
**Cláudio Pascoal**

Bachelor in Computer Science Engineering

# 3D Convolutional Neural Networks for Identifying Protein Interfaces

Dissertação para obtenção do Grau de Mestre em

**Engenharia Informática**

Orientador: Ludwig Krippahl, Assistant Professor,
NOVA University of Lisbon

Júri

Presidente: Rui Nóbrega, Assistant Professor, FCT UNL - Computer Science Dep.
Arguente: Rui Rodrigues, Assistant Professor, FCT UNL - Math Dep.


FACULDADE DE
CIÊNCIAS E TECNOLOGIA
**UNIVERSIDADE NOVA** DE LISBOA

**February, 2021**

**3D Convolutional Neural Networks for Identifying Protein Interfaces**



Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

# Acknowledgements

Firstly, I would like to thank my adviser, Professor Ludwig Krippahl, for the guidance and advises during the preparation and elaboration of this project. Furthermore, I want to thank all my colleagues, whom I spent so many years learning with and who kept me motivated. Importantly, I want to thank my family, who supported me and cheered for me all the time! Finally, I want to thank all my friends that, in one way or another, encouraged and helped me.

I also want to thank my cat, Genji, for sitting and sleeping next to me while I was developing this project.

# Abstract

Protein interaction is a fundamental part of nearly all biochemical processes and proteins evolved specific surface regions for molecular recognition and interaction. These regions are different from the remaining surface, with different amino acid compositions, geometry and chemical properties. Detecting protein interfaces can lead to a better understanding of protein interactions granting advantages to fields such as drug design and metabolic engineering.

Most of the existing interface predictors use structured data, clearly defined data types usually obtained from data sets. However, proteins are very complex molecules and there is not a single property capable of distinguishing the interface from the rest of the protein surface to all types of proteins. Indeed, deep learning arises as an adequate approach able to capture feature from unstructured data as images, texts, sensor data and volumes. In here, the aim was to identify interface regions in known protein spatial structures together with their biochemical properties by exploring new applications of 3D convolutional neural networks.

For this, some state-of-the-art convolutional neural networks architectures were explored in order to find an architecture that suits this problem, and even more, have good performance. Other state-of-the-art machine learning predictors are also considered to identify the best biochemical properties to be added as new channels.

Afterward, the interface predictions will be compared with the ground-truth, obtained by calculating the distances of atoms between the different chains of the protein complexes.

**Keywords:** Machine learning, Deep learning, Neural networks, Bioinformatics, Protein, Interface prediction, Convolutional neural networks

# Resumo

A interação entre proteínas é fundamental em todos os processos biológicos e bioquímicos. As proteínas são compostas por regiões específicas que permitem o reconhecimento molecular e, consequentemente, interações com outras moléculas. Normalmente, estas regiões são estruturalmente diferentes da restante molécula sendo caracterizadas e compostas por aminoácidos diferentes, propriedades químicas e geometria diversa. A detecção das interfaces das proteínas pode ser uma mais valia no contexto de perceber a interação entre as mesmas e consecutivamente, ser vantajoso para o design de novos fármacos (ou drug design) e engenharia metabólica.

As previsões de interfaces usam maioritariamente dados estruturados, ou seja, dados bem definidos normalmente obtidos em bancos de dados. No entanto, as proteínas são moléculas complexas o que impossibilita a distinção da sua interface, uma vez que não existe uma propriedade única e específica para todas. Deste modo, o deep learning é uma ferramenta fundamental porque usa características de dados não estruturados, como por exemplo a informação espacial da proteína, imagens, textos, dados de sensores ou volumes.

O objetivo principal deste projeto é identificar regiões de interfaces através de estruturas tri-dimensionais de proteínas conhecidas juntamente com as respetivas distribuição espacial das suas propriedades, usando redes neuronais de convolução. Neste trabalho foram estudados algoritmos de deep learning para encontrar a rede neuronal mais adequada ao problema que pretendemos resolver com o melhor desempenho. Outros algoritmos de previsão foram considerados para identificar quais as melhores propriedades bioquímicas a serem usadas como novos canais de input.

Seguidamente, as previsões do modelo foram comparadas com as interfaces reais, que foram obtidas pelo cálculo das distâncias dos átomos entre cadeias diferentes do mesmo complexo.

**Palavras-chave:** Aprendizagem automática, Aprendizagem profunda, Redes neuronais, Bioinformática, Proteína, Previsão de interfaces, Redes neuronais de convolução

# Contents

# List of Figures

Chapter 1

# Introduction

The main goal of this chapter is to highlight the motivation and main aims of this work. Moreover, a brief biochemical background will be provided for further understanding of how we can use proteins structures and their residues properties to predict their interfaces. Finally, it will also be explained machine learning in the context of biochemical issues.

## 1.1 Motivation

Proteins are essential molecules to all living organisms. Every cell produces and uses significant amounts of proteins to perform their own and specific biological function. Importantly, they can act as enzymes, antibodies, transporters and hormones.[24] Commonly, these molecules are described as linear compositions - chains - of amino acids. There are 20 amino acids, and each one has different characteristics such as polarity, hydrophobicity and electric charge. Moreover, protein chains can assume lengths from twenty to thousands of amino acids.

One of the essential protein characteristics is the ability to detect and bind to other molecules through specific areas, which are known as interfaces. These interfaces, commonly located in the protein's surface, have specific biochemical characteristics and allow the generation of complexes (multi-chained proteins) with their own function. Interestingly, according to Protein Data Bank (PDB), there are over 3500 proteins/structures with unknown function. Identifying chains interfaces in the laboratory is very time demanding, and most of the existing machine learning algorithms used to study interfaces use structured data. The addition of Deep Learning algorithms such as Convolutional Neural Networks (CNNs) to this context is a significant improvement since deep learning applications seem to achieve better performances with non-structured data than traditional

machine learning algorithms.

The biochemical properties of the amino acids, their ability to bind with other molecules and their 3D structure are essential features to understand proteins' functions. Indeed, studying and predicting proteins interfaces with CNNs open new doors to discover new proteins functions, and consequently, new possibilities for drug design and to new therapeutic solutions.

## 1.2 Objectives

Many of the existing protein interfaces predictors use structured data (data that can be represented in a table) with the traditional machine learning algorithms such as random forests or Support Vector Machines. Structure data means that there is no spatial relationship between their samples attributes, in this case, amino acids characteristics. By using 3D Convolutional Neural Networks(CNNs), it is possible to extract features from the characteristics of the amino acids based on where they are located, and based on which characteristics the amino acids located around them have. The amino acids characteristics can be either structural or physicochemical. Moreover, the goal was to build a 3D CNN that can be trained using protein chains' amino acids and their corresponding characteristics represented in a 3D environment; and then predict which of those amino acids belong to those protein chains' interfaces. Indeed, in this project, there were executed three steps until the discussion of the results:

1. Dataset Creation (Chapter 3) - in this step, the aim was to build 3D representations of protein chains and their amino acids characteristics. It involved filtering and downloading protein structural information (3D coordinates) from the PDB and, afterwards, more amino acid properties were calculated using state-of-the-art algorithms. The true protein interfaces were then calculated, and then, the relevant amino acids' attributes for interface classification were mapped, using the 3D coordinates, into 3D arrays (tensors). At the end of this step, the dataset was divided into three subsets: training set, validation set and test set. The training set was used to train the 3D CNN; the validation set was used to validate the training phase, and finally, the test set was used to evaluate the CNN predictions.

2. 3D CNN Implementation (Chapter 4) - after studying several state-of-the-art 3D CNNs implementations and architectures (Section 2.2.3), it was decided to use the U-Net architecture since it was positively suited in other Semantic Segmentation problems. For example, a Semantic Segmentation problem could consist in identifying which pixels within an image represent a cat or a dog. In this case, the problem consists in the identification of which amino acids, within a protein, belong to the protein's interface. The details about the architecture and its parameters are discussed in the corresponding section.

3. 3D CNN Training and Evaluation (Chapter 5 ) - in this step, the final 3D CNN was trained with different combinations of amino acids characteristics until the one with the best score was obtained. Moreover, the predictions on the test set, produced by the model trained with the best combination (the one which obtained the best validation score during the training phase), were evaluated based on the Confusion Matrix scores. Additionally, the performance of the model was represented by the AUC-ROC curve. Furthermore, the best model was tested with an independent test set from the Protein Docking Benchmark to compare its performance with other similar predictors.

## 1.3 Proteins

Proteins are complex macromolecules that are present in all types of cells. Curiously, a single cell may have a variety of proteins that differ in size (from small to large peptides), in complexity, function, and behavior. Additionally, they bind to other molecules to execute different biological functions that could not be performed otherwise. An excellent functional example is hemoglobin since it is responsible for the transportation of oxygen in red blood cells when bound to iron molecules. Furthermore, proteins can be represented in four levels: primary structure, secondary structure, tertiary structure and, for proteins with more than one chain (complexes), the quarternary structure (Section 1.3.2). Therefore, the building blocks and the four main structure representations of proteins will be described in the following sections [24].

### 1.3.1 Amino Acids

There are 20 types of amino acids, with different and specific characteristics. When amino acids bind together to create a polypeptide chain or protein complex, water molecules are released, leaving their residues in the chain. Therefore, for now on, the name 'residues' refer to the amino acids in protein chains. Indeed, all of the amino acids are alpha-acids being composed by a carboxyl group (COOH), an amino group (NH2) and the R-group, or side chain (Figure 1.1). Moreover, they are all linked to the same carbon atom called the alpha-carbon.

The distinction between the 20 types of amino acids is their R-group, or side chain because it has specific characteristics concerning size, shape, and electric charge. Importantly, these characteristics have an impact in the amino acid's solubility in water which has great importance in complex creation and thus in interface prediction [24].

Physicochemical and structural properties of amino acids help in defining if they belong to an interface. The most noticeable ones are solvent accessibility, hydrophobicity and charge.

**Solvent Accessibility** - Usually, protein interface predictions use this property, and it is one of the most discriminatory. Studies have proved that high solvent accessibility of a

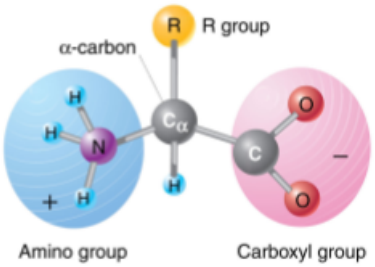


Figure 1.1: The structure of amino acids: a carboxyl group ($COO^-$), an amino group ($NH_3^+$) and the side chain (R group) bounded to the alpha-carbon ($\alpha$-carbon). Adapted from: iGenetics, 3rd edition, (2012).

residue indicates its participation in an interface [7]. Areas buried within the molecule with no exposure without access on the surface are less likely to participate in protein-ligand interaction since binding sites take place in the surface of the protein. Interface's residues are more solvent-accessible [33] when they are in an uncomplexed form. However, a theory called O-Ring shows that a ring of other residues surrounds the residues that make contact. This ring occludes water molecules from the center of the interface where contact residues are present [25]. For this reason, knowing the solvent accessibility of a residue and its neighbors may contribute to determining whether the first belongs to an interface or not.

**Hydrophobicity** - Hydrophobicity is a measure that represents the tendency of amino acids to prefer non-soluble environments over soluble environments. Hydrophobic residues tend to interact with each other, and they prefer to interact with each other over interacting with the soluble environment. On the contrary, it is known that pairing hydrophobic and hydrophilic residues result in low contact values. Therefore, hydrophobicity is very relevant for detecting protein interfaces. In addition to the O-Ring theory, that was mentioned above, the hydrophobicity of the residues located in the O-Ring may be relevant since these residues occlude water from the contact residues [25]. Therefore, knowing a residue and its neighbors hydrophobicity may help to predict its presence in contact sites. In below, a list of the hydrophobic amino acids is presented [9]:

1. Very Hydrophobic - Cysteine, Isoleucine, Leucine, Methionine, Phenylalanine, Tryptophan and Valine.
2. Less hydrophobic - Alanine, Glycine, Histidine, Proline, Serine, Threonine and Tyrosine.
3. Part hydrophobic - Arginine and Lysine.

**Charge** - There are five charged amino acids when incorporated in a protein chain. Aspartate, Glutamate and Histidine, when they are in more acidic environments (lower pH),

they accept a proton, which is charged positively. Therefore, they are basic amino acids. On the contrary, Arginine and Lysine, when in more basic environments (higher pH), they lose a proton. Therefore, these amino acids are acids. Indeed, opposite charges tend to attract each other, and the same ones repulse one another. For this reason, one can assume that charges can help in detecting whether a residue can create contact or not. Negatively charged residues have displayed low contact values. However, pairs formed by positively charged residues have an average contact tendency. Additionally, Arginine and Lysine, which have a negative charge, had their orientation so that their charged parts would be very distant, resulting in a diminished repulsion between them.[16]. The charged amino acids are:

1. Negative charge - Aspartate and Glutamate.
2. Positive charge - Histidine, Arginine and Lysine.

**Residue type** - Amino acids differ from each other, and their characteristics imply some specific environments in order to pair. It is known that Arginine, Tryptophan and Tyrosine are present in 13.3%, 21% and 12.3% interface hotspots, respectively. However, Leucine, Serine, Threonine and Valine are found not to be frequently present in hotspots [31]. Residues link to each other by sharing hydrogen atoms. The nitrogen atom in the backbone donates one hydrogen, and then the carbonyl oxygen receives it, culminating in hydrogen bonds. This bonding plays an important role in 3D structures by supporting their secondary structure. Moreover, residues can be grouped by their hydrophobicities and charges. There are four standard groups:

1. Polar (hydrophilic) - Serine, Threonine, Tyrosine, Asparagine and Glutamine.
2. Non-Polar (hydrophobic) - Glycine, Alanine, Valine, Cysteine, Proline, Leucine, Isoleucine, Methionine, Tryptophan and Phenylalanine.
3. Acidic (negatively charged) - Aspartate and Glutamate.
4. Basic (positively charged) - Histidine, Arginine and Lysine.

### 1.3.2 Protein Structures

Although containing hundreds of individual bonds rotating freely, a protein has its own unique 3D structure due to its particular chemical and structural characteristics. Therefore, to study protein structures is essential to understand their function. Briefly, in here it is described all four levels of representation of protein structures: Primary Structure (section 1.3.2.1), Secondary Structure (section 1.3.2.2), Tertiary Structure (section 1.3.2.3) and Quaternary Structure (section 1.3.2.4). Figure 1.2 provides the visual context of the four levels of representation.

#### 1.3.2.1 Primary Structure

The primary structure is the raw ordered sequence of the amino acids, without any specification of the chain's geometry. This sequence is the core of the protein structure due to its impact in shape, and global characteristics of the molecule [18].

#### 1.3.2.2 Secondary Structure

The idea behind the secondary structure is to categorize the residues of the primary structure, considering the appropriate description of the shape that the peptide chain acquires. The 3D coordinates of residues are not part of this classification, only their sequence's shape also known as conformation. Hydrogen bonds induce residues spatial arrangements and the most common patterns are alpha-helices, beta-strands and loops, which are described below.

**Alpha helices** are created when the backbone of the peptide chain roles into a helix shape having hydrogen bonds stabilizing the structure along with the conformation. This stabilization occurs in a specific pattern: the carbonyl oxygen of the residue R1 will accept the hydrogen donated by the nitrogen of the residue R5 (4 residues apart) resulting in 3.6 residues per turn. There are other patterns, for example, three residues apart in very short segments or at the end of the helix, and five residues apart which causes helix holes along its axis reducing the efficiency of the structure. Although the last patterns are rare.[18].

Additionally, **beta strands** are also considered secondary structures induced by hydrogen bonding although the pairing of amino acids occurs between two sequences of the same chain. Indeed, some sequences of amino acids can interact with others, even if they are distant in the primary structure. Moreover, this linkage between two segments has been described to occur in two different ways [18]:

1. **Parallel beta strands** if the sequences are increasing in the same direction in both sides of the hydrogen bonds.
2. **Antiparallel beta strands** if the sequences are increasing in opposite directions in the ends of the hydrogen bonds.

Furthermore, it is also possible to find non-regular secondary structures, called **loops**, since they cannot form hydrogen bonds with other parts of the protein. Their loop-like shape identifies them, usually having the end and beginning of the loop residues close to each other in space. Loops have been found to act as connectors, either connecting secondary structures of one protein but also being involved in complex formation [18].

#### 1.3.2.3 Tertiary Structure

Commonly, the tertiary structure of a protein is its 3D shape. It is represented by all the secondary structures position and their spatial coordinates. For example, the combinations of beta-strands and alpha helices form the hydrophobic core of a globular protein.

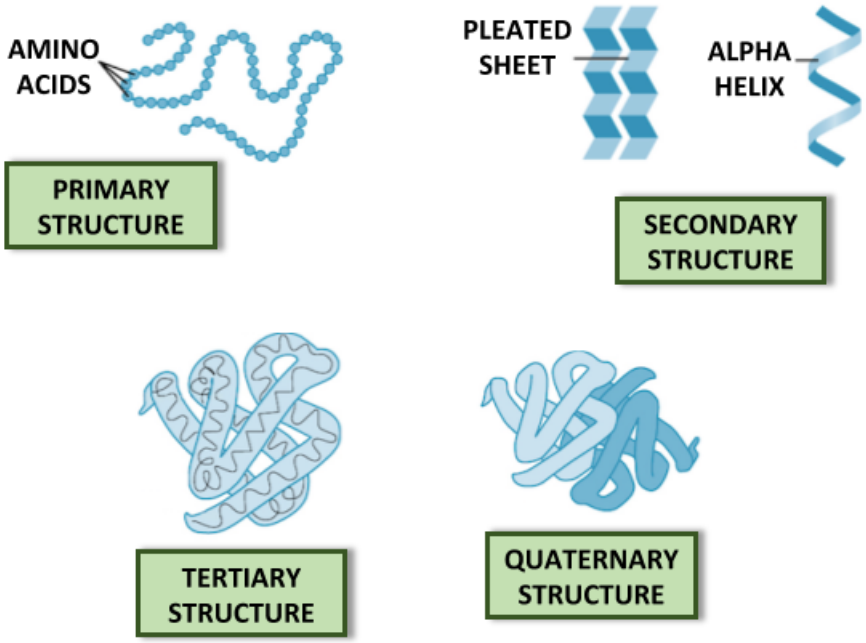


Figure 1.2: The four levels of protein structure: Primary structure, Secondary structure, Tertiary structure amd Quaternary structure. Adapted from: khanacademy.org.

Moreover, the placement of interconnecting loops is also important in creating complexes. The tertiary structure is described as the most stable form of a protein (lowest energy conformation) which allows staying in that conformation unless some functionality requires a flexible change [18].

#### 1.3.2.4 Quaternary Structure

The quaternary structure is the representation of the spatial relationship of several tertiary structures of a protein complex. A complex is known to be the binding of one or more polypeptide chains by non-covalent bonds. The nomenclature of these structures is given according to the number of subunits: monomer for one subunit, dimer for two subunits, trimer, tetramer or pentamer. Briefly, protein complexes that are constituted with equal subunits are named as homodimers. Although, when the subunits are different from each other, they are named as heterodimers [18].

### 1.3.3 Interface Characteristics

The binding sites between proteins and their ligands have been extensively studied in order to understand whether specific amino acids properties can interfere or even belong to the interface.

It is known that homodimers often create permanent complexes, and it has been found that their interfaces are mostly hydrophobic. Furthermore, studies have also demonstrated that large hydrophobic and uncharged polar residues are more present in protein

interfaces than charged residues and curiously they are more frequently found in homodimers. By analyzing surface patches, Jones and Thornton found that, in general, surface interfaces have a planar geometry. Therefore, for some types of complexes, their residues' solvent-accessible surface is higher in interface surface patches opposing to non-interface surface regions [14]. Additionally, interfaces areas tend to be proportional to the entire area of the whole protein surface [23, 28]. Regarding all studies, they all have found that none of the physical or chemical amino acid characteristics was unique and linear for all classes of complexes concerning protein interfaces.

### 1.3.4 Obligatory and Non-Obligatory Interfaces

A polypeptide chain of a protein-protein complex is called obligatory if it remains bound to another chain throughout its functional lifetime. On the contrary, there are a few protein-protein complexes in which proteins separate from each other under specific biological conditions. Usually, non-obligatory structures are stable in bound and unbound forms while obligatory chains are detected by being always in the bound form [13].

Studies have concluded that interaction patterns in interfaces of obligatory and non-obligatory chains are different and, importantly, obligatory chains contacts are mostly non-polar. Moreover, obligatory chains have more interface contacts (20 contacts on average) than non-obligatory chains(13 contacts on average). Additionally, the centers of both non-obligatory and obligatory chains interfaces are hydrophobic, revealing that they are non-polar. Therefore, the periphery is more polar than the center. However, non-obligatory interfaces are more polar than the obligatory ones, probably because they interact with the solvent when in the unbound form in order to stay stable [13]. Moreover, there is a significant tendency for obligatory chains to have larger interface areas, non-polar interface center, and to involve secondary structural elements across the interface to stabilize, namely beta-sheets. On the other hand, non-obligatory contacts occur in loops, generally.

All the mentioned features have subtle variations between the types of complexes, and because of that, a single feature cannot be used to predict different types of complexes [13]. However, separating the two types of chains in two different sets may be advantageous because the neural network will probably detect different features and patterns on each data set.

