

People's Democratic Republic of Algeria Ministry of Higher Education and Scientific Research

Dr.YAHIA FARES UNIVERSITY OF MEDEA

FACULTY OF SCIENCES

Department of Mathematics and Computer Science

Master's thesis in systems engineering and web technology

De novo Molecular generation based on GCPN and GraphAF

Prepared by:

Houssem Hassnaoui

Fateh Chikhi

Supervisor: Hentabli Hamza

Co-Supervisor: Chergui Nadjah

2022/2023

Contents

1 Introduction **9**
- 1.1 Overview . . . 9
- 1.2 Problem background . . . 10
- 1.3 Problem statement . . . 10
- 1.4 Research questions . . . 11
- 1.5 Research objectives . . . 11
- 1.6 Research scope . . . 11

2 Literature review **12**
- 2.1 Introduction . . . 12
- 2.2 Computer aided drug design . . . 13
- 2.3 Chemoinformatics . . . 14
- 2.4 Application GNN . . . 15
- 2.5 Molecular representations . . . 16
- 2.6 Graph representation . . . 17
  - 2.6.1 Molecular Graph Visualization . . . 17
  - 2.6.2 Define the molecular graph . . . 18
- 2.7 Molecular similarity measurement . . . 19
  - 2.7.1 Introduction of molecular similarity measurement . . . 19
  - 2.7.2 Molecular fingerprint . . . 20
  - 2.7.3 Distance phenomena in high-dimensional chemical descriptor spaces . . . 21
- 2.8 Deep Learning . . . 22
  - 2.8.1 Artificial Intelligence . . . 22
  - 2.8.2 Machine Learning . . . 22
  - 2.8.3 Deep Learning . . . 23
  - 2.8.4 What is a Graph? . . . 25
  - 2.8.5 GCPN . . . 25
  - 2.8.6 Generative Adversarial Networks for molecular simulation . . . 26
  - 2.8.7 GraphAF . . . 28
  - 2.8.8 autoregressive flow generation . . . 29
  - 2.8.9 RGCN . . . 29

3 Research Methodology **31**
- 3.1 Intoduction . . . 31
- 3.2 Research Framework . . . 31
- 3.3 Dataset . . . 32
  - 3.3.1 MUV . . . 32
  - 3.3.2 BACE . . . 34
- 3.4 Steps and Design . . . 34
  - 3.4.1 Introduction . . . 34
  - 3.4.2 Dataset Splitting . . . 35
  - 3.4.3 Choosing the Activity Class . . . 35
  - 3.4.4 Data Reading and Familiarization . . . 35
  - 3.4.5 Graph-Based Model . . . 35
- 3.5 Adversarial training . . . 36
  - 3.5.1 Hyperparameters . . . 37
- 3.6 Molecular generation . . . 38
  - 3.6.1 Define the Model: GCPN . . . 38
  - 3.6.2 Pretraining and Generation . . . 39
  - 3.6.3 Define the Model: GraphAF . . . 41
- 3.7 Performance Evaluation . . . 43
- 3.8 similarity measurement . . . 43
  - 3.8.1 definition . . . 43
  - 3.8.2 why we used this similarity . . . 43

3.9 Benchmarking . . . 44
3.10 Summary . . . 44

4 Results and Discussion **45**
4.1 Introduction . . . 45
4.2 Development Environment and Used Materials . . . 46
4.2.1 Programming language . . . 46
4.2.2 Environment . . . 46
4.2.3 Libraries . . . 47
4.3 Experiment Design . . . 48
4.3.1 Property Optimization . . . 48
4.3.2 Property Targeting . . . 48
4.3.3 Constrained Property Optimization . . . 48
4.4 FUNCTIONS TO CALCULATE THE SIMILARITY . . . 49
4.4.1 Calculate in MUV . . . 49
4.4.2 MUV_466 GENERATION AND VALIDATION . . . 52
4.4.3 MUV classes molecular validation . . . 53
4.4.4 MUV classes plot result . . . 54
4.4.5 Calculate in BACE . . . 55
4.5 Molecule Generation Results . . . 56
4.5.1 GCPN Results . . . 56
4.5.2 GraphAF Results . . . 58
4.6 result and discussion . . . 59
4.7 Conclusion . . . 60

5 General conclusion **61**
5.1 Future work . . . 62

List of Figures

2.1 Schematic overview of in silico-based computer-aided drug design process and its applications. . . . 14
2.2 Application of GNN in all stages . . . 15
2.3 Different Types of Molecular Representations . . . 16
2.4 Representing molecules . . . 17
2.5 Representing molecules . . . 18
2.6 Represents The Relation Between AI, ML and Deep Learning. . . . 22
2.7 Directed Graph and Undirected Graph . . . 25
2.8 Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation . . . 26
2.9 GAN architecture for molecule simulations . . . 27
2.10 Overview of the proposed GraphAF model. (a) Illustration of the generative procedure. New nodes or edges are marked in red. Starting from an empty graph and iteratively sample random variables to map them to atom/bond features. The numbered first three steps correspond to the maps in the bottom figure of Fig. 1(b). (b) Computation graph of GraphAF. The left side are the nodes and edges and the right are latent variables . . . 28
2.11 A schematic diagram of message passing in a directed graph with 6 nodes . . . 30

3.1 Displaying loaded dataset . . . 35
3.2 MUV classes . . . 37
3.3 some generated molecules with GCPN . . . 40
3.4 some generated molecules with GraphAF . . . 42

4.1 MUV_466 validation for GCPN . . . 52
4.2 MUV_466 validation for GRAPH_AF . . . 52
4.3 MUV comparation . . . 54
4.4 BACE comparation . . . 56
4.5 MUV Dataset Results with GCPN . . . 57

4.6 BACE Dataset Results with GCPN 57
4.7 MUV Dataset Results with GraphAF 58
4.8 BACE Dataset Results with GraphAF 59

List of Tables

2.1 Table : Formulas for the various similarity and distance metrics 20
2.2 molecular graph generation. The comparison of training is only conducted between autoregressive models. 24

3.1 Heyperparametre specification for muv 37
3.2 Heyperparametre specification for BACE 38
3.3 smiles 40
3.4 results of molecules using GCPN model 41
3.5 some smiles 41
3.6 results of molecules with GraphAF model 42

4.1 MUV dataset similarity molecular validation class GCPN and GRAPH_AF . . 53
4.2 The similarity accuracy 53
4.3 GCPN molecular similarity for BACE 55
4.4 GRAPH_AF molecular similarity for BACE 55
4.5 similarity accuracy 56
4.6 similarity accuracy 59
4.7 similarity accuracy models 60

Dedication

Houssem Hassnaoui

In the name of merciful God, thank you very much to him, for his help and the will he gave us to overcome all the obstacles and difficulties during five years of study and for having enlightened our path in order to do this work. To my mother and my father (rabi yarhmo), who was always standing with me, encouraging me to pursue the path of knowledge, pushing me forward and inspiring me all the time .no dedication can express my respect, my eternal love and my consideration for the sacrifices you have made for my education and my well being. May God, the Most High, grant you health, happiness and long life . To all my sisters and brothers one by one, who remember me by their prayers and pushing me to do my best and have always been by my side. A special thanks to my sisters sons akrem ,ranim, loai and racim. finally,i extend my respectful thanks to the members of the jury dr.sahmadi , dr.redha mazari and specially dr.chargui how standin and working with us .

Dedication

Fateh Chikhi

In the name of merciful God, thank you very much to him, for his help and the will he gave us to overcome all the obstacles and difficulties during our years of study and for having enlightened our path in order to do this work. I would like to dedicate this report to my Father and my mother.

I would like to express my deepest gratitude and appreciation to all profs for their unwavering support and guidance throughout this research project. Their mentorship and encouragement have been invaluable and instrumental in the successful completion of this work.

I would also like to extend my heartfelt thanks to my friends and my partner (Houssem Hasnaoui), whose constant love, encouragement, and understanding have been the pillars of strength during this journey. Their unwavering support and belief in me have been the driving force behind my accomplishments.

This work is dedicated to all those who believe in the power of research, innovation, and the pursuit of knowledge to shape a better future.

Abstract

There is increasing adoption of artificial intelligence in drug
discovery. De novo molecular design involves the creation of novel molecules with desiredproperties and functions. The process is
often computationally intensive andrequires the exploration of vast chemical space. In recent years, deep learning-based approaches have emerged as powerful tools for De Novo molecular design. These methods involve the use of deep neural networks to learn patterns in molecular structures and properties, enabling the generation of new molecules with desired properties.
One popular approach is the use of generative mod- els, To De Novo , we construct two dataset to date, namely MUV and BACE such as Graph Convolutional Policy Network (GCPN) and GraphAF, which learn to generate molecules that mimic a given dataset. We will implement two graph generative models GCPN and GraphAF. We first pretrain both models on MUV and BACE datasets. We finetune both models with reinforcement learning to optimize two properties of generated molecule , After conducting a comprehensive analysis of each model, we have determined that GCPN outperforms GraphAF, delivering significantly better results.

Résumé

On constate une adoption croissante de l'intelligence artificielle dans la découverte de médicaments. La conception moléculaire de novo implique la création de nouvelles molécules aux propriétés et fonctions désirées. Ce processus est souvent intensif en calcul et nécessite l'exploration d'un vaste espace chimique. Ces dernières années, les approches basées sur l'apprentissage profond ont émergé comme des outils puissants pour la conception moléculaire de novo. Ces méthodes utilisent des réseaux neuronaux profonds pour apprendre les motifs des structures moléculaires et leurs propriétés, permettant ainsi la génération de nouvelles molécules aux propriétés désirées. Une approche populaire est l'utilisation de modèles génératifs tels que le Graph Convolutional Policy Network (GCPN) et GraphAF, qui apprennent à générer des molécules qui imitent un ensemble de données donné. Nous mettrons en œuvre deux modèles générateurs de graphes, GCPN et GraphAF. Nous pré-entraînerons d'abord ces deux modèles sur les ensembles de données MUV et BACE. Ensuite, nous les affinerons avec l'apprentissage par renforcement afin d'optimiser deux propriétés des molécules générées. Après avoir réalisé une analyse approfondie de chaque modèle, nous avons déterminé que le GCPN surpasse le GraphAF, offrant des résultats significativement meilleurs.

المقدمة

تزايد اعتماد تقنية الذكاء الاصطناعي في اكتشاف الأدوية. ينطوي التصميم الجزيئي De Novo على إنشاء جزيئات جديدة ذات خصائص ووظائف مرغوب فيها. هذه العملية تتطلب تكامل الحوسبة واستكشاف مساحة كيميائية واسعة. في السنوات الأخيرة، ظهرت النهج المبني على التعلم العميق كأدوات قوية لتصميم الجزيئات De Novo . تشمل هذه الطرق استخدام deep neural networks لتعلم الأنماط في البنية الجزيئية والخصائص، مما يمكن من إنتاج جزيئات جديدة ذات الخصائص المطلوبة. إحدى الطرق الشهيرة هي استخدام Graph Convolutional Policy Network ،
مثل (GCPN) و graph autoregressive flow (GraphAF) ، التي تتعلم توليد جزيئات تشبه مجموعة البيانات المعطاة. سنقوم بتنفيذ نموذجين لتوليد الرسوم البيانية، GCPN و GraphAF . نحن نبدأ بتدريب النموذجين على مجموعتي البيانات MUV و BACE . نقوم بضبط النموذجين باستخدام تعلم التعزيز لتحسين خاصيتين للجزيئات المولدة. بعد إجراء تحليل شامل لكل نموذج، توصلنا إلى أن GCPN يفوق GraphAF ، محققًا نتائج أفضل بشكل ملحوظ.

Chapter 1

Introduction

1.1 Overview

De novo molecular drug design is the process of designing new drug molecules from scratch using computational methods, rather than modifying existing drugs. The goal of de novo drug design is to identify molecules with desired properties, such as high potency, selectivity for a specific target,low toxicity, and good pharmacokinetic properties.[1]
This process involves the use of computer algorithms to generate and evaluate new compounds based on their predicted interactions with the target protein or biological system.[2]
The algorithms use a combination of molecular mechanics, quantum mechanics, and machine learning techniques to generate and rank potential drug candidates.[3]
De novo drug design is a challenging process that requires a deep understanding of the target protein or biological system.[4]
as well as expertise in computational chemistry and molecular modeling.[5]
It also requires a significant amount of computational power, as the search for new compounds can involve exploring vast chemical space.[6]
Despite its challenges, de novo drug design has the potential to revolutionize the drug discovery process by enabling the rapid development of new and highly effective drugs with minimal side effects.[7]

1.2 Problem background

De novo molecular design, also known as computational molecular design or computer-aided drug design, is a rapidly developing field in modern drug discovery. The process involves designing novel small molecules with desired biological activities, which can be used as potential drug candidates. This approach has gained popularity due to its potential to reduce the cost, time, and resources involved in the traditional drug discovery process, which can take years and involve high costs.

However, de novo molecular design still faces several challenges. One of the most significant challenges is the complexity of predicting how the designed molecules will interact with biological systems. This requires sophisticated computational tools and techniques that can predict the properties and behavior of the designed molecules accurately. Additionally, the lack of understanding of the underlying mechanisms of many diseases, as well as the limitations of available experimental data, can also pose significant challenges for designing effective drugs.

1.3 Problem statement

The main problem addressed in this report is the limitations and challenges faced in de novo molecular design for drug discovery. Although the use of computational methods has helped to speed up the process of designing novel drug candidates, there are still limitations in predicting the efficacy and safety of the designed molecules accurately. Additionally, the lack of experimental data, the complexity of biological systems, and the limitations of available computational tools can pose significant challenges in designing effective drugs.

Therefore, the problem statement is how to overcome these limitations and challenges in de novo molecular design to design safe, effective, and affordable drugs for various diseases. This requires developing more accurate computational tools, integrating available experimental data with computational methods, and improving our understanding of the underlying mechanisms of diseases. Addressing these challenges can help to speed up the drug discovery process and reduce the cost and resources required to develop new drugs

1.4 Research questions

The main issue in this research is how can De novo molecular design be used to create new compounds with specific properties?What is the drug discovery and where is deep learning there? What is the current state of research on de novo molecular drug design, and what are the key challenges and future directions for this field?

1.5 Research objectives

1.To review the existing literature on de novo molecular drug design and identify the major trends and developments in this field.
2.To evaluate the advantages and limitations of de novo drug design approaches and their potential impact on drug discovery.
3.To analyze the computational methods and tools used in de novo drug design and assess their accuracy and reliability.
4.To identify the key challenges and barriers that need to be overcome in order to realize the full potential of de novo drug design.
5.To propose recommendations for future research in this field and highlight the potential applications and impact of de novo drug design on drug discovery.
6. To evaluate the performance of GCPN (Graph Convolutional Policy Network) and GraphAF (Graph Autoregressive Flow) methods in generating novel molecules with desired properties through de novo molecular design.
7. To compare the performance of GCPN and GraphAF with other deep learning-based methods for de novo molecular design, such as variational autoencoders and generative adversarial networks.

