

Faculty of Sciences and Technology  
Department of Informatics Engineering

# RECURRENT MODELS FOR DRUG GENERATION

Angélica Santos Carvalho

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering, advised by Prof. Joel P. Arrais and Prof. Bernardete Ribeiro and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

July 2019



UNIVERSIDADE D  
COIMBRA



## **Abstract**

Drug discovery aims to identify potential new medicines through a multidisciplinary process, including several scientific areas, such as biology, chemistry and pharmacology. Nowadays, multiple strategies and methodologies have been developed to discover, test and optimise new drugs. However, there is a long process from target identification to an optimal marketable molecule. The main purpose of this dissertation is to develop computational models able to propose new drug compounds. In order to achieve this goal, the artificial neural networks explored and trained to generate new drugs in the form of Simplified Molecular-Input Line-Entry System (SMILES). The explored neural networks model were Recurrent Neural Network (RNN), Long-Short Term Memory (LSTM), Gated Recurrent Unit (GRU) and Bidirectional Long-Short Term Memory (BLSTM). A consistent dataset was chosen, and the generated SMILES by the model were syntactically and biochemically validated. In order to restrict the generation of SMILES, a technique denominated Fragmentation Growing Procedure was used, where made it possible to choose a fragment and generate SMILES from that. To analyse the recurrent network that fits the best and the respective parameters, some tests were performed, and the network contained in the model that reached the best result, 98% of valid SMILES and 93% of unique SMILES, was an LSTM with two layers. The technique to restrict the generation was used in the best model and reached 99% of valid SMILES and 79% of unique SMILES.

## **Keywords**

Drug Discovery, Deep Learning, Recurrent Models, Validation, Fragmentation Growing Procedure



## Resumo

A descoberta de medicamentos visa identificar potenciais novos medicamentos através de um processo multidisciplinar, incluindo várias áreas científicas, como a biologia, a química e a farmacologia. Atualmente, múltiplas estratégias e metodologias têm sido desenvolvidas para descobrir, testar e otimizar novos medicamentos. No entanto, há um longo processo que vai desde a identificação de alvos até uma molécula comercializável. O objetivo principal desta dissertação é desenvolver um modelo computacional capaz de propor novos compostos. Para atingir este objetivo, foi explorado e treinado um modelo recorrente para gerar um novo *Simplified molecular-input line-entry system* (SMILES). As *Artificial Neural Network* (ANN) estudadas nesta dissertação foram *Recurrent Neural Network* (RNN), *Long-Short Term Memory* (LSTM), *Gated Recurrent Unit* (GRU) e *Bidirectional Long-Short Term Memory* (BLSTM). Um conjunto de dados consistente foi escolhido e os SMILES gerados pelo modelo foram sintática e bioquimicamente validados. Para restringir a geração de SMILES, foi utilizada uma técnica denominada *Fragmentation Growing Procedure*, onde é possível escolher um fragmento e gerar SMILES a partir dele. Para analisar a rede recorrente que melhor se ajusta e os respectivos parâmetros, foram realizados alguns testes e a rede contida no modelo que atingiu o melhor resultado, 98% SMILES válidos e 93% SMILES únicos, foi uma LSTM com 2 camadas. A técnica de restrição de geração foi utilizada no melhor modelo e atingiu 99% dos SMILES válidos e 79% dos SMILES únicos.

## Palavras-chave

*Drug Discovery, Deep Learning, Modelos Recurrentes, Validação, Fragmentation Growing Procedure*



# Acknowledgement

First, I would like to thank my thesis supervisors, Prof. Joel P. Arrais, and Prof. Bernardete Ribeiro, for accepting me in this project, for the availability and all the advices during the execution of this dissertation. I would also like to thank the laboratory group, LARN, that received me always with a smile and some advice.

My sincere thanks to the University of Coimbra, the University of Aveiro and BSIM Square for the opportunity of being part of this project, and allowed me to learn from it.

I am profoundly grateful to the University of Coimbra and its people that make it possible for me to reach this point and taught me a lot about informatics and life.

I would also like to acknowledge my jury, Prof. Pedro Abreu and Prof. João Barata for the time and the constructive criticism and the people that read my thesis and their valuable comments.

My research would have been impossible without the aid and support of my close friends that were present on my academic journey and allowed me to complete this step of my life with a smile and some joy in between. A special thank to the people that were in my intermediate defence, took notes that simplified my progress, and made me company during the long nights of work and in the nights of relaxing.

Finally, the most important thanks are for my family, but particularly for my parents and sister that always were there to support me and make me believe that I could be and do whatever I want, with success. Furthermore, I would like to thank again to my sister and highlight the exhaustive corrections, the right words in right moment and all the conversations, since without it I could not deliever this dissertation, specially with this quality.





# Contents

<b>Acronyms</b> .....	<b>ix</b>
<b>List of Figures</b> .....	<b>xi</b>
<b>List of Tables</b> .....	<b>xv</b>
<b>Chapter 1 Introduction</b> .....	<b>1</b>
1.1 Context .....	1
1.2 Motivation .....	2
1.3 Goals .....	3
1.4 Document Structure .....	4
<b>Chapter 2 State of the Art</b> .....	<b>5</b>
2.1 From Artificial Intelligence to Deep Learning .....	6
2.1.1 Applications .....	9
2.2 Drug Discovery .....	9
2.2.1 Steps of Drug Discovery .....	10
2.2.2 Computational Drug discovery .....	11
2.2.3 Related Works .....	15
<b>Chapter 3 Methods</b> .....	<b>19</b>
3.1 Dataset .....	20
3.2 Tools .....	23
3.3 Recurrent Neural Networks Architectures .....	23
3.3.1 Simple Recurrent Neural Networks (RNN) .....	24
3.3.2 Long-Short Term Memory (LSTM) .....	25
3.3.3 Gated Recurrent Unit (GRU) .....	28
3.3.4 Bidirectional Long-Short Term Memory (BLSTM) .....	29

3.4	Parameters of Recurrent Architectures .....	30
3.5	Methodology.....	32
3.6	Validation .....	38
<b>Chapter 4</b>	<b>Results and Discussion .....</b>	<b>41</b>
4.1	Phase 1 .....	41
4.1.1	Initial Parameters .....	42
4.1.2	Number of Epochs for the Sub-dataset 1.....	45
4.2	Phase 2.....	49
4.2.1	Number of Epochs for the Sub-dataset 2.....	49
4.2.2	Batch-size for the Sub-dataset 1 .....	51
4.2.3	Batch-size for the Sub-dataset 2 .....	53
4.3	Phase 3.....	55
4.3.1	Optimizers .....	56
4.3.2	Dropout .....	57
4.3.3	Softmax Temperature .....	60
4.4	Phase 4.....	64
4.4.1	Sub-datasets Size.....	65
4.4.2	Layers .....	66
4.5	Best Model and Optimal Parameters.....	67
4.6	Fragmentation Growing Procedure .....	69
4.7	General Discussion .....	72
<b>Chapter 5</b>	<b>Conclusion.....</b>	<b>73</b>
<b>References</b>	<b>.....</b>	<b>75</b>

# Acronyms

ADME	Absorption, Distribution, Metabolism and Excretion
AI	Artificial Intelligence
ANN	Artificial Neural Network
BLSTM	Bidirectional Long Short-Term Memory
DL	Deep Learning
DNN	Deep Neural Network
FDA	Food and Drug Administration
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
ML	Machine Learning
QSAR	Quantitative Structure Activity Relationships
RNN	Recurrent Neural Network
SMILES	Simplified Molecular Input Line Entry Specification



# List of Figures

Figure 1 - Relations between AI, ML, DL and ANN concepts .....	6
Figure 2 - A mostly complete graph of neural networks .....	8
Figure 3 - Scheme of the proposed methodology .....	20
Figure 4 - Dataset component .....	20
Figure 5 - Most important feature of an RNN – Hidden Layer.....	24
Figure 6 - An unrolled RNN .....	24
Figure 7 - RNN formulas .....	25
Figure 8 - LSTM cells connected to each other .....	25
Figure 9 - LSTM core line.....	26
Figure 10 - Forget gate layer.....	26
Figure 11 - Input gate layer .....	27
Figure 12 - Update of the cell state.....	27
Figure 13 - Output gate layer .....	28
Figure 14 - GRU gating mechanism formulas .....	29
Figure 15 - BLSTM .....	29
Figure 16 - Dropout example .....	31
Figure 17 - SMILES generator component .....	32
Figure 18 - Dictionary of existing characters .....	33

Figure 19 - Mapping SMILES.....	34
Figure 20 - Padding SMILES.....	34
Figure 21 - One-hot encoding.....	35
Figure 22 - Models trained .....	36
Figure 23 - Model using LSTM with 2 layers .....	36
Figure 24 - Example of the code of the implementation of the model.....	37
Figure 25 - Prediction symbol by symbol .....	37
Figure 26 - Fragmentation growing procedure.....	38
Figure 27 - Validation component.....	38
Figure 28 - Use of MolVS to validate the SMILES .....	39
Figure 29 - Models in test – Phase 1.....	42
Figure 30 - Results with the initial parameters .....	44
Figure 31 - Epochs for the sub-dataset 1 - RNN - 1 layer.....	47
Figure 32 - Epochs for the sub-dataset 1 - RNN - 2 layers .....	47
Figure 33 - Epochs for the sub-dataset 1 - LSTM - 1 layer .....	47
Figure 34 - Epochs for the sub-dataset 1 - LSTM - 2 layers.....	47
Figure 35 - Epochs for the sub-dataset 1 - GRU - 1 layer .....	48
Figure 36 - Epochs for the sub-dataset 1 - GRU - 2 layers .....	48
Figure 37 - Epochs for the sub-dataset 1 -BLSTM - 1 layer.....	48
Figure 38 - Epochs for the sub-dataset 1 - BLSTM - 2 layers .....	48
Figure 39 - Models in test – Phase 2.....	49

Figure 40 - Epochs for the sub-dataset 2 - LSTM - 1 layer .....	50
Figure 41 - Epochs for the sub-dataset 2 - LSTM - 2 layers.....	50
Figure 42 - Epochs for the sub-dataset 2 - GRU - 1 layer .....	51
Figure 43 - Epochs for the sub-dataset 2- GRU - 2 layers.....	51
Figure 44 -Batch size for the sub-dataset 1 - LSTM - 1 layer.....	52
Figure 45 - Batch size for the sub-dataset 1 - LSTM - 2 layers .....	52
Figure 46 - Batch size for the sub-dataset 1 - GRU - 1 layer.....	53
Figure 47 - Batch size for the sub-dataset 1 - GRU - 2 layer.....	53
Figure 48 - Batch size for the sub-dataset 2 - LSTM - 1 layer.....	54
Figure 49 - Batch size for the sub-dataset 2 - LSTM - 2 layers .....	54
Figure 50 - Batch size for the sub-dataset 2 - GRU - 1 layer.....	55
Figure 51 - Batch size for the sub-dataset 2 - GRU - 2 layer.....	55
Figure 52 - Models in test – Phase 3 .....	55
Figure 53 - Optimizers - LSTM - 2 layers.....	57
Figure 54 - Optimizers - GRU - 2 layers.....	57
Figure 55 - Dropout - LSTM - 2 layers - Adam.....	60
Figure 56 - Dropout - LSTM - 2 layers - RMSprop.....	60
Figure 57 - Dropout - GRU - 2 layers - Adam.....	60
Figure 58 - Dropout - GRU - 2 layers - RMSprop.....	60
Figure 59 - Software temperature - LSTM - 2 layers - Adam – 0.3.....	63
Figure 60 - Software temperature - LSTM - 2 layers - Adam – 0.2.....	63

Figure 61 - Software temperature - LSTM - 2 layers - RMSprop – 0.3 .....	63
Figure 62 - Software temperature - GRU - 2 layers - Adam- 0.1 .....	64
Figure 63 - Software temperature - GRU - 2 layers - RMSprop – 0.1 .....	64
Figure 64 - Models in test – Phase 4.....	64
Figure 65 - Sub-datasets size - LSTM - 2 layers .....	66
Figure 66 - Layers of the best model.....	67
Figure 67 - Best Model - Validity.....	69
Figure 68 - Best Model - Dataset replicates.....	69
Figure 69 - Best Model - PubChem .....	69
Figure 70 - Best Model - Databases .....	69
Figure 71 - SMILES used in the fragmentation growing procedure .....	70
Figure 72 - Fragmentation growing procedure - Benzamidine - Validity .....	70
Figure 73 - Fragmentation growing procedure - 3-Methylpyrazole - Validity.....	70
Figure 74 - SMILES generated with the fragment Benzamidine .....	71
Figure 75 - SMILES generated with the fragment 3-Methylpirazole .....	71



# List of Tables

Table 1 - Available public data in PubChem .....	21
Table 2 - Examples of SMILES of some compounds.....	22
Table 3 - Results with the initial parameters (%).....	44
Table 4 – Results for the parameter number of epochs for the sub-dataset 1 (%) .....	46
Table 5 - Results for the parameter number of epochs for the sub-dataset 2 (%) .....	50
Table 6 – Results for the parameter batch size for the sub-dataset 1 (%).....	52
Table 7 - Results for the parameter batch size for the sub-dataset 2 (%) .....	54
Table 8 - Results for the parameter optimizer (%) .....	57
Table 9 - Results for the parameter dropout (%).....	59
Table 10 - Results for the parameter Software temperature (%).....	62
Table 11 - Results for the parameter sub-datasets size (%).....	65
Table 12 - Results for the final test with different number of layers (%) .....	67
Table 13 - Best Model analysis .....	68
Table 14 - Results of fragmentation growing procedure (%).....	70



# Chapter 1

## Introduction

### 1.1 Context

Artificial Intelligence (AI) is a reality in all areas of science. Its application brings, as main advantages, the process to analyse scientific data in a systematic, precise and reproducible way [1].

Machine Learning (ML) is a sub discipline of AI. In this decade, the most promising form of ML used is Deep Learning (DL) [2], which allows ML to support large quantities of raw data and discovers the representations necessary for detection or classification [3].

There are numerous methods that use ML algorithms, they are used for very distinct purposes. Examples of methods used in ongoing works are: Support Vector Machine (SVM), Bayes Method, Regression, Random Forests and Neural Network. These methods have been adopted in several biomedical applications, such as diagnose, prognosis, virtual screening, progress of cancer, regulator research and others contexts [4].

Thus, in the field of medicine, multiple works explores the use of DL in cardiology [5], ophthalmology [6], radiology [7], pneumology [8], psychiatry [9] and oncology, such as lymphoma [10] and breast cancer [11], among other specialities.

Moreover, ML have been applied in pharmaceutical area. Modern ML techniques have been used to learn Quantitative Structure Activity Relationships (QSAR) and to develop AI programs that accurately predict in silico how chemical modifications might influence biological behaviour, such as Absorption, Distribution, Metabolism and Excretion (ADME) properties [12].

Therefore, DL have been successfully used in drug discovery, as it is possible to observe in several projects [13]. Drug discovery aims to identify potential new medicines through a multidisciplinary process, including scientific areas, such as biology, chemistry and pharmacology [14]. This process is composed of six main steps: target identification, library development and chemical synthesis, high-throughput screening, lead optimization, clinical trials and drug approval [13].

In order to discover a new medicine, bioinformatics use different methodologies; each of them has distinct goals, inputs, outputs, and are target-oriented or diversity-oriented [15]. There are many computational use cases that lay on DL to help in different steps of drug discovery process. Some of those cases are the property and activity prediction, drug-target interactions, synthesis planning, virtual and reverse screening, low-data drug discovery and, finally, *de novo* drug design [16, 17].

The *de novo* drug design approach aims to generate new chemical structures, through recurrent models as more common types of neural network chosen. SMILES (Simplified Molecular Input Line Entry Specification) are the elected type of compound representation [18-20].

## 1.2 Motivation

Presently, all over the world, health systems face several challenges. The rising of the number of different diseases, commonly in the same person, called multimorbidity, the disability due to aging, the epidemiological transition, the higher expectations in health services and society, the increasing expenses in health and poor productivity, are some of the most pressing ones. The current scenario of economic austerity policies that are constraining investment in health systems does not help to identify those problems and challenges [3].

Drug Discovery is a process that consumes time, labour, money, and, it is very risky. In fact, developing a drug usually takes 10-15 years, 1.8 billion US dollars and the success

rate of developing a new molecular entity is only 2.01%, on average. The FDA reports that the number of drugs approved by itself has been declining since 1995, although, the investment in this area has been gradually increasing [21]. The majority of the drug candidates fail to become an approved drug in the late steps of clinical trials due to some unexpected side effect or toxicity problems.

Computational drug discovery has the potential to improve efficiency and effectiveness of the experimental process of drug discovery [3]. These computational techniques allow the reduction of human mistakes, usually made during drug design, the reduction of the amount of candidate compound, the reduction of the time needed for testing each candidate, the improvement of the optimization and, also, the improvement of the recapitulation of disease biology [13].

Moreover, new systems and methods are being developed in order to detect toxicity of candidate compound at early stages of drug discovery, these strategies combined with *in vitro* and *in vivo* biological testing can drastically decrease the time and the cost of the process, and also improve safety evaluation [13, 22].

Finally, these technologies combined with the availability of huge databases of chemicals or compounds potentialize the shortening of the time needed for the whole process of drug discovery from drug design until the clinical trials [13].

### 1.3 Goals

The main goal of this dissertation is to explore the use of multiple architectures of recurrent neural networks to generate new and valid molecules that can be used as drug candidates in further studies. Toward this aim SMILES, a linear textual representation of molecules, will be used, and the explored models will be inspired on the current state of the art for text generation.

Furthermore, there are secondary aspirations, such as:

- Define a dataset of SMILES that can be used as a consistent benchmark;

- Evaluate multiple recurrent networks and find its optimal parameters.
- Define a procedure to restrict the generation of SMILES.
- Define a procedure to syntactically and biochemically validate the generated SMILES;

## 1.4 Document Structure

This dissertation is constituted of five chapters. This chapter includes the context, the motivation and the goals for this work. The second chapter corresponds to the State of the Art and consists in the explanation of the context of this study and other strategies/options already created to solve the existing problems. The third chapter corresponds to the Methods, where the methodology used to achieve the goals are explained. The fourth chapter presents the Results and Discussion of the applied methodology. The obtained results for the several tests performed are presented and analysed to the proposed and created method. Finally, the fifth chapter presents the Conclusions of this work, and also future steps of this study

# Chapter 2

## State of the Art

In this chapter, it will be defined concepts of AI, ML, DL and Artificial Neural Networks (ANN), and it will be clarified how they relate with each other and with drug discovery. Moreover, some examples of strategies used nowadays in drug discovery will be explained.

AI, ML and DL are fundamental concepts in this dissertation. As it is possible to observe in Figure 1, these three concepts are intimately related, since ML is a sub discipline of AI, and DL is a form of ML.

## 2.1 From Artificial Intelligence to Deep Learning

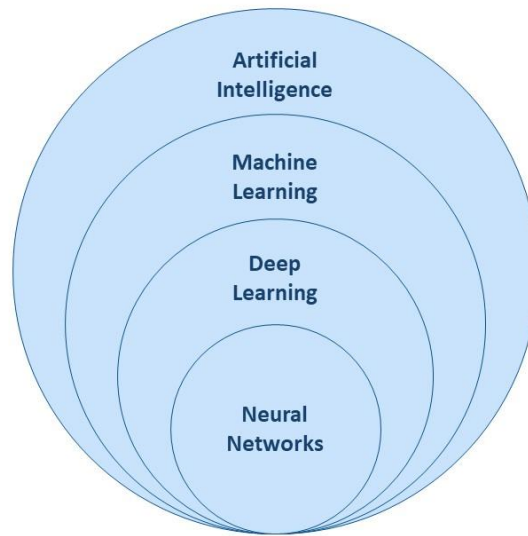


Figure 1 - Relations between AI, ML, DL and ANN concepts

The term AI was proposed by John McCarthy in 1956, however there were intelligent and thinking machines way before that time. The first references about medical AI date to the early 1970s [23]. Since then, AI and technology, in general, have been evolving very rapidly, allowing the developing of solutions in fields such as industrial production processes, logistics, mobility, shopping, personal assistant and digital marketing [24]. AI can be defined as an extensive scientific discipline with its roots in philosophy, mathematics and computer science, that has the purpose of learning and developing systems that show properties of intelligence [3].

