الجمهورية الجزائرية الديموقراطية الشعبية

POPULAR DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليم العالي والبحث العلمي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جامعة د. يحي فارس بالمدية

YAHIA FARES UNIVERSITY OF MÈDÉA



Graduation Project

A thesis submitted in partial fulfillment of the requirements for the award of the degree of Master of Computer Science

Specialty: Systems Engineering and Web Technology

TITLE

Mol2text : Automatic Molecular Description Using Deep Learning

Proposed and Directed by:

Mr. Hentabli Hamza

Prepared and Presented by:

Mr. Mihoubi Houssem

Ms. Yesseri Marwa

Academic year 2021-2022

ACKNOWLEDGEMENTS

First, I would like to thank ALLAH S.W.T. for all the achievements that I have gained today. Next, I wish to extend my grateful appreciation to all those who have contributed directly and indirectly to the preparation of this study. I would like to take this opportunity to thank my supervisor, Associate Professor Dr. Hentali Hamza for attention, encouragement and guidance throughout the length of this study. Not forgetting my beloved family for all supports and understandings they gave me.

Second, I would like to thank my evaluators for their critics and comments. All have helped me improve my work as well as my report writing.

Lastly, I am grateful to all my colleagues, friends, staff, and lecturers in Faculty of Computer Science and Department Of Mathematics And Computer Science, University Yahia Fares Medea and for their help and support at every step during this study.

DEDICATION

First of all, we must thank the God who gave us the strength and courage to succeed in carrying out this humble act.

We lead this humble act to: Our dearest parents neither dedication nor word expressing our gratitude and deep respect.

Your affection.

I pray to God Almighty to preserve and grant health, happiness and long life.

Our dear parents.

And to our brothers, too.

For all our family.

And to all our friends without mentioning the names.

By the way, we would like to say that we are lucky to have people who love us.

And those who are always there for us.

Abstract

In recent years, automatic image annotation (AIA) has become an emerging research topic aiming at reducing human labeling efforts for large-scale image collections . AIA refers to the process of automatically labeling the images with a predefined set of keywords or concepts representing image semantics .the aim of AIA is to build associations between image visual contents and concepts ;the aim of this reserach is to develop a novel method of Automatic Molecules annotaion MOL2text ,based on deep learning graph neural network (GNN) ,to build associations between molecules structure and concepts of key word .

Key words

Molecular description , deep learning , Automatic image annotation

Résumé

Ces dernières années, l'annotation automatique d'images (AIA) est devenue un sujet de recherche émergent visant à réduire les efforts d'étiquetage humain pour les collections d'images à grande échelle. AIA fait référence au processus d'étiquetage automatique des images avec un ensemble prédéfini de mots-clés ou de concepts représentant la sémantique de l'image. Le but de l'AIA est de créer des associations entre le contenu visuel de l'image et les concepts ; le but de cette recherche est de développer une nouvelle méthode d'Annotation des molécules MOL2text, basé sur le réseau neuronal de graphe d'apprentissage profond (GNN), pour construire des associations entre la structure des molécules et les concepts de mots clés.

Mots Clés

Apprentissage Automatique , Description Moléculaire , Annotation Automatique des images

ملخص

في السنوات الأخيرة ، أصبح التعليق التوضيحي التلقائي للصور (AIA) موضوعًا بحثيًا ناشئًا يهدف إلى تقليل جهود وضع العلامات البشرية لمجموعات الصور على نطاق واسع. يشير AIA إلى عملية التصنيف التلقائي للصور بمجموعة محددة مسبقًا من الكلمات الرئيسية أو المفاهيم التي تمثل دلالات الصور. الهدف من AIA هو بناء روابط بين محتويات ومفاهيم الصورة المرئية ؛ الهدف من هذا البحث هو تطوير طريقة جديدة للتلقائي إعلان الجزيئات MOL2text ، استنادًا إلى الشبكة العصبية للرسم البياني للتعلم العميق (GNN) ، لبناء روابط بين بنية الجزيئات ومفاهيم الكلمة الرئيسية.

الكلمات المفتاحية

التعلم العميق , وصف الجزيء , شرح تلقائي للصورة

Table of Contents

Figure List IX
Table List X
CHAPTER 1 INTRODUCTION
Introduction 1
1.1 Background of the Problem 2
1.2 Problem Statement 4
1.3 The objective 5
1.4 Importance of the study 6
1.5 Scope of study 6
1.6 Thesis outline 7
CHAPTER 2 LITERATURE REVIEW
Introduction 8
2.1 Cheminformatics 9
2.2 Molecular Representations 11
2.2.1 MDL Molfile 13
2.2.2 STRING REPRESENTATIONS 14
2.2.3 Simplified molecular-input line-entry system (smiles) 15
2.2.4 Extended-connectivity fingerprints 16
2.3 Molecular similarity measurement 17
2.3.1 Similarity Coefficients 18
2.3.2 Fingerprint similarity 20
2.4 Deep learning 20
2.4.1 Convolution Neural Network (CNN) 22
2.4.2 RECURRENT NEURAL NETWORK (RNN) 23
2.4.3 LONG-SHORT-TERM MEMORY (LSTM) 24
2.5 Deep learning model for automatic image annotation 25
2.6 Automatic image annotation 26
2.7 Discussion 27
2.8 Summary 28

CHAPTER 3 RESEARCH METHODOLOGY

Introduction....30
3.1 Research Framework....30
3.2 Dataset....31
3.3 Design and steps....32
3.3.1 Molecule and Caption Dataset....32
3.3.2 Prepare molecule fingerprint Data....32
3.3.3 Prepare Text Data....38
3.3.4 Develop Deep Learning Model....41
3.3.4.1 Loading Data....41
3.3.4.2 Defining the Model....45
3.3.4.3 Fitting the Model....46
3.3.5 Evaluate Model....47
3.4 Summary....50

CHAPTER 4 EVALUATION AND TEST

Introduction....51
4.1 Environment....51
4.1.1 Hardware used....55
4.2 Splitting the data....55
4.3 Develop the Model....56
4.4 Result model....58
4.5 Summary....63

GENERAL CONCLUSION

Conclusion....67

REFERENCES

REFERENCES....68

Figure List

Figure 2.1 A hierarchy of chemical structure representations, increasing in information content from left to right and top to bottom 10

Figure 2.2 Overview of the different classes of molecular representations 12

Figure 2.3 Example of a connection table and end line within an MDL molfile V2000 for the molecule leucine generated by ChemDraw 13

Figure 2.4 Structure graph of dicycloverine hydrochloride 14

Figure 2.5 Some examples of the smiles stings for a variety of molecule 16

Figure 2.6 Example for fingerprint. 17

Figure 2.7 A molecular fingerprint representation 20

Figure 2.8 The general Convolution Neural Network configuration 22

Figure 2.9 The max-pooling operation using 2×2 filters 23

Figure 2 .10 A simple Recurrent Neural Network model 23

Figure 2 .11 The architecture of LSTM. 24

Figure 2 .12 A general deep learning model 26

Figure 2 .13 A general system framework for AIA 27

Figure 3.1 PaDEL descriptor calculator graphical user interface 33

Figure 3.2 PaDEL descriptor calculator graphical user interface 34

Figure 3.3 PaDEL descriptor: calculate the 12 fingerprint interface 35

Figure 3.4 Example for Generate fingerprint matrix from molecule smile 37

Figure 4.1 Running this process for the molecules use rdkit 54

Figure 4.2 Plot of the Caption Generation Deep Learning Model 57

Table List

Table 2.1 Various representations of the dicycloverine hydrochloride molecule 15

Table 2.2 Distance coefficient 19

Table 3.1 Example of the dataset 31

Table 3.2 The 12 selected fingerprint used in this research 36

Table 4.1 Organization of train data 55

Table 4.2 The structure of model 56

Table 4.3 The description generated 58

CHAPTER 1
INTRODUCTION

Introduction

Advancements in medical and pharmaceutical chemistry have relied on the tried-and-true trial-and-error method. Due to the lengthy and painstaking nature of this approach, only a few compounds with drug-like properties could be identified after several years of investigation. Furthermore, it was not possible to grasp why some compounds had the drug-like feature while just a few others linked to these molecules do not. Chemical biology, or the systematic study of drug-like chemicals in biological systems, is a relatively new field.

To begin with, chemistry has generated a massive amount of data, which is continually growing. More than 45 million chemical compounds have been identified, with the number growing by millions each year. Combinatorial chemistry and high-throughput screening are two new approaches that create massive volumes of data. Only by putting all of this data and information in databases can it be controlled and made accessible [1].

Second, many chemical issues are too complicated to be addressed using first-principles approaches and theoretical computations. This holds true for the connections between a compound's structure and its biological activity, as well as the impact of reaction circumstances on chemical reactivity. All of these chemistry difficulties need unique methods for organizing enormous volumes of chemical structures and data, extracting information from data, and modeling complicated interactions. Cheminformatics approaches can help in this situation [2].

So Cheminformatics is the use of computer and informational approaches to a variety of chemical challenges. Cheminformatics and chemical informatics are other terms for the same thing. Pharmaceutical firms employ these in silico tools in the drug discovery process. Cheminformatics may also be used to analyze data for a variety of businesses, including paper & pulp, dyes, and other related sectors. The storing of information about chemicals is the most common use of Cheminformatics. Chemoiformatics also includes quantitative structure–activity connection analysis. Many in silico chemoinformatic methods are now available for predicting the physiochemical characteristics and biological activity of a wide range of chemical compounds. As a result, Cheminformatics aids in the identification of possible therapeutic targets as well as the understanding of physical, chemical, and biological aspects of a variety of chemical molecules. Cheminformatics outputs may also be used to guide the direction of wet laboratory studies [3].

On the other hand, the pharmaceutical business is receiving a lot of Cheminformatics attention right now. When a prospective medicine is being developed, high-throughput screening methods are used to screen libraries of compounds for activity using one or more relevant assays. The binding, potency, selectivity, and functional activity of candidate compounds are then assessed. Pharmaceutical firms increasingly employ sophisticated modeling tools to forecast the toxicity of compounds long before they are tested on humans, saving time and money while avoiding animal sacrifice, and allowing them to quickly reject chemical compounds [1].

Chemical informatics can specialize in molecular modeling, chemical structure coding and searching, chemical data visualization, and a variety of other topics. Structure(SMILES), molecular connectivity (SDfile) ,3D chemical structure, pharmacophore , structure similarity, molecular similarity and diversity ,chemical property, HTS and activity fingerprint, lead identification and optimization, combinatorial library design and synthesis, and metabolic profile should all be available from a Cheminformatics system [4].

Cheminformatics is a broad word that refers to the generation, organization, management, retrieval, analysis, dissemination, visualization, and use of chemical data. Clearly, converting data to information and information to knowledge is a task that is required in every discipline of chemistry, not only drug creation. As a result, we agree that Cheminformatics methodologies are required in all fields of chemistry [2].

1.1 Background of the Problem

In the recent years, the low number of novel therapeutics approved by the industrialised nation‘s medical institutions such as the US has caused great concern about the productivity levels and apparently declining innovation levels of the pharmaceutical industry. Reasons cited for this ineffectiveness are the hierarchical organisational structures of vertical integrated companies with their rigid and inflexible research units. In addition, there is the danger of innovation stagnating through the rising transaction costs of the bringing together of current research results [5]. The pressure led to find new active compounds and bring them to the market as quickly as possible has led many pharmaceutical companies to use Cheminformatics technologies in their product discovery and development processes. These companies hope that the drug discovery process could be accelerated dramatically by increasing the capabilities of producing and testing compounds.

Cheminformatics originally emerged as a vehicle to help the drug discovery and development process, however Cheminformatics now plays an increasingly important role in many areas of biology, chemistry, and biochemistry.

Some fundamental problems in processing chemical information had to be solved. Foremost of all was the need to make the computer understand the language of chemists, the representation of chemical structures by structure diagrams and the representation of chemical reactions by reaction equations. This chemical language is largely graphical in nature and therefore molecule editors have been developed to communicate with the computer in a graphical manner. Over the years various methods for storing molecular information have been developed[6]. However, it became quite clear that molecules should be represented in a way that allows access to each atom and each bond in a molecule. Standards for exchanging such structural information have been agreed upon both in form (SMILES). Databases with chemical information had a big impact on how chemical research is being done. The high amount of chemical in-formation available can only be managed through chemo-informatics methods. Without databases modern scientific research in chemistry and related fields would simply be in-conceivable.

Chemists use combinatorial chemistry to produce enormous and diverse libraries of structures, in which a large number of compounds are created by systematically and repetitively mixing sets of different types of reagents, known as building blocks. Because compounds in the libraries may be randomly tested or screened to locate a promising medicinal compound, these libraries can be utilized as a source of new potential medications. Using Cheminformatics to improve the capabilities of testing chemicals, it is now feasible to test hundreds of thousands of these compounds in a short period of time. Computers may help with this process in a variety of ways, such as generating captions for molecules [7].

We aims to pave the way for computational drug discovery and accelerate the advancement of molecular artificial intelligence in a practical manner. It was difficult to construct a caption for a molecule from a variety of large-scale datasets. With the availability of large-scale annotated data, such as the chembl [8]dataset, and the deployment of deep learning algorithms, this changed dramatically. Molecule classification networks trained on the big chembl dataset have been found to generalize effectively and may be utilized as generic molecule extraction for a variety of purposes. This has led to the effective use of deep networks to a variety of

different computer vision applications, including molecular captioning. We suggested a unique approach in this thesis to investigate the task of autonomous molecular description creation.

The goal of this project is to build a unique ligand-based searching strategy that uses a network and fingerprints to search a vast chemical library in order to improve the efficacy of similarity algorithms. In addition, if numerous reference structures and fingerprints are available, this approach should be able to manage them. The availability of various options the use of reference structures and a variety of fingerprint kinds can assist improve the approach of generating a generic caption for a molecule.

1.2 Problem Statement

The comparable property principle asserts that structurally similar molecules are more likely to show similar characteristics than structurally different compounds. This is the foundation of molecular captioning. The notion of molecular captioning is now frequently applied in rational medication development.

Each approach can uncover some actives that the others would overlook. Furthermore, certain approaches may perform better on some tasks than others [6]. It's tough to determine the best approaches ahead of time. As a result, it's best to combine different similarity approaches wherever possible.

Many research in the field of information retrieval have shown that retrieval models based on inference networks perform significantly better than conventional models. Numerous strategies in the field of Cheminformatics come from the field of information retrieval, where there are many molecular captioning between them [9] .The goal of this thesis is to create a molecular captioning technique.

Smile Given an input molecule (smile), our objective is to construct a caption. We'll just look at approaches for creating single-sentence captions. As a result, a caption may be exactly described as a sentence, which is a series of words with the length of the sentence being the length of the sentence. The molecule (smile) is entered in the first stage to transform it to a fingerprint. This method is deterministic, and the retrieved feature vectors have a constant size for each input. We may test several strategies for extracting feature vectors from molecules and compare their performance on the automated captioning challenge objectively and subjectively. For single molecule classification, for example, features derived from smiles and fingerprints trained on the chembl [8] database are used, The language model, which takes the

visual feature vector If as input and learns a probability distribution across phrases, is the next step. Because sentences are made up of a series of words, they lend themselves to being represented using sequential models, such as the Long-Short Term Memory (LSTM) variety. We developed the concept and made our code available to the public.

It's also debatable how to evaluate a molecular captioning system because we have to compare the produced caption to a few distinct reference captions. We also employ a number of automatic evaluation indicators derived from machine translation research. We also offer the results of a human review based on the automatic captioning challenge. We examine the benefits and weaknesses of our solution after analyzing the outcomes of the trials.

1.3 The objective

In terms of molecular description production, differentiates between two approaches: models that produce original descriptions for a particular molecule and models that build description filling templates or retrieve descriptions associated with comparable molecules; we will concentrate on the former [10].

We'll limit the scope of this thesis to creating descriptions of molecules that include since it's a fascinating issue on which we can concentrate our efforts. Related efforts in automatic molecular description generation haven't always concentrated on a specific molecule issue, instead attempting to cover all conceivable descriptions and molecules.

The project's key goals are as follows:

- Review and analyze the state-of-the-art techniques of generating molecule descriptions.
- Study and understand Deep Learning models and their applications.
- Investigate several fingerprint types for use molecular representation.
- Study different Deep Learning frameworks and tools for molecular description

1.4 Importance of the study

The molecule captioning concept asserts that molecules with comparable structural characteristics would have similar physicochemical and biological properties. Many rational drug design initiatives have been built on this foundation.

Technological improvements in synthesis and high throughput screening [11]have expanded the capacity to synthesize enormous libraries of compounds and screen hundreds of thousands of compounds in a short amount of time over the previous decade.

As a result of these advancements, computer-based approaches [12] for compound selection and assessment are becoming increasingly important. Furthermore, as computer power has increased, molecule captioning software may now be run on very huge databases of compounds.

1.5 Scope of study

The scope of this study is confined to the following in order to fulfil the above-mentioned objectives:

1. The datasets that will be used in this investigation will be limited to chemical data from the chembl and ID databases.

2. Produce a molecule Caption

3. Putting models for description creation to the test.

1.6 Thesis outline

This thesis is divided into four chapters:

The first chapter introduce our theme and thesis.

The second chapter cover the background of molecular caption, we discuss about Cheminformatics, molecular representations and deep learning.

The third chapter describes the research methods, methodology to develop the algorithm and all the steps of algorithm.

Finally, the last section of the chapter presents the algorithm's results.

