UNIVERSIDADE D
COIMBRA

Ulyana Motresku

IN SILICO APPROACHES FOR THE DETECTION OF ZINC ION IN REGULATORY PROTEINS

Thesis submitted to the Faculty of Science and Technology of the University of Coimbra for the degree of Master in Biomedical Engineering with specialization in Clinical Informatics and Bioinformatics, supervised by PhD Alexandra Carvalho and MSc Beatriz Columbano Almeida.

September, 2022

In silico approaches for the detection of zinc ion in regulatory proteins

Thesis submitted to the
University of Coimbra for the degree of
Master in Biomedical Engineering

Author:

Ulyana Motresku

Supervisors:

Alexandra Carvalho, PhD

Beatriz Columbano Almeida, MSc

Centre for Neuroscience and Cell Biology - University of Coimbra
Rational Protein Engineering Group

Coimbra, 2022

In silico approaches for the detection of zinc ion in regulatory proteins

Author:
Ulyana Motresku

Supervisors:
Alexandra Carvalho, PhD
Beatriz Columbano Almeida, MSc

Acknowledgements

Gostaria de agradecer, em primeiro lugar, a ambas as minhas orientadoras por todo o apoio que me prestaram ao longo deste ano letivo. À Doutora Alexandra Carvalho, cuja suas experiências científicas e académicas foram indispensáveis para a concretização deste projeto. À Beatriz Almeida, aluna de doutoramento, que para além de todas as horas despendidas em discussões e esclarecimentos, também me ajudou a evoluir tanto a nível científico como pessoal.

Quero exprimir a minha profunda gratidão à minha família, os meu pais, a minha irmã e o Cláudio por todo o amor e carinho incondicional. O que alcancei foi graças ao vosso esforço, aos vossos ensinamentos, à vossa educação e confiança. Sem vocês eu não conseguiria fazer tudo aquilo que fiz até hoje. A determinação, responsabilidade e ética que me foram ensinados fazem de mim aquilo que sou como pessoa.

Por fim, quero agradecer aos amigos que Coimbra me deu, à Rute, à Ana, à Margarida, à Beatriz e ao Telmo. Estes últimos anos só foram possíveis graças ao vosso apoio, graças ao nosso apoio uns pelos outros em todas as ocasiões. Não posso deixar de mencionar a meu profundo apreço às Ana's e ao Bruno que fazem parte de uma nova fase da minha vida.

Resumo

As proteínas reguladoras são macromoléculas complexas essenciais para o funcionamento biológico. A literatura recente tem mostrado grandes lacunas na caracterização de proteínas reguladoras, principalmente relacionadas com a falta do ião zinco nas estruturas dos cristais e à ausência de informações sobre seu papel. Assim, uma melhor caracterização molecular é essencial para aumentar a nossa compreensão das proteínas reguladoras dependentes de metais, uma vez que estas têm grande valor em áreas como a biomedicina e biotecnologia. Por este motivo, implementamos abordagens *in silico* para prever resíduos com ligação ao ião zinco em fatores de transcrição.

Primeiro, construímos o nosso próprio conjunto de dados com fatores de transcrição e, em seguida, extraímos um conjunto de características heterogéneas contendo características baseadas na sequência e estrutura das proteínas. Três modelos do estado da arte foram implementados e otimizados com nossos dados, *Convolutional Neural Networks* (CNN), *Long-Short Term Memory Neural Networks* (LSTM) e *Gated Recurrent Units* (GRU), bem como validados com um conjunto de dados de benchmark.

As características baseadas na sequência relacionadas aos resíduos cisteína e histidina, bem como a estrutura secundária onde o resíduo está localizado são as características com maior correlação linear com o alvo. Os modelos LSTM e GRU sofrem *overfitting* obtendo os mesmos valores em F1-Measure para o treino 65% e em F1-Measure em teste 43%. O modelo CNN reporta os menores valores F1-Measure em teste de 41%. Portanto, GRU é o nosso melhor modelo com valores de F1-Measure de 65,165%-treino, 52,926%-validação

e 42,898%-teste.

Neste projeto usamos a sequência de aminoácidos completa em vez de apenas resíduos de ligação específicos, o que é uma vantagem sobre os modelos implementados do estado da arte. Além disso, os nossos resultados de treino abrem as portas para melhorias quando as sequências de aminoácidos são usadas em tarefas semelhantes.

Palavras Chave: Fatores de Transcrição, Locais de ligação de zinco, Estrutura de proteínas, Coordenação ao zinco, Aprendizagem computacional, Química computacional

Abstract

Regulatory proteins are complex macromolecules essential for biological functioning. Recent literature has shown large gaps in regulatory proteins characterisation, mainly pertaining to the lack of zinc ion in the crystal structures and the absence of information about its role. Thus, an improved molecular characterisation is essential to increase our understanding of metal-dependant regulatory proteins since they have great value in biomedical and biotechnological fields. Therefore, we implemented *in silico* approaches to predict zinc-binding residues in transcription factors.

First, we constructed our own dataset with transcription factors, and then we extracted a set of heterogeneous features containing sequence and structure based features. Three state of the art models were implemented and optimised with our data, Convolutional Neural Networks (CNN), Long-Short Term Memory Neural Networks (LSTM) and Gated Recurrent Units (GRU), as well as validated with a benchmark dataset.

The sequence based features related do the cysteine and histidine residues and the secondary structure where the residue is localised are the features with higher linear correlation with the target. The LSTM and GRU models overfitted obtaining training F1-Measure of 65% both and low test F1-Measure of 43% both. The CNN model reports the lowest values with test F1-Measure of 41%. Therefore, GRU is our best model with F1-Measure values of 65.165%-training, 52.926%-validation and 42.898%-testing.

Here we used the complete amino acid sequence instead of just specific binding residues, which is an advantage over the state of the art implemented models. Also, our training

results opened the doors for improvement when amino-acid sequences are used in similar tasks.

Keywords: Transcription factors, Zinc binding-sites, Protein structure, Zinc coordination, Machine Learning, Computational Chemistry

Contents

List of Figures

List of Tables

Acronyms

CNN Convolutional Neural Networks.

DSSP Hydrogen Bond Estimation Algorithm.

FANTOM Fast Newton-Raphson Torsion Angle Minimizer.

GRU Gated Recurrent Units.

LSTM Long-Short Term Memory Neural Networks.

ML Machine Learning.

PAM Point Accepted Mutation.

PR AUC Precision-Recall Area Under the Curve.

ProCos Protein Composition Server.

PSI-BLAST Position-Specific Iterative Basic Local Alignment Search Tool.

RF Random Forest.

SASA Solvent Accessible Surface Area.

SVM Support Vector Machine.

TF Transcription Factors.

Chapter 1

Introduction

Bioinformatics is the branch of computer science that analyses large collections of biological data. It is a interdisciplinary field that combines computer science, mathematics, statistics, physics and biology [1].

The cornerstones of bioinformatics were settled in the 1960s with the development of computer techniques for protein sequence analysis. First, the *de novo* peptide sequence assembler was developed [2], followed up by the first protein sequence database [3] and then the amino acid substitution model for phylogenetics [4]. At the beginning of the new millennium, the use of the Internet associated with next-generation sequencing led to an exponential increase in data and a rapid proliferation of bioinformatics tools [5]. Currently, bioinformatics faces numerous challenges, including dealing with Big Data and ensuring reproducibility of results.

Computational chemistry is the branch of chemistry that solves chemical and biological problems. It relies on quantum mechanics or classical molecular mechanics calculations [6] to compute the structures and properties of molecules and solids [6, 7]. Computational chemistry frequently requires the assistance and validation of experimental setups. Along with bioinformatics, computational chemistry has emerged as a reliable method for unravelling the complexities of macromolecules, such as proteins, carbohydrates, nucleic

acids, and lipids [7]. These techniques make the bridge between raw biological data to specific knowledge about a given biological process. However, understanding the results and models provided by bioinformatics remains a barrier to the development of other areas or the complete application of the findings produced by these experiments [7].

Proteins are macromolecules composed of a sequence of amino acids. They perform essential functions in organisms, such as storage, transport, catalysis of metabolic reactions, DNA replication, stimuli response, and structure support to cells and organisms [8]. Recent studies have shown that 30%-40% of proteins require one or several metal cofactors to tune their biological function, stability, structure, and regulation [9]. These proteins are designated as metal-biding proteins or metalloproteins [10]. The most common metal ions found in proteins are: calcium (Ca^{2+}), cobalt (Co^{2+}), copper (Cu^{2+}), iron (Fe^{3+} or Fe^{2+}), manganese (Mn^{2+}), magnesium (Mg^{2+}), potassium (K^{2+}), sodium (Na^{2+}), nickel (Ni^{2+}), and zinc (Zn^{2+}) [9]. Zinc ion is the second most abundant transition metal in living organisms after iron. Zinc is a common catalytic cofactor because of its chemical properties: Lewis acid strength, lack of redox reactivity and fast ligand exchange [11].

In eukaryotes, most zinc proteins function in the regulation of gene expression, such as regulatory proteins [11]. Regulatory proteins, such as transcription factors (TF) play a critical role in the biology of microorganisms. TF repress, de-repress or activate gene transcription through a tightly regulated direct interactions mediated by various unique domains or motifs such as helix–turn–helix domains, helix–loop–helix domains, zinc fingers, homeodomains, leucine zippers and β-sheet DNA-binding proteins [12, 13]. The transcriptional control results from an interplay between regulatory DNA sequences and site-specific DNA-binding proteins. Therefore, the TF acquire a tertiary shape compatible with the surface of the DNA. When the TF and the DNA are close enough, the TF establish numerous atomic interactions, such as hydrogen bonds, ionic interactions, and hydrophobic interactions, which allow the regulation of that gene [14].

The traditional methods that are used to identify metal-binding conformation or bind-

ing residues include biochemical and biophysical experiments, such as mass spectrometry, X-ray crystallography, high-throughput X-ray absorption spectroscopy, surface Plasmon resonance, and isothermal titration calorimetry [9, 15]. However, these technologies are too costly and time-consuming. So computational tools have been developed to predict metal-binding residues [15]. There are several algorithms to predict zinc-binding sites, Nan Ye *et al.* classified them according to their basic design and scheme. Therefore, the models can be divided into four categories learning−, docking−, template− and meta−based methods [9]. In this work, we aim to use learning-based models to predict zinc-ion binding residues by implementing deep learning algorithms to construct a prediction model.

To reach the goal of our project, we focus on predicting zinc-ion binding residues using features extracted from the protein sequences and their respective structures. We studied the implemented state of the art methods and applied them to our specific problem. As a final result, we implemented three deep learning classification algorithms that are able to predict zinc-binding sites.

1.1 Motivation

The most accurate way to detect zinc-binding residues still through biological techniques, such as the ones mentioned above. However, due to the huge amount of proteins it is not possible to do experiments for all. Therefore, *in silico* approaches predict a higher amount of interactions per unit of time, although the results are sometimes inaccurate [9]. Recent literature has shown large gaps in regulatory proteins characterisation, mainly pertaining to the lack of zinc ion in the crystal structures and the absence of information about its role [16, 17]. This misleading information can be a reflection of the crystallographic conditions or due to the molecular analysis complexity.

Regulatory proteins when inducible by small molecules or drugs, have a great value in the biomedical and biotechnological fields [18]. Thus, an improved molecular character-

isation can be helpful to increase our understanding about metal-dependent regulatory proteins. To fulfil this lack of information, it is important to study new models able to better understand the features extracted from the protein's sequence and structure.

1.2 Objectives

This project aims to develop a computational model to detect zinc in TF, using the proteins tridimensional structure and amino acid sequence composition. To reach this goal, several steps were followed:

1. The study of Machine Learning approaches to predict zinc-binding in proteins.
2. Study of protein and amino acid properties to be selected as features.
3. Dataset, feature selection, and feature engineering.
4. Study of Deep Learning models and their different variations.
5. Model training, tunning and testing.
6. Analysis of the model with benchmark dataset.

These steps described in the following chapters show the workflow of this thesis.

Our main goal is to achieve a set of models that can predict zinc-ion binding residues, Scikit-learn [19] and Keras [20] were used as the principal resources, among other libraries.

1.3 Outline

The rest of this document is organised as follows: Chapter 2 contains background knowledge and the state of the art. Chapter 3, contains the methods used and Chapter 4 the results and discussion. Finally in Chapter 5 the conclusion of the work and future perspectives are presented

Chapter 2

State of the Art

In this chapter, we introduce the concepts necessary for predicting zinc ions in proteins.

We start with biological background, and then we formally define our problem and present various classification methods that have been used in similar tasks [21–24]. Subsequently, we present and analyse the different feature extraction techniques for proteins. This state of the art analysis provides a better understanding of previous strategies, allowing us to build a better classifier in an easy and efficient way.

2.1 Biological Background

2.1.1 Proteins - Brief Overview

Proteins are essential organism components and play a role in nearly every process within cells [8]. Some proteins, enzymes, catalyse biochemical reactions and are essential for metabolism. Other proteins play critical roles in cell signalling, immune responses, cell adhesion, cell cycle, and cell shape [25].

Protein synthesis can be divided into two steps: first, the transcription of the DNA into RNA, and second, the translation of the RNA into proteins. There are 20 amino

acids commonly found as residues in proteins; however, other less common amino acids also occur [25]. These proteins can be free metabolites or constituents that have had common amino acid residues modified after protein production. An amino acid contains an α-carboxyl group, an α-amino group, and a side-chain or specific R group, as represented in Figure 1. They can be classified into five groups on the basis of the polarity and charge (at pH 7) of their side-chains, namely the aliphatic with apolar R groups, the non charged with polar R groups, the aromatic, the positively charged and the negatively charged [25].

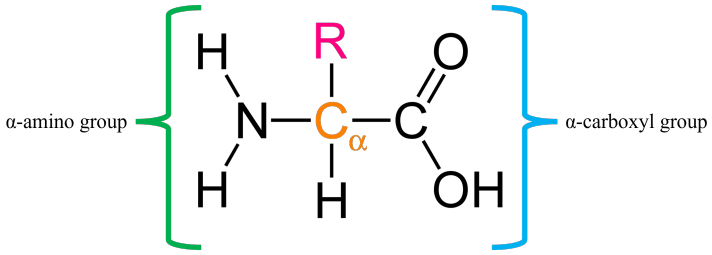


Figure 1: Amino acid structure. An amino acid has an α-carbon (orange), an α-amino group (green), an α-carboxyl group (blue) and a specific R group (pink).

Protein structure is classified in four different levels of complexity: primary, secondary, tertiary, and quaternary, Figure 2. The primary structure consists of the linear polypeptide chain with one-dimensional information, Figure 2A. In the secondary structure, hydrogen bonds are formed within the polypeptides, originating a three-dimensional arrangement, such as an α-helix or β-sheet, Figure 2B. Then the overall three-dimensional structure of a polypeptide is called its tertiary structure, Figure 2C. It is primarily due to interactions between the side-chains of the amino acids. When a protein is composed of several polypeptide chains, the arrangement of these chains is called the quaternary structure of the protein, Figure 2D [25].

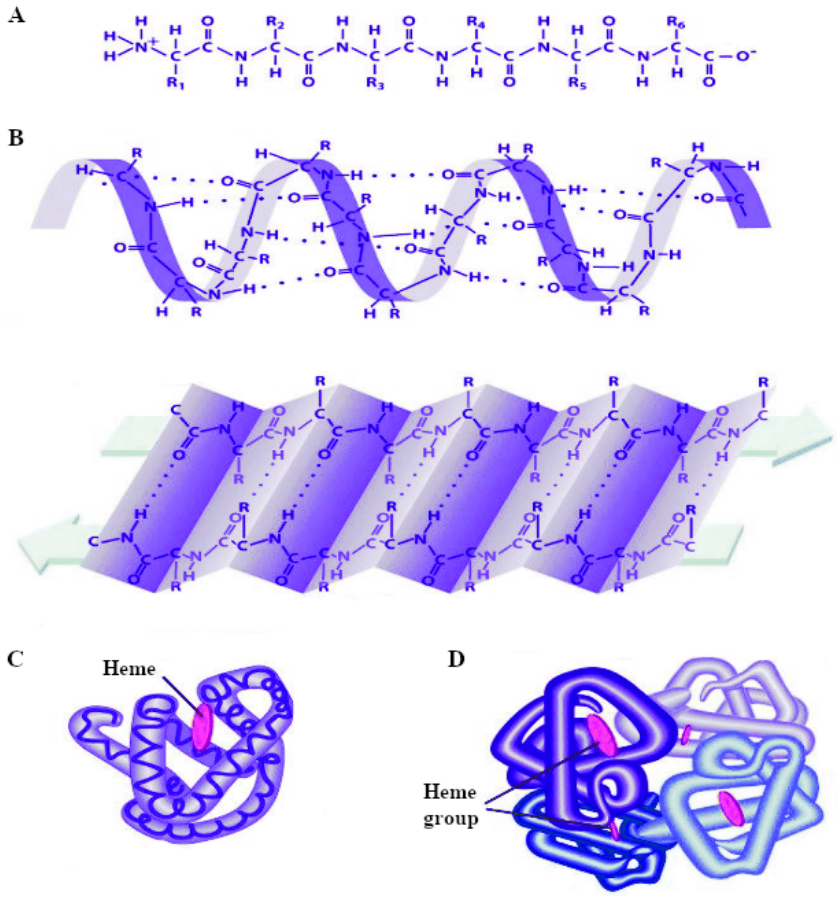


Figure 2: Four levels of protein structure. (A) Primary structure - constituted by an amino end and a carboxyl end; (B) Secondary structure - the first scheme represents an alpha helix, in which the dots represent the hydrogen bonds between residues at different locations; in the second scheme, the alpha helix it is shown in a pleated sheet schematic view. (C) Tertiary structure and (D) Quaternary structure. (Adapted from Ritika Gupta, 2017[26])

2.1.1.1 Primary Structure

The primary structure, as defined previously, is the ultimate determinant of the overall conformation of a protein. The primary structure of any protein in its current state results from mutation and selection over the evolutionary time [25]. During the synthesis of the protein, the order in which the amino acids are connected defines a set of interactions between amino acids [25]. Therefore, the order of the side-chain structures and resulting interactions are very important since early interactions affect later interactions. Thus,

the primary structure is essential for the other features that a protein possesses, and the following structures depend on these properties [25].