ML is a sub discipline of AI, where algorithms learn how to predict something from associations in datasets, in others words, it is possible to say that ML is the application of statistical models to data using computer programs [3].

ML is a powerful tool for interpreting biological data. However, DL techniques, a non-traditional form of ML algorithms, are very popular and more effective ways of reaching good results. They are based on models with fewer assumptions about underlying data and they can handle more complex data [3].



DL methods allow a machine to learn from large amounts of raw data with the purpose of discover how to represent it in order to promote its detection or classification. These methods depend on multiple layers of representation of the data with consecutive transformations, which highlight the more relevant information and censor irrelevant one. The representations of data can assume multiple levels of abstraction [3, 25].

DL methods have been responsible for significative advances in ML. Moreover, DL can find complex structures in large datasets. It uses a backpropagation algorithm in order to show how the machine should change its internal parameters, which are used to compute the representation in each layer from the representations in the previous one [25].

DL uses ANNs. There are different types of ANN, with different architectures and purposes. ANNs are a type of computer model which use the human brain functioning as an inspiration. These neural networks are very efficient at solving well defined problems that involve pattern recognition and categorisation, while traditional computer systems are programmed to develop applications. This property makes neural networks very adaptable to solve a wide variety of problems, even problems that traditional computing approaches could not solve until now [26, 27].

There are many types of ANN architectures, with different purposes and applications. Figure 2 displays a complete chart of Neural Networks, where it is possible to highlight some, such as Deep Feed Forward (DFF), Recurrent Neural Network (RNN), Long-short Term Memory (LSTM) and Convolutional Neural Networks (CNN).

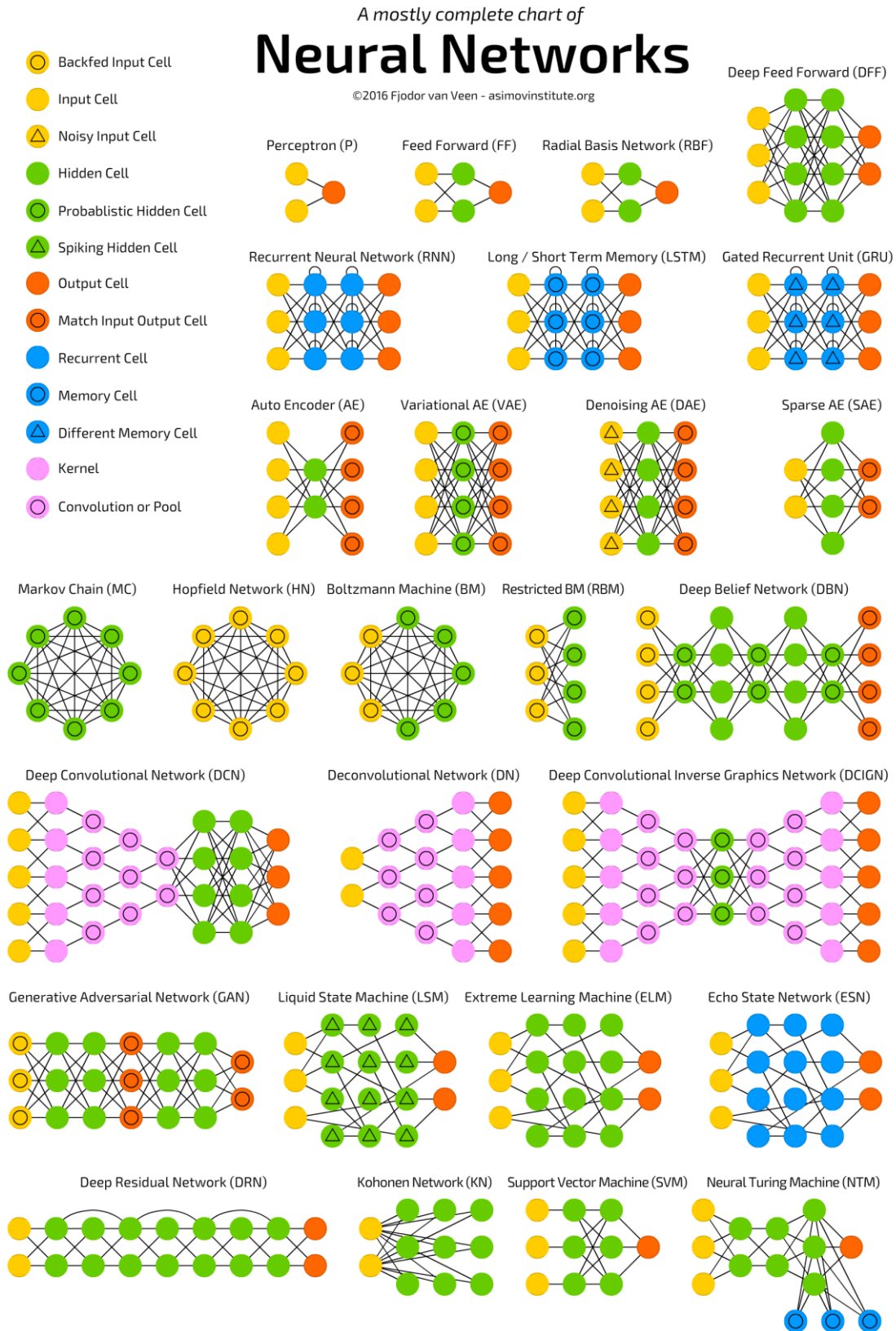


Figure 2 - A mostly complete graph of neural networks, obtained from [28]

### 2.1.1 Applications

DL has been used extensively in a wide range of fields, especially through the utilization of ANNs.

CNNs have provided major breakthroughs in image classification, since they were created with the purpose of extract information from raw signals. For example, they can be used in face recognition, scene labelling, image classification, action recognition, human pose estimation and document analysis [29]. A more specific example it is radiology, since it can interpret the images taken from the human body [30].

RNNs and its variants were created to deal with time series event. For example, they can be used in text summarization [31], translation [32], speech recognition [33] and text recommendations [34] and automatic music composition [35].

## 2.2 Drug Discovery

Drug discovery is a multidimensional problem that involves the research of different properties of natural and synthetic compounds, such as safety, pharmacokinetics and efficacy, during drug candidate selection. AI, ML and DL, with the evolution of technology started to become part of this process [3].

The traditional drug discovery consumes a lot of resources, and most of the drug candidates fail to become an approved drug in late steps of clinical trials, due to some unexpected side effect or toxicity problem. Computational drug discovery has the potential to improve efficiency and effectiveness of the process, through aspects, such as reduction of human mistakes, reduction in the amount of candidate compound, reduction of the time needed for testing, detection of candidate compound toxicity in early stages, improving safety evaluation and, finally, decreasing the time and the cost of the all process [3, 13, 22].

ANN started to be used in molecular informatics and drug discovery, approximately, two decades ago, and since then the use of DL in drug discovery evolved significantly.

Nowadays, the interest in adapting advanced ANN architectures for medical and pharmaceutical research was renewed, and every day appears a different and new strategy [36].

### 2.2.1 Steps of Drug Discovery

The process of drug discovery, normally, is triggered by the appearing of a disease or a clinical condition that does not have a therapy available in the market. Under the necessity to find a new drug, the drug discovery process starts, the steps of this process are enumerated below [13].

1. Target Identification
2. Target Validation
3. Lead Identification
4. Lead Optimization
5. Preclinical Development
6. Clinical trials
7. Registration

In Portugal, the registration of the new drug into the market needs the approval of INFARMED (“*Autoridade Nacional do Medicamento e Produtos de Saúde*”, created by *Decreto-Lei n.º 495/99 de 18, de Novembro* [37] and updated by *Decreto-Lei n.º 269/2007, de 26 de Julho* [38]) (and, also, the approval of European Medicines Agency (EMA) [39]. In the United States of America (USA), the agency that regulates the entry of drug in the market is called Food and Drug Administration (FDA) [40].

The drug discovery process is slow and expensive. For those reasons, the investment in computational methods is risen and covers the first four steps of this process, target identification, target validation, lead identification and lead optimization.

## 2.2.2 Computational Drug discovery

Computational drug discovery evolved significantly in the last decade, some use cases examples will be explained in this subsection, such as property and activity prediction, drug-target interactions, synthesis planning, virtual and reverse screening, *de novo* drug design, and low-data drug discovery.

### Property and Activity Prediction

In drug discovery, it is essential that the candidate compounds have a set of criteria and properties in order to succeed in clinical trials. Therefore, compound optimization is a multidimensional challenge, since it is crucial to optimize or exclude compounds that could fail in following phases of the drug discovery process because of its properties [16].

There were already methods that use ML algorithms to predict properties, such as Support Vector Machines (SVM) [16, 41], Random Forests (RF) [42, 43] and Bayesian learning [44]. However, DL algorithms achieved better results than the previous algorithms, since Deep Neural Networks (DNN) can handle a huge amount of descriptors without the need of feature selection, use dropout to avoid overfitting and have a set of hyperparameters that can be used to adjust the model to the problem in hands [17]. The success of these methods helps the drug repositioning strategy. This strategy consists in identifying new usage for the existing drugs on the market [45].

Over time, there were some DL methods to predict activity and properties. When the compounds were represented for its molecular descriptors, the easier way to achieve the purpose was to use a fully connected DNN [46]. Years later, two distinct methods emerged, the first one used a variant of RNN, called UGRNN, which first converted molecular structures into vectors of the same length as the molecular representation, and then used them to fully connect a neural network [47]. The second method was similar to the UGRNN, since it also used neural networks to generate a vector that would be used to train the network. That second method was called graph convolution model [17, 48].

Until now, the molecules were represented by its descriptors, but the opportunity to represent the molecule by a SMILES string appeared, and it became the input necessary to build a predictive model, using LSTMs, without the need to generate molecular descriptors. Furthermore, images of 2D drawings of molecules were used with CNNs and achieved great results comparing with the other models. This possibility of feeding networks with the direct structure of the molecules and without the need of descriptors was a revolutionary and essential feature that distinguish DL from the other ML methods [17].

### **Drug-Target Interactions**

The identification of drug-target interactions is a massive part of the drug discovery process. The drug-target interactions are usually classified in a binary way, 0 or 1, whether there is or not an interaction between them. However, protein-ligand interactions assume a continuum of binding strength values, called binding affinity. The knowledge of affinity data is increasing in databases. Consequently, it allows the use of advanced learning techniques such as DL architectures in the prediction of binding affinities in drug-target pairs [49].

Drug-target interactions were, until recently, approached as a binary classification problem. However, this approach neglected important information about protein-ligand interactions, since binding affinity provides information on the strength of the interactions, and it can be expressed in measures such as dissociation constant ( $K_d$ ), inhibition constant ( $k_i$ ) and the half maximal inhibitory concentration ( $IC_{50}$ ) [49].

Nowadays, the use of this type of characterization using the binding affinity score, instead of the binary classification, provides datasets more realistic and predicts more values for the strength of the interaction between drug-target pairs [49]. Some works use CNNs to score the protein-ligand interaction [17, 50].

## Synthesis Planning

The organic synthesis phase is one of the most critical step in drug discovery process, since it is necessary to synthesize the new molecules in order to proceed the compound optimization path and to identify molecules with improved properties. Since this phase is really important, its planning is essential for success [16, 51, 52].

Some computational approaches have been implemented to help the synthesis planning. Three main aspects can be highlight: 1) prediction of the result of a reaction with a given set of reagents, 2) prediction of the yield of a chemical reaction, 3) as well as retrosynthetic planning [16, 53, 54].

Synthesis planning may be divided in forward synthesis and retro synthesis.

Two main approaches for forward synthesis predictions are: 1) the combination of quantum chemical descriptors with manual encoded rules and ML to predict a reaction and its products, and 2) training DNN with millions of reactions[55, 56]. One approach used for retrosynthetic analysis uses RNNs fed with reactants and products in form of SMILES string in an encoder-decoder architecture [16, 57].

## Virtual and Reverse Screening

Virtual Screening has emerged, in the past decades, to identify interactions between components, as drugs and their targets. Find these interactions is crucial for drug discovery process. However, experimental approaches to achieve it are ineffective due to feasibility problems. They are labour intensive, costly and time consuming [13]. Despite of recent advances in combinatorial chemistry and high-throughput screening, which allowed chemists to synthesize large amounts of compounds, just a small percentage could be manufactured. Virtual screening uses several computational techniques that allow chemists and bioinformatics to reduce a huge virtual library to a more manageable size. Then, it will be possible to identify which structure are most likely to bind to a drug target, typically a protein, receptor or enzyme [58].

It is possible to say that virtual screening has emerged as a reliable, inexpensive method for identifying candidate compounds, and, because of that, it has a positive impact on the drug discovery process [59].

Conventional virtual screening consists of a group of compounds that are screened against a chosen specific target, and the interactions are identified as its candidate. On the other hand, there is, also, the reverse virtual screening method that works the other way around, since the goal is to identify a candidate target, so a compound is sought against multiple proteins [13].

### **Low-data Drug Discovery**

DNNs have been demonstrated to be crucial in the field of prediction, when inferring the properties and activities of small-molecule compounds. Nevertheless, these techniques require a large amount of training data and this aspect is a limitation of these techniques. One-shot learning associated with DL, mainly, with ANN, can reach good predictions, using small amounts of training data, unlike most of the ML techniques used. Thus, one-shot learning can use a small dataset, or even just one image or sample in order to generate very accurate predictions [60].

Some successful works done with limited data available as dataset used LSTM networks, but a recent type of architecture has been used in DL too, called memory augmented neural networks [17, 61].

### **De novo Drug Design**

Approximately 25 years ago, the development of the *de novo* design with the aim to generate new compound without reference compounds started to show results. Since then, numerous approaches arose [16, 62, 63].



One of the first methods used was Variational AutoEncoder (VAE). Its purpose was to generate chemical structures. This method was used as a molecular descriptor generator and was coupled with a Generative Adversarial Network (GAN), which is a special neural network architecture [64, 65].

Another promising method developed used RNNs to generate new compounds. This method relies on a large dataset constituted of SMILES strings and generates new valid SMILES strings that were not in the used dataset for the training [19]. RNN can be classified as a generative model for molecule structures, since this type of ANN learns the probability of the distribution of characters in SMILES strings to generate new valid ones. This method allied with other techniques generate improved results. Two of the techniques used were transfer learning and reinforcement learning. The first one consists in training the RNN with a 1st general dataset, and then training with a 2nd target-specific dataset. The second technique uses score functions to evaluate the generated SMILES and achieve SMILES with chosen properties [17, 66].

### 2.2.3 Related Works

In this subsection, it will be shown three related studies which share goals and/or methods with this dissertation. The goals of this dissertation (already exhibited in the previous chapter), are included in the *de novo* drug design use case explained before, more precisely, in the recurrent models that aim to generate new chemical structures.

In the work “Drug Analogs from Fragment Based Long Short-Term Memory Generative Neural Networks” [20], the authors used LSTM generative neural networks with SMILES of drug-like molecules from four sources, such as ChEMBL, DrugBank. Fragments from commercial catalogues and FDB17. They created 6 different primary datasets with different features to test and used transfer learning with 10 different drugs that cover a broad range of size and complexity. These 6 datasets and 10 drugs were combined for the training, which means that every dataset was trained with each selected drug. However, the hyperparameters of the LSTM and its layers were kept in all of the tests. The model chosen is constituted by three LSTM layers, a dropout layer and a time distributed layer. The

dropout rate was 0.2, the activation function was softmax and the optimizer was Adagrad. For the primary dataset, the number of epochs was 50, but the batch-size was dependent on the dataset, 16, 32 or 64. For the transfer learning the number of epochs was 20 and batch size 4. After the training and prediction, the valid SMILES were retained, the duplicates were removed, and SMILES, with undesirable functional groups, were also removed. The last step was to select high similarity structure analogs and conclude how the primary dataset influences this similarity, depending on the selected drug. This implementation was done using Python, Keras, TensorFlow, Numpy and RDKit [20].

The second related work was “Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks” [19]. This work used LSTM to generate models for molecular structures, allied to transfer learning in order to restrict the generated SMILES. To encode the SMILES, the authors used one-hot encoding, and the prediction was done symbol by symbol. The end of line (EOL) symbol represents the end of one SMILES generated. As main dataset, they used 1.4 million canonical SMILES taken from ChEMBL. The LSTM model trained had three stacked LSTM layers, each one followed by a dropout layer, with a dropout ratio of 0.2, a batch-size of 128 and Adam as optimizer. They used a standard target prediction to verify whether the generated molecules were active on the desired targets. They chose three different targets and the tests with transfer learning were made with them. The percentage of generated SMILES was 97.7%, according to CDK toolkit, SMILES duplicated, or part of the training dataset was removed. After these steps, the authors checked whether the molecules generated could be considered as valid starting points to drug discovery process, using internal AstraZeneca filters, and the percentage obtained was 75%. Thus, they conclude that the implemented model could generate a huge number of SMILES with similar physico-chemical properties to the training molecules and that transfer learning enables the creation of new molecules with the desired activity. To implement this work they used tools as Keras, Scikit-Learn and XG Boost [19].

The third chosen was “Generative Recurrent Networks for *De Novo* Drug Design” [18]. This work aimed to predict SMILES for specific molecular targets. The authors used LSTM neural networks and their chosen compound representation notation was SMILES. They used transfer learning in order to restrict the generation of the compounds and also a fragment-based strategy. The main dataset used was constituted by 541,555 SMILES taken

from ChEBML22; and, for the transfer learning, they used three different datasets of varying sizes. The pre-processing of the dataset consisted in padding, mapping and one-hot encoding. The trained LSTM had 2 layers and a dropout rate was 0.3. Softmax was used as activation function, and the number of epochs for the main dataset were 22, and for the transfer learning was set to 12. The generated SMILES were validated with RDKit, and, when the softmax temperature were 0.5, they obtained 98% of valid SMILES. The final step consisted in a fragment-based strategy, in this case benzamidine, as a start point of the prediction of each SMILES, 97% of the generated SMILES were valid.



# Chapter 3

## Methods

This chapter explains the methods and methodologies used in this thesis, from the dataset until the validation process. Figure 3 summarizes the relationship between each component of the methodology chosen and those are explained in the following sections. The first section describes the dataset used to train the models implemented and better explains the definition of SMILES. Furthermore, some examples are given in order to help the reader to understand the content of the dataset. The second section identifies and justifies the tools used to achieve the goals at stake. The third section describes the four types of ANNs that were implemented and analysed.

These first three sections prepare the reader for the fourth section, which contains all the methodology and strategy used in the created model. The fifth section explains the definitions of the parameters of recurrent networks architectures that were used and tested in this work. The sixth and last section is about validation. In fact, one of the goals of this work is to be able to validate the generated SMILES by the model explained in this chapter.

