POLITECNICO DI MILANO

School of Industrial and Information Engineering
Department of Electronics, Information and Bioengineering
Master of Science Degree in Computer Science and Engineering

T-Tox: A new deep learning model to predict mutagenicity of chemicals

Supervisor:
Prof. Ing. Marco COLOMBETTI
Co-Supervisor:
Prof. Giuseppina GINI

Thesis of:
Francesco ZANOLI 877471

Academic year 2017-2018

Zanoli F.
T-Tox: A new deep learning model to predict mutagenicity of chemicals

francesco.zanoli@mail.polimi.it

Politecnico di Milano
Computer Science Department
Sessione Laurea: 20 Dicembre 2018

Al mio Nonno.

Abstract

Since the early 1900's scientists have searched for a valid method to determine the toxicity of chemicals and to understand toxic processes. In the years, innovation and discoveries have revolutionized the entire industry of pharmacology and chemistry. Testing chemicals started from the trial on animals, passed through the tests in vitro to arrive at the brand new technology of the test on-chip. Today the registered chemical structures are about 28 million, while experimental toxicity data are available for a few hundred thousands of them. Defining properties and effects for all the available chemicals is such a huge challenge due to the cost of the experimentation and to legislative restrictions. Therefore prediction is the only available solution, but it poses many challenges in terms of accuracy and interpretability. Recently, advancements in understanding the human brain provided the basis to create new machines that can think, solve problems, find patterns and even create artworks. In the last several years the biggest enterprises and research groups compete in order to develop the best model that can recognize images and texts, and act as the human brain. The goal of our work is to apply these new cutting-edge researches on machine learning and deep learning to the field of biology and toxicology. The aim is to discover the link between the molecular form of a chemical and its biological effects. To do that we adopted Deep Learning methods, modified the most successful architectures, Inception and LSTM, and combined them to build a model called T-Tox. This model contains three main parts, Toxception, SMILES-Net and C-Tox that respectively perform image analysis, text analysis and classification. Our results are based on a dataset of about 20000 molecules tested for mutagenicity with the AMES test, an in-vitro assay on *Salmonella*. The results obtained by the analysis of the architecture proposed overcome the current state of the art. Besides, the model does not have any knowledge of chemistry. We proposed a method to extract the new knowledge generated from the architecture comparing it with the existing one, obtaining good results especially in the structural alert generation. The advantages of the new approach over traditional model building are that the chemical structure, as obtained from public databases, is a sufficient input and there is no need for specific expertise in chemistry and biology. Moreover, the system is robust to noise. On the other hand, the disadvantage is the computation time needed to build the model.

Sommario

Dai primi anni 90, la scienza cerca un metodo valido per determinare la tossicità di un composto chimico. Negli anni l'innovazione e le nuove scoperte hanno rivoluzionato completamente l'industria della chimica e della farmacologia. I test sui composti sono partiti dai trial sugli animali, passando per i test In Vitro fino ad arrivare ai test in chip. Ad oggi i composti chimici registrati sono circa 28 milioni di cui solo su alcune migliaia sono stati eseguiti gli esperimenti tossicologici. Definire le proprietà e gli effetti per tutte le sostanze è un progetto talmente grande da essere quasi irrealizzabile. Per questo motivo la predizione è l'unica soluzione possibile per ottenere i dati di tossicità per tutti i composti. Questo è però ostacolato da diverse problematiche riguardanti l'accuratezza e l'interpretabilità dei risultati. Recenti sviluppi nella compresione del cervello umano hanno permesso di creare macchine in grado di pensare, risolvere problemi, ricercare pattern e perfino creare opere d'arte. Negli ultimi anni, industrie e ricercatori, stanno gareggiando per sviluppare il modello migliore nella classificazione di testo ed immagini, come lo farebbe un uomo. L'obbiettivo della nostra ricerca è quello di utilizzare questi nuovi metodi di machine learning e deep learning e applicarli al campo della tossicologia. Lo scopo è quello di scoprire le connessioni tra la struttura molecolare e gli effetti collaterali di un composto chimico. Per fare questo, abbiamo descritto e modificato le migliori architetture, Inception e LSTM. Le abbiamo poi combinate per creare un nuovo modello chiamato T-Tox.Quessto modello è composto da 3 parti principali: Toxception, SMILES-Net, C-Tox, che si occupano rispettivamente di analisi delle immagini, analisi del testo e classificatione. I nostri risultati sono basati su un database di circa 20000 molecole testate attraverso l'AMES test. Questo test è un test In Vitro che sfrutta la Salmonella. I risulati ottenuti dall'analisi dell'architettura proposta sorpassano lo stato dell'arte. In aggiunta il modello non contiene nessuna base di conoscenza, questo rende possibile l'analisi e l'estrazione di quello che il modello ha imparato. Questo permette inoltre di comparare la conoscenza estratta con della conoscenza esistente. In particolare i risultati migliori sono ottenuti nella generazione delle SA. I vantaggi di questo nuovo approccio sono principalmente nella mancanza di conoscenza iniziale che permette di non dover avere esperienza umana per poter creare un modello simile. Inoltre il sistema è resistente al rumore. Lo svantaggio però è il tempo computazionale richiesto per creare il modello.

Ringraziamenti

Prima di tutto, vorrei ringraziare il Professor Marco Colombetti, il relatore di questa tesi, per i suoi preziosi consigli e per avermi fornito tutti gli strumenti di cui avevo bisogno per portare a compimento la mia tesi.
Un grazie speciale anche alla Professoressa Giuseppina Gini, oltre che per l'aiuto fornitomi e la grande conoscenza che mi ha donato, per la disponibilità e precisione dimostratemi durante tutto il periodo di stesura. Mi ha trasmesso la passione e l'entusiasmo per poter affrontare al meglio questo percorso. Senza di Lei questo lavoro non avrebbe mai preso vita.
Inoltre, vorrei ringraziare l'istituto di ricerca Mario Negri, in particolare Giuseppa Raitano, Alessio Gamba ed Emilio Benfenati, per il supporto tecnico ed i dati condivisi.

Un ringraziamento speciale alla mia famiglia, Mila, Roberto, Elisabetta: è grazie al loro sostegno e al loro incoraggiamento se oggi sono riuscito a raggiungere questo traguardo. Mi hanno insegnato i valori che sono alla base di una vita onesta, la curiosità, la capacità di crearsi nuovi obbiettivi e la voglia di mettersi in discussione. Non potrò mai ringraziarvi abbastanza.

Contents

Chapter 1

Introduction

Humans have been creating and discovering chemicals for many centuries. Over the last hundred years, we have also developed a science called toxicology, which studies the effects of the discoveries on a living organism. In the last decades, this science has gained more and more interest from industries, authorities, and the mass market.
Nowadays people are worried about what they eat, touch and breath. This interest has spurred industries and scientists in the development of new and efficient methods to test the side effects of chemicals on the environment and human health. The methods adopted have been traditionally derived from toxicology, a branch of pharmaceutical research. Toxicology heavily depends on testing methods performed on living beings. Since the introduction of computers, toxicology has evolved embedding the machines as instruments to reach its primary goal: the definition of a model (a mathematical function) to describe toxicity. Those models are based on the principle that the chemical structure is responsible for the effects on the living systems; the acronym QSAR (Quantitative Structure Activity Relationship) is indeed used to name them.
Moreover, the interest of the population has also moved from the bare toxic effect to the testing methods used. In just a few decades the testing done on animals that were socially accepted in the '50s, has become to be considered an atrocity. The pressure on industry to change its testing methodology has pushed the research for an alternative solution. In this scenario, toxicology and the progress in computer science become suddenly really interconnected. The new achievements in pattern recognition, data mining, and artificial intelligence, made in the last decades, tremendously improved computational toxicology. The prediction methods that were traditionally based on parametric methods are now often based on non-parametric methods that better find complex non-linear relationships.

Computational models, called in the toxicology jargon in-silico models, have a wide range of applicability; on the long term they will significantly help companies to save money and to exploit at most their knowledge bases, and for this reason, this is a

trending topic that has been receiving a lot of interest. System regulators constantly monitor this field with interest because, even if they do not yet propose a proprietary solution, they consider this topic as a real and valuable spot.
However, there is a very problematic issue scientists have to face while developing such kind of solutions: not always the data available are enough to create valid and flexible models. For this reason, one of the hottest research trends in the community is to find a standard way to store and share these data. International public bodies, as EPA (Environmental Protection Agency) in the USA, are actively working to providing free and certified datasets.
On the other hand, big companies such as Pharmacy, Cosmetics, and Food industries, are not sharing their precious information about tests. This slackens the evolution of the predictive toxicology domain.

1.1 Aims and goals of the thesis

Toxicology, as well as biology, is based on natural laws. Discovering those natural laws is the main objective of biology. However, this task is long and difficult as any induction task. In toxicology the laws to discover can be arranged at different abstraction levels: the lowest one is to find a relationship between the chemical structure and the biological effect. At higher levels, the effect could be explained as a process with time steps, or can be related to portions of the DNA, and so on. In any case, these laws are extremely inclined to pattern creation. Therefore it is intuitive that the application of advanced pattern recognition methods, such as deep learning, could give important results if applied to toxicology. This idea has been recently proposed in literature with the aim of reorganizing the chemical knowledge making use of its vast size (28 millions of molecules are registered in the official CAS database).
This thesis, on the one hand, presents a synthesis of the technical and practical knowledge present in the literature about computation toxicology and deep learning and on the other hand, it aims at constructing an architecture based on the most advanced deep learning techniques so demonstrating in practice that advanced machine learning methods offer a viable solution. The model is built on data available for a widely used test, the Ames test for mutagenicity, which gives a Boolean result. In addition, this thesis aims at proposing a standard procedure to treat and manage the data used to create the model.
Finally, this work has the ambitious goal of designing the new architecture depending only on raw data (the test results) and no other a priori or external knowledge. That is, no knowledge will be introduced in the modeling steps by the human expert and no data features will be a priori computed. The only input we provide is the definition of the chemical 2D structure in text format and the Boolean value of mutagenicity. Moreover, from the developed model we aim at extracting the knowledge self-generated during training in order to either compare it with the existing one or

to analyze it as new knowledge.
Please consider that we are not interested yet in making our model accepted by regulators; our aim is to make it accepted by researchers.

1.2 Contribution

The main contribution of this thesis is designing a dual architecture which groups together two of the most efficient deep learning techniques, devoted to image and text analysis, increasing the prediction accuracy for computational toxicology. The dual architecture, called T-Tox, integrates two networks trained on the 2D chemical structures; Toxception sees the structures as images, while SMILES-Net sees them as text.
A characterizing contribution of this work is the idea of training a computational toxicity model without using external expert knowledge. This proposal goes against the most common practices found in literature but provides better results when compared to the results provided by previous models.
In addition, we precisely describe the preprocessing pipeline, and in particular the development of novel data treatment, an essential part of each model based on deep learning techniques.
Another contribution of this work is the study of different hyperparameters applied to known structures as Inception and LSTM proposed by Szegedy et al. (2017) [79] and Hochreiter and informatik (1997) [38].
This works also makes available an easy-to-use software: a python module to use the developed Ames model and to train other models based on custom data. In conclusion, this thesis adds a new application for deep text classification and image classification which has never been used before in the literature for this specific tasks.
Finally, through this work, we opened a new branch of in-silico research using deep learning; if further deepened, could potentially improve the design of similar models.

1.3 Overview

From a high-level point of view, the structure of this thesis reflects the research plan we followed: we started by studying the basic concept of toxicity and its domain definition, discussing the main challenging parts and analyzing the possible focus points. We moved then towards the current state of the art in computational toxicology, analyzing the mainly adopted methods, and finding out that their architecture was not intuitive and too much constrained. For this reason, we decided to move towards deep learning, a more complex but recent, intuitive and fascinating solution. We studied the possible classification methods, from image to text passing through basic features, and for each of them we went deeply into the theory behind it.
In the scope of deep learning classification, we selected and validated a set of possible hyper-parameters.

As last step, we selected the best performing model for each of the types studied and we merged it together to create the final architecture.
In order to target this work on a real working reality, we created a dataset of chemicals grouped from the literature and we trained and tested our model and every sub-model on this data.
For feasibility reasons, we oriented the models to save resources and to exploit at most available data and computational power.
This thesis is organized into eight chapters; the chapters describing our model architecture are composed by three main subparts: state of the art of the current techniques used, the proposed implementation, and the results on that part.
The chapters contain:

- Chapter 2 is a high-level overview of Toxicity. It presents the main definitions used in this domain, the classes used to identify a chemical, and the possible tests on this chemical. We also describe the AMES test (in section 2.4.2) with its pros and cons. The topics introduced are fundamental to understand the basic concepts used in the following chapter. In addition, we describe the open challenges of toxicity as the tests variability (section 2.5.1). In section 2.5.2 we add some considerations about the lack of data, a problem proper of this domain

- In Chapter 3 we present the current state of the art for computation toxicology, covering the most used techniques of QSAR (section 3.3.2) and the theory behind them. We added subsection (section 3.1) about the primary users and creators of these techniques, and explaining the main reasons of interest in the domain and its development. We also introduce a basic explanation of the basis of this work by citing some interesting propositions from literature (section 3.2.2).

- Chapter 4 is the introduction and definition of our project and specifies the whole architecture of the model.
  It contains a small state of the art of deep learning applied to toxicology and the main challenges we faced in carrying out our work. It discusses the reason for choosing the AMES test (section 4.3.1). It also contains the whole data preprocessing method (section 4.3) with the main problems encountered in the development. In addition, it contains a detailed explanation of the evaluation techniques used on the model.

- In Chapter 5 we discuss the first part of the model, i.e., Toxception, which makes a visual classification using an architecture similar to the one proposed by Szegedy et al. (2017) [79]. Moreover, section 5.2 explains the theory behind state of the art for image classification by deeply analyzing the mathematical formulas used by the network.
  In addition, we discuss the hyper-parameters chosen to test the possible different networks (section 5.4). Finally in section 5.5 we report the results obtained from the model tested on the AMES test dataset.

- Chapter 6, is structured as Chapter 5 but it contains the part of the model that performs text classification, called SMILES-Net. As before, in this chapter we analyze the current state of the art techniques as LSTM, bidirectional cell, and attention layer. In section 6.5 we report the proposed architecture for this subpart. In particular in this chapter, we also discuss the attention mechanism and the method used to extract knowledge from the network.

- In Chapter 7 we report C-Tox, the final feed-forward network proposed to integrate the results of the two networks based on images and text. We also discuss some basic techniques concerning neural networks in general, for example the optimization algorithm, the learning rate and the activation function. Then in section 7.2 we state our proposal and we describe the workaround we used to save computational time using transfer learning, also explained in this chapter.

- Chapter 8 contains the discussion of the whole document. We discuss the final proposal for each part of the architecture, as well as the advantages of not using apriori knowledge. In addition, we insert considerations concerning some common mistakes that can be made in the evaluation of our proposal and results. Finally, this chapter contains the description of the distribution methods we decided to adopt to share our model and architecture.

- In Chapter 9 we derive our conclusions and we point out which are the most promising research directions for future work.

Chapter 2

Toxicology and toxicity

Toxicology is a discipline, overlapping with biology, chemistry and pharmacology, that involves the study of the adverse effects of chemical substances on living organisms and the practice of diagnosing and treating exposures to toxins and toxicants. The first formal study of these effects was done by Orfila (1813) [64]. These studies divide and classify the existing chemicals in order to determinate the risk of using them based on the general process reported in figure 2.1. In the centuries these experiments have changed the subjects and their protocols. A brief of the history of toxicology is in paragraph 2.1. They started with *In vivo* testing, where actual living organism as animals or even humans were treated with specific substances and the effects were reported analyzing the blood and the reaction. Then *In vitro* testing was applied, where the subject was no longer from the animal's domain but instead was a particular virus or bacteria able to simulate some specific function of the human organism. Finally, the introduction of the computer gave origin To *In silicio* testing, that is the new edge technique that allows scientist to predict the result of toxicity based on the previous assays or based on reasoning and knowledge of chemical and chemistry. Of course all of the three categories still available in toxicity experiments even if government and communities are pushing the scientist to the latter in order to stop animal usage.
In this chapter, we are going to define (in paragraph 2.2) some basic definitions that are useful to understand the choices and the results explained in this paper. Moreover in 2.3 and 2.4 we explain the different possible tests used in different epochs and the evolution this domain has been subjected to. Finally in paragraph 2.5 we analyze the primary challenges of Toxicology. It is important to remark that all the knowledge presented in this chapter it is only a basic introduction of the fundamental concepts, in order to have a better understanding of the work realized.

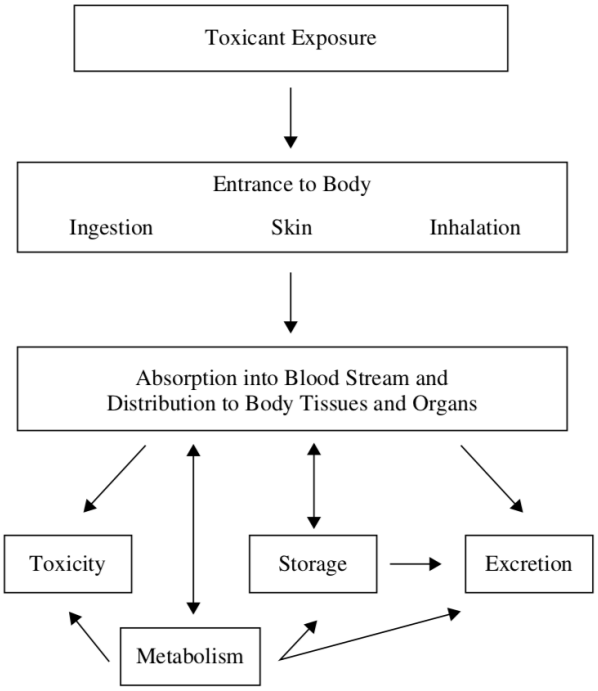


Figure 2.1: The toxicity process in a general scheme

2.1 A Brief History of Toxicology

Much of the early history of toxicology has been lost and in much that has survived toxicology is of almost incidental importance in manuscripts dealing primarily with medicine. Some, however, deal more specifically with toxic action or with the use of poisons for judicial execution, suicide or political assassination. The Egyptian papyrus, Ebers, dating from about 1500 BC, is ranked as the earliest surviving pharmacopeia, and the surviving medical works of Hippocrates, Aristotle, and Theophrastus. The early Greek poet Nicander treats, in two poetic works, animal toxins (Therica) and antidotes to plant and animal toxins (Alexipharmica). There appear to have been few advances in either medicine or toxicology between the time of Galen and Paracelsus. It was the latter who, despite frequent confusion between fact and mysticism, laid the groundwork for the later development of modern toxicology by recognizing the importance of the dose-response relationship. His famous statement *"All substances are poisons; there is none that is not a poison. The right dose differentiates a poison and a remedy"* succinctly summarizes that concept. His belief in the value of experimentation was also a break with earlier tradition. There were some significant developments during the eighteenth century. Probably the best known is the publication of Ramazini's Diseases of Workers in 1700, which led to his recognition as the father of occupational medicine. The correlation between the occupation of chimney

sweeps and scrotal cancer by Percival Pott in 1775 is almost as well known, although it was foreshadowed by Hill's correlation of nasal cancer and snuff use in 1761. Orfila, a Spaniard working at the University of Paris in the early nineteenth century, is generally regarded as the father of modern toxicology. He clearly identified toxicology as a separate science and, in 1815, published the first book devoted exclusively to toxicology. An English translation in 1817, was entitled A General System of Toxicology or, A Treatise on Poisons, Found in the Mineral, Vegetable and Animal Kingdoms, Considered in Their Relations with Physiology, Pathology and Medical Jurisprudence. Since then, advances have been numerous, too numerous to list in detail. They have increased our knowledge of the chemistry of poisons, the treatment of poisoning, the analysis of toxicants and toxicity, modes of toxic action and detoxication processes, as well as specific molecular events in the poisoning process. With the publication of her controversial book, The Silent Spring, in 1962, Rachel Carson became an important influence in initiating the modern era of environmental toxicology. Her book is often credited as the catalyst leading to the establishment of the US Environmental Protection Agency and she is regarded, by many, as the mother of the environmental movement. It is clear, however, that since the 1960s toxicology has entered a phase of rapid development and has changed from a science that was largely descriptive to one in which the importance of mechanisms of toxic action is generally recognized. Since the 1970s, with increased emphasis on the use of the techniques of molecular biology, the pace of change has increased even further, and significant advances have been made in many areas, including chemical carcinogenesis among many others. [3].

2.2 Definitions

It is mandatory to clarify some definitions we used to elaborate this work. That is to define a common language before starting the conceptual and practical work on this domain. This part has also been introduced to analyze the different standards of the current state of the art due to the presence of different entities and active organizations that use different protocols and standards.
Moreover, there are different usages of the same words, especially between scientific and normal words. So strict definitions are needed in order to clarify the differences between the common terms used in day to day speech and the proper words used in literature.

2.2.1 Chemical

Chemical is defined as *any substance, or a mixture of substances.* We can distinguish different types of chemicals:

- **Substance** chemical elements and their compounds in the natural state or obtained by any production process, including any additive necessary to preserve

the stability of the product and any impurities deriving from the process used, but excluding any solvent which may be separated without affecting the stability of the substance or changing its composition.

- **Element** the simplest form of matter. There are currently 118 known elements in the periodic table.
- **Chemical compound** a substance consisting of two or more elements combined or bonded together so that its constituent elements are always present in the same proportions.
- **Mixture** a combination or a solution composed of two or more substances in which they do not react.

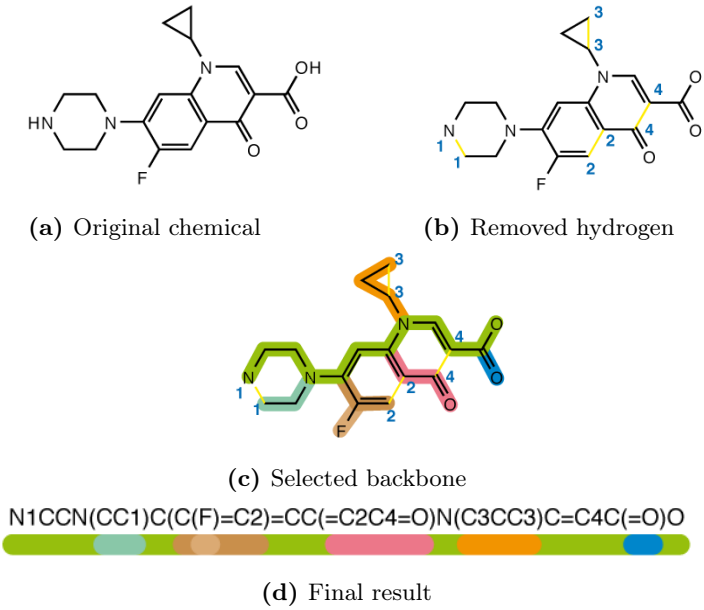
In this paper, we are going to mainly use the definition of Chemical compound and Substance as all the literature studies and experiments have concentrated on these type of chemicals.

2.2.2 SMILES

Simplified Molecular input line entry system or SMILES; it is a specification in the form of a line notation for describing the structure of chemical species using short ASCII strings. This specification was first described in *SMILES. Algorithm for generation of unique SMILES notation* by Weininger et al. (1989) [82] and edited in 2007. It allows scientist to pass from a 2D/3D representation of a chemical into a simple string and to do the inverse process from the string to the image. SMILES allows not only an easy description of the chemicals but also a practical way to transport information in a compressed form. Typically multiple SMILESs can be written in different ways for a single molecule, that is because there exist different algorithms to encode the structure into a string. Each algorithm has a single unique string for a single molecule. The algorithm used in this thesis is called *Canonicalization algorithm* and the SMILES derived from it is then called *Canonical SMILES*. In terms of procedure, the algorithm is graph based. The string is obtained by printing the symbol nodes encountered in a depth-first tree traversal of a chemical graph. The chemical graph is first trimmed to remove hydrogen atoms and cycles are broken to turn it into a spanning tree. Where cycles have been broken, numeric suffix labels are included to indicate the connected nodes. Parentheses are used to indicate points of branching on the tree. The resultant SMILES form depends on the choices: of the bonds chosen to break cycles, of the starting atom used for the depth-first traversal, and of the order in which branches are listed when encountered. The entire process is represented in figure 2.2.

The dictionary used to transform the figure into a string can be split into different categories:

- *Atoms* are represented in branches // except if they are either organic (B, C, N, O, P, S, F, Cl, etc.) or if they have no change. Branches are also omitted if there



(a) Original chemical

(b) Removed hydrogen

(c) Selected backbone

(d) Final result

Figure 2.2: The SMILES algorithm applied to a molecule of Ciprofloxacin

is a number of Hydrogen omitted by the SMILES that create a neutral charge.

- *Bonds* A bond is represented using one of the symbols . ,-, =,# , $, :, / Bonds are assumed to be single. Double, triple, and quadruple bonds are represented by the symbols =, # , $ respectively.
- *Rings* structures are written by breaking each ring at an arbitrary point (although some choices will lead to a more legible SMILES than others) to make an acyclic structure and adding numerical ring
- *Branching* are described with parentheses, as in figure 2.2 The first atom within the parentheses, and the first atom after the parenthesized group, are both bonded to the same branch point atom.

2.2.3 Toxicity

The EPA (U.S. Environmental Protection Agency) defines it as:

> *The degree to which a substance (a toxin or poison) may be harmful to the environment or hazardous to your health if inhaled, ingested or absorbed through the skin.*

Different entities have different definitions, in particular we can determinate different classes depending on the effect that the substances have on the organism. This classification is determined by approved testing measures or calculations and has determined cut-off levels set by governments and scientists. Currently, many countries

have different regulations regarding the types of tests, numbers of tests and cut-off levels. The most used classifications in the literature are written by EPA (2003) [24] and EU of the European Union (1992) [61].

Name	**EPA' classes**	**European classes**
Class I	*Extremely hazardous*	*Very toxic*
Class II	*Highly hazardous*	*Toxic*
Class III	*Moderately hazardous*	*Harmful*
Class IV	*Slightly hazardous*	*Corrosive*
Class V	-	*Irritant*
Class VI	-	*Sensitizing*
Class VII	-	*Carcinogenic*
Class VIII	-	*Mutagenic*

Table 2.1: Two different type of classification from the different standardization about the toxicity level in compounds

In particular EPA classification is based on the *LD50* concentration parameter. The European classification instead is based on the attributes of the substances. The main differences between this two classes are that the former is strictly based on parametric measures, these must be above a certain threshold. The latter instead is based on different tests and experiments that define a more massive and flexible categorization. At the same time, it creates a more complex environment by introducing ambiguities. An objective method to classify a brand new substance is explained in detail in paragraph 2.3

In this paper, we consider the European classification as it describes more in detail the causes of the adverse effects and allows us to analyze the different characteristics of the studied substances better. Moreover we can split this eight classes into two main groups: *Acute toxin* and *Chronic toxin*. That is based on the different time of exposure that the organism needs to be poisoned by the toxin. In the former, we have almost immediate effects and this frequently implies an immediate lethal effect for the organism. The latter instead causes a poisoning with a medium/long-term exposure. It also could lead to a lethal effect but only after specific amount of time that depends on the organism and its characteristics.

Today the study of acute toxicity is mandatory for all the substances introduced in the market and it is also easier than chronic toxicity to observe using in vitro testing. The studies of chronic toxicity have been introduced only after the Second World War with the help of the no smoking campaigns and the discovery of thousands of colorants. This type of toxicity is, as imaginable, difficult to prove either with in vitro or in vivo experiments. Many tests have been proposed and different criteria have been created to classify this kind of substances. Some of these tests, the one considered for this work, are explained in paragraph 2.4. The interest in chronic toxicity is high because

the effects caused by the chemical are often not curable, as in the case of cancer. Moreover the cost and the amount of resources needed to discover it still really high. Scientists have then moved their interest into the prediction of chronic toxicity in order to decrease the cost of testing and fulfilling all the requirements imposed by the legislature.
It is interesting to point out the definition of mutagenicity:

> *Mutagenicity refers to a chemical or physical agent's capacity to cause mutations (genetic alterations). Agents that damage DNA causing lesions that result in cell death or mutations are genotoxins. All mutagens are genotoxic, but not all genotoxins are mutagens as they may not cause retained alterations in DNA sequence. (John Tainer, 2013)*

This class is the least dangerous for human toxicity, and at the same time, it is also the most studied. That is due to the high tendency of mutagenic compounds to catalyze the effects of other classes. In fact, the presence of mutagen helps cancerogenic substances and increases their level of toxicity. It is explainable with some physiologic response of the cell in the presence of specific substances.

2.3 Classification Methodology

The Occupational Safety and Health Administration (OSHA) had tried to standardize, the classification process, independently from the classes chosen. It is composed of different phases and can be adapted to all the different tests. The *Hazard Classification Guidance* by OSHA (2016) [65] define not only these processes for toxic molecule but also for all the hazardous categories defined by the different countries. It defines the process as:

> *Hazard classification is the process of evaluating the full range of available scientific evidence to determine if a chemical is hazardous, as well as to identify the level of severity of the hazardous effect. When complete, the evaluation identifies the hazard class(es) and associated hazard category of the chemical.*

In particular Hazard in this context is interpreted as:

> *Hazard refers to an inherent property of a substance that is capable of causing an adverse effect.*

Although the *Hazard Classification Guidance* treat all the different categories of hazardous chemicals, in this paragraph we are going to concentrate only on the *Health hazard*, that is the classification explained in the introduction of chapter 2 and adopted by the European Union. In particular, some sub-classes are defined in order to point out the differences between chemical belonging to the same class.

Class	**Category**			
Acute Toxicity	1	2	3	4
Irritant	1A	1B	1C	2
Sensitization	1A	1B		
Mutagenicity	1A	1B	2	
Carcinogenicity	1A	1B	2	
Reproductive Toxicity	1A	1B	2	Lactation

Table 2.2: The subset of the classes of human toxicity defined in the introduction of chapter 2

Before exploring the different phases of the classification process it is important to remark that even in this process there are some differences between *substances* and *chemical compounds* (see paragraph 2.2.1), that is because different links between elements produce important effects on the physic properties of the substances analyzed. The classification process is composed of four phases that can be singularly analyzed and discussed.

2.3.1 Inventory and Data collection

The first step of this process consists in listing out all the different compounds, elements and substances we want to include in the research. That can be done by analyzing the literature or creating a *Market requested compounds list* based on the production of new chemicals that companies have made or they have planned to create in the next periods.
Once the list of chemicals is complete, it is important to proceed with data collection, that is the task of regrouping all the information about the chemicals. In particular, there are three main categories we need to collect about the data:

- *Chemical Identity.* It includes the chemical name along with common name and synonyms, *CAS* (Chemical Abstract Services) numbers. That is a unique number assigned by the American Chemical Society. These IDs are useful to standardize the collected data and to avoid the usage of replicated data. In fact, due to the large number of different standards the probability to have a misinterpretation of the chemical name or to have replicated data is high if we do not apply a specific cleaning function on the data. Another factor for which a well defined chemical identity is important is that some chemical properties as SMILES descriptor, described in paragraph 2.2.2, can be represented for the same compound in different forms, this implies the impossibility to detect duplicates in the collected dataset using only some descriptors.

- *Physical and chemical properties.* These are the empirical data of the substances or mixture. Data gathered from observation or by tests performed on the chemical. For the majority of chemicals, there are available public databases with

plenty of information about them. For the newly discovered compounds instead, data need to be extracted from newly realized tests.

- *Health effects.* These are the result of the tests explained in paragraph 2.4. The data included in this set are the observable effects that the molecules have on an organism, considering humans and the environment (water, soil and air organisms). In particular, this effect can be observed through blood examination, microscope, MRI, etc, and are expressed as categories (for instance mutagen or not mutagen) or as doses.

All the data used in this work are discussed in paragraph 4.3 and are collected from literature and databases created by public, private companies and research institutes.

2.3.2 Data analysis

The third step of the classification process is data analysis, which is the most demanding in term of technical expertise. This step is based on the classification guideline asked by governments and public bodies; therefore it can dramatically change from time to time and from country to country. In this step, different technologies have been introduced as an alternative to the human technical expertise passing from statistics through machine learning arriving at deep learning. The main goal of this path is to improve the human decision in order to pass the *Knowledge barrier*, that is the maximum knowledge that each individual can manage. In this third step, using automatized reasoning, it is possible to find not only classes of the chemicals in the data collected but also some more general rules that could help the development and the study of new molecules. More details about the stand-alone techniques are given in the description of the state of the art in chapter 3.
The classification of chemicals can be derived from data resulting from the test. Some classes can be determinate by *Weigh of evidence* (WoE) using expert judgment. That is the usage of all the available data on that chemical, even general data, that together can lead to a specific endpoint. Different standardizations provide different criteria to derive the class from WoE. Some of the tests used to find some WoE are explained in paragraph 2.4.

2.4 Possible tests

Many different types of data about toxicity can be extracted from a single test. In particular, we can distinguish between two main types of test: *In vitro*, such as cell cultures or tissue slices, and *In Vivo* that include laboratory tests on animals and volunteer humans. Specifically for toxicity, there are different categories of test defined in *Guideline for testing of chemicals* by OECD/OCDE (2001) [60]. We are going to analyze some possible tests dividing the explanation in two categories as reported by Shaikh and D.G. (2016) [75] and Oghenesuvwe Erhirhie et al. (2018) [62].

The *In vitro* tests explained in this paragraph are only a small part of the total and with some time there are going to be even more *In vitro* tests and less *In vivo* studies as explained by Krewski et al. (2010) [47].