2.1.1.2 Secondary and Tertiary Structure

The secondary structure derives from the hydrogen bonds formed between the backbone amino acids atoms. They are formed between the partially negative oxygen atom and the partially positive nitrogen atom [27]. It is important to refer that the hydrogen bonds that compose secondary structure do not include the ones involving the amino acid side-chains. The hydrogen bonds can coil or fold the polypeptide chain, generating patterns that contribute to the protein shape. As a result of the often repetition of some patterns, they have been identified [27]. The Hydrogen Bond Estimation Algorithm (DSSP) defines eight types of secondary structure:

- G - 3-turn helix
- H - 4-turn helix (α helix)
- I - 5-turn helix (π helix)
- T - hydrogen bonded turn
- E - extended strand in parallel and/or anti-parallel β-sheet conformation
- B - residue in isolated β-bridge
- S - bend
- C - coil

The tertiary structure is the overall three-dimensional shape formed by the interactions of the side-chains of the various amino acids. As an example, amino acids with polar properties are hydrophilic and can be in contact with water, whereas nonpolar amino acids are hydrophobic, so they will cluster at the core of the protein, avoiding contact with the surrounding water [28]. Therefore, the properties of the R groups will highly influence the protein's tertiary structure and global shape [28].

2.1.2 Role of the Zinc Ion in Regulatory Proteins

Due to the filled d-shell orbital, Zn^{2+} has ligand-field stabilisation energy of zero in all ligand geometries. Therefore, no geometry is inherently more stable than another. This property can be used by zinc metalloproteins to alter the reactivity of the metal ion. Hence, it is an essential factor for the ability of this ion to catalyse chemical transformations accompanied by changes in the metal coordination geometry without destabilising the structure. [10, 11].

Zinc sites have several functions, Andreini C. et al. in 2011 [11] shows that in Zn-superfamilies 58% of the zinc sites have a structural role, 18% have a catalytic role, 4% have a regulation role, 2% have a substrate role, and 18% have an unknown role. In non-redundant zinc proteins, 55% of the zinc sites have a structural role, 35% have a catalytic role, 2% have a regulation role, 1% have a substrate role, and 7% have an unknown role [11].

Cysteine, histidine, aspartic acid, and glutamic acid residues are the most common residues that coordinate with zinc ions. Moreover, the characteristic spacings between the ligands defines different zinc motifs in proteins. These motifs can be annotated in sequences of proteins that do not have their three-dimensional structure determined [29]. As an example, in zinc fingers, where zinc plays a structural role and has tetrahedral geometry, three coordinations spheres occur: (i) two cysteines and two histidines, (ii) three cysteines and one histidine, or (iii) four cysteines. These coordinations can occur with any permutation in the order of the ligands, as shown in Figure 3. Therefore, the arrangement of the zinc interacting with the ligand can be linear, Figure 3A, intertwined, Figure 3B, clustered, Figure 3C, or interleaved, Figure 3D [29].

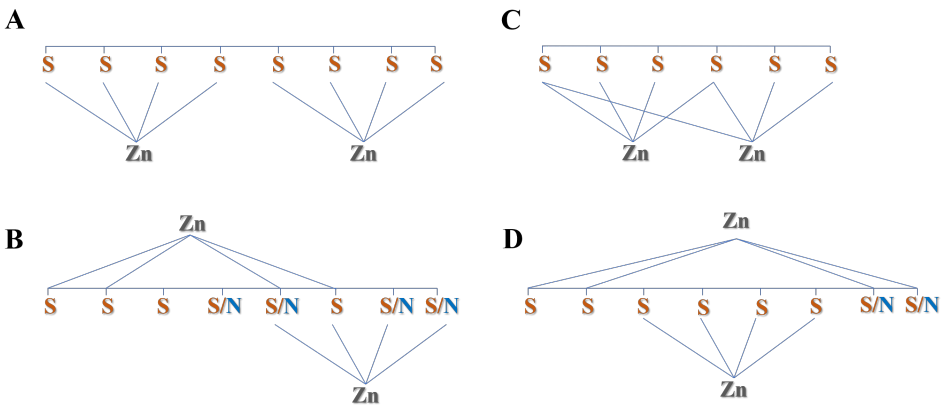


Figure 3: Zinc/sulfur coordination in protein domains. (A) Linear arrangement; (B) Intertwined arrangement; (C) Clustered arrangement; (D) Interleaved arrangement (S/N indicates that histidine can replace cysteine as a ligand in some members of these protein families.) (Adapted from Wolfgang Maret, 2005 [29])

Thus, it is possible to predict zinc-binding sites with the primary structure of a protein. However, the predictions are problematic when the spacing between the ligands is highly variable, ligands are shared by zinc ions, or ligand stem from different interacting proteins or subunits [29]. Then, it is also important to analyse features of the secondary and tertiary structure to improve performance of the prediction algorithms.

Figure 4A shows an example of a regulatory protein with zinc fingers, where the zinc ion has a structural function, and the Figure 4B presents an example of a protein where the zinc site has a regulatory function.

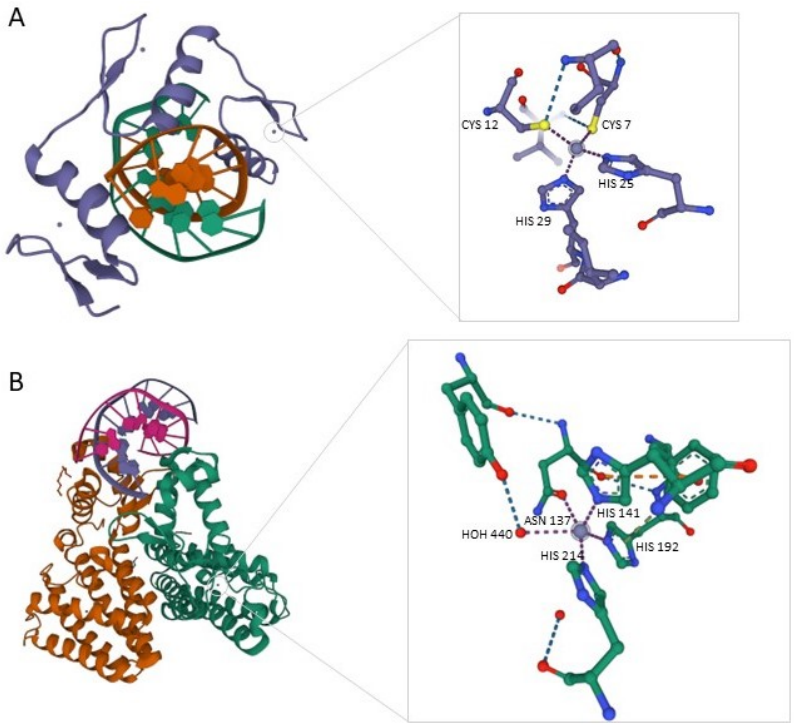


Figure 4: Zinc Ion in Regulatory Proteins (A) Schematic representation of the ZIF268 zinc finger-DNA complex (1AAY) and a close-up view of the Zn^{2+} binding site. (B) Schematic representation of structure of the transcriptional repressor Atu1419 (VanR) from *Agrobacterium fabrum* in complex a palindromic DNA (C2221 space group) (6ZA3) and a close-up view of the Zn^{2+} binding site. Zinc ion is represented by the grey sphere.

2.2 Classification Problem

This work intends to improve prediction of zinc ions in regulatory proteins, which is a binary classification problem. Therefore, our problem can be defined by the following function (1):

$$p_i = ZincIonDetector(x_i, y_i),\ where\ x_i\ \in P \subseteq \mathbb{R}^{\mathbb{N}} \tag{1}$$

ZincIonDetector is the algorithm that will detect zinc ion-binding to a residue i, by giving his prediction p_i. x_i represents the features of a single amino acid i that belongs

to a protein P, and y_i represents the desired output of the classification algorithm, where $y_i \in [0, 1]$, $y = 0$ defines the non-binding to zinc ion class and $y = 1$ defines the zinc ion-binding class. $\mathbb{R}^{\mathbb{N}}$ is the dataset of regulatory proteins and their respective features.

The algorithm uses feature-target training data, (x_i, y_i), in such a way that *ZincIonDetector* correctly predicts the classification of new data, test data, that was not supplied to the algorithm. The problem of metal ion-binding site prediction, particularly zinc ion-binding site prediction, has been the object of many approaches. Therefore, it is essential to make a brief description and analysis of some classifiers used for similar tasks. During this project, some of the analysed models were implemented to evaluate their performance on our specific problem.

2.2.1 State of the Art for Detection of Zinc Ion in Proteins

The computation-based methods for the prediction of metal-binding sites can be divided into four categories, learning-, docking-, template- and meta-based methods, as shown by Ye N. et al.[9]. Learning-based methods consider the detection of metal-binding residues as a classification problem, and use machine learning techniques. The docking-based methods have the objective of finding the appropriate binding conformation together with the appropriate target binding residues by examining the surface of the protein and then, with the help of a scoring function assess the binding pose. The template-based methods compare the structure of a given unknown protein with an optimal known template structure. Finally, the meta-based algorithms aim to build more accurate classifiers by combining existent predictors [9]. However, the focus of this project is on learning-based algorithms, specifically the study of machine learning approaches.

To solve problems similar to our project, different Machine Learning algorithms where tested, such as Random Forest (RF) algorithms [22], Support Vector Machine (SVM) [21, 23], Convolutional Neural Networks (CNN) [23, 24], Long-Short Term Memory Neural Networks (LSTM) [24], and Gated Recurrent Units (GRU) [24].

2.2.1.1 Random Forest Classifiers

A Random Forest (RF) [22, 30] based classifier is a supervised machine learning method, structured by an assemblage of decision tree classifiers, where each tree casts a unit vote for the most popular class, the class that they have predicted. Then, based on that polling, the prediction is made upon the class that had more "votes" [22, 30]. The use of RF classifiers has its advantages, such as effective handling over missing data, does not need hyper-parameter tuning to produce an acceptable prediction, and it also contours some issues of overfitting in decision trees [31].

Jiangning Song *et al.* [22] proposed the MetalExplorer [32], an RF based classifier and correctly classified eight different types of metal-binding sites (Ca^{2+}, Co^{2+}, Cu^{2+}, $Fe^{2+/3+}$, Ni^{2+}, Mg^{2+}, Mn^{2+} and Zn^{2+}) in proteins. They used heterogeneous features, sequence-, structure-, and residue contact network-based features. This model has prediction metrics calculated for each type of metal having a general precision of 60%, and a recall value that ranges from 59% to 88% depending on the specified metal. In particular for the zinc ion, this model has a prediction of 60% and a recall of 62.2% [22].

2.2.1.2 Support Vector Machine

A Support Vector Machine (SVM) [33, 34] is a supervised machine learning algorithm that can be used for prediction or regression problems, yet it is frequently used for classification problems [33]. SVMs aim to find a hyperplane in an N-dimensional space, where N is the number of features. The hyperplane is the decision boundary between classes, and its position is supported by the data points that are closer to it. These data points structure the support vectors [34]. SVMs have good performance when the margin of separation is evident and have low computational cost since the decision function is the subset of training points. However, the performance is low when working with large sets of data and when the dataset is noisy [33].

Xiuzhen Hu *et al.* in 2016 proposed the IonSeq model. IonSeq predicts nine metal ions

(Zn^{2+}, Cu^{2+}, Fe^{2+}, Fe^{3+}, Ca^{2+}, Mg^{2+}, Mn^{2+}, Na^{+}, K^{+}) and four acid radical ion ligands (CO_3^{2-}, NO_2^{2-}, SO_4^{2-}, PO_4^{3-}) [21]. The model is based on a modified AdaBoost algorithm [21] that creates a set of multiple training datasets to contour the issue of class imbalance. Then, with each dataset an SVM predictor model is created, and a final classifier is assembled by combining all the models [21]. This model uses sequence-based features. The prediction metrics where calculated for each individual metal/ligand, therefore sensitivity values of this model range between 5.57% and 77.14%, and the specificity value is 99%. Specifically for the zinc ion, this model has a sensitivity of 43.56% and specificity of 99.75% [21].

Ismail Haberal and Hasan Ogul, in 2017, proposed an SVM based model to predict zinc ion-binding to histidines and cysteines. This model uses sequence-based features, such as Point Accepted Mutation (PAM) [23, 35], that measures the rate at which point mutations that swap out one amino acid residue for another have been incorporated in a gene lineage over the course of evolution [36]. The model reached a precision of 65% and 81% recall [23].

2.2.1.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a deep neural network with a mechanism inspired by the biological visual cortex. This network is feed-forward since the information flows only in one direction in the model; therefore, there are no feedback connections [24, 37]. Convolutional layers extract features from the input training data outputting abstracted feature maps. The first layer extracts simpler features, and subsequent convolutional layers extract more complex features [24]. CNNs are a remarkable innovation for computer vision. They are helpful for problems with image data, such as image classification, object detection and segmentation. However, these networks also can be applied to problems with text data, time series data, and sequence input data [38].

Ismail Haberal and Hasan Ogul proposed, in 2017, the DeepMBS an CNN based model to predict zinc ion-binding to histidines and cysteines, claiming that their model was one of the first applications of deep learning models in the prediction of metal-binding sites [23]. This model uses sequence-based features, such as PAM [39], explained earlier. The model reached a precision of 79% and 82% recall [23].

Later, in 2019, the same authors proposed another model based on CNNs [24]. The model uses sequence-based features, such as PAM [35, 36], Protein Composition Server (ProCos [40] that obtain parameters useful for the functional characterisation of the proteins), and the binary representations of amino acids. A classifier was made for each of these feature groups, therefore, the prediction metrics were also calculated for each feature group. Precision values range between 67% to 79% and recall values range between 81.2% and 82% [24].

2.2.1.4 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are deep learning algorithms. RNN have as objective to work with sequence prediction problems, such as text data, speech data, classification problems, regression problems, and generative models [38]. Unlikely traditional deep learning models, the output of a RNN depends on the preceding elements of the sequence; in our context RNNs process one residue after the other [41]. RNNs, in contrast with feedforward algorithms, share parameters from one layer to another, such as weight parameters. However, these networks have short-term memory, which can cause the loss of information when exposed to sequences that are long enough [42]. Therefore, Long-Short Term Memory Neural Networks (LSTM) and Gated Recurrent Units (GRU) are the created solution for this problem.

Long-Short Term Memory Neural Networks (LSTMs) are a special type of RNN. They were designed to overcome the problem of short-memory on RNNs. The key

to LSTMs is the cell state, which is responsible for "remembering" values along the data sequence. Therefore, the cell state transports relevant information from start to finish. This information can be changed by the gates, which are different neural networks that learn which information is pertinent to keep or forget during training [42, 43]. LSTMs have four gates, the input gate, the forget gate, the output gate, and the update gate.

Ismail Haberal and Hasan Ogul in 2019 proposed an LSTM based model to predict zinc ion-binding to histidines and cysteines, with sequence-based features, that are explained in the above section [24]. Once again the classifiers were tested for each feature group obtaining precision values that range between 66% to 73.2% and recall values that range between 80.8% and 81% [24].

Gated Recurrent Units (GRUs) are identical to LSTMs but have fewer parameters, because they have only two gates, and do not have the cell state [42]. Therefore, this models may train faster and may not need the same amount of data as LSTM to generalise. However, LSTMs may have better performance in large datasets [42, 44].

Ismail Haberal and Hasan Ogul proposed an GRU based model to predict zinc ion-binding to histidines and cysteines. this model is based on sequence-based features, that are explained CNN section [24]. The model was tested for each set of features having precision values that range between 68.8% to 74.6% and recall values that range between 78% and 81% [24].

2.3 Feature Extraction

In machine learning (ML), features are the independent variable acting as an input for the system, and the prediction by the model is the dependent variable. Therefore, features represent the protein for the model. Feature extraction is the most crucial step when working with any classifier because the quality of the features will highly influence the model's results. In this section, we will explain some of the techniques used to extract features from proteins.

2.3.1 Sequence-based Features

The majority of the models implemented for the detection of zinc ion-biding use sequence-based features. These features are obtained directly from the protein's primary structure. Therefore, the most used feature is the amino acid composition of the protein, followed by the physiochemical properties of the residues, then the evolutionary profile of the protein, using Position Accepted Mutations (PAM) [9, 36], Position Matrix Scoring (PMS) [9], Position Weight Matrix (PWM) [9], Evolutionary Matrix Scoring (EMS) [9] or Position Specific Scoring Matrix (PSSM) [9, 22], the calculation of the conservation score, the analysis of the amino acid pairs, and finally, position related features, such as sequence length [9, 22].

Since these features derive from the primary structure, they are extracted from FASTA files. FASTA format is the most standard way to represent a sequence of a protein [45]. It is a text-based format to represent sequences of nucleotides or sequences of peptides, where the amino acid or the nucleic acid are represented by a single-letter code [45]. This format begins with the symbol ">" in the first line followed by a brief description of the sequence, followed by lines of sequence data [45]. The Table 1 presents the accepted amino acid codes by FASTA format.