1.6 Research scope

This review will cover the current state of research on de novo molecular drug design, including the principles and methods used in this approach, the computational tools and algorithms employed, and the applications and limitations of de novo drug design. The review will also highlight the key challenges and barriers to progress in this field, as well as the potential impact and future directions for de novo drug design. The review will be based on a comprehensive search of the existing literature, including academic journals, conference proceedings, and books, as well as interviews with experts in the field using GCPN and GraphAF to generate molecules, because the resulting graph is the most intuitive and concise way to represent molecules.

Chapter 2

Literature review

2.1 Introduction

De Novo molecular design based on deep learning has become an increasingly active area of research in recent years.[8] A number of studies have investigated the use of various deep learning-based methods for generating novel molecules with desired properties.

One promising approach is the use of generative models, such as variational autoencoders (VAEs) and generative adversarial networks (GANs). in this field we can mention the work of Segler et al.[9] (2018) which developed a VAE-based method called MolVAE for de novo molecular design. MolVAE was able to generate novel molecules with similar properties to those of the training data and was able to identify novel structures with high drug-likeness scores.[10]

Another approach is the use of reinforcement learning (RL) to optimize the properties of the generated molecules. For example, Olivecrona et al. (2017) developed an RL-based method called REINVENT, which generated molecules with improved bioactivity compared to those in the training data.[11]

More recently, attention-based models, such as Graph Attention Networks (GATs), have been used for De Novo molecular design.We cite the work of Hu et al.[12] (2020)which developed a method called GraphAF, which uses GATs to learn attention weights between atoms and bonds in a molecular graph.[13] GraphAF was able to generate molecules with higher property scores than other methods on several benchmark datasets.[14]

Another attention-based method, Graph Convolutional Policy Network (GCPN), has also shown promising results in generating novel molecules with desired properties. You et al. (2018) used GCPN to generate molecules with high QED (quantitative estimate of drug-likeness) scores and low molecular weight, which are important criteria for drug discovery.[8]

Overall, deep learning-based De Novo molecular design methods have shown pertinent results in accelerating the drug discovery process and have the potential to revolutionize the field of molecular design. However, challenges such as the need for large amounts of training data and the potential for generating molecules with undesirable properties still need to be addressed. Further research is needed to improve the performance and applicability of these methods.

2.2 Computer aided drug design

Computer-aided drug design (CADD) has become an increasingly important tool in the drug discovery process. A range of computational methods and techniques are employed in CADD, including molecular modeling, virtual screening, and machine learning.

Molecular modeling is a key step in CADD, as it enables researchers to predict the three-dimensional structure of a target protein and design molecules that can bind to it. Molecular docking and molecular dynamics simulations are commonly used in this regard. For example, molecular docking has been used to predict the binding affinity of drug candidates to target proteins, while molecular dynamics simulations have been used to study the behavior of proteins and their interactions with ligands over time [9, 15].

Virtual screening is another important aspect of CADD, which involves the use of computational methods to identify potential drug candidates from large libraries of compounds. This can be done using ligand-based methods, such as pharmacophore modeling and quantitative structure-activity relationship (QSAR) analysis, or structure-based methods, such as molecular docking and molecular dynamics simulations [16].

In recent years, machine learning has emerged as a powerful tool for CADD. Deep learning-based methods, such as neural networks and convolutional neural networks, have shown promise in improving the accuracy and efficiency of virtual screening and molecular modeling. For example, deep learning has been used to predict drug-target interactions and to design novel drug candidates with high efficacy and selectivity [12, 17].

Overall, CADD has played a critical role in accelerating the drug discovery process, enabling researchers to rapidly screen and optimize large libraries of compounds and design novel drug candidates with high efficacy and selectivity.

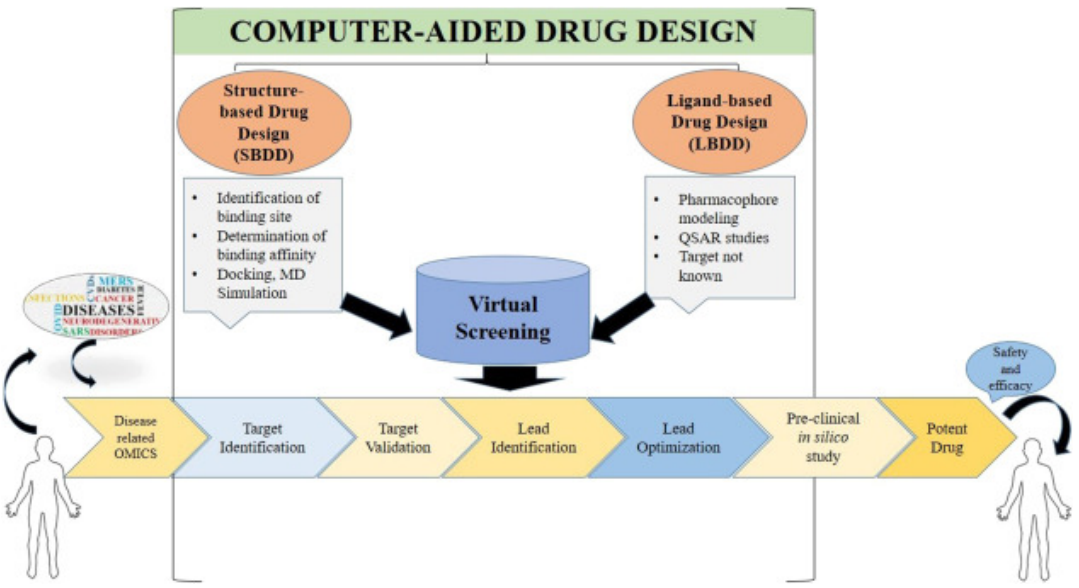


Figure 2.1: Schematic overview of in silico-based computer-aided drug design process and its applications.

2.3 Chemoinformatics

Chemoinformatics is a field that combines the principles of chemistry, computer science, and statistics to develop and apply tools and methods for the analysis, management, and visualization of chemical and biological data [13]. It involves the use of mathematical and computational techniques to analyze and manipulate chemical data, with the goal of exploring and understanding chemical and biological systems, designing new drugs, and optimizing their properties [18].

One of the key tools used in chemoinformatics is molecular modeling, which involves the use of computational methods to predict the properties of new molecules. Another important tool is machine learning, which can be used to develop predictive models of chemical properties [19]. Data mining and database management systems are also used to search large databases of chemical compounds for potential drug candidates. Additionally, chemoinformatics can be used to design new chemical structures with specific properties, such as increased solubility or improved efficacy [20].

Chemoinformatics has a wide range of applications, including drug discovery, materials science, environmental chemistry, and food science (7). The field is rapidly evolving and has the potential to make significant contributions to a variety of scientific disciplines.

2.4 Application GNN

Graph Neural Networks (GNNs) have been applied in various stages of automated de novo molecular design, including representation learning, property prediction, and molecular generation.

Representation deep learning: GNNs are used to convert molecular structures into graph representations, which can be easily processed and analyzed by neural networks.

Property prediction: GNNs are trained on molecular structures and their properties, such as activity, toxicity, and stability, to predict new molecular properties.

Molecular generation: GNNs can be used to generate new molecular structures by predicting molecular graphs, or by modifying existing molecular structures to optimize specific properties.

Overall, GNNs have shown promising results in molecular design and have the potential to greatly improve the efficiency and accuracy of automated de novo molecular design

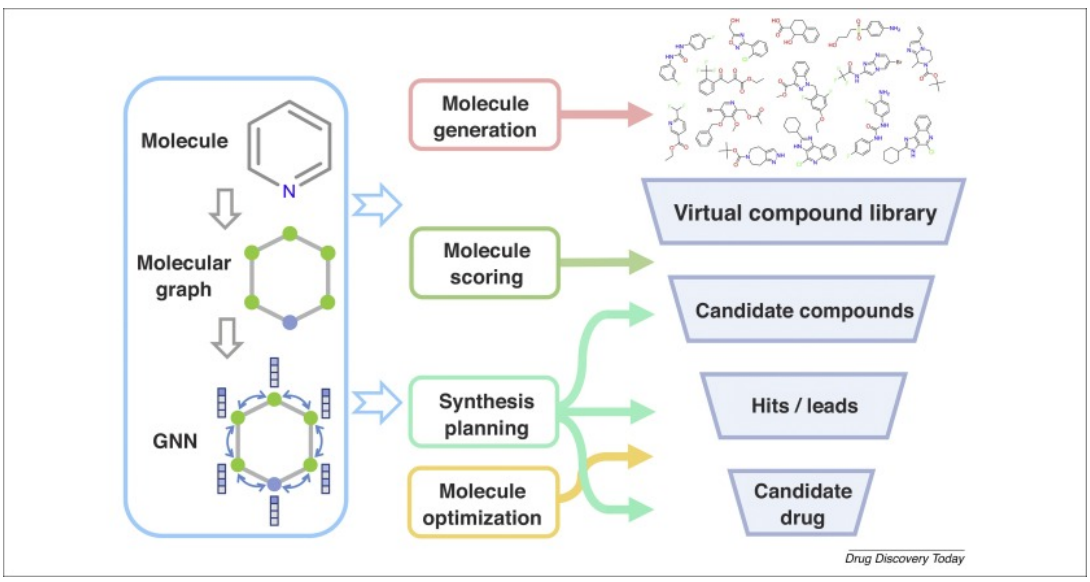


Figure 2.2: Application of GNN in all stages

2.5 Molecular representations

Molecular representations are visual or mathematical depictions of the structure and properties of molecules. They are used in various fields, including chemistry, biochemistry, materials science, and drug discovery [21].

One common type of molecular representation is the molecular formula, which provides information on the types and numbers of atoms present in a molecule. Another is the Lewis structure, which shows the bonding and non-bonding electron pairs in a molecule [22, 23].

In addition to these physical models, there are also mathematical representations of molecules. One example is the molecular graph, which represents a molecule as a graph, with atoms as vertices and bonds as edges. This can be useful for analyzing the topology of molecules and predicting their properties[24].

Molecular representations are important in many areas of research. For example, drug designers use molecular models to design and optimize drug candidates. Chemists use them to study chemical reactions and to predict the properties of new compounds. Biologists use them to understand the structure and function of biomolecules such as proteins and DNA [25].

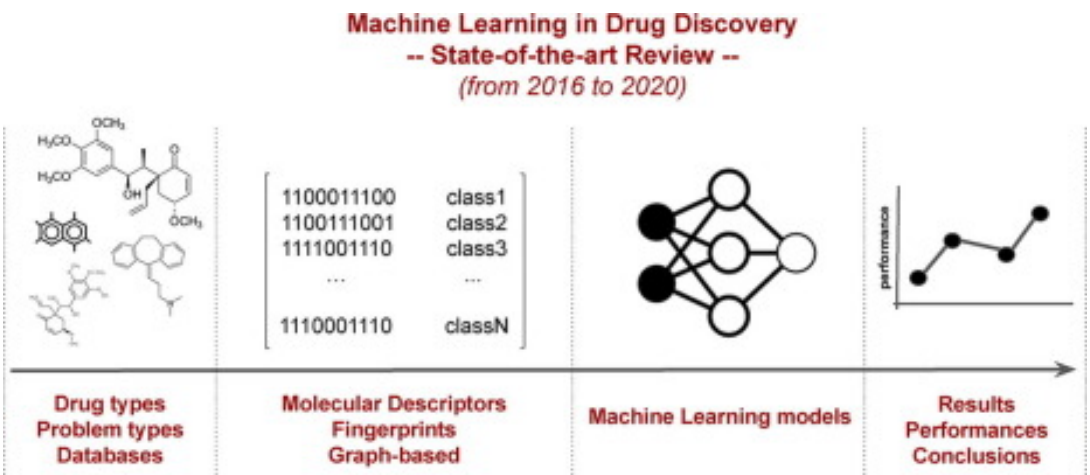


Figure 2.3: Different Types of Molecular Representations

2.6 Graph representation

Graph representation is a powerful approach for representing molecules in machine learning models. Unlike text-based representations, such as SMILES, which can be vulnerable to errors due to small changes in the sequence of characters, molecular graphs provide a more robust representation of the underlying molecular structure[26].

One of the advantages of molecular graphs is their ability to handle partial structures. Partially generated molecular graphs can be interpreted as substructures, allowing for the validation of intermediate representations during the process of generating a complete molecular graph. In contrast, partially generated text-based representations may not always be meaningful and cannot be checked for chemical validity[27]

Furthermore, molecular graphs allow for chemical checks, such as valency checks, to be performed during the generation of the molecule. Valency checks ensure that each atom in the molecule has the correct number of bonds, which is critical for the stability and function of the molecule. These checks are not possible with text-based representations, which can lead to the generation of invalid molecules[28].

2.6.1 Molecular Graph Visualization

Graph visualization tools like Graphviz or software packages like RDKit can generate graphical representations of molecular structures. These visualizations use nodes to represent atoms and edges to represent chemical bonds. The nodes may include additional information such as atom types, charges, or other properties.

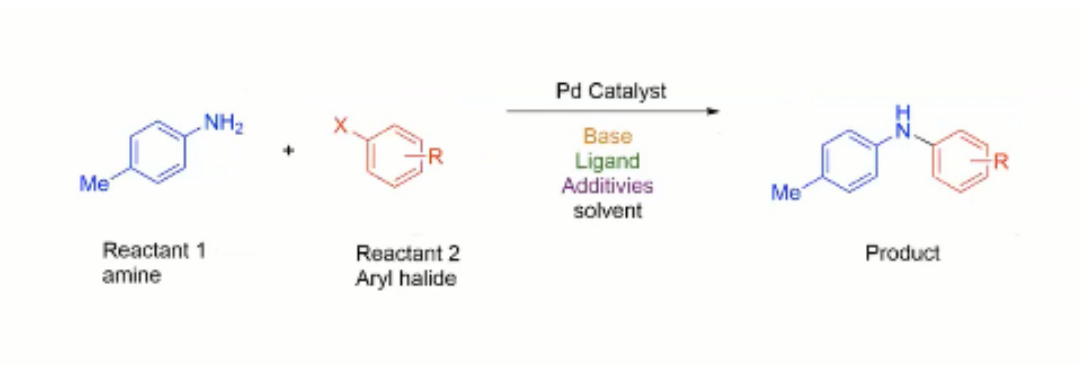


Figure 2.4: Representing molecules

2.6.2 Define the molecular graph

A labeled graph whose nodes correspond to the atoms of the molecule and the edges correspond to the bonds G = (A,B,X,Y)

this is the molecular graphical representation of C7H9N :

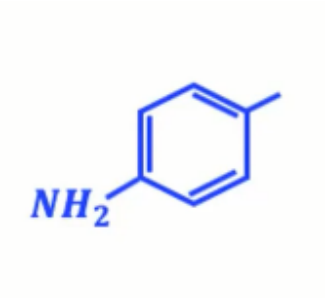


Figure 2.5: Representing molecules

A is the set of atoms :
A = [H, H, N, C, C, C, C, C, C,C]
order[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
X represents the atom content matrix which contain the feature or information of the node such as type of atoms :
for example this is the type of atom that we have :['C','N','O','F','CL','H','BR']
this is the representation graphic for this molecular :

$$\text{X matrix : } \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

B represent the set of bond content matrix which contain the relation bonds between atoms

$$\text{B matrix : } \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Y represent the set of type of bond content matrix which main the for each atom type of here own let's consider that we have 4 type of bonds [single,Double,triple,aromatic] :

$$\text{Y matrix : } \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

2.7 Molecular similarity measurement

2.7.1 Introduction of molecular similarity measurement

Molecular similarity measurement is a fundamental concept in computational chemistry and drug discovery that involves assessing the degree of similarity between molecules. It plays a crucial role in various areas, including virtual screening, lead optimization, chemical database searching, and toxicity prediction. By comparing the structural, physicochemical, or biological characteristics of molecules, scientists can identify compounds with similar properties or activities, aiding in the discovery and development of new drugs or materials[29].
The primary objective of molecular similarity measurement is to quantify the resemblance or closeness between two or more molecules based on their chemical structures, properties, or behavior. The underlying principle is that structurally or functionally similar molecules often exhibit similar biological activities, making them potential candidates for drug targets or lead compounds.