2.4.1 *In Vivo*

As it is imaginable, advises against the usage of this type of tests are more and more present in the domain of toxicology. That is because the usage of living animal and the possible danger humans are submitted to. There are different types of *In Vivo* tests that can be used based on the different toxicity being analyzed:

- *Acute toxicity tests.*Acute toxicity appears almost immediately (hours/days) after an exposure. Acute exposure is usually a single dose or a series of doses received within 24 hours. These tests have different steps and each step uses groups of animals of the same sex, each group is treated with a specific dose. Absence or presence of compound-related mortality of the animals dosed at one step will determine the next step: no more tests needed, another test with the same dose or another test with a higher/lower dose. In this type of test usually, rodents are used and the sex chosen is the one predicted to be more sensitive to the chemical. Weight, age and temperature are defined by the OECD in order to standardize the execution of the test. This type usually uses LD_{50} as a parameter to evaluate the results. Standardized tests are available for oral, dermal, and inhalation exposures. The most famous tests of this type are *Graphical method of Miller and Tainter*, *Arithmetical method of Reed and Muench*. The lethal dose is calculated using the arithmetical *method of Karber* which is:

$$LD_{50} = LD_{100} - \sum_{n=1}^{N} \frac{(a * b)}{n} \tag{2.1}$$

  where

  a : Dose difference
  b : Mean mortality
  n : Group population
  $LD100$: Least dose required to kill the 100% of the population

  Some other tests can be found in the literature as *Lorkeaazs method* and *Fix dose procedure (FDP)*. The most used recently is the *Up and down procedure (UDP)* that consist of the use of only one animal per group at which incremental doses are administrated. If the animal dies, the successive group will receive a decremented dose.

- *Skin sensitization tests.* In the skin irritation test, 0.5 g of a test substance is applied to the surface of an animal's skin. During the observation period (14

Tests	Species	Age	N of animals	Dosage	Observation period
Acute Toxicity	Rats,Rabbits	Young,Adults	5 per group	Three dose level	14 days
Subchronic Toxicity	Rodents, Rubbit, Dogs	Young,Adults	10 rodents, 4 dogs	Three dose	90 days
Chronic Toxicity	Rodents, Rats, Dogs	Young,Adults	20 rodents, 4 dogs	Three doses	12-24 months

Table 2.3: The parameters used from the different *In vivo* tests

days), signs such as erythema and edema are assessed. The most common is the *Draize eye irritancy test*, but some *In vitro* experiments can be used. At the end of the study, the animals are sacrificed and pathological changes are evaluated. Skin sensitization tests are carried out usually using the guinea pig as a model.

- *Subchronic toxicity tests.* Subchronic toxicity results from repeated exposure for several weeks or months. It is a common human exposure pattern for some pharmaceutical and environmental agents. Some parameters about these tests can be found in table 2.3

- *Chronic toxicity tests.* These studies extend over a long period of time, and they involve large groups of animals. Chronic toxicity represents cumulative damage to specific organ systems and takes many months or years to become a recognizable clinical disease. The process to determinate chronical toxicity passes by a long-term administration of the chemical until the damage exceeds the threshold for chronic diseases. Ultimately the damage becomes so severe that the organ stops its normal functioning and a variety of toxic effects may result.

- *Neurotoxicity studies.* They may be employed to evaluate the specific histopathological and behavioral neurotoxicity of a chemical and are used to characterize neurotoxic responses such as neuropathological lesions and neurological dysfunctions (loss of memory, sensory defects, and learning and memory dysfunctions). Usually, neurotoxicological studies are carried out in adult rodents. The test substance may be administered for 28 days or even more than 90 days, and neurological changes are evaluated.

Table 2.3 is a summary of the number of animals and the specifics used.

2.4.2 *In Vitro*

A lot of *In vivo* test can also be realized using *In vitro* instead. Besides a reduction in costs, the main driver of this choice is the consideration that animal welfare becomes unavoidable. To reduce animal use as well as to make more specific tests on cells and

tissues, we are approaching in *in silico* or computational toxicology, which may be a final goal. Even the new silicio based technology can be considered *In Vitro* tests. That is because basically *In Vitro* are simple experiments realized in a tube by cultivating human or animal cells into a glass. The most famous of these are Cancerogenicity studies, but there are plenty of unknown branches of toxicology that can be studied with *In Vitro* tests. One big example is *Mutagenicity studies*. In this section, we explain and discuss three main *In Vitro* tests used by computational biology.

AMES As explained in 2.3 Mutagenicity is the ability of chemicals to cause changes in the genetic material in the nucleus of the cells in ways that allow the changes to be transmitted during cell division. To test the mammalian environment mutagenicity, there is the *AMES test* designed by Ames et al. (1973) [12]. This test is in the middle between *in vivo* and *In vitro*. It is based on rat liver and it is executed in a test tube. The rat liver homogenate is prepared to produce a metabolically active extract. The extract is combined with Strain 1 of *Salmonella typhimurium*, which is a bacteria that carry mutations in genes involved in histidine synthesis. These strains are auxotrophic mutants (normally indicate with his^-), that means they require histidine to grow, but they cannot produce it. A part of the homogenate and bacterial strain is combined with a suspected mutagenic substance and a part is kept as it is, so in the absence of histidine, the bacteria are unable to grow on minimal medium. The method tests the capability of the tested substance in creating mutations that result in a return to a "prototrophic" state (indicated with his^+) so that the cells can grow on a histidine-free medium. The induction of revertant colonies indicates that some his^- bacteria have mutated (reverted) to his^+, and therefore that substance tested is a mutagen. A representation of the process can be found in figure 2.3. Different bacterial strains are sensitive to different types of mutation. Use of a liver homogenate simulates the metabolic breakdown of the suspected mutagen in a mammalian system and more accurately predicts mutagenicity of substances ingested by humans. This test has some limits; in fact, Salmonella typhimurium is a prokaryote; therefore it is not a perfect model for humans. Rat liver is used to mimic the mammalian metabolic conditions so that the mutagenic potential of metabolites formed by parent molecules in the hepatic system can be assessed. However, there are differences in metabolism between human and rat that can affect the mutagenicity of chemicals being tested. Moreover, the reproducibility of this test is around eighty-five percent as reported by Piegorsch and Zeiger (1991) [67]. This implies that the accuracy of *Computation methods* expressed in chapter 3 cannot reach 100% of accuracy.
Mutagens identified in the Ames test need not necessarily be carcinogenic, and further tests are required for any potential carcinogen identified in the test. Drugs that contain the nitrate moiety sometimes come back positive for Ames when they are indeed safe. The nitrate compounds may generate nitric oxide, an important signal molecule that can give a false positive.

We decided to analyze the AMES test, as reported in section 4.3, for two main reasons. First of all, even if it is an *In Vitro* test, this means that it is not subjected to the *Three R rule* by Russell et al. (1959) [70], its cost it is around thousands of euros for each tested chemical. A second reason is that this test is nowadays considered by the regulators as the standard for initial mutagenicity and toxicity test. That is why it is used by industries and organization also during the screening process when the chemical substance it is not synthesized. These reasons make the AMES test one of the most interesting tests to analyze and to study in order to create a correct model able to predict and to replace the current tests. It is one of the aims of this thesis, as discussed in the next chapters.

HCS *High-content screening (HCS)* in cell-based systems uses living cells as tools in biological research to elucidate the workings of normal and diseased cells. HCS is also used to discover and optimize new drug candidates. High content screening is a combination of modern cell biology, with all its molecular tools, with automated high-resolution microscopy and robotic handling. Cells are first exposed to chemicals or RNAi reagents. Changes in cell morphology are then detected using image analysis. Changes in the amounts of proteins synthesized by cells are measured using a variety of techniques. The technology may be used to determine whether a potential drug is disease modifying or for data acquisition for chemical properties.

HTS *High-throughput screening (HTS)* is a method for scientific experimentation especially used in drug discovery realized using robotics, data processing/control software, liquid handling devices, and sensitive detectors. To prepare for an assay, the researcher fills each well of the plate with some biological entity that they wish to conduct the experiment upon, such as a protein, cells, or an animal embryo. After some incubation time has passed to allow the biological matter to absorb, bind to, or otherwise react (or fail to react) with the compounds in the wells, measurements are taken across all the plate's wells, either manually or by a machine. The results of each experiment are written as a grid of numeric values, with each number mapping to the value obtained from a single well.

2.5 Challenges in toxicity

As we have seen in the chapter, in toxicology domain there are many possibilities to improve the test results, to decrease the cost and the abuse of animals during studies. Toxicology is considered at the moment one of the most challenging domains in which it is possible to introduce innovation. That is mainly due to the need of increasing accuracy with a small amount of data. In particular, we can divide the main difficulties into subcategories.

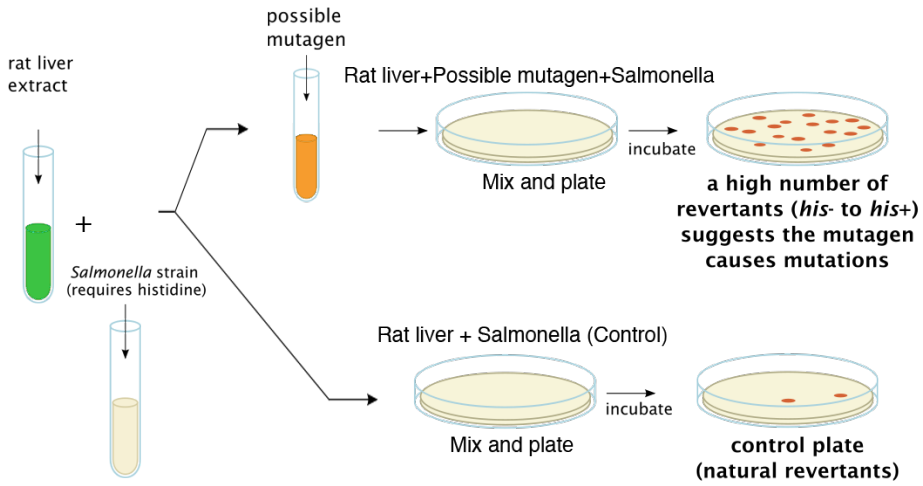


Figure 2.3: The process of the AMES test

2.5.1 Accuracy and variability

As already explained in paragraph 2.2.3 due to the high number of regulations there is not standardization. It means that even in the literature different studies realized in different countries are not always comparable. Moreover, the accuracy of each test could vary based on the type of test, on the days used and on the standardization rules adopted. For these reasons, each study has its own domain applicability and variability. For example, the AMES test that uses *Salmonella typhimurium* is applicable only to mutagens that react in the presence of histidine as presented by Ames et al. (1973) [12]. For example, mixing AMES MPF with AMES will lead to some less accurate results. Depending on the type of test the accuracy could drastically change as we can see in table 2.4. In this work, as written in paragraph 4.3, we collect different tests from different studies and papers, for this reason the sensitivity and the specificity have an upper bond that, at least for the work concerning the AMES test is insurmountable, as also explained in paragraph 2.4.2.
Many of these studies also depend on the number of elements present in the population. As we saw in paragraph 2.4, for environment and humanity reasons, the number of animals is limited and really small. In chapter 3 there are some statistical and computational techniques that allow scientists to overpass this limit with some approximations.

In addition, in the last decade, it has become clear that many non-toxic chemicals produce misleading positive results in some of the regulatory toxicity assays. These doubtful conclusions cost a lot of time and money, as they trigger additional testing of apparently toxic candidates, in both *In vitro* and *In vivo*, to discover whether the suggested hazard is genuine. This, in turn, means that clinical trials can be put on

Tests	Sensitivity	Specificity
Bacterial reversion (Ames)	60	77
Chromosome aberrations	70	55
Mammalian mutation	81	48
Ames MPF	58	63
RAD54-GFP	39	82
DEL	86	80
GADD45a-GFP	87	95

Table 2.4: Some results comparison of different AMES tests reported by Kamber et al. (2009) [43] and Kirkland et al. (2005) [46]

hold. An example is carcinogenicity as explained by Kirkland et al. (2005) [46].
Table 2.4 shows result from the paper by Kamber et al. (2009) [43] and Kirkland et al. (2005) [46] that provides the sensitivity and the specificity of different AMES tests. These reasons combined with the problems concerning the datasets available, discussed in paragraph 2.5.2, transform computational toxicology into a really challenging domain.

2.5.2 Data

In 2009 the *The Toxicity Data Landscape for Chemicals* written by Judson et al. (2009) [40] reported 28 million chemicals already discovered. Of this millions, only three millions were tested on animals and humans, and only one million turned out to have a toxic assay summary. All the successive studies on other compounds are predictions or simulations. It is understandable that with this numbers, find an accurate and optimized model is a big challenge. In particular, the goal of decreasing the cost implies a reduced budget and time to improve the dataset already existing. Moreover, the heterogeneity of the data is very high and this reduces, even more, the reachable precision. It must also be considered that the knowledge about the functioning of the organisms is still insufficient with respect to the complexity of the systems. In particular, the mechanism of the human body, in some parties, is too complex to be reproduced by the modern computer. That is because, even if all the processes are extremely rational and logic, the numbers of possibilities is huge. A possible analysis and prediction technique could be then used to search and define patterns in order to reduce the number of possibilities and the computation time.

The last problem concerning data is the differences between the collection and the procedure used to save them. To do computation toxicology, it is mandatory to have data in machine-readable form; this includes the usage of a common ontology to allow the machine to understand the different sources. The main problem here is the presence of various ontologies. Almost all the datasets use a proprietary ontology that is different from the others. In a scenario like the one just described it is intuitive to

understand the tremendous effort needed to combine different databases from different organizations. The strategy adopted in this work is to develop a new method that does not require apriori knowledge about chemistry and chemicals. In this manner, the use of different ontologies is not required as the machine does not use any information or ontology and can treat all the sources as a unique one.

Chapter 3

Computational toxicology

As already presented in chapter 2, toxicology has improved in the centuries and new branches of this science are currently in development. In particular, we can identify two main complementary areas: the first is the largely descriptive process of determining the effect of a large number of chemicals on the function of various organ systems. These researches have created a large knowledge base of toxicity effects, and recent efforts are looking forward to organize these information into a unique database as described by U.S. Environmental Protection (2010) [80]. The second area is the process of identifying the mode of action of the majority of the agents, usually at a molecular or cellular level. The two parallel tracks have been important in constructing a conceptual framework that can be used from international and governmental organization to assure public health and risk assessment. One of the most important drawbacks is that both mechanisms are labor and resources intensive: for these reasons analyzing all the chemicals on the market is an impossible task as reported by Krewski et al. (2010) [47]. A proposed solution, written in *Computational Toxicology: Realizing the Promise of the Toxicity Testing in the 21st Century* by Rusyn and Daston (2010) [71], has been to create a more rapid screening method based on a mechanism of understanding toxicity. However, mechanistic researches have been reductionist and may not be fully capable of characterizing the full spectrum of targets for agents that affect multiple systems at roughly the same concentration and/or have pleiotropic effects. For these reasons a new branch called *Computational toxicology* or *In Silico* tests has been created. In particular, it can be defined as:

> *Computational toxicology is the application of high-powered computing to manage and detect patterns and interactions in large biological and chemical data sets.*

Computational toxicology takes advantage of three significant technological breakthrough: high-information-content data streams, novel biostatistical methods, and the computational power to analyze these data (Judson et al. (2009) [40]).

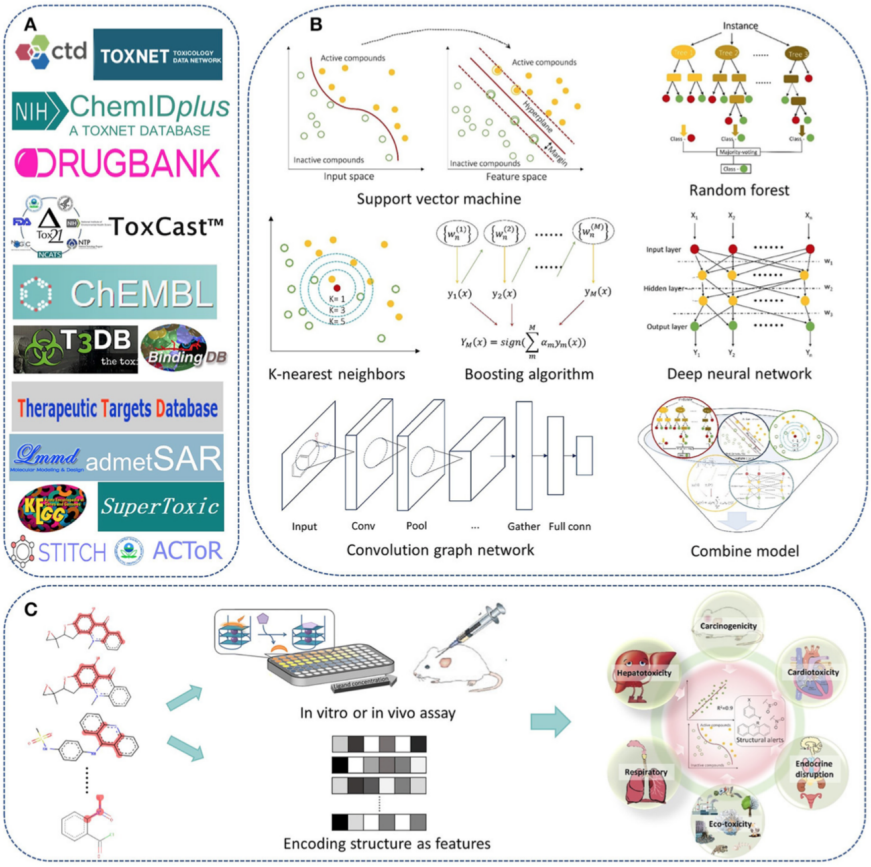


Figure 3.1: A summary of the main methods used in computational toxicology and explained in this chapter

There are various methods used to compute the effects and the reasons of toxicity in chemicals; in figure 3.1, a basic and summarized schema about the possible techniques is represented.

In this chapter, we are going to describe the current state of the art in computational toxicology focusing our attention on the newest and most recent methods. In particular we distinguish two main types of analysis: *Features analysis* in paragraph 3.3 and *Reasoning analysis* in paragraph 3.2.2. Moreover in paragraph 3.1 we discuss the main users of these technologies, the interests and the goals these subjects have in order to obtain a comprehensive understanding of the drivers, and the investments of this domain. In paragraph 3.2 some basic machine learning methods are explained to introduce the themes of the paragraph 3.3.2.

3.1 Users

At the moment there is a great interest from international and governmental agencies in the research of alternative techniques as the ones proposed in this work. That is driven by the necessity of increasing the accuracy about acute and chronicle toxicity and by the new regulations that restrict, or even prohibit (in the case of cosmetics), the *In vivo* tests, to reduce the usage of animals for toxicological trials. Also, industries, of all kinds, are interested in the development of more cheap and fast ways to test chemicals in order to decrease the cost and to increase productivity.
As different users have different interests the explanation must be divided into two paragraphs: 3.1.1 discusses the technical improvements and the cost reduction that computational toxicology can have for industries. Paragraph 3.1.2 instead explains the results required by regulators and governments to accept as *No-hazardous* a specific chemical.

3.1.1 Industries

When we speak about chemicals, it is normal to imagine the pharmaceutic industry as the most interested and the most active in the domain. The truth is that all the companies that use non-tested or non-yet-approved chemicals need to develop a process to do empirical tests on these substances. This goes from pharmaceutical to food and chemical industries.
The analytic process is composed of different phases: *Identification of the composition*, that is the collection of the different mixtures or elements present in the material being analyzed. *Profiling of compounds*. It means the characterization of the different elements present with the list of the features for each one. *Toxicity lead*, this is the identification of all the possible toxic components. That is usually done with some basic tests depending on the type of toxicity we are looking for, an example is the AMES test.
A detailed schema about this process reported to the pharmaceutic world, indicating the cost of each step, can be found in figure 3.2. In there we can see how the costs exponentially increase while the studies keep going forward. It is obvious that with such costly methodology, it is in the interested of pharmaceutic and no-pharmaceutic company to improve the techniques used, in order to reduce the cost.
This type of users usually have also a Research and Development (R&D) department inside the company, in order to speed up the process of production and to invest in new testing techniques. For this users, computational toxicology can be used in different ways: pre-clinical and clinical trial evaluation, drug design and structural alerts. All these methods are explained in paragraph 3.3 and 3.2.

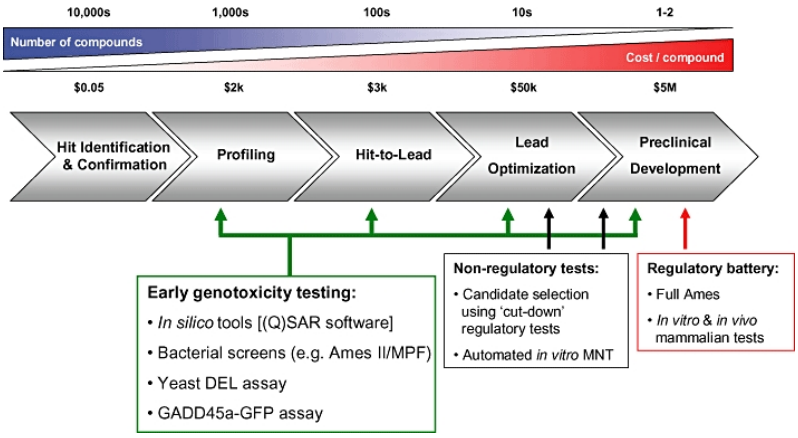


Figure 3.2: The pharmaceutic test process on a chemical in order to be sold on the market. On the top the cost of each individual step.

3.1.2 Regulators

Regulators for toxicology are all the set of individuals or organizations that control and decide the standard and the threshold for a chemical to be considered toxic. Famous regulators include *US Food and Drug Administration (FDA)*, *Environment Health Perspectives (EHP)*, *Environmental Protection Agency (EPA)*,*National Center for Biotechnology Information (NCBI)* etc. In Europe, the main body responsible for the certification of chemicals is *ECHA*, which applies the REACH regulation. These organizations periodically publish the accepted features and characteristics for a chemical to be commercialized. At the very beginning, the regulations concerned only *In Vivo* and *In Vitro* experiments, while *In Silico* was not accepted as a test in order to sell the chemical. With the increase in accuracy and continuous development, regulators have started including more and more models. The current accepted models are discussed in paragraph 3.3 and 3.2. The most important characteristic of these models is transparency. In fact, they are also called *White Box*. That is because regulators need a complete understanding of the functioning of the methods, and of their partial results, in order to correctly evaluate their functionality.
Regulators are also in charge of collecting data, study new methods, and defining the standardization accepted. In the literature many different papers and databases can be found, some are discussed in paragraph 4.3, the most famous are *AcTor*, *ToxCast* and *Tox21*. In those datasets, chemical features and toxicity effects are reported following the accepted rules chosen by the organization who create the collection. These subjects are involved in the development of *Computational toxicology* as it could lead to a new era of chemical control, that means, for example, the assurance that the sold substances are, with a certain precision, hazard-free.

Year	Regulators	Industries	Researcher
2010	<50	<1000	millions
2000	<50	<100	millions
1970	<15	<100	thousands
1960	<10	<50	thousands
1940	<10	<20	thousands

Table 3.1: Data extracted by Judson et al. (2009) [40]. Industries refer only to pharmaceutical companies.

3.1.3 Researchers

These are no-profit organizations, universities and R&D departments. These are considered the main creators of new methods and technologies. Some do it to help the development of toxicology some other instead, do it for the interested of the subjects in paragraph 3.1.1 and 3.1.2. Researchers usually are groups of scientists from different domains that share all the knowledge they have to a common goal, in this case, the creation of new methods to improve toxicology. This group also writes papers and creates databases that are approved and certified by the regulators. In the latest decade, the number of papers produced by researchers has doubled (Judson et al. (2009) [40]).

Between these three categories we can recognize the pharmaceutical industries as the one who has the most quantitative and qualitative data that unfortunately are not public. Researchers are the most active in the domain and they are the most important drivers of changes and improvements. The regulators are, as the name says, the controllers of the current standardization and are the ones who decide the timing of evolution toxicology can be subjected to. In table 3.1 there are numbers of these three categories in different years by Judson et al. (2009) [40], these data includes only the most famous agencies and researchers in USA and Europe.

3.2 Machine learning approaches

Historically, the most popular models, QSAR, (explained in section 3.3.2) have used statistical methods and parametric models on a family of compounds with similar characteristics. With new studies on different chemicals, some new non-linear techniques, taken from machine learning, were needed. In this paragraph a few techniques used in sections 3.3 and 3.3.2 are explained. Some of these can also be found in chapter 4.

3.2.1 Literature methods

In the literature, most of the papers and the models developed use a probabilistic and statistical approach with the main goal to find a mathematical correlation between

their data and specific phenomenon. This pattern research method has been applied since the beginning of computational toxicology. In particular scientists have used methods from computer science, based on mathematical theorems. Some example are *Bayes Classifier* or *SVM*, that are well known in the literature and really used in pattern recognition. Other common techniques derived from computer science are: *Decision tree* and *Random forest* that allow for classifying anything based on its features. In these classification algorithms, it is worth to include also the *K-Mean* that can reduce the computation time and the user expertise required to create the model. However, all the methods described have a common characteristic: at some time in the creation process, they require the interaction with an expert system (usually human) in order to select the features on which it will be based the whole process. In fact, all of the methods just cited are extremely powerful if they are applied to a consistent set of properties. It obvious that these choices must be driven by someone/something which is able to distinguish the differences and the correlation between the different possibilities.

3.2.2 Neural networks and Deep Learning

These are methods usually used from other domains and recently applied to toxicology. Neural networks (NN) are a biologically-inspired programming paradigm, firstly described by McCulloch et al. (1943) [55], which enables a computer to learn from observational data. They can also be described as a mathematical function that maps a given input to the desired output. NN is not an algorithm but can be considered a framework to process complex input data in machine learning. The same network can be applied to many different tasks depending on the data set it has been trained with. A neural network is a collection of connected units, called nodes, that are divided in three categories: *Input layers*, *Hidden layers* and *Output layers* as shown in figure 3.3. Each node is connected to the successive with a link. Each link has a weight w_i that is a real number which is multiplied for the output of each node, and it is passed to the next adjacent one. A *bias* can be added to the sum of these weights to facilitate the network's training. NNs are based on the *Perceptron*, design by Rosenblatt (1958) [69] that can be put in sequence and in parallel to obtain a net similar to figure 3.3. The break-through for neural network can be found in *The roots of backpropagation* by Werbos (1976) [83] which describes the method to train multi-layer networks in a feasible and efficient way. This algorithm allows to calculate the gradient of the *loss function* with respect to the weights in the NN. The weights updates can be done via stochastic gradient descent using:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial C}{\partial w_{ij}} + \xi(t) \tag{3.1}$$

where

η : is the learning rate
C : is the loss function
$\xi(t)$: is a stochastic therm

The choice of the loss and activation function depends on the problem analyzed. The most relevant for this work are *Mean Square Error (MSE)* and *xentropy*.
Concerning Computational toxicology, the main interesting aspect about NNs is that, opposite to the method cited in section 3.2.1, they do not require any user expertise or any apriori knowledge to analyze and discover patterns or to classify. In fact, using the backpropagation algorithm, the network is able to select the most interesting features without any specific suggestion. The main drawback of this method is, however, the strict connections that the results have with the input data. It means that, depending on the quality and on the quantity of data passed to the network the created model will be more or less accurate. It is also the main reason because, until these last years, the most famous computational toxicology methods did not use NNs.
In the last years, data scientists and computational toxicologist have started using neural networks in many different ways, especially applied to toxicology. The main usage is the adoption of neural networks to solve the toxic endpoint classification problem. As we said, it requires many data but also gives high-quality results.
Another common usage done with NNs is the features selection. The main problem cited in section 3.2.1 is the expertise required to select the parameters to work on using the methods cited above. One proposed solution is to use neural networks to select the best chemical features, to achieve the best classification result using some of the common machine learning cited before.
There is also a third possibility for the usage of neural networks. It is the analysis of the data in order to extract the necessary logic knowledge to explain the physical process. It means: given a set of data that concern some toxic chemicals and an NN architecture. We feed the NN with the dataset, and we extract from the network enough knowledge to describe why a particular compound is toxic for a specific endpoint. Unfortunately, it still under development and its usage is really limited because it requires a tremendous amount of data in order to convert the probabilistic knowledge extracted from the architecture into a logic knowledge, able to describe the phenomena.

3.3 Features based prediction models

Since the 1990s, new technologies have been developed and widely applied to produce large amounts of chemical and biological data. Today, high-throughput screening (HTS) and high-content screening (HCS), explained in 2.4.2, are performed on a routine basis in both academia and the pharmaceutical industry, providing information on the biological activity of thousands of compounds The main idea of these models is to extract patterns and find connections on the data collected by these tests. As

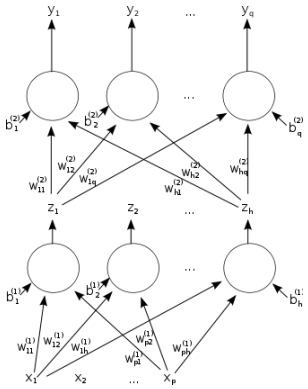


Figure 3.3: A basic neural network with: weights, bias and two hidden layers.

reported by Nigsch et al. (2009) [57] we can divide the features based methods into different categories. All of them can be realized using statistical or machine learning approaches.

3.3.1 Computational filters

A common and easy approach is the research of toxicity-conferring molecular fragments, that are small parts of a molecule that can be associated with toxicity effect following statistical methods. This type of test, applied on a dataset with different endpoints, gives different results. In particular Sander (2006) [72] reported these results:

Endpoint	Accurancy
mutagenicity	90%
tumorigenicity	90%
Skin irritation	90%
Reproductive toxicity	70%

Table 3.2: Result of Sander (2006) [72] for fragment filtering

Results for similar endpoints can be found in literature reported by Foggia et al. (2006) [29], Llorens et al. (2002) [54] and Simon-Hettich et al. (2006) [76].
The main problem with such a method is the consistency of the *Structural Alert* (SA) provided. Frequently the fragments used are too small to be unambiguously linked to the chemicals. Moreover, the set of rules to search inside the input dataset must be decided and these rules are usually linked to the specific database, therefore, not transferable to other dataset.

3.3.2 QSARs and SARs

Quantitative Structure-Activity Relationships (QSARs) are typically used to develop models that are specific to a single defined endpoint to which usually correspond an *In Vivo* or *In Vitro* test. As implicit in the name, such models use only the chemical structure of the molecule to create an association with the biological endpoint. Such models typically predict the activity of a chemical with respect to this specific biological target, or the likelihood of a specific toxic effect taking place. Moreover, in the development of new drugs, such models can be used to flag compounds that are likely to cause adverse effects as proposed by Whitebread et al. (2005) [84].
The use of these models within regulatory bodies, however, is much less established and still under active development as reported in paragraph 3.1.2. The common rules nowadays are two: use more than one model to assess the chemical toxicity and use or design a *white box* in order to analyze the partial results.

There are many possible technologies to create a QSAR; the most used are explained below, the majority of them is taken from the machine learning algorithm, also reported by Baskin (2018) [15], in section 3.2. Some famous models to predict the mutagenicity emulating the Ames test are discussed in this section. Moreover, many papers have studied the different results for the different endpoints in order to compare the different models.

CAESAR In the CAESAR model for mutagenicity, reported by Ferrari and Gini (2010) [28], there is an SVM classifier followed by a *Structural alert* (SA) search that helps to reduce the number of false negative, i.e., the number of chemicals predicted as non-toxic that are toxic. The model uses 25 descriptors, four of which are global, the other are atoms count made on the 2D fragment. The model is a cascade with a C-SVM, firstly designed by Vapnik and A.Lerner (1963) [81] that is part of the SVM family. It finds the *maximum margin separator hyperplane* in the input space. It uses a *Radial basis function* as kernel function. Once the chemical is passed through The C-SVM, and if it is marked as mutagenic the process ends. Instead, if it is non-mutagenic, the chemical is analyzed with a collection of 10 SA taken from Benigni et al. (2007) [18]. If the result is still non-mutagenic another set of 5 SA can be applied. This process allows a great result, near to the 85% reproducibility of the AMES test.

SARpy From a survey of the (Q)SARs for chemicals, it appears that a number of important models are based on the mechanistic knowledge formalized into the SAs, and are aimed at detecting SAs in the query chemicals. *SARpy* that stands for SAR in PYthonm described by Ferrari et al. (2013) [27] is a new ad hoc SAR approach aimed at finding relevant fragments in a transparent way, extracting a set of rules directly from data without any *apriori* knowledge. The fragment candidates to become SAs are automatically selected on the basis of their prediction performance on a training

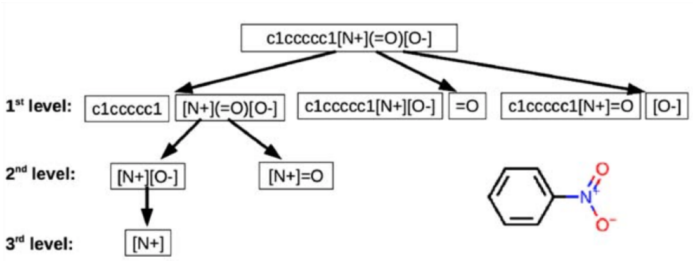


Figure 3.4: An example of the process used in SARpy.

set. The output of SARpy consists of a set of rules in the form:

$$IF\ contains\ <SA>\ THEN\ <apply\ label>$$

where the SA is expressed as SMILES, explained in paragraph 2.2.2.
The mining process done by this method is the research of the correlation between the incidence of a molecular substructure and the activity of the molecules containing it. It is composed of three phases, that are also in figure 3.4.

- *Fragmentation*: The recursive algorithm considers every sub-sequence of the SMILES. In particular, it performs a rough fragmentation of the input structures, but by iterating each fragmentation step on the output of the previous one, it collects substructures of increasing complexity until the in-depth fragmentation of the original structures is completed.

- *Evaluation*: Every substructure is treated as an SA and it is matched on the training set to calculate the correlation. The algorithm uses a *likelihood ratio* that is:
$$Likelihood_{ratio} = \frac{TruePositive}{FalsePositive}\frac{negative}{positive} \tag{3.2}$$

- *Rule set extraction*: A reduced set of rules is extracted from the huge set obtained before. This phase is also an iterative process that consists of: Order the list of potential alerts by likelihood ratio. Select the top-ranked one, add it to the rule set, remove it from the list of potential alerts, and update TP, FP and likelihood ratios values of the remaining potential alerts. Finally, return to point 1.

This method is particularly efficient and has been used in many successive papers to extract a fragment from the data set studied, including mutagenicity and carcinogenicity.

3.3.3 Target based prediction

Quantitative Target-specific Toxicity Prediction Model (QTTPM) approach integrates molecular dynamics (MD) simulation and machine learning. It uses a variable called

dyPLIDs that represents the dynamic interaction between Ligands and receptors. Developed by Gong et al. (2018) [32] it differs from the QSAR as it concentrates on the organic target of the compounds. It incorporates simulation testing and random forest method in order to evaluate specific organ toxic chemicals. At the same time the most important innovation introduced by QTTPM model, it is its biggest limitations as it implies the test and the simulation to be applied only on a specific cell.