Table 1: Accepted amino acid codes by FASTA format

Single-letter code	**Three-letter code**	**Amino acid name**
A	ALA	Alanine
B	ASX	Aspartate or Asparagine
C	CYS	Cysteine
D	ASP	Aspartate
E	GLU	Glutamate
F	PHE	Phenylalanine
G	GLY	Glycine
H	HIS	Histidine
I	ILE	Isoleucine
K	LYS	Lysine
L	LEU	Leucine
M	MET	Methionine
N	ASN	Asparagine
P	PRO	Proline
Q	GLN	Glutamine
R	ARG	Arginine
S	SER	Serine
T	THR	Threonine
U		Selenocysteine
V	VAL	Valine
W	TRP	Tryptophan
Y	TYR	Tyrosine
Z	GLX	Glutamate or Glutamine
X		any
*		translation stop
-		gap of indeterminate length

2.3.2 Structure-based Features

Structure-based features are obtained from the secondary and tertiary structure of the protein. These types of features are less used since most proteins do not have a determined structure. Therefore, the most used structural feature is the secondary structure of the residue, followed by solvent exposure of the residue, then the solvent accessibility of the residue, the B-factor, and the residue spatial cluster properties [9, 22].

These features are extracted by specific software from PDB data. The PDB data can have several formats, such as PDB, mmCIF and XML files. These files include the experimentally determined 3D crystal structure or Nuclear Magnetic Resonance (NMR) of the protein by listing the atoms that compose it. They can also include citation information and more details of the structure in the header of the file [46].

Chapter 3

Dataset & Methods - Detection of zinc ion in regulatory proteins

After the analysis of the state of the art, we decided to focus on deep learning methods since these approaches are the latest in the field. Therefore, we tested with a dataset generated by us the models proposed by Ismail Haberal and Hasan Ogul in [24].

Despite the number of features that can be collected from a protein, there is not an optimal set of features for the prediction of zinc ion-biding residues. The models created to solve this problem have some differences, such as distinct algorithms ranging from linear regression models, where the obtained results are good and computational costs keep low, to deep learning approaches that are showing improvement with higher performances. Various sets of features are also chosen, being the ones related to the sequence of the protein easier to obtain, although the structure based ones can offer more complex information to the algorithms. Therefore, it is not easy to make a direct comparison between them and find an optimal set. However, the majority of the models uses only sequence-based features. Hence, our approach is based on choosing sequence-based features as well as structure-based features to produce a model with heterogeneous features. Therefore, we chose the binary representation of the aminoacid sequence, PSSM and its

conservation score, secondary structure features as well as solvent accessible surface area related to tertiary structure.

A typical ML model workflow consists of the following processes: raw data collection, raw data preprocessing, feature extraction, feature preprocessing and selection or reduction, model training and classification, postprocessing, and performance evaluation [47]. In this chapter, we will describe the workflow of our methods.

3.1 Generating Dataset

Due to the specificity of our problem, we created a dataset of regulatory proteins with zinc-binding sites. To achieve that, we made two approaches. In the first approach TF databases were searched and then filtered for the proteins with zinc-binding sites. The second approach was by searching for databases with zinc-binding proteins and filter the proteins with regulatory functions.

3.1.1 First Approach

In the first approach, we searched for the latest databases with TF. Therefore, we chose three databases, the HOCOMOCO database [48, 49], the TransmiR database [50, 51], and the Jaspar 2020 database [52, 53].

The HOCOMOCO database contains TF represented by their amino acid sequence and the belonging species. The TransmiR database contains the nucleotide sequence of the TF, as well as the belonging species. The Jaspar 2020 database contains only the amino acid sequence of the TF. Subsequently, by using the UniProt [54], ID mapping [55] and entry retrieving [56] we programmatically extracted the proteins structure IDs for the HOCOMOCO, Jaspar 2020, and TransmiR databases. Moreover, for the TransmiR data, we also extracted the protein sequence IDs, ending up with a total of 21,233 proteins,

described by the gene ID, sequence ID and structure ID. In the Table 2 we present a brief summary of this process.

Table 2: First approach database processing

Database	**Year**	**Present data**	**Extracted data**	**Nº Proteins**
HOCOMOCO [48]	2017	Specie & AA Sequence ID	Structure ID	6,793
TrasmiR [50]	2018	Specie & N Sequence ID	AA Sequence ID & Structure ID	10,704
Jaspar 2020 [52]	2020	AA Sequence ID	Structure ID	3,736
Total				21,233

AA - Amino acid; N-Nucleotide

Then, the next step was the overlapping of the information of the three databases. First, we overlapped the trio gene ID, sequence ID and structure ID, reducing the set of proteins to 16,669 entries. Secondly, we overlapped only the structure IDs reducing even more the set to 13,260 entries. Finally, when downloading the PDB files programmatically, we excluded more 6,139 proteins due to the lack of PDB files in the RCSB PDB [57]. Thus, the final set of proteins was composed of 7,121 TF. This number is very high due to the fact that, sometimes , the set contains the same protein in different conformations, and for that reason, the same sequence can have many structure IDs.

The following stage was filtering the proteins with zinc-binding sites from the ones that did not have zinc-binding sites. For this task, we used the library BioServices [58] from Python, to search for the monomers bounded to the protein. Thus, we end up with 1,052 entries with zinc-binding residues and 6,069 entries without zinc-binding residues.

The last step was the feature extraction from the proteins. However, during this process, we encounter some issues with the labelling of the class of the residues (zinc-binding or non-zinc-binding). Since a vast part of the files were older entries, some of the fields of the PDB files were not correctly filled. Hence, we were not able to label the sequences programmatically. After the feature extraction and labelling of a part of the dataset we decided to analyse it, concluding that it did not have the quality expected since the length of the sequences with zinc-binding residues was very low, between 30 and 200, with

a majority of lengths lower than 50 amino acids. Moreover, the Linear Pearson's correlation coefficient between the features and the target vector was analysed. The Figure 5A presents the values of the correlation coefficients between the sequence based features and target showing a strong correlation between zinc-binding sites with the amino acids cysteine and histidine. The Figure 5B shows the correlation coefficient between evolutionary based features and target presenting again the strong correlation with the amino acid cysteine. However, the structure based features presented shallow values, as shown in Figure 5C. For these reasons, we decided to go for a different approach to construct our dataset.

Pearson's Linear Correlation Coefficient is the most popular linear correlation coefficient and it is defined as [59]:

$$rho(a,b) = \frac{\sum_{i=1}^{n}\left(X_{a,i} - \bar{X_a}\right)\left(Y_{b,i} - \bar{Y_b}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_{a,i} - \bar{X_a}\right)^2 \sum_{j=i}^{n}\left(Y_{b,j} - \bar{Y_b}\right)^2}} \tag{2}$$

where X and Y are matrices with columns X_a e Y_b, having means $\bar{X_a} = \sum_{i=1}^{n}\left(X_{a,i}\right)/n$ and $\bar{Y_a} = \sum_{j=1}^{n}\left(Y_{b,j}\right)/n$. The result of the correlation coefficient ranges between -1 and 1. When the value is 0, there is no linear dependency between the variables, i.e., the feature has no linear dependency with the target. A positive correlation coefficient means that X_a and Y_b are simultaneously higher or lower than their respective means. Thus, when the correlation coefficient is negative, X_a and Y_b are on opposite sites of their respective means. In conclusion, the greater the absolute value, the stronger the relation between the variables [59, 60].

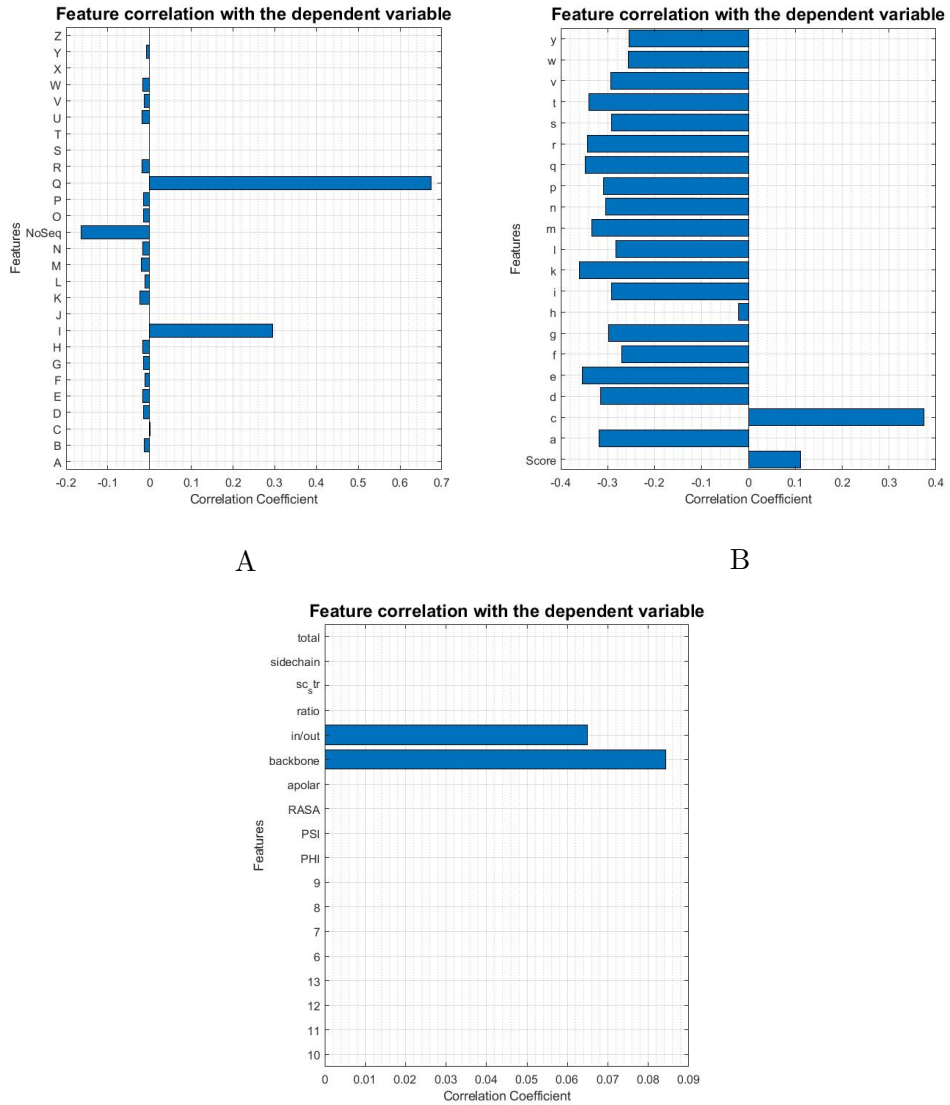


Figure 5: Feature correlation with the target vector of the first approach. (A) Presents the correlation with the sequence-based features. (B) Presents the correlation with evolutionary-based features. (B) Presents the correlation with structure-based features. The features are detailed bellow in Table 3

3.1.2 Second Approach

In the second approach, we searched for the most recent databases with zinc-binding proteins. We chose the ZincBindDB [39], which was released in 2019 and has 24,992 zinc-binding sites, this database uses the structural data from the RCSB PDB [57] and

identifies every detail about the binding site in the protein. This database presents an API [61] for programmatic access.

The first step was filtering regulatory proteins from the database. Therefore, we query for regulatory proteins with the following keywords: *transcription*, *transcription factor*, *regulatory proteins*, and *regulation site*. Obtaining for each query 8, 586, 85, and 1 entries, respectively, making a total of 662 entries. Before the next step, we removed repeated entries. Then, we query for the chains with zinc-binding sites with the structure IDs, obtaining a set of files with structure ID, chain ID, and zinc ion ID.

In the following stage, we finally constructed our set of proteins by arranging a table with the columns *Structure ID*, *Chain ID*, *Zinc ion ID*, and *Sequence*. Subsequently, we produced the FASTA files for each chain, as well as we downloaded the PDB files for each protein. After removing entries without PDB files, we accomplished our dataset with the positive class, proteins with zinc-binding residues, composed of 507 proteins, 1037 chains, and 1732 zinc-binding sites.

To add negative class examples (proteins without zinc-binding sites), we chose 3000 random proteins from the the set without zinc-binding sites obtained in the first approach. From this set, we excluded the proteins with sequence length bigger than 5000 and lower than 30 amino acids. Finally, we chose 1725 random chains and added to our dataset the respective FASTA and PDB files.

3.1.3 Validation Dataset

The dataset that we used to validate our model was the benchmark dataset proposed by Passerini *et al.* [62] adopted by Haberal *et al.* [24].

This dataset contains 2,727 unique protein chains with 640 zinc-binding sites. However, it covers eight metal ion or complex binding sites, such as zinc, heme, iron/sulfur, copper, cadmium, iron, nickel, and others. Thus, due to the difference in the number of proteins with and without zinc-bing sites, we chose only 640 chains without zinc-binding

residues to constitute the validation dataset. This dataset has 0.33% of residues that bind to zinc and 99.67% that do not.

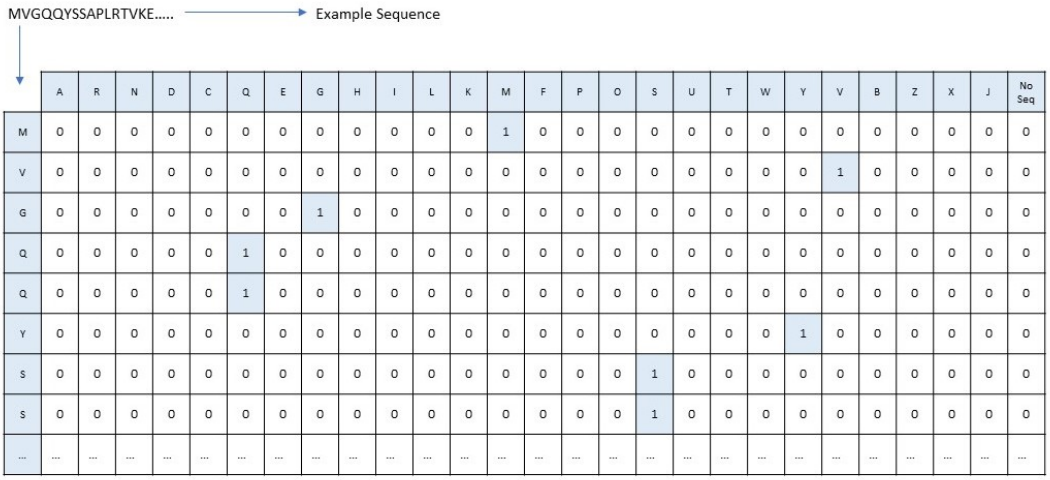
3.2 Feature Extraction & Engineering

After obtaining the final dataset, we proceed to the feature extraction. In contrast to Haberal *et al.* [24], which only used sequence-based features, we find it interesting to experiment with their proposed models, a set of heterogeneous features, with sequence based features as well as with structure based features.

3.2.1 Sequence based features

To extract sequence based features, we used two methods, the binary representation of the amino acids, and the Position Specific Scoring Matrix (PSSM) as well as the respective conservation score. The binary representation of the amino acid consists in the representation of each amino acid by a 27-vector one-hot vector. Hence, each amino acid has the assigned value of 1 in its position in the list of amino acids, and the remaining 26 values are 0, as shown in Figure 6.

The PSSM is generated from the Position-Specific Iterative Basic Local Alignment Search Tool (PSI-BLAST). This tool builds alignments generated by the BLASTp, that it is a sequence similarity search method for proteins [63]. Briefly, BLASTp reports alignments that score above a specified threshold. These alignments are identified by comparing a protein sequence with a specified database. Therefore, the PSSM, by storing the scores of each position of the alignment in a matrix, catches the conservation pattern of the alignment [63]. Due to the iterative nature of PSI-BLAST, this algorithm has a bigger capacity to detect distant sequence similarities. Therefore, has been demonstrated that PSI-BLAST is capable of detecting conservation even in the three-dimensional struc-



	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	O	S	U	T	W	Y	V	B	Z	X	J	No Seq
M	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
G	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 6: Binary representation of an amino acid sequence

ture of the proteins by sequence searches [63–65].

To calculate the PSSM, we used the BLAST program from NCBI [64] services programmatically with the BioPython library [66], using the default E-value = 0.005 cutoff, with three iterations, and the *pdb* database [67] since we are working only with proteins with documented tridimensional structure. Hence, for each protein we obtained matrices of features with Lx20 size, where L is the sequence length.

The conservation score was calculated for each residue from the values of the PSSM and is defined as:

$$Score_i = -\sum_{j=1}^{20} p_{i,j} log_2 p_{i,j} \tag{3}$$

where $p_{i,j}$ is the frequency of amino acid j at position i. The PSSM and conservation score can also be designated as evolutionary based features.

3.2.2 Structure based features

Structure based features were obtained with two algorithms, the hydrogen bond estimation algorithm (DSSP) [68, 69] and the Fast Newton-Raphson Torsion Angle Minimizer

(FANTOM) [70–72].

With the DSSP program, we obtained the secondary structure in which the residue is involved. Therefore, we extracted 12 features: secondary structure, relative accessible surface area, protein backbone torsion angles PHI and PSI, and eight other features that describe the hydrogen bounds, giving their location and energy.

The FANTOM program has a fast routine, GETAREA [72], to calculate the Solvent Accessible Surface Areas (SASA) of individual atoms and their gradients. Hence, we obtained the total SASA, the apolar area, the contribution for the SASA from the backbone and from the side-chains, the ratio of side-chain and surface area to "random coil", as well as an classification of if the residue is exposed to the solvent or if it is buried in the protein structure.

Table 3 summarises the features extracted from the protein's sequence and structure. Having features derived from the primary, secondary and tertiary forms.