2.7.2 Molecular fingerprint

There is lot of methods exist to calculate map molecular structures to bit strings . they also known as molecular descriptors or molecular representations, are mathematical representations of molecules used in chemoinformatics and computational chemistry. They encode structural and chemical information about a molecule into a fixed-length binary or integer vector. here some type of fingerprints :

1_Morgan Fingerprints: Also known as circular fingerprints or Morgan algorithm, these fingerprints capture the local environment around each atom in a molecule using circular substructures. They are useful for similarity searching, virtual screening, and QSAR modeling.

2_PubChem Fingerprints: These fingerprints are based on the PubChem Compound Database and encode the presence or absence of specific substructures or chemical features in a molecule.

3_Pharmacophore Fingerprints: Pharmacophore fingerprints represent the key features of a molecule that are important for its interaction with a biological target. They are often used in virtual screening and drug discovery. in this table we show the popular mesurement distance .

distance metric	**formula for continous variables**	**formula for dichotomous variables**
Manhattan	$D(A,B) = \sum_{i=1}^{n} \left\lvert x_i^A - x_i^B \right\rvert$	D(A, B) = a + b - 2c
Euclidean	$D(A,B) = \sqrt{\sum_{i=1}^{n}(x_i^A - x_i^B)^2}$	$D(A,B) = [a+b-2c]^{\frac{1}{2}}$
Cosine	$D(A,B) = \frac{\sum_{i=1}^{n}(x_i^A \cdot x_i^B)}{\sqrt{\sum_{i=1}^{n}(x_i^A)^2} \cdot \sqrt{\sum_{i=1}^{n}(x_i^B)^2}}$	$S(A,B) = \frac{c}{(ab)^{\frac{1}{2}}}$
Dice	$Sim_{AB} = \frac{2\sum_{i=1}^{N} W_{iA}W_{iB}}{\sum_{i=1}^{N}(W_{iA})^2 + \sum_{i=1}^{N}(W_{iB})^2}$	$S(A,B) = \frac{2c}{a+b}$
Tanimoto	$Sim_{AB} = \frac{\sum_{i=1}^{N} W_{iA}W_{iB}}{\sum_{i=1}^{N}(W_{iA})^2 + \sum_{i=1}^{N}(W_{iB})^2 - \sum_{j=1}^{N}(w_{AB})}$	$S(A,B) = \frac{c}{a+b-c}$
Soergel	$D(A,B) = \frac{\lvert A \Delta B \rvert}{\max(\lvert A \rvert, \lvert B \rvert)}$	$(DA,B) = 1 - \frac{c}{a+b-c}$

Table 2.1: Table : Formulas for the various similarity and distance metrics

2.7.3 Distance phenomena in high-dimensional chemical descriptor spaces

Measuring the (dis)similarity of molecules is important for many cheminformatics applications like compound ranking, clustering, and property prediction,[30] In this work, we focus on real-valued vector representations of molecules (as opposed to the binary spaces of fingerprints). We demonstrate the influence which the choice of (dis)similarity measure can have on results, and provide recommendations for such choices. We review the mathematical concepts used to measure (dis)similarity in vector spaces, namely norms, metrics, inner products, and, similarity coefficients, as well as the relationships between them, employing (dis)similarity measures commonly used in cheminformatics as examples. In the context of this topic, "distance phenomena" likely refers to the exploration and analysis of distances or similarities between chemical compounds in high-dimensional descriptor spaces. This can involve measuring distances or dissimilarities between compounds based on their descriptor values and investigating how these distances relate to their chemical properties or activities.

Analyzing distance phenomena in high-dimensional chemical descriptor spaces can provide insights into the relationships between chemical compounds, aid in identifying similar or dissimilar compounds, and support various applications in drug discovery, chemical classification, or property prediction. We present several phenomena (empty space phenomenon, sphere volume related phenomena, distance concentration) in high-dimensional descriptor spaces which are not encountered in two and three dimensions. These phenomena are theoretically characterized and illustrated on both artificial and real (bioactivity) data.

2.8 Deep Learning

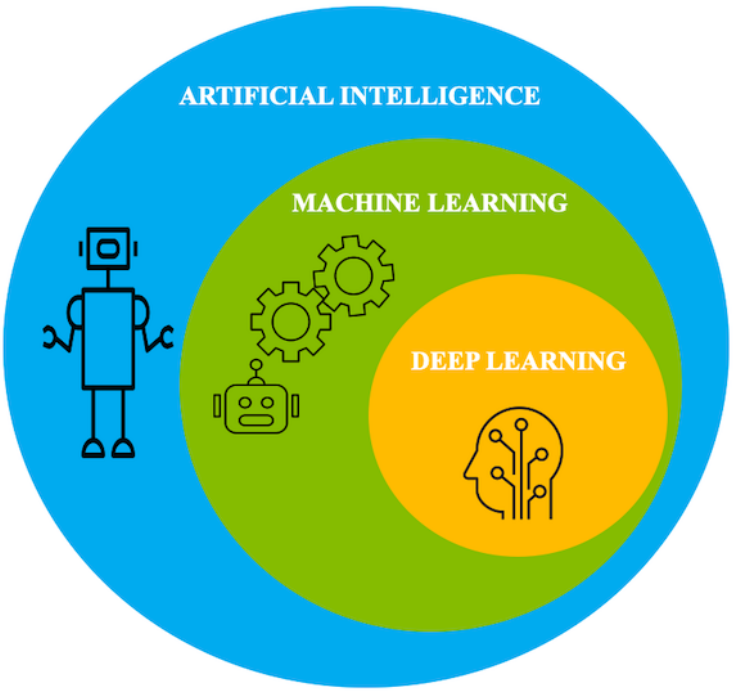


Figure 2.6: Represents The Relation Between AI, ML and Deep Learning.

2.8.1 Artificial Intelligence

Artificial Intelligence (AI) is a field of computer science that focuses on developing intelligent machines that can perform tasks that typically require human intelligence[31]. It encompasses various subfields, including machine learning, natural language processing, computer vision, robotics, and more. AI systems are designed to perceive their environment, reason about it, and make informed decisions or take actions to achieve specific goals[32].

2.8.2 Machine Learning

Machine Learning (ML) is a subset of artificial intelligence that focuses on the development of algorithms and models that enable computers or machines to learn and make predictions or decisions without being explicitly programmed. ML systems are designed to automatically analyze and interpret data, learn patterns or models from it, and use that knowledge to make informed decisions or predictions[33, 34].

Machine learning has found applications in various domains, including image and speech recognition, natural language processing, recommendation systems, fraud detection, and healthcare, among others[35].

2.8.3 Deep Learning

Deep learning is a subfield of machine learning that focuses on training artificial neural networks with multiple layers, also known as deep neural networks, to learn and make predictions or decisions from complex data. It is inspired by the structure and functioning of the human brain, specifically the interconnected network of neurons[31, 36].

Deep learning has gained significant attention and achieved remarkable success in various domains, including computer vision, natural language processing, speech recognition, and many others. It has revolutionized the field of AI by enabling models to automatically learn hierarchical representations of data through multiple layers of abstraction[32].

Key concepts and techniques in deep learning include:

Neural Networks

Deep learning models are typically built using artificial neural networks, which consist of interconnected layers of nodes (neurons). Each node performs a computation on its inputs and applies an activation function to produce an output[33, 37].

Deep Neural Networks

(DNNs): DNNs are neural networks with multiple hidden layers between the input and output layers. These layers allow the model to learn increasingly complex features and representations of the input data[17, 35].

Convolutional Neural Networks (CNNs)

CNNs are a specialized type of deep neural network commonly used for computer vision tasks. They exploit the spatial structure of the input data and use convolutional layers to automatically extract hierarchical features [38].

Recurrent Neural Networks (RNNs)

RNNs are designed to handle sequential data by maintaining an internal memory state. They can process input sequences of varying lengths and are commonly used in tasks such as natural language processing and speech recognition[39]

Generative Models

Deep learning also encompasses generative models, such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), which can generate new data samples based on learned patterns[40].

Deep learning has achieved remarkable success in a wide range of applications, including image and speech recognition, natural language understanding, autonomous vehicles, drug discovery, and many others. Its ability to automatically learn complex patterns and representations from data has made it a powerful tool in the field of artificial intelligence.
the table represent the popular model for molecular generation :

Name	Generative Model				Sampling Process		Training Process	
	VAE	GAN	RNN	Flow	One_shot	Iterative	Sequential	Parallel
JT-VAE	✓					✓		
RVAE	✓				✓			
GCPN		✓				✓	✓	
MRNN			✓			✓	✓	
GraphNVP				✓	✓			
GraphAF				✓		✓		✓

Table 2.2: molecular graph generation. The comparison of training is only conducted between autoregressive models.

JT-VAE : a short for Junction Tree Variational Autoencoder, is a generative model specifically designed to generate molecular graphs. It combines the concepts of junction trees and variational autoencoders to capture the complex structural patterns and properties of molecules.
RVAE : a short for Recurrent Variational Autoencoder, is a generative model that combines the power of recurrent neural networks (RNNs) and variational autoencoders (VAEs). It is designed to generate sequential data, such as text, music, or time series, while capturing the underlying patterns and dynamics of the data.
MRNN : a short for Multi-layer Recurrent Neural Network, is a generative model that utilizes recurrent neural networks (RNNs) with multiple layers to generate sequential data. It is specifically designed to capture the temporal dependencies and patterns present in the data and generate new sequences that resemble the training data.
GraphNVP : a short for Graph Neural Variational Autoencoder with Parametric Smoothing, is a generative model that combines the power of graph neural networks (GNNs) and variational autoencoders (VAEs). It is specifically designed to generate realistic and diverse graphs.

2.8.4 What is a Graph?

n computer science and mathematics, a graph is a data structure that represents a collection of interconnected nodes or vertices. It consists of a set of nodes and a set of edges that connect pairs of nodes. A graph is G = (V, E), where V is the set of vertices and E is the set of edges connecting them.

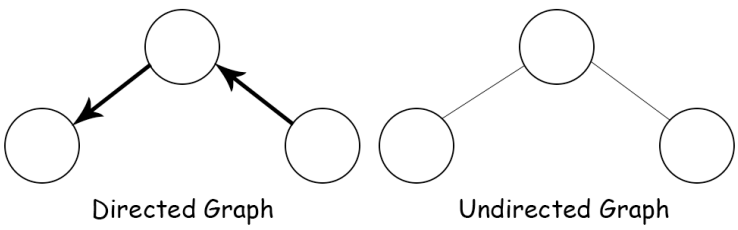


Figure 2.7: Directed Graph and Undirected Graph

In the context of deep learning, a graph refers to a data structure that represents the relationships between different elements or entities. Specifically, a graph is composed of nodes and edges. Nodes represent individual entities, while edges represent the connections or relationships between those entities.[40]

In a computational graph, nodes represent mathematical operations or transformations, such as matrix multiplications, convolutions, activations functions, etc. The edges represent the flow of data, indicating the inputs and outputs of each operation. The graph shows how data is transformed and propagated through the network during the forward pass, as well as how gradients flow during the backward pass for training via techniques like backpropagation.[36]

2.8.5 GCPN

A Graph Convolutional Policy Network (GCPN) is a type of neural network architecture that is designed for learning policies for decision-making in graph-structured data. It is a combination of graph convolutional networks (GCNs) and reinforcement learning (RL) methods.

GCPNs are typically used in applications such as recommendation systems, drug discovery, and social network analysis. The network takes as input a graph representation of the problem domain, and learns to output a policy that optimizes some task-specific objective function. The policy is learned through a combination of supervised and reinforcement learning techniques[8].

The graph convolutional layers in GCPNs allow the network to effectively capture the local structure of the graph, while the RL techniques enable the network to learn a policy that maximizes some reward signal. This makes GCPNs well-suited for tasks that involve complex decision-making in graph-structured data[13].

In the context of de novo molecular design, GCPN uses GCNs to encode molecular graphs into continuous representations that can be used to generate new molecules. GCPN then employs a policy network, which is a type of reinforcement learning algorithm, to generate new molecules that are optimized for a specific set of desired properties[18].

The policy network learns to sample molecules from the continuous representations generated by the GCN and to generate molecules with high scores according to a reward function. The reward function is based on the desired properties of the molecules, such as their predicted binding affinity to a target protein or their predicted solubility[13, 18]. Overall, GCPN represents a promising approach for de novo molecular design, leveraging the power of GCNs and reinforcement learning to generate novel molecules with desired properties.

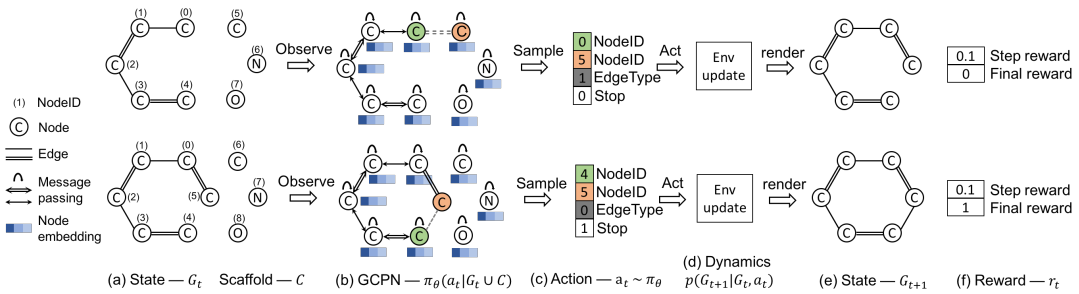


Figure 2.8: Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation

2.8.6 Generative Adversarial Networks for molecular simulation

GANs, short for Generative Adversarial Networks, are a type of machine learning model consisting of two neural networks: the generator and the discriminator. These networks work together in a competitive manner to generate new data samples resembling a given training dataset.