3.4 Existing softwares

Some of the methods and models explained in 3.2, 3.3.2 and 3.3.3 are non-commercial, other instead are integrated into commercial software and are used by regulators to validate new discoveries. Some of these programs are made by regulators themselves or by research institutes. Some famous names are:

- *VEGA HUB* [10]. It can access a series of QSAR models for regulatory purposes. It has been developed based on the models produced by *EU Demetra* project, about the hazard of pesticide, and by the CAESAR project for the study of the human toxicology.
- *DEREK*, is an expert system for the prediction of toxicity (genotoxicity, carcinogenicity, skin sensitization, etc.)
- *CAESAR*, created by Cassano et al. (2010) [19], is a java web application that allows the access to all the toxicity predictive models developed within the CAESAR Project.
- *DEMETRA* is an EU-funded project. This project aim has been to develop predictive models and software which give a quantitative prediction of the toxicity of a molecule.
- *T.E.S.T.* Toxicity Estimation Software Tool enables users to estimate acute toxicity easily, by EPA (2016) [25]
- *Toxtree* is a full-featured and flexible user-friendly open source application, which is able to estimate toxic hazard by applying a decision tree approach

Chapter 4

Three-Tox

Chapters 2 and 3 have introduced the concept of toxicity and the most common techniques used in computational toxicology. In this chapter we start designing the model proposed in this work.
As already said, we wanted to free our model from the usage of *apriori knowledge*, which is partial and may constitute a bottleneck more than an opportunity. To do so, we designed the architecture to be the most modular and independent as possible. We divided the architecture into three main parts: two of them designed to extract the necessary data features from two different forms of input for the same chemical. The third part is used for finding the non-linear correlation between the features extracted and the label associated with the input. In this chapter, we first analyze the current open challenges we had to face during the development of this work, in section 4.1. Then in section 4.2 we describe the current state of the art of deep learning applied to computational toxicology. Finally in section 4.3 and 4.4 we explain the data collected, the pre-processing method proposed and the central proposal of the work.

4.1 Open Challenges

As we have seen in the precedent chapters, Computational Toxicology still has challenging problems to be solved. With our proposal, we want to add a few more steps to solve them. There are three main open challenges in this domain. The first and more intuitive one is about data. Available results of tests should be organized, standardized and ideally made available in public locations. Unfortunately, neither standardization and global public sources exist at this time. There are many different sources with different regulations that could lead to inconsistent data and to not comparable methods. This limits the development of a global model that could be applied to the whole domain. This constraint also implies the need to create a proprietary set of tests in order to analyze the specific phenomena.
Moreover, even where a big and well-defined source of data exists, computational tox-

icology is extremely influenced by the current domain knowledge available and by the expertise of the data scientist who is creating or using a specific model. It leads to the definition of the second big challenge of computational toxicology, that is the bias introduced by the human conducting the experiments or building the models. As stated in section 3.2.1 the current state of the art is biased by the chemical and biological knowledge we currently have. That is because both the settings and the interpretation of the models are influenced by previous researches and by the best practices defined in different periods. Nowadays features selection is normally based on statistical methods where data already exist; or it is driven by previous researches where data needs to be collected through tests. In both cases, there is a constraint that ties together the results and a certain applicable domain (called endpoint in toxicology). In this work, we want to eliminate at least one of these constraints by creating an architecture that is unrelated to the available chemical knowledge. This choice helps not only the reduction of the bias we described but also opens this domain to a bigger number of scientists, including computer science engineers, that right now can not work on this domain without a deep understanding of chemistry and biology. However, the proposal of this thesis still needs the presence of a database and therefore still it needs to individuate an endpoint; this task is driven by an analysis of the importance of the endpoint and by the availability of experimental tests on the chemicals.
The third big challenge everyone has to face entering the computational toxicology field is the necessity of regulation and definition for every step taken during the creation process. That is the documentation and the test required by the regulators in order to validate the model and to allow its usage in production and marketing. This implies the possibility to analyze the partial result of the model as well as the final results. We do not aim to create a method to be approved from the regulators, but we propose however a method that allows extracting the knowledge developed during the training part. Doing that we allow the introduction of a new concept of knowledge that it is not derived from the human experience but instead from the model itself.

4.2 State of the art in deep learning for computational toxicology

The applications of Deep learning models have tremendously enlarged their applicability domain in the last years, starting from *Image analysis* to *NLP* passing through *Vocal recognition*. Also, the applications in medicine are copious. Yet, computational toxicology seems not to be really affected by this evolution. In fact, there exist only a few studies that apply a deep learning architecture to the toxicology classification. The most famous is Goh et al. (2017) [31] who proposed an existing solution developed by *Google* and applied to toxicology, achieving great results. Different considerations drive this choice. First of all the presence of data that, especially in deep learning, becomes of fundamental importance. However, we decided to apply deep learning

techniques developed *ad hoc* and inspired by the most advanced researches in computer science in order to avoid the disadvantages of the traditional way QSAR models are built. In particular, as we want to disconnect the model from the bias derived by the presence of apriori knowledge, we decided to use deep learning techniques that allow the architecture to be only data-driven without introducing anything but the chemical itself.

4.3 Data

As we said, the data is one of the most important parts of computational toxicology and computation methods in general. There exists a variety of different sources that contain data linked to toxicology; in table 4.1 are listed some of the most complete and used in the literature, grouped by the type of chemicals they contain. In order to accomplish the first goal of this work, to avoid introducing apriori knowledge in the model, we need a big and coherent source of chemicals with the related toxicity. Moreover, to develop the best model we also need to simplify the structural complexity. It means that we treat only one toxic endpoint while maintaining the possibility to add others in a second version. Specifically, we considered only mutagenicity studied through the AMES test. The main reasons are explained in section 4.3.1. Moreover, the creation of our database has followed a specific process. It started with collecting data from the literature, followed by the data pre-processing. Sections 4.3.2 and 4.3.3 describe both the two phases.

4.3.1 Why AMES

We decided to create only one function correlating the chemical structure to one toxic endpoint, mutagenicity. The mutagenicity, as explained in chapter 2 is tested using the AMES test. We chose this particular test for different reasons. The first parameter that drove our choice is the interest reported by all the different users listed in section 3.1. This test is necessary to all the chemical producers in order to commercialize any substance. The interest in the computational model for this test is also driven by the cost the AMES test has. It is an *In Vitro* test, that is generally cheaper and faster than the *In Vivo*. Yet, the AMES test is comparable in cost with any other *In Vivo* tests. Moreover, the different standardizations defined for this test do not allow the industries to use the researches made by others in different countries. That is because different governments have different regulations concerning the specifications of salmonella strains in the AMES test. Therefore, usually, it is not possible to mix the results of the same substance tested with other test characteristics.
The second factor that influenced our choice is the number of data available in literature and on the web. Due to the high level of interest in the domain, it is possible to find thousands of chemicals tested with the AMES test. However this data are usually private, this means that we can use them to build the model as we are a research

		N	Vitro	Vivo	C	H	A	M
Drugs	Withdrawn	578			✓	✓		
	WHOCC	-		✓				✓
	HSDB	-				✓	✓	
Compounds	Open TG-Gates	170	✓	✓	✓	✓	✓	
	PubChem	500.000.000	✓	✓	✓			✓
	ZINC15	200.000.000	✓	✓	✓			✓
	ToxCast, Tox21	8.599	✓					✓
	Toxline	-			✓	✓	✓	✓
	ChemIDPlus	400.000	✓		✓			✓
	EBKB	-			✓	✓	✓	✓
	EcoTox	11.000				✓	✓	
	CandLInventory	-			✓			✓
	Tox and Disease	-			✓			✓
	ATSDR	-			✓			
	AcTor	-			✓			
	DssTox	-	✓		✓			✓
Molecules	TOXMAP	-			✓	✓		
Genes	GeneTox	3000				✓	✓	
Risk Assasement	ITER	-				✓	✓	
	Danish QSAR DB	600.000				✓	✓	✓
	OpenFoodTox	4.000			✓	✓	✓	
	CEE-TV	20.000				✓	✓	
Pathway	KEGG	8.599	✓			✓	✓	✓

Table 4.1: A representation of the data contained in the different databases analyzed.

N : is the number of elements contained in the database.
$Vitro$: is checked if the data contained are derived from *In Vitro* tests.
$Vivo$: is checked if the data contained are derived from *In Vivo* tests.
C : is true if the database contains other features about the chemicals.
H : true if the reported tests concern Humans
A : true if the reported tests concern Animals
M : true if the reported tests concern other molecules

institute, but we can not share this data with anyone.
Of course, the choice of the AMES also implies some drawbacks. The most important is the different standardization of the available data, which requires data pre-processing and more caution depending on the source they come from. Different standardization also implies different standard for the storage and the categorization of the chemicals used. In particular, as the chemical structure is the only input value, we are interested in the algorithm used to write the SMILES that identifies the chemical; since a structure can be defined with different SMILES, we need a consistent method to produce it.

4.3.2 Data Collection

As we said, the execution of the AMES test is expensive so the results are often proprietary. In order to collect a sufficient number of chemicals tested we researched in the literature and on the web. This choice, however, has a cost, as explained in section 4.3.1. As the collection of chemicals was partially realized on the web, we had to distinguish between trustworthy sources and suspicious sources. We defined a trusted source as

> *A source that is published from certified authorities or that has been used in computational model approved by the regulators*

Unfortunately, as we said before, there are not too many trusted source due to the list of requirements requested by the regulators. The main trusted source we could find is created by NIHS *National Institute of Health Sciences, Japan (DGM/NIHS)* as part of their *Ames/QSAR International Challenge Project*. It contains around twelve thousands compounds. Most of the chemicals in this dataset are pharmaceutical or industrial products. This is another important point to investigate as the origin of the chemicals analyzed could influence the distribution of the toxicity and therefore the accuracy of the results. Between the trusted sources we also included two databases created by the *National Cancer Institute (NCI)*. These are: *GeneTox*[2] and *CCRIS*[1]. We also used data from the *VEGA Hub* that merges most of the cited database.
We also used suspicious sources, one specifically: *CGX* [22]. This source contains various AMES test type for each chemical and defines as positive to AMES test, a chemical where more than three tests result positive for it. However, we wanted to have a more strict definition of a mutagenic chemical. That is because we wanted to reduce/minimize the number of false negative predictions, i.e., chemicals predicted as negative which are actually toxic. In order to do that we considered as positive to AMES test the chemical if just one of the five tests resulted positive. This choice is based on the request of the regulators to have a high level of sensitivity in order to be sure that the compound tested is not toxic.
Between trusted and suspicious sources we also found a list of really cited sources that however are not included in the definition of trusted. The source are created

by Kazius et al. (2005) [44], Helma et al. (2004) [37], Feng et al. (2003) [26], Judson et al. (2005) [41], and are all regrouped by Hansen et al. (2009) [35] in its article: *Benchmark Data Set for in Silico Prediction of Ames Mutagenicity.* Another paper that groups together the different databases is written by Benfenati et al. (2018) [16], who also described the current state of the art about computational toxicology.
All these different sources contain pharmaceuticals, pesticides and industrial products. As we have so many sources and so many regulations, during the collection process we paid particular attention in the selection of the SMILES. In particular, excluding the trusted sources we just mentioned, for every result found we compared it with the trusted databases in order to find duplicates, and we eliminated all the incoherent tuples. Moreover, we kept the database source during the whole pre-processing phase in order to eliminate at each step the duplicated derived from a dataset that was not marked as trusted. This caution is necessary because, due to the high number of possible AMES test, and to the fact that we do not use any other knowledge but the data, these grouped data need to be coherent and well structured. Unfortunately, it was not possible to collect also the full specifications of the conducted tests. That is because the different sources do not always report the standards applied to obtain the results.
Another important information we would like to have is the use of each compound. The chemical type tested in fact could explain some results from the proposed model, and also could define better the results obtained for the AMES test. In fact, there are different regulations for the toxicity test depending on the final usage the chemical will be used for. Unfortunately, in all the databases we collected there is no such information. One method to obtain it could be to create a system able to understand the natural language to search each specific compound on PubChem[6].

4.3.3 Data Preprocessing

Figure 4.1 presents the whole preprocessing method we used on the data collected from the literature. The *stage 1* is described in section 4.3.2. The output of the literature review is split into two, based on the source of the data. The data coming from trusted sources are used as a baseline to remove the duplicates and the incoherent data in the second set. In *stage 4* we performed a basic search on both sets in order to discover if some of the data were replicated or had different values. The basic rules used to remove the tuple are:

<IF> *trusted.contains(suspicious)*
<THEN> *remove from suspicious*

<IF> *suspicious.duplicate* <AND> $toxicity_1 == toxicity_2$
<THEN> *remove the duplicates*

<IF> *suspicious.duplicate* <AND> $toxicity_1 ! = toxicity_2$
<THEN> *remove the negative duplicates*

As we can see in the third rule, we made another safe choice. In order to reduce the number of false negatives, we kept the toxic values. It is done because we wanted to balance the presence of toxic compounds as the data collection is unbalanced and has more non-toxic chemical.
This operation is repeated also in *stage 6, stage 8* and *stage 11*.

The normalization function in *stage 5* consists in the *cleaning process* of the SMILES string collected. With *cleaning process* we mean the process of balancing the charge of the molecules, the recognition and removal of disconnected substructures. We could have some specific compounds that are chemically unbalanced (i.e., a wrong number of electrons and protons that define the charge). Moreover, some SMILES could be disconnected, i.e., composed of two disconnected substructures. This normalization process is essential for two reasons: the former is that the software used in *stage 7* accepts only chemicals in a specific form. The other reason is that this process allows in giving all the collected compounds a similar standardization. The output of this process is a set of chemicals that are all balanced and have no disconnected strings. They still, however, have different standardization about the SMILES form. Another module, which is *duplicates removal*, is performed in order to keep the database coherent during the process. In listing 4.1 is reported the code used, with the related comments, to realize this function.

Listing 4.1: The function used to normalized and Neutralize the charges and the disconnected string in the chemicals collected

```
def NeutraliseCharges(self, reactions=None):
        if reactions is None:
            if reac is None:
                reac=InitialiseNeutralisationReactions()
            reactions=reac
        mol = Chem.MolFromSmiles(SMILE)
        replaced = False
        for i,(reactant, product) in enumerate(reactions):
            while mol.HasSubstructMatch(reactant):
                replaced = True
                rms = AllChem.ReplaceSubstructs(mol,
                            reactant,
                            product)
                mol = rms[0]
        if replaced:
            return (Chem.MolToSmiles(mol,True), True)
        else:
            return (SMILE, False)
```

Stage 7 consist in the evaluation of the collected compounds by the software *VEGA HUB*[10]. This allows to transform the SMILES string and to rewrite it using the same algorithm. This action is fundamental because, as explained in section 2.2.2, different algorithms produce different SMILES strings. This process is also helpful to find new duplicates in the dataset. In fact, in *stage 8* we remove the duplicates again with the

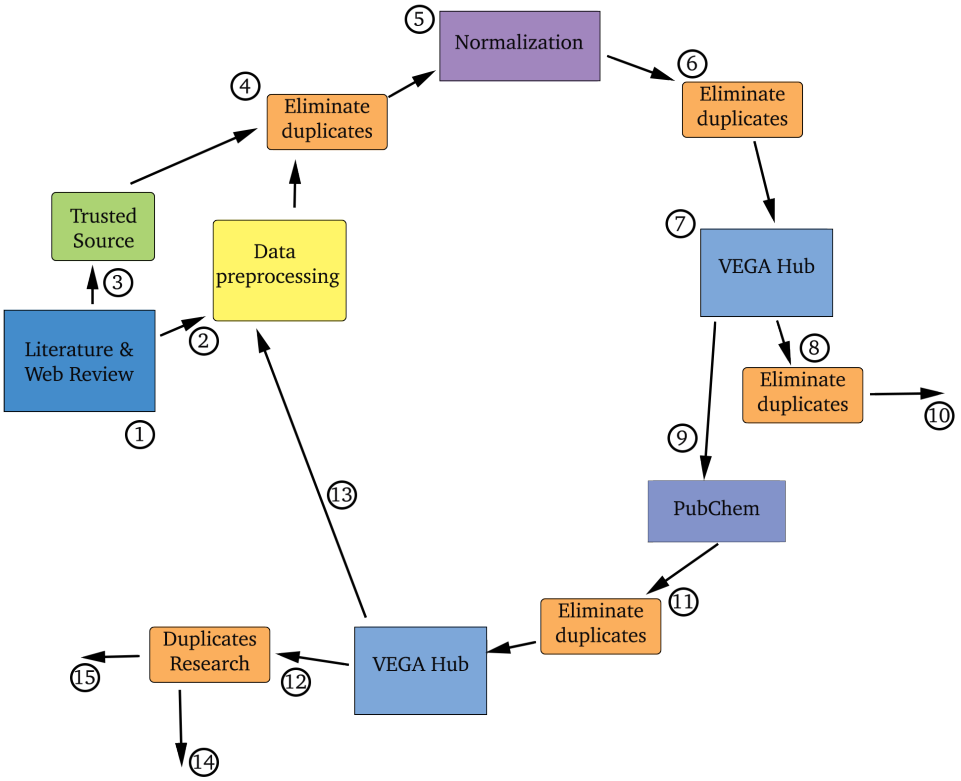


Figure 4.1: The process followed to collect and create the database.

function described before. The output, in *stage 10*, is then stored in the database. However, the software eliminates also a list of strings, possibly due to errors in the string format, that *VEGA* does not recognize as SMILES. In order to not lose useful chemicals we inserted a second preprocessing phase. It is composed of various steps that allow recovering some of the discarded compounds.

We wanted to maintain the number of chemicals as high as possible while also maintaining the quality of the dataset. For this reason, we introduced *stage 9*. In *stage 9* we used the compounds discarded from *VEGA*, and using the available API, we consulted the *PubChem*[6] database. This because using this API we could search for different SMILES string corresponding to the same chemical. It is done by either using the CAS-Number, that is a unique identifier for chemical, or by submitting a SMILES string.
The results obtained from this query have been then removed from the duplicates (we also considered the compounds stored in *stage 10*) and parsed again inside *VEGA*, in *stage 11,14*. As described, *VEGA* can discard some of the input string, so we added the final discarded values as new suspicious source and we restarted the cycle, in *stage 13*.

Different from the other duplicate removal block, in *stage 12* we performed a duplicate retrieval. Instead of deleting the copies we kept the duplicates in another database storing them in a different form. We used a code proposed by O'Boyle (2018) [59] to generate others smiles of the same molecules. That is because, as explained in 2.2.2, different SMILES string generates different image structure. The reasons to create this "*parallel database*" of duplicates is that we wanted to test if the proposed model is *SMILES invariant*, this means that the predictions do not rely only on the input but that the network proposed is actually learning some basic knowledge of chemistry.

4.3.4 Final Database

The final database created has almost 22 thousand, 21.923 to be specific, compounds with AMES test results, where the initial literature collection contained around 32 thousand chemicals. Moreover, we collected a database of duplicate composed of 2 thousand compounds (2080). Denoting the importance of the preprocessing phase is important. Even if it reduced the number of data to work with, it helps to maintain the model accuracy and the quality of the results.
This database has some critic points that need to be remarked in order to interpret the outcome of this work correctly. One big challenge in this database is the high level of unbalanced data as is visible in table 4.2. It is due to the distribution of known toxic chemicals present at the writing time. It implies that the result could not pass an upper bound in the specificity and sensitivity values.

In order to have a bigger picture on the data collected and to define an applica-

	Number	Percentage
Positive	8127	33,85%
Negative	15876	66,14%
Total	24003	100%

Table 4.2: Evaluation of database and its balance.

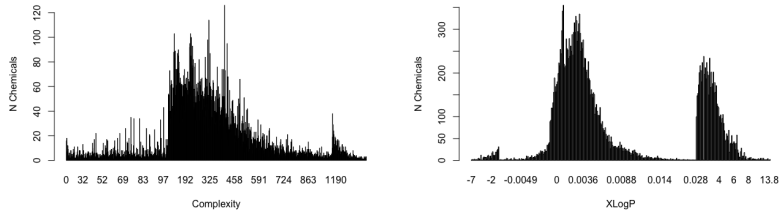
bility domain we completed the database with some basic information concerning the features normally used in machine learning and QSAR models. We did not use however these features in the architecture and we used them only to compare the database created with the others presented in literature.
We used some features that are not explained in the introduction: *Complexity*, defined by Whitesides and Ismagilov (1999) [85] as:

> The characteristic of the behavior of a system or model whose components interact in multiple ways and follow local rules, meaning there is no reasonable higher instruction to define the various possible interactions.

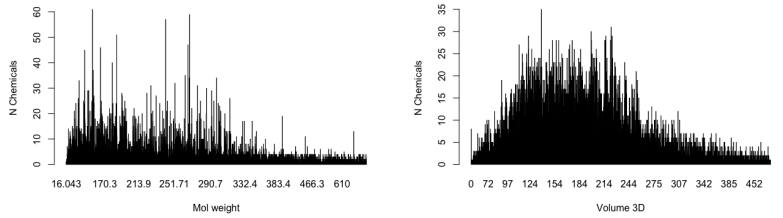
In particular in figure 4.2a shows that the data collected have in general *middle complexity*, *i.e.* the structures of the chemicals have elements strictly connected to each other but the chemical itself is less reactive with other substances.
It is also interesting to compare in figure 4.2d and 4.2c the volume of the molecule with the molecular weight, that is the sum of the atomic weights of each constituent element multiplied by the number of atoms of that element. The former is Gaussian-distributed and the latter instead has an uniform distribution, which implies that the chemicals considered have a high internal density.
Finally figure 4.2b shows the *XlogP* that is an atom-additive method for calculating the octanol/water partition coefficient (logP). It gives the logP value for a give compound by summing up the contributions from component atoms and correction factors. The logP value is also known in physic as the logarithm of the partition coefficient. That is defined as a particular ratio of the concentrations of a solute between the two solvents (a biphase of liquid phases), specifically for un-ionized solutes. It is calculated with either formula 4.2 or 4.1 depending if the chemical is ionized or not. As showed in the figure 4.2b then the chosen compounds are either water-soluble or water-repellent, there are middle options.

$$\log\ P_{\text{oct/wat}} = \log\left(\frac{[\text{solute}]^{\text{un-ionized}}_{\text{octanol}}}{[\text{solute}]^{\text{un-ionized}}_{\text{water}}}\right) \tag{4.1}$$

$$\log\ P^{\text{I}}_{\text{oct/wat}} = \log\left(\frac{[\text{solute}]^{\text{I}}_{\text{octanol}}}{[\text{solute}]^{\text{I}}_{\text{water}}}\right) \tag{4.2}$$



(a) Complexity distribution **(b)** XLogP distribution



(c) Molecular weight distribution **(d)** Volume distribution

Figure 4.2: Characterization of the chemicals collected, taken from the data of PubChem.

4.4 Architecture

T-Tox stands for *Three-Tox*. This name is inspired by the fact that the proposed model contains three different networks, *Toxception*, *SMILES-Net* and *C-Tox*. Figure 4.3 shows a schematic representation of the architecture.

The first two, *Toxception* and *SMILES-Net* receive in input a SMILES string indicating the chemical form of the compound, and they transform it in different ways. Toxception is the graphical analyzer of the model and is charged with the creation of the image of the compound starting from the SMILES. On the image, it proceeds to a features extraction using CNN.
SMILES-Net instead takes the input as it is, in the SMILES format and uses a method called SmileEmbedding to convert the string into numeric form. Successively it passes the chemical through an LSTM network. It also performs the feature extraction on the string. The outputs of those two models are then merged together and passed to C-Tox, that is a basic feed-forward network.
Both the networks also have a visual output applied to the input that allows the users to visualize the results. It is really useful to test the knowledge learned from the networks, as one of the aims of this thesis is to develop a prediction model that does not use apriori chemical knowledge and gives a basic toxicity knowledge as output. It means that it can be applied to any toxic endpoint and it does not require any chemical features; it only requires as input the chemical formula (SMILES) and a boolean vector indicating the toxic endpoint.
In the following chapters, we explain the different parts of the proposed model. Each of

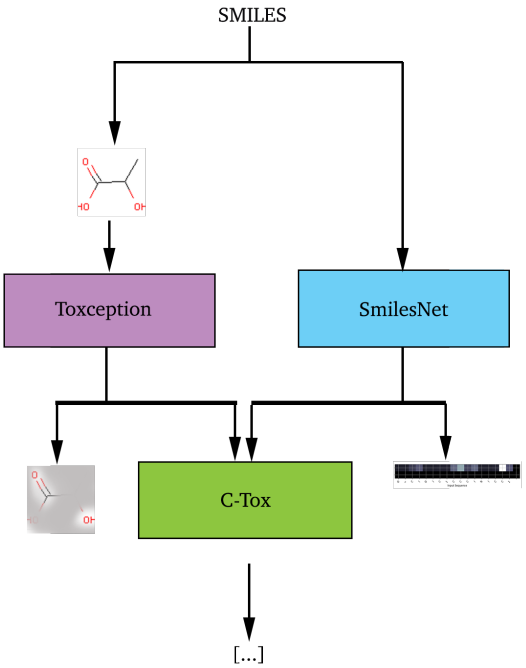


Figure 4.3: A schematic representation of architecture of T-Tox.

the chapters contains an explanation about the current state of the art of the chosen method (not yet applied in toxicology). Moreover, it describes the implementation and the results that each of the parts gave while trained alone, except for C-Tox that reports the result of integrating the models.

4.4.1 The output

As output of this model, we decided to use a number from zero to one for each endpoint. This means that the model executes a process similar to regression. Considering the AMES test endpoint, it could be obvious to define it as a categorization, multiple or binary. However, the classification inserts a big constraint that may erroneously simplify the model. In fact, as we saw in section 2.2.3, a chemical can move from toxic to non-toxic simply by changing the dose given to a specific individual. Moreover, the toxicity is tested on a subset of the population and this could not apply to all the individuals. These reasons make the definition of classification, that does not have grades of toxicity but only boolean values, too restrictive. Plus, the output of the model can be interpreted as the probability for the compound to belong to a specific class.
However as we have only boolean values as labels, and due to some considerations, which see mutagenicity as a one-shot event not related to dose, we keep the classification methods available, and we evaluate the prediction by rounding the output values

of C-Tox. We did this distinction by using two loss functions during the training process: *MSE*, *Mean Square Error* and *xentropy.*
MSE is defined as

$$C(w,b) = \frac{1}{2*n}\sum_{x}\left\|y(x)-a\right\|^2 \tag{4.3}$$

Where:

w : is the collection of weights of the network,
b : is the bias matrix of the network,
n : is the total number of training inputs,
a : is the vector of outputs from the network when x is input.

The *Cross-entropy* loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of 0.012, when the actual observation label is 1 would be bad and it will result in a high loss value. It can be calculated as:

$$-\left(\tilde{y_l}\log y + (1-\tilde{y_l})\log 1 - y\right) \tag{4.4}$$

Where:

y_l : is the label
y : is the predicted value
log : is the natural log

As we said before, we used the model only on one endpoint but we actually realized it to support different endpoints. So, in this case, the *Cross-entropy* would be:

$$-\sum_{c=1}^{M} y_{l,o,c}\log y_{o,c} \tag{4.5}$$

Where:

$y_{l,o,c}$: is the label for the observation O of the class C
y : is the predicted value for the observation O of the class C
log : is the natural log
M : is the number of classes

4.4.2 Evaluations

As explained in section 4.4.1, and in the next chapters. We consider this problem as a single class toxicology regression problem, this means that each molecule has as output a number between 0 and 1 that indicates the probability (given our training

set) to belong to a particular class, in our case, to be toxic. For traditional single-label binary or multiple classification models, most of the performance metrics are calculated based on the count of true positive (TP), true negative (TN), false positive (FP), and false negative (FN). We define accuracy as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \tag{4.6}$$

Accuracy is not a reliable metric for the real performance of a classifier, because it will yield misleading results if the data set is unbalanced as in our case. For this reason we decided to apply two additional metrics:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{4.7}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{4.8}$$

These formulas can help the analysis of the results and are used in this work to detail the values obtained by the accuracy output.
Moreover the performance metrics can be divided into two main groups: *example-based* and *labeled based* metrics. In the label-based there are: *subset accuracy*, *Jaccard similarity coefficient*, *hamming-loss*,*precision*, *recall*. Yang et al. (2018) [86]. Showed in formula 4.10.

$$\text{Precision} = \frac{1}{n}\sum_{i=1}^{n}\frac{\|Y_i \cap Z_i\|}{\|Z_i\|} \tag{4.9}$$

$$\text{Recall} = \frac{1}{n}\sum_{i=1}^{n}\frac{\|Y_i \cap Z_i\|}{\|Y_i\|} \tag{4.10}$$

where

Y_i : represent the real i-th label
Z_i : represent the predicted i-th label
n : is the number of instances

Using this equations it is possible to define some other basic metrics that are used in machine learning and are useful to compare the results between different works.

F1 the F1 score is a measure of a test's accuracy. It considers both the precision p and the recall r of the test, to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples.

$$\text{F1} = \frac{2 * TP}{2 * TP + FP + FN} \tag{4.11}$$

MCC The Matthews correlation coefficient (MCC), introduced 1975, is used in machine learning as a measure of the quality of binary classifications. It takes into

account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are not comparable in size. The MCC is in essence a correlation coefficient between the observed and predicted binary classifications; it returns a value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 means no better than random prediction and -1 indicates total disagreement between prediction and observation.

$$\text{MCC} = \frac{TP * TN - FP * FN}{\sqrt{(TN + FP)(TN + FN)(TP + FP)(TP + FN)}} \tag{4.12}$$

4.4.3 Frameworks

All the technologies and theories explained in this work has been implemented using *Python 3.6* language. The complete code can be found on GitHub. In particular, some frameworks have been used in order to decrease the complexity of the implementation of the work realized. In this section, we analyze the technologies used in order of importance.

Tensorflow *TensorFlow* [9] is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the *Google Brain* team within Google's AI organization, it comes with strong support for machine learning and deep learning, and the flexible numerical computation core is used across many other scientific domains. It is used especially for neural network development and it is the most common software used by researchers and industries. The choice of using *Tensorflow* instead of other common frameworks is mainly driven by the existence of a pre-written driver for the GPUs used and by the highly detailed documentation.

Keras *Keras* [5] is a high-level neural networks API, written in Python and capable of running on top of *Tensorflow*, *CNTK*, or *Theano*. It was developed with a focus on enabling fast experimentation. It is a common choice to speed up the writing phase of the code. This software is used to abstract the code from the backend part, in this case, tensorflow. It allows disconnecting the code layer from the GPU instructions completely.

RDKit *RDKit* [7] is a collection of cheminformatics and machine-learning software written in C++ and Python. The core data are structured in C++ and it has a wrapper coded in Java, C# and python. This framework allows to analyze the chemicals with different operations:

- Create the 2D and 3D structures starting from the SMILES string.

- Obtain atomic charge of the molecules and balance the electron number at the atom level.
- Descriptor calculators and similarity search
- Fingerprint generation for machine learning

This software is used in many software tools listed in section 3.4 and it is used in this work especially in section 4.3.3

Talos *Talos* [8] is an hyper-parameter optimization tool based on python and used on keras. It helps the choice of the main hyper-parameters by analyzing different possibilities and showing a detailed report about how well the machine is performing, in which time and with which parameters. This util allows to speed up the process of running all the different possible tests.

Other Python packages A few more packages have been used in order to analyze and plot the results. In particular the most famous are:

- Numpy used to manage the array and tensor
- Sklearn, that implements the basic method of machine learning and dataset management
- Matplot, in order to plot the result and to manage images.
- Cameo, to create images. RDKit also uses it.
- Open-CSV used to manage *csv* files and to collect information from the different sources.

For all of them, there is an active community that supports and help the development of the packages and their usage.

Chapter 5

Toxception

Image analysis and object recognition are two of the most active research topics at this time. The main idea is to train the computer as a child, by showing different examples of the same object, in order to make the machine understand and recognize the object based on the images features. This problem is usually called *Large Scale Visual Recognition Challenge* (LSVRC). In this chapter, we are going to analyze, in section 5.1, the different techniques used in different times to do this task, in order to understand the currently available technologies. In section 5.2 we discuss the main theory behind Convolutional neural network. Then in section 5.3 we explain the proposed network, Toxception, and its theory based on Szegedy et al. (2017) [79] and Goh et al. (2017) [31]. In section 5.4 we explain why we chose this model based on some comparisons.

5.1 LSVRC history

A first solution to the LSVRC problem was proposed by Krizhevsky et al. (2012) [48] with the *AlexNet*. Since then, every year, the *ImageNet* contest, *i.e.* a challenge on the big dataset ImageNet [4], declares the winner by publishing the network which proved to have the best accuracy and the smallest error. A graph with the evolution of the error and the networks in the different years can be found in figure 5.1. More recently, deep learning has also begun to emerge in other fields, such as high-energy particle physics, astrophysics and bioinformatics. In chemistry, a few notable recent achievements include DNN-based models winning the Merck Kaggle challenge for activity prediction in 2012 and the NIH Tox21 challenge for toxicity prediction in 2014. In LSVRC the most powerful model turned out to be *Convolutional Neural Network*(CNN). The functioning and the theory behind them are explained in section 5.2. The most active actors of this research domain are *Google*, *Microsoft* and *Facebook*. In fact, they define the current state of the art since the first presentation of a deep neural network used to analyze images.

