Alejandro García Martínez

De Novo molecule generation for biomedical research

Degree Final Project

Supervised by Dr Marta Sales Pardo
and Dr Roger Guimerà Manrique

Bachelor's degree in Biomedical Engineering

UNIVERSITAT ROVIRA i VIRGILI

Tarragona
2023

Abstract

In recent years, multiple generative models have been developed for *De Novo* molecule generation. Generative models are a potentially powerful tool for generating molecules and exploring chemical space. Various areas of biomedical research can benefit from their use, such as drug discovery, metabolomics, and more.

In this study, we have adapted the model proposed by You et al., a Graph Convolutional Policy Network, to generate molecules with a specific molecular formula as a parameter. We have verified the validity and realism of the generated molecules. Additionally, we have generated in silico spectra for the generated molecules to propose new metabolites for the spectra present in the ARUS database. Finally, we have developed a model (Random Forest Regressor) to predict the aqueous solubility of the generated molecules.

Keywords: deep generative models; graph convolutional policy network; metabolomics; drug discovery.

Acknowledgments

I would like to express my gratitude to Marta and Roger for all the help and guidance they have provided me in this project, from which I have learned a lot.

I want to thank my colleagues from SEES:lab for the support they have offered me from day one.

I want to express my most sincere gratitude to David Girbau and Xavier Correig, who have guided me during these years.

Finally, to my parents for their unconditional support.

Contents

1 Introduction 1

1.1 Aims of the project 3

2 Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation 4

2.1 Artificial Neural Networks 4
2.2 Molecular graph representation 5
2.3 Graph Convolutional Network (GCN) 6
2.4 Reinforcement Learning 7
2.5 Architecture and Operation of GCPN 8
2.5.1 Problem definition 8
2.5.2 Molecule Generation Environment 9
2.5.3 Graph Convolutional Policy Network 10
2.5.4 Model Training 11

3 Implementation of the model including the molecular formula 12

3.1 The importance of including the molecular formula in the generative model 12
3.2 Model development 13
3.2.1 Expert data 14
3.2.2 Planarity 15

4 Evaluation of Generated Molecules 17

4.1 Output Data Structure 17
4.2 Analysis of the number of generated molecules 18
4.3 Filtering of generated molecules 19
4.3.1 Filtering the $C_8H_9NO_2$ dataset 20
4.3.2 Filtering the $C_{17}H_{19}NO_3$ dataset 21
4.3.3 Filtering the $C_9H_{11}BrN_2O_5$ dataset 22
4.4 Mol2Vec 24
4.5 Distribution of generated molecules 24
4.6 Chemical structure similarity 27

5 Identification of Spectra from the ARUS Database using the Modified Generative Model 30

6 Prediction of aqueous solubility (logS) of generated molecules 34

6.1 Training data 35
6.2 Quantitative Estimation of Drug-likeness descriptors 35
6.3 Random Forest Regressor for LogS Prediction 36
6.4 Prediction of logS for generated molecules with the molecular formula C8H9NO2 37

7 Conclusion 39

8 Bibliography 40

A Programming code 43

A.1 Code description 43
A.2 Run 43
A.3 Code 43

B Additional figures 44

1 Introduction

Medicine is undergoing significant transformations driven by the introduction of new technologies and the growth of innovative disciplines such as proteomics, metabolomics, genomics, epigenomics, precision medicine and robotics. Artificial intelligence (AI) plays a fundamental role in the transformation of medicine, with widespread application across the majority of medical and biomedical disciplines [1].

The development of new drugs has been one of the major challenges in biomedical research, and this challenge became particularly evident during the global pandemic caused by SARS-CoV-2. The drug discovery process involves exploring a discrete chemical space composed of 10^{60} drug-like compounds. However, out of this set, it is estimated that only 10^{8} have therapeutic relevance. Additionally, a report conducted by a renowned consulting firm on the top 20 pharmaceutical companies in 2022 estimates that the process of developing a drug and bringing it to the market takes an average of 10 to 15 years, with an investment of around 2.3 billion dollars [2] [3].

In recent years, a significant portion of biomedical research has focused on introducing artificial intelligence and developing computational methods in the drug discovery process. One solution that has received significant attention is Deep Generative Models. DGMs are neural networks consisting of numerous layers, trained to approximate complex high dimensionality probability distributions from samples. Following training, DGMs can be used to estimate the probability of each observation and generate new samples using the underlying distribution [4]. DGMs can be employed to generate new molecules (de novo molecule generation). It provides us with a powerful tool for quickly, efficiently, and cost-effectively exploring chemical space [2].

Metabolomics is another discipline that greatly benefits from DGMs. Metabolomics is a growing discipline that studies metabolites, a set of small molecules with a molecular weight of less than 1.5-2 kDa derived from cellular metabolism. Metabolites provide information about metabolic pathways and complex networks within biological systems [5]. DGMs are capable of generating hundreds of thousands of metabolites. DGMs are potentially a powerful tool to explore the chemical space of metabolites and their identification as we will discuss later.

Metabolomics is widely used in the detection of biomarkers, the study of disease mechanisms, identification of new drug targets, enhanced knowledge of drug metabolism, customization of drug treatments and monitoring of therapeutic outcomes [6]. A common scenario is the search for biomarkers in biofluids for the early detection of cancer. For example, Qi et al. conducted a metabolomic analysis on plasma samples to identify metabolites that could enable the early diagnosis of lung cancer. Qi et al. were able to identify five key metabolites (almitic acid, heptadecanoic acid, 4-oxoproline, tridecanoic acid and ornithine). Furthermore, they were able to identify a set of metabolites that allowed the distinction of cancer types (adenocarcinoma, squamous cell carcinoma, small cell lung cancer) [7].

Metabolomics utilizes advanced high-performance analytical techniques, such as nuclear magnetic resonance (NMR) and mass spectrometry (MS), for the measurement of metabolites on a sample. These two techniques are the most widely used in metabolomic studies and are complementary [5]. NMR and MS provide, as a result, the spectra corresponding to the metabolites present in the sample. There is also a growing interest in optical spectroscopy techniques, such as Raman spectroscopy and infrared spectroscopy [8].

In metabolomics, there are three main workflows: targeted analysis, semi-targeted analysis and non-targeted analysis (Fig. 1). The targeted experiments analyze and quantify a small number of specific known metabolites. These experiments follow a driven-hypothesis approach. Semi-targeted metabolomics is an intermediate analysis between non-targeted metabolomics and targeted metabolomics. It aims to quantify a number of known metabolites in the sample while also attempting to discover unknown metabolites. Untargeted metabolomics (non-targeted metabolomics) is performed to collect and analyze all possible metabolites in a sample [5]. A common procedure in non-targeted metabolomics relies on the use of the LC-MS technique. After sample preparation, the compounds are separated from the physical sample using liquid chromatography (LC). Subsequently, electrospray ionization (ESI) is applied (incorporated within the mass spectrometer itself). ESI generates gas-phase ions from the analyte molecules. Finally, the ions enter the mass spectrometer, which generates mass spectra for each chromatographic peak [9].

In this phase of the procedure, it is necessary to identify the metabolite that corresponds to each spectrum. However, it is not a simple process. The identification of metabolites is a challenging task in metabolomics, with a large number of unknown metabolites that may have biomedical relevance. For example, the National Institute of Standards and Technology has developed a public database called ARUS (Annotated Recurrent Unidentified Spectra) that includes thousands of spectra that frequently appear in the analysis of plasma and urine samples. The corresponding molecular formulas of these spectra are known, but the structure of the molecules is unknown [10].

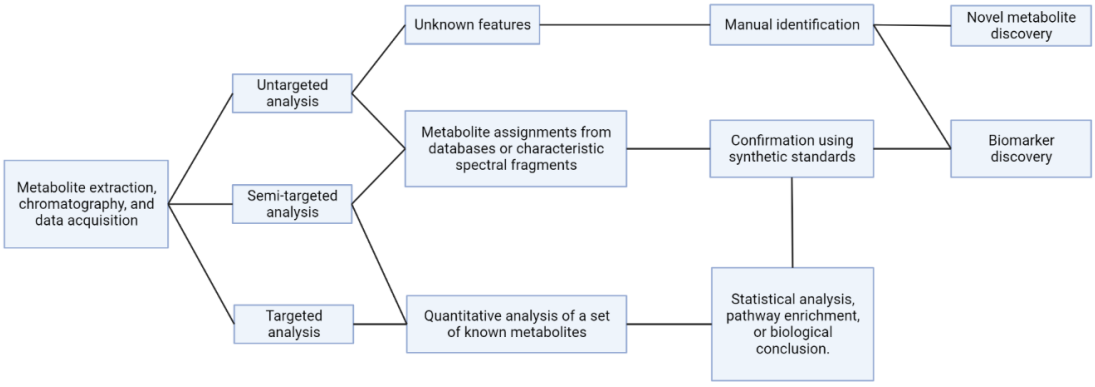


Figure 1. Targeted, Semi-targeted, and Untargeted Metabolomics workflows. Adapted from "Metabolomics: A Primer," by X. Liu, J.W. Locasale, 2017, Trends in Biochemical Sciences

For spectrum identification, traditionally mass spectra libraries have been used, where the query spectrum of an unknown compound is compared against a database of reference MS or MS/MS spectra. The candidate molecules from the database are ranked based on the similarity of their spectra. However, the primary drawback of these methods is that the reference database, often incomplete, merely represents a small fraction of the molecules in reality. As a result, if the reference spectrum of the targeted compound is not contained in the database, it leads to unreliable matching results [11].

Numerous computational methods have been proposed to solve the problem of metabolite identification [11]. In recent years, there has been a growing interest in the development of computational methods capable of generating in silico spectra. Some well-known methods include QCEIMS, CFM-ID, or MassFrontier [12].

As mentioned earlier, generative models are a promising tool for the exploration of the chemical space of metabolites. The combination of generative models and algorithms for generating in silico spectra offers the possibility of proposing new metabolites for the identification of spectra.

In recent years, several generative models have been employed for de novo molecule generation. For example, Gomez-Bombarelli et al. developed a Variational Autoencoder (VAE) for de novo molecule generation [13]. Guimaraes et al. also proposed ORGAN, a model based on Generative Adversarial Network (GAN) and reinforcement learning (RL) [14]. In this study, our focus is on the model proposed by You et al. This model is explained in the paper published by You et al., titled "Graph Convolutional Policy Network (GCPN) for goal-directed molecular graph generation" [15]. As the title indicates, a Graph Convolutional Policy Network (GCPN) is used. GCPN consists of a graph convolutional network that incorporates reinforcement learning (RL) to generate goal-directed graphs. This approach allows the generation of molecules where the generative process can be focused on specific objectives while considering chemical rules. This fact is of great importance as it allows the optimization of essential chemical properties in molecule development, such as druglikeness or the octanol-water coefficient (logP).

1.1 Aims of the project

The objectives of this project are:

- Study the architecture and operation of the Graph Convolutional Policy Network (GCPN) proposed by You et al.
- Modify the algorithm to receive as a parameter a certain molecular formula and generate realistic molecules with the specified molecular formula.
- Find a mathematical description of the generated molecules.
- Check how realistic the generated molecules are.
- Apply a tool for in silico spectrum generation to the generated molecules in order to propose new candidates for the unidentified spectra in the ARUS library (Annotated Recurrent Unidentified Spectra).
- Develop a computational method to predict the aqueous solubility (logS), a significant parameter in compounds with the potential to become drugs, of the generated molecules

2 Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation

In this section we examine the operation and architecture of the Graph Convolutional Policy Network (GCPN). To achieve this objective, we present an overview of the key concepts and elements that constitute the GCPN. Subsequently, we investigate the overall functioning of the GCPN specifically for Goal-Directed Molecular Graph Generation.

2.1 Artificial Neural Networks

As discussed in the introduction section, neural networks, or artificial neural networks, play a fundamental role in Deep Generative Models. Artificial neural networks (ANNs) consist of layered nodes, including an input layer, one or more hidden layers, and an output layer. Taking inspiration from the human brain's structure, ANNs comprise a vast number of interconnected nodes or neurons. These neurons are simple units that assign weights to their incoming connections (Fig. 2). During network activation, the node receives data through the input connections, and each data point is multiplied by its corresponding weight. The results are then aggregated to obtain a single value. Subsequently, the value is passed through an activation function, which determines the output of the node. If the output exceeds a predefined threshold, the node becomes activated and transmits the result to the next layer. Initially, the weights and thresholds are initialized with random values and are continuously adjusted during the training process. Various activation functions are available, with notable examples being the sigmoid function, the Rectified Linear Unit (ReLU), and Leaky ReLU [16] [17]. 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), among others.

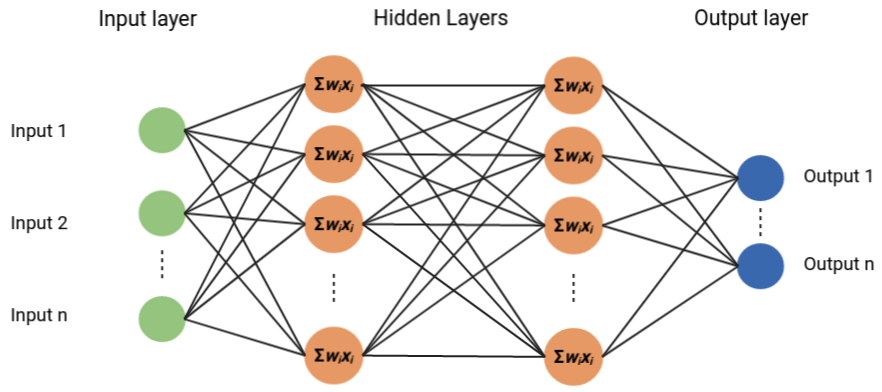


Figure 2. Neural networks organized in layers (input layer, hidden layer, output layer). We can look at the densely interconnected neurons and the data multiplied by the weights and added. Adapted from TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning (chapter 4), by SRamsundar, B. and Zadeh, R.B., 2018, O'Reilly Media. Created with BioRender.com

2.2 Molecular graph representation

Throughout history, chemists have shown interest in the representation of molecules. They have developed multiple types of two-dimensional representations, such as skeletal structures, Newman Projections, or Lewis structures. In recent times, the development of new disciplines such as bioinformatics and cheminformatics has led to the need for developing new representations that enable the computational processing of chemical compounds, such as SMILES (Simplified Molecular Input Line Entry System), molecular graph representation, fingerprints, and InChI (International Chemical Identifier) [18].

Deep generative models for de novo molecule generation developed in recent years have mainly relied on two types of representations: SMILES and molecular graph representation [2]. However, You et al. decided to use molecular graph representation as it is more robust than SMILES, which is a text-based representation of a molecule that encodes its structure using a linear string of characters. In addition, molecular graph representation allows for valence and chemical validity checks [15].

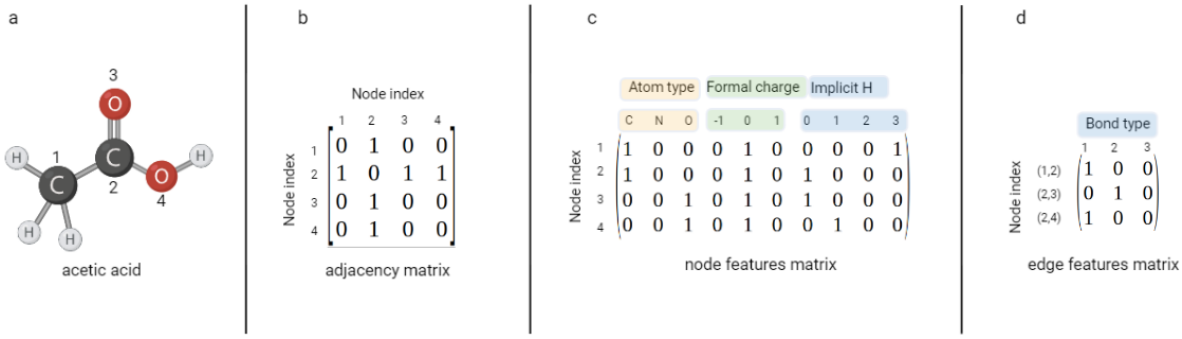


Figure 3. a) The graph representation of acetic acid assigns numbers 1 to 4 to the nodes, excluding the hydrogens. b) The adjacency matrix for the acetic acid graph is constructed by ordering the rows and columns based on the node index. c) The node features matrix for the acetic acid graph comprises vectors that encode different properties, including the atom type, charge presence, and the number of implicit hydrogens. d) The edge features matrix for the acetic acid graph encodes the type of link between nodes, distinguishing between single, double, and triple bonds. Adapted from "Molecular representations in AI-driven drug discovery: a review and practical guida," by L. David, A.Thakkar, R.Mercado, O.Engkvist, 2020, Journal of Cheminformatics, Volume(12), page 56. Created with BioRender.com

A graph is defined as a tuple that consists of a set of nodes (V) and a set of edges (E), where each edge connects pairs of nodes within the set V. In the specific context of a molecular graph, the nodes represent the atoms that constitute the molecule, while the edges correspond to the links or bonds connecting these atoms. In order to process a graph using a computer, it is essential to represent the nodes and edges using linear data structures, such as vectors and matrices. The graph information is stored in three matrices (Fig. 3):

- The Adjacency matrix is a square matrix that describes the connections between nodes. Each element of the matrix, denoted as a_{ij}, represents the connection between nodes i and j. If there is an edge between the nodes, a_{ij} is assigned a value of 1. Otherwise, if there is no edge, a_{ij} is assigned a value of 0 (Fig. 3b).
- The Node feature matrix contains information about the nodes in the graph. It can include various features related to the nodes. For example, it is common to encode the atom type as a node feature (Fig. 3c) [19].
- The edge features matrix encodes the identity of the bonds within a graph. It encompasses various features that can be represented. One widely used encoding scheme is based on the type of bond (single,double or triple) (Fig. 3d) [18].

2.3 Graph Convolutional Network (GCN)

Graphs exhibit a complex structure that often complicates the discovery of the knowledge they hold. The complexity of graphs is attributed to their non-Euclidean nature, the absence of a consistent structure, and scalability [20]. To handle this type of data, graph neural networks (GNNs) were introduced. GNNs are specialized neural networks designed to operate on graph-structured data. They take a graph as input and produce a graph as output. This enables predictions to be made at the node, edge, or graph level [21]. There are various types of GNNs, but in this study, we will focus on the Graph Convolutional Network (GCN) as it is a key component in the model proposed by You et al [15]. In contrast to traditional neural networks, GCNs are not fully connected.

