University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Data Science

Linear Interacting Peptides (LIPs) Detection in Protein Sequences Based on Embedding Models

Supervisor
Prof. Damiano Piovesan
University of Padova

Master Candidate
Riccardo Carangelo

Academic Year

2023-2024

To Mary

Abstract

The crucial functional and structural roles of proteins in almost all essential vital functions of living organisms represent a clear proof of their significance across a vast range of research areas, with a particular attention on the biomedical fields. The interaction profile of proteins, given by the sum of the interactions of their basic constituents, i.e., the amino acids, are at the heart of proteins functionality. Protein interactions are the key to understanding biological processes, with profound implications in clinical and biotechnological researches. An exhaustive study of such a problem, however, remains a challenge of enormous complexity, which requests a considerable effort, with decades of study from thousands of researcher groups, employing computing power, advanced laboratories, and a huge integrative and coordinated work, only possible thanks to advances in laboratory science and to infrastructures provided by modern bioinformatics. A problem that is still open and in an active research phase is concerned with the analysis and understanding of so-called intrinsically disordered regions of proteins and, in particular, with the interactions that these regions are able to establish. This paper focuses on special interaction regions, i.e., linear interacting peptides, which are binding protein regions usually found within disordered proteins and that exhibit very specific structural features. These particular interacting regions are presented and discussed in detail, providing a proper exploratory analysis, with the aim of knowing them better and, finally, developing a predictive model for the classification of such peptides. The approach used in this work is cross-disciplinary, looking at the problem, from both a biological and a data-driven points of view, merging concepts derived from structural biochemistry and molecular biology with concepts and tools from the data science and data analysis fields. The ultimate goal is to provide a better understanding of these peculiar regions of protein interaction. During this paper, the variability and complexity of these regions will be shown through data analysis pipelines and specific machine and deep learning tools, highlighting their predictive power, but also addressing their limitations in approaching a problem of great biological complexity.

Contents

Listing of figures

Listing of tables

Listing of acronyms

Molecular Biology and Genetics

DNA	Deoxyribonucleic Acid
RNA	Ribonucleic Acid
mRNA	Messenger RNA
tRNA	Transfer RNA

Proteins biochemistry

PPI	Protein-Protein Interaction
PDI	Protein-DNA Interaction
PLI	Protein-Lipid Interaction
PCI	Protein-Carbohydrate Interaction
RIN	Residue Interaction Network
IDP	Intrinsically Disordered Protein
IDR	Intrinsically Disordered Region
LIP	Linear Interacting Peptide
RSA	Relative Solvent Accessibility

Bioinformatics and Biotechnologies

MSA	Multiple Sequence Alignment
FASTA	FAST-All
pLDDT	predicted Local Distance Difference Test
NMR	Nuclear Magnetic Resonance
HTS	High Throughput Screening
CAID	Critical Assessment of Intrinsic protein Disorder
SIFTS	Structure Integration with Function, Taxonomy and Sequence
RCSB PDB	Research Collaboratory for Structural Bioinformatics Protein Data Bank
STRING	Search Tool for the Retrieval of Interacting Genes/Proteins

RING	Residue Interaction Network Generator
FLIPPER	Fast Linear Interacting Peptides Predictor

Information Technologies

GPU	Graphics Processing Unit
JSON	JavaScript Object Notation
TSV	Tab-Separated Values
FTP	File Transfer Protocol

Data Science

KL divergence	Kullback-Leibler divergence
NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers
PLM	Protein Language Model
EDA	Exploratory Data Analysis
CNN	Convolutional Neural Network
ROC curve	Receiver Operating Characteristic curve
PR curve	Precision-Recall curve
AUC score	Area Under the Curve score
t-SNE	t-Distributed Stochastic Neighbor Embedding
PCA	Principal Component Analysis
BCE	Binary cross-entropy
TPR	True Positive Rate
TNR	False Negative Rate
FPR	False Positive Rate
FNR	False Negative Rate

1 Introduction

This introductory section is about molecular biology and bioinformatics behind this work and is dedicated to provide a basic, but comprehensive, presentation about some important concepts and tools that are directly relevant to the subject matter of this project, with the aim of establishing an essential framework which can offer an appropriate understanding of what has been done. The chapter is divided into four main sections. The first section explains some key structural biochemistry and molecular biology principles, beginning with a concise introduction to proteins and the process to build them through gene expression, which follows the principle of the central dogma of molecular biology. Then, a section is dedicated to protein structures, functions and interactions, with a focus on intrinsically disorder regions and linear interacting peptides. A second section is dedicated to the bioinformatics databases and tools used in the project, with a short explanation of their functioning; in particular, this section introduces the following resources: UniProtKB, PDB, MobiDB, RING, FLIPPER, ProtTrans, and CAID. The third section briefly presents the field of protein features predictions with a particular attention to protein language models and their importance in computational biology. A final fourth section is aimed to present the approach and the techniques used for carrying out this project.

1.1 Proteins

Proteins are macromolecules that are commonly observed in living organisms and which play a crucial role in a wide range of processes carried out by cells, thus being essential to life by serving as structural and functional elements [2]. Examples of these roles are catalysis of metabolic reactions, support and signaling, transportation of other molecules, ability to build specific organic structures, and ability to exercise control and protection from pathogens [2, 4]. This pervasive functional role of proteins in all the functions and structures of life justifies the enormous importance of these macromolecules in many research fields, like healthcare, environmental sustainability, and industry [5, 6, 7, 8].

1.1.1 Composition and structure of proteins

Protein units are continuous single polypeptides composed of a sequence of amino acids, each originating from a specific combination of 20 distinct amino acid types [2, 4]. Each amino acid consists of a central carbon atom (the α-carbon or C_α) that establishes four covalent single bonds with an amino group, a carboxyl group, a hydrogen atom and a specific side chain, which is also called "R group" and is the only variable part that determines the identity of an amino acid among the 20 types mentioned above (Figure 1.1)) [4, 1]. The structural basis of these amino acid polymers is the peptide bond, which is a covalent bond established between the carboxyl group of one amino acid and the amino group of the next amino acid in the peptide chain, formed in a condensation reaction that releases a water molecule (with a thermodynamically unfavorable forward reaction, but with a reverse activation energy that is too high too get back to the reagents) [1]:

$$R^{I}COOH + H_2NR^{II} \rightleftharpoons R^{I}COHNR^{II} + H_2O$$

where, the non-variable portion is directly implied in the formation of the peptide chain and, in general, the series of amino acids connected by the peptide bonds creates the protein's backbone, which consists in the makeup of N-C_α-C repeating units (where N represents the nitrogen atom of the amino group, C_α represents the carbon atom of the α-carbon, and the last C represents the carbon atom of the carboxyl group).

One of the most important aspects of the proteins is their arrangement in space, (i.e., their conformation), which must be in most cases very specific to allow a correct performance of the protein function [2]. The protein conformation is driven by the amino acid composition and sequence order and, in general, the structural organization of proteins can be divided into different levels following a complexity growth (Figure 1.2) [2]:

- the **primary structure** is the linear sequence of amino acids, linked together by their peptide bonds. Considering all possibilities, a polypeptide chain of n amino acids could, theoretically, assemble into 20^n dis-

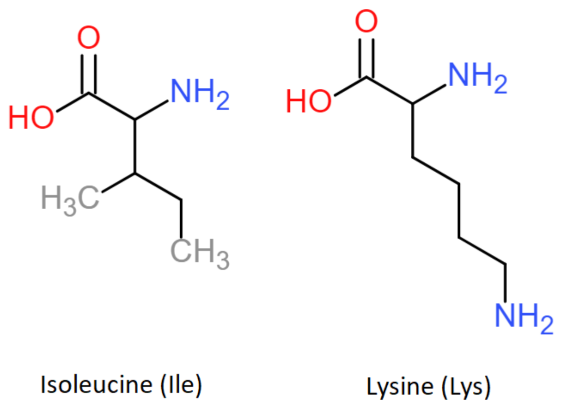


Figure 1.1: Skeletal diagrams of two amino acids, Isoleucine (on the left), a hydrophobic amino acid, and Lysine (on the right), a positive charged amino acid. The structures highlights the terminal groups of the molecules, like oxygen-based groups (in red), nitrogen-based groups (in blue), and carbon-based groups (in gray). It is easy to see the variable side chains, which differ for every amino acid, and the non-variable portion (above in the molecules), in which carboxyl and amino groups are emphasized. Example structures have been taken from *Lehninger principles of biochemistry*, by Nelson *et al.* [1].

tinct formations, if left to chance. Yet, the primary structure of a protein derives by specific inherited genetic information, ensuring that each protein is the result of a precise biological aim. The primary structure forms the backbone of the protein and the order of the amino acids in this very first simple structure is fundamental for driving the spatial organization towards the final protein conformation, inducing the secondary and tertiary structures;

- the **secondary structure** provides a more complex organization, with basic three dimensional regular patterns formed by the hydrogen bonds of elements of the backbone interacting with each other (thanks to a partial negative charge of the oxygen atoms and the partial positive charge of the hydrogen atoms attached to the nitrogen atoms). The main secondary structure motifs that can be observed in proteins are (Figure 1.3):
  - *α-helices*, made by hydrogen bonds between every fourth amino acid;
  - *β-sheets*, made by two or more segments of the polypeptide chain that are positioned side by side (called ”β-strands”) and connected by hydrogen bonds among amino acids of the parallel segments;

- the **tertiary structure** is induced by the side chains of the amino acids, engaging in a variety of chemical interactions with determined roles, like hydrogen bonds, hydrophobic interactions, disulfide bridges, and Van der Waals forces ((Figure 1.3));
- the **quaternary Structure** is given by the assembly of multiple polypeptide chains into a functional pro-

tein complex, forming an arrangement that is essential for carrying out the final function.

As mentioned above, amino acid side chains are unique for each of the 20 amino acids that constitute proteins. However, it is possible to associate these chains with each other, depending on their intrinsic characteristics, giving rise to a taxonomy that allows a grouping of amino acids into classes; in particular, it is possible to distinguish four different classes of amino acids, basing on the charge profile of their side chains [4, 1]:

- **non-polar amino acids**, which tend to avoid water molecules, establishing hydrophobic interactions with each other and compacting into a globular conformation that tends to move away from the watery medium, in the innermost portions of the protein;
- **polar amino acids**, which can be found in active sites or on the surface of the proteins, forming bonds that make secondary and tertiary structures and also participating in catalytic reactions;
- **positively and negatively charged amino acids**, which are able to form ionic bonds, contributing to protein structural stability and catalytic activity.

This class division of amino acids is a fundamental concept for understanding some conformational aggregation behaviors of proteins, where the pattern of amino acids that makes up a protein region determines its charging characteristics and, therefore, its conformational aggregation characteristics [1]. An important example of this concept is given by the presence of hydrophobic globular regions that are rigidly aggregated, due to a rich presence of non-polar amino acids; other examples are the regions which are specifically present on the protein interface with the medium, interacting with water molecules or other molecules, or even regions with greater conformational and interaction freedom, that are less subject to the structural rigidity typical of other regions, thanks to the presence of charged or polar amino acids [2, 1].

1.1.2 From DNA to Proteins: The Central Dogma of Molecular Biology

The making of proteins is a complex process, involving other macromolecules of enormous importance, such as DNA and RNA, which allow the genetic information to be safely stored and then converted into function, as stated by the central dogma [2]. The central dogma of molecular biology is a concept that refers to the flow of the genetic information inside living organisms; the main idea behind this concept can be captured by the statement "DNA makes RNA and RNA makes proteins", sentence that highlights the main characteristics of the concept, i.e., its unidirectionality [2]. In facts, biological information travels in only one direction, from genes to proteins, and cannot be transferred back (this kind of concept admits, of course, exceptions) [2]. Firstly stated by Francis Crick in 1957, the dogma was indeed more a hypothesis than an actual dogma, leaving room for further

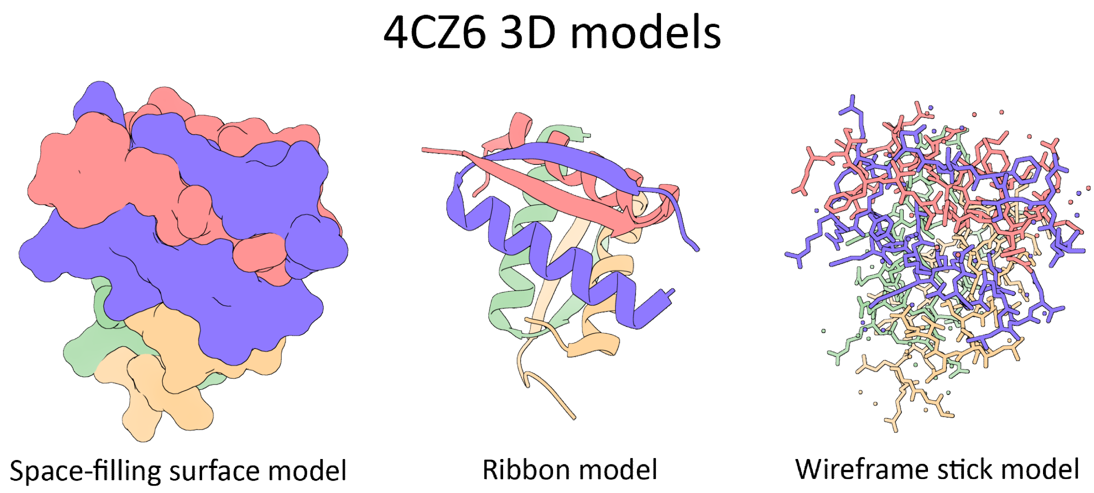


Figure 1.2: Various three dimensional representations of the folded truncated tetramerization domain of the p53 anti-tumor protein, obtained from *Danio rerio* by using biocrystallography and X-ray diffraction techniques. The PDB data file was taken from the Protein Data Bank (PDB, protein ID 4CZ6). All the structures have been thematically colored to highlight the four different peptide chains that interact with each other. On the left, it is possible to observe the space-filling surface representation, which emphasizes the globular structure of the protein and it is also useful to visualize the protein surface. In the middle, it is possible to see the ribbon representation, commonly used to easily visualize the most frequent secondary structures, like, for instance, α-helices (shown as spring-like structures) and β-sheets (depicted as arrows). It is also possible to see random coils (i.e., regions that lacks a specific secondary structural organization, represented using thin strings [2]). This representation is also useful to visualize the tertiary structure folding of each chain and the aggregation of the different folded chain to give the final quaternary structure. On the right, it can be seen the less intuitive wire-frame stick representation of the protein domain, which shows the chemical structure of the protein's underlying molecules, highlighting also proximal atoms and molecules that surround the protein surface (like ligands and other solution molecules). This representation provides a way to directly see the backbone and the side chains of the single amino acids of the protein.

discoveries of slightly deviating situations and exceptional cases [2]. Anyway, even if exceptions are widely known today (important examples are epigenetics, reverse transcription, prions and RNA editing), the central dogma of molecular biology remains an important foundation for representing the genetic information flow in every living organism, since it is in most cases a reliable system for describing the flow of the genetic information [2, 4].

As will be shown below, the processes by which DNA produces RNA and RNA produces proteins are transcription and translation, respectively. In creating new proteins, transcription and translation implement a mechanism of conversion of the genetic information into function that is called ”gene expression” [2, 4]. In addition to transcription and translation, there is the replication, which is essential for the transmission of DNA and is considered as the most important inheritance mechanism in living organisms, since it provides every cell a distinct copy of the DNA molecule, which has a critical role in the storage and preservation of genetic information; through replication two identical versions of DNA are generated from a single parent DNA molecule [2, 4]. In the transcription process, specific portions of the DNA molecules (the genes) can be used to make a messenger RNA (mRNA), which plays a transmission role, since it is a fundamental intermediate that carries genetic information [2, 4]. The RNA ribonucleotide composition is directly determined by the nucleotide composition of the DNA constituting the transcribed gene [2]. In addition, a newly generated mRNA may go through further maturation steps (like splicing, capping, polyadenylation, and so on) [2]. Although transcription can also be a tool for producing special types of RNA (mostly employed in regulatory processes), the main function of this biological process is the creation of messenger RNAs for new proteins generation [2]. Translation is the final step that directly makes the primary sequence of new proteins; here, the genetic information contained in the mature mRNA transcript is used by ribosomes, where transfer RNAs (tRNAs, a special type of RNAs involved in translation) is used as a template for making a new protein [2]. Even in this case, codons determine the specific amino acid composition and order of the protein, i.e., by triples of ribonucleotides contained in the mRNA; in particular, for every three ribonucleotides, a specific amino acid is added to the sequence of the new protein, in a specific process that involves the presence of some intrinsic redundancy, since several triplet combinations can be associated with the positioning of the same amino acid) [2, 4]. After the assembly of the protein, its three-dimensional structure is naturally achieved through a precise folding, guided by the amino acid sequence composition [2, 1]. The common arrangement in space of the mature protein, called native conformation, is indeed fundamental and strictly related to the ability of the protein to perform its natural function [2, 1]. This concept makes the accuracy of the transcription and translation processes a critical step, in which a protein canonical function can be invalidated even with a small error in translating the RNA sequence [2, 1]. While replication, transcription, and translation are essential processes to produce proteins, they are themselves performed by a plethora of extremely complex protein-based mechanisms. Therefore, in a complex synergistic framework, all these three processes are interested by a mutualistic mechanism, which is still a subject of study, debate and refinements [2].

1.1.3 Intrinsically Disordered Proteins

One of the key messages of the previous section concerns the importance of reaching a highly specific conformation during the protein folding process and the extreme fragility that such a process have in impairing the proper performance of the function for which a given protein is intended. Once this concept is understood, an apparent major contradiction lies in the fact that many proteins actually possess regions (which can also be quite significant in terms of size) of intrinsic disorder, which do not negatively affect protein function, but are sometimes even fundamental to the performance of the latter [9, 10]. There are even cases in which proteins totally or partially lack a fixed structure, which can assume a specific conformation only in the presence of specific interaction partners [10, 11]. These particular kind of proteins forms a distinct category and provide a view on protein structural organization and function that challenges the traditional way of associating to each protein a determined and unique conformation that makes the function possible [9]. This particular property of IDPs allows them to be versatile proteins, often adopting fixed three-dimensional conformation upon binding to other molecules, but facilitating at the same time distinct binding requirements [10, 9]. IDPs play several crucial roles, participating in different important functions, mostly related to chromatin regulation, transcription, and cell signaling, while being widely studied in various aspects [12, 13]. As it happens with all functions and structural features of proteins, even the intrinsic disorder feature is encoded in the protein's amino acid chain; it is, indeed, possible to say that, generally, for being an IDR, a protein region must have a low content of bulky hydrophobic amino acid residues, while presenting a high proportion of polar and charged amino acids, which give rise to a low hydrophobicity region with a low probability of collapsing into a globular conformation [10]. In this way the peptidic region is more prone to disorder, by enhancing repulsion (due to the presence of net charges) and interaction with water molecules composing the medium that surrounds the protein [10].