## 1.4 Machine Learning

Machine learning (ML) is an important field in computer science that joins artificial intelligence and the possibility to study and develop systems capable of learning from known data. Briefly, the ability of a program to learn, adapt and predict something is roughly compared to learning from experience, which makes ML a powerful tool nowadays. For

instances, explicit programming, designed by humans beings, find some tasks too difficult to solve, due to their complexity or unacceptable computational costs. However, ML systems' tasks are described in how the system should process a type of dataset (a collection of features). For this reason, ML is a multi-disciplinary field, since any data can be computed and learned by its systems through specific learning algorithms that learn to predict correct outputs. Firstly, the learning or training process of an algorithm is responsible for the adaptation of its own properties and actions based on the training set, in order to improve the results. Then, after trained, the algorithm is submitted to a test set (data that was not used for training) and, for each element, predicts an output based on the previously learned patterns.

There are three main types of algorithms: 1) Supervised learning; 2) Reinforcement Learning, and 3) Unsupervised learning [4]. All of them approach ML problems in their own way. Supervised learning algorithms learn by examples. This means that each example of the training set is labelled with the correct classification, which allows the model to generalize and produce correct answers through all the input data by detecting feature patterns. On the opposite, unsupervised learning algorithms experience data sets without labels, learning patterns and aggregating attributes from the data. Usually, the goal is to learn the entire probability distribution, which generated the data as density estimation. Some of these algorithms are used for clustering.

Moreover, if a problem consists in classifying an element discretely according to a set of possible classes, then it is a classification problem. However, if the wanted prediction is a continuous value, then it is a regression problem. Indeed, this work focuses on a supervised method solving a classification problem because the data is labelled by the correct protein chains' interface residues and the two classes are "atom is an interface atom"and "atom is not an interface atom".

### 1.4.1 Deep Learning

Deep Learning (DL) is a subset of ML algorithms which learn in layers. That is, these algorithms involve learning through several layers which allow the computer to build a hierarchy of complicated concepts based on simple contexts. For that reason, DL models learn to perform tasks directly from simple information like text, images or sounds (unstructured data), with outstanding performance, sometimes better than human-level performance. Additionally, in ML, feature engineering is an essential job to improve performance, and it usually requires domain knowledge for fine-tuning. However, DL algorithms can perform feature extraction by themselves. Therefore, these algorithms tend to outperform ML algorithms. DL is the technology behind the innovative technologies like voice control, driver-less cars and computer vision.

Recently, this type of algorithms became very popular in bioinformatics, computational biology and medical informatics since deep networks such as recurrent neural networks and auto-encoders have been studied and used to predict protein structures

and protein classification. Convolutional neural networks have also been used for enzyme classification and prediction of protein properties. This networks can outperform traditional approaches when dealing with spatial representations of data [5].

#### 1.4.1.1 Computer Vision

This project focuses on a Computer Vision task, called Semantic Segmentation. Computer Vision is an interdisciplinary scientific field that consists of how computers can gain high-level understanding from visual contexts (images, videos or any spatial representation). There are several Computer Vision tasks, some of them described below:

1. **Image Classification** - To classify an entire image based on its content. For example, to identify an image representing a cat as "cat", or an image representing a truck as "truck". This discrete label of the main object in the images is the most fundamental building block in Computer Vision.

2. **Image Classification with Localization** - Similar to the Image Classification but in this task, the computer should be able to localize where the object is present in the image. Usually, this localization is identified by a bounding box around the object.

3. **Object Detection** - Similar to the Image Classification with Localization, but now, the computer is able to identify several objects in the same image.

4. **Semantic Segmentation** - In semantic image segmentation, the goal is to label each pixel of an image with the corresponding class of what that pixel is representing. The output of this task is a high-resolution image (with the same size as the input images) whose pixels are labelled according to a particular class. Thus it is pixel-level image classification.

Indeed, in this project, the proteins' interface identification was approached as a Semantic Segmentation task, where the goal was to identify the interface atoms in 3D spatial representations of the proteins. Therefore, for a 3D spatial representation (protein), a voxel (name of a pixel in a 3D context) level classification is performed, where each voxel corresponds to a protein's atom. Therefore, each voxel is labelled with one of the two classes: interface atom or non-interface atom.

#### 1.4.1.2 Convolutional Neural Networks

Usually, Computer Vision tasks are performed by Convolutional Neural Networks (CNNs), a DL method that consists in performing feature extraction and classification in unstructured data, such as images, videos or other spatial representations. This section describes the different operations that are typically used in CNNs.

Usually, CNNs perform better than other types of neural networks since they can capture image aspects, such as edges or circles, successfully. For example, if a 3x3 image is

transformed into a 9x1 vector and then fed into a Multi-Level Perceptron for classification, the local spatial dependencies and shapes cannot be captured. Nevertheless, it is essential that the spatial relationships of the input volumes such as edges, holes, curved shapes or planar regions, are maintained. Therefore, the capture of these features is possible, using 3D CNNs. Forwardly, the CNN components that make the network work are described below.

**Input Data** - The input data needs to be in the form of a tensor: 1) a vector, 2) a matrix full of values for each pixel, considering an image, or 3) a rank-3 tensor, which contains values for each voxel (3D pixel), in the case of volumes. Moreover, the resolution of the tensors can be different depending on the type of data.

**Multiple Channels** - Sometimes, one data tensor is not enough to capture all the critical features to make useful predictions. However, the addition of more tensors with the same shape, but with different and relevant values, can help the model to find more useful characteristics from the input. Each input tensor is called a channel. One good example is the image data: an image can be viewed as three matrices, thus three channels, each one considering different colors: red, green and blue (RGB), with values from 0 to 255.

**Convolutional Neuron** - A convolutional neuron is responsible for the convolution of its input, which can be an element of the data set or the output of the previous layer. The operation happens by applying a filter to the neuron's receptive field, producing a feature map as the output. First, a brief description of what a filter and the receptive field are:

1. **Receptive Field** - The receptive field is a piece of the input tensors, a window of the matrix or a small volume of the 3D data. Additionally, the receptive field slides through the input tensor with a pace - Stride - and each data piece suffers the convolution operation. If a CNN uses three input channels, the receptive field is applied to all channels equally to perform the convolution operations.

2. **Filter** - Filters are tensors with the same dimension as the receptive field. These filter values are the weights that the network will optimize in order to represent a useful feature that helps to make a good prediction. Moreover, filters start with random values. However, when the network is trained, they should represent a relevant pattern or shape that helps in identifying a class.

The output of a convolutional neuron is the result of the tensor multiplication between the filter and all the receptive fields of the neuron input creating a new tensor, for each input channel. These output tensors are called feature maps, and they will be the input to the next layer.

**Convolution Layer** - Convolutional layers are groups of convolutional neurons, or filters. Typically, it is preferable to have a smaller number of filters in the first layers and then gradually apply more filters, layer after layer. This approach is desirable because the level of the detected features increases throughout the network layers. The first layers usually

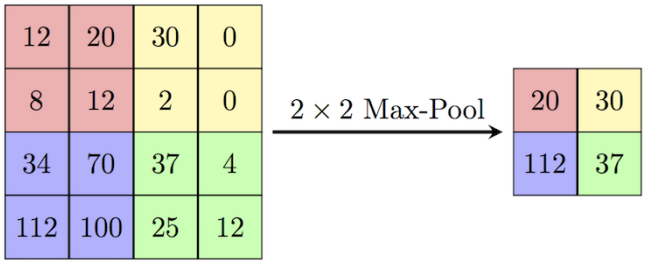


Figure 1.3: Pooling Operation. Adapted from: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling.

focus on small patterns and shapes like small edges, lines and curves, while deeper layers detect larger patterns such as faces, cars or even movements. The growth of the level of complexity of the features is a result of another operation called Pooling, explained next.

**Pooling** - A convolutional layer can produce a vast number of feature maps, and their size may be equal to their inputs or smaller depending on the convolution parameters choice. Pooling operations make those feature maps less dense by, for each zone of the feature map (a window in a matrix or a volume in a tensor), transcribing the maximum pixel or voxel value to the entire zone and reducing the overall resolution of the feature map. A visual example is provided in Figure 1.3. Usually, max-pooling is more used comparing to average-pooling because it captures the feature that is the most relevant. Finally, pooling layers can be used after convolutional layers to obtain the most relevant parts of the feature maps and, at the same time, reduce the computational complexity.

Besides the described components of the CNNs, other beneficial neural network techniques are commonly used (described below).

**Dropout** - The dropout technique is the shutdown of random neurons in the networks during the training phase. This technique is applied in convolutional neural networks by selecting random neurons and setting all their filter values to 0, forcing it to relearn a feature again. The main objective of this technique is to obtain different perspectives during training and also to control overfitting.

**Fully Connected Layer** - The fully connected layers (or dense layer) in a convolutional network are multilayer perceptrons that have the goal to map the feature maps, obtained from the convolutions, into a class probability distribution. This mapping is done successfully with an activation function. Finally, the calculated probabilities are used to determine the loss function value, based on the ground-truth, which is crucial to the model training. In figure 1.4 is an example of a network with a fully connected layer.

**Backpropagation** is the most popular optimization strategy for neural networks. The goal of this strategy is to find the ideal weight values for all the neural network neurons to minimize the result of the error function calculated in the last layer, training the model. A neural network produces an output, or prediction, once a data sample is submitted to it. With that output and the corresponding label (ground-truth), an error (or cost) is

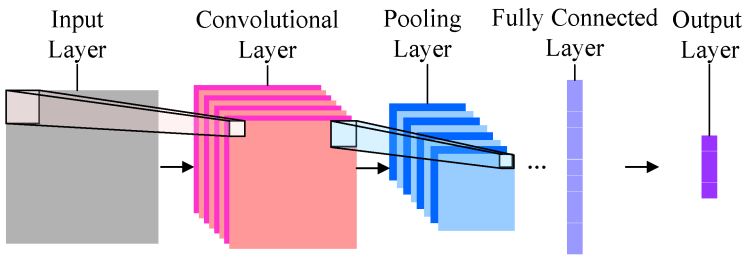


Figure 1.4: Convolutional neural network. Adapted from: https://www.mdpi.com/2078-2489/7/4/61.

calculated with an error function (or loss function). The derivative of this error function is its gradient. The idea is to use the error function gradient to know how the neuron weights, in all the network, should change to minimize the error. Therefore, the partial derivative of the error for each neuron weight is calculated in order to update them.

Briefly, CNNs use successive combinations of convolutional filters, pooling operations, activation functions and fully-connected layers between the input and output. They are built based on how the brain works as they learn to focus on the crucial spatial features that help to solve supervised tasks [5]. This algorithm performs convolutions, i.e. taking in an input image, apply the same filter on different image regions and consequently optimize the filter weights, through backpropagation, training the model. This operation allows the understanding of particular aspects of the data samples that identify the object in the image, making the model learn the right values of the filter weights. CNNs can have infinite architectures using a vast number of combinations of components. Nevertheless, the challenge is to find the best network for a specific problem, with the best parameters and with the best performance.

## 1.5 Convolutional Neural Networks to Identify Protein Interfaces

This project focuses on considering the protein interfaces prediction as a Semantic Segmentation task, by using their 3D structure disposition mixed with the residues physical and chemical features. Indeed, the 3D CNN algorithm was the most logical choice, for this context, because it shows promising results in other visual tasks. Additionally, this approach focuses mostly on the proteins chains' shape, since all the residues attributes are mapped into it, which allows a more detailed extraction of the critical spatial contexts. Many of the existing interface predictors use proteins sequences and structural residue information, however, mapping this information into a 3D context is more realistic and can help in determining interface sites, even with fewer features.

Indeed, our presented algorithm showed average results when comparing with other methods that did not approach this problem as a computer vision task. Therefore, this project shows that the protein interface prediction problem can be approached with 3D

CNNs. However, the amino acids information and the parameters in the network should be carefully optimized for better performance.

Chapter 2

# State of the Art

The main state-of-the-art methods and technologies that are relevant to the implementation of the current project are described in this chapter.

Firstly, a discussion about the tools and methods used to obtain the structural and physicochemical data is presented. Additionally, to understand/build an efficient and suitable CNN, some existing architectures and training strategies are described in detail. Finally, it is discussed how the evaluation of the prediction is performed, followed by a description of the needed tools to implement this project.

## 2.1 Data Extraction

In this section, the methods and tools for protein structural and physicochemical features extraction are covered.

### 2.1.1 Structural Data

#### 2.1.1.1 Classes of Proteins

Firstly, it is necessary a pre-selection of the kind of proteins that should be used as input. As previously mentioned in Section 1.3.3, interfaces are easily distinguished in homodimers by their hydrophobicity. Furthermore, proteases are characterized by having serine and histidine residues very active in their interaction sites, and for that, they are also easily detected using proteases primary structure. Two studies had these facts into account. Regarding one study about the prediction of interface residues in complexes [11], they split the data obtained from the PDB, and they trained and tested their neural networks with different types of protein. The set of proteins was split based on protein characteristics: 1) chain length (small proteins, medium proteins and large proteins); and 2)

being homodimers or heterodimers. However, the second study about predicting interactions sites but in heterocomplexes [15] focused on evolutionary conservation and surface disposition.

Another type of protein classification to be considered is whether the molecule interface is obligatory or non-obligatory. Section 1.3.4 discusses how the two types of interfaces have different structural and physicochemical characteristics. However, in this work, heterodimers and homodimers with obligatory chains were considered.

#### 2.1.1.2 3D Models

The 3D structures of proteins are retrieved from the files obtained from PDB [6]. Each protein file contains all its atoms and their 3D coordinates, allowing a spatial distribution of the structure. Considering the studies mentioned in the Section 2.1.1.1, there is another difference in the way how amino acids of each protein are represented. One of them used all atoms of the amino acids distributed through the 3D environment, on the contrary to the other study, which used only the alpha-carbon to represent the 3D structure of proteins.

In this work, the 3D representations were produced with all the amino acids' atoms. These representations helped in representing the orientation and size of the amino acids, which contributes to the interface classification.

Protein structures can be mapped into 3D grids with different resolutions. Another issue is how to fit the proteins in their volumes since they have different shapes and sizes. Indeed, two options can be considered: 1) to produce representations with different sizes according to the proteins' sizes, or 2) scale all proteins to one size. Moreover, the first option would lead to proteins representation in different resolutions, which is unwanted. Nevertheless, the second option is more favorable since the model of this work have the capacity of being aware size differences between structures, probably being considered an essential feature [5]. Additionally, using a fully convolutional network could avoid the different input resolutions problem since these networks do not use a fully connected layer as an output layer. However, in this work, the protein chains were cut, rotated and then placed in 3D tensors with size 64x64x32.

#### 2.1.1.3 Solvent Accessibility

Solvent accessibility, also known as Accessible Surface Area (ASA) is a property obtained from an algorithm that simulates a probe surrounding the surface of a molecule. This probe is a simulation of the solvent, in this case, water, travelling around the 3D structure of the protein and calculating the solvent accessibility of each residue [17].

Some programs calculate solvent accessibility. NACCESS [41](1992) efficiently implements the algorithm, and it is easily used through a command line. FreeSASA (2016) is a software in which the basis is the same as NACCESS; however, it presents some advantages relatively to NACCESS. For instances, it has very few dependencies, and it provides

Python and C APIs and benefits from multicore processing [30]. Both tools calculate ASA one molecular one structure at a time. One tool, PSAIA [29], can process entire sets of protein structures. Additionally, it performs its calculations on bound or unbound forms of the protein chains, which is of the interest of this project since only unbound structures were considered. Alternatively, ASA values can be obtained from a data bank called DSSP, which is generated from the PDB database [41].

Moreover, there are other attributes than ASA that PSAIA can obtain. RASA (Relative Accessible Surface Area) which is another measure of solvent exposure of the residue. The formula obtains RASA: $RSA = ASA/MaxASA$, where ASA is the solvent accessible surface area and MaxASA is the maximum accessible surface area for the residue. Additionally, DPX of a residue is the mean of the minimum distances from the residue's atoms and all solvent accessible atoms. Therefore, a large DPX of a residue means that it is buried inside the protein. On the contrary, if DPX is close to zero, it means that the residue is located in the chain's surface. In this work, ASA, RASA and DPX obtained from PSAIA were used.

### 2.1.2 Physicochemical Data

#### 2.1.2.1 Hydrophobicity

The representation of hydrophobicity can be binary or through scales. A binary representation of a residue could be: hydrophobic or non-hydrophobic. However, several available scales are more precise because they maintain a ratio between the amino acids with numerical values. Unfortunately, with the vast number of scales, some representations are contradictory due to different calculation methods [10]. One study used 144 different scales and averaged their values. This resulted in a distinction of three classes: Hydrophobic, Hydrophilic and Ambivalent, that may be used as 1, -1 and 0 values, respectively [42].

For this project, the relative values of the residues' hydrophobicities were considered, which allowed better feature extraction of the model. The hydrophobicities values were obtained from AAindex[19].

### 2.1.3 Data Augmentation

Protein structures do not have any specific spatial orientation. They can have any orientation on 3D conformational space unlike objects such as cars or tables. Indeed, the Cartesian coordinates of the PDB protein models are space and time snapshots of their overall dynamic structure. However, the orientation of the protein is unrelated to the protein properties. In order to extract features from several orientations of a structure, the dataset must be augmented by rotating the structures during the training phase [5].

There are two options: 1) rotate the structures based on the probabilities of flipping around each axis and the combination of the flips or 2) create new copies of the structure,

in which each one is rotated $360/n$ degrees around all axis, resulting in $n$ x 3 + 1 samples. The first one considers combinations of rotations, producing rotations that the second option cannot create. The second one is related to the interpretation of convolution that is weight sharing across translations since the same filter/weights travel across all the volume. Creating rotation samples is, implicitly, sharing weights across rotations [27].

In this project, a different approach was applied. All protein chains of the dataset were cut into slices and then rotated in 6 different orientations. Therefore, the samples of the resulting dataset are slices of protein chains in several orientations.

## 2.2 Convolutional Neural Networks

### 2.2.1 Convolutional Neural Networks in Other Protein Contexts

In this section are described several possible implementations of convolutional neural networks. In the context of this project, the goal was to find the model and the data that best suit protein interface detection. Indeed, there are studies regarding the study of proteins that used CNNs and obtained excellent results.

Torng and colleagues used 3D CNNs to identify protein functional sites (2018 [40]), zones of the protein that react with other molecules. Regarding this project, the purpose of their study is very similar to this project once it also considers the use of multiple channels, where each one represents the spatial distribution of an element. They also use three interpolated convolutional layers with three pooling layers, ending with a softmax probability estimation. Then, then optimize the weights during training and finally, they obtained the results. This study does not only detect the zone where proteins bind to other molecules, but also detect residues that help in the chemical reactions. Furthermore, they used multiple channels with different types of data.

Additionally, another study used 3D CNNs to detect symmetries and repeating patterns in protein structures (2018 [34]) using only one channel. They used as input a rank-3 tensor that was further submitted to two convolutional layers and then flattened. Afterwards, the resulting vector is used as input to a three-layer fully connected network. Moreover, this study focuses on patterns and symmetries of 3D structures.

The mentioned studies used different architectures; they both provided excellent results using 3D CNNs in 2018. Joining a vast number of other recent studies, CNNs are making a significant impact in solving learning problems. Moreover, these algorithms are being applied to all kinds of scientific fields becoming increasingly popular.

### 2.2.2 Prediction of Protein Interaction Sites

In this section are briefly described some of the existing interaction sites predictors. This studies and their results are used for comparison in the evaluation stage of this project, regarding the obtained AUC score.

#### 2.2.2.1 PSIVER

PSIVER is a protein interaction sites predictor which uses protein chain sequence features ( position-specific scoring matrix (PSSM) and predicted accessibility). These attributes are used for training a Naive Bayes Classifier with the objective of predicting the correct interface residues. Indeed, this classifier obtained a Dice score of 34.8% and an AUC of 0.62, testing with the DBD 3.0 (protein-protein docking benchmark set version 3.0).

Similarly to this project, they used a non-redundant dataset, where the protein sequences had less than 25% of sequence similarity. Additionally, they also used the amino acid positions and their solvent accessibility [32].

#### 2.2.2.2 PPiPP

This method uses PSSM and binding propensity scores between the types of residues to train a two-stage neural network to predict interacting pairs of residues. Additionally, they made a comparison between predicting the residue pairs (training analyzing the protein chain and the ligand) and predicting chains' interface residues without their ligands information.

This model successfully predicted PPI sites and achieves an area under the receiver operating characteristic (ROC) curve (AUC) of 0.73 when identifying residue pairs. Additionally, the model achieved an AUC of 66.1 when identifying single chains' interface residues [3].

#### 2.2.2.3 SSWRF

This method is an ensemble of SVM and sample-weighted random forests with the objective of dealing with the class imbalance that exists when predicting protein-protein interaction sites. Indeed, there is a more considerable amount of residues that are not interacting (negative class) than the number of residues that are interacting with another chain (positive class). This imbalance leads to a decrease in the overall performance of traditional machine learning classifiers. Moreover, they used PSSM, hydrophobicity and solvent accessibility from the target residues, and their neighborhood, as features to predict if the target residue is interacting or not. They further analyzed the proposed SSWRF using the DBD as the test set, achieving a Dice score of 35.1% and an AUC of 72.9% [44].