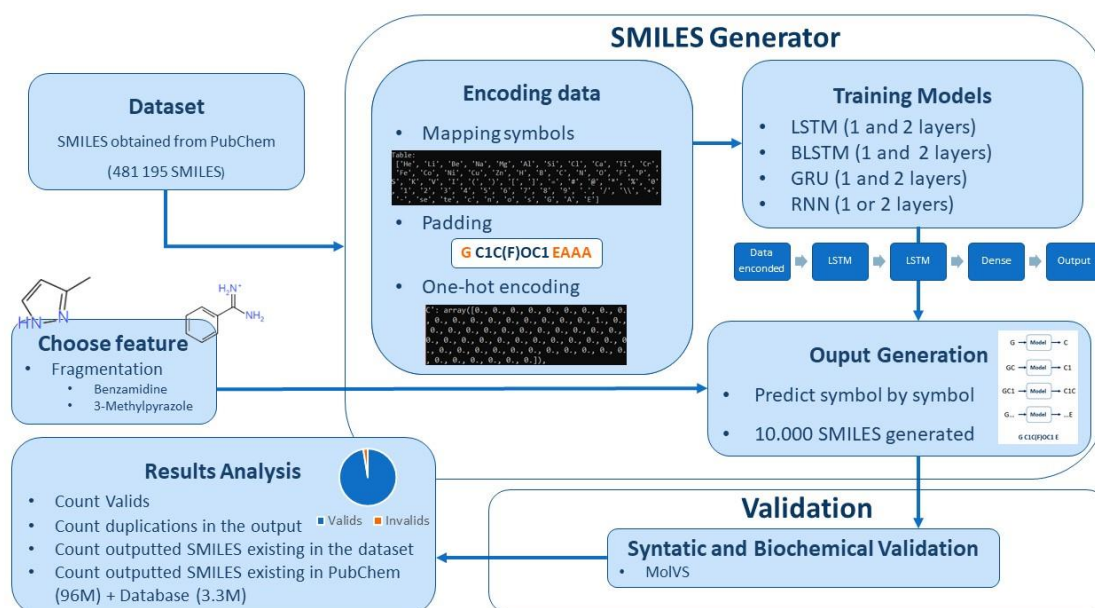


Figure 3 - Scheme of the proposed methodology

### 3.1 Dataset

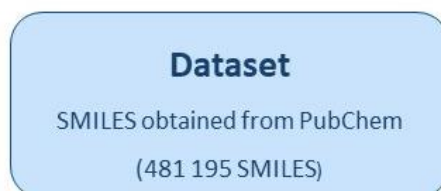


Figure 4 - Dataset component

The dataset (Figure 4) used to train the model implemented was obtained from PubChem. PubChem is an open chemistry database from the National Institutes of Health (NIH) launched in 2004. This database allows the users to upload their scientific data, which allows that others researcher use it; for that reason, it became an essential resource for scientists, students and the general public. PubChem is mostly constituted by small molecules, but also by larger ones such as nucleotides, lipids, carbohydrates and chemically-modified macromolecules [67]. Table 1 contains the types and quantity of data existing in

PubChem. The set of data with 481,194 canonical SMILES, used as dataset in this work, was taken from the almost 96M of compound in SMILES notation from this database.

In the tests, this dataset was divided into two sub-datasets, since the training of the networks is done in two steps. First the network is trained with the sub-dataset 1, and then with the sub-dataset 2.











Table 1 - Available public data in PubChem, data taken from [68]

Data Collection	Quantity
Compounds	95,689,405
Substances	234,675,576
BioAssays	1,340,530
Bioactivities	265,364,936
Gene Targets	58,029
Protein Targets	17,847
Taxonomy Targets	3,746
Literature	29772917
Patents	3142716

Simplified Molecular Input Line Entry Specification (SMILES) is a line notation for entering and representing molecules and reactions, as the name suggested. This notation is a simple way of representing a compound, one of the main advantages to choose SMILES as

input and output of the model. Table 2 shows some typical examples of compounds with its respective SMILES string [69]. These examples have the purpose to familiarize the reader with this notation, but it is important to clarify that a compound can be written, in this notation, in very different ways. However, there are a standard way of writing SMILES, designated as canonical SMILES.

Table 2 - Examples of SMILES of some compounds [44]

Compound name	SMILES	Structure
Ethane	CC	
Carbon dioxide	O=C=O	
Hydrogen cyanide	C#N	
Triethylamine	CCN(CC)CC	
Acetic acid	CC(=O)O	
Cyclohexane	C1CCCCC1	
Benzene	c1ccccc1	
Hydronium ion	[OH3+]	
Deuterium oxide	[2H]O[2H]	
Uranium-235	[235U]	



## 3.2 Tools

The model and the validation tests were all written in the same programming language, Python. This is a simple language with multiple free libraries available, that can be crucial to the goal of this dissertation.

There were five libraries used in the model implementation: NumPy, Keras, TensorFlow MolVS and PubChemPy. NumPy is the fundamental package for scientific computing with Python [70], this was used to build and shape the input and output matrices. The second library used was Keras. It is a high-level neural networks API that can run on top of TensorFlow, CNTK, or Theano. It was developed in Python with the goal of allowing fast experimentation [71]. The third library was TensorFlow that is an open source library for high performance numerical computation [72]. The fourth library, MolVS was used to validate the generated SMILES by the model, its function will be explained in section 3.5. Finally, the fifth library, PubChemPy was used to analyse the SMILES, since this library allows the user to search for compounds in the PubChem database.

There are many options of libraries to build DL model, but the choice was Keras, because: 1) previous and similar works used this library, 2) the internet is full of examples using Keras, and 3) documentation is really complete and useful. The reasons to choose TensorFlow were similar to Keras.

All code was written in Sublime text 3 and was executed in the command line of a Windows and an Ubuntu operating system.

## 3.3 Recurrent Neural Networks Architectures

This section explains how the four types of variations of RNN used in this dissertation work and highlights the major differences between them. The ANNs will be presented in the following order, firstly simple RNN, then LSTM, GRU and finally, BLSTM.

### 3.3.1 Simple Recurrent Neural Networks (RNN)

A simple RNN is a type of ANN where the output from the previous step are fed into the current step as an input. In cases that the purpose is to predict the next element of a sentence, it is essential to know which was the previous word. RNNs were created to solve issues in this field, and a hidden layer is the most important feature of an RNN (Figure 5), because it is this component that allows the network to keep some information about a sequence [73].

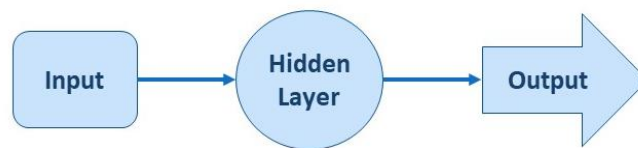


Figure 5 - Most important feature of an RNN – Hidden Layer, based on [74]

This type of ANN has a “memory” since it remembers all information about the weights that were already calculated. The same parameters are used for each input as it completes the same assignment on all the inputs or hidden layers to produce the output. This strategy reduces the complexity of parameters, in contrary to other types of ANNs [75]. In more detail, in other ANNs, each layer is independent and has its own set of weights and biases. Each layer does not memorize the previous outputs. However, RNNs using the same parameter in all of the inputs join together all of the hidden layers into a single recurrent layer, as it is shown in Figure 6 [68].

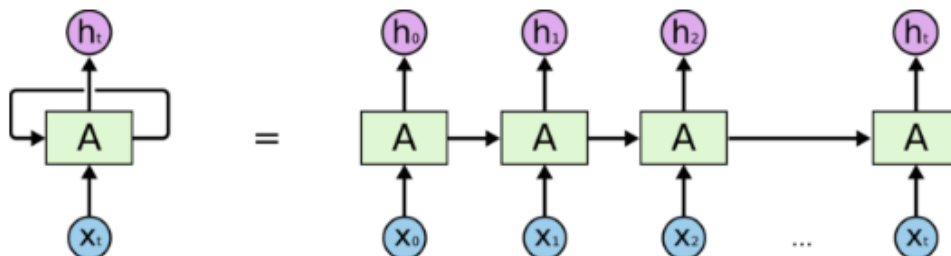


Figure 6 - An unrolled RNN, obtained from [76]

There are three formulas that make this process possible, the first one has as its own purpose to calculate the current state, the second one allows the application of the activation function (tanh), and the third formula is to calculate the output. The three formulas and its variables are shown in Figure 7 [77].

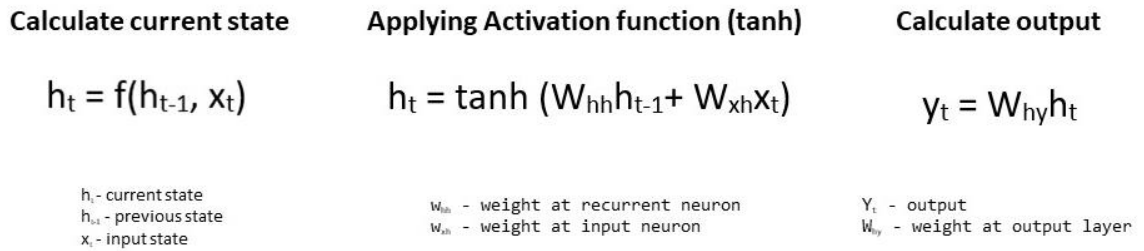


Figure 7 - RNN formulas, obtained from [74]

### 3.3.2 Long-Short Term Memory (LSTM)

An LSTM is an ANN classified as a type of an RNN. LSTM solves the main problems of RNNs, namely the failure to derive context from time steps, which are much far behind. LSTM networks structure remains the same as in RNN, whereas the repeating module does more operations, such as: forget gate operation, input gate operation and output gate operation. A simple LSTM cell consists of 3 gates that are represented in Figure 8 [77, 78].

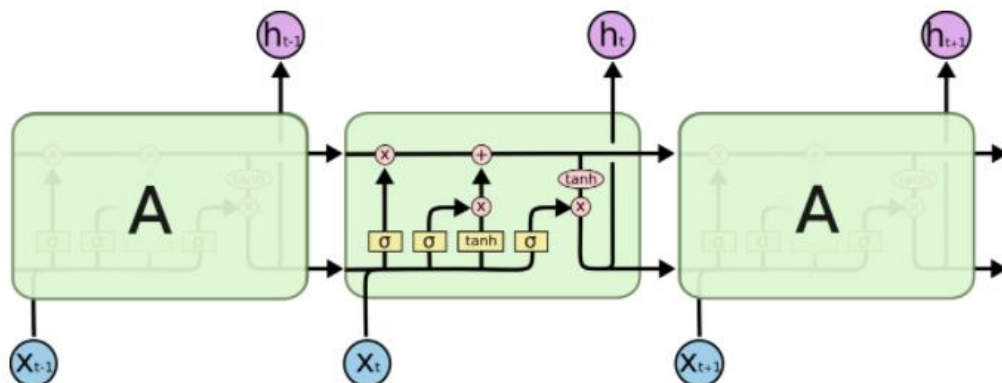


Figure 8 - LSTM cells connected to each other, obtained from [79]

An LSTM structure finds its key in the cell state, the horizontal line running through the top of the cell, shown in Figure 9. The cell works like a conveyor belt that runs through the entire chain. It is easy to keep the information unchanged, although the LSTM can remove or add information to the cell state. These changes are regulated by structures called gates, that are composed of a sigmoid neural net layer and a pointwise multiplication operation. Each sigmoid layer outputs a number between 0 and 1, where 0 means that no information is let through, and 1 means that everything will be let through [77, 78].

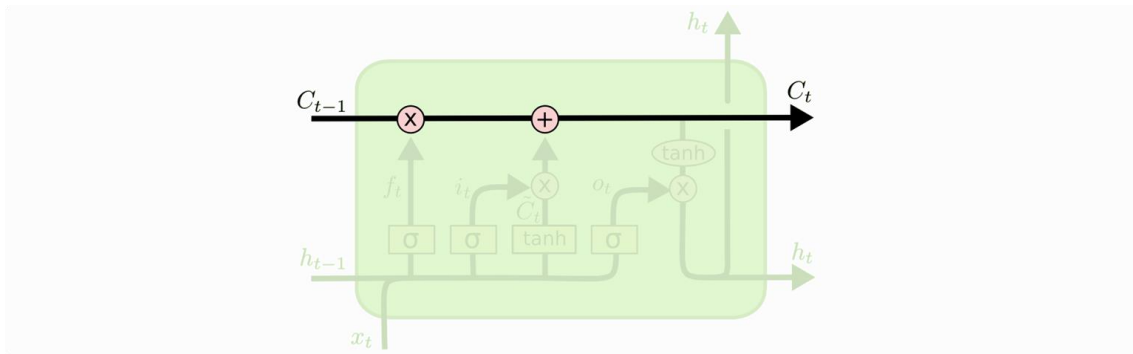
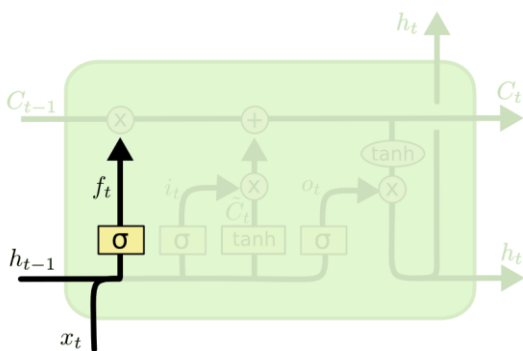


Figure 9 - LSTM core line, obtained from [80]

The first step, in the flow of information in an LSTM, is to decide which information will be forgotten. This decision called “forget gate layer”, is made by a sigmoid layer. This gate and its calculation are represented in Figure 10 [77, 78].



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

Figure 10 - Forget gate layer, obtained from [80]

The second step is to decide what new information will be stored in the cell state. To do that, first a sigmoid layer called “input gate layer” decides which values of the information will be updated. Secondly, a tanh layer creates a vector of new candidate values, that could be added to the state. This gate and its calculation are represented in Figure 11 [77, 78].

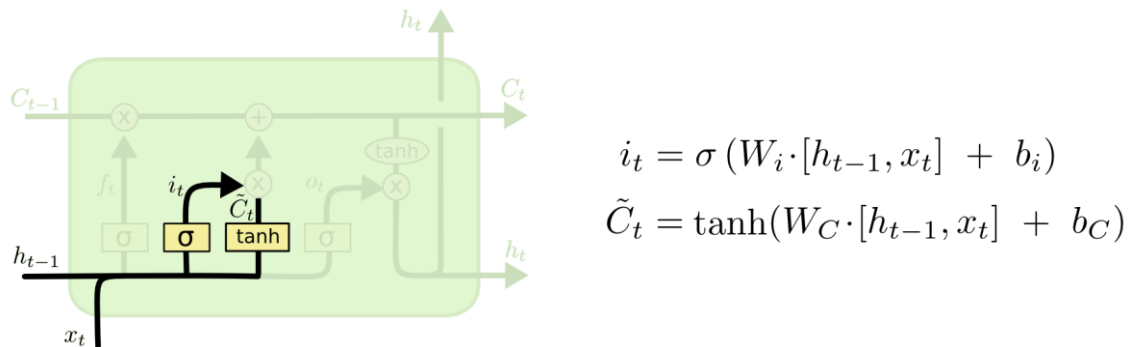


Figure 11 - Input gate layer, obtained from [80]

The third step consists in the update of the old cell into the new cell state. In order to update it, it will be necessary to combine the two values before ( $i_t$  and  $\tilde{C}_t$ ) to create an update to the state. This update is represented in Figure 12 [77, 78].

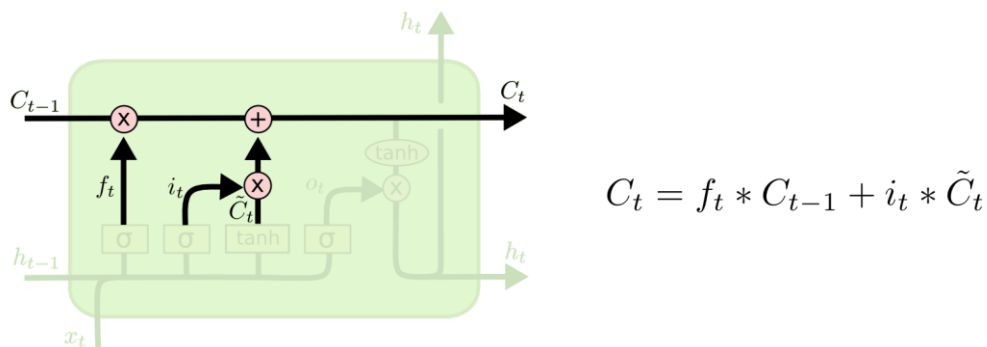
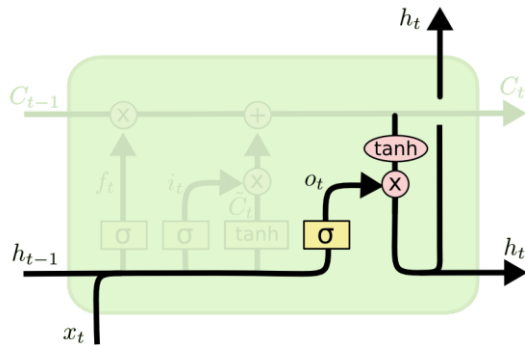


Figure 12 - Update of the cell state, obtained from [80]

Finally, the output must be decided based on the cell state. First, a sigmoid layer called output gate layer decides what parts of the cell state will be outputted. Secondly, the cell state will be put through a tanh layer to push the values to be between -1 and 1, and then multiply it by the output of the sigmoid gate. Figure 13 represents the gate and its calculations [77, 78].



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

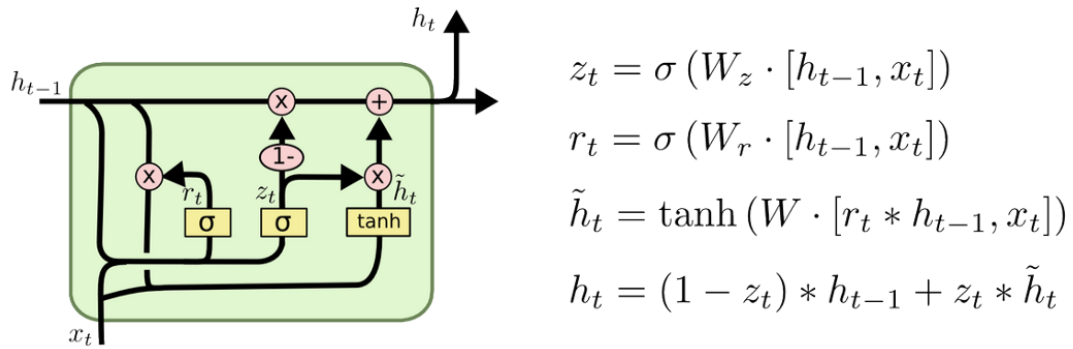
$$h_t = o_t * \tanh(C_t)$$

Figure 13 - Output gate layer, obtained from [80]

### 3.3.3 Gated Recurrent Unit (GRU)

GRU is one variant of LSTM. This ANN keeps the resisting vanishing gradient properties of LSTM, but internally it is faster and simpler than an LSTM.

LSTM and GRU have two main differences. 1) LSTM has 3 gates input, output and forget gate, as it was explained before, and GRU has only two gates, an update z and a reset gate r. 2) GRU does not have a persistent cell state distinct from the hidden state, as in LSTM. The update gate z determines how much of memory must be kept, and the reset input r decides how to combine the new input with the previous value. Figure 14 represents a GRU cell and its formulas [73].



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

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

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 14 - GRU gating mechanism formulas, obtained from [81]

### 3.3.4 Bidirectional Long-Short Term Memory (BLSTM)

Unidirectional LSTM or, simply, LSTM just preserves information of the past, because its inputs are only from the past and fed into the normal time order. However, bidirectional LSTM uses two hidden states combined to run the inputs in both directions; forward and backwards (Figure 15). This feature allows the network to save the data from the next inputs, which means that at any point in time there are information of the previous and the following steps. In simple words, BLSTM is an LSTM that can access to information from the past and from the future, and combine them to calculate the weights just like an LSTM [82].

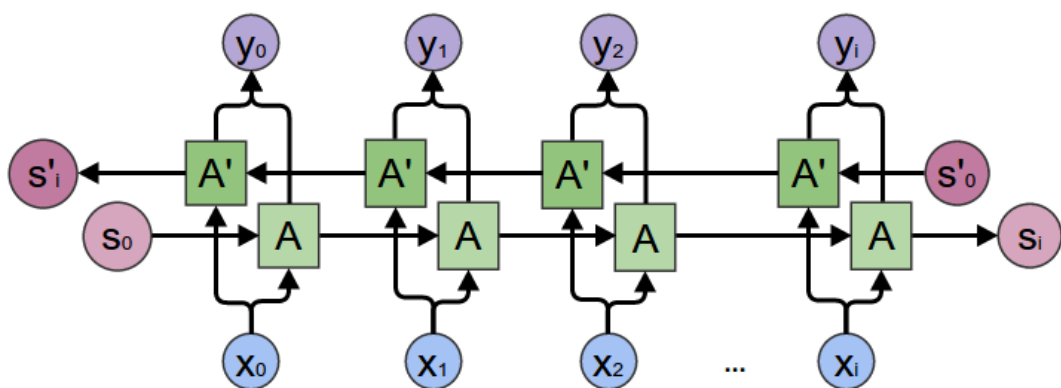


Figure 15 - BLSTM, obtained from [83]

## 3.4 Parameters of Recurrent Architectures

The RNNs architecture used in this work have a group of hyperparameters that can be changed and adapted to the purpose of the work in hands.