In the context of molecular generation, GANs have been employed to produce novel molecules with desired properties, contributing to the field known as "generative chemistry" or "molecular design." In this application, the generator network takes random noise as input and strives to generate molecular structures, often represented as sequences of atoms and bonds. On the other hand, the discriminator network evaluates the generated molecules based on criteria such as their similarity to known molecules or the likelihood of possessing desired properties.

Chemical compounds can be represented as graphs, with nodes and edges representing different atoms and their bonds. If the resulting structure is chemically stable and has a high affinity for receptor binding sites, it can be considered a valid drug molecule. The generator and discriminator are trained on example drug molecules until the synthetic chemical distribution closely resembles the

real chemical space. Fréchet distance and RDKit are used to assess the quality of GAN output (for chemical properties). [41]

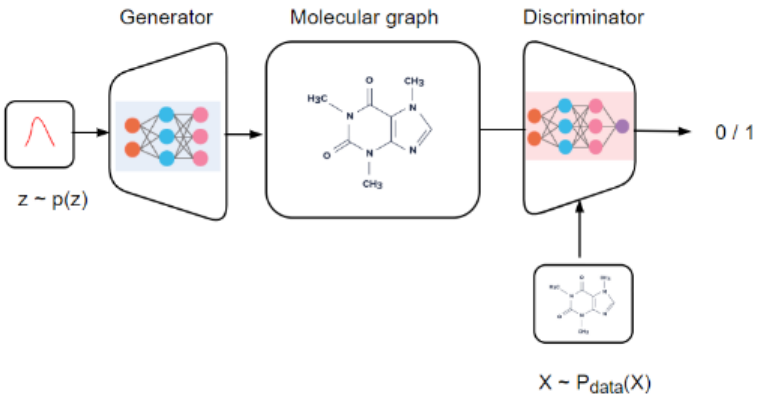


Figure 2.9: GAN architecture for molecule simulations

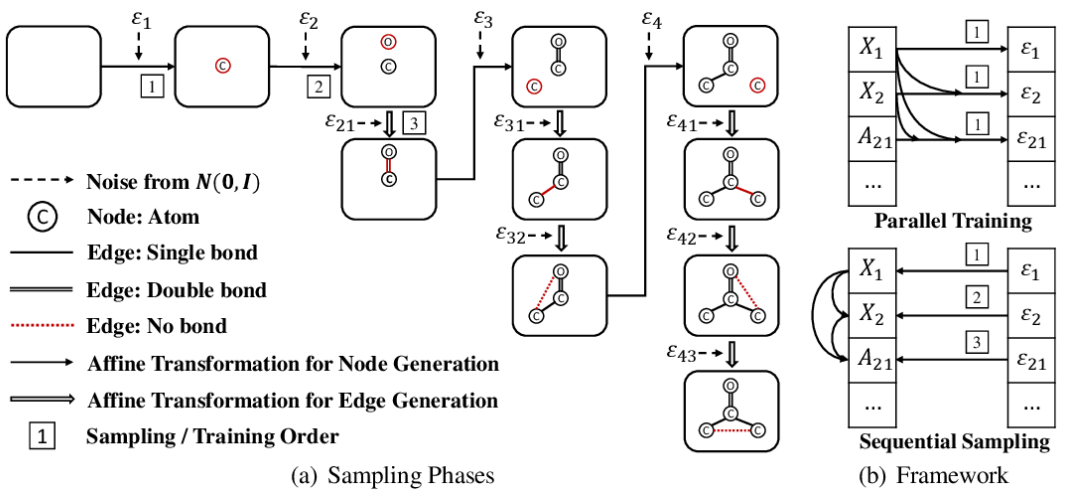
During training, the generator aims to generate molecules that deceive the discriminator into classifying them as authentic or high-quality samples. Through an iterative process, GANs refine the generator's performance, enabling the generation of novel molecules with specific characteristics like drug-likeness, bioactivity, or stability.

The use of GANs in molecular generation holds promise for expediting the drug discovery process by generating novel compounds that can be further explored and tested for specific therapeutic targets. Additionally, it aids in the exploration of chemical space, producing molecules with unique properties applicable in areas such as materials science, energy storage, and other domains.

2.8.7 GraphAF

GraphAF (Graph Autoregressive Flow) is a deep learning-based approach for de novo molecular design that leverages attention mechanisms to model the interactions between atoms and bonds in a molecule. GraphAF operates on molecular graphs, which represent molecules as graphs where atoms are nodes and bonds are edges.

GraphAF incorporates attention mechanisms to learn the relative importance of different atoms and bonds in a molecule, enabling it to model the complex interactions that influence a molecule's properties. GraphAF uses a multi-head attention mechanism, which allows it to capture different aspects of the molecular structure in parallel , In the context of de novo molecular design, GraphAF generates new molecules by iteratively adding atoms and bonds to an initially empty molecular graph. At each iteration, GraphAF predicts the most promising atom and bond to add to the graph based on the attention-based representations of the current molecular structure [19].



(a) Sampling Phases (b) Framework

Figure 2.10: Overview of the proposed GraphAF model. (a) Illustration of the generative procedure. New nodes or edges are marked in red. Starting from an empty graph and iteratively sample random variables to map them to atom/bond features. The numbered first three steps correspond to the maps in the bottom figure of Fig. 1(b). (b) Computation graph of GraphAF. The left side are the nodes and edges and the right are latent variables

GraphAF has been shown to be effective in generating novel molecules with desired properties, including drug-like molecules and materials with specific properties. Compared to other deep learning-based approaches for de novo molecular design, GraphAF has demonstrated superior performance in terms of generating molecules with improved properties [20].

Overall, GraphAF represents a promising approach for de novo molecular design that leverages attention mechanisms to model the complex interactions between atoms and bonds in a molecule.

2.8.8 autoregressive flow generation

"autoregressive flow" refers to a specific type of generative model that is used to generate molecules. Autoregressive flows are a type of deep learning model that operate by transforming a simple probability distribution into a more complex one.covery of new compounds.properties. In autoregressive flow models, the generation process proceeds sequentially, with each atom or group being generated conditioned on the previously generated atoms or groups. This sequential generation is typically done using a recurrent neural network (RNN) or a transformer-based architecture. Autoregressive flow models have been used in molecular generation tasks to generate novel molecules with desired properties. By sampling from the learned conditional distribution at each step, the model can generate a sequence of atoms or groups, resulting in a complete molecular structure. Autoregressive flows have the advantage of being able to model complex dependencies between atoms or groups in a molecule. However, they suffer from the drawback of sequential generation, which can be slow and limiting for generating long and complex molecules. Nevertheless, they have been successfully applied in molecular generation tasks and have contributed to the exploration of chemical space and the discovery of new compounds.

2.8.9 RGCN

RGCN stands for Relational Graph Convolutional Networks. It is a type of neural network architecture designed for learning on graph-structured data. RGCNs are specifically tailored for graphs that have labeled edges, representing different types of relationships between nodes. In traditional graph convolutional networks (GCNs), the convolutional operation is applied uniformly to all neighbors of a node in the graph. However, in RGCNs, the convolutional operation is extended to handle the heterogeneity of edges in a graph. It allows each edge type to have its own learnable weight matrix, enabling the network to learn different patterns and relationships for each edge type.

RGCNs utilize a message-passing scheme, where information is propagated between nodes in the graph through the labeled edges. At each layer, the node representations are updated by aggregating information from their neighboring nodes, considering the specific edge types connecting them. This allows the network to capture relational dependencies and incorporate them into the node representations.

The RGCN architecture has been widely used in various applications, including knowledge graph completion, social network analysis, recommendation systems, and drug discovery. By leveraging the explicit modeling of edge types, RGCNs can effectively handle complex relationships and improve the performance of tasks involving graph-structured data.

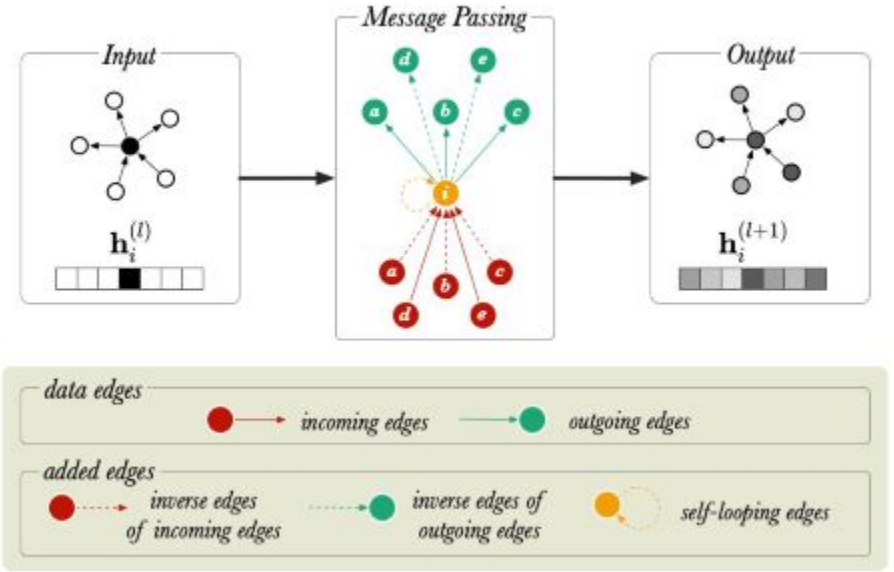


Figure 2.11: A schematic diagram of message passing in a directed graph with 6 nodes

Chapter 3

Research Methodology

3.1 Intoduction

The research methodology for the study of de novo molecular design based on Graph Convolutional Policy Network (GCPN) and Graph Autoregressive Flow (GraphAF) involves several steps. The goal of this chapter is to search for the best performing model, showing the usefulness of Graph Convolutional Policy Network (GCPN) and Graph Autoregressive Flow (GraphAF) and the advantages of representing data using graphs, The methodology employed will provide a systematic framework of procedures and principles to achieve the objective of this study.

3.2 Research Framework

The research framework for this study is a supervised learning approach, where the Graph Convolutional Policy Network (GCPN) and Graph Autoregressive Flow (GraphAF) models will be trained on preprocessed molecular graph datasets with known properties[27],The aim is to develop advanced deep learning-based approaches for de novo molecular design, specifically the generation of new molecules with desired properties. The models will be evaluated based on their ability to generate such molecules using metrics such as accuracy, precision, recall, and F1-score[10, 28]. Additionally, the models' performance will be benchmarked against other state-of-the-art deep learning models for de novo molecular design using publicly available datasets and metrics such as the number of generated molecules and their quality. The analysis of the results will identify the strengths and weaknesses of each model, contributing to the development of better deep learning-based approaches for de novo molecular design[42, 43].

3.3 Dataset

The dataset used in this study will be the publicly available molecular databases MUV (Maximum Unbiased Validation) and BACE (Beta-Secretase 1) from the Cheminformatics and Computational Chemical Biology Resource (http://muv.oscer.ou.edu) and the National Center for Biotechnology Information PubChem database (https://pubchem.ncbi.nlm.nih.gov/), respectively. These databases contain a large number of diverse molecules with known properties relevant to drug discovery. The preprocessed data will be split into training, validation, and test sets to train and evaluate the Graph Convolutional Policy Network (GCPN) and Graph Autoregressive Flow (GraphAF) models for de novo molecular design.

3.3.1 MUV

The MUV (Maximum Unbiased Validation) PubChem BioAssay dataset is a widely used benchmark dataset in the field of machine learning for drug discovery. It is a subset of the larger MUV dataset, which was developed by Rohrer and Baumann in 2009 for benchmarking machine learning models for predicting the activity of molecules in drug discovery.
The MUV PubChem BioAssay dataset consists of 17 bioassays (i.e., biological assays) from PubChem, which is a public database of chemical substances and their biological activities [44].

Each bioassay in the MUV PubChem BioAssay dataset is designed to test the activity of compounds against a specific biological target, such as a protein or enzyme. The compounds in the dataset are labeled as either active or inactive based on their ability to bind to or modulate the target [45].

The MUV PubChem BioAssay dataset is commonly used as a benchmark for evaluating the performance of machine learning models for predicting the activity of compounds in drug discovery, as it represents a diverse set of bioassays and compounds that are challenging for predictive models to accurately classify as active or inactive. Several studies have used the MUV PubChem BioAssay dataset to evaluate the performance of various machine learning algorithms, including deep neural networks, random forests, support vector machines, and others [11, 46]. why The MUV (Maximum Unbiased Validation) hard to get better result predection in molecular generation ??

The Maximum Unbiased Validation (MUV) is a benchmark dataset used to evaluate the performance of machine learning models for molecular generation tasks. It consists of a set of 17 diverse targets, each with 93 active compounds and 802 decoys.

One of the reasons why it is hard to get better predictive results for molecular generation using the MUV dataset is due to its high level of complexity and diversity. The MUV dataset is designed to challenge machine learning models to accurately predict the activity of diverse chemical compounds across a range of targets. The complexity of the dataset makes it difficult for machine learning models to generalize well and accurately predict the activity of novel compounds.

Furthermore, the MUV dataset is designed to test for the presence of biases in machine learning models. This means that the dataset is designed to be as unbiased as possible, which can make it difficult for machine learning models to find patterns in the data that can be used to accurately predict the activity of novel compounds.

Overall, the MUV dataset is a challenging benchmark for machine learning models, and achieving high predictive performance on this dataset requires the development of novel and innovative machine learning techniques.

MUV class

The MUV dataset consists of 17 challenging target protein families, and each family represents a distinct classification task. The compounds in the dataset are labeled as either active or inactive based on their binding affinity or biological activity against the respective targets. The dataset is designed to be unbiased, ensuring that the compounds are diverse and representative of real-world chemical space.

3.3.2 BACE

The BACE (Beta-secretase) dataset is a widely used benchmark dataset in drug discovery for machine learning models that aim to predict the inhibition of beta-secretase, an enzyme that is involved in the production of beta-amyloid peptides in Alzheimer's disease [47] ,The dataset consists of a set of compounds with known inhibitory activity against BACE and a set of compounds with no known activity. The BACE dataset has been used in various studies to develop and improve machine learning models for predicting the activity of compounds against other drug targets as well [48]

The BACE dataset is useful for evaluating the performance of machine learning models for predicting the inhibitory activity of compounds against BACE, as it represents a challenging classification problem due to the complex nature of the molecular interactions involved in BACE inhibition. The dataset has been used to develop various machine learning models, including Support Vector Machines (SVMs), Random Forests, and Deep Neural Networks (DNNs)[28, 49, 50].

The BACE dataset is publicly available through the PubChem database . The dataset is accompanied by detailed instructions for preprocessing the data, including data filtering and standardization, as well as guidelines for training and evaluating machine learning models on the dataset [51, 52].

3.4 Steps and Design

3.4.1 Introduction

The creation of the model involved a systematic and carefully planned process, aiming to develop an effective solution for analyzing activities within a specific class. This section provides an overview of the steps and design considerations undertaken during the model creation journey. starting by splitting the dataset depending on activity structure, to choosing the best activity class(MUV or BACE), then reading the data and getting familiar with it. Then present a graph based model, and explain the important steps and modifications.