CHAPTER 2
LITERATURE REVIEW

Introduction

The computer revolution and the introduction of high-throughput screening tools in drug development paved the way for computational analysis and visualization of bioactive compounds throughout the previous century's technical advancements. It was important to express molecules in a syntax that could be read by computers and understood by scientists from many domains in order to achieve this goal. Chemical representations have grown in quantity throughout time, owing to the rapid advancement of computers and the difficulty of creating a representation that incorporates all structural and chemical features. Many of the most common electronic molecular and macromolecular representations utilized in drug development are based on graph representations. Furthermore, We also discuss how these representations may be used in AI-assisted drug development. Our goal is to give a quick primer on structural representations, which are critical in AI drug development. Molecules are complicated structures, and representations must account for a wide range of features, as well as the differences in nature of small molecules and macro molecules. The advancement of cheminformatics and bioinformatics has resulted in a more rapid and efficient drug development process, as well as a better knowledge of molecular behaviour. We believe that practicing cheminformaticians, students, and anybody else interested in learning more about the underlying molecular representations in cheminformatics that may be employed in AI-driven drug discovery applications will find this review useful.

Recently, much attention has been given to image annotation due to the massive increase in image data volume. One of the image retrieval methods which guarantees the retrieval of images in the same way as texts are automatic image annotation (AIA). Consequently, numerous studies have been conducted on AIA, particularly on the classification-based and probabilistic modeling techniques. Several image annotation techniques that performed reasonably on standard datasets have been developed over the last decade. In this paper, a review of the image annotation method was conducted, focusing more on deep learning models. Automatic image annotation (AIA) methods were also classified into five categories, including i) Convolutional Neural Network (CNN) [13] based on AIA, ii) Recurrent Neural Network (RNN) based on AIA, iii) Deep Neural Networks (DNN) based on AIA, iv) Long-Short-Term Memory (LSTM) based on AIA, and v) Stacked auto-encoder (SAE) based on AIA. An assessment of the five varieties of AIA methods was also offered based on their principal notion, feature mining technique, explanation precision, computational density, and examined aggregated data. Moreover, the evaluation metrics used to evaluate AIA methods were

reviewed and discussed. The need for careful consideration of methods throughout the improvement of novel procedures and datasets for image annotation assignment was highly demanded. From the analysis of the achievements so far, it is certain that more attention should be paid to automatic image annotation.

2.1 Cheminformatics

Cheminformatics [2] is the application of computer approaches to chemical problems, with a focus on structural data manipulation. While this type of research has been done for a long time, its use has exploded in recent years, owing in part to the massive amounts of data created by high-throughput screening and combinatorial chemistry. Concurrent advances in computing capacity have enabled the creation of a slew of novel techniques.

'Representation and Manipulation of 2D Molecular Structures' [10] is the definition of Cheminformatics. It goes on some of the techniques for storing chemical structures in huge databases, as well as how to look for specific substructures inside molecules. Two-dimensional databases are used to store and retrieve full chemical structures and their related attributes, but they can also be used to look for substructures, or identical portions of structures. A database might be searched for all structures having a benzene ring, for example. Implementation approaches include a variety of options.

'3D Molecular Structure Representation and Manipulation' [10] is the title of the paper. 3D descriptors include information about the relative orientation of those atoms in space, in addition to the types of atoms and their connectivity. The steric and electrical characteristics of the molecules are affected by these orientations. 3D approaches are more difficult to work with than 2D methods, but they are particularly useful for finding molecules with comparable characteristics but radically distinct underlying structures. The fact that molecules large enough to be of interest can adopt many low-energy conformations adds to the complexity.

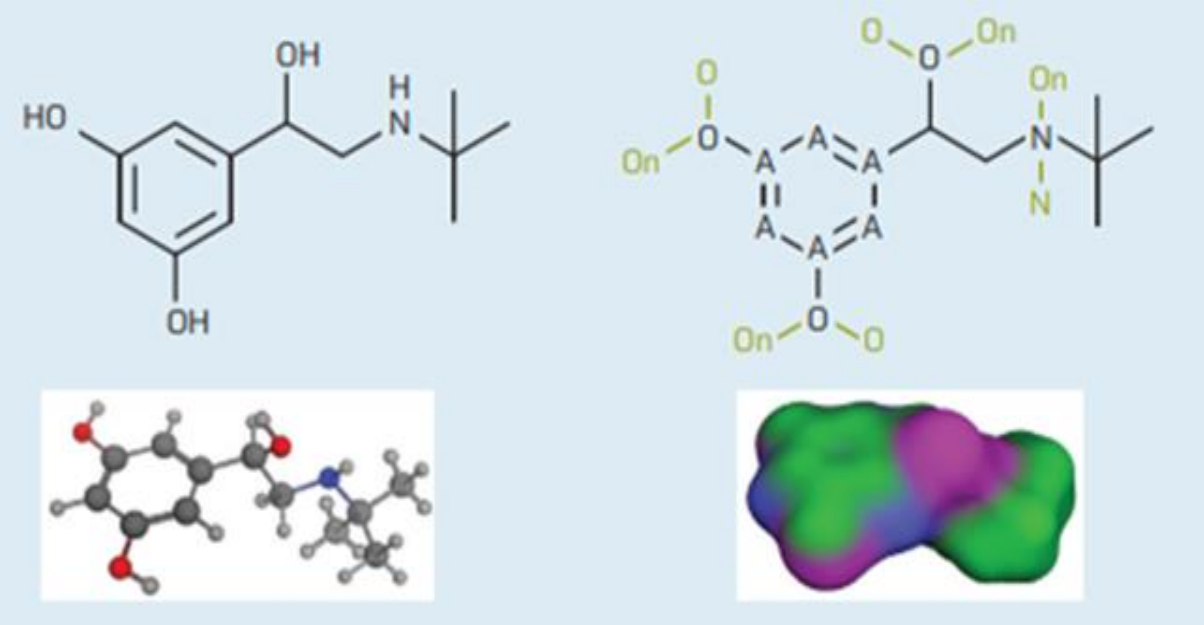


Figure 2.1 A hierarchy of chemical structure representations, increasing in information content from left to right and top to bottom

To balance the competing goals of computing efficiency and accurately capturing the variety of low-energy conformations for a single molecule, many techniques have been created. The Cambridge Structural Database and the Protein Databank are the two most well-known databases of 3D structures, each comprising single, high-resolution crystal or NMR structures of particular molecules, and both are briefly detailed. Because crystal structures for all chemicals are not available, databases of theoretical structures are employed instead. There are several methods for creating and searching databases of theoretical structures (including confined systematic search, clique identification, maximum likelihood, and evolutionary algorithms).

Instead of working with full structures, molecular descriptors are sometimes used. Molecular descriptors are numerical numbers that are used to describe the properties of molecules. . Fragment screens and pharmacophore keys are examples of three-dimensional descriptors.

Substructure or pharmacophore searches can be replaced by similarity searches. Similarity searching takes a complete molecule and searches for other compounds that are similar to the query molecule, rather than identifying a specific component of the molecule and searching for other molecules that contain that same piece. The similarity of each molecule in the database to the query molecule is rated. The subject of how to define 'similar' is complicated, and several

2D and 3D similarity metric methods are examined. Fingerprints, similarity coefficients, and the greatest common subgraph are examples of 2D approaches. The alignment of the molecules is required for most 3D approaches, and numerous methods for determining the molecular overlay that maximizes similarity are addressed.

'Selecting a Variety of Compounds' is a means' selecting a variety of compounds'. Instead of picking related compounds, a group of compounds that covers the most structural space may be necessary. Navigation across relevant chemical space and selection of potentially helpful subsets are made possible by diversity approaches. Cluster analysis, both hierarchical and non-hierarchical, dissimilarity-based approaches, cell-based methods, and optimization methods are among the methodologies explored. The chapter concludes with an overview of how the various strategies function and when each might be appropriate.

Virtual screening, or totally in silico grading and ranking of expected activity of chemicals. The speed of the scoring function is critical since virtual combinatorial libraries can contain billions of chemicals. Simple principles and filters for predicting drug-likeness, as well as structure-based screening approaches like protein–ligand docking, are described. Practical issues are explored, such as how to set up computational experiments in a timely and scientifically valid manner. Prediction of ADMET (absorption, distribution, metabolism, excretion, and toxicity) features, those pesky biological processes that keep active molecules from becoming marketable medications, takes up a significant portion of the book.

2.2 Molecular Representations

Robust representation of molecules [10] is required for accurate functioning of the ML models. An ideal molecular representation should be unique, invariant with respectto different symmetry operations, invertible, efficient to obtain, and capture the physics, stereo chemistry, and structural motif. Some of these can be achieved by using the physical, chemical, and structural properties[14].

Molecules can be represented on a piece of paper using a two-dimensional (2D) scheme such as the one of dicycloverine hydrochloride in Scheme 1, but there are many different options for representing [15] compounds computationally. When computers were first commercialized, strings of alphanumerical characters were preferred, as these required less memory to store and less computational power to process, but as computers developed and memory/processing power became less expensive, less compact representations (that are more

flexible and less ambiguous) became more wide-spread. An overview of the different classes of molecular representation can be seen in Figure 1

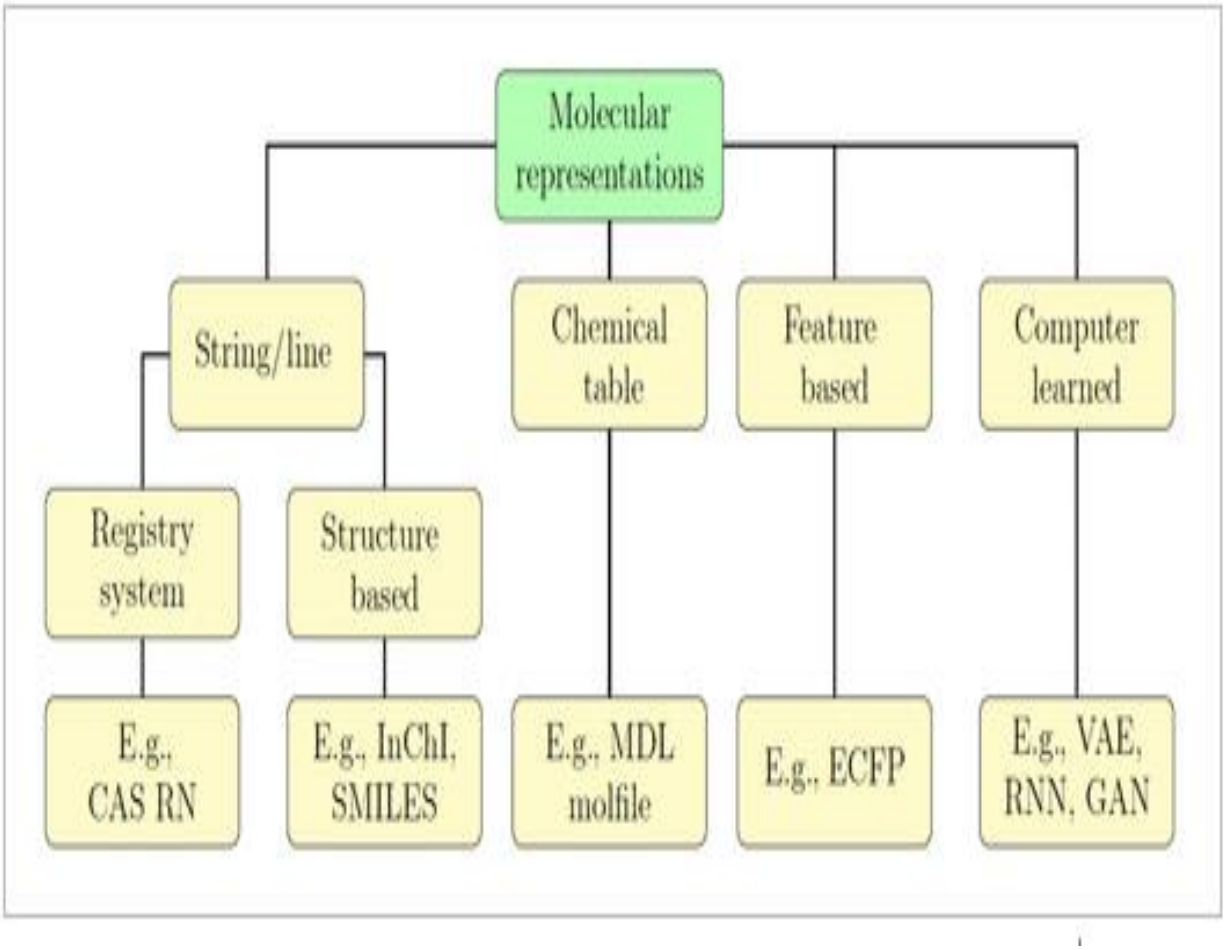


Figure 2.2 Overview of the different classes of molecular representations

2.2.1 MDL Molfile

MDL molfiles consist of three main sections: a header block (containing title, timestamp and an optional comment), a CTab, and an end line which must read “M END.”[16][17] The CTab consists of a number of sections, best understood by considering **Figure 2.3** .

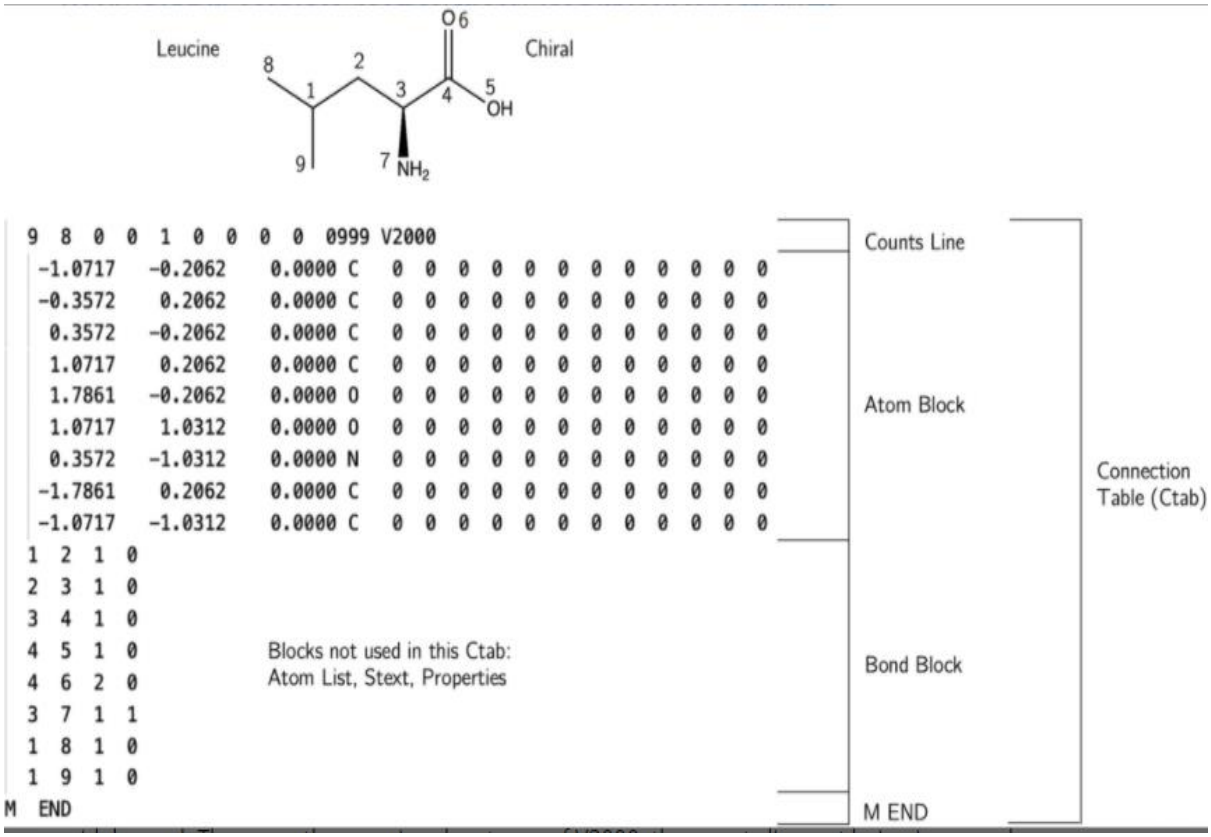


FIGURE 2.3 Example of a connection table and end line within an MDL molfile V2000 for the molecule leucine generated by ChemDraw

MDL molfiles [18] only support single, double, and triple bonds, which is insufficient for molecules with bonds where simple electron sharing in a covalent bond is insufficient. The introduction of a zero-bond order has been suggested as a way to (partially) solve this problem [19] . Furthermore, if the number of hydrogen atoms is not explicitly mentioned, nontrivial valency may result in an incorrect number of assumed hydrogen atoms; this is a serious

problem that is difficult to overcome (simply requiring the number of H-atoms to be specified might compromise back-compatibility).

2.2.2 STRING REPRESENTATIONS

String representations generally consist of characters from the American Standard Code for Information Interchange (ASCII) character encoding standard, and are more compact and easier for humans to read and write than other representations. An example showing various string representations for the molecule dicycloverine hydrochloride (**figure 2.4**) can be seen in **Table 2.1** [18].