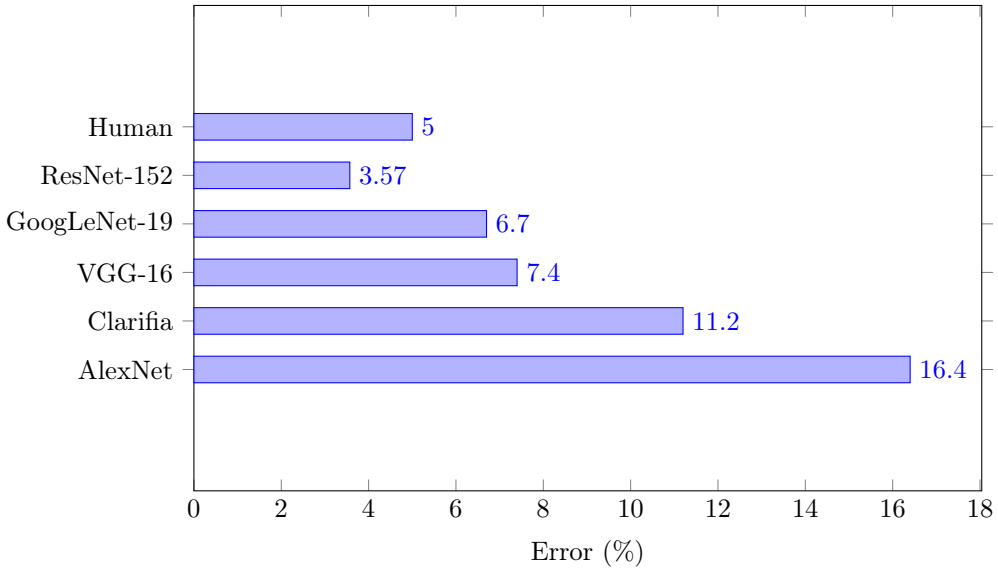


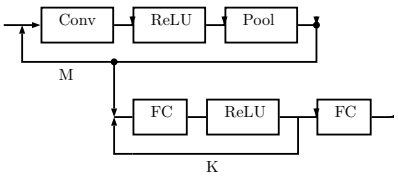
Figure 5.1: The error distribution in the years: AlexNet (2012), Clarifia (2013), VGG-16 (2014), GoogLeNet-19 (2014), ResNet-152 (2015)

Starting with *LeNet-5* by LeCun et al. (1989) [50], CNNs have typically had a standard structure with stacked convolutional layers, followed by one or more fully-connected layers. With GoogLeNet by Szegedy et al. (2015) [77], a new concept of inception is introduced allowing to apply three different filters on the image (1x1,3x3,5x5). Note that the introduction of 1x1 could seem counterintuitive, but it reduces the number of input channels and allows a faster training. Then with the Microsoft network, ResNet by He et al. (2016) [36], the networks started going wilder instead of going deeper. The filers are placed in parallel, reducing the computation time. Finally in *Inception-v4* by Szegedy et al. (2017) [79], Google provided a way to simplify and increase the performances of the predecessors. The accuracy resultant from this new network is comparable with the ResNet, that achieved 3.7% in the Top-5 error, with tremendously lower computation time. The resource management is a critical task and due to the hardware we dispose of, explained in chapter 4 we decided to use the *Inception-ResNet* architecture that it is similar to *Inception-v4* in term of performance and computation time. Moreover, it introduces even more spatial reduction, that is really helpful in our case.

5.2 CNN

The first modern CNN was developed by LeCun et al. (1995) [51]. He used a CNN to classify handwritten digits. Later, machine learning, and deep learning, in particular, have taken big leaps forward regarding accuracy and efficiency, causing a gain in popularity. A CNN is composed of different parts that together form the network.

Each part will be discussed in detail, but the basic usage of CNNs is on the following form:



where

$Conv$: stands for Convolution Layer
$ReLU$: stands for Rectified Linear Unit
$Pool$: stands for Pooling Layer
FC : stands for Fully-Connected Layer
M, K : are the numbers of times each operation is performed

A convolutional layer is usually composed of several feature maps (with different weight vectors), so that multiple features can be extracted at each location. Once a feature has been detected its exact location becomes less important, as long as its approximate position relative to other features is preserved. Therefore, each convolutional layer is followed by an additional layer which performs a local averaging and a subsampling, the pooling layer, reducing the resolution of the feature map. The activation layer (ReLU) after the convolutional layer is used to control the effect of the squashing non-linearity. This triad can be repeated in sequence to increase the number of feature maps and to decrease the spatial resolution. The fully connected layer is used then to find the correlation between the label and the features contained in the feature maps. LeCun et al. (1995) [51]

5.2.1 Convolutional Layer

They utilize the fact that an image is a high-dimensional input, consisting of small features that together form the image. To extract the features, it executes different steps. A single step of convolution multiplies and sums the pixel values of an image with the values of a filter. This filter can be of shape $N * N$. Next, the filter is shifted to a different position, and the convolutional step is repeated until all pixels are processed at least once. Figure 5.2.The resulting matrix eventually detects edges or transitions between dark and light colors and eventually event more complex forms. The more filters are applied, the more details the CNN is capable of recognizing. In really disperse images it is possible to skip some pixels, this process is called *Stride convolution* and consists in moving the filter of the strides chosen. The features extraction process could be obtained using a basic feed-forward network on the raw image. Although, a very high number of neurons would be necessary, due to the huge input sizes associated with the images, where each pixel is a relevant variable. For

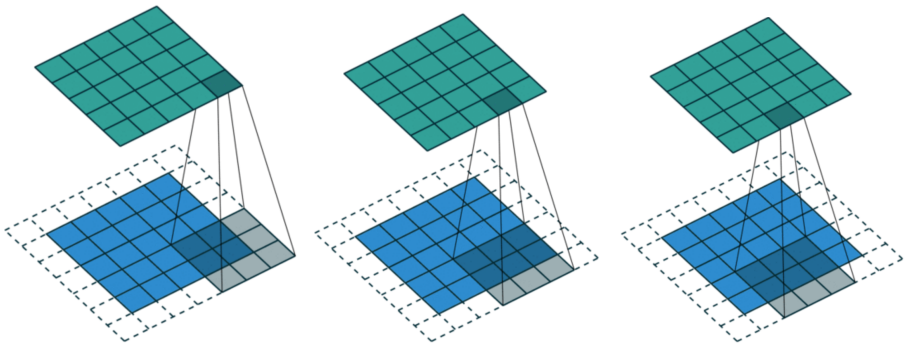


Figure 5.2: Three steps of the convolutional process

instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters. Moreover, the inclusion of the rectified linear unit (ReLU) aims to apply an *elementwise* activation function such as sigmoid to the output of the activation produced by the previous layer. This non-linear function is necessary for the network to be able to represent non-linear relationships between neurons.

5.2.2 Pooling Layer

It is possible to insert different types of pooling based on the goal to reach and the problem analyzed. It takes as parameter the dimension of the output mask. In figure 5.3 a filter = 2 is used. The most famous types are *max pooling* and *average pooling*, but max pooling is generally preferred. Max pooling selects the maximum value of all selected squares to make feature detection more robust. Average pooling uses the average of all values. Neither of this two pooling methods requires parameters, so backpropagation also does not need to learn anything. The max pooling uses formula 5.1 and the average pooling instead uses formula 5.2.

$$f(\vec{x}) = \max(\vec{x}) \tag{5.1}$$

$$f(\vec{x}) = \text{avg}(\vec{x}) \tag{5.2}$$

The main reason to choose the max pooling is that it performs really well in most of the systems, but there is no mathematical explanation for this choice.

5.2.3 Fully-Connected Layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. It is also suggested by

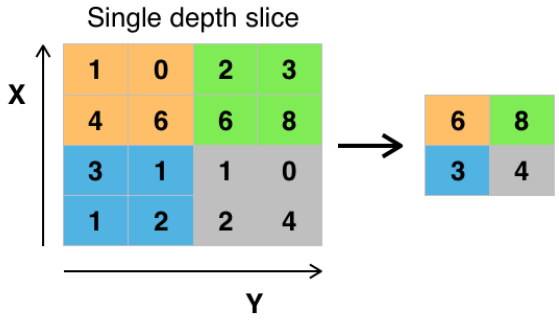


Figure 5.3: The max pooling operation on a simple example

O'Shea (2015) [66], that ReLU may be used between these layers, to improve performance. It is in principle the same as the traditional multi-layer perceptron neural network (MLP).

A distinguishing feature of CNNs is that many neurons share the same filter, this reduces memory footprint because a single bias and a single vector of weights are used across all receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights. This combined with the deep architecture by Szegedy et al. (2017) [79] allows our system to be computationally feasible and to have a regular training time.

5.3 Toxception

Inspired by Szegedy et al. (2017) [79] and Goh et al. (2017) [31] we decided to call this part of the work *Toxception*. It is based on the inception-ResNet v2, also known as *GoogLeNet*. In addition, it uses the work done by Goh et al. (2017) [31] about the optimization realized on a similar system that analyzes Tox21 database and HIV virus. The main difference between this and our network is the absence of apriori chemical knowledge used by Goh network. In order to understand the network used,

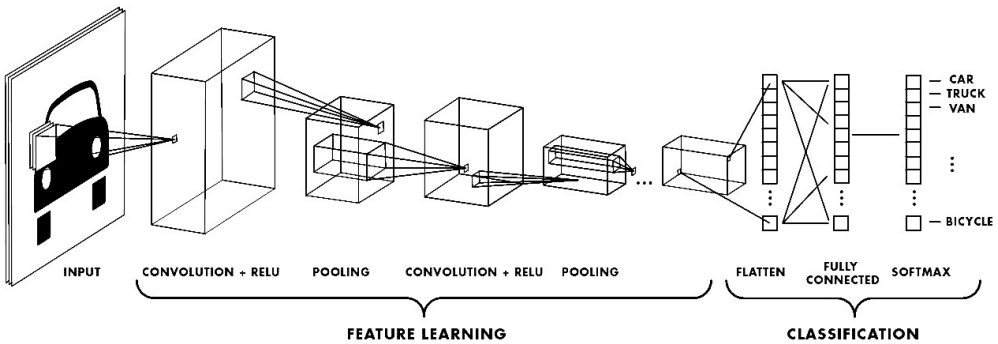


Figure 5.4: A sample schema of a convolutional neural network.

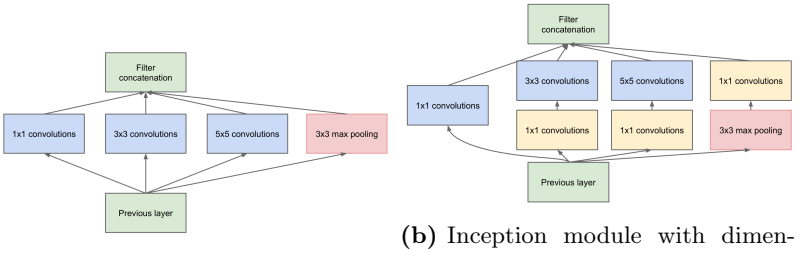
we must follow the theory since the first *Inception* (Szegedy et al. (2015) [77]) network. Before starting this journal, however, we explain how we generate the images from the SMILES strings, in section 5.3.1.

5.3.1 RDKit's image generation

Two modules compose the function realized by RDKit to convert a SMILES to a two-dimensional drawing: a SMILES parser to convert the SMILES back to its parent spanning tree, and a SMILES drawer to convert this spanning tree to a two-dimensional structure drawing.
The parser module generates a parse tree from the input SMILES, in which each atom is encoded by a node object in a linked tree data structure. The topology of the parse tree is identical to the spanning tree used to generate the SMILES string. In practice, the parser uses a simple CFG grammar to avoid ambiguities. In addition to generating the parse tree, the parser can identify the location of an erroneous symbol.
The SMILES drawer module converts the parse tree obtained from the SMILES to a 2D-structure drawing. The module positions acyclic atoms, atoms in fused rings and atoms in Spiros based on Euclidean and molecular geometry according to the VSEPR model. The placement of bridged ring-systems with $n - rings \geq 2$ is treated as a two-dimensional graph embedding problem, solved based on graph theoretic distances as described by Kamada and Kawai. The algorithm sets up a virtual dynamic system, where weighted topological distances between all vertices are modeled as springs. Whereas other spring embedders such as the *Eades* and *FruchtermanReingold* algorithms, which have been adapted to depict molecular structures, The spring introduce repulsive electrical forces between no connected vertices to keep them apart. The drawer, at the end of the process, saves the image to a PNG or SVG file.

5.3.2 Inception

Inspired by a neuroscience model of the primate visual cortex by Serre et al. (2007) [74] the *Inception* network used a series of filters of different sizes to handle multiple scales. Moreover, this network includes deeply the concept of *Network-in-Network* by Lin et al. (2013) [52] in order to increase the representational power of neural networks. In this model, additional $1x1$ convolutional layers are added with a dual purpose: most critically, they are used mainly as dimension reduction modules to remove computational bottlenecks, that would otherwise limit the size of our networks. This allows increasing both, the depth and the width, of our networks without a significant performance penalty. The main drawbacks of the increase of width and depth are the increase of the computational resources and the increase of the number of parameters, that make the network more prone to overfitting.
A fundamental way of solving both of these issues would be to introduce sparsity and replace the fully connected layers by the sparse ones, even inside the convolutions. Besides mimicking biological systems, this would also have the advantage of firmer



(a) Inception module, naive version

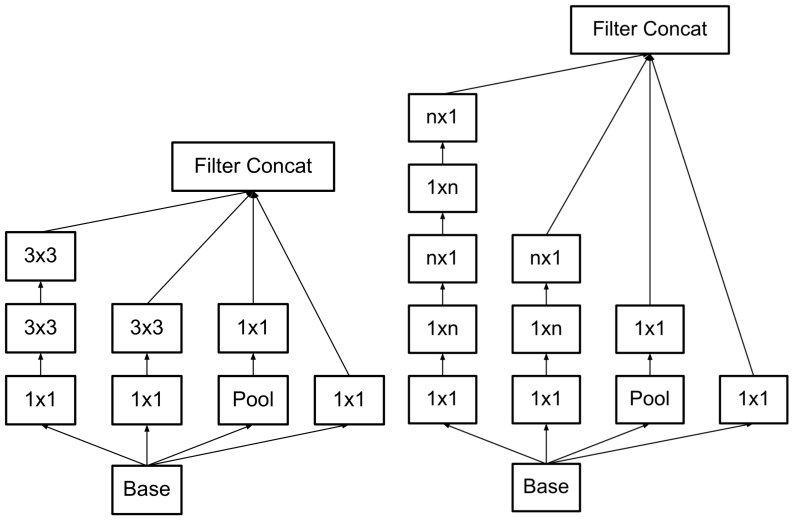
(b) Inception module with dimensionality reduction

Figure 5.5: Inception module

theoretical underpinnings.
The main idea of the Inception architecture is to consider an optimal local sparse structure of a convolutional vision network and to convert it by readily available dense components. This is done by building a subset group composed of convolutional blocks. These clusters form the units of the next layer and are connected to the units in the previous layer. We assume that each unit from an earlier layer corresponds to some region of the input image and these units are grouped into filter banks. These "Inception modules", in figure 5.5, are stacked on top of each other, their output correlation statistics are bound to vary: as features of higher abstraction are captured by higher layers, their spatial concentration decreases. Moreover, in order to judiciously reduce dimension wherever the computational requirements would increase too much otherwise, the *Inception module with dimensionality reduction*, in figure 5.5b, is created. It includes $1x1$ convolutions that are used to compute reductions before the expensive $3x3$ and $5x5$ convolutions. Besides being used as reductions, they also include the use of rectified linear activation making them dual-purpose. It is based on the concept of embedding in order to represent the information in a dense and compress form. The first *Inception* network was a combination of these blocks with an occasional max-pooling layer with stride two, leading to 22 deep layers.

In order to improve *Inception-v1*, Szegedy et al. (2016) [78] listed out some main principle that should be used in the architecture design phase:

- Avoid representational bottlenecks, especially early in the network. That is because, theoretically, information content cannot be assessed merely by the dimensionality of the representation as it discards important factors like correlation structure.
- Higher dimensional representations are easier to process locally within a network. Increasing the activations per tile in a convolutional network allows for more disentangled features.
- Spatial aggregation can be done over lower dimensional embeddings without much or any loss in representational power.



(a) Inception modules where each $5x5$ convolution is replaced by two $3x3$ convolution.

(b) Inception modules after the factorization of the nxn convolutions.

Figure 5.6: Inception-v3 modules.

- Balance the width and depth of the network. Optimal performance of the network can be reached by balancing the number of filters per stage and the depth of the network.

The main improvement added by *Inception-v3* is then the factorization of convolution with larger filter size. In fact, with suitable factorization, we can end up with more disentangled parameters and therefore with faster training. For example, a $5x5$ convolution with n filters over a grid with three filters is $25/9 = 2.78$ times more computationally expensive than a $3x3$ convolution with the same number of filters. The innovation introduced by Szegedy et al. (2016) [78] is the possibility to decompose into a multi-layer network with fewer parameters maintaining the same input size and output depth. To do so it replaces the $5x5$ convolution with two layers of $3x3$ convolution (compare figure 5.5b with 5.6a). To push even further the factorization process, Szegedy et al. (2016) [78] proposed to use asymmetric convolutions, e.g., $nx1$. For example, using a $3x1$ convolution followed by a $1x3$ convolution is equivalent to sliding a two-layer network with the same receptive field as in a $3x3$ convolution. The factorization proposed in figure 5.6b has proven to be really efficient in the late layer of the network, so two new block types have been created in order to recreate the *Inception* architecture. The final *Inception-v3* is 42 layer in depth and it increases the number of parameters.

5.3.3 Residual Network

He et al. (2016) [36] introduced in 2015 the idea of *Residual Connection* based on the

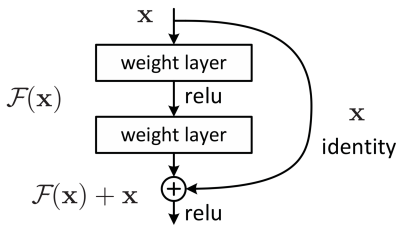


Figure 5.7: Residual learning: a building block

concept of *Residual Learning*. Let us consider $H(x)$ as an underlying mapping to be fit by a few stacked layers. With x denoting the inputs to the first of these layers. If one hypothesizes that multiple nonlinear layers can asymptotically approximate complicated functions, then it is equivalent to hypothesize that they can asymptotically approximate the residual functions, *i.e.*, $H(x) - x$. So rather than expect stacked layers to approximate H(x), we explicitly let these layers approximate a residual function $F(x) := H(x) - x$. The original function thus becomes $F(x) + x$. In this reformulation, if identity mappings are optimal, the solvers may simply drive the weights of the multiple nonlinear layers toward zero to approach identity mappings. As reported by He et al. (2016) [36], the learned residual functions, in general, have a small response. In the solution proposed for ResNet, each building block is considered with residual learning and it is expressed as

$$y = F(x, W_i) + x \tag{5.3}$$

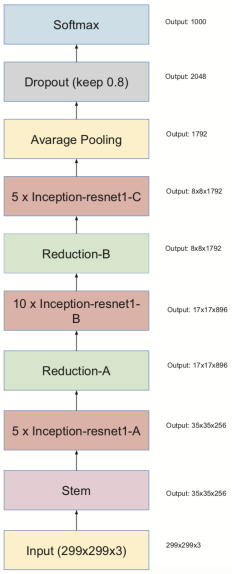
where

x	:	is the input of the vector of the layers considered
y	:	is the output of the vector of the layers considered
$F(x, W_i)$	:	represents the residual mapping to be learned. For example in figure 5.7 that has two layers it is $F = W_2\sigma(W_1x)$. Where σ represents the ReLU.

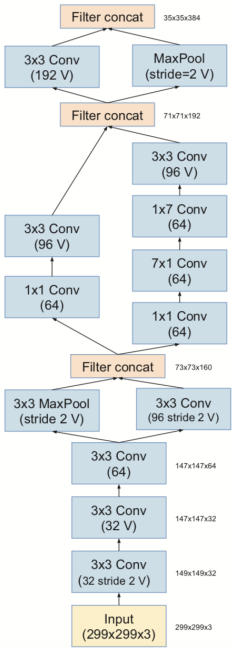
As it is imaginable the sequence of layers considered must be longer than one in order to take advantages of the absence of extra weights to calculate.
The network realized by He et al. (2016) [36] is similar to *VGG-19* but it contains residual connection as in figure 5.7, it has lower complexity and has 152 layer depth.

5.3.4 Inception-ResNet-v2 altered, our model

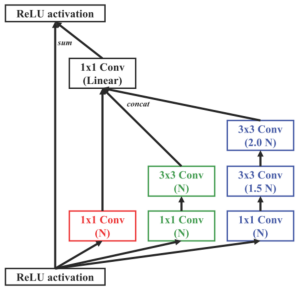
Szegedy et al. (2017) [79] combines the work of He et al. (2016) [36] and Szegedy et al. (2016) [78] by creating the 152 layer network that is represented in figure 5.8. The main differences between this and the older ones are the training time, that is highly reduced by the introduction of the *Residual connection*. Also, the *Stem* block is introduced to replace all the linear layers present in the precedent networks.



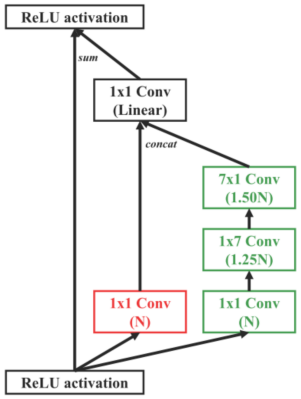
(a) Inception-ResNet-v2



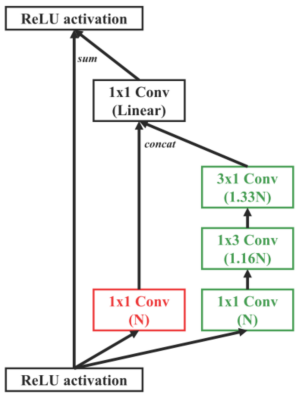
(b) The stem block.



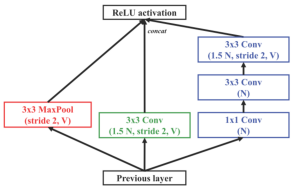
(c) Inception-ResNet-A.



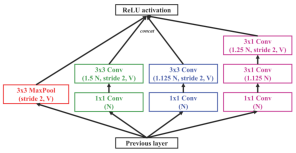
(d) Inception-ResNet-B.



(e) Inception-ResNet-C.



(f) Inception-ResNet-Reduction-A.



(g) Inception-ResNet-Reduction-B.

Figure 5.8: In figure 5.8a is the overall schema for the Inception- Resnet-v1 and Inception-Resnet-v2 network. While the schemas are the same for both networks, the composition of the stem and interior modules differ. In 5.8b is detailed the composition of the stem. The other figures represent the different block of the network. Layers are denoted in colored boxes and are assumed to have a ReLU activation layer after the specified convolution layer, with a stride of 1, and 'same' padding unless otherwise noted. Each block has N convolutional filters for each layer, and the variations are indicated as multiples of N.

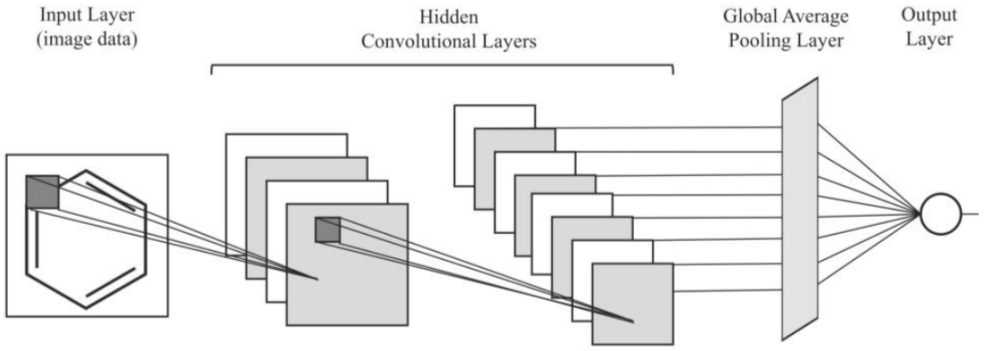


Figure 5.9: Simple scheme of Toxception

To adapt the network to our problem we have changed the size of the input from $299x299x3$ to $80x80x3$ changing also the different number of nodes for each type. Differently from Goth [31], we did not use data augmentation, and we passed the images to the network exactly as they exit from the generation function. That is because the chemistry knowledge passed to the network and image processing done by [31] has turned out not to improve the performance of the model. The resultant network has 650881 parameters. A simple final network scheme is in figure 5.9 and figure 5.8.
Differently from [79] we used a personalized *learning rate scheduler* and *optimizer scheduler* as suggested by [31]. In particular, we trained the network with two algorithms. First, we tried *Adam Optimizer*, then we used *RMSProp (Root Mean Square Propagation)* optimizer for half of the training epochs and we switched to *SDG (Stochastic gradient descent)*. All are described in section 7.1.2. The parameters used are reported in table 5.1

Optimizer	Parameter	Value
RMSProp	Initial lr	$1e^{-3}$
	Rho	0.9
	Epsilon	$1e^{-8}$
	Decay	0
SDG	Initial lr	$1e^{-3}$
	Momentum	0.9
	Gamma	$1e^{-8}$

Table 5.1: Parameters used of the optimization algorithms

5.4 Optimization process

In this section, we want to give an overview of the different combinations of parameters tested for Toxception, in order to have a better comprehension of our results reported in section 5.5. Also, we want to explain and analyze the process followed to evaluate and optimize the network. In table 5.2 we regrouped all the values by the different cited papers in order to have a global vision of the state-of-the-art performances. As it can be deduced from figure 5.8, Toxception is a big network and this implies a high number of Hyper Parameters to manage and optimize. Testing all the different possible combinations is impossible. We need to construct a tree with the most important parameters that could affect the efficiency and the computation resources used by the network. To realize this task we used *Talos*, explained in paragraph 4.4.3, which allows the user to decide the main important parameters to change and run all the possible permutations of them, testing the accuracy, the loss function, the specificity, etc. In this section, we report the design choices we made in order to optimize the network.
We decided to leave unchanged the convolution layers proper to each block in order to not modify the nature of the network itself. We kept as a variable only the number of neurons present for the first layer creating a causal sequence between the different blocks in order to maintain input dimensions coherent. We tested different *Dropout* values (0.4, 0.2, 0.1, 0) to understand which value prevents the network from overfitting without affecting the performances.
In order to reduce the complexity of the network, and seen the low complexity of the images. We decided to reduce the *Stem block* and we tested the different solutions to analyze if the block was also needed in simple 2D structure images. Finally, we tested different epochs and early stopping technique in order to evaluate the tendency of the network to overfitting and to guarantee the best model possible. The whole data concerning the result are regrouped and explained in section 5.5

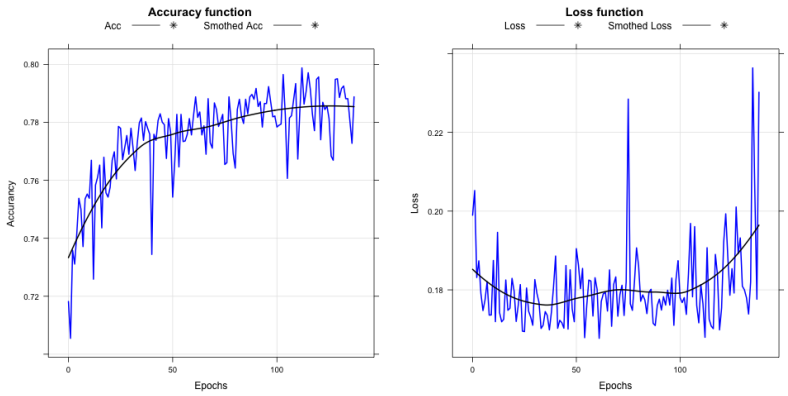
Model	Top-1 err.	Top-5 err.
VGG-16	25.2%	7.32%
GoogLeNet	-	6.66%
BN-inception	21.99%	4.82%
ResNet	19.38%	3.57%
Inception-v4	17.7%	3.8%
Inception-ResNet-v2	17.8%	3.7%

Table 5.2: Result from the ILSVRC challenge from the different network proposition.

5.5 Results

We used the 20% of the data collected as the validation set. In addition, we used different metrics in order to optimize and to evaluate the network: the loss on the validation set, the accuracy on the validation set. Even if this last one does not normally

prove the consistency of the model, in image analysis classification it is considered a good metrics to evaluate the model. Moreover, we use sensitivity and the specificity on the validation set. From these two measures, it is possible to calculate the ROC curve and to extract then the AUC, area under the ROC curve. In order to discover the potentiality of our model, we tested both MSE and cross-entropy loss function (called xentropy in this section).
Also, we analyzed the probability distribution given by the network without rounding the final values of the vector. It is a best practice to analyze this values in order to avoid the phenomenon of the false result, which happens when the classification results are correct, but the outcome of the network is actually too weak. Moreover, this study allows the extraction of the probability for a particular compound to belong to a specific endpoint. Therefore, especially in SMILES-Net, it is really useful to understand the model itself.
Using talos, we ran multiple times the network applying the cross-validation method,i.e., changing the hyperparameters combination. The framework returns a table containing all the different networks tested. In table 5.4 and 5.3 we reported the best results for each important combination found. In particular, the tables contain some abbreviation that we are explained here in order to have a better understanding. Tox_Basic indicates the Toxception network without *Stem* block. The notation Tox instead includes also the usage of the *Stem* block. The subscript next to each name, $_{A\ B}$ indicates if the network has been trained using, *RMSProp* optimizer for 100 epochs and then re-trained with the *SDG*, for $_A$. For $_B$, instead, we used the *Adam* optimizer for the number of epochs indicated in the tables. The two optimizers just cited are indicated with their abbreviation: *RMS* and *Adam*. Between the parameters listed, *Neurons* indicates the number of neurons present in the first Inception block, the values of the others are not cited as they are strictly related to that one. Table 5.3 reports all the network trained with *Mean square error (MSE)*. 5.3 reports all the test done with the *cross-entropy* function.
The numbers chosen are taken from the literature, in particular from Goh et al. (2017) [31]. We start all the training with a *learning rate* of $1e^{-}3$ and we apply an *early stopping* technique.



(a) accuracy **(b)** loss

Figure 5.10: Detailed result using MSE without the stem block and the *Adam* optimizer. In particular we can observe how the accuracy and the loss function follow the normal training pattern with the increase of the epochs. All the metrics reported here are calculated on the validation set, some initial values are cut out in order to have a better vision of the values and the trend.

Architecture	Parameters		Metrics					
			Acc	Loss	Val Acc	Val Loss	Spec	Sens
Tox_basic_{A1}	Epochs	*200*	0.998	0.01	0.812	0.161	0.588	0.84
	Optimizer	*Adam*						
	Neurons	16						
Tox_basic_{B1}	Epochs	*200*	0.997	0.01	0.801	0.152	0.623	0.882
	Optimizer	*RMS*						
	Neurons	16						
Tox_{A1}	Epochs	*200*	0.854	0.116	0.787	0.165	0.675	0.771
	Optimizer	*Adam*						
	Neurons	32						
	Epochs	*200*	0.965	0.367	0.789	0.18	0.632	0.891
	Optimizer	*Adam*						
	Neurons	16						
Tox_{B1}	Epochs	*300*	0.823	0.136	0.767	0.175	0.382	0.931
	Optimizer	*RMS*						
	Neurons	32						
	Epochs	*200*	0.954	0.041	0.790	0.172	0.65	0.874
	Optimizer	*RMS*						
	Neurons	16						

Table 5.3: Toxception Optimization results using *Mean Square Error (MSE)* as loss function

Architecture	Parameters		Metrics					
			Acc	Loss	Val Acc	Val Loss	Spec	Sens
Tox_basic_A	Epochs	*200*	0.988	0.021	0.801	0.171	0.624	0.875
	Optimizer	*Adam*						
	Neurons	16						
Tox_basic_B	Epochs	*200*	0.842	0.133	0.776	0.169	0.415	0.791
	Optimizer	*RMS*						
	Neurons	16						
Tox_A	Epochs	*200*	0.962	0.091	0.779	0.38	-	-
	Optimizer	*Adam*						
	Neurons	16						
Tox_B	Epochs	*200*	0.831	0.141	0.759	0.178	0.392	0.922
	Optimizer	*RMS*						
	Neurons	16						

Table 5.4: Toxception Optimization results using *xentropy* as loss function

Looking at the tables, there are different aspects to point out. First of all, it is clear the effect of the *Stem* on the network. In fact, in all the simulations, the usage of the *Stem* block limits the learning. It can be explained by the simplicity of the data passed into Toxception. As already said, the images are composed by a lot of white space; this implies that the division of the input into small features can considerably increase the resources needed from the network to converge. Observing Tox_{B1} and Tox_{B2} it is visible as the increase of the training epochs allows the network to converge and to minimize the loss function. However, the results achieved by these networks concerning the accuracy are still not so high compared with the one obtained in the basic networks.

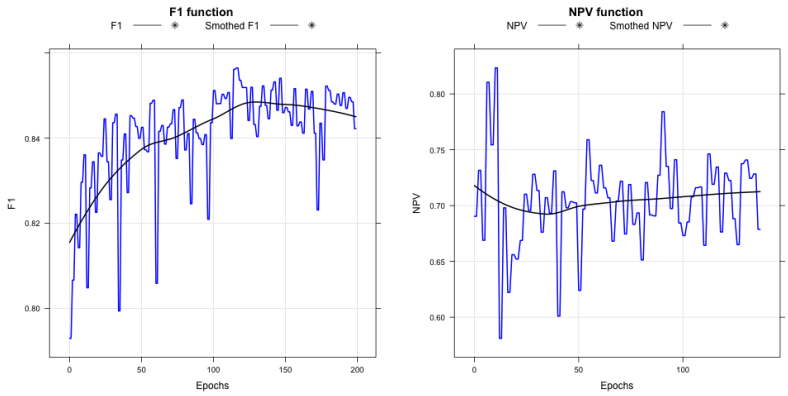
The second interesting point that appears from these analyses is the values of *Sensitivity* and *Specificity*. A low value of the sensitivity implies a high number of false negatives (because we use as thesis: the compound is toxic). The main goal of this thesis, and of all the prediction models that are developed, is to have a really low number of false negatives. That is because false negative indicates some compound that should be marked as toxic, but it is instead predicted as non-toxic. The implications of a high number of false negatives are obvious. As we can see in the tables, this number in the Toxception is small as the sensitivity is more than 85%. However, the values of specificity, in all the combinations, are not really high. It is due to precautionary choices we have made during the data collection and the data pre-processing. As reported in section 4.3, while collecting the data we decided to include all the AMES test results in the literature but, in order to create a safe model, we considered as *mutagenic* all the compounds that appeared more than once with opposite results. As we explained in section 2.4.2, this contradiction can be caused by different variables, as for example the test guideline, but in order to not have a high number of false negatives, we marked them as toxic. That is why, even if with other models we

increase this values, we will always have an upper bound for this metric.
The third aspect, is the comparison between *RMSProp* and *Adam* Optimizer. As we can see in both tables, 5.4 and 5.3, the *Adam* optimizer seems to perform better than the *RMS* in the whole set of metrics. That could be due to the particular learning rate schedule that we implemented in the usage of RMS. We split the training epochs into two and we trained the network for half using the RMS and for another half using the SDG. We wanted to test the proposition made by Goh et al. (2017) [31] in order to understand the domain applicability of the methods proposed in the article. The results of this research end-up in the discovery that the method proposed by Goh et al. (2017) [31] performs actually well applied to simple compound given the assumption that the network has received some basic knowledge of chemistry. The alteration proposed by this work to *Chemception* allows the model to be more adaptive and at the same time to keep high performances. The proposed method also can adopt a different loss function. *Chemception* instead gives low results if trained with *Cross-entropy*.
Another parameter that needs to be considered, not included in the table, is the computational time needed to execute the training and the prediction. Based on the hardware explained in chapter 3, *Toxception* takes around 130 milliseconds per step, each epoch is composed of 600 steps. This makes the time spent by the networks for the training process around four hours. The prediction instead is less time consuming as we do not need to calculate the weight of each neuron. To evaluate the whole dataset the net take around five minutes, this means around twelve milliseconds per compound. The time is an important variable in this work as the optimization process run around 600 combinations and must be really efficient in order not to take too much time.