In this work, some of the parameters were tested in order to choose the best value for each parameter and to reach the best possible result. The parameters in test were:

- Number of epochs for the sub-dataset 1;
- Number of epochs for the sub-dataset 2;
- Batch size for the sub-dataset 1;
- Batch size for the sub-dataset 2;
- Optimizer;
- Dropout;
- Softmax temperature;
- Number of SMILES in the sub-dataset 1;
- Number of SMILES in the sub-dataset 2;
- Number of layers

The first and second parameters were the number of epochs for the training of each sub-dataset. An epoch is when an entire dataset is passed through the neural network once, in another words is an iteration on a dataset [84].

The third and fourth parameters were the batch-size used to train each sub-dataset. As it was explained before, an epoch is when an entire dataset is passed through the neural network once, but it is not possible to pass the entire dataset into the network at once, so the dataset is divided into number of batches or parts.

The fifth parameter was the optimizer, which has the function of shaping and moulding the model into its most accurate form by tampering with the weights. In these tests, there were tested the following optimizers, available in Keras [75]:

- Adam;
- Stochastic gradient descent (SGD);
- RMSprop;
- Adagrad;
- Adadelta;
- Adamax;
- Nesterov Adam (Nadam).



The sixth parameter was dropout, which is a technique to reduce overfitting in neural networks by preventing complex co-adaptations on training data [85]. Basically, a percentage of units (both hidden and visible) in a network are ignored during the training phase. This choice is random [86]. For example, if the dropout value is 0.3, it means that 30% of the units are going to be ignored. Figure 16 shows an example of a network without dropout (A) and a network with dropout (B).

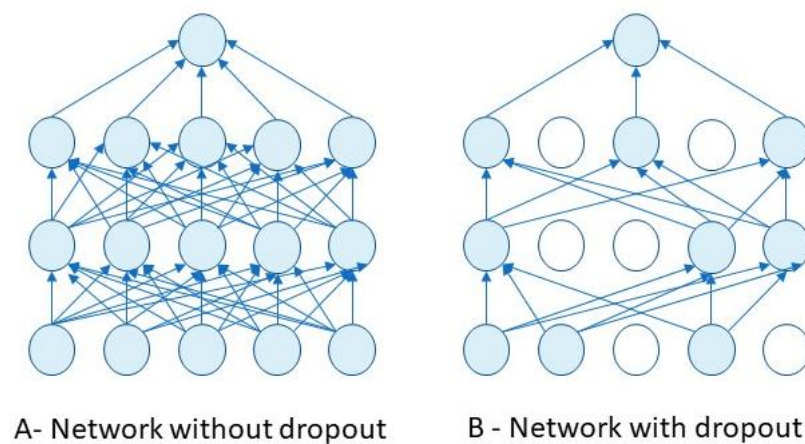


Figure 16 - Dropout example

The seventh parameter was softmax temperature. Softmax is an activation function. Activation functions introduce non-linear properties to the networks and their main purpose is to convert the input signal of a networks' node to the output signal. In Keras there are some distinct activation functions but softmax is the only one that produced results for the models in test [87]. Softmax function normalizes the candidates at each iteration of the network based on their exponential values by ensuring that the outputs of the network are all between 0 and 1 at every timestep.

Temperature is a hyperparameter of neural networks used to control the randomness of predictions by scaling the logits before applying softmax. The following formula shows the calculations done by softmax, where  $q$  is the probability vector,  $T$  is temperature and  $z$  is the class probabilities with logit produced by the network.

$$q_i = \frac{e^{\frac{z_i}{T}}}{\sum_j e^{\frac{z_j}{T}}}$$

The eighth parameter used in the tests is not a hyperparameter. However, the number of SMILES in the training sub-dataset is an important aspect that influences the training and consequently the results obtained.

Finally, the number of layers in each model, that is not a hyperparameter too. It is a situation that had to be taken into consideration. Despite the fact that the 8 models chosen were already using 1 and 2 layers, in one point of the test there was the need of trying another number of layers.

### 3.5 Methodology

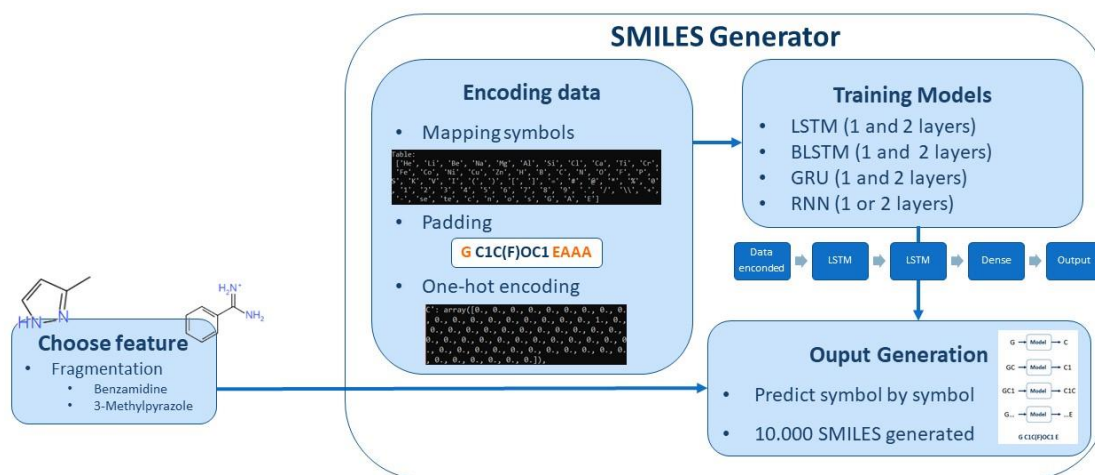


Figure 17 - SMILES generator component

The methodology used to generate SMILES can be divided into three main parts: encoding data, the training model and the output generation. The first two parts are done

twice, one for each sub-dataset. Furthermore, there are a secondary element integrated in the output generation step, which also allows choosing features for the SMILES generation. A scheme of this methodology is shown in the SMILES generator component in Figure 17.

## Encoding Data

Firstly, the program reads the sub-dataset, then the encoding data starts. This segment of the SMILES generation process has three primary steps: mapping symbols, padding and one-hot encoding.

Mapping symbols rely on a dictionary that contains all of the possible characters included in SMILES. Figure 18 shows this dictionary constituted of atoms, special characters and padding characters, which will be explained in detail. Each SMILES existing in the dataset is read and saved in a string format, and at that point each different atom or special character is tokenized into a char type. For example, in Figure 19 it is possible to observe that the SMILES presented was decomposed into atoms and special character, carbon represented by a “C” is an atom, chlorine represented by “Cl” is another atom, and a parenthesis, represented by “(” is a special character.

### Dictionary

```
atoms = [
    'H', 'He', 'Li', 'Be', 'B', 'C', 'N', 'O', 'F', 'Ne', 'Na', 'Mg', 'Al', 'Si', 'P', 'S', 'Cl', 'Ar', 'K', 'Ca', 'Sc', 'Ti', 'V', 'Cr', 'Mn',
    'Fe', 'Co', 'Ni', 'Cu', 'Zn', 'Ga', 'Ge', 'As', 'Se', 'Br', 'Kr', 'Rb', 'Sr', 'Y', 'Zr', 'Nb', 'Mo', 'Tc', 'Ru', 'Rh', 'Pd', 'Ag',
    'Cd', 'In', 'Sn', 'Sb', 'Te', 'I', 'Xe', 'Cs', 'Ba', 'La', 'Ce', 'Pr', 'Nd', 'Pm', 'Sm', 'Eu', 'Gd', 'Tb', 'Dy', 'Ho', 'Er', 'Tm',
    'Yb', 'Lu', 'Hf', 'Ta', 'W', 'Re', 'Os', 'Ir', 'Pt', 'Au', 'Hg', 'Tl', 'Pb', 'Bi', 'Po', 'At', 'Rn', 'Fr', 'Ra', 'Ac', 'Th', 'Pa', 'U',
    'Np', 'Pu', 'Am', 'Cm', 'Bk', 'Cf', 'Es', 'Fm', 'Md', 'No', 'Lr', 'Rf', 'Db', 'Sg', 'Bh', 'Hs', 'Mt', 'Ds', 'Rg', 'Cn', 'Uut',
    'Uuq', 'Uup', 'Uuh', 'Uus', 'Uuo'
]
special = [
    '(', ')', '[', ']', '=', '#', '@', '*', '%', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '.', '/', '\\', '+', '-', 'se', 'te', 'c', 'n', 'o', 's'
]
padding = ['G', 'A', 'E']
```

Figure 18 - Dictionary of existing characters

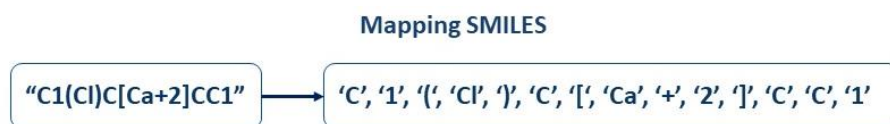


Figure 19 - Mapping SMILES

The second step is padding the SMILES, for that it is necessary to count the length of each SMILES and keep the length of the longest one. For example, as shown in Figure 20, the longest size is 9, then, character "G", meaning "go" is added at the start of each SMILES, and character "E" is added to the end, meaning "end" (Figure 20 - B). After adding these two letters, obviously, the length of the longest string increases from 9 to 11. At last, every SMILES in the dataset is normalized. To achieve this, it is needed to add "A"s after the "E", so that all the SMILES can have the same size. In the example, it is observed that the SMILES "CC", in step B, just has as size 4 and need to reach 11, so 7 "A"s were added.



Figure 20 - Padding SMILES. A – Count the length of SMILES. B – Add "G" and "E". C – Normalize the length and padding with character "A"

The third step is one-hot encoding. In this step, each character in each SMILES will be transformed into a one-hot encoded array. For example, in Figure 21, the SMILES "CF(Cl)C" has as its first character the "G", so it is necessary to search for "G" in the dictionary, keep its position and then generate an array with zeros, and in the position saved will be a "1". The same process is to be done for all the characters of the SMILES. At the end, there is an array with all the characters encoded, ready to use as training set to the networks.

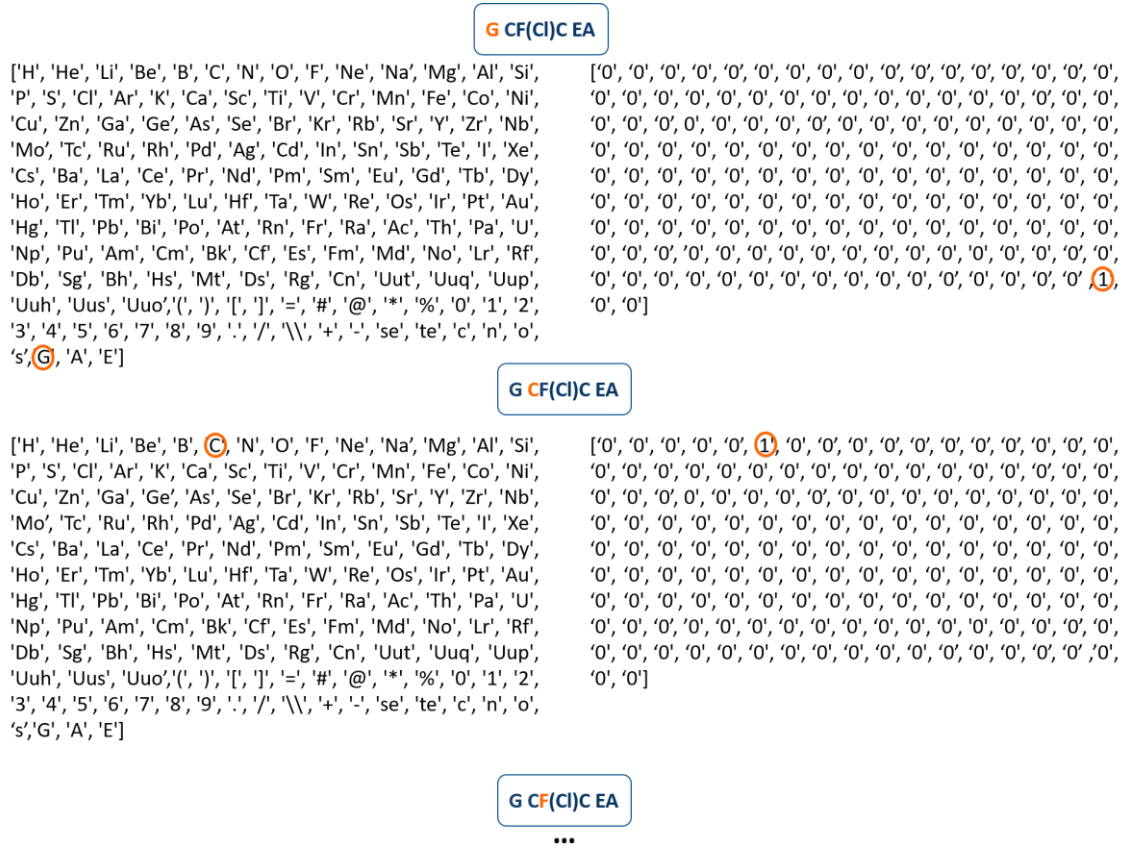


Figure 21 - One-hot encoding

### Training Models

After these three steps of the encoding data part, an array is ready to feed the models. There are 8 models used in this dissertation, constituted by 4 ANNs: RNN, LSTM, GRU and BLSTM. Each of them was used with 1 and 2 layers, as it is possible to observe in Figure 22.

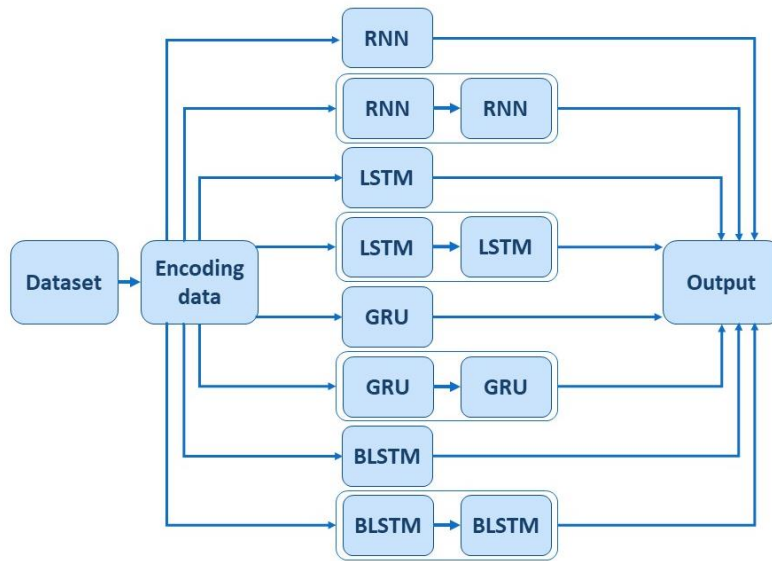


Figure 22 - Models trained

The models are built with the tool Keras, where the layers are already implemented, and it is only necessary to choose some parameters or choose the default values. For example, Figure 23 shows the layers existing in the model using LSTM with 2 layers, and Figure 24 shows one simple implementation of that model. In this example, there are 2 LSTM layers and one Dense layer. Dense layer is a linear operation which every input is connected to every output by a weight [88]. The other models are implemented in the same way, it is just necessary to choose the number of layers, 1 or 2 and the type of ANN.



Figure 23 - Model using LSTM with 2 layers

```

model = Sequential()
model.add(LSTM(50, input_shape = (49,1), dropout=0.3, return_sequences=True))
model.add(LSTM(50, input_shape = (49,1), dropout=0.3, return_sequences=True))
model.add(Dense(46, activation='softmax'))
model.compile(loss= 'categorical_crossentropy', optimizer='adam')

```

Figure 24 - Example of the code of the implementation of the model in Figure 23 with Keras

## Output Generation

After the training done by the networks in the chosen model, it is necessary to choose the strategy to predict the SMILES. In this dissertation, the strategy chosen was to predict symbol by symbol, which means that, in each iteration, a symbol is predicted depend on the previously predicted symbol. Figure 25 shows an example of this process.

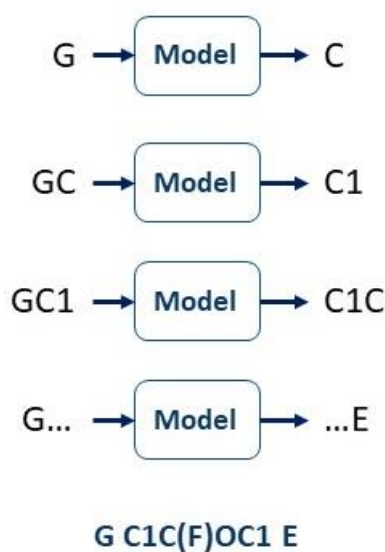


Figure 25 - Prediction symbol by symbol

## Fragmentation Growing Procedure

The methodology explained already does not use an oriented generation of SMILES. However, the fragmentation growing procedure adds this feature to the methodology. The

procedure consists in choosing a compound and make it part of all the SMILES generated in that run. For example, in Figure 26 the compound chosen to be the fragment was 3-Methylpyrazole. SMILES A and B are two examples of generated SMILES using this fragment. In that case, the fragment is at the beginning of the generated SMILES, but, depending on the fragments and its features, it could be in a different position.

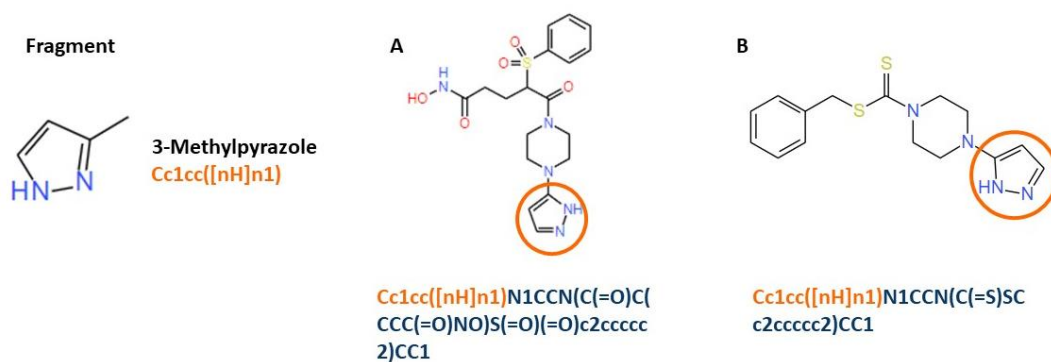


Figure 26 - Fragmentation growing procedure

### 3.6 Validation

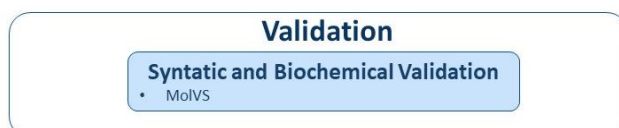


Figure 27 - Validation component

The SMILES generated by the model proposed had to be syntactically and biochemically validated (Figure 27), in order to evaluate the model, and study the set of ideal parameters, which generate the major percentage of valid possible SMILES. To achieve this goal, a tool named MoIVS was chosen. This tool, written in Python, provides very convenient functionalities for molecular standardization and validation. MoIVS was released under the MIT License, but it is allowed for commercial use, modifications, distribution, sublicensing



and private use. Furthermore, MolVS is built using the RDKit chemical framework [89], which is an open source toolkit for cheminformatics with a huge set of functions in this field [90].

In order to use this tool, it was necessary to install it, but also Python. It is simple to use tool, and it is well documented. In generated SMILES validation process, it was just necessary to use a function called “`validate_smiles()`”, as observed Figure 28 shows. In the examples shown, this tool worked, first with benzene, evaluating it as a valid SMILE, and then, secondly evaluating “`CCC1`” as the invalid SMILES that it is.

```
>>> from molvs import validate_smiles
>>> validate_smiles('C1=CC=CC=C1')
['INFO: [FragmentValidation] benzene is present']
>>>
>>> validate_smiles('CCC1')
[19:31:27] SMILES Parse Error: unclosed ring for input: 'CCC1'
['ERROR: [IsNoneValidation] Molecule is None']
```

Figure 28 - Use of MolVS to validate the SMILES

According to the chosen tool, a SMILES is identified as invalid when [91]:

- RDKit failed to parse the input format;
- The molecule has no atoms;
- 1,2-dichloroethane is present;
- Certain fragments are present;
- The system it is not overall neutral;
- The molecule contains isotopes.