After cleaning the data from the feature extraction process, we ended up with a dataset that contains:

- 1037 chains and 1724 zinc-binding sites (507 proteins)
- 1701 chains for the class of non-zinc-binding sites

The new values for the correlation coefficients, showed in Figure 7, are better than the values from the data in the first approach. The overall absolute values are larger, showing a better relation of the features with the target. Specifically, cysteine and histidine amino acids have the most significant values showing their strong relation with zinc-binding sites since they are the most abundant residues in these sites, Figure 7A, the same property about cysteine is presented in Figure 7B. The third most significant feature is the secondary structure, Figure 7C third bar, demonstrating that the residues

Table 3: Summary of the extracted features

Features		**Nº**
Binary representation of the amino acids	A, R, N, D, C, Q E, G, H, I, L, K M, F, P, O, S U, T, W, Y, V B, Z, X, J, NoSeq	27
PSSM	A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V	20
Conservation Score	$Score_i = -\sum_{j=1}^{20} p_{i,j} log_2 p_{i,j}$	1
DSSP	Secondary Structure RASA PHI PSI Hydrogen bound 1 (energy and location) Hydrogen bound 2 (energy and location)	12
SASA	Total SASA Apolar area Backbone Influence Side chain Influence Ratio In/Out	6
Total		66

that form zinc-binding sites may have a prefered secondary structure.

During the process of collection of the dataset, we tried to balance the number of sequences for both classes of this problem. However, since we are classifying residues, our dataset is extremely unbalanced, having 0.51% of residues that bind to zinc and 99.49% that do not.

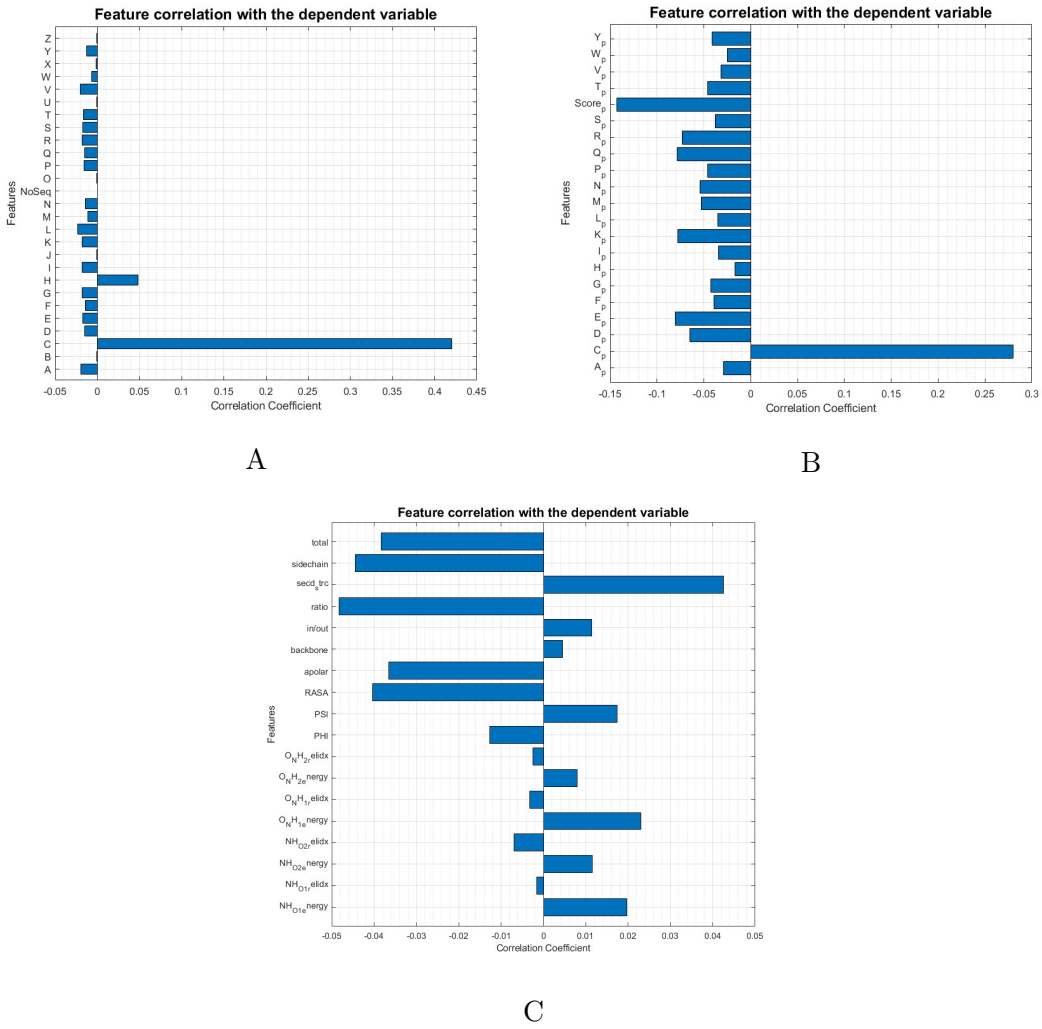


Figure 7: Feature correlation with the target vector of the second approach. (A) Presents the correlation with the sequence-based features. (B) Presents the correlation with evolutionary-based features. (C) Presents the correlation with structure-based features.

3.2.3 Feature Selection

In deep learning algorithms, it is not usual to apply methods for feature selection since the model has the capacity to work with the features that have more impact on the prediction. However, we made an approach of feature selection to analyse the behaviour of the model with the complete dataset and with a reduced dataset. We realise that the

chances of the performance enhancement of the model with a smaller dataset are low. However, if the performance is not impaired, it is an advantage to work with a smaller dataset since we can shorten memory usage and time consumption, resulting in lower computational cost while training.

There are three types of feature selection methods: the filters, the wrappers and the embedded. The filters preprocess the features independently from the classifier, the wrappers select the features depending on a given classifier, and the embedded methods are integrated into the classification algorithm [73].

Since feature selection may not have a big positive impact on the performance of the algorithm, we followed a very simple approach by choosing the Boruta algorithm [74, 75], which is a wrapped filter based on RF classifier. The Boruta algorithm starts by creating random copies of all features, the shadow features. Then, it trains the RF with the extended set of features applying a measure to calculate the importance of each feature, the default measure is mean decrease accuracy. Therefore, in each iteration, the algorithm checks if the real feature is more important than the shadow feature, removing the ones that do not have importance [74, 75].

Python has the Boruta library [74, 76], with very useful methods that simplify the task by retrieving a list with the ranks of each feature (where 1 means best feature). Thus, the model determined that the least important features are the binary representation of N (asparagine), Q (glutamine), M (methionine), O (pyrrolysine), U (selenocysteine), B (aspartate), Z (glutamate), X (any), J (leucine) and NoSeq (no sequence). In conclusion, the method excluded 10 features, hence, we can train the models with a datatset of 66 features or a dataset with 56 features, and compare their performances.

3.3 Implemented Models

3.3.1 Convolutional Neural Networks

As referred earlier, CNN are very powerful neural networks for image analysis. They are composed by multilayer feedforward, with many hidden layers as well as different types of layers [37, 77]. The number of weights to be learned in CNN algorithms can be reduced since some layers may not be fully connected. Although this algorithm is not the most appropriate for sequential data problems, there are some adaptions that can be performed to the data to make this application possible for CNNs [77].

The CNN model proposed by Haberal *et al.*[24] is composed of four convolution layers (2D CNN layers), two pooling layers and two multi-layers perceptron layers, like the following:

- 2D Convolutional 1 with input of 280, output of 128, and kernel size of 3x3
- 2D Convolutional 2 with input of 128,output of 96, and kernel size of 3x3
- 2D Convolutional 3 with input of 96, output of 64, and kernel size of 3x3
- 2D Convolutional 4 with input of 64, output of 32, kernel size of 3x3 and ReLU activation
- 2D Max Pooling with input of 32 and output of 64, with a pooling window of 2x2
- Dropout layer with a rate of 0.15
- Dense layer with output of 64 and Rectified Linear Unit (ReLU) activation
- Dropout layer with a rate of 0.15
- Classification layer (dense layer) with output of 2 and Softmax activation

The convolution layer creates a tensor of outputs by convolving a created convolution kernel with the layer input. Therefore, with this process, the input image is turned into an abstracted feature map, designated as an activation map. The input for this layer is the (number of inputs) x (input height) x (input width) x (input channels) and returns an output of (number of inputs) x (feature map height) x (feature map width) x (feature

map channels) [77].

Max Pooling layers reduce de dimensions of data by dividing their input into rectangular pooling regions and then computing the maximum of each region for the output feature map [77].

Fully connected layers, such as dense layers, are multilayer perceptrons that flatten the input matrix to an output vector. All of the neurons of these layers are connected to the neurons of the previous layer. In classification problems, the output size of the last fully connected layer is the number of classes of the problem.

Dropout layers set random input elements to zero with a given probability [77].

ReLU activation function applies a threshold operation where all the negative values are set to zero, $R(z) = max(0, z)$. This function is especially useful in CNNs because these neural networks were developed for image processing, where the data is presented in the range between $[0, 1]$, characterising the pixel's intensity. Therefore, does not make sense to have negative values in the feature map, and ReLU activation rectifies the values. However, this activation function is not useful in all applications [77].

Softmax activation function takes as input a vector of raw outputs from the previous layer and returns a vector of probability scores [77]. With x as the input vector and y as the output vector, the r^{th} component of the output is:

$$y_r\left(x\right) = \frac{e^{a_r(x)}}{\sum_{j=1}^{k} e^{a_j(x)}} \; where \; a_r(x) = ln(P(x, \theta|c_r)P(c_r)) \tag{4}$$

where k is the number of classes, and the sum of the probabilities of each class in the prediction is equal to 1. This function is particularly useful for classification problems in the classification layer, returning the probability distribution of each class, therefore is highly adopted in multi-label problems.

CNNs models require specific pre-processing. Therefore the first step is to normalise

the values of the features between 0 and 1, to mimic the greyscale pixel's intensity [37]. The normalisation was made with the Min-Max Scaling approach [37], where the feature values are scaled between 0 and 1.

Then finally, we constructed our images by extracting windows from the data. Haberal *et al.* [24] experimented their models with various windows size and concluded that windows of 15 amino acids were the best size. Therefore, to classify each amino acid, we made a window of size 15, where we classify the amino acid in the middle position. To simplify the process, we ignored the first and last seven amino acids since they do not influence on zinc-binding sites. The Figure 8 shows the scheme of the process:

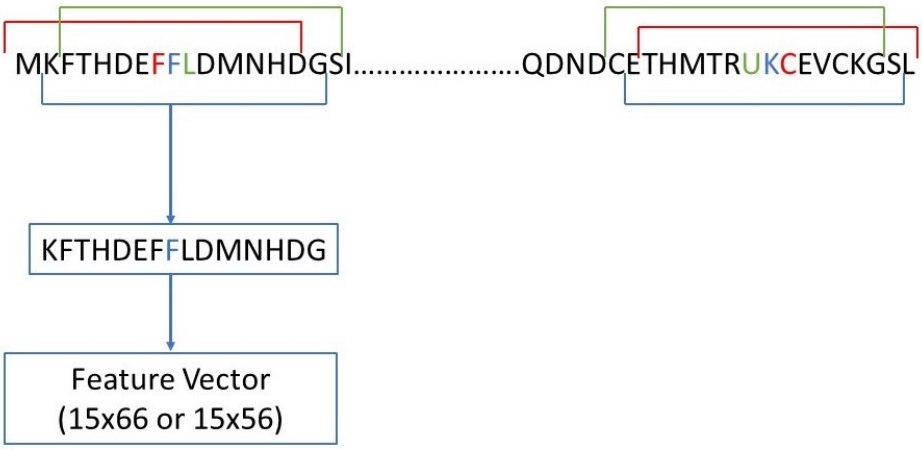


Figure 8: Schematic of window extraction

3.3.2 Long-Short Term Memory Neural Networks

LSTMs are RNN created to solve the memory problems of RNNs. They have a high number of serial blocks that constitute the long term memory, but each block only uses the state of the previous block, which defines the short term memory [42, 77].

Haberal *et al.* [24] proposed an LSTM model with four LSTM layers, one dropout layer, and one fully connected layer, such as:

- LSTM 1, 2, 3, and 4 with an output of 50
- Dropout layer with a rate of 0.2
- Classification layer (dense layer) with output of 2 and Softmax activation

The core of the LSTM block, as presented in Figure 9, is the cell state c, and the parameters to be learned are the input weight matrix (5), the recurrent weight matrix (6) and the bias (7).

$$W = [W_i \ W_f \ W_g \ W_o] \tag{5}$$

$$R = [R_i \ R_f \ R_g \ R_o] \tag{6}$$

$$b = [b_i \ b_f \ b_g \ b_o] \tag{7}$$

First, c_{t-1}, the previous cell state, passes through the forget gate, where it is multiplied by f_t function (8). Where x_t is the input features, h_{t-1} is the previous hidden state, and σ_g is the sigmoid activation function (9), that returns values between 0 and 1.

$$f_t = \sigma_g(W_i x_t + R_i h_{t-1} + b_i) \tag{8}$$

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{9}$$

Then, in the input gate, the cell candidate (10) where σ_c is the tanh activation function that returns values between -1 and 1, is multiplied by i_t function (11). The obtained result is summed in the update gate with the cell state generating the new cell state c_t. Finally, c_t passes through the tanh activation function and is multiplied by o_t function (12), in the output gate generating a new hidden state. The new cell state and the new hidden state will be the input of the next block, and so on.

$$g_t = \sigma_c(W_g x_t + R_g h_{t-1} + b_g) \tag{10}$$

$$i_t = \sigma_c(W_i x_t + R_i h_{t-1} + b_i) \tag{11}$$

$$o_t = \sigma_c(W_o x_t + R_o h_{t-1} + b_o) \tag{12}$$

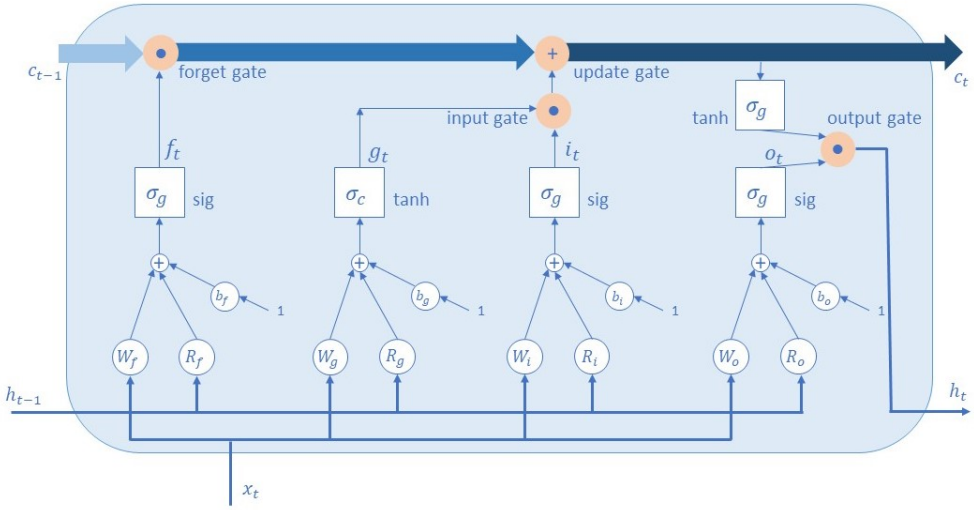


Figure 9: LSTM block (Adapted from António Dourado [77])

3.3.3 Gated Recurrent Units

GRU are the new generation of RNNs. They are very similar to LSTMs, however, they lack the cell state and some gates [42, 44]. The last model proposed by Haberal *et al.* [24] was a GRU model with three GRU layers, one dropout layer, and one dense layer, like the following:

- GRU 1, 2, and 3 with an output of 256
- Dropout layer with a rate of 0.2
- Classification layer (dense layer) with output of 2 and Softmax activation

A GRU block has an update gate that has the same function as the forget and input gates from the LSTM. The second gate is the reset gate that, with a sigmoid activation function, decides which information passes or not. The cell state was substituted by the hidden state, and all the operations are made in the hidden state [42, 44].

For these two models, the pre-processing of the data was made by standardisation,

where the features were transformed by subtracting the mean and then divided by the standard deviation. This transformation creates a dataset with features that have a mean of zero and unit standard deviation.

LSTMs and GRUs accept sequential data. Therefore, two approaches for the length of the sequence were made. The first approach was by expanding all the sequences to a length of 2143, which represents the longest sequence in the dataset. And the second approach was setting the length of the sequences to 500 amino acids, with an overlap of 50 amino acids. Therefore the sequences with a lower length were filled with zeros to reach the ideal length, and the sequences with higher length were divided into windows with an overlap of 50 amino acids.

3.4 Experiments

The dataset split was made with the following proportions: 60% for training data, 20% for validation and the remaining 20% for testing. The division into these three sets was made taking into account the original proportions of the classes of the dataset to not aggravate the imbalance.

To adapt the explained models to our problem, we changed the activation function in the CNNs from ReLU to the sigmoid activation function, $sigmoid(x) = \frac{1}{1+e^{-x}}$, which also transforms negative values into positive, as needed in CNNs. The classification layer in the three models was substituted by a dense layer with output 1 and sigmoid activation function. We defined the loss function the binary cross-entropy, which calculates the cross-entropy between true and predicted labels. We also chose the Adam optimiser.
All three models were trained with 100 epochs and batch size of 100 for CNN and LSTM, for GRU the batch size was 1000.

Due to the unbalanced dataset, we helped the algorithms by setting the bias of the output layer as the logarithm of the proportion of positive examples in the dataset, in our case $log(0.0051)$ and for the validation dataset $log(0.0033)$. The models were trained with 5-fold cross-validation, where the test dataset from the dataset split was always preserved for the testing, while the model was trained and validated with the rest of dataset (80%). We used the Stratified K-Folds from scikit learn, to ensure the proportion of the data in the folds. The same strategy was used for the benchmark dataset, however, since this datatset has fewer samples we used 3-fold cross-validation instead.

3.4.1 Experiment 1

The first experiment, had the aim to evaluate the model with the parameters explained earlier, the same used in the state of the art model. Therefore, we intended to examine which feature set and sequence length would bring better results, as well as analysing the model's general performance to outline the path to the second experience. In summary, we will test:

- CNN model with sequences with window size of 15 and with 66 and 56 features.
- LSTM model with sequences of length 2143 and 500 with 66 and 56 features.
- GRU model with with sequences of length 2143 and 500 with 66 and 56 features.