#### 2.2.2.4 DLPred

DLPred is a method that uses PSSM, physical properties such as solvent accessibility and residues hydrophobicities to train long-short term networks to identify protein interaction sites. They used a set of protein sequences instead of a set of single residues, to retain the whole protein sequence. Moreover, they have also tested this model with the test set of the DBD and achieved a Dice score of 54.7%, and an AUC score of 0.811 [45].

#### 2.2.2.5 PAIRpred

This method uses information from the two bounded protein chains to predict pairs of interacting residues from both of them. PAIRpred extracts sequence and structure features about residue pairs using pairwise kernels that are further used to train an SVM classifier. This method achieved a remarkable AUC score of 87% when evaluated with the test set from DBD 4.0 when trained for residues pairs detection. However, for protein chains' binding site detection, another SVM-based predictor was developed with the same characteristics as the previous one. The second predictor achieved an AUC value of 75,4% [2].

### 2.2.3 Network Architectures

Throughout the years, CNN architectures have been evolving according to different problems. Traditional convolutional network architectures with successive convolution layers and fully connected layers evolved and got better for image recognition (complete image classification). However, fully convolutional architectures (without fully connected layers) also advanced to perform better concerning semantic segmentation (pixel per pixel classification). Indeed, the goal of each layer in all CNN types is the same: to extract features from the spatial relationships of the input data. In this section, some of the most known architectures are briefly reviewed, highlighting some essential characteristics in the context of this project.

#### 2.2.3.1 Traditional Convolutional Neural Networks

Usually, as described in section 1.4.1.1, convolutional neural networks are composed by the stacking of several convolution and pooling layers. Therefore, a fully connected layer applies a function in order to distribute values through all the resulted feature maps from the last pooling layer. These values are used in the loss function to penalize incorrect filter values and to benefit the correct ones. Finnaly, the model is trained through backpropagation.

**AlexNet** - AlexNet became very popular due to its innovative characteristics. Its creators explored the depth of the model and its relationship to the model performance. Since a deep architecture is essential for high performance and it is highly demanding computationally, they used GPU processing to accelerate the training phase. This network consists in 8 weighted layers: 5 convolutional layers and 3 fully connected layers. Moreover, 3 max pooling layers are used after the convolutional layers 1, 2 and 5. The network architecture is displayed in Figure 2.1. The first convolutional layer has 96 filters of size $11 \times 11$ with padding with 2 pixels and a stride of 4 pixels. However, the rest of the convolution layers have a padding and a stride set as 1 pixel. The second convolutional layer as 256 filters of size $5 \times 5$. Layers 3, 4 and 5 have 384, 384 and 256 filters respectively, with size of $3 \times 3$ [21].

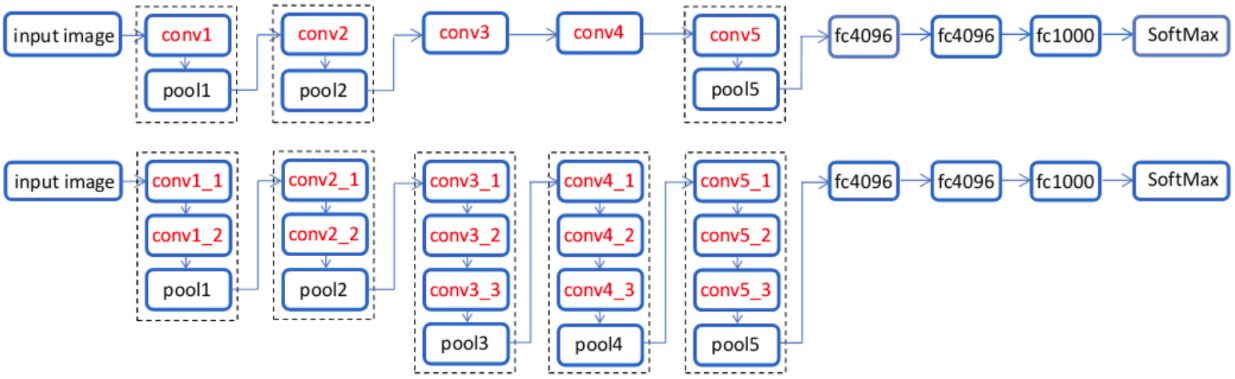


Figure 2.1: AlexNet and VGG Architectures. Adapted from:https://technology.condenast.com/story/a-neural-network-primer

**VGG** - This architecture was built based on AlexNet characteristics and showed good results in image classification problems. This network has 16 weighted layers and it is organized by convolution blocks. The first two convolution blocks are composed by two convolution layers, an activation (ReLU) and a max pooling layer, each. Blocks 3, 4 and 5 are composed by 3 convolutions, one activation (ReLU) and a max pooling layer, each (Figure 2.1). Additionally, the network has 3 fully connected layers after the convolution blocks with the same characteristics as the AlexNet. All the filter sizes in VGG convolution layers are of $3 \times 3$ and the stride is set to 1.

The receptive field in the first layers is smaller and has a stride, in contrast with AlexNet's first layer which has a big receptive field with a stride of 4 pixels. This causes the retainment of more background information, which may be unrelated to the problem ultimately affecting the final prediction. Moreover, two convolutions with kernel size of $3 \times 3$ obtain information with the same receptive field as one convolution with kernel size $5 \times 5$, but with less parameters. Therefore, VGG is deeper than AlextNet and, for that reason, has more parameters to be adapted. Concluding that the VGG performs better than AlexNet.

#### 2.2.3.2 Fully Convolutional Network

In opposite to the traditional convolutional networks, in which the final layer is a fully connected network, FCNs final layers are mainly convolutional. This approach makes the network susceptible to any input size. Usually, these networks are used for semantic segmentation (label each pixel with the class of its enclosing object or region). Moreover, after the last convolutional layer, upsample layers are added to ensure that the output size becomes equal to the input size. This is highly necessary to generate a pixel-by-pixel error by the error function for the given classes [26]. In here, two FCNs are discussed:

**SegNet** - SegNet is an encoder-decoder network which takes advantage of the convolution blocks of the VGG network (some convolutions, followed by the activation function

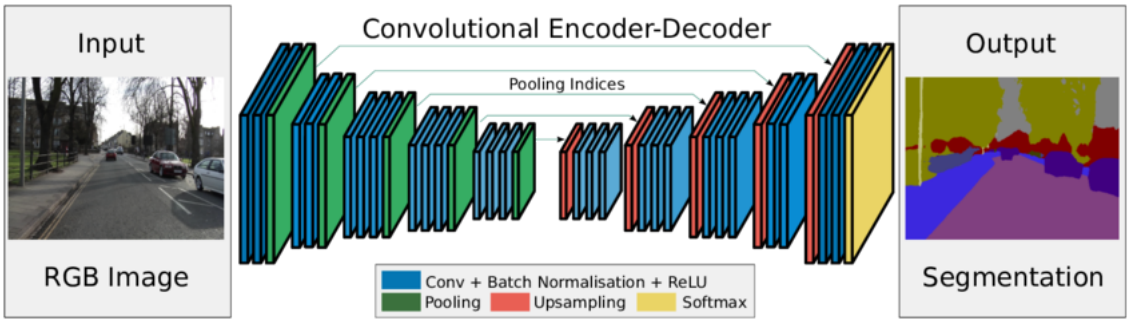


Figure 2.2: SegNet Architecture. Adapted from: Badrinarayanan, V., Kendall, A., Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(12), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

(ReLU) and ending with max-pooling). However, instead of using fully connected layers like VGG, the resulting feature maps, after the last pooling layer, are upsampled using upsampling layers. Furthermore, they are contextualized by using more convolutions. The upsampling-convolution process continues until the output feature maps are at the same size as the input data. Finally, the output layer produces pixel-by-pixel predictions. Segnet was used in semantic segmentation problems with great success [8]. The Figure 2.2 shows the Segnet architecture.

**U-Net** - uses almost the same architecture as the Segnet and adds the use of skip connections. This technique consists in adding results from the downsample side feature maps to the correspondent size upsampled feature maps. Indeed, U-Net uses multiscale information via skip connections to capture both coarse level and fine level information at the deconvolutional layers [38]. This network has more weights to update during training due to the skip connections. Therefore, it performs better than the SegNet, but the training phase takes a longer time. Figure 2.3 displays the U-Net architecture and how the features of the convolution side of the network are added to the deconvolution side.

In this project, the goal is to identify which atoms belong to the interface of protein chains; therefore, it is a pixel-by-pixel classification problem. Indeed, the U-Net was used as architecture since it is the network that obtains the best results in semantic segmentation. However, some parameters depend on the type of data. For example, the U-Net-based network used in this project treats 3D data instead of 2D, and the receptive fields must adapt to the problem.

### 2.2.4 Activation Functions

The activation function simulates brain neurons reaction over a particular stimulus, and if the neuron becomes activate or not. Moreover, this function is a non-linear transformation applied to the results of the convolutions. Importantly, activation functions introduce non-linear properties to the networks. Although linear equations are easier to solve, they

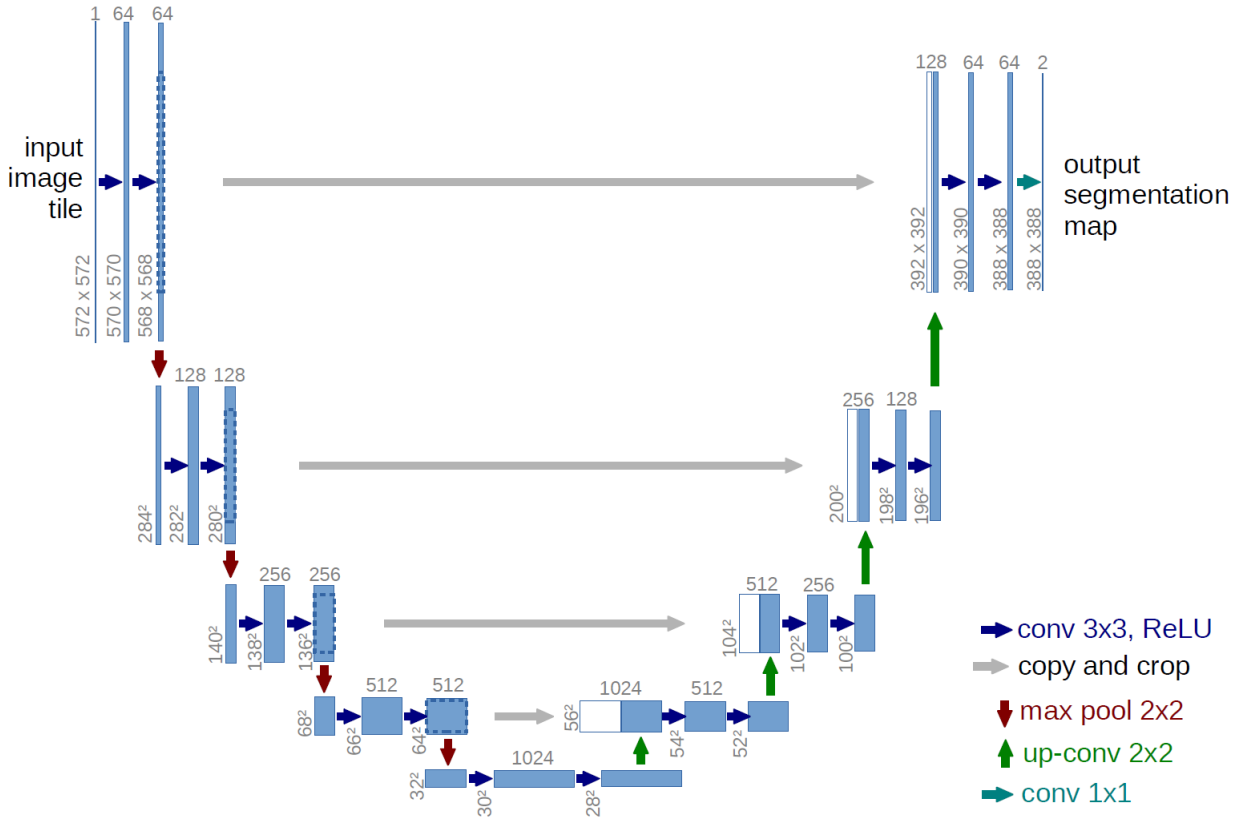


Figure 2.3: Unet Architecture. Adapted from: Ronneberger, O., Fischer, P., Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9351, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

are limited in their complexity. However, the goal of deep neural networks is to represent any function, and for that, the use of hidden layers with activation functions is crucial.

Sigmoid function, in Figure 2.1, consists in receiving one value and outputting a number between 0 and 1. It was one of the first activation functions in neural networks because the value 0 means that the neuron does not activate and the value 1 means that the neuron fully activates, the function is:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

However, when $x$ is very close to 0 or 1, the gradient is almost equal to 0 (vanishing gradient problem). During backpropagation, this local gradient is multiplied, and the error function gradient will get values even closer to 0, resulting in the loss of signal of that neuron. Additionally, the function is not zero-centered meaning that when the output of the function varies between 0 and 1, the gradient updates go too far in both directions.

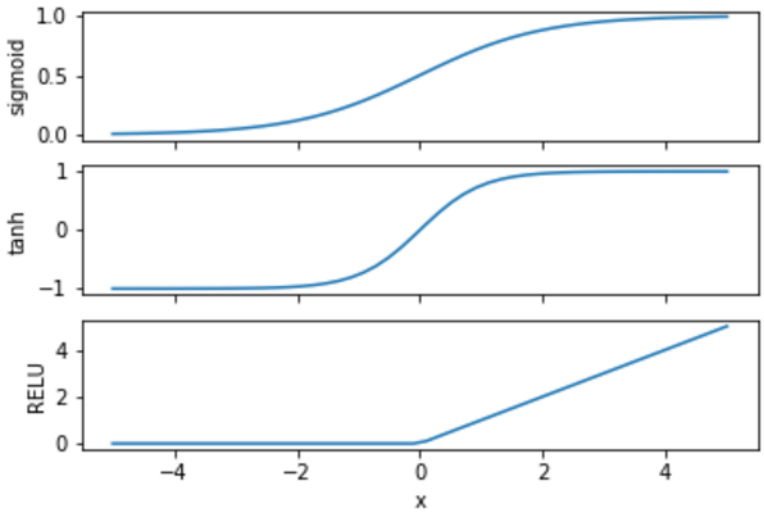


Figure 2.4: Activation functions: Sigmoid, Tanh and ReLU. Adapted from:https://technology.condenast.com/story/a-neural-network-primer

The hyperbolic tangent function, or tanh (Figure 2.1), is similar to the sigmoid function. However, its output is zero-centered, in the range of -1 to 1, which makes the filter optimization easier. For this reason, tanh is preferred to the sigmoid, but it also suffers from the vanishing gradient problem. The tanh function is:

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.2}$$

The rectified linear unit, or ReLU (Figure 2.1), is an activation function that became very popular in the last few years. Given a value $x$ it outputs 0 if $x$ is negative and outputs $x$ if positive. This function does not involve expensive operations like the other mentioned functions, which makes the learning phase much faster and, additionally, avoids the vanishing gradient issue [22]. ReLU is defined by:

$$g(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{2.3}$$

However, ReLU can cause neurons to die during training. Significant gradients flowing through a ReLU neuron can cause weight updates that make them never activate again. From that point on, gradients will be equal to 0. To solve this problem, the Leaky ReLU was introduced. The difference between this variant and the regular ReLU is on the negative side of the function. Instead of being 0 when $x$ is negative, it has a small negative tolerance avoiding the 0 value of gradients.

In this project, ReLU function was considered for the hidden layers since it is the most popular function due to having the best performance and fewer problems. If a vast number of neuron had "died"during the training phase, then the leaky ReLU would be

considered. The Sigmoid function was used in the output layer, producing the probability of each voxel belonging to the protein interface.

## 2.3 Evaluation

Nowadays, there is still a widespread concern regarding the evaluation of CNNs performance. Importantly, there is a need to assess the expected error in order two compare different architectures and to optimize model parameters.

Algorithms cannot be evaluated only based on the training data. Suppose an algorithm performs very well on training data. In that case, it might be overfitting/overtraining, meaning that the model is learning noise in that specific training data and, therefore, fails to generalize. Additionally, this situation leads to prediction errors when testing data different from the training set. Moreover, the validation set must be different from the training set. Importantly, it is guaranteed that the data used by the learning mechanism is different from the data used to evaluate the error and prediction. However, only a validation set is not enough when testing different models because both validation and training sets may be small and sensitive to noise, leading to false conclusions [4]. Additionally, CNN's usually perform differently from run to run due to the random attribution of filter values at the beginning. A solution is to calculate an average of multiple run errors.

Using a validation set indirectly affects the training phase of an algorithm, since it is used as guidance to obtain better predictions. Indeed, the validation set helps to determine when to stop learning in order to avoid overfitting [4], and it can help to optimize parameters like the input resolutions, the number of filters and the number of channels in CNNs.

After the model was trained and optimized, a test set was used to evaluate the model predictions. The test set corresponded to 15% of the available data.

### 2.3.1 Cross-validation

Obtaining the average of multiple validation runs implies the use of many training and validation sets. However, sometimes, data sets are limited, and there is not enough data the allows the use of many training and validation sets. For that reason, cross-validation can be used, with the same data is, but it is split into different sets. One existing problem on cross-validation is that data is shared between different combinations of training and validation sets. However, if the sets are large enough, error estimates are robust, while overlapping data should be small [4].

The most common method of executing cross-validation is the **K-Fold**. The data set is randomly split in $K$ equal sections. All the combinations of one of the $K$ sections being the validation set and $K-1$ being the training set are used. Importantly, to keep a large training set, validation set are kept small. Additionally, there is a considerable amount of

overlap among the training sets. For these reasons, $K$ value needs to be carefully chosen [4].

### 2.3.2 Measures for Predictions Evaluation

In this project, the notions of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) will allow the determination of several measures of comparison between the predicted structures and the corresponding ground-truths. TP voxels will correspond to correctly identified as interface amino acids. In opposite, TN voxels will be considered as non-interface residues. FN voxels will correspond to residues that are represented in the ground-truth as interface residues, but the model failed to identify them. On the other hand, FPs will correspond to the voxels which have been mispredicted as interface residues.

Volumetric measures have been already used to compare a predicted segmentation I with its associated reference mask M [20]. The most popular measure used to compare two segmentations is the Dice coefficient. It is a ratio defined as twice the size of the intersection between the two volumes, normalized by the sum of their sizes:

$$D = \frac{2|I \cap M|}{|I| + |M|} = \frac{2TP}{2TP + FP + FN} \tag{2.4}$$

The obtained coefficient can have values in the range of 0 to 1, where the value 0 corresponds to entirely different segmentations, and the value 1 corresponds to identical ones.

Moreover, considering all existing positives (P'), all existing negatives (N') and the total number of instances (S) this measurements will be calculated:

1. True positive rate, Recall or Sensitivity: $\frac{TP}{P'}$
2. False positive rate: $\frac{FP}{N'}$
3. Precision: $\frac{TP}{TP+FP}$
4. Specificity: $\frac{TN}{N'}$
5. Error: $\frac{FP+FN}{S}$
6. Accuracy: $\frac{TP+TN}{S} = 1 - error$

Additionally, the AUC (Area Under the Curve) and ROC (Receiving Operator Characteristics) curve is also an important performance measurement. The ROC curve is given by the TPR (True Positive Rate) against the FPR (False Positive Rate) obtained with different thresholds of the probability of identifying true positives. The AUC is the area from under the ROC curve to the $x$ axis. A good model has an AUC value close to 1, which means it has a good measure of separability between true positives and false positives, that is, the distribution of TNs is concentrated below the threshold, and the distribution

of TPs is located above the threshold. On the contrary, an AUC close to zero means that the distributions of TPs and TNs are overlapping, meaning that the model could not separate the predicted positives between classes.

## 2.4 Tools

Along with all known programming languages, Python is the chosen language to implement the proposed work. It is one of the most common programming languages used in machine learning implementation due to its easy syntax and easy access to a vast number of machine learning libraries with good state-of-the-art models.

**NumPy** - This Python package contains multidimensional arrays and mathematical functions to provide numerical computation without affecting performance [43].

**Matplotlib** - This graphical package for Python provides high quality graphical for several 2D plot types [39].

**TensorFlow** - TensorFlow (TF), similar to **Theano** and **PyTorch** is a framework developed by Google that provides tools to develop and train deep learning algorithms. Both TF and Theano serve as a base to a high-level library, called **Keras** which facilitates in building neural networks [1]. All three frameworks support GPU computation; however, Theano can operate in a single GPU while TF can perform in several GPUs. Additionally, TF has lower compile time than Theano which, in large models, takes long compile times. Moreover, an advantage that TF has is that it detects and uses the computer devices (GPUs) by itself while with PyTorch, everything needs to be moved to the device in order to use it. In short, the characteristics that make TF a better choice are: 1) the compilation time is very shot, 2) TF comes with a supporting tool called TensorBoard for an in-depth visualization of the training process [37], 3) Tensorflow has a much bigger community meaning that it is easy to find solutions and to learn, and 4) its device management is better.

**Biopython** - Provides several libraries regarding very bioinformatic problems, including modules for reading and writing different sequence file formats, and deals with 3D protein structures [12].

**Scikit-Learn** - This Python library provides efficient implementations for most of the known machine learning algorithms. Additionally, it supports validation, and feature manipulation [35].

**UCSF Chimera** - An open source software that allows the interaction and visualization with the PDB proteins. It was used to generate images to complement this document [36].

**PSAIA** - The software that was used to calculate the ASA and RASA values for all the amino acids of all the protein chains of the dataset. It uses an algorithm that simulates a solvent around the protein chains [29].