GCN is a specific type of GNN that utilizes convolutional operation. The application of convolution on graphs has been a challenging task for researchers and has been extensively studied in recent years, primarily due to the non-Euclidean nature of graphs. Various techniques have been developed, which can be grouped into two categories: spectral graph convolutions and spatial graph convolutions. Spectral graph convolutions are rooted in the concept of the graph Fourier Transform. In contrast, spatial graph convolutions rely on aggregating node representations from their local neighborhoods. The spatial graph convolution is favored for its simplicity and lower computational requirements making it more popular in practice [20].

GCNs have been developed that employ both types of techniques. The operation of the majority of GCNs can be divided into the Message Passing phase and the Readout phase. Typically, the initial phase is implemented through a GCN layer that takes the adjacency matrix and the node feature matrix as input. In this phase, each node receives messages from its neighboring nodes, which are generated based on their features. The Message Passing phase consists of two steps: aggregation and update. In the Aggregate step, nodes gather messages from their neighbors by performing a weighted sum operation. This is followed by the Update step, where the aggregated messages undergo a non-linear transformation, such as the ReLU function, to generate updated node representations. If multiple GCN layers are stacked, a deep GCN can be created, allowing the passing of messages between neighbors that are more than one hop away. In the Readout phase, the learned node embeddings serve as the final representations of the nodes or the entire graph [22].

2.4 Reinforcement Learning

Reinforcement learning (RL) is a essential concept in the GCPN model and is considered one of the three main categories of machine learning, alongside supervised learning and unsupervised learning. Reinforcement learning problems involve learning how to map different situations to appropriate actions in order to maximize a numerical reward signal. These are closed-loop problems because the actions taken by the learning system impact subsequent inputs. The learner is not instructed on which actions to perform. Instead, it must explore and try different actions to determine which ones will produce the highest reward [23].

The Reinforcement learning framework consists of two fundamental elements: the agent and the environment. The agent is an autonomous entity (software program), that explores and learns from the environment. The environment encompasses everything outside the agent and is dynamic. The agent and the environment interact constantly [23][24].

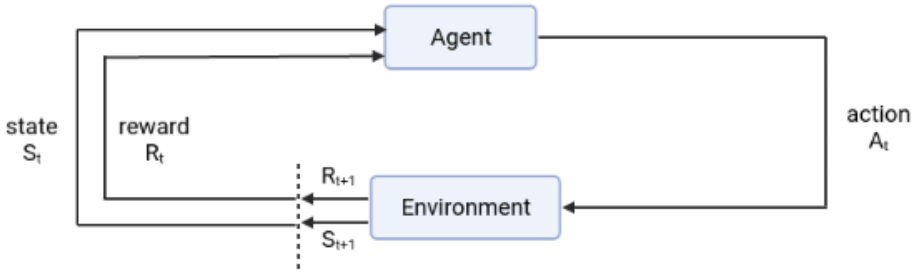


Figure 4. A diagram that represents the interaction between the agent and the environment within the framework of reinforcement learning. Adapted from *Reinforcement Learning: An Introduction* (p. 54), by Sutton, R. S., & Barto, A. G., 2014, The MIT Press. Created with BioRender.com.

At each time step *t*, in a sequence of discrete steps *t*=0,1,2..., the agent and the environment interact with each other (Fig. 4). The agent is provided with a representation of the environment's state, denoted as s_t, at each time step *t*. The state s_t is an element of the set S, which encompasses all possible states, also referred to as the action space. Upon evaluating s_t, the agent chooses an action a_t from the set $A(s_t)$, encompassing all available actions in s_t. Subsequently, the environment undergoes a state transition in response to the action chosen by the agent. The dynamic transition establishes the transition probabilities. After one time step, the agent receives a numerical reward r_{t+1} and receives the next state, denoted as S_{t+1} [23]. At each time step, the agent maps the states to the probabilities of selecting each possible action. This mapping is commonly referred to as the agent's policy, represented as $_t$ [24].

Another key concept is value, which represents the expected cumulative rewards that an agent anticipates receiving in the long run. It is accompanied by a corresponding function.The value allows the agent to select actions that will provide greater long-term rewards, rather than only focusing on actions that yield higher short-term rewards [24].

Reinforcement learning problems can be represented as Markov Decision Processes. Markov Decision Processes (MDP) is a mathematical framework employed for modeling decision-making problems. It is a tuple (S, A, P, R), where S represents the set of states, A denotes the set of actions, P signifies the transition function, and R represents the reward function. MDP relies on the Markov Property, which is of great significance as it indicates that the environment only depends on the current state and action, and is independent of the system's past [25].

As we have studied, the objective of RL is to maximize the long-term rewards received by the agent. To achieve this, it is necessary to optimize the policy π_t. There are multiple algorithms that allow for this optimization, known as reinforcement learning algorithms. Some of the well-known algorithm groups include model-based methods, value-based methods, and policy gradient methods. You et al. decided to use Proximal Policy Optimization (PPO). This algorithm was developed by OpenAI in 2017 and employs gradient-based optimization techniques. The goal of PPO is to avoid large updates to the policy π_t by clipping the ratio between the two policies to a certain range of values [26].

2.5 Architecture and Operation of GCPN

Once we have explored the various elements and concepts present in the model at a general level, we study in more detail the architecture and operation of the Graph Convolutional Policy Network.

2.5.1 Problem definition

The graph G is represented by the adjacency matrix A, node feature matrix F, and edge-conditioned adjacency tensor E. E is a multidimensional tensor that encodes the adjacency matrix of the graph as well as any additional edge features.

The main objective of the model is to generate molecules that optimize a certain property. To achieve this, You et al. incorporate two sources of prior knowledge into the generative model. The first source is embedded within the model's code itself. For example, it includes information about chemical valency. The second source is a dataset consisting of 250,000 real molecules from the Zinc database. The Zinc database contains millions of commercially available molecules and is widely used in pharmaceutical research. The model aims to generate molecules that resemble those present in the dataset. To accomplish this, an adversarially trained discriminator is incorporated, which allows for comparison between the generated molecules and real molecules and generates a numerical value indicating their similarity.

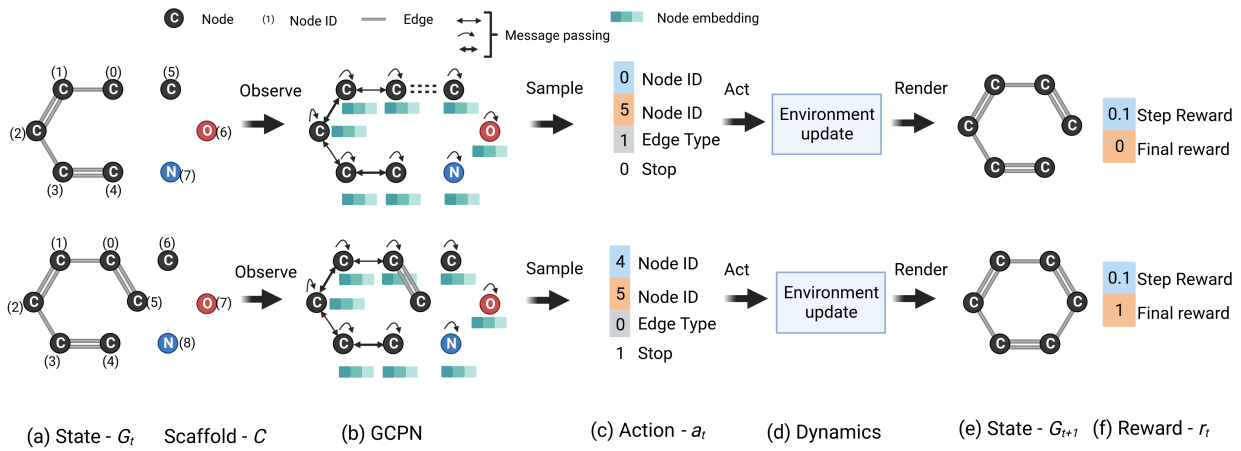


Figure 5. Overview of the generative process. Each row corresponds to a step of the generative process. The first row corresponds to an intermediate step in the process. The second row corresponds to the final step. (a) The intermediate graph G_t defines the state. The set of C atoms are added for the GCPN calculation [C,N,O,S,P,F,I,Cl,Br]. (b) GCPN employs message passing to encode the state into node embeddings. Subsequently, it generates a policy π_0. (c) An action of the policy is sampled. It is constitute of four elements: two Node IDs, edge type and stop. (d) The environment checks the chemical valency. Subsequently, it returns to the following state G_{t+1} (e) and the reward (f). Adapted from "Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation," by J. You, B. Liu, R.Ying, V.Pande, J.Leskovec, 2018, NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, Pages 6412–6422. Created with BioRender.com

2.5.2 Molecule Generation Environment

The environment is described by four fundamental aspects: state space, action space, state transition dynamics, and reward design. We analyze each of them.

- State Space. Each state of the environment, denoted as s_t, corresponds to an intermediate graph generated, denoted as G_t, at each time step. The RL agent has complete visibility of G_t. It is important to highlight that the initial graph, G_0, always starts with a carbon atom. Fig. 5(a) shows an intermediate graph before the action is performed, while Fig. 5(e) shows it after the action has been performed.
- Action Space. The action space is distinct, fixed-dimensional, and homogeneous. A set of scaffold subgraphs is defined, represented as C_1,C_2...C_n. These will be added during the generation of the graph. The set C corresponds to a set of atoms: carbon, nitrogen, oxygen, sulfur, phosphorus, fluorine, iodine, chlorine and bromine. An important concept is the extended graph, which is based on the union of G_t and C (Fig. 5(a)). There are two possible types of actions. The first type consists of generating a link between a node

in G_t and C_i. The second type consists of joining two already existing nodes within G_t. After selecting the action, any scaffold subgraphs that are not connected are removed.

- State Transition Dynamics. It incorporates domain specific rules. The policy network suggests actions to the environment, but some of these actions may be infeasible. In such cases, the environment rejects these actions, and the state remains unchanged as it adheres to a set of predefined rules. Importantly, this model incorporates chemical rules. Fig. 5(d) illustrates the successful completion of both actions after passing the valency check.
- Reward design. The model differentiates between two types of rewards that influence the agent's behavior: intermediate rewards and final rewards (Fig. 5(f)). Intermediate rewards include step-wise validity rewards, which involve assigning a positive or negative reward based on whether or not valency is exceeded. They also include the intermedial adversarial rewards. Final rewards are calculated considering the scores of various significant properties such as logP, penalized logP, druglikeness, synthetic accessibility, etc. Final rewards also include certain penalties if the molecule does not pass certain filters and the final adversarial rewards. Later on, we will analyze each of these rewards in detail.

2.5.3 Graph Convolutional Policy Network

GCPN acts as an agent. The GCPN takes an intermediate graph G_t and a set of subgraphs C as input, and outputs the action a_t that predicts the new bond to add (Fig. 5(b)(c))

The model computes the node embeddings of an input graph using a GCN. Specifically, a variant of GCN is employed that allows incorporating categorical edge types. The approach involves conducting message passing over each edge type for L layers. At the l layer, all the messages from different edge types are aggregated to compute the node embeddings of the next layer, represented as H^{l+1}.

The final node embedding matrix X is obtained at the output of the last graph convolutional layer. X is the base on which actions predictions are made. Each action a_t consists of four components: two nodes, a prediction of the edge type, and a prediction of termination. When selecting nodes to form a bond, it is important to consider that the first node will always be part of G_t, while the second node can either be part of C (available atoms) or be a node of G_t.

A predicted probability distribution is used to sample each of the four components that constitute a_t. The following equations govern the process:

$$a_t = CONCAT(a_{first}, a_{second}, a_{edge}, a_{stop}) \tag{1}$$

$$
\begin{aligned}
f_{first}(s_t) &= SOFTMAX(m_f(X)), & a_{first} &\sim f_{first}(s_t) \in \{0,1\}^n \\
f_{second}(s_t) &= SOFTMAX(m_s(X_{a_{first}}, X), & a_{second} &\sim f_{second}(s_t) \in \{0,1\}^{n+c} \\
f_{edge}(s_t) &= SOFTMAX(m_e(X_{a_{first}}, X_{a_{second}})), & a_{edge} &\sim f_{edge}(s_t) \in \{0,1\}^b \\
f_{stop}(s_t) &= SOFTMAX(m_t(AGG(X))), & a_{stop} &\sim f_{stop}(s_t) \in \{0,1\}
\end{aligned} \tag{2}
$$

The letter m represents a Multilayer Perceptron (MLP) [27], which is a type of artificial neural network. The letter is associated with each of the MLPs (first, second, edge, stop). $SOFTMAX$ is a mathematical function that maps an input vector of real numbers to a probability distribution over a set of classes. The application of $SOFTMAX$ is performed on the output of the MLP [23].

m_f takes the final node embedding matrix as input and, in combination with the $SOFTMAX$ function, produces a vector that represents the probability distribution for selecting each node. To select the second node a_{second}, it is crucial to include the information of the first selected node a_{first} by concatenating its embedding $X_{a_{first}}$. Afterwards, m_s maps the embedding to a probability distribution encompassing all potential nodes that can be chosen as the second node in the graph. Then, m_e predicts the link based on the embeddings of the first and second nodes. Finally, the stop prediction is accomplished by incorporating the node embeddings into a graph embedding, which is subsequently mapped to a scalar value by m_t. After this process, a_t is ready to be sent to the environment.

2.5.4 Model Training

As we have mentioned before, it is essential to incorporate information from real molecules into the model. This information allows the model to generate realistic molecules and incorporate certain properties of interest for the designer. To incorporate this information, the model includes a Generative Adversarial Network framework that generates adversarial rewards. In general, a GAN consists of two competing networks: the generator and the discriminator. The generator generates synthetic data and passes it to the discriminator, which compares it with real data and tries to determine if it is real or synthetic. The objective is for the generator to be able to generate data that is indistinguishable from real data for the discriminator. Therefore, You et al. introduced a discriminator, specifically a graph convolutional network, which aims to differentiate between real and generated molecules by comparing a real molecule from the dataset with the generated molecule. Additionally, to incorporate more information, a subgraph of the molecule is used, not the complete molecule. The training is divided into two parts:

- Supervised Learning. The generative model is trained using a supervised learning approach. Its goal is to train the model to generate realistic molecules. The policy is trained by imitating the actions taken in real observed graphs and the GAN framework is employed. Model parameters are updated and optimized using the gradient descent algorithm.
- RL training. The primary objective is to optimize rewards, and Proximal Policy Optimization (PPO) is employed to optimize the model parameters.

3 Implementation of the model including the molecular formula

In this section, we examine the modifications implemented to the GCPN model to generate realistic molecules with a specific molecular formula specified by the user as a parameter. The code can be found in Appendix A.

3.1 The importance of including the molecular formula in the generative model

The molecular formula is an essential concept of chemistry that represents the type and number of atoms that constitute a molecule. In most cases, a single molecular formula represents multiple molecules. These molecules have the same types and numbers of atoms but differ in their spatial or structural arrangement, resulting in different properties. Molecules with the same molecular formula are known as isomers.

Including the molecular formula in the generative model allows us to generate the maximum possible number of isomers for the specified molecular formula. This fact has various applications.

As mentioned in the introduction, pharmaceutical research faces the challenge of exploring the discrete chemical space formed by 10^{60} drug-like compounds. The generative model proposed by You et al. allows for the generation of a large number of molecules, significantly facilitating the exploration of this space. However, the distribution of molecular formulas is unknown. It is possible that the model may not generate molecules with certain molecular formulas. By imposing a molecular formula, we ensure the generation of thousands of isomers for that formula. Therefore, we could create a dataset that includes millions of different molecular formulas and systematically introduce them into the generative model, generating thousands of isomers. This approach would enable us to systematically, effectively, and comprehensively explore the chemical space.

As mentioned before, isomers exhibit different properties. However, when they share the same molecular formula, they can also have certain similar or even identical properties, such as molecular weight. This fact allows researchers to have a preliminary idea of which formulas may be more useful and generate thousands of isomers using the generative model.

It is also of great utility in metabolomics. As mentioned in the introduction, one of the major challenges in metabolomics is the identification of spectra. There is a large number of spectra that frequently appear in plasma and urine samples, for which their corresponding molecular formula is known but the structure of the molecules is unknown. These spectra are grouped in the ARUS database (Annotated Recurrent Unidentified Spectra), maintained by the National Institute of Standards and Technology [10]. With the modified generative model, we can generate thousands of isomers for each of the molecular formulas in the ARUS database. The generated isomers, along with in silico spectrum generation tools, can help us identify each of the spectra.

3.2 Model development

To incorporate the molecular formula into the model, we have made modifications to the molecule environment. The MoleculeEnv class defines the molecule environment. MoleculeEnv contains the methods to define the molecule environment, add atoms and bonds, modify bonds, reset the process, generate rewards, check chemical validity, get final results and get expert information from the molecule database.

As we explained in the previous section, the set C contains the atomic symbols for carbon, nitrogen, oxygen, sulfur, phosphorus, fluorine, iodine, chlorine, and bromine. The agent selects one of the atoms from this set to add it to the intermediate graph G_t. The agent can select any atom from the set C as many times as necessary. Therefore, molecules can incorporate atoms up to the established limit of 38 atoms. This limit cannot be modified as it leads to multiple errors due to variations in the size of several tensors.

Taking this explanation into account, we have proposed a solution for generating molecules with a specific molecular formula (Fig. 6). The solution is based on having two lists. The first list (atom_list) includes only the atomic symbols of the atoms that make up the molecular formula, and it is immutable. It replaces the set C. Therefore, the agent can select any type of atom from atom_list to add it to the intermediate graph G_t. The use of this list allows us to use only the specified types of atoms in the molecular formula, but it does not allow us to establish the number of atoms. The second list (formula_list) solves this problem. Formula_list is composed of the atoms that formed the molecular formula, and they are repeated as many times as indicated by the formula. Formula_list is updated every time an atom is added to G_t, replacing the atom type used with the ASCII character $. When all elements of formula_list have been replaced with a $, no new atoms can be added to G_t, but intramolecular bonds can still be established.