1.1.4 Protein Interactions

Proteins are able to establish several kind of interactions with many different molecules and, in general, proteins can interact with all the other types of main macromolecules that can be found in an organism (i.e., other proteins, lipids, carbohydrates, and DNA) sometimes forming intricate networks of interactions that are essential to regulation and performance of vital functions (Figure 1.3); it is, indeed, possible to observe in proteins a great variability of regions used for interacting with other molecules, with cases of considerable structural and organizational complexity [1]. The main types of protein interactions are performed by electrostatic forces, hydrogen bonding, and hydrophobic effect, making the interaction network of each protein rich of different types of specific interactions, suitable for performing distinct functions [1].

This paragraph presents the main types of protein interactions, with particular descriptive emphasis on protein-protein interactions. Protein-protein interactions (PPIs) are common interactions when forming protein functional complexes and for transmit several kinds of signals in the cell; protein-carbohydrates interactions (PCIs) are usually a central component of the cellular recognition ability, but are also part of some of the most important metabolic pathways; protein-lipid interactions (PLIs) can be crucial for keeping the integrity of the membrane and for signaling tasks; finally, protein-DNA interactions (PDIs) are fundamental in regulating gene expression [2, 1]. Among all these interaction types, PPIs are the most common protein interactions and are considered central for every cellular process, since they are the baseline for assembling protein complexes and for the regulation of all cellular process (Figure 1.4) [1]. Therefore, these interactions are of particular interest in research, even if understanding and characterizing all of them is a very challenging task, due to the natural complexity of the problem [14]. Protein interaction studies in biomedical and biotechnology fields are of paramount importance, as a complete understanding could lead to strong practical applications in several areas of biomedical research and biotechnology, like, for instance, drug design, genetic engineering, and pharmaceutical clinical research [5, 6, 7, 8].

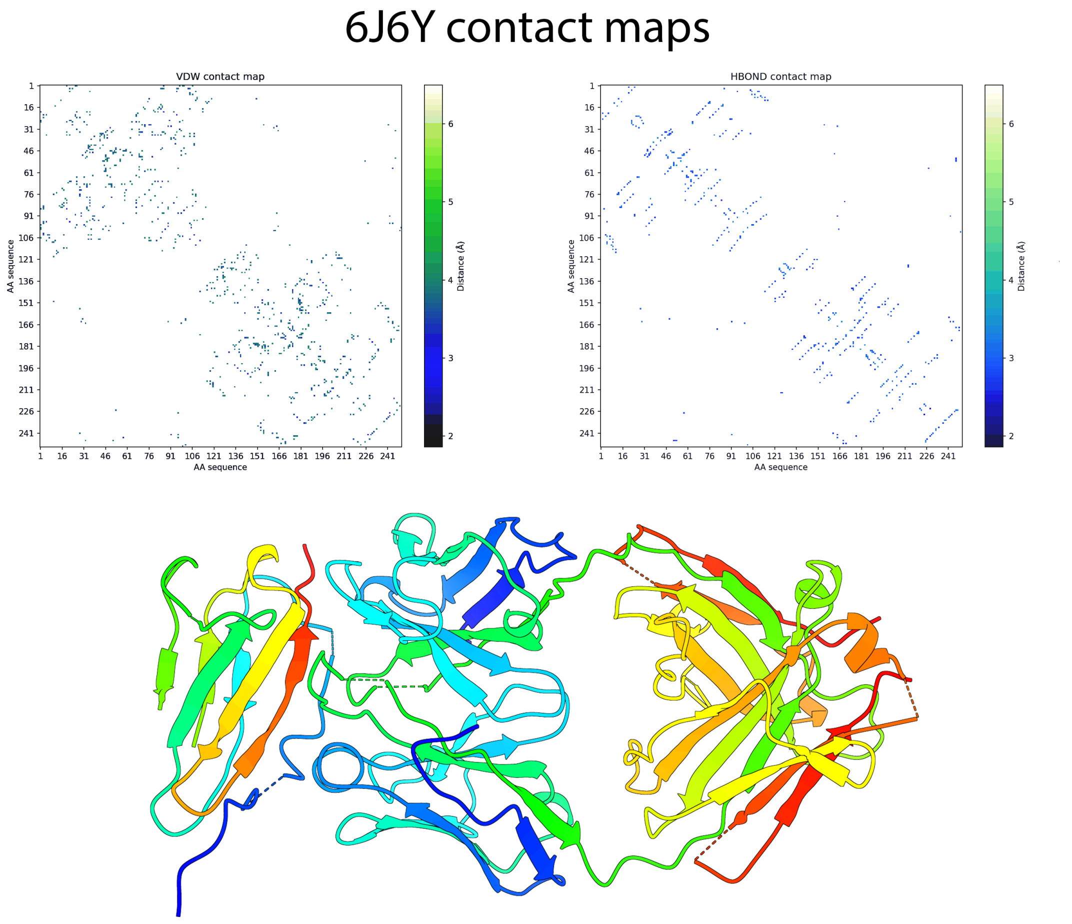


Figure 1.3: Contact maps and three dimensional representation of the folded truncated tetramerization domain of FGFR4 D2 - Fab complex (biological assembly 1), obtained from *Homo sapiens sapiens* by using biocrystallography and X-ray diffraction techniques. The plots refer to a specific state of the protein conformational ensemble. The PDB data file was taken from the Protein Data Bank (PDB, protein ID 6J6Y). The contacts were calculated using the Residue Interaction Network Generator 2.0 (RING 2.0). The plots above present the contact maps showing two very specific kinds of interactions of the protein's amino acids, gradient-colored based on the distance in ångström and showing the amino acid sequences, named "AA sequence", in both x- and y-axis (with a colored dot if two coordinate positions interact). In the top-left corner, it is possible to see the van der Waals forces contact map (*VDW contact map*), which shows the interactions that take an active part in the formation of the tertiary structure. In the top-right corner, it is possible to see the hydrogen bonds contact map (*HBOND contact map*), which shows the interaction that are responsible for the generation of the secondary structures. Below, it is shown the ribbon representation with rainbow coloring of the protein to which the contact maps refer; it is interesting to compare the maps with the protein arrangement in space, in order to appreciate the great amount of contacts that take part in the final protein conformation, which shows a compact three dimensional folds and several β-sheets.

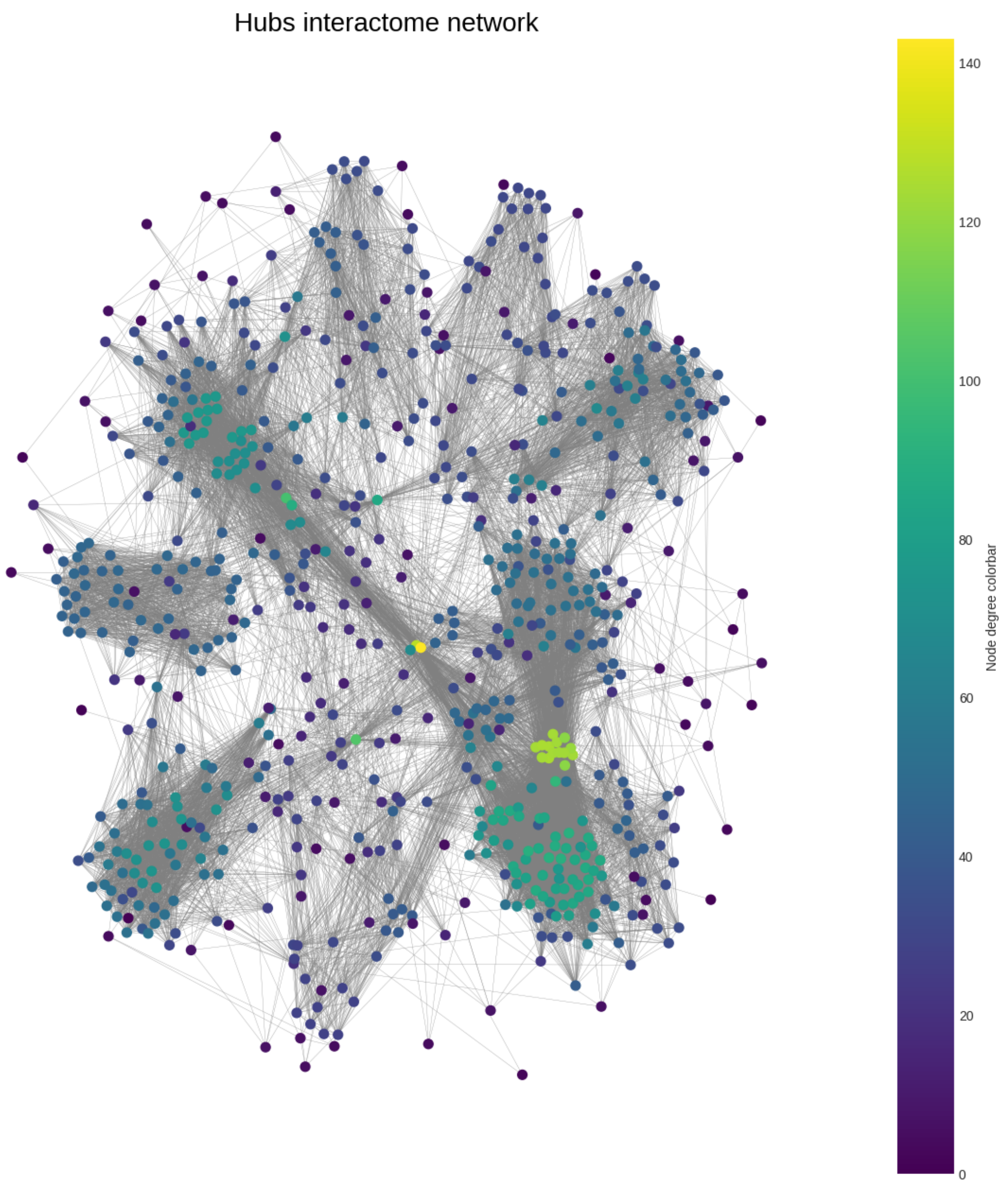


Figure 1.4: Sub-graph composed by the hub nodes (with their immediate neighbors) selected for the protein interactome network of *Homo sapiens sapiens*, taking proteins as nodes and physical interactions between pairs of proteins as edges, while showing the degree of each node protein using a gradient coloration. Data about proteins and their interactions has been selected from the Search Tool for the Retrieval of Interacting Genes/Proteins (STRING) database [3], selecting human proteins and their edges with the highest confidence data (STRING score greater or equal than 0.900). This densely connected and high confidence network shows the most connected proteins, i.e., hub node proteins; it is easy to notice the great level of connectivity in these hub proteins, which is an evidence of their importance within cells (as, for example, master regulators for important processes, such as modifications to the DNA molecule). It is finally possible to appreciate at first glance an organizational topology that shows small cliques with higher levels of local connections, which may be regions that represent regulatory hubs in a scale-free topology context, or large protein complexes.

1.1.5 Linear Interacting Peptides

When considering the complex field of protein interactions in cells, a comprehensive understanding of the subject cannot exclude the crucial role of the short protein motifs that take part in the cellular interaction network [15]. In this context, LIPs are short amino acid sequences usually (but not always) found in IDRs and which shows the ability of acting as binding interfaces, thus playing an essential role in many physiological activities of organisms [16]. LIPs present the following structural characteristics in comparison to non-LIPs residues [16]:

- highly linear structure (linearity refers to the spatial distance between the C_α of the first and last residue of a specific region, divided by the length of such sequence);
- higher inter-chain over intra-chain contacts;
- both higher RSA and ΔRSA;
- equal distribution of secondary structures;
- higher presence of random coil conformations.

From this list of features it is possible to deduce a strongly flexible nature of LIPs that makes them dynamic elements in the context of cellular mechanisms, allowing the protein to interact with several other macromolecules (i.e. peptide regions in the same protein or regions from other proteins or nucleic acids), enabling in many cases the disorder-to-order transition of the IDRs in which they're included [17]. LIPs will be the main object of study of this paper and the project described by it, with a focus on the analysis of their biochemical characteristics, in order to highlight their complexity and the consequent difficulty for any potential task aimed to an effective identification of these regions through the use of predictors.

Figure 1.5 shows an example of a chain containing a LIP region, which interacts with two other chains of a single protein complex. In the figure, it is possible to appreciate a ribbon representation of the LIP chain lying on the surface of the other two chains, wrapping both of them by establishing PPIs.

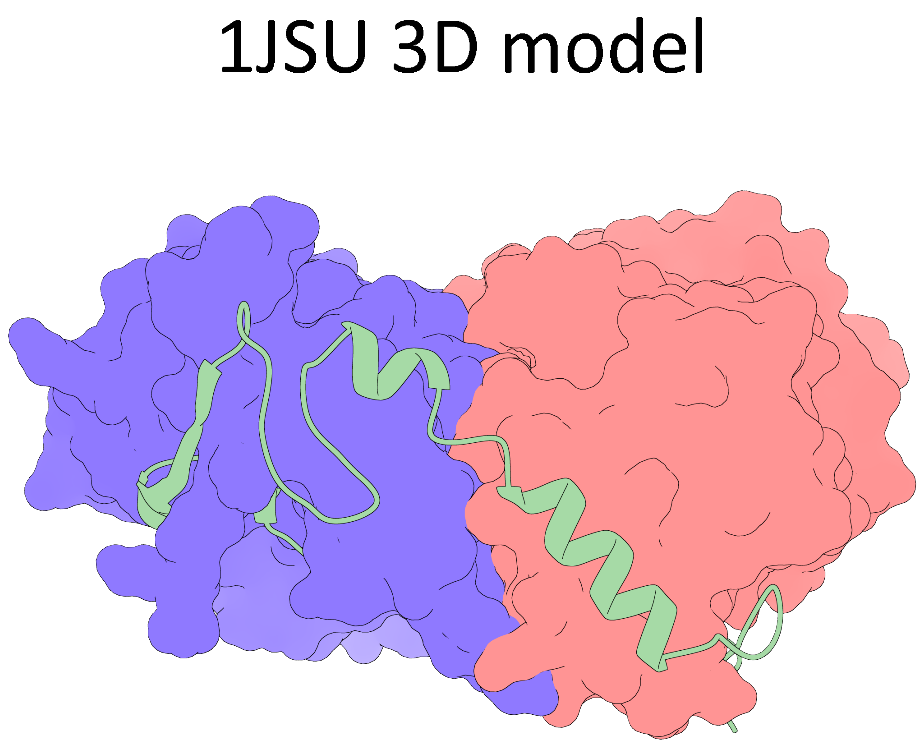


Figure 1.5: Three dimensional representations of the P27 (KIP1)/cyclin A/CDK2 complex, obtained from *Homo sapiens* by using biocrystallography and X-ray diffraction techniques. The PDB data file was taken from the Protein Data Bank (PDB, protein ID 1JSU). All the chains were thematically colored: the LIP chain is colored in green and uses a ribbon representation, while the other two chains were colored in pink (chain A) and in violet (chain B), both represented using a space-filling surface model, in order to make it possible to appreciate the interaction of these chains with the LIP chain, which lies on the surface of the space-filling model.

1.2 Protein Data

In the past decade, bioinformatics has become a fundamental field of modern biological research, providing a powerful analytical tool for an era that is dealing with an enormous amount of biological data that offer the potential for great discoveries [18, 19]. Nowadays, bioinformatics offers the methodologies needed to untangle the complexity of cellular mechanisms, also thanks to its strong collaborative and contributory nature, aspects that are essential for tackling and integrating the large-scale problems addressed by modern science [20, 21]. Of particular interest is the high-throughput approach, i.e., a systematic and fast approach for exploring a large amount of biological data at a large scale, using many computing resources [22]. This approach has become fundamental in several bioinformatics fields that require such kind of analysis, among which high-throughput sequencing (HTS) stands out [22, 23]. Once asserted the great importance of taking into account a bioinformatic approach while facing a modern molecular biology problem, it is possible to introduce some important tools, services, and databases. Some of them have been used for this project, in order to retrieve data and information, while others are simply important to better understand the origin of the data and the problem itself. This chapter will be dedicated to briefly describe such useful bioinformatics tools and databases, which are mostly focused on storage and retrieval of a vast amount of biological data about protein sequences and structures.

1.2.1 UniProtKB Database

The UniProtKB database is the primary component of UniProt, which is composed of two components: UniProtKB/Swiss-Prot and UniProtKB/TrEMBL; while the former is manually annotated and provides detailed and accurate protein annotations, the latter offers automated annotations, presenting material that does not have manual curation and that awaits for annotation [24, 25]. The manual annotation in UniProtKB/Swiss-Prot is performed by expert curators who are devoted to the analysis of the existing literature and of the computational data, in order to ensure data accuracy and to provide high-quality databases for protein research [26]. Within these two big sub-sections, the UniProtKB database stores and manages a huge amount of data, including extensive cross-references and annotations relative to many protein features, in terms of structure, function, and interaction, allowing, in this way, in-depth analyses and efficient querying across multiple databases [25]. This integrative approach is a hallmark of UniProtKB and provides a powerful tool that cross-references to more than 120 external databases, promoting the integration of information from a wide spectrum of biological fields [27].

1.2.2 Protein Data Bank

The Protein Data Bank is a big and continuously updated platform that provides information about the three-dimensional structural characteristics of proteins and nucleic acids, obtained through NMR, X-ray crystallography, and cryo-EM [28, 29]. PDB was created as the first open-access digital resource for biological sciences and it is now a unique global repository of structural data for biological molecules, with a huge archive of detailed information about experiment-based structures [29, 30]. As of today, PDB, with more than 200000 experimental structures and more than 1 million computed structure models [31], supports a wide range of scientific research fields and parallel databases, from basic molecular biology to clinical research and drug design [32, 33].