To understand these set of rules some tests were made with valid and invalid SMILES with different types of errors. The dataset chosen, and described in section 3.1, was validated with this tool. All the SMILES followed this procedure, all were tested for validation.



# Chapter 4

## Results and Discussion

This chapter contains the results obtained from the tests done with the methodology implemented in this thesis, and their discussion.

The main purpose of these tests was to evaluate the set of parameters that reaches the higher ratio of valid SMILES. In each set of tests, only one parameter evaluated, therefore it was possible to obtain the ideal value for each parameter selected.

The tests are divided into four phases, since it was possible to, gradually, exclude the least efficient models. In other words, the tests started with 8 models and ended up with just the best model with the best parameters.

Furthermore, after the best model was selected, it was possible to test the fragmentation growing procedure.

### 4.1 Phase 1

As it was explained in the previous chapter, there were 8 models in test, which used 4 different artificial neural networks, RNN, LSTM, GRU and Bidirectional, as it is presented in Figure 29, each model were attributed an ID. Each type was implemented with 1 and 2 layers.

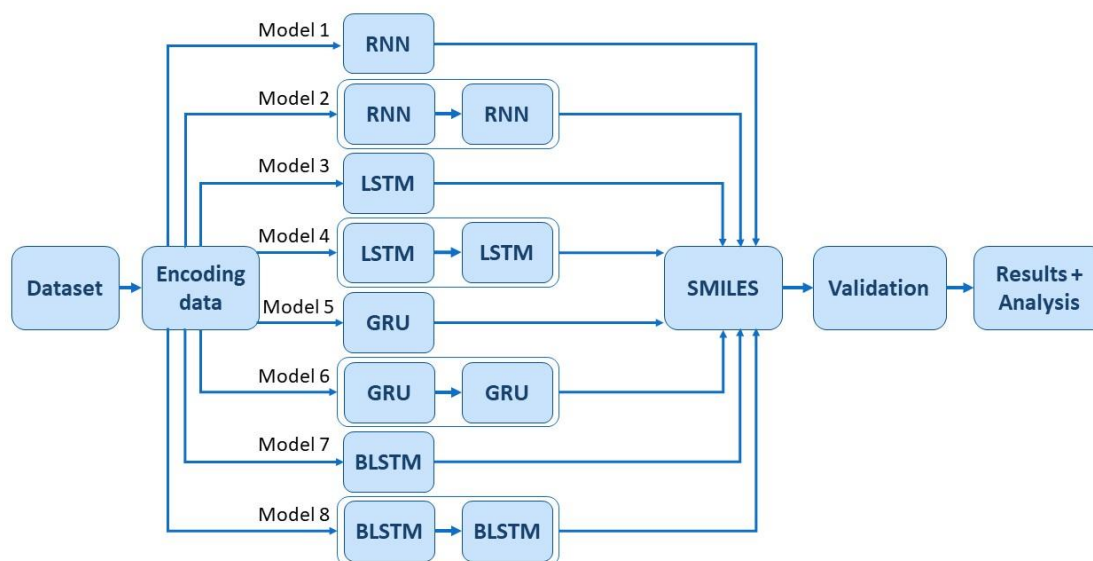


Figure 29 - Models in test – Phase 1

Fi

These 8 models were tested with a set of initial parameters and then the set of tests done to choose the ideal value for the parameter number of epochs for the sub-dataset 1.

#### 4.1.1 Initial Parameters

To start the tests, it was necessary to choose a value for each parameter applied, that choices were done taking into consideration some of related works [18-20], as discussed in chapter 2.

The initial parameters chosen were:

- Number of epochs for the sub-dataset 1: 22;
- Number of epochs for the sub-dataset 2: 15;
- Batch-size for the sub-dataset 1: 256;
- Batch-size for the sub-dataset 2: 16;
- Optimizer: Adam;

- Dropout: 0.3;
- Softmax temperature: 1;
- Number of SMILES in the sub-dataset 1: 100,000;
- Number of SMILES in the sub-dataset 2: 100,000.

This first test was done with this set of parameters in order to have an initial idea of which were the results achieved by each of the 8 models studied.

Table 3 contains the percentage of valid and unique SMILES that were generated with the 8 different models in test. The “valid” SMILES are the SMILES that passed through the validation process (See more details in Section 3.6), while the “unique” SMILES are the generated SMILES that were not repeated. For example, for the model RNN – 1 Layer, there were generated 10,000 SMILES, 1.55% were valid and 1.53% were unique. The results are all expressed in percentage, although all the tests performed in this study generated 10,000 SMILES.

Analysing Table 3, it is possible to observe that the models 1 and 2 reached low results, such as 1.55% and 2.50% of valid SMILES, and 1.53% and 2.23% of unique SMILES, respectively. The models 3 and 4 reached percentages of valid SMILES of 60.99% and 71.03%, and 60.98% and 71.02% of unique SMILES, respectively. Similarly, the models 5 and 6 reached values such as 60.79% and 67.75% of valid SMILES, and 60.79% and 67.74% of unique SMILES, respectively. Finally, the models, 7 and 8 reached results of 73.97% and 1.54% of valid SMILES, and 0.19% and 0.04% of unique SMILES, respectively.

Considering Table 3 and Figure 30, it was possible to understand that the models tested, except model 7 and 8, have similar percentages of valid SMILES and unique SMILES; which means that there were a reduced number of repeated SMILES. Conversely, in models 7 and 8, almost all of the generated SMILES were equal, since the percentage of unique SMILES is almost zero. These models correspond to the models that use BLSTM, which should be an improvement of LSTMs, but in this case and with the pre-processing done, this network did not present good results. The main incompatibility of BLSTM in this methodology is the padding, since BLSTM read forward and backward and because of the characters “G”, “E” and “A”, it will very often predict one of those characters and the SMILES will be completed. This fact justifies the redundancy of the SMILES generated.

Comparing the efficiency of the models, models 1 and 2 presented very low percentages of SMILES generated. These models use RNN, which is the simpler network in test and lose the context of the training data unlike the other three type of tested networks.

Models 3, 4, 5 and 6 presented much higher percentage of valid and unique SMILES than the other four models and seem to be the more consistent models.

Table 3 - Results with the initial parameters (%)

ID	Models	Valid (%)	Unique (%)
1	RNN - 1 Layer	1.55	1.53
2	RNN - 2 Layers	2.50	2.23
3	LSTM - 1 Layer	60.99	60.98
4	LSTM - 2 Layers	71.03	<b>71.02</b>
5	GRU - 1 Layer	60.79	60.79
6	GRU - 2 Layers	67.75	67.74
7	BLSTM - 1 Layer	<b>73.97</b>	0.19
8	BLSTM - 2 Layers	1.54	0.04

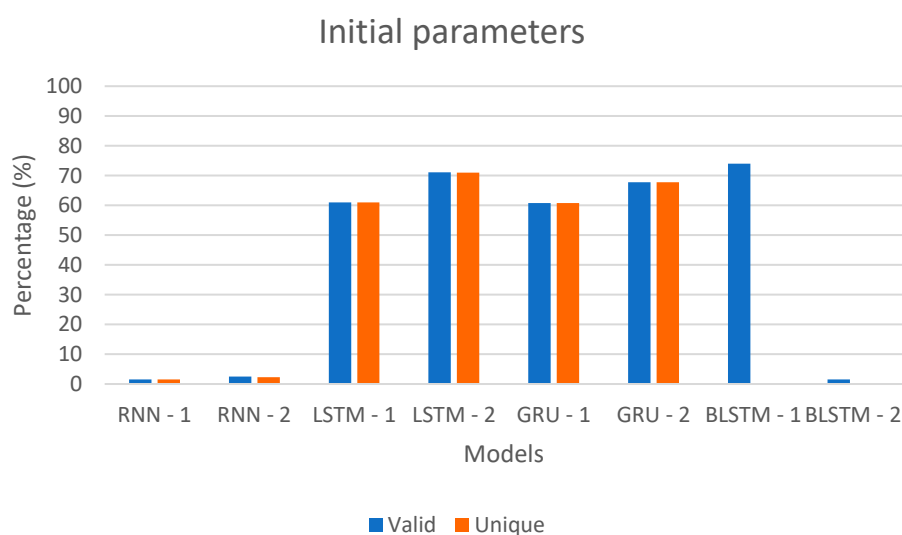


Figure 30 - Results with the initial parameters

### 4.1.2 Number of Epochs for the Sub-dataset 1

The first parameter tested was the number of epochs for the sub-dataset 1.

In the tests performed in the 8 models, in order to reach the ideal value for this parameter, the number of epochs for the sub-dataset 1 used were 4, 8, 12, 16, 20, 24 and 28. For example, if the number of epochs was 4, the sub-dataset 1 would be used 4 times to train the network.

Table 4 shows the results of the 8 models for the set of values that were chosen to be tested for this parameter. The 1<sup>st</sup> and 2<sup>nd</sup> models reached 3.24% and 2.32% of valid SMILES, respectively, when the number of epochs was 8. The 3<sup>rd</sup> and 8<sup>th</sup> models had their best results when the number of epochs was 20, 62.56% and 20.57% of valid SMILES, respectively. Finally, 4<sup>th</sup>, 5<sup>th</sup>, 6<sup>th</sup> and 7<sup>th</sup> models presented their best result at 24 epochs, 77.59%, 67.14%, 67.42% and 90.55%, respectively.

Considering Table 4 and Figure 31 to Figure 38, it was possible to observe that model 1 and 2 present the worst values of valid SMILES, while models 7 and 8, although the values of valid SMILES were relevant, the number of unique SMILES was also really low. Conversely, models 3 to 6 have shown the best percentages. These results are aligned with previous subsection and share the same reasons. Thus, to choose the best value for this parameter, models 1, 2, 7 and 8 were ignored, since the generated SMILES were almost all invalid or equal between them. Then, analysing the models 3 to 6, the choice had to be between 20 and 24 epochs. However, the best number seems to be 24 epochs, and for now, the best model appears to be model 4.

Table 4 – Results for the parameter number of epochs for the sub-dataset 1 (%)

		Models							
	ID	1	2	3	4	5	6	7	8
No of epochs		RNN 1 Layer	RNN 2 Layers	LSTM 1 Layer	LSTM 2 Layers	GRU 1 Layer	GRU 2 Layers	BLSTM 1 Layer	BLSTM 2 Layers
4	Valid (%)	2.48	0.34	58.37	76.80	63.85	64.91	57.94	1.78
	Unique (%)	2.12	0.33	58.36	76.79	63.85	64.88	0.19	0.08
8	Valid (%)	<b>3.24</b>	<b>2.32</b>	59.68	76.99	62.19	58.54	94.17	0.01
	Unique (%)	2.57	2.16	59.68	76.99	62.18	58.53	0.24	0.01
12	Valid (%)	1.12	0.32	60.19	73.24	61.91	59.18	66.58	1.09
	Unique (%)	1.11	0.31	60.19	73.23	61.91	59.18	0.25	0.06
16	Valid (%)	2.87	0.66	61.80	75.81	61.63	66.37	65.78	0.03
	Unique (%)	2.62	0.63	61.80	75.80	61.62	66.37	0.17	0.03
20	Valid (%)	1.27	2.15	<b>62.56</b>	72.90	63.37	56.22	62.04	<b>20.57</b>
	Unique (%)	1.15	1.98	62.55	72.90	63.36	65.22	0.26	0.08
24	Valid (%)	2.22	0.49	58.54	<b>77.59</b>	<b>67.14</b>	<b>67.42</b>	<b>90.55</b>	1.96
	Unique (%)	2.09	0.45	58.54	77.59	67.14	67.41	0.31	0.06
28	Valid (%)	1.79	0.25	59.75	75.21	64.77	59.73	33.78	1.34
	Unique (%)	1.75	0.24	59.75	75.21	64.77	59.73	0.20	0.05



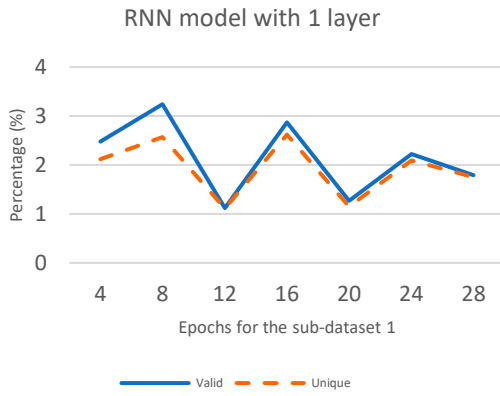


Figure 31 - Epochs for the sub-dataset 1 - RNN - 1 layer

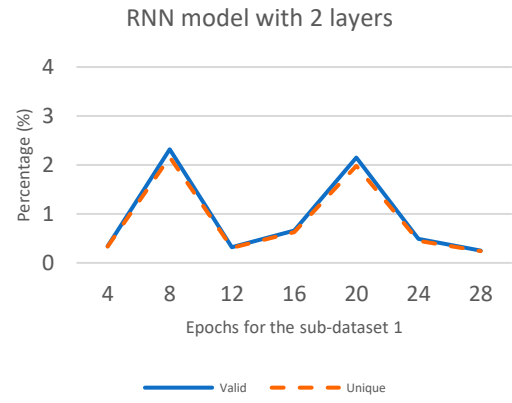


Figure 32 - Epochs for the sub-dataset 1 - RNN - 2 layers

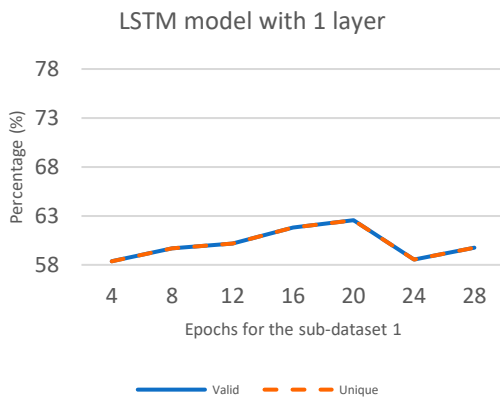


Figure 33 - Epochs for the sub-dataset 1 - LSTM - 1 layer

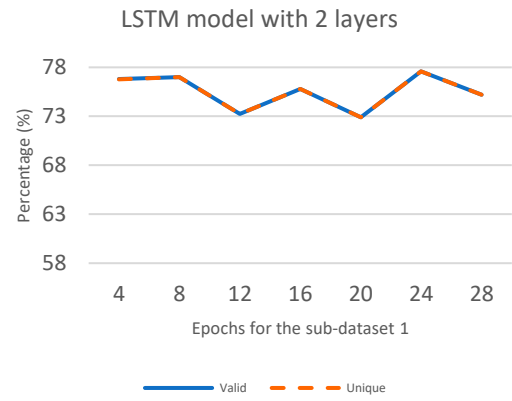


Figure 34 - Epochs for the sub-dataset 1 - LSTM - 2 layers

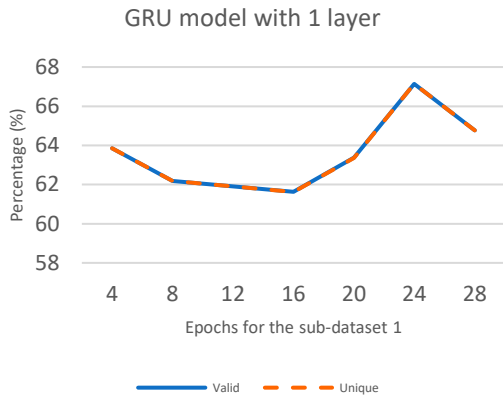


Figure 35 - Epochs for the sub-dataset 1 - GRU - 1 layer

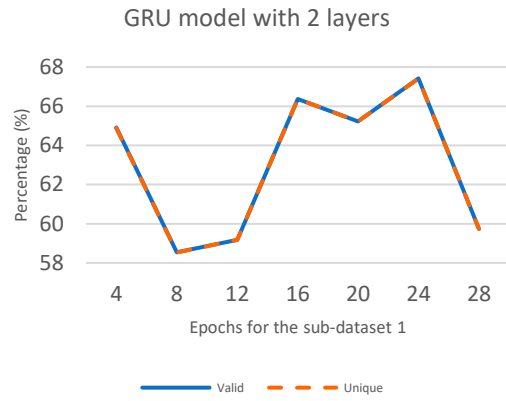


Figure 36 - Epochs for the sub-dataset 1 - GRU - 2 layers

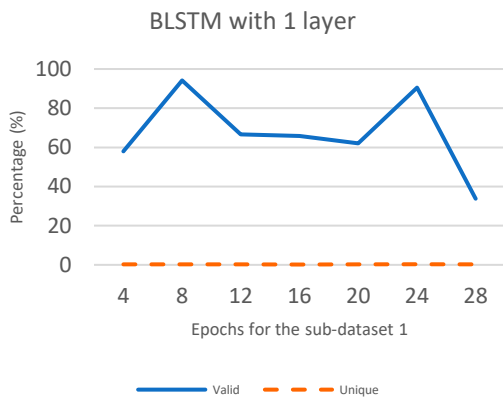


Figure 37 - Epochs for the sub-dataset 1 -BLSTM - 1 layer

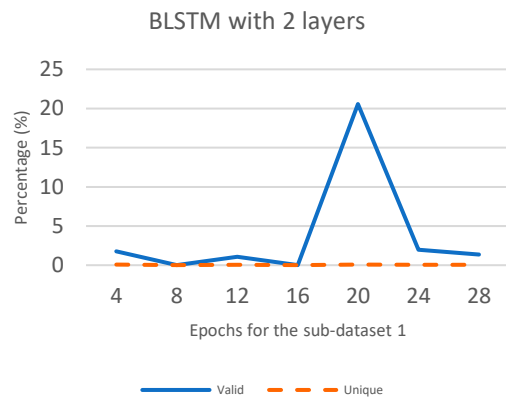


Figure 38 - Epochs for the sub-dataset 1 - BLSTM - 2 layers

After the previous set of tests and its analysis, the worst models were excluded from the study, and were not included in following tests performed. The results of models 1, 2, 7 and 8 were very disappointing and much lower than the other 4 models. At this stage of the study, the models using RNN and Bidirectional were eliminated, and the models that stayed in test were the ones using LSTM and GRU.

## 4.2 Phase 2

Figure 39 represents the models in test in phase 2. These models were the ones that reached best results in phase 1.

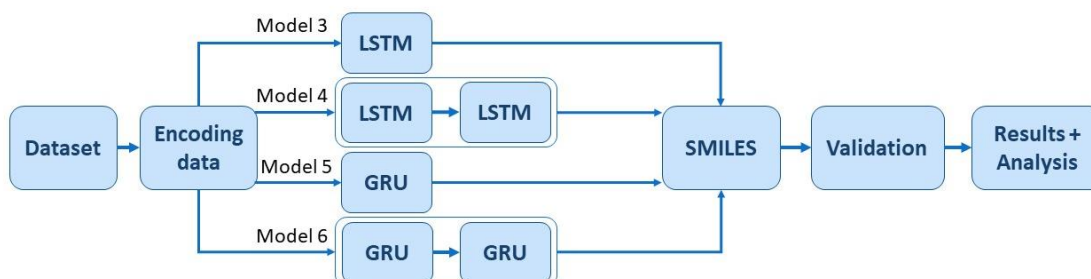


Figure 39 - Models in test – Phase 2

These four models were tested to reach the ideal values for the parameters number of epochs for the sub-dataset 2, batch-size for the sub-dataset 1 and batch-size for the sub-dataset 2.

### 4.2.1 Number of Epochs for the Sub-dataset 2

The second parameter tested was the number of epochs for the sub-dataset 2. The values tested in the remaining models were 10, 15 and 20.

Table 5 presents the results of the tests performed in order to choose the ideal value of epochs for the sub-dataset 2. The model 3 achieved 61.15% of valid SMILES with 20 epochs, and models 4, 5 and 6 achieved 77.59%, 67.14% and 67.42% of valid SMILES with 15 epochs, respectively

Analysing Table 5 and Figure 40 to Figure 43, the value selected, as the best value, were 15 epochs for the sub-dataset 2, which means that the sub-dataset 2 will be passed through 15 times to train the network.