In order to better analyze the results just explained we decided to run some other training for the most interesting networks in the tables including some additional metrics. In particular, we reported below the analysis executed on three main configurations:

- Tox_basic with Adam optimizer and MSE loss function, in figure 5.11 and 5.12
- Tox_basic with RMS optimizer and MSE loss function, in figure 5.14.
- Tox with RMS optimizer and MSE loss function, in figure 5.13.

The black line reported in all the graphs is the smoothed function obtained from the blue curve. There are a few remarkable facts to comment. The first is that in all the configuration proposed the metrics tend to converge pretty fast. This phenomenon is particularly clear in figure 5.14.
Another point can be seen in figure 5.14. Observing all the metrics, we can see how the model responds to the introduction of a new learning rate schedule. A small hump represents this in the middle of the training. However the NPV function seems less affected by this phenomenon. It is due to the precautionary choices taken in the data

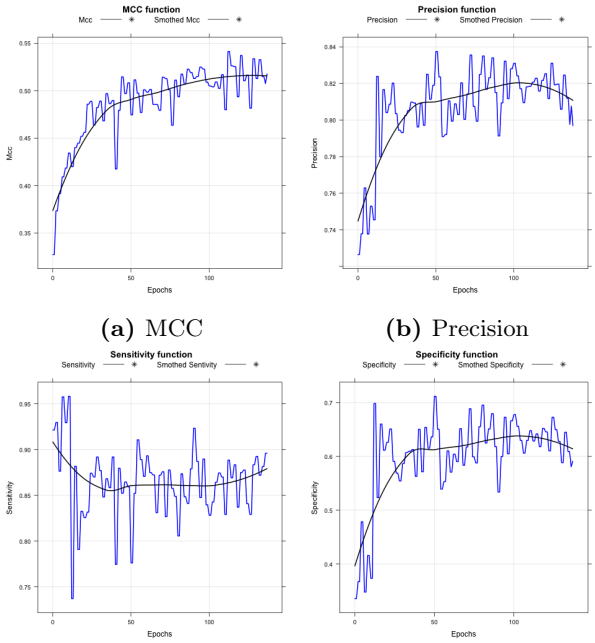


(a) F1 **(b)** NPV

Figure 5.11: Detailed result using MSE without the stem block and the *Adam* optimizer. In particular it is interesting to observe the NPV metric, which is stable while the epochs go up without really dropping its value. The F1 function, used to evaluate classifications gives good result without any noticeable trend. All the metrics reported here are calculated on the validation set, some initial values are cut out in order to have a better vision of the values and the trend.

collection phase, explained before.
Another interesting information we can get from these graphs is that precision, sensitivity and specificity get really fast to their asymptotic value and form a plateau after a few epochs. It seems to be a learning problem, but as we can see from the loss function in all the combination tested this is not the case. Moreover, the stability and the values of the loss function, in figure 5.14 and 5.13, suggest also that the proposition of the learning scheduler made by Goh et al. (2017) [31] is valid and if it is applied to the right model could give better results.
As we wanted to compare the best models we reported in figure 5.13 the results obtained using the *MSE* loss function, the *RMS* optimizer and the stem module just after the input. In particular, seen the results of the RMS we wanted to compare these results with the one obtained from the basic model of *Toxception*. From figure 5.14, we can see that our initial hypothesis, about the complexity of the architecture, is validated by these comparisons.

Analyzing singularly each graph we can also point out some aspects. First of all, in figure 5.10b, it is visible when the network starts overfitting. The last ten epochs are really noisy compared to the others and at the end there are a few high values. In this graph it is possible to see the early stop mechanism in action. We configured the network in order to stop training if, for ten consecutive epochs, the value of the loss function increased with respect to the previous epoch. This trend is also visible in figure 5.12b, 5.12d and 5.11a. The last important point to consider in the Tox_basic_A is the schedule of the learning rate of the Adam optimizer. We do not have a graph for it but in figure 5.11a and 5.11b it is possible to deduct from the smoothed black



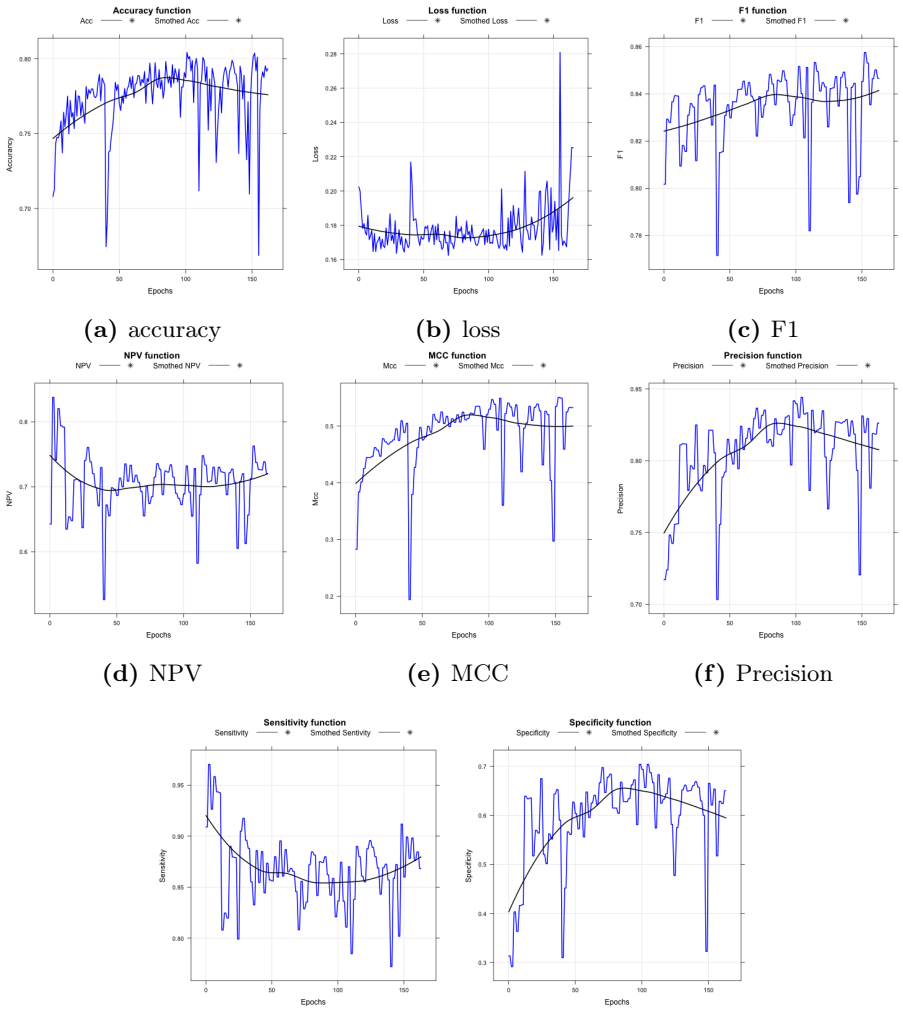
(a) MCC **(b)** Precision

(c) Sensitivity **(d)** Specificity

Figure 5.12: Detailed result using MSE without the stem block and the *Adam* optimizer. We can see how the sensitivity decreases in favor of the specificity. These two metrics are well summarized by the *MCC* where we can see how the MCC function gradually grows with a big step from the begging to the end during the learning process. The precision instead stays stable after a few dozens of epochs. All the metrics reported here are calculated on the validation set, some initial values are cut out in order to have a better vision of the values and the trend.

line when the optimizer changed the learning rate by observing a few small humps in epoch 50^{th},100^{th} and 120^{th}.

For what concerns the Tox_B we can also say something more. As before we can see the effect of the early stop technique in figure 5.13a, 5.13b and 5.13f. In addition, in this graphs, it is clear that the noise is higher than the other network. We attribute this to the presence of the stem block, which introduces more sparse features to the convolutions layers. This effect also proves our assumption proposed in section 5.4. Finally, considering Tox_basic_B, we remark more smooth graphs, especially for figure 5.14a, 5.14f and 5.14h. This is due to two main factors: first of all we did not cut the initial values because we wanted to have a more high level view of the training process. In figure 5.14e and 5.14d this phenomena is pretty clear. In addition the RMSProp optimizer has a smoother management of the learning rate, as we used two different algorithm combined together.
Our final proposal is in chapter 8, combined with the decision taken for the other sub-model.

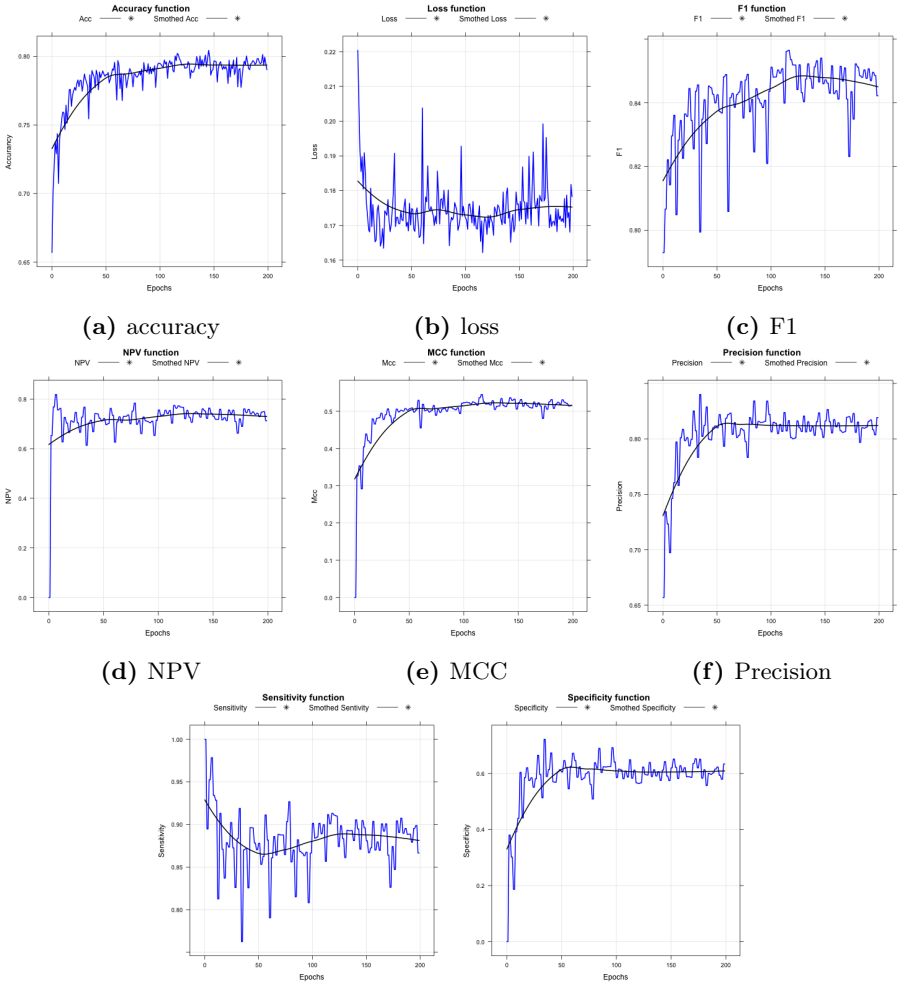


(a) accuracy **(b)** loss **(c)** F1

(d) NPV **(e)** MCC **(f)** Precision

(g) Sensitivity **(h)** Specificity

Figure 5.13: Detailed result using *MSE* with the stem block and the *RMS* optimizer. All the metrics reported here are calculated on the validation set, some initial value are cut out in order to have a better vision of the values and the trend.



(a) accuracy **(b)** loss **(c)** F1

(d) NPV **(e)** MCC **(f)** Precision

(g) Sensitivity **(h)** Specificity

Figure 5.14: Detailed result for regression without the stem block With the RMS optimizer. All the metrics reported here are calculated on the validation set, some initial values are cut out in order to have a better vision of the values and the trend.

Chapter 6

SMILES-Net

The second part of this work concerns the textual representation of a chemical. We decided to explore a completely different domain compared to chapter 5 in order to cover the widest possibilities concerning the methods that could work well in computer toxicology. The standard text classification has always been important in data mining. With the growth of deep learning, neural networks have also been adopted on text. Nowadays the main applications are conversational agent, machine translation, and document classification. In this chapter, we introduce the state of the art techniques of text classification using neural networks. Concerning computational toxicology, however, this particular mechanism has never been used to classify toxic chemicals. We will analyze the main reasons in the result section, as our model is built using this technique highlighting some drawbacks that could affect future choices in other works. In particular in section 6.1 we analyze the main open problems of textual treatment. In 6.2 we also analyze the main techniques used to represent text in such a way that it is understandable to the computer. In section 6.3 we discuss the module that follows the feature extraction and document translation. Finally in section 6.7 we discuss our results.

6.1 Open problems

As already said there are different applications of text analysis. The market is really interested in three domains:

- Document classification,
- Conversational agents,
- Machine translation.

6.1.1 Document Classification

Usually based on natural language, document classification has become popular in the last years due to its indexing and space saving capabilities. In fact, using classification, we can store and manage a large amount of data, consuming a moderate amount of resources. It is really useful in domains such as medicine, searching engine, database, etc. It consists in assigning a document or a text to one or more classes. The documents to be classified may be texts, images, music, etc. Each kind of document possesses its particular classification problems. In this chapter, we only treat text classification.
The traditional machine learning techniques used in text classification are almost the same explained in section 3.2. The new cutting-edge techniques use deep learning to find better patterns among names, entities and other words inside the document. The main idea is to consider every part of the object to classify as a token. A document is then a set of tokens combined with some unknown rules. Hrala and Král (2013) [39] has written an accurate article about the different methods used; a few of them are explained in this chapter.

6.1.2 Conversational agent

The goal to speak with a computer has always been in the researchers' agenda, and developing a computer able to sustain a conversation is an active research field. The idea to interact with a computer using natural language has always been really attractive for many tech companies, and some recent publications from Google and Apple seem not too far from this goal. The deep usage of neural networks, GPUs and TPUs, has incredibly improved our capacity in understanding natural language and the capability of a machine to understand and use it. Conversational agents are especially used from industries to manage technical issues with customers, to answer simple questions and to have a basic interaction with the users. The most famous systems of conversational agent are *Siri*[13], *Google Assistant*[34] and *Amazon Chatbot*[14].

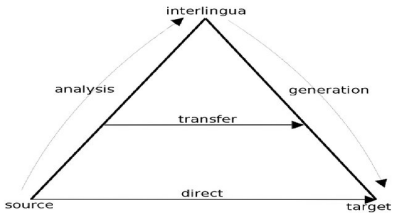


Figure 6.1: The possible solutions for machine translation. Starting from the most basic method in the bottom and going up with the structured methods.

6.1.3 Machine translation

Machine translation was the first NLP problem analyzed, with various solutions proposed in the years. It consists in taking as input a sentence in one language and giving as output the sentence in another language. The output language can be either a human language or a machine-readable language. The first step is to find a statistical correlation between the most frequent words in the input language and the most frequent words in the output language. The most recent method proposed is called *Bilingual* and is based on the translation of the sentence into a temporary language, from which it is possible to translate it to all the known languages. In the paper by Zou et al. (2013) [88] this technique was used to create a word2vec system.

6.2 Feature extraction

Every supervised learning algorithm needs a special representation called *Vector Space Modeling (VSM)*. That is because a neural network works only on numbers. The process of translating a document into numbers is called *Feature extraction*. The methods explained in this section are based on the concept of "*term*". The definition of a term depends on the application. Typically terms are single words, keywords, or phrases. If words are chosen as terms, the dimension of the vector is the number of words in the vocabulary. In our case to represent chemical structures, a term is a single character or at most a couple of them.
Given a raw piece of text T, first a vocabulary v is extracted. The vocabulary is an ordered set containing all the unique words in T; it is usually sorted by word occurrence, and it has a size, v_s. By means of the vocabulary, each word is assigned an index i, a number between 0 and v_s. With this index, we build a vector $v(w)$ of shape $(1, v_s)$. Practically, considering a word w, its vectorial representation has the following form:

$$v(w) = (0, 0, ..., 0, 0, 1, 0, 0, ..., 0, 0), \tag{6.1}$$

Having only one 1 in position i. This representation is called *1-hot-encoding*; it is exactly the same used in chemistry with the name of fingerprint. Even if this method is able to assign a vector to each word in a vocabulary, the extracted features are very sparse, and this is not a first-rate property. For this reason, usually a second step is performed, to a get a handier compact representation. In order to transform the vector into a dense vector, there are different methods. The simplest class of techniques is *BOW (Bag of words)* methods that represent the document as a vector:

$$d_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j}) \tag{6.2}$$

Each dimension corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero. The main problem with these methods is they do not consider word order and for this reason, they can not include any semantic information

in the document vector. To overcome this problem the most used technique is TD-IDF. Other options are possible, first of all using a neural network able to find a correct pattern to translate the text into numbers; this particular technique is called *Doc2Vec* if applied to a document, *Word2Vec* if applied to single words.

6.2.1 TF-IDF

It stands for *Term frequency-Inverse document frequency*. Frequency of occurrence of a token from vocabulary in each document consists in the term frequency. The number of documents, in which a token occurs, determines the Inverse document frequency. It means that if a token frequently occurs in a document that token has high TF, but if that token frequently occurs in the majority of documents then it reduces the IDF. This is done to penalize the *Stop words*, *i.e.*, common words that are present in all the documents as *a, the, etc.* In this way, the words concerning the context of the document get better evaluated by the algorithm. The weights are calculated with:

$$w_{i,j} = TF_{i,j} X \log \frac{N}{IDF_j} \tag{6.3}$$

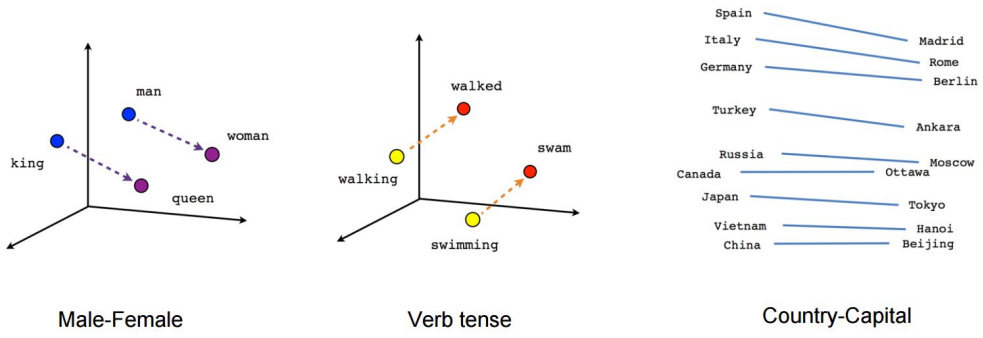
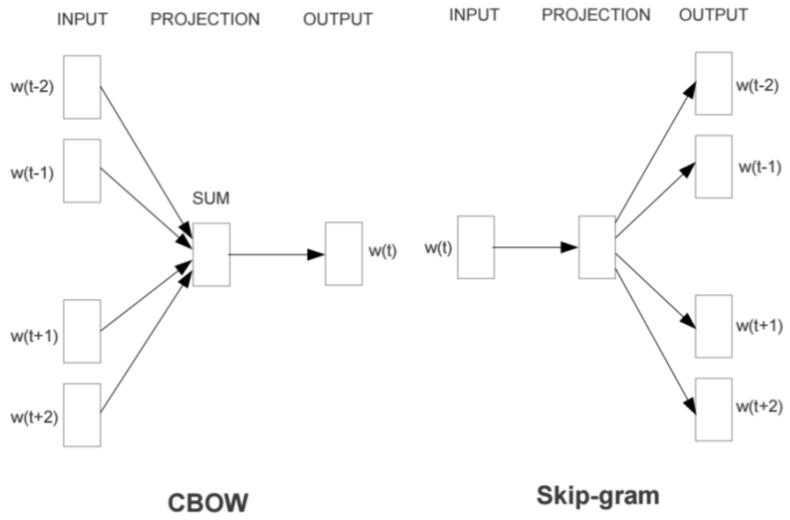


Figure 6.2: Words relationship in CBOW.

6.2.2 Word2Vec

These techniques are based on the work of Mikolov et al. (2013) [56] that proposed two simple methods to construct the word vector: *CBOW* (Continuous Bag-of-word) and *Skip-gram*. These methods use adjacent words to predict the next one. A simple explanation is "*show me your friends, and I will tell who you are*".

Continuous bag of words creates a sliding window around the current word, to predict it from "context", *i.e.*, the surrounding words. Each word is represented as a feature vector. After training, these vectors become the word vectors. As said before, vectors which represent similar words are close by different distance metrics, and additionally encapsulate numeric relations, such as the king-queen=man from figure 6.2.



(a) CBOW. **(b)** Skigram.

Figure 6.3: Word2Vec by Mikolov et al. (2013) [56]

The second algorithm (Skip-gram) is actually the opposite of CBOW: instead of predicting one word each time, it uses one word to predict all surrounding words, called "context". Skip gram is much slower than CBOW but is more accurate with infrequent words. An example of how the algorithm works can be expressed by taking the sentence *I like playing football in the backyard with my cousin.* Considering CBOW mode, a training sample is constructed by picking an index between 0 and $num - words$ in sentence, for example $t = 3$, and by extracting the relative word, that is $w(t = 3) = football$. The training task then consists in generating $w(t = 3)$ from the context words: *like, playing, in, the.* For Skip-gram mode instead, there is a specular situation where the network has to generate the context words *like, playing, in, the* from $w(t = 3) = football$.

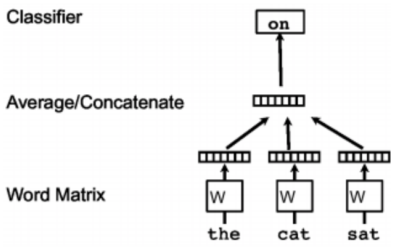
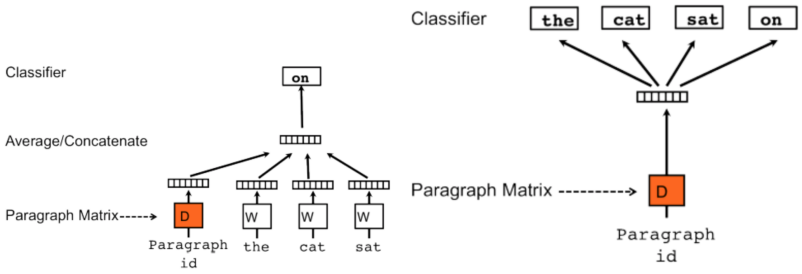


Figure 6.4: CBOW.

6.2.3 Doc2Vec

The goal of doc2vec is to create a numeric representation of a document, regardless of its length. However, unlike words, documents do not come in logical structures such as words, so other methods have to be found. The proposition of Quoc and Mikolov (2014) [68] is to use the same method of word2vec introducing another vector Paragraph ID, transforming the two methods in 6.2.2 into *Distributed Memory version of Paragraph Vector (PV-DM)* similar to CBOW and *Distributed Bag of Words version of Paragraph Vector (PV-DBOW)* similar to Skip-gram. These methods are shown in figure 6.5.

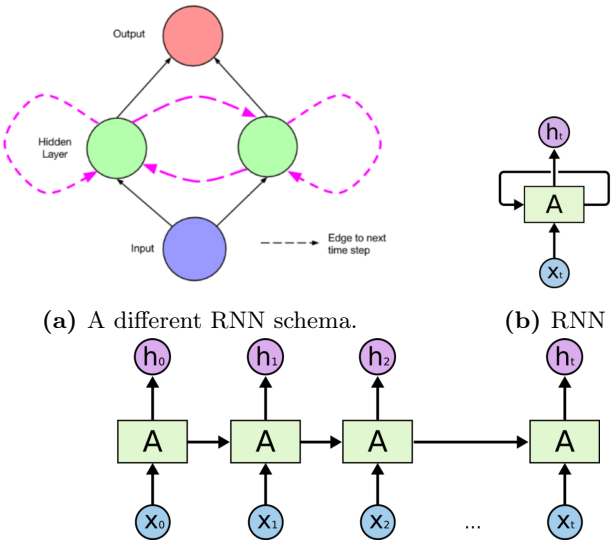


(a) PV-DM. **(b)** PV-DBOW.

Figure 6.5: Doc2Vec by Quoc and Mikolov (2014) [68]

6.3 RNN

The main problem with CNNs, explained in section 5.2, it is that they perform poorly when given a sequence of data, *i.e.*, audio clip which contains a sequence of spoken words in natural language. Feed-forward networks and CNN take a fixed length as input, but, in many cases the input length is variable. It is possible to overcome this issue by padding all the inputs to a fixed size. However this workaround will perform worst than RNN. *Recurrent Neural Network* (RNN), represented in figure 6.6b, are neural networks where connections between nodes form a directed graph along a sequence. Each node at a time step takes an input from the previous node and this can be represented using a feedback loop. It is possible to unfurl the feedback loop to obtain the network in figure 6.6c. This allows to exhibit temporal dynamic behavior or to analyze input in sequence. Recurrent neural networks can have many architectures, the most interesting for our work are : *LSTM* (Hochreiter and informatik (1997) [38]), *Elman networks*, *Jordan networks*, *GRU* (Junyoung et al. (2015) [42]). The equations used to calculate the output are based on the different timesteps and allow to have a concept of time inside the network. It is easier to imagine the network as a structure with two hidden layers connected to each other that receive the same input and affect the same output. The network can be interpreted not as cyclic, but rather as a deep network with one layer per time step and shared weights across time steps. The



(a) A different RNN schema. **(b)** RNN

(c) RNN explosed

Figure 6.6: RNN by rnn (2015) [11]

equations of the output value are:

$$g(x) = \tanh x \tag{6.4}$$

$$a_t = g(W_{hh} * h_{t-1} + W_{xh} * x_t) \tag{6.5}$$

$$a_t = \tanh W_{hh} * h_{t-1} + W_{xh} * xt \tag{6.6}$$

$$h(t) = W_{hy} * a_t \tag{6.7}$$

Where

W_{hx}	:	is the matrix of conventional weights between the input and the hidden layer
W_{hh}	:	is the matrix of recurrent weights between the hidden layer and itself at adjacent time steps
W_{hy}	:	s the matrix of recurrent weights between the hidden layer and the output
a_t	:	represent the output from the previous node
h_t, h_{t-1}	:	are respectfully the output of the hidden layer itself and the "previous" hidden layer.

Backpropagation in recurrent neural networks occurs in the opposite direction of the arrows drawn in figure 6.6c. The exciting part of backpropagation in RNN is that it occurs from right to left. Since the parameters are updated from final time steps to initial time steps, this is termed as backpropagation through time.
In the last few years, there has been incredible success applying RNNs to a variety of

problems: speech recognition, language modeling, translation, image captioning, etc. Some studies, however, have discovered some critical problems in the usage of RNN (Lipton (2015) [53] and Bengio et al. (1994) [17]). One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they would be extremely useful. However, the main problem pointed out by Bengio et al. (1994) [17] is that a basic RNN structure as the one just explained cannot remember input values for a long gap of time/words.

6.3.1 LSTM Neural Networks

Long Short-Term Memory networks (LSTM) are a special kind of RNN, capable of learning long-term dependencies. Introduced by Hochreiter and informatik (1997) [38] they work tremendously well on a large variety of problems. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time/words is practically their default behavior.
All RNN have the form of a chain of repeating modules of neural networks. In standard RNNs, this repeating module has an elementary structure, such as a single tanh layer. The LSTM still have the chain structure, but they have four layers interacting between each other.

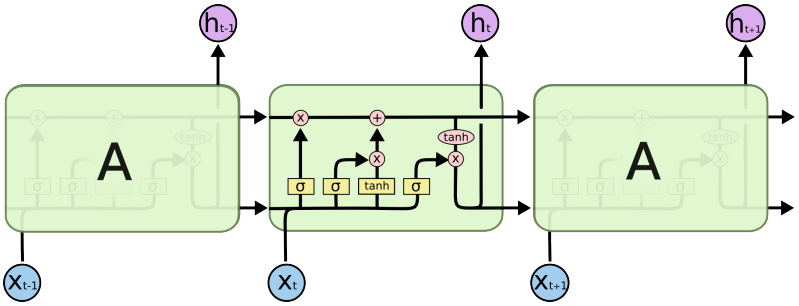


Figure 6.7: LSTM structure with the correlated notation

At each time step, the cell computes the current state Ct by combining several values

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f), \tag{6.8}$$

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i), \tag{6.9}$$

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o), \tag{6.10}$$

$$\widetilde{C_t} = \tanh(W_c * [h_{t-1}, x_t] + b_c), \tag{6.11}$$

$$C_t = f_t * c_{t-1} + i_t * \widetilde{C_t}, \tag{6.12}$$

$$h_t = o_t * \tanh(C_t) \tag{6.13}$$

$$y_t = W_y * h_t \tag{6.14}$$

At time step t the cell state C_{t-1} flows horizontally on the top of the architecture. By tweaking the parameters of the first σ called f_t or *forget gate layer*, that goes from 0 to 1, the network is able to decide how much of C_{t-1} is going to be maintained in C_t. Changing the second σ called i_t or *Input gate layer*, the LSTM changes the new output with $\widetilde{C_t}$, that is the possible new predicted value.

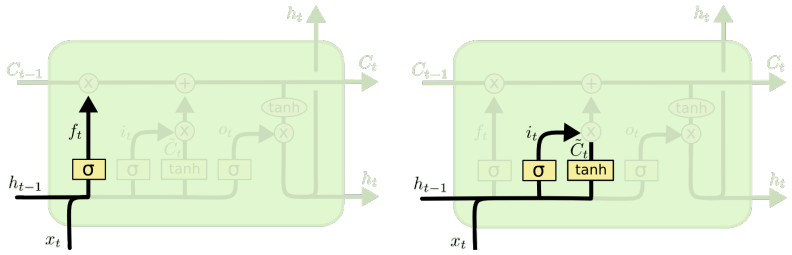


Figure 6.9: The first two steps of the LSTM at work. Images by [63]

Finally, we decide what is the output based on our cell state, after filtering. To do this, the network uses a sigmoid layer which decides what part of the cell state is going into the output. Then a tanh layer transforms the output between 1 and -1 and multiplies it by the output of the sigmoid gate so that the final result is only the parts we decided. This last gate o_t is called *Output gate layer*

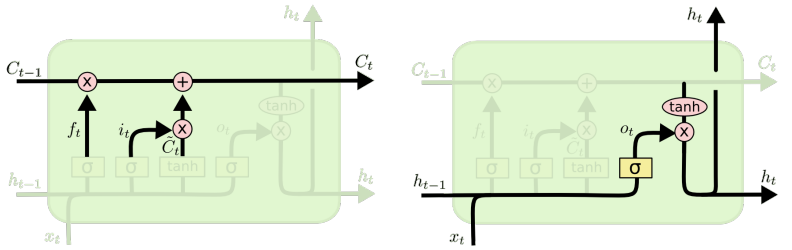


Figure 6.10: The last two steps of the LSTM at work. Images by [63]

There are many different possible structures for an LSTM and even the most "simple" proposed by Hochreiter and informatik (1997) [38] is really effective in reducing the *Vanishing gradient* that is the main problem in RNN. A big improvement to LSTM it is the use of attention mechanism that allows to visualize and understand what such an intricate network has learned. This particular improvement is explained in section 6.4.

6.3.2 GRU Neural Networks

Gated Recurrent Unit, GRU in short, in another structure of RNN initially proposed by Junyoung et al. (2015) [42]. Unlike LSTM it does not have a cell state and has two gates instead of three. A gated recurrent unit uses an *Update gate*, z_t, and a *Reset gate*, r_t. The update gate, using a σ function, decides on how much of information from the past should be let through and how much should be discarded. The output is calculated with these equation:

$$z_t = \sigma(W_z * [h_{t-1}, x_t]) \tag{6.15}$$

$$r_t = \sigma(W_r * [h_{t-1}, x_t]) \tag{6.16}$$

$$\widetilde{h_t} = \tanh(W * [r_t * h_{t-1}, x_t]) \tag{6.17}$$

$$h_t = 1 - z_t * h_{t-1} + z_t * \widetilde{h_t} \tag{6.18}$$

Even though GRU is computationally more efficient than an LSTM network, due to the reduction of gates, it still comes second to LSTM network in terms of performance. Therefore, GRU can be used when we need to train faster and do not have much computation power at hand.

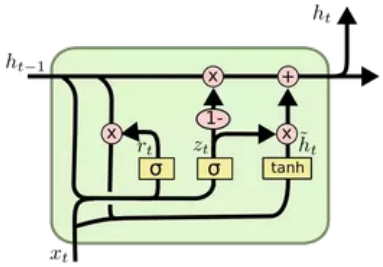


Figure 6.11: The GRU network structure.

6.3.3 Bidirectional Recurrent Neural networks

One problem with all the RNN networks explained until now is that they only learn from previous input. Sometimes it is necessary to analyze the future input in order to interpret the current input correctly. The most common example is "*He said, Teddy bears are on sale*" and "*He said, Teddy Roosevelt was a great President*". In order to correctly predict the meaning of the word "*Teddy*" we need to eliminate the ambiguity linked to it. In our work, this corresponds to the need to interpret the SMILES sequence. For example "*He*" and "*HC*": in the former, the string represents a single element, in the latter, the string is a chemical compound composed of Hydrogen and Carbon. To manage this problem bidirectional recurrent neural networks (BRNN) were proposed by Schuster and Paliwal (1997) [73]. The main idea is to use two links in two different directions between the layers.