We ran the code with the modifications. However, occasional errors occurred that interrupted the execution. We identified two types of errors. The first type of error was caused by actions that generated a bond between the intermediate graph and an out-of-range NodeID. The second error occurred when the action contained two identical NodeIDs, indicating an attempt to create a self-bond. To solve these errors, we established a series of Boolean conditions that discarded such actions.

Later, we executed the code again. We decided to input the molecular formula $C_8H_9NO_2$. The generated molecules only contained the specified atom types in the atom_list. However, only a few molecules utilized all the atoms from the formula_list. The majority had utilized between 3 and 7 atoms from the formula_list. The made modifications allowed us to specify the atom types that would form the molecules and limit the number of atoms according to the specifications of the molecular formula. However, to predominantly generate molecules with the specified atoms, it was necessary to incorporate information from real molecules that presented the specified molecular formula. We replaced the dataset from Zinc, consisting of 250,000 molecules, with 2,500 molecules with the molecular formula $C_8H_9NO_2$. The utilization of subgraphs from real molecules within the GAN framework enables this significant reduction. Despite the reduction, a lot of information is still incorporated. When we ran the code again, the majority of molecules exhibited the same atoms as Formula_list. Therefore, it is necessary to change the dataset every time a molecular formula is introduced. Hence, a different model is generated for each molecular formula.

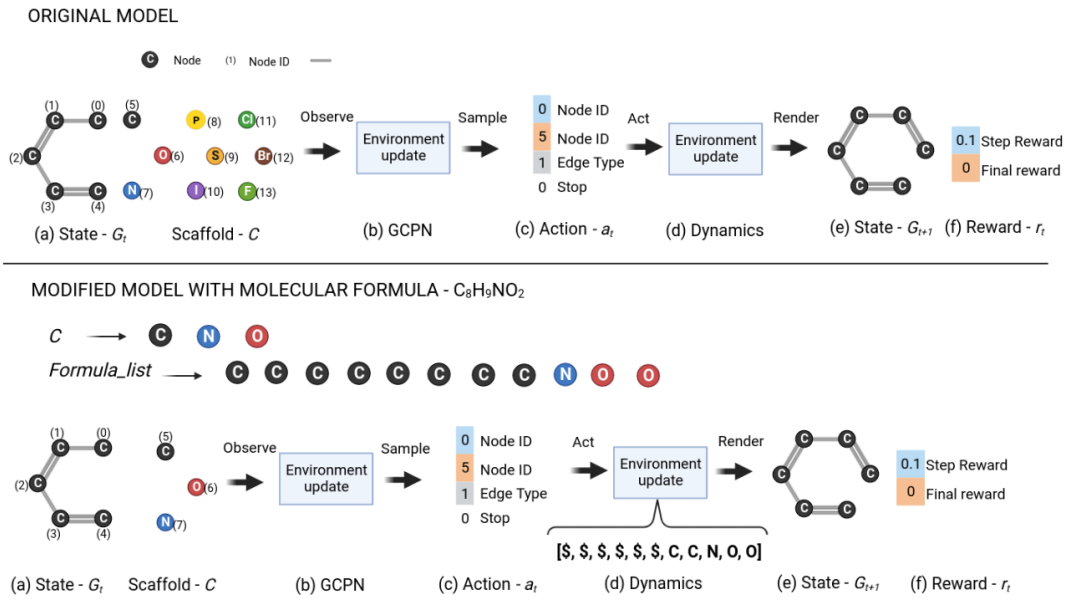


Figure 6. Comparison between the original model and the model modified with the molecular formula. The input provided is $C_8H_9NO_2$. We can observe the set C from the original model formed by [C,O,N,P,S,F,I,Cl,Br]. The set C from the modified model (atom_list) is formed by [C,O,N]. We can observe the formula_list formed by the atoms indicated by $C_8H_9NO_2$. It is important to emphasize that six carbon atoms have been used and replaced with the symbol $

3.2.1 Expert data

We obtain the molecules for the data training from the PubChem database. PubChem is a database of molecules maintained by the National Center for Biotechnology Information. It contains 115 million compounds and 306 million substances [28]

	CanonicalSMILES
0	CC1=CC(=C(C=C1)[N+](=O)[O-])C
1	CC(=O)NC1=CC(=CC=C1)O
2	CC1=CC(=CC=C1)OC(=O)N
3	C1OC2=C(O1)C=C(C=C2)CN
4	CC1=CC(=C(C=C1)N)C(=O)O
...	...
2955	CC1=C(C=CN=C1OC)C=O
2956	CC(=O)NC1=CC=C(C=C1)O
2957	CC(=NO)C1=CC=C(C=C1)O
2958	CC1=C(C=CC(=C1)C=NO)O
2959	CC1=CC(=C(C=C1)C=NO)O

Figure 7. Canonical Smiles corresponding to $C_8H_9NO_2$ extracted from PubChem.

PubChem allows for searching and downloading all the isomers for a specific molecular formula. We downloaded the results in CSV format. This file contains a lot of information, specifically 40 columns. However, we are only interested in the column that contains the canonical SMILES (Fig. 7). The canonical SMILES variant represents the unique, standardized representation of a molecular structure. The model works with molecular graph representation; however, the molecules stored in the training dataset (expert data) and the generated molecules are stored as SMILES.

3.2.2 Planarity

Planarity is a property exhibited by certain graphs, indicating that the graph can be represented on a plane without any edge crossings. However, the complexity of graphs often makes it challenging to visually determine their planarity. To determine whether a graph is planar or non-planar, methods such as Kuratowski's theorem or planarity testing techniques are employed. In the case of small organic molecules, their molecular graph representations are generally planar.

We depicted the molecules and observed that many of them displayed bond crossings (Fig. 8). Consequently, we decided to verify whether these molecular graph representations were planar or non-planar. We installed the planarity package, which allowed us to quickly determine the planarity of the molecular graphs. We observed that a significant number of molecular graphs were non-planar. The model is constrained to generate molecules considering the molecular formula. Hence, it is more likely for the agent to choose unusual actions that generate non-planar molecular graphs.

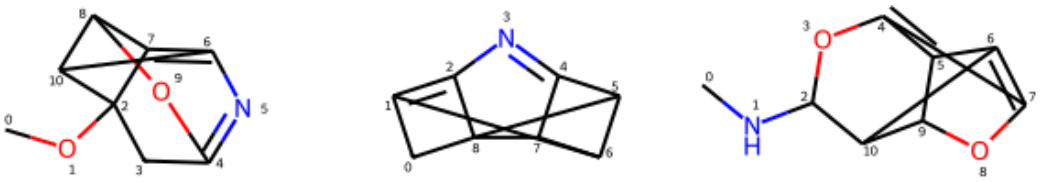


Figure 8. Non-planar molecular graphs generated for $C_8H_9NO_2$. In the case of the first and third molecules, they consist of all $C_8H_9NO_2$ atoms (excluding hydrogen). The second molecule respects the type and number of atoms, but it does not include oxygen.

We proposed a solution that proved to be effective and reduced the number of non-planar molecules. The solution is based on modifying the generation of rewards, specifically the generation of chemical validity rewards. Chemical validity rewards are granted at the end of the generation of each molecule. In the original model, all final graph rewards are rewarded with +2 (reward_valid), and if they are chemically invalid, they are penalized with -5 (reward_valid). However, we observed that the chemical validity filter was quite permissive. Additionally, the molecule must pass two filters (Zinc, Steric). Each filter, if not passed, penalizes with -1 (reward_valid). The modified model incorporates a boolean condition that checks whether

the graph is planar or non-planar. If the molecular graph is non-planar, it is penalized with -5 (reward_valid).

We also included this boolean condition in the generation of intermediate rewards. We check whether the intermediate graph is planar after generating a bond. If it is not planar, it is penalized with -1 (reward_step). Only the first bond that makes the molecular graph non-planar is penalized

After incorporating these changes, we retrained the model and visualized the molecules. We observed a significant reduction in the number of non-planar molecules, and they exhibited a more realistic appearance.

4 Evaluation of Generated Molecules

After studying the modifications made to the model, we analyze the generated molecules. The results are stored in a CSV file. This file contains the generated molecules in SMILES format, along with various rewards and information from different filters established by You et al. The rewards assigned to each molecule provide us with information about its realism and feasibility. Therefore, we select those molecules that meet the exact molecular formula and have certain rewards above predefined thresholds. Additionally, we have employed Mol2Vec. Mol2Vec is a method that generates fixed-dimensional numerical embeddings for each molecule based on its chemical structure. This method is applicable to known molecules as well as de novo molecules. This type of representation allows us to numerically compare the similarity between generated and real molecules [29]. All the analysis code and experiments have been developed in Python and can be found in Appendix A.

4.1 Output Data Structure

As mentioned earlier, the model generates a CSV file with the generated molecules and their rewards. Next, we analyze the 12 columns that compose the dataset (Fig. 9).

smile	reward_valid	reward_qed	reward_sa	final_stat	rew_env	rew_d_step	rew_d_final	cur_ep_ret	flag_steric_strain_filter	flag_zinc_molecule_filter	stop
C#COON1C#C1	0.0	0.266166	0.417471	-1.865897	-1.878092	0.000000	1.013263e+00	-2.065803	False	False	False
C1=C2COC3=C4N1C234	1.0	0.407639	0.445083	-1.645807	-0.658003	0.000000	1.007315e+00	-0.633300	False	True	True
O=C=C1N=C1C1=C=CO1	1.0	0.364593	0.494661	-1.469809	-0.482004	0.000000	1.267836e+00	-0.207414	False	True	False
CC12C(=O)C13N2C31C2=C1O2	1.0	0.413098	0.413441	-1.946472	-0.958667	0.000000	1.034391e+00	-0.730534	False	True	False
OC12C3=NC1(C3)C2=C1C#C1	1.0	0.445035	0.334599	-2.171720	-1.159525	0.000000	9.717817e-01	-1.104537	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...
C1=CC2=CC3=C(NCC23)O1	1.0	0.529786	0.453403	-3.782249	-2.794444	0.000000	1.528703e-05	-3.810196	False	True	True
NC1=C2CC(=C2O)C(=O)CC1	1.0	0.537317	0.640682	-1.971457	-0.983652	0.000000	2.092417e-05	-1.857760	False	True	True
O=C1C2=C3C4C2C134	1.0	0.412762	0.413345	-1.775935	-0.788130	0.000000	7.315633e-14	-1.853927	False	True	True
CCC1=CC2=NC(=O)OC2C1	2.0	0.571548	0.612032	-0.698052	1.289753	0.000226	8.164992e-04	0.400351	True	True	True
C#CC1(C(C)=O)C(=O)C1C=N	2.0	0.342201	0.549231	-1.305873	0.681932	0.002226	1.574119e-04	-0.298099	True	True	True

Figure 9. Output dataset for $C_8H_9NO_2$.

- **smile**: Column containing the generated molecules in SMILES format.
- **reward_valid**: The column represents the chemical validity of the generated molecules. Each generated molecule is initially assigned a reward of +2. If the molecule is not valid, it is penalized with -5. We have observed that the chemical validity filter is quite permissive. If the molecular graph is non-planar, a penalty of -5 is applied. Finally, the steric strain filter and the zinc molecule filter penalize molecules that do not pass them with a score of -1.
- **reward_qed**: This reward is highly significant for computational drug discovery. The reward_qed is calculated using the Quantitative Estimate of Drug-likeness (QED) method. QED is a well-known computational approach that generates a score ranging from 0 to 1 to assess whether a chemical compound exhibits drug-like properties [30]. The method is implemented in RDKit (Appendix B).

- **reward_sa**: Reward that quantifies the synthetic accessibility. The reward is normalized between 0 and 1, where values closer to 1 indicate a higher ease of synthesis for the molecule.
- **final_stat**: The final reward corresponds to a specific property and is calculated differently based on the property being optimized such as penalized logP, QED, etc.. The default optimization is for penalized logP, which considers the ring size and synthetic accessibility when calculating the logP score.
- **rew_env**: The reward received by the agent after taking an action is a combination of various components. If the molecule is complete, it includes the reward_valid, final_stat, and reward_step. When the molecule is still being constructed, the reward_env is equal to the reward_step.
- **rew_d_step**: It represents the intermediate adversarial reward generated by the GAN, and its values range from -1 to 1.
- **rew_d_final**: It represents the final adversarial reward generated by the GAN, and its values range from -1 to 1
- **cur_ep_ret**: Reward that combines rew_env, rew_d_step, and rew_d_final. Since it encompasses multiple rewards, it provides us with a lot of information about each molecule.
- **flag_steric_strain_filter**: Filter designed to reject unrealistic molecules. As mentioned earlier, if the molecule does not pass the filter, it is penalized with -1 (reward_valid).
- **flag_zinc_molecule_filter**: Filter designed to detect problematic functional groups. As mentioned earlier, if the molecule does not pass the filter, it is penalized with -1 (reward_valid).
- **stop**: It is a boolean variable that informs us whether the agent signaled the end of the process (True) or if the process ended for another reason (False).

4.2 Analysis of the number of generated molecules

First, we focus on analyzing the number of molecules generated by the modified model with the specified number of atoms in the molecular formula (excluding hydrogen). In this analysis, we were specifically interested in observing the potential influence of the number of atoms present in the molecular formula and the size of the training dataset.

First, we employed $C_8H_9NO_2$. We decided to use this molecular formula because it has a significant number of known isomers (2983), it is composed of three different types of atoms (excluding hydrogen), and it consists of a small number of atoms (11). The generative model generated 80119 molecules. Out of these, 39885 molecules had the specified number of atoms according to the molecular formula, which represents 49.78% of the total (Fig. 10(a)). Therefore, they constituted the majority of the molecules. The remaining molecules mostly consumed 9 or 10 atoms from the Formula_list. These results demonstrated that the

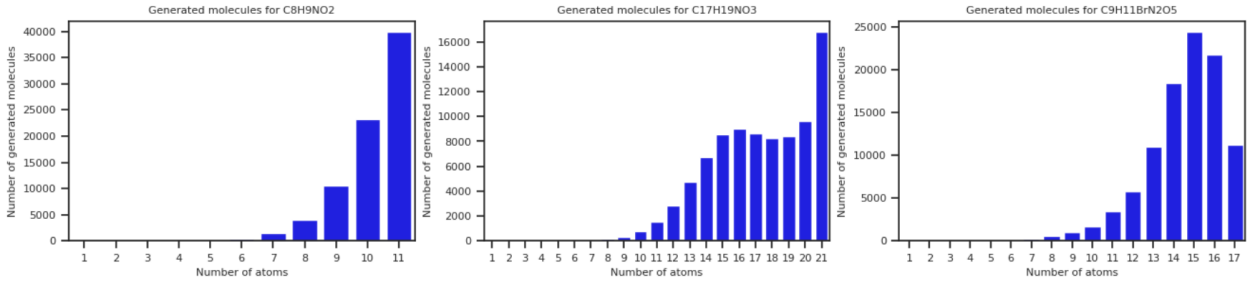


Figure 10. Number of molecules generated with different numbers of atoms for $C_8H_9NO_2$, $C_{17}H_{19}NO_3$ and $C_9H_{11}BrN_2O_5$

modifications made, along with the appropriate training dataset, allow us to generate mostly molecules with the specified type and number of atoms according to the molecular formula.

Next, we introduced $C_{17}H_{19}NO_3$ into the model. We decided to use this formula because it consisted of the same types of atoms as in the previous case but in a larger quantity. This allowed us to observe the influence of a higher number of atoms on the results, which translates to more possible combinations. $C_{17}H_{19}NO_3$ consists of 16,121 known isomers. The DGM generated 86,675 molecules, out of which 16,796 had the specified number of atoms according to the molecular formula. As shown in Fig. 10(b), the bar for 21 atoms is much larger than the rest. However, it represents 19.37% of the total. As the number of atoms increases, the number of possible combinations also increases, as reflected in the bars ranging from 15-20 atoms with values around 9000. Therefore, the modifications applied to the model also allow for the generation of larger molecules. However, the percentage of molecules that use all the atoms in the molecular formula decreases.

As mentioned earlier, the model is still capable of generating realistic molecules despite a significant reduction in the training dataset, thanks to the utilization of subgraphs by the GAN framework. In the PubChem database, we noticed that certain molecular formulas have very few isomers. Therefore, we wanted to assess whether the modified DGM was capable of generating molecules using all the atoms from the Formula_list while using a significantly reduced training dataset. For this purpose, we introduced $C_9H_{11}BrN_2O_5$ into the model. This molecular formula has 120 known isomers. The generative model produced 100,026 molecules, of which 11,245 had the specified number of atoms (17), representing 11.2% of the total. In Fig. 10(c), we observe that the column for 17 atoms is the fourth highest bar. These results indicate that a larger training dataset is necessary to obtain more information about the molecules and predominantly generate molecules with the appropriate molecular formula. However, considering that there are numerous molecular formulas with few known isomers, the modified model remains a useful tool for exploring chemical space.

4.3 Filtering of generated molecules

In order to obtain realistic molecules that satisfied the specified molecular formula, including hydrogen atoms, it was necessary to apply a filtering process to the results. This was accomplished by utilizing the rewards obtained from the output data.

We utilized two types of rewards: reward_sa and cur_ep_ret. As previously explained, reward_sa evaluates synthetic accessibility and ranges from 0 to 1. Higher values, close to 1, indicate greater ease of synthesizing the molecules. We observed that molecules with low values were unrealistic. Therefore, we decided to select molecules with a value of 0.7 or higher. On the other hand, cur_ep_ret encompasses rewards from the environment and rewards from the GAN, indicating similarity to real molecules in the training dataset. This reward covers a wider range of values, including negative values. Therefore, we selected molecules with a cur_ep_ret value greater than or equal to 1.

The filtering process continued with the selection of molecules that had exactly the same molecular formula as the one inputted into the generative model, including hydrogen atoms.

Finally, we wanted to check if any of the generated molecules existed or were completely new. To search for the molecules in PubChem, we used the PubChemPy package [31], which allows interacting with PubChem in Python. We added a column to each dataset indicating whether the molecule was found in PubChem or if it was new.