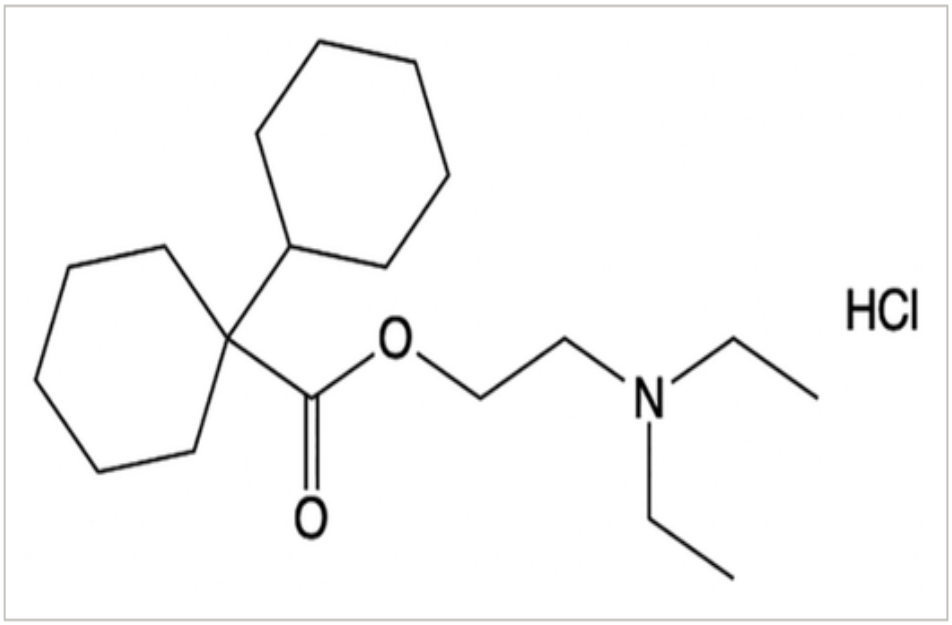


Figure 2.4: Structure graph of dicycloverine hydrochloride

TABLE 2.1 : Various representations of the dicycloverine hydrochloride molecule

Geniric names	dicycloverine HCI , binacole , bentyl , dibent , dyspas , and so on
Mol.formula	$C_{19}H_{36}CINO_2$
IUPAC name	2-(diethylamino)ethyl 1-cyclohexylcyclohexane-1-carboxlate hydrochloride
CAS RN	67-92-5
Canoninal SMILES	CCN(CC)CCOC(=O)C1(CCCCC1)C2CCCCC2.CL
InChel	InChl=1S/C19H35NO2.CIH/c1-3-20(4-2)15-16-22-18(21)19 (13-9-6-10-14-19)17-11-7-5-8-12-17 /h17,3-16H2 ;1H InChlKey :GUBNMFJOJGDCEL-UHFFFAOYSA-N
WLN	L6TJA-AL6TJAVO2N&2&GH

2.2.3 Simplified molecular-input line-entry system (smiles)

SMILES[18] represents (organic) molecules with a string of ASCII characters. Atoms are simply represented with the same one- or two-letter symbol that is used in the periodic table. Single bonds can either be implicit or represented with −, and double, triple, and quadruple bonds with =, #, and $, respectively. Rings are represented with a number after the (arbitrarily chosen) initial atom in the ring and closing atom (e.g., C1CCNCC1 and N1CCCCC1 are equivalent). Branching is represented with parentheses around the branch, for example, 4-ethylheptane (a 7-C backbone with a two-carbon sidechain on the fourth carbon) would be CCCC(CC)CCC. Branches can also be nested within other branches by adding more parenthesis pairs. Aromaticity can be represented with either alternating single/double bonds, or by writing aromatically bonded atoms in lower case. To illustrate this, consider all of these equivalent representations of benzene: c1ccccc1, C1═C_C═C_C═C1, C1═CC═CC═C1.

Although it can be easy to write down a SMILES [20] string that is syntactically correct since there are many ways to write the same thing, this non canonical nature of SMILES [21] can make (sub)structure searches in a database difficult. various algorithms have been developed to canonicalize SMILES [22] strings including Universal SMILES, Rdkit SMILES and CANGEN.

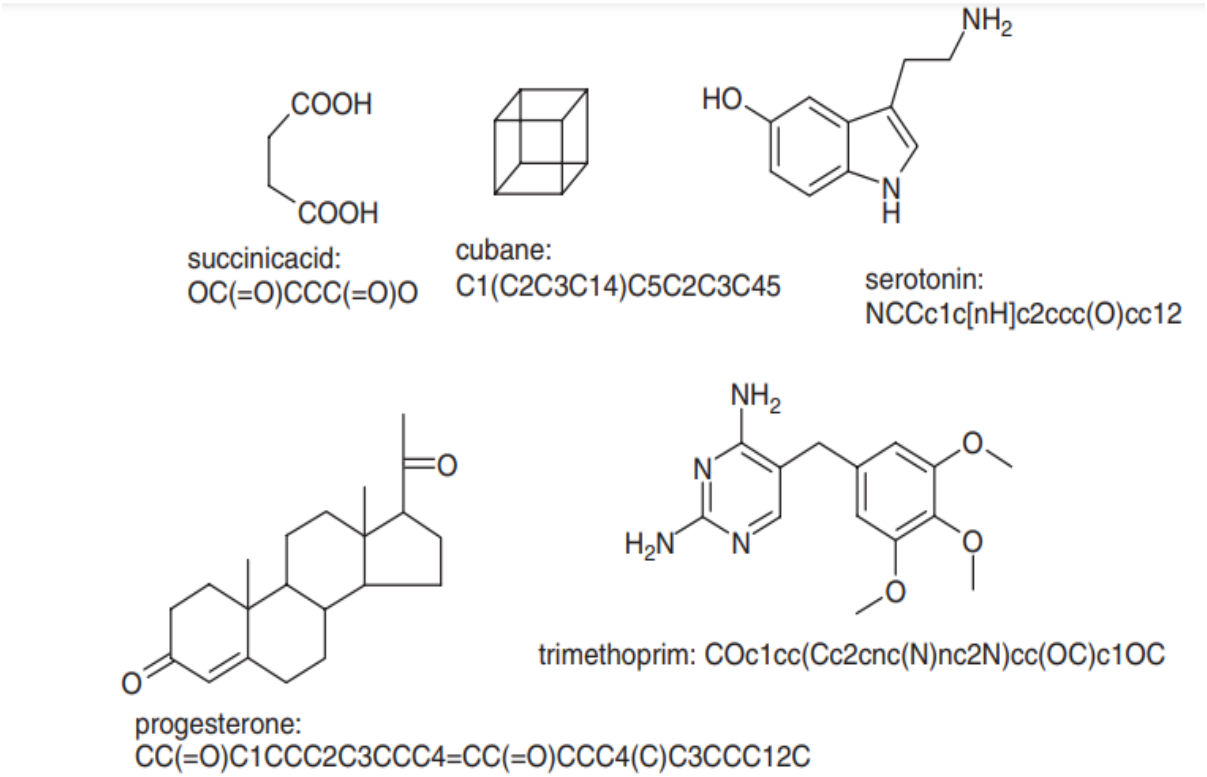


FIGURE 2.5 some examples of the smiles stings for a veriety of moleclule

2.2.4 Extended-connectivity fingerprints

Molecular fingerprints [23] are intended to represent the presence (or absence) of substructures within molecules often in a sparse vector, and they generally fall into one of two categories: matching substructures in a molecule to substructures in an expert-defined set, or algorithmic enumeration and hashing of substructures in a molecule. The ECFP is one of the most widely used chemical fingerprinting techniques due in large part to the popular open-source python package RDKit [24] having an implementation of it, called “Morgan fingerprint.” However, chemical fingerprinting techniques existed before the ECFP, see for example HOSE and FREL. Comparing and contrasting fingerprinting methods can be found in the literature, so this text will instead give a brief overview of how ECFPs are generated and may be used in ML models.

In generating an ECFP is using information about each atom in a molecule to yield a descriptor of the atom and its immediate environment that is invariable with how the atoms in the molecule are numbered [25]. Any set of properties which fulfill these criteria could be used. For ECFPs the property set came from the daylight atomic invariants rule: the number of

connections, number of nonhydrogen bonds, atomic number, sign of charge, absolute charge, and number of attached hydrogens.

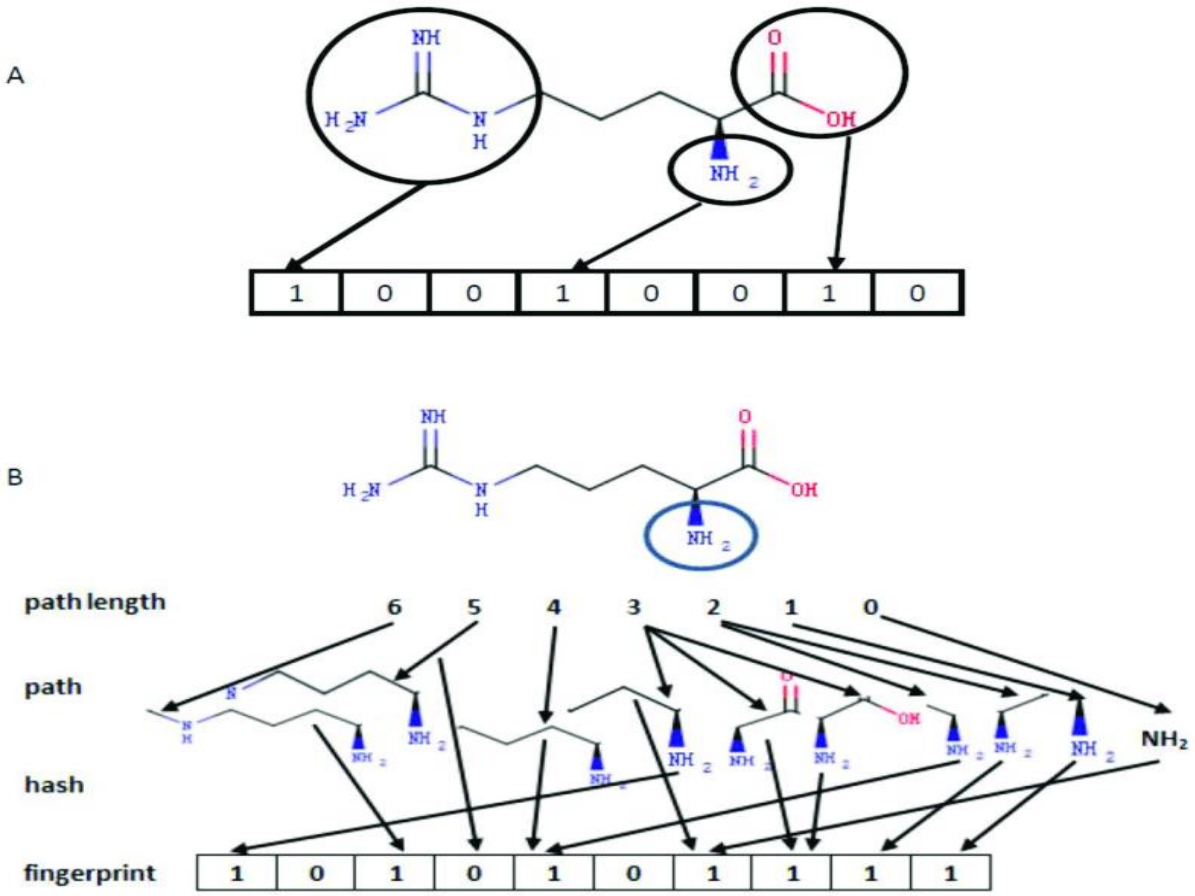


FIGURE 2.6 Example for fingerprint

2.3 Molecular similarity measurement

A degree of symmetry in either analogy or resemblance between two or more concepts or objects is referred to as similarity. The concept of similarity is based on either precise or approximate pattern repeats in the compared objects[11].

Similarity measures quantify the relatedness of two molecules with a large number if their molecular descriptions are closely related and with a small number (large negative or zero) when their molecular descriptions are unrelated. There are many measures available to quantify the degree of similarity between a pair of molecules. The computational requirements of these

measures vary depending on the level of detail used to represent the molecules that are being compared.

2.3.1 Similarity Coefficients

The degree of resemblance between two structures is measured using similarity coefficients, which are numerical values. There are many different types of similarity measurements in use. Edit distance, for example, has been used to compare the similarity of two molecular structures. Edit distance is a string-based measure of the number of operations required to translate the representation of one structure to the representation of another structure.

Distance coefficients, association coefficients, correlation coefficients, and probabilistic coefficients are the four primary forms of similarity coefficients[26]. The similarity coefficients are shown in table 2.2.

The distance between structures in a molecular space is measured using distance coefficients. The agreement between the binary, multi-state, or continuous character representations of two molecules is measured by association coefficients, which are pair-functions. Some association coefficients that have been used to assess compound similarity. The degree of connection between sets of data repercentage the molecules, such as the proportionality and independence between pairs of real-valued molecular descriptors, is measured by correlation coefficients. While not commonly employed in evaluating molecular similarity[27], probabilistic coefficients focus on the distribution of descriptor frequencies across members of a data collection, emphasizing a match on a seldom occurring variable.

Table 2.2 Distance coefficient

Similarity/Distance Coefficient	Formula	
Tanimoto Coefficient	$Tani = \frac{c}{a+b-c}$	(1)
Dice Coefficient	$Dice = \frac{2a}{2a+b+c}$	(2)
Cosine Coefficient	$Cos = \frac{a}{\sqrt{(a+b)(a+c)}}$	(3)
Euclidean Distance	$ED(a,b) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2}$	(4)
Hamming Distance	$HM(a,b) = a+b-2c$	(5)

Each coefficient computes the similarity between two molecular fingerprints, *X* and *Y*, of length *n*, in which *a* is the number of bits set in both *X* and *Y*, *b* is the number of bits set exclusively in *X*, *c* is the number of bits set exclusively in *Y* and *d* is the number of bits set in neither *X* or Y.

Fingerprint Distance Metrics :

- Quantifying similarity or distance (dissimilarity) between two molecular fingerprints
- Common relationship between distance (dissimilarity) and similarity metrics:

distance = 1 - similarity

- Can be applied to continuous or binary variables

Common similarity/distance measure in chemical informatics:

- Similarity metrics:
  - Tanimoto coefficient
  - Dice coefficient
  – Cosine coefficient
- Distance metrics:
  - Euclidean distance
  – Hamming distance

2.3.2 Fingerprint similarity

All of the unique fragments that exist in a molecule are hashed using a hashing technique to fit inside the length of the bit string in hashed fingerprints. Because it does not rely on a pre-existing list of structural components, this method allows for more generalizations. The nature of the chemical structures in the database, rather than the pieces in any specified list, characterizes the fingerprints created. This technique indexes a set of patterns rather to utilizing a fragment dictionary[28].

- Bit string representations of chemical structure and properties
- 2D and/or 3D features of a molecule are typically encoded as a vector of binary values

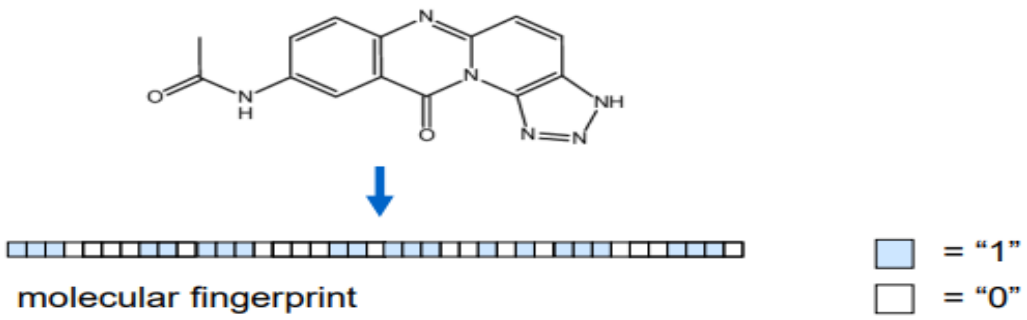


Figure 2.7 A molecular fingerprint representation

- Keyed fingerprints:
  - one bit position is associated with exactly one predefined feature
  - "1": presence of the feature in the molecule
  - "0": absence of the feature in the molecule Hashed fingerprints:
  - one molecular feature is mapped to several bit positions applying a hash function
  - it follows that a bit position is often set to "1" by more than one feature

2.4 Deep learning

Deep learning is a type of machine learning that allows computers to learn from their mistakes and comprehend the world as a hierarchy of concepts[29]. There is no need for a

human computer operator to expressly describe all of the information that the computer requires since the computer learns through experience. A graph of these hierarchies would be many layers deep, allowing the machine to understand sophisticated concepts by constructing them out of simpler ones. This book covers a wide variety of subjects related to deep learning. The work provides a mathematical and conceptual grounding in linear algebra, probability theory and information theory, numerical computation, and machine learning, among other topics. It covers applications such as natural language processing, speech recognition, computer vision, online recommendation systems, bioinformatics, and videogames, as well as deep learning techniques used by industry practitioners, such as deep feed forward networks, regularization, optimization algorithms, convolutional networks, sequence modeling, and practical methodology[30]. Finally, the book provides research views on linear factor models, auto encoders, representation learning, structured probabilistic models, Monte Carlo techniques, the partition function, approximation inference, and deep generative models, among other theoretical subjects.