# Chapter 3

# Data

In this chapter, it is described and justified how approximately 6 000 PDB files are transformed into nearly 700 000 tensors. Rank-3 tensors represent data, like matrices, but they represent it in 3D. For example, the image of a cat can be represented in a 2D matrix (pixel-by-pixel), and a rank-3 tensor can represent a 3D model of a cat (voxel-by-voxel). In this project, these tensors represent spatial protein information where the indexes are the atom coordinates, and the values correspond to amino acid characteristics (for example, their hydrophobicity or amino acid group). 3D CNNs use this type of representation as to their input data. Therefore, it was necessary to convert the data from the PDB files into rank-3 tensors.

The first section is focused on the Protein Data Bank files, which contain the proteins' coordinates. This information is essential to build 3D representations and to extract features based on the location and neighbourhood of the atoms.

Afterwards, in the second section, is explained how the interface residues were labelled. This step was needed to compare the outputs of the neural network with their labels. Moreover, these labels allowed the optimization of the CNN's filter weights and the calculation of the metrics with the confusion matrix, evaluating the performance of the models.

Finally, in the third section is described how the rank-3 tensors were generated, containing different amino acids characteristics. Having the possibility of combining different types of data is useful to compare different models and their performances.

## 3.1 PDB Files

The PDB files, which can be downloaded from the PDB website, contain information about proteins structures. The PDB contains over 170 000 structures, which contain

chains of different types. However, not all of them are relevant to this project. By performing an advanced search in the PDB website, the following filters were applied:

1. Number of Protein Instances (Chains) per Assembly >= 2: To obtain only protein complexes, whose chains have interfaces.
2. Number of DNA Instances (Chains) per Assembly = 0; Number of RNA Instances (Chains) per Assembly = 0; Number o NA Hybrid Instances (Chains) per Assembly = 0: These chains are unwanted because they are not protein instances.
3. Experimental Method equals X-Ray Diffraction: NMR solution is another standard method but only 13 000 units in the PDB were obtained by it. However, 150 000 of the protein structures in PDB were obtained by X-Ray Diffraction. Therefore, filtering by this method guaranties more variety and quantity in the data set.
4. Refinement Resolution <= 2.5 Angstroms: Resolution of 1 Angstrom are highly ordered, and it is easy to detect every atom. On the contrary, by having a resolution of 3 Angstroms or higher, only the basic contours of the protein can be detected. Therefore, only structures obtained by X-Ray Diffraction with a resolution below 2.5 Angstroms were used.
5. Sequence similarity = 30% : This filter is critical because it reduces redundancy in the data set. The PDB has a set of clusters based on the sequence similarity between proteins. Having this filter set to 30% means that for each group of proteins that share 30% of their sequence, only one protein structure of that group is considered as a sample for this data set.

Finally, after filtering the PDB structures, a total number of 8 569 PDB files were downloaded. Briefly, a PDB file has two sections: 1) the header that contains the main details about each protein structure and the X-Ray Crystallography parameters used, and 2) the coordinates that represent the spatial data for each atom in the protein. Concerning the header, the tag RESOLUTION was obtained, which represents the resolution of the X-Ray Crystallography used to obtain the protein crystals and therefore, its structure. Regarding the coordinates section, each atom is represented by:

1. **Atom Number** - atoms are numbered in sequence through the file following the protein chain;
2. **Atom Type** - the type or element of the atom (for example ca = alpha C or o = carbonyl O);
3. **Residue Name** - the name of the residue in the three-letter amino acid abbreviation;
4. **Residue Number** - number of the residue in sequence through the protein chain;
5. **Chain** - the chain where the atom belongs;

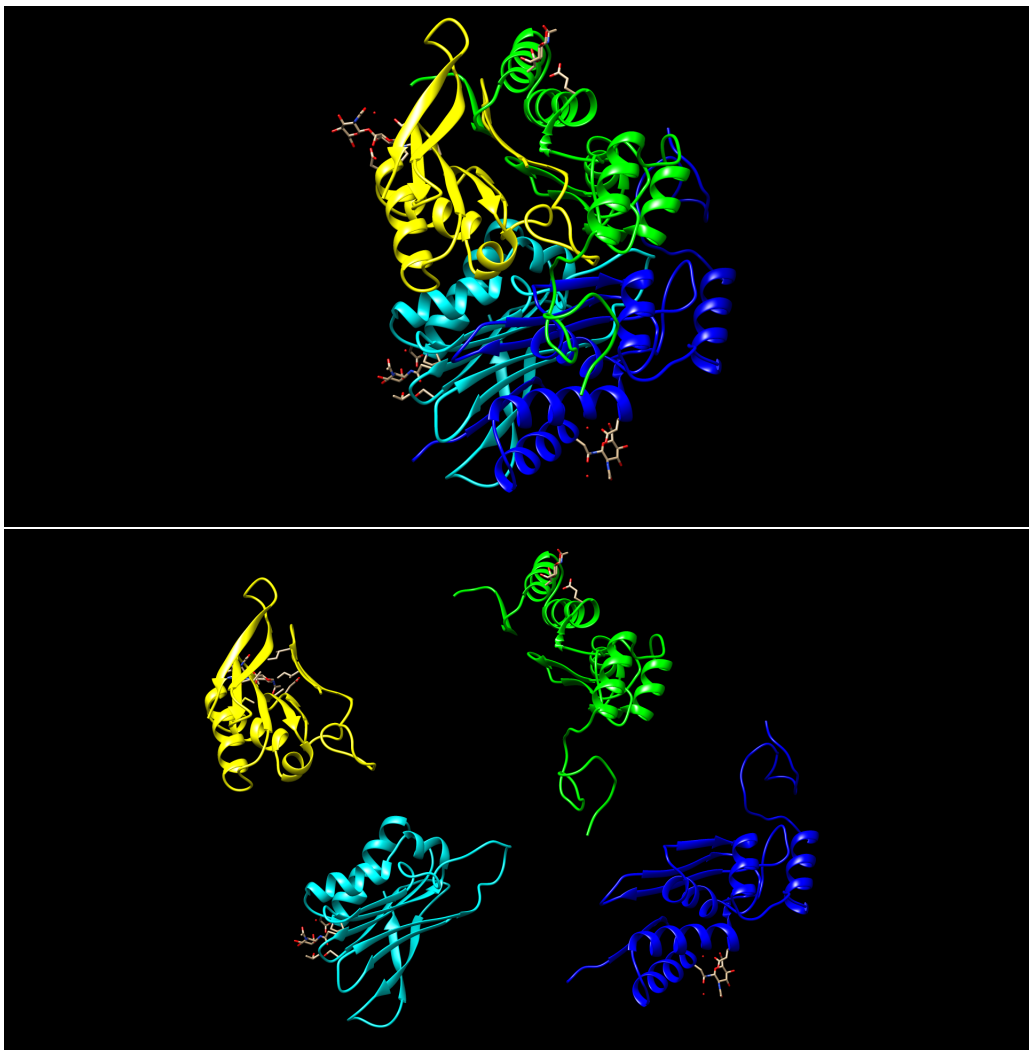

Figure 3.1: Protein Structures: Protein Complex called Human Aspartyglucosaminidase (1APY in PDB), each chain is represented by a distinct color. In the upper image, the chains are in their bound form and in the lower image, they are in their unbound form..

6. **The Coordinates** - x, y and z coordinates of the atom, in angstroms from the unit-cell origin (the center cell of the structure);

7. **Occupancy** - the fraction of unit cells that contain the atom;

8. **Temperature Factor** - this value represents the uncertainty of the atom's position due to disorder or thermal vibrations.

In the present Project, several characteristics of the atom were used, regarding the PDB files. The Atom Number was used to keep track of the indexes as an id. Moreover, the Residue Name was also used to allow the gathering of amino acids into amino acid groups along with the appliance of biophysical properties (such as the hydrophobicity) from the AAIndex for each amino acid. The Chain attribute allowed the split of the whole

protein complex by its chains, and the Coordinates were used to represent the amino acids and their features spatially.

While the PDB files were processing, more filters were applied:

1. **Only structures with more than one chain** - even with the PDB filter, a large number of protein structures contained only one chain. Those structures were discarded because only proteins with several chains could be used to calculate the chains' interfaces.

2. **Only chains with more than 20 residues** - when calculating chains interface, some small chains were detected, and when calculating their interface, it was noticed that all the amino acids were considered interface amino acids. This interfaces could mislead the neural network to consider all the chains' residues as interface residues. Moreover, all the chains were cut into slices, and all those slices were rotated to the same orientation. When small chains were cut, the resulting slices would contain a small number of atoms that would not contribute to the network training phase.

3. **Only chains with more than four residues in its interface** - It is known that interfaces are areas of the chain's surface which contains several interacting residues. Having a chain with only one or two interacting residues may be misleading to the neural network because they are probably not interacting with other chains. Indeed, the goal of this filter is to let the neural network focus only on large groups of interacting residues.

Finally, after all the filtration applied on the PDB website and during the PDB files processing, 6 131 PDB files generated a total number of 19 096 chains. The next section will describe how each chain interface was calculated. These interfaces were then used as labels in the neural network.

## 3.2 Interface Calculation

3D CNNs use rank-3 tensors as data representation, and this Project faces a semantic segmentation problem. Indeed, the goal is to predict the 3D representation of the interface atoms of the chains from the protein complexes (monomers). Moreover, to make the CNN update its filters weights, its output must be compared to the real interface representation (the ground-truth). Therefore, the Chain attribute from the PDB files was used to separate the protein complexes atoms by their corresponding chains. Furthermore, the distance between amino acids from different chains was calculated. This distance was used to evaluate the interaction between atoms from different chains, that is if they belonged to their corresponding chain's interface.

Calculating the distances for each atom of each chain takes a long time, especially if a complex has more than two chains. In this sense, to avoid spending too much time

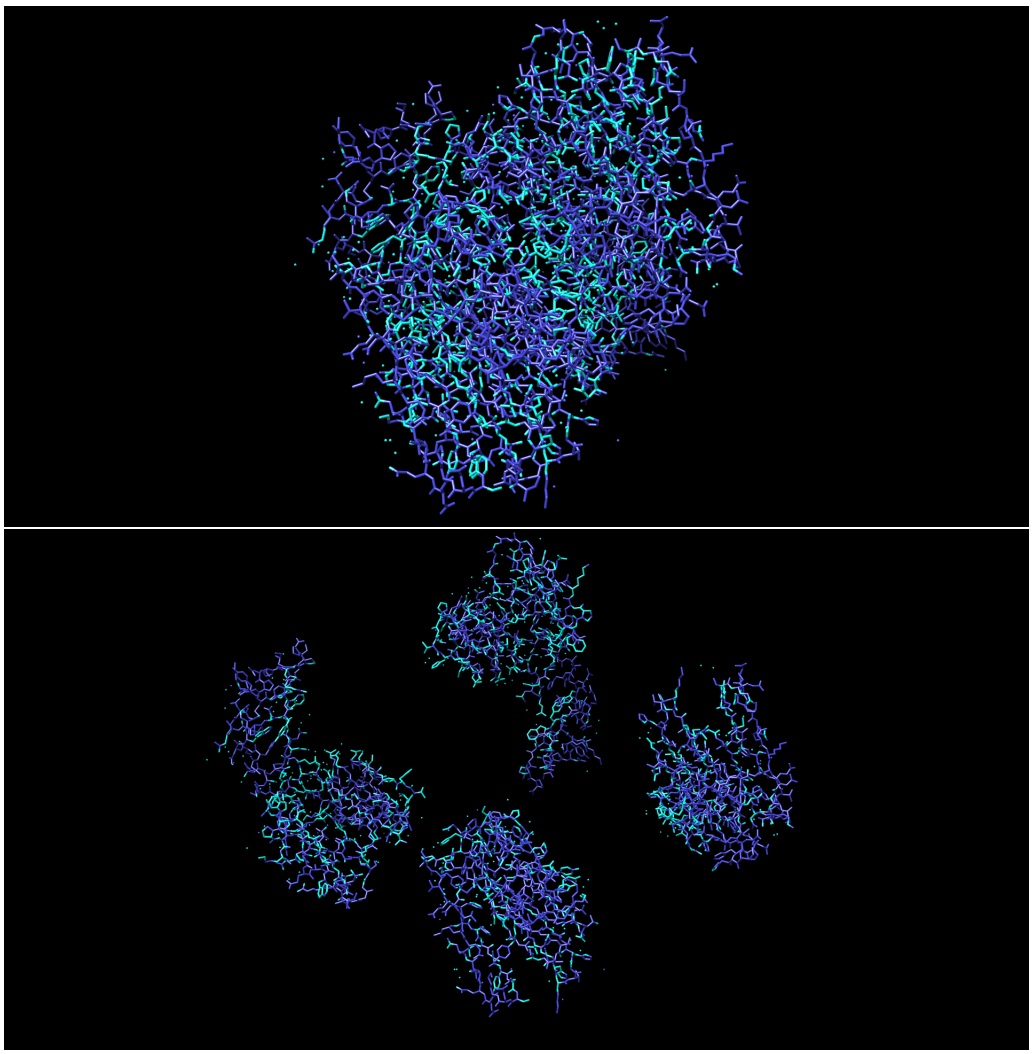

Figure 3.2: Protein Interfaces: Protein Complex called Human Aspartyglucosaminidase (1APY in PDB). The atoms in dark blue represent atoms of non-interface residues; and the light blue atoms represent atoms from interface residues. In the upper image, the chains are in their bound form and in the lower image, they are in their unbound form.

calculating distances, a K-Dimensional Tree (KDT) was used. This data structure is a binary tree that considers that every leaf is a k-dimensional point. In the case of the non-leaf nodes, they represent hyperplanes that divide the space into two sides. Each non-leaf has a dimension associated and divides the set of 3D points into two subtrees. For example, if for a particular split the y-axis is chosen, the points with higher and smaller y value than the node will be on the right and on the left subtree, respectively. This method brought a faster way to find connecting atoms between chains. 5 Angstroms was the used distance offset. Finally, it was possible to build 3D spatial representations of the chains interfaces to serve as ground-truth to the 3D CNNs (Figure 3.4).

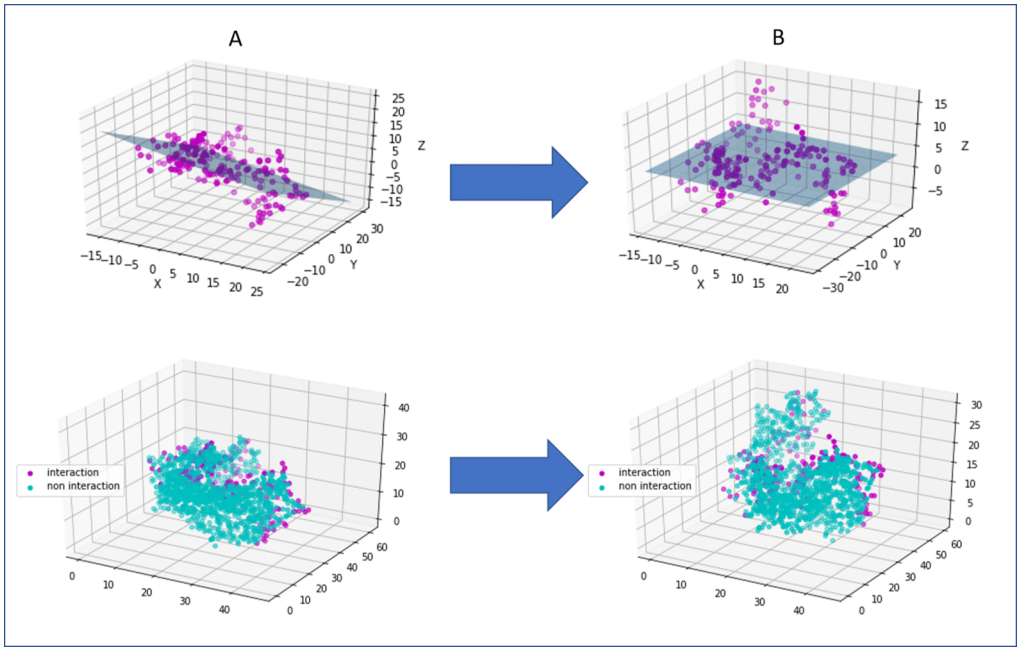


Figure 3.3: Representation of the rotation of a protein chain based on the linear regression applied to its interface atoms coordinates. On the left side (A), the chain (A below) and its interface (A above) before the rotation (interface regression plane is still tilted); on the right side (B), the chain (B below) and its interface (B above) after the rotation (interface regression plane is perpendicular to the z-axis)

## 3.3 Creating Chain Slices

Regarding the protein size, even though the splitting of the protein into chains was performed, some chains remained too big in a 3D context. For example, the computation resources could not process a (100, 100, 100) input, especially with several channels. Although cropping or resizing were options, these operations could result in the loss of information. This could happen because: 1) cropping a very large chain could cut away its interface because it is located in the surface of the structure; and 2) resizing could cause overlapping atoms due to the 1 Angstrom resolution.

### 3.3.1 Orienting Protein Chains based on Interface Linear Regression

To overcome large protein chain sizes, the alternative was to split the chains into slices. However, this splitting could not be done randomly because a slice with at least one entire interface is required to obtain better features. Moreover, protein interfaces may have different forms and orientations, and for that reason, all slices should be represented in the same orientation to maintain a stable representation environment. Preferably, the orientation followed the direction of one coordinate axis allowing an easier cut in the chain. To guarantee that this is accomplished, a Multiple Linear Regression (MLR) was applied on the interface atoms. MLR consists in modelling the linear relationship between independent variables and dependent ones. Specifically, in 3D coordinates cases,

it can be used to calculate the plane that fits in a set of points. Concerning this project, this method was used to find the equation of the plane that fits the interface atoms of a protein chain. Once the equation was obtained, its normal vector was also obtained.

Importantly, the slices were rotated in order to their interface plane become perpendicular to the z-axis. Therefore, a 3D rotation matrix was built to rotate the interface plane's normal onto the z-axis unit vector  = (0,0,1). For instance, the rotation matrix that transforms the vector **a** into vector **b** was built in the following way:

1. Let $v = a \times b$
2. Let $s = \|v\|$
3. Let $c = a \cdot b$
4. Let $V_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$
5. The rotation matrix R is given by: $R = I + V_\times + V_\times^2 + \frac{1-c}{s^2}$

By applying the rotation matrix to all the coordinates of the atoms set, the protein structures adopted the position in which the interface plane is perpendicular to the z-axis. Afterwards, the chain structure could be cut while maintaining at least one of its interface intact. The chains were cut in half. Regarding the z-axis, resulting in two slices, containing the bottom and the top half of the chain structure. Moreover, the left, right, front and back halves of the structure were also obtained. In total, six slices were generated, and at least one of the slices included an entire interface while the remaining slices may or may not have included parts of interfaces. Therefore, there were slices without interface atoms, serving as negative slices to the CNN. In this stage, the six slices have different orientations, and CNN's have rotation sensitivity, which means that shapes and patterns represented in different angles are considered different features. To overcome this problem, all slices were rotated to the same direction.

## 3.4 Tensor Generation

After the generation of the slices through rotation and slicing methods, they needed to be converted into rank-3 tensors to obtain their 3D representation in a way that the CNNs could use. As previously mentioned, rank-3 tensors are like matrices but in 3D. They can represent 3D models if their indexes act as 3D coordinates. For example, an atom with the coordinates (10,10,5) can be represented in the indexes (10,10,5) of a rank-3 tensor. Additionally, the value of that position in the tensor can be any atom characteristic (hydrophobicity, corresponding amino acid group, charge and more). However, the chains slices coordinates needed some transformations before being converted into

tensors. Firstly, the atoms' coordinates were updated to be all positive, and to start at the point (0,0,0) because the indexes of the tensors must be positive. Therefore, this was obtained by finding the minimum coordinates for each axis and then applying translation transformations to all coordinates to obtain the desired final position. This transformation consisted in finding the offset between the value zero and the minimum atom coordinates of each slice, in all three axes. The offset was negative if the minimum coordinates were superior to zero and positive otherwise. Then, the offsets of all axis were added to all coordinates. As a result, the new minimum atom coordinates of the slices is (0,0,0).

Once the coordinates were ready, the decision of what amino acid characteristic should be used had to be taken. However, some rank-3 tensors and 3D CNNs relationship characteristics must be clarified. Any value can be inserted to a position in a tensor, but the CNN can interpret the values of rank-3 tensors in different ways. For example, if in the same tensor some values equal to 1 and others equal to 2, the positions with the values equal to 2 will be considered more relevant than the positions with the values equal to 1. Therefore, to represent categorical features of amino acids, such as the groups of amino acids, the best way is to use binary tensors, with values equal to 1 to represent atoms of a certain amino acid type in specifics positions and values equal to 0 representing the opposite. However, some characteristics needed to be compared between atoms, such as their hydrophobicity. As previously mentioned, interfaces are usually hydrophobic, and there are types of amino acids more hydrophobic than others. In these cases, a tensor with different values according to the atoms hydrophobicity (referring to its amino acid type) was wanted.

Furthermore, different rank-3 tensors were created representing different amino acids characteristics distributed through the slices' atoms coordinates as indexes. Moreover, the rank-3 tensors were joined together, creating a rank-4 tensor which was the input to the 3D CNNs. Each rank-3 tensor of the input is called a channel, and, in the first convolution layer, they were processed independently. The results of the convolutions applied to each channel were added, resulting in the feature maps of that layer.