4.3.1 Filtering the $C_8H_9NO_2$ dataset

We applied the filter formed by the conditions reward_sa >=0.7 and cur_ep_ret >=1, and 1941 molecules passed the filter. Next, we filtered the molecules by exact molecular formula ($C_8H_9NO_2$) and obtained 585 molecules. Additionally, we found 165 molecules in the PubChem database. This result was of great significance as it indicated that the generative model was capable of generating realistic molecules and confirmed that the filtering conditions were correct (Fig. 11).

	smile	reward_valid	reward_qed	reward_sa	final_stat	rew_env	rew_d_step	rew_d_final	cur_ep_ret	strain_f	zinc_f	stop	formula	database
	CC1=CC=CC(=CC(=O)O)N1	2.0	0.549872	0.725062	-0.446523	1.541282	0.019655	0.737387	1.377404	True	True	True	C8H9NO2	False
	CC(=O)C(=O)C1=C(C)C=C1N	2.0	0.577871	0.733301	-0.579755	1.408050	0.001054	0.769918	1.135343	True	True	True	C8H9NO2	False
	NCCC1C2=C1OC(=O)C=C2	2.0	0.666570	0.704152	-0.657548	1.330257	0.000454	0.825659	1.147087	True	True	True	C8H9NO2	False
	C=CC1=C(CO)C=CC(O)=N1	2.0	0.660475	0.755140	-0.360480	1.627324	0.010165	0.640629	1.302077	True	True	True	C8H9NO2	False
	CC=CC=C1C(=O)C(=O)N1C	2.0	0.310463	0.735280	-0.533057	1.454748	0.013818	0.459611	1.033689	True	True	True	C8H9NO2	False
	...	...	...	...	...	...	...	...	...	...	...	...	...	...
	CC=C(O)C1=CC=C(O)C=N1	2.0	0.600489	0.794312	-0.037410	1.950395	0.001272	0.026791	1.020526	True	True	True	C8H9NO2	False
	O=C1NC2=C(CCCC2)C1=O	2.0	0.512670	0.793541	-0.317344	1.670461	0.001329	1.360160	2.048113	True	True	True	C8H9NO2	True
	CC(=O)C1=C(N)C=CC(O)=C1	2.0	0.359583	0.888592	0.178982	2.166787	0.003216	1.801816	3.049306	True	True	True	C8H9NO2	True
	CC1=C(C)C(O)=C(C=O)C=N1	2.0	0.613670	0.795028	-0.148583	1.839222	0.001003	1.501396	2.400920	True	True	True	C8H9NO2	False
	O=C1CCCNC2=C1C=CO2	2.0	0.612822	0.768003	-1.306347	0.681458	0.010810	1.701537	1.465015	True	True	True	C8H9NO2	True

Figure 11. Resulting dataset from the filtering process for $C_8H_9NO_2$. The original columns of the output dataset (rewards, filters, stop) can be observed, as well as the columns corresponding to the molecular formula and the existence of the molecule in PubChem

We depicted the molecules to perform a qualitative analysis (Fig. 12). We were able to observe that the molecules exhibited a realistic appearance, with identifiable functional groups, rings, and common structures. We also noticed less common structures such as squares and heptagons.

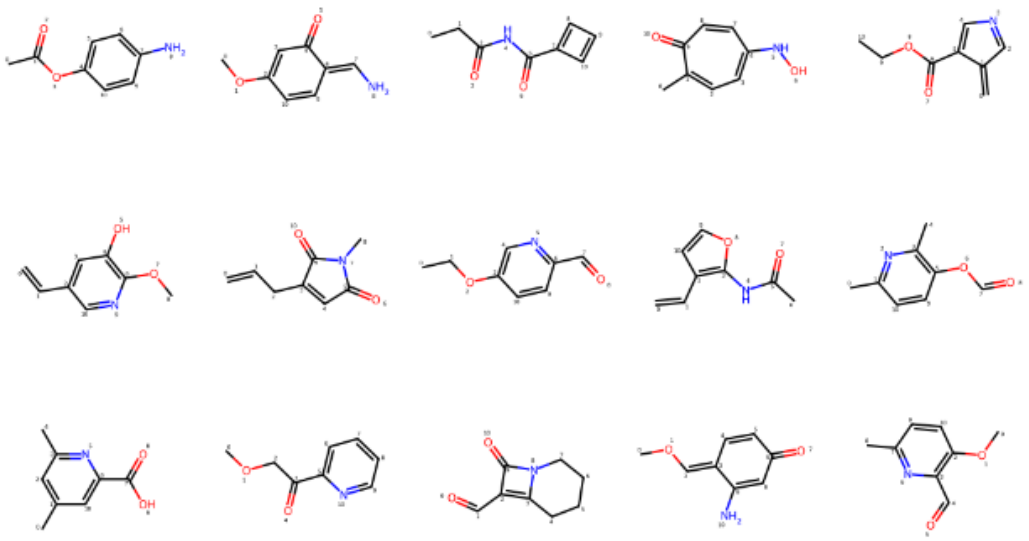


Figure 12. Some generated molecules for $C_8H_9NO_2$ are shown. Realistic and diverse molecules can be observed, including common functional groups and rings. There are also less common structures such as squares and heptagons

4.3.2 Filtering the $C_{17}H_{19}NO_3$ dataset

We applied the filter formed by the conditions reward_sa >=0.7 and cur_ep_ret >=1, and 3,261 molecules passed the filter. Next, we filtered the molecules by exact molecular formula ($C_{17}H_{19}NO_3$) and obtained 83 molecules (Fig. 13).

If we compare the molecules that pass the first filter for $C_{17}H_{19}NO_3$ (3,261) and $C_8H_9NO_2$ (1,941), there is a notable difference, considering that approximately the same number of molecules were generated. We attribute this difference to the size of the training dataset. In the case of $C_8H_9NO_2$, it consisted of 2,500 isomers out of the 2,983 available in PubChem. The training dataset for $C_{17}H_{19}NO_3$ was composed of 14,000 isomers out of the 16,121 present in PubChem. By incorporating more information, the model is able to generate a greater number of realistic and diverse molecules.

We also observe that the number of molecules that exactly match the specified molecular formula decreases compared to the previous case. None of the molecules were found in PubChem.

smile	reward_valid	reward_qed	reward_sa	final_stat	rew_env	rew_d_step	rew_d_final	cur_ep_ret	strain_f	zinc_f	stop	formula	database
C=CC=CC1=CC2=C(C(C(=O)CC)=C2)N(CCOC)C1=O	2.0	0.723275	0.733476	-0.065572	1.922233	0.000667	0.466442	1.888926	True	True	True	C17H19NO3	False
CC=CC(C=CC1=CC=CC=C1)=C(O)C(=NC=O)OCC	2.0	0.285035	0.744555	0.242115	2.229920	0.001784	0.383035	2.141394	True	True	True	C17H19NO3	False
C=CC(C=O)NC(=O)C1=CC(OCC)=C(C2=CCC2)C=C1	2.0	0.618827	0.752470	0.053639	2.041444	0.000341	0.157774	1.524321	True	True	True	C17H19NO3	False
C=CC(=C)NC1=CC(O)=C1C1=CC(O)=C(OCCC)C=C1	2.0	0.669349	0.750105	0.252215	2.240020	0.013505	0.034708	1.880231	True	True	True	C17H19NO3	False
C=CC1=CC=C(OC2=C(CO)C=CC=C2)C(OC)=C1CN	2.0	0.855742	0.836694	0.434792	2.422597	0.002784	1.209856	3.094455	True	True	True	C17H19NO3	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
CCOCC1(C2=CC=CC(C(=O)O)=C2)CC2=C(C)C=C2N1	2.0	0.873116	0.704558	-0.100656	1.887149	0.005174	0.342821	1.858532	True	True	True	C17H19NO3	False
C=CC=C(O)C1=CC(OC)=C(C)C(C2=C(N)C(OC)=C2)=C1	2.0	0.643278	0.742964	0.148736	2.136541	0.000008	0.000397	1.576882	True	True	True	C17H19NO3	False
C=CC=C(C=C)CC(=O)N1COC2=C1C(C)=CC(OC)=C2	2.0	0.778510	0.730838	0.122011	2.109816	0.002372	0.000879	1.635659	True	True	True	C17H19NO3	False
COC(=O)C(C)C(=O)N(CC1=CC=CC=C1)CC1=CC=C1	2.0	0.595046	0.781336	0.058692	2.046497	0.039323	1.723504	3.189811	True	True	True	C17H19NO3	False
COC1=CC(C(=O)OCC2=CC=CC(C)=C2)=CC(CN)=C1	2.0	0.858079	0.903837	0.615524	2.603329	0.006312	0.622984	3.078640	True	True	True	C17H19NO3	False

Figure 13. Resulting dataset from the filtering process for $C_{17}H_{19}NO_3$. The original columns of the output dataset (rewards, filters, stop) can be observed, as well as the columns corresponding to the molecular formula and the existence of the molecule in PubChem

As in the previous case, we depicted the molecules to perform a qualitative analysis (Fig. 14). We observed that the molecules had a realistic appearance, but there were also unfamiliar structures present, such as triangles and squares.

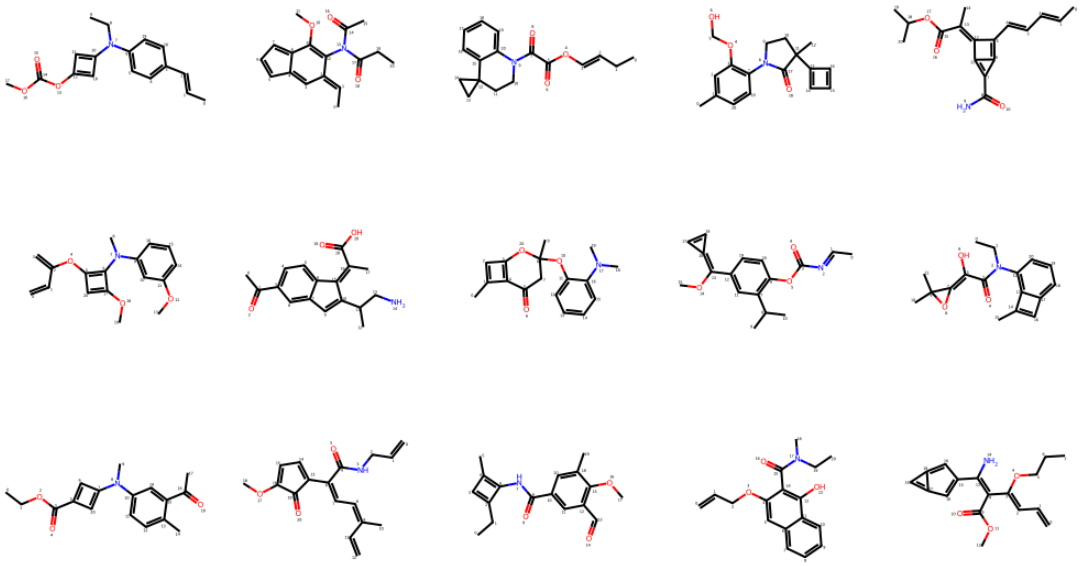


Figure 14. Some generated molecules for $C_{17}H_{19}NO_3$ are shown. Realistic and diverse molecules can be observed, including common functional groups and rings. There are also less common structures such as squares and heptagons

4.3.3 Filtering the $C_9H_{11}BrN_2O_5$ dataset

We applied the filter formed by the conditions reward_sa>=0.7 and cur_ep_ret >=1, and 370 molecules passed the filter. Next, we filtered the molecules by exact molecular formula and obtained 16 molecules (Fig. 15). The number of molecules that meet the conditions

of reward_sa and cur_ep_ret> is much lower than in the two previous cases. The model incorporates limited information (100 isomers out of the 120 present in PubChem), resulting in an increase in unrealistic molecules and a lower variability of generated molecules. This is reflected in the fact that out of the 16 resulting molecules from the filtering process, 3 are found in PubChem (Fig.16). However, the model allowed us to generate 13 new molecules. Therefore, the modified generative model with very small training datasets can be useful for various purposes, but the potential lack of variability should be taken into consideration.

smile	reward_valid	reward_qed	reward_sa	final_stat	rew_env	rew_d_step	rew_d_final	cur_ep_ret	strain_f	zinc_f	stop	formula	database
CCN1C(=O)C(CBr)CC(=O)N(C(=O)O)C1=O	2.0	0.764007	0.717193	-1.673214	0.314882	0.011352	2.728031	2.506586	True	True	True	C9H11BrN2O5	False
O=C1C=C(Br)N(C2(CCO)OCCO2)C(=O)N1	2.0	0.723107	0.702089	-0.919516	1.068580	0.004183	1.720093	2.294252	True	True	True	C9H11BrN2O5	False
COC(CO)CC(=O)N1C(=O)NC(=O)C=C1Br	2.0	0.718943	0.707258	-0.901354	1.086741	0.002373	1.527485	1.941040	True	True	True	C9H11BrN2O5	False
CC(=O)OCC(O)(Br)CN1C=CC(=O)NC1=O	2.0	0.559084	0.715992	-0.905593	1.082502	0.005039	2.320674	2.951770	True	True	True	C9H11BrN2O5	False
COC1=CC(N(C)C(=O)O)=C(Br)N1CC(=O)O	2.0	0.878297	0.766648	-0.194613	1.793482	0.000297	0.001442	1.034297	True	True	True	C9H11BrN2O5	False
CCOC(CC(=O)O)N1C=C(Br)C(=O)NC1=O	2.0	0.814936	0.734248	-0.578276	1.409819	0.029162	1.176024	2.175352	True	True	True	C9H11BrN2O5	False
O=C(O)CCN1C(=O)CCC(=O)N1C(=O)CBr	2.0	0.722626	0.764189	-0.600724	1.387371	0.040928	2.392874	3.469144	True	True	True	C9H11BrN2O5	False
O=C(O)CCN(C(=O)CBr)N1C(=O)CCC1=O	2.0	0.558010	0.770124	-0.579362	1.408733	0.013045	2.275113	3.086664	True	True	True	C9H11BrN2O5	True
O=C1NC(=O)N(C2OCCOC2CO)C=C1Br	2.0	0.746381	0.704420	-0.942083	1.046013	0.000099	1.584552	1.820280	True	True	True	C9H11BrN2O5	False
CC(CO)N1C(=O)C(Br)=CN(CC(=O)O)C1=O	2.0	0.774043	0.746978	-0.741267	1.246828	0.000018	2.676979	3.485454	True	True	True	C9H11BrN2O5	True
O=C(O)CN1C(=O)CCN(C(=O)CCBr)C1=O	2.0	0.739799	0.806778	-0.380437	1.607658	0.000231	0.000001	1.052304	True	True	True	C9H11BrN2O5	False
O=C1NC(=O)N(C2COC(CO)OC2)C=C1Br	2.0	0.746381	0.759263	-0.744694	1.243401	0.011401	0.374653	1.175442	True	True	True	C9H11BrN2O5	False
COCC1OC(N2C=C(Br)C(=O)NC2=O)C1O	2.0	0.762710	0.706756	-0.934047	1.054048	0.000136	0.456797	1.229595	True	True	True	C9H11BrN2O5	False
COC(=O)CC(OC)N1C=C(Br)C(=O)NC1=O	2.0	0.791242	0.739221	-0.630492	1.357603	0.000572	0.228438	1.104051	True	True	True	C9H11BrN2O5	False
O=C(O)CCN(C(=O)CBr)N1C(=O)CCC1=O	2.0	0.558010	0.770124	-0.579362	1.408733	0.036397	2.250854	3.159548	True	True	True	C9H11BrN2O5	True
O=C(O)CN(C(=O)CCBr)N1C(=O)CCC1=O	2.0	0.558010	0.759397	-0.617970	1.370125	0.022090	0.035894	1.042710	True	True	True	C9H11BrN2O5	False

Figure 15. Resulting dataset from the filtering process for $C_9H_{11}BrN_2O_5$. The original columns of the output dataset (rewards, filters, stop) can be observed, as well as the columns corresponding to the molecular formula and the existence of the molecule in PubChem.

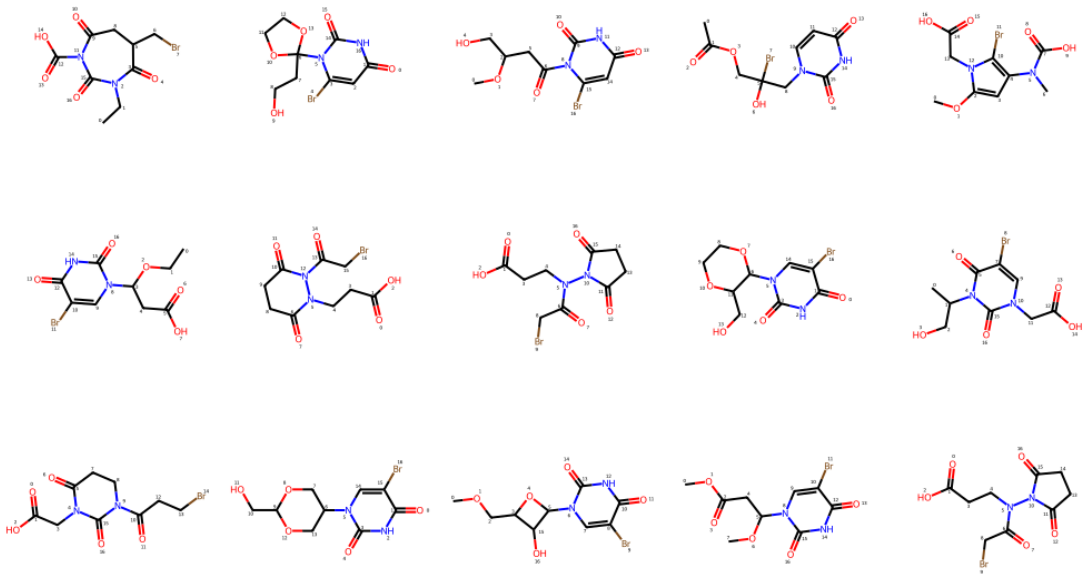


Figure 16. Generated molecules for $C_9H_{11}BrN_2O_5$

4.4 Mol2Vec

The numerical representation of molecules is an essential aspect of cheminformatics. In 2018, an unsupervised method was proposed for learning vector representations of molecular substructures. This method, called Mol2Vec, is based on Word2Vec, a well-known technique in natural language processing (NLP) [29]. Word2Vec is an unsupervised learning technique utilized to generate embeddings for words in a given text corpus, with the goal of capturing semantic and syntactic relationships [32].

Mol2Vec utilizes the substructures obtained from the Morgan algorithm. The Morgan algorithm examines the substructures surrounding the heavy atoms of a molecule within a defined radius, which is typically measured in terms of the number of bonds. Each substructure is assigned a distinct identifier, and these identifiers are then hashed into a fixed-size vector. This approach often draws a parallel between the substructures derived from the Morgan algorithm and words in natural language processing, where chemical compounds can be seen as sentences [29].

