for energy storage and conversion has the potential to revolutionize renewable energy production and usage. This could significantly improve the efficiency of batteries and fuel cells, paving the way for a more sustainable energy future [10].

Furthermore, in agriculture, creating specific molecules can lead to the development of new, more selective pesticides that are less harmful to the environment [11].

To sum up, this project focuses on exploring how generative models based on graph neural networks (GNNs) can accelerate the design of specific molecules and facilitate the identification of unknown molecules, critical areas that promise significant advances in drug discovery, metabolomic analysis, and other relevant areas of scientific research and applied engineering. These abilities hold immense promise, with the potential to profoundly impact various fields, transforming industries and enhancing the quality of life for people around the world [12].

1.1 Aims of the project

The overarching goal of the project is to modify existing tools for molecule generation to create a tool for the generation of molecules with specific chemical structures.

The specific aims to achieve this goal are:

- To understand vectorial representations of molecular structures using Mol2Vec and convert SMILES to Mol2Vec embeddings.
- To learn how to use graph neural network (GNNs) based generative diffusion models for molecule generation.
- To modify these models to incorporate Mol2Vec embeddings into the generation process to generate molecules with a specific chemical structure.
- To evaluate the generated molecules.
- To assess practical applications and implications of the enhanced molecular generation model in various fields, such as pharmaceuticals and materials science.

2 Neural Networks and Diffusion Models for Molecular Generation

In this section, we explore both graph neural networks and diffusion models, examining their fundamental principles, their applications in structured data analysis, and how they complement each other to address challenges in graph and network modeling. To construct a generative model for molecules, it is crucial to understand the representation of molecular structures, the architecture of neural networks suitable for graph data, and the mechanisms of diffusion processes in generative models.

Building these models generally requires several key steps: representing molecules as graphs, using neural networks designed to handle this graph-structured data, and employing diffusion models to refine and generate molecular graphs. The integration of these components enables the generation of novel molecular structures that adhere to chemical principles and desired properties.

2.1 Molecular Representation in AI-driven drug discovery

Since the 19th century, scientists have been interested in how molecules are represented. Traditionally, they have been represented using the chemical formula or as structure diagrams with bonds and atoms, including Lewis structures, skeletal diagrams, ball-and-stick models, etc. [13], [14]. However, recently, the advancement of fields such as cheminformatics and bioinformatics require new molecular representations to facilitate the computational processing of chemical substances. Some of these representations are SMILES (Simplified Molecular Input Line Entry System), InChI (International Chemical Identifier), graphs, and fingerprints [5], [15].

2.1.1 Molecular Graph Representation

In this representation, a molecule is described as a graph where atoms are represented as nodes and chemical bonds as edges connecting these nodes (Fig 1). This form aligns well with many machine learning techniques, particularly graph neural networks (GNNs), which are designed to handle data in graph form. [16]

Nodes are typically marked with features that describe the atom, such as atomic number, atom type (e.g., carbon, hydrogen, oxygen), hybridization state, and whether it is part of a ring structure. Similarly, edge features might include bond type (single, double, triple, or aromatic) and whether the bond is in a ring [17].

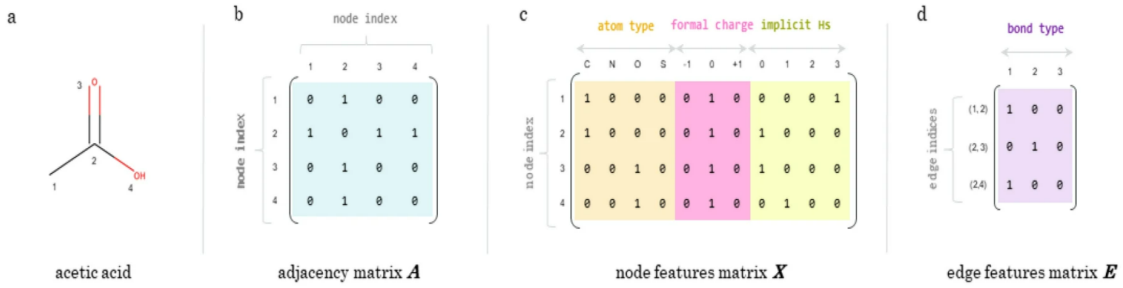

Figure 1. Example for acetic acid. **a** Graph representation of acetic acid with nodes numbered from one to four. **b** Adjacency matrix, A, for an acetic acid graph with the corresponding node ordering on the left. **c** Node features matrix, X, which one-hot encodes a few selected properties. **d** Edge features matrix, E, where each edge feature vector is a one-hot encoding of single, double, or triple bonds. "Implicit Hs" stands for the number of implicit hydrogens on a given node. Figure from [15].

Recently, research has been conducted on the use of molecular graph representations within ML and AI realms, particularly in the fields of drug discovery and molecular property prediction. This includes the development of algorithms and models that can learn from them to make predictions about molecular properties and behaviors.

2.2 Artificial Neural Networks (ANNs)

ANNs play a key role in generation models and form the foundation on which GNNs are built. They are a type of machine learning algorithm inspired by the structure and function of the human brain. ANNs are composed of interconnected nodes, similar to neurons, and can learn to perform specific tasks by processing and analyzing data.

In ANNs, nodes are organized in layers: an input layer, where neurons are fed part of the input data; hidden layers, where most of the learning occurs; and an output layer, which can have one or more nodes depending on the task. Nodes in each layer connect to adjacent layers using connections with adjustable weights W_{ij}. Each node processes information by multiplying the incoming signal S_j from the previous node j by its corresponding weight W_{ij}, and then passing the result through an activation function f to determine its final state a_i. Here, i denotes the current node and j denotes the nodes in the previous layer (Fig. 2). This process can be mathematically represented as:

$$a_i = f\left(\sum_j W_{ij} S_j\right)$$

If this output reaches a threshold, the node sends its state to the next layer. Initially set at random, weights and thresholds are adjusted during training using optimization algorithms like backpropagation, which update the weights to reduce errors. Activation functions like sigmoid or Rectified Linear Unit (ReLU) allow the network to model complex data relationships [18], [19], [20], [21], [22]. These functions are defined as:

$$\text{Sigmoid function:} \quad f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

$$\text{ReLU function:} \quad f(x) = \max(0, x) \tag{2}$$

There are multiple types of neural networks, such as Feedforward Neural Network (FNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Graph Neural Network (GNN) [18], [19]. We will be using GNNs because they are the most suitable type of ANN to represent molecules. GNNs excel at capturing the relationships and interactions between the atoms in a molecule, allowing for a more accurate and detailed representation of molecular structures.

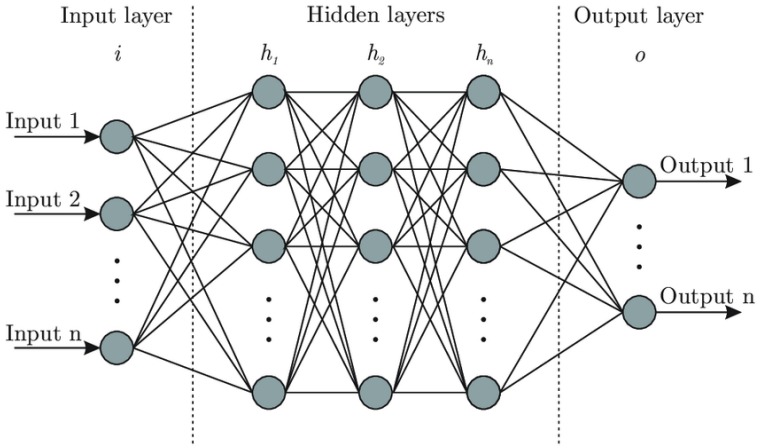

Figure 2. Neural Networks Architecture. Figure from [23].

2.2.1 Graph Neural Networks (GNNs)

GNNs are special types of neural networks capable of working with a graph data structure. They are used in predicting nodes, edges, and graph-based tasks.

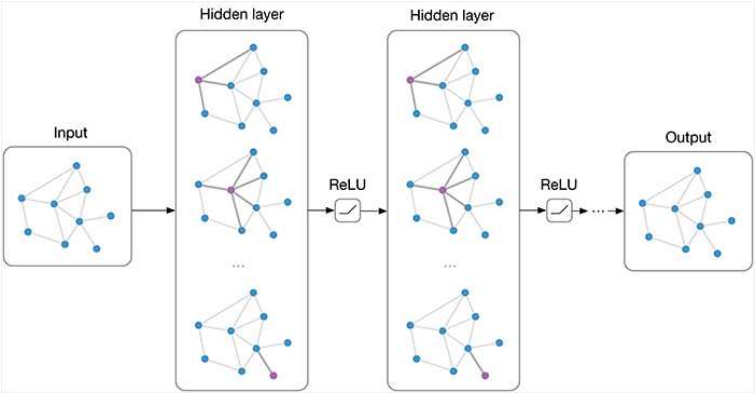

Figure 3. Multi-layer Graph Neural Network (GNN) with first-order filters. Figure from [24].

Each node and each edge in the graph is represented by a feature vector. GNNs operate iteratively, where each node communicates with its neighbors by exchanging the information encoded in their feature vectors. At each step, a node aggregates this information, typically by summing or concatenating it, and incorporates this into its own feature vector (Fig. 3). This process allows each node to gradually accumulate information from its neighbors, capturing the relationships within the graph. GNNs consist of layers that update node features based on their neighbors' features, with non-linear activation functions applied between layers. The network is trained to minimize a loss function using techniques like stochastic gradient descent (SGD). GNNs are effective for tasks such as molecular modeling [25], [26].

2.3 Diffusion Models

Generative models are a class of machine learning models that can generate new data based on training data. Diffusion models are a type of generative model that generate data by progressively destroying data by adding noise to a dataset (Fig. 4) and then learning to recover the data by reversing this noising process, or in other words, applying the denoising process (Fig. 5). So, they can generate coherent images from noise. Diffusion models have been widely used for generating images from text, and recent advancements have further extended their utility in generative AI and deep learning for various applications, including drug development.

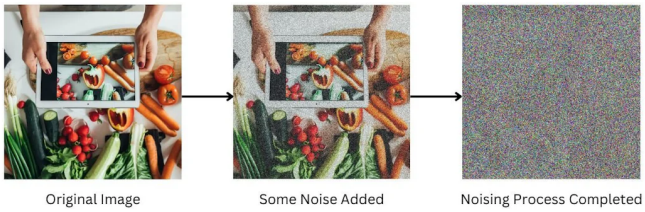

Figure 4. Example of a noising process. Figure from [27].

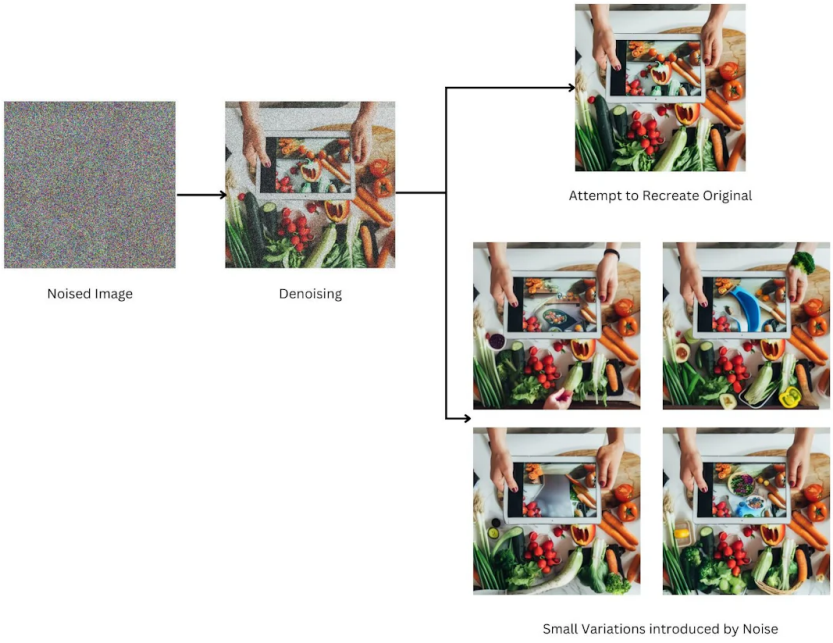

Figure 5. Example of a denoising process. Figure from [27].