1.2.3 SIFTS service

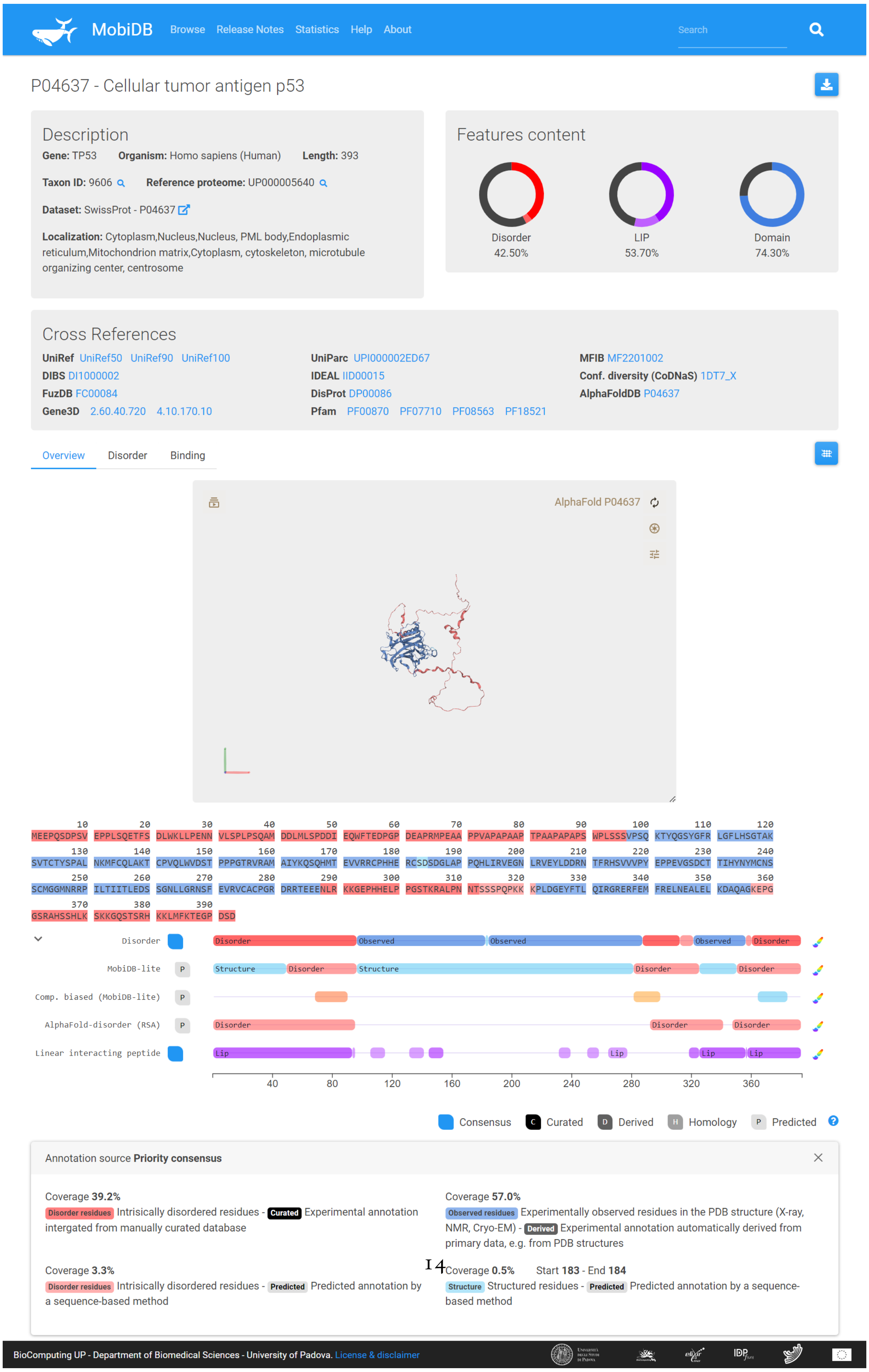
SIFTS is a project involving PDB and UniProt, who collaborated together to make a tool for mapping at the residue-level protein entries, with a system of information release on a weekly basis [34]. The service also provides an annotation system from several other databases and includes cross-references to other biological resources, with the possibility of exporting files via FTP [34].

1.2.4 MobiDB database

MobiDB is a centralized database for IDPs annotated information, which covers all the UniProt sequences and offers annotations and predictions taken from several sources with an online visualization interactive tool [35, 36]. MobiDB features three levels of annotations [37, 38]:

1. manually curated data, extracted from the DisProt database [39] and expanded using disorder annotations that include information from several databases;
2. indirect data, inferred from the structures of the PDB entries of many proteins, can be considered as an indication of intrinsic disorder; indirect data were taken by using missing residues from X-ray crystallography and from structures and mobile NMR, while integrating also high temperature X-ray structures and overall conformation diversity;
3. predicted data, derived from several predictors and expanded with various types of annotations on backbone rigidity, secondary structure preferences, and disordered regions.

Finally, MobiDB provides tools for performing comparative analyses, allowing the identification of inter-organisms characteristics, and instruments for cross-referencing to other protein databases [40]. An example of the content page of MobiDB for a specific protein is shown in Figure 1.6.



P04637 - Cellular tumor antigen p53

Description

Gene: TP53 **Organism:** Homo sapiens (Human) **Length:** 393

Taxon ID: 9606 **Reference proteome:** UP000005640

Dataset: SwissProt - P04637

Localization: Cytoplasm,Nucleus,Nucleus, PML body,Endoplasmic reticulum,Mitochondrion matrix,Cytoplasm, cytoskeleton, microtubule organizing center, centrosome

Features content

Disorder	LIP	Domain
42.50%	53.70%	74.30%

Cross References

UniRef UniRef50 UniRef90 UniRef100	**UniParc** UPI000002ED67	**MFIB** MF2201002
DIBS DI1000002	**IDEAL** IID00015	**Conf. diversity (CoDNaS)** 1DT7_X
FuzDB FC00084	**DisProt** DP00086	**AlphaFoldDB** P04637
Gene3D 2.60.40.720 4.10.170.10	**Pfam** PF00870 PF07710 PF08563 PF18521	

```
        10         20         30         40         50         60         70         80         90        100        110        120
MEEPQSDPSV EPPLSQETFS DLWKLLPENN VLSPLPSQAM DDLMLSPDDI EQWFTEDPGP DEAPRMPEAA PPVAPAPAAP TPAAPAPAPS WPLSSSVPSQ KTYQGSYGFR LGFLHSGTAK
       130        140        150        160        170        180        190        200        210        220        230        240
SVTCTYSPAL NKMFCQLAKT CPVQLWVDST PPPGTRVRAM AIYKQSQHMT EVVRRCPHHE RCSDSDGLAP PQHLIRVEGN LRVEYLDDRN TFRHSVVVPY EPPEVGSDCT TIHYNYMCNS
       250        260        270        280        290        300        310        320        330        340        350        360
SCMGGMNRRP ILTIITLEDS SGNLLGRNSF EVRVCACPGR DRRTEEENLR KKGEPHHELP PGSTKRALPN NTSSSPQPKK KPLDGEYFTL QIRGRERFEM FRELNEALEL KDAQAGKEPG
       370        380        390
GSRAHSSHLK SKKGQSTSRH KKLMFKTEGP DSD
```

🟦 Consensus C Curated D Derived H Homology P Predicted

Annotation source **Priority consensus**

Coverage **39.2%**
Disorder residues Intrisically disordered residues - Curated Experimental annotation intergated from manually curated database

Coverage **57.0%**
Observed residues Experimentally observed residues in the PDB structure (X-ray, NMR, Cryo-EM) - Derived Experimental annotation automatically derived from primary data, e.g. from PDB structures

Coverage **3.3%**
Disorder residues Intrisically disordered residues - Predicted Predicted annotation by a sequence-based method

Coverage **0.5%** Start **183** - End **184**
Structure Structured residues - Predicted Predicted annotation by a sequence-based method

BioComputing UP - Department of Biomedical Sciences - University of Padova. License & disclaimer

Figure 1.6: MobiDB complete example page for the p53 protein.

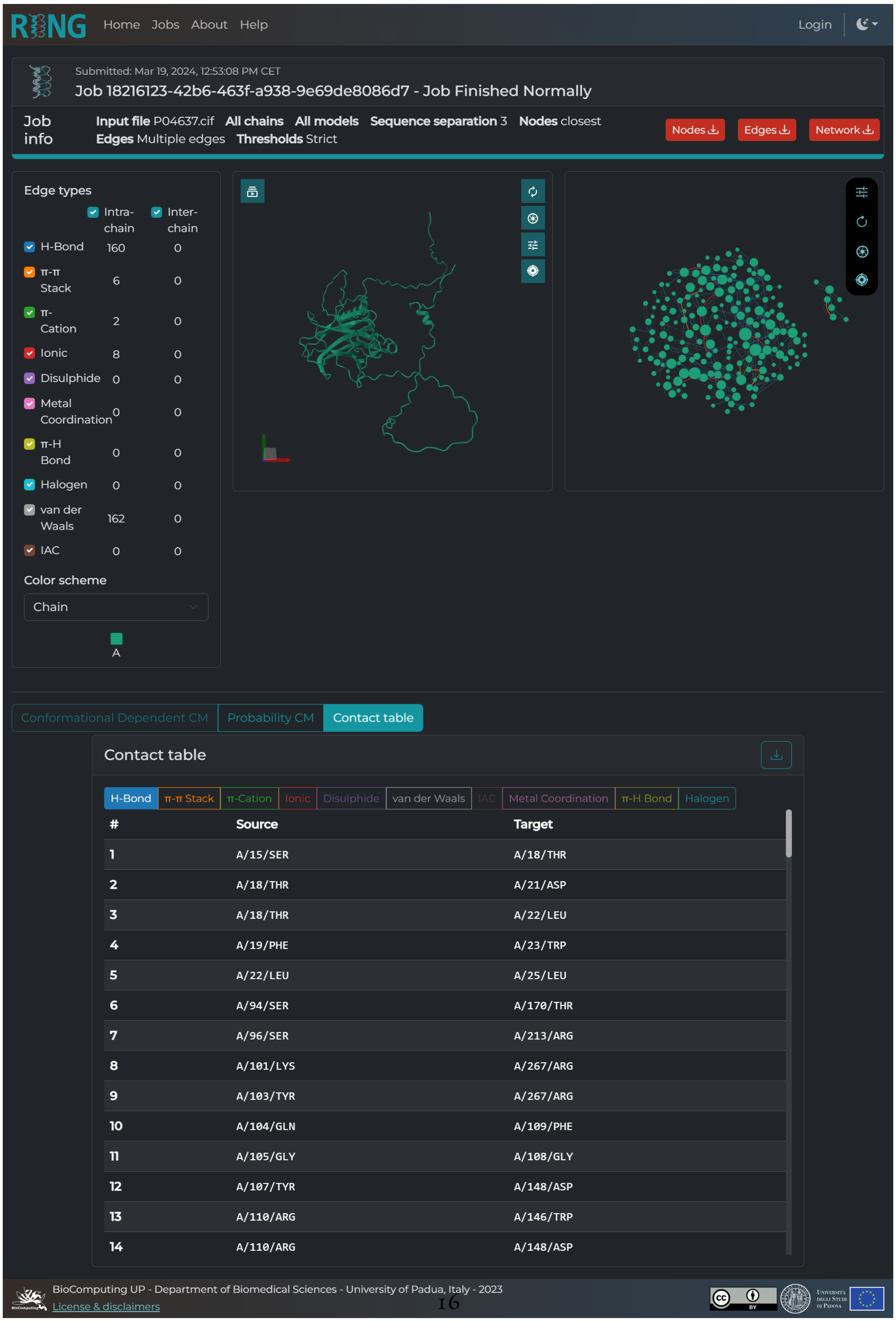
1.3 Protein Structure and Features Prediction

The ability to predict protein structures and features has wide implications in modern biomedical research and pharmaceutical development, while providing a powerful *in silico* system for protein functional characterization [41]. Nowadays, the protein prediction field has proven to be a powerful research instrument, which can be used to provide great help to protein function studies, by combining general concept and tools from molecular biology and computer science, but also exploiting methods and models from machine and deep learning [42]. This chapter is dedicated to some of the prediction tools implied in the field of protein structures and features research, briefly introducing and describing those who have been an essential base for this project, without exploring their technical aspects.

1.3.1 Residue Interaction Network Generator

As already discussed above in several sections, PPIs are an extremely important features in living organisms, just as they are for scientific research. One of the many ways of visualizing and characterizing PPIs is through Residue Interaction Networks (RINs), which are a conceptual framework that provides a way to represent protein structures using networks in which nodes represent amino acid residues and edges represent interactions, with the purpose of obtaining additional information about proteins' structure and function both at protein and residue levels [43]. By taking into account the absolute importance of studying protein structures in terms of intrinsic interaction patterns for understanding protein conformations and functions, it is easy to conclude the importance of RINs in the process of exploring and understanding the structural properties and the conformation dynamics of proteins [44].

In this context, a powerful instrument is RING, an advanced tool for creating high-quality RINs at the atomic level, with the purpose of providing efficient and accurate information about intra- and inter-chain interactions; RING is able to identify non-covalent bonds in multi-state protein structures (from both molecular dynamics and conformational ensembles), allowing a visualization of RINs through Cytoscape or web-browser [45].



Submitted: Mar 19, 2024, 12:53:08 PM CET

Job 18216123-42b6-463f-a938-9e69de8086d7 - Job Finished Normally

Job info **Input file** P04637.cif **All chains** **All models** **Sequence separation** 3 **Nodes** closest **Edges** Multiple edges **Thresholds** Strict

Edge types

	Intra-chain	Inter-chain
H-Bond	160	0
π-π Stack	6	0
π-Cation	2	0
Ionic	8	0
Disulphide	0	0
Metal Coordination	0	0
π-H Bond	0	0
Halogen	0	0
van der Waals	162	0
IAC	0	0

Color scheme

Chain

🟩 A

Contact table

#	Source	Target
1	A/15/SER	A/18/THR
2	A/18/THR	A/21/ASP
3	A/18/THR	A/22/LEU
4	A/19/PHE	A/23/TRP
5	A/22/LEU	A/25/LEU
6	A/94/SER	A/170/THR
7	A/96/SER	A/213/ARG
8	A/101/LYS	A/267/ARG
9	A/103/TYR	A/267/ARG
10	A/104/GLN	A/109/PHE
11	A/105/GLY	A/108/GLY
12	A/107/TYR	A/148/ASP
13	A/110/ARG	A/146/TRP
14	A/110/ARG	A/148/ASP

BioComputing UP - Department of Biomedical Sciences - University of Padua, Italy - 2023
License & disclaimers

Figure 1.7: RING complete example page for the p53 protein.

1.3.2 Fast Linear Interacting Peptides Predictor

Fast Linear Interacting Peptides Predictor (FLIPPER) is an automated method developed and presented by Monzon *et al.*, (2021) [16], for detecting LIPs using a random forest classifier that takes as input several structural protein features as input and returns a binary LIP classification as output, helping to distinguish LIP regions from non-LIPs regions. The features considered by FLIPPER include some of the canonical LIPs' characteristics, like intra- and inter-chain contacts, secondary structure content, solvent accessibility in both bound and unbound states, structural linearity, and chain length, all used by the model to predict the propensity of each amino acid to be part of a LIP region with a high accuracy. After training with successful results, FLIPPER has been used to process the entire PDB dataset, identifying in this way various LIPs based on different binding modes and partner molecules. FLIPPER is now used to generate prediction data about LIPs for the MobiDB database [46].

1.3.3 AlphaFold and AlphaFold-Disorder

AlphaFold is a powerful AI tool developed by Google DeepMind for predicting the three dimensional structure of proteins starting from their primary structures (i.e., the plain amino acid sequences); AlphaFold has proven to be extremely effective, regularly reaching an accuracy level that is competitive with experimental data [41, 47]. Shortly after its release, AlphaFold has soon turned out to be an essential tool for improving the current knowledge of protein structures, thanks to its use to a large-scale processing of all known protein sequences, enabling a huge and sudden expansion of what is the structural coverage of protein sequences [47]. Furthermore, the deep learning models used for making such a powerful protein prediction system were able to set a new standard, outperforming previous methods with their very high performances [41]. Specifically, a tool that was important for this project is AlphaFold-Disorder, which is an algorithm that exploits the predictive power of AlphaFold to extrapolate structural and functional protein features, like the presence of secondary structures, the predicted Local Distance Difference Test (pLDDT, i.e., an estimated probability of AlphaFold prediction score confidence) score [48], and the RSA, subsequently combining them to predict the presence of interactions and disordered regions at an amino acid level [49].

1.3.4 ProtTrans

A novel and promising field of study is Protein Language Models (PLMs) research, which derives from the intersection of computational biology and machine learning, with the aim of exploiting protein sequences in order to understand their functional and structural characteristics [50]. This new field of study uses NLP models, usually employed in the study of human language, to analyze amino acid sequences that constitute proteins, encoding such sequences into vector representations to try to capture structural and functional sequence-enclosed properties [50]. This type of approach has resulted in new high-performance methods which utilize deep learning techniques for providing a significant improvement in predicting protein structure, taking advantage of several advanced techniques, like transformers with attention mechanisms (MSA Transformer, Rao *et al.*, 2021 [51]) or auto-encoders (proteinBERT, Brandes *et al.*, 2022 [52]).

One of the most important works in this field is provided by the ProtTrans project, developed by Elnaggar *et al.* (2021) [53]. In this project several language models have been tested on proteins sequences in order to explore their limits, while the effects of two auto-regressive models (i.e., Transformer-XL and XLNet) and four auto-encoder models (i.e., BERT, Albert, Electra, and T5) have been tested on extensive datasets and compared to existing state-of-the-art solutions based on evolutionary information and MSA. ProtTrans models are able to outperform those state-of-the-art methods, representing an important leap in the process of integrating deep learning techniques in protein research. Protein embeddings can be a powerful tool for the analysis and prediction of structural and functional characteristics of proteins, although the black box nature of the deep learning models involved in the embedding generation makes it complex to understand the full potential and the meaning of these vector representations. In fact, it is important to bring to attention the fact that strengths and limitations of protein embeddings are still the subject of active study and experimentation (this fundamental concept will be addressed and explored in more detail in the **Results and Discussion** chapter).

1.3.5 Critical Assessment of protein Intrinsic Disorder prediction

It must be clear at this point the level of challenge constituting by studying the protein IDRs, as well as understanding their structural features and functional role in the intricate pattern of the molecular processes of life.

In a context in which the large part of the knowledge about IDRs is based on predictions, the CAID experiment, established in 2018, takes its place as a community based instrument to asses the state of the art in predicting IDRs and their binding portions in a blind-test setting; every tool evaluated must be able to assign a score to each amino acid residue, basing on its propensity to be labelled in a certain way, depending on the specific predictive task associated and starting only from the protein sequence[54, 55]. The CAID experiment presents several predictors (with accuracy results) evaluated on specific sets of proteins, using a round organizational system: the predictors evaluated at each round are presented publicly on specific dates [54, 55]. Specifically, the challenge can be divided into three tasks [55]:

- disorder prediction, which uses two test sets, *Disorder NOX* and *Disorder PDB*, containing 210 and 348 proteins, respectively (Figure 1.8 and Figure 1.9); *Disorder NOX* and *Disorder PDB* are both sets of proteins made for the disorder prediction task; however, in the first case (i.e., *Disorder NOX*) the positive examples consist of amino acids that possess a DisProt disorder annotation (excluding X-ray missing residues), while the negative examples are anything that is not characterized by the disorder annotation in DisProt; in the second case (*Disorder PDB*), the positive examples are the positions with DisProt disorder annotation (including X-ray missing residues), while the negative examples consist only of the positions that are observed in the PDB, i.e., the positions that are characterized by annotation derived from protein structure study techniques, such as NMR, X-ray crystallography or cryo-EM;

- binding prediction, which uses a *Binding* test set of 78 proteins (Figure 1.10); the positive examples of this dataset are sequences that establish interactions and are located within disordered regions;
- linker prediction, which uses a *Linker* test set of 40 proteins (Figure 1.11); linkers are defined in DisProt as unstructured regions that provide separation and allow movements between two adjacent functional regions.