Undergraduate and graduate students interested in employment in business or research, as well as software developers interested in incorporating deep learning into their products or platforms, can benefit from deep learning. One of these fields is chemistry, which deals with a wide range of extremely difficult challenges such as retro synthesis, reaction optimization, and drug creation. Historically, they have faced intense criticism from chemists, who have viewed computational techniques based on hand coded heuristics and rules with mistrust. These issues are very similar to the ones that deep learning has grown to dominate, such as computer vision and natural language processing. 12 As a result, the adoption of these technologies in chemistry has steadily increased, with many displaying considerable increases in predicted accuracy and capacity to emulate human decision making. Given the widespread use of deep learning in chemistry, it's critical for chemical researchers who aren't aware with the intricacies of the techniques to become acquainted with how they work.

Deep learning in chemistry, explaining the core ideas in the broadest possible sense, without emphasis on mathematical detail, and with reference to chemical applications. This understanding will then be used to provide a broad overview of the influences deep learning has so far had across applied and theoretical chemistry.

2.4.1 Convolution Neural Network (CNN)

CNN sometimes known as ConvNet, is a type of deep feed-forward ANN that is commonly used in visual image processing. It's a well-known DL architecture that was triggered by the active items' anticipated visual sensitivity tool. In the literature, there are other CNN architectural versions[31], including LeNet-5, Alexnet, VGG, GoogleNet, and Deep Residual Learning .However, the underlying components of both CNN [32] variations are comparable. Figure 2 shows the three main layers of the well-known LeNet-5 (convolutional, pooling, and fully-connected).

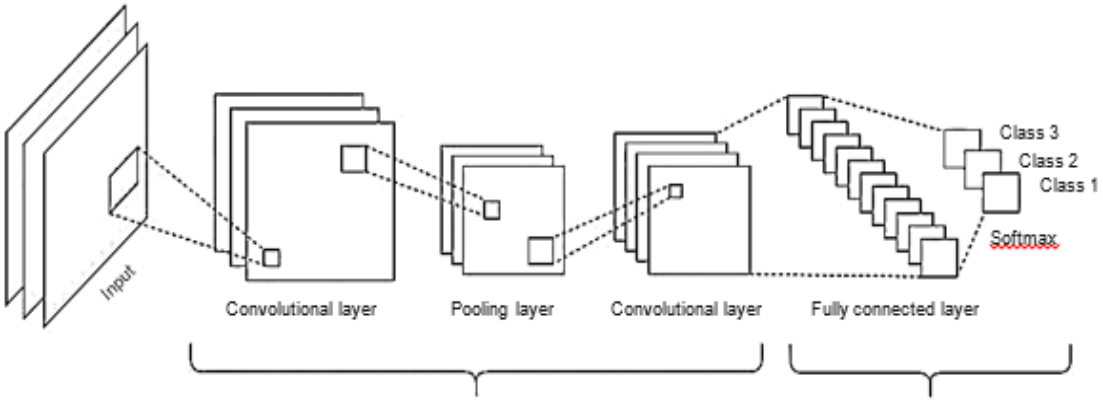


Figure 2.8 The general Convolution Neural Network configuration

Max Pooling Layer typically entails the periodic insertion of a pooling layer in-between consecutive conventional layers in a ConvNet design, as seen in Figure 3.The aim of this layer is to limit the amount of computation and the number of considerations in the network by gradually reducing the spatial size of the representation, hence preventing overfitting. The functioning of the pooling layer is independent of the response's depth slice since it spatially resizes it using the MAX operation. The most frequent type of pooling layer is a pooling layer with 2 2 filters, which is applied with two down samples per complexity portion of the response by two along both width and height, leaving 75% of the activations .

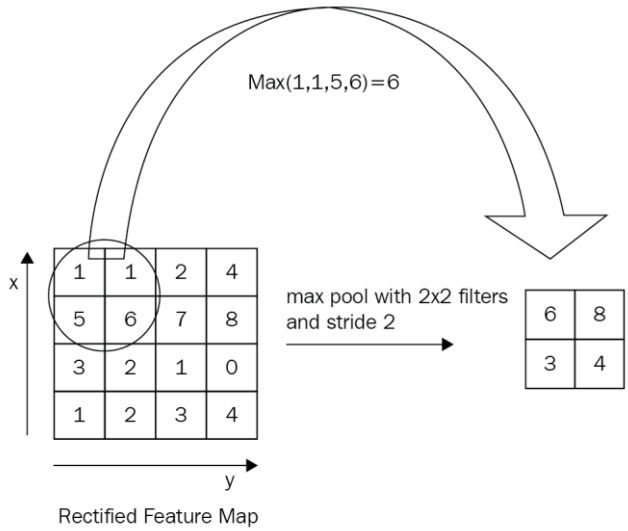


Figure 2.9 The max-pooling operation using 2 × 2 filters

2.4.2 RECURRENT NEURAL NETWORK (RNN)

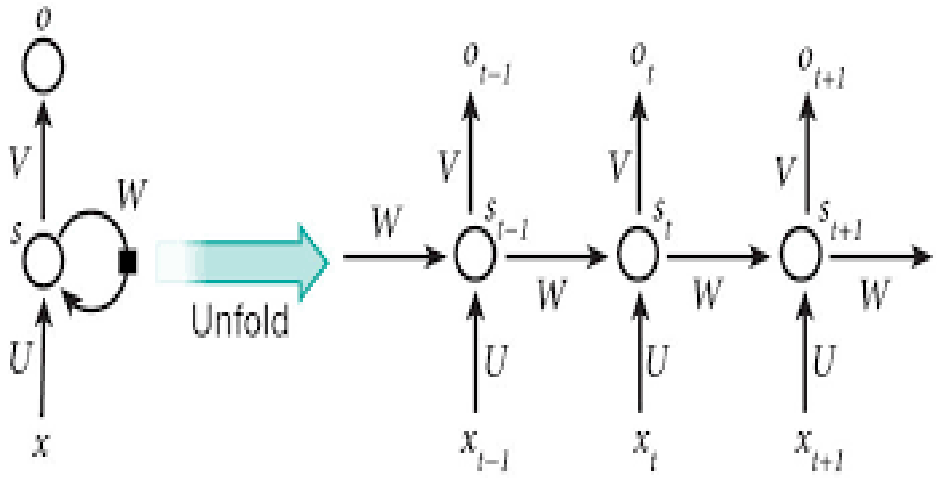


Figure 2.10 A simple Recurrent Neural Network model

x_i: inputs, h_i: hidden states (memory), o_i: outputs, U: input weights, W: hidden weights (recurrent connection), V: output weights.

The RNN, also called the Elman network, has three layers in each period.The design of RNN is illustrated in Figure 4. The three layers in each of its periods are the feedback word layer w, the recurrent layer r, and the output layer y, and the activation of these layers at time t is represented as V(t), r(t), and y(t), respectively. V(t) is the existing word trajectory that can take the form of a simple 1-of-N coding representation h(t) (i.e. the one-hot representation; it

is binary, and its dimension is the same as that of the dictionary size with just one non-zero component). The calculation of Y (t) can proceed thus (Mikolov et al. 2010): x(t) = [w(t) r (t − 1)];r(t)=f1(U.x(t)); y(t) = g1(V. r(t)); here, x (t) represents the concatenation vector for w (t) & r (t-1), while f1 (:) is the element-wise sigmoid & g1 (:) is the softmax function. U & V are the learnable weights. A simple RNN model is typified in Figure 4. In the RNN, the size of the network is a function of the input sequence length. The recurrent layers connect the sub-networks at different time frames. Hence, when executing back propagation, there is a need to ensure the inaccuracy is disseminated back in time via recurrent connections. It's a method of facilitating object instances annotation. In the approach, a polygon that outlines an object is predicted using Polygon-RNN [33], while the corrections are easily incorporated from an annotator in the loop. The system's analysis showed that it achieved an improved annotation speed of up to factor 4.74 and its annotation agreement was the same as that performed by human annotators. This approach is advantageous because it offers structurally plausible object annotations and permits predefined annotation accuracy by the annotator in just a few clicks.

2.4.3 LONG-SHORT-TERM MEMORY (LSTM)

Although traditional RNN finds it difficult to study long-term dynamics due to the issue of exploding and vanishing gradients, the LSTM network was proposed, and the core of its architecture (refer to Figure 2.11) is a memory cell for storing the state of the cells over time; there are also gates for controlling how and when the states of the cells can be updated.

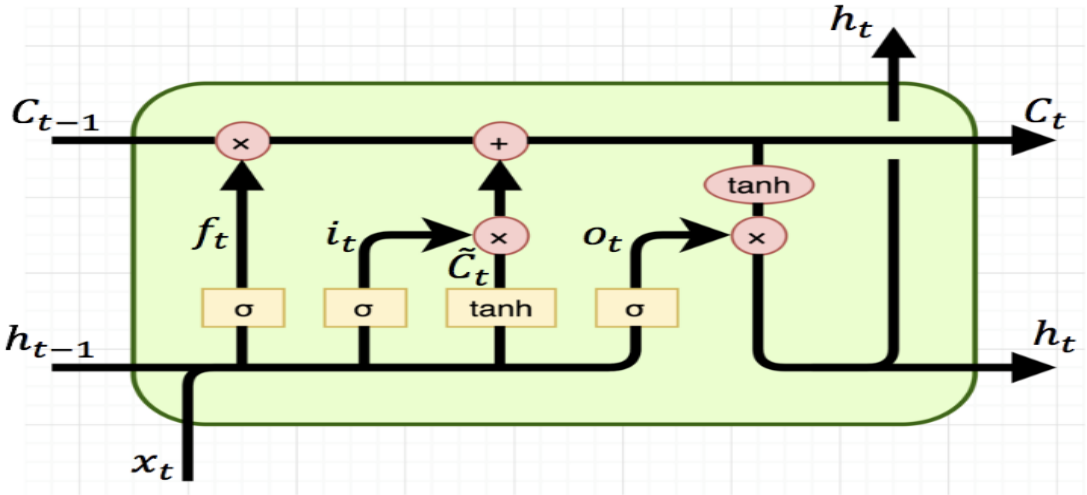


Figure 2.11 The architecture of LSTM

There are various variations of gates and memory cells with varying connections. In the LSTM model[34], word production is mostly determined by the word's embedding at the current time-step and the previous hidden state (this includes the pre-information of the image). This procedure is repeated until the last token of a phrase is encountered. Meanwhile, as the method progresses, the picture data served at the start becomes less relevant. The produced words at the beginning of a series are likewise susceptible to the same problem as a result, generation for a long sentence at the conclusion of the sentence can be done essentially blindly. Despite the LSTM's capacity to preserve long-term memory to some extent, it is still a problem for sentence generation. The LSTM model by introducing semantic data as additional feedback to each LSTM block unit. With this modification, the model can better describe an image's content without diving into the common but unrelated phrases.

2.5 Deep learning model for automatic image annotation

Deep learning [30] is responsible for significant advances in cutting-edge AI-based research including speech recognition, entity recognition, and machine translation. Because of its deep architecture, deep learning might be utilized to handle a variety of difficult AI challenges. As a result, deep learning is now being used to a variety of new jobs and domains, in addition to traditional chores like surface recognition, etymological prototypes, and object detection. For example, a recurrent neural network (RNN) was used to de-noise voice data. The finding of gene expression and clustering patterns using SAE, on the other hand, has been reported. Another study uses a neural network to produce pictures with various styles, while relying on deep learning to allow for simultaneous sentiment analysis across many modalities. Deep learning-related research will grow in popularity in this period.The pragmatic findings suggest that deep learning performs better than conventional machine learning processes. Some have suggested that it's because deep learning can roughly imitate the brain's functions (several layers of neural networks placed one after another, similar to the traditional brain architecture). Deep learning, on the other hand, lacks a robust speculative background. Due to their training on the feature extraction component, deep learning [35] technologies often outperform predictable ML systems. Using deep learning methods, feature hierarchies are taught, with features from the lower hierarchy being combined to produce features from the upper hierarchy. The autonomous learning of features at various abstraction levels will allow for a technique to gain sophisticated functionality, allowing the system to directly translate input to output without relying on human-crafted features .During picture recognition, for example, the

standard procedure is to remove/fetch and input the preset features to SVM. The retrieved characteristics are also optimized in deep learning schemes, which is why deep learning approaches perform better. The rationale for using deep learning is depicted in Figure 2.12 . Deep learning outperformed standard machine learning as data amount increased. Deep learning performance is poor when there is little data since it requires a large amount of data to accomplish perfect learning. Traditional machine learning algorithms, on the other hand, work better with small data due to their handcrafted rules. The figure below summarizes this fact.

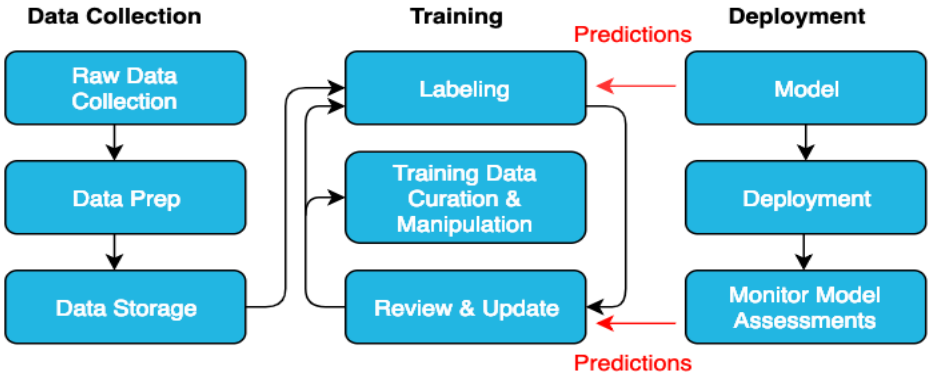


Figure 2.12 A general deep learning model

2.6 Automatic image annotation

Massive volumes of photographs being produced, stored, and posted to the Web for sharing and viewing due to the rapid development of multimedia and networks.

People, for example, are constantly uploading and sharing their photos on websites, posing a dilemma for picture sharing service providers in terms of organizing and retrieving such a big number of images.

Automatic image annotation (AIA) [36] approaches are a type of efficient scheme for bridging the semantic gap between original images and their semantic information.

Traditional annotation models, on the other hand, only operate well with carefully created manual features.

AIA(Automatic Image Annotation) [36] is a technology which automatically annotate the image with its semantic tags. The technology is often applied to the areas such as image retrieval, classification, and the medical domain.

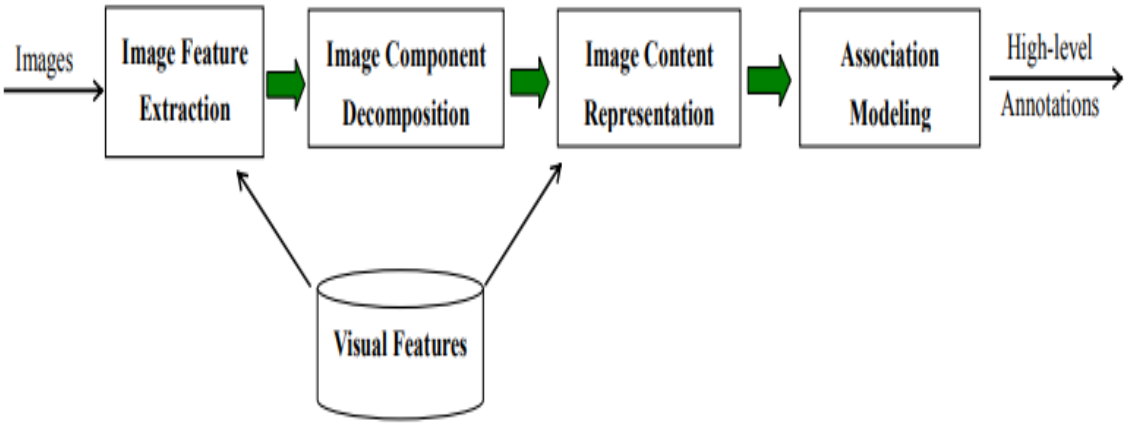


Figure 2.13 A general system framework for AIA

Image features are frequently picked or extracted for use in multimedia processing methods such as recognition, classification, tagging, and so on. Appropriate characteristics should be discriminative and compact, as well as representing the intrinsic substance of the source image.

A variety of strategies were developed to attain these goals. Some existing studies use latent subspace to reveal the knowledge that exists between visual content and semantic information.

2.7 Discussion

Molecules are complex structures and their representations must account not only for a wide variety of properties, such as stereochemistry and valence, but also for the different nature of these small molecules and macromolecules. The rise of cheminformatics and bioinformatics has led to a faster and more efficient drug discovery process as well as to a better understanding of molecular behaviour. In this review, we presented various popular notations and representations for small molecules, polymers, and proteins and their most common uses related to AI within computational drug discovery. We hope that this review will benefit practising cheminformaticians, students, and anyone else interested to learn more about the

underlying molecular representations in cheminformatics that can be used in AI-driven drug discovery applications.

This chapter presents a review of various methods to AIA based on deep learning. The reviewed approaches are Convolution neural network (CNN)[13] based on AIA, , Long-Short-Term Memory (LSTM) based on AIA, An assessment of the five types of AIA methods was offered based on their original idea, feature mining technique, annotation correctness, computational complexity, and datasets. Moreover, the assessment metrics used to assess AIA methods were reviewed and the advantages and issues associated with each technique were explained. The major challenge of the AIA techniques based on deep learning is related to high-dimensional feature analysis. All the existing features currently have the problem of not sufficiently describing the images. There is no existing feature that is significant enough to represent the high variation between images efficiently. The AIA technique's challenge is to reduce the semantic gap between low-level visual image features captured by machines and high-level semantic concepts perceived by humans. Many studies have been conducted on mining the image-image, image-label, and label-label correlation. Open issues, such as class-imbalance and weak-labeling of the training dataset. Finally, some of the research on image Annotation systems is inclined to accomplish high accuracy and squat recall; the strength of image annotation is to ensure a balance between precision and recall by making sure the recall scores improve without maintaining precision. On the other hand, some image annotation models take a long time and computational complexity in the training phase, making them computationally intensive when faced with large training datasets. It is necessary to highlight the need for careful consideration of these aspects when building new image annotation techniques and datasets for future tasks.