The number of epochs needed to train the network with the sub-dataset 2 is smaller than with the sub-dataset 1, since the model was already trained, and 15 epochs were enough to reach the best result.

Table 5 - Results for the parameter number of epochs for the sub-dataset 2 (%)

Models					
	ID	3	4	5	6
<b>No epochs</b>		LSTM 1 Layer	LSTM 2 Layers	GRU 1 Layer	GRU 2 Layers
<b>10</b>	Valid (%)	56.89	72.43	60.89	64.91
	Unique (%)	56.88	72.40	60.89	64.91
<b>15</b>	Valid (%)	58.54	<b>77.59</b>	<b>67.14</b>	<b>67.42</b>
	Unique (%)	58.54	77.59	67.14	67.41
<b>20</b>	Valid (%)	<b>61.15</b>	74.80	59.78	63.07
	Unique (%)	61.15	74.80	59.77	63.06

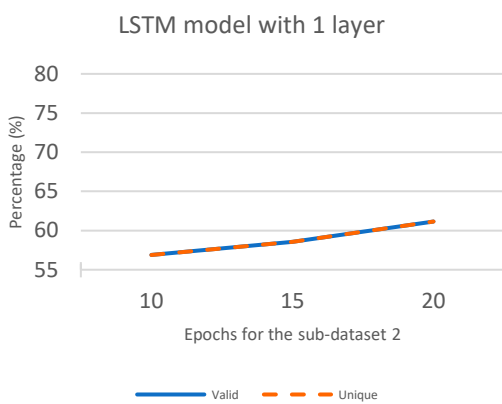


Figure 40 - Epochs for the sub-dataset 2 - LSTM - 1 layer

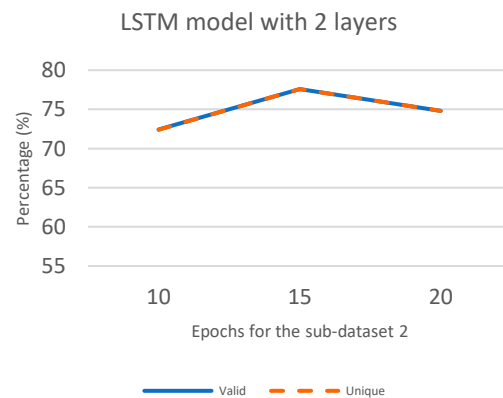


Figure 41 - Epochs for the sub-dataset 2 - LSTM - 2 layers

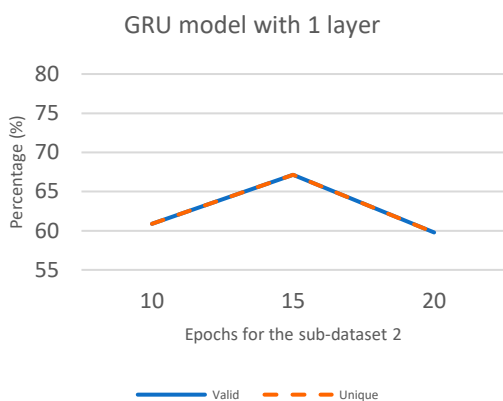


Figure 42 - Epochs for the sub-dataset 2 - GRU - 1 layer

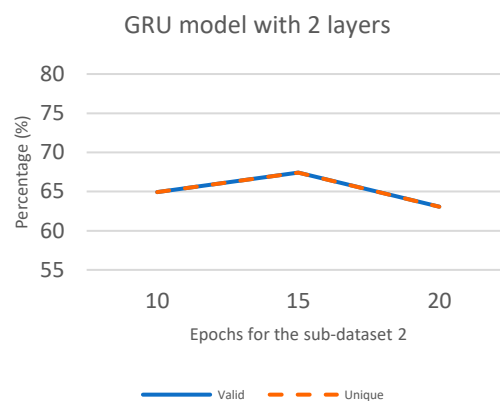


Figure 43 - Epochs for the sub-dataset 2- GRU - 2 layers

#### 4.2.2 Batch-size for the Sub-dataset 1

The third parameter analysed was the batch size for the sub-dataset 1.

In this subsection, the values tested were 64, 128, 256 and 512 for the model using LSTM and GRU.

Table 6 showed that model 3 archived 64.52% of valid SMILES when the batch size was equal to 64, model 4 and model 6 archived 78.14% and 69.12% of valid SMILES, respectively, when the batch was 128. Moreover, model 5 archived 67.14% of valid SMILES when the batch size selected was 256.

Analysing Table 6 and Figure 44 to Figure 47, it was possible to observe that the best result obtained was using the model 4, while the 2<sup>nd</sup> best model was model 6. As it happened in the previous tests, the best model are the ones with 2 layers. Thus, the value chosen for this parameter in study was 128, since the bests results were obtained with this value.

Table 6 – Results for the parameter batch size for the sub-dataset 1 (%)

Models					
	ID	3	4	5	6
<b>Batch size</b>		LSTM 1 Layer	LSTM 2 Layers	GRU 1 Layer	GRU 2 Layers
<b>64</b>	Valid (%)	<b>64.52</b>	76.34	62.64	39.38
	Unique (%)	64.52	76.32	62.64	39.38
<b>128</b>	Valid (%)	60.65	<b>78.14</b>	62.80	<b>69.12</b>
	Unique (%)	60.65	78.13	62.80	69.12
<b>256</b>	Valid (%)	58.54	77.59	<b>67.14</b>	67.42
	Unique (%)	58.54	77.59	67.14	67.41
<b>512</b>	Valid (%)	60.10	69.11	61.44	62.39
	Unique (%)	60.10	69.11	61.44	62.37

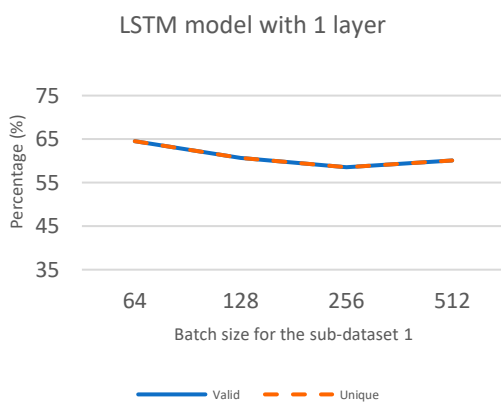


Figure 44 -Batch size for the sub-dataset 1 - LSTM - 1 layer

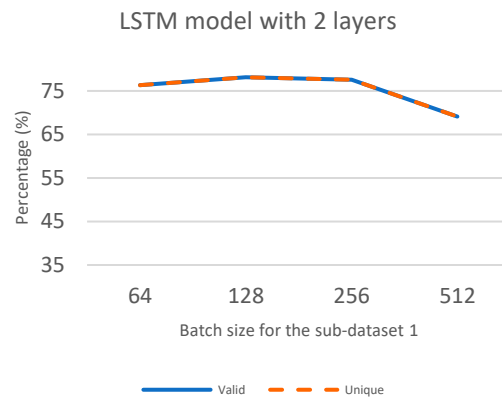


Figure 45 - Batch size for the sub-dataset 1 - LSTM - 2 layers

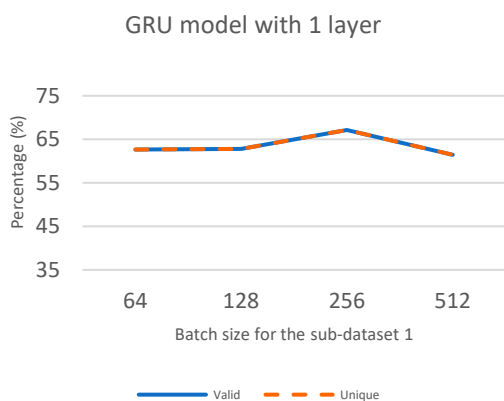


Figure 46 - Batch size for the sub-dataset 1 -  
GRU - 1 layer

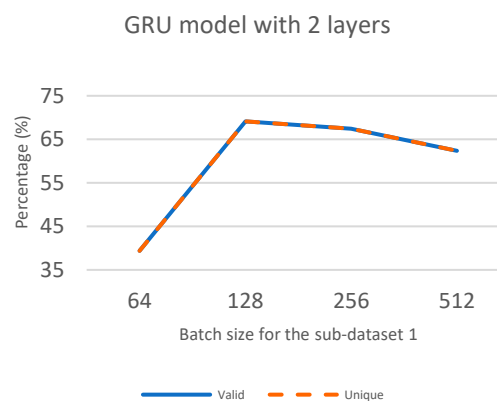


Figure 47 - Batch size for the sub-dataset 1 -  
GRU - 2 layer

### 4.2.3 Batch-size for the Sub-dataset 2

The fourth parameter analysed was the batch size for the sub-dataset 2. All the other parameters were maintained, and the batch size values tested were 8, 16, 32 and 64.

Table 7 shows the percentages of generated SMILES in each model tested. Once again valid and unique SMILES percentages are similar per each model. Therefore, only the percentage of valid SMILES was observed. Analysing these results, the best percentages obtained per each model were: 60.78% of valid SMILES by model 8, when the batch size was 8; 78.14% and 69.12% of valid SMILES for model 4 and model 6, respectively, using batch size 16; and 63.41% of valid SMILES for model 5, when the batch size was 32.

Considering Table 7 and Figure 48 to Figure 51, it was possible to observe that the best result obtained was again using the model 4, while the 2<sup>nd</sup> best model was model 6. Thus, the value chosen for this parameter in study was 16, since the best results were obtained with this batch size.

The ideal value for the batch-size of the sub-dataset 2 is smaller than for the batch-size for the sub-dataset 1. That means that the sub-dataset 2 should be split into less parts, since the network was already trained, the second training do not need be that precise.

Table 7 - Results for the parameter batch size for the sub-dataset 2 (%)

Models					
	ID	3	4	5	6
<b>Batch size</b>		LSTM 1 Layer	LSTM 2 Layers	GRU 1 Layer	GRU 2 Layers
<b>8</b>	Valid (%)	<b>60.78</b>	76.76	59.07	23.66
	Unique (%)	60.78	76.74	59.07	23.66
<b>16</b>	Valid (%)	60.65	<b>78.14</b>	62.80	<b>69.12</b>
	Unique (%)	60.65	78.13	62.80	69.12
<b>32</b>	Valid (%)	61.05	74.16	<b>63.41</b>	67.76
	Unique (%)	61.05	74.15	63.40	67.75
<b>64</b>	Valid (%)	56.84	75.41	62.82	68.29
	Unique (%)	56.83	75.40	62.81	68.28

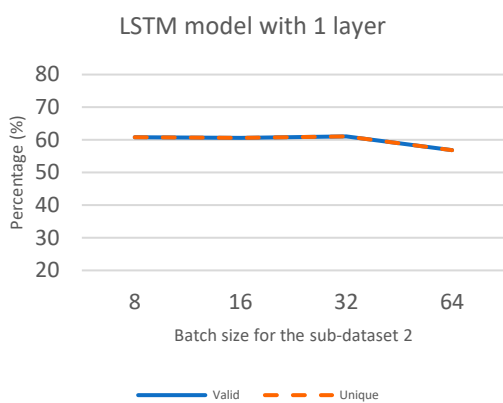


Figure 48 - Batch size for the sub-dataset 2 - LSTM - 1 layer

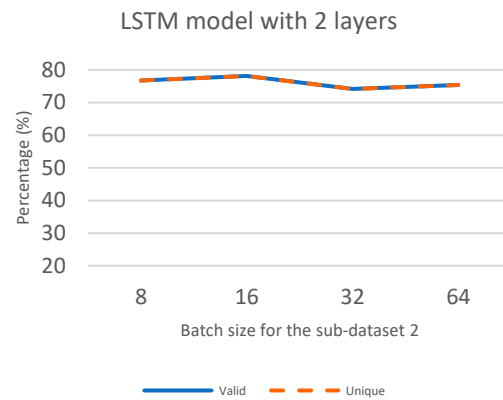


Figure 49 - Batch size for the sub-dataset 2 - LSTM - 2 layers



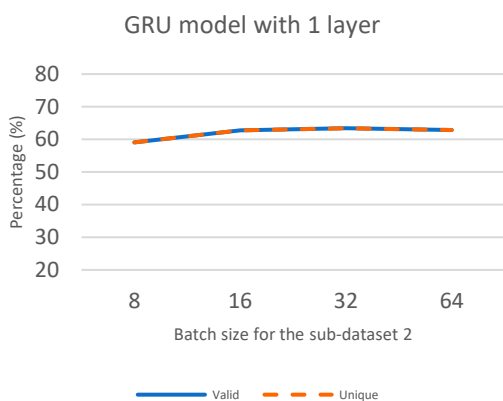


Figure 50 - Batch size for the sub-dataset 2 - GRU - 1 layer

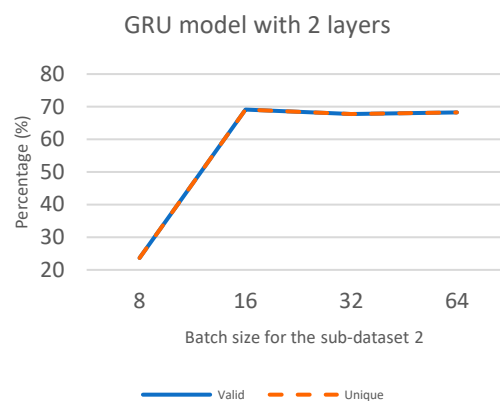


Figure 51 - Batch size for the sub-dataset 2 - GRU - 2 layer

At this point, it was possible to conclude that models 4 and 6 were always the best models in test, which correspond to the models with 2 layers. Therefore, model 3 and 5 were now on excluded from the study.

### 4.3 Phase 3

The remaining models at this phase of the study, phase 3, are shown in Figure 52.

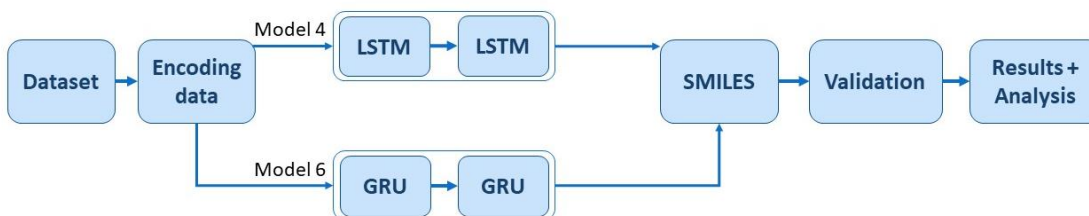


Figure 52 - Models in test – Phase 3

The models 4 and 6 were tested to reach to the ideal values of the parameters optimizers, dropout and softmax temperature.

### 4.3.1 Optimizers

The fifth parameter analysed during this study was the optimizer.

shows the results obtained by the last two models in study using the optimizers selected. Model 4 showed the best results. Two optimizers presented similar results, 78.14% and 78.69% of valid SMILES for Adam and RMSprop, respectively. Model 6 reached a 69.12% of valid SMILES, with Adam optimizer.

Thus, taking in consideration, Figure 53 and Figure 54, where model 4 presented the best results, and the percentages of valid SMILES very similar for both optimizers RMSprop and Adam, both were selected to the next steps. Regarding, model 6 only Adam will be considered, which means that instead of 2 models to test the next parameters, there will be used 3 models.

The results for the optimizers tested were consistent, where the model 4 presents result better results than 6. However, SGD optimizer is the worse in these tests, which means that this optimizer is not successful in this type of model. Furthermore, in Nadam optimizer it is visible a huge different between the two types of networks and that has to be related with the two differences between LSTM and GRU, the gates and the persistent cell state.

Table 8 - Results for the parameter optimizer (%)

Optimizers									
ID	Models		Adam	SGD	RMSprop	Adagrad	Adadelata	Adamax	Nadam
4	LSTM 2 Layers	Valid (%)	78.14	0.21	<b>78.69</b>	39.96	57.43	71.31	66.95
		Unique (%)	78.13	0.21	78.67	39.95	57.42	71.30	66.94
6	GRU 2 Layers	Valid (%)	<b>69.12</b>	0.31	62.00	49.19	59.48	68.03	0.11
		Unique (%)	69.12	0.31	61.98	49.19	59.48	68.03	0.11

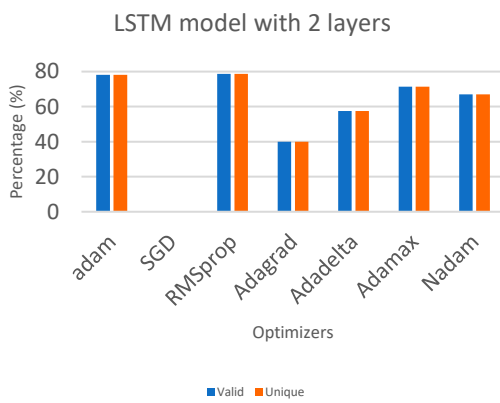


Figure 53 - Optimizers - LSTM - 2 layers

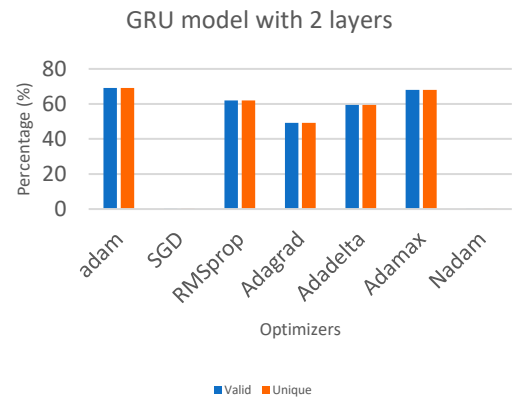


Figure 54 - Optimizers - GRU - 2 layers

### 4.3.2 Dropout

The sixth parameter analysed was dropout.

For these tests, there were used the values 0.1; 0.2; 0.3; 0.4 and 0.5 of dropout applied to the 3 models previously selected (subsection 4.3.2).

Table 9 shows the percentages of generated SMILES created by the models under study. Considering the most efficient results, model 4A archived 78.69% of valid SMILES when the dropout was 0.3, while the model 4B best result was using 0.2 of dropout, which gave rise to 78.57% of valid SMILES (similar value to 0.2 dropout, 78.14% of valid SMILES). Models 6A and 6B archived 67.09% and 72.25% of valid SMILES, respectively, when dropout was 0.1.

Analysing Table 9 and Figure 55 to Figure 58, the choice of the ideal dropout value for all models in study was difficult. Therefore, it has been decided to keep the best value of dropout for each model. Thus, each one of the 4 models tested in this step kept the value of dropout that made them reach the best result. However, model 4B presented approximated results for two values of dropout, 0.2 and 0.3, and in order to make sure that the best model would be achieved, the decision was to take them both in consideration for the next tests.

As it was already explained before, the dropout is the term used to designate the percentage of units (hidden or visible) that are dropped. The successful values of dropout were the smaller ones, since the percentage of units removed cannot be very significant because relevant data would be removed too.