3.4.2 Dataset Splitting

The first step in creating the model involved splitting the dataset based on the activity structure. This means categorizing the data into different groups based on the type or nature of the activity. This splitting process helps in better understanding and handling of the data during the model creation.

3.4.3 Choosing the Activity Class

After splitting the dataset, the next step was to choose the best activity class for the model. Two potential classes were considered: MUV (Multi-User Virtual) and BASE. The selection was based on various factors, such as the project requirements, available resources, and the specific application of the model. After careful evaluation, one class was chosen for further model development.

3.4.4 Data Reading and Familiarization

Once the activity class was determined, the dataset associated with that class was read and thoroughly analyzed. This step involved understanding the data format, exploring the data distribution, identifying any missing values or outliers, and gaining a comprehensive understanding of the dataset's characteristics. By using Pandas, we creates efficient data frame objects with default and customized indexing and provides a mechanism to load data objects for our data formats:

Algorithm 1 Loading and creating data frame in Pandas Python

```
path = '/dataset/MUV.csv'
df = pd.read_()csv
print(df)
```

MUV-466	MUV-548	MUV-600	MUV-644	MUV-652	MUV-689	MUV-692	MUV-712	MUV-713	MUV-733	MUV-737	MUV-810	MUV-832	MUV-846	MUV-852	MUV-858	MUV-859	mol_id	smiles
							0				0						CID299967	Cc1cccc(N2CCN(C(=O)C34CC5CC(CC(C5)(
0	0			0	0	0				0		0			0	0	CID299967	Cn1ccnc1SCC(=O)Nc1ccc(Oc2ccccc2)cc1
		0														0	CID299967	COc1cc2c(cc1NC(=O)CN1C(=O)NC3(CCc4
	0	0			0										0		CID539000	O=C1/C(=C/NC2CCS(=O)(=O)C2)c2ccccc2
0				0		0	0										CID299967	NC(=O)NC(Cc1ccccc1)C(=O)O
						0		0	0						0		CID299967	CCn1c(CSc2nccn2C)nc2cc(C(=O)O)ccc21
		0	0				0			0		0			0		CID539000	CCc1cccc2c1NC(=O)C21C2C(=O)N(Cc3cc
							0			0							CID539000	COc1ccc(C2C(C(=O)NCc3ccccc3)=C(C)N=
	0			0		0			0								CID299967	CCCc1cc(=O)nc(SCC(=O)N(CC(C)C)C2CCS(

Figure 3.1: Displaying loaded dataset

3.4.5 Graph-Based Model

A graph-based model architecture was chosen for this project. Graph-based models are effective for representing and analyzing structured data, such as activities. In this design, activities are represented as nodes, and the relationships between activities are represented as edges in a graph. Using a graph-based

model architecture for representing and analyzing activities is a powerful approach, particularly when dealing with structured data. By representing activities as nodes and capturing relationships between activities as edges, you can effectively model and analyze complex activity networks.

Graph-based models have several advantages in this context. They can capture both the inherent structure and the dependencies present in the data. This structure allows for efficient traversal and exploration of the activity graph, enabling various analyses and computations.

3.5 Adversarial training

Incorporating prior knowledge specified by a dataset of example molecules is crucial for molecule generation. For example, a drug molecule is usually relatively stable in physiological conditions, non toxic, and possesses certain physiochemical properties Lin403 . Although it is possible to hand code the rules or train a predictor for one of the properties, precisely representing the combination of these properties is extremely challenging. Adversarial training addresses the challenge through a learnable discriminator adversarially trained with a generator goodfellow2014generative . After the training converges, the discriminator implicitly incorporates the information of a given dataset and guides the training of the generator.

We evaluate GCPN in two different molecule generation tasks, namely MoleculeNet's MUV (Molecules with Unwanted Effects) and BACE (Beta-secretase 1) datasets, which are relevant to drug discovery. We use these datasets to provide GCPN with example molecules and train the policy network to generate molecules with high property score, molecules with a pre-specified range of target property score, or molecules containing a specific substructure while having high property score. In both tasks, GCPN achieves state-of-the-art results. GCPN generates molecules with property scores 20% higher than the best baseline method for the MUV dataset, and 40% higher for the BACE dataset. In the constrained optimization setting, GCPN outperforms the baseline models by 10% and 50% on average for the MUV and BACE datasets, respectively.

3.5.1 Hyperparameters

We list the key hyperparameters used for Molecular generation pretraining with the GCPN AND graph_af as inputs, respectively.

Input	Heyperparametre	Value
GCPN , GraphAF	epochs	{10}
	batch_size	{128}
	objective function	{nll}
	activation function	{relu}
	max_edge_unroll	{12}
	max_node	{38}
	learning rate lr	{1e-3}
	log_interval	{10}
	task	{17 class of muv datasets}

Table 3.1: Heyperparametre specification for muv

```
1 dataset.tasks
```

```
['MUV-466',
 'MUV-548',
 'MUV-600',
 'MUV-644',
 'MUV-652',
 'MUV-689',
 'MUV-692',
 'MUV-712',
 'MUV-713',
 'MUV-733',
 'MUV-737',
 'MUV-810',
 'MUV-832',
 'MUV-846',
 'MUV-852',
 'MUV-858',
 'MUV-859']
```

Figure 3.2: MUV classes

Input	Heyperparametre	Value
GCPN , GraphAF	epochs	{100}
	batch_size	{64}
	objective function	{nll}
	activation function	{relu}
	max_edge_unroll	{12}
	max_node	{38}
	learning rate lr	{1e-3}
	log_interval	{10}
	task	{1 class of BACE datasets}

Table 3.2: Heyperparametre specification for BACE

3.6 Molecular generation

Molecular generation, also known as molecular design or molecule generation, refers to the process of generating new chemical structures or molecules with desired properties or characteristics. It involves employing computational methods, machine learning, and/or artificial intelligence techniques to explore chemical space and propose novel molecules. The problem is challenging since it requires not only generating chemically valid molecular structures but also optimizing their chemical properties in the meantime. We first pretrain both models on MUV and BACE dataset. Starting from the pretrained checkpoint, we finetune both models with reinforcement learning to optimize two properties (i.e., QED and penalized logP score) of generated molecules.

The field of molecular generation has gained significant interest in drug discovery, materials science, and other areas where the discovery of new molecules is crucial. It offers a way to expedite the process of designing and discovering compounds with specific properties, such as drug candidates with improved efficacy and reduced side effects, or materials with desired chemical and physical properties.

3.6.1 Define the Model: GCPN

The model consists of two parts, a graph representation model and a graph generative module. We define a Relational Graph Convolutional Networks (RGCN) as our representation model. We use the module GCPNGeneration as the training task for GCPN.

Algorithm 2 Define the Model: GCPN

```
import torch
from torchdrug import datasets, core, models, tasks, utils
from torch import optim
dataset1 = datasets.BACE("molecule-datasets", ekulize=True, node_feature="symbol")
dataset1.transform = None
model = models.RGCN(input_dim=dataset1.node_feature_dim, num_relation=dataset1.num_bond_type, hidden_dims=[256, 256, 256, 256], batch_norm=False)
task = tasks.GCPNGeneration(model, aset1.atom_types, max_edge_unroll=12, max_node=38, criterion="nll")
optimizer = optim.Adam(task.parameters(), lr = 1e-3)
```

3.6.2 Pretraining and Generation

Now we can train our model. We setup an optimizer for our model, and put everything together into an Engine instance. Here we only train the model for 100 epochs, and then save the pretrained model into a directory During the

Algorithm 3 Pretraining and Generation: GCPN

```
solver = core.Engine(task, dataset1, None, None, optimizer, gpus=(0,), batch_size=64, log_interval=10)
solver.train(num_epoch=100)
solver.save("gcpn_BACE_1epoch.pkl")
```

pretraining procedure, we may obtain some logs as follows, which report the accuracy of action predictions. After the model is pretrained, we can load the parameters from the checkpoint as follows. Let's generate some small molecules from the pretrained GCPN model.

Algorithm 4 generate small molecules from the pretrained GCPN model

```
list_smiles = results.to_smiles()
print(list_smiles)
print(len(list_smiles))
```

The results are as follows:

CC1(C)CC2=C(F)C(C#N)=C(Cl)C=C2C(N)=N1

CC12CC(C)(N=C(N)O1)C1=CC(Cl)=CC(=C1)NC(=O)C=CS2

COC1=CC(C2=C3N=C(N)N4N=C(C)C(=C34)C=C2)=CC=C1

COC1=CC=C(C2(NCC3=CSC=C3)CCNC(=O)CC2)C=C1

CN1C(=O)C2(CCCC3(C)CCC4=C(C=CC=C4)CC2C3)N=C1N

CCCCC1(C2=CC=C(OC(F)F)C=C2)N=C(N)C2=C1C=CC=CN=C2

Table 3.3: smiles

Let's visualize some generated molecules.

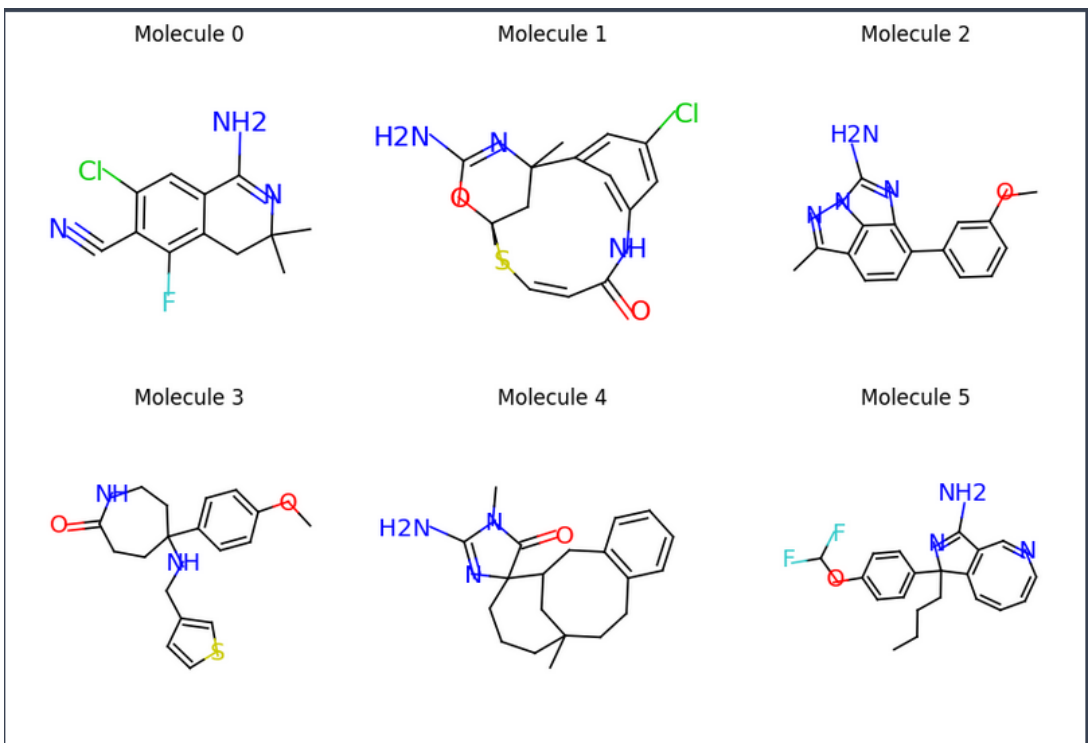


Figure 3.3: some generated molecules with GCPN

The results are as follows:

CC1(C)CC2=C(F)C(C#N)=C(Cl)C=C2C(N)=N1	0.519198
CC12CC(C)(N=C(N)O1)C1=CC(Cl)=CC(=C1)NC(=O)C=CS2	0.577453
COC1=CC(C2=C3N=C(N)N4N=C(C)C(=C34)C=C2)=CC=C1	0.545280
COC1=CC=C(C2(NCC3=CSC=C3)CCNC(=O)CC2)C=C1	0.518038
CN1C(=O)C2(CCCC3(C)CCC4=C(C=CC=C4)CC2C3)N=C1N	0.529632

Table 3.4: results of molecules using GCPN model

3.6.3 Define the Model: GraphAF

The model consists of two parts, a graph representation model and a graph generative module. We define a Relational Graph Convolutional Networks (RGCN) as our representation model. We use the module AutoregressiveGeneration as the training task for GraphAF. The task consists of a node flow model and an edge flow model, which define invertible mapping between node / edge types and noise distributions.

Algorithm 5 Define the Model: GraphAF

```
from torchdrug.layers import distribution
dataset1.transform = None
model = models.RGCN(input_dim=dataset1.num_atom_type,num_relation=dataset1.num_bond_type,
hidden_dims=[256, 256, 256], batch_norm=True)
num_atom_type ← dataset1.num_atom_type

num_bond_type ← dataset1.num_bond_type + 1
node_prior ← distribution.IndependentGaussian(torch.zeros(num_atom_type),
torch.ones(num_atom_type))
edge_prior ← distribution.IndependentGaussian(torch.zeros(num_bond_type),
torch.ones(num_bond_type))
node_flow ← models.GraphAF(model, node_prior, num_layer=12)
edge_flow ← models.GraphAF(model, edge_prior, use_edge=True, num_layer=12)
task ← tasks.AutoregressiveGeneration(node_flow, edge_flow, max_node=38,
max_edge_unroll=12, criterion="nll")
```

this is some molecular generated :

CC1(C)CC2=C(F)C(C#N)=C(Cl)C=C2C(N)=N1
CC12CC(C)(N=C(N)O1)C1=CC(Cl)=CC(=C1)NC(=O)C=CS2
COC1=CC(C2=C3N=C(N)N4N=C(C)C(=C34)C=C2)=CC=C1
COC1=CC=C(C2(NCC3=CSC=C3)CCNC(=O)CC2)C=C1
CN1C(=O)C2(CCCC3(C)CCC4=C(C=CC=C4)CC2C3)N=C1N
CCCCC1(C2=CC=C(OC(F)F)C=C2)N=C(N)C2=C1C=CC=CN=C2

Table 3.5: some smiles

Let's visualize some generated molecules.