2.8 Summary

The use of machine learning in molecular/material research is quickly going beyond the traditional technique. ML molecular descriptors are still widely used today, whereas DL machine learning features are also widely employed and have a good success rate. However, one of the most important features of DL models, the underlying fundamental principles, has yet to be discovered. As a result, it's only useful in molecular research, where physical and chemical complicated factors play a key part. In the case of solids, however, the atomic environment descriptor is combined with the cut off distance to restrict the number of surrounding atoms that are utilized to construct the representation. Another important issue is

to take use of the task's invariances. Molecules, for example, may be rotated and moved in space without modifying their characteristics, therefore predictions are unaffected. Scalability, transferability, and data-size-extensivity are some well-known ML science difficulties. The principles, models, algorithms, and challenges linked with the types of DL based on AIA methodologies were covered in the preceding sections. The capacity to understand complicated associations, create powerful features, and eliminate the need for manual selection are all advantages of DL-based AIA approaches, according to the researchers.

CHAPTER 3
RESEARCH METHODOLOGY

Introduction

This chapter presents the methods that will be used molecule description. The methodology employed will provide a systematic framework of procedures and principles to achieve the objective of this study. ." In this chapter, you'll learn how to utilize deep neural network models to automatically produce descriptions for molecules like molecules. After reading this article, you will be aware of the following:

The difficulty of developing written descriptions for molecules and the necessity to integrate computer vision and natural language processing advancements. About the elements that comprise a neural feature captioning model, namely the feature extractor and language model.

3.1 Research Framework

The study methods will be conducted according to the workflow process.

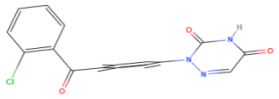
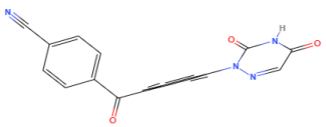
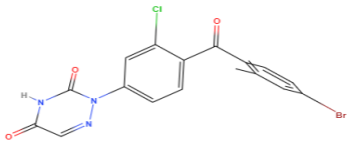
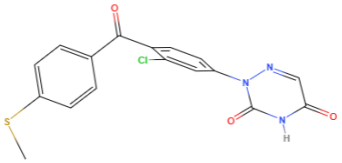
Methodology implementation phases:

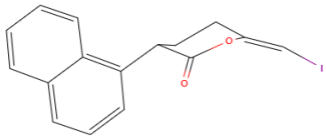
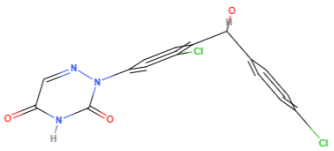
- **phase 1**
    - Literature review on molecular textual description
- **phase 2**
    - Analysing and preparing molecule datasets
    - Identifying and calculate fingerprint for molecular inside the data
- **phase 3**
    - Develop deep learning model to generate textual description for molecule
    - Training and testing description generation models
- **phase 4**
    - Test and evaluate the model
    - Generate new captain model

3.2 Dataset

ChEMBL is a massive, open-access drug discovery database that intends to gather Medicinal Chemistry data and expertise throughout the pharmaceutical development process[8]. The database's practical applications include finding chemical tools for a target of interest, assessing compound selectivity, training machine learning models (e.g. for target prediction), assisting in drug repurposing hypotheses, assessing target tractability, and integrating with other drug discovery resources. The capacity to focus on the infrastructural components of the data, as well as the varied dynamic that occurs between giant pharma, SMEs [Small and Medium Enterprises], and academia, has been a beneficial development for the group's work.

Table 3.1 example of the dataset

Description	SMILES	Molecule
The compound was tested for the in vitro inhibition of platelet 12-lipoxygenase at a concentration of 30 uM	[N+](C)(C)(Cc1ccccc1)CCCCCCCCCC.[Cl-]	
Compound was evaluated for its ability to mobilize calcium in 1321NI cells	Cc1cc(-n2ncc(=O)[nH]c2=O)ccc1C(=O)c1ccc(C#N)cc1	
Cytotoxic Activity was evaluated against 143B (TK-) tumor cells	Cc1cc(Br)ccc1C(=O)c1ccc(-n2ncc(=O)[nH]c2=O)cc1Cl	
In vitro cell cytotoxicity against 143B-LTK cell lines expressed in HSV-1 TK	CSc1ccc(C(=O)c2ccc(-n3ncc(=O)[nH]c3=O)cc2Cl)cc1	

The compound was tested for minimum inhibitory concentration in Staphylococcus aureus,147N penicillin sensitive Staphylococcus aureus	O=C1O/C(=C\I)CCC1c1cccc2ccccc12	
(Concentration that cause 50% activation of human reticulocyte 15-lipoxygenase (15-HLO)	O=c1cnn(-c2ccc(C(O)c3ccc(Cl)cc3)c(Cl)c2)c(=O)[nH]1	

3.3 Design and steps

Captioning a molecule entails creating a human-readable written description for an object, such as a molecule. It's a simple problem for a person to solve, but it's a difficult challenge for a computer to solve since it requires both comprehending the content of a molecule and translating that knowledge into intelligible language. Deep learning algorithms have recently overtaken traditional methods and are attaining state-of-the-art results for the task of automatically creating descriptions for molecules, known as captions

3.3.1 Molecule and Caption Dataset

We provide a dataset collection for sentence-based molecule description, which consists of 999 molecules (smile) each coupled with one distinct caption that provides clear explanations of molecule.

- **Dataset.Sdf** An archive of all molecules.
- **description.txt** An archive of all text descriptions for molecules.

3.3.2 Prepare molecule fingerprint Data

PaDEL-Descriptor is a software for calculating molecular descriptors and fingerprints. PaDEL-Descriptor was developed using the Java language and consists of a library component and an interface component. The library component allows it to be easily integrated into quantitative structure activity relationship software to provide the descriptor calculation feature while the interface component allows it to be used as standalone software. The software uses a Master/Worker pattern to take advantage of the multiple CPU cores that are present in most

modern computers to speed up calculations of molecular descriptors. The software has several advantages over existing standalone molecular descriptor calculation software. It is free and open source, has both graphical user interface and command line interfaces, can work on all major platforms (Windows, Linux, MacOS), supports more than 90 different molecular file formats, and is multithreaded. PaDEL-Descriptor is a useful addition to the currently available molecular descriptor calculation software.

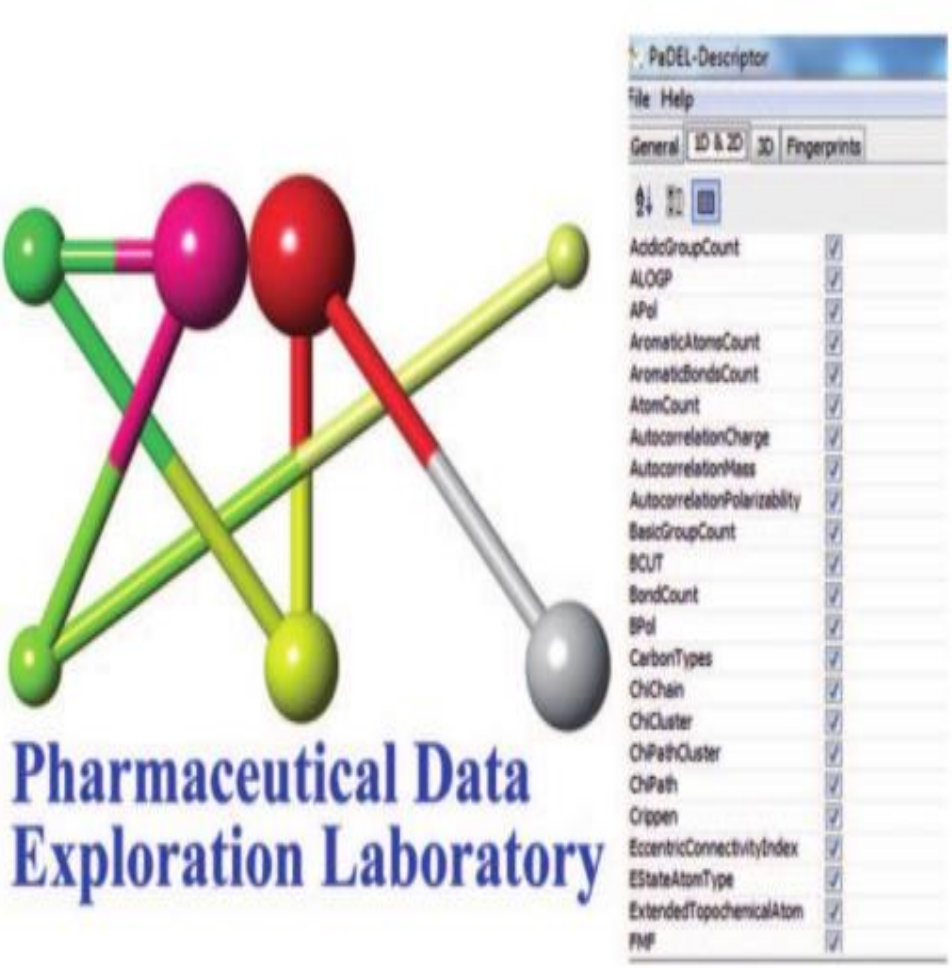


Figure 3.2 PaDEL descriptor calculator graphical user interface

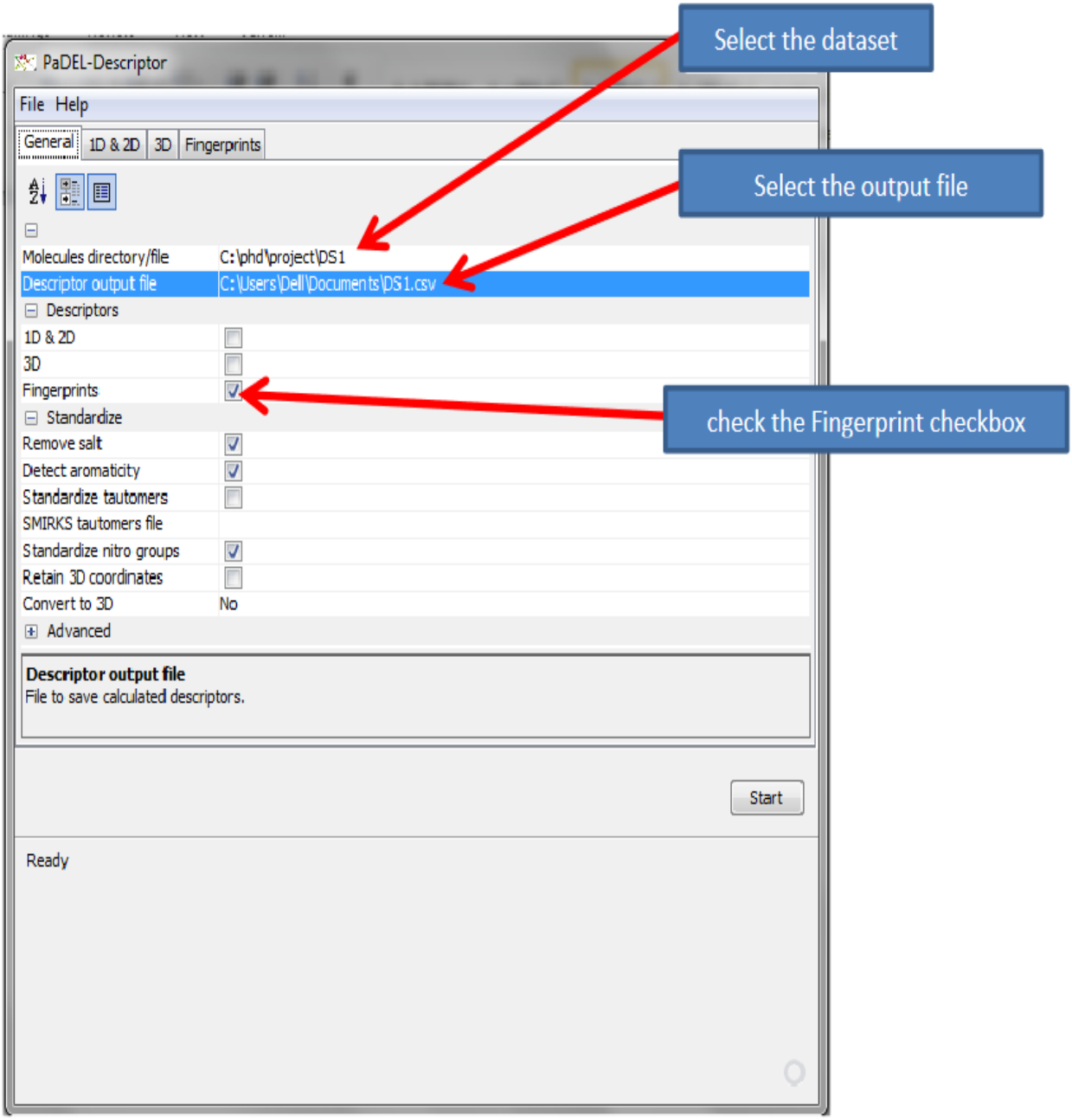


Figure 3.3 PaDEL descriptor calculator graphical user interface

The PaDEL software currently calculates 1875 descriptors (1444 1D, 2D descriptors and 431 3D descriptors)

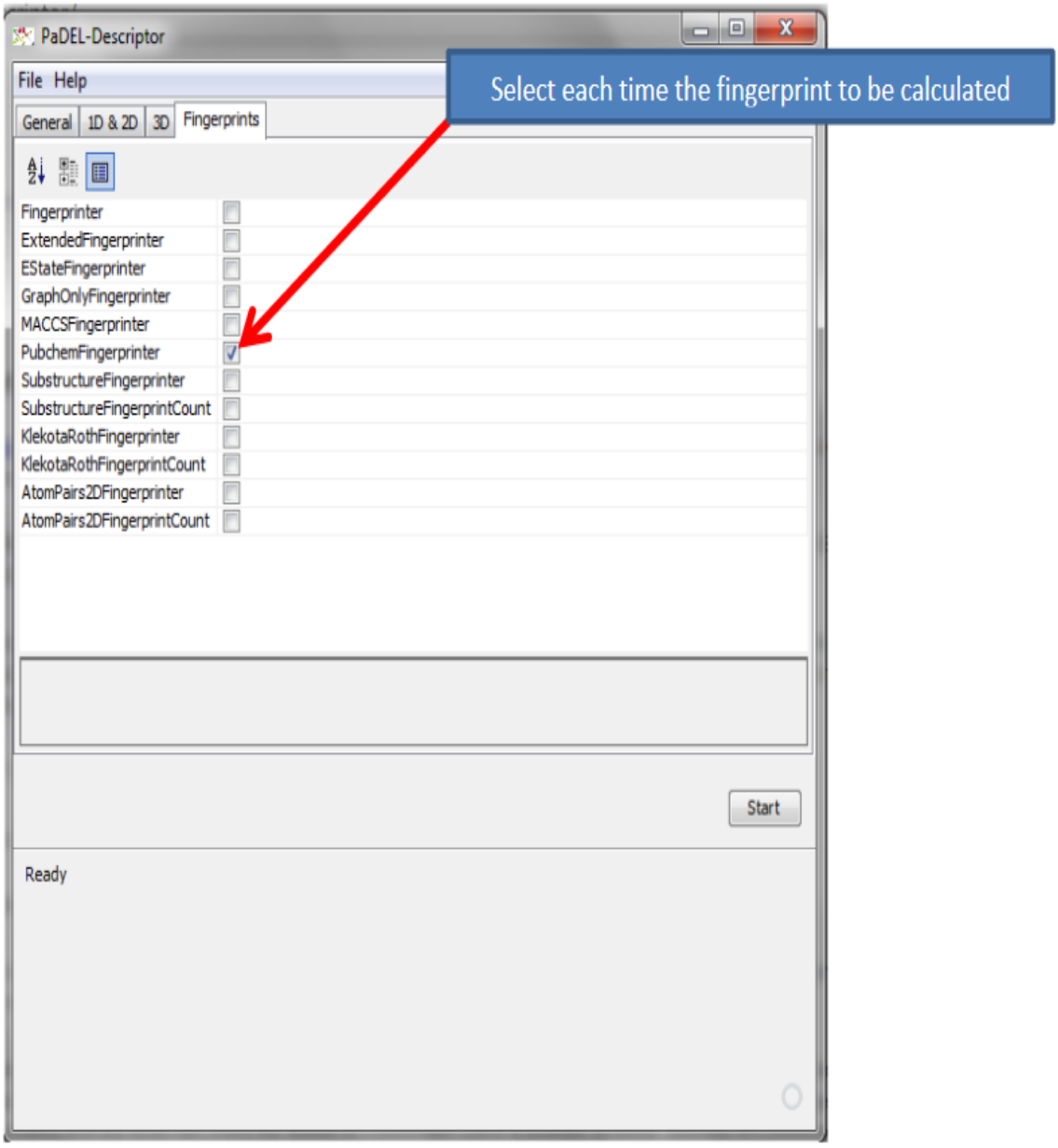
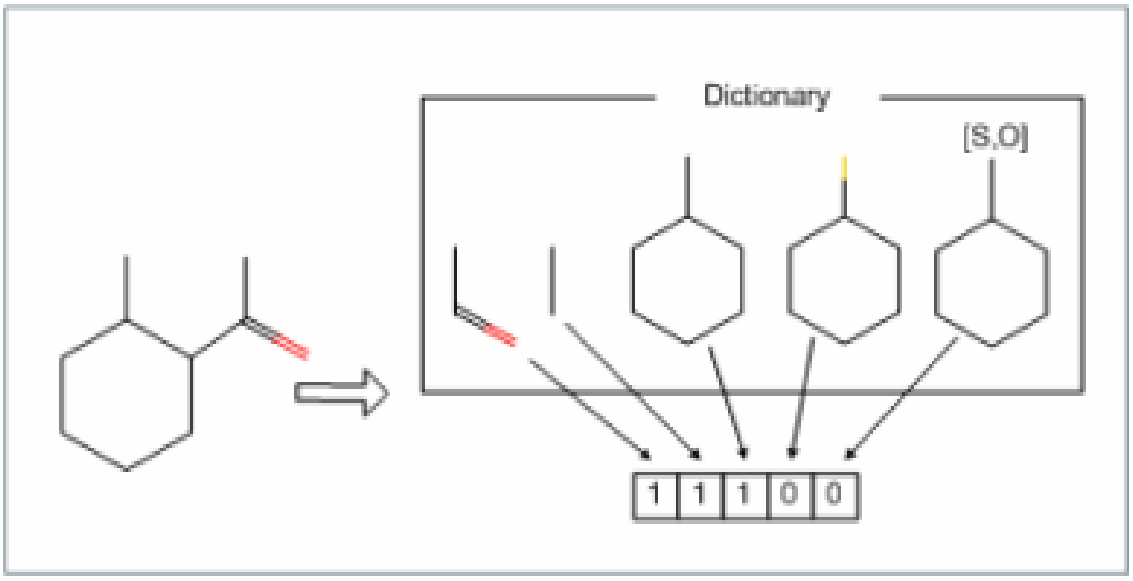