Among these tasks, the one that is most related to the specific task of this project is the second of the list, i.e., the binding prediction one. However, it is important to highlight some important differences between these two tasks. In fact, while this project focuses on the classification of LIP regions, the CAID binding task focuses on generic interacting amino acid sequences found within disordered regions. LIP regions possess specific structural features (as already noted in detail in the biological part of this introduction), and, although they are predominantly found in disordered regions, they can also be observed to a lesser extent in ordered regions (this aspect will be analyzed in detail in the following chapter). Therefore, the task of this project differs slightly from what can be observed in CAID, although the specific binding CAID task can be useful for this project, as will be seen later in this paper.

By looking at Figures 1.8, 1.9, 1.10, and 1.11, it is possible to notice that the disorder prediction task has reached good results using the *Disorder PDB* test set, while binding and linker predictions tasks still need improvements.

Another important plot considered in the CAID results web page is the ”Runtime vs AUC” plot, which is a way for the CAID challenge to measure the execution time in relation with the AUC score. In this case, the execution time is calculated as the average time for the prediction of a sequence of 1000 residues against the measure of the area under the ROC curve Figure 1.12.

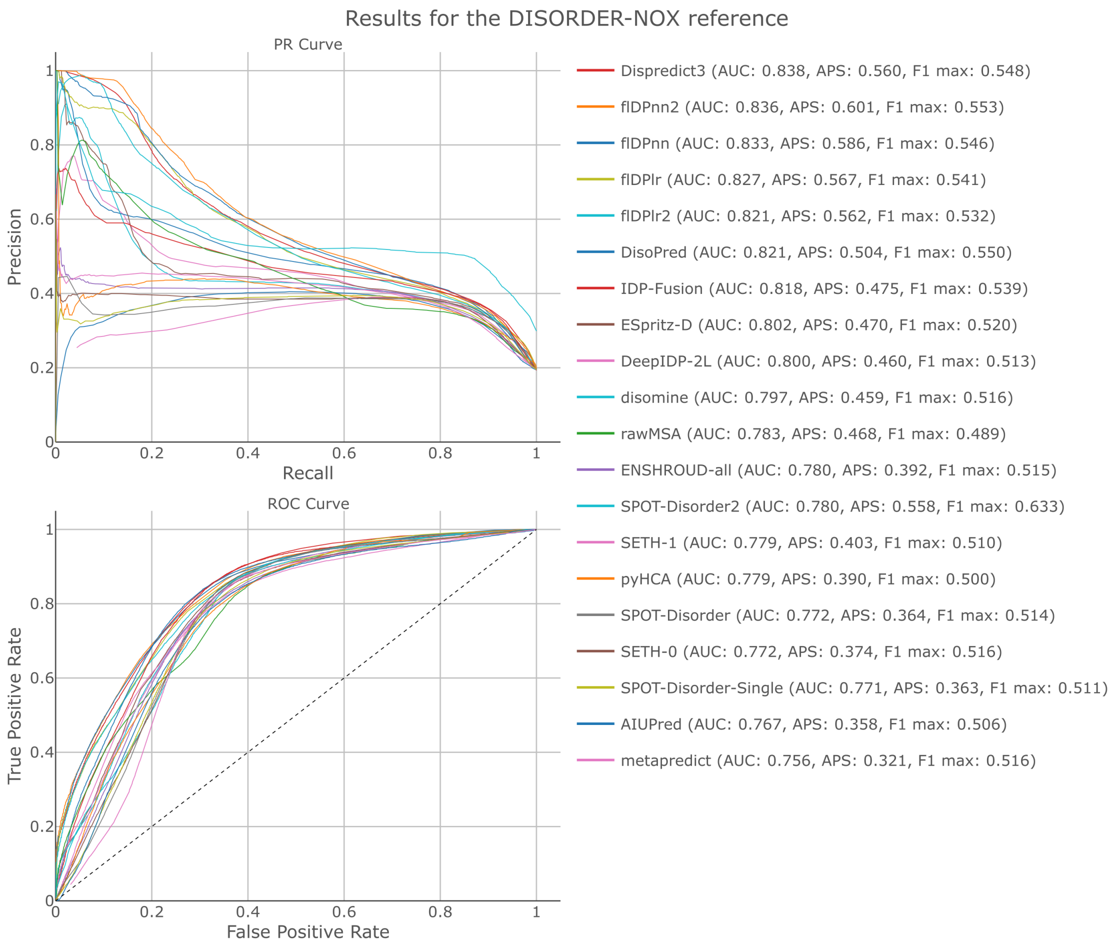


Figure 1.8: ROC and PR curves for the top 20 best models in predicting the disorder state in the *Disorder NOX* dataset (sorted by AUC score). Best results are provided by the Dispredict3 model, with an AUC score of 0.838 and a max F1 score of 0.548.

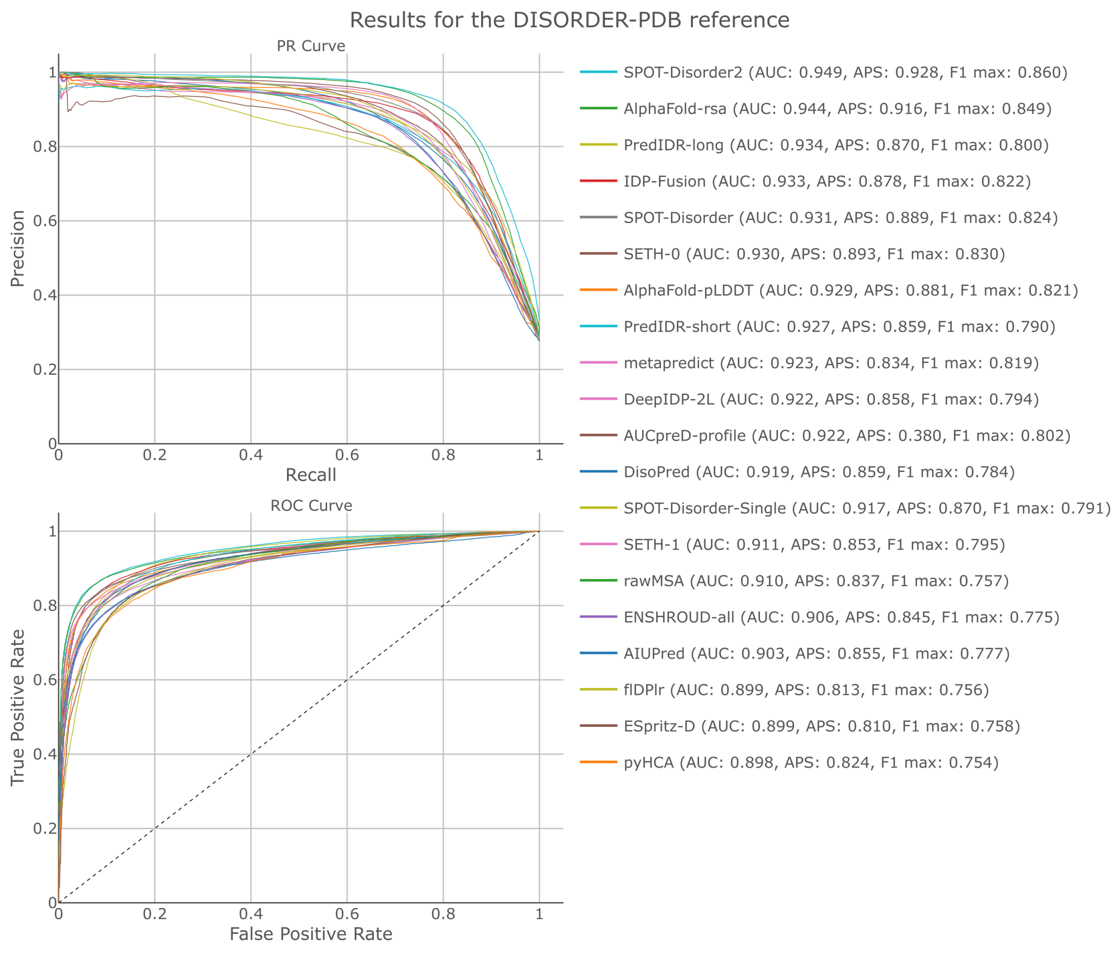


Figure 1.9: ROC and PR curves for the top 20 best models in predicting the disorder state in the *Disorder PDB* dataset (sorted by AUC score). Best results are provided by the SPOT-Disorder2 model, with am AUC score of 0.949 and a max F1 score of 0.860. The *Disorder PDB* dataset, if compared with the *Disorder NOX* one, provides better quality results and brings out the best from the predictors used, with excellent results in terms of accuracy.

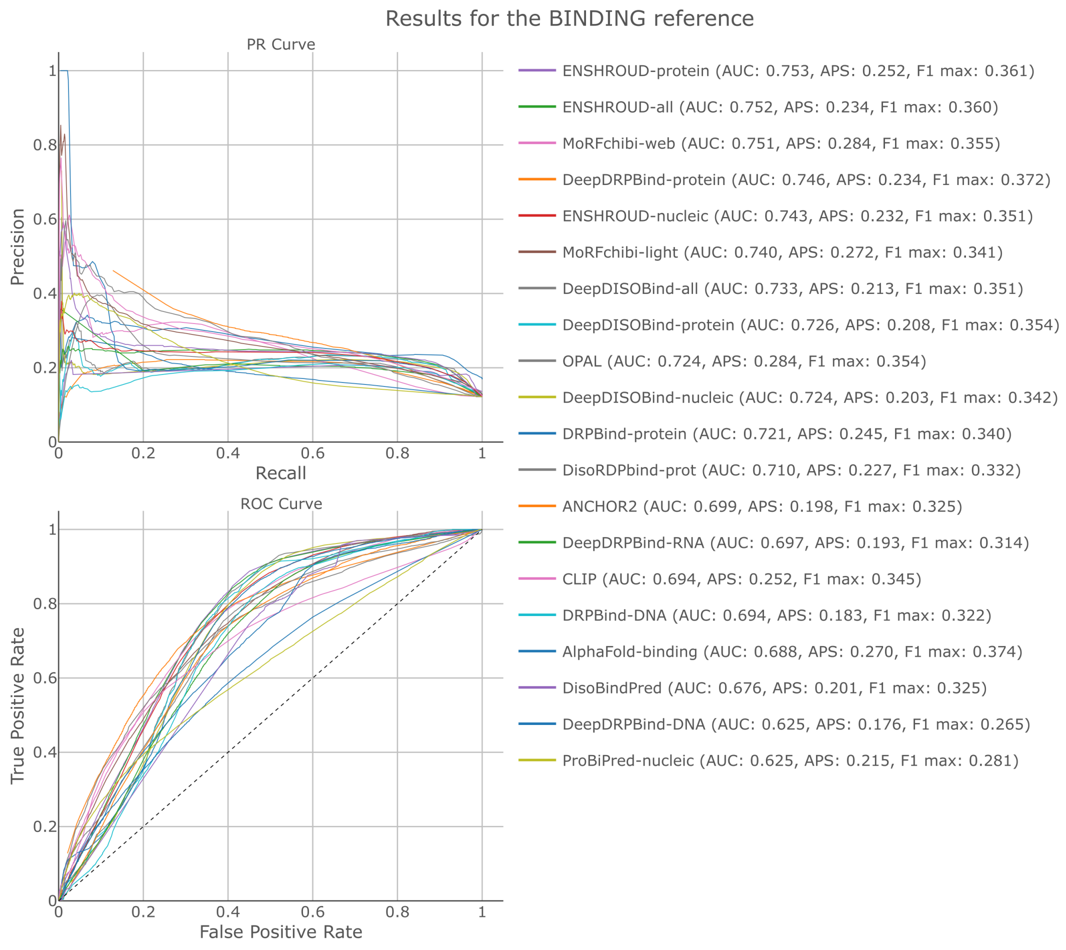


Figure 1.10: ROC and PR curves for the top 20 best models in predicting the presence of binding amino acids within disordered regions in the *Binding* dataset (sorted by AUC score). Best results are provided by the ENSHROUD-protein model, with an AUC score of 0.753 and a F1 score of 0.361.

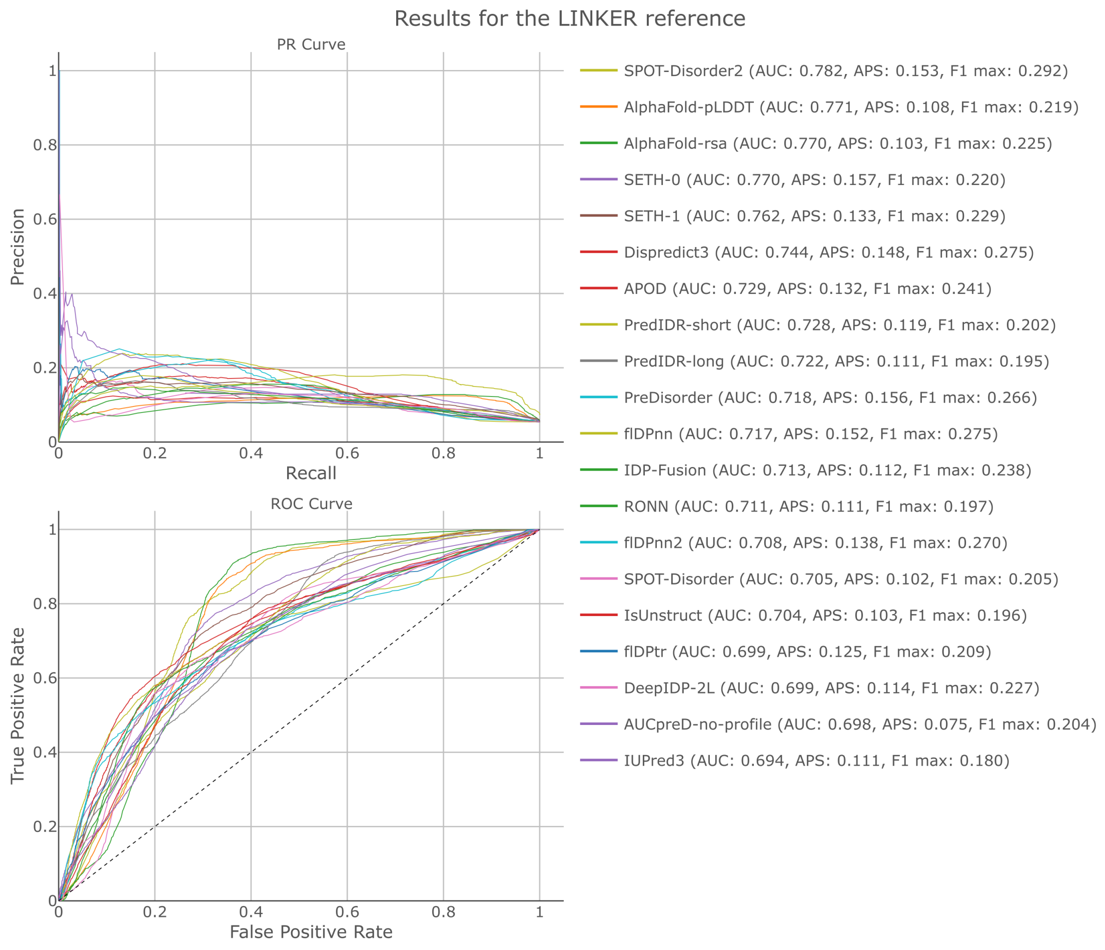


Figure 1.11: ROC and PR curves for the top 20 best models in predicting the presence of linkers in the *Linker* dataset (sorted by AUC score). Best results are provided by the SPOT-Disorder2 model, with an AUC score of 0.782 and a F1 score of 0.292.

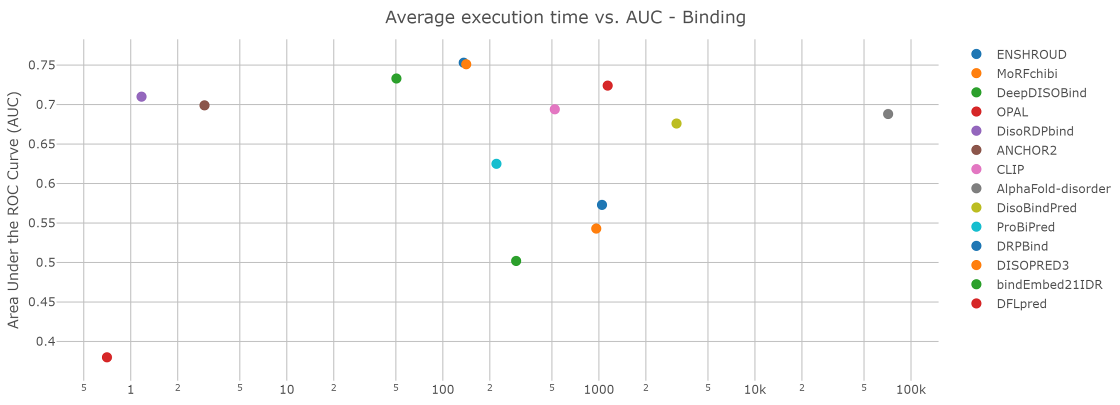


Figure 1.12: Runtime plot for the CAID binding prediction task. The figure shows the area under the ROC curve for some binding prediction algorithms, visualized against the averaged execution time; best algorithms tend to be in the upper left corner of the plot, with a runtime as close to zero as possible and with an area under the ROC curve as high as possible.

1.4 Approach and Techniques for Embedding-Based LIPs Detection

The project presented in this paper combines concepts derived from molecular biology and from data science, in order to formulate an effective analysis of linear interacting peptides, with the final goal of creating a deep learning model for a better understanding of limits and potential in predicting LIP regions.