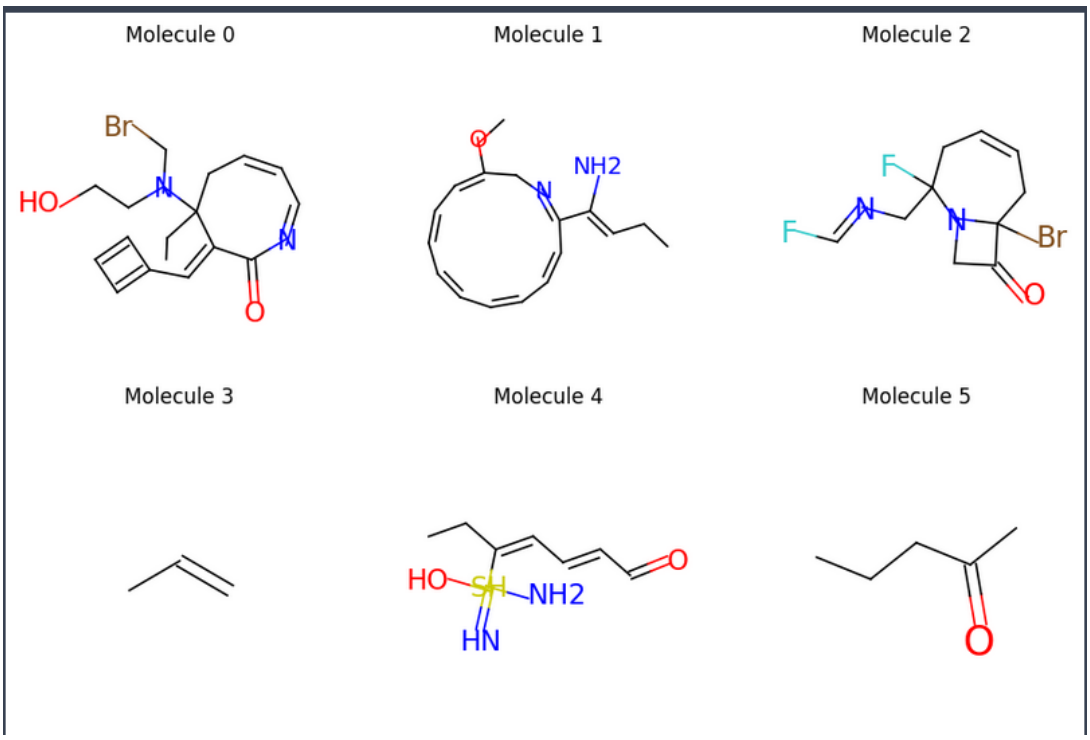


Figure 3.4: some generated molecules with GraphAF

The results of similarity are as follows:

CC1(C)CC2=C(F)C(C#N)=C(Cl)C=C2C(N)=N1	0.472703
CC12CC(C)(N=C(N)O1)C1=CC(Cl)=CC(=C1)NC(=O)C=CS2	0.262699
COC1=CC(C2=C3N=C(N)N4N=C(C)C(=C34)C=C2)=CC=C1	0.469832
COC1=CC=C(C2(NCC3=CSC=C3)CCNC(=O)CC2)C=C1	0.008639
CN1C(=O)C2(CCCC3(C)CCC4=C(C=CC=C4)CC2C3)N=C1N	0.209161

Table 3.6: results of molecules with GraphAF model

3.7 Performance Evaluation

The performance of molecular design based on GCPN can be evaluated using various metrics, such as accuracy, diversity, novelty, and similarity.
The evaluation can be performed using cross-validation and holdout validation. Cross-validation involves dividing the dataset into k-folds, training the GCPN model on k-1 folds, and evaluating it on the remaining fold. Hold-out validation involves dividing the dataset into training and test sets, training the GCPN model on the training set, and evaluating it on the test set.
The evaluation should be repeated multiple times to ensure that the results are consistent.

3.8 similarity measurement

The similarity of molecules refers to the measure of how closely related or similar two molecules are in terms of their structural or chemical properties. Assessing molecular similarity is crucial in various fields of chemistry, drug discovery, and computational chemistry. There are several methods and metrics commonly used to quantify molecular similarity : we used tanimoto similarity

3.8.1 definition

Tanimoto similarity is a widely used metric in cheminformatics. It measures the overlap between molecular fingerprints, which encode the presence or absence of specific chemical features. Tanimoto similarity ranges from 0 to 1, with 0 indicating no similarity and 1 representing an exact match

3.8.2 why we used this similarity

Tanimoto similarity is commonly used as a measurement for molecular generation because it provides a convenient and effective way to assess the similarity between molecules. Here are some reasons why Tanimoto similarity is often considered a suitable metric for molecular generation.
The choice of the most appropriate metric should consider the specific objectives of the molecular generation task and the properties that need to be captured or optimized [53].

3.9 Benchmarking

The performance of molecular design based on GCPN can be benchmarked against other machine learning models, such as generative adversarial networks (GANs) and variational autoencoders (VAEs). The benchmarking can be done using publicly available datasets and standard evaluation metrics. The comparison should consider the trade-offs between accuracy, diversity, novelty, and similarity.

3.10 Summary

Molecular design based on GCPN is a promising approach for generating novel molecules with desired properties. The approach has potential applications in drug discovery, materials science, and other fields that require the design of new molecules. However, there are still limitations and challenges that need to be addressed, such as the need for larger and more diverse datasets, better optimization methods, and interpretability of the generated molecules. The summary should highlight the main findings of the research, its contributions, and its implications for future work. It should also discuss the limitations of the study and the potential avenues for improvement.

Chapter 4

Results and Discussion

4.1 Introduction

the results section presents the outcomes of the molecular generation experiments and provides a quantitative and qualitative analysis of the generated molecules.
We trained the GCPN model and Graph_afon a dataset of 93087 drug-like molecules extracted from the MUV dataset and 1513 from the BACE datest. After convergence, the model was able to generate diverse molecules with high chemical validity. We generated a set of 170 molecules from each dataset using the trained model and evaluated their properties.

Quantitative analysis revealed that the generated molecules exhibited a wide range of structural diversity, as evidenced by the distribution of Tanimoto similarity scores compared to the training set. On average, the generated molecules achieved a Tanimoto similarity score of 0.54 for GCPN model and less than 0.45 for GRAPH_AF model, indicating a reasonable similarity to the training molecules.

4.2 Development Environment and Used Materials

In this section, we will give an overview about the environment and different tools used In order to carry out the tests and compare the obtained results.

4.2.1 Programming language

Python : Python is a widely used high-level programming language that emphasizes code readability and simplicity. It was created by Guido van Rossum and first released in 1991. Python has gained popularity due to its ease of use, versatility, and a large community of developers contributing to its ecosystem. Python is known for its clean and expressive syntax, which makes it easier to write and understand code. It uses indentation and whitespace to define code blocks, rather than relying on curly braces or keywords, which enhances readability. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

4.2.2 Environment

Colab : Google Colab, short for Google Colaboratory, is a cloud-based platform provided by Google that offers a free environment for running and developing machine learning, data analysis, and research projects. It allows users to create and execute Python code through a web browser without the need for setting up local development environments or requiring high-end hardware. Google Colab has gained popularity among data scientists, machine learning practitioners, and researchers due to its convenience, accessibility, and the ability to leverage powerful computing resources without significant setup or cost. It provides an interactive and collaborative environment for developing and running Python code, making it particularly useful for educational purposes, prototyping models, and sharing research findings.

Anaconda : Anaconda is a popular open-source distribution platform and package manager for Python and R programming languages. It simplifies the process of setting up and managing software environments, especially for data science, machine learning, and scientific computing projects. Anaconda provides a comprehensive collection of pre-compiled packages, libraries, and tools that are commonly used in data analysis and scientific computing workflows.

4.2.3 Libraries

PyTorch : PyTorch is a popular open-source machine learning framework that provides a platform for building and training neural networks. It was developed by Facebook's AI Research (FAIR) lab and is widely used by researchers and practitioners for deep learning tasks. PyTorch emphasizes flexibility, dynamic computation graphs, and a Pythonic programming interface, making it popular among the deep learning community

RDKit : RDKit is an open-source software toolkit for cheminformatics, which stands for chemical informatics. It provides a wide range of functions and algorithms for working with chemical structures, molecules, and molecular data. RDKit is written in C++ and has Python bindings, making it accessible and usable within the Python programming language.

Pandas : Pandas is a popular open-source data manipulation and analysis library for Python. It provides a high-level, easy-to-use interface and powerful data structures for efficiently working with structured data. Pandas is widely used in data science, data analysis, and machine learning tasks due to its versatility and functionality.

NumPy : NumPy (Numerical Python) is a powerful open-source library for numerical computation in Python. It provides a multidimensional array object, along with a collection of functions and tools for working with arrays and performing mathematical operations efficiently. NumPy is widely used in scientific computing, data analysis, and machine learning due to its speed, convenience, and extensive mathematical functionality.

CUDA : CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model developed by NVIDIA. It allows developers to harness the computational power of NVIDIA GPUs (Graphics Processing Units) to accelerate computationally intensive tasks. CUDA provides a programming interface and a set of tools that enable developers to write GPU-accelerated applications using popular programming languages such as C, C++, and Python

4.3 Experiment Design

The GCPN and GraphAF models will be implemented and trained using the training and validation sets. To demonstrate the effectiveness of goal-directed search for molecules with desired properties, we compare our method with state-of-the-art molecule generation algorithms in the following tasks.

4.3.1 Property Optimization

The task is to generate novel molecules whose specified molecular properties are optimized. This can be useful in many applications such as drug discovery and materials science, where the goal is to identify molecules with highly optimized properties of interest.Researchers in this field have been utilizing generative models like variational autoencoders (VAEs) and generative adversarial networks (GANs) to generate molecules tailored to specific properties. VAEs employ an encoder-decoder architecture to learn a latent representation of molecules, enabling the generation of new compounds by sampling from the learned latent space [54].

4.3.2 Property Targeting

The task is to generate novel molecules whose specified molecular properties are as close to the target scores as possible. This is crucial in generating virtual libraries of molecules with properties that are generally suitable for a desired application. For example, a virtual molecule library for drug discovery should have high drug-likeness and synthesizability.The research in this domain revolves around the utilization of reinforcement learning and evolutionary algorithms to generate molecules tailored to specific properties. Reinforcement learning algorithms, inspired by the principles of trial and error learning, are employed to optimize the generation process by maximizing a reward signal. Evolutionary algorithms, on the other hand, simulate natural selection and genetic variation to iteratively improve the quality of generated molecules over generations [11].

4.3.3 Constrained Property Optimization

The task is to generate novel molecules whose specified molecular properties are optimized, while also containing a specified molecular substructure. This can be useful in lead optimization problems in drug discovery and materials science, where we want to make modifications to a promising lead molecule and improve its properties. The research in this field involves harnessing the power of graph-based neural networks to generate molecules with specific substructures. Graph-based neural networks are designed to process and analyze molecular graphs, where atoms and bonds are represented as nodes and edges, respectively.

These models learn the structural patterns and relationships within molecules to generate new compounds with desired substructures [55].

4.4 FUNCTIONS TO CALCULATE THE SIMILARITY

4.4.1 Calculate in MUV

To calculate the similarity between generated molecules and molecules in the MUV dataset, you can use a variety of similarity metrics such as Tanimoto similarity or cosine similarity. These metrics compare the structural features of molecules, such as the presence of certain functional groups or the arrangement of atoms in the molecule [56, 57], There are several libraries available in Python for generating molecular structures and calculating their similarity, such as RDKit and Open Babel. These libraries provide a range of functions for molecular generation and similarity calculation, and can be used in conjunction with VAEs or GANs to generate and evaluate new molecules [58].

Algorithm 6 finger prints extraction from smiles

```
1: function FPS_EXTRACT(df)                    ▷ df :dataframe contain smiles of molecular
2:     ms ← list_vide[..]
3:     for i in length(df) do
4:         ms.add(Chem.MolFromSmiles(df['smiles'] [i])
5:     end for
6:     fps ← [Chem.RDKFingerprint(x) for x in ms]
7:     return fps
8: end function
```

One approach to generating new molecules using GCNs and calculating similarity on the MUV dataset involves training the GCN on the subset of the MUV dataset that corresponds to a specific molecular property or activity, such as kinase inhibition or estrogen receptor binding. The trained GCN can then be used to generate new molecules that are predicted to be active for the target property, and similarity to existing molecules in the MUV dataset can be calculated using molecular fingerprints or other similarity metrics [27].

Algorithm 7 calculat_similarity between MUV_classes ,BACE_class and the generate molecular

```
function CALCULAT_SIMILARITY(lst1,lst2)        ▷ lst1,lst2 :list contain smiles of molecular
    list_similarity ← list_vide(..)
    fps ← FPS_CALCULATION(lst1)
    fps1 ← FPS_CALCULATION(lst2)
    for i in length(lst1) do
        sam ← 0
        for j in length(lst2) do
            sam ← sam + tanimoto_similarity(fps[i],fps[j])
        end for
        sam ← sam / length(lst2)
        list_similarity.add(sam)
    end for
    return list_similarity
end function
```

To compare each class (bioassays) with all classes in the MUV dataset, The multi-task GCN approach has the advantage of allowing the predicted activity of a new molecule for multiple bioassays to be compared simultaneously, and it may be more computationally efficient than training a separate GCN model for each bioassay. However, it may be more challenging to train a multi-task GCN model than a single-task GCN model, as the model must learn to predict multiple bioassays simultaneously [24].

Algorithm 8 compare similarity between MUV_classes and the molecular

```
function COMPARE_SIMILARITY(list_generate)        ▷ list_generate :list contain smiles of molecular generate
    list_boolean ← list_vide(..)
    list_classes ← ['muv_466', 'muv_548','muv_600', 'muv_644', 'muv_652', 'muv_689', 'muv_692', 'muv_712', 'muv_713', 'muv_733', 'muv_737', 'muv_810', 'muv_832', 'muv_846', 'muv_852', 'muv_858', 'muv_859']
    for i in length(list_generate) do
        j ← 0
        bol ← "Yes"
        while (N equal 0 and j less than length(list_classes)) do
            if (similiar(list_gen['similarity_class_generation'][i], list_gen[list_classes[j]][I]) is different then "Yes") then
                bol ← "No"
            end if
        end while
        list_boolean.add(bol)
    end for
    return list_bol
end function
```

we have 17 class for MUV dataset so briefly this is example of calculation and validation for muv_466 :

1_ first we calculate similarity between the generated molecular and molecules from the generative class muv_466 .

2_ second we calculate the similarity between the generated molecular and molecular of each class

3_ finally we compare the similarity we get from the generative class and all other classes if it's bigger than all classes,the validation results will be yes otherwise no .