Figure 3.4 PaDEL descriptor: calculate the 12 fingerprint interface

The PaDEL software currently calculates 12 types of fingerprints (total 16092 bits) show in the picture and represented in the table below

Table 3.2 The 12 selected fingerprint used in this research

Fingerprint	**Number**	**Remarks**	**Description**
CDK fingerprint	1024	FP	Fingerprint of length 1024 and search depth of 8
CDK extended fingerprint	1024	ExtFP	Extends the Fingerprinter with additional bits describing ring features
Estate fingerprint	79	EStateFP	E-State fragments
CDK graph only fingerprint	1024	GraphFP	Specialized version of the Fingerprinter which does not take bond orders into account
MACCS fingerprint	166	MACCSFP	MACCS keys
Pubchem fingerprint	881	PubchemFP	Pubchem fingerprint
Substructure fingerprint	307	SubFP	Presence of SMARTS Patterns for Functional Group Classification by Christian Laggner
Substructure fingerprint count	307	SubFPC	Count of SMARTS Patterns for Functional Group Classification by Christian Laggner
Klekota-Roth fingerprint	4860	KRFP	Presence of chemical substructures
Klekota-Roth fingerprint count	4860	KRFPC	Count of chemical substructures
2D atom pairs	780	AP2D	Presence of atom pairs at various topological distances
2D atom pairs count	780	APC2D	Count of atom pairs at various topological distances



	1	2	3						1023	1024
1	1	1	0	0	0	1	0	1	1	0
2	0	0	0	1	0	0	0	0	0	1
	1	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	1
998	0	0	0	0	0	0	0	0	1	0
999	1	1	0	0	0	0	0	1	0	0

Figure 3.5 Example for Generate fingerprint matrix from molecule smile

3.3.3 Prepare Text Data

The dataset contains multiple descriptions for each molecule and the text of the descriptions requires some minimal cleaning.

Clean Text for Machine Learning with Python

First, we will load the file containing all of the descriptions.

```
data = pd.read_csv('C:/Users/hp-01/Desktop/draw molecule/Data.csv')


#description      canonical_smiles
Smiles =[]
Descripe =[]
for index, row in data.iterrows():
  Smiles.append(row['canonical_smiles'])
  Descripe.append(row['description'])


Smiles = np.array(Smiles)
Descripe = np.array(Descripe)

print(len(Smiles))
print(len(Descripe))
```

Each molecule has a unique identifier. This identifier is used on the molecule chembl_id and in the text file of descriptions.

After that, we'll go over the list of molecular descriptions one by one. The following provides the method load_descriptions(), which returns a dictionary of molecular IDs to descriptions given the loaded document content. A collection of one or more textual descriptions corresponds to each molecular identification.

```
# extract descriptions for molecules
def load_descriptions(doc):
    mapping = dict()
    # process line
    i = 0
    for line in doc:
        if i not in mapping:
            mapping[i] = list()
        mapping[i].append(line)
        i = i+1
    return mapping


descriptions = load_descriptions(Describe)
print('Loaded: %d ' % len(descriptions))
```

The description wording must then be cleaned. The descriptions have already been tokenized and are simple to use.

We will clean the text in the following ways in order to reduce the size of the vocabulary of words we will need to work with:

- Convert all words to lowercase.
- Remove all punctuation.
- Remove all words that are one character or less in length (e.g. ‘a’).
- Remove all words with numbers in them.

Below defines the clean_descriptions() function that, given the dictionary of molecule identifiers to descriptions, steps through each description and cleans the text.

```
def clean_descriptions(descriptions):
    # prepare translation table for removing punctuation
    table = str.maketrans('', '', string.punctuation)
    for key, desc_list in descriptions.items():
        for i in range(len(desc_list)):
            desc = desc_list[i]
            # tokenize
            desc = desc.split()
            # convert to lower case
            desc = [word.lower() for word in desc]
            # remove punctuation from each token
            desc = [w.translate(table) for w in desc]
            # remove hanging 's' and 'a'
            desc = [word for word in desc if len(word)>1]
            # remove tokens with numbers in them
            desc = [word for word in desc if word.isalpha()]
            # store as string
            desc_list[i] =  ' '.join(desc)

clean_descriptions(descriptions)
```

We may summarize the amount of the vocabulary once it has been cleansed.

In an ideal world, we'd have a language that's both expressive and compact. A smaller vocabulary means a smaller model that can be trained more quickly.

To gain an indication of the extent of our dataset vocabulary, we may convert the clean descriptions into a set and publish its size.

```
 # convert the loaded descriptions into a vocabulary of words
def to_vocabulary(descriptions):
    # build a list of all description strings
    all_desc = set()
    for key in descriptions.keys():
        [all_desc.update(d.split()) for d in descriptions[key]]
    return all_desc


# summarize vocabulary
vocabulary = to_vocabulary(descriptions)
print('Vocabulary Size: %d' % len(vocabulary))
```

Finally, with one molecule identification and description per line, we may save the dictionary of molecule identities and descriptions to a new file entitled des.txt.

The save_descriptions () method saves the mapping of IDs to descriptions to file, provided a dictionary holding the mapping and a filename.

```
# save descriptions to file, one per line
def save_descriptions(descriptions, filename):
    lines = list()
    for key, desc_list in descriptions.items():
        for desc in desc_list:
            lines.append(str(key) + ' ' + desc)
    data = '\n'.join(lines)
    file = open(filename, 'w')
    file.write(data)
    file.close()


save_descriptions(descriptions, 'des.txt')
```

3.3.4 Develop Deep Learning Model

In this section, we will define the deep learning model and fit it on the training dataset.

This section is divided into the following parts

3.3.4.1 Loading Data

First, we must load the prepared smile molecule and text data so that we can use it to fit the model.

We are going to train the data on all of the molecules and captions in the training dataset. While training, we are going to monitor the performance of the model on the development dataset and use that performance to decide when to save models to file.

The train and development dataset have been predefined in the trainmol.txt and devmol.txt files respectively, that both contain lists of molecule file smiles. From these smiles, we can extract the molecule identifiers and use these identifiers to filter molecules and descriptions for each set.

The function load_set() below will load a pre-defined set of identifiers given the train or development sets filename.

```
# load a pre-defined
def load_set(filename):
    doc = load_doc(filename)
    dataset = list()
    # process line by line
    for line in doc.split('\n'):
        # skip empty lines
        if len(line) < 1:
            continue
        # get the molecule identifier
        identifier = line.split('.')[0]
        dataset.append(identifier)
    return set(dataset)
```

Now, we can load the moluceles and descriptions using the pre-defined set of train or development identifiers.

Below is the function load_clean_descriptions() that loads the cleaned text descriptions from ‘descriptions.txt‘ for a given set of identifiers and returns a dictionary of identifiers to lists of text descriptions.

The model we will develop will generate a caption given a molecule, and the caption will be generated one word at a time. The sequence of previously generated words will be provided as input. Therefore, we will need a ‘first word’ to kick-off the generation process and a ‘last word‘ to signal the end of the caption.

We will use the strings ‘startseq‘ and ‘endseq‘ for this purpose. These tokens are added to the loaded descriptions as they are loaded. It is important to do this now before we encode the text so that the tokens are also encoded correctly.

```
# load clean descriptions into memory
def load_clean_descriptions(filename):
    # load document
    doc = (filename)
    descriptions = dict()
    for line in doc.split('\n'):
        # split line by white space
        tokens = line.split()
        # split id from description
        mol_id, mol_desc = tokens[0], tokens[1:]
        # skip molecules not in the set

        if image_id not in descriptions:
            descriptions[mol_id] = list()
            # wrap description in tokens
        desc = 'startseq ' + ' '.join(mol_desc) + ' endseq'
            # store
        descriptions[mol_id].append(desc)
    return descriptions
```

Before it can be submitted to the model as input or compared to the model's predictions, the description language must be encoded to numbers.

The first step in encoding data is to establish a consistent mapping between words and unique integer values. The Tokenizer class in Keras may be used to learn this mapping from the imported description data.

The to_lines() method converts a dictionary of descriptions into a list of strings, while the create tokenizer() function creates a Tokenizer based on the imported molecular description text.

```
def to_lines(descriptions):
    all_desc = list()
    for key in descriptions.keys():
        [all_desc.append(d) for d in descriptions[key]]
    return all_desc

# fit a tokenizer given caption descriptions
def create_tokenizer(descriptions):
    lines = to_lines(descriptions)
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(lines)
    return tokenizer

# prepare tokenizer
tokenizer = create_tokenizer(train_descriptions)
vocab_size = len(tokenizer.word_index) + 1
print('Vocabulary Size: %d' % vocab_size)
```

The text can now be encoded.

Every description will be broken down into words. One word and a molecule will be given to the model, which will then produce the next word. The model will then be given the first two words of the description as input, and the molecule will produce the following word. The model will be trained in this manner.

```
X1,    X2 (text sequence),    y (word)
mol startseq,
mol startseq,       ,
mol startseq,       ,     ,
mol startseq,       ,     ,       ,
mol startseq,       ,     ,       ,   ,
mol startseq,       ,     ,       ,   ,  , endseq
```

The created words will be concatenated and recursively supplied as input to generate a caption for a molecule later, when the model is employed to generate descriptions.

Given the tokenizer, a maximum sequence length, and a dictionary with all descriptions and molecules, the function create_sequences() transforms the data into input-output pairs for training the model. The model takes two input arrays: one for molecular fingerprints and another for encoded text. The model only has one output, which is the next encoded word in the text sequence.

The input text is converted to integers and fed into a word embedding layer. A prediction will be generated by the model, which will be a probability distribution over all terms in the vocabulary.

As a result, the output data will be a one-hot encoded version of each word, which will reflect an idealistic probability distribution with 0 values at all word locations except the real word position, which will have a value of 1.

```
def create_sequencess(tokenizer, max_length, desc_list, vocab_size, fingerprint):
    X1, X2, y = list(), list(), list()
    # walk through each description for the molecule
    h =0
    for desc in desc_list:
        seq = tokenizer.texts_to_sequences([desc])[0]
        for i in range(1, len(seq)):
            in_seq, out_seq = seq[:i], seq[i]
            in_seq = pad_sequences([in_seq], maxlen=max_length)[0]
            out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]
            print(out_seq)
            X1.append(fingerprint[h])
            h=h+1
            X2.append(in_seq)
            y.append(out_seq)
    return array(X1), array(X2), array(y)
```

We'll need to figure out how many words the longest description can include. Below is a tiny auxiliary function called max_length().

```
# calculate the length of the description with the most words
def max_length(descriptions):
    lines = to_lines(descriptions)
    return max(len(d.split()) for d in lines)
```

We now have enough data to load the training and development datasets, as well as turn the loaded data into input-output pairs for deep learning model fitting.

3.3.4.2 Defining the Model

We will describe the model in two parts:

- **Feature Extractor :** This is a model that has been pre-trained on the molecule dataset. We used the retrieved characteristics predicted by this model as input after pre-processing the molecule using the fingerprint.
- **Sequence Processor :** is a program that allows you to create a sequence of events This is a word embedding layer for processing text input, followed by a recurrent neural network layer using Long Short-Term Memory (LSTM).
- **Decoder :** (for lack of a better name). Both the feature extractor and sequence processor output a fixed-length vector. These are merged together and processed by a Dense layer to make a final prediction.

The Feature Extractor model expects a vector of 1024 items as input molecule (smile) characteristics. A Dense layer transforms them into a 256-element representation of the molecule.

The Sequence Processor model needs input sequences of a certain length (34 words) to be fed into an Embedding layer that ignores padded values using a mask. After that, there's an LSTM layer with 256 memory units.

A 256-element vector is produced by both input models. In addition, both input models employ regularization in the form of a 50% dropout. Because this model configuration learns relatively quickly, this is done to avoid overfitting the training dataset.

```
 # define the captioning model
def define_model(vocab_size, max_length):
     # feature extractor model
    inputs1 = Input(shape=(32,32,3))
    fe1 = Dropout(0.5)(inputs1)
    fe2 = Dense(256, activation='relu')(fe1)
     # sequence model
    inputs2 = Input(shape=(max_length,))
    se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
    se2 = Dropout(0.5)(se1)
    se3 = LSTM(256)(se2)
    # decoder model
    decoder1 = add([fe2, se3])
    decoder2 = Dense(256, activation='relu')(decoder1)
    outputs = Dense(vocab_size, activation='softmax')(decoder2)
    # tie it together [smile, seq] [word]
    model = Model(inputs=[inputs1, inputs2], outputs=outputs)
    model.compile(loss='categorical_crossentropy', optimizer='adam')
    # summarize model
    print(model.summary())
    plot_model(model, to_file='model.png', show_shapes=True)
    return model
```

3.3.4.3 Fitting the Model

We can now fit the model to the training dataset given that we know how to define it.

The model learns quickly and overfits the training data soon. As a result, we'll keep an eye on the trained model's performance on the holdout development dataset. We will save the entire

model to file when the model's skill on the development dataset improves at the conclusion of an epoch.

We may then utilize the stored model with the best skill on the training dataset as our final model at the conclusion of the run.

This may be accomplished by establishing a ModelCheckpoint in Keras and instructing it to monitor the minimal loss on the validation dataset, as well as saving the model to a file that contains both the training and validation losses.

```
# define checkpoint callback
filepath = 'model-ep{epoch:03d}-loss{loss:.3f}-val_loss{val_loss:.3f}.h5'
checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min')
```

We can then specify the checkpoint in the call to fit() via the callbacks argument. We must also specify the development dataset in fit() via the validation_data argument.

We will only fit the model for 20 epochs, but given the amount of training data, each epoch may take 30 minutes on modern hardware.

```
# fit model
model.fit([X1train, X2train], ytrain, epochs=20, verbose=2, callbacks=[checkpoint], validation_data=([X1test, X2test], ytest
```

3.3.5 Evaluate Model

After the model has been fitted, we may assess the accuracy of its predictions using the holdout test dataset.

We'll test a model by creating descriptions for all of the molecules in the test dataset and using a standard cost function to assess the predictions.

To begin, we must be able to use a trained model to provide a description for a molecule.

This requires sending in the start description token 'startseq,' creating one word, and then repeatedly running the model with created words as input until the end of sequence token 'endseq' or the maximum description length is achieved.

This behavior is implemented by the function generate_desc(), which creates a textual description from a trained model and a produced molecule as input. It uses the word_for_id() function to convert an integer prediction to a word.

```
 # map an integer to a word
def word_for_id(integer, tokenizer):
    for word, index in tokenizer.word_index.items():
        if index == integer:
            return word
    return None

# generate a description for an molecule
def generate_desc(model, tokenizer, mol, max_length):
    # seed the generation process
    in_text = 'startseq'
    # iterate over the whole length of the sequence
    for i in range(max_length):
        # integer encode input sequence
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        # pad input
        sequence = pad_sequences([sequence], maxlen=max_length)
        # predict next word
        yhat = model.predict([mol,sequence], verbose=0)
        # convert probability to integer
        yhat = argmax(yhat)
        # map integer to word
        word = word_for_id(yhat, tokenizer)
        # stop if we cannot map the word
        if word is None:
            break
        # append as input for generating the next word
        in_text += ' ' + word
        # stop if we predict the end of the sequence
        if word == 'endseq':
            break
    return in_text
```

We will generate predictions for all molecule in the test dataset and in the train dataset.