In order to generate the Mol2Vec model, the training of a Word2Vec algorithm is necessary. Word2Vec takes molecular sentences as input. To generate these molecular sentences, the Morgan algorithm is applied to a large corpus or dataset of molecules. For instance, Jaeger et al. utilized 19 million compounds to train their initial model. The Morgan algorithm generates corresponding identifiers, which are then ordered based on the structure determined by the canonical SMILES representation. Through this process, a lookup table is created, comprising molecular substructures alongside their corresponding vectors [33].

To obtain the Mol2Vec embedding for a new molecule, the Morgan algorithm is applied to that molecule, and subsequently, the identifiers are extracted. These identifiers are then queried within the pre-trained model. Finally, the vectors associated with all substructures are aggregated in order to represent the molecule. It should be emphasized that the resulting vector presents a fixed size [33].

For our study, SEES:lab provided us with a Mol2Vec model trained on a dataset consisting of approximately 26,000 molecules. It is important to highlight that the Skip-Gram algorithm was used to train the model. Skip-gram is commonly employed to predict the context given a word. Additionally, a window size of 15 was used, meaning that the model considers 15 words before and after the target word. The generated embeddings had a fixed length of 300.

4.5 Distribution of generated molecules

The Mol2Vec model allows us to position the molecules in a 300-dimensional space. The distribution of the embeddings in this space provides us with information about the similarity or dissimilarity of the molecules based on their chemical structure. We decided to calculate the Euclidean distance between the Mol2Vec embeddings, which indicates how close or far apart the vectors are in the Mol2Vec space. Therefore, we were able to extract information about their distribution.

We decided to calculate the Euclidean distance between the Mol2Vec embeddings of the molecules resulting from the filtering process with the molecular formula $C_8H_9NO_2$. Specifically, we chose to use this formula because 165 out of the 585 generated molecules were

found in PubChem. This provided us with the opportunity to calculate the Euclidean distance while distinguishing three groups of pairs: between real molecules (real-real group), between synthetic molecules (synthetic-synthetic group), and between real and synthetic molecules (real-synthetic group).

We introduced the molecules from the $C_8H_9NO_2$ (Fig. 11) dataset into the Mol2Vec model and calculated the Mol2Vec embeddings for each molecule. The results from the Mol2Vec model included four columns: mol, sentence, mol2vec, and FinalVec, which we added to our dataset.

The mol column contains the molecule in the RDKit's mol format. Sentence represents the molecular sentence for each molecule. The mol2vec column indicates the size of the vector, which, as mentioned before, is always 300 dimensions. Finally, the FinalVec column corresponds to the Mol2Vec embeddings (Fig. 17).

mol	sentence	mol2vec	FinalVec
<rdkit.Chem.rdchem.Mol object at 0x7fca29d85580>	(2246728737, 422715066, 3217380708, 3227968710...	(300,) dimensional vector	[0.52712375, -1.6234407, -0.509044, -2.307523,...
<rdkit.Chem.rdchem.Mol object at 0x7fca29d85620>	(2246728737, 3545365497, 2246699815, 387109951...	(300,) dimensional vector	[0.78995013, -1.0518501, -2.4209511, -1.502192...
<rdkit.Chem.rdchem.Mol object at 0x7fca29d856c0>	(847957139, 2592785365, 2245384272, 787341104,...	(300,) dimensional vector	[-0.1369437, -1.5959965, -2.202025, -2.5434742...
<rdkit.Chem.rdchem.Mol object at 0x7fca29d85760>	(2246997334, 3696402029, 2246703798, 977565923...	(300,) dimensional vector	[0.08449754, -0.8936355, -3.3543036, -1.440043...

Figure 17. mol, sentence, mol2vec and FinalVec columns added to the dataset of molecules resulting from the filtering process with the molecular formula $C_8H_9NO_2$

Next, we checked if the dataset contained duplicate molecules. We observed that there were some molecules that were repeated. Therefore, we removed the duplicates, leaving only unique molecules in the dataset. At the end of this process, our dataset consisted of 548 unique molecules, out of which 151 were present in PubChem. We designed a function to calculate the Euclidean distance between all pairs of Mol2Vec vectors in the dataset. We applied the function and generated a symmetric dataframe with 548 rows and 548 columns. Naturally, all elements on the main diagonal were 0 since they corresponded to the distance of the Mol2Vec embedding of each molecule with itself. Additionally, we added the columns for SMILES and database, which, as mentioned before, indicate whether the molecule is present in PubChem or not. This information would facilitate generating the dataframe for each group of pairs. The dataframe can be found in Appendix B (Fig. 34). From this dataframe, we generated a separate dataframe for each group: real-real (151x151), synthetic-synthetic (397x397), and real-synthetic (151x397).

From each of the corresponding dataframes for the groups, we extracted a vector containing all the distances stored within them. In the case of real-real and synthetic-synthetic groups, we took into consideration their symmetry to avoid having duplicate values.

We used these three vectors to generate a histogram and determine the distribution of the Euclidean distance for each group. The three histograms are depicted in the same figure, normalized, and using 100 bins (Fig. 18).

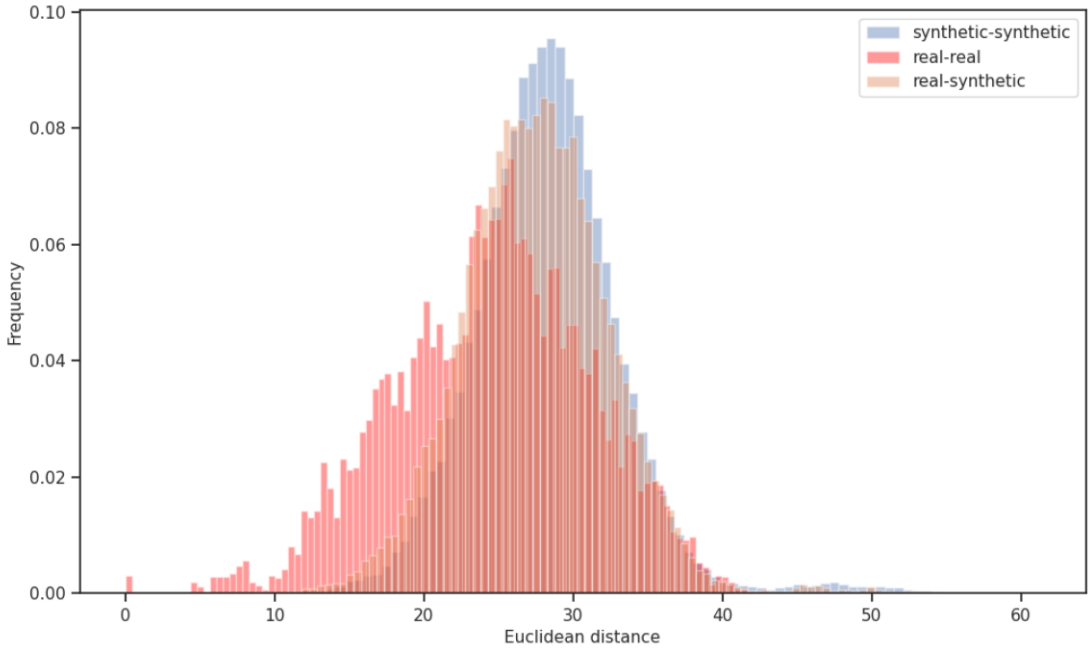


Figure 18. Distribution of Euclidean distance for real-real, synthetic-synthetic and real-synthetic groups

It can be observed that there is significant overlap among the three histograms. This overlapping suggests that the molecules generated by the generative model have a similar representation to real molecules. Therefore, this indicates that the generated molecules are feasible and realistic.

To obtain more precise information for each group, we plotted a box plot for each group. In Fig. 19, we observe that the average Euclidean distance is very similar in all three cases. The range of normal values for the real-real group is greater than that for the synthetic-synthetic and real-synthetic groups. This is consistent with the histogram, where we can see that the Euclidean distance values for these two groups are concentrated around 25 and 33, which is also reflected in the interquartile range of both groups' box plots. The normal value range of the real-real group includes the normal value range of the synthetic-synthetic and real-synthetic groups, as well as many of their outliers. These results suggest that the new molecules have a similar representation to the real ones.

In the synthetic-synthetic and real-synthetic groups, we can also observe outliers that fall outside the normal value range of the real-real group. This fact may indicate variability in the results, which is beneficial for exploring chemical space, discovering metabolites, etc. However, if the value is very high, it may also indicate an unrealistic molecule.

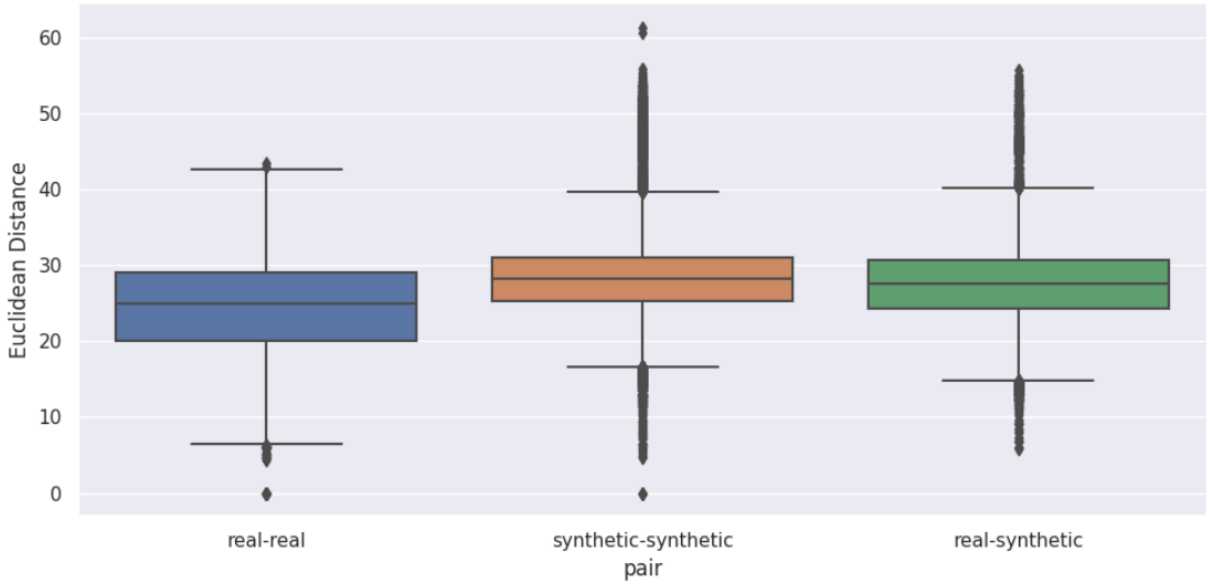


Figure 19. Boxplot of Euclidean Distance for the real-real, synthetic-synthetic, and real-synthetic groups

4.6 Chemical structure similarity

The Tanimoto Similarity is a widely used tool in cheminformatics for measuring similarity between two vectors. In the context of cheminformatics, these vectors typically represent fingerprints. A fingerprint is composed of a predefined set of substructures and features that can be found in a chemical structure. Each substructure or feature is encoded as either 1 (presence) or 0 (absence), resulting in a binary vector. One popular fingerprint generation algorithm is the Morgan algorithm. The Tanimoto Similarity is employed to compare the similarity between chemical structures using their fingerprints [34]. It produces a value ranging from 0 to 1, where a value of 1 indicates identical fingerprints.

We employed Tanimoto Similarity on the generated and filtered molecules with the molecular formula $C_8H_9NO_2$ (Fig. 11). We calculated the Tanimoto Similarity (or Tanimoto Index) between the molecules within each group: real-real, synthetic-synthetic, and real-synthetic. Morgan Fingerprints were used for the calculations. We took into consideration the results of the Euclidean distance from subsection 4.5.

The process was similar to the calculation of the Euclidean distance. Firstly, we designed a function that allowed us to calculate the Tanimoto Index between all possible pairs of molecules in the $C_8H_9NO_2$ dataset. The results were stored in a dataframe with 548 rows and 548 columns. The dataframe was symmetric, with a value of 1 in the elements on the main diagonal. To facilitate further analysis and the generation of dataframes for each group, we added a column that contains the SMILES and another column indicating the presence or absence of the molecule in PubChem (database column). The dataframe can be seen in Appendix B Fig. 35.

Next, we used the dataframe to generate a separate dataframe for each group: real-real (151x151), synthetic-synthetic (397x397), and real-synthetic (151x397).

Subsequently, we generated a vector for each of the three dataframes that contained all the Tanimoto Similarity values for the group. In the case of the real-real and synthetic-synthetic group dataframes, we took into account that they were symmetric to avoid duplicating values.

Finally, we constructed a dataset (Fig. 20) that included the Tanimoto Similarity and Euclidean Distance between all possible pairs of molecules. Moreover, we labeled each pair with its respective group

	Euclidean distance	Tanimoto distance	pair
0	22.922793	0.149	real-real
1	35.137311	0.048	real-real
2	32.667285	0.080	real-real
3	35.606643	0.068	real-real
4	33.579583	0.062	real-real
...	...	...	...
149873	20.496880	0.173	real-synthetic
149874	19.800677	0.192	real-synthetic
149875	34.959327	0.036	real-synthetic
149876	21.722369	0.115	real-synthetic
149877	29.260332	0.047	real-synthetic

Figure 20. The dataset consisted of all possible pairs of molecules from the $C_8H_9NO_2$ dataset, including their Tanimoto Similarity value and the Euclidean distance between their Mol2Vec embeddings.

We represented each pair considering the group they belonged to in a scatter plot (Fig. 21). On the x-axis, we represented the Euclidean distance, and on the y-axis, we represented the Tanimoto Similarity. As shown in Fig. 21, most of the points are located at an Euclidean distance between 20 and 40, which is consistent with the histogram we saw in the previous subsection (Fig. 18). Furthermore, the scatter plots are overlapped, suggesting that the Euclidean Distance and Tanimoto Similarity values for the molecules are comparable across the different groups. This indicates that the modified model is capable of generating new realistic and valid molecules for a given molecular formula.

We observe that as the Euclidean distance between the Mol2Vec embeddings decreases, the value of Tanimoto similarity increases. This indicates that molecules with closely located Mol2Vec embeddings in the Mol2Vec space exhibit higher structural similarity and may possess similar properties.

We wanted to visually verify this relationship. We were provided with a database containing approximately 1.9 million compounds. We selected 172,740 molecules, all of which had a molecular weight below 300 g/mol. Taking into account that the molecular formula used in the generative model, $C_8H_9NO_2$, has a molecular weight of 151.163 g/mol, we generated Mol2Vec embeddings for the 172,740 molecules. We then searched for the four Mol2Vec embeddings with the lowest Euclidean distance for each of the Mol2Vec embeddings corresponding to the generated and filtered molecules with the molecular formula $C_8H_9NO_2$. Fig. 22 shows the result. The first column includes the representation of molecules generated by the generative model, and the following four columns include the four most similar

molecules based on the Euclidean distance between the Mol2Vec embeddings. As shown in Fig. 22, the selected molecules from the database exhibit a very similar structure, including the same functional groups, atom types, etc. This indicates, as we have seen previously, that molecules with closely located Mol2Vec embeddings have a higher degree of structural similarity (Appendix B Fig. 36).

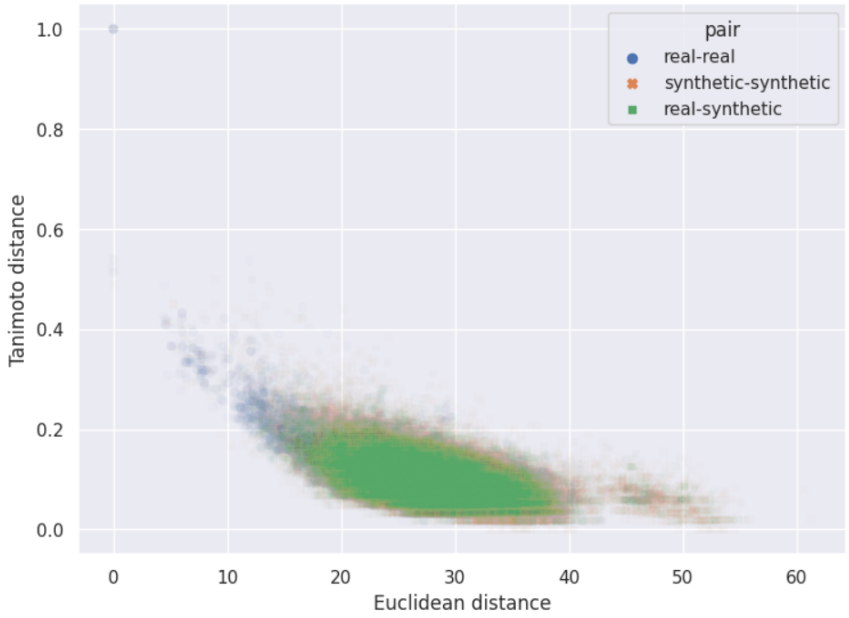


Figure 21. Dispersion of Tanimoto Similarity for all pairs of molecules from the real-real, synthetic-synthetic and real-synthetic group based on the Euclidean distance of their Mol2Vec vectors

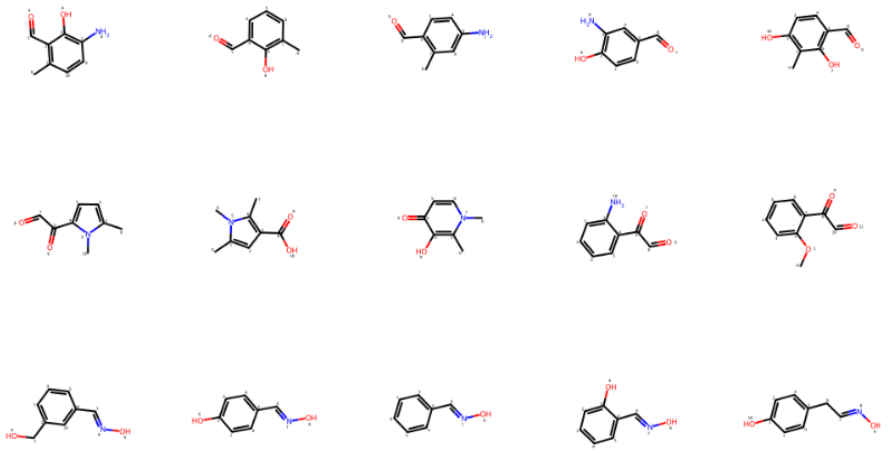


Figure 22. The first column displays the molecules generated by the DGM for $C_8H_9NO_2$, and the subsequent four columns contain the molecules from the database with the highest similarity considering the lowest Euclidean distance between their Mol2Vec embeddings

5 Identification of Spectra from the ARUS Database using the Modified Generative Model

As we explained earlier, one of the main challenges in metabolomics is to identify the spectrum with its corresponding molecule. As mentioned in Section 3.1, there is a database maintained by NIST that contains spectra frequently observed in mass spectrometry of urine and plasma samples, for which the molecular formula is known but not the structure of the molecule. We decided to investigate whether the molecular formulas we had used so far were also present in the database. We were able to find 12 spectra corresponding to $C_8H_9NO_2$ (9 belonging to plasma samples and 3 to urine samples), 1 spectrum for the molecular formula $C_{17}H_{19}NO_3$, and no spectrum for $C_9H_{11}BrN_2O_5$.