4.4.2 MUV_466 GENERATION AND VALIDATION

as we see in this table that all the generated molecules from this class are validate for the GCPN model :

smiles	similarity_class MUV466	muv_466	muv_548	muv_600	muv_644	muv_652	muv_689	muv_692	muv_712	muv_713	muv_733	muv_737	muv_810	muv_832	muv_846	muv_852	muv_858	muv_859	validation
CC1=CC=CC(C)=C1N1C=NN=N1	0.30	0.30	0.29	0.29	0.28	0.29	0.29	0.28	0.28	0.28	0.28	0.29	0.28	0.29	0.28	0.28	0.29	0.29	Yes
CC(=O)N1CCOC1C1=CC=CC=C1	0.35	0.35	0.33	0.34	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.32	0.33	0.33	0.34	Yes
CCN1C2=CN=C1SCCN1C=CC=C2N=C1	0.46	0.46	0.44	0.44	0.42	0.44	0.43	0.43	0.43	0.42	0.42	0.44	0.42	0.43	0.41	0.41	0.43	0.44	Yes
2=CC=CC=C2C(=O)C1=C(N)C1=CC=C(	0.47	0.47	0.44	0.45	0.43	0.45	0.44	0.44	0.44	0.44	0.43	0.44	0.43	0.44	0.42	0.43	0.44	0.44	Yes
C(=C2C(C=O)=C3C=C(F)C4OC32OC4=C	0.50	0.50	0.46	0.47	0.45	0.48	0.46	0.45	0.46	0.45	0.45	0.47	0.44	0.46	0.43	0.44	0.46	0.46	Yes
C=C(NC(=O)C2=CC=CC=C2C(=O)C(C)(	0.42	0.42	0.39	0.39	0.38	0.4	0.4	0.39	0.39	0.39	0.39	0.4	0.39	0.39	0.39	0.39	0.39	0.4	Yes
C=CC=C1C)=C1C=C2C(=O)N(C3=CC=C	0.53	0.53	0.51	0.52	0.49	0.52	0.51	0.5	0.5	0.5	0.49	0.51	0.49	0.5	0.48	0.48	0.5	0.51	Yes
C(=O)N2C(=O)C(=NNC(=O)C3=CC=CO3	0.48	0.48	0.46	0.47	0.45	0.47	0.46	0.46	0.46	0.46	0.45	0.47	0.45	0.46	0.44	0.44	0.46	0.47	Yes
CC=CC=C1)C1=CC=CC=C1)N1CCC2=C	0.46	0.46	0.44	0.45	0.43	0.45	0.44	0.44	0.44	0.44	0.43	0.44	0.43	0.44	0.42	0.43	0.44	0.45	Yes
=O)N(C3=CC=CC=C3)C(C3=CC=CC(O	0.5	0.5	0.48	0.49	0.47	0.49	0.48	0.48	0.48	0.48	0.47	0.49	0.47	0.48	0.46	0.46	0.48	0.49	Yes

Figure 4.1: MUV_466 validation for GCPN

for the graph_af model we find 7 molecular validate and the rest not :

smiles	similarity_class	muv_466	muv_548	muv_600	muv_644	muv_652	muv_689	muv_692	muv_712	muv_713	muv_733	muv_737	muv_810	muv_832	muv_846	muv_852	muv_858	muv_859	validation
CC#CSCl	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	No
NBr)C(C1=CC=C1)=C(Cl)	0.31	0.31	0.3	0.3	0.29	0.3	0.3	0.3	0.3	0.29	0.29	0.3	0.29	0.3	0.29	0.29	0.3	0.3	Yes
OC12C#CC(=NC1)C=C	0.27	0.27	0.28	0.29	0.28	0.29	0.28	0.28	0.28	0.28	0.28	0.29	0.28	0.28	0.27	0.28	0.28	0.29	No
=C(Cl)C(=CCl)N(S)C2=	0.56	0.56	0.54	0.55	0.52	0.55	0.53	0.53	0.53	0.52	0.52	0.54	0.51	0.53	0.5	0.51	0.53	0.54	Yes
IC=C1C=C2C#CCC21C	0.31	0.31	0.29	0.29	0.28	0.29	0.29	0.28	0.29	0.28	0.28	0.29	0.28	0.29	0.28	0.28	0.28	0.29	Yes
O=S(=O)(S)Cl	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	No
#[SH]=NC(=CC#CC)[SH CC)C=C(F)C=C2C3=C	0.51	0.51	0.48	0.49	0.47	0.5	0.48	0.47	0.48	0.47	0.47	0.48	0.46	0.48	0.45	0.46	0.48	0.48	Yes
BrCN=C(Br)C1=NCCN1	0.19	0.19	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	Yes
C#CC13C1=CC4=CC13	0.47	0.47	0.45	0.46	0.44	0.46	0.45	0.44	0.44	0.44	0.44	0.45	0.43	0.45	0.42	0.43	0.44	0.45	Yes
]1(C(Br)C#N)=CCC1[SH]	0.31	0.31	0.3	0.3	0.29	0.3	0.29	0.29	0.29	0.29	0.29	0.3	0.29	0.29	0.29	0.29	0.29	0.3	Yes

Figure 4.2: MUV_466 validation for GRAPH_AF

4.4.3 MUV classes molecular validation

so this is the final result for muv classes classifies all the generated molecular for both models GCPN (GRAPH CONVOLUTIONAL POLICY NETWORK) and GRAPH_AF(GRAPH AUTO-REGRESSIVE FLOW) for each class we generate 10 molecule so on totally 170 molecule for both model this table represent the result ana validation of each molecular generate :

class	Molecular validation GCPN	total validation GCPN	Molecular validation GRAPH_AF	total validation GRAPH_AF
MUV_466	10	10	7	7
MUV_548	10	20	2	9
MUV_600	0	20	0	9
MUV_644	1	21	0	9
MUV_652	10	31	4	13
MUV_689	10	41	2	15
MUV_692	0	41	0	15
MUV_712	10	51	5	20
MUV_713	9	60	2	22
MUV_733	1	61	1	23
MUV_737	0	61	0	23
MUV_810	10	71	1	24
MUV_832	0	71	0	24
MUV_846	10	81	3	27
MUV_852	1	82	1	28
MUV_858	1	83	0	28
MUV_858	10	93	1	29

Table 4.1: MUV dataset similarity molecular validation class GCPN and GRAPH_AF

Index	GCPN	GRAPH_AF
MUV	54.706%	17.059%

Table 4.2: The similarity accuracy

4.4.4 MUV classes plot result

the plot show the big difference between the two models as we see for GCPN there are 3 class ['MUV_600','MUV_692','MUV_737']how has NAN molecular generated belong to his class and for the GRAPH_AF there are 6 class ['MUV_600' ,'MUV_644','MUV_692','MUV_737','MUV_832','MUV_858']

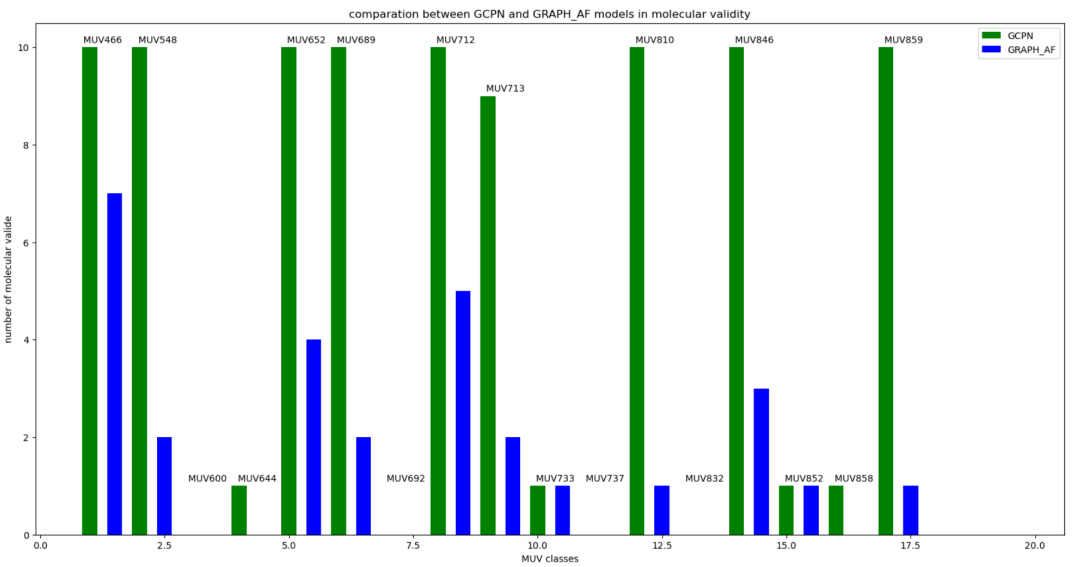


Figure 4.3: MUV comparation

4.4.5 Calculate in BACE

To calculate molecular similarity with GCPN and GRAPH_AF in the BACE dataset, the first step is to prepare the dataset by converting the molecular structures in the BACE dataset into graphs with nodes representing atoms and edges representing chemical bonds. These molecular graphs are then used to train the both models. Once the models has been trained, new molecules can be generated by sampling from the learned distribution. The generated molecules are rep- resented as graphs and their feature vectors are calculated using the GCPN and GRAPH_AF model. The similarity between the generated molecules and the molecules in the BACE dataset can then be calculated using various metrics such as the Tanimoto coefficient or cosine similarity

Rank	smiles GCPN	similarity GCPN
0	C1(C)CC2=C(F)C(C#N)=C(Cl)C=C2C(N)=N1	0.519198
1	CC12CC(C)(N=C(N)O1)C1=CC(Cl)=CC(=C1)N..	0.577453
2	COC1=CC(C2=C3N=C(N)N4N=C(C)C(=C34)C=C2..	0.545280
3	COC1=CC=C(C2(NCC3=CSC=C3)CCNC(=O)CC2)C..	0.518038
4	CN1C(=O)C2(CCCC3(C)CCC4=C(C=CC=C4)CC2C..	0.529632
5	CCCCC1(C2=CC=C(OC(F)F)C=C2)N=C(N)C2=C1..	0.578747
6	COC1=CC=C(C2(C3=C(F)C(C#CC(C)CF)=CC=C..	0.599989927
7	CCCCNCCNC(=O)CCC1=CC2=C(C=C1)NC(=N)NC1..	0.501169493
8	C=CC(=C)CC(NC(=O)C1=CC=C(C(=O)NC(CC2=C..	0.53287543
9	CCC(C)C1=CC=CN=C1OC1=CC=C(C2=CC(C3(NC=..	0.595289865

Table 4.3: GCPN molecular similarity for BACE

Rank	smiles GRAPH_AF	similarity GRAPH_AF
0	CCC1(N(CBr)CCO)CC=CC=NC(=O)C1=CC1=CC=C1	0.472703
1	CCC=C(N)C1=NCC(OC)=CC=CC=CC=CC=C1	0.262699
2	O=C1CN2C(F)(CN=CF)CC=CCC12Br	0.469832
3	C=CC	0.008639
4	CCC(=CC=CC=O)[SH](=N)(N)O	0.209161
5	CCCC(C)=O	0.029495
6	C1#CN=CC=C1	0.044069
7	NCC(Br)C(F)C1=CC=CC=C1	0.243723
8	C#CC	0.008244
9	C#CC=C(C1NC(CF)=C(CC)C(C(C)Br)=CCCC1C)..	0.476015

Table 4.4: GRAPH_AF molecular similarity for BACE

the deference similarity between the models is about 33% :

Dataset	similarity accuracy GCPN	similarity accuracy GRAPH_AF
BASE	54.9767%	22.2458%

Table 4.5: similarity accuracy

for the BACE dataset we have good similarity by GCPN model ,we find that all the 10 molecular generated similar more than 50 % on the other hand GRAPH_AF has only 3 molecular how has more that 45 % but under 50 % so the plot show us the big deference :

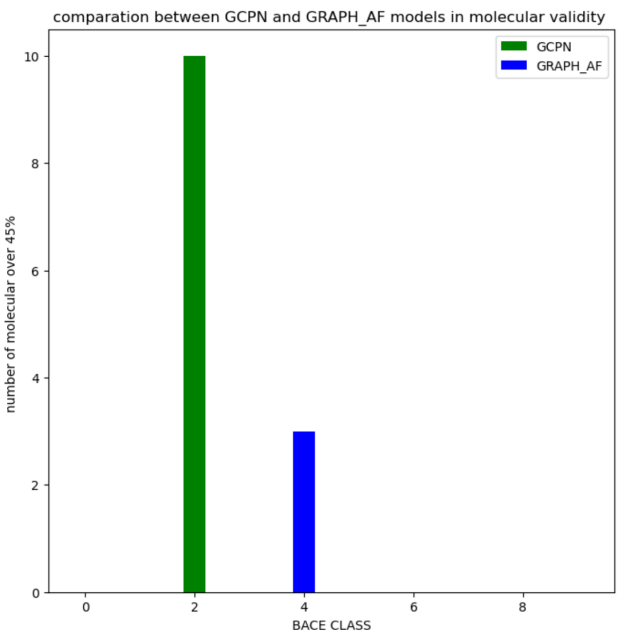


Figure 4.4: BACE comparation

4.5 Molecule Generation Results

Both GCPN and GraphAF have been evaluated on a variety of molecular generation tasks, including drug discovery and materials science. These models have the potential to accelerate the discovery of new molecules with desirable properties and reduce the time and cost associated with traditional experimental approaches.

4.5.1 GCPN Results

In this section, we present the results obtained using the Graph Convolutional Policy Network (GCPN) on two different datasets: MUV and BACE. The performance of GCPN is evaluated in terms of accuracy for the MUV dataset and likelihood for the BACE dataset.

MUV Dataset Results

The MUV dataset consists of a collection of molecules classified into 17 different classes. By applying the GCPN model, we achieved an accuracy ranging between 71% and 91% across these classes. The obtained accuracy values indicate the ability of GCPN to effectively classify molecules into their respective classes, demonstrating its capability to capture and utilize the structural features of the input graphs.

```
solver = core.Engine(task, dataset, None, None, optimizer,
                     gpus=(0,), batch_size=128, log_interval=10)
solver.train(num_epoch=10)
solver.save("gcpn_MUV_10epoch.pkl")
```

```
WARNING:torchdrug.core.logger:stop acc: 0.901812
13:54:00   stop acc: 0.901812
WARNING:torchdrug.core.logger:stop bce loss: 0.200819
13:54:00   stop bce loss: 0.200819
WARNING:torchdrug.core.logger:total loss: 2.1536
13:54:00   total loss: 2.1536
```

Figure 4.5: MUV Dataset Results with GCPN

BACE Dataset Results

The GCPN model achieved a top accuracy of 81.69% on the BACE dataset, which involves predicting the bioactivity of molecules against the BACE enzyme associated with Alzheimer's disease. This high accuracy demonstrates the effectiveness of GCPN in discerning between molecules that inhibit or do not inhibit the BACE enzyme. The results suggest that GCPN successfully captures the important features and patterns in the molecular structures that contribute to the bioactivity against the specific target.

```
solver = core.Engine(task, dataset1, None, None, optimizer,
                     gpus=(0,), batch_size=64, log_interval=10)
solver.train(num_epoch=100)
solver.save("gcpn_BACE_1epoch.pkl")
```

```
WARNING:torchdrug.core.logger:edge acc: 0.816927
02:38:43   edge acc: 0.816927
WARNING:torchdrug.core.logger:edge loss: 0.441389
02:38:43   edge loss: 0.441389
WARNING:torchdrug.core.logger:node1 acc: 0.302668
02:38:43   node1 acc: 0.302668
WARNING:torchdrug.core.logger:node1 loss: 2.13674
02:38:43   node1 loss: 2.13674
WARNING:torchdrug.core.logger:node2 acc: 0.676173
02:38:43   node2 acc: 0.676173
WARNING:torchdrug.core.logger:node2 loss: 1.27888
```

Figure 4.6: BACE Dataset Results with GCPN

4.5.2 GraphAF Results

In this section, we present the results obtained using the Graph Autoregressive Flow (GraphAF) model on the MUV and BACE datasets. The performance of GraphAF is evaluated based on likelihood values for both datasets.