Table 9 - Results for the parameter dropout (%)

Models					
	ID	4A	4B	6A	6B
<b>Drop out</b>		LSTM 2 Layer - Adam	LSTM 2 Layers - RMSprop	GRU 2 Layers - Adam	GRU 2 Layers - RMSprop
<b>0.1</b>	Valid (%)	77.53	77.34	<b>67.09</b>	<b>72.25</b>
	Unique (%)	77.50	77.34	67.07	72.22
<b>0.2</b>	Valid (%)	78.05	<b>78.57</b>	65.74	65.65
	Unique (%)	78.00	78.57	65.74	65.64
<b>0.3</b>	Valid (%)	<b>78.69</b>	78.14	62.00	69.12
	Unique (%)	78.67	78.13	61.98	69.12
<b>0.4</b>	Valid (%)	72.11	72.73	54.77	62.47
	Unique (%)	72.10	72.69	54.77	62.47
<b>0.5</b>	Valid (%)	67.87	68.79	53.73	54.61
	Unique (%)	67.84	68.79	53.72	54.61



Figure 55 - Dropout - LSTM - 2 layers - Adam

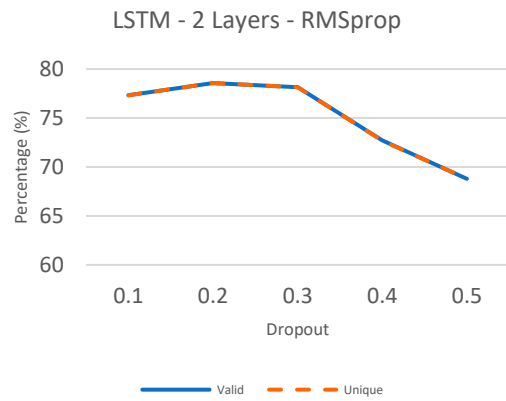


Figure 56 - Dropout - LSTM - 2 layers - RMSprop

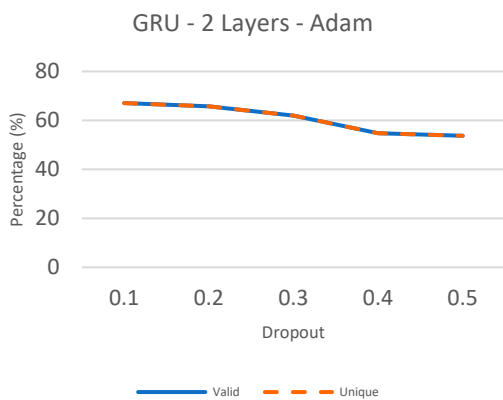


Figure 57 - Dropout - GRU - 2 layers - Adam

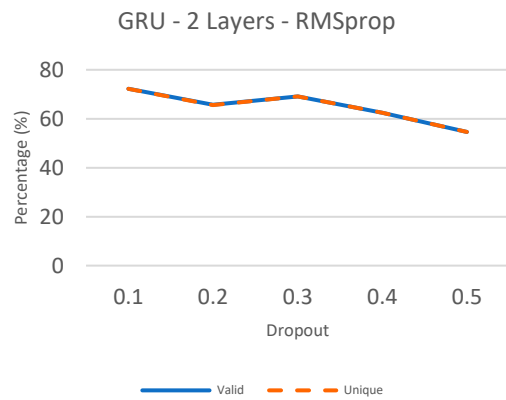


Figure 58 - Dropout - GRU - 2 layers - RMSprop

### 4.3.3 Softmax Temperature

The seventh parameter evaluated was the softmax temperature.

Softmax function normalizes, at each iteration, the candidates, based on their exponential values, by establishing that network's outputs are all between one and zero.

Temperature is a variable of softmax used to control the randomness of predictions by scaling the logits before applying softmax [87].

In this subsection, the values of temperature used were 0.3; 0.5, 0.8; 1. and 1.5.

Table 10 presents the results obtained by the models at different temperatures. Models 4A1, 4A2, 4B and 6A reached their best results when the temperature was set at 0.5, 95.36%, 95.01%, 97.43% and 94.47% of valid SMILES, respectively, while the model 6B achieved 94.14% of valid SMILES.

Considering Table 10 and Figure 59 to Figure 63, it was straightforward that the best temperature value was 0.5. Therefore, it is the best choice to use in the study. As the temperature increases the difference between valid and unique SMILES is smaller. However, from 0.5 the of valid and unique decreases significantly.

Table 10 - Results for the parameter Software temperature (%)

Models						
	ID	4A1	4A2	4B	6A	6B
<b>Softmax temperature</b>		LSTM 2 Layers - Adam 0.3	LSTM 2 Layers - Adam 0.2	LSTM 2 Layers - RMSprop 0.3	GRU 2 Layers - Adam 0.1	GRU 2 Layers - RMSprop 0.1
<b>0.3</b>	Valid (%)	89.80	92.29	97.06	93.21	<b>94.14</b>
	Unique (%)	56.93	63.03	41.59	50.98	46.03
<b>0.5</b>	Valid (%)	<b>95.46</b>	<b>95.01</b>	<b>97.43</b>	<b>94.47</b>	92.62
	Unique (%)	89.57	91.81	89.88	89.81	86.06
<b>0.8</b>	Valid (%)	86.50	89.94	89.77	85.98	81.54
	Unique (%)	86.43	89.79	89.55	85.89	81.43
<b>1.0</b>	Valid (%)	78.14	78.57	78.69	72.25	67.09
	Unique (%)	78.13	78.57	78.67	72.22	67.07
<b>1.2</b>	Valid (%)	54.29	62.27	61.97	53.57	50.67
	Unique (%)	54.28	62.27	61.97	53.57	50.67
<b>1.5</b>	Valid (%)	27.87	33.87	35.61	28.08	24.73
	Unique (%)	27.85	33.87	35.61	28.07	24.73



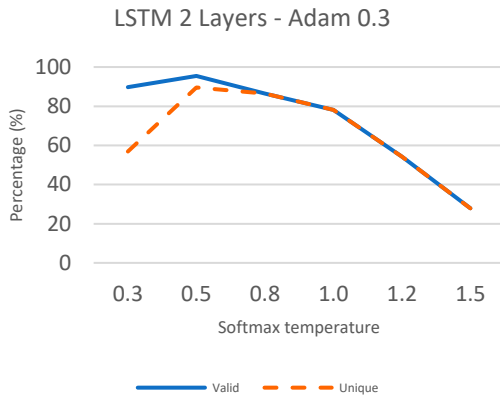


Figure 59 - Software temperature - LSTM - 2 layers - Adam - 0.3

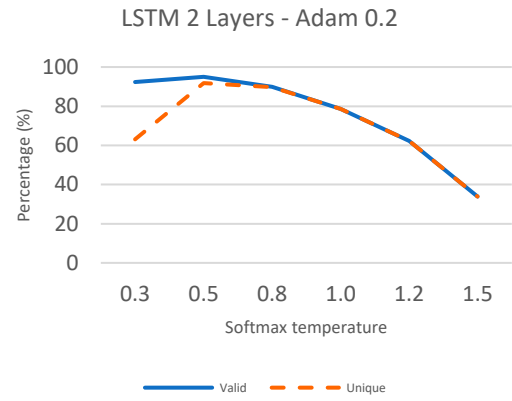


Figure 60 - Software temperature - LSTM - 2 layers - Adam - 0.2

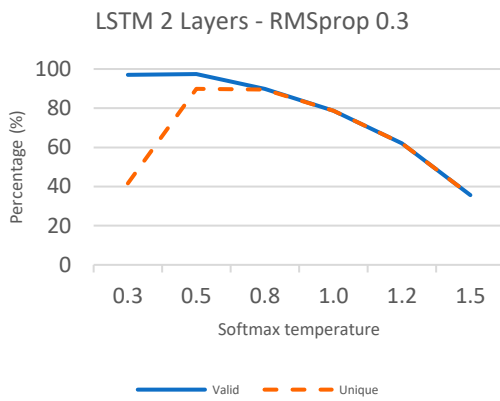


Figure 61 - Software temperature - LSTM - 2 layers - RMSprop - 0.3

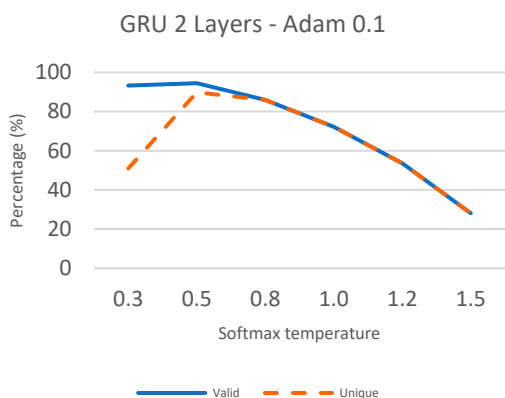


Figure 62 - Software temperature - GRU - 2 layers - Adam- 0.1

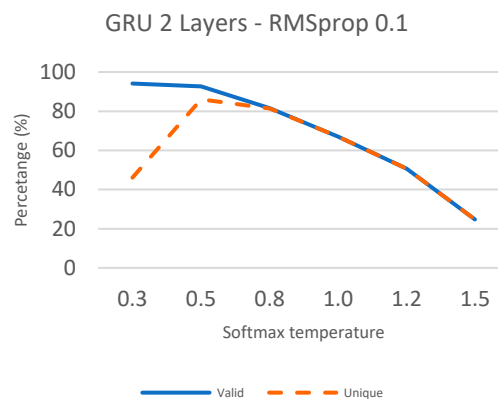


Figure 63 - Software temperature - GRU - 2 layers - RMSprop - 0.1

After the previous tests and its analysis, it was possible to observe a tendency that models using LSTM were always better than models using GRU. For this reason, model 5A and 5B were removed from the models in test.

## 4.4 Phase 4

At this point, there were just one original model being tested, the one that uses LSTM networks with 2 layers (Figure 64), even though there were 3 sets of parameters being used for this model.



Figure 64 - Models in test – Phase 4

The model still in test was used to test the sub-datasets size and the number of layers.

#### 4.4.1 Sub-datasets Size

The eighth and ninth parameters tested were the number of SMILES in the sub-dataset 1 and the sub-dataset 2. In the several tests already done, the number of SMILES in each sub-dataset were 100,000 SMILES, however in this specific test the number of SMILES in each sub-dataset increased to 240,000.

Analysing Table 11 and Figure 65, it was observed that the best result corresponded to the model 4A1, 98.04% of valid SMILES. The increase in the number of SMILES gave rise to a rise of valid SMILES from 95.46% to 98.04%, a great improvement. This fact means that a bigger dataset could improve the results, but due to lack of memory available in the machines, these tests could not be completed.

At this phase, the best sub model was 4A1, LSTM with 2 layers, with Adam optimizer and dropout equals to 0.3. That meant that the ideal parameters were set for the best model, and the best model was model 3.

Table 11 - Results for the parameter sub-datasets size (%)

Models			
ID	4A1	4A2	4B
	LSTM 2 Layers – Adam 0.3	LSTM 2 Layers – Adam 0.2	LSTM 2 Layers – RMSprop 0.3
<b>Valid (%)</b>	<b>98.04</b>	96.86	96.30
<b>Unique (%)</b>	93.38	89.74	90.69

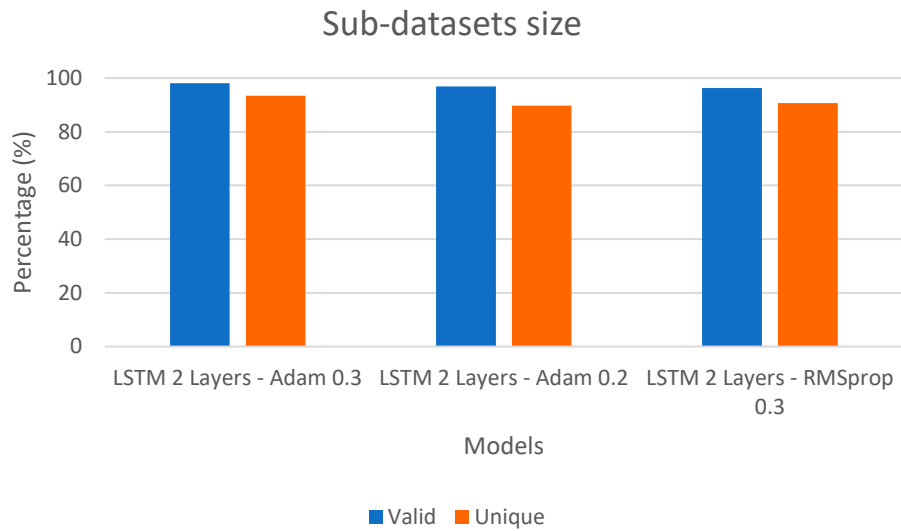


Figure 65 - Sub-datasets size - LSTM - 2 layers

#### 4.4.2 Layers

The best model, model 4, was chosen. However, to ensure that the choice was the right one, this model was subject to a final test. The final test consisted in testing the model with 3, 4 and 5 layers. Since it was already established that the model 3 with just 1 layer was worse than the model 4 with 2 layers, maybe the increase of layers may have a positive impact in efficiency.

Results of this final test was presented in Table 12.

Analysing Table 12 and Figure 66, it was possible to see that the choice taken before was the right one and the best model should have just 2 layers.

Table 12 - Results for the final test with different number of layers (%)

No Layers	Valid (%)	Unique (%)
2	98.04	93.38
3	97.25	90.16
4	91.40	88.74
5	96.59	91.29

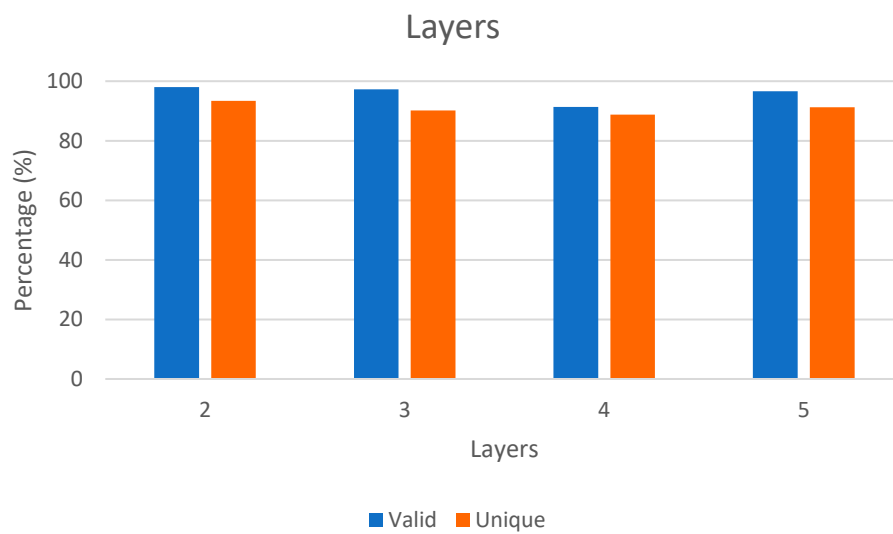


Figure 66 - Layers of the best model

## 4.5 Best Model and Optimal Parameters

Taking all the tests in consideration, the optimal value for each parameter, which originates the best model, was selected. Thus, the best model was model 4, LSTM with 2 layers, and the best values per each parameter were:

- Number of epochs for the sub-dataset 1: 24;
- Number of epochs for the sub-dataset 2: 15;
- Batch-size for the sub-dataset 1: 128;
- Batch-size for the sub-dataset 2: 16;
- Optimizer: Adam;

- Dropout: 0.3;
- Softmax temperature: 0.5;
- Number of SMILES in the sub-dataset 1: 240,000;
- Number of SMILES in the sub-dataset 2: 241,000.

The percentages achieved with the best model and the ideal values for each parameter was: 98.04% of valid SMILES and 93.38% of unique SMILES.

At this point the best model and parameters were chosen, therefore, a different analysis was done with the generated SMILES. Besides the percentages of valid and unique SMILES, the generated SMILES were: 1) compared with the dataset, to know how many SMILES were duplicated; 2) searched in the PubChem database; 3) searched in another collection of databases and datasets, both to understand how many SMILES were already included known in these databases.

The best model was run 30 times with the ideal values for the parameters (Table 13). In average, there were 97.51% of valid SMILES with a deviation of 0.31% (Figure 67), 91.43% of unique SMILES with a deviation of 0.92% and 1.85% already existed in the dataset of the training (Figure 68). Furthermore, 7.39% of the generated SMILES were found in PubChem (Figure 69), while 2.56% were found in the database that contains a collection of different datasets and databases (Figure 70). Yet, on average each run took 13 hours and 48 minutes, the loss of the training of the sub-dataset 1 was 0.4462 and loss of the sub-dataset 2 was 0.4388.

Table 13 - Best Model analysis

	Valid (%)	Unique (%)	Dataset replicates (%)	PubChem (%)	Databases (%)
<b>LSTM - 2 layer (average)</b>	97.51	91.43	7.39	1.85	2.56

Best Model - Validity

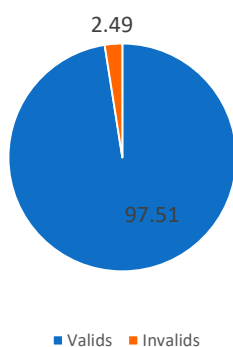


Figure 67 - Best Model - Validity

Best Model - Pubchem

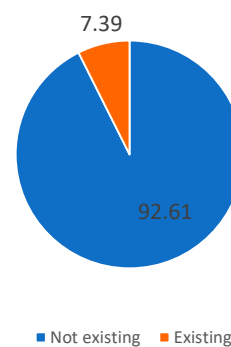


Figure 69 - Best Model - PubChem

Best Model - Dataset replicates

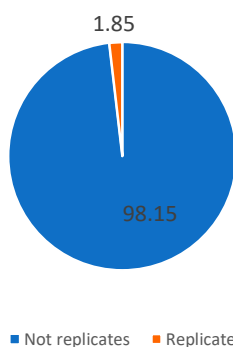


Figure 68 - Best Model - Dataset replicates

Best Model - Databases

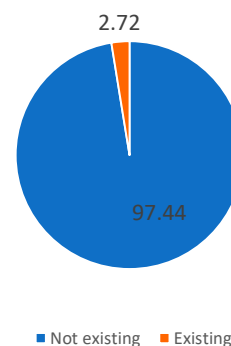


Figure 70 - Best Model - Databases

## 4.6 Fragmentation Growing Procedure

The fragmentation growing procedure, previously explained in Chapter 3, is presented in this section with the purpose to ensure to the reader that this model can actually achieve good results.

The fragmentation growing procedure was composed of tests taken with two fragments of SMILES, benzamidine and 3-Methylpyrazole (Figure 71). Table 14 shows the results obtain in this strategy. The run using benzamidine, as a fragment, reached 84.77% of valid SMILES, (Figure 72) and 39.65% of unique SMILES. More efficiently, the run using 3-

Methylpyrazole achieved an almost perfect result, 99.14% of valid SMILES (Figure 73) and 77.98% of unique SMILES.

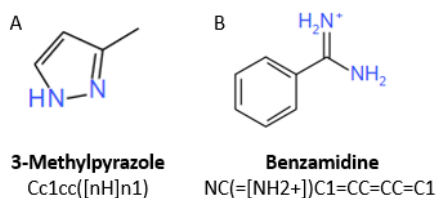
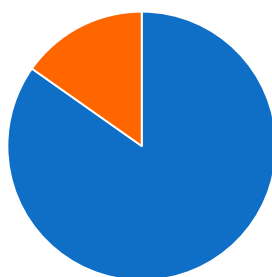


Figure 71 - SMILES used in the fragmentation growing procedure. A - 3-Methylpyrazole. B - Benzamidinium

Table 14 - Results of fragmentation growing procedure (%)

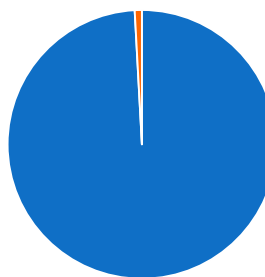
Name	Fragment	Valid	Unique
<b>Benzamidinium</b>	NC(=[NH2+])C1=CC=CC=C1	84.77	39.65
<b>3-Methylpyrazole</b>	Cc1cc([nH]n1)	<b>99.14</b>	77.98

Benzamidinium - Validity



■ Valid ■ Invalid

3-Methylpyrazole - Validity



■ Valid ■ Invalid

Figure 72 - Fragmentation growing procedure - Benzamidinium - Validity

Figure 73 - Fragmentation growing procedure - 3-Methylpyrazole - Validity



In Figure 74 and Figure 75, there are presented 6 SMILES generated with the fragmentation growing procedure. The first 3 SMILES of Figure 74 are examples of SMILES generated with benzamidine, as the fragment, and Figure 75 shows examples of SMILES which fragment was 3-Methylpyrazole. Comparing the SMILES Figure 74 of with Figure 71B, it was possible to observe that the fragment is actually in the SMILES. The same happens in Figure 75 SMILES.

This strategy presented successful results, which means that it was possible to choose fragments, and therefore features existing in the generated SMILES.