The way how amino acids types and characteristics were represented in rank-3 tensors is described below:

1. **Amino Acid Groups** - To avoid a large number of input data channels in the CNNs, amino acids were grouped by their characteristics. Across the numerous amino acids characteristics, in this Project, the following were considered: 1) hydrophobicity, and 2) charges, which define the standard four amino acid groups: polar, non-polar, acidic and basic groups. Therefore, the chains slices' atoms were split by their residue type group, and a rank-3 tensor represented each group with the values "0"and "1"distributed through the coordinates of the atoms (tensor indexes).

2. **Charge** - The first thought was to create tensors with the charge information of each residue. However, this information is already explored through the amino acid

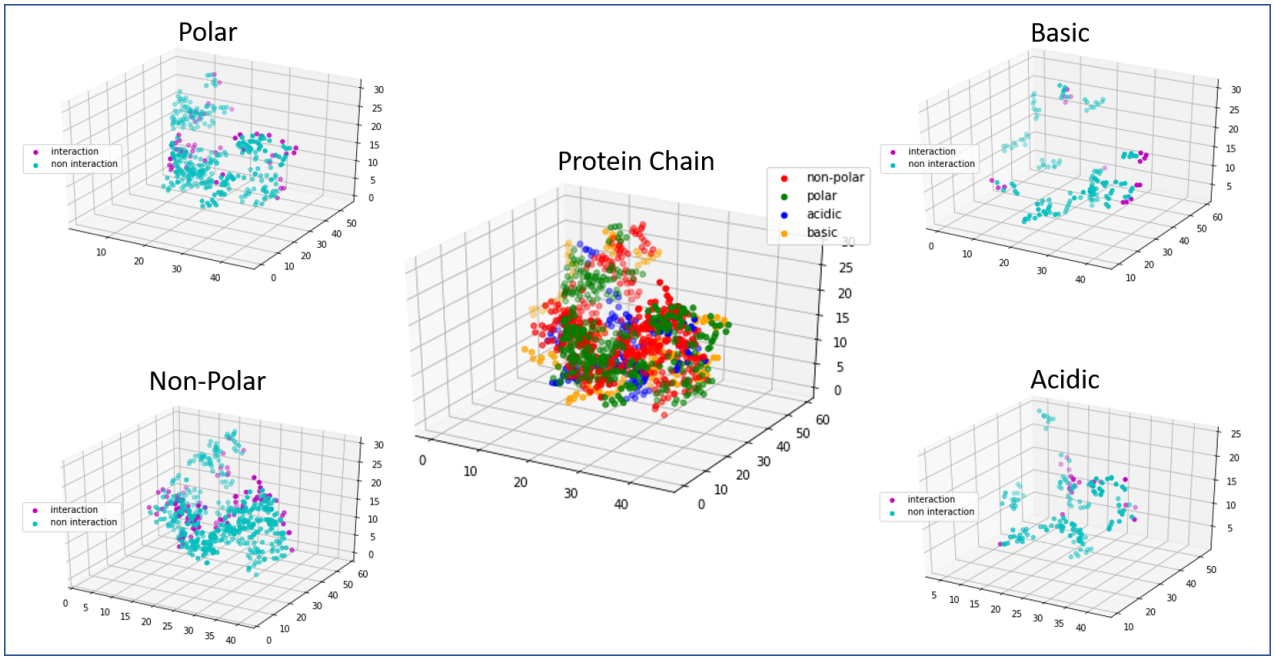


Figure 3.4: Representation of each amino acids group of a protein chain. In the center, the protein chain with all the amino acids groups together (non-polar, polar, acidic and basic). Each group spatial representation is also displayed with the visualization of the interacting atoms of the specific group.

groups tensors because acidic and basic amino acids are negatively and positively charged, respectively, while polar and non-polar amino acids are uncharged. Therefore, there is no need to create rank-3 tensors with the charge information of the slices.

3. **Hydrophobicity** - a tensor with the hydrophobicity information is essential. Although polar and non-polar tensors already represent hydrophobic and non-hydrophobic amino acids, this data is binary. Moreover, it is known that some amino acids are more hydrophobic than others, which is correlated with interface characteristics since these residues can be more hydrophobic than non-interface residues. Therefore, a rank-3 tensor was created based on the slice's atoms coordinates (indexes) with the values of the amino acids' hydrophobicities obtained from the AAIndex.

Moreover, some physical properties were obtained using PSAIA: the area of solvent accessibility, the relative area of solvent accessibility and the DPX for residues from each protein chain in the data set.

1. **Area of Solvent Accessibility (ASA)** - It is known that the interface residues are located in the protein's structure surface. Thus, they have a higher value of ASA. This information is useful because it is known that more hydrophobic residues are usually located inside the protein structure. However, interacting hydrophobic residues are located on the surface. Therefore, hydrophobic residues with higher

Table 3.1: Amino acid attributes.

| Residue Type | Hydrophobicity | Group | Max ASA |
|---|---|---|---|
| GLU | -1.10 | Acidic | 194 |
| PHE | 1.35 | Non-Polar | 197 |
| MET | 1.00 | Non-Polar | 188 |
| ALA | 0.02 | Non-Polar | 106 |
| VAL | 1.13 | Non-Polar | 142 |
| THR | -0.77 | Polar | 142 |
| CYS | 0.77 | Polar | 135 |
| HIS | 0.26 | Basic | 184 |
| SER | -0.97 | Polar | 130 |
| LEU | 1.14 | Non-Polar | 164 |
| LYS | -0.41 | Basic | 205 |
| ILE | 1.81 | Non-Polar | 169 |
| ASP | -1.04 | Acidic | 163 |
| TYR | 1.11 | Polar | 222 |
| GLN | -1.10 | Polar | 198 |
| GLY | -0.80 | Polar | 84 |
| ARG | -0.42 | Basic | 248 |
| ASN | -0.77 | Polar | 157 |
| PRO | -0.09 | Non-Polar | 136 |
| TRP | 1.71 | Non-Polar | 227 |

ASA can mean that they are interacting in the surface of the protein chain. For this reason, tensors with the ASA values for each residue, in all atoms coordinates, were created.

2. **Relative Area of Solvent Accessibility (RASA)** - This is the measure that compares the ASA of a residue with the standard maximum ASA value for that type of residue (fourth column in Table 3.1).

Moreover, one more attribute was calculated, which represents the tendency of a residue to be located inside the protein. This can be calculated by analysing the abundance of each residue in the whole protein, $AR_W$, and in the surface of the protein, $AR_S$. $AR_{iW}$ is obtained by dividing the number of residues of the type $i$ for the total number of residues in the protein, $N_W$:

$$AR_{iW} = \frac{N_{iW}}{N_W} \tag{3.1}$$

$AR_{iS}$ is obtained by dividing the number of residues of the type $i$ for the total number of residues in the protein, $N_S$:

$$AR_{iS} = \frac{N_{iS}}{N_S} \tag{3.2}$$

Table 3.2: Total number of protein complexes, number of chains, number of slices and tensors used for training.

| Number of Protein Complexes | Total Number of Chains (Monomers) | Total Number of Slices | Total Number of Tensors (For Training) |
|---|---|---|---|
| 6131 | 19096 | 114576 | 572880 |

Finally, the tendency for a residue of the type $i$ to be located inside the protein, $T_i$ is given by:

$$T_i = \frac{AR_{iW}}{AR_{iS}} \tag{3.3}$$

The tendencies of each residue were calculated to construct a tensor with that information. This attribute is useful because it makes possible the identification of residues that tend to be inside the protein but, in some cases, they are not, they are located in the surface of the protein which may indicate that the residue is interacting. In the table x is represented the tendencies for each residue type.

Concluding, from a set of 6 131 PDB files that represent all the protein complexes that were used, 19096 chains were considered, and each chain was described in 6 rotated slices. Finally, from each slice, nine different tensors were created. In Table 3.1 is presented the number of each type of data. In total, discarding the interaction residues tensor for training, there are 572880 tensors with relevant information that will be used to train the CNNs. Below are listed the nine types of rank-3 tensors generated for each slice, and that were used by the CNNs:

1. Hydrophobicities
2. Polar Residues
3. Non-Polar Residues
4. Acidic Residues
5. Basic Residues
6. Accessible Surface Area
7. Relative Accessible Surface Area
8. Tendency of location inside the protein for residue
9. Interaction Residues (Ground-Truth)

Table 3.3: Data set division based on the number of chains.

| Training Set (70%) | Validation Set (15%) | Test Set (15%) |
|---|---|---|
| 13367 | 2864 | 2864 |

## 3.5 Loading the Data

In this section, it is described how the data set was divided into training, validation and test sets. Moreover, it is explained how each set was loaded to be processed by the CNN models.

Importantly, each chain was considered a sample. Slices could not be samples because it would imply having slices of the same chain in different sets. Additionally, there are many slices that do not contain any interface residues, and that could cause imbalances in the sets due to the random factor of the data set division. So, the chains were shuffled, and the first 70% were the training set, followed by the next 15%which were in the validation set, and the last 15% were in the test set (Table 3.2).

During the CNNs training, it was impossible to load all data at once. At this step, the solution was to build a custom generator using the Keras Sequence functionality, which allows to load and preprocess the data in batches. The use of this custom generator had one primary focus: to consider one sample as one chain. This means that all six slices of one chain would enter in the CNN together.

# Chapter 4

# 3D CNN Implementation

This chapter focuses on explaining how CNNs were built and adapted to perform well in the context of this work. Since the main focus of this work is to build a model that performs 3D Semantic Segmentation in 3D models, several existing architectures served as base models. There are many choices to make when building a CNN, but the architectures described in Section 2.2.2 have obtained good results. In the next section is described how each component of the CNN was built and why. Furthermore, the final section of this chapter explains how the final CNN is composed.

## 4.1 Building CNN Components

**Input Layer** - The input size was set to $64 \times 64 \times 32 \times number\_of\_channels$. Firstly, the dimensions should be divisible by two in all convolution layers allowing a uniform pooling of $2 \times 2 \times 2$ and the stride value was set as two in the network. Secondly, because this input size includes most of the dimensions of the slices, meaning that all the smaller sizes were padded (added zeros around the structure until the tensor had the input size) and some of the bigger slices were cropped. Additionally, the number of channels varied to obtain different models, each one trained with different data.

**Convolution Blocks** - Convolution blocks were built using one or several convolution layers, batch normalization, an activation function and a max pooling layer, following the VGG model (Section 2.2.3.1). Importantly, organizing the network by blocks facilitates its building, editing and representation. Indeed, each block is composed by:

1. **Convolution Layer** - Each convolution layer has a number o filters (or kernels) which are rank-3 tensors whose values adapt through backpropagation according to the context of the relevant input data. Having more kernels in a convolution

layer corresponds to a larger number of shapes and patterns that can be captured. However, if some filters discovered all the relevant features and there are more filters in the same layer, then some may adapt to misleading features causing wrong predictions. Therefore, in this project, different filter numbers were tested.

Additionally, filters can have different sizes, meaning that they can extract features with different sizes, which can lead to misleading features by capturing background information. Therefore, convolutions with kernel sizes set as $3 \times 3 \times 3$, $5 \times 5 \times 5$ and $7 \times 7 \times 7$ correspond to different receptive fields. However, stacking convolution layers with the kernel size of $3 \times 3 \times 3$ can simulate higher receptive fields while having fewer parameters to update. For example, stacking two convolution layers with kernel sizes of $3 \times 3 \times 3$ ($3 \times 3 \times 3 \times 2 = 54$ parameters per filter) correspond to one convolution with the kernel size of $5 \times 5 \times 5$ ($5 \times 5 \times 5 = 125$ parameters per filter); and stacking three kernels with the size of $3 \times 3 \times 3$ ($3 \times 3 \times 3 \times 3 = 81$ parameters per filter) correspond to the receptive field of a kernel with the size of $7 \times 7 \times 7$ ($7 \times 7 \times 7 = 343$ parameters per filter); if the strides are set as 1. Additionally, large receptive fields generate low-resolution feature maps because the evaluated area of the input is large.

Furthermore, padding is set as "same", meaning that zeros were added around the feature maps to maintain the same size as the input. Otherwise, the feature maps would be smaller after each convolution. Indeed, the VGG network only downsamples the feature maps using the max pooling layer (Section 2.2.3.1).

In this project, all convolution layers had their filter size set to $3 \times 3 \times 3$, with stride set to 1 and padding set to "same". The number of filters and the number of convolution layers will be discussed in the next section.

2. **Batch Normalization** - Batch Normalization is a deep learning technique that is used to normalize the input of neural networks and the inputs of the activation functions in hidden layers. Indeed, it enables higher learning rates, scores and faster training by keeping a zero mean and unit variance along with the network. In this project, this method was used because otherwise, the training phase would take a significantly longer time.

3. **Activation Function** - ReLU activation function was used in convolution blocks because it is very quick to perform since it considers that a gradient is equal to 1 if the neuron is activated (Section 2.2.4). It is proven to perform better than other activations such as sigmoid or tanh in the hidden layers of the network.

4. **3D Max Pooling Layer** - Each convolution block has a max pooling layer to downsample the resulting feature maps from the convolution layers. This means that the resolution of the feature maps is reduced and only the maximum values of each region in the feature map are used in the downsampled feature map. Each pooling

layer has a window with a size set to $2 \times 2 \times 2$ and stride set to 2. These parameters makes that the result of the pooling layer is half of the size of its input. For example, if the input of the max pooling layer size is $64 \times 64 \times 32$ then its output result is $32 \times 32 \times 16$.

5. **Dropout** - this technique consists of the shutdown of random neurons in the networks during the training phase. In this project, experiments were made with and without dropout to obtain different and more generalized feature maps. In the end, the results presented the effect of the dropout in the models, with different probabilities.

**Number of Convolution Blocks** - the input has a resolution of 64×64×32 for each channel. Therefore, in this Project, after each convolution block, the feature maps resolutions are halved by the max pooling layer which means that after four convolution blocks the feature maps should have a size of $4 \times 4 \times 2$.

**Upsampling Blocks** - Regarding this Project, upsampling blocks are responsible for upsampling their input and extract features of the upsampled feature maps, based on the SegNet architecture. There is also an Add Layer which adds the feature maps obtained in the convolution blocks, to the feature map obtained from the first convolution layer of this block, based on the U-Net architecture (Section 2.2.3.2). Using this technique allows the use of relevant information from the features maps obtained from the convolution blocks (which is more fine-grained) and also the relevant information of the upsampled feature maps (more coarse-grained). After the Add layer, there is one more convolution, followed by the activation function. There is the same number of upsampling blocks as convolution blocks, and the output of the last upsampling block has the same size as the network input, $64 \times 64 \times 32$. In short, each upsampling block contains:

1. One 3D Upsampling Layer
2. One Add Layer
3. One 3D Convolution
4. Batch Normalization
5. Activation Function

**Output Activation Function (Loss Function)** - The output layer is a convolution with one filter with kernel size $1 \times 1 \times 1$ and a Sigmoid activation. The result of the activation is a value from 0 to 1 for each voxel in the $64 \times 64 \times 32$ tensor. Furthermore, those values were compared with the ground truth (the true spatial representation of the amino acid interfaces of the evaluated slice) using a loss function. The chosen loss function is the dice coefficient loss, which is the metric that better represents a segmentation.

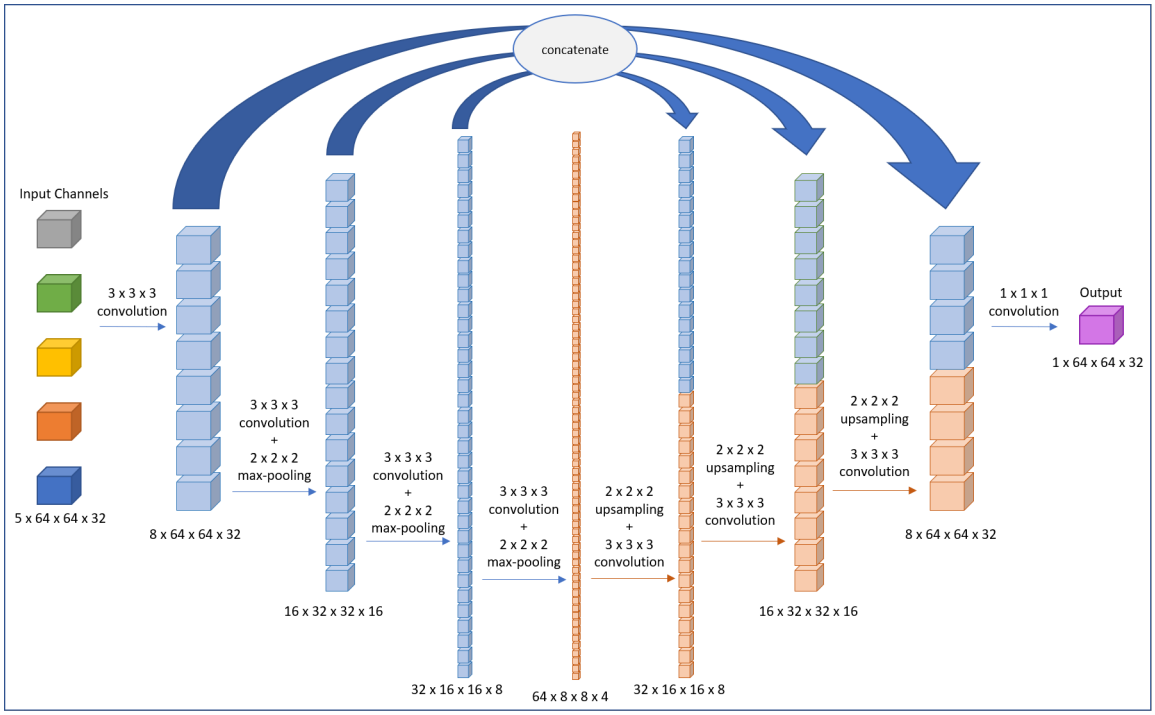


Figure 4.1: U-Net Architecture.

## 4.2 Full Architecture

Overall, the architecture used in this Project was based on the U-Net architecture and it was composed by a convolution side and and upsampling side. The convolution side had five convolution blocks, containing one convolution layer, batch normalization, the ReLU activation function and finally the max pooling layer. Moreover, the upsampling side of the network had four upsampling blocks, containing one upsampling layer, one concatenate layer which contains the upsampled feature maps plus the feature maps of the convolution side with the same size, one convolution layer, batch normalization and ReLU. Finally, the architecture is represented in Figure 4.1. This CNN was implemented using Keras.

Chapter 5

# Results and Discussion

This chapter contains all the training and test results with different network configurations. The main goal of this project is to identify interface regions of protein chains based on their known spatial structures and biochemical properties using 3D CNNs.

Therefore, the first stage of experiments is about the data that should be used. Indeed, the CNN was trained with five different inputs, corresponding to different channels. It is essential to notice that in this stage, the number of filters is small, and its primary purpose is to find the best combination of data. The first combination only contained the four channels with amino acid groups (polar, non-polar, acidic and basic residues). Afterwards, the channels containing the Hydrophobicity, ASA, RASA and Tendency of being inside the protein were added, one by one. Eight channels compose the last combination. Finally, the model that achieved the best validation dice value was evaluated in the next step.

The second step consists in increasing the number of filters on the model that had the best dice value in the validation set. The goal in this stage was to optimize the network to the type of data. Indeed, if the data is more complicated, which is the case of ASA and RASA, whose values are different for every residues or atoms, a network with more filters may have a better performance. Finally, the model which had the best dice value in the validation score was evaluated in the next stage.

The third stage consists of producing predictions using the test set with the best model of the second stage. These predictions were then analyzed with the confusion matrix metrics and the AUC ROC curve, evaluating the overall performance of the model. Moreover, these predictions were grouped in categories for a more in-depth analysis.

The fourth and final stage focuses on producing predictions on the DBD test set. The results were then compared with other state-of-the-art predictors which obtained results with the same dataset.

These experiments were made with 19 096 chains (with six slices each, corresponding

Table 5.1: Data set division based on the number of chains.

| Training Set (70%) | Validation Set (15%) | Test Set (15%) |
|---|---|---|
| 13367 | 2864 | 2864 |

to 114 576 slices), explicitly using 13 367 chains for training, 2864 for validation and 2864 for the test (Table 5.1). However, in the fourth stage, the DBD dataset was used to compare with similar predictors. Moreover, these experiments were processed in an NVIDIA GeForce RTX 2070 Super with 8Gb of memory. Additionally, in this Project, Adam was the chosen optimizer with the learning rate as 0.01 for all the tests and the dice coefficient was the metric used to compare the model outputs with the ground-truth (the chains interfaces' spatial representations).

## 5.1 Stage 1 - Training with different data

In this section are presented all of the training results of five models, trained with different combinations of data. The 3D CNN architecture had five convolution blocks, where each convolution block had one convolution layer with stride set to 1, kernel size set to $3 \times 3 \times 3$ and padding set to *same*. The number of filters in the convolution layer of each convolution block doubled from one block to the next, until the fifth convolution block. This means that the first block started with eight filters in its convolution layer, the second block had 16 filters in its convolution layer, then the third block had 32 filters in its convolution layer, the fourth had 64 filters, and the fifth had 128 filters. Moreover, following the fifth convolution block, the upsampling side of the network had four upsampling blocks where in each convolution layer the number of filters halved from the previous one.