The repeating module in a Bidirectional RNN could be a conventional RNN, LSTM or GRU. The structure and the connections of a bidirectional RNN are represented in figure 6.12. There are two types of connections, one going forward in time, which helps to learn from previous representations, and another going backward in time, which helps to learn from future representations. [11].
BRNNs can be trained using similar algorithms to RNNs because the two directional neurons do not have any interactions. However, when backpropagation is applied, additional processes are needed because updating input and output layers cannot be done at once. General procedures of training are two. In forward pass, forward states and backward states are passed first; then output neurons are passed. In backward passes, output neurons are passed first, then forward states and backward states are passed next. After forward and backward passes are done, the weights are updated.

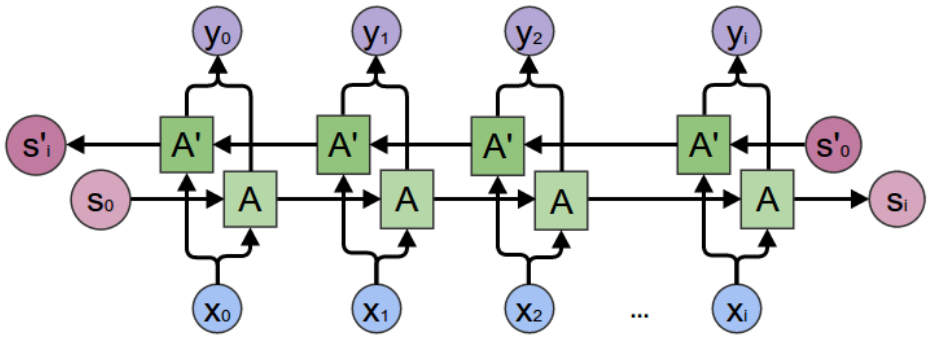


Figure 6.12: An example of bidirectional architecture.

6.4 Attention mechanism

In animals visual cortex the brain tends to focus its attention on a specific part of the data received from the eyes. That is because, in order to correctly interpret the scene, it needs to concentrate on a specific part. We follow the same process while we read an English sentence. We first focus on the subject and the verb, then we move and observe the articles and the adjectives. In the same way, a neural network can define the important parts of the input it needs to analyze first in order to make a correct prediction. This information has proven to be really performance effective, especially in RNNs. Attention mechanism was first developed by Larochelle and Hinton (2010) [49] to be applied in Image visualization. Also Denil et al. (2011) [23] studies the enhancement introduced by this method. The first application to text analysis of attention was realized by Cho et al. (2015) [21]. Cho et al. proposed a simple *Encoder-Decoder* structure with the usage of an attention mechanism in order to reduce the computation resources needed to achieve good accuracy.
The idea is to use a layer connected to the RNN that receives the context vector and

calculates the weight this has on the final prediction. More formally, taken as input a vector $(x_1, .., x_T)$ we transform it into a sequence of vector $(h_1^e, ..., h_T^e)$. From this, we can extract the probability by equation 6.19.

$$p_\sigma(y_t|c_t, y_{t-1}, ..., y_1) = softmax(g(v(h_t, c_t))) \tag{6.19}$$

From 6.19 it is possible to introduce the *Attention weight* in order to speed up the calculus process.

$$\alpha_t = \frac{e^{p_\sigma(t)}}{\sum_{s=1}^{S} e^{p_\sigma(s)}} \tag{6.20}$$

The creation of this weight allows the attention mechanism to be trainable and to be subjected to backpropagation. In this way, it is possible to enhance the performance of the network.

6.5 SMILES-Net

The characteristics explained in the precedents sections, *i.e.* the ability to treat sequential input and the great performances, are the reasons why we decide to apply them in this work. In this section, we discuss the choices made during the architecture design process, and the final model obtained. In section 6.5.1 we discuss the embedding used for this unusual problem. Meanwhile in section 6.5.3 we discuss the main advantages in using *Attention mechanism* and how we decide to use it.
The molecule classification using RNNs has never been realized in the literature. We took inspiration from Zhou et al. (2015) [87] and Goh et al. (2017) [30], combining different methodologies and creating a whole new network. The idea proposed in this work is to use a simple text classifier based on a bidirectional LSTM/GRU, combined with a SMILES-embedding. At the end of the network, we also insert an attention layer, a dropout layer, and a dense layer in order to, respectfully, extract the context vector, avoid over-fitting and classify the correct endpoint.

6.5.1 SmileEmbedding

The idea proposed by Goh et al. (2017) [30] is to use a set of SMILES as input for a network composed of two LSTM, in order to extract the vector representation of the SMILES given by the network. We actually do a different process, but we started replicating this solution. In order to have a good representation of the SMILES, we mixed this method with the word2vec, explained in 6.2.2.
Let us define some terminology in order not to create misunderstanding. The word "*word*" in this section represents the indivisible sequence of characters that can be found in a string, in our specific case, words are the elements present in the *Periodic table*. In particular the complete dictionary of our database contains also some special

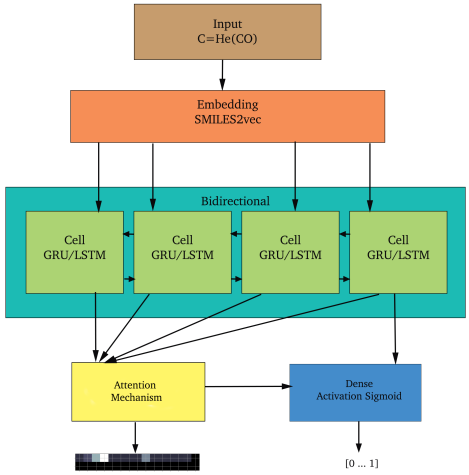


Figure 6.13: A schematic representation SMILES-Net

characters, *i.e.* *],/,@,etc.*, that can be commonly found in a SMILES string. We created the dictionary of our dataset by iterating on each string in order to find either an element of the periodic table or a special character; this task is simplified by the fact that, all the elements are one or two characters long, with the first letter always written in capital letters and the second always in minuscule. Once the dictionary is created we pass it to an embedding layer. The embedding layer is based on the method explained in section 6.2.2. For example given the string "*(Cl)CC*" we pass to the embedding layer the dimension of our dictionary and the input length, combined with the input in numbers

$$[1, 2, 3, 4, 5] \qquad (6.21) \qquad (x_1, ...x_n) \qquad (6.22)$$

Once the layer has been trained, it creates a table with all the possible tensors to use. In particular, the embedding adds a new dimension. Our example becomes

$$[[0.7, 1.7], ..., [4.1, 2.0]] \qquad (6.23) \qquad ([x_1, h_1], ...[x_n, h_n]) \qquad (6.24)$$

It might seem counter-intuitive at first, but the underlying automatic differentiation engine manages to optimize these vectors associated with each input integer just like any other parameter of our model. Once the string has been transformed into a vector, we can train our classification network.

6.5.2 Cells

As explained in section 6.3 the choice of the type of RNNs to insert in the network is really important. For this reason, we defined the type of the cell used as hyper-parameter in the optimization process. However, in this section we define some standards and

the characteristics the cell must have in order to fulfill our requirements. In particular, as explained in section 2.2.2, the sequence of symbols in the SMILES represents the chemical connection between elements. It means that different ordered sequence and small changes to the string can totally revolutionize the result of the toxicity test. For example, taken two strings

CC(C)CN3C#N *CC(C)CNC*

It is possible to remark that only a few letters at the end are different. In order to avoid ambiguity in all the phases of the computation, we decided to use a bidirectional cell. It allows us to have the previous input but also the future input, in order to easily distinguish the differences, as explained in section 6.3.3.

6.5.3 Attention and Fragment extraction

As we mentioned in the introduction of this chapter and in section 3.1.2, one of the main goals of this thesis is to create an enhanced and transparent system. To do that we need to extract and analyze the knowledge that the network creates during the training procedure; besides accuracy, we need to evaluate the model in terms of the functional groups (subparts of the molecule) that are found, so to compare our model with existent knowledge. We add to the network an attention layer, described in section 6.4.

Listing 6.1: The function used to extract the fragment from SMILES-Net

```
def getFeatures(text,threshold):
    d = data.SMILE2Int(text,vocab)
    d = pad_sequences(np.array([d]),
                      maxlen=max_size,
                      padding='post')
    predicted_text = "0"
            if pred_model.predict(d).round()[0][0]==1
            else "1"
    results = pred_model.predict(d)
    ...
    pred = proba_model.predict(d)
    predicted = np.zeros((input_length,2))
    \# get the activation map
    activation_map = np.squeeze(pred[1][0])
    for i in range(0,input_length):
        if predicted_text=='1':
            predicted[i,0]=activation_map[i]
        else:
            predicted[i,1]=activation_map[i]
    predicted = np.rot90(predicted)
    prob = max(activation_map)
    sequences = extractSequences(activation_map, threshold)
    ...
```

This layer allows to create a correlation map between the output of the recurrent network. This matrix is used by the dense layer to focus on the important parts of the parsed string. Section 6.4 explains how attention works. Moreover, we extract the context vector created by the network for each string of our dataset creating a matrix composed of vectors. Using a threshold on the correlation value, we can select adjacent characters, forming a string which represents an important part of the SMILES that is used to predict the final endpoint. We can create a correlation map between each character of the SMILES string and the final prediction. Instead of the big matrix, we can also draw the graph using a correlation based plot that helps the user to understand what it is going on inside the network.
An example of such results is in figure 6.14. We also extract the fragment in string format with the function in listing 6.1 iterated over all the SMILES in the dataset. This operation is fundamental because many methods, as explained in section 3.3.2, are able to extract these fragments, and experts need them in order to make a decision. It is important to note that in line 22 we passed to the function *extractSequences* (a basic iteration over the string and the activation map) the variable threshold that we set to 0.05. That is because from a SMILES it is possible to get almost infinite combinations of characters and a filter is needed in order to have meaningful data to analyze. Moreover, the code uses two models, in line 9 and line 11; the first one, *predModel*, is the network explained in this section. The second, *probaModel*, is instead the same model explained in this section without the dense layer at the end. That is because we need to obtain the context vector from the network in order to extract the fragments. The two models are already trained and the context vector extracted are then based on the final weight values. This technique uses a particular method, called *Transferable learning*, to not train over and over the network. The details can be found in chapter 7 but the main idea is to save the weights of a model into a file and load them into another model as starting weights. In this ways, there is no need to train the new network again. The dots (...) inserted in the code indicate basic python instructions, needed to run the code. In chapter 8 we compare our fragments with one famous SAR model, *SARpy*. Some of the resulting strings are in table 6.1. As we can see, there are two probabilities predicted by the network: the probability of being a mutagen and the probability not to be toxic. That is because of the possibilities to have multiple endpoints. It is also a countercheck to evaluate the correct functioning of the network.

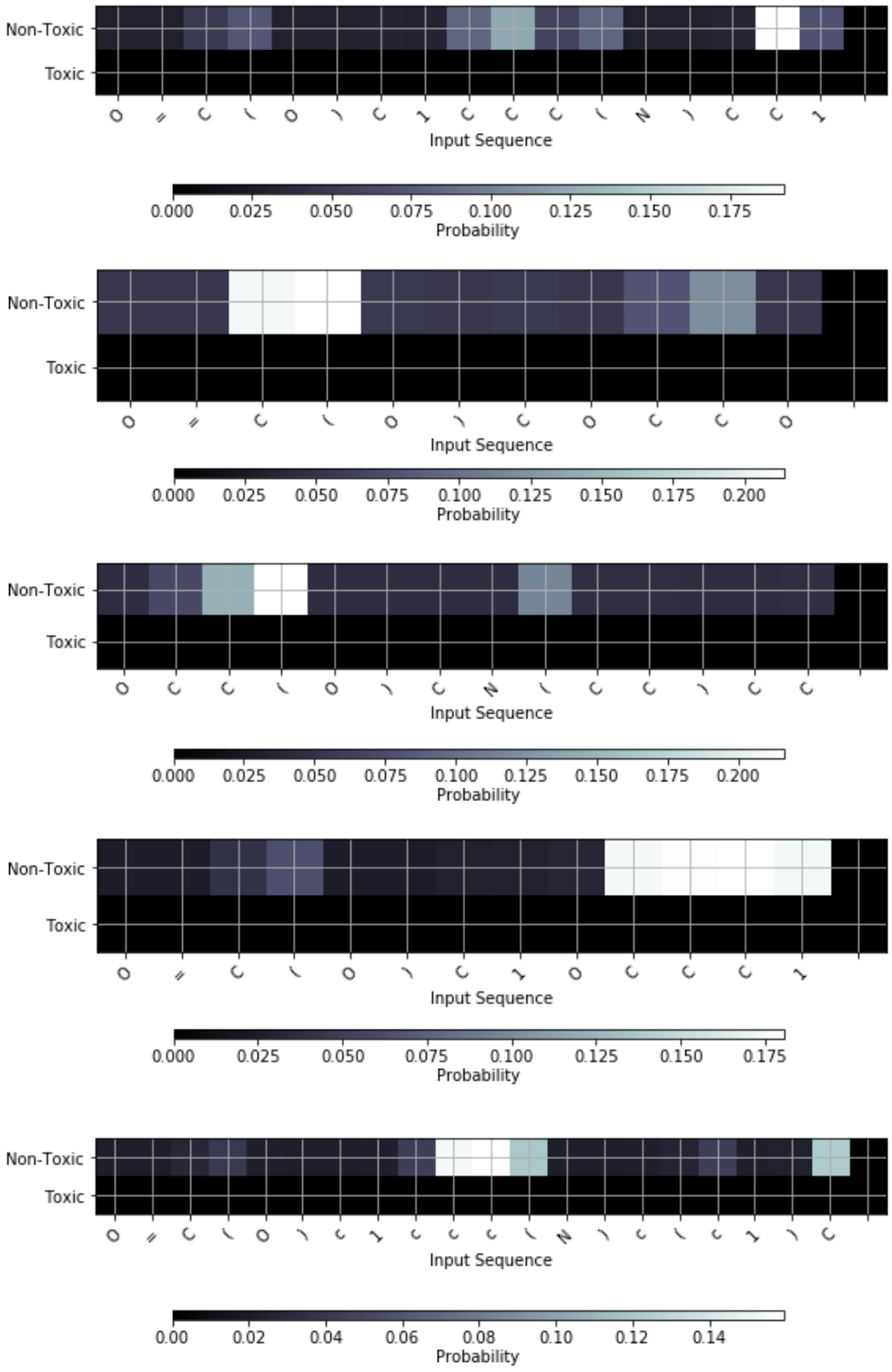


Figure 6.14: Some correlation maps of the fragment extracted.

Fragment	Mutagen Prob	No-Toxic Prob
=C(N1)N(=	0.017980922	0.98201907
N=C1Cl)N(=	0.018495886	0.9815041
C=C(	0.019155307	0.98084474
=C(S1)N(=	0.021767307	0.97823274
=NØ)N(=O)=	0.022063345	0.9779367
S(=O)(=	0.026124591	0.9738754
=N)NN(=O)=O	0.027941832	0.9720581
C(=N)NN(=O)=O	0.028735904	0.9712641
C=C(O1)N	0.029339418	0.97066057
N=C1N(=O)=	0.031243872	0.9687561
#CCc1cc	0.98075986	0.019240092
N#CCc1cc	0.98075986	0.019240092
CCCCI	0.9816908	0.018309135
CCCCCI	0.9816908	0.018309135
ICI	0.98542863	0.014571338

Table 6.1: Some fragments in string formats

6.6 Optimization process

As we did in section 5.4, we optimized the network using *Talos*. The main difference between these two processes is the list of hyper-parameters used. As this network was never theorized in the literature, we had to perform a more detailed analysis to find the best parameters to use. The main challenge of the optimization process, in this case, is the time needed to run all the different networks with different values. Differently from chapter 5, *SMILES-Net* takes more time to run, due to the usage of LSTM. In the evaluation process, we also added the comparison of the extracted fragments with the fragments resulting from SARpy on the same database. In this section we discuss only the different parameters and method use, the results are reported in section 6.7. We focused our attention on the computation resources used and on the performance metrics defined in chapter 4. In particular, we want to achieve the most powerful network with the smallest number of resources possible. For these reasons, we chose to analyze two different cell types. Maintaining the *Bidirectional* cell we studied the LSTM compared with the GRU. We also analyzed the type of operations inside of the cell itself, in particular we tested: *multiplication*,*sum* and *concatenation*. In these types we looked for the optimum *learning rate* starting from 0.5 using this sequence: 0.5,0.1,0.01,0.001,0.0001. We introduced an early-stop technique to reduce the possibility of over-fitting, and we looked at the number of epochs needed to the network to converge. Also, the loss function of our network was used as a parameter. In particular we used: *mean squared error*,*cross-entropy* and *log-cosh*.

In order to discover the full possibilities of *SmileNet* we tested different optimizers with different learning rate schedule: *Adam, RMSProp*. We also analyzed the number of neurons for each layer, that is the numbers of features that is possible to extract

from each row of the dataset. It is one of the most important factors in computational resources uses by the network as the increasing of this number indicates the increase in the number of parameters to train in the model.

Hyper-parameter	Application Range
Learning Rate	[0.0001,0.05]
Batch size	[16-128]
Epochs	[25-200]
Cell	*GRU,LSTM*
Loss function	*Mean-Squarred-Error,Log-Cosh,Binary-Crossentropy*
Optimizer	*Adam, RMSProp*
Dropout	[0-0.5]
Neuron number	[20-400]

Table 6.2: A summary of the hyper-parameters chosen. The square parenthesis indicates a continuous range.

Finally, we studied the optimal dropout rate to avoid over-fitting and the batch size of our input. That is the number of samples that will be propagated through the network during a single iteration. The number of iterations of one epoch is calculated by the total training dataset divided by the *batch size* chosen. The correct definition of this parameters is mandatory to save resources during the training. In fact, even if the first idea would have been to use only one iteration per epoch to train faster, the usage of batch reduce the memory required by the network and its computation time use. At the same time, the smaller is the batch, the more accurate will be the gradient estimation during training. In the table 6.2 there is a summary of the hyper-parameters analyzed and the tested range.

6.7 Results

SMILES-Net is a new application for computational toxicology. For this reason, we had to deeply analyze the network in order to extract all the possible pieces of information about it. In order to do that we run twenty-five different combination of hyper-parameter, this allows finding the correlation between the parameters and their effects on the performance results. The number of tests could seem too weak compared to the assays done on *Toxception* and *C-Tox*, the reason why this is such a small number is that *SMILES-Net*, due to its recurrent structure takes a lot of time to run. In particular, one step is around 2 seconds. This, multiplied for the number of steps per epochs (around 600) and for the number of epochs gives an average training time of 20 hours. The expensive training time is also the reason why we did not evaluate, in a first stage, the network using sensitivity and specificity. The usage of these metrics, in fact, implies a second prediction at the ends of each epoch. This increase, even more, the computational time needed.

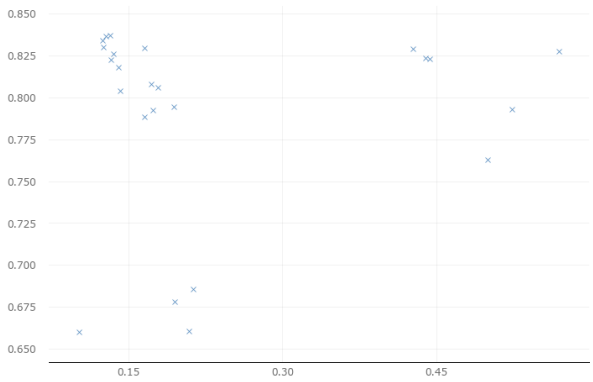


Figure 6.15: SMILES-Net optimization results distribution. The x-axis is the values obtained for the validation loss metrics. The y-axis instead is the validation accuracy.

Once again we used 20Another interesting aspect to point out is the differences between *LSTM* and *GRU*. In particular, from the tables, it is clear that with a sufficient number of epochs the *LSTM* can achieve better results. It seems from the general histogram that the LSTM performs worse than the GRU cell. This is actually misleading, figure 6.15 shows an average score for all the combinations that contain an LSTM cell. Again here the time is an essential metrics. The usage of the GRU cell, as explained before is the most time efficient approach compared to the performances.

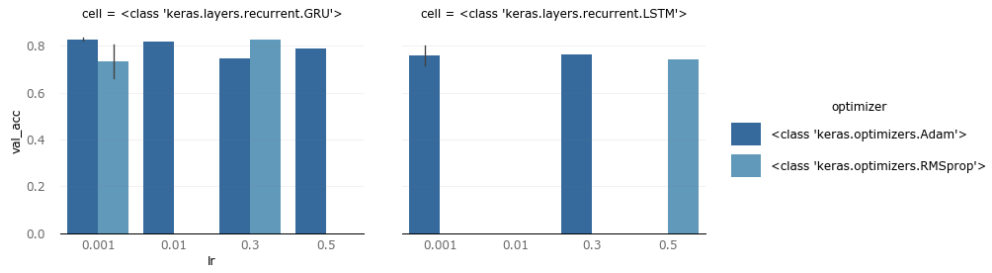
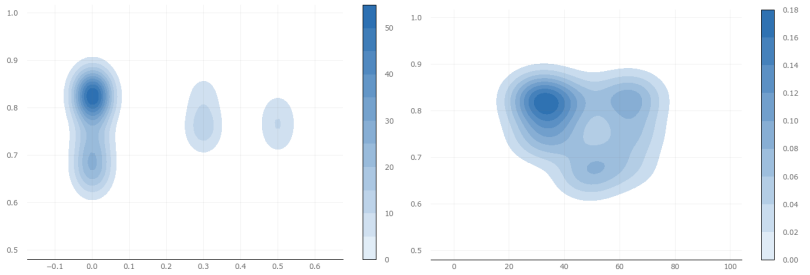


Figure 6.16: Average values distribution of the validation accuracy, in the dependent axis, with the correlated parameters on the independent axis. In particular, the y-axis is the learning rate, the colon is the different types of cells, and the color represents the optimizer used.

The batch size is another parameter to analyze. In fact, we can see from figure 6.17, how the values distribution is concentrated in the proximity of the best validation accuracy. We trained the model with a batch size from 16 to 64 and it is interesting to remark as the best values, even in the list of optimal results in table 6.3 and 6.4, are obtained between 32 and 64, of course in order to reduce the calculation of the GPU will be useful to use a power two. That is because all the calculus can be converted into a binary format with the less effort. However, a small value of the batch size is usually preferred, but from these results, we can notice that the smallest value is

discarded. That is because the optimization process was implemented with a time limit of each assay. A small batch size implies a better performance but also a more extended training, the values obtained are a good compromise between the drawbacks of the batch size.



(a) *lr* vs. Validation accuracy **(b)** Batch size vs validation accuracy

Figure 6.17: The graphs show the distribution of the validation accuracy depending on the values of batch size and learning rate.

Observing the tables we can denote again how the *crossentropy* function is still higher than the *MSE*, that because the *crossentrpy* measures the distance between the distribution probability of the prediction rapport to the real value. This indicates the correctness of the prediction, however the minimized function tends to start, in average, far away from the convergence point. This means that we would need even more training time in order to perfectly reach the minimum. This implies the insertion of other dropout neurons in order to prevent the metrics to over-fit. This is proven by the fact that only one of the combination tested have a loss function bellow 0.20 and this best option has the highest number of epochs. Moreover it is useful to point out that all the training executed do not reach the maximum level of accuracy. In a rough analysis, this would mean that the network could be trained even more. Actually analyzing in details each training we can see that the validation loss starts increasing, and the model starts over-fitting after only a few epochs after the reported one. It means that on this database the model could not learn more than a certain upper bound.

Independently from the different parameters used the model have a clear learning limit. In order to overpass this, there are a few possible operations that could be done. The most important is explained in detail in section 9.1

As we did for *Toxception*, we extrapolated the best results in order to evaluate the other metrics calculable during the optimization process. We calculated all the other metrics explained before in order to obtain a comparison between our proposed methods and the current machine learning state of the art. At first glance the two graphs in figure 6.18 the results seem not to move away from the results of *Toxception*. In fact, except for the most harmonic form of the accuracy and loss function, the values are really similar. Going deeper into the other metrics however shows a peculiar trend in all the analyzed functions, *MCC*, *Specificity* and *Sensitivity*. Like all the other

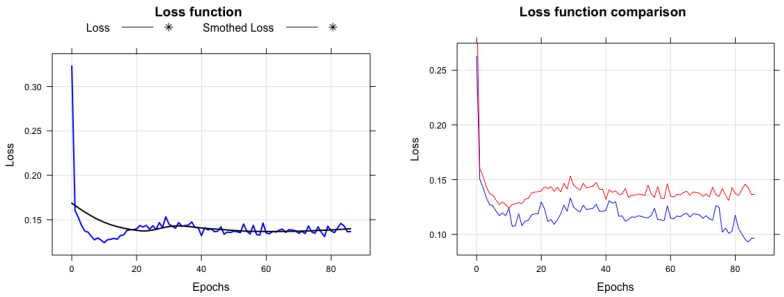
metrics they stabilized after a small period. However, observing the parameters and their variations we can see how the values are almost the same, varying of only a few hundredths. It can be explained with two phenomena: the first is the training time, in fact, the two model used different units of measurement in term of iterations or training epochs. The second argument to consider is that the input passed into *SMILES-Net* are strings that have almost never been modified. The input of *Toxception* instead are generated from this string by a framework that introduces a small error in the process of image generation. It means that the image generation function does not create an exact precise graph based on the string but makes an image 80x80px. This sampling process always introduces a decrease of the accuracy. In particular, when the compound structure is really complex the dimension of the image become a big limitation and the accuracy considerably decrease. It is shown on our models by the fact that more complex molecules could be more inclined to be toxic and could than augment the number of false negative predicted by the network. That is why both the *MCC* and the sensitivity of *SMILES-Net* are higher and more stable if compared with Toxception. Observing the validation accuracy and the validation loss we can see that the accuracy remain stable while the loss function gets to the minimum and starts increasing, we decided to do not trust the validation accuracy to define the starting over-fitting point, but we chose the validation loss instead. In the graph 6.18 it is remarkable as validation loss (in red) and the loss function (in blue) follow the same pattern and start diverging at the end of the training. It means that the model is still actually training and the techniques applied to the network to avoid the over-fitting are actually working. However, we can see the gap between the two line that it is stable and around 0.02 in the center of the graph.
The last aspect of this part we want to point out is the drop in the loss function at the beginning of the training followed by a sudden jump at a stable value. It is certainly not a great result and can indicate various things. First of all, it can indicate that the model is too small and we need to go deeper as it could mean that the model gets all the information that it needs at the beginning. However, see the results we interpret this gap as an indicator of the data quality. It means that the dataset is not big enough (even if it is not small) to reach the best result on the studied problem.

Architecture	Parameters							Metrics	
	Ep	Lr	Neurons	BS	Drop	Opt	Loss	Val Acc	Val Loss
SMILES-Net_GRU_Sum	100	0.001	400	64	0.2	*Adam*	*MSE*	0.836	0.131
	100	0.01	400	32	0.2	*Adam*	*MSE*	0.817	0.140
	50	0.3	400	32	0.2	*Adam*	*MSE*	0.807	0.172
	50	0.001	400	64	0.2	*RMS*	*logcosh*	0.803	0.141
	50	0.5	100	32	0.1	*Adam*	*MSE*	0.788	0.165
SMILES-Net_GRU_Mul	50	0.001	100	64	0.2	*RMS*	*logcosh*	0.66	0.104
SMILES-Net_LSTM_Sum	100	0.001	400	32	0.2	*Adam*	*MSE*	0.836	0.127
	100	0.001	400	32	0.2	*Adam*	*MSE*	0.830	0.125
	50	0.001	200	50	0.2	*RMS*	*MSE*	0.677	0.195
SMILES-Net_LSTM_Concat	50	0.001	200	32	0.2	*Adam*	MSE	0.829	0.165
	50	0.001	200	64	0.1	*Adam*	MSE	0.724	0.194
	50	0.001	400	41	0.1	*Adam*	MSE	0.722	0.174
	50	0.01	100	64	0.4	*RMS*	MSE	0.685	0.213
	50	0.001	100	64	0.4	*RMS*	MSE	0.660	0.208
SMILES-Net_LSTM_Mul	100	0.001	400	32	0.2	*Adam*	*MSE*	0.822	0.106
	50	0.001	200	32	0.1	*Adam*	*MSE*	0.806	0.176

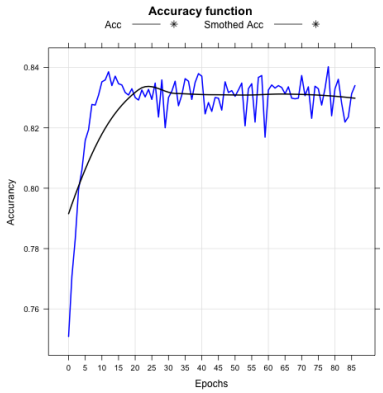
Table 6.3: SMILES-Net Optimization results using *MSE*

Architecture	Parameters							Metrics	
	Ep	Lr	Neurons	BS	Drop	Opt	Loss	Val Acc	Val Loss
SMILES-Net_GRU_Sum	200	0.001	400	64	0.2	*Adam*	*xentropy*	0.823	0.131
SMILES-Net_GRU_Mul	100	0.001	400	32	0.1	*Adam*	*xentropy*	0.823	0.439
SMILES-Net_GRU_Concat	50	0.3	200	48	0.1	*RMS*	*xentropy*	0.827	0.570
SMILES-Net_LSTM_Sum	100	0.001	400	32	0.2	*Adam*	*xentropy*	0.834	0.124
	100	0.001	200	32	0.2	*Adam*	*xentropy*	0.828	0.421
	50	0.5	400	64	0.2	*RMS*	*xentropy*	0.792	0.524
	100	0.3	400	64	0.2	*Adam*	*xentropy*	0.762	0.499

Table 6.4: SMILES-Net Optimization results using *xentropy*

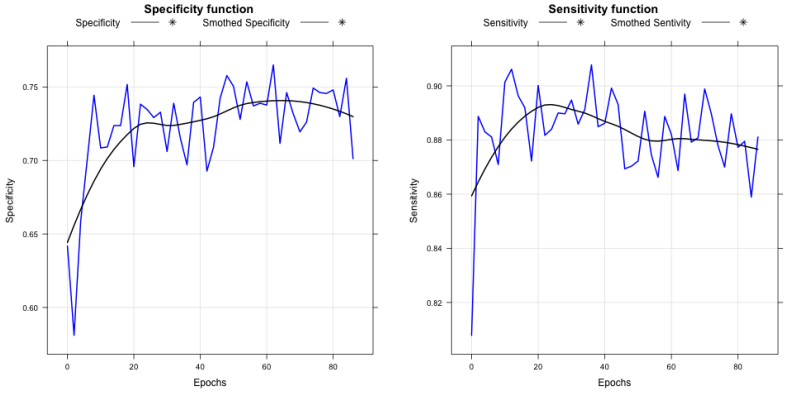


(a) loss **(b)** loss comparison

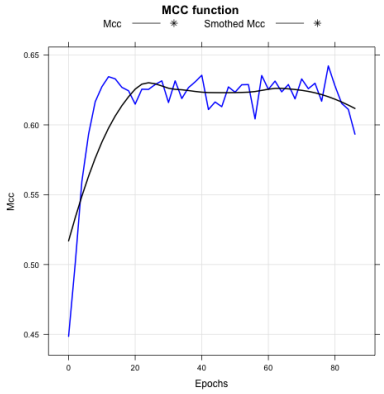


(c) accuracy

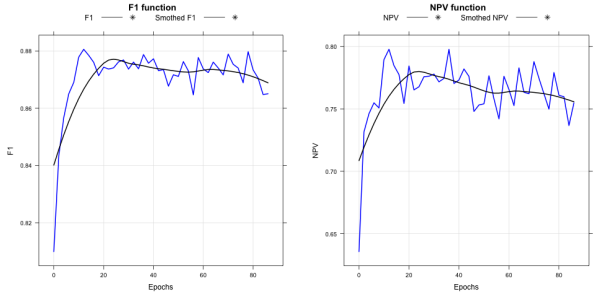
Figure 6.18: Accuracy and loss function for SMILES-Net trained with *MSE* loss function and *Adam* as optimizer. All the metrics reported here are calculated on the validation set, some initial values are cut out in order to have a better vision of the values and the trend.



(a) Specification **(b)** Sensitivity



(c) MCC



(d) F1 **(e)** NPV

Figure 6.19: Sensitivity, Specificity, MCC, F1 and NPV for SMILES-Net trained with *MSE* loss function and *Adam* as optimizer. All the metrics reported here are calculated on the validation set, some initial values are cut out in order to have a better vision of the values and the trend. It is interesting to note the clear and harmonic trend that all the curve have. Avoiding of course the small perturbations that are however not really relevant.

Chapter 7

C-Tox

In the preceding chapters, we explained two complementary models that analyze molecules in two different ways. In this chapter, we are going to merge them creating a single unique net in order to achieve even higher efficiency and effectiveness in the mutagen prediction. The networks explained before, *SMILES-Net* and *Toxception* can be used not only for prediction but also for data featuring, That is the generation of features, *i.e.* important parameters, about the input. As the basic methods used in the current state of the art, we used a simple feed-forward network to find the correlation between these features and to classify the endpoint. We call it, C-Tox. In particular, in this chapter we are first going to analyze the basic theory behind feed-forward networks, in section 7.1, then in section 7.2 we explain in detail C-Tox and its architecture. Finally, as we did in chapters 5 and 6, in section 7.3 we explain the optimization parameters used and the evaluation methods adopted on the complete model, T-Tox. Also, in section 7.4 we run the whole model, T-Tox composed by the three networks.