The **Materials and Methods** chapter is dedicated to the comprehensive, technically addressed, and in-depth discussion of everything that was used to achieve the project results, presented in two distinct sections, *Datasets* and *Models*. In the *Datasets* section the training, validation, and test datasets used for this project are discuss. A first part of this section shows how such datasets were processed, adding important further information, such as amino acid position-specific disorder labels and details about the interactions established by each amino acid. These two features were obtained using AlphaFold-Disorder and RING, respectively. With this additional information, it was possible both to identify some problematic LIP regions in the dataset (in terms of effective interactions) and to extract interesting features regarding the characteristics of LIPs, in terms of their presence in protein regions classified as ordered or disordered. The identification of LIP regions that showed a small amount of effective interactions made it possible to clean the training dataset, by removing regions that may constitute false positives, which, although in reduced quantity in this dataset, would still be able to increase the level of overfitting, reducing also the model's ability to generalize. At the same time, the introduction of disorder information made it possible to identify important differences between LIP portions within ordered regions and LIP portions within disordered regions (the latter constitute, based on the definition given above, the majority of LIP regions). A second part of the *Datasets* section is dedicated to the exploratory data analysis performed on the processed data, in order to check the training and datasets through the use of descriptive statistics and distribution plots, but also using dimensionality reduction techniques, such as PCA and t-SNE. This specific analysis was important to be able to spot patterns, anomalies, and other useful information that can be used when building the final predictive model.

The *Models* section presents some theory about convolutional neural networks, introducing then the model used in this project and the data preparation that was done for building the input set for training the final CNN model. In particular, the training dataset introduced and analyzed above was finally split into a training set and a validation set, using a 5-fold stratified cross validation, a technique which is useful for unbalanced datasets, allowing to apply a k-fold cross validation, while keeping the proportions and distributions of positives and negatives in each fold. Single protein sequences were used as inputs for the CNN model, after applying a 0-padding to equalize all the sequence lengths, getting in this way inputs of 1024 feature embeddings (one for each amino acid position) and sequence lengths of 1000 (i.e., the single amino acids, used as examples plus the 0 positions made by the 0-padding step). Finally, batches were extracted to train the model. Such model is a CNN with a combination of pointwise convolutionals layer (with a kernel size of 1), for reducing the number of feature of each input, and depthwise convolutional parallel layers, used for analyzing two different classes of LIP regions, i.e., the ordered LIP regions and the disordered LIP regions (with kernel sizes of 9 and 19, which are the average length of ordered LIP sequences and disordered LIP sequences, respectively, as will be seen later). The weights for each convolutional layer were initialized using a Xavier initialization, while outputs of each of convolution went through a ReLU activation function, for introducing non-linearity, and through a batch normalization. The final outputs of these three convolutional layers were then combined and a dropout was used. The model uses an AdamW optimizer with a binary cross-entropy loss function, that integrates an output sigmoid activation function. For taking into account the datasets imbalance, the weights of the model were themselves unbalanced, for giving a proportional importance basing on the magnitude of dataset imbalance. For making a more stable model, it was necessary to implement a L2-regularization, with a contribution from both the batch normalization and the dropout layer. A the same time, a weight decay of 10^{-3} was used, while the learning rate was kept low to a value of 10^{-5}. The final model hyperparameters were chosen through a grid-search for tuning the number of channels as output for the pointwise convolution and the size of the input batches, seeking a maximization of the F1-score on the validation set. In this way, a value of 512 output channels for the pointwise convolution and a batch size of 16 were chosen. The 5 fold stratified cross validation was trained for 50 epochs, using a variant of the early stopping trigger that was able to provide a consensus for both the number of epochs of each fold and the values of the descending loss function of the model.

One of the most important things to consider before starting to describe in detail materials and methods of this project is the inherent difficulty of predicting interacting sequences within disordered regions. As can be seen in Figure 1.10, this task still has much room for improvement, since even the best results currently available still show rather poor metrics, with a max F1 score for the CAID challenge case that does not go above 0.361. This specific issue will be covered in detail in the **Results and Discussion** chapter and can be justified by the intrinsic great variability in the characteristics of interacting disordered regions, which poses a challenge of great complexity. Considering this important fact and considering the binding CAID task problem as a reference to take into account such difficulty, the main goal of this project is to try to raise the bar in this very specific field, without the need to obtain a model with high predictive potential, but rather with the intent to contribute as much as possible to the study and understanding of these important and elusive biochemical structures.

2
Materials and Methods

This chapter focuses on discussing datasets, tools, and models used during the course of this project, through two sections, one for the datasets and the other for the models. Specifically, the first *Datasets* section is dedicated to the training and test datasets used to analyze LIP regions and to perform predictions. Such datasets are introduced and described, paying particular attention to their characteristics and their processing. It also shows the results obtained from the exploratory data analysis. In this section, several tools used to carry out the dataset analysis and processing are presented, providing descriptive statistics metrics and visualizations, while introducing also processing algorithms, such as t-SNE and PCA. The **Models** section is dedicated to deep learning, talking about the model chosen for the prediction of LIP regions. In particular, this section provides several theoretical notions about Convolutional Neural Networks and some other general deep learning concepts, while talking about the CNN model developed for carrying out the task of this project. This CNN model is carefully described in detail, justifying architectural and parameters choices, while also showing its potentialities and limitations, exposing the processes that led to the development of a stable model and describing the hyperparameters tuning step.

2.1 Datasets

This section is dedicated to give a detailed exposition of the process of creating the datasets that were useful for a correct analysis of the problem and for training the prediction algorithms. It often happens in the field of Data Science that the process of creating, cleaning and maintaining datasets that must be used in learning models is a preponderant phase, both in terms of effort and time demand, while developing or performing any artificial intelligence pipeline. It can happen, in many cases, that the attention to the dataset is the most important, time consuming, and difficult phase, which may, however, be rewarded with relevant results, as long as the data have been handled in the right way, with meticulousness and a critical thinking approach. Specifically, in this project, the importance and difficulty of generating an adequate training set were proof of what was just stated above. The effort required at this stage was indeed significant, while the time requirement was remarkable. However, this process made it possible to create a dataset which was relevant to the problem, generating a training dataset that could be efficiently used for training and validate a convolutional neural network, as will be shown in detail in the section dedicated to models.

2.1.1 Data preprocessing and analysis

This section presents the source dataset retrieved from MobiDB for building the training set and the two test datasets for finally testing the models, i.e., the TE440 dataset, retrieved from the Biomine Kurgan Lab website [17, 56], and the CAID Binding dataset, retrieved from the CAID website [57]. This section also provides a complete explanation of the processes performed to build the final training and test datasets. In particular, it is possible to divide the preprocessing into two distinct steps. Firstly, the protein sequences from the training dataset were enriched using supplementary useful information from RING and AlphaFold-Disorder, and filtered; subsequently, sequence embeddings, generated using ProtTrans, were integrated to both the the training and test datasets.

Dataset Preprocessing

The dataset used to build the training set was provided from the FLIPPER output dataset, available on the MobiDB website. As mentioned in the previous chapter, FLIPPER was used on the entire PDB. However, in order to formulate a better idea of the output, PDB file-level predictions were mapped to their corresponding UniProt files by the authors, using the SIFTS service [16]. The final file is freely available on the MobiDB homepage (by navigating the section dedicated to dataset and downloading the ”Linear interacting peptide (LIP) - derived” dataset) in both JSON and TSV formats. This file was downloaded through the MobiDB programmatic access system and underwent a preliminary analysis, using Python 3 and several external Python libraries.

The MobiDB dataset introduced above was enriched with interaction data from RING, disorder data generated with AlphaFold-Disorder, and protein embeddings calculated using ProtTrans, using an amino acid level resolution. Specifically, RING outputs were used for determining the actual amount of interactions inside LIP regions. In this way, it was possible to extrapolate a coverage level for each LIP in the dataset, in terms of actual quantity of interactions. The calculation was simple, considering, for each distinct LIP region in each protein, the ratio of the amount of interactions provided by RING to the length of the LIP sequence. For a LIP sequence in which every amino acid has an interaction identified by RING, the coverage would be 1.0, while for a LIP region containing amino acids that do not establish any kind of interaction, according to RING, the coverage would be 0.0. All cases in the spectrum between these two extreme examples possess intermediate coverage values. This process was useful to avoid including sequences with low coverage values, by establishing an arbitrary threshold for discarding LIP regions that have no or few interactions. Such LIP regions are more prone to increase the number of false positives during the training phase of the CNN model. Once identified, LIP amino acid positions included in LIP regions with a coverage under the chosen threshold were transformed into negative positions.

AlphaFold-disorder was used to identify disordered regions in all protein sequences of the training set. In this way, it was possible to distinguish LIP amino acids that fall out disordered regions from LIP amino acids that fall within ordered regions. The idea is to ascertain whether there are significant differences between these two subsets, as this could inform the training of the CNN model and potentially yield better results, as will be extensively discussed later in this paper. Specifically, by using the information brought by AlphaFold-disorder on RSA and pLDDT, it was possible, for each amino acid position, to identify disordered regions as follows [49]:

- to each position for which was true $(\text{RSA} \geq 0.581) \vee (\text{pLDDT} < 0.688)$, a disorder label was added;
- to each position that did not fall within the case above, an "order" label was added, identifying ordered regions in a complementary manner.

At this point, all the labels for disordered and LIP regions were processed using a simple filter, implemented to homogenize the regions. Specifically, the filter takes as input the binary labels and removes short LIP regions that don't meet a specified length criterion (by flipping each "1" label to "0"), while merging together close LIP regions (by flipping each 0 label to 1), using the same length threshold. In this particular case, the length threshold was chosen to be strictly less than 5. In this way, the filter removed regions that were shorter than 5 amino acids positions and joined together LIP regions that were strictly closer than 5 amino acid positions, thus creating uninterrupted LIP regions longer than 5 amino acids.

Finally, ProtTrans was used to generate embeddings for each protein in the LIPs dataset. Embeddings are amino acid-based and give, for each position, 1024 numeric features made up of real numbers that are mostly close to 0. In this way, a protein sequence with length n can generate an embedding matrix of shape $n \times 1024$

(represented with a NumPy array). The Figure 2.1 shows, as an example, the distribution of the first of these features (the values of the distribution are those taken by the feature along the entire dataset).

It is important, at this point, to point out the limitation problem related to the generation of the outputs of AlphaFold, which consequently affects AlphaFold-Disorder. Such limitation regards the fact that AlphaFold has a limit of 2700 amino acids per sequence [47]. Therefore, in order to be able to use AlphaFold-Disorder to discern ordered and disordered regions within the training set, it was necessary to discard some excessively long sequences. In particular, 128 sequences were excluded from the training set, corresponding to 1.23 % of the total dataset. The alternative idea of cutting sequences to not exclude any proteins would have led to potential problems in the reliability of the outputs, since AlphaFold relies on complete sequences to generate its outputs. Complete sequences, as already seen in the introduction chapter, are essential to determine the final structure and thus its properties and this is why it was inevitable to exclude some too long sequences.

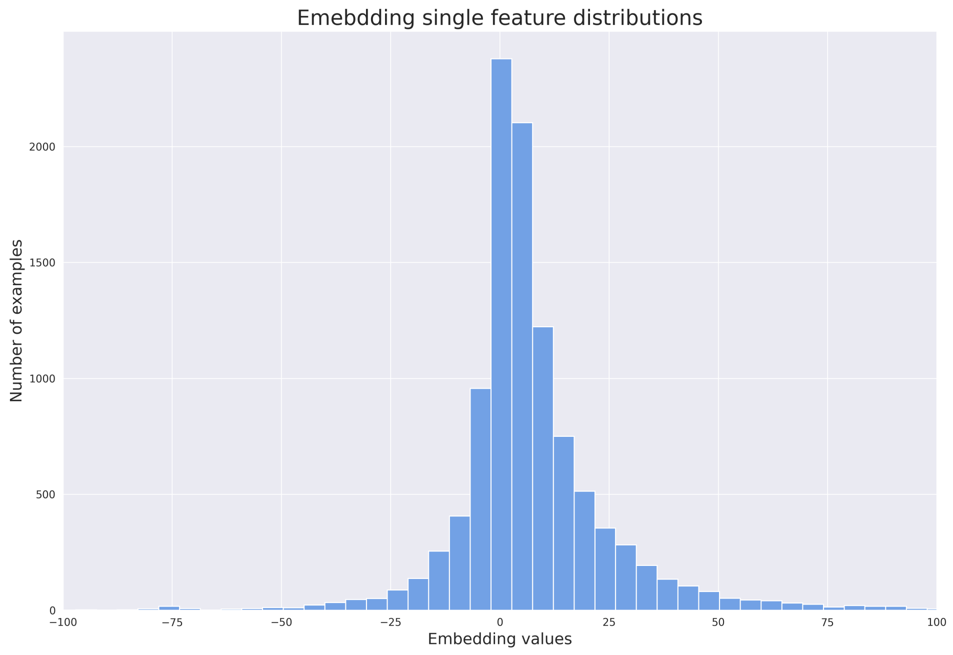


Figure 2.1: 100 bins histogram showing the distribution of the first of the 1024 features; it can be seen that, even with the presence of some outliers, the values are more or less centered on zero and that the distribution, seems to be roughly symmetric, with a mean close to zero. The distribution of the other features is analogous, which is why, by arbitrarily choosing the first feature, it is possible to say that the distribution shown here is representative for the whole group of features.

Exploratory Data Analysis

This subsection is dedicated to EDA, an important step, with the aim of starting to learn about the data, identifying problems, properties and patterns, in order to correctly select the models and their characteristics. This fundamental stage ensures a robust foundation for the formal modeling part of the pipeline, allowing a decision-making process that is consistent with both the scientist's purposes and the inherent properties of the data. In particular, this subsection shows some descriptive statistical analyses and some data visualizations, talking about features and problems of the data from a biochemical point of view.

Starting from the source dataset, that is the MobiDB derived LIPs, it is possible to identify a total number of 228328 rows (of which 6 contain missing values) and 6 columns, explained below:

- **acc**, containing the UniProt IDs for 8278 proteins;
- **feature** showing the evidence-feature-source triplet, i.e., a system that indicates the quality of the annotation, the type of annotation, and the origin of the annotation (more information about this labeling on the MobiDB help section [40]);
- **start..end** containing the coordinate of the LIP regions on the protein sequence;
- **content_fraction and content_count**, i.e., the relative and absolute numbers (respectively) of residues annotated with a specific feature over the length of the protein sequence (i.e., the number LIP residues on the total number of amino acids);
- **length**, that is the total length of each sequence;

As also stated by the MobiDB website, the coverage of this dataset is 7.1 %; the very first information that is possible to notice from this percentage is that the dataset is very imbalanced. Thus, according to FLIPPER predictions, LIP regions are relatively uncommon within proteins. To further analyze this aspect, the fraction of amino acids with LIP character within all proteins in the dataset was calculated, in order to determine the distribution of the percentage of protein sequences responding to the property of LIP, shown in Figure 2.2. In the plot it is shown how 80.28 % of the LIP sequences are located in the first quartile of the distribution, proving that, in about 4 out of 5 cases, LIP regions constitute a portion which is less than or equal to 25 % of the total sequence of the protein in the dataset.

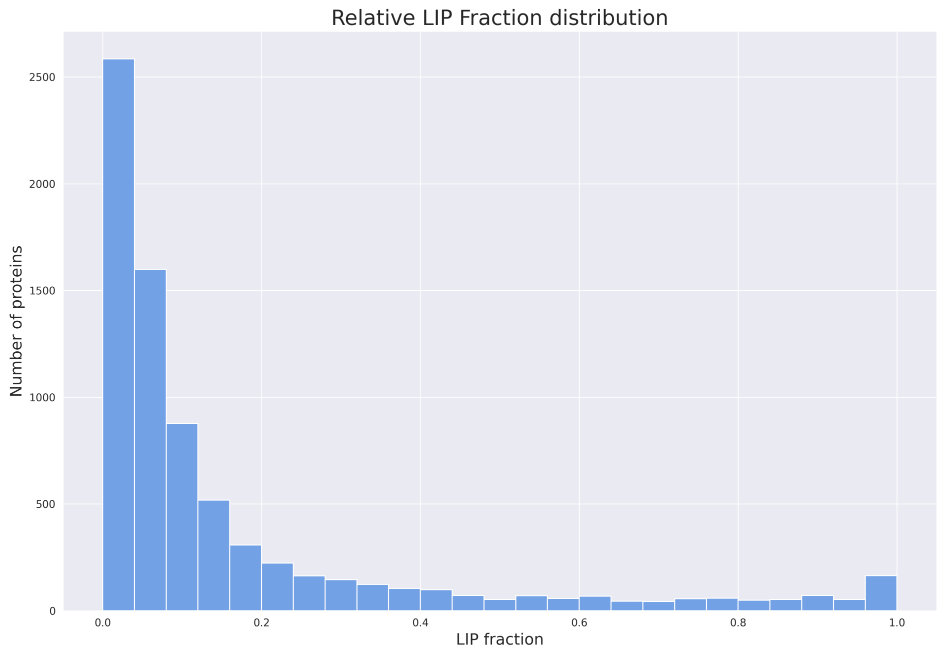


Figure 2.2: Relative distribution of LIPs' fraction in the protein sequences of the MobiDB derived LIPs dataset; the 25 bins histogram highlights the fact that the vast majority of the LIPs actually represents a very small portion of the proteins, with 25 % of the smallest LIPs consisting of 80.28 % (i.e., 6152 elements) of the total sequences and 75 % of the largest LIPs consisting of 19.72 % (i.e., 1.511 elements) of the total sequences. Thus, it is possible to see that 80.28 % of the sequences are located in the first quartile, Q_1, of the distribution shown in this plot.

LIPs Separation Analysis

As can be seen in Figure 2.3, LIP regions that fall within ordered portions of the proteins are in the minority, with a number of 3872 (i.e., 28.31 % of the total regions) and an average length of 9.20 amino acids; on the other hand, LIP regions that fall within disordered portions of the protein, in clear majority, are 9803 (i.e., 71.69 % of the total), with an average length of 18.92 amino acids. There are two immediate observations from these results. Firstly, the fact that disordered LIP regions are in the majority is in line with the definition of LIP as a linear peptide mostly located within disordered regions and capable of establishing interactions; however, the amount of LIP regions within ordered portions of the proteins is symptomatic of the fact that such structures may also regularly lie within structured portions with more defined folding (assuming, of course, the reliability of both FLIPPER and AlphaFold-Disorder data). Secondly, It is easy to notice the significant difference in length between the two sets of LIP sequences. This could be justified by two different scenarios:

1. distinct LIPs exist within ordered regions; in this case it is possible that such regions might show peculiar

structures with peculiar amino acid compositions in order to exist in such ordered protein portions or that the ordered regions containing LIPs are actually regions with low levels of order that allows the presence of short LIPs;

2. the ordered LIP regions identified are not distinct LIPs, but they are rather regions belonging to LIPs that possess most of their sequence in disordered regions, but which overrun into ordered boundary regions.