Table 4 presents the implemented models.

Table 4: Summary of Experiment 1

Model	Layers
CNN	2D Conv1(input_shape, 128, (3,3))
	2D Conv2(128, 96, (3,3))
	2D Conv3(96, 64, (3,3))
	2D Conv4(64, 32, (3,3), sigmoid)
	2D MaxPool(32, 64, (2,2))
	Dropout(0.15)
	Dense(64, sigmoid)
	Dropout(0.15)
	Dense(1, sigmoid)
LSTM	LSTM 1(input_shape, 50)
	LSTM 2(50)
	LSTM 3(50)
	LSTM 4(50)
	Dropout(0.2)
	Dense(1, sigmoid)
GRU	GRU 1(input_shpe, 256)
	GRU 2(256)
	GRU 2(256)
	Dropout(0.2)
	Dense(1, sigmoid)
Adam optimiser, binary_crossentropy loss	

3.4.2 Experiment 2

The second round of experiences was conditioned by the results of the previous experience and by the computational resources since the networks already have great complexity. Therefore, here we tried different hyperparameters to enhance the performance of the models. For the CNNs we tested the dense layer with outputs of 32 and 96, the dropout layer was tested with the rate of 0.1. The LSTM model was tested with the output of 20

and 30, and the dropout layer was tested with 0.1 and 0.3 rates. GRU model was tested with the output of 64 and 128, and the dropout was lowered to 0.1.

Finally, the best parameter combination were tested with the learning rate of 0.0001 and 0.00001 since the default rate is 0.001.

Then, the best set of the hyperparameters was tested with the benchmark dataset.

3.5 Metrics

The final step after selecting the models and organising the experiments is the choice of the evaluation metrics. In our particular problem, the metrics were chosen under the condition of an unbalanced dataset.

Therefore, the first step is obtaining the confusion matrix of the model predictions, as shown in Table 5, where ZB is the class zinc-binding residue and NZB is the class non zinc-binding residue.

Table 5: Confusion Matrix

		Predicted	
		ZB	**NZB**
Real	**ZB**	TP	FN
	NZB	FP	TN

TP (True Positives) is the number of well classified zinc-binding residues, TN (True Negatives) is the number of well classified non zinc-binding residues, FP (False Positives) is the number of non zinc-binding residues classified as zinc-binding residues, and FN (False Negatives) is the number of zinc-binding residues classified as non zinc-binding residues.

We must choose the appropriate metrics for models developed with an unbalanced dataset, such as our case which has the positive outcome as the minority class with 0.51% of examples. Therefore, we selected the Precision, Recall, F1-Measure and the Precision-

Recall Area Under the Curve (PR AUC). Although accuracy is the most used metric, it is not appropriate for unbalanced problems since it calculates the rate of correct predictions by the total predictions. Hence, it will have a high value even with a no-skill model due to the high number of correct predictions for the negative class [78].

Precision (13) is the fraction of correctly assigned positive class that belongs to the positive class. Recall (14) shows the positive rate by summarising how well the positive class was predicted. The F1-Measure (15) combines precision and recall balancing both concerns. PR AUC is the calculation of the score of the area under the curve appropriated for unbalanced datasets, Figure 10.

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

$$F1\_Measure = 2 \times \frac{Precision + Recall}{Precision \times Recall} \tag{15}$$

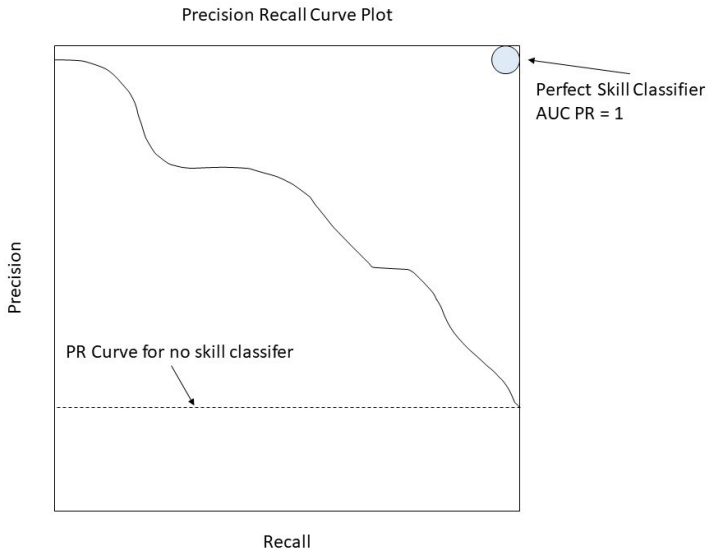


Figure 10: Precision Recall Curve (Adapted from Tour of Evaluation Metrics for Imbalanced Classification, 2020 [79])

In summary, the aim is to achieve the best F1-Measure possible, that it is obtained with the best values for Precision and Recall, as well as the best PR AUC value, closest to 1 as possible.

Chapter 4

Results and Discussion

In the current chapter, we will present and analyse the results of the proposed experiments. As a starting point for the discussion, we present the results from the chosen state of the art methods [24] applied to the benchmark dataset [62], Table 6.

Table 6: Ismail Haberal and Hasan Ogul Results [24]

	Features	F1-Measure (%)	Precision (%)	Recall (%)
CNN	PAM	80.5	79	82
	BR	75.6	77.8	81.2
	PC	73	67	82
LSTM	PAM	73.6	73.2	80.8
	BR	72	66	81
	PC	72.6	72.4	81
GRU	PAM	73.6	74.6	81
	BR	72,2	68.8	81
	PC	72	71	78

PAM - Point Accepted Mutation; BR - Binary representation of amino acids; PC - ProCos

4.1 Experiment 1

In experiment 1 we implemented the state of the art methods with slight changes for our dataset, as explained in the previous chapter. Therefore, we present the mean and standard deviation of the results from the 5-fold cross-validation. The Tables 7-15 show the results organised by the number of features of the dataset (Nº F.) and window size of the data (W).

The CNN model, has low performance with our dataset. Table 7 presents the results for the training stage. The dataset with 66 features presented the highest value for the F1-Measure with 39.123%. However, the dataset with 56 features presents lower and stable standard deviation values of the metrics. In Tables 8 and 9, the results of validation and test are showed, respectively. An opposite behaviour is detected, since the dataset with 56 has the highest value for the F1-Measure with 32.974% for validation and with 30.8% for the test set. The metrics for the dataset with 56 features have lower standard deviation values. This model presents a gap between the values of the metrics for training and validation/test. However, the loss function for training and validation with 56 features shows a good fit since validation values keep up with the training values (Figure 11) (for more information please check Appendix A - Figure A.1). The application of the algorithm to the dataset with 56 features presents more stable values, with a lower standard deviation. Thus, the following experiences with CNN model will be made with only this dataset.

Table 7: Experiment 1 - CNN - Training results

Nº F.	W	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
66	15	**39.123**	57.782	**29.652**	0.462	1.655	2.267	1.805	0.089
56	15	38.710	**57.822**	29.125	**0.517**	**1.247**	**0.643**	**1.329**	**0.013**

SD - Standard Deviation

Table 8: Experiment 1 - CNN - Validation results

Nº F.	W	Mean				SD			
		F1_M%	Precision%	Recall%	PR AUC	F1_M	Precision	Recall	PR AUC
66	15	31.467	57.674	**23.808**	0.519	15.576	5.877	11.463	0.039
56	15	**32.974**	**62.756**	22.574	**0.525**	**3.910**	**2.102**	**3.273**	**0.018**

SD - Standard Deviation

Table 9: Experiment 1 - CNN - Testing results

Nº F.	W	Mean				SD			
		F1_M%	Precision%	Recall%	PR AUC	F1_M	Precision	Recall	PR AUC
66	15	29.416	56.864	**22.006**	0.496	14.689	3.892	10.786	0.029
56	15	**30.800**	**61.292**	20.834	**0.502**	**4.713**	**2.299**	**3.776**	**0.012**

SD - Standard Deviation

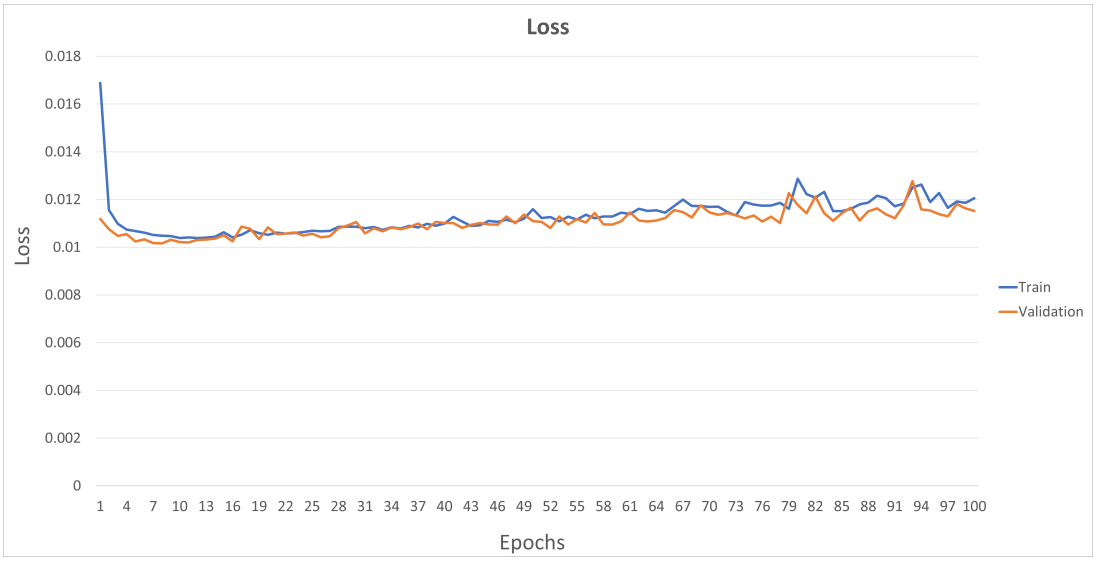


Figure 11: Loss of CNN model, with 56 features

The LSTM model shows overall better results than the CNN model. During the training stage, Table 10, the results are closer to those of the state of the art. For the dataset with the 56 features and window size of 500 residues, we obtained an F1-Measure of 70.115%. Tables 11 and 12 present the highest results for the dataset with 56 features and with the window size of 2143 residues, with F1-Measure values of 49.614% for validation and 40.660% for test. The standard deviation of the metrics in this model does not influence the choice for a better dataset since the values are low as well as scattered between the datasets, however, in the next experiment we will use the window with 500 residues for the reduction in computational costs. In this model we see an even bigger gap between the values of training and validation/test. Figure 12 presents the loss curves showing that the model suffers from overfitting, since the validation set has higher values for the loss function (for more information please check Appendix A - Figure A.2). In the next experience we aim to reduce this gap, because for an ideal fit the curves should almost overlap.

Table 10: Experiment 1 - LSTM - Training results

Nº F.	W	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
66	2143	68.715	76.919	62.410	0.842	1.159	0.881	1.225	**0.009**
66	500	69.551	78.202	62.985	0.853	**0.684**	**0.873**	**0.841**	**0.009**
56	2143	69.740	77.732	63.357	0.850	1.899	1.642	1.958	0.018
56	500	**70.115**	**78.517**	**63.810**	**0.857**	1.376	1.846	1.183	0.012

SD - Standard Deviation

Table 11: Experiment 1 - LSTM - Validation results

Nº F.	W	Mean				SD			
		F1_M%	Precision%	Recall%	PR AUC	F1_M	Precision	Recall	PR AUC
66	2143	49.252	**54.738**	45.016	**0.559**	3.062	**2.253**	3.473	0.049
66	500	49.547	52.737	46.673	0.515	4.325	4.962	3.197	0.056
56	2143	**49.614**	52.036	**47.199**	0.525	**2.370**	2.485	**2.112**	**0.044**
56	500	48.289	52.328	44.941	0.509	4.466	4.254	4.067	0.051

SD - Standard Deviation

Table 12: Experiment 1 - LSTM - Testing results

Nº F.	W	Mean				SD			
		F1_M%	Precision%	Recall%	PR AUC	F1_M	Precision	Recall	PR AUC
66	2143	39.759	**43.806**	36.207	**0.385**	**1.436**	2.762	1.904	**0.042**
66	500	39.848	41.554	37.702	0.346	2.365	**1.934**	2.614	0.062
56	2143	**40.660**	42.584	**38.096**	0.360	2.492	2.855	2.546	0.076
56	500	38.547	41.384	35.698	0.332	2.047	2.552	**1.664**	0.071

SD - Standard Deviation

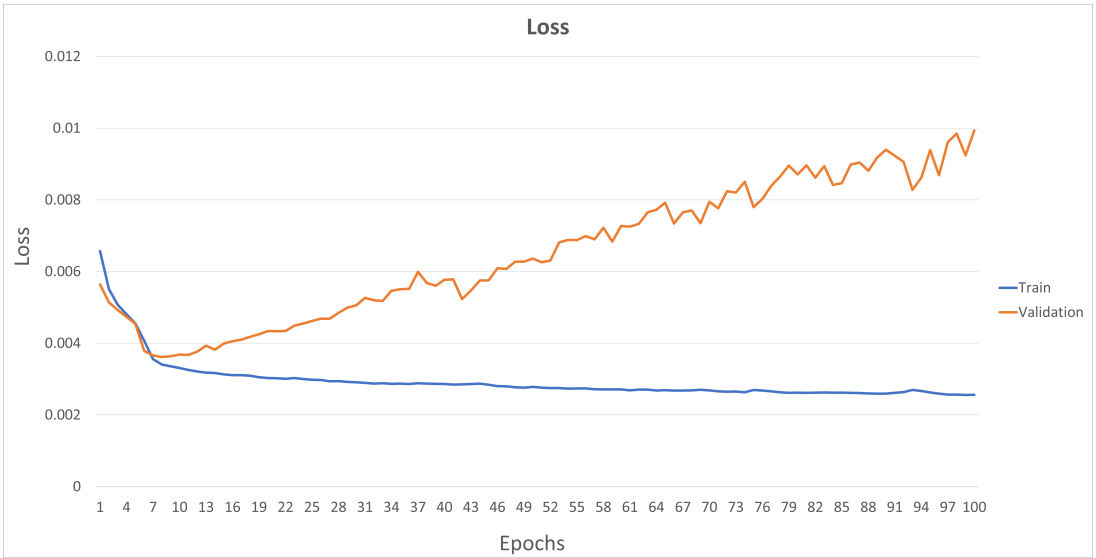


Figure 12: Loss of LSTM model, with 56 features and window size 500 residues

The GRU model presents the best results so far. Table 13 shows the highest F1-Measure of 72.251% for the dataset with 66 features and window size of 500 residues. This dataset has also the lowest values for the standard deviation of the metrics. However, during the validation and test the dataset with the best performance is the one with 56 features and window size of 500, with F1-Measure of 47.645% and 39.260%, respectively (Tables 14 and 15). Likewise, the previous model we have the same problem with overfitting when analysing the loss curve, as shown in Figure 13 (for more information please check Appendix A - Figure A.3).

Table 13: Experiment 1 - GRU - Training results

Nº F.	W	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
66	2143	71.622	78.560	66.029	0.872	1.198	**0.497**	1.681	0.011
66	500	**72.251**	**78.956**	**67.013**	**0.877**	**0.609**	0.889	**1.238**	**0.005**
56	2143	71.716	78.546	66.336	0.872	1.716	1.304	2.009	0.013
56	500	72.149	78.800	66.861	0.874	0.759	0.799	1.355	0.007

SD - Standard Deviation

Table 14: Experiment 1 - GRU - Validation results

Nº F.	W	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
66	2143	47.447	**52.150**	43.627	**0.451**	**2.145**	3.180	**0.830**	0.041
66	500	46.248	51.258	42.254	0.396	4.813	3.855	4.944	0.034
56	2143	46.574	51.553	42.522	0.430	3.495	**3.017**	3.004	0.045
56	500	**47.645**	48.813	**46.972**	0.421	5.107	3.665	5.764	**0.026**

SD - Standard Deviation

Table 15: Experiment 1 - GRU - Testing results

Nº F.	W	Mean				SD			
		F1_M%	Precision%	Recall%	PR AUC	F1_M	Precision	Recall	PR AUC
66	2143	37.253	40.263	34.164	0.290	2.244	2.824	1.648	0.073
66	500	37.648	**41.327**	34.526	0.292	1.648	2.227	1.131	0.084
56	2143	37.953	40.495	34.381	0.291	**1.208**	**2.112**	**0.732**	0.072
56	500	**39.260**	39.886	**38.566**	**0.294**	2.844	2.367	4.251	**0.060**

SD - Standard Deviation

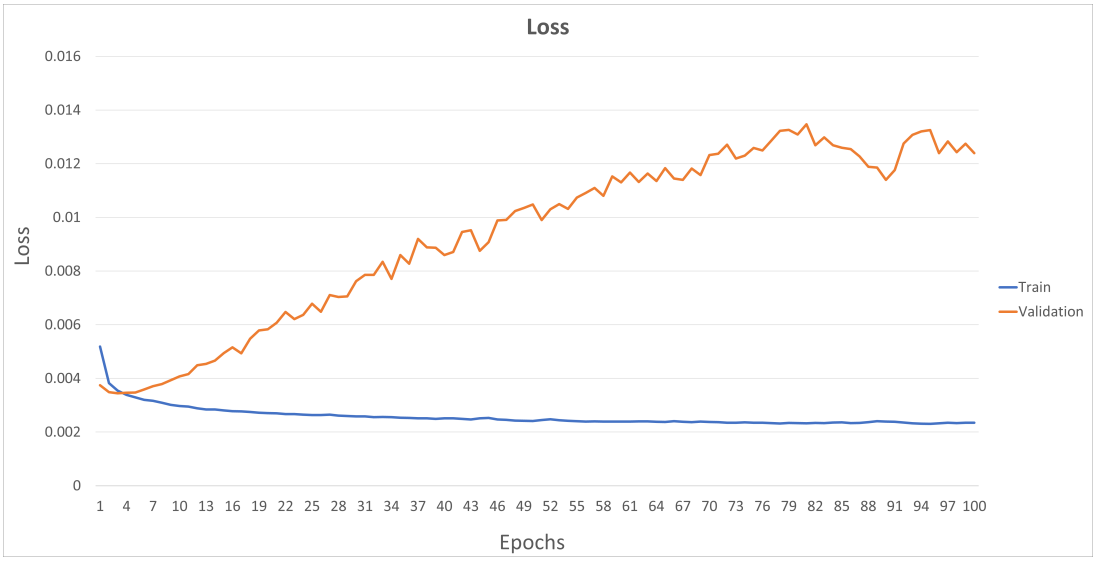


Figure 13: Loss of GRU model, with 56 features and window size 500 residues

4.2 Experiment 2

The state of the art methods are quite complex, however high complexity does not mean a better performance. Therefore, in this experiment, we evaluate the performance of the models with different parameters and sometimes simpler parameters.