Denoising process shows the attempt to de-noise the image in an attempt to recover the original image through several iterations. Different results of the reconstructed image can be observed, representing different degrees of success in denoising. There are small variations that can affect the final fidelity of the recovered image [27], [28], [29].

Our model works as a diffusion model. Just as noise can alter details of an image making perfect restoration not possible, in the context of molecules, adding noise consists of distorting the chemical structure of the molecule. In terms of molecular graphs, adding noise is thus equivalent to swapping pairs of bonds (or edges in the molecular graph); for instance, a swap could correspond to removing two bonds connecting atoms (1,2) and (3,4) and generating two bonds connecting atom pairs (1,3) and (2,4). The de-noising process that a diffusion generative process should follow would then consist of identifying those edges that need to be removed (destroyed) and those that need to be created.

2.4 Generative Diffusion Model

The generative model we use is a deep generative diffusion model that relies on GNNs to create new molecules, developed by Manuel Ruiz Botella as part of his PhD project [30] (Fig. 6). This model operates by first representing each node in the graph, which corresponds to an atom, with a feature vector. Similarly, each edge between the nodes, representing a chemical bond, is also described by a feature vector. Additionally, global features of the graph represent information at the molecular level. The node and edge features are then fed into a GNN, which processes this information iteratively. This iterative process allows each node to exchange information with its neighbors, and the GNN produces embeddings for each node, encapsulating aggregated information from its neighbors.

Once the node embeddings are generated, they are concatenated for every possible pair of nodes, incorporating the graph embedding at this stage. This concatenation allows the model to consider all possible interactions between atoms within the molecule. Attention modules are then used to process these concatenated node pairs, determining the relevance of each pair of nodes in the context of the molecular structure. The output from the attention module is used to predict which bonds should be created to form the molecular structure and which bonds should be removed to correct the structure.

These predictions are then used to adjust the molecular structure by creating new bonds and destroying unnecessary ones. The model is capable of performing complex operations such as reversing double bonds and swapping bond positions based on these predictions. This approach allows the generative model to create new molecules by capturing both local features, such as nodes and edges, and global features of the entire graph. Advanced attention mechanisms guide the generation of bonds and the reconstruction of the molecular structure, ensuring that the resulting molecules are accurate and structurally sound.

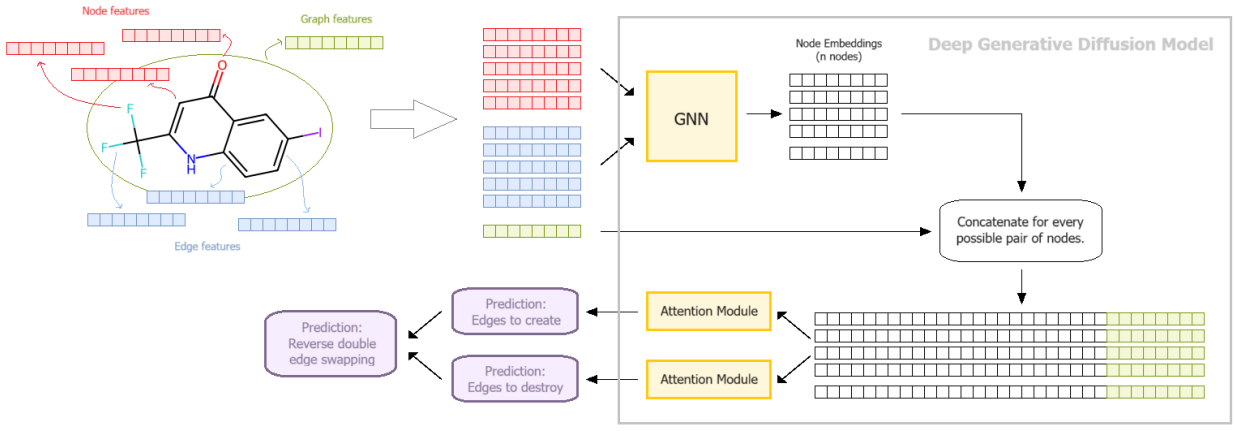

Figure 6. Scheme of the Deep Generative Diffusion Model.

3 Model Implementation

In this section, we explore how we have adapted the model to incorporate additional information through Mol2Vec embeddings, enhancing the quality of the generated molecules and the efficiency of training. The original model was able to generate molecules similar to the original, but by incorporating Mol2Vec embeddings, we aim to direct the generation process more accurately. This ensures that the reconstructed molecules closely match the desired structural and chemical properties.

We use Mol2Vec embeddings because they efficiently represent molecular structure in a vector format, simplifying the integration of additional information into the model. Mol2Vec captures the chemical context of molecular fragments, which is essential for guided molecular generation based on structure. Leveraging these embeddings allows the model to generate more accurate and chemically plausible molecules.

3.1 Mol2vec

Mol2vec is an unsupervised machine learning algorithm that learns vector representations of molecular substructures, inspired by the Word2vec natural language processing technique. This method converts molecules into numerical vectors, or embeddings, that capture their structural and chemical characteristics. These vector representations are useful for applications such as predicting chemical properties and clustering chemical compounds, facilitating their use in various machine learning algorithms [31], [32], [33].

To obtain these Mol2vec embeddings, we have created a Python script that transforms the SMILES of the molecules we get from a pickle file into vector embeddings. This procedure is performed by using the RDKit library to convert SMILES into molecular objects, followed by the application of the mol2vec model to generate 300-dimensional vectors (Fig. 7) [31].

smiles	mol	sentence	mol2vec	embedding
NC(Cn1ccc(=O)n(Cc2ccccc2C(=O)O)c1=O)C(=O)O	<rdkit.Chem.rdchem.Mol object at 0x7ccdf6717ae0>	(847957139, 2599973650, 2245273601, 4274980665...	(300,) dimensional vector	[0.8277235627174377, -1.5075547695159912, -3.8...
N=C(N)N[C@@H](CC(=O)[O-])C(=O)[O-]	<rdkit.Chem.rdchem.Mol object at 0x7ccdf6717b50>	(849275503, 2669055056, 2246699815, 2438720939...	(300,) dimensional vector	[-0.4702579081058502, -0.9614687561988831, -0....
CC(=O)O[C@@]12CO[C@@H]1C[C@H](O)[C@@]1(C)C(=O)...	<rdkit.Chem.rdchem.Mol object at 0x7ccdf6717d10>	(2246728737, 3545365497, 2246699815, 266499585...	(300,) dimensional vector	[-4.459695339202881, -7.249135494232178, -10.0...
CC(C)C(NC(=O)CN)C(=O)O	<rdkit.Chem.rdchem.Mol object at 0x7ccdf6717d80>	(2246728737, 3537119515, 2245273601, 300433380...	(300,) dimensional vector	[-1.1858932971954346, 0.905653715133667, -2.28...
COc1cc2c(cc1OC)CN(S(N)(=O)=O)CC2	<rdkit.Chem.rdchem.Mol object at 0x7ccdf6717df0>	(2246728737, 3975275337, 864674487, 2076190208...	(300,) dimensional vector	[1.7059106826782227, -3.1860127449035645, -1.9...

Figure 7. Conversion of SMILES to Mol2vec vectors.

3.1.1 Generative Diffusion Model with Mol2Vec Embeddings

We have made adjustments to the original generative model by incorporating Mol2Vec embeddings. This improvement aims to increase the accuracy and quality of the generated molecules by providing additional molecular context during the generation process.

In this adapted model (Fig. 8), after generating the node embeddings using the GNN, Mol2Vec embeddings are incorporated along with the graph embeddings and the concatenated node pairs before passing them to the attention modules. Incorporating these embeddings provides the model with additional information about the chemical environment of the atoms, enhancing its ability to generate molecules that are more accurate and chemically plausible.

The attention modules then process these enriched node pair embeddings, determining their relevance within the molecular structure. The results from the attention modules are used to predict which bonds must be created or destroyed to form or correct the molecular structure, similar to the original model. Incorporating Mol2Vec embeddings enhances the model's ability to generate molecules that are not only structurally similar to the originals but also exhibit the desired chemical properties. This improvement leverages the chemical context provided by Mol2Vec to guide the generative process with greater precision, resulting in the generation of higher quality molecules.

The figure shows how Mol2Vec embeddings are integrated into the attention modules.

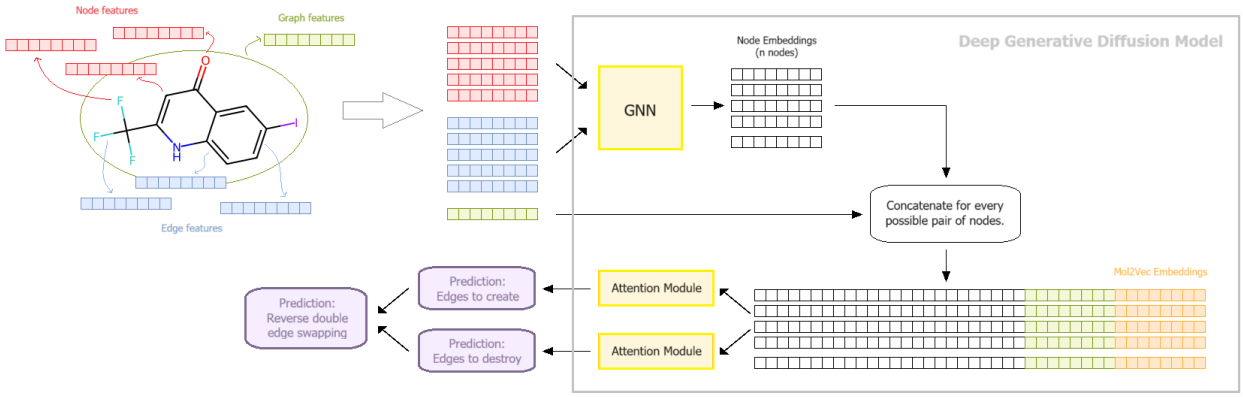

Figure 8. Scheme of the Generative Model with Mol2Vec Embeddings Implementation.

3.2 Model Training

The training process of our model involves graph neural networks (GNNs), multi-head attention layers, and Mol2Vec embeddings as key components.

Initialization

Before training begins, it is essential to initialize the model by setting up the architecture of the GNN layers and the multi-head attention layers, as well as initializing the random weights.

Noise Addition and Degradation

Training begins by iteratively adding noise to the original molecules, degrading them until they become almost indistinguishable from pure noise. This process is executed by a function that selects pairs of nodes in the molecular graph and alters their links. The model must learn to reverse these steps to reconstruct the original molecules.

Processing the Noisy Graph

During each iteration, the GNN processes the noisy graph to generate node and graph embeddings that capture the molecule's structural information. These embeddings, along with Mol2Vec embeddings, are then used by the multi-head attention layers to calculate attention weights, determining the relative importance of different nodes and bonds.

Predicting Molecular Structure

The attention layers estimate the probabilities of forming or breaking bonds within the molecular structure. These predictions are achieved by processing pairs of node embeddings with attention mechanisms. Additionally, the global graph embedding and Mol2Vec embedding are used, and the results are refined using subsequent reduction layers.

Loss Calculation and Backpropagation

The loss is calculated by comparing the model's predictions with the original molecular structures using a loss function that quantifies the difference between the generated structure and the target structure. Once the loss has been calculated, the model's weights are adjusted through a process called backpropagation. Backpropagation involves calculating the gradient of the loss function to determine how to adjust the weights of the GNN and the attention layers [34]. This adjustment is carried out using optimization algorithms such as Stochastic Gradient Descent (SGD), which modify the model parameters in the direction that reduces the loss.

Dataset Division and Training

The dataset is divided into three subsets: 80% for training, 10% for validation, and 10% for testing. The process of adding noise, forward propagation, loss calculation, and backpropagation is repeated through multiple iterations, referred to as epochs. An epoch is one complete pass through the entire training dataset [35]. During this process, the model is trained with the training dataset, while the validation dataset is used to tune the model's hyperparameters and prevent overfitting. The test dataset, which contains molecules not seen during training, is used to evaluate the model's performance and its ability to generalize to new data. This training framework ensures that the model learns to generate accurate molecular structures by effectively reversing the degradation process, guided by the combined information from GNN embeddings and Mol2Vec embeddings.