The loss function was the Dice score which compared the outputs of the CNN with the corresponding labels. This comparison made all the filters' weights update through backpropagation. The metrics that were tracked during training, for both training set and validation set, were the Dice score, Precision and Recall. Precision because it was essential to be able to analyse how the ratio between true positives (TPs) and positive predictions, including false positives (FPs), evolved during training. The recall was also significant to understand how did the network identify the correct atoms during training (TP's against positive labels, including true negatives (TNs).

U-Net stands for the name of the architecture. Moreover, the different types of data that were used to train the U-Net models were: four channels representing the four amino acid groups (AG), one channel representing the residues hydrophobicities (H), one channel representing the accessible surface are of all residues (ASA), one channel representing the relative accessible surface area (RASA) and one more channel representing the tendency of the residues to be located inside the protein (T). Each channel is representing

all the atoms of the amino acids, that is, in the positions of the atoms is represented the value of the corresponding amino acid attribute. For example, in the hydrophobicities channel, the position of an atom of Histidine represents the Histidine's hydrophobicity.

In the Table 5.2 it is possible to verify that the model trained with AG and the model trained with AG + H channels had validation dice scores below the validation dice score of the models trained with AG + H + ASA, AG + H + ASA + RASA and AG + H + ASA + RASA + T. The groups of the amino acids proved to be an important characteristic to identify interface residues since they represent some amino acids features: hydrophobic amino acids in the polar group; hydrophilic amino acids in the non-polar group; positively charged amino acids in the basic group and negatively charged amino acids in the acidic group. The model trained with these groups achieved a validation dice score of 0.5429, a recall of 0.6501 and a precision of 0.4891. Therefore, the model identified 65% of the true interface atoms. However, the model had a large number of false positives that were caused by the absence of more complex characteristics which resulted in the precision value of 0.4891.

By adding a channel representing the hydrophobicity of the residues through all the atoms' coordinates, the level of complexity of the input data increased. Indeed, having a measure of comparison between different residues helped in finding more patterns that could correctly classify the residues as protein interfaces. The model trained with the amino acid groups and the hydrophobicities values (AG + H) achieved a validation dice score of 0.5618, a recall of 0.6708, and a precision of 0.5071. Comparing to the U-Net model trained with AG, the U-net model trained with AG + H had a better overall score. This happened because in the first model, the measure of comparison regarding the residues' hydrophobicities, was binary: a residue was hydrophobic or hydrophilic. However, with this second model, more in-depth comparisons were made: a residue could be more hydrophobic than others. Indeed, the obtained results proved that having a channel with more complex data, by having different values for different types of residues, helped the model in extracting better features and thus better performance in the distinction between interface atoms and non-interface atoms.

Moreover, the channel representing the residues' accessible surface area (ASA) was added. Indeed, the U-Net model was trained with the channel combination AG + H + ASA. Therefore, the complexity of the data was increased because the ASA values depend on the whole chain structure. The same residue can have different ASA values depending on where it is located in the chain. If the residue is located on the surface of the chain, in contact with a solvent, then the ASA value is larger compared to when the same residue is located within the chain structure, away from the surrounding solvent. Indeed, the interface residues are located on the surface of protein chains, and they are mostly hydrophobic. Therefore, hydrophobic residues with high ASA values have more chances of belonging to the chain interface. For this reason, the model trained with the combination AG + H + ASA obtained better scores than the previous model (AG + H). It achieved a validation dice score of 0.5766, a recall of 0.6719, and a precision value of

Table 5.2: Training results of the best validation score epoch from all the trained models in Stage 1, with eight filters in the first convolution layer, which were trained with different channels combinations.

| Models | Training Dice | Validation Dice | Training Recall | Validation Recall | Training Precision | Validation Precision |
|---|---|---|---|---|---|---|
| U-Net-8f (AG) | 0.5518 | 0.5429 | 0.6845 | 0.6501 | 0.4835 | 0.4891 |
| U-Net-8f (AG+H) | 0.5707 | 0.5618 | 0.6846 | 0.6708 | 0.5111 | 0.5071 |
| U-Net-8f (AG+H+ASA) | 0.5830 | 0.5766 | 0.6979 | 0.6719 | 0.5238 | 0.5304 |
| U-Net-8f (AG+H+ASA +RASA) | 0.5851 | 0.5776 | 0.6982 | 0.6897 | 0.5264 | 0.5187 |
| U-Net-8f (AG+H+ASA +RASA+T) | 0.5874 | 0.5751 | 0.6982 | 0.6726 | 0.5301 | 0.5265 |

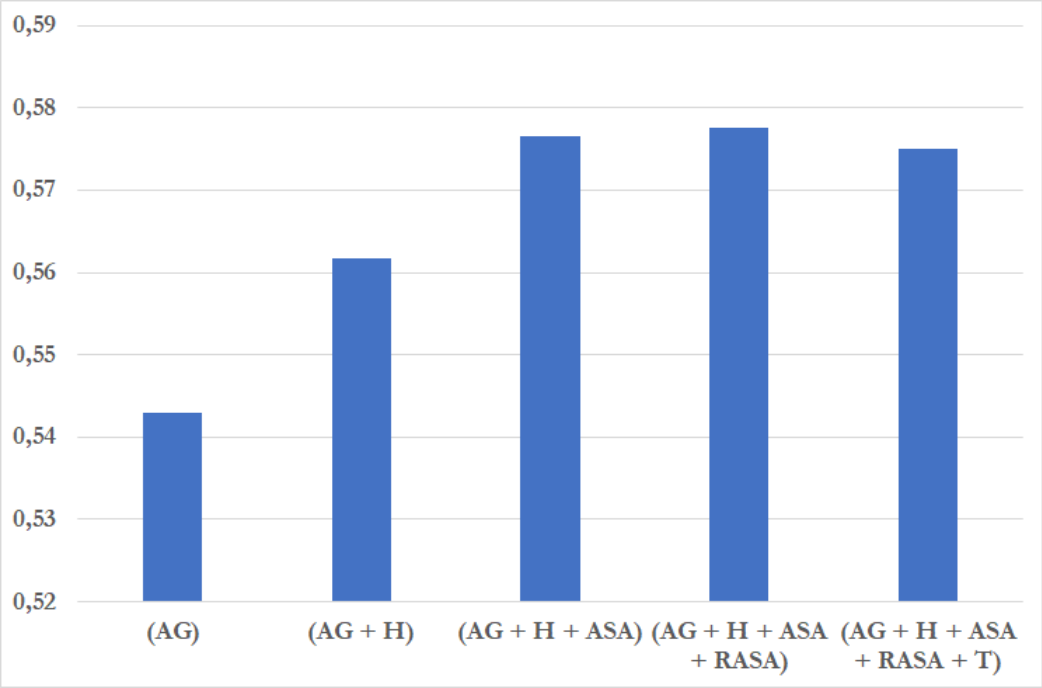


Figure 5.1: Representation of the best Dice score in the validation set of the models trained with different combinations of data.

0.5304. Comparing this model to the previous one (AG + H), the recall values are very similar, which means that both models identified the same ratio of actual interface atoms (67%). However, the precision value is 3% higher in this model. This happens because this model is performing better distinctions between interface and non-interface residues, thus, reducing the number of false positives in its predictions, increasing the precision value.

Furthermore, the next model was trained with the addition of the channel containing the residues relative accessible surface areas (RASA) for each residue, which were calculated by dividing the ASA values for the standard ASA value of the corresponding residue [table x]. Therefore, the U-Net model trained with AG + H + ASA + RASA obtained a validation dice score of 0.5776, a recall of 0.6897, and a precision with the value of 0.5187. Indeed, comparing with the model trained with the combination AG + H + ASA, the validation dice values are very similar. This happened because RASA is obtained from the ASA values and, therefore, they may maintain the same ratios between different types of residues as the ASA values. This means that both ASA and RASA represent how exposed to a solvent are the residues. By having the same kind of representation, these models produced a very similar dice score.

The fifth model used the combination AG + H + ASA + RASA + T, in total eight channels. The addition of the channel that represented the tendency of residues to be located inside the protein structure (T) did not improve the scores of the previous models. It achieved a validation dice score of 0.5751, a recall of 0.6726, and a precision value of 0.5265. Again, the data represented in T was obtained using the ASA value, which was essential to determine if a residue was located on the surface or not. Similar to the feature RASA, this data did not help the model to perform better than the U-Net trained with the combination of channels AG + H + ASA.

Finally, comparing all the trained models (Figure 5.1), the models trained with the combinations of channels AG + H + ASA, AG + H + ASA + RASA, and AG + H + ASA + RASA + T had the best validation dice scores. Since the contexts of the channels representing RASA and T were very similar, and both of them were calculated using the ASA values, adding them to the model did not bring any large improvement. Therefore, the U-Net model that was tested in Stage 2 was trained with the channel combination of AG + H + ASA.

## 5.2 Stage 2 - Training with different number of filters

The next approach was to find the right number of filters to achieve the best predictions in the validation data in Stage 1. Briefly, filters are the weighted kernels that are applied in the input data to extract features. The kernel weights are updated, optimized and further used to make useful predictions. Indeed, increasing the number of nodes/filters in a convolution layer increases the number of weights and the number of patterns that the network can learn. All experiments were performed using the kernel size previously defined, 3×3×3. The filters in each convolution layer are independent and, hopefully, they converge into different shapes. However, having too many filters may result in having more patterns than the needed, while the training phase takes a longer time. For example, a model that is trained to detect cats in images. If all cat features can be extracted already and more filters are added, then the model may start to capture misleading features, such as the bell of the cat. In this case, the model may consider other animals using bells as a

Table 5.3: Representation of the metrics from the best validation score epoch for each model.

| Models | Training Dice | Validation Dice | Training Recall | Validation Recall | Training Precision | Validation Precision |
|---|---|---|---|---|---|---|
| U-Net-4f | 0.5737 | 0.5659 | 0.6942 | 0.6675 | 0.5112 | 0.5153 |
| U-Net-8f | 0.5830 | 0.5766 | 0.6979 | 0.6719 | 0.5238 | 0.5304 |
| U-Net-16f | 0.5938 | 0.5829 | 0.7018 | 0.7199 | 0.5377 | 0.5105 |
| U-Net-32f | 0.5975 | 0.5885 | 0.6933 | 0.6706 | 0.5410 | 0.5417 |

cat, generating false positives. In this context, if there are too many filters, some atoms may be considered interface when they are not, increasing the number of false positives and reducing the dice value in the validation set. Moreover, having more parameters to update makes the network train slower and having a small number of filters may not be enough for the network to learn the necessary contexts.

In this stage, the models U-Net were trained with the combination of channels AG + H + ASA. Moreover, the models were trained with a different number of filters in the convolution layers of the convolution blocks. The number of filters doubled in each convolution layer, which means that if the first layer had four filters, the next one had eight filers, the following one had 16 filters, then 32 filters and finally 64 filters. In the upsampling side of the CNN, the first upsampling layer contained 32 filters in the convolution layer, and the number of filters halved in each convolution layers of the following upsampling blocks. The nomenclature of the models was made based on the number o filters of the first convolution layer to be simpler to identify. Indeed, the model with eight filters in the first layer was already trained in the first stage. Moreover, new models were trained with the same data but with four filters, 16 filters and 32 filters in the first convolution layer.

In the Table 5.3 we can see that the model U-Net-4f-AG+H+ASA, with four filters in the convolution layer and with 47,953 parameters to update (weights) in total, had a validation dice score of 0.5659, a recall of 0.6675 and a precision of 0.5153. The best model of Stage 1, the U-Net-8f-AG+H+ASA, with eight filters in the first convolution layer and with 190,017 weights, had a validation dice score of 0.5766, a recall of 0.6719 and a precision of 0.5304. Afterwards, the U-Net-16f, with 16 filters in the first convolution layer and a total number o weights of 756,481, achieved a validation dice score of 0.5829, a recall of 0.7199 and a precision of 0.5105. Finally, the model U-Net-32f, with 32 filters in the first convolution layer and a total of 2,715,849 weights, obtained a validation dice score of 0.5885, with a recall of 0.6733 and a precision value of 0.5417. The model U-Net-32f-AG+H+ASA achieved the best validation score (Figure 5.2). Indeed, adding more filters improved the performance of the CNN.

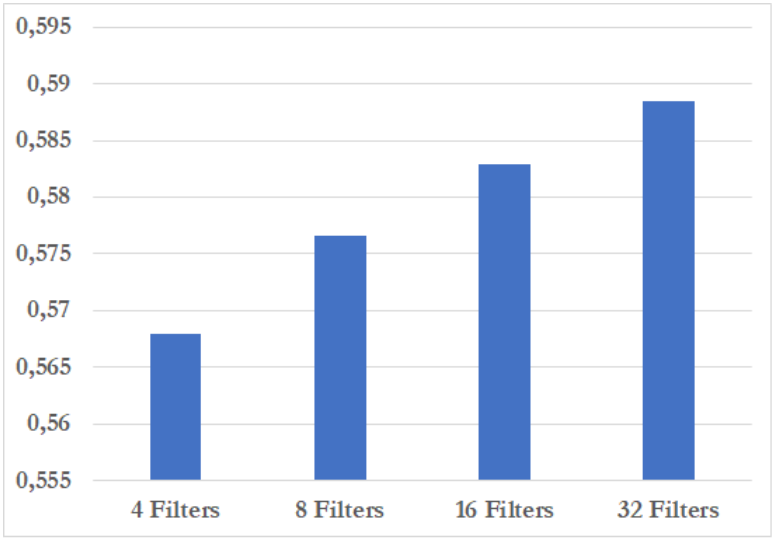


Figure 5.2: Representation of the best Dice scores in the validation set of each model trained with the combination of channels AG + H + ASA, and different number of filters in the first convolution layer.

## 5.3 Stage 3 - Evaluation

In this section are presented and discussed the prediction results of the model with the best training results in Stage 2, the U-Net-32f trained with the channels combination AG + H + ASA. The prediction phase consists in submitting the test set to the model, containing protein chains that the network has never processed. Therefore, this is considered the most crucial phase of the entire Project because it defines how well the model operates with new data. Moreover, the used metrics that evaluated the best model in the prediction phase were based on the confusion matrix. This matrix is a table displaying the number o positives and negatives of both prediction and ground-truth. These values allow a more detailed analysis of the performance of the algorithm. It is important to notice that the predictions have a large number of true negatives because of the tensor values which do not correspond to any atom. Therefore, those values were not counted; that is, only voxels that correspond to the position of an atom were evaluated.

Additionally, several probability thresholds were tested for two reasons. The first one is to find the threshold that best identifies the true positives and false positives. The second one is to be able to build the AUC ROC curve, which evaluates the model capability to separate the true positives and false positives, that is, the overall performance of the model. Moreover, this curve is built using the values true positive rate (or recall) and false positive rate obtained from testing the model with different probability thresholds.

The model was tested with the thresholds from the value zero to the value one, with all the different activations from the obtained predictions, as thresholds. The recall and the false positive rate for each test were used to draw the AUC ROC curve in the Figure 5.2. The obtained AUC has a value of approximately 75,21%. The following results are

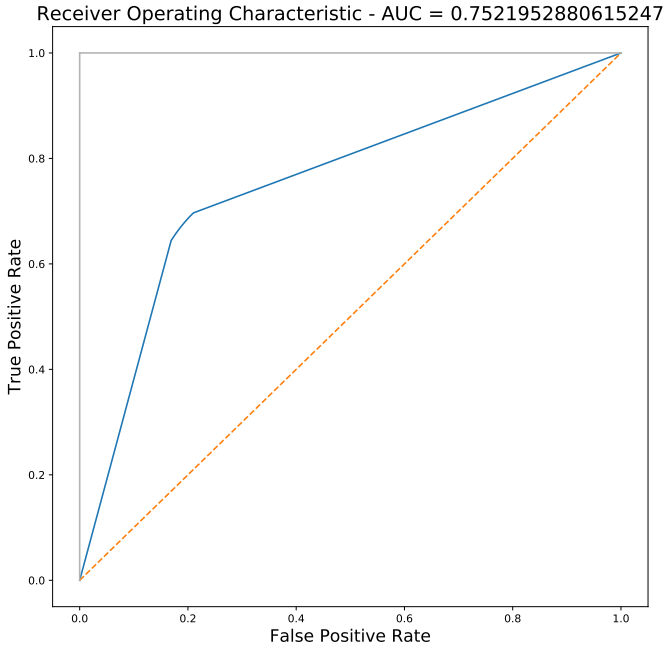


Figure 5.3: AUC ROC curve of the U-Net-32f trained with the channels combination AG + H + ASA. The obtained AUC is approximately 0.7521

Table 5.4: Evaluation report of the U-Net-32f trained with the combination of channels AG + H + ASA. Positives and Negatives correspond to the ground-truth; P-Positives and P-Negatives correspond to the positives and negatives of the predictions; FDR represents the False Discovery Rate; TP stands for True Positives; FP corresponds to False Positives; and FPR stands for False Positive Rate

| U-Net-32f (AG+H+ASA) | | | | | |
|---|---|---|---|---|---|
| Positives | Negatives | P-Positives | P-Negatives | Accuracy | FDR |
| 2653175 | 7903107 | 3237395 | 7318887 | 0.7778 | 0.4523 |
| TP | FP | Dice | Recall | Precision | FPR |
| 1772809 | 1464586 | 0.6019 | 0.6681 | 0.5476 | 0.1853 |

the analysis of the model tested with the 0,90 threshold.

The test set contains 17 184 slices of chains, which correspond to 2 864 chains. These slices have in total 2 653 175 interface atoms and 7 903 107 non-interface atoms. Indeed, the model considered 3 237 395 atoms as interface atoms and 7 318 887 atoms as non-interface atoms. Moreover, 54,76% (precision) of the positive predictions are true positives and 66,81% (recall) of the real interface atoms were identified as positive by the model. Therefore, the dice score of these predictions is of 60,19%, which is slightly higher than the training and validation scores, 59,75% and 58,85%, respectively. By having similar dice scores in all sets (train set, validation set and test set), it is possible to conclude that the model could generalize its predictions for different protein chains, especially

Table 5.5: Table representing evaluation report for each amino acid group.

| U-Net-32f (AG + H + ASA) | | | | | | |
|---|---|---|---|---|---|---|
| Group | TP | FP | Interface | Dice | Recall | Precision |
| Polar | 484808 | 421574 | 738402 | 0.5895 | 0.6565 | 0.5348 |
| Non-Polar | 768730 | 535868 | 1061610 | 0.6497 | 0.7241 | 0.5892 |
| Basic | 330584 | 311909 | 509838 | 0.5737 | 0.6484 | 0.5145 |
| Acidic | 188687 | 195235 | 343325 | 0.5189 | 0.5495 | 0.4914 |

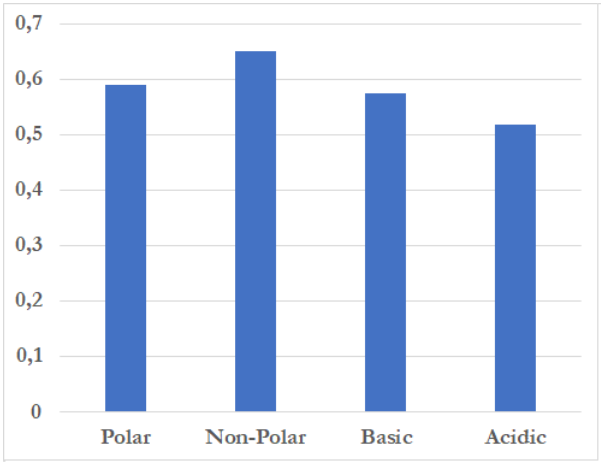


Figure 5.4: Graph representing the Dice values for each amino acid group.

when the redundancy of the protein complexes' sequences were minimized.

Moreover, the same evaluation was made regarding the four amino acid groups (Polar, Non-Polar, Basic and Acidic) to understand how the model performed in each group (Table 5.5). The Polar group had a Dice value of 58,95%; the Non-Polar group obtained a Dice score of 64,97%; the Basic group present a Dice score of 57,37%, and the Acidic group got a Dice value of 51,89%. Indeed, the Non-Polar group obtained the best score Figure of all four groups because this group represent the hydrophobic amino acids, whose presence in the chains interfaces is abundant. The Acidic group (amino acids with a positive charge) had the least Dice score, which proves that this group has the lowest contact rate of the four groups. The Basic and Polar groups had Dice scores in between the other two groups; however, the Polar group showed a higher contact rate in the prediction.