The function below named evaluate_model() will evaluate a trained model against a given dataset of molecule descriptions. The actual and predicted descriptions are collected and evaluated collectively using the corpus BLEU score that summarizes how close the generated text is to the expected text.

BLEU scores are one metric that may be used to assess the model's ability.

```
# evaluate the skill of the model
def evaluate_model(model, descriptions, mols, tokenizer, max_length):
    actual, predicted = list(), list()
    # step over the whole set
    for key, desc_list in descriptions.items():
        # generate description
        yhat = generate_desc(model, tokenizer, mols[key], max_length)
        # store actual and predicted
        references = [d.split() for d in desc_list]
        actual.append(references)
        predicted.append(yhat.split())
    # calculate BLEU score
    print('BLEU-1: %f' % corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))
    print('BLEU-2: %f' % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
    print('BLEU-3: %f' % corpus_bleu(actual, predicted, weights=(0.3, 0.3, 0.3, 0)))
    print('BLEU-4: %f' % corpus_bleu(actual, predicted, weights=(0.25, 0.25, 0.25, 0.25)))
```

BLEU scores are used in text translation for evaluating translated text against one or more reference translations.

Here, we compare each generated description against all of the reference descriptions for the molecule. We then calculate BLEU scores for 1, 2, 3 and 4 cumulative n-grams.

All of this may be used with the routines from the previous section for data loading. To build a Tokenizer so that created words may be encoded as input sequences for the model, we must first load the training dataset. It's crucial that we utilize the same encoding strategy for the produced words as we used for the model's training.

We then use these functions for loading the test dataset.

3.4 Summary

This chapter provided extensive methodology of Deep Learning Model to describe molecule, it has four Phases, starting from the first phase which has: collecting the chembl data, Second phase is defining the model of transforming the molecule smile to molecule fingerprint. Third, develop the molecular description model. Finally, we evaluted the model, each stage of these stages plays an important role in accomplishing this study.

CHAPTER 4
EVALUATION AND TEST

Introduction

In this chapter we will present and discuss the different approaches we implemented it, starting by describing the Dataset, preprocessing the data, data augmentation and the environment and tools we used in our models architecture with our modifications and improvement that have been added, compiling, training and validating the model, and last but not least, a conclusion following a discussion of results.

4.1 Environment

We used Python programming language in our experiment. Python is an easy to learn, powerfulprogramming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid applicationdevelopment in many areas on most platforms.

To conduct our experiment, we used the following modules:

Anaconda

Anaconda is a free program that gives you access to a toolbox designed for research and science. Installing Anaconda offers you access to a variety of environments that let you work in Python or R. These environments, often known as integrated development environments (IDEs), are platforms or programs that make coding easier. For writing text, they are akin to text processors like Microsoft Word, Google Doc, and Pages, but they are much more. Many important capabilities are available in IDEs for writing, editing, and debugging code, visualizing and inspecting data, storing variables, presenting results, and collaborating on projects. The programming language does not differ from the presentation and idiosyncrasies of an IDE. As a consequence, switching Anaconda IDEs does not result in a significant change in your Python code. Understanding Python's syntax is the most difficult part of the learning curve. You'll be able to easily transfer your coding skills from one IDE to another after you've mastered one. One IDE is not inherently superior than another; each has its own set of advantages and disadvantages, and the one you select is a matter of personal taste.

Jupyter



Jupyter Notebook is a web-based interactive development environment that runs in your current browser. It is incredibly versatile and easy to experiment with because each block of code may be run individually. Different sorts of texts can be utilized in the same Notebook as a result of this. As a result, you may utilize code outputs, visualizations, equations, and plain text all in one spot.

This allows you to easily create and share documents, as well as show your code and findings in an organized and appealing manner. It is ideal for collaborations because it is web-based and allows users to easily share Notebooks with others. JupyterLab is a Jupyter Notebook add on that adds a lot of functionality. Jupyter Notebooks are combined with other apps in JupyterLab, such as a command-line Terminal, to create JupyterLab.

Rdkit



Rdkit is a collection of Cheminformatics and machine learning tools written in C++ and Python. Which is more important, it allows to work with many representations of chemical data and has a power to extract almost each chemical descriptor from the data you have [24].

- **Reading molecules**

The majority of the basic molecular functionality is found in module:

Individual molecules can be constructed using a variety of approaches

```
from rdkit import Chem
m = Chem.MolFromSmiles('Cc1ccccc1')
m = Chem.MolFromMolFile('data/input.mol')
stringWithMolData=open('data/input.mol','r').read()
m = Chem.MolFromMolBlock(stringWithMolData)

m1 = Chem.MolFromSmiles('CO(C)C')
m2 = Chem.MolFromSmiles('c1cc1')



m1 is None
m2 is None
```

- **Writing molecules**

Molecules can be converted to text using several functions present in the rdkit.chem module for example, for SMILES:

```
m = Chem.MolFromMolFile('data/chiral.mol')
Chem.MolToSmiles(m)
'C[C@H](O)c1ccccc1'
Chem.MolToSmiles(m,isomericSmiles=False)
'CC(O)c1ccccc1'


Chem.MolToSmiles(Chem.MolFromSmiles('C1=CC=CN=C1'))
'c1ccncc1'
Chem.MolToSmiles(Chem.MolFromSmiles('c1cccnc1'))
'c1ccncc1'
Chem.MolToSmiles(Chem.MolFromSmiles('n1ccccc1'))
'c1ccncc1'


Chem.Kekulize(m)
Chem.MolToSmiles(m,kekuleSmiles=True)
'C[C@H](O)C1=CC=CC=C1'
```

- **Working with 2D molecules: Generating Depictions**

The RDKit has a library for generating depictions (sets of 2D) coordinates for molecules. This library, which is part of the AllChem module, is accessed using the rdkit.Chem.rdDepictor.Compute2DCoords function:

```
m = Chem.MolFromSmiles('c1nccc2n1ccc2')
AllChem.Compute2DCoords(m)


template = Chem.MolFromSmiles('c1nccc2n1ccc2')
AllChem.Compute2DCoords(template)

ms = [Chem.MolFromSmiles(smi) for smi in ('OCCc1ccn2cnccc12','C1CC1Oc1cc2ccncn2c1','CNC(=O)c1nccc2cccn12')]
for m in ms:
    _ = AllChem.GenerateDepictionMatching2DStructure(m,template)
```

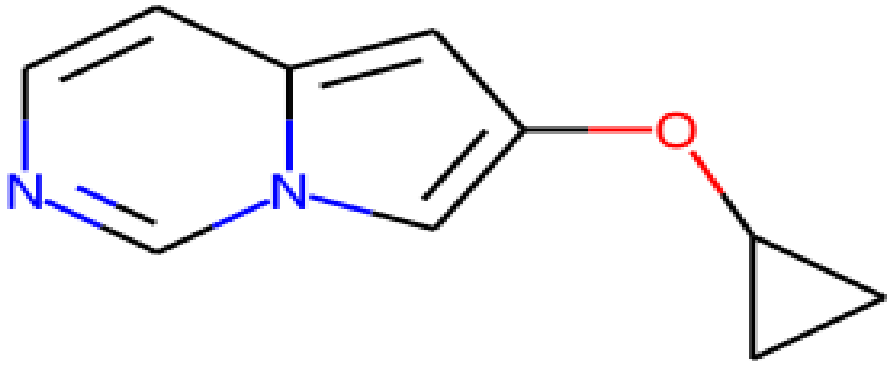


Figure 4.1 Running this process for the molecules use rdkit

Keras

Keras is a high-level neural network API that runs on top of TensorFlow or Theano. It was originally created in Python. It was created with the goal of allowing for quick experimentation. This package provides an R-based interface to Keras. All of the objects supplied by functions in this package are native R objects or raw pointers to python objects, allowing users to use the whole keras API.

(1) accurate, manual parsing of R inputs to Python,

(2) R-sided documentation

(3) examples produced using the API are the package's key features. It lets users to load and execute popular pre-trained models like VGG-19, among other things.

Tensorflow



TensorFlowTM is an open source software package that uses data flow graphs to do numerical computations. The graph's nodes represent mathematical operations, while the graph's edges represent the multidimensional data arrays (tensors) that are exchanged between them. With the flexible architecture, you may use a single API to distribute compute to one or more CPUs or GPUs in a desktop, server, or mobile device. TensorFlow was created by Google Brain Team researchers and engineers as part of Google's Machine Intelligence research group for the aim of performing machine learning and deep neural network research, although the technology is broad enough to be used in a number of different disciplines.

4.1.1 Hardware used

As for the components are:

-GPU Memory with 8 GB.

-CPU intel core i5-7200U

-RAM with 8 GB

-Disk with 256 GB of storage.

4.2 Splitting the data

This dataset contains 999 molecules. Divided into two sets training.

Table 4.1 Organization of training data

Categories	**Number of smiles**	**Number of description**
Training	700	700
Testing	299	299

4.3 Develop the Model

We will define a deep learning based on the “merge-model” described

To get a sense for the structure of the model, specifically the shapes of the layers, see the summary listed below.

Table 4.2 the structure of the model

```
Model: "model_6"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_96 (InputLayer)          [(None, 34)]         0           []

 input_95 (InputLayer)          [(None, 1024)]       0           []

 embedding_24 (Embedding)       (None, 34, 256)      7424        ['input_96[0][0]']

 dropout_48 (Dropout)           (None, 1024)         0           ['input_95[0][0]']

 dropout_49 (Dropout)           (None, 34, 256)      0           ['embedding_24[0][0]']

 dense_72 (Dense)               (None, 256)          1048832     ['dropout_48[0][0]']

 lstm_24 (LSTM)                 (None, 256)          525312      ['dropout_49[0][0]']

 add_24 (Add)                   (None, 256)          0           ['dense_72[0][0]',
                                                                  'lstm_24[0][0]']

 dense_73 (Dense)               (None, 256)          65792       ['add_24[0][0]']

 dense_74 (Dense)               (None, 7579)           7453        ['dense_73[0][0]']

==================================================================================================
Total params: 1,654,813
Trainable params: 1,654,813
Non-trainable params: 0
```

We also create a plot to visualize the structure of the network that better helps understand the two streams of input.

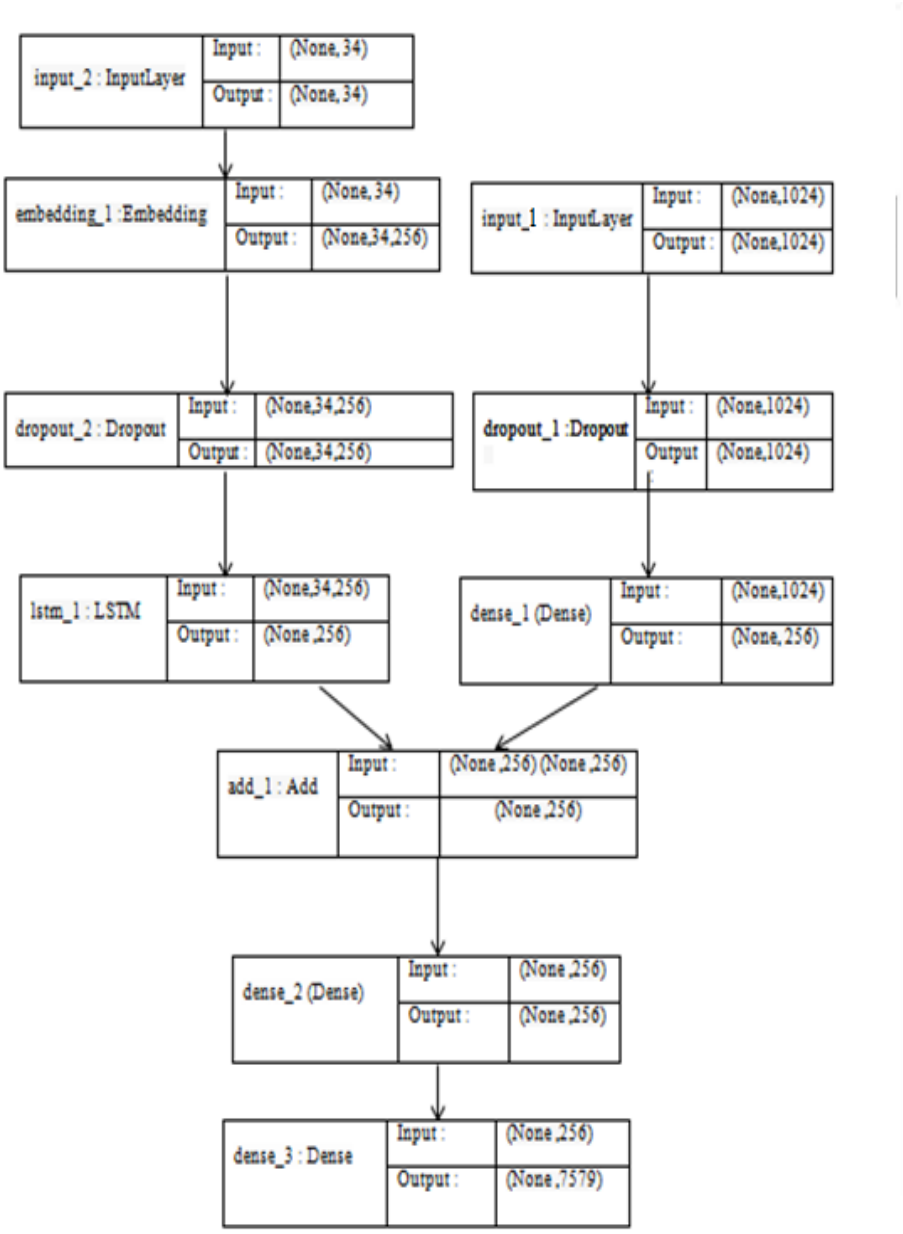


Figure 4.2 Plot of the Caption Generation Deep Learning Model

Now that we know how to define the model, we can fit it on the training dataset.

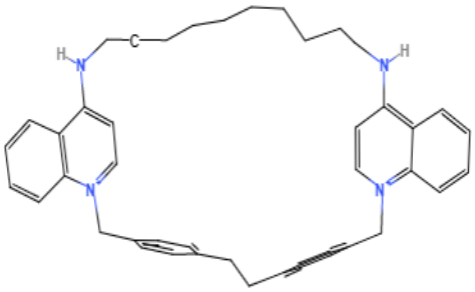
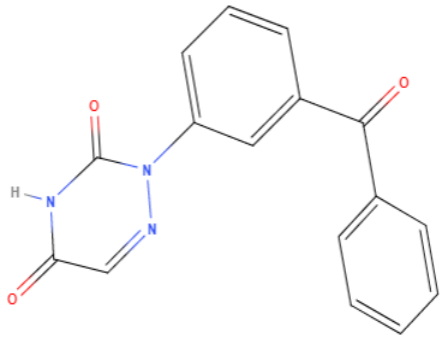
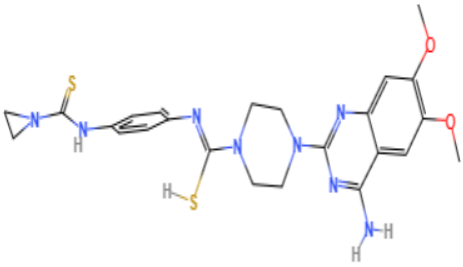
The model learns fast and quickly overfits the training dataset. For this reason, we will monitor the skill of the trained model on the holdout development dataset. When the skill of the model on the development dataset improves at the end of an epoch, we will save the whole model to file .We will only fit the model for 20 epochs.