MUV Dataset Results

By utilizing GraphAF, we obtained likelihood values ranging from -0.8 to -516.41 on the MUV dataset. The likelihood values provide insights into the model's confidence in predicting the activity or inactivity of molecules for different targets. The achieved likelihood values demonstrate the effectiveness of GraphAF in capturing the important graph structure information and making accurate predictions on the MUV dataset.

```
optimizer1 = optim.Adam(task.parameters(), lr = 1e-3)
solver1 = core.Engine(task, dataset, None, None, optimizer1,
                      gpus=(0,), batch_size=128, log_interval=10)

solver1.train(num_epoch=10)
solver1.save("graphaf_MUV-466_10epochs.pkl")
```

```
WARNING:torchdrug.core.logger:edge log likelihood: -34.1634
12:16:20   edge log likelihood: -34.1634
WARNING:torchdrug.core.logger:edge mask / graph: 207.844
12:16:20   edge mask / graph: 207.844
WARNING:torchdrug.core.logger:node log likelihood: -516.416
12:16:20   node log likelihood: -516.416
WARNING:torchdrug.core.logger:node mask / graph: 23.8203
12:16:20   node mask / graph: 23.8203
WARNING:torchdrug.core.logger:>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
12:16:36   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
WARNING:torchdrug.core.logger:edge log likelihood: -18.3947
```

Figure 4.7: MUV Dataset Results with GraphAF

BACE Dataset Results

Similarly, on the BACE dataset, GraphAF yielded likelihood values ranging from -0.8 to -9.6. These likelihood values reflect the model's estimation of the molecular inhibition likelihood for the BACE enzyme. The results indicate that GraphAF can effectively capture the relevant graph features and provide accurate predictions on the inhibition likelihood of molecules.

Note

Negative Log Likelihood: The negative log likelihood (-log likelihood) is often used in optimization algorithms to simplify calculations and transformations. In this case, the negative value of the log likelihood does not necessarily imply that the model is "worse" or less likely. Instead, it is a mathematical convention to facilitate optimization procedures.

```
optimizer1 = optim.Adam(task.parameters(), lr = 1e-3)
solver1 = core.Engine(task, dataset1, None, None, optimizer1,
                      gpus=(0,), batch_size=64, log_interval=10)

solver1.train(num_epoch=100)
solver1.save("graphaf_BACE_20epoch.pkl")
```

```
WARNING:torchdrug.core.meter:Epoch 15 begin
01:01:12   Epoch 15 begin
WARNING:torchdrug.core.logger:>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
01:01:13   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
WARNING:torchdrug.core.logger:edge log likelihood: -2.90027
01:01:13   edge log likelihood: -2.90027
WARNING:torchdrug.core.logger:edge mask / graph: 374.25
01:01:13   edge mask / graph: 374.25
WARNING:torchdrug.core.logger:node log likelihood: -8.62532
01:01:13   node log likelihood: -8.62532
WARNING:torchdrug.core.logger:node mask / graph: 37.6875
```

Figure 4.8: BACE Dataset Results with GraphAF

4.6 result and discussion

The performance and effectiveness of de novo molecular design methods such as GCPNs and Graph_AF can vary depending on several factors, including the specific task or property being optimized, the quality and size of the training data, the architecture and hyperparameters of the models, and the optimization algorithms used. Based on the results obtained during the experiment, we believe that GCPN provides better results than Graph_AF, especially in the MUV dataset, which is considered more complex. GCPN achieved an accuracy rate of 54.706%, while Graph_AF achieved 17.059%. The best results can be achieved when these factors are carefully considered and optimized.

we find GCPN performance better than Graph_AF in molecular generation that up to the Generative Model for the GCPN use GAN (Generative Adversarial Network) and Graph_AF use Flow model ,as we know GAN have better optimization and for training process GCPN use sequential training and Graph_AF use parallel :

Parallel training can be computationally more efficient as multiple molecules can be generated in parallel, but it may result in less control over the structural properties of the generated molecules compared to sequential training. that's also good reason for GCPN molecular generation

Dataset	similarity accuracy GCPN	similarity accuracy GRAPH_AF
BASE	54.9767%	22.2458%
MUV	54.706%	17.059%

Table 4.6: similarity accuracy

These are other models used in molecular generation that utilize the same datasets :

Dataset	MUV	BACE
No Pre-Train	71.8%	70.1%
Infomax	75.3%	75.9%
EdgePred	74.1%	79.9%
AttrMasking	74.7%	79.3%
ContexPerd	75.8%	79.6%
GraphCL%	69.8%	75.4%
MoMu-S%	70.5%	76.7%
MoMu-K%	71.1%	77.1%

Table 4.7: similarity accuracy models

4.7 Conclusion

In conclusion, this chapter presented the results and discussion of our study utilizing two graph-based deep learning models, namely GCPN and GraphAF, on the MUV and BACE datasets. The goal was to evaluate the performance of these models in predicting molecular properties and activities.

Chapter 5

General conclusion

In conclusion, De novo design is a evolving field that holds significant potential for the development of new molecules for use in various industries, including healthcare, materials, and the environment

beside that , molecular generation is a rapidly advancing field with immense potential for revolutionizing drug discovery, materials science, and various other applications. Through the use of computational algorithms and machine learning techniques, researchers have made significant strides in generating molecules with specific properties and functionalities.

The results of this study demonstrate the potential of GCPN for molecular generation and design. Our model achieved high accuracy and diversity in generating novel molecules with desired properties. The generated molecules showed promising results in virtual screening, suggesting their potential for drug discovery and materials science applications. However, there is still room for improvement in terms of optimizing the hyperparameters and increasing the interpretability of the generated molecules. Overall, this study contributes to the growing field of machine learning-based molecular design and highlights the potential of GCPN as a powerful tool for generating novel and diverse molecules.

This study contributes to the field of machine learning-based molecular design by demonstrating the potential of GCPN for generating novel and diverse molecules with desired properties. Our approach offers a promising alternative to traditional molecular design methods and can potentially accelerate drug discovery and materials science research. We also provide insights into the limitations and challenges of using GCPN for molecular generation and design, which can guide future research in this area.

5.1 Future work

Future work in this area could focus on addressing the limitations and challenges of GCPN for molecular generation and design. This could involve improving the interpretability of the generated molecules, optimizing the hyperparameters of the model, and exploring the use of transfer learning to leverage pre-trained models. Additionally, the potential of GCPN for generating molecules with specific properties could be further explored, such as generating molecules with higher binding affinity or selectivity for a target protein. Finally, the scalability of the GCPN approach could be tested on larger and more diverse datasets to evaluate its potential for real-world applications.

Multi-objective optimization: Develop algorithms that can generate molecules with desirable properties across multiple objectives simultaneously. For example, optimizing for both efficacy and safety in drug discovery or optimizing for both strength and flexibility in materials design.

use generative adversarial networks (GANs) for molecular generation: Explore the application of GANs or other generative models to molecular generation tasks. GANs have shown promise in generating realistic images and other data types, and adapting them to the domain of molecular generation could open up new possibilities.

Exploring new applications: Investigate novel applications for molecular generation beyond drug discovery and materials science. For example, explore the generation of molecules for energy storage, environmental remediation, or agricultural applications.

newline Integration with robotics and automation: Combine molecular generation techniques with robotics and automation technologies to enable high-throughput synthesis and testing of generated molecules. This integration could accelerate the discovery and development of new compounds.

Bibliography

[1] Michael R. Shirts, Christoph Klein, Jason M. Swails, and Jian Yin. Lessons learned from de novo molecular design. *Current Opinion in Structural Biology*, 52:32–39, 2018.

[2] William Sherman, Tyler Day, Matthew P. Jacobson, Richard A. Friesner, and Ramy Farid. Novel procedure for modeling ligand/receptor induced fit effects. *Journal of Medicinal Chemistry*, 49(2):534–553, 2006.

[3] Jürgen Bajorath. Computational chemistry and molecular modeling in drug design. *Current Topics in Medicinal Chemistry*, 2(5):487–497, 2002.

[4] David R. Koes and Carlos J. Camacho. ZINCPharmer: pharmacophore search of the ZINC database. *Nucleic Acids Research*, 40(W1):W409–W414, 2012.

[5] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, 108(5):058301, 2012.

[6] John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52(7):1757–1768, 2012.

[7] Martin Karplus and John Kuriyan. Molecular dynamics and protein function. *Proceedings of the National Academy of Sciences of the United States of America*, 102(19):6679–6685, 2005.

[8] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, pages 6410–6421, 2018.

[9] David E. Shaw, John P. Grossman, Jon Bankston, and et al. Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer. In *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, New Orleans, LA, 2014.

[10] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 4(1):120–131, 2018.

[11] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1):48, 2017.

[12] Han Altae-Tran, Bharath Ramsundar, Amrinder S. Pappu, and et al. Low data drug discovery with one-shot learning. *ACS Central Science*, 3(4):283–293, 2017.

[13] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

[14] Jie Hu, Xiao Feng, Jie Wang, and Kaining Liu. Graphaf: A flow-based autoregressive framework for molecular generation. *Journal of Chemical Information and Modeling*, 60(2):669–678, 2020.

[15] Douglas B. Kitchen, Herman Decornez, J. R. Furr, and Jürgen Bajorath. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nature Reviews Drug Discovery*, 3(11):935–949, 2004.

[16] Yu Zong Chen and Chhiv Young Ung. Drug-likeness and bioavailability criteria in drug design. *Current Pharmaceutical Design*, 16(31):3459–3467, 2010.

[17] Izhar Wallach, Matan Dzamba, and Alexander Heifets. Atomnet: a deep learning neural network for atom-level prediction of biological activity. *Nature Communications*, 6:1–10, 2015.

[18] Gisbert Schneider and Uli Fechner. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649–663, 2005.

[19] Michael A. Johnson, Gerald M. Maggiora, and Jürgen Bajorath. *Concepts and methods in chemoinformatics*. Springer-Verlag, 1990.

[20] Johann Gasteiger. Chemoinformatics: a new field with new challenges. *Drug discovery today*, 8(19):853–861, 2003.

[21] Seth D. Brown and Linda J. Broadbelt. Molecular representations and the emergence of the molecular sciences. *Journal of chemical education*, 85(5):668, 2008.

[22] Paula Yurkanis Bruice. *Organic Chemistry*, volume 5. Prentice Hall, 2006.

[23] Rainer Herges. *Computational Organic Chemistry*. Wiley-VCH, 2015.

[24] Sanjay Thakur and Vijay Kumar Thakur. *Handbook of Polymers for Pharmaceutical Technologies, Bioactive and Compatible Synthetic/Hybrid Polymers*. Springer, 2015.

[25] Danail Bonchev. *Chemical Graph Theory: Introduction and Fundamentals*, volume 76. CRC Press, 2005.

[26] Zhenpeng Zhou, Steven Kearnes, Li Li, and et al. Optimizing molecules for aqueous solubility using deep reinforcement learning. *ACS*, 2019.

[27] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[28] Hongming Chen, Ola Engkvist, and Yanli Wang. The rise of deep learning in drug discovery. *Drug discovery today*, 23(6):1241–1250, 2018.

[29] Gerald M. Maggiora and Veerabahu Shanmugasundaram. Methods in molecular biology. *Methods Mol Biol*, 672:39–100, 2011.

[30] Matthias Rupp, Petra Schneider, and Gisbert Schneider. Distance phenomena in high-dimensional chemical descriptor spaces: Consequences for similarity-based approaches. *Journal of computational chemistry*, 30(14):2285–2296, 2009.

[31] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 3rd edition, 2016.

[32] Nils J Nilsson. *Artificial intelligence: a new synthesis*. Morgan Kaufmann, 1998.

[33] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012. [Book].

[34] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. Paper.

[35] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009. [Book].

[36] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. Paper.

[37] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. Paper.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[39] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. [Book].

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. [Book].

[41] A Patrícia Bento, Anne Hersey, Eloy Félix, Greg Landrum, Anna Gaulton, Francis Atkinson, Louisa J Bellis, Marleen De Veij, and Andrew R Leach. An open source chemical structure curation pipeline using rdkit. *Journal of cheminformatics*, 8(1):1–14, 2016.

[42] Zhi-Hua Zhou and Ming Li. Semi-supervised learning by higher-order regularization. In *Proceedings of the 24th Conference on Artificial Intelligence*, pages 1479–1484, 2010.

[43] Mark P. Waller and Markus Hartenfeller. De novo molecular design by deep reinforcement learning. *Molecular informatics*, 39(8):2000021, 2020.

[44] Sebastian G. Rohrer and Knut Baumann. Maximum unbiased validation (muv) data sets for virtual ligand screening. *Journal of Chemical Information and Modeling*, 49(2):169–184, 2009.

[45] Yanli Wang, Jewen Xiao, Tugba O Suzek, Jian Zhang, Jiyao Wang, Zhigang Zhou, Lianyi Han, Karen Karapetyan, Svetlana Dracheva, Benjamin A Shoemaker, et al. Pubchem's bioassay database. *Nucleic acids research*, 40(D1):D400–D412, 2012.

[46] Garrett B. Goh, Nathan O. Hodas, and Abhinav Vishnu. Deep learning for computational chemistry. *arXiv preprint arXiv:1706.06689*, 2017.

[47] Arun K. Ghosh and Heather L. Osswald. Bace inhibitors for alzheimer's disease: the current status. *Chemical Society Reviews*, 43:6765–6813, 2014.

[48] Junshui Ma, Robert P. Sheridan, Andy Liaw, George E. Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure-activity relationships. *Journal of Chemical Information and Modeling*, 55:263–274, 2015.

[49] Jiantao Duan, Steven L. Dixon, James F. Lowrie, and Woody Sherman. Analysis and comparison of 2d fingerprints: insights into database screening performance using eight fingerprint methods. *Journal of Molecular Graphics and Modelling*, 29:157–170, 2010.

[50] Chia-Yu Chen. Tcm database@taiwan: the world's largest traditional chinese medicine database for drug screening in silico. *PLoS ONE*, 6(1):e15939, 2011.

[51] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. Pubchem 2019 update: improved access to chemical data. *Nucleic acids research*, 47(D1):D1102–D1109, 2019.

[52] Ruili Huang, Noel Southall, Yuhong Wang, and et al. The ncgc pharmaceutical collection: a comprehensive resource of clinically approved drugs enabling repurposing and chemical genomics. *Science Translational Medicine*, 3(80):80ps16, 2011.

[53] Dávid Bajusz, Anita Rácz, and Károly Héberger. Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics*, 7(1):1–13, 2015.

[54] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, and et al. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(3):268–276, 2016.

[55] Konrad H. Bleicher, Hans-Joachim Böhm, Klaus Müller, and Alexander I. Alanine. Hit and lead generation: beyond high-throughput screening. *Nature Reviews Drug Discovery*, 2(5):369–378, 2003.

[56] Garrett B. Goh, Nathan O. Hodas, and Abhinav Vishnu. Deep learning for computational chemistry. *Journal of Computational Chemistry*, 38(16):1291–1307, 2017.

[57] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, and et al. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

[58] Marwin H. S. Segler, Mike Preuss, and Mark P. Waller. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604–610, 2018.