3.3 Code Modifications

The integration of Mol2Vec embeddings into our model required significant modifications across several components of the codebase. These modifications spanned data loading, model architecture, and the sampling process, each contributing to the enhanced performance and accuracy of our molecular generation model.

The first major component that needed adjustment was the data loader. The original script was responsible for extracting SMILES string for the training, validation, and test datasets. To incorporate the Mol2Vec embeddings, we introduced a new list called `m2v_list`, which was processed alongside the existing `smiles_list`. At each point where `smiles_list` extracted, `m2v_list` was also extracted to ensure that Mol2Vec embeddings were available for each molecule. New variables `train_m2v`, `validation_m2v` and `test_m2v` were calculated to contain the Mol2Vec embeddings of the respective datasets, and then were included in the `train_dl`, `validation_dl` and `test_dl` data loaders, allowing the model to access and use them during the training and evaluation phases. This integration ensured that the Mol2Vec embeds were seamlessly incorporated into the data loading process, providing enriched molecular representations that improved the model's ability to understand and predict molecular properties.

Significant changes were also made to the main training script. Each time data was read from the `train_dl` DataLoader, the script now extracted the `mol2vec_emb` along with the other molecular data. For each molecule, the `mol2vec_tensor` was concatenated with the graph embedding `gemb`. This ensured that the model leveraged the enriched molecular representations provided by the Mol2Vec embeddings, potentially improving its ability to predict and generate molecular structures accurately.

Additionally, for control experiments, we created a false `mol2vec_emb` consisting of a 300-dimensional vector of zeros, allowing for a comparison between the model's performance with real Mol2Vec embeddings versus a null representation.

To integrate the Mol2Vec embeddings into the model architecture, the input size of the `self.mlpGraph` layer was increased to accommodate the additional 300-dimensional Mol2Vec embeddings. Specifically, the input size of `self.mlpGraph` was expanded from `NGFEAT` to `NGFEAT+300`. Furthermore, the reshaping of `xA_m` was also adjusted to include the Mol2Vec embeddings, ensuring that these embeddings were correctly integrated.

Finally, the sampling process, which is crucial for generating molecules, was updated to select the most likely molecule instead of a random one from the top 20, guaranteeing that the best possible molecule was chosen based on the model's predictions. This change ensured that the generated molecules were of the highest possible quality.

4 Evaluation of Generated Molecules

Once the model is fully trained, we proceed to sample the molecules. We use a Python script to apply the trained model, aiming to reverse the molecular structure's decomposition steps as accurately as possible. To evaluate the effectiveness of the model, we split our dataset into two parts: a training file and a test file. The training file contains 12,012 molecules that were used during the training process, whereas the test file includes 1,502 molecules that were not exposed to the model during training.

The test file is particularly critical to our analysis, as it allows us to validate the model's ability to generalize to new data, which is essential to demonstrate that the model can be effectively applied in real-world situations. This validation is key to ensuring the model's utility and reliability in diverse and unforeseen situations.

4.1 Output Data Structure

The Python script generates a comprehensive CSV file that tabulates the properties of the generated molecule, the original, and the noised molecules. This file is essential for assessing the chemical and physical properties of each set of molecules, allowing a direct comparison and assessment of the model's accuracy in replicating and altering molecular structures. The following table illustrates the molecular structure representations (SMILES) and calculated properties, such as TPSA and ring count, among other descriptors (Fig. 9):

	smiles_gen	smiles_ori	smiles_cero	TPSA_ori	TPSA_gen	TPSA_cero	...	RingCount_ori	RingCount_gen	RingCount_cero
0	CCCNc1cc(F)cc(F)c1F	NCCc1ccc(C(F)(F)F)cc1	suciedad	26.02	12.03	12.03	...	1	1	3
1	CN(C)C(=O)NC(=O)NC(=O)NC(=O)n1cccc1P(=O)=O	Nc1ncnc2c1ncn2C1OC(CO)C2OP(=O)(O)OC21	suciedad	154.84	146.68	122.79	...	4	1	8
2	CCCN1C(=O)OC2=CC(P(=O)=O)=CC=C(Cl)C=C(C1=O)C2Cl	O=C(COc1ccccc1P(=O)(O)O)Nc1cccc(Cl)c1Cl	suciedad	95.86	80.75	79.23	...	2	2	7
3	O=C1CCOC(=O)C2=CC(F)=CC(F)=C(F)C=C2N1	CCOC(=O)c1c[nH]c2c(F)c(F)c(F)cc2c1=O	suciedad	59.16	55.40	50.72	...	2	2	6
4	CCC(C)=CC(=Cc1nc1=O)C(=O)OC(C)=O	COC(=O)CC1CC(c2ccc(O)cc2)=NO1	suciedad	68.12	73.33	82.77	...	2	1	7
...	...	...	...	...	...	...	...	...	...	...
1497	C=CN1C(=O)NC(=O)C(O)C2C3NC1C2OP(=O)(O)O3	Nc1ccn(C2O[C@H](CO)[C@H]3OP(=O)(O)O[C@@H]23)c(...	suciedad	146.13	137.43	147.15	...	3	3	4
1498	CCC(=O)NC(=O)C1=C2C=C2C=C(SC)S1	CCOC(=O)c1c(-c2cccs2)csc1N	suciedad	52.32	46.17	41.49	...	2	2	7
1499	CC1C(O)C1C(O)C(=O)N1C(=O)NN=CC(=O)NC1=O	O=c1[nH]c(=O)c2ncn([C@@H]3O[C@H](CO)[C@@H](O)[...	suciedad	153.46	148.40	117.51	...	3	2	8
1500	O=C1SCCc2cc(Br)ccc21	O=C1CCSc2ccc(Br)cc21	suciedad	17.07	17.07	17.07	...	2	2	3
1501	CN=C1Nc2ccccc21	Nc1ccc2cc[nH]c2c1	suciedad	41.81	24.39	47.96	...	2	2	7

1502 rows × 123 columns

Figure 9. CSV output detailing the properties of original, generated, and most noised molecules.

4.2 Similarity Calculation Method

In this section, we describe the methodology used to calculate the similarity between the molecules generated by the model and the original molecules. Molecular similarity is a crucial measure in computational chemistry and drug design, as it allows us to evaluate how well the model can reproduce the structural and functional characteristics of the original molecules. To achieve this, we use two main techniques: molecular fingerprints and Mol2Vec embeddings.

4.2.1 Tanimoto coefficient

The Tanimoto coefficient $T(A, B)$ is a measure of similarity between two sets, often used in chemoinformatics to compare molecular fingerprints. This coefficient ranges from 0 to 1, where 0 indicates no similarity and 1 indicates complete identity [36]. It is defined as:

$$T(A, B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{3}$$

Where:

- $|A \cap B|$ is the number of common features (bits set to 1) in both fingerprints A and B.
- $|A|$ is the number of features (bits set to 1) in fingerprint A.
- $|B|$ is the number of features (bits set to 1) in fingerprint B.

In this study, we utilize two types of fingerprints: MACCS keys (substructure key-based) and Morgan fingerprints (circular). Each fingerprint captures different aspects of molecular similarity. MACCS keys focus on the presence or absence of specific substructures (Fig. 10), while Morgan fingerprints provide a detailed description of the molecular topology based on atom connectivity (Fig. 11). By using both, we aim to achieve a comprehensive assessment of molecular similarity, considering both specific substructures and overall molecular topology [37], [38]. We have decided to calculate the average of the Tanimoto coefficients from both fingerprints, providing a single metric that balances the information captured by each type, thus ensuring a robust and holistic evaluation of molecular similarity.

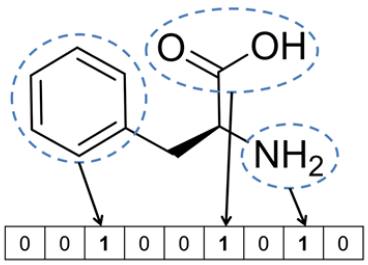

Figure 10. In substructure key-based fingerprints, bits are set according to the substructures that are present in the molecule. (1 if the given substructure is present and 0 if absent.) Thus, each bit position corresponds to a specific substructure. Figure from [38].

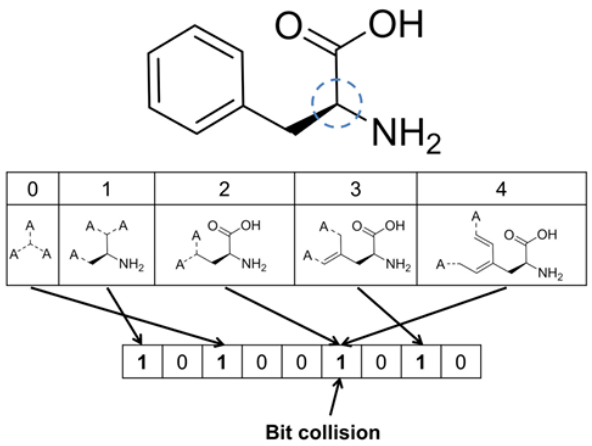

Figure 11. Extended connectivity fingerprints encode atom environments up to a predefined diameter (4 bonds in this example). Terminal atoms also account for bonds not part of the fragment ("A" denotes "any atom"). The central atom here is the α-carbon, but this process repeats for each heavy atom. Figure from [38].

We established thresholds for the Tanimoto coefficient (Tc) to assess the performance of our generative model in producing molecules that maintain structural fidelity to the original ones [39].

- $Tc \geq 0.85$: These molecules are considered highly similar, indicating that the generated molecule closely resembles the original in terms of structure and substructure.
- $0.7 \leq Tc < 0.85$: These molecules are considered moderately similar, suggesting an acceptable level of similarity that may still be useful for certain applications.
- $Tc < 0.7$: These molecules are considered poorly similar, indicating significant structural differences from the original molecule.

4.2.2 Euclidean distance

The Euclidean distance is a classical measure of the similarity between two vectors that represents the direct distance between them in a multidimensional space [40]. In the context of molecules, this distance can be used to quantify how different the vector representations of two molecules are.

To calculate the Euclidean distance between two molecules, we converted the molecules to their vector representations using Mol2Vec embeddings. Once we have these vectors, we can apply the Euclidean distance formula:

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4}$$

Where:

- n is the number of dimensions (features) in the vectors.
- x_i is the value of the i-th feature in the first vector.
- y_i is the value of the i-th feature in the second vector.

We plotted a histogram to visualize the distribution of the Euclidean distances of our molecules across the dataset. The histogram below shows the frequency of different Euclidean distance ranges, helping us to identify the typical distances between the generated and original molecules (Fig. 12).

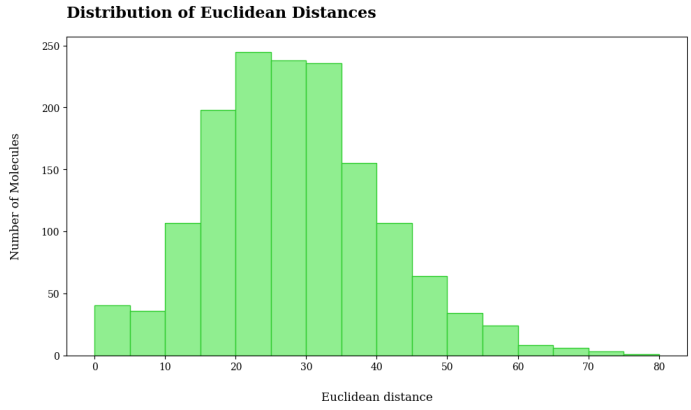

Figure 12. Distribution of Euclidean Distances between generated and original SMILES.

4.3 Analysis of Molecular Properties

In this section, we evaluate the quality of the generated molecules by comparing them to the original ones based on several important molecular properties. The properties selected for this analysis are TPSA, Ring Count, Mol logP, Mol MR, qed, and the numbers of single and double bonds. These properties are crucial to evaluate the functionality, stability, and potential pharmaceutical application of the molecules.