In both cases, it is important to consider and study this separation more in detail, since the final prediction of the CNN model will be amino acid-based and, therefore, it is important to help the model discriminate between amino acids belonging to disordered regions and amino acids belonging to ordered regions. This is why it is worth exploring further the separation between disordered and ordered LIP regions, in order to better visualize and deal with it. For this purpose, two distinct dimensionality reduction techniques were used, i.e., Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE), with a silhouette score evaluation for the separation of ordered and disordered LIPs' embeddings. The first one was used to reduce dimensionality, while the second one to check for data separation, both in terms of disordered and ordered LIPs, and in terms of the tendency of individual embeddings to form distinct clusters. The separability is quantified using a silhouette score.

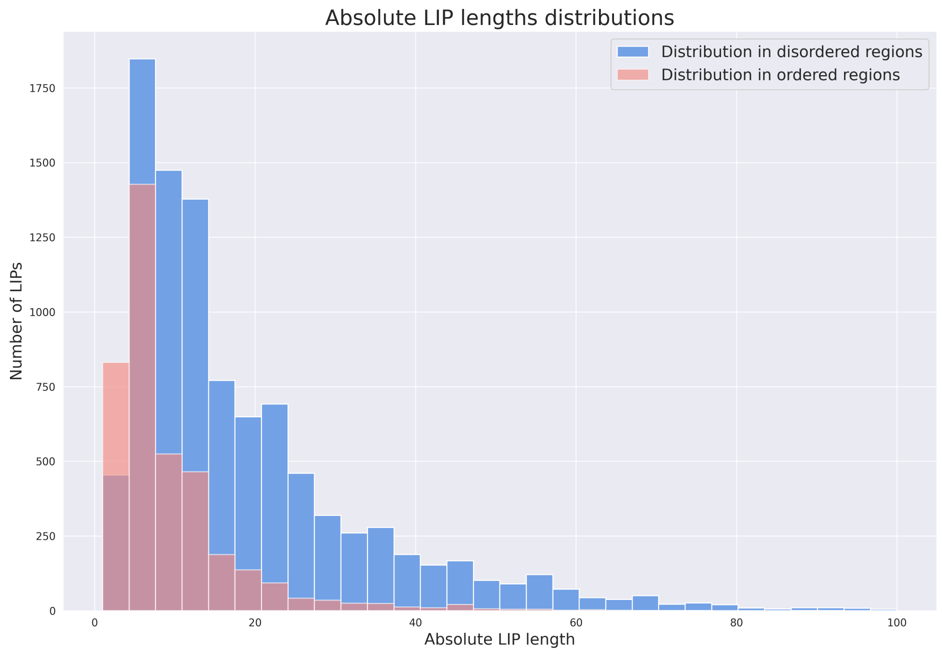


Figure 2.3: Absolute distributions of the grouped LIPs sequences; in soft blue, LIPs that fall within disordered regions (with an average length of 18.92), represented with a 30 bins histogram; in pink, LIPs that fall within ordered regions (with an average length of 9.20), represented with a 30 bins histogram (with $\alpha = 0.6$); x-axis values are absolute length values. It is easy to observe the difference in the amount of sequences and the average for the two groups of LIPs, even if the two distributions seem to be comparable in terms of shape.

Principal Component Analysis

PCA is a statistical method for reducing the size of a dataset by taking the number of features to an lower number, generating the so-called principal components, i.e., a few linear combinations of the original variables that maximally explain the variance of all the variables [58]. PCA operates in real or complex vector spaces that posses an inner product, performing an orthogonal linear transformation of the coordinate system in order to maximize the variance on the first coordinate (i.e., the first principal component) by some scalar projection of the data [59]. Therefore, the first principal component is required to have the maximum variance possible and, under the constraint, the second component is calculated to be orthogonal to the first component [60].

Formally, given the matrix $\mathbf{X} \in M_{I\times J}(\mathbb{F})$ (i.e., $\mathbf{X}$ is a matrix with $I \in \mathbb{N}$ rows and $J \in \mathbb{N}$ columns with entries in the field $\mathbb{F}$), representing the dataset table, by denoting the individual variables of $\mathbf{X}$ as $\mathbf{x}_j \forall j \in \{1, \ldots, J\}$, such that, all the x_j are in a I-dimensional vector space on the field $\mathbb{F}$, then the PCA transformation $\mathbf{t}$ is defined as a linear combination of such vectors [61]:

$$\mathbf{t} = \sum_{j=1}^{J} w_j \mathbf{x}_j \tag{2.1}$$

for some vectors $w_1, ..., w_j \in \mathbb{F}^{1\times I}$. Since the vector **t** is a linear combination of the $\mathbf{x}_j$, it belongs to the same vector space and, thus, it is possible to use the matrix notation to write $\mathbf{t} = \mathbf{X}\mathbf{w}$, such that $\mathbf{w} = (w_1, ..., w_j)$ [61]. The variance contained in the matrix **X** is relevant to the problem and, therefore, it is important to transfer as much of this variance as possible to **t**, thus providing (in the case where the variability of **t** is appreciable) a good summary of the $\mathbf{x}_j$ variables; in this context, the maximization problem is expressed as follows [61]:

$$\underset{||\mathbf{w}||=1}{\arg\max}\, \mathrm{Var}[\mathbf{t}] = \underset{||w||=1}{\arg\max}\, \mathbf{t}^\top \mathbf{t} = \underset{||w||=1}{\arg\max}\, \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} \tag{2.2}$$

where it is assumed that the matrix **X** is mean-centered and, accordingly, all linear combinations are mean-centered as well. The formulation briefly presented here can adapted for the extrapolation of both the first components and the other components of the PCA transformation [60, 61].

t-Distributed Stochastic Neighbor Embedding

t-SNE is an unsupervised non-linear algorithm, used for data exploration through dimensionality reduction, firstly introduced as "stochastic neighbor embedding" by Geoffrey in 2003 [62] and further developed to be the currently utilized t-SNE by Van der Maaten *et al.* in 2008 [63]. These authors were able to develop a technique which is very popular today for visualizing high-dimensional datasets in a lower dimensional space. The mathematical idea behind t-SNE is based on the steps presented below.

Step 1: Similarity Probabilities Calculation in High Dimension

t-SNE starts by calculating probabilities that represent similarities between points in the original high-dimensional dataset [62, 63]. Formally, for each pair of points x_i and x_j, in the set of high dimensional points $X = \{x_1, ..., x_n | n \in \mathbb{N}\}$, t-SNE computes a conditional probability $p_{j|i}$ that represents the probability of x_i choosing x_j as its neighbor (it can be interpreted as a measure of dissimilarity), if other neighbors were picked based on a Gaussian probability distribution centered on x_i; the conditional probability $p_{j|i}$ is defined as [62, 63]:

$$p_{j|i} = \frac{\exp\left(-\frac{||x_i - x_j||^2}{2\sigma_i^2}\right)}{\sum_{\substack{k=1 \\ k\neq i}}^{n} \exp\left(-\frac{||x_i - x_k||^2}{2\sigma_i^2}\right)} \tag{2.3}$$

where $||x_i - x_j||^2$ is the squared Euclidean distance between x_i and x_j, and σ_i is the standard deviation of the Gaussian centered on x_i. The aim of t-SNE is to optimize σ_i for each point x_i.

Step 2: Similarity Probabilities Calculation in Low Dimension

For each pair of points in the low-dimensional map y_i and y_j, t-SNE calculates similar probabilities, but using this time a t-Student distribution with one degree of freedom, instead of the Gaussian, for extrapolating conditional probability q_{ij} (this provides a distribution with heavier tails) [62, 63]:

$$q_{ij} = \frac{\exp\left(-||y_i - y_j||^2\right)}{\sum_{\substack{k=1 \\ k \neq i}}^{n} \exp\left(-||y_i - y_k||^2\right)} \tag{2.4}$$

Step 3: Minimizing the Kullback-Leibler Divergence

The aim of t-SNE is to minimize the Kullback-Leibler (KL) divergence between the two probability distributions P (in high dimensionality) and Q (in low dimensionality), in order to preserve local structures while also revealing global structures [62, 63]:

$$KL(P_i||Q_i) = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2.5}$$

The minimization is usually performed using gradient descent and the result is a low-dimensional map that reflects the similarities between points in the most reliable way possible compared to the original high-dimensional dataset [62, 63]. Therefore, t-SNE transforms Euclidean distances into similarity probabilities, while keeping local closeness through a cost function based on KL divergence and using a t-Student distribution to compute low-dimensional probabilities.

Separation Analysis

As mentioned before, for datasets with a large number of features, it is necessary, before proceeding with a t-SNE, to apply a dimensionality reduction technique, which may vary depending on the sparsity of the data [64]. Since the training dataset embeddings' sparsity (measured as the ratio of zero positions to total positions within embeddings) is less than 10^{-3} % and considering also the large number of features brought by the embedding generation process (i.e., 1024 embedding features), the use of a PCA before t-SNE was necessary to reduce such a high number of features, in order to suppress noise as much as possible and speed up the process of calculating distances between examples operated by t-SNE [64]. The best parameters' set were chosen through a grid-search, in an attempt to maximize the silhouette score, which gives a numeric value for evaluating in a quantitative manner the magnitude of separation between ordered and disordered LIP regions. Silhouette score is a coefficient used for quantifying the magnitude of separation between two groups of points, which is calculated for each example in the points dataset as follows [65]:

$$S = \frac{(b-a)}{\max_{\mathbb{R}}(a,b)} \tag{2.6}$$

where a is the mean intra-cluster distance for each sample, while b is the mean nearest-cluster distance for each sample (i.e., the distance between a specific given sample and the nearest cluster that the sample is not a part of). The S value is in $[-1, 1]_{\mathbb{R}}$, where 1 indicates the best separation, 0 indicates adjacent clusters with boundaries that match, and -1 indicates perfectly overlapping clusters [65].

The implementations of silhouette score, PCA and t-SNE used in this project were all provided by Scikit-learn, while the grid-search was manually implemented for optimizing the following parameters: number of components for PCA and perplexity, learning rate, early exaggeration, and number of components for t-SNE. Finally, other parameters were left with their default values. Following the Scikit-learn user guide [66, 66], it is possible to explain the parameters chosen for the grid-search:

- the PCA number of components is the number of principal components, i.e., how many new variables there are in the output; for high dimensional data with low sparsity a reasonable value of principal components suggested by the Scikit-learn guide is of 50;
- t-SNE number of components corresponds to the dimension of the embedded space, that is, even in this case, the number of new variables in the output;
- t-SNE perplexity is related to the number of nearest neighbors used in other learning algorithms and it is usually larger for larger datasets, still remaining smaller in value than the number of examples of the input; the Scikit-learn guide suggests here to keep the value of the perplexity between 5 and 50;
- t-SNE early exaggeration determines the compactness of natural clusters from the original space within the embedded space, as well as the amount of space that will separate them. With higher values, the distance between natural clusters in the embedded space will be greater; the choice of this parameter is not considered critical;
- t-SNE learning rate influences the learning speed; the user guide suggests to keep it between 10 and 1000; a too low learning rate value, causes the majority of points to appear compressed within a dense cloud, with a few outliers.

Following these instructions, the number of principal components of PCA was determined by exploring the positive values under 50, while the dimension of the t-SNE embedded space was kept on 2; the perplexity value, considering the between 5 and 50, suggested by the user guide, and considering also the big size of the dataset, was searched slightly below 50; the early exaggeration value was searched between 24 and 36. This specific choice was arbitrary, since the user guide lacks of suggestions, but also considering the low importance in the right choice of this parameter; finally, the learning rate was kept low, in order to ensure a good learning process, but not too low, so as to avoid an excessive compression of the data points, thus the explored values were above 10, but below 500. After the grid-search process the final selected values were 15 principal components, 50 for the perplexity, 36 for the early exaggeration, and 250 for the learning rate, getting a final silhouette score of 0.0041, which indicates that the two groups are very close, with almost overlapping boundaries; it is not easy, indeed, to discern the two LIP groups. The plot derived from the combination of PCA and t-SNE used in combination with the grid-search best parameters' set is shown in Figure 2.4. In this plot, two different clustering patterns are evident, with disordered LIPs that tend to form more compact clusters and ordered LIPs that present instead a more scattered distribution, often forming a kind of "satellite cloud" around the more compact clusters of ordered LIPs. Also, the plot shows the formation of a number of quite distinct structures that this type of approach was able to isolate and which may find their reason in other LIP characteristics, which are, however, beyond the main scope of this paper, but that could find interesting applications in additional studies aimed at refining techniques for predicting LIP regions.

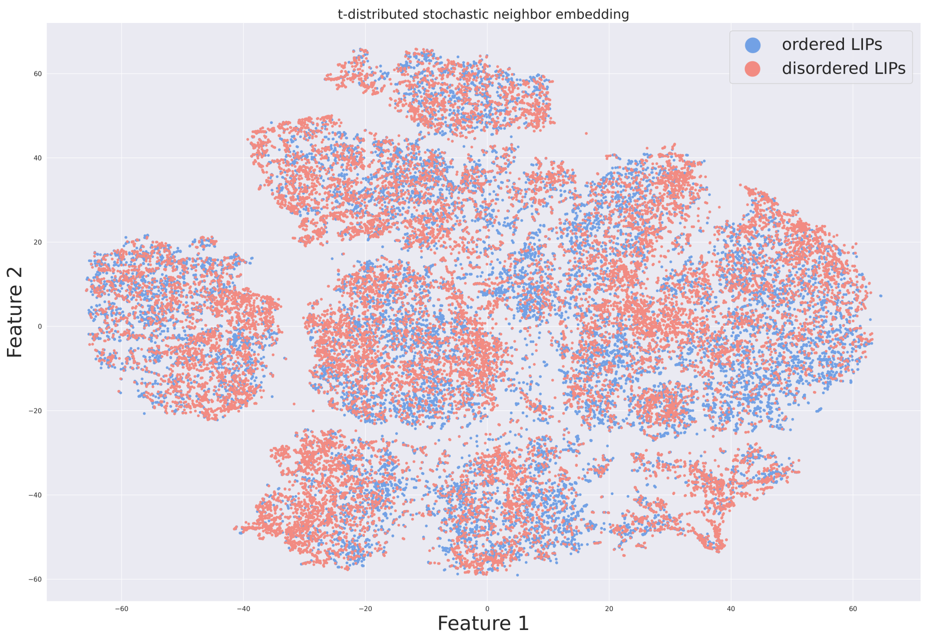


Figure 2.4: Bidimensional plot followed by PCA and t-SNE dimensionality reductions steps. Ordered and Disordered LIPs were labelled to highlight the grouping (with ordered LIP positions in light blue and disordered LIP positions in pink). The silhouette score value for this plot is 0.0041. The plot shown here is a good example for describing the complexity of the problem addressed in this project.

LIPs Coverage Analysis

Coverage analysis for each LIP sequence was able to identify regions with no actual interactions and regions with extremely few interactions, allowing their elimination from the set of the positives. In order to help this process, a coverage distribution plot has been made Figure 2.5. This plot seems to have a reduced, but not absent, symmetry, with the most common coverage values being around the "middle part" of the curve, which shows an average of 50.27 % and a standard deviation of 23.44 %. Considering this fact, a specific number of labels belonging to low coverage LIP sequences was discarded, by flipping their labels to from "1" to "0". This choice allowed to reduce the amount of low coverage regions, with the aim of also reducing overfitting by directly affecting false positive examples. For this project, in order to keep a conservative approach, while ensuring that all values with very low levels of interaction were eliminated, an arbitrary coverage threshold of 10 % was chosen (such threshold allow to consider all values below the percentile 3.5 on the data distribution considered in Figure 2.5), so as to discard LIP regions in which fewer than 1 in 10 amino acids establish interactions. In this way, it was possible to find a total of 341 sequences with a coverage low than 10 %, of which 224 were regions with a coverage of 0 %, i.e., regions without any interaction. Although it was important to remove these LIP regions to ensure better accuracy for the classification models, it is also important to consider the fact that the removed sequences correspond to 3.36 % of the total LIP sequences. For this reason, this step should improve performance and reduce overfitting, without, however, being decisive for the quality of the accuracy metrics. Additional studies in this specific field could focus on analyzing results from applying a higher coverage threshold for choosing which sequences to retain among the positives, trying to maximize accuracy metrics during the model training process. However, this passage must be performed carefully, because applying too high threshold values could lead to a rapid degradation in the quality of the final results, ending up discarding true positive LIP regions in a dataset that is already highly unbalanced.

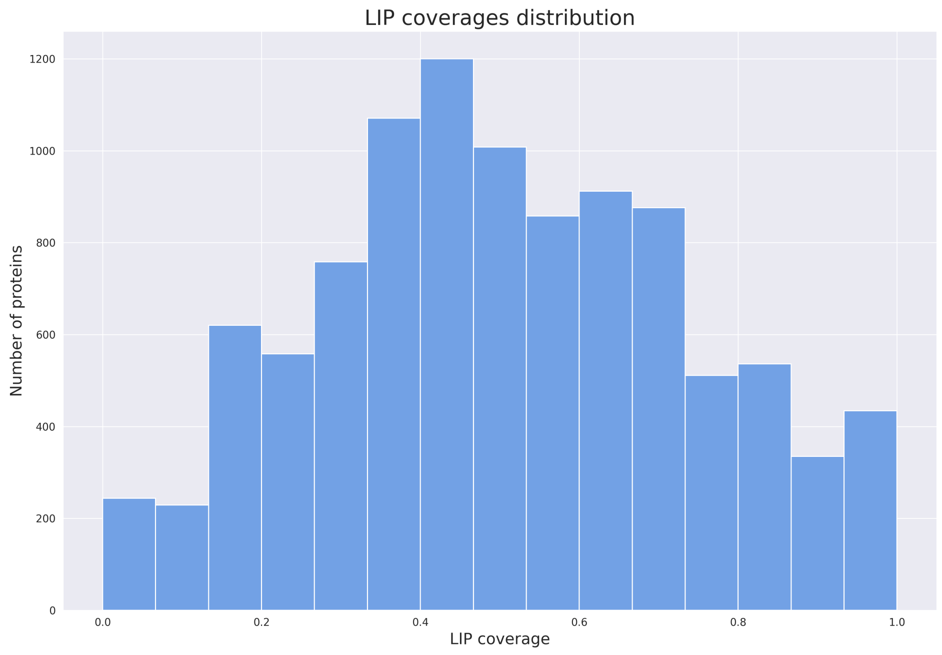


Figure 2.5: 15 bins histogram showing the distribution of the coverages within the LIP regions of the dataset. By looking at this distribution, it is possible to see that there is no prevalent amount of low coverage values.