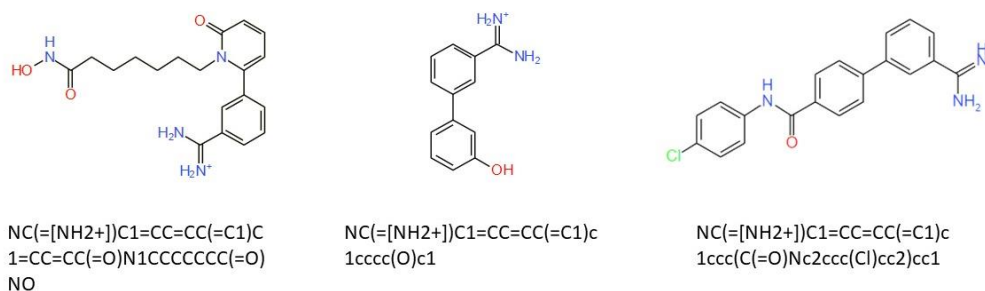


Figure 74 - SMILES generated with the fragment Benzamidine

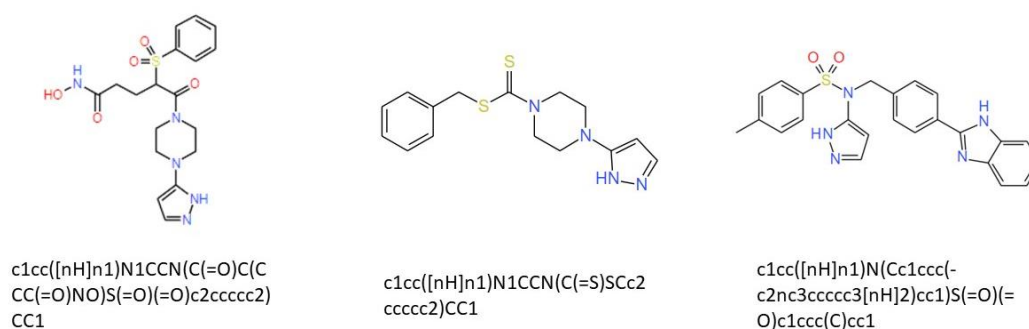


Figure 75 - SMILES generated with the fragment 3-Methylpyrazole

## 4.7 General Discussion

After analysing all the tests done, it is possible to conclude that the best recurrent network to achieve the main goal is an LSTM with 2 layers, using Adam optimizer. The best percentage of valid SMILES obtained was 98.04%, but in  $97.5\% \pm 0.31\%$  of valid SMILES. The parameter that influenced more the results was softmax temperature, since it was noticed that existed a significant increment in the result when that parameter was chosen. Regarding the fragmentation growing procedure, the best result was obtained with the 3-Methylpyrazole, 99.14%.

Comparing with previous works, the percentage of valid SMILES obtained in those studies was similar to the average of valid SMILES obtained in this work ( $97.5\% \pm 0.31\%$ ) [18, 19]. Regarding the fragmentation growing procedure, the best value obtained in this work (99.14%) is bigger than the related work of Gupta et al., (97%) which applied the same strategy [18]. These comparisons are useful in the way that it is possible to conclude that the results obtained in this work are coherent, respecting its purpose.

The parameters chosen after each set of tests are similar to the ones used by the authors in the related works. However, the optimizer parameter was the more surprising one, since two related works presented used Adam [18, 19] while the other one used Adagrad [20]. Conversely, the optimizer tested in this work, RMSprop optimizer, presented good results, sometimes better than Adam that was the optimizer chosen. RMSprop was a surprise since it is not usual to see authors using this optimizer in works in this area.

The main limitations in the performance of these tests were time and memory, since the available machines to perform the tests had a good GPU with low memory or a good RAM with only CPU. In fact, the lack of memory prevented the use of a bigger dataset in the study, which could lead to slightly better results. The excessive duration of each run was also a problem. The absence of available time to complete this work lead to the exclusion of the worse models while the tests were done.

Finally, the second dataset can be used in transfer learning, since this model was implemented taking this aspect into account and its ready for that. Despite, this work does not cover this strategy, it could be a possibility in the future.

# Chapter 5

## Conclusion

The strategy applied in this study gave rise to 97.5% of valid SMILES, on average. It was a percentage close to 100%, which means that the developed model fulfilled its main purpose, to generate new molecules using the SMILES format.

The choice of the dataset and the validation of the model are fundamental for the triumph of the model and, therefore, of this work, because the first one trains the model and the second validates it. In one hand, a consistent dataset is a key player for the success of the neural networks training, since the good results obtained are directly proportional to the suitability of the dataset chosen. In the other hand, the procedure for the validation covers the syntactically and biochemically validation of the generated SMILES. This validation was essential to validate the whole work since the better way to evaluate the model was by analysing its results.

The fragmentation growing procedure proved to be a good strategy to restrict the feature space of the generated SMILES. For this reason, this element completed the main component adding this aspect, where the best result obtained was 99% of valid SMILES.

In fact, from this work and the tests performed, it can be concluded that the implemented system is a delicate set of small components, such as datasets, validations and hyperparameters of each layer of the recurrent models. Each component and variable must be well studied and tested to achieve the best results, in this case is essentially the percentage of valid SMILES.

In this work, some techniques and variables were defined. However, they could be improved. For example, the model could be trained with a bigger dataset. However, that was not possible because of the lack of powerful machines, that could not tolerate more data. Thus, with a more powerful machine available the results obtained could be better, while

with a GPU the time for each execution could be more efficient. Moreover, the transfer learning technique remained to be used, since the purpose of this work could be achieved without that technique. However, choosing the second dataset with specific characteristics, the model developed could be used with that technique.

In summary, this work tested four types of recurrent networks, RNN, LSTM, GRU and BLSTM with a wide variety of values for a set of hyperparameters of the networks. From the tests, it was clear that the best model used was LSTM with 2 layers and the optimizer chosen was Adam.

# References

1. Gil, Y., et al., *Amplify scientific discovery with artificial intelligence*. Science, 2014. **346**(6206): p. 171.
2. Majaj, N.J. and D.G. Pelli, *Deep learning-Using machine learning to study biological vision*. J Vis, 2018. **18**(13): p. 2.
3. Panch, T., P. Szolovits, and R. Atun, *Artificial intelligence, machine learning and health systems*. J Glob Health, 2018. **8**(2): p. 020303.
4. Fan, S., et al., *Machine Learning Methods in Precision Medicine Targeting Epigenetics Diseases*. Curr Pharm Des, 2018.
5. Rogers, M.A. and E. Aikawa, *Cardiovascular calcification: artificial intelligence and big data accelerate mechanistic discovery*. Nat Rev Cardiol, 2018.
6. Lu, W., et al., *Applications of Artificial Intelligence in Ophthalmology: General Overview*. J Ophthalmol, 2018. **2018**: p. 5278196.
7. Mazurowski, M.A., et al., *Deep learning in radiology: An overview of the concepts and a survey of the state of the art with focus on MRI*. J Magn Reson Imaging, 2018.
8. Zeng, X. and G. Luo, *Progressive sampling-based Bayesian optimization for efficient and automatic machine learning model selection*. Health Inf Sci Syst, 2017. **5**(1): p. 2.
9. McKernan, L.C., E.W. Clayton, and C.G. Walsh, *Protecting Life While Preserving Liberty: Ethical Recommendations for Suicide Prevention With Artificial Intelligence*. Front Psychiatry, 2018. **9**: p. 650.
10. Im, H., et al., *Design and clinical validation of a point-of-care device for the diagnosis of lymphoma via contrast-enhanced microholography and machine learning*. Nat Biomed Eng, 2018. **2**(9): p. 666-674.
11. Sadoughi, F., et al., *Artificial intelligence methods for the diagnosis of breast cancer by image processing: a review*. Breast Cancer (Dove Med Press), 2018. **10**: p. 219-230.
12. Zhou, Y., et al., *Exploring Tunable Hyperparameters for Deep Neural Network with Industrial ADME Data Sets*. J Chem Inf Model, 2018.
13. Rifaioğlu, A.S., et al., *Recent applications of deep learning and machine intelligence on in silico drug discovery: methods, tools and databases*. Brief Bioinform, 2018.
14. *Drug Discovery*. [cited 2019 30/12/2018]; Available from: <https://www.nature.com/subjects/drug-discovery>.

15. Schreiber, S.L., *Target-oriented and diversity-oriented organic synthesis in drug discovery*. Science, 2000. **287**(5460): p. 1964-9.
16. Hessler, G. and K.H. Baringhaus, *Artificial Intelligence in Drug Design*. Molecules, 2018. **23**(10).
17. Chen, H., et al., *The rise of deep learning in drug discovery*. Vol. 23. 2018.
18. Gupta, A., et al., *Generative Recurrent Networks for De Novo Drug Design*. Mol Inform, 2018. **37**(1-2).
19. Segler, M.H.S., et al., *Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks*. ACS central science, 2018. **4**(1): p. 120-131.
20. Awale, M., et al., *Drug Analogs from Fragment-Based Long Short-Term Memory Generative Neural Networks*. Journal of Chemical Information and Modeling, 2019. **59**(4): p. 1347-1356.
21. Xue, H., et al., *Review of Drug Repositioning Approaches and Resources*. Int J Biol Sci, 2018. **14**(10): p. 1232-1244.
22. Singh, S., B.K. Malik, and D.K. Sharma, *Molecular drug targets and structure based drug design: A holistic approach*. Bioinformation, 2006. **1**(8): p. 314-20.
23. Patel, V.L., et al., *The coming of age of artificial intelligence in medicine*. Artif Intell Med, 2009. **46**(1): p. 5-17.
24. Wirth, N., *Hello marketing, what can artificial intelligence help you with?* International Journal of Market Research, 2018. **60**(5): p. 435-438.
25. LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning*. Nature, 2015. **521**: p. 436.
26. Florio, T., S. Einfeld, and F. Levy, *Neural networks and psychiatry: candidate applications in clinical decision making*. Aust N Z J Psychiatry, 1994. **28**(4): p. 651-66.
27. Starzomska, M., *[Use of artificial neural networks in clinical psychology and psychiatry]*. Psychiatr Pol, 2003. **37**(2): p. 349-57.
28. *Asimov Institute*. [cited 2019 11/01/2019]; Available from: <http://www.asimovinstitute.org/?s=neural+networks>.
29. Bhandare, A., et al., *Applications of convolutional neural networks*. International Journal of Computer Science and Information Technologies, 2016. **7**(5): p. 2206-2215.
30. Yamashita, R., et al., *Convolutional neural networks: an overview and application in radiology*. Insights into imaging, 2018. **9**(4): p. 611-629.
31. Nallapati, R., et al., *Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond*. 2016. 280-290.
32. Auli, M., et al., *Joint language and translation modeling with recurrent neural networks*. 2013. 1044-1054.

33. J. Han, K., et al., *Deep Learning-Based Telephony Speech Recognition in the Wild*. 2017. 1323-1327.
34. Bansal, T., D. Belanger, and A. McCallum. *Ask the gru: Multi-task learning for deep text recommendations*. in *Proceedings of the 10th ACM Conference on Recommender Systems*. 2016. ACM.
35. Choi, K., G. Fazekas, and M. Sandler, *Text-based LSTM networks for automatic music composition*. arXiv preprint arXiv:1604.05358, 2016.
36. Gawehn, E., J.A. Hiss, and G. Schneider, *Deep Learning in Drug Discovery*. *Molecular Informatics*, 2016. **35**(1): p. 3-14.
37. INFARMED - decreto-lei nº 495/99. Available from: [https://www.infarmed.pt/documents/15786/1065790/008\\_DL\\_495\\_99\\_VF.pdf?fbclid=IwAR10ozzD0IEYR0hqWRYOs0wOM6Fd99aXFAAGu1dYfqNhcbbQXG\\_mbLc\\_RM](https://www.infarmed.pt/documents/15786/1065790/008_DL_495_99_VF.pdf?fbclid=IwAR10ozzD0IEYR0hqWRYOs0wOM6Fd99aXFAAGu1dYfqNhcbbQXG_mbLc_RM).
38. *Decreto-lei nº269/2007*. Available from: [https://dre.pt/pesquisa/-/search/636710/details/maximized?fbclid=IwAR1q\\_wI6r3RTGvT7JPUy4stg6iApOKcQCQLsfHHWVt68c4hniK2l0ntonM4](https://dre.pt/pesquisa/-/search/636710/details/maximized?fbclid=IwAR1q_wI6r3RTGvT7JPUy4stg6iApOKcQCQLsfHHWVt68c4hniK2l0ntonM4).
39. Hoebert, J.M., et al., *Future of the European Union regulatory network in the context of the uptake of new medicines*. *Br J Clin Pharmacol*, 2013. **76**(1): p. 1-6.
40. Rouse, R., et al., *Translating New Science Into the Drug Review Process: The US FDA's Division of Applied Regulatory Science*. *Ther Innov Regul Sci*, 2018. **52**(2): p. 244-255.
41. Cortes, C. and V. Vapnik, *Support-vector networks*. *Machine Learning*, 1995. **20**(3): p. 273-297.
42. Breiman, L., *Random Forests*. *Machine Learning*, 2001. **45**(1): p. 5-32.
43. Svetnik, V., et al., *Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling*. *Journal of Chemical Information and Computer Sciences*, 2003. **43**(6): p. 1947-1958.
44. Rogers, D., R.D. Brown, and M. Hahn, *Using Extended-Connectivity Fingerprints with Laplacian-Modified Bayesian Analysis in High-Throughput Screening Follow-Up*. *Journal of Biomolecular Screening*, 2005. **10**(7): p. 682-686.
45. Hameed, P.N., et al., *A two-tiered unsupervised clustering approach for drug repositioning through heterogeneous data integration*. *BMC Bioinformatics*, 2018. **19**(1): p. 129.
46. Ma, J., et al., *Deep Neural Nets as a Method for Quantitative Structure–Activity Relationships*. *Journal of Chemical Information and Modeling*, 2015. **55**(2): p. 263-274.
47. Lusci, A., G. Pollastri, and P. Baldi, *Deep Architectures and Deep Learning in Chemoinformatics: The Prediction of Aqueous Solubility for Drug-Like Molecules*. *Journal of Chemical Information and Modeling*, 2013. **53**(7): p. 1563-1575.

48. Morgan, H.L., *The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service*. Journal of Chemical Documentation, 1965. 5(2): p. 107-113.
49. Ozturk, H., A. Ozgur, and E. Ozkirimli, *DeepDTA: deep drug-target binding affinity prediction*. Bioinformatics, 2018. 34(17): p. i821-i829.
50. Ragoza, M., et al., *Protein-Ligand Scoring with Convolutional Neural Networks*. J Chem Inf Model, 2017. 57(4): p. 942-957.
51. Szymkuć, S., et al., *Computer-Assisted Synthetic Planning: The End of the Beginning*. Angewandte Chemie International Edition, 2016. 55(20): p. 5904-5937.
52. Engkvist, O., et al., *Computational prediction of chemical reactions: current status and outlook*. Drug Discovery Today, 2018. 23(6): p. 1203-1218.
53. Law, J., et al., *Route Designer: A Retrosynthetic Analysis Tool Utilizing Automated Retrosynthetic Rule Generation*. Journal of Chemical Information and Modeling, 2009. 49(3): p. 593-602.
54. Chen, J.H. and P. Baldi, *Synthesis Explorer: A Chemical Reaction Tutorial System for Organic Synthesis Design and Mechanism Prediction*. Journal of Chemical Education, 2008. 85(12): p. 1699.
55. Kayala, M.A. and P. Baldi, *ReactionPredictor: Prediction of Complex Chemical Reactions at the Mechanistic Level Using Machine Learning*. Journal of Chemical Information and Modeling, 2012. 52(10): p. 2526-2540.
56. Kayala, M.A., et al., *Learning to Predict Chemical Reactions*. Journal of Chemical Information and Modeling, 2011. 51(9): p. 2209-2222.
57. Liu, B., et al., *Retrosynthetic Reaction Prediction Using Neural Sequence-to-Sequence Models*. ACS Central Science, 2017. 3(10): p. 1103-1113.
58. Walters, W., M. T Stahl, and M. Murcko, *Virtual Screening – An Overview*. Vol. 3. 1998. 160-178.
59. Lyne, P.D., *Structure-based virtual screening: an overview*. Drug Discovery Today, 2002. 7(20): p. 1047-1055.
60. Altae-Tran, H., et al., *Low Data Drug Discovery with One-Shot Learning*. ACS Cent Sci, 2017. 3(4): p. 283-293.
61. Graves, A., et al., *Hybrid computing using a neural network with dynamic external memory*. Nature, 2016. 538(7626): p. 471-476.
62. Schneider, P. and G. Schneider, *De Novo Design at the Edge of Chaos*. Journal of Medicinal Chemistry, 2016. 59(9): p. 4077-4086.
63. Hartenfeller, M. and G. Schneider, *Enabling future drug discovery by de novo design*. Wiley Interdisciplinary Reviews: Computational Molecular Science, 2011. 1(5): p. 742-759.



64. Goodfellow, I.J., et al., *Generative adversarial networks*. Advances in neural information processing systems, 2014. **3**: p. 2672-2680.
65. Kadurin, A., et al., *druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico*. Mol Pharm, 2017. **14**(9): p. 3098-3104.
66. Jaques, N., et al. *Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control*. in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 2017. JMLR. org.
67. *PubChem - documents*. [cited 2019; Available from: <https://pubchemdocs.ncbi.nlm.nih.gov/about>.
68. *PubChem - data*. [cited 2019; Available from: <https://pubchemdocs.ncbi.nlm.nih.gov/statistics>.
69. *Daylight*. [cited 2019 03/01/2019]; Available from: <http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html>
70. *NumPy*. [cited 2019 03/01/2019]; Available from: <http://www.numpy.org>
71. *Keras: The Python Deep Learning library*. [cited 2019 02/01/2019]; Available from: <https://keras.io>.
72. *Tensor Flow - An open source machine learning library for research and production*. [cited 2019 02/01/2019]; Available from: <https://www.tensorflow.org>.
73. Chung, J., et al., *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014.
74. *Introduction to Recurrent Neural Network*. [cited 2019; Available from: <https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/>.
75. *Keras - Optimizers*. [cited 2019; Available from: <https://keras.io/optimizers/>.
76. *An Introduction to Recurrent Neural Networks*. [cited 2019; Available from: <https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912>.
77. Gajbhiye, A., et al. *An Exploration of Dropout with RNNs for Natural Language Inference*. in *Artificial Neural Networks and Machine Learning – ICANN 2018*. 2018. Cham: Springer International Publishing.
78. Goldberg, Y., *A primer on neural network models for natural language processing*. J. Artif. Int. Res., 2016. **57**(1): p. 345-420.
79. *Introduction – Why do we need Sequence Models??* [cited 2019 20/01/2019]; Available from: <https://towardsdatascience.com/introduction-to-sequence-models-rnn-bidirectional-rnn-lstm-gru-73927ec9df15>.

80. *Understanding LSTM networks.* [cited 2019 20/01/2019]; Available from: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
81. *Deep Learning Basics: Gated recurrent unit (GRU).* [cited 2019; Available from: <https://medium.com/mlrecipies/deep-learning-basics-gated-recurrent-unit-gru-1d8e9fae7280>.
82. Graves, A., S. Fernández, and J. Schmidhuber, *Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition.* 2005. 799-804.
83. *Understanding Bidirectional RNN in PyTorch.* [cited 2019; Available from: <https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66>.
84. *Keras - Sequential.* [cited 2019; Available from: <https://keras.io/models/sequential/>.
85. *Keras - Dropout.* [cited 2019; Available from: <https://keras.io/layers/core/#dropout>.
86. Gal, Y. and Z. Ghahramani. *A theoretically grounded application of dropout in recurrent neural networks.* in *Advances in neural information processing systems.* 2016.
87. Hinton, G., O. Vinyals, and J. Dean, *Distilling the Knowledge in a Neural Network.* 2015.
88. *Keras - Dense Layer.* [cited 2019; Available from: <https://keras.io/layers/core/>.
89. *MolVS.* [cited 2019; Available from: <https://molvs.readthedocs.io/en/latest/index.html>.
90. *RDKit.* [cited 2019; Available from: <https://www.rdkit.org/docs/Overview.html>.
91. *MolVS - Validation.* [cited 2019; Available from: <https://molvs.readthedocs.io/en/latest/api.html#id1>.