- *TPSA (Topological Polar Surface Area)*: It measures the polar surface area of a molecule, influencing its ability to interact with biological targets and its permeability across cell membranes. It is key to the bioavailability and effectiveness of the drug.
- *Ring Count*: The number of rings affects the stability and reactivity of the molecule. Cyclic structures influence chemical and biological behavior, so maintaining a similar number of rings is essential.
- *Mol logP (Logarithm of Octanol/Water Partition Coefficient)*: Measures the hydrophobicity of a molecule, affecting its solubility and its ability to cross cell membranes, crucial for drug absorption and distribution.
- *Mol MR (Molecular Refractivity)*: Related to the polarizability of the molecule, it influences intermolecular interactions. It is important for understanding how a molecule interacts with biological receptors and other components.
- *qed (Quantitative Estimate of Drug-likeness)*: It measures the similarity of a molecule to other known drugs, considering factors such as bioavailability, chemical stability, and interactions with biomolecules.
- *Single and double bonds*: The number of single and double bonds affects the structure and chemical reactivity of the molecule, determining its flexibility and rigidity, as well as its ability to form interactions.

Evaluating these properties between the original and generated molecules allows us to determine if the generated molecules retain the essential characteristics of the original ones. This ensures their suitability for practical applications in pharmaceutical chemistry and other areas, making them likely to be effective in real-world scenarios.

To do this, we have calculated the absolute difference between the property values of the original and the generated molecules, as well as between the generated and the zero reference molecules. The purpose of this comparison is to determine which of these differences is closer to zero, which would indicate greater similarity. This will allow us to determine whether the properties of the generated molecules are more similar to the original or the noised molecules.

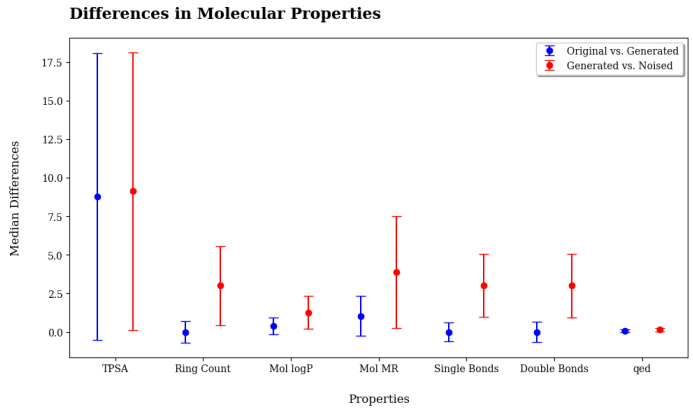

Figure 13. Median differences and standard deviations in molecular properties (TPSA, Ring Count, Mol logP, Mol MR, Single Bonds, Double Bonds, and QED) between original, generated, and noised molecules.

The graph (Fig. 13) shows that the generated molecules are, in general, more similar to the original molecules than to the zero reference molecules, which validates the effectiveness of the generating model used. However, there are variations in how well the different molecular properties are replicated. TPSA presents the greatest difficulty in being accurately replicated, as indicated by the large median differences and high variability. This may be due to the complexity of capturing the topological polar surface in the molecular generation process. Mol logP and qed are the properties that replicate best in the generated molecules, with low median differences and low variability. These properties are crucial for drug design, indicating that the generating model is promising for applications in medicinal chemistry. Ring Count, Single Bonds, and Double Bonds show low median differences with moderate to low variability, suggesting that the basic structural properties of the molecules replicate reasonably well. Finally, Mol MR has moderate medium differences and some variability, indicating acceptable replication but room for improvement.

To sum up, although there are some properties that are more challenging to replicate, the model demonstrates a solid ability to reproduce most molecular properties with high accuracy, especially those crucial for the viability of the molecules as potential drugs.

To delve deeper into these results, we will analyze the molecular properties based on the similarity of the generated molecules to the original ones. Specifically, we will categorize the generated molecules into three groups according to their Tanimoto coefficient: High ($Tc \geq 0.85$), Moderate ($0.7 \leq Tc < 0.85$), and Low Similarity ($Tc < 0.7$) (Fig. 14).

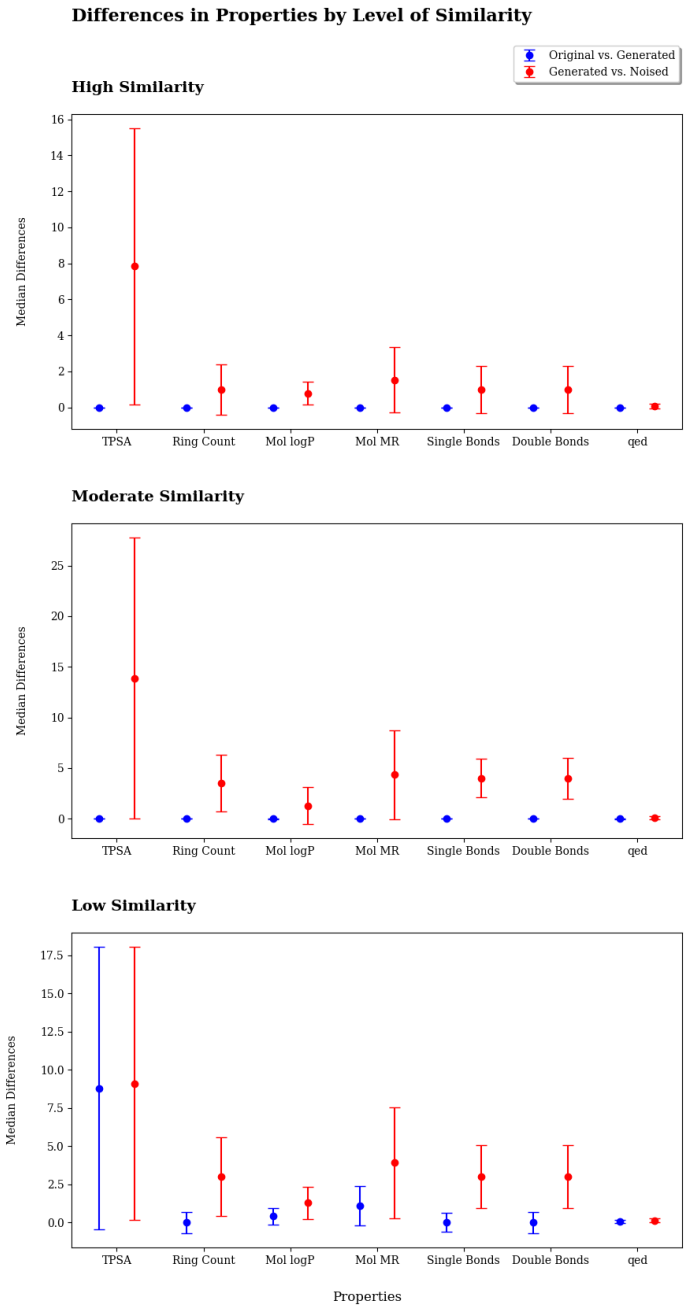

Figure 14. Median differences and standard deviations in molecular properties (TPSA, Ring Count, Mol logP, Mol MR, Single Bonds, Double Bonds, and QED) by level of similarity between original, generated, and noised molecules.

By creating separate graphs for each of these groups, we aim to identify trends and variations in the replication accuracy of different molecular properties across varying levels of structural similarity. This will help us determine how well the generative model performs in preserving crucial properties as the structural similarity to the original molecules changes.

The analysis demonstrates that the generative model preserves molecular properties with high accuracy in the high and moderate similarity groups. The zero median differences in TPSA, Ring Count, Mol logP, Mol MR, Single Bonds, Double Bonds, and qed in these groups reflect the outstanding ability of the model to replicate these properties accurately. In the low similarity group, TPSA has high median differences, indicating that this property is more difficult to replicate when structural similarity decreases. In addition, the high variability observed reflects inconsistency in its generation. However, the other properties still show low median differences, suggesting that the model can replicate these features with reasonable accuracy even with low similarity. Overall, although some structural properties hold up better than others as similarity decreases, the model proves to be robust in preserving key features in most cases.

We can conclude that the reconstructions are valid when the similarity is higher than 0.7, as the model accurately maintains most of the important molecular properties. This ability to generate molecules with high fidelity to the originals is crucial for applications in pharmaceutical chemistry and other areas that require accuracy in molecular features.

4.4 Reconstruction Quality Analysis through Multiple Samplings

In this section, we analyze the results from sampling the molecules across 10 different generations. By examining multiple instances, we aim to identify patterns and consistencies in the generative model's performance, evaluating its stability and reliability in maintaining desired properties and structural fidelity.

We evaluated how frequently each molecule was reconstructed with high, moderate, or low similarity and also calculated descriptive statistics on the Tanimoto coefficients obtained from these iterations. The mean indicates the average similarity, providing an overview of how closely the generated molecules match the originals. The standard deviation reflects the variability or consistency of the reconstructions, showing how much the similarity scores fluctuate across iterations. The maximum and minimum value represent the highest and lowest similarity scores, respectively, highlighting the best and worst cases of molecular reconstruction (Fig. 15). This helps us understand how well the model reproduces the properties of the original molecules across different iterations and identifies any patterns or deviations in the reconstructions.

	Original SMILES	Great	Moderate	Poor	mean	std	min	max
0	NCCc1ccc(C(F)(F)F)cc1	0	0	10	0.249841	0.048559	0.177174	0.350945
1	Nc1ncnc2c1ncn2C1OC(CO)C2OP(=O)(O)OC21	0	0	10	0.323183	0.040551	0.249658	0.368897
2	O=C(COc1ccccc1P(=O)(O)O)Nc1cccc(Cl)c1Cl	0	0	10	0.329749	0.071438	0.242176	0.486364
3	CCOC(=O)c1c[nH]c2c(F)c(F)c(F)cc2c1=O	0	0	10	0.308427	0.067493	0.224044	0.469444
4	COC(=O)CC1CC(c2ccc(O)cc2)=NO1	0	0	10	0.249930	0.042683	0.184211	0.320060
...	...	...	...	...	...	...	...	...
1497	Nc1ccn(C2O[C@H](CO)[C@H]3OP(=O)(O)O[C@@H]23)c(...	0	0	10	0.406757	0.055408	0.332638	0.511905
1498	CCOC(=O)c1c(-c2cccs2)csc1N	0	0	10	0.232970	0.039474	0.199400	0.301695
1499	O=c1[nH]c(=O)c2ncn([C@@H]3O[C@H](CO)[C@@H](O)[...	0	0	10	0.389329	0.037982	0.343056	0.444089
1500	O=C1CCSc2ccc(Br)cc21	1	0	9	0.399141	0.201165	0.180599	0.870968
1501	Nc1ccc2cc[nH]c2c1	0	0	10	0.316821	0.109281	0.190691	0.540625

1502 rows × 8 columns

Figure 15. Summary of reconstruction quality for each molecule sampled 10 times.

To visualize this, we created a bar chart showing the number of molecules that achieved a good similarity with valid properties ($Tc \geq 0.7$) at least once in the 10 samplings compared to those that consistently had low similarities ($Tc < 0.7$) (Fig. 16). This graph helps us to identify which types of molecules it is able to reconstruct and which are not.

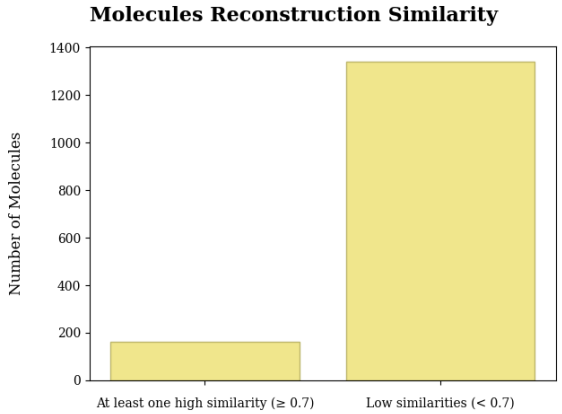

Figure 16. Molecules Reconstructions Similarity across 10 iterations.

The graph reveals that the generative model has a limited ability to produce high similarity reconstructions. However, it is remarkable that there are 163 molecules that are correctly reconstructed at least once, which is a significant achievement for this model indicating that it has good potential. This result suggests that, although improvements in the model are needed to achieve greater accuracy and consistency in generating molecules that more closely resemble the originals, the current progress is already promising and represents an important step in the right direction.