2.1.2 Exploratory Data Analysis Insights

Now, it is possible to draw some conclusions from the results of this EDA. The thing that can be immediately noticed is the complexity of the data of the training dataset, which shows a high unbalance, with LIP regions that show a number of hidden features that allows the organization in distinct subgroups (as seen in the PCA/t-SNE plot). At the same time, it is important to take into account the presence within the dataset of LIP regions identified as positive examples, even though they present very low (or zero) levels of interactions. Furthermore, it appears that ordered and disordered LIP regions have some separation in their embeddings, a phenomenon that reflects a separation in terms of structural and functional characteristics. Consequently, when building the predictive model, the separation between LIP amino acids that fall into ordered regions and LIP amino acids that fall into disordered regions will be taken into account. Furthermore, coverage analyses might suggest the presence of unreliable positive examples, possibly recognised as false positives by FLIPPER. This aspect is of great importance in the evaluation phase of the model, since can affect the positive examples identification ability of the model.

2.1.3 The Test Datasets

In this project two distinct datasets were used to test the CNN model predictive ability and the limits in the LIPS classification task. The first one is the TE440 dataset, while the second one is the CAID Binding dataset, which was already introduced in the previous chapter. Both datasets contain manually curated protein sequences in which the LIP binary labels for each amino acid position are provided. These test datasets were used after the completion of the training and validation steps for the model, in order to conduct a final test of its capabilities on real, curated, and reliable data that the model had never seen before.

It is important to mention that these two datasets differ and respond to different tasks. As shown below, while the TE440 dataset contains labels of LIP regions in proteins, the CAID Binding dataset contains labels on amino acids that perform general interactions and are exclusively located within disordered regions. Therefore, the latter is a slightly different dataset than the TE440 dataset, which contains information on linear peptides that perform interactions, i.e., a subcategory of binding regions, which are also predominantly located in disordered regions, being, however, also present in ordered regions, as was seen above. On the other hand, the CAID binding dataset shows no information about binding regions in ordered sequence portions, while assigning positive labels to all binding amino acids in disordered regions, whether they are LIP regions or otherwise. In this way, the CAID Binding dataset leaves out some useful information to the goal of this project (i.e., ordered LIP regions binding labels), while at the same time introducing some not useful information (i.e., disordered non-LIP region binding labels). However, although the task related to the CAID Binding dataset is different than the one proposed by this project, it can still be of great utility to evaluate the performance of the model on this dataset as well, as will be seen in the section discussing the results. This is because it was very important, considering the complexity of this problem, to put into context the results obtained during the model evaluation steps, which are predictably lower in terms of performance when compared with those obtained on the TE440 dataset.

TE440 DATATEST

The TE440 dataset was built using the manually curated LIPs that can be found on MobiDB [17]. The file is provided in TXT format, with groups of three lines, in which the first line shows the UniProt identifier of the protein, the second line the protein sequence, and the third line the binary labels for each amino acid in the sequence. The latter was used as target variable for the model in the test phase. The TE440 dataset contains a total of 259063 negative amino acid positions (96.04 % of the total positions) and 10687 positive amino acid positions (3.96 % of the total positions). The distribution of the sequence coverages (in terms of LIP positions content) is shown in Figure 2.6.

Using the same procedure seen above, sequence embeddings for these proteins were calculated with ProtTrans and added to the test set for each amino acid position. Even in this case, it is important to discuss a limitation problem, which affects, in this case, the GPU used to run ProtTrans when computing sequence embeddings. Indeed, a limit of 4000 amino acids was imposed for the length of the sequences, so that the process could be completed without error. This problem did not affect the training dataset, since proteins longer than 2700 amino acids had already been discarded to allow a proper execution of AlphaFold-Disorder. However, the problem affects the test set, from which it was unfortunately necessary to delete 4 protein sequences (corresponding to 0.91 % of the total test dataset):

- the dystonin protein (UniProt ID Q03001), with a sequence length of 7569 amino acids;
- the histone-lysine N-methyltransferase 2D (UniProt ID O14686), with a sequence length of 5536 amino acids;
- the E3 ubiquitin-protein ligase HUWE1 (UniProt ID Q7Z6Z7), with a sequence length of 4373 amino acids;
- the DNA-dependent protein kinase catalytic subunit (UniProt ID P78527), with a sequence length of 4127 amino acids.

Even in this case, as discussed above for AlphaFold-Disorder, it was not possible to avoid excluding such sequences, since complete sequences are essential to get reliable embeddings.

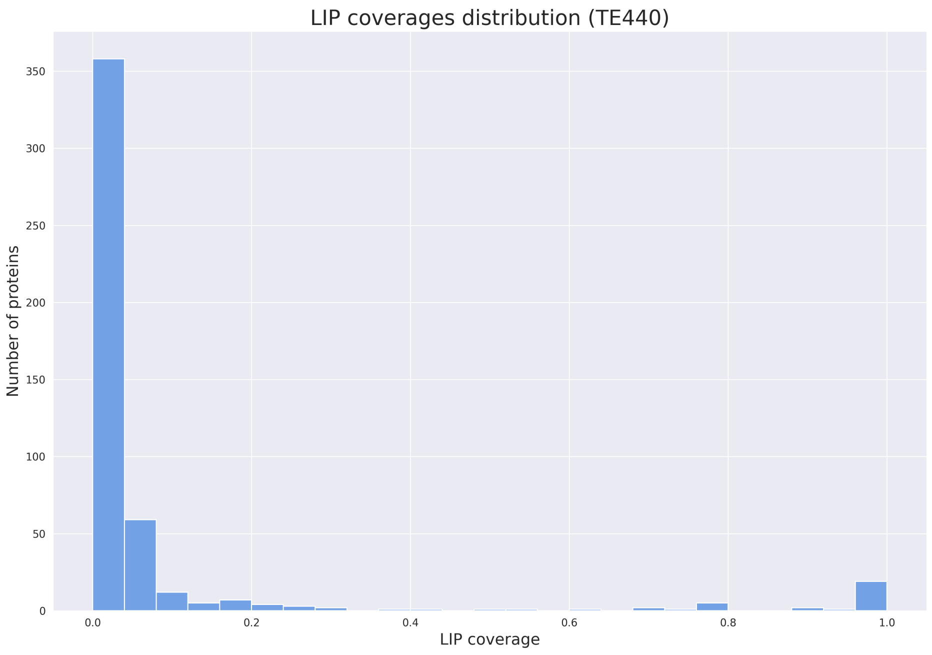


Figure 2.6: 25 bins histogram showing the distribution of the coverages within LIP regions of the TE440 dataset. By looking at this distribution, it is possible to see that there is a prevalent amount of low coverage values.

CAID Binding dataset

The CAID Binding dataset is composed of 78 protein sequences, providing, in the same format shown above for the TE440 dataset, a FASTA file containing DisProt IDs, complete protein sequences and binary labels for each protein; in this dataset positive examples are sequences that establish interactions and are located within disordered regions [55]. The CAID Binding dataset contains a total of 54568 positive amino acid positions (86.94 % of the total positions) and 8196 negative amino acid positions (13.06 % of the total positions). The distribution of the sequence coverages (in terms of LIP positions content) is shown in Figure 2.7.

The same procedure described above was used here to add embeddings for each amino acid position in this dataset. Also, even in this case, it was necessary to delete 1 protein sequence for the embedding calculation (corresponding to 1.30 % of the test set), due to the 4000 length limitation. The deleted protein is the replicase polyprotein 1a (UniProt ID P0DTC1), with a length of 4405 amino acids.

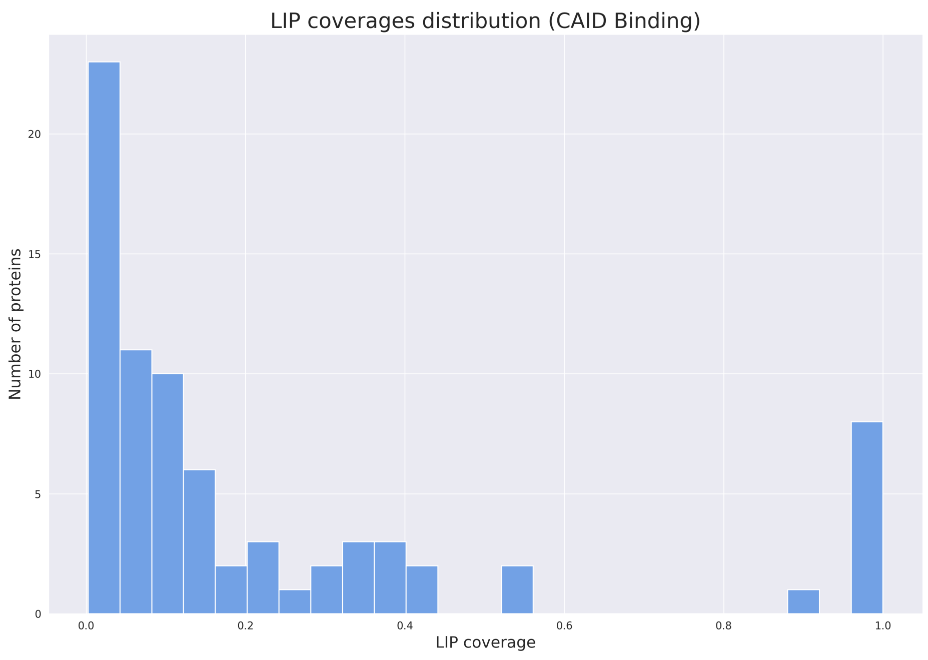


Figure 2.7: 25 bins histogram showing the distribution of the coverages within LIP regions of the CAID Binding test dataset. Again, as was seen for the TE440 datasets, it is possible to see that there is a prevalent amount of low coverage values.

2.2 Models

This section is dedicated to discuss in detail the convolutional neural network model used for this project to classify LIP regions. The chapter provides several theoretical concepts, related to Convolutional Neural Networks and to deep learning in general, showing the logic behind them and their main features. The section continues with a section dedicated to explain the data preparation for the model and a section presenting and discussing the CNN model proposed for LIPs classification.

The specific characteristics of the data that were used as input for the binary classification task of predicting LIP amino acid positions within protein sequences justifies the decision to use a CNN model. CNNs, as it is explained below, are mostly known for their ability to work with images and, by drawing a parallel with the domain of image analysis, it is possible to see whole protein embedding sequences, which are bidimensional arrays of numbers, as single channel (i.e., monochromatic) images. As with pixels in a regular image, embeddings are related to each other, since every embedding is made in a way that takes into account the specific position of an amino acid relative to its chemical surround in the folded protein. From this perspective, the choice of using a kernel-based convolution mechanism, typical of CNNs, becomes clearer, by pursuing the idea of leveraging the ability of CNNs to capture and interpret local patterns in the inputs that can also occupy very small spaces within the input, as is the case, for example, in tasks of recognizing small details within large and complex images.

2.2.1 The Convolutional Neural Network

Convolutional neural networks (CNNs) are a class of deep learning algorithms that are mostly used for grid-structured data processing tasks, with an architecture that makes them particularly well-suited for tasks involving time series (as one dimensional grids), images (as a two dimensional grids) and videos (as three dimensional grids) [67, 68, 69]. CNNs have been enormously successful in recent years, with vast fields of application and are particularly well-known for their contributions in computer vision and general image processing [70, 71, 72, 73, 74]. This section is dedicated to introduce the main concept of these networks, presenting the general idea behind CNNs and explaining the functioning of their architecture through some important basic notions, with a particular focus on kernel and convolution concepts.

Convolution Operator

In its general form, convolution is a mathematical operation, denoted with the sign $*$ and performed on two real-valued functions, resulting in a third real-valued function; given $x, w : \mathbb{R} \to \mathbb{R}$ [75]:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t-a) \forall t \in \mathbb{Z} \tag{2.7}$$

$$s(t) = (x * w)(t) = \int_{-\infty}^{+\infty} x(a)w(t-a)da \forall t \in \mathbb{R} \tag{2.8}$$

where, in CNNs terminology, x (or, more in general, the first argument) is called **input**, w (or, more in general, the second argument) is called **kernel**, and the output is called **feature map**. While equation 2.8 is for continuous signals, equation 2.7 is for discrete signals [75].

This general convolution definition presented above must be adapted to the CNN paradigm, with some changes [75, 76]:

- firstly, only the equation 2.7 is actually used for computer science applications, where discretization happens;
- secondly, the CNN paradigm requires the convolution operator to be applied to tensor objects (which have the form of multidimensional arrays), thus transforming the infinite sums to finite sums of tensor

entries, by assuming that the convolution is zero everywhere outside the finite set of points for which the values are stored, getting in this way finite sums of finite arrays entries;

- finally, convolution operator can be used on more than one axis at a time (as happens, for instance, in the case of two dimensional kernel used on image inputs).

With these premises, the CNN's two-dimensional kernel convolution is defined as follows [75]:

$$S(i,j) = (I * K)(i,j) = \sum_{i=1}^{m} \sum_{j=1}^{n} I(i-m, j-n)K(m,n) \tag{2.9}$$

where S is the CNN convolution, K is the kernel, I is the input and the two sums represent the kernel application along two distinct dimensions. The output is a feature map of the input that is able to highlight specific local aspects of it, like edges and corners in convolutional image processing [75].

It is important to point out the fact that, for performance efficiency reasons, in many neural network libraries (including PyTorch, which is the library used for this project) the convolution is implemented in a slightly different form, i.e., the cross-correlation form for two dimensional kernels [77]. Therefore, the convolution operation becomes a discrete matrix multiplication [75, 76]:

$$S(i,j) = (I * K)(i,j) = \sum_{i=1}^{m} \sum_{j=1}^{n} I(i+m, j+n)K(m,n) \tag{2.10}$$

This form is the equivalent to the convolution shown above, but without flipping the kernel, giving, in this way, the ability to CNNs to be able to learn the proper kernel values in the proper place in an invariant way with respect to the flipping; thus, a model with kernel flipping is able to learn a kernel that is flipped in comparison to a kernel learned by a model without kernel flipping [75].

The architectural concept of CNN kernel is at the heart of CNNs and draws inspiration from images properties and from the organisation of the animal visual cortex neuroanatomy, where individual neurons respond to stimuli in a limited region of the visual field, known as the receptive field [75]. The receptive field in a CNN refers to the region in the input image that a particular CNN's feature detector (i.e., the kernel) is able to apply, in the sense of neural connections, enabling the network to focus on small, local features first, gradually building up to more global and complex features, which provide CNNs their exceptional feature learning and recognition capabilities; this specific kind of paradigm leads to three important properties of CNNs [75]:

- sparse interactions, that is the phenomenon of having a sparse connectivity due to the receptive field induced by the application of a small kernel. Inputs (like, for instance images) are usually high-dimensional collections of entries and hidden layers in traditional fully connected networks are usually larger than the input size. Thus, even for a shallow network, the number of weights would be huge. This concept raises an obvious practical problem, in terms of computational memory and time required for processing. In this context, CNNs adopt a sparse interaction system, due to the smaller size of their kernels in comparison to the input. By using a kernel of a fixed small size, it is possible to process a very big input data without loosing the ability to detect small meaningful features;

- parameter sharing, that is the fact of using the same parameters for more than one function in a model. While traditional neural networks use each entry of the weight matrix exactly one time, by multiplying it for the whole input, CNNs uses few entries of the kernel at every position of the input, reducing the storage requirement and allowing to apply to one input a value of weight that is tied to a value of weight applied somewhere else;
- equivariant representations to translation, i.e., a property caused by the form of parameter sharing that allows the ability of keeping a stable interpretation of an image under translations (even if convolution is naturally not equivariant to changes in scale or rotation).

These three concepts lead to a deep learning model that has a higher computational efficiency, both in terms of memory and computational time, as well as showing better statistical efficiency, being able to capture the proximity correlation typical of the type of data for which these networks are designed (as happens, for instance, for nearby pixels in images), while remaining also able to capture even small important features in very big inputs [75, 76].

2.2.2 Data Preparation

This section explains in detail how the data were prepared for the CNN model, showing techniques and strategies used for generating input data.

The original total training dataset was split into training and validation sets, so as to provide a reliable basis for grid-searching, with the aim to tune and optimize specific chosen hyperparameters. For the validation step, the k-fold cross validation method was preferred over the hold-out method, since the higher reliability of the former over the latter was demonstrated on many tasks [78, 79]. To correctly build the dataset using a k-fold cross validation, it was of absolute importance to take into account the fact that LIP sequences are uncommon within proteins. Therefore, as widely shown above, the dataset contains far more non-LIP amino acid positions than LIP amino acid positions. For this reason, the preparation of the dataset for training and validating the CNN model was carried out using a 5-fold stratified cross validation, which is a version of the k-fold cross validation that is specialized for unbalanced datasets. This technique pays particular attention to preserve, in each fold, both an equal proportion of positives and negatives and an equal distribution of the embedding values among the folds, with reference to the source dataset. Also, by choosing $k = 5$ it was possible to ensure that the absolute number of positive cases per each fold was not too small. To do so, the Scikit-learn Python external library was used, since, as stated in the library user guide [80], its `StratifiedKFold` function is designed to generate folds that preserve both the proportions between positives and negatives and to get also a distribution of the examples of the folds that is as close as possible to the one of the source dataset.

At this point, the input units for the CNN model were whole sequence-embeddings arrays of shape $n \times 1024$, for each sequence of length n, while the CNN model should have generated outputs consisting in single arrays of length n, containing binary values for labeling each amino acid position in the sequence. Considering the need for the CNN inputs to be uniform in their sizes [76], it was necessary to deal with the variability of n, by applying 0-padding to the embedding sequences before obtaining the final input for the model. This technique allowed to add zeros to enlarge the array to a specific maximum value, which was uniform for all the dataset. unfortunately, because of hardware limitations, it was necessary to adopt a strategy to avoid incurring a memory error, since the padding resulted in large data, as all sequences would be enlarged to reach the length of the longest sequence. By setting a limit threshold of 1000 amino acids it was possible to ensure that padding did not produce inputs with a too large size, obtaining unit input sequences no longer than 1000 amino acids and, excluding in this way, longer than 1000 amino acids sequences. This steps removed a total of 423 protein sequences, corresponding to the 4.85 % of the total training set. In this way, it was possible to keep a reasonable length limit without loosing information, since all sequences larger than this threshold were discarded. When applying the 0-padding, it was important to keep in memory the original length of the sequence, since, when evaluating the quality of the output, it is necessary to mask the portions of the input that do not contribute to the prediction. For this reason, masks have been calculated for each protein sequence. Those masks are applied at the time of evaluating the output, in order to allow the validation of only those positions that correspond to true amino acids. The last step consisted in a batch splitting of the training and validation dataset, for avoiding memory errors feeding whole folds to the model and to stabilize the training process by providing a more consistent estimate of the model gradient [81].

The final dataset got with this processing step consisted of 8299 proteins sequences for the training and validation set (where, for each fold, the algorithm selects 6639 proteins for the training step and 1660 proteins for the validation step), 436 protein sequences for the TE440 test set, and 77 protein sequences for the CAID Binding test set. The input units of these sets are padded array of embeddings with a shape of 1000×1024.