The 17 184 predictions were categorized by how good they were. These categories are described below:

- Perfect prediction (**A1**) - Predictions that corresponded perfectly to their labels, where there is a number of TPs larger than zero and equal to the number of positive atoms in the label. Moreover, the number of FPs is equal to zero. This category contains 154 predictions which is a minimal number. However, the average number of atoms in these chain slices is 87. Therefore, these structures are very small when compared with the structures in the other categories and, for that reason, all the

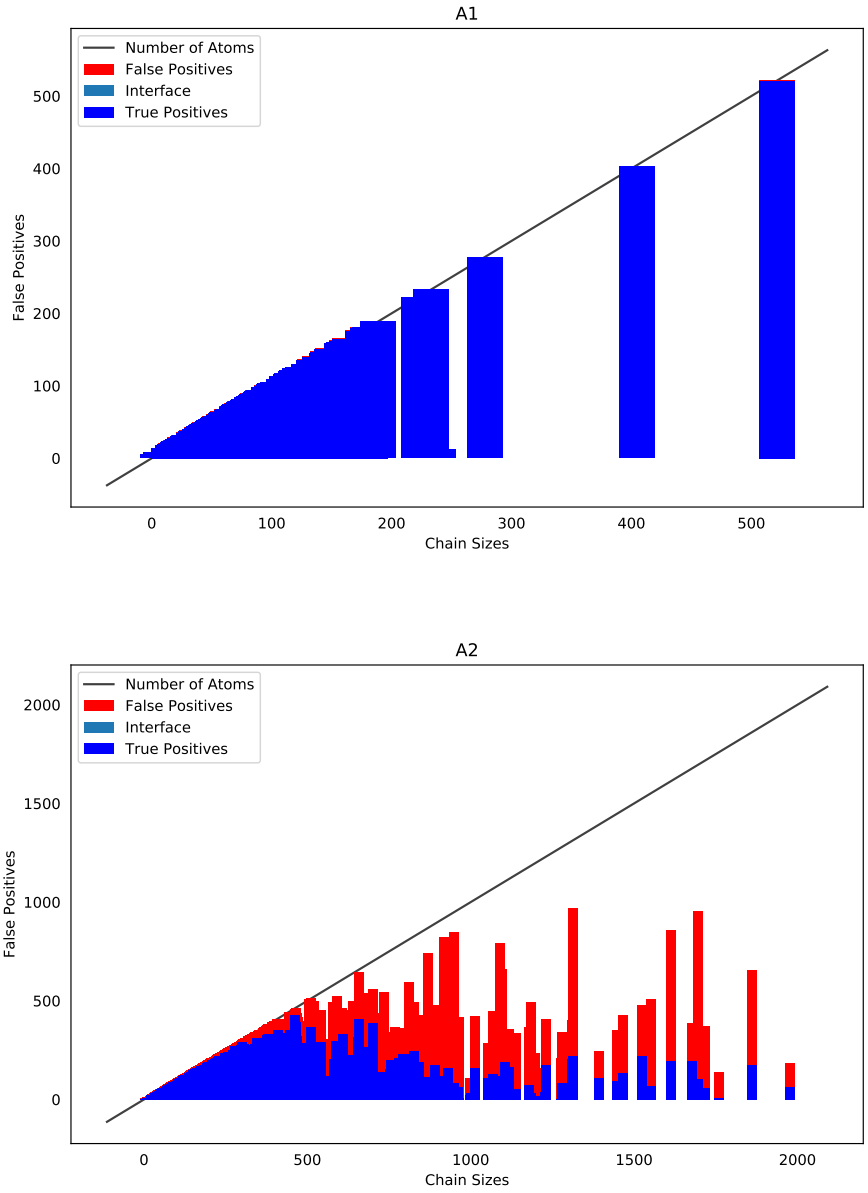


Figure 5.5: Distributions of true positives and false positives regarding the number of atoms of the protein interfaces predictions for the categories A1 and A2.

atoms of the structures residues were considered interacting atoms and the model identified them easily. In the top graph of the Figure 5.5, it is possible to see that the true positives are equal to the number of atoms in most of the predictions.

- Predicted more than true interface (**A2**) - Predictions whose interface was wholly correct but the model considered more atoms as interface atoms. That is, the number of TPs is greater than zero and equal to the number of positive atoms in the label. However, the number of FPs is greater than zero. This category contains 1873 predictions, almost 11% of the total number of predictions. In the bottom graph

Table 5.6: Table representing the number of predictions (Nr Predictions), average number of atoms (Avg Nr Atoms), average protein interface atoms (Avg PI Atoms), average number of true positives (Avg TP) and average number of false positives; for each prediction category.

| Category | Nr Predictions | Avg Nr Atoms | Avg PI Atoms | Avg TP | Avg FP |
|---|---|---|---|---|---|
| A1 | 154 | 87 | 84 | 84 | 0 |
| A2 | 1873 | 230 | 85 | 85 | 72 |
| B1 | 171 | 409 | 0 | 0 | 0 |
| B2 | 1069 | 415 | 0 | 0 | 89 |
| C1 | 64 | 255 | 122 | 92 | 0 |
| C2 | 8683 | 686 | 209 | 165 | 373 |
| D1 | 355 | 546 | 140 | 22 | 0 |
| D2 | 3197 | 768 | 161 | 47 | 82 |
| N1 | 518 | 546 | 69 | 0 | 0 |
| N2 | 1083 | 655 | 51 | 0 | 77 |

of the Figure 5.5 is shown that for small chains, almost all of the atoms were true interface atoms, and therefore, the number of true positives was close to the total number of atoms. The model identified the remaining atoms as false positives to complete the whole proteins. However, for larger chains, the number of interface atoms were smaller, and the number o false positives were higher than for smaller chains. Indeed, these false positives may correspond to atoms of residues that could interact with other chains. Indeed, adding this number to the number of predictions in the category A1, 2027 predictions had the true interface wholly identified.

- Perfect predictions without an interface (**B1**) - Predictions that did not contain interface atoms and whose labels also did not contain any interface atoms. That is the number of TPs equal to zero and FPs equal to zero. There are 171 predictions in this category which were predictions made on slices of chains without interface atoms. Therefore, the model correctly identified all atoms of these structures as atoms from non-interface residues.

- No interface but predicted false interface (**B2**) - Predictions that contain interface atoms but whose label did not contain any interface atoms. That is a number of TPs equal to zero and FPs greater than zero. This category contains 1069 predictions. Indeed, these are predictions which represent false interfaces when compared to the ground-truth. The Figure 5.6 shows that for smaller chains, the model identified almost the whole proteins as false positives. For larger proteins, it identified false positives that might correspond to atoms of residues that can interact with other chains.

- Predicted half or more interface without false positives (**C1**) - Predictions that contained more than half of the correct interface atoms without false positives. The number of TPs is equal to or greater than half of the true number of interface atoms,

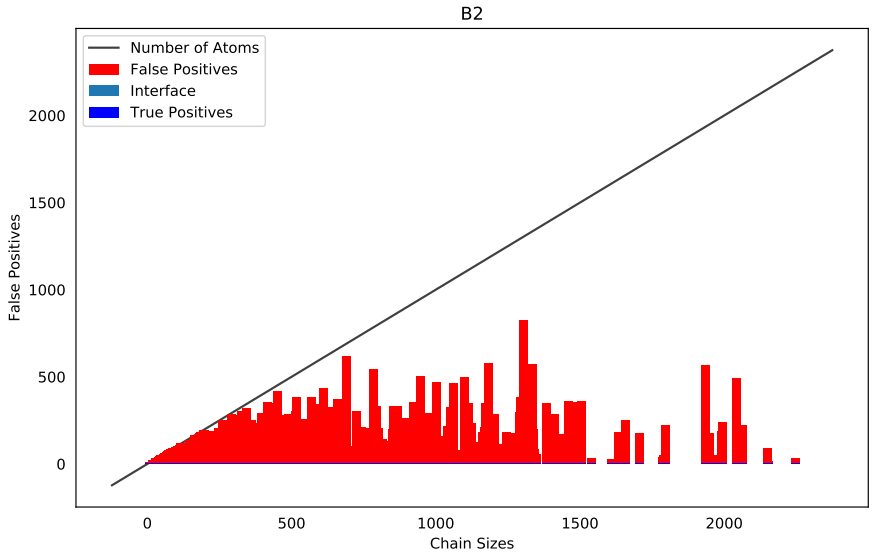


Figure 5.6: Distributions of true positives and false positives regarding the number of atoms of the protein interfaces predictions for the category B2.

and the number of FPs is equal to zero. This category has the lowest number of predictions, 81, and they are mostly small chains. Additionally, the average number of atoms of the slices in this category is 255, which corresponds to small structures comparing with the other categories.

- Predicted half or more interface with false positives (**C2**) - Predictions that contained more than half of the correct interface atoms and contained false positives. The number of TPs is equal to or greater than half of the true number of interface atoms, and the number of FPs is greater than zero. This category contains the largest number of predictions: 8 683. Indeed, approximately 50% of the predictions have correctly identified 50% or more of their true interface. The presence of false positives may indicate that there are more possibilities of residues interactions with other chains. In the Figure 5.7 is shown that small chains had almost all atoms considered as interface atoms, and the additional atoms were considered as false positives by the model. For larger proteins, the model had a more expected behavior because the number of true positives is close to the number of protein interface atoms. However, the average number of false positives is still too large, in Table 5.6 we can see that it doubles the average number of true positives. Indeed, identifying the exact set of interface atoms without false positives is a challenging task using only these residues characteristics because there is not one specific feature that identifies interface residues.

- Predicted less than half interface without false positives (**D1**) - Predictions that contained less than half of the correct interface atoms without false positives. The

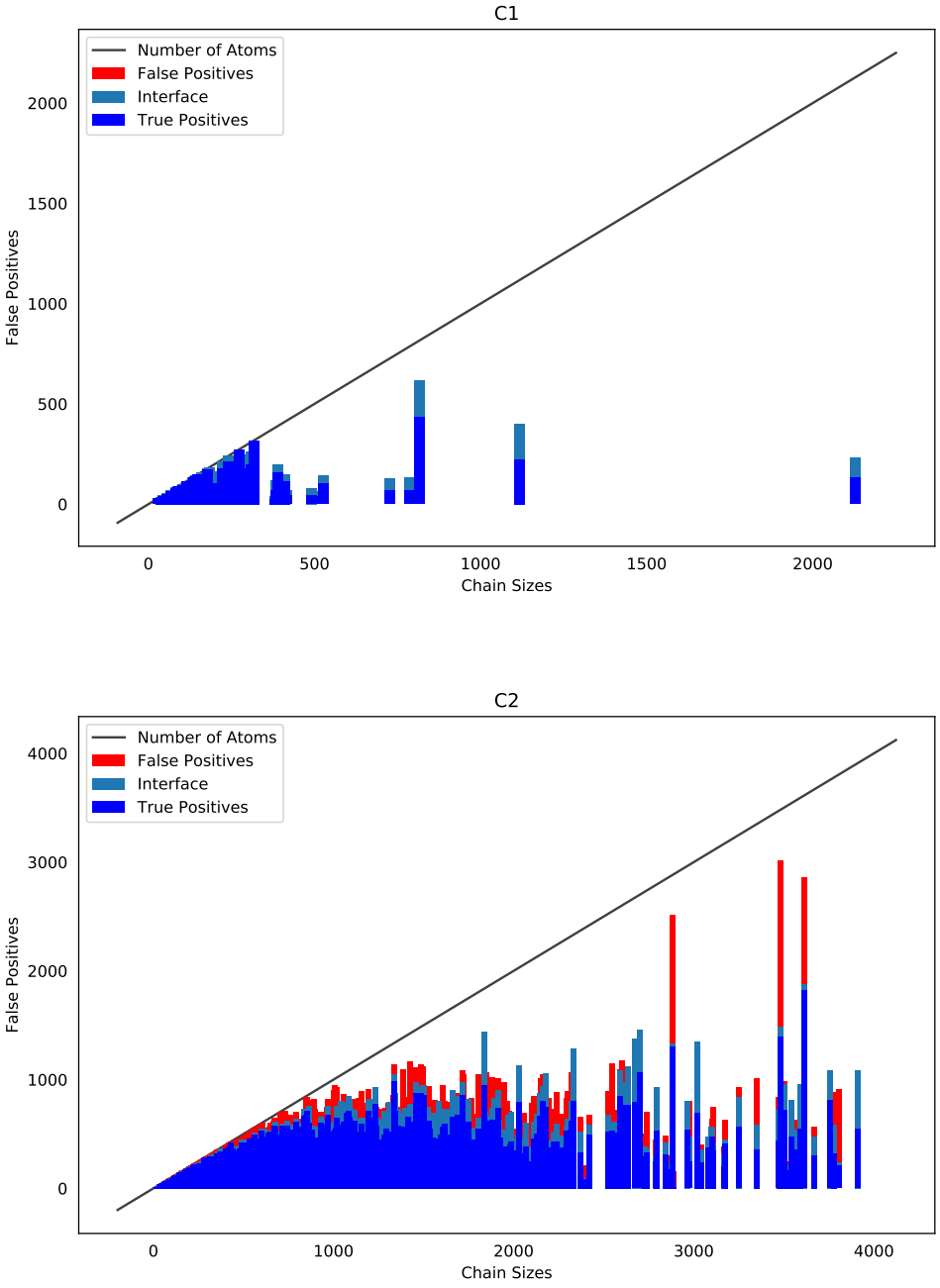


Figure 5.7: Distributions of true positives and false positives regarding the number of atoms of the protein interfaces predictions for the categories C1 and C2.

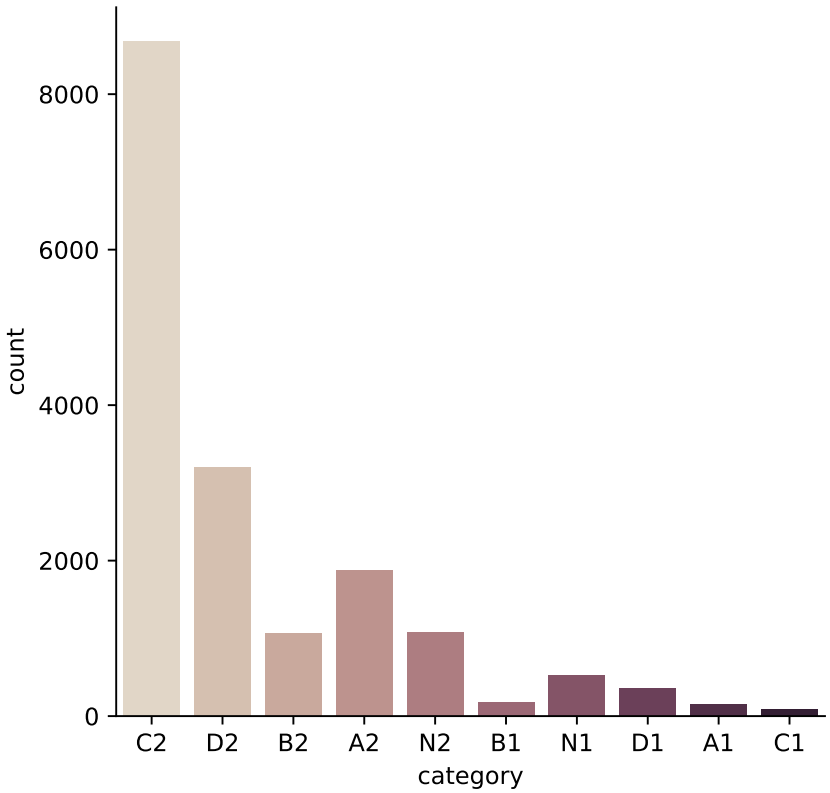


Figure 5.8: Graph representing the number of slices of each prediction category.

number of TPs is smaller than half of the true number of interface atoms, and the number of FPs is equal to zero. This category contains 355 predictions. Figure 5.9 shows that the model correctly identified a small number of interface atoms in these predictions. In average, these chain slices had 140 interface atoms and only identified an average of 22 per prediction (Table 5.6).

- Predicted less than half interface with false positives (**D2**) - Predictions that contained less than half of the correct interface atoms and contained false positives. The number of TPs is smaller than half of the true number of interface atoms, and the number of FPs is greater than zero. This category contains 3 197 elements. The behavior of the model regarding this category is very similar to the category D1, with the addition of false positives. The average number of FP is lesser than the double of the average number of TP.

- Predicted none interface without false positives (**N1**) - Predictions that contained none of the correct interface atoms without false positives. The number of TPs is equal to zero, and the number of FPs is equal to zero. This category contains 518 empty predictions. Indeed, the model failed to identify any interface atom in these predictions.

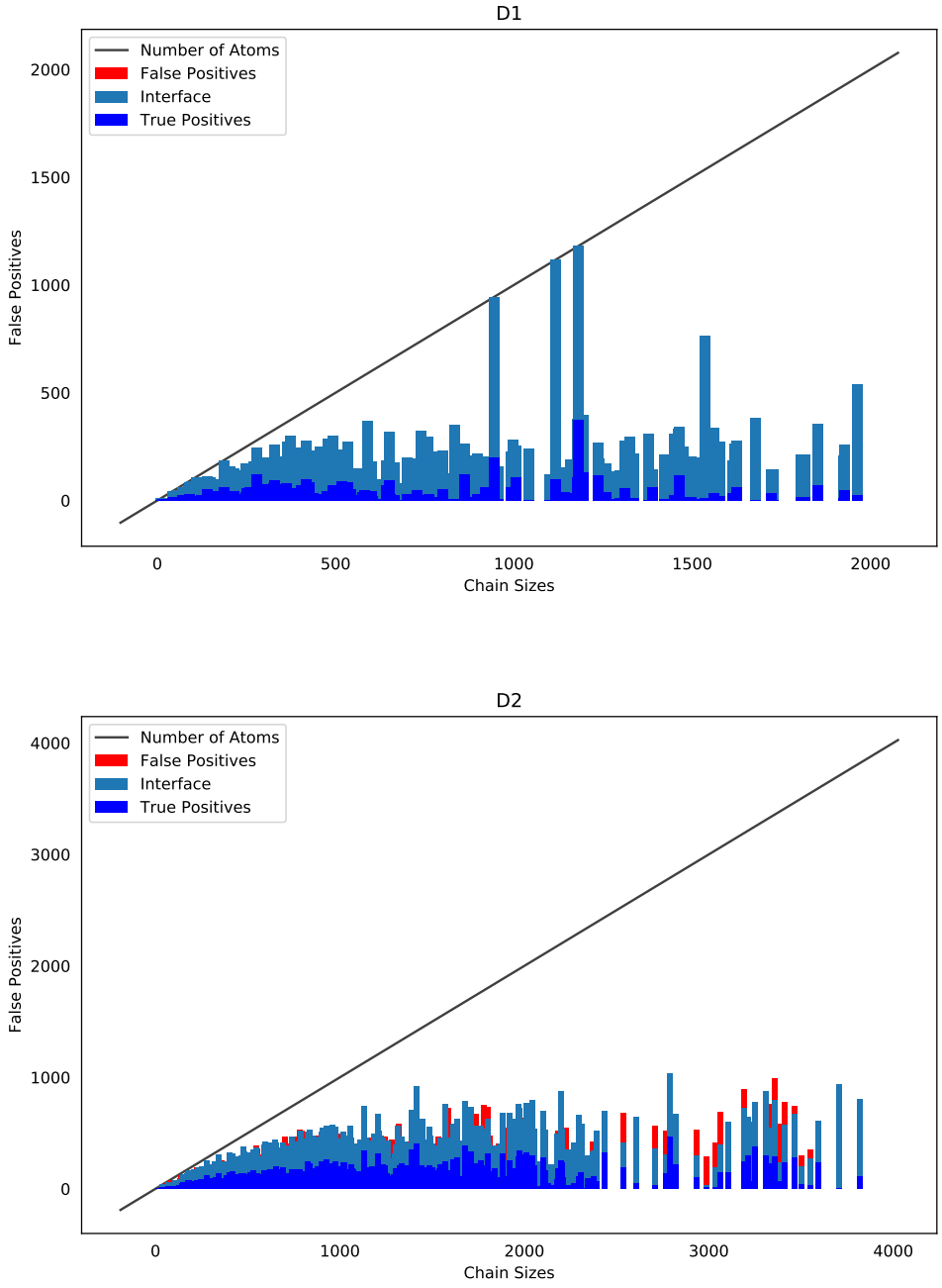


Figure 5.9: Distributions of true positives and false positives regarding the number of atoms of the protein interfaces predictions for the categories D1 and D2.

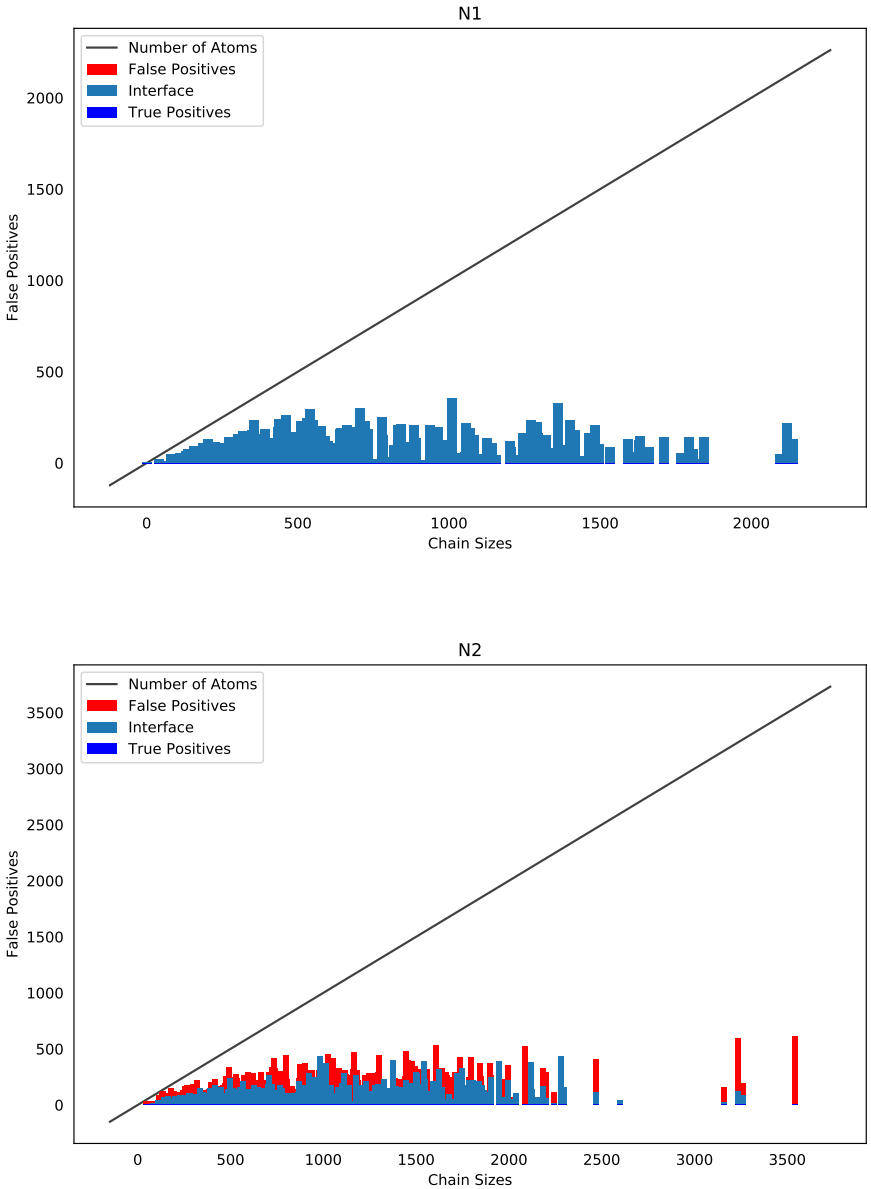


Figure 5.10: Distributions of true positives and false positives regarding the number of atoms of the protein interfaces predictions for the categories N1 and N2.