4.4.1 Analysis of physico-chemical properties

We then select the subset of molecules that have been correctly reconstructed at least once for further analysis. Our goal is to better understand the characteristics that determine the quality of the generated molecules, identifying what factors influence some molecules to be well reconstructed while others are not. The factors we analysed are the number of heavy atoms (Fig. 17), the exact mol weight (Fig. 18) and the mol logP (Fig. 19).

To carry out this analysis, we calculate the percentage of molecules that are correctly reconstructed as a function of various factors present in the original molecules. This approach allows us to identify whether there is a correlation between these factors and the ability of the model to generate molecules with high structural similarity.

- *Number of Heavy Atoms*

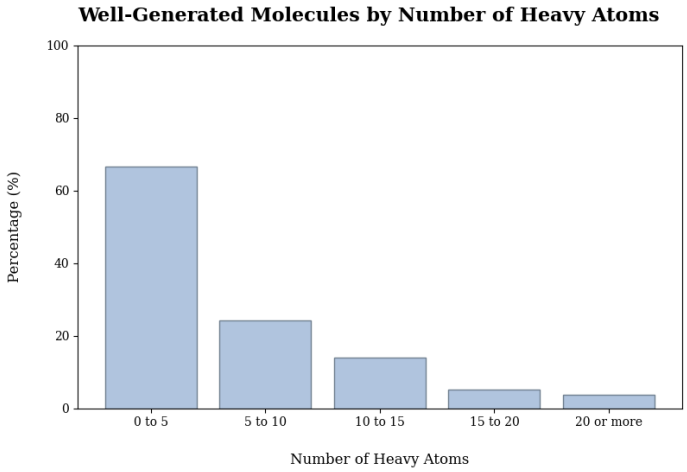

Figure 17. Percentage of Well-Generated Molecules by Number of Heavy Atoms.

It is evident that the model performs significantly better in generating smaller molecules with a high level of similarity to the original ones. Specifically, molecules with 0-5 heavy atoms have a high reconstruction success rate, with over 60% being well-generated. As the number of heavy atoms increases, the percentage of successful reconstructions decreases. For molecules with 5-10 heavy atoms, the success rate drops to about 20%, and it continues to decline further for molecules with more than 10. This trend shows that the generative model has more difficulty accurately reconstructing larger and complex molecules. The decrease in performance with increasing molecule size highlights an area for potential improvement in the model, suggesting that enhancing its capability to handle larger molecular structures could lead to more consistent and accurate reconstructions.

- *Molecular Weight*

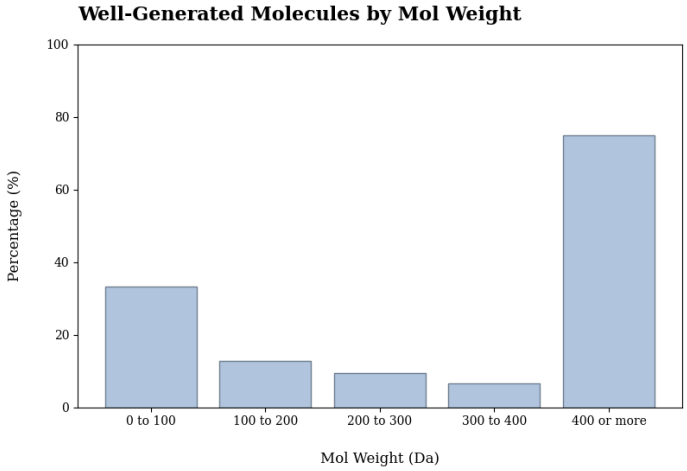

Figure 18. Percentage of Well-Generated Molecules by Molecular Weight.

The model performs well when generating low molecular weight molecules, indicating that these are easier to reconstruct. As the molecular weight increases, the percentage of well-generated molecules decreases. For medium-sized molecules, the model presents additional challenges. The increase in the percentage of well-reconstructed molecules in the higher molecular weight range is unexpected and may indicate that these molecules have specific characteristics that make them easier to reconstruct.

- *Partition coefficient (mol logP)*

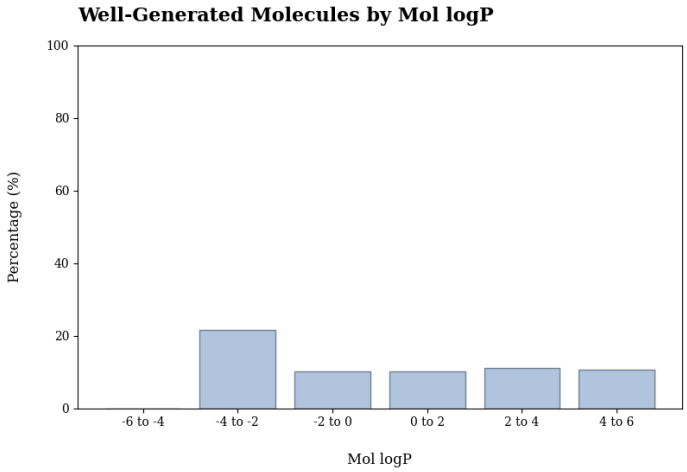

Figure 19. Percentage of Well-Generated Molecules by mol logP.

Unlike the trend observed with the number of heavy atoms and molecular weight, Mol logP does not seem to have a significant impact on molecule generation. The percentages of well-generated molecules remain relatively constant across the different ranges of Mol logP, although it appears that those with negative values tend to be generated with higher fidelity. This observation suggests that hydrophilicity (indicated by negative Mol logP values) could positively influence the generative model's ability to accurately reconstruct molecules. We cannot be certain, but it seems that either hydrophobicity is not a determining factor or it has a negative influence.

4.4.2 Variability in Similarity

In this section, we develop an analysis of the variability in the similarity of the molecules generated through the 10 samplings based on the calculation of the standard deviation. This analysis will allow us to understand how the similarity varies with respect to the original ones in different iterations of the model, that is, how consistent the model is. A high standard deviation will indicate higher variability and therefore lower consistency in the molecular reconstructions, while a low standard deviation will suggest high consistency (Fig. 20).

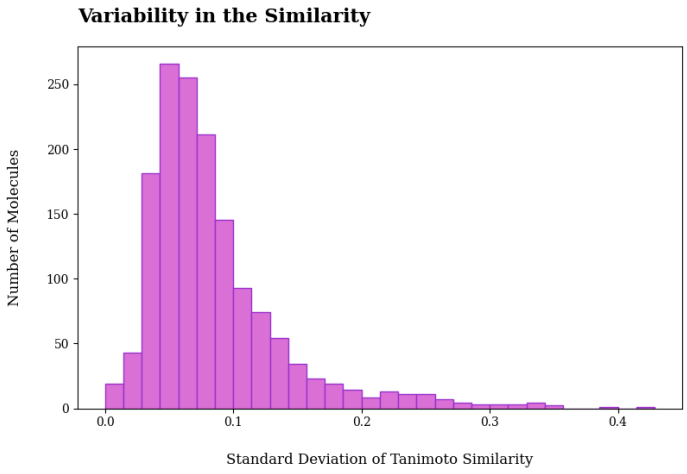

Figure 20. Variability of Symmetry in Generated Molecules.

This graph shows that most molecules have a standard deviation of Tanimoto similarity less than 0.1. This indicates that, in general, our model is quite consistent in reproducing molecules similar to the original ones over multiple iterations. It has a high capacity to maintain the structural fidelity of the original molecules, which is promising for its application in fields such as pharmaceutical chemistry and materials design. However, the presence of molecules with high variability in similarity reflects inconsistencies in the generation of certain molecules, suggesting that there are areas where the model can improve.

4.4.3 Analysis of Specific Cases

In this section, we analyze specific cases to more precisely identify the strengths and weaknesses of our generative model. To do this, we have selected molecules that represent different behaviors and classified them into three categories based on their average similarity and standard deviation of similarity (Fig. 1):

Table 1. Summary of the type of selected molecules with their similarity statistics.

Average Similarity	Standard Deviation	Similarity Category
1.00	0.00	Consistently high
±0.6	±0.2	Moderate variability
±0.2	<0.05	Consistently low

We have selected two molecules for each category, illustrating examples of consistently high similarity, moderate variability, and consistently low similarity (Fig. 21. This helps provide a clearer view of how the model works in different contexts.

	Original SMILES	Great	Moderate	Poor	mean	std	min	max	Heavy Atoms	Mol Weight	MolLogP
239	CNS(N)(=O)=O	10	0	0	1.000000	0.000000	1.000000	1.000000	6	110.014998	-1.59070
542	N=C(N)NC(N)=O	10	0	0	1.000000	0.000000	1.000000	1.000000	7	102.054161	-1.45183
393	CCC(Cc1ccc2c(c1)OCO2)NC	0	0	10	0.185405	0.032991	0.160778	0.270930	15	207.125929	1.95580
1142	O=C(O)CN(CC(=O)O)Cc1c(O)cccc1O	0	0	10	0.249739	0.029623	0.191582	0.293218	18	255.074287	0.06900
195	NC(=O)c1ccc(Br)cc1	4	0	6	0.640553	0.341596	0.254085	1.000000	10	198.963276	1.54800
1034	CCSc1nc2ccccc2[nH]1	3	0	7	0.506211	0.347786	0.183608	1.000000	12	178.056469	2.67490

Figure 21. Filtered molecules for analysis of individual cases

Our Python script for molecule sampling generates an image for each molecule in which we can observe the denoising process. The diagram (Fig. 22) shows the original molecule followed by the noisy molecule and then the intermediate steps of the denoising process until reaching the generated molecule.

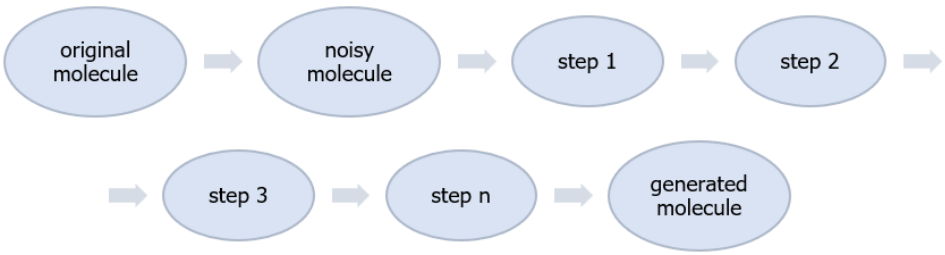

Figure 22. Scheme to interpret the images of the molecules.

Category: Consistently high

- *Molecule 239:* CNS(N)(=O)=O

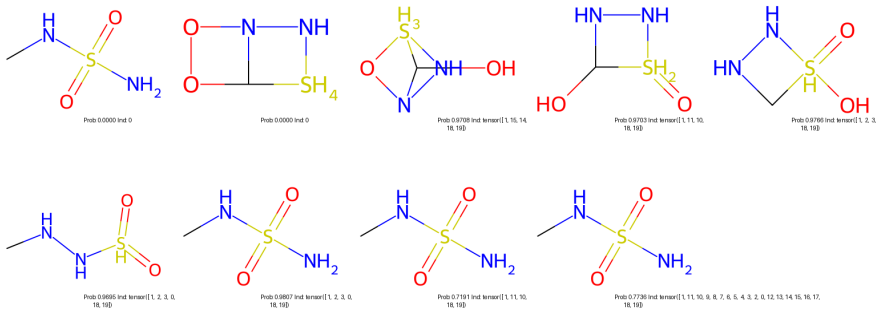

Figure 23. Great reconstruction of *CNS(N)(=O)=O*.

- *Molecule 542:* N=C(N)NC(N)=O

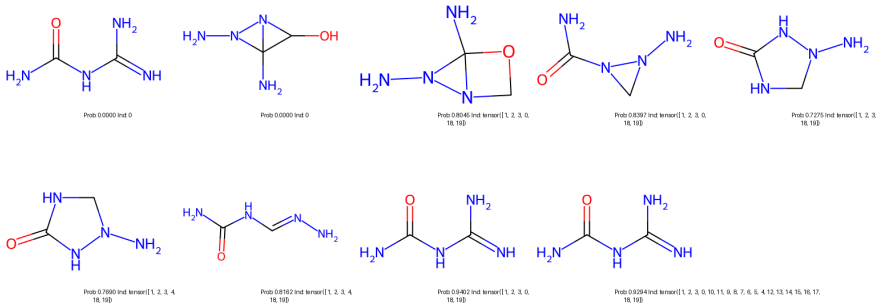

Figure 24. Great reconstruction of *N=C(N)NC(N)=O*.