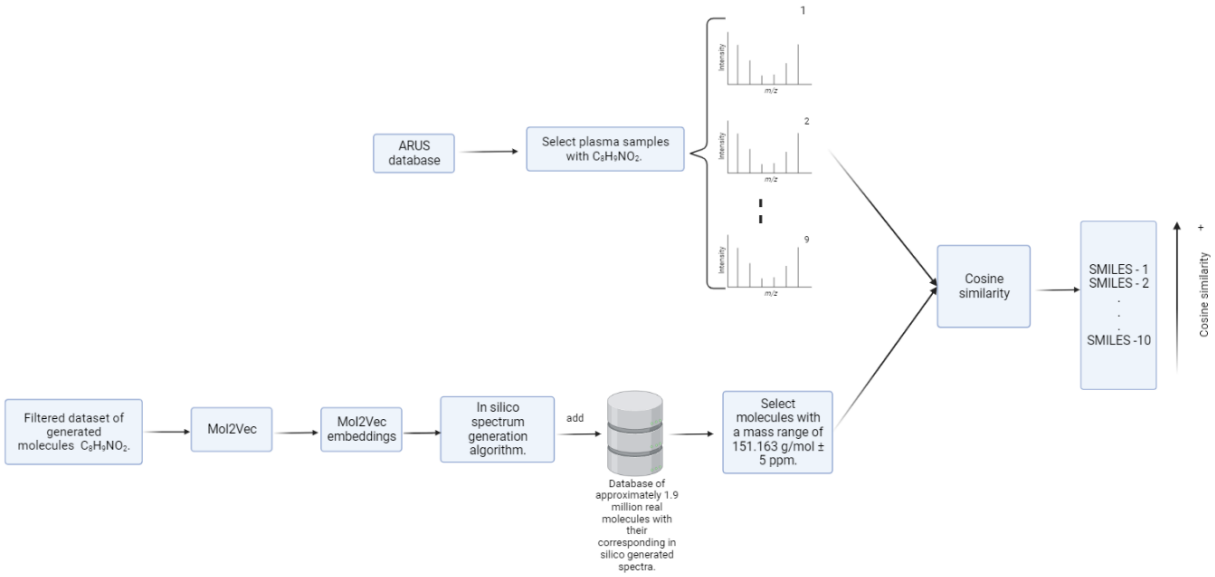


Figure 23. Workflow to identify spectra from the ARUS database with the molecular formula $C_8H_9NO_2$.

SEES:lab provided us with a tool capable of generating in silico spectra from Mol2Vec embeddings. Therefore, we were able to generate spectra for the molecules generated by the DGM for each molecular formula and compare them with the unidentified spectra from ARUS. We focused on the spectra of $C_8H_9NO_2$ for plasma samples, as there is a greater number of spectra in ARUS.

SEES:lab provided us with a database of approximately 1.9 million real molecules with their corresponding in silico generated spectra. We applied the algorithm to the Mol2Vec embeddings of the generated molecules filtered with the formula $C_8H_9NO_2$, generated the spectra, and added them to the database. On the other hand, we selected the unidentified spectra of $C_8H_9NO_2$ for plasma samples from the ARUS database.

We filtered the molecules in the database considering the molecular weight of $C_8H_9NO_2$, which is 151.163 g/mol $\pm$5 ppm. The spectra are represented as vectors, which facilitates comparison. Therefore, we were able to calculate the similarity between the vectors from ARUS and the vectors in the specified range (151.163 g/mol $\pm$5 ppm) that includes real molecules

from the database and molecules generated by the generative model. To do this, we applied cosine similarity. Cosine similarity is a measure used to determine the similarity between two vectors in a multi-dimensional space. It calculates the cosine of the angle between the two vectors, resulting in a value ranging from -1 to 1. For each ARUS spectrum, we selected the top 10 candidates from the database with the highest cosine similarity values (in descending order) (Fig. 23).

The results are stored in a dataframe where each row corresponds to an unidentified ARUS spectrum of $C_8H_9NO_2$. The dataframe has 7 columns: file name, MS/MS, Formula, mz, smiles_10, cosine_10, and generated_m. The file name indicates the type of sample, which in this case is all plasma, and the technique used (fragmentation by collision cell (HCD) and ion trap (IT)). The MS/MS and mz columns contain the spectrum of the unknown molecule from ARUS, with MS/MS also including the intensity of each peak. The Formula column contains the molecular formula. Each cell in the smiles_10 column contains 10 SMILES strings from the database that have the most similar spectrum to the unknown molecule. Each cell in the cosine_10 column contains the 10 cosine similarity values corresponding to the 10 molecules. Finally, the generated_m column indicates whether the selected molecules are from the $C_8H_9NO_2$ dataset of the generative model or if they are real molecules previously present in the database. The dataframe can be seen in Fig. 24.

	File name	MS/MS	Formula	mz	smiles_10	cosine_10	generated_m
0	plasma_HCD	53.0386:1.63 54.0338:4.54 55.0178:8.32 55.0542...	C8H9NO2	[53.0386, 54.0338, 55.0178, 55.0542, 56.0496, ...	[C=C(O)C1=C(C2CC2)NC1=O, CC1CCC(C#N)=C1C(=O)O,...	[0.3622089663122213, 0.31980107453341566, 0.31...	[yes, yes, yes, yes, yes, no, yes, no, yes, yes]
1	plasma_HCD	56.9649:4.43 58.0651:1.29 59.073:3.43 65.0386:...	C8H9NO2	[56.9649, 58.0651, 59.073, 65.0386, 67.0542, 7...	[CC1=C(O)C=CC(C(N)=O)=C1, CC(=O)C1=C(C=O)C=C1C...	[0.27216552697590873, 0.27216552697590873, 0.2...	[yes, yes, yes, yes, no, yes, yes, yes, yes, yes]
2	plasma_HCD	50.7501:1.16 56.9648:9.9 57.0334:1.2 58.0651:3...	C8H9NO2	[50.7501, 56.9648, 57.0334, 58.0651, 58.5603, ...	[C=C1CCC(=O)C(=O)C=C1N, CC(=O)C1=C(C=O)C=C1CN,...	[0.2773500981126146, 0.2749859704614352, 0.251...	[yes, yes, yes, no, yes, yes, yes, yes, yes, no]
3	plasma_HCD	110.0601:88.14 111.0441:1.36 124.0392:2.24 134...	C8H9NO2	[110.0601, 111.0441, 124.0392, 134.06, 146.996...	[CC(=O)N(C(C)=O)C1=CC=C1, CC(=O)N(C(C)=O)C1=CC...	[0.17677669529663687, 0.17677669529663687, 0.1...	[yes, yes, yes, yes, yes, yes, no, yes, yes, yes, yes]
4	plasma_HCD	109.0523:3.11 110.0601:60.63 128.019:1.15 134....	C8H9NO2	[109.0523, 110.0601, 128.019, 134.0601, 151.06...	[CC(=O)N(C(C)=O)C1=CC=C1, CC(=O)N(C(C)=O)C1=CC...	[0.1889822365046136, 0.1889822365046136, 0.188...	[yes, yes, yes, yes, yes, yes, no, yes, yes, yes, yes]
5	plasma_HCD	55.0178:2.73 55.0541:5.05 56.0495:7.71 58.065:...	C8H9NO2	[55.0178, 55.0541, 56.0495, 58.065, 65.0386, 6...	[CC1=C(C(=O)OC#N)CCC1, O=C1NC2=C(C=C1O)CCC2, O...	[0.2760671660461932, 0.27006573887713, 0.27006...	[yes, yes, no, yes, yes, yes, yes, yes, yes, yes]
6	plasma_HCD	52.6794:1.98 55.0179:3.38 55.0544:15.56 56.049...	C8H9NO2	[52.6794, 55.0179, 55.0544, 56.0495, 57.2473, ...	[CC1=C(O)C=CC(C(N)=O)=C1, CC1=CC(C(N)=O)=CC=C1...	[0.30685820596610786, 0.30685820596610786, 0.3...	[yes, yes, yes, yes, no, yes, yes, yes, yes, yes]
7	plasma_HCD	53.8091:4.04 56.691:3.31 56.9649:323.51 58.065...	C8H9NO2	[53.8091, 56.691, 56.9649, 58.0651, 59.073, 60...	[NCCC1=CC=CC(=O)C1=O, CC(=O)Nc1ccc(O)cc1, CC(=...	[0.1889822365046136, 0.17407765595569785, 0.17...	[yes, no, no, no, no, no, yes, yes, yes, yes]
8	plasma_IT	58.0651:14.28 58.3429:1.05 59.0729:8.21 62.993...	C8H9NO2	[58.0651, 58.3429, 59.0729, 62.9938, 71.7131, ...	[CCC(=O)C(=O)C1=C(N)C=C1, CC1C(C=O)=CNC=C1C=O,...	[0.23008949665421113, 0.21081851067789195, 0.2...	[yes, no, yes, yes, yes, yes, yes, yes, yes, yes]

Figure 24. Dataframe Containing Unidentified Spectra for $C_8H_9NO_2$ from Plasma Samples in ARUS Database. For each unknown spectrum, 10 molecules are proposed based on cosine similarity between the unidentified spectrum and synthetic spectra of molecules in the database (151.163 g/mol $\pm$5 ppm)

As seen in the generated_m column (Fig. 24), the majority of the candidate molecules belong to the group of molecules generated by the DGM. This indicates that the modified generative model can be a very useful tool in spectrum identification by proposing new candidates.

Next, we provide the three candidate molecules (Fig. 25,26,27) for the first unidentified spectrum in the dataframe. All of them are generated by the generative model and none of them are found in PubChem. They have cosine similarity values of 0.362, 0.319, and 0.314, respectively. We can observe that the spectra are quite different but share a significant number of peaks.

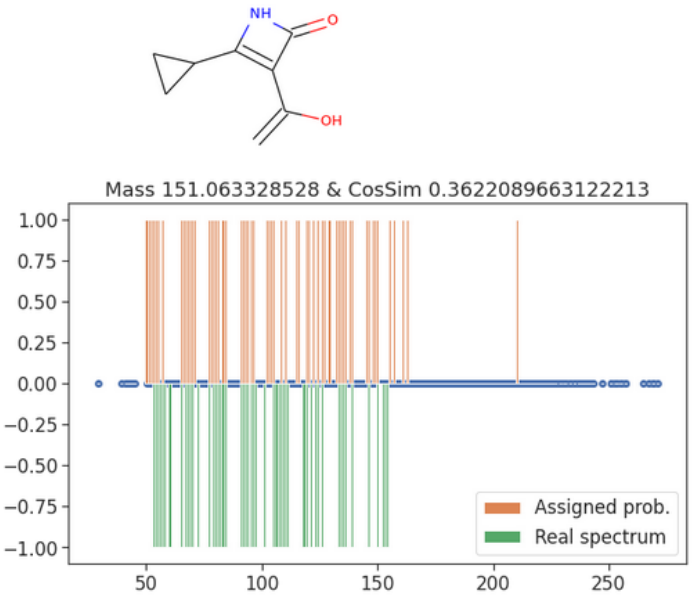


Figure 25. Comparison between the first unidentified spectrum of ARUS $C_8H_9NO_2$ (plasma) and the spectrum of the first candidate. The candidate is generated by the generative model. They present a cosine similarity of 0.36

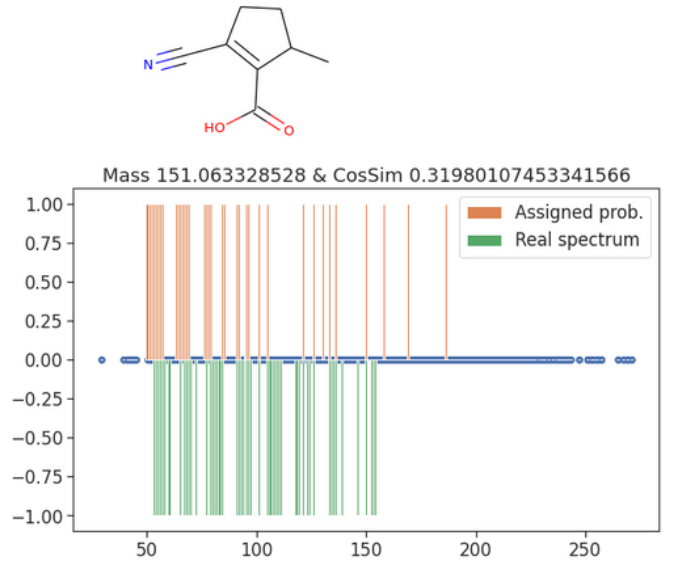


Figure 26. Comparison between the first unidentified spectrum of ARUS $C_8H_9NO_2$ (plasma) and the spectrum of the second candidate. The candidate is generated by the generative model. They present a cosine similarity of 0.319

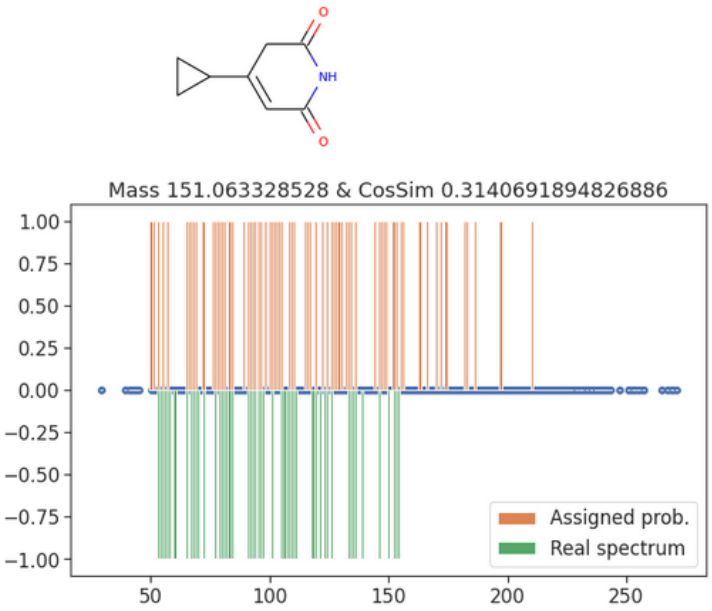


Figure 27. Comparison between the first unidentified spectrum of ARUS $C_8H_9NO_2$ (plasma) and the spectrum of the third candidate. The candidate is generated by the generative model. They present a cosine similarity of 0.313

6 Prediction of aqueous solubility (logS) of generated molecules

Throughout history, pharmaceutical research has shown great interest in identifying good drug candidates. Various methods have been developed to assess druglikeness, which refers to the set of properties that determine whether a compound has the potential to become a drug. One of the most popular criteria is Lipinski's rule, also known as the rule of five. This criterion establishes a series of minimum conditions for a molecule to be a potential candidate. It states that the number of hydrogen bond acceptors should be less than 5, the number of hydrogen bond donors should be less than 5, the molecular weight should not exceed 500, and the octanol-water partition coefficient (logP) should not exceed 5 [30]

Different druglikeness filters have been developed, such as Lipinski, Ghose, Veber, and others [35]. In recent years, the popularity of a computational method known as Quantitative Estimate of Druglikeness (QED) has grown. QED allows for the estimation of druglikeness and generates a value ranging from 0 to 1, with 1 being the best possible result. The method considers eight descriptor properties to calculate the score: molecular weight (MW), octanol-water partition coefficient (logP), number of hydrogen bond donors (HBD), number of hydrogen bond acceptors (HBA), polar surface area (PSA), number of rotatable bonds (ROTB), number of aromatic rings (AROM), and number of structural alerts (ALERTS) [30].

In recent years, there has also been a growing interest in improving the aqueous solubility of drugs [36] [37], as it is a crucial parameter that affects the administration and absorption of drugs, especially when administered orally. Aqueous solubility is commonly expressed using a base 10 logarithm (logS) [38].

Historically, the logS value of molecules has been determined experimentally, which is a costly and time-consuming process. However, various computational models have been proposed to predict logS based on fingerprints such as Dragon descriptors and Morgan fingerprints, as well as directly on the molecular graph. These computational approaches offer a faster and more cost-effective alternative for estimating the aqueous solubility of molecules, aiding in drug discovery and development processes [39]. We decided to implement a Random Forest Regressor capable of predicting logS values using Mol2Vec embeddings and QED properties.

A random forest regressor is a machine learning algorithm used for regression tasks, where the goal is to predict a numerical value. It is a ensemble learning method. The random forest regressor combines the predictions of multiple decision trees to generate a more accurate prediction. It works by constructing a multitude of decision trees during the training phase. Each decision tree is built on a randomly selected subset of the training data and a subset of the input features. During prediction, the random forest regressor averages the predictions from all the individual decision trees to provide the final output [40].

Next, we analyze the development process and the results of the predictive model.

6.1 Training data

To train our model, we utilized the AqSolDB compound database. This database consists of 9,982 molecules, each with its corresponding logS value. It also provides additional information for each molecule, such as molecular weight, number of valence electrons, number of rotatable bonds, number of rings, logP, and more. AqSolDB, developed by Sorkun et al. in 2019, has become a reference database for the development of machine learning and deep learning algorithms for logS prediction [41].