First, we start by the CNN model, where we tested this model with different output sizes for the dense layer, like 32 and 96, and one more dropout rate of 0.1. Since the

dataset with 56 features presents more stable and higher values we excluded the datatset with 66 features from the experiments with this model.

In the training process, Table 16, the model with the dense layer with output size of 96 and dropout rate of 0.1 achieved the best result with a value of F1-Measure of 42.460%, higher than the previous model. During validation, Table 17, and test, Table 18, the dense layer with output size of 32 and the dropout rate of 0.1 had the best performance with a F1-Measure of 40.717% and 38.291%, respectively. The difference in performance between the training and validation/test also decreased.

Table 16: Experiment 2 - CNN - Training results

DLS	D	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
32	0.1	42.162	**60.122**	32.962	**0.552**	3.792	3.385	5.254	**0.010**
32	0.15	37.972	55.888	28.909	0.500	3.129	1.566	3.535	0.016
64	0.1	40.639	57.076	31.552	0.525	3.365	2.099	3.384	0.026
96	0.1	**42.460**	58.707	**33.303**	0.541	**1.764**	**0.733**	**2.071**	0.012
96	0.15	39.822	56.440	30.917	0.510	2.194	1.332	3.084	**0.010**

SD - Standard Deviation; DLS - Dense Layer output size; D - Dropout Rate

Table 17: Experiment 2 - CNN - Validation results

DLS	D	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
32	0.1	**40.717**	59.714	**32.117**	**0.545**	**5.129**	6.269	7.634	0.036
32	0.15	37.150	58.599	29.102	0.508	9.240	4.858	11.007	0.031
64	0.1	38.666	59.954	29.223	0.538	5.357	**4.293**	**6.241**	**0.021**
96	0.1	34.550	**61.921**	25.154	0.536	8.115	5.958	8.296	0.031
96	0.15	32.739	61.300	24.119	0.514	9.975	7.751	9.504	**0.021**

SD - Standard Deviation; DLS - Dense Layer output size; D - Dropout Rate

Table 18: Experiment 2 - CNN - Testing results

DLS	D	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
32	0.1	**38.291**	58.477	**30.050**	**0.517**	6.593	6.359	9.324	0.034
32	0.15	35.057	54.790	27.475	0.482	9.107	**3.171**	10.574	0.021
64	0.1	37.498	59.845	27.993	0.516	**4.354**	6.335	**5.191**	0.027
96	0.1	33.629	**62.046**	24.365	0.521	8.239	6.176	8.494	0.018
96	0.15	31.508	60.567	23.161	0.495	9.987	7.151	10.006	**0.014**

SD - Standard Deviation; DLS - Dense Layer output size; D - Dropout Rate

The LSTM model was tested with two more layer sizes for the LSTM layers, 20 and 30, and two more dropout rates, 0.1 and 0.3. We present the results for the dataset with 56 features and window size of 500 since in the previous experiment was the one with slightly better results.

During training, Table 19, the model with LSTM layer size of 50 and dropout rate of 0.1 has the highest F1-Measure of 70.199%. However, when validating, Table 20, LSTM layer size of 20 and dropout rate of 0.1 had the best performance with F1-Measure of 52.342%. During testing, Table 21, LSTM layer size of 30 and dropout rate of 0.3 had the highest value of F1-Measure of 41.695%. These experiments do not show a big improvement from the previous LSTM model as well as do not improve the decrease of the gap of the results between training and validation/test.

Table 19: Experiment 2 - LSTM - Training results

LS	D	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
20	0.1	65.539	72.532	59.903	0.789	1.685	1.865	1.376	0.027
20	0.3	64.133	72.867	57.744	0.784	2.201	2.478	2.428	0.035
30	0.1	68.724	74.435	64.138	0.826	2.126	1.802	2.424	0.026
30	0.3	67.520	75.475	61.039	0.817	1.868	**1.596**	2.016	0.025
50	0.1	**70.199**	76.082	**65.450**	**0.847**	1.836	2.925	**1.145**	0.022
50	0.3	69.273	**76.764**	63.465	0.842	**1.231**	1.806	1.632	**0.017**

SD - Standard Deviation; LS - LSTM Layer Size; D - Dropout Rate

Table 20: Experiment 2 - LSTM - Validation results

LS	D	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
20	0.1	**52.342**	**60.645**	**46.214**	**0.629**	4.964	4.451	5.675	0.064
20	0.3	48.828	59.253	42.509	0.618	4.175	6.852	6.504	0.061
30	0.1	49.351	54.253	45.767	0.616	4.547	**3.946**	4.520	0.053
30	0.3	49.607	55.102	45.845	0.599	4.352	6.485	3.646	0.050
50	0.1	48.985	53.538	45.534	0.544	**3.766**	5.190	**3.243**	0.050
50	0.3	48.891	53.793	45.387	0.558	4.885	4.822	6.710	**0.038**

SD - Standard Deviation; LS - LSTM Layer Size; D - Dropout Rate

Table 21: Experiment 2 - LSTM - Testing results

LS	D	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
20	0.1	41.314	48.466	36.299	**0.499**	**0.928**	3.898	1.895	**0.038**
20	0.3	40.692	**48.829**	35.544	0.487	3.513	2.862	4.726	0.045
30	0.1	39.194	42.684	35.883	0.460	2.139	3.108	**1.737**	0.051
30	0.3	**41.695**	43.658	**39.260**	0.435	2.455	**1.840**	3.356	0.079
50	0.1	38.782	41.830	36.284	0.371	3.161	3.281	3.391	0.088
50	0.3	40.071	43.084	37.271	0.388	2.612	2.771	3.315	0.079

SD - Standard Deviation; LS - LSTM Layer Size; D - Dropout Rate

The GRU model was tested with two more layer sizes for the GRU layers, 64 and 128, and with the dropout rate of 0.1. We present the results for the dataset with 56 features and window size of 500, from the same reasons as the LSTM model.

The highest performance during training, Table 22, was from the model with GRU size of 256 and dropout rate of 0.1 with a F1-Measure of 71.560%. During validation, Table 23, and testing, Table 24, the highest value was achieved by the GRU layer of size 64 and dropout rate of 0.2. In theses experiments the problem of overfitting kept up.

Table 22: Experiment 2 - GRU - Training results

LS	D	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
64	0.1	69.941	76.819	64.295	0.845	0.976	1.612	1.810	0.011
64	0.2	69.854	77.018	64.140	0.845	1.482	1.651	2.032	0.017
128	0.1	71.200	77.959	66.487	0.863	0.840	1.557	1.204	**0.010**
128	0.2	71.122	77.654	65.933	0.858	0.729	**1.553**	1.021	0.012
256	0.1	**71.560**	**79.204**	**65.790**	**0.871**	**0.709**	1.934	**0.805**	**0.010**

SD - Standard Deviation; LS - GRU Layer Size; D - Dropout Rate

Table 23: Experiment 2 - GRU - Validation results

LS	D	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
64	0.1	49.643	**54.049**	46.265	**0.607**	6.366	5.652	7.322	0.053
64	0.2	**50.069**	53.081	**47.839**	0.601	4.302	5.110	4.212	0.037
128	0.1	48.581	52.550	45.518	0.551	4.310	**2.966**	5.872	0.066
128	0.2	48.985	52.588	46.448	0.561	**3.109**	3.837	3.913	0.064
256	0.1	47.435	51.714	44.154	0.491	3.833	5.116	**3.339**	**0.036**

SD - Standard Deviation; LS - GRU Layer Size; D - Dropout Rate

Table 24: Experiment 2 - GRU - Testing results

LS	D	Mean				SD			
		F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
64	0.1	39.128	**42.682**	35.960	**0.449**	**1.641**	2.425	2.567	**0.071**
64	0.2	**40.118**	41.968	**38.381**	0.441	2.797	**2.360**	3.638	0.091
128	0.1	39.152	42.062	36.484	0.376	2.383	2.434	3.133	0.082
128	0.2	39.120	41.465	36.793	0.363	2.473	2.395	3.164	0.083
256	0.1	37.093	40.409	34.387	0.326	1.936	2.473	**2.306**	0.072

SD - Standard Deviation; LS - GRU Layer Size; D - Dropout Rate

Now that we analysed the behaviour of the models with different hyperparameters, we will analyse the changes in performance due to a different learning rate. Until now the models were tested with a default learning rate of 0.001. However, LSTM and GRU models showed overfitting, so we decide to test the learning rate values of 0.0001 and 0.00001. We will only present the results for the learning rate of 0.0001 since that for the 0.00001 the performance of all models was not expected.

To reduce the number of performed tests, we chose the better hyperparameters found above. Therefore, we tested:

- CNN model with dense layer size output of 96 and dropout rate of 0.1, we also reduced the number of epochs for 50 during training.
- LSTM model with LSTM layer size of 50 and dropout rate of 0.1.
- GRU model with GRU layer size of 256 and 0.2.

Once again, the dataset tested was the one with 56 features and window size of 500 for LSTM and GRU, and for CNN with window size of 15.

For the CNN model, Table 25, the change of the learning rate shows improvement. During training, validation and testing the F1-Measure values increased to 53.685%, 43.072% and 41.015%, respectively. In Figure 14 the loss function shows a slight overfitting (for more information please check Appendix B - Figure B.1), however, the validation and test values for this model are higher than in the previous ones.

Table 25: Learning Rate Experiment - CNN

	Mean				SD			
	F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
Train	53.685	65.394	45.544	0.673	0.555	0.793	0.832	0.007
Validation	43.072	68.462	31.701	0.629	1.383	6.369	2.511	0.015
Test	41.015	67.997	29.783	0.612	3.043	5.558	4.058	0.005

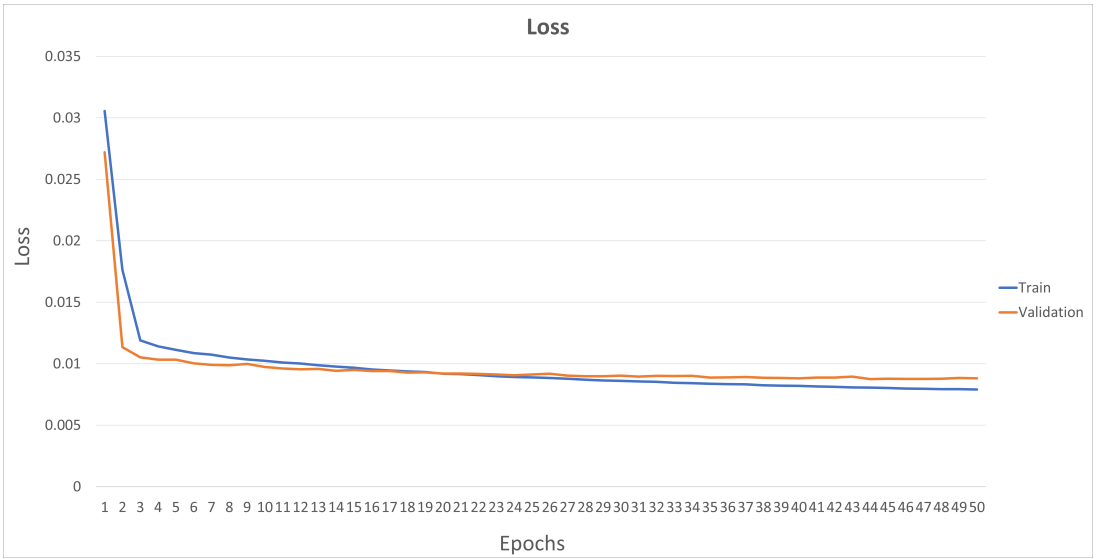


Figure 14: Loss of CNN model

For the LSTM model, Table 26, the overall values for training and validation are lower, with F1-Measure of 64.581% and 53.568%, respectively. However, the test value for F1-Measure increased for 43.120%. Even though, this model still has the problem of overfitting the curve is more stable than in the previous model (Figure 15) (for more information please check Appendix B - Figure B.2).

Table 26: Learning Rate Experiment - LSTM

	Mean				SD			
	F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
Train	64.581	70.995	59.692	0.772	2.610	2.047	2.940	0.037
Validation	53.568	58.351	49.425	0.648	4.373	4.038	4.654	0.065
Test	43.120	46.668	39.537	0.510	2.219	3.242	1.649	0.027

Figure 15: Loss of LSTM model

The GRU model, Table 27, presentes a similar analysis as the previous one (LSTM model), where the overall values for training are lower, with F1-Measure of 65.162%. However, the validation and test values for F1-Measure increased for 52.926% and 42.898%, respectively. The Figure 16 shows the loss function curve, where the model is still overfitting, however, with a more stable curve.

Table 27: Learning Rate Experiment - GRU

	Mean				SD			
	F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
Train	65.162	73.259	59.122	0.787	1.371	1.804	1.392	0.022
Validation	52.926	58.001	49.428	0.660	5.281	5.470	5.681	0.062
Test	42.898	47.132	39.661	0.546	1.754	3.532	3.079	0.017

Figure 16: Loss of GRU model

Since the change in learning rate decrease the gap between the loss functions and stabilise them, we will test the validation dataset with these hyperparameters. However, for the LSTM and GRU models we will also test with the dataset with 66 features and a window size of 500, since we want to analyse the behaviour of these proteins in both representations.

The CNN model with the validation dataset, Table 28, presents values similar to the ones with our dataset. The performance for F1-Measure for training, validation and test is 53.812%, 53.045% and 48.896%, respectively.

Table 28: CNN - Validation Dataset

	Mean				SD			
	F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
Train	53.812	63.156	46.901	0.567	2.266	1.522	2.626	0.028
Validation	53.045	56.936	48.655	0.497	5.229	2.983	6.707	0.036
Test	48.896	56.420	43.881	0.489	3.321	3.245	6.452	0.037

The LSTM model, Table 29, with the dataset with 56 features, during training and validation presented values very similar to the ones with our dataset, with F1-Measure of 63.791% and 52.545%, respectively. However, with the test set the results where not good with F1-Measure values of 11.061%. The same model applied to the dataset with 66 features had very low F1-Measure values (training-2.946%, validation-0.752% and testing-0.715%).

Table 29: LSTM - Validation Dataset

	Mean				SD			
	F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
56 - features								
Train	63.791	70.297	58.681	0.752	1.571	1.707	1.563	0.024
Validation	52.545	58.117	48.264	0.623	3.015	4.231	3.506	0.056
Test	11.061	36.316	6.772	0.102	2.124	4.487	1.509	0.014
66 - features								
Train	2.946	18.771	1.580	0.200	2.095	13.521	1.131	0.048
Validation	0.752	18.254	0.393	0.131	0.551	13.792	0.287	0.042
Test	0.715	11.607	0.376	0.091	0.787	12.264	0.384	0.004

The GRU model, Table 30, for the dataset of 56 features, presents values for the F1-Measure of 68.002%, 50.473% and 8.613%, respectively, for training, validation and testing. On the other hand, the dataset of 66 features, presentes the best values of all experiments with F1-Measure of 83.270%, 60.816% and 49.931%, respectively, for training, validation and testing.

Table 30: GRU - Validation Dataset

	Mean				SD			
	F1_M$_{\%}$	Precision$_{\%}$	Recall$_{\%}$	PR AUC	F1_M	Precision	Recall	PR AUC
				56 - features				
Train	68.002	73.757	63.344	0.809	0.313	0.446	0.268	0.010
Validation	50.473	58.500	44.671	0.633	2.999	4.037	2.376	0.057
Test	8.613	39.846	4.966	0.183	0.588	2.359	0.488	0.005
				66 - features				
Train	83.270	87.298	79.762	0.930	2.752	1.505	3.834	0.028
Validation	60.816	70.153	55.026	0.656	10.169	3.829	13.546	0.120
Test	49.931	54.512	44.545	0.469	3.718	2.740	5.922	0.016

4.3 Discussion

In experiment 1, we started by evaluating the CNN model that presents low values compared to those obtained by Haberal [24]. These values, were somehow expected since we are using a model with strong abilities for image analysis in sequence data. Another point that justifies these values is the low number of positive examples that the model has available to learn. For our dataset, the classifier presents low skill since its PR AUC values range between 0.46-0.53. However, the loss learning curve shows a good fit between training and validation data. LSTM and GRU models during training present values close to the ones obtained by Haberal [24], with PR AUC values higher than 0.84. Both models suffer from overfitting; however, it is not a surprise because this problem is the main challenge when working with datasets so unbalanced as ours. The classical solution for unbalanced data is oversampling or undersampling. Due to the specificity of the problem, we can not just eliminate proteins or create fake positive examples. Therefore, in the next experiment (2), we analysed the behaviour of these models with other hyperparameters and regulated the learning rate.