Both molecules (Fig. 23) (Fig. 24) have shown exceptional performance. In all iterations, the generative model managed to reproduce them with perfect similarity, achieving an average Tanimoto coefficient of 1.00, and thus also a standard deviation of 0.00, and a minimum and maximum of 1.00. These molecules have 6 and 7 heavy atoms, an exact molecular weight of 110 and 102 Da, and a mol logP of -1.59 and 1.45, respectively. The relatively low number of atoms and molecular weight, and the negative mol logP, indicating high hydrophilicity, appear to be positive contributing factors to the high accuracy in their reconstruction. This result suggests that our model has a high capacity to handle and reproduce small molecules with these specific features.

Category: Moderate variability

- *Molecule 195:* NC(=O)c1ccc(Br)cc1

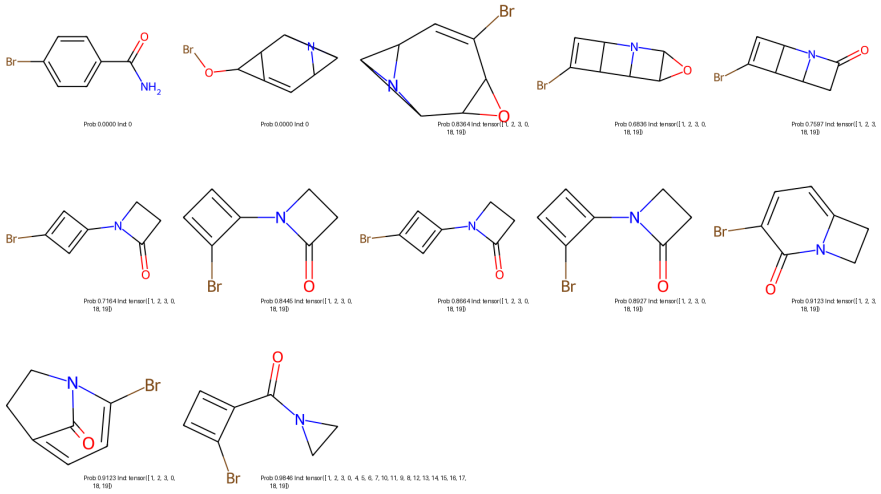

Figure 25. Poor reconstruction of NC(=O)c1ccc(Br)cc1.

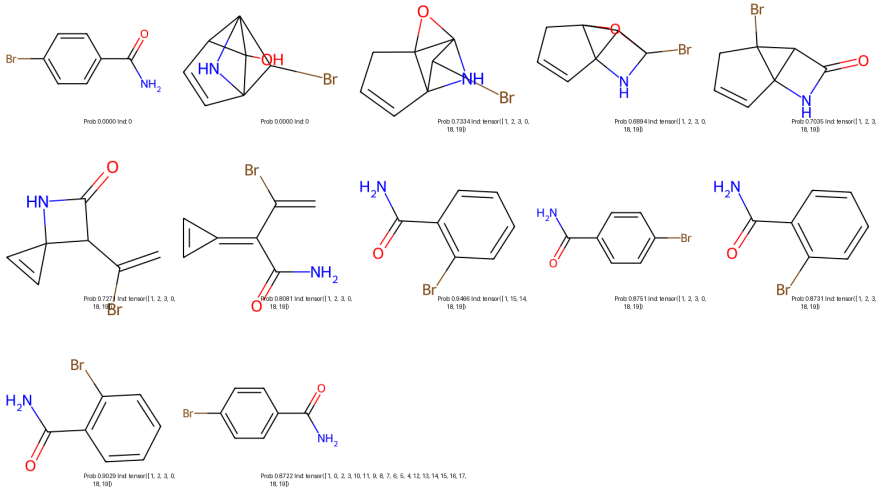

Figure 26. Great reconstruction of NC(=O)c1ccc(Br)cc1.

In this case, the model showed varied results over the 10 iterations. This molecule was successfully reconstructed with perfect similarity in 4 iterations (Fig. 26), while in the 6 remaining iterations, it had difficulty reversing the noise, resulting in final generated molecules that were quite different from the original (Fig. 25). The average Tanimoto coefficient for these reconstructions was 0.64, which, although below the threshold of 0.7

for a valid reconstruction, is still relatively close. However, the standard deviation of 0.34 indicates considerable variability in the reconstruction quality. The Tanimoto coefficient ranged from a minimum of 0.25 to a maximum of 1, highlighting the inconsistency in the model's performance. This molecule has 10 heavy atoms, an exact molecular weight of 198.96 Da, and a MolLogP of 1.55. These specific features, indicating a relatively high number of heavy atoms and a considerable molecular weight, seem to influence the accuracy of the reconstruction and suggest areas for further refinement of the model.

- *Molecule 1034:* CCSc1nc2ccccc2[nH]1

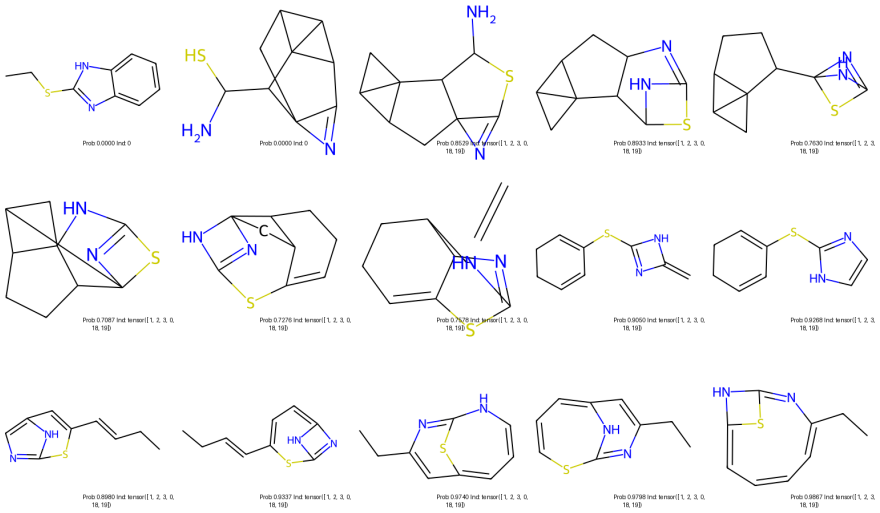

Figure 27. Poor reconstruction of CCSc1nc2ccccc2[nH]1.

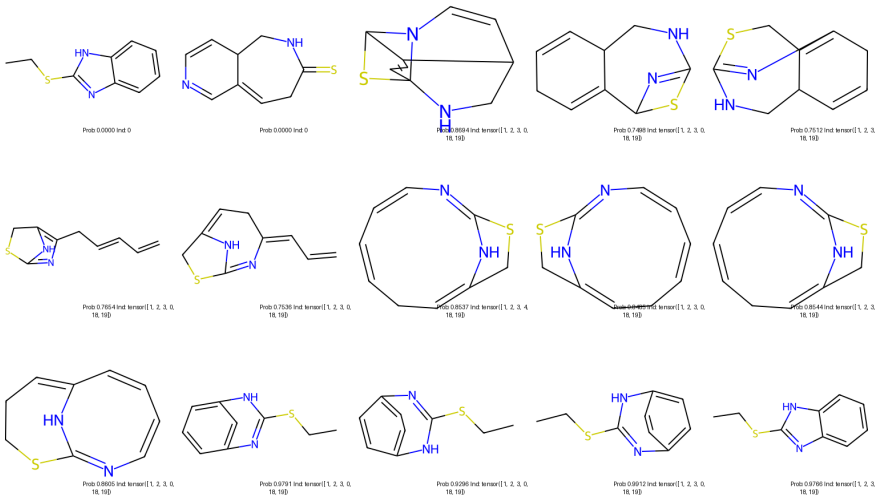

Figure 28. Great reconstruction of CCSc1nc2ccccc2[nH]1.

Similarly, for this molecule, the model also exhibited variability in performance. Out of the 10 iterations, it successfully reconstructed accurately in 3 instances (Fig. 28), while in 7 iterations it did not manage to produce a structure that closely matched the original (Fig. 27). The average Tanimoto coefficient was 0.50, with a standard deviation of 0.35, indicating significant variability. The Tanimoto coefficient ranged from a minimum of 0.18 to a maximum of 1. This molecule has 12 heavy atoms, an exact molecular weight of 178 Da, and a MolLogP of 2.67. This molecule also has a relatively high number of atoms, and consequently a high molecular weight, characteristics that the model does not handle very well, leading to frequent reconstruction failures.

Despite the challenges observed in some iterations, the successful reconstruction in some samplings indicates that the model has the potential to handle and accurately reproduce complex molecules, though further refinements may be needed to ensure consistent performance across different samplings.

Category: Consistently low

- *Molecule 393:* CCC(Cc1ccc2c(c1)OCO2)NC

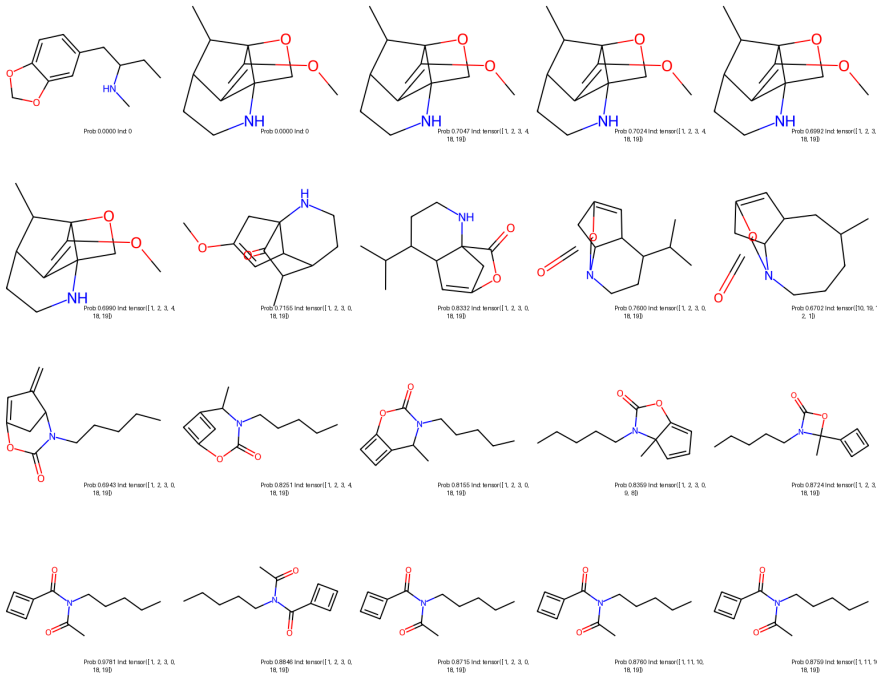

Figure 29. Poor reconstruction of CCC(Cc1ccc2c(c1)OCO2)NC.

- *Molecule 1142:* O=C(O)CN(CC(=O)O)Cc1c(O)cccc1O

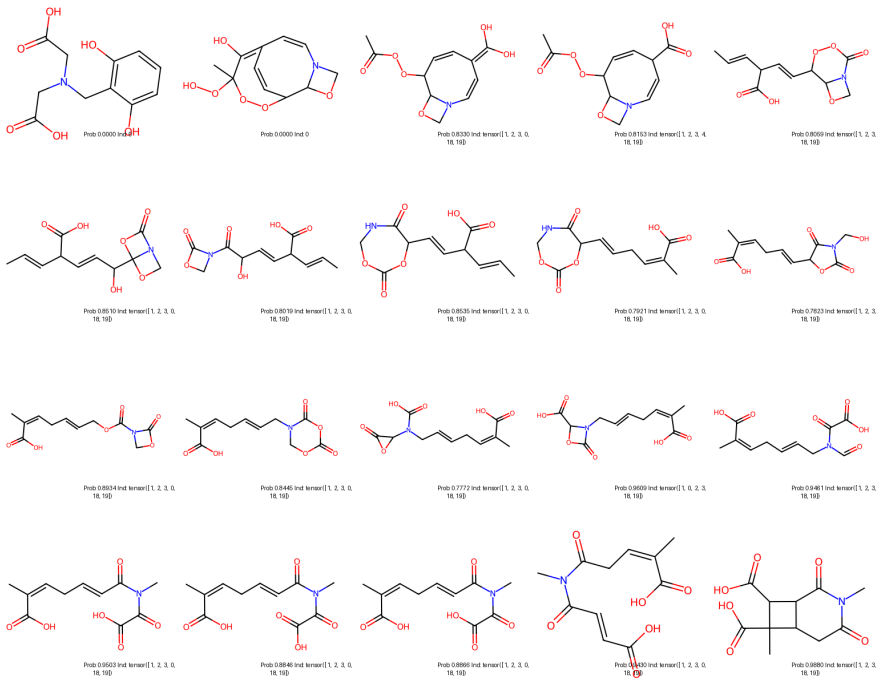

Figure 30. Poor reconstruction of O=C(O)CN(CC(=O)O)Cc1c(O)cccc1O.