We noticed that AqSolDB contained some inorganic molecules, which were not relevant for our purposes. Therefore, we developed a function to filter out these inorganic molecules. As a result, we removed 25 molecules. Later, we removed 6 SMILES that were causing errors. After these changes, the database contained 9,651 molecules. We stored the molecules in SMILES format along with their corresponding logS values (solubility_dataset), while discarding the remaining columns from the original database (Fig. 28).

	smiles	solubility	is_Organic
0	CCCCCCCCCCCCCCCCCC[N+](C)(C)C.[Br-]	-3.616127	True
1	C1=CC2=C3C(=C1)C(=O)NC3=CC=C2	-3.254767	True
2	C1=CC(=CC=C1C=O)Cl	-2.177078	True
3	CC(C1=CC=CC=C1)C2=CC(=C(C(=C2)C(=O)O)[O-])C(C)...	-3.924409	True
4	C1C(O1)CN(CC2CO2)C3=CC=C(C=C3)CC4=CC=C(C=C4)N(...	-4.662065	True
...	...	...	...
9977	CCCCNC1=CC=C(C=C1)C(=O)OCCN(C)C	-3.010000	True
9978	CC1(C2CC3C(C(=O)C(=C(C3(C(=O)C2=C(C4=C1C=CC=C4...	-2.930000	True
9979	CC1=CC(=C(C=C1)C(C)C)O	-2.190000	True
9980	CC(C)C(CCCN(C)CCC1=CC(=C(C=C1)OC)OC)(C#N)C2=CC...	-3.980000	True
9981	CC(=O)CC(C1=CC=CC=C1)C2=C(C3=CC=CC=C3OC2=O)O	-4.780000	True

Figure 28. Solubility_dataset. The SMILES from AqSolDB, the logS of each molecule, and a column indicating whether the molecule is organic can be observed. This column allowed us to remove the inorganic substances

6.2 Quantitative Estimation of Drug-likeness descriptors

RDKit allows generating the QED score and computationally extracting each of the 8 property descriptors for each molecule. This RDKit function is very useful as it can be applied to both known real molecules and de novo generated molecules. We applied this function to the molecules from AqSolDB in the solubility_dataset and added molecular weight, HBD, HBA, PSA, ROTB, AROM, and and logP calculated using the ALOGP method to each molecule in the dataset. We did not take into consideration the structural alerts, which are substructures associated with certain toxicological risks, for the prediction of logS.

We also incorporated the Mol2Vec embeddings corresponding to each molecule into the solubility_dataset. The resulting dataset can be seen in Fig. 29.

	smiles	solubility	is_Organic	Generate-MW	Generate-HBA	Generate-HBD	Generate-PSA	Generate-ROTB	Genereta-AROM	Generate-ALOGP	mol2vec	FinalVec
0	CCCCCCCCCCCCCCCCCC[N+](C)(C)C.[Br-]	-3.616127	True	392.510	0	0	0.00	17	0	3.95810	(300,) dimensional vector	[-3.1532447, -10.467522, -4.5587573, -5.430258...
1	C1=CC2=C3C(=C1)C(=O)NC3=CC=C2	-3.254767	True	169.183	1	1	29.10	0	2	2.40550	(300,) dimensional vector	[-0.21332638, -4.94754, -4.898317, -2.9384081,...
2	C1=CC(=CC=C1C=O)Cl	-2.177078	True	140.569	1	0	17.07	1	1	2.15250	(300,) dimensional vector	[0.3167037, -2.2978127, -2.3034618, -1.9181324...
3	CC(C1=CC=CC=C1)C2=CC(=C(C(=C2)C(=O)O)[O-])C(C)...	-3.924409	True	756.226	6	2	120.72	10	6	9.52150	(300,) dimensional vector	[5.0138774, -14.494184, -12.087434, 1.5173029,...
4	C1C(O1)CN(CC2CO2)C3=CC=C(C=C3)CC4=CC=C(C=C4)N(...	-4.662065	True	422.525	6	0	56.60	12	2	2.48540	(300,) dimensional vector	[0.3777204, -7.9683495, -6.0686083, -2.1634223...

Figure 29. Solubility_dataset consists of the SMILES, logS, QED descriptor properties, and Mol2Vec embeddings

As we explained earlier, each Mol2Vec embedding consists of 300 positions. To facilitate predictions, we decomposed each vector and arranged it into 300 columns. Starting from the solubility_dataset, we generated four dataframes. The first one contains all the descriptor properties, the second one contains all the descriptor properties except for logP, the third one contains only the Mol2Vec embeddings, and finally, the last dataframe contains all the variables. LogP (logarithm of the partitioning coefficient between n-octanol and water) has been accepted as a descriptor of the lipophilicity of a molecule. There is a certain correlation between logP and logS [38].

6.3 Random Forest Regressor for LogS Prediction

We generated a Random Forest Regressor model for each of the four mentioned dataframes in subsection 6.2. In all cases, we separated the target variable, solubility (label), from the rest of the numerical variables (features). Non-numerical variables (SMILES) were removed. We performed data splitting for each dataframe, selecting 80% of the data for training and 20% of the data for testing (train_features, train_labels, test_features, test_labels). We trained each Random Forest Regressor using train_features and train_labels with 50 trees.

Afterwards, we input the test_features into each model to make predictions of aqueous solubility. To verify if the predictions aligned with the actual values, we generated four scatter plots comparing the real values against the predicted values (Fig. 30). To facilitate interpretation, we included a line of equality. As can be observed in Fig. 30, the points in all four graphs cluster around the line of equality, indicating that the predictions are quite accurate and there is correlation.

However, we can observe that in some graphs, the values are more dispersed. Specifically, when using only the Mol2Vec and QED vectors (excluding logP). We evaluated the performance of the model by calculating the root mean square error (RMSE) (Fig. 31). As shown in Fig. 31, the lowest RMSE (1.16) corresponds to the predictions made with all values (Mol2Vec+QED), followed by predictions using all QED descriptor properties (1.30). The RMSE increases when using only the Mol2Vec embeddings (1.44), and considering the QED properties descriptors excluding logP, the RMSE further increases (1.71).

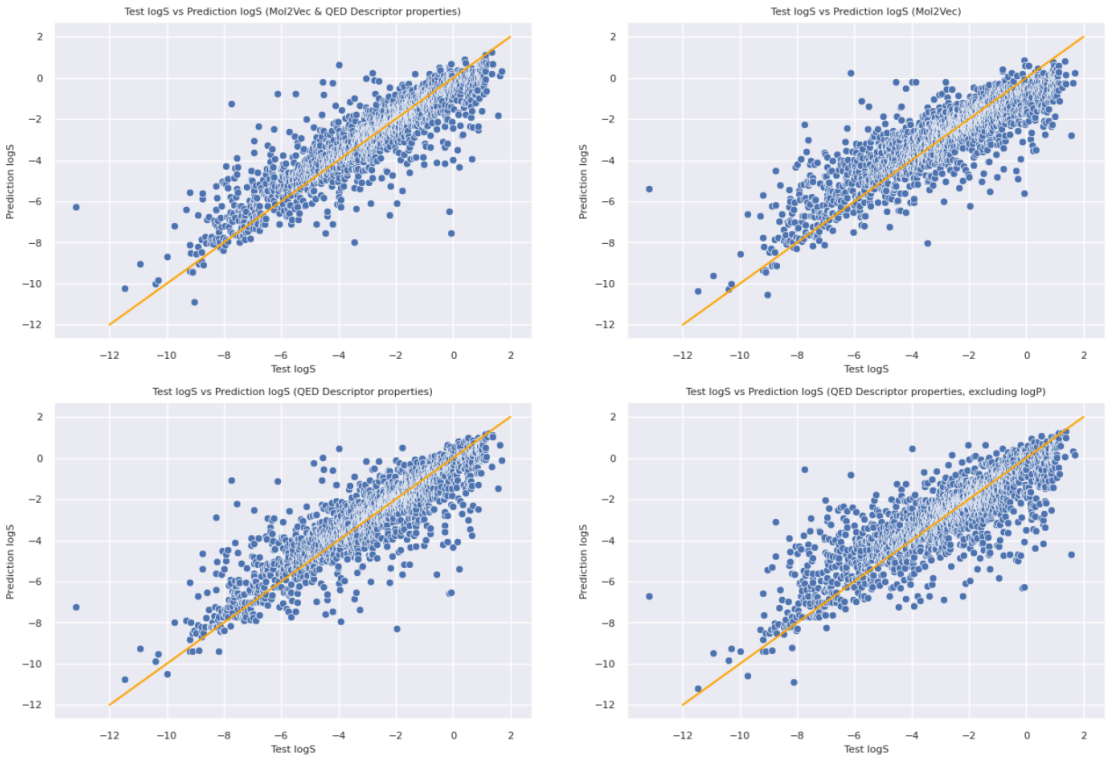


Figure 30. Real logS values of training against predicted logS values for each group of features (all, Mol2Vec, QED Descriptor properties, and QED Descriptor properties excluding logP)

	Features used	rmse
0	Mol2Vec+QED property descriptors	1.16
1	Mol2Vec	1.44
2	QED property descriptors	1.30
3	QED property descriptors (no logP)	1.71

Figure 31. Root mean square error for each Random Forest Regressor model

6.4 Prediction of logS for generated molecules with the molecular formula C8H9NO2

Considering the previous results, we decided to use the Random Forest Regressor model trained with the Mol2Vec embeddings along with the QED property descriptors to make predictions of logS for the generated and filtered molecules with the molecular formula $C_8H_9NO_2$. To achieve this, we started with the dataframe of generated and filtered molecules for $C_8H_9NO_2$ along with their corresponding Mol2Vec embeddings. Using the QED function, we calculated the QED descriptor properties for each molecule. We removed all reward columns and any columns that did not contain numerical values, and then inputted the dataframe into the model to make the logS predictions. The results were stored along with

the corresponding SMILES and rewards in a dataset.

Overall, compounds are classified based on their logS values as highly soluble if the value is above 0, soluble within the range of -2 to 0, slightly soluble between -2 and -4, and insoluble for values below -4. Taking this classification into account, we filtered the dataset and selected molecules with a logS value higher than -4 and a QED score equal to or greater than 0.7. The filtering process led us to identify two molecules that met these criteria. These two molecules are not present in PubChem (Fig. 32). The first molecule has a QED score of 0.71, while the second molecule has a QED score of 0.7. They both have a synthetic accessibility reward score of 0.74 and 0.77, indicating a certain ease of synthesis. Both molecules pass the zinc molecule filter established by You et al., indicating that they do not have problematic functional groups. Additionally, they have logS values of -0.72 and -2, respectively, indicating solubility. Therefore, we have discovered two new potential drug candidates (Fig. 33).

The model has demonstrated its power as a tool for exploring the chemical space of drugs and proposing valid and realistic drug candidates.

	smile	r_valid	reward_qed	reward_sa	final_stat	rew_env	rew_d_step	rew_d_final	cur_ep_ret	steric_strain	zincfilter	stop	formula	database	FinalVec	logS prediction
	CCCOC1=NC2=C1C(O)=C2	2.0	0.715691	0.740194	-0.355275	1.632530	0.002237	0.434871	1.076084	True	True	False	C8H9NO2	False	[0.516166, -3.5558205, -2.1490946, -0.88711905...	-0.728000
	C=CC1=C(NC(C)=O)OC=C1	2.0	0.700728	0.770818	-0.081301	1.906503	0.008529	1.418952	2.505775	True	True	True	C8H9NO2	False	[0.3274684, -0.14788409, -2.0584974, 0.0408742...	-2.007483

Figure 32. Resulting molecules from the filtering process (reward_qed>=0.7 and logS>=-4)

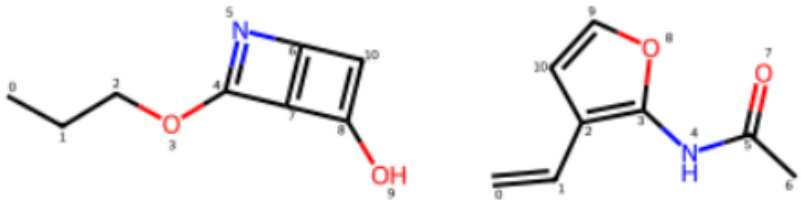


Figure 33. New potential drug candidates generated by DGM with $C_8H_9NO_2$ molecular formula

7 Conclusion

The primary goal of the study was to modify the generative model introduced by You et al. in order to generate de novo molecules with a specific molecular formula. Our aim was to provide a tool for systematic, efficient, and thorough exploration of the chemical space.

We have been able to study in detail the generative model (GCPN) proposed by You et al. We have developed the necessary adaptations to predominantly generate molecules with a specified molecular formula as a parameter. Furthermore, it has been of great importance to demonstrate that the molecules were realistic and feasible.

To achieve this, we employed the Mol2Vec model, which allowed us to obtain a 300-dimensional embedding for each of the generated and filtered molecules for $C_8H_9NO_2$ (Fig. 11). We have observed that the distribution of Euclidean distances between the Mol2Vec embeddings ($C_8H_9NO_2$ dataset) for the real-real, synthetic-synthetic, and real-synthetic pair groups is overlapping, indicating that the generated molecules have a similar representation to real molecules. We have determined that the Tanimoto similarity values for the molecules are comparable across the three pair groups. We have also observed that the structural similarity between two molecules is higher when their corresponding Mol2Vec embeddings are close to each other in the Mol2Vec space. Hence, we have been able to verify that the generated molecules are realistic and feasible.

We generated the in silico spectra of the generated molecules for $C_8H_9NO_2$ (Fig. 11). Subsequently, we added them to an in silico spectra database (provided by SEES:lab). We filtered by molecular mass and compared the spectra from the database with the spectra from the ARUS database for plasma samples with the molecular formula $C_8H_9NO_2$ (a total of 9 spectra). Ten candidates from the database were proposed for each ARUS spectrum. The majority of candidate molecules belonged to the group of molecules generated by the DGM. Therefore, we have been able to determine that our adapted generative model is a promising tool for spectrum identification due to its ability to generate new candidates.We have considered that it would be highly useful to create a database of generated molecules along with their corresponding in silico generated spectra. This database would provide researchers with additional resources for the identification of spectra and enhance their capabilities in this field.

Finally, we implemented a random forest regressor for the prediction of aqueous solubility using the Mol2Vec embeddings and the computationally calculated QED descriptor properties. We used the model to predict the aqueous solubility of the generated and filtered molecules for $C_8H_9NO_2$ (Fig. 11). We filtered the results to find potential drug candidates in the dataset, considering a QED value greater than 0.7 and that the compounds were at least slightly soluble. After applying these filters, we were able to identify two new drug candidates, neither of which is found in PubChem. This fact has shown us that our adapted model is a useful and promising tool for the discovery of new drugs.

In conclusion, we have successfully achieved the established objectives. The adapted model provides us with numerous possibilities for biomedical research.

8 Bibliography

[1] F. Piccialli, V. Di Somma, F. Giampaolo, S. Cuomo, and G. Fortino, "A survey on deep learning in medicine: Why, how and when?" *Information Fusion*, vol. 66, Sep. 2020. DOI: 10.1016/j.inffus.2020.09.006.

[2] Y. Cheng, Y. Gong, Y. Liu, B. Song, and Q. Zou, "Molecular design in drug discovery: a comprehensive review of deep generative models," *Briefings in Bioinformatics*, vol. 22, no. 6, Aug. 2021, bbab344, 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.

[3] Deloitte, *Measuring the return from pharmaceutical innovation*, https://www2.deloitte.com/us/en/pages/life-sciences-and-health-care/articles/measuring-return-from-pharmaceutical-innovation.html, 2022.

[4] L. Ruthotto and E. Haber, "An introduction to deep generative modeling," *GAMM-Mitteilungen*, vol. 44, May 2021. DOI: 10.1002/gamm.202100008.

[5] X. Liu and J. W. Locasale, "Metabolomics: A primer," *Trends in Biochemical Sciences*, vol. 42, Feb. 2017. DOI: 10.1016/j.tibs.2017.01.004.

[6] D. S. Wishart, "Emerging applications of metabolomics in drug discovery and precision medicine," *Nature Reviews Drug Discovery*, vol. 15, no. 7, pp. 473–484, 2016. DOI: 10.1038/nrd.2016.32.

[7] S.-a. Qi, Q. Wu, Z. Chen, *et al.*, "High-resolution metabolomic biomarkers for lung cancer diagnosis and prognosis," *Scientific Reports*, vol. 11, Jun. 2021. DOI: 10.1038/s41598-021-91276-2.

[8] C. Lima, H. Muhamadali, and R. Goodacre, "The role of raman spectroscopy within quantitative metabolomics," *Annual Review of Analytical Chemistry*, vol. 14, Jun. 2021. DOI: 10.1146/annurev-anchem-091420-092323.

[9] Ö. C. Zeki, C. C. Eylem, T. Reçber, S. Kır, and E. Nemutlu, "Integration of gc–ms and lc–ms for untargeted metabolomics profiling," *Journal of Pharmaceutical and Biomedical Analysis*, vol. 190, p. 113 509, 2020, ISSN: 0731-7085. DOI: https://doi.org/10.1016/j.jpba.2020.113509. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0731708520313959.

[10] *Annotated recurrent unidentified spectra (arus)*. [Online]. Available: https://chemdata.nist.gov/dokuwiki/doku.php?id=chemdata:arus.

[11] D. H. Nguyen, C. H. Nguyen, and H. Mamitsuka, "Recent advances and prospects of computational methods for metabolite identification: A review with emphasis on machine learning approaches," *Briefings in Bioinformatics*, vol. 20, no. 6, pp. 2028–2043, Aug. 2018, ISSN: 1477-4054. DOI: 10.1093/bib/bby066. eprint: https://academic.oup.com/bib/article-pdf/20/6/2028/31789414/bby066.pdf. [Online]. Available: https://doi.org/10.1093/bib/bby066.

[12] I. Blaženović, T. Kind, J. Ji, and O. Fiehn, "Software tools and approaches for compound identification of lc-ms/ms data in metabolomics," *Metabolites*, vol. 8, May 2018. DOI: 10.3390/metabo8020031.

[13] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, *et al.*, "Automatic chemical design using a data-driven continuous representation of molecules," *ACS Central Science*, vol. 4, no. 2, pp. 268–276, Jan. 2018. DOI: 10.1021/acscentsci.7b00572. [Online]. Available: https://doi.org/10.1021%2Facscentsci.7b00572.

[14] G. Guimaraes, B. Sanchez, P. Farias, and A. Aspuru-Guzik, "Objective-reinforced generative adversarial networks (organ) for sequence generation models," May 2017.