As we referred earlier, we had no expectation of having positive changes in the performance of the models by testing them with the dataset with 66 features or 56 features. Therefore, meeting our expectations, the difference in the performance between these two sets is 1 unit, sometimes higher for the dataset with 66 and other times for the one with 56. However, the standard deviation of the metrics was always lower for the dataset with 56 features, showing more stability of the obtained values when experimenting. The size of the window, 2143 or 500, has low influence on the performance in the experiment 2, to decrease computational cost, we choose to use the window with 500 residues.

In experiment 2, for the CNN model, the dense layer output size of 96 and dropout rate of 0.1 enhanced the performance values. The application of a new learning rate helped, even more, the algorithm since the loss function reached lower values than before. However, the loss for the validation set has values slightly higher than the training set. Analysing the overall values, we were able to improve the skill of the model for our dataset since the PR AUC ranges between 0.61-0.67. The LSTM algorithm with an LSTM layer size of 50 and a dropout rate of 0.1 reached the best values for the performance. The change in the learning rate decreased the performance, but the gap between the loss function of training and validation was decreased along with the stabilisation of the tendency. Despite the decrease in training performance, we find this model the best within our experiments since it presents more stability in the loss curve.The GRU model has the best performance, with a GRU layer size of 256 and a dropout rate of 0.2. The changes in the learning rate affected the model only during training because the values for validation and test increased. The conclusions of this model are the same as the ones for the LSTM model, where the change in the learning rate decreased the gap between the loss functions and stabilised them.

To validate our models, we tested them with a validation dataset. The CNN model

presents values similar to the ones with our dataset. With the validation dataset with 56 features, the LSTM and GRU models present similar values to the ones with our dataset. However, they also present overfitting, with F1-Measure for test bellow 11%. For the validation dataset with 66 features, the LSTM model presents unexpected values, and GRU presents the best values of all models.

In summary, the CNN model presents the lower results, however, the more stable ones since the loss function presents a good fit. LSTM and GRU have equivalent performance. Due to the conditions of overfitting we can argue about the heterogeneous set of features that we selected since our values in test are so low that are not comparable with the results of the state of the art [24]. Moreover, our training results opened the doors for improvement when amino acid sequences are used instead of only specific binding residues, which is an advantage over the state of the art implemented models and for future similar tasks.

Chapter 5

Conclusion & Future Perspectives

The present study concerned the development of three deep learning algorithms, Convolutional Neural Networks (CNN), Long-Short Term Memory Neural Networks (LSTM) and Gated Recurrent Units (GRU), for the prediction of zinc-binding residues in regulatory proteins, which was our main objective. To achieve this, several steps were taken, namely:

- The construction of a specific dataset containing transcription factors with zinc-binding sites, as well as the extraction of features with diverse methods for a heterogeneous set containing sequence and structure based properties from the proteins.
- The analysis and implementation of three deep learning models, CNN, LSTM and GRU, along with their training and hyperparameter optimisation.
- The application of the final models to a benchmark dataset to validate our results.

Moreover, regardless of the computational complexity of the training phase, the proposed methodology is relatively straightforward with quick processes for the feature extraction, basic postprocessing steps and, finally, the implementation of the models for classification. To comprehend the relation between the features and the target we assessed the linear correlation coefficient concluding that sequence based features related

do the cysteine and histidine residues, as well as the secondary structure where the residue is localised are the features that have the higher linear correlation with the target.

The CNN model presents the lowest results; however, they are corroborated by the validation dataset. Thus, it can be said that the problem may be the application of a sequence data problem to a network specialised in image classification. The LSTM and GRU models are both overfitting, therefore is difficult to make an objective analysis. These models in the training stage present relatively good results with room for improvement. However, in the validation and testing phase, they present low skills. This behaviour can have two sources: the first source is the dataset imbalance, second is the representativeness, as much as we have been careful to maintain the proportions during all the processes that split the dataset, some coordination patterns have a residual presence in the dataset, therefore the training set may not contain them and when validating or testing the model the performance decreases because of the lack of knowledge of these patterns. The validation dataset is overfitting, either.

In conclusion, the analysis of the training results shows an opening for improvement when amino acid sequences are used instead of just specific binding residues. This is an advantage over models implemented with state of the art technology, since when we use complete amino acid sequences we are dealing with more residues than just the four/five residues present in the coordination sphere. Which means that we can use much more information to accurately predict the presence of zinc ion in the TF, and also inspect other important residues for the coordination sphere. Furthermore, considering that we only used regulatory proteins with zinc-binding sites, the number of patterns available for the dataset was highly limited, as there is a small number of annotated proteins to work with complex machine learning algorithms. To truly evaluate the actual application of methods such as these, more data is needed to build more comprehensive datasets.

Concerning future perspectives, once this approach is relatively simple, there is space

for further improvements. In particular, it is important to solve the overfitting problem, however, options like oversampling or undersampling are not suitable since we cannot construct fake positive samples or eliminate the negative examples. Although, there are other strategies that can be implemented, such as hybrid approaches where data- and algorithm-level approaches can be combined. The use of methods for feature selection, such as the embedded filter, can also make big improvements since we can analyse the impact of each feature in the prediction allowing us to discard the ones with no meaning for the problem, as well as reducing the computational cost by only working with meaningful features. Another interest approach for metal-binding residue prediction are the Natural Language Processing (NLP) algorithms that only use the sequence of amino acids for feature extraction.

References

[1] Bayat A. "Science, medicine, and the future: Bioinformatics". In: *BMJ* 324 (2002). DOI: 10.1136/bmj.324.7344.1018.

[2] Dayhoff MO. and Ledley RS. "Comprotein: a computer program to aid primary protein structure determination". In: *Proceedings of the December* Fall Joint Computer Conference (1962), pp. 4–6. DOI: 10.1145/1461518.1461546.

[3] Margaret O. Dayhoff. *Atlas of protein sequence and structure*. Vol. 1. National Biomedical Research Foundation, Silver Spring [Md.], 1965.

[4] Dayhoff MO., Schwartz RM., and Orcutt BC. *Atlas of protein sequence and structure*. Washington, DC: National Biomedical Research Foundation, 1978. Chap. 22: a model of evolutionary change in proteins.

[5] Jeff Gauthier et al. "A brief history of bioinformatics". In: *Briefings in Bioinformatics* 20.6 (2018), pp. 1981–1996. DOI: 10.1093/bib/bby063.

[6] C. J. Cramer. *Essentials of Computational Chemistry*. John Wiley Sons, 2002.

[7] NF Brás et al. "Computational Biochemistry". In: *Reference Module in Chemistry, Molecular Sciences and Chemical Engineering* (2015). DOI: 10.1016/B978-0-12-409547-2.10833-9.

[8] Branden C and Tooze J. *Introduction to Protein Structure*. New York: Garland Pub, 1999.

[9] Nan Ye et al. “A Comprehensive Review of Computation-Based Metal-Binding Prediction Approaches at the Residue Level”. In: *BioMed Research International* 2022 (2022), p. 19. DOI: 10.1155/2022/8965712.

[10] Keith A. McCall, Chih chin Huang, and Carol A. Fierke *et al.* “Function and Mechanism of Zinc Metalloenzymes”. In: *American Society for Nutritional Sciences* (2000), p. 10. DOI: 10.1093/jn/130.5.1437S.

[11] Claudia Andreini, Ivano Bertini, and Gabriele Cavallaro *et al.* “Minimal Functional Sites Allow a Classification of Zinc Sites in Proteins”. In: *PLoS ONE* 6.10 (2011), p. 13. DOI: 10.1371/journal.pone.0026325.

[12] Meiying Zheng, David R. Cooper, and Nickolas E. Grossoehme *et al.* “Structure of Thermotoga maritima TM0439: implications for the mechanism of bacterial GntR transcription regulators with Zn^{2+}-binding FCD domains”. In: *Biological Crystallography* D.65 (2009), pp. 356–365. DOI: 10.1107/S0907444909004727.

[13] Ian M. Adcock and Gaetano Caramori. “Transcription Factors”. In: *Asthma and COPD* (2009), pp. 373–380. DOI: 10.1016/B978-0-12-374001-4.00031-6.

[14] Peter F. Johnson and Steven L. McKnight. “Eukaryotic Trasncriptional Regulatory Proteins”. In: *Annual Review of Biochemistry* 58 (1989), pp. 799–839. DOI: 10.1146/annurev.bi.58.070189.004055.

[15] Hui LI, Dechang PI, and Chuanmin CHEN. “An Improved Prediction Model for Zinc-Binding Sites in Proteins Based on Bayesian Method”. In: *Technical Gazette* 26.5 (2019), pp. 1422–1426. DOI: 10.17559/TV-20190730093945.

[16] Deepti Jain. “Allosteric Control of transcription in GntR Family of Transcription Regulators: A Structural Overview”. In: *International Union of Biochemistry and Molecular Biology* 67.7 (2015), pp. 556–563. DOI: 10.1002/iub.1401.

[17] Armelle Vigouroux et al. "Characterization of the first tetrameric transcription factor of the GntR superfamily with allosteric regulation from the bacterial pathogen *Agrobacterium fabrum*". In: *Nucleic Acids Research* 49.1 (2021), pp. 529–546. DOI: 10.1093/nar/gkaa1181.

[18] Noah D Taylor et al. "Engineering an allosteric transcription factor to respond to new ligands". In: *Nat Methods* 13.2 (2016), pp. 177–183. DOI: 10.1038/nmeth.3696.

[19] *scikit-learn*. URL: https://scikit-learn.org/stable/. (accessed: 30.01.2021).

[20] *Keras*. URL: https://keras.io/. (accessed: 30.01.2021).

[21] Hu X. et al. "Recognizing metal and acid radical ion-binding sites by integrating *ab initio* modeling with template-based transferals". In: *Bioinformatics* 32.21 (2016), pp. 3260–3269. DOI: 10.1093/bioinformatics/btw396.

[22] Song J. et al. "MetalExplorer, a Bioinformatics Tool for the Improved Prediction of Eight Types of Metal-binding Sites Using a Random Forest Algorithm with Two-step Feature Selection". In: *Current Bioinformatics* 11 (2016). DOI: 10.2174/2468422806666160618091522.

[23] Haberal I. and Ogul H. "DeepMBS: Prediction of protein metal binding-site using deep learning networks". In: *Fourth International Conference on Mathematics and Computers in Sciences and in Industry* (2017). DOI: 10.1109/MCSI.2017.13.

[24] Haberal I. and Ogul H. "Prediction of Protein Metal Binding Sites Using Deep Neural Networks". In: *Molecular Information* 38 (2019). DOI: 10.1002/minf.201800169.

[25] David L. Nelson and Michael M. Cox. *Lehninger - Principles of Biochemistry*. 5th ed. W. H. Freeman and Company, 2008.

[26] Ritika Gupta et al. "In Silico Structure Modeling and Characterization of Hypothetical Protein $YP_0$04590319.1*PresentinEnterobacter aerogens*". In: *Proteomics Bioinformatics* 10.6 (2017), pp. 152–170. DOI: 10.4172/jpb.1000436.

[27] Rehman I., Farooq M., and Botelho S. *Biochemistry, Secondary Protein Structure*. StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing, 2022. URL: https://www.ncbi.nlm.nih.gov/books/NBK470235/.

[28] Rehman I., Kerndt CC., and Botelho S. *Biochemistry, Tertiary Protein Structure*. StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing, 2022. URL: https://www.ncbi.nlm.nih.gov/books/NBK470269/.

[29] Wolfgang Maret. "Zinc coordination environments in proteins determine zinc functions". In: *Trace Elements in Medicine and Biology* 19 (2005), pp. 7–12. DOI: 10.1016/j.jtemb.2005.02.003.

[30] Breiman L. "Random Forests". In: *Machine Learning* 45 (2001), pp. 5–32. DOI: 10.1023/a:1010933404324.

[31] Onesmus Mbaabu. *Introduction to Random Forest in Machine Learning*. URL: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/. (accessed: 06.08.2022).

[32] Song J. et al. *MetalExplorer*. URL: http://metalexplorer.erc.monash.edu.au/.

[33] Suni Ray. *Understanding Support Vector Machine(SVM) algorithm from examples (along with code)*. URL: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/. (accessed: 06.08.2022).

[34] Rohith Gandhi. *Support Vector Machine — Introduction to Machine Learning Algorithms*. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. (accessed: 06.08.2022).

[35] Jones DT., Taylor WR., and Thornton JM. "The rapid generation of mutation data matrices from protein sequences". In: *Comput Appl Biosci* 8.3 (1992). DOI: 10.1093/bioinformatics/8.3.275.

[36] Depth Tutorials and Information. *Point Accepted Mutation (Molecular Biology)*. URL: http://what-when-how.com/molecular-biology/point-accepted-mutation-molecular-biology/. (accessed: 30.08.2022).

[37] Aditya Sharma. *Convolutional Neural Networks in Python with Keras*. URL: https://www.datacamp.com/tutorial/convolutional-neural-networks-python. (accessed: 20.06.2022).

[38] Jason Brownlee. *When to Use MLP, CNN, and RNN Neural Networks*. URL: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/. (accessed: 20.06.2022).

[39] Ireland SM. and Martin ACR. "ZincBindDB". Version 1.2.0. In: (2019). DOI: 10.1093/database/baz006. URL: https://github.com/samirelanduk/ZincBindDB.

[40] L. Rishishwar et al. "ProCoS: Protein composition server". In: *Bioinformation* 5.5 (2010). DOI: 10.6026/97320630005227.

[41] IBM Cloud Education. *Recurrent Neural Networks*. URL: https://www.ibm.com/cloud/learn/recurrent-neural-networks. (accessed: 20.06.2022).

[42] Michael Pi. *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. URL: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. (accessed: 20.06.2022).

[43] Siddharth Yadav. *Intro to Recurrent Neural Networks LSTM — GRU*. URL: https://www.kaggle.com/code/thebrownviking20/intro-to-recurrent-neural-networks-lstm-gru/notebook. (accessed: 21.06.2022).

[44] Yujian Tang. *Build a GRU RNN in Keras*. URL: https://pythonalgos.com/build-a-gru-rnn-in-keras/. (accessed: 21.06.2022).

[45] Zhang Lab. *What is FASTA format?* URL: https://zhanggroup.org/FASTA/. (accessed: 30.06.2022).

[46] PDB-101. *Introduction to PDB Data*. URL: https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/introduction. (accessed: 30.06.2022).

[47] Bernardete Ribeiro. *Pattern Recognition - Introduction, Fundamentals & Concepts*. 2021.

[48] Kulakovskiy IV. et al. "HOCOMOCO: towards a complete collection of transcription factor binding models for human and mouse via large-scale ChIP-Seq analysis". In: *Nucleic Acids Research* 46 (2018). DOI: 10.1093/nar/gkx1106.

[49] Kulakovskiy IV. et al. *HOCOMOCO*. URL: https://hocomoco11.autosome.org/. (accessed: 26.01.2022).

[50] Tong Z. et al. "TransmiR v2.0: an updated transcription factor-microRNA regulation database". In: *Nucleic Acids Research* 47 (2019). DOI: 10.1093/nar/gky1023.

[51] Tong Z. et al. *TransmiR v2.0 database*. URL: http://www.cuilab.cn/transmir. (accessed: 26.01.2022).

[52] Fornes O. et al. "JASPAR 2020: update of the open-access database of transcription factor binding profiles". In: *Nucleic Acids Research* 48 (2020). DOI: 10.1093/nar/gkz1001.

[53] Fornes O. et al. *Jaspar 2022*. URL: https://jaspar.genereg.net/. (accessed: 26.01.2022).

[54] The UniProt Consortium. "UniProt: the universal protein knowledgebase in 2021". In: *Nucleic Acids Research* 49 (2021). DOI: 10.1093/nar/gkaa1100.

[55] UniProt. *ID Mapping*. URL: https://www.uniprot.org/help/id_mapping. (accessed: 30.01.2022).

[56] UniProt. *Programmatic access - Retrieving individual entries*. URL: https://www.uniprot.org/help/api_retrieve_entries. (accessed: 30.01.2022).

[57] H.M. Berman et al. *RCSB PDB - (2000) The Protein Data Bank Nucleic Acids Research, 28: 235-242*. URL: rcsb.org. (accessed: 30.01.2022).

[58] Cokelaer et al. "BioServices: a common Python package to access biological Web Services programmatically". In: *Bioinformatics* 29.24 (2013). DOI: 10.1093/bioinformatics/btt547. URL: https://bioservices.readthedocs.io/en/main/index.html.

[59] MathWorks. *corr*. URL: https://www.mathworks.com/help/stats/corr.html?s_tid=srchtitle_corr_1. (accessed: 23.03.2022).

[60] Wikipedia. *Pearson correlation coefficient*. URL: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient. (accessed: 23.03.2022).

[61] Sam Ireland. *ZincBind*. URL: https://api.zincbind.net/. (accessed: 30.05.2022).

[62] Passerini A. et al. "Identifying Cysteines and Histidines in Transition-Metal-Binding Sites Using Support Vector Machines and Neural Networks". In: *PROTEINS: Structure, Function, and Bioinformatics* 65 (2006). DOI: 10.1002/prot.21135.

[63] Bhagwat M. and Aravind L. *Comparative Genomics: Volumes 1 and 2*. Totowa (NJ): Humana Press, 2007. Chap. 10 PSI-BLAST Tutorial. URL: https://www.ncbi.nlm.nih.gov/books/NBK2590/.

[64] Altschul SF, Madden TL andSchäffer AA, and et al. "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs". In: *Nucleic Acids Research* 25 (1997). DOI: 10.1093/nar/25.17.3389.

[65] Aravind L. and Koonin EV. "Gleaning non-trivial structural, functional and evolutionary information about proteins by iterative database searches". In: *J. Mol. Biol* 287.5 (1999). DOI: 10.1006/jmbi.1999.2653.