Both molecules (Fig. 29) (Fig. 30) have shown consistently low performance. In all iterations, the generative model failed to reproduce them accurately, achieving an average Tanimoto coefficient of 0.185 and 0.25, a standard deviation of 0.033 and 0.296, a minimum of 0.16 and 0.19, and a maximum of 0.27 and 0.29. These molecules have 15 and 18 heavy atoms, an exact molecular weight of 207.13 and 255.07 Da, and a MolLogP of 1.96 and 0.069, respectively. The relatively high number of atoms and molecular weight appear to be negative contributing factors to the low accuracy in their reconstruction. This result highlights the limitations of our model in handling and reproducing larger and more complex molecules with these specific features.

4.5 Evaluating the Influence of Mol2Vec Embeddings

In this section, we assess the impact of Mol2Vec embeddings on our model's performance. We conducted an experiment where we retrained the same model, but instead of using Mol2Vec embeddings, we used zero vectors of the same dimensions. This allowed us to directly compare the results and validate the positive impact of Mol2Vec embeddings.

We first compared the number of successful reconstructions ($Tc \geq 0.7$) of an iteration between both training methods. As shown in the figure (Fig. 31), the model trained with Mol2Vec embeddings greatly outperformed the model trained with zero vectors, demonstrating the positive impact of incorporating Mol2Vec embeddings in the training process.

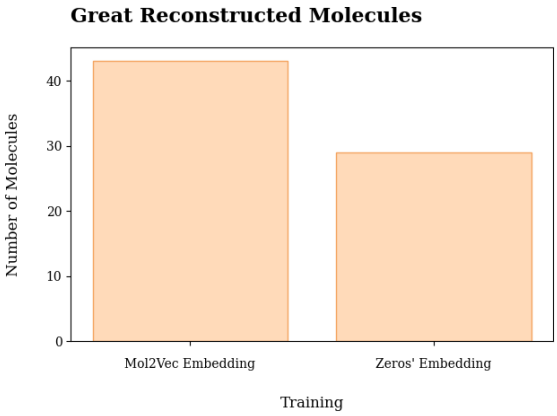

Figure 31. Number of successful reconstructions ($Tc \geq 0.7$) for both training methods.

We also compared the two key metrics: the Tanimoto coefficients and the Euclidean distances between the original and generated molecules. The graph below (Fig. 32) shows box plots summarizing these metrics for both training methods.

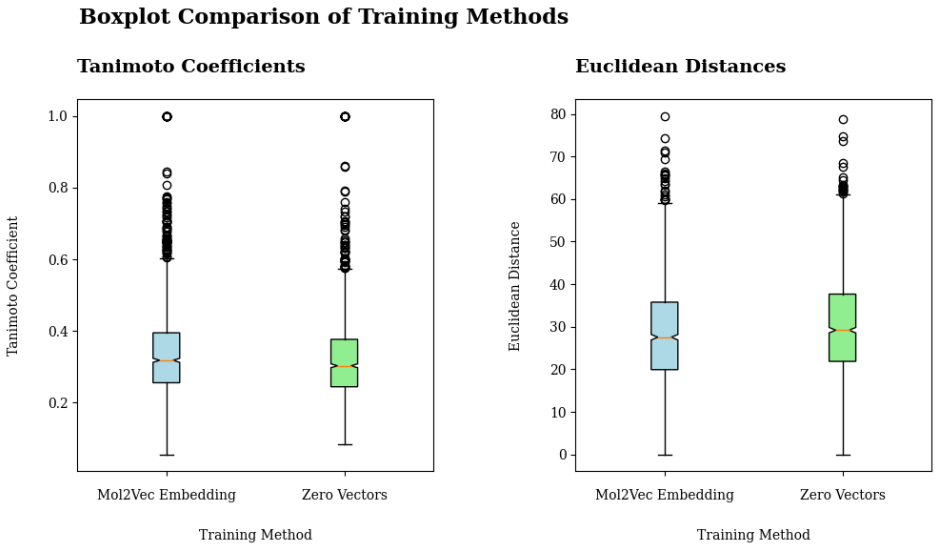

Figure 32

We can observe that the Tanimoto coefficients are higher and less dispersed for the model trained with Mol2Vec embeddings compared to the model trained with zero vectors. This indicates that the use of Mol2Vec embeddings significantly improves the similarity between the generated molecules and the original ones, providing higher accuracy in the molecular reconstruction. In terms of Euclidean distances, the model trained with Mol2Vec embeddings also shows lower variability and smaller values. This means that the molecules generated with Mol2Vec embeddings are not only more similar in terms of their key structural features but are also closer in feature space, resulting in greater consistency in the reconstruction.

To sum up, the results of all analyses indicate that the use of Mol2Vec embeddings provides a clear advantage in generating more accurate and coherent molecules, both in terms of structural similarity and proximity in feature space.

5 Conclusion

Our analysis demonstrates that while the incorporation of Mol2Vec embeddings has shown potential in directing the reconstruction of molecules, the overall results are not yet optimal. This indicates that, while a promising approach, it requires significant improvement. We observed variability in the performance of the model across different types of molecules. While the model shows promising results for some molecular structures, it struggles with others, indicating that further tuning and possibly architectural adjustments are necessary. Using Mol2Vec embeddings can help guide the generative process, but additional strategies are required to consistently enhance performance.

There are several strategies that could enhance the effectiveness of our model. One potential improvement is to integrate Mol2Vec embeddings before the GNN. Graph Neural Networks (GNNs) are specifically designed to operate on graph-structured data, making them ideal for processing molecular structures where atoms are nodes and bonds are edges. By incorporating these embeddings earlier, the GNN can utilize this rich structural information from the beginning of the learning process. Mol2Vec embeddings provide detailed information about the chemical environment of each atom, including properties derived from similar molecules. By integrating these embeddings before the GNN, we allow the network to leverage this additional context during the message-passing and aggregation phases. This could result in more informative node features being propagated through the network, improving the model's ability to learn meaningful patterns and relationships within the molecular graph. If the GNN were given richer initial features, it could start learning more complex representations from the first layer. This could potentially reduce the number of layers or iterations needed for the GNN to converge on an accurate model, thus improving learning efficiency and final performance.

Another straightforward yet effective approach is to extend the training duration. Longer training times allow the model to iterate more on the training data, which can help it to fine-tune weights and learn more nuanced patterns. With each additional iteration, the loss value tends to decrease, moving closer to zero, indicating improved model performance. This is particularly important in complex tasks such as molecular generation, where subtle details can significantly impact the quality of the generated molecules.

Training a model to generate molecules with valid and desired properties has significant implications across various fields. In the pharmaceutical sector, it can facilitate drug discovery by generating candidate molecules with specific therapeutic properties. In metabolomics, it can enhance the study of known and unknown metabolites and their interactions within biological systems. In agriculture, it aids in developing agrochemicals with specific functionalities to improve crop yield and resistance. In materials science, it helps create materials with tailored

properties for industrial applications. Improving the model enables more accurate and efficient generation of molecules that meet desired criteria, significantly impacting these industries and contributing to scientific advancements. By refining our approach and leveraging the detailed chemical context provided by Mol2Vec embeddings, we can achieve more precise and targeted molecular generation, leading to transformative advancements in various scientific and industrial domains.

References

[1] Wikipedia. "Machine learning." (2024), [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning.

[2] IBM, "The benefits of ai in healthcare," *IBM Insights*, Jul. 2023. [Online]. Available: https://www.ibm.com/think/insights/ai-healthcare-benefits.

[3] "Artificial intelligence in drug discovery and development," *Drug Discovery Today*, Oct. 2020. DOI: 10.1016/j.drudis.2020.10.010. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7577280/.

[4] "A survey on deep learning in medicine: Why, how and when?" *Information Fusion*, 2021. DOI: 10.1016/j.inffus.2020.09.006. [Online]. Available: https://doi.org/10.1016/j.inffus.2020.09.006.

[5] "Generative chemistry: Drug discovery with deep learning generative models," *Journal of Molecular Modeling*, vol. 27, no. 3, p. 71, 2021. DOI: 10.1007/s00894-021-04674-8. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10984615/.

[6] "Molecular design in drug discovery: a comprehensive review of deep generative models," *Briefings in Bioinformatics*, vol. 22, no. 6, bbab344, Aug. 2021, ISSN: 1477-4054. DOI: 10.1093/bib/bbab344. eprint: https://academic.oup.com/bib/article-pdf/22/6/bbab344/41089800/bbab344.pdf. [Online]. Available: https://doi.org/10.1093/bib/bbab344.

[7] C. Proteomics. "Metabolomics in drug development." (2024), [Online]. Available: https://metabolomics.creative-proteomics.com/metabolomics-in-drug-development.htm?gad_source=5&gclid=EAIaIQobChMIk9PQj5efhgMV4YpoCR3GWAZ5EAAYAiAAEgJrHvD_BwE.

[8] U. C. de Madrid. "Nuevas moléculas y polímeros orgánicos semiconductores para dispositivos optoelectrónicos." (2024), [Online]. Available: https://www.ucm.es/otri/complutransfer-nuevas-moleculas-y-polimeros-organicos-semiconductores-para-dispositivos-optoelectronicos.

[9] "A database of molecular properties integrated in the materials project," Aug. 2023. DOI: 10.26434/chemrxiv-2023-tz7x8.

[10] T. Lewis, "Methods and the types of energy storage molecules," *Nanotechnol Lett*, 2021. [Online]. Available: https://www.pulsus.com/scholarly-articles/methods-and-the-types-of-energy-storage-molecules-9548.html.

[11] "Applying molecular docking to pesticides," *Pest Management Science*, vol. 79, no. 11, pp. 4140–4152, Nov. 2023. DOI: 10.1002/ps.7700. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/ps.7700.

[12] "Deep learning as an opportunity in virtual screening and drug design," *Frontiers in Chemistry*, vol. 5, p. 65, 2018. DOI: 10.3389/fchem.2017.00065. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5785775/.

[13] LibreTexts. "Molecular representations." (2024), [Online]. Available: https://chem.libretexts.org/Courses/Howard_University/General_Chemistry:_An_Atoms_First_Approach/Unit_2:__Molecular_Structure/Chapter_5:_Covalent_Bonding/Chapter_5.8:_Molecular_Representations.

[14] LabXchange. "Introduction to neural networks." (2024), [Online]. Available: https://www.labxchange.org/library/pathway/lx-pathway:c8b529f2-c589-487b-876f-6d35e0076eb3/items/lx-pb:c8b529f2-c589-487b-876f-6d35e0076eb3:html:d7e509b0.

[15] "Molecular representations in ai-driven drug discovery: A review and practical guide," *Journal of Cheminformatics*, vol. 12, p. 56, 2020. DOI: 10.1186/s13321-020-00460-5. [Online]. Available: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-020-00460-5.

[16] Y. Kwon, D. Lee, Y. Choi, S. Kim, H. Oh, and J. Lee, "Compressed graph representation for scalable molecular graph generation," *Journal of Cheminformatics*, vol. 12, p. 58, 2020. DOI: 10.1186/s13321-020-00463-2. [Online]. Available: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-020-00463-2.