- Predicted none interface with false positives (**N2**) - Predictions that contained none of the correct interface atoms and contained false positives. The number of TPs is smaller than half of the true number of interface atoms, and the number of FPs is greater than zero. This category contains 1 082 predictions. Indeed, the model failed to identify true interface atoms in these samples. Instead, only false interface atoms were identified.

After analyzing the model's performance and predictions, it is possible to understand that, for the protein information that was used to train the model, there is one crucial factor that influenced the model's behavior, which is the chain size, or in this case, the chain slice size. Many of the small chain slices (with less than 500 atoms) had their interface defined as the whole chain, which is observable in the category A1, where the number of true positives is equal to the number of interface atoms, which is equal to the number of atoms in the slice. Additionally, the small chain slices that had not all the atoms as interface had false positives that, together with the true positives, fulfilled the whole chain slice (A2 and C2). This phenomenon happens in small chains because: 1) the model may be adapting too much to their residues' solvent accessibility area (ASA). Indeed, the majority of small chains have a larger percentage of residues with higher values of ASA, since they are located closer to the surface. Therefore, the model attributes a positive probability for those residues. 2) While calculating the chains' interfaces, small chains that are surrounded by other chains have most of their atoms labelled as interface atoms. Indeed, these two factors together influenced the model to recognize, in most of the small chain slices, all their atoms as interface atoms. However, in the categories D1, D2, N1 and N2, this phenomenon did not happen. Instead, in D1's and D2's small chain slices, even with numbers of interface atoms close to the total number of atoms, had smaller numbers of true positives and false positives. Furthermore, in N1's and N2's, none of the correct interface atoms were recognized by the model. Indeed, the chain/slice size is a significant factor for this model to recognize the correct interface atoms. With better hardware, it is possible to train models with the entire chains (with larger input sizes), and a right approach would be to train models for different sizes of chains. This approach would result in having different models to identify interfaces of different sized chains.

Moreover, there were much more predictions that contained false positives than predictions that contained none (Figure 5.3). This mismatch between the true interface and the predictions happens because the model or the data are not good enough to make a better distinction between true positives and false positives. More detailed optimization of the 3D Convolutional Neural Networks would benefit the solving of this problem, which is approached in many different ways by the scientific community. Parameters like the receptive field of the convolution layers, the learning rate of the optimizer and the application of dropout could improve the performance of these networks when solving protein interface recognition problems.

Table 5.7: Comparison with other state of the art predictors results.

| Model | AUC (%) |
|---|---|
| SPPIDER | 54.1 |
| PSIVER | 57.5 |
| PPiPP | 63.8 |
| PAIRpred | 75.4 |
| This Project | 69.9 |

Additionally, all types of residues can be present in the chains' surfaces and with no specific characteristics that differ interface and non-interface residues. For that reason, it is harder to fit learning models for solving this problem, especially with a non-redundant dataset like the one that is used in this project. Indeed, to find a model that correctly identifies the correct interfaces of unbound protein chains is still a challenge. However, using 3D CNNs can, indeed, contribute to this study.

## 5.4 Stage 4 - Comparison with Similar Predictors

This section presents the results of the final stage of this project, focusing on an independent test set, the DBD. This set was used in many other protein interaction sites and, therefore, it could be used as a benchmark for comparisons between the performance of the developed predictor and others, presented in Section 2.2.2. The measure for comparison is the AUC obtained in the evaluation of the predictors using the test set of the DBD. Moreover, the results of this project were compared with the SPPIDER, PSIVER, PPiPP and PAIRpred, because they had results using unbound structures, like in this project.

The model developed in this project obtained an AUC value of 69.9% (Figure 5.11). PSIVER, PPiPP, SSWRF and DLPred used only sequence-based features, and PAIRpred and our model used both sequence and structural-based features. Moreover, PAIRpred used a larger amount of amino acid features than this project which made their method to have better results than our method, an AUC of 75.4%. However, even with a small number of features (amino acid groups, hydrophobicity and ASA) our model still outperformed SPPIDER, PSIVER and PPiPP methods, with AUC values of 54.1%, 57.5% and 63.8%, respectively (Table 5.7).

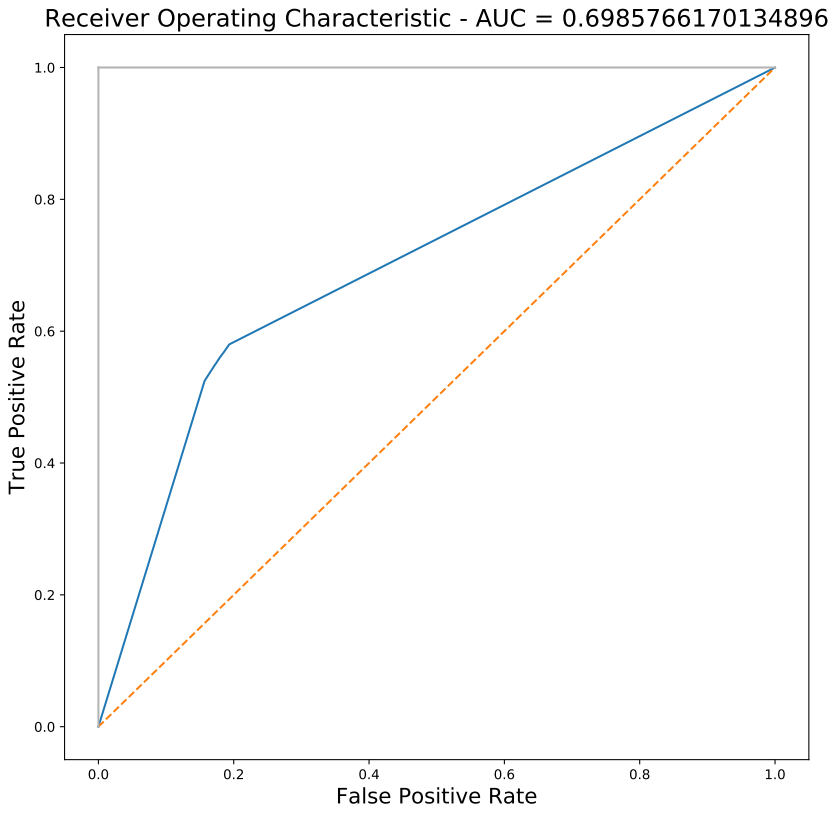


Figure 5.11: ROC Curve and AUC value of the U-Net-32f trained with the combination of channels AG + H + ASA, with the test set of the DBD.

Chapter 6

# Conclusions

In this chapter are described the overall conclusions of this project including the limitations, the most challenging aspects and the possible future work that could complement this implementation.

## 6.1 Conclusions

In conclusion, the main objectives of this project were accomplished. The dataset with the 3D representations of several amino acid characteristics was successfully created. Indeed, the results showed that CNNs achieve better results by adding non-redundant data to the input channels (AG, H and ASA). On the contrary, the channels representing RASA and T, whose values are obtained using ASA, did not improve the performance of the CNN. Moreover, the 3D CNN models were successfully implemented and trained. These models were constructed based on state-of-the-art U-Net architecture. The results proved that these networks could be useful when dealing with Semantic Segmentation problems regarding 3D data. Moreover, by increasing the CNN's number of filters, and consequently increasing the number o weights that could be updated, the results got better.

The model trained with 32 filters in the first convolution layer, with a total of 2,715,849 updated weights whose input channels contained the representations of the amino acid groups, the residues hydrophobicities and the residues accessible surface area (AG + H + ASA), obtained the best dice score in the validation data. Moreover, the dice and AUC scores of these model predictions using the test set from the original dataset, were of 0,5885 and 0.699, respectively. Indeed, this model is not good enough to be used in areas such as drug design or protein engineering because it is not accurate enough. The true positive rate must be higher, and the false positive rate lower. However, this experiment

could still outperform some of the existing protein interface predictors. Additionally, while the studied state-of-the-art used datasets with a small number of samples, in this experiment, there were used all of the non-redundant protein complexes present in the PDB. Indeed, even with a great variety of protein complexes and chains, the model still had a reasonably good score.

The comparison between the developed predictor and the other predictors in Stage 4, using a common test set showed that, although our model was not able to outperform PAIRpred, it still outperformed SPPIDER, PSIVER and PPiPP scores in unbound proteins. Indeed, this method, which approaches protein interface prediction as a computer vision problem, can be used in further experiments with other types of amino acid features.

One limitation of this experiment was the available hardware. CNNs are very complex models, and it can be tough to achieve the best results because they require the optimization of several parameters, and the training phase may take a long time. Therefore, with more GPU devices, the training would be faster, and it would be possible to compare more models with different configurations. Some of the parameters that could be optimized in addition to the number of filters are the batch size and the optimizer learning rate.

Having more GPU devices would be beneficial for another reason: the size of the data and the size of the network. This experiment was limited to the 8GB of the GPU device. For that reason, the chain structures had to be cut into slices to reduce the samples size, which can make an impact in the final results. Furthermore, it would be interesting to make the same experiments as this project but with whole protein chains. Additionally, by increasing the available memory, the CNN could have more filters in its layers, which means more weights to update and the possibility of better results.

Lastly, while developing this thesis, the author faced interesting challenges which promoted the development of practical skills such as learning the Python language, deep learning algorithms and frameworks; and the ability to apply computer science to other areas. The first challenge was to understand the context of this study: to learn about proteins and why their interfaces study was necessary. Secondly, in-depth research about CNNs was made to understand the algorithm mechanics and what experiments could be executed; and finally, the combination of the two fields required extensive research and trial and error to obtain the final dataset which was used to train and test the CNNs.

# Bibliography


[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, G. Brain, I. Osdi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. "TensorFlow : A System for Large-Scale Machine Learning This paper is included in the Proceedings of the TensorFlow : A system for large-scale machine learning." In: (2016).

[2] F. u. A. Afsar Minhas, B. J. Geiss, and A. Ben-Hur. "PAIRpred: Partner-specific prediction of interacting residues from sequence and structure." In: *Proteins: Structure, Function and Bioinformatics* 82.7 (2014), pp. 1142–1155. ISSN: 10970134. DOI: 10.1002/prot.24479.

[3] S. Ahmad and K. Mizuguchi. "Partner-aware prediction of interacting residues in protein-protein complexes from sequence data." In: *PLoS ONE* 6.12 (2011). ISSN: 19326203. DOI: 10.1371/journal.pone.0029104.

[4] E. Alpaydin. *Introduction to Machine Learning Second Edition*. 2nd ed. 2010. ISBN: 9780262012430.

[5] A. Amidi, S. Amidi, D. Vlachakis, V. Megalooikonomou, and N. Paragios. "EnzyNet : enzyme classification using 3D convolutional neural networks on spatial representation." In: 2012 (2018), pp. 1–18. DOI: 10.7717/peerj.4750.

[6] M. Amsom, A. Society, R. St, and I. I. Lipscomb. "The Protein D a t a Bank : A Computer-based Archival File for Macromolecular Structures The Protein Data Bank is a computer-based archival file for macromolecular ( a ) Scope." In: (1977), pp. 535–542.

[7] T. T. Aumentado-armstrong, B. Istrate, and R. A. Murgita. "Algorithmic approaches to protein-protein interaction site prediction." In: (2015), pp. 1–21. DOI: 10.1186/s13015-015-0033-9.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495. ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2644615. arXiv: 1511.00561.

[9] M. J. Betts and R. B. Russell. "Amino Acid Properties and Consequences of Substitutions." In: 4 (2003).

[10] K. M. Biswas, D. R. Devido, and J. G. Dorsey. "E valuation of methods for measuring amino acid hydrophobicities and interactions." In: 1000 (2003), pp. 637–655.

[11] H. Chen and H.-x. Zhou. "Prediction of Interface Residues in Protein – Protein Complexes by a Consensus Neural Network Method : Test." In: 35 (2005), pp. 21–35. DOI: 10.1002/prot.20514.

[12] P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, and F. Kauff. "Biopython : freely available Python tools for computational molecular biology and bioinformatics." In: 25.11 (2009), pp. 1422–1423. DOI: 10.1093/bioinformatics/btp163.

[13] S. De, O Krishnadev, N Srinivasan, and N Rekha. "Interaction preferences across protein-protein interfaces of obligatory and non-obligatory components are different." In: 16 (2005), pp. 1–16. DOI: 10.1186/1472-6807-5-15.

[14] P. Dixon. "W4 Dixon." In: (2017), pp. 55–83.

[15] P. Fariselli, F. Pazos, A. Valencia, and R. Casadio. "Prediction of protein – protein interaction sites in heterocomplexes with neural networks." In: 1361 (2002), pp. 1356–1361.

[16] F. Glaser, D. M. Steinberg, I. A. Vakser, N. Ben-tal, and I. E-mail. "Residue Frequencies and Pairing Preferences at Protein – Protein Interfaces protein – protein interfaces of known high-resolu- residue – residue contact preferences . The residue statistical strength of the data set . Differences be- tween amino acid distributions were observed for interfaces with buried surface area of less than 1 , 000 Å 2 versus interfaces with area of more than abundant in small interfaces . The largest residue –." In: 102.August 2000 (2001), pp. 89–102.

[17] N. Haven. "The Interpretation of Protein Structures : Estimation of Static Accessibility." In: (1971).

[18] F. J. Burkowski. "Structural Bioinformatics: An Algorithmic Approach." In: (2019).

[19] S. Kawashima, H. Ogata, and M. Kanehisa. "AAindex : Amino Acid Index Database." In: 27.1 (1999), pp. 368–369.

[20] J. Kleesiek, G. Urban, A. Hubert, D. Schwarz, M. Bendszus, and A. Biller. "SC Skull Stripping." In: *NeuroImage* (2016). ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2016.01.024. URL: http://dx.doi.org/10.1016/j.neuroimage.2016.01.024.

[21] A. Krizhevsky, I. Sutskever, and G. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Neural Information Processing Systems* 25 (2012). DOI: 10.1145/3065386.

[22] B. A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: (2012).

[23] M. C. Lawrence and P. M. Colman. "Shape Complementarity at Protein/Protein Interfaces." In: *Journal of Molecular Biology* 234.4 (1993), pp. 946–950. ISSN: 0022-2836. DOI: https://doi.org/10.1006/jmbi.1993.1648. URL: http://www.sciencedirect.com/science/article/pii/S0022283683716487.

[24] A. Lehninger, D. L. Nelson, and M. M. Cox. *Lehninger Principles of Biochemistry*. Fifth Edition. W. H. Freeman, 2008. URL: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20{\%}255C{\&}path=ASIN/1429224169.

[25] J. Li and Q. Liu. "' Double water exclusion ': a hypothesis refining the O-ring theory." In: 25.6 (2009), pp. 743–750. DOI: 10.1093/bioinformatics/btp058.

[26] J. Long, E. Shelhamer, and T. Darrell. "Fully Convolutional Networks for Semantic Segmentation." In: ().

[27] D. Maturana and S. Scherer. "VoxNet : A 3D Convolutional Neural Network for Real-Time Object Recognition." In: (2015), pp. 922–928.

[28] A. J. McCoy, V Chandana Epa, and P. M. Colman. "Electrostatic complementarity at protein/protein interfaces 1 1Edited by B. Honig." In: *Journal of Molecular Biology* 268.2 (1997), pp. 570–584. ISSN: 00222836. DOI: 10.1006/jmbi.1997.0987. URL: http://linkinghub.elsevier.com/retrieve/pii/S0022283697909876.

[29] J. Mihel, M. Šikić, S. Tomić, B. Jeren, and K. Vlahoviček. "PSAIA - Protein structure and interaction analyzer." In: *BMC Structural Biology* 8 (2008), pp. 1–11. ISSN: 14726807. DOI: 10.1186/1472-6807-8-21.

[30] S. Mitternacht. "FreeSASA : An open source C library for solvent accessible surface area calculations [ version 1 ; referees : 2 approved ] Referee Status :" in: 189 (2016), pp. 1–10. DOI: 10.12688/f1000research.7931.1.

[31] I. S. Moreira, P. A. Fernandes, and M. J. Ramos. "Hot spots—A review of the protein – protein interface determinant amino-acid residues." In: October 2006 (2007), pp. 803–812. DOI: 10.1002/prot.

[32] Y. Murakami and K. Mizuguchi. "Applying the Naïve Bayes classifier with kernel density estimation to the prediction of protein-protein interaction sites." In: *Bioinformatics* 26.15 (2010), pp. 1841–1848. ISSN: 13674803. DOI: 10.1093/bioinformatics/btq302.

[33] Y. Ofran and B. Rost. "Protein – Protein Interaction Hotspots Carved into Sequences." In: 3.7 (2007). DOI: 10.1371/journal.pcbi.0030119.

[34] G. Pagès, S. Grudinin, G. Pagès, S. Grudinin, D. Using, and G. Pag. "DeepSymmetry : Using 3D convolutional networks for identification of tandem repeats and internal symmetries in protein structures To cite this version : HAL Id : hal-01903624 DeepSymmetry : Using 3D convolutional networks for identification of tandem repeats and internal symmetries in protein structures." In: (2018), pp. 0–24.

[35] F. Pedregosa, R. Weiss, and M. Brucher. "Scikit-learn : Machine Learning in Python." In: 12 (2011), pp. 2825–2830.

[36] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, and T. E. Ferrin. "UCSF Chimera - A visualization system for exploratory research and analysis." In: *Journal of Computational Chemistry* 25.13 (2004), pp. 1605–1612. ISSN: 01928651. DOI: 10.1002/jcc.20084.

[37] L. Rampasek and A. Goldenberg. "TensorFlow : Biology ' s Gateway to Deep Learning ?" In: *Cell Systems* 2.1 (2016), pp. 12–14. ISSN: 2405-4712. DOI: 10.1016/j.cels.2016.01.009. URL: http://dx.doi.org/10.1016/j.cels.2016.01.009.

[38] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9351 (2015), pp. 234–241. ISSN: 16113349. DOI: 10.1007/978-3-319-24574-4_28. arXiv: 1505.04597.

[39] E. G. K. Thiruvathukal and B. J. D. Hunter. "M ATPLOTLIB : A 2D G RAPHICS E NVIRONMENT." In: (2007), pp. 90–95.

[40] W. Torng and R. B. Altman. "Structural bioinformatics High precision protein functional site detection using 3D convolutional neural networks." In: September (2018), pp. 1–10. DOI: 10.1093/bioinformatics/bty813.

[41] W. G. Touw, C. Baakman, J. Black, T. A. H. Beek, E Krieger, P Joosten, and G. Vriend. "A series of PDB-related databanks for everyday needs." In: 43.October 2014 (2015), pp. 364–368. DOI: 10.1093/nar/gku1028.

[42] G. Trinquier and Y.-h. Sanejouand. "Which effective property of amino acids is best preserved by the genetic code ?" In: 11.3 (1998), pp. 153–169.

[43] S. V. D. Walt, S. Africa, and M. S. Feb. "The NumPy array : a structure for efficient numerical computation." In: (2011), pp. 1–8. arXiv: arXiv:1102.1523v1.

[44] Z. S. Wei, K. Han, J. Y. Yang, H. B. Shen, and D. J. Yu. "Protein-protein interaction sites prediction by ensembling SVM and sample-weighted random forests." In: *Neurocomputing* 193 (2016), pp. 201–212. ISSN: 18728286. DOI: 10.1016/j.neucom.2016.02.022. URL: http://dx.doi.org/10.1016/j.neucom.2016.02.022.

[45] B. Zhang, J. Li, L. Quan, Y. Chen, and Q. Lü. "Sequence-based prediction of protein-protein interaction sites by simplified long short-term memory network." In: *Neurocomputing* 357 (2019), pp. 86–100. ISSN: 18728286. DOI: 10.1016/j.neucom.2019.05.013. URL: https://doi.org/10.1016/j.neucom.2019.05.013.