7.1 Feed forward neural networks

Deep feed-forward networks, also often called feed-forward neural networks, or multi-layer perceptrons(MLPs), are the quintessential deep learning models. The goal of a feedforward network is to approximate some function f^*. For example, for a classier, y = $f^*(x)$ maps an input x to a category y. A feed-forward network denes a mapping $y = f(x;\sigma)$ and learns the value of the parameters σ that result in the best function approximation.Goodfellow et al. (2016) [33]. These networks are the basis for the two models explained before. They are also a mandatory "must know" for a computer science engineer. That is why we give a brief explanation of their theory only at the end of this work. As we know, the inspiration of neural networks is our brains, and they emerged from a very popular machine learning algorithm named *perceptron* by Rosenblatt (1958) [69].

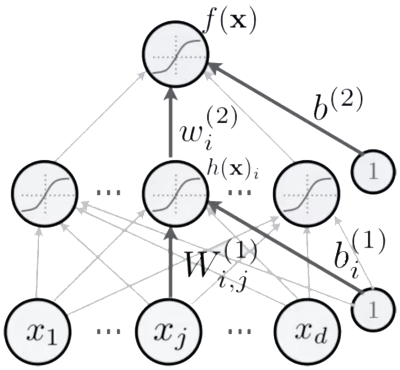


Figure 7.1: A schematic representation of a neural network.

A representation of a simple neural network is in figure 7.1. The leftmost layer in this network is called the *Input layer*, and the neurons within the layer are called input neurons. The rightmost or *Output layer* contains the output neurons, or, as in this case, a single output neuron. The middle layer is called a *hidden layer* since the neurons in this layer are neither inputs nor outputs. $W^i_{(i,j)}$ denotes the weight for the connection from the j^{th} neuron in the $(l1)^{th}$ layer to the i^{th} neuron in the i^{th} layer. We use b_i for the bias for the i^{th} neuron. $h(x)$ is the activation function, that is the function that decides the value of the output for each layer. These networks are really interesting for their main property to approximate every finite dimension function. In fact the *Universal approximation theorem* by Chen and Chen (1995) [20] states:

> *A feedforward network with a linear output layer and at least one hidden layer with any"squashing" activation function (such as the sigmoid activation function) can approximate any Borel measurable function from one nite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units.*

It means that there exists a network large enough to achieve any degree of accuracy we desire. In order to find the correct function $f(x)$ we define a *Cost function* that defines the error function to minimize. That is because in these networks it is easier to evaluate the correct value of the weights and bias with a smooth predefined function instead of maximizing an unknown one. The most famous are the *Mean square error* or *MSE* and the *Cross-entropy*. Neural networks are usually trained by using iterative, gradient-based optimizers that merely drive the cost function to a very low value. In particular we use the following equations:

$$\Delta C \simeq \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2. \tag{7.1}$$

This can be written as

$$\Delta C \simeq \nabla C * \Delta v \tag{7.2}$$

where

$$\nabla C \equiv (\frac{\partial C}{\partial v_1}, ...\frac{\partial C}{\partial v_m})^T \tag{7.3}$$

$$\Delta v \equiv (\Delta v_1, ...\Delta v_m)^T \tag{7.4}$$

From these it is possible to prove that

$$\Delta v = -\eta \nabla C \tag{7.5}$$

$$\eta = \frac{\epsilon}{\|\nabla C\|} \tag{7.6}$$

From these equation we can derive then the update rules for basic neural network that are:

$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k} \tag{7.7}$$

$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial w_k} \tag{7.8}$$

In particular η is also called learning rate and it defines the step that the network does at each timestep, this indicates how fast the network is going to learn.
These are the basic equations used by the network, in the precedent chapters we explained the *Backpropagation algorithm* that can efficiently improve the network performances. In this part of the project we decided to keep the network as simple as possible in order to perform the prediction in the smallest time. In particular we used two principal techniques to achieve this goal. The first one described in section 7.1.3 is based on the idea that we can save the training of the network without repeating it each time we need to do a prediction. The second main technique used is the dropout, presented in section 7.1.4, that is the process of discarding some random weight based on probability. Before explaining these methods we report the basic theory of the activation function and the optimizer used to train the networks. That is because because the proposed work is based on this basic topics and uses both to train the whole model.

7.1.1 Activation function

The activation function of a node defines the output of that node, or neuron, given an input or set of inputs. Every, or almost every, neuron has an equivalent activation function. It exists a numerous amount of possible activation functions. Some of them can be found in figure 7.2. The activation function is really important in feed-forward network as it is the main parameter that influences the change trend of the weights. For example, the fist common function described in literature was *Sigmoid*.

$$\frac{1}{1 + e^{-x}} \tag{7.9}$$

This particular function is really sensible when the input is around 0, this means that when we have small weight, due to the gradient descent algorithm application, we can have the *Gradient vanishing problem*. It means that the network cannot improve too much because its weights are too small and do not affect the output. To solve this problem there are other functions as the *Tanh*.

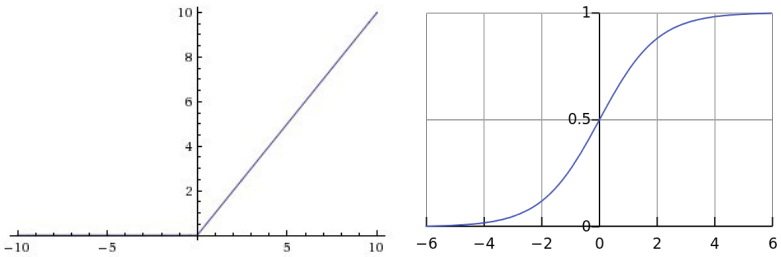
$$\frac{2}{1+e^{-2x}} - 1 \tag{7.10}$$

It is really similar to the sigmoid but it changes dramatically near 0 in order to avoid the problem described before. The most currently used is the *ReLu*.

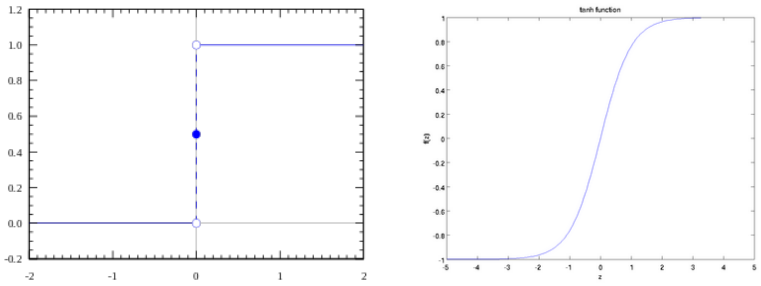
$$max(0, x) \tag{7.11}$$

This function, as stated by Chen and Chen (1995) [20] can approximate all the functions with enough hidden layers.

$$\frac{1}{1+e^{-x}} \tag{7.12}$$



(a) ReLU **(b)** Sigmoid function



(c) Step function **(d)** Tanh function

Figure 7.2: Different activation functions

7.1.2 Optimizer

The *Optimization algorithms* helps us to minimize (or maximize) an Objective function (also called Error function) $E(x)$ which is simply a mathematical function dependent on the model internal learnable parameters which are used in computing the target values from the set of predictors used in the model. The different optimization algorithms are called Optimizer. In this section, we discuss just three of them, which are the main algorithms used in this work. The first two are the *RMSProp (Root Mean Square Propagation)* optimizer and the *SDG (Stochastic gradient descent)*. Both are of the same family of optimizers and use the stochastic gradient descent algorithm to optimize the learning. SDG was the first to be discovered, and it updates the weights with

$$w := w - \eta \nabla Q(w) \tag{7.13}$$

as the gradient $\nabla Q(w)$ of our loss function is not known apriori in the iterative algorithm the values is approximated with a gradient at a single step:

$$w := w - \eta \nabla Q_i(w) \tag{7.14}$$

Moreover, as we use batching in training the $\nabla Q_i(w)$ is calculated not on the single data row but on the batch, in order to be more precise.
RMSProp instead introduces the *adapting learning rate*. It adapts the learning rate for each of the parameters. The idea is to divide the learning rate for weight by a running average of the magnitudes of recent gradients for that weight. First, the running average is calculated in terms of means square,

$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2 \tag{7.15}$$

So the parameters are updated as

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w) \tag{7.16}$$

The third optimizer is the Adam algorithm, *Adaptive Gradient Algorithm*. It was presented by Kingma and Ba (2014) [45] in 2015, and it is an optimization algorithm that combines the advantages of the other two extensions of stochastic gradient descent. It maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g., natural language and computer vision tasks). It means that the algorithm performs well also in noisy problems. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients. It has three parameters that can be optimized:

- *Learning-rate*. The proportion that weights are updated, Larger values results in faster initial learning before the rate is updated. Smaller values instead slow

learning right down during training

- β_1 The exponential decay rate for the first moment estimates.

- β_2. The exponential decay rate for the second-moment estimates. This value should be set close to 1.0 on problems with a sparse gradient (e.g., NLP and computer vision tasks).

7.1.3 Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. This optimization allows rapid progresses or improved performances when modeling the second task. Nevertheless, transfer learning is popular in deep learning given the enormous resources required to train deep learning models or the large and challenging datasets on which deep learning models are trained. There are two main methods to use transfer learning: *Develop model approach* and *Pre-trained model approach*. In the former, we create and train our first model, and we use it to perform a second task. In the latter instead we take some model from the literature, and we reuse them as a starting point for our proposal. In this work, we used the first methodology, and we created two main methods with the task of featuring chemical molecules. We reuse then the trained model into T-Tox, and we predicted the toxicity endpoint of these molecules based on the features extracted. This technique is based on a straightforward concept. We can store the weights and the architecture of one model inside a matrix. Once our model is stored, we can then decide to load these data into a similar/equal structure that is able to reuse the same weights. In particular we used the *.hd5* notation to save our models and we used a particular function of *Keras* to retrieve them.

7.1.4 Dropout

The definition of dropout in neural networks is

> The term "dropout" refers to dropping out units (both hidden and visible) in a neural network.

More formally this means that, at each training stage, individual nodes are either dropped out of the network with probability $1-p$ or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. The result after dropout is showed in figure 7.3

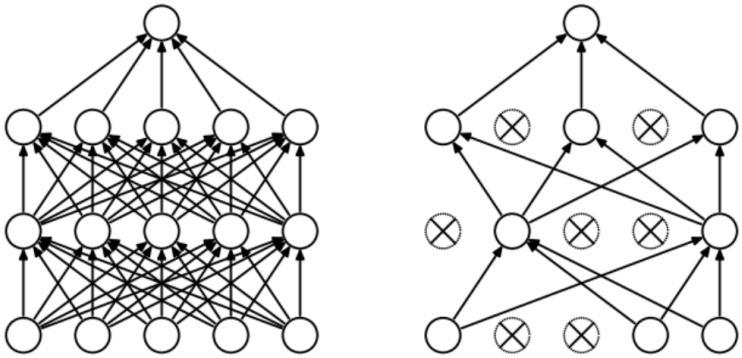


Figure 7.3: A schematic representation of neural network without and with dropout.

This regularization reduces over-fitting by adding a penalty to the loss function. By adding this penalty, the model is trained such that it does not learn an interdependent set of features weights. Moreover, dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. However, it also increases the number of iterations required to converge, even if it reduces the time per epochs during the training.

7.2 C-Tox

C-Tox derives from *Classifier for toxicology*. We decided to call it C-Tox because this final proposal is the last part of our model which connects the network explained in chapter chap:Chem and 6. In particular, it performs the classification for the whole model, T-Tox.
In order to achieve a classification, we had to remove the fully connected layers present in Toxception and SMILES-Net. This makes the connection between the pool layer and the attention layer direct to the fully connected layer of C-Tox. That is because we wanted to use a more detailed classification, that was able to find a more similar function to the non-linear toxicity function searched. The whole architecture is shown in figure 7.4

C-Tox is composed of several fully connect layers which have an activation function and a dropout layer just after the output. The number of this triples is discussed in section 7.3 as we tested from two to one hundred repeated block. At the end of these blocks, we added the final fully connected layers, also separated by dropout layers. The last dense layers are used to decrease the number of neurons belonging to each layer gradually.
In order to avoid time-consuming training, we did not train the first two block of this network again. We used instead the *Transfer Learning* technique, explained in section 7.1.3. With this method, we could use the weights determined in the previous training. It allows training only the last part of the model, C-Tox. Doing that, the training time is tremendously reduced.

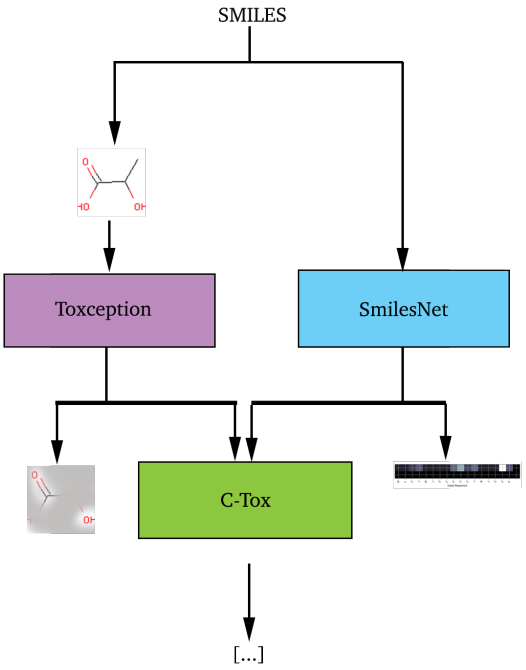


Figure 7.4: A schematic representation of T-Tox

7.3 Optimization process

Differently, from the other chapters in this last section, we do not treat only the sub-network. In this case, we actually discuss the optimization of the whole model. However, we decided to put this part in this chapter as it is the end of the network. It would have been impossible to train C-Tox without training the whole model with it. Besides, as we were working on the final proposal, we did more tests than in the previous sub-net.

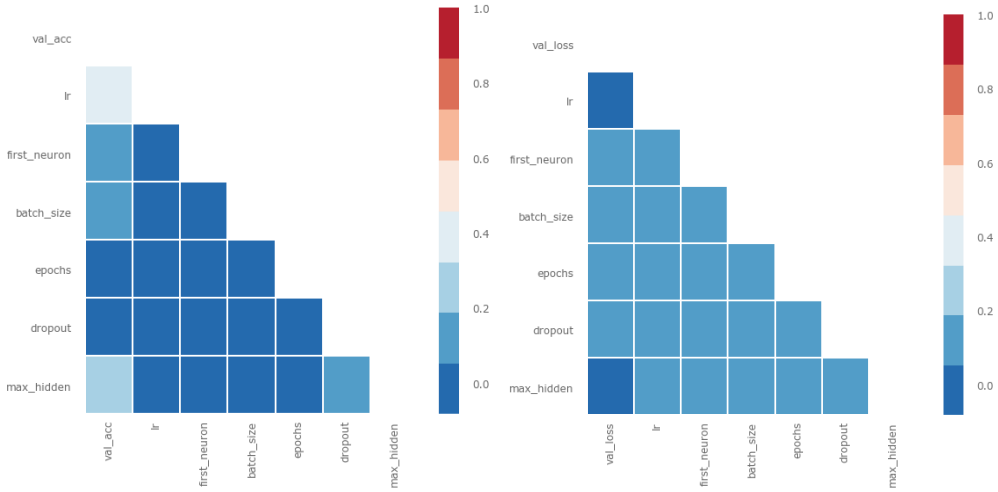
In order to understand which network could work better for our task, we prepared a large number of simulations with a high number of hyperparameters in order to discover the whole network responses and performances. We adopted as before two possible type of optimizer: *Adam* and *RMSProp* and we changed the optimizer parameters into a bigger range than the one tested in the previous networks. We also added a specific test on the different possibilities for the activation function of each layer. In order to study in detail the network we tested a combination of the possible activation functions explained in section 7.1.1. We also tested different combinations of dropout in order not to lose too much accuracy without overfitting.

7.4 Results

As we have already done for the previous networks, also for T-Tox we want to link the parameters with the network performances. This time, contrary to the other we also looked for the parameters that affect the validation accuracy. That is because, this time, opposite to the other optimization processes, we trained a feed-forward network that, as explained before it is faster to train. Just to give a comparable measure each iteration is about ten milliseconds, this means that an epoch is around 6 seconds. In this case, we tested the network from one to one hundred epochs. So each training is around 15 minutes on average. However, we tested 900 different combinations that translate in two computation days. An important change in this chapter is the introduction of a new parameter abbreviated with *MHL* that stands for *Max Hidden Layers* and indicate the number of hidden layers of the feed-forward network. As we have explained before, the number of hidden layers can tremendously increase the over-fitting phenomenon. For this reason, we wanted to test the architecture in order to define the minimum number of layers necessary to have good performances and the best compromise between this two factors.
The first data to extract from these analyses is the correlation map between the hyper-parameters used. It allows understanding the connections between the parameters and also the effects that the parameters have on the metrics used. From figure 7.5 where we have both the metrics used in the optimization process we can see how some of the parameters do not affect at all the results, accuracy in particular. At the same time, the validation loss is based on most of the proposed parameters. It indicates that the choice of the specified parameters is correct and, especially in such a big dataset of assays, that there is an actual correlation between them. Looking at the correlation map of the validation loss we can add a few more comments to better understand the results in the figures.

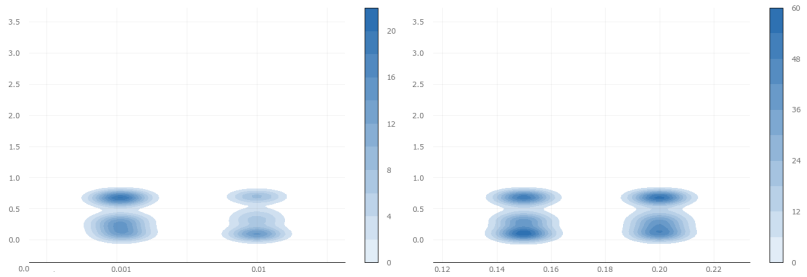
The *MHL* do not really influence the loss function. This is intuitive as the over-fitting problem mainly affect accuracy. However, as we inserted the dropout, the validation loss is compensated by the presence of this parameter, more than the accuracy. The batch size and the number of neurons for each layer are surprisingly high correlated. In fact, we expected to see the learning rate as highly linked parameters. In figure 7.6a we can find an explanation for this behavior. The learning rate is almost equally distributed between the four main possibilities. This is due to the different loss functions tested. The *xentropy* and the *MSE* have different units, and this affects the distribution of the aggregate results.

The epochs, in figure 7.7a, do not shows any particular trend. That is because epochs do not influence too much the loss function. We applied the Early stop method to prevent even more the overfitting. This, however, makes the distribution of the epochs too sparse to be meaningful for the loss function. In figure 7.13 we have a clear view of the sparse values collected. We can notice however that the two main clusters are concentrate near 0.6 and above 0. On the other hand, epochs are affecting



(a) Correlation map for validation accuracy. **(b)** Correlation map for validation loss.

Figure 7.5: Correlation maps for validation accuracy and validation loss function.



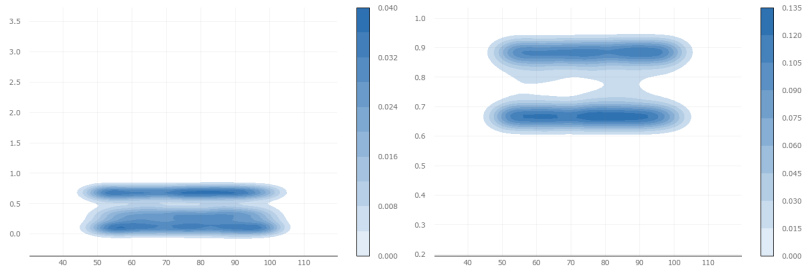
(a) Learning rate **(b)** Dropout

Figure 7.6: Learning rate and dropout distribution, on the x-axis, related to the value of validation loss function on axis y.

accuracy more than loss function. It gives us a new perspective. The loss function tends to converge faster than the accuracy. It implies that even with a lot of techniques to avoid overfitting if we increment too much the complexity of the network we do not obtain better results.

As for the epochs, also the MHL suggests us that increasing the parameters is not always the best choice. In table 7.1, the best result is obtained using only two hidden layers. However, figure 7.8 shows that, in a general configuration, the architecture with 11 hidden layers will perform better. In chapter 8 we validate our sentences "*The simplest is better*". For other applications, however, we cannot suggest using apriori the same configurations.

The last parameter that is highlighted by the correlation map is the batch size. Again here, the best value for this parameters would have been 64, as we can see from the graph, in figure 7.9, however, the concentration of the best accuracy is 32. This



(a) Validation loss function **(b)** Validation Accuracy

Figure 7.7: Epochs number distribution, on x-axis, related to the value of validation accuracy and loss function on axis y.

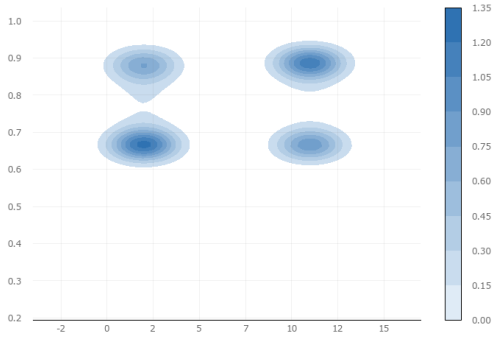


Figure 7.8: Hidden layers' number distribution, on the x-axis, related to the value of validation accuracy function on axis y.

is coherent with the literature which suggests better results with a low batch value. The graph, however, is very sparse. From table 7.1, where we reported the best values obtained, it is visible however that also 64 as batch size gives excellent results.

In order to summarize the results just explained we reported figure 7.10 and 7.11. These two histograms show the distribution of all the tested combinations. An interesting point of view about these results is the learning rate of 0.2 that gives great results only using 11 hidden layers. With a big network and a few epochs for training, a high learning rate is a possible solution to create a model. However, this increases the value of the loss function. That is the main reason because we do not consider this value while evaluating the table of the results.

In figure 7.12, we can see the correlation between loss function and accuracy. There are two detailed to remarks. First of all the best results are correctly concentrate on the bottom right corner, with some sparse and worse results in the neighborhood. The second point is that from this graph we can see the early stop technique in action. Comparing the optimization with and without early stop technique we can remark as the line of dots it is not present when we do not stop the training. That is because the stop can happen for two reasons: either the network has started overfitting, or, more probably, the combination of parameters tested is not correct. This last option

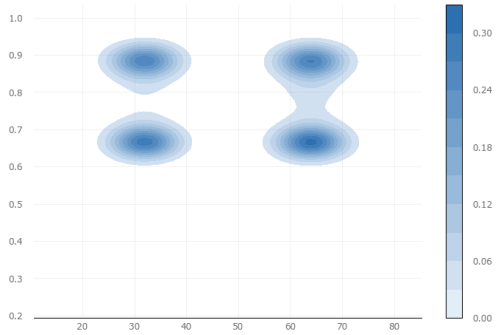


Figure 7.9: Batch size' distribution, on x-axis, related to the value of validation accuracy function on axis y.

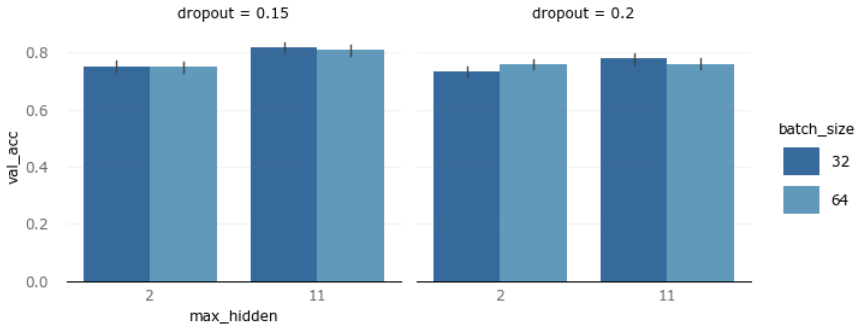


Figure 7.10: Histogram representing the relation between validation accuracy, on the y-axis, the learning of hidden layers, on the x-axis, the batch-size, in color, and the dropout, in the two graphs.

is the reason for the line on the left part of the points. We stop the network while it starts getting worse but most of the time that is because the mixed proposed it is not suitable for the problem.

Even if we discussed table 7.1 in chapter 8 it is worthy spend a few words for it. The table follows the structure of the other chapters. On the most left column, there is the name of the architectures used. This time, the name is composed of the batch size selected followed by the number of hidden layers present in the network.
An interesting point for table 7.1 is that, as for SMILES-Net, we can see that the MSE perform better than the cross-entropy loss function. Again this is due to the number of epochs. We tested the network on 300 hundred epochs to prove this hypothesis, and in fact, we achieve the 90% of accuracy and the 0.09% for the loss function. The other columns prove that it is not necessary to use high parameter values to achieve the best result. It is also coherent with the latest literature of computation toxicology.

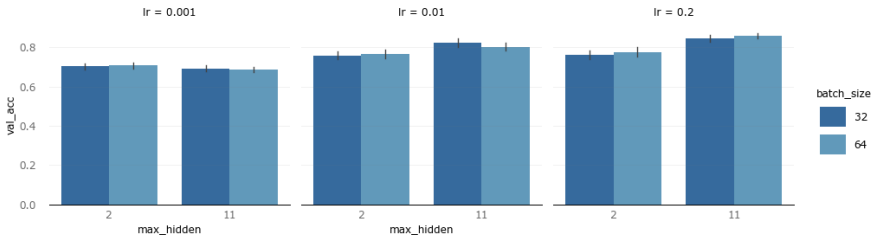


Figure 7.11: Histogram representing the relation between validation loss function, on the y-axis, the number of hidden layers, on the x-axis, the learning rate, in color, and the dropout, in the two graphs.

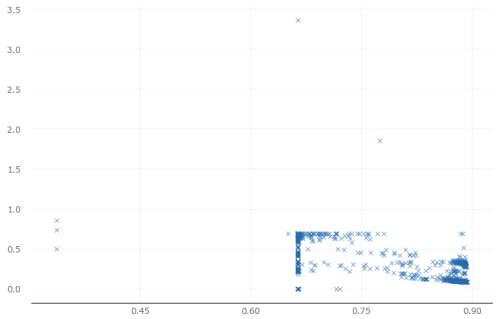


Figure 7.12: The distribution of the validation loss function based on the validation accuracy.

Architecture	Parameters							Metrics			
	Ep	Lr	Neurons	BS	Drop	Loss	MHL	Acc	Loss	Val Acc	Val Loss
C-Tox-32-2	60	0.01	300	32	0.15	*MSE*	2	0.893	0.081	0.891	0.083
	100	0.001	500	32	0.2	*MSE*	2	0.887	0.087	0.885	0.091
C-Tox-64-2	85	0.01	100	64	0.15	*MSE*	2	0.897	0.079	0.893	0.084
	90	0.001	500	64	0.15	*MSE*	2	0.889	0.090	0.875	0.092
C-Tox-32-11	80	0.01	500	32	0.2	*MSE*	11	0.893	0.083	0.891	0.085
	100	0.001	300	32	0.15	*MSE*	11	0.889	0.085	0.880	0.086
C-Tox-64-11	85	0.01	200	64	0.15	*MSE*	11	0.893	0.084	0.892	0.088
	100	0.001	500	64	0.15	*MSE*	11	0.886	0.086	0.884	0.089
C-Tox-32-2	70	0.01	100	32	0.15	*xentropy*	2	0.890	0.293	0.884	0.294
	80	0.001	10	32	0.15	*xentropy*	2	0.871	0.319	0.85	0.349
C-Tox-64-2	80	0.01	100	64	0.15	*xentropy*	2	0.889	0.281	0.886	0.295
	84	0.001	300	64	0.15	*xentropy*	2	0.782	0.689	0.689	0.692
C-Tox-32-11	90	0.01	500	32	0.2	*xentropy*	11	0.891	0.290	0.892	0.291
	70	0.001	500	32	0.15	*xentropy*	11	0.889	0.287	0.887	0.301
C-Tox-64-11	85	0.01	500	64	0.15	*xentropy*	11	0.895	0.278	0.891	0.288
	75	0.001	500	64	0.15	*xentropy*	11	0.887	0.301	0.881	0.305

Table 7.1: C-Tox Optimization results for regression and classification

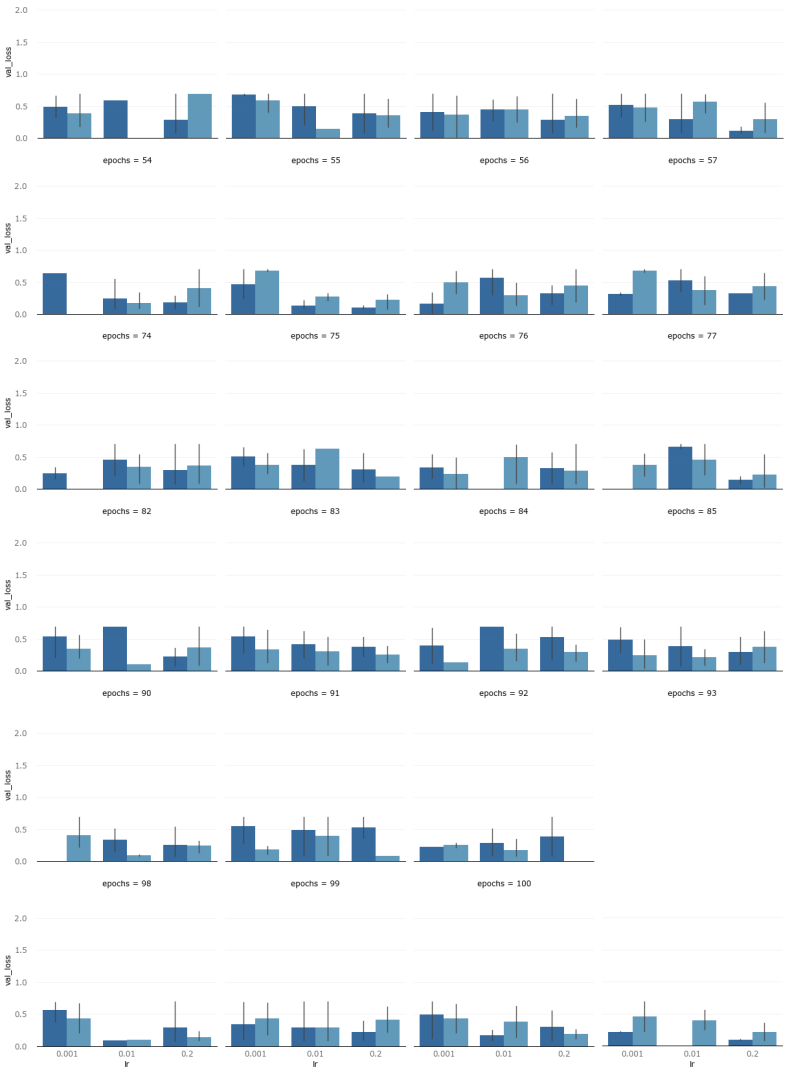


Figure 7.13: Histogram representing the relation between validation loss function, on the y-axis, the learning rate, on the x-axis, the batch-size, in the color, and the epochs, in the graphs.

Chapter 8

Discussion

We faced many different thematics during this work, from the data collection to the result of the model proposed passing through the knowledge extraction from the networks in the architecture. In this chapter, we are going to discuss all these topics adding some considerations and some concerns. Moreover, we will add some future work or some better implementation that can be added to the proposed work.

8.1 Knowledge

Presenting this model, we stated a significant and strong choice. To not add external knowledge to any part of the model. It is obvious that this strong choice has some advantages and some drawbacks. Both of them needs to be discussed and examined in depth. With this assumption, we also add the possibility to obtain and extract knowledge from the model. As we did not insert other information than the SMILES and the label, the knowledge is completely new, generated from the machine. It also needs a deeper analysis, and in this case, we need to evaluate the knowledge extracted from the architecture proposed. For clearness, we analyze each sub-model separately, and we compare then the whole results together.

8.1.1 Apriori knowledge

An important matter to analyze is the choice to not have *apriori knowledge* in the networks. This decision, as already said in the precedent chapters, was taken to completely abstract the model from error and uncertainty deriving from the human expertise applied to other models. Observing the results however we do not believe that the absence of this knowledge penalizes the proposed model. Comparing the architecture proposed by Goh et al. (2017) [31] we can see that the introduction of the chemical properties in the images passed to the network only adds a few hundredth in the final result. If we compare the Chemception to our final model, T-Tox, the latter overcomes the former with different points in accuracy, specificity, sensitivity,

and error. The same choice has also been applied to SMILES-Net and C-Tox.
Concerning SMILES-Net we could have inserted the dictionary generated from other sources in order to avoid a part of the interdependence between the data and the network results. Moreover, we could have allowed the network to use an external source, taken from the literature, containing the most used *Structural Alerts*, in order to focus its attention on specific sub-strings. However, it is clear how this process could be affected and influenced by external knowledge. We decided not to add this source inside the architecture, coherently with the absence of other knowledge, but we used external SAs to evaluate the results of our network in term of acquired knowledge. The case of the SAs is explanatory of how the external knowledge derived by the previous researches can influence the outcome of any architecture. The usage of structures calculated with common machine learning methods can completely reverse the outcome of the architecture that would not be able to learn, but only to replicate the already present knowledge.
The most controversial part of this choice is the decision to not introduce any knowledge also in the final *feed-forward*, *C-Tox*, that is in charge of classifying the input based on the features extracted by the precedent networks. It is undeniable that we could have used some knowledge in this last part by introducing some known features to sustain the already selected values. A part of the future work could be to compare the results of the two networks with different knowledge to validate our choice. However, the consideration we did until this part of the network still applicable to C-Tox. The introduction of new features also adds a tremendous bias. That is because we can not introduce all the features of the chemicals, this would be not effective and too costly in term of resources, and the selection done on the features to pass to the network creates a dependency between the result and the expertise applied by some external entity, different from the network itself.