[66] Peter JA Cock et al. "Biopython: freely available Python tools for computational molecular biology and bioinformatics". In: *Bioinformatics* 25.11 (2009), pp. 1422–1423.

[67] Sayers EW et al. "Database resources of the national center for biotechnology information". In: *Nucleic Acids Research* 50 (2022). DOI: 10.1093/nar/gkab1112.

[68] Touw WG. et al. "A series of PDB related databases for everyday needs". In: *Nucleic Acids Research* 435 (2015). DOI: 10.1093/nar/gku1028.

[69] Kabsch W and Sander C. "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features". In: *Biopolymers* 22 (1983). DOI: 10.1002/bip.360221211.

[70] Oezguen N. et al. *FANTOM 4.2*. URL: https://bose.utmb.edu/fantom/. (accessed: 30.05.2022).

[71] Fraczkiewicz R. and Braun W. "Exact and Efficient Analytical Calculation of the Accessible Surface Areas and Their Gradients for Macromolecules". In: *Journal of Computational Chemistry* 19.3 (1998). DOI: 10.1002/(SICI)1096-987X(199802)19:3<319::AID-JCC6>3.0.CO;2-W.

[72] Negi S. et al. *GETAREA 1.0 beta*. URL: http://curie.utmb.edu/area_man.html. (accessed: 30.05.2022).

[73] Bernardete Ribeiro. *Pattern Recognition - Chapter 6 – Feature Selection*. 2021.

[74] Aaron Lee. *Boruta Feature Selection (an Example in Python)*. URL: https://towardsdatascience.com/simple-example-using-boruta-feature-selection-in-python-8b96925d5d7a. (accessed: 30.05.2022).

[75] Debarati Dutta. *How to perform feature selection (i.e. pick important variables) using Boruta Package in R ?* URL: https://www.analyticsvidhya.com/blog/2016/03/select-important-variables-boruta-package/. (accessed: 30.05.2022).

[76] *boruta$_p$y*. URL: https://github.com/scikit-learn-contrib/boruta_py. (accessed: 30.05.2022).

[77] António Dourado. *Computação Neuronal e Sistemas Difusos - Chapter 5 – Dynamic Networks and Deep Learning*. 2020.

[78] Bernardete Ribeiro. *Pattern Recognition - Chapter 5 – ROC Curves*. 2021.

[79] Jason Brownlee. *Tour of Evaluation Metrics for Imbalanced Classification*. URL: https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/. (accessed: 28.11.2021).

Appendices

Appendix A

Experiment 1

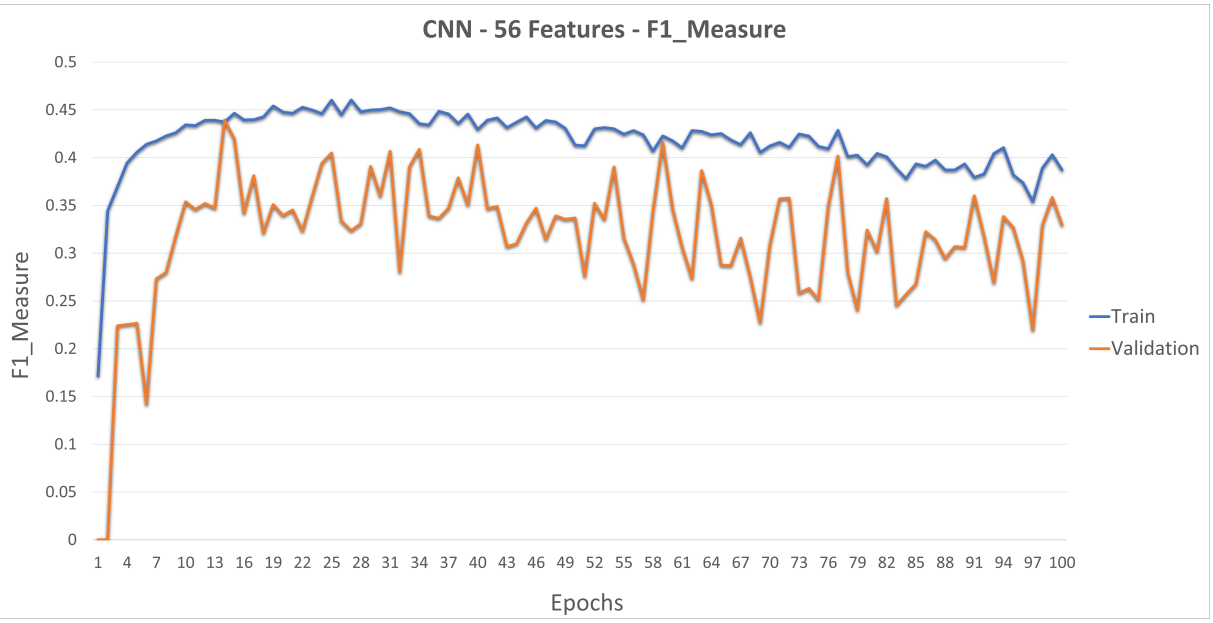


Figure A.1 - F1-Measure evolution of CNN model, with 56 features and window size of 15 residues

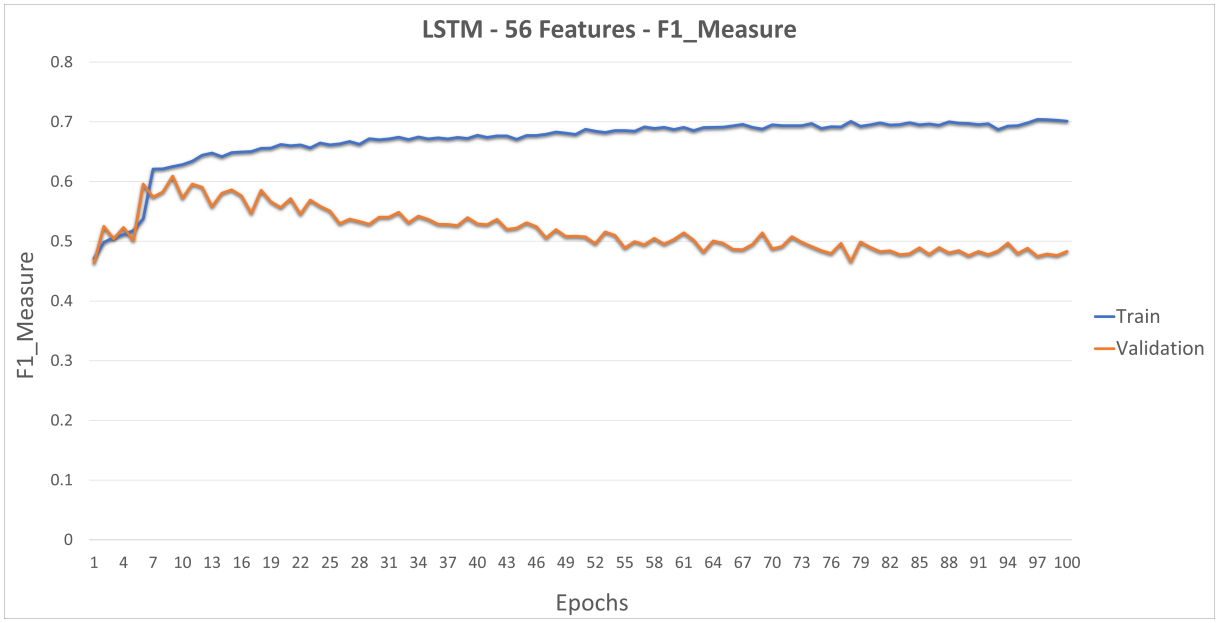


Figure A.2 - F1-Measure evolution of LSTM model, with 56 features and window size of 500 residues

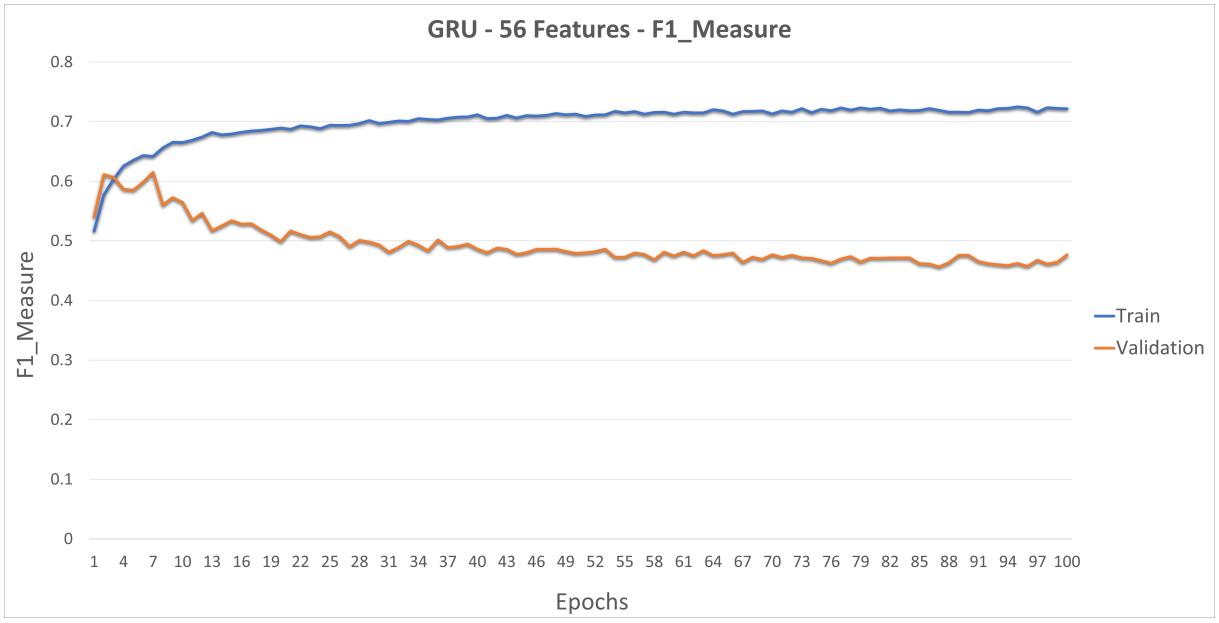


Figure A.3 - F1-Measure evolution of GRU model, with 56 features and window size of 500 residues

Appendix B

Experiment 2

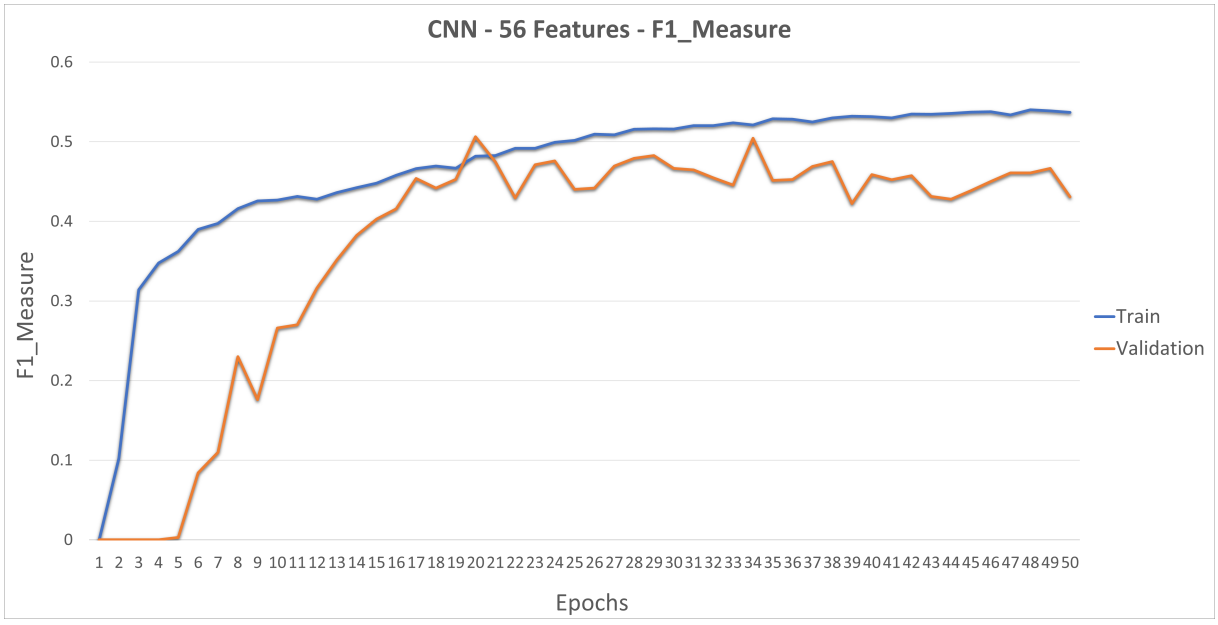


Figure B.1 - F1-Measure evolution of CNN model, with 56 features and window size of 15 residues

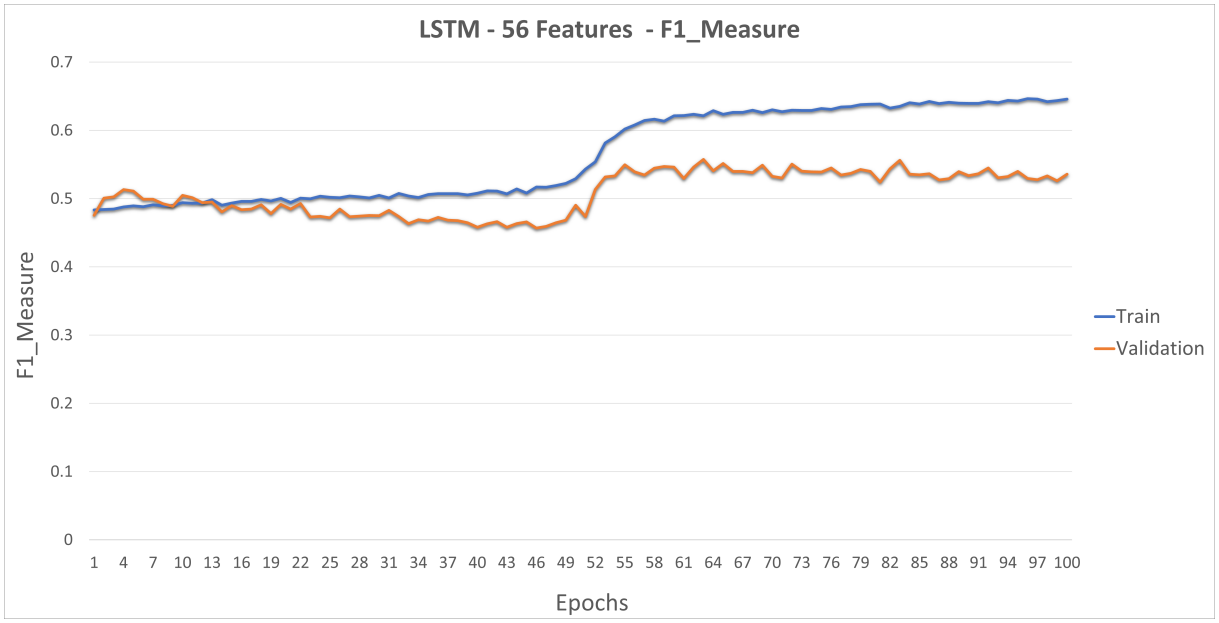


Figure B.2 - F1-Measure evolution of LSTM model, with 56 features and window size of 500 residues

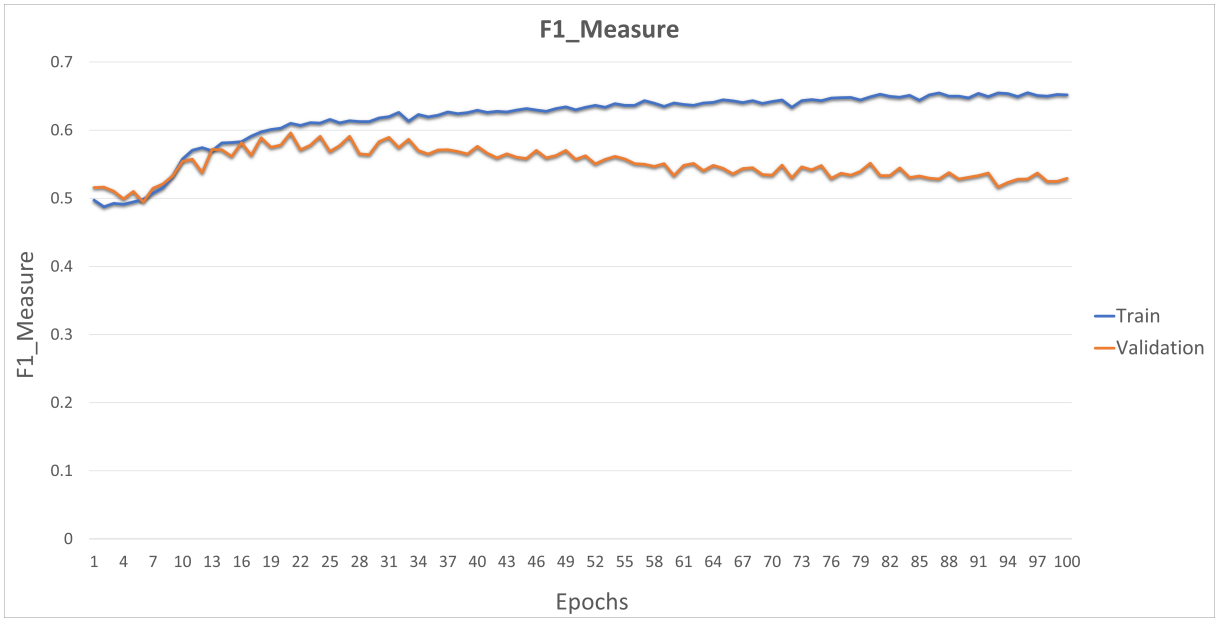


Figure B.3 - F1-Measure evolution of GRU model, with 56 features and window size of 500 residues