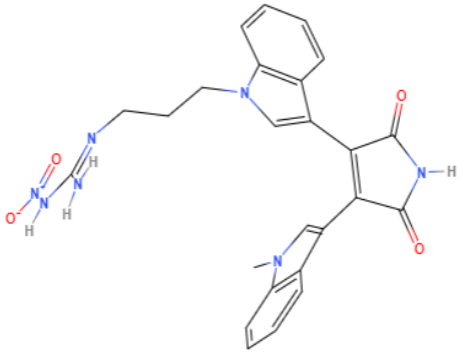
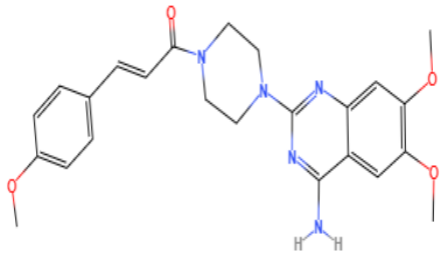
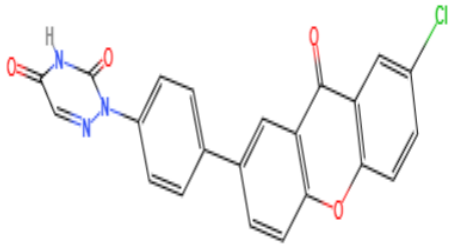
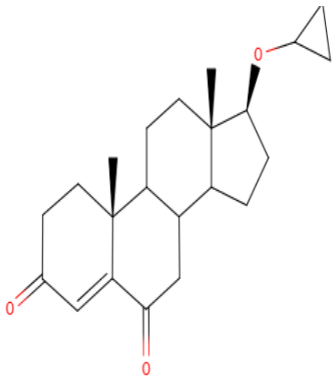
4.4 Result model

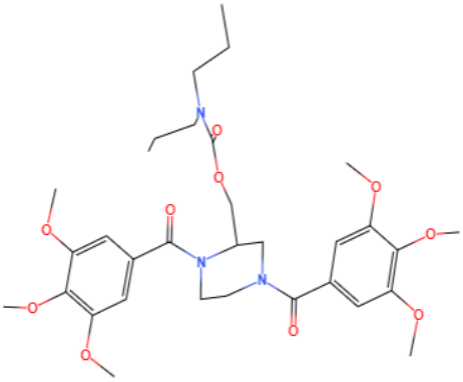
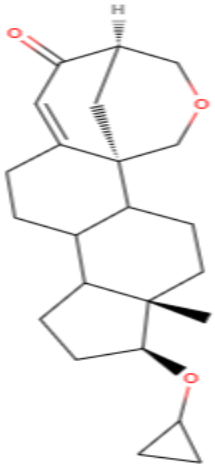
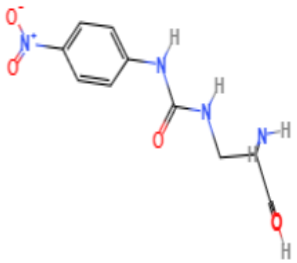
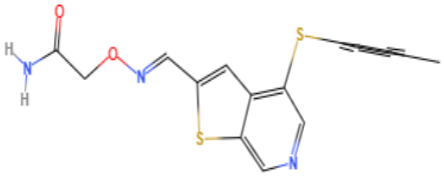
After training the model, we may evaluate the accuracy of its predictions using a dataset

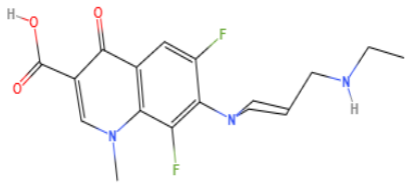
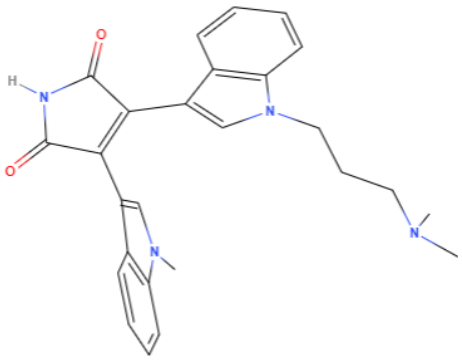
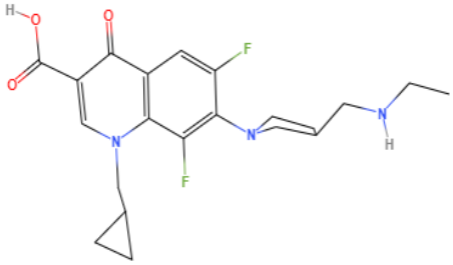
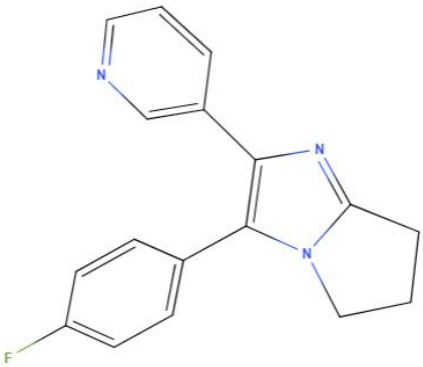
We will test a model by generating descriptions of all the molecules in the test dataset **Table 4.3**

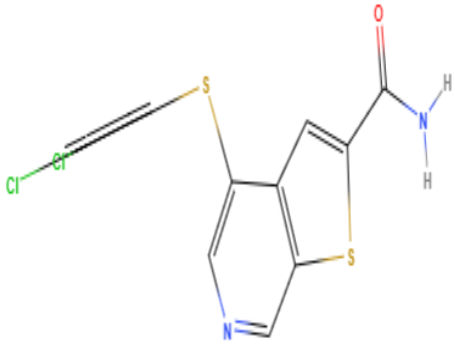
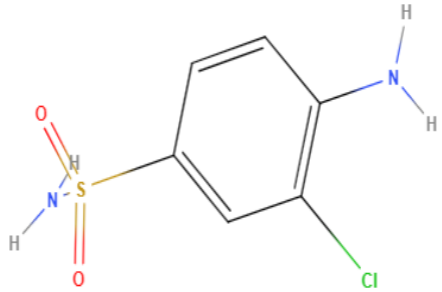
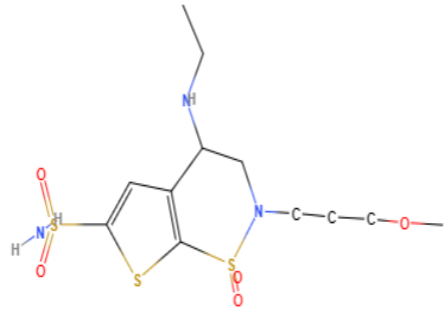
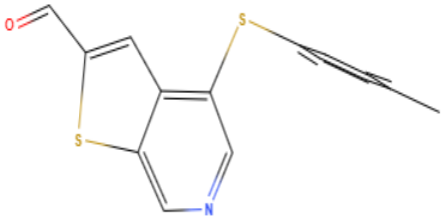
Table 4.3 The description generated

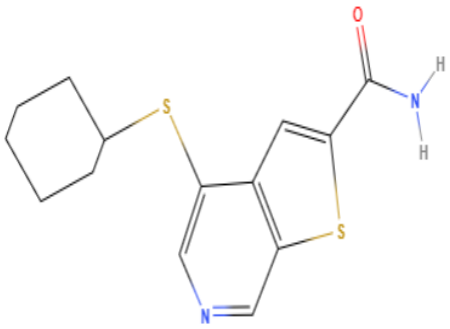
Molecule	Real Description	Prediction Description
	Inhibitory activity against soybean 15-lipoxygenase was evaluated	Inhibitory activity against lipoxygenase was evaluated
	Cytotoxicity against ovarian cancer cell line (1A-9) of humans was determined	Cytotoxicity ovarian cancer cell line humans was determined
	Percentage inhibition of human ovarian cancer (1A9) cell lines	Percentage inhibition tumor ovarian cancer cell lines

	Inhibition of 5'-inosine monophosphate dehydrogenase type I (IMPDH I)	Inhibition monophosphate dehydrogenase IMPDH
	Dissociation constant towards 16S rRNA construct A	Dissociation constant towards construct
	Percentage inhibition of human ovarian cancer (1A9) cell lines	Percentage inhibition human cancer lines
	The compound was tested in vitro for the anti-HBV activity against 2.2.15 cells	The compound was tested vitro for activity against

	Inhibitory concentration against human neutrophil elastase (HNE)	Inhibitory concentration against human elastase
	Compound was evaluated for its inhibitory effect on thymidine [3H]dThd) incorporation into DNA of CEM/MTX cells in culture at conc. 10e-5	Compound was evaluated for its inhibitory effect thymidine incorporation cells culture
	In vitro inhibition of 1,3-beta-glucan synthase in Candida albicans membrane assay	In vitro inhibition synthase Candida membrane assay
	Inhibition of 5-Desaturase involved in ergosterol biosynthesis	Inhibition ergosterol biosynthesis

	In vitro anticancer activity against 11 NCI NSCLC cell lines; inactive	vitro anticancer activity against cell lines
	Inhibition of 11 beta-hydroxylase from rat adrenal gland	Inhibition hydroxylase from adrenal gland
	Tested for inhibition of 12-LO (12-lipoxygenase) as an inhibitor of 12(S)-HETE biosynthesis in mouse epidermal homogenate	Tested for inhibitionas an inhibitor biosynthesis mouse epidermal homogenate
	cytotoxicity against 41M cells incubated for 24 hr in MTT assay	cytotoxicity against incubated for assay

	Compound was tested for agonistic activity against 5-HT uptake	Compound tested agonistic activity
	Agonist efficacy on 5-hydroxytryptamine 1A receptor subtype	Agonist efficacy hydroxytryptamine subtype
	Binding affinity towards 5-HT1F (human cloned receptor) in CHO cells using [3H]5-HT as radioligand	Binding affinity towards human cloned receptor radioligand.
	Stimulatory activity of intragastric pressure was tested in the rat	Stimulatory activity intragastric pressure tested rat



Chymotryptic inhibitory activity against 26S proteasome	Chymotryptic inhibitory activity proteasome

4.5 Summary

The outcomes of our model implementation are presented in this chapter. First and foremost, it is separated into two sections: the first section is about data, including preprocessing, data augmentation, and the production of text data into batches, and the second section is about validation, training, and testing.The second phase is about the model, such as training, compiling, testing, and obtaining results.

GENERAL CONCLUSION

Conclusion

We have completed the project's key objectives, which were set forth from the outset. We researched and learnt several Deep Learning techniques and tools, which are currently highly significant in the field of Artificial Intelligence. We've also gone over the whole bibliography on the molecule description generation issue.

We completed this project, which required a large amount of computational power to train Deep Learning models. As a Deep Learning framework, we utilized Keras on top of Tensorflow for visualizing the training process. we also looked at the many accessible and well-known molecular databases to see which ones would be the best fit for our research.

Finally, we were able to create several Deep Learning models that automatically provide molecular descriptions. for building the models' outputs, we used state of the art Deep Learning techniques for Computer Vision and Natural Language Generation: Convolutional Neural Networks for extracting the molecule fingerprint and Recurrent Neural Networks for generating the sentence using state of the art Deep Learning techniques for Computer Vision and Natural Language Generation. on a large and well-known molecular dataset, these models were effectively trained and evaluated.

After completing the project's core objectives, there are several avenues in which future effort might be directed in order to widen the scope. to begin with, data quantity and quality are critical for effective training; hence, with more and more representative instances in the dataset and more accurately labeled examples, the model might be more accurate and yield better assessment results.

Finally, while Convolutional Neural Networks are the most advanced, they have several restrictions that reduce the model's accuracy. it would be a nice route for this project to take in the future.

REFERENCES

REFERENCES

[1] Y. Chen and J. Kirchmair, “Cheminformatics in Natural Product-based Drug Discovery,” *Mol. Inform.*, vol. 39, no. 12, pp. 1–16, 2020, doi: 10.1002/minf.202000171.

[2] J. K. Wegner *et al.*, “Cheminformatics,” *Commun. ACM*, vol. 55, no. 11, pp. 65–75, 2012, doi: 10.1145/2366316.2366334.

[3] J. GASTEIGER and K. FUNATSU, “Cheminformatics – An Important Scientific Discipline,” *J. Comput. Chem. Japan*, vol. 5, no. 2, pp. 53–58, 2006, doi: 10.2477/jccj.5.53.

[4] J. Gasteiger, “Solved and unsolved problems of Cheminformatics,” *Mol. Inform.*, vol. 33, no. 6–7, pp. 454–457, 2014, doi: 10.1002/minf.201400068.

[5] B. Munos, “Can open-source R&D reinvigorate drug research?,” *Nat. Rev. Drug Discov.*, vol. 5, no. 9, pp. 723–729, 2006, doi: 10.1038/nrd2131.

[6] “Summary of themes from the 2021 Executive Roundtable Examining Experiences Implementing and Accommodating the ICH E6 (R2) Guidance,” vol. 6.

[7] P. Willett, “Chapter 6 Subset-Selection Methods For Chemical Databases Methods for Subset Selection,” pp. 115–116.

[8] E. L. Willighagen *et al.*, “The ChEMBL database as linked open data,” pp. 1–12, 2013.

[9] J. A. DiMasi, R. W. Hansen, and H. G. Grabowski, “The price of innovation: New estimates of drug development costs,” *J. Health Econ.*, vol. 22, no. 2, pp. 151–185, 2003, doi: 10.1016/S0167-6296(02)00126-1.

[10] L. David, A. Thakkar, R. Mercado, and O. Engkvist, “Molecular representations in AI-driven drug discovery: a review and practical guide,” *J. Cheminform.*, vol. 12, no. 1, pp. 1–22, 2020, doi: 10.1186/s13321-020-00460-5.

[11] Huang, “similarity _based_vertuelle-screen_using,” *To BHMA ToY A$Σ$K$Λ$H$Π$IOY$*, vol. 9, no. 1, pp. 76–99, 2010.

[12] J. Newton, “The best of both worlds,” *Chem. World*, vol. 13, no. 7, pp. 68–69, 2016.

[13] Y. Ma, Y. Liu, Q. Xie, and L. Li, “CNN-feature based automatic image annotation method,” *Multimed. Tools Appl.*, vol. 78, no. 3, pp. 3767–3780, 2019, doi: 10.1007/s11042-018-6038-x.

[14] M. Krenn *et al.*, “SELFIES and the future of molecular string representations,” pp. 1–34, 2022, [Online]. Available: http://arxiv.org/abs/2204.00056.

[15] R. Sayle and J. Mayfield, “Advances in (standardization of) organometallic and inorganic structure representations,” no. March, 2022.

[16] Accelrys, “CTfile Formats,” *CTfile Formats*, no. October, 2012, [Online]. Available: http://www.symyx.com/downloads/public/ctfile/ctfile.jsp.

[17] A. Dalby *et al.*, “Description of Several Chemical Structure File Formats Used by Computer Programs Developed at Molecular Design Limited,” *J. Chem. Inf. Comput. Sci.*, vol. 32, no. 3, pp. 244–255, 1992, doi: 10.1021/ci00007a012.

[18] D. S. Wigh, J. M. Goodman, and A. A. Lapkin, “A review of molecular representation in the age of machine learning,” *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, no. August 2021, pp. 1–19, 2022, doi: 10.1002/wcms.1603.

[19] A. M. Clark, “Accurate specification of molecular structures: The case for zero-order bonds and explicit hydrogen counting,” *J. Chem. Inf. Model.*, vol. 51, no. 12, pp. 3149–3157, 2011, doi: 10.1021/ci200488k.

[20] A. C. K. To and R. For, “S TEPPING B ACK TO SMILES T RANSFORMERS FOR,” pp. 1–14, 2022.

[21] N. M. O’Boyle, “Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI,” *J. Cheminform.*, vol. 4, no. 9, pp. 1–14, 2012, doi: 10.1186/1758-2946-4-22.

[22] D. Weininger, “J. Chem. Inj Comput.,” *J. Chem. Inf. Comput. Sci.*, vol. 28, pp. 31–36, 1988.

[23] S. Honda, S. Shi, and H. R. Ueda, “SMILES Transformer : Pre-trained Molecular Fingerprint for Low Data Drug Discovery,” 2018.

[24] G. Landrum, “RDKit: Open-source Cheminformatics,” *Http://Www.Rdkit.Org/*, vol. 3, no. 04, p. 2012, 2006, [Online]. Available: http://www.rdkit.org.

[25] H. Gao, T. J. Struble, C. W. Coley, Y. Wang, W. H. Green, and K. F. Jensen, “Using Machine Learning to Predict Suitable Conditions for Organic Reactions,” *ACS Cent. Sci.*, vol. 4, no. 11, pp. 1465–1476, 2018, doi: 10.1021/acscentsci.8b00357.

[26] S. J. Delany and R. Goebel, *Case-Based Reasoning*, no. July. 2013.

[27] C. J. Krebs, “Part IV: Estimating community parameters. Chapter 12 Similarity coefficients and cluster analysis,” *Ecol. Methodol.*, pp. 484–530, 2014, [Online]. Available: https://www.zoology.ubc.ca/~krebs/downloads/krebs_chapter_12_2017.pdf.

[28] A. Cereto-Massagué, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallvé, and G. Pujadas, “Molecular fingerprint similarity search in virtual screening,” *Methods*, vol. 71, no. C, pp. 58–63, 2015, doi: 10.1016/j.ymeth.2014.08.005.

[29] A. C. Mater and M. L. Coote, “Deep Learning in Chemistry,” *J. Chem. Inf. Model.*, 2019, doi: 10.1021/acs.jcim.9b00266.

[30] P. J. Chun, T. Yamane, and Y. Maemura, “A deep learning-based image captioning method to automatically generate comprehensive explanations of bridge damage,” *Comput. Civ. Infrastruct. Eng.*, pp. 1–15, 2021, doi: 10.1111/mice.12793.

[31] K. Osako, R. Singh, and B. Raj, “Complex recurrent neural networks for denoising speech signals,” *2015 IEEE Work. Appl. Signal Process. to Audio Acoust. WASPAA 2015*, pp. 16–20, 2015, doi: 10.1109/WASPAA.2015.7336896.

[32] P. Karpov, G. Godin, and I. V. Tetko, “Transformer-CNN: Swiss knife for QSAR modeling and interpretation,” *J. Cheminform.*, vol. 12, no. 1, pp. 1–12, 2020, doi: 10.1186/s13321-020-00423-w.

[33] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, “Deep captioning with multimodal recurrent neural networks (m-RNN),” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, vol. 1090, no. 2014, pp. 1–17, 2015.

[34] C. Zhang and Y. Tian, “Automatic video description generation via LSTM with joint two-stream encoding,” *Proc. - Int. Conf. Pattern Recognit.*, vol. 0, pp. 2924–2929,

2016, doi: 10.1109/ICPR.2016.7900081.

[35] P. A. Naidu, S. Vats, G. Anand, and V. Nalina, “A Deep Learning Model for Image Caption Generation,” vol. 8, no. 6, 2020.

[36] L. Ballan, T. Uricchio, L. Seidenari, and A. Del Bimbo, “A cross-media model for automatic image annotation,” *ICMR 2014 - Proc. ACM Int. Conf. Multimed. Retr. 2014*, no. April, pp. 73–80, 2014, doi: 10.1145/2578726.2578728.