[15] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, *Graph convolutional policy network for goal-directed molecular graph generation*, 2019. arXiv: 1806.02473 [cs.LG].

[16] S.-H. Han, K. W. Kim, S. Kim, and Y. C. Youn, "Artificial neural network: Understanding the basic concepts without mathematics," *Dementia and Neurocognitive Disorders*, vol. 17, pp. 83–89, 2018.

[17] B. Ramsundar and R. Zadeh, *TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning*. O'Reilly Media, 2018, ISBN: 9781491980453. [Online]. Available: https://books.google.es/books?id=rtlEtAEACAAJ.

[18] L. David, A. Thakkar, R. Mercado, and O. Engkvist, "Molecular representations in AI-driven drug discovery: A review and practical guide," *J. Cheminform.*, vol. 12, no. 1, p. 56, Sep. 2020.

[19] A. D. White, "Deep learning for molecules and materials," *Living Journal of Computational Molecular Science*, vol. 3, no. 1, p. 1499, 2021. DOI: 10.33011/livecoms.3.1.1499. [Online]. Available: https://dmol.pub.

[20] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: A comprehensive review," *Computational Social Networks*, vol. 6, Nov. 2019. DOI: 10.1186/s40649-019-0069-y.

[21] A. S. Daigavane, B. Ravindran, and G. Aggarwal, "Understanding convolutions on graphs," *Distill*, 2021.

[22] T. H. Do, D. M. Nguyen, G. Bekoulis, A. Munteanu, and N. Deligiannis, "Graph convolutional neural networks with node transition probability-based message passing and dropnode regularization," *CoRR*, vol. abs/2008.12578, 2020. arXiv: 2008.12578. [Online]. Available: https://arxiv.org/abs/2008.12578.

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018, ISBN: 0262039249.

[24] [Online]. Available: https://es.mathworks.com/content/dam/mathworks/ebook/gated/predictive-maintenance-ebook-all-chapters.pdf.

[25] M. van Otterlo and M. A. Wiering, "Markov decision processes: Concepts and algorithms," 2012.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. arXiv: 1707.06347 [cs.LG].

[27] H. Taud and J. Mas, "Multilayer perceptron (mlp)," in *Geomatic Approaches for Modeling Land Change Scenarios*, M. T. Camacho Olmedo, M. Paegelow, J.-F. Mas, and F. Escobar, Eds. Cham: Springer International Publishing, 2018, pp. 451–455, ISBN: 978-3-319-60801-3. DOI: 10.1007/978-3-319-60801-3_27. [Online]. Available: https://doi.org/10.1007/978-3-319-60801-3_27.

[28] *Pubchem compound database*. [Online]. Available: https://pubchem.ncbi.nlm.nih.gov/.

[29] S. Jaeger-Honz, S. Fulle, and S. Turk, "Mol2vec: Unsupervised machine learning approach with chemical intuition," *Journal of Chemical Information and Modeling*, vol. 58, Dec. 2017. DOI: 10.1021/acs.jcim.7b00616.

[30] R. Bickerton, G. Paolini, J. Besnard, S. Muresan, and A. Hopkins, "Quantifying the chemical beauty of drugs," *Nature chemistry*, vol. 4, pp. 90–8, Feb. 2012. DOI: 10.1038/nchem.1243.

[31] *Pubchempy documentation*. [Online]. Available: https://pubchempy.readthedocs.io/en/latest/.

[32] V. K. Ayyadevara, "Word2vec," in *Pro Machine Learning Algorithms : A Hands-On Approach to Implementing Algorithms in Python and R*. Berkeley, CA: Apress, 2018, pp. 167–178, ISBN: 978-1-4842-3564-5. DOI: 10.1007/978-1-4842-3564-5_8. [Online]. Available: https://doi.org/10.1007/978-1-4842-3564-5_8.

[33] S. Altaner, S. Jaeger-Honz, R. Fotler, *et al.*, "Machine learning prediction of cyanobacterial toxin (microcystin) toxicodynamics in humans," *ALTEX: Alternativen zu Tierexperimenten*, Jul. 2019. DOI: 10.3390/toxins11070388..

[34] D. Bajusz, A. Rácz, and K. Héberger, "Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations?" *Journal of Cheminformatics*, vol. 7, May 2015. DOI: 10.1186/s13321-015-0069-3.

[35] S. Mignani, J. Rodrigues, H. Tomas, *et al.*, "Present drug-likeness filters in medicinal chemistry during the hit and lead optimization process: How far can they be simplified?" *Drug Discovery Today*, vol. 23, no. 3, pp. 605–615, 2018, ISSN: 1359-6446. DOI: https://doi.org/10.1016/j.drudis.2018.01.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1359644617304671.

[36] K. T. Savjani, A. K. Gajjar, and J. K. Savjani, "Drug solubility: Importance and enhancement techniques," *ISRN Pharmaceutics*, vol. 2012, 2012.

[37] M. Ishikawa and Y. Hashimoto, "Improvement in aqueous solubility in small molecule drug discovery programs by disruption of molecular planarity and symmetry," *Journal of Medicinal Chemistry*, vol. 54, no. 6, pp. 1539–1554, 2011, PMID: 21344906. DOI: 10.1021/jm101356p. eprint: https://doi.org/10.1021/jm101356p. [Online]. Available: https://doi.org/10.1021/jm101356p.

[38] J. Wang, G. Krudy, T. Hou, W. Zhang, G. Holland, and X. Xu, "Development of reliable aqueous solubility models and their application in druglike analysis," *Journal of Chemical Information and Modeling*, vol. 47, no. 4, pp. 1395–1404, 2007, PMID: 17569522. DOI: 10.1021/ci700096r. eprint: https://doi.org/10.1021/ci700096r.

[39] J. Meng, P. Chen, M. Wahib, *et al.*, "Boosting the predictive performance with aqueous solubility dataset curation," *Scientific Data*, vol. 9, p. 71, Mar. 2022. DOI: 10.1038/s41597-022-01154-3.

[40] Y. Liu, Y. Wang, and J. Zhang, "New machine learning algorithm: Random forest," in *International Conference on Information Computing and Applications*, 2012.

[41] M. Sorkun, A. Khetan, and S. Er, "Aqsoldb, a curated reference set of aqueous solubility and 2d descriptors for a diverse set of compounds," *Scientific Data*, vol. 6, Aug. 2019. DOI: 10.1038/s41597-019-0151-1.

Appendices

A Programming code

A.1 Code description

The official implementation of ”Graph Convolutional Policy Network for Goal Directed Molecular Graph Generation” is in a public GitHub repository. Four files stand out:

- run_molecule.py. It is the main code. Hyperparameters (parameters that control the behavior of the model during training) can be tuned.
- molecule.py It contains the molecule environment code
- gcn_policy.py. It is the GCPN.
- pposgd_simple_gcn.py PPO algorithm specifically tuned for GCN policy.

A.2 Run

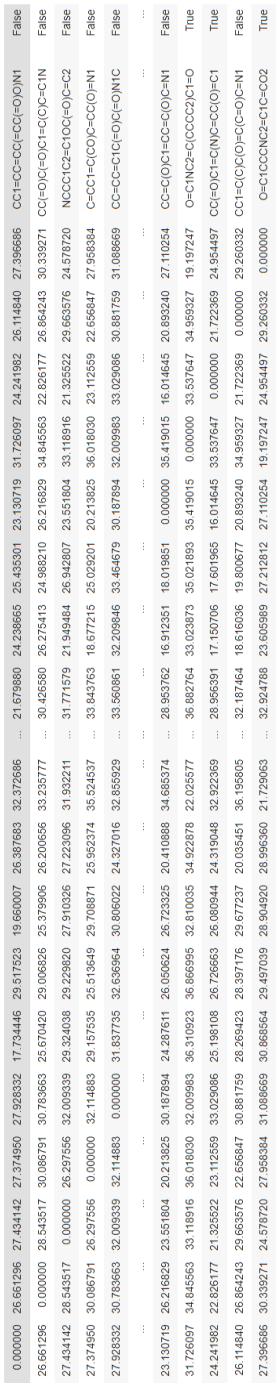
The code can be run as a single process or as several simultaneous processes. Training is a slow process, so it is necessary to run it in several simultaneous processes. Taking into account the technical characteristics of the computer we have used 8 CPUs.

It is also important to note that two folders are generated. The molecule_gen folder contains a csv file with the results. These include the generated molecule in SMILES format and the corresponding rewards. On the other hand, TensorFlow generates a ckpt folder that saves the model every 256 steps.

A.3 Code

The modified generative model and the codes can be found in the following GitHub repository: https://github.com/AlejandroGM2000/TFG_AlejandroGarcia.git. Please note that they are generally private, so you will need to request access.

B Additional figures



	0	1	2	3	4	5	6	7	8	9	...	540	541	542	543	544	545	546	547	smile	database
0.000000	26.661296	27.434142	27.374950	27.928332	17.734446	29.517523	19.660007	26.387683	32.372686	...	21.679880	24.238665	25.435301	23.130719	31.726097	24.241982	26.114840	27.396686	CC1=CC=CC(=C1)C(=O)N1	False
26.661296	0.000000	28.543517	30.086791	30.783663	25.670420	29.006826	25.379906	26.200656	33.235777	...	30.426580	26.275413	24.988210	26.216829	34.845563	22.826177	26.864243	30.339271	CC(=O)C(=O)C1=C(C)C=C1N	False
27.434142	28.543517	0.000000	26.297556	32.009339	29.324038	29.229820	27.910326	27.223096	31.932211	...	31.771579	21.949484	26.942807	23.551804	33.118916	21.325522	29.663576	24.578720	NCCC1C2=C1OC(=O)C=C2	False
27.374950	30.086791	26.297556	0.000000	32.114883	29.157535	25.513649	29.708871	25.952374	35.524537	...	33.843763	18.677215	25.029201	20.213825	36.018030	23.112559	22.656847	27.958384	C=CC1=C(CO)C=CC(O)=N1	False
27.928332	30.783663	32.009339	32.114883	0.000000	31.837735	32.636964	30.806022	24.327016	32.855929	...	33.560861	32.209846	33.464679	30.187894	32.009983	33.029086	30.881759	31.088669	CC=CC=C1C(=O)C(=O)N1C	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
23.130719	26.216829	23.551804	20.213825	30.187894	24.287611	26.050624	26.723325	20.410888	34.685374	...	28.953762	16.912351	18.019851	0.000000	35.419015	16.014645	20.893240	27.110254	CC=C(O)C1=CC=C(O)C=N1	False
31.726097	34.845563	33.118916	36.018030	32.009983	36.310923	36.866995	32.810035	34.922878	22.025577	...	36.882764	33.023873	35.021893	35.419015	0.000000	33.537647	34.959327	19.197247	O=C1NC2=C(CCCC2)C1=O	True
24.241982	22.826177	21.325522	23.112559	33.029086	25.198108	26.726663	26.080944	24.319048	32.922369	...	28.956391	17.150706	17.601965	16.014645	33.537647	0.000000	21.722369	24.954497	CC(=O)C1=C(N)C=CC(O)=C1	True
26.114840	26.864243	29.663576	22.656847	30.881759	28.269423	28.397176	29.677237	20.035451	36.195805	...	32.187464	18.616036	19.800677	20.893240	34.959327	21.722369	0.000000	29.260332	CC1=C(C)C(O)=C(C=O)C=N1	False
27.396686	30.339271	24.578720	27.958384	31.088669	30.868564	29.497039	28.904920	28.996360	21.729063	...	32.924788	23.605989	27.212812	27.110254	19.197247	24.954497	29.260332	0.000000	O=C1CCCNC2=C1C=CO2	True

Figure 34. Dataframe that contains the Euclidean distance between all the Mol2Vec embeddings of the generated molecules for $C_8H_9NO_2$.

	0	1	2	3	4	5	6	7	8	9	...	540	541	542	543	544	545	546	547	smile	database
0	1.000	0.146	0.050	0.069	0.074	0.224	0.107	0.192	0.105	0.049	...	0.356	0.105	0.125	0.113	0.073	0.113	0.151	0.062	CC1=CC=CC(=CC(=O)O)N1	False
1	0.146	1.000	0.098	0.058	0.085	0.152	0.100	0.146	0.143	0.118	...	0.174	0.143	0.143	0.083	0.061	0.209	0.125	0.070	CC(=O)C(=O)C1=C(C)C=C1N	False
2	0.050	0.098	1.000	0.068	0.073	0.070	0.068	0.086	0.103	0.083	...	0.069	0.103	0.067	0.053	0.053	0.091	0.051	0.095	NCCC1C2=C1OC(=O)C=C2	False
3	0.069	0.058	0.068	1.000	0.036	0.071	0.127	0.069	0.068	0.032	...	0.070	0.145	0.105	0.135	0.017	0.113	0.109	0.046	C=CC1=C(CO)C=CC(O)=N1	False
4	0.074	0.085	0.073	0.036	1.000	0.077	0.074	0.074	0.157	0.071	...	0.075	0.073	0.054	0.078	0.078	0.058	0.075	0.067	CC=CC=C1C(=O)C(=O)N1C	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
543	0.113	0.083	0.053	0.135	0.078	0.096	0.113	0.073	0.154	0.034	...	0.115	0.154	0.132	1.000	0.018	0.167	0.184	0.048	CC=C(O)C1=CC=C(O)C=N1	False
544	0.073	0.061	0.053	0.017	0.078	0.036	0.017	0.054	0.053	0.109	...	0.055	0.071	0.034	0.018	1.000	0.037	0.036	0.121	O=C1NC2=C(CCCC2)C1=O	True
545	0.113	0.209	0.091	0.113	0.058	0.140	0.113	0.113	0.091	0.070	...	0.115	0.111	0.154	0.167	0.037	1.000	0.115	0.066	CC(=O)C1=C(N)C=CC(O)=C1	True
546	0.151	0.125	0.051	0.109	0.075	0.157	0.109	0.130	0.170	0.050	...	0.154	0.216	0.192	0.184	0.036	0.115	1.000	0.047	CC1=C(C)C(O)=C(C=O)C=N1	False
547	0.062	0.070	0.095	0.046	0.067	0.048	0.133	0.062	0.095	0.111	...	0.063	0.078	0.078	0.048	0.121	0.066	0.047	1.000	O=C1CCCNC2=C1C=CO2	True

Figure 35. Dataframe that contains the Tanimoto similarity between all the generated molecules for $C_8H_9NO_2$.

	SmilesGenerate	FirstSimilarSmiles	FirstED	SecondSimilarSmiles	SecondED	ThirdSimilarSmiles	ThirdED	FourSimilarSmiles	FourED
3312	CC1=CC=CC(=CC(=O)O)N1	C1=CC(=O)C(C(=C1)C(=O)O)N	17.235823	C1C(=CC(=O)O)NC1=O	18.164725	C(=C1C(C(=O)N1)O)C(=O)O	18.294496	C1=CC(=O)C(=N)C(=C1)C(=O)O	18.622194
3956	CC(=O)C(=O)C1=C(C)C=C1N	CC1=CC=C(O1)C(=O)C(=O)C	18.464374	CC1=CC(=C(C=C1)C(=O)C(=O)C)O	19.845195	CC(=O)C(=O)C1=CC=CO1	20.377866	CC(=O)C1=C(CCC1=N)O	20.417820
4255	NCCC1C2=C1OC(=O)C=C2	C1=CC(=CC(=C1)O)CCN	18.191392	C1=CC(=CC=C1CCN)O	18.587865	C1=CC(=C(C=C1O)CCN)O	18.649597	CCCC1=CC(=CC(=O)O1)O	18.651368
5371	C=CC1=C(CO)C=CC(O)=N1	C=CC1=CC(=O)NC(=O)N1	13.584311	C=CC1=CNC(=O)NC1=O	15.092981	CC1=NC(=CC=C1)C=C	15.364752	CC1=NC=C(C=C1)C=C	15.678105
5577	CC=CC=C1C(=O)C(=O)N1C	CC=CC=CC1=CC(=O)CCN1	22.546209	C1C2N(C1=O)C(C(=CC=O)O2)C(=O)O	24.043416	CN1C(=O)C(C(=O)NC1=O)C=O	24.187450	CN1C(=O)CC(=C(C1=O)O)OC	24.198024
...	...	...	...	...	...	...	...	...	...
119650	CC=C(O)C1=CC=C(O)C=N1	CC=C(C)C1=CC=CC=N1	12.106321	CC=C(C)C1=CN=CC=C1	12.306055	C1=CC(=C(N=C1)C(=O)O)O	12.433172	C1=CC(=O)NC=C1C(=O)O	12.770432
119681	O=C1NC2=C(CCCC2)C1=O	C1CCC2=C(C1)C(=O)NC2=O	10.440604	C1CC2=C(CCC(=O)N2)C(=O)C1	11.502955	C1CC2=C(C1O)C(=O)NC2=O	16.132496	C1CC2=C(C(=O)CC2)NC1	17.021944
119814	CC(=O)C1=C(NC=CC(O)=C1	C1=CC(=C(C=C1O)C(=O)O)N	7.849352	CC(=O)C1=CC=CC=C1N	8.852031	C1=CC(=C(C(=C1N)O)C(=O)O	8.898838	CC1=C(C=CC(=C1O)O)C(=O)C	9.058594
120259	CC1=C(C)C(O)=C(C=O)C=N1	CC1=CN=C(C(=C1C=O)O)C	6.326220	CC1=NC=C(C(=C1O)CO)C=O	9.818225	CC1=CC(=C(C(=C1)C=O)O)C	11.193215	CC1=NC=C(C(=C1O)C(=O)O)C=O	11.410491
120293	O=C1CCCNC2=C1C=CO2	CC1=CC2=C(C=C1)NCCC2	13.092314	C1CC2=C(C=CC(=C2)O)NC1	13.717723	C1CNC2=CC=CC=C2C1=O	13.820270	CC1=CC2=C(C=C1)NCCC2=O	13.905943

Figure 36. The 'SmilesGenerate' column contains the SMILES of the generated molecules for $C_8H_9NO_2$. There are four other columns that contain SMILES. These SMILES correspond to real molecules extracted from the dataset provided by SEES:lab. They are the molecules in the dataset that have the smallest Euclidean distance between their Mol2Vec embedding and the Mol2Vec embedding of the generated molecule. They are sorted in ascending order. The Euclidean distance is stored in the respective columns