8.1.2 Extracted knowledge

The main advantages of our choice to not use external knowledge is that we are sure that everything we can extract from the network is self-generated. It means that the knowledge we are able to extract from the network does not come from other sources, but it is derived only from the SMILES. This implies different things. First of all, it opens new possibilities for knowledge generation. In the literature, many methods use statistical analysis on a set of data to extract and generate SAs. The most famous are described by Benigni et al. (2007) [18], Ferrari et al. (2013) [27] just to cite some. The main advantage of our method is that using neural networks it is possible to non-linear modeling relationships. Therefore the knowledge extracted could better describe the complex model we want to analyze.
In order to extract the knowledge from our model, we used the Attention mechanism that is explained in detail in section 6.4. In particular, we did not apply this mechanism on C-Tox but only Toxception and SMILES-Net. That is because we wanted to study

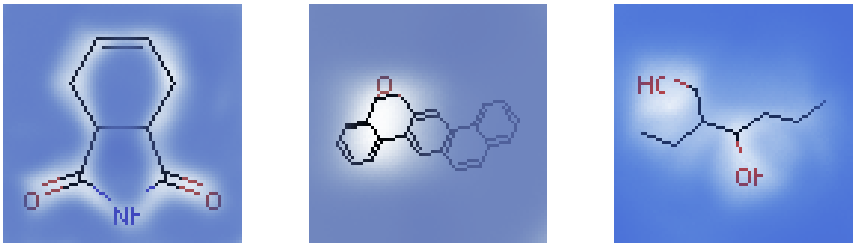


Figure 8.1: The visual attention results from Toxception

the different possibles representations that the network proposes and not the features selected by the model. The attention layer has been applied to both Toxception and SMILES-Net with the same methodology, however, the possibility of knowledge extraction is different in the two sub-networks.
We start with the extraction from Toxception because it is the one which has more problems to solve. In figure 8.1 we reported an example of visualization results. We can see from the images that the network identify a part of the chemicals as the important structure that helped to classify the compound. However, this visualization cannot be processed from an algorithm. The main problem deriving from these results is that we are not able to extract any detail of the image but just to add an overlay layer that visually indicates which part is taken. Of course, the layer gives a correlation matrix that is understandable from the following network but the problem comes when we try to translate this knowledge into another form, in order to compare it with the existing one. One possible solution could be to create a parallel graph, based on the molecule, analyze and apply the correlation maps given by the attention mechanism to this graph. In this way, we could have a description of the structural parts included. It is clear that this process is really complex and the realization could become really challenging for some molecules. So the realization of this system is left to future work on this model.

Everything changes for SMILES-Net as the inputs of the network are strings, we can see them as the structure of our input on which we apply the correlation matrix of the attention mechanism. It means that taking the input string, we are able to calculate the weight that each character has on the final prediction. This permits to asses the most important sub-strings used in the prediction of both, toxic and no toxic chemicals. It also means that we can construct a map, reported in figure 6.14, that allows a visual comprehension of the extracted results. The most important string during the prediction process are the *Structural alerts*. The strings obtained can be used to compare the quality of the prediction done comparing them with the known SAs. This is important also because could lead to the development of new SAs. An example of the string extracted from our model can be found in 8.1. An example of SARpy' is instead in 8.2.

$$O = C(c1ccccc1)C(O)(C)C \tag{8.1}$$

$$O = [N+]([O-])c1ccc(N = N)c(c1)[N+](= O)[O-] \tag{8.2}$$

The extraction method is explain in detail in section 6.5.3.In this section, we want to discuss the advantages of this method and the comparison with the current state of the art. The most important things to get from the correlation matrix of the attention mechanism is the value related to each character. In fact, each matrix can be interpreted as the correlation matrix of the specific string compared with the output. It means that we can get different SAs by imposing different threshold to the correlation values we want the searched string to have. Moreover, to compare the results with the literature, we ran almost 18 thousands compounds of our dataset into SARpy and we extracted the most important active and inactive SAs. These are the structural alerts that correspond to toxic compounds and the structural alerts that correspond to non-toxic compounds. In table 8.1 we reported the analysis for the different thresholds used. In particular, the occurrence is calculated by summing up the number of time the specific SA can be found in each row of the database. The percentage of true positive, true negative, false positive and false negative is calculated on the number of occurrences. The minimum length is imposed by the code used to do the computation as we discarded all the string with less than three characters. That is because we wanted to avoid a specific SA to indicate only an element without indicating any chemical structure. The results obtained are coherent with the results from chapter 6. Especially the false positive and the false negative are coherent with the values obtained for specificity and sensitivity. However, some consideration must be done on this table.

First of all, we can point out that, even if the percentage of specificity and sensitivity reflects the results obtained from SMILES-Net, we have the best results for the specificity using a threshold of 0.15, opposite for the sensitivity where the best result is instead the smallest values 0.01. This is because, as we said in section 4.3, our dataset is unbalanced, see table 4.2 for more details. It means that using smaller fragments as SA we generalize too much and this reflects on the results in an increase of the false negative. Using bigger string instead we specialize our model, that is able, but at the same time we also increase the percentage of error while treating negative chemicals. Moreover, we can see how the maximum length decrease with the increase of the thresholds. It is obvious that this happens because we add more constraints to the acceptable characters and it is also correlated with the SARpy occurrences results. In fact, as the length increase, the number of occurrences in the SARpy database start decreasing. This is because longer strings are more difficult to be contained in the external source of SAs.

Th	Characteristics		Metrics					
	Min Len	Max Len	TP (%)	TN (%)	FP (%)	FN (%)	Occ	SARpy Occ
0.1	3	9	28	35	24	13	324 190	9 823
0.15	3	6	25	38	19	18	206 375	5 475
0.01	3	96	28	31	30	11	938 083	24 894
0.02	3	49	26	31	27	16	1 317 264	29 559
0.05	3	19	26	30	29	15	1 075743	26 160

Table 8.1: SMILES-Net SAs extraction analysis

Th	: stands for threshold and it is the correlation value required to consider the character.
$MinLen$	: is the length of the smallest fragment found.
$MaxLen$	: is the length of the biggest fragment found.
TP	: indicates the percentage of the true positive occurrences.
TN	: indicates the percentage of the true negative occurrences.
FP	: indicates the percentage of the false positive occurrences.
FN	: indicates the percentage of the false negative occurrences.
Occ	: is the sum of the number of times the fragment is contained in a SMILES of the dataset.
$SARpyOcc$	: is the sum of the number of times a fragment is contained in the SAs of SARpy.

It is also interesting to denote the number of occurrences in the SARpy set. All the values are higher than five thousand. Yet the SAs returned by SARpy are only seven hundred. This enormous difference between the two units it is due to the fact that SARpy returns really long structures, our model instead gives short SAs. This difference allows the SAs from SMILES-Net to be found more than one for each SA of SARpy. This result is also valuable as it indicates that, even if in a smaller form, our model is still finding meaningful structures that can lead to the correct prediction of the input.

The results obtained suggest that the best threshold values to used in term of accuracy are 0.1 and 0.15. However, we proposed for our model a threshold of 0.01. That is because we still want to guarantee a good level of sensitivity. Moreover, with a deep analysis of the result, we found out that the 0.01 value is the one that allows finding the highest number of long strings, comparable to SARpy, and that the long fragments found can be found multiple times in SARpy. However, the proposed threshold is the one that requires the most post-processing work in order to have understandable and readable results. In fact, the original number of strings found was higher than the one reported here. Yet, we applied a cleaning process on the output of SMILES-Net in order to remove duplicates and very short string, only one or two characters.
We did not extract any knowledge from the C-Tox. That is because we do not have

any comparison mechanism with the current state of the art as the proposed model is the first of its type and we were not looking for new knowledge. However could be interested in future improvement to extract the weight matrix of C-Tox in order to analyze which features perform better than the others. It could lead to a remarkable improvement in term of model efficiency. With the development of a new model based on the one proposed in this work, it could also be interesting to compare which network selects the most interesting features.
Finally, to evaluate our model, T-Tox, we ran two different analysis: the first one based on the dataset created from the duplicates, described in section 4.3.3. The prediction on this database gives 0.889 of accuracy, 0.08 sensitivity and 0.12 of specificity. This excellent values derived from the fact that the model has actually been trained on these values, in a different form. This experiment gives us the possibility to prove our initial hypothesis: the model became sensible to the structure and not to the SMILES.
The second analysis is a verification of the knowledge of the other models taken from the literature, compared with T-Tox. In *A large comparison of integrated SAR/QSAR models of the Ames test for mutagenicity*, Benfenati et al. (2018) [16] took 10 QSAR/SAR models from the literature and predicted 18 thousand compounds using them. Two hundred of them resulted in either false positive or false negative for all the ten models. We predict these 200 compounds with our model and we obtain an accuracy of 0.62 and 0.1 FP. The accuracy value is not high. The causes can be various. First of all, some of the tested chemicals may be outside the domain applicability of our architecture. Moreover, we did not pass the 200 molecules through the preprocessing described in section 4.3.3. Besides, it must be considered that all the other model predicted them wrong. It could mean that the AMES tests must be repeated on these compound in order to have an assurance of the conditions and the validity of the tests. However, the values of false negative are shallow. This is coherent with the result explained before and indicates that the precautionary choices made had the desired effects.

8.2 Unsupervised learning over supervised learning

Unsupervised learning is defined as the capability of the algorithm to learn without any indication of the values to learn. In our specific case could be interpreted in two ways. Either the capability of the model to learn without passing it the label that classifies the input or it can be seen as the capability of the architecture to propose the right features without passing it the desired solution. The second interpretation just described could be misleading. It would seem in fact that our proposal it is actually autonomously learning which features describe better the input. However, this is not the case. As explained in chapter 7 we used a peculiar technique called *Transfer learning* with which we are able to train the sub-networks just once without losing performances. It is important to clarify that the training of the two sub-model, even

if it is done separately and with others goals, it is not unsupervised. We used a simple workaround in order to obtain the weights we needed in the final model. In order to do that, we trained the two network, Toxception, and SMILES-Net separately, as two classifiers. Then in the final model, we use the same weights of the classification, but instead of using the same layers we removed the final dense layer. This remotion allows both networks not to classify but to furnish the features necessary for the final classification. Another way of seeing this is that we replace the final dense classification of the two layers with a more sophisticated feed-forward network. The concept does not change and all the learning still receiving the input and the respective label related to it.
With this explanation, it is now clear that we cannot talk of unsupervised learning as all of the sub-networks proposed are trained with the same labels.

8.3 The final model

In section 5.5, 6.7 and 7.4 we discuss the main results obtained training each model separately. In particular, we optimized each sub-network using different metrics based on the main idea that the best model in each training gives the best result combined with the other. However, we did not list the choice we made to finalize the architecture. That is because these choices need to be discussed in details. Also, we did not mention another aim of this work that is the distribution of the architecture proposed.

8.3.1 Model proposal

The choices of the parameters used for every sub-models are based on two main concepts: *Simplicity*, we wanted to maintain the architecture as simple as possible in order to reduce the computation resources consumed. *Performances*, while reducing the resources we also wanted to have the best performances we could achieve. The choices listed below are then a compromise of this two factors.
Concerning Toxception we propose the basic network which does not contain the stem block and is optimized using *Adam* algorithm. The loss function which performs better is the *cross-entropy*. The other training parameters adopted are reported in table 8.2. This choice was mostly driven by the necessity to have good values of sensitivity while maintaining the other values, validation loss, and specificity, in a good interval. These parameters go against the results reported by Goh et al. (2017) [31] but still aligned with the most common techniques used in the literature. The choice of do not include the stem block in the network, as proposed by Szegedy et al. (2017) [79] is due to the simple concept we explained at the beginning of this section. This simplification allows the network to be faster and to increase the performances as we removed the screening input part.

The choice of the parameters for SMILES-Net instead has been influenced by our resources, that is because we could not test a deeper architecture and this has limited

Architecture	Parameters		Metrics					
			Acc	Loss	Val Acc	Val Loss	Spec	Sens
Tox_basic_A	Epochs	*200*	0.988	0.021	0.801	0.171	0.624	0.875
	Optimizer	*Adam*						
	Neurons	16						
	Batch size	32						
	Dropout	0.2						
	Lr	0.001						

Table 8.2: The solution proposed for the Toxception sub-model.

Architecture	Parameters		Metrics					
			Acc	Loss	Val Acc	Val Loss	Spec	Sens
LSTM_Sum	Epochs	*100*	0.962	0.10	0.830	0.125	0.741	0.821
	Optimizer	*Adam*						
	Neurons	400						
	Batch size	32						
	Dropout	0.2						
	Lr	0.001						

Table 8.3: The solution proposed for the SMILES-Net sub-model.

the possibilities between which we could choose. However the best option this time was obtained using the MSE function, the Adam optimizer and one hundred as training epochs. The other parameters are reported in table 8.3. The most critical parameter for this choice was the time. However, we did not choose the best option in term of resources usage. In fact, the cell proposed is an LSTM with sum. As we explained in chapter 6 the more efficient would have been the GRU cell. Therefore it is intuitive that we did not take only the resources into account. In fact, we also included the specificity and sensitivity metrics into the reasoning process. The one proposed is the solution which has the best absolute specificity and sensitivity between all the combinations tested. An interesting point is the choices of the loss function. In the literature, the most used is usually the cross-entropy loss function. Yet, our results suggest that the best choice for our model is the MSE. This can be attributed to various causes; the most important are also discussed in section 6.7. The first cause can be the number of epochs. As we already said, the limited number of training epochs could, in fact, also affect this specific metric. Another probable cause is the peculiarity of the input proposed. It means that the LSTM has never been used in the literature to analyze this type of text, it could be that the function to approximate is better reachable using the Mean Square Error function. The others values proposed still aligned with the literature except for the batch size that is linked entirely to the GPU memory and its values in our case could vary between 16 and 128.

Finally, the solution chose for C-Tox was based entirely on the performances as the

Architecture	Parameters		Metrics					
			Acc	Loss	Val Acc	Val Loss	Spec	Sens
C-Tox-64-2	Epochs	*85*	0.897	0.079	0.893	0.084	0.741	0.821
	Optimizer	*Adam*						
	Neurons	100						
	Batch size	64						
	Dropout	0.15						
	Lr	0.01						
	Hidden layers	2						

Table 8.4: The solution proposed for the C-Tox sub-model.

resources taken are less compared to the other sub-models. We also wanted, in this case, to keep the network as simple as possible, this time the main reason was that create and an over-parametric network can lead to random prediction and inconclusive results. This sentence is also proven by the optimization process realized on C-Tox that is explained in section 7.4. In fact, also in our proposal, we take the smallest number of hidden layer tested: two. This is understandable as C-Tox contains already three other hidden layers that are fixed and allow to scale the number of neurons for each layer gradually. The optimizer still Adam also in this sub-model and the loss function adopted is the MSE. Actually, the choice of the MSE, in this case, can be argued and a good improvement to this work could be to test the same parameters using the cross-entropy instead. As already reported in chapter 7 the main reason because the values are not comparable should be that the network did not train enough to reach the optimum. Differently from the other, in C-Tox we proposed a batch size of 64 in order to speed up the process even more. Also, the dropout is less than the other propositions. That is because a dropout follows each dense layer and the usage for each of them of high values could probably lead to a deterioration of the performances. The summit of all the parameters proposed is reported in table 8.4.

8.3.2 Model distribution

One of the goals of this thesis was to develop a new model and to realize an easy way to make it available for the community in order to do that we split this work into two different parts. The first one is the architecture proposed and the second is the trained model on the architecture. Unfortunately, not all the data present in the database are public and we have not the right to distribute them. However, we still can publish the model created on the dataset as it is impossible to derive the origin from it.
With the architecture proposed we create a package in *Python* that allow everyone to use the model, and each sub-model, on its own dataset. The code is structured in such a way that it is possible to change and manipulate almost all the parameters of the architecture from the learning rate to the number of layers in each sub-model. The package also implements the optimization process and its graphics manipulation

allowing the users to use it in all its flexibility. Moreover from the package is possible to use each sub-model in a stand-alone version and to integrate it with other sub-modules. The knowledge extraction is present in the code. We did not add the code to analyze the knowledge extracted as it has been realized with a different programming language. The package limits the structure of the data that is possible to insert to the SMILES string and its label. That is because we wanted to guarantee a usage coherent with the main principle of this work. We chose to insert as requirements for the package: *Tensorflow, Keras, RDKit, numpy* and some other minor packages that are fundamental for the usage of the code proposed. It is not possible however to change the Keras backend. That is because the code written it is not compatible with other, as Theano or PyTorch. This choice is really constraining, but we decide anyway to do it as the packages chosen are the most commonly used and the most updated. They are also really flexible and can be used in almost all the problems. From the package, it is also possible to use the proposed model that it is also accessible through the API.

The API allows a controlled access interface in order to interrogate the model and to make predictions about specific compounds provided by the users. The API also furnishes the respective visual knowledge about the input introduced. The input for the API is limited to a maximum of one thousand compounds per request; this is done to avoid server crash and to prevent the API to be attacked using overflow hacking techniques.

The choice of distinguishing the technologies used to access the model and to the package is based on the resources management. We wanted the user to use the model independently from his/her resources, and at the same time, we want the user to be able to train the architecture on its own hardware that could be more efficient than ours.

Chapter 9

Conclusions

The goal of our research was to use the new cutting-edge researches on machine learning and deep learning and apply them to the field of biology and toxicity. The aim was to discover the link between the molecular form of a chemical and its biological effects. To do that we adopted deep Learning methods, modifying the most successful architectures, Inception and *LSTM*, combining them together to build a model called *T-Tox*.

In Chapter 2 we deeply discussed basic definitions for the toxicology domain, and we discussed the chemicals classification. We gave particular importance to the tests used nowadays. Explaining the possible variants of *In Vivo* and *In Vitro*. In addition, we analyzed in details the *AMES* test [12]. The goal of this chapter was to introduce the reader to the toxicology domain. We wanted to give an overview of the main challenges of toxicology as well as the possibilities and the improvements that this domain can be subject to.

In Chapter 3 we shown the current state of the art, explaining in detail the recent methods based on *Neural Networks* and *QSAR/SAR* [18]. In addition, we discussed the reasons for the interests in toxicology and computation toxicology, also describing the main users. We gave evidence of the existing work and its possible implication. Moreover, we added an overview of the current available software, as *SARpy* [27] and *VEGA* [10]. In this part, it is visible the complete, or almost complete, absence of deep learning techniques in the state of the art to the best of our knowledge. We also reported some results comparison proposed by Benfenati et al. (2018) [16], that can be used to evaluate our proposal.

In Chapter 4,5, 6 we framed our project. We explained the main challenges we encountered, and we showed how we designed our model. This thesis aimed to produce a flexible model that can be used on real toxicity tests. We also wanted to test our

proposal on a useful real-case. We regrouped and proposed a database of almost 22 thousand chemicals with their AMES test results. This dataset was dirty and composed of many different sources. In order to help the job of the model, we designed a pre-processing pipeline to standardize and normalize the structures. The result of this pipeline is our final training set. We started from data exploration, where we understood the properties of the datasets in the literature. Then, we designed a procedure for SMILES level cleaning using *VEGA*. We empowered the pre-processing pipeline with an interesting block: a function linked to *PubChem* able to get information from specific chemicals. We performed this cyclic obtaining our database proposal. The dataset creation is a fundamental phase as we aim to not use apriori knowledge except for the SMILES strings.
We wanted to propose an innovative architecture and we split the project into three parts, (Toxception, SMILES-Net and C-Tox) combining them at the end. For Toxception, in chapter 5, we studied the most famous *Image classificator* and we chose Inception by Szegedy et al. (2017) [79] as base for our model. We reduced the network, and we introduced an attention layer. The final network is a CNN based with 140 layers and over 1 thousand parameters. The introduction of the attention layer allows the analysis of the important part of the images and highlights it in the output. The performances of this model are around 80% of accuracy which is almost the current state of the art results.
SMILES-Net, in chapter 6, is based on *Text classification*, we studied the different classification techniques, LSTM, GRU and bidirectional. We deeply analyzed the SMILES structure and we proposed a new SMILES2Vect technique that helps the model understand the strings. We inserted the attention layer at the end of the network again. One of the aims of this thesis was to extract new knowledge generated by our model. SMILES-Net is the biggest source of knowledge for our proposal. The attention layer returns the contribution each character gives to the prediction. This allows to create *Structural Alert* that compared with the existing knowledge base gives comparable results.
The final sub-model proposed is C-Tox, in chapter 7 which performs the classification on the features extracted by Toxception and SMILES-Net. We proposed an interesting solution for combining all the models into one, saving a lot of computational time. We loaded the weights of the models trained before into the final architecture T-Tox. This saves the training time of the whole set of parameters.

In Chapter 8 we analyzed our results and we discussed the main advantages and drawbacks of our proposal. The final model give outstanding results compared with the current deep learning technique *Chemception* [31]. The results are also comparable with the QSAR models present in the literature, the main advantage of our proposal is that there is no apriori knowledge and the dataset do not require human effort and expertise to be created. Moreover, it must be considered that the literature methods have a small applicability domain and, especially the SAs models based, do not classify

the molecules if it does not contain any known structural alert. It means an increase in accuracy, but it implies a high number of discarded input. Our proposal, on the other hand, is based on mathematical analysis and do not discard any input.
The results reported in this work are however comparable with the literature models and our proposal has special advantages that are not present in any other architecture.

9.1 Future Work

The aim of this thesis was also to propose a new research branch in order to go deeply into the possibilities offered by this new innovative architecture. In the final chapter, 8, we discussed the main drawbacks and the blind spots of our model. Every part of our architecture can be improved. We propose three main streams of future work. One for each sub-model.
Concerning Toxception it could be interesting to analyze different image size. This could lead to a better prediction and a better knowledge extraction. We also proposed to insert the graph analysis proposed in chapter 8. It will improve the evaluation of the generated knowledge.
For SMILES-Net we proposed to analyze the Structural Alert extracted, explained in section 8.1.2. We just compared them with the existing knowledge. However, it would be interesting to evaluate the accuracy of the new SA by analyzing other datasets.
In C-Tox we propose to make a comparison between the features selected from Toxception and SMILES-Net and the most common features. It could lead to new discoveries concerning the feature selection also in other more traditional methods. Besides, for this part, could be interesting to analyze the weights of each sub-model, Toxception and SMILES-Net, in order to study which models give the best features. Finally, we also proposed to improve T-Tox. The most interesting proposition would be to add a classification to a new endpoint. Actually, the code proposed already contains the structure to predict other endpoints. However, this new classification needs the search and the construction of a whole new database.

Bibliography

[1] Chemical carcinogenesis research information system (ccris). URL https://bit.ly/2Su2zuw.

[2] Gene-tox. URL https://bit.ly/2zeXGN4.

[3] A brief history of toxicology. URL https://aibolita.com/sundries/22157-a-brief-history-of-toxicology.html.

[4] Imagenet. URL https://bit.ly/2OiQYLD.

[5] Keras. URL https://bit.ly/2q0PpZ7.

[6] Pubchem. URL https://bit.ly/1t5jyUe.

[7] Rdkit. URL https://bit.ly/2OYLjj9.

[8] Talos. URL https://bit.ly/2yL9gQJ.

[9] Tensorflow. URL https://bit.ly/1MWEhkH.

[10] Vega hub. URL https://bit.ly/2EQaD64.

[11] Introduction to sequence models—rnn, bidirectional rnn, lstm, gru, 2015. URL https://bit.ly/2FoKGev.

[12] Bruce Ames, Frank D. Lee, and William E. Durston. An improved bacterial test system for the detection and classification of mutagens and carcinogens,. *Proc Natl Acad Sci USA*, 1973.

[13] Apple. Siri. URL https://apple.co/1FvsbZq.

[14] AWS. Amazon chatbots. URL https://amzn.to/2fREaOp.

[15] Igor Baskin. Machine learning methods in computational toxicology. *Computation toxicology*, 2018.

[16] Emilio Benfenati, Azadi Golbamaki, Giuseppa Raitano, Alessandra Roncaglioni, Serena Manganelli, Frank Lemke, Ulf Norinder, Elena Lo Piparo, Masamitsu Honma, Alberto Manganaro, and Giuseppina Gini. A large comparison of integrated sar/qsar models of the ames test for mutagenicity. *SAR and QSAR in Environmental Research*, 2018.

[17] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 1994.

[18] Romualdo Benigni, Tatiana Netzeva, Emilio Benfenati, and Cecilia Bossa et all. The expanding role of predictive toxicology: An update on the (q)sar models for mutagens and carcinogens. *Journal of Environmental Science and Health, Part C*, 25(1):53–97, 2007. doi: 10.1080/10590500701201828. URL `https://doi.org/10.1080/10590500701201828`.

[19] Antonio Cassano, Alberto Manganaro, and Emilio Benfenati. Caesar models for developmental toxicity. *Chem Cent J.*, 2010.

[20] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[21] KyungHyun Cho, Aaron C. Courville, and Yoshua Bengio. Describing multimedia content using attention-based encoder-decoder networks. *CoRR*, abs/1507.01053, 2015. URL `http://arxiv.org/abs/1507.01053`.

[22] Raffaella Corvi, Madia, and Federica. Eurl ecvam genotoxicity and carcinogenicity consolidated database of ames positive chemicals. *European Commission, Joint Research Centre (JRC)*, 2018.

[23] Misha Denil, Loris Bazzani, Hugo Larochelle, and Nando de Freitas. Learning where to attend with deep architectures for image tracking. *CoRR*, abs/1109.3737, 2011. URL `http://arxiv.org/abs/1109.3737`.

[24] EPA. *Chemical Hazard classification and labeling: comparation of opp requirements and the GHS*. 2003.

[25] U.S. EPA. *User's Guide for T.E.S.T.(Toxicity Estimation Software Tool): A Program to Estimate Toxicity from Molecular Structure.* 2016.

[26] Jun Feng, Laura Lurati, Haojun Ouyang, Tracy Robinson, Yuanyuan Wang, Shenglan Yuan, and S. Stanley Young. Predictive toxicology: benchmarking molecular descriptors and statistical methods. *Journal of Chemical Information and Computer Sciences*, 43(5):1463–1470, 2003. doi: 10.1021/ci034032s. URL `https://doi.org/10.1021/ci034032s`. PMID: 14502479.

[27] T. Ferrari, D. Cattaneo, G. Gini, N. Golbamaki Bakhtyari, A. Manganaro, and E. Benfenati. Automatic knowledge extraction from chemical structures: the case of mutagenicity prediction. *SAR and QSAR in Environmental Research*, 2013.

[28] Thomas Ferrari and Giuseppina Gini. An open source multistep model to predict mutagenicity from statistical analysis and relevant structural alerts. *Chemistry Central Journal*, 4(1):S2, Jul 2010. ISSN 1752-153X. doi: 10.1186/1752-153X-4-S1-S2. URL `https://doi.org/10.1186/1752-153X-4-S1-S2`.

[29] P. Foggia, A. Limongiello, F. Tufano, and M. Vento. Learning graphs from examples: An application to the prediction of the toxicity of chemical compounds. *Pattern Recognition Artificial Intelligence*, 2006.

[30] G. B. Goh, N. O. Hodas, C. Siegel, and A. Vishnu. SMILES2Vec: An Interpretable General-Purpose Deep Neural Network for Predicting Chemical Properties. *ArXiv e-prints*, December 2017.

[31] Garrett Goh, Charles Siegel, Abhinav Vishnu, Nathan O. Hodas, and Nathan Baker. Chemception: A deep neural network with minimal chemistry knowledge matches the performance of expert-developed qsar/qspr models. 2017.

[32] Ping Gong, Sundar Thangapandian, Gabriel Idakwo, Nan Wang, and Chaoyang Zhang. Quantitative target-specific toxicity prediction modelling. 2018.

[33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[34] Google. Google duplex. URL `https://assistant.google.com/`.

[35] Katja Hansen, Sebastian Mika, Timon Schroeter, Andreas Sutter, Antonius ter Laak, Thomas Steger-Hartmann, Nikolaus Heinrich, and Klaus-Robert Müller. Benchmark data set for in silico prediction of ames mutagenicity. *Journal of Chemical Information and Modeling*, 49(9):2077–2081, 2009. doi: 10.1021/ci900161g. URL `https://doi.org/10.1021/ci900161g`. PMID: 19702240.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[37] Christoph Helma, Tobias Cramer, Stefan Kramer, and Luc De Raedt. Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. *Journal of Chemical Information and Computer Sciences*, 44(4): 1402–1411, 2004. doi: 10.1021/ci034254q. URL `https://doi.org/10.1021/ci034254q`. PMID: 15272848.

[38] Sepp Hochreiter and Fakultat informatik. Long short-term memory. 1997.

[39] Michal Hrala and Pavel Král. Evaluation of the document classification approaches. In Robert Burduk, Konrad Jackowski, Marek Kurzynski, Michał Wozniak, and Andrzej Zolnierek, editors, *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*, pages 877–885, Heidelberg, 2013. Springer International Publishing. ISBN 978-3-319-00969-8.

[40] J. Judson, Richard A., and Dix DJ. The toxicity data landscape for environmental chemicals. *Environmental Health Perspectives*, 2009.

[41] Philip N. Judson, Paul A. Cooke, Nancy G. Doerrer, Nigel Greene, Robert P. Hanzlik, Christopher Hardy, Andreas Hartmann, David Hinchliffe, Julie Holder, Lutz Müller, Thomas Steger-Hartmann, Andreas Rothfuss, Mark Smith, Karluss Thomas, Jonathan D. Vessey, and Errol Zeiger. Towards the creation of an international toxicology information centre. *Toxicology*, 213(1):117 – 128, 2005. ISSN 0300-483X. doi: https://doi.org/10.1016/j.tox.2005.05.014. URL `http://www.sciencedirect.com/science/article/pii/S0300483X05002404`.

[42] Chung Junyoung, Caglar Gulcehre, and Kyunghyun Cho. Gated feedback recurrent neural networks. 2015.

[43] A. Kamber, Fluckiger-Isler S., Engelhardt G., Jaeckh R., and Zeiger E. Comparison of the ames ii and traditional ames test responses with respect to mutagenicity, strain specificities, need for metabolism and correlation with rodent carcinogenicity. *Mutagenesis*, 2009.

[44] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48 (1):312–320, 2005. doi: 10.1021/jm040835a. URL `https://doi.org/10.1021/jm040835a`. PMID: 15634026.

[45] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

[46] I. Kirkland, Aardema M., Henderson L., and Muller L. Evaluation of the ability of a battery of three in vitro genotoxicity tests to discriminate rodent carcinogens and non-carcinogens. i. sensitivity, specificity and relative predictivity. *Mutat Res.*, 2005.

[47] Daniel Krewski, Daniel Acosta Jr., and Melvin Andersen. Toxicity testing in the 21st century: a vision and a strategy. 2010.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. 2012.

[49] Hugo Larochelle and Geoffrey E Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In J. D. Lafferty, C. K. I.

Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1243–1251. Curran Associates, Inc., 2010. URL http://papers.nips.cc/paper/4089-learning-to-combine-foveal-glimpses-with-a-third-order-boltzmann-machine.pdf.

[50] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541. URL https://doi.org/10.1162/neco.1989.1.4.541.

[51] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1995.

[52] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[53] Zachary Chase Lipton. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015. URL http://arxiv.org/abs/1506.00019.

[54] O. Llorens, J. Perez, and HO. Villar. Investigation of structural and electronic biases in mutagenic compounds. 2002.

[55] J. McCulloch, S. Warren, and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4): 115–133, Dec 1943. ISSN 1522-9602. doi: 10.1007/BF02478259. URL https://doi.org/10.1007/BF02478259.

[56] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL http://arxiv.org/abs/1301.3781.

[57] Florian Nigsch, NJ Maximilan Macaluso, John BO Mitchell, and Donatas Zmuidinavicius. Computational toxicology: an overview of the sources of data and of modelling methods. *Expert Opin. Drug Metab. Toxicol.*, 2009.

[58] NIHS. Ames/qsar international collaborative study. URL https://bit.ly/2z7Rg2g.

[59] Noel O'Boyle. Generating multiple smiles, 2018. URL https://baoilleach.blogspot.com/2018/04/generating-multiple-smiles.html?m=1.

[60] OECD/OCDE. Guideline for testing of chemicals. 2001.

[61] Council of the European Union. Directive on the approximation of laws, regulations and administrative provisions relating to the classification, packaging and labelling of dangerous substances. 1992.

[62] Earnest Oghenesuvwe Erhirhie, Chibueze Peter Ihekwereme, and Emmanuel Emeka Ilodigwe. Advances in acute toxicity testing: strengths, weaknesses and regulatory acceptance. 2018.

[63] Christopher Olah. Understanding lstm networks, 2015. URL `https://bit.ly/1S6gmjZ`.

[64] Mathieu Orfila. *Traité des poisons (Toxicologie générale)*. 1813.

[65] OSHA. *Hazard Classification Guidance*. 2016.

[66] Keiron Teilo O'Shea. An introduction to convolutional neural networks. 2015.

[67] W. Piegorsch and E. Zeiger. Measuring intra-assay agreement for the ames salmonella assay,. *Statistical Methods in Toxicology*, 1991.

[68] V. Le Quoc and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. URL `http://arxiv.org/abs/1405.4053`.

[69] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958.

[70] William Moy Stratton Russell, Rex Leonard Burch, and Charles Westley Hume. The principles of humane experimental technique. 1959.

[71] I. Rusyn and GP Daston. Computational toxicology: Realizing the promise of the toxicity testing in the 21st century. *Environmental Health Perspectives*, 2010.

[72] M. Von Korff Sander. Toxicity-indicating structural patterns. *Chemical Informatique Models*, 2006.

[73] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997.

[74] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2007.

[75] Nusrat K. Shaikh and Maheshwari D.G. An overview on- toxicity testing methods. 8:3834–3849, 06 2016.

[76] B. Simon-Hettich, A. Rothfuss, and T. Steger-Hartmann. Use of computer-assisted prediction of toxic effects of chemical substances. *Toxicology*, 2006.

[77] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[78] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[79] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. 2017.

[80] Agency U.S. Environmental Protection. Toxrefdb: Toxicity reference database. 2010.

[81] VN. Vapnik and A.Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 1963.

[82] M. Weininger, Weininger A., and Weininger JL. Smiles. algorithm for generation of unique smiles notation. *Journal of Chemical Information and Modeling.*, 1989.

[83] Paul Werbos. *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting.* Buch, 1976.

[84] S. Whitebread, J. Hamon, D. Bojanic, and L. Urban. Keynote review: in vitro safety pharmacology profiling: an essential tool for successful drug development. *Drug Discovery Today*, 2005.

[85] M. George Whitesides and Rustem F. Ismagilov. Complexity in chemistry. *Science*, 1999.

[86] Hongbin Yang, Lixia Sun, Weihua Li, Guixia Liu, and Yun Tang. In silico prediction of chemical toxicity for drug design using machine learning methods and structural alerts. *Frontiers in Chemistry*, 2018.

[87] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. A C-LSTM neural network for text classification. *CoRR*, abs/1511.08630, 2015. URL `http://arxiv.org/abs/1511.08630`.

[88] Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. Bilingual word embeddings for phrase-based machine translation. 2013.

List of Figures

List of Tables