[17] DeepChem. "Featurizers." (2024), [Online]. Available: https://deepchem.readthedocs.io/en/latest/api_reference/featurizers.html.

[18] MathWorks. "What is a neural network?" (2024), [Online]. Available: https://www.mathworks.com/discovery/neural-network.html.

[19] IBM. "Neural networks." (2024), [Online]. Available: https://www.ibm.com/topics/neural-networks.

[20] A. W. Services. "What is a neural network?" (2024), [Online]. Available: https://aws.amazon.com/es/what-is/neural-network/.

[21] Databricks. "Artificial neural network." (2024), [Online]. Available: https://www.databricks.com/glossary/artificial-neural-network.

[22] GeeksforGeeks. "Neural networks: A beginner's guide." (2024), [Online]. Available: https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/.

[23] T. D. Science. "Designing your neural networks." (2024), [Online]. Available: https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed.

[24] T. Kipf. "Graph convolutional networks." (2024), [Online]. Available: https://tkipf.github.io/graph-convolutional-networks/.

[25] GeeksforGeeks. "What are graph neural networks?" (2024), [Online]. Available: https://www.geeksforgeeks.org/what-are-graph-neural-networks/?ref=ml_lbp.

[26] T. A. Summer. "Graph neural networks." (2024), [Online]. Available: https://theaisummer.com/Graph_Neural_Networks/.

[27] S. AI. "Diffusion models guide." (2024), [Online]. Available: https://scale.com/guides/diffusion-models-guide#what-are-diffusion-models.

[28] AssemblyAI. "Diffusion models for machine learning: Introduction." (2024), [Online]. Available: https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/.

[29] Coursera. "Diffusion models: A comprehensive guide." (2024), [Online]. Available: https://www.coursera.org/articles/diffusion-models.

[30] M. Ruiz-Botella, S.-P. M, and R. Guimerà, *A study of example*, unpublished, 2024.

[31] K. Samoturk, T. Laino, and J. Gervasoni, *Mol2vec: An unsupervised machine learning approach to learn vector representations of molecular substructures*, 2024. [Online]. Available: https://github.com/samoturk/mol2vec.

[32] V. Kisin, *Tutorial: Machine Learning in Chemistry Research - RDKit & mol2vec*, 2024. [Online]. Available: https://www.kaggle.com/code/vladislavkisin/tutorial-ml-in-chemistry-research-rdkit-mol2vec#mol2vec---learning-vector-representations-of-molecular-subctructures.

[33] K. Samoturk, T. Laino, and J. Gervasoni, "Mol2vec: Unsupervised machine learning approach with chemical intuition," *Journal of Chemical Information and Modeling*, vol. 58, no. 12, pp. 2391–2400, 2018. DOI: 10.1021/acs.jcim.7b00616. [Online]. Available: https://pubs.acs.org/doi/10.1021/acs.jcim.7b00616.

[34] Wikipedia. "Backpropagation." (2024), [Online]. Available: https://en.wikipedia.org/wiki/Backpropagation.

[35] Simplilearn. "What is an epoch in machine learning?" (2024), [Online]. Available: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-epoch-in-machine-learning?utm_source=frs_article_page&utm_medium=top_share_option&utm_campaign=frs_copy_share_icon.

[36] FeatureBase. "Tanimoto similarity and jaccard indexes with featurebase." (2024), [Online]. Available: https://featurebase.com/blog/tanimoto-similarity-in-featurebase.

[37] K. Gao, D. D. Nguyen, V. Sresht, A. M. Mathiowetz, M. Tu, and G.-W. Wei, "Are 2d fingerprints still valuable for drug discovery?" *Physical Chemistry Chemical Physics*, vol. 22, no. 16, pp. 8373–8390, 2020. DOI: 10.1039/d0cp00305k. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7224332/.

[38] D. Bajusz, A. Rácz, and K. Héberger, "Chemical data formats, fingerprints, and other molecular descriptions for database analysis and searching," in Dec. 2017, ISBN: 9780124095472. DOI: 10.1016/B978-0-12-409547-2.12345-5.

[39] K. Szilágyi, B. Flachner, I. Hajdú, *et al.*, "Rapid identification of potential drug candidates from multi-million compounds' repositories. combination of 2d similarity search with 3d ligand/structure based methods and in vitro screening," *Molecules*, vol. 26, no. 18, p. 5593, 2021. DOI: 10.3390/molecules26185593. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8468386/.

[40] S. A. Bero, A. K. Muda, Y.-H. Choo, N. A. Muda, and S. F. Pratama, "Weighted tanimoto coefficient for 3d molecule structure similarity measurement," *Computational Intelligence and Technologies (CIT) Research Group, Center of Advanced Computing and Technologies, Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka*, 2024.

Appendices

A Programming Code

There are four files that stand out in our project:

- *main_single_fast_v3_quadrupletes.py*: This is the main script responsible for training the model. It includes the core training loop, handles the data loading, and manages the saving of model checkpoints. During the training process, a directory is created to save the state of the model at each epoch. These checkpoints are stored in files named `model_epoch_{epoch}.pth` within a folder named with the current date.
- *data_loader.py*: This script handles the preparation and loading of the dataset. It ensures that Mol2Vec embeddings are incorporated alongside the SMILES strings.
- *models.py*: This file contains the definition of the neural network architecture, including the modifications to integrate Mol2Vec embeddings into the model.
- *sample_fast_v3_extra.py*: This script manages the sampling process for generating molecules. It includes the logic for selecting and processing the molecular structures during the sampling phase. During the sampling process, a directory is creted to store the reconstruction process for all generated molecules, saving composite images and data for each step of the reconstruction named `composite_molecule_(...).png`.

Training the model is a slow process, so it is necessary to run it for a long period of time. In my case, I had the code running for 10 days on a computer with 8 CPUs and it reached epoch 456.

A.1 Data Loader Modifications

We have modified the data loader to include Mol2Vec embeddings alongside the SMILES strings in training and validation processes. These changes ensure that Mol2Vec embeddings are seamlessly integrated into the data loading process.

```
def build_dataset(all_smiles, all_m2v, ftr=0.8, fva=0.1, ... :

    # Prepare the whole dataset with all the embeddings
    g_list, e_list, s_list, m2v_list, nmol = [], [], [], [], 0
    ...
    train_gr = GraphDataset(g_list[:Ntr])
    train_e = TensorDataset(e_tensor[:Ntr])
    train_s = s_list[:Ntr]  # Train SMILES
    train_m2v = m2v_list[:Ntr]  # Train VECTORS

    validation_gr = GraphDataset(g_list[Ntr:Ntr+Nva])
    validation_e = TensorDataset(e_tensor[Ntr:Ntr+Nva])
    validation_s = s_list[Ntr:Ntr+Nva]  # Validation SMILES
    validation_m2v = m2v_list[Ntr:Ntr+Nva]  # Validation VECTORS

    test_gr = GraphDataset(g_list[Ntr+Nva:])
    test_e = TensorDataset(e_tensor[Ntr+Nva:])
    test_s = s_list[Ntr+Nva:]  # Test SMILES
    test_m2v = m2v_list[Ntr+Nva:]  # Test VECTORS

    combined_dataset_tr = PairedDataset(train_gr, train_e, train_s, train_m2v)
    combined_dataset_va = PairedDataset(validation_gr, validation_e, validation_s, validation_m2v)
    combined_dataset_te = PairedDataset(test_gr, test_e, test_s, test_m2v)

    train_dl = DataLoader(combined_dataset_tr, batch_size=bs, collate_fn=graph_collate_fn)
    validation_dl = DataLoader(combined_dataset_va, batch_size=bs, collate_fn=graph_collate_fn)
    test_dl = DataLoader(combined_dataset_te, batch_size=bs, collate_fn=graph_collate_fn)

return train_dl, validation_dl, test_dl, nbonds_perc
```

A.2 Training Script Modifications

In this code, we first extract the Mol2Vec embedding tensor for each molecule and then concatenate this embedding with the existing graph embedding (`gemb`). This concatenated tensor is subsequently used as part of the training data. We also provide an alternative approach for training with a 300-dimensional zero vector for comparative analysis.

```
def main(train_dl, test_dl, model, checkpoint, executor):
    ...
    for train_graph_b, train_edge_b, smiles, mol2vec_emb, atoms in train_dl:
        ...
        count = 0
        contador_molecula = 0
        for sigma_i in sigma_list:
            ...
            for result, contador_molecula in zip(results, range(len(results))):
                ruido, gemb, nemb, distances, edge_index, edge_attr, natoms, num = result

                # Extract the Mol2Vec embedding tensor
                mol2vec_tensor = torch.from_numpy(mol2vec_emb[count].astype(np.float32))

                # Concatenate the Mol2Vec embedding with the graph embedding
                gemb = torch.cat([gemb.unsqueeze(0), mol2vec_tensor.unsqueeze(0)], dim=1)

                # For training with a 300-dimensional zero vector
                # zero_emb_tensor = torch.zeros(1, 300)
                # gemb = torch.cat([gemb.unsqueeze(0), zero_emb_tensor], dim=1)

                # contador_molecula += 1
```

A.3 Model Architecture Modifications

One of the key changes was to adjust the size of the input layer in the `self.mlpGraph` neural network from `NGFEAT` to `NGFEAT+300`. This modification allows the model to include the 300-dimensional Mol2Vec embeddings alongside the original graph features, enriching the input data with detailed molecular context.

Additionally, the reshaping step for the `xA_m` tensor was updated to accommodate the new dimensions, ensuring that the Mol2Vec embeddings are correctly integrated into the model's processing pipeline. These changes enable the generative model to utilize the enriched molecular representations provided by Mol2Vec embeddings, ultimately improving its ability to predict and generate more accurate and chemically plausible molecular structures.

```
class GATN_35_onlyGNNv3_quadlogits_GIN(torch.nn.Module):
    def __init__(self):
        super().__init__()
        ...
        self.mlpGraph = nn.Sequential(
            Linear(NGFEAT+300, 2*NGFEAT),  # Modification to include Mol2Vec embeddings
            nn.ReLU(),
            Linear(2*NGFEAT, 2*NGFEAT)
        )

    def forward(self, data):
        ...
        xA_m = xA.reshape([shBatch, NGFEAT+300])  # Modification to include Mol2Vec embeddings
        xann = self.mlpGraph(xA_m)
```

A.4 Sampling Process Modifications

In the sampling process, we had to modify the selection mechanism for choosing the most likely molecular structures. In this code, we sort the candidates by their probabilities in descending order using `torch.argsort` and then select the top candidate. This deterministic approach replaces the previous method of randomly selecting a candidate from the top 20, ensuring that the most probable molecular structure is chosen for further processing. These changes are critical for improving the reliability and quality of the generated molecules.

```
def process_batch(conjunto, model, num, generacion, str_date, cantidad):
    df_generated_temporal = pd.DataFrame()
    valid_graph_b, valid_edge_b, smiles, mol2vec, _ = conjunto
    ...
    for graph, tensor, smi, mol2vec_t in zip(...):
        ...
        indice_eleccion = torch.argsort(probabilidades_softmax, descending=True)
        for indice in indice_eleccion:

            # Obtain the selected quadruplet
            cuadrupletas = list(probabilidades_cuatrupletas.keys())
            eleccion = cuadrupletas[indice.item()]
            ...
            if componentes_act <= componentes_ant:
                contador_molecula += 1

                # Store the molecule and its probability
                molecules.append(mol)
                probabilities.append(probabilidades[indice].item())
                indices.append(indice_eleccion)

                if contador_molecula == vueltas:
                    ...
                    # Adding the SMILES and descriptors to the DataFrame
                    data = {"smiles_gen": smiles_str, "smiles_ori": smi, 'smiles_cero': "suciedad"}
                    for d in DESCRIPTORES:
                        data[d + "_ori"] = des_ori[d]
                        data[d + "_gen"] = des_gen[d]
                        data[d + "_cero"] = des_cero[d]
                    for i, d in enumerate(GDESCRIPTORES):
                        data[d + "_ori"] = desg_ori[i]
                        data[d + "_gen"] = desg_gen[i]
                        data[d + "_cero"] = desg_cero[i]

                    # Draw and save the molecule
                    img = Draw.MolToImage(mol)

    df_generated_temporal.to_csv(...)
    return df_generated_temporal
```