2.2.3 Proposed CNN Model for LIPs Detection

This section finally presents the deep learning model designed for this LIP prediction binary classification task. The model was fully implemented using PyTorch 2.1.2 (cu121) in a Jupyter Notebook environment (version 4.1.5), for a better control of the workflow during code prototyping. This section is dedicated to a meticulous explanation of the structural features of the model, proceeding then to explain in detail its functioning, the problems related to its development, and the final chosen solutions to overcome such problems.

Model Architecture

The architecture of the CNN model is inspired by the work done by Bernhofer & Burkhard (2022) [82], which made a CNN model for predicting transmembrane proteins using ProtTrans embeddings as inputs. In this case, the authors exploited the properties of the convolution operator for a feature engineering process aimed to reduce the dimensionality of the embedding features through the application of the pointwise convolution. Then, they use parallel convolutions and output concatenation to discriminate distinct secondary structures in the proteins.

Following a similar line of thought, the main idea was to develop a network that was able to reduce the embedding dimensionality and, then, perform distinct identification tasks for ordered and disordered LIP regions, so as to take advantage of the separation of these two groups. All the convolutional blocks of the model consist of a first convolutional layer, followed by the application of a ReLU activation function (to introduce non-linearity) and a final batch normalization layer. In particular, the CNN model workflow is the following (the complete diagram of the model architecture is presented in Figure 2.8):

- the model starts with a first convolutional block that uses a pointwise convolution with kernel size 1 as first layer;
- after the first pointwise convolutional block, there are two depthwise convolutional blocks that are placed in parallel; both these blocks take as input the output of the previous pointwise convolutional block, using a kernel size of 9 and 19, i.e., the average length of the ordered LIP regions and the disordered LIP regions, respectively [82];
- at this point, the outputs of the three distinct convolutional blocks (the pointwise one and the depthwise two) are concatenated together;
- a dropout with a probability is applied;
- finally, a last plain pointwise convolutional layer is applied to the concatenated results for generating the output.

During the execution of the model, a masking of the 0-positions added with padding is used for taking into account only the real amino acid positions during the calculation of the loss and the evaluation metrics. The specific architecture and workflow features of the model presented above are discussed in detail below.

Weight Initialization

The weights for the three convolutional layers were initialized using a Xavier uniform initialization, which is a state-of-the-art technique commonly used for initializing weights and presented by Glorot & Bengio in 2010 [83]. The main idea of this technique is of taking into account the number of input and output units of each layer for establishing the size of the random initialization, allowing both activations and gradients to flow efficiently throughout the forward- and back-propagation phases [83, 84, 85].

In this project, the Xavier initialization is provided by PyTorch [86], in which the input tensor is filled with values sampled using a Xavier uniform distribution $\mathcal{U}(-a, a)$, where:

$$a = \text{gain} \times \sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}} \tag{2.11}$$

with gain being an optional scaling factor, while fan_in being the number of input signals that feed into the layer and fan_out being the number of output signals that come out of the layer, respectively. The way in which the Xavier uniform initialization is defined allows to maintain the variance across layers during the training of the model, allowing a good information flow through the network and preventing the gradient from becoming too large or small, reducing, in this way, the risk of incurring in exploding or vanishing gradients [87].

Pointwise Convolution

Pointwise convolution is a special type of convolution, which uses a kernel of size equal to 1 [88]. This kind of convolutional layer is used twice in this project. The first pointwise convolution is used with the aim of engineering the embedding features, by reducing them from 1024 to a lower number (decided by the results of a grid-search). The second use of this convolution is at the end of the model, where it combines the outputs for each of the three previous layers (Figure 2.8), generating in this way the binary scores for each of the two classes.

Depthwise Convolution

Depthwise convolution is a convolutional layer that applies a bidimensional kernel to each channel independently, preserving, in this way, individual feature information within each channel, while reducing computational cost compared to standard convolution and generating sliding weighted sums for each dimension [89]. In the case of this project, two parallel depthwise convolutional layers (as can be seen in Figure 2.8) with kernel sizes of 9 and 19 were used, respectively corresponding to the average length of order LIPs and disordered LIPs. The input channels for these depthwise convolutional layers are the size-reduced feature embedding outputs of the first pointwise convolution.

Optimizer and Regularization

The chosen optimizer for this model, i.e., the AdamW algorithm, is a stochastic optimization method that modifies the classic weight decay implementation of the Adam optimization algorithm [90]. Before exploring AdamW algorithm, it is important to provide an appropriate introduction to the Adam algorithm, on which AdamW is based.

The Adam algorithm (whose name derives from "adaptive moments") [91] is an optimizer that follows the main idea of combining the properties of momentum and RMSProp [92], adjusting the learning rate using different parameters from estimates of first and second moments of the gradients and correcting (using bias) first and second-order moments, due to their initialization at the origin, ensuring accurate updates from the beginning [91]. The weight update at time step $t \in \mathbb{N}$ from the parameter θ_{t-1} to the parameter θ_t are performed as follows [91]:

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.12}$$

With η being the learning rate, ε a number representing a small quantity to prevent zero division, and β_1 and β_2 forgetting parameters. In this context, $\hat{m}_t$ and $\hat{v}_t$ are the first and second order moments, respectively, obtained as follows. Considering the loss function J, the update equations for biased first moment estimate m_t and bias-corrected first moment estimate $\hat{m}_t$ is:

$$\begin{cases} m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla_\theta J(\theta_t) \\ \hat{m}_t = \dfrac{\theta_t}{1 - \beta_1^t} \end{cases} \tag{2.13}$$

while, the update equations for biased second raw moment estimate v_t and bias-corrected second raw moment estimate $\hat{v}_t$:

$$\begin{cases} v_t = \beta_2 v_{t-1} + (1 - \beta_2) \nabla_\theta J(\theta_t)^2 \\ \hat{v}_t = \dfrac{v_t}{1 - \beta_2^t} \end{cases} \tag{2.14}$$

where $\nabla_\theta J(\theta_t)^2 = \nabla_\theta J(\theta_t) \odot \nabla_\theta J(\theta_t)$.

The AdamW algorithm is based on the idea that L_2 regularization (i.e., the weight decay) is not effective in the Adam algorithm, proposing thus a decoupled weight decay as a more efficient alternative that improves regularization. This different implementation separate the weight decay from the optimization steps [90]. In particular, AdamW adjusts the equation 2.12 its weight decay term to appear in the gradient update [90]:

$$\theta_{t+1,i} = \theta_{t,i} - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} + \lambda \theta_{t,i} \right) \tag{2.15}$$

where λ is the weight decay coefficient, applied directly to the parameters θ_t, alongside the adaptive learning rate update. This modification allows for a more consistent and effective implementation of L_2 regularization; by correcting the integration of weight decay, AdamW often achieves better generalization performance compared to the traditional Adam optimizer, especially in complex models and datasets [90].

For the purpose of this project, $L2$-regularization plays an important role, since the stability of the CNN model was an essential part of the implementation (as will be discussed later in the paper). For this reason, it was decided to test both Adam and AdamW, in order to choice the best one in terms of regularization performance.

Loss Function

Before starting to talk about the loss function, it is important to introduce the important concept of cross-entropy. Cross-entropy is defined as a measure of the difference between two probability distributions P and Q over the same set of events X, defined as follows [93]:

$$H(P, Q) = -\sum_{x \in X} P(x) \log Q(x) \tag{2.16}$$

where $P(x)$ is the probability of the event x. Cross-entropy function is widely used as a loss function in machine and deep learning and corresponds to the multinomial logistic loss applied to the outputs of a neural network, when the softmax is used, allowing in this way to address the difficult computational problem of directly minimizing the zero-one classification loss [94].

In particular, for a classification problem with C classes, the cross-entropy can be used as a loss function as follows[75]:

$$H(y, \hat{y}) = -\sum_{c=1}^{C} y_c \log \hat{y}_c \tag{2.17}$$

where y is the actual distribution of the labels, that can be represented as a one-hot encoded vector y_c with a 1 in the correct class entry position, and zero in all the other positions, while $\hat{y}$ is the predicted probability distribution over classes, with $\hat{y}_c$ being the probability of class c.

In the special case in which $C = 2$, the formula becomes [95]:

$$H(y, \hat{y}) = -y \log(\hat{t}) - (1 - y) \log(1 - \hat{y}) \tag{2.18}$$

The loss function used in this project is the Binary Cross-Entropy (BCE), using the `BCEWithLogitsLoss` PyTorch implementation. As mentioned in the official documentation [96], this particular implementation integrates the BCE into a sigmoid function, in order to provide a version with a higher numerical stability.

Batch Normalization

Batch normalization is used to ensure stability and consistency to the model during the learning process, accelerating it by addressing the problem of internal covariate shift [97]. This issue occurs when the distribution of each layer's input changes during the training phase, slowing down the convergence. With batch normalization the batch statistics are used to adjust the activations of each layer [97]. In the case of this project, the PyTorch 2D batch normalization for an input x to generate an output y used the expected values $\mathbb{E}$ and the variance Var of the input, calculating the normalization as follows [98]:

$$y = \frac{x - \mathbb{E}[X]}{\sqrt{\mathrm{Var}[x] + \varepsilon}} \cdot \gamma + \beta \tag{2.19}$$

where γ and β are learnable parameters with the same size of the input x, while ε is a constant, which is added to the batch variance for numerical stability. In an effort to maximize the quality of the results, batch normalization was put after ReLU (Figure 2.8) [99].

Activation Functions

The activation function used in this project is the Rectified Linear Unit (ReLU), a very important activation function, widely used in many neural network architectures due to its simplicity and efficiency. Such function is defined as [100]:

$$\mathrm{ReLU}(x) = \max_{\mathbb{R}}\{0, x\} \forall x \in \mathbb{R} \tag{2.20}$$

Thanks to this definition, it is possible to activate a neuron only if its input is strictly positive, setting to zero all negative values, introducing in this way non-linearity and allowing the network to quickly learn complex patterns while mitigating the gradient problems that are typical of other activation functions [100].

Another activation function is the sigmoid, used at the end of the model, before the output, but right after the application of the binary cross-entropy loss function. The sigmoid function $\sigma : \mathbb{R} \to [0, 1]_{\mathbb{R}}$ is defined as follows [75]:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \forall x \in \mathbb{R} \tag{2.21}$$

Considering the codomain of the function and its trend, the importance of this function in the context of neural networks is evident, since it maps input values in the range between 0 and 1, thus allowing the output to be interpreted as probabilities, making this function ideal for binary classification tasks, contributing also to introduce non-linearity into the network, to enhance learning of complex relationships between inputs and outputs [75]. This function was for generating probability scores which, when rounded to unity, provided a list of binary labels useful for classifying LIP nad non-LIO amino acids.

Dropout

The dropout layer was placed at the end of the model. The dropout technique transforms into zeros random feature channels in order to make the model to better generalize, by imposing the deactivation of a random subset of neurons in the network, including their incoming and outgoing connections, during the training phase, multiplying the output of each neuron by a binary mask [101]. The aim of this deactivation is to force the model to learn a more robust set of features that are not based on a too small set of neurons, introducing in this way a redundant representation of the data that lead to a more distributed learning process for enhancing generalization [101]. With such an approach, it is possible to emulate a large ensemble of networks with shared weights and slightly different architectural configurations, reducing in this way the sensitiveness of the overall network to specific characteristics of the training data [101].

CNN Model Workflow

The model described above generates per-residue predictions for each input sequence of length n, generating a binary-valued array of the same length n, after masking the padded entries.

Before starting the execution of the model code, a specific numerical seed was set of 4 for each randomness point within the code. A fixed random seed ensures the reproducibility of the code, avoiding instability in results due to randomness, ensuring, in this way, the consistency needed to be able to repeat the execution of the code. This is important, since it makes possible to verify the results, comparing the performance of different models, and obtaining a baseline to be able to properly debug the code.

The CNN model underwent a stage of grid-search, for evaluating the best set of hyperparameters to use in the final version. This tuning process focused on two specific parameters, the size of the batches, in which the training and validation sets are split, and the number of channels for the output of the first pointwise convolution layer used for features reduction. The grid-search was manually implemented with the aim to maximize the F1-score and allowed to select the following parameters: a reduction from 1024 to 512 of the features by the pointwise convolution and a batch size of 16.

The final model was trained for 50 epochs for each fold. The number of epochs was imposed so as to obtain a set with an equal number of loss values and metrics for each fold. For this reason, a canonical early stopping function was not usable, but rather the number of epochs was chosen such that it would guarantee that the loss function of all folds would simultaneously give delta values that were less than a certain fixed threshold for the last consecutive n epochs. This approach emulates a delta and a patience mechanism canonical implementation in a more elaborate way, ensuring both an equal number of epochs for each fold and the convergence of the models for each fold above a certain threshold. In general, it is important to keep the same number of training epochs for each fold, in order to ensure comparability. In this case, the chosen patience and delta thresholds, were of 5 epochs and $10 * -2$, respectively. This decision was imposed by the limitations of the computational resources available to train the model, which would have required too much time to train the model for a larger number of epochs, considering also the need to complete many other computational resources-demanding steps, such as grid-searches on PCA and t-SNE, but also the two distinct 5 fold stratified cross-validation grid-searches on the final CNN model, one for finding the best hyperparameters set for model stability (in the regularization phase), while the other for finding the best hyperparameters set for the best performance (in the canonical tuning phase). The time required to complete all this tasks and also the training of such a demanding model would not have been compatible with the deadline demands of this project. Despite this important limitations, the idea of imposing a manual stopping threshold of 50 epochs was able to guarantee a satisfactory level of loss convergence, as it will be seen in the next chapter.

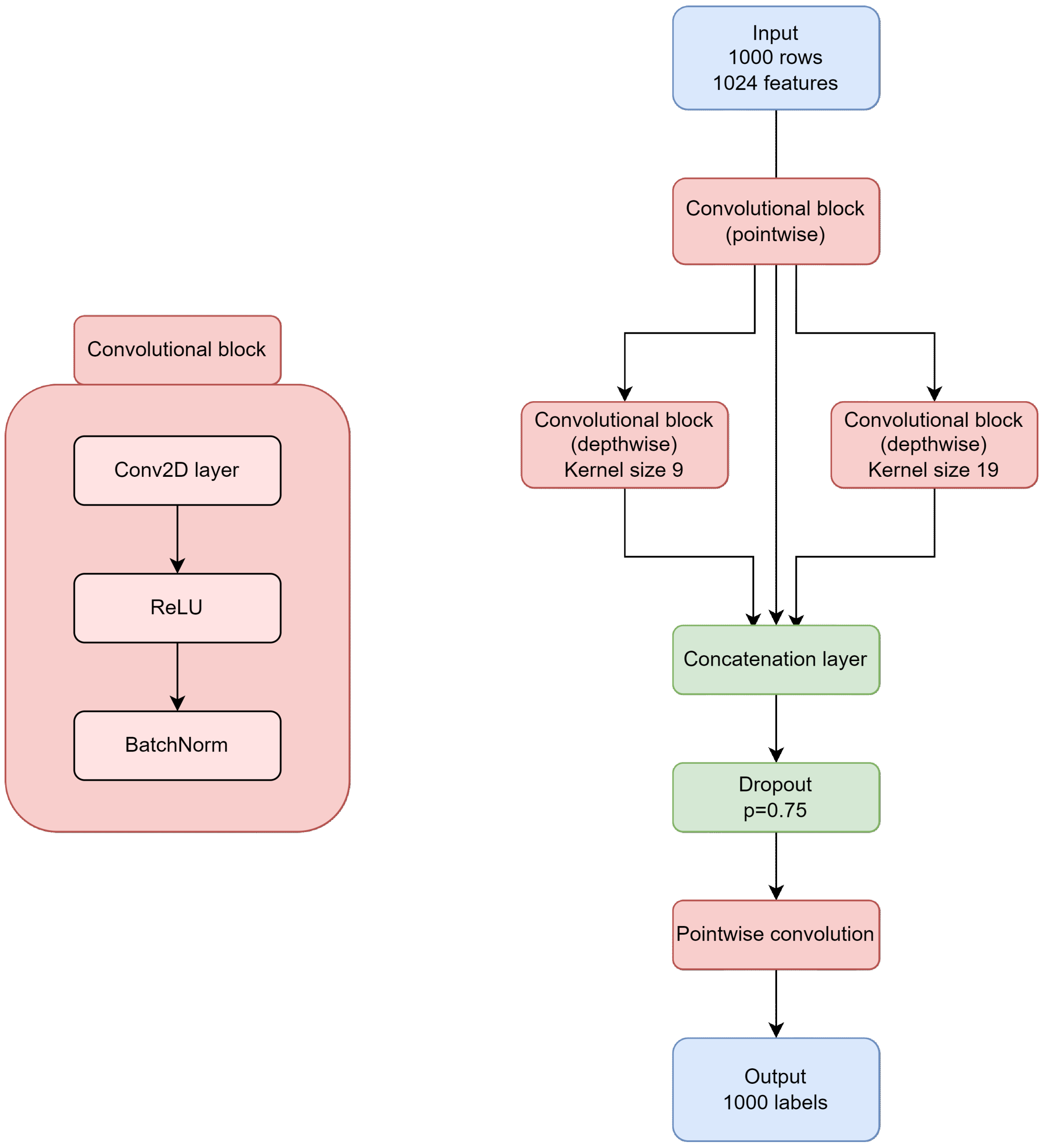


Figure 2.8: Representative diagram of the architecture of the CNN model used in this project; on the left, it is possible to observe the structure of each convolutional block, which include a convolutional layer, a layer where the ReLU activation function is applied, and a final batch normalization layer; on the right, it is possible to see the overall structure of the CNN model, where the two parallel layers and the first pointwise convolution layer can be appreciated.

The Stability Problem

The initial steps of the training were affected by major stability and convergence problems of the model, which concretized, in particular, in several glaring anomalies in the loss function trend behavior throughout the training epochs, both for the training and validation datasets, which showed difficulties in convergence, big fluctuations and oscillations, and overfitting phenomena. To overcome these issues, various tests were conducted in an attempt to act on the model's generalization capabilities and gradient stability during the training phases [97, 75]. Therefore, an important part of the project was devoted to a regularization analysis aimed at obtaining stable behaviors during the training process. For this purpose, the main idea that solved the problem was the introduction of a hybrid-regularized model, associated with a careful choice of the correct optimizer (between Adam and AdamW) and an appropriate value of learning rate.

The regularization was, thus, a combination of several distinct techniques, which did not provide any noteworthy improvements when used alone. The main idea behind this choice was taken from previous studies on effectiveness of hybrid-regularized networks on stability and performance [102, 101]. Following this concept, batch normalization layers were added at the end of each convolutional block [99] and Xavier initialization was used for the weights of each convolutional layer [87], while several models were trained using different weight decay, dropout, and learning rate values while alternating the usage of Adam and AdamW optimizers, monitoring the behavior of the loss function during the epochs on both training and validation sets. With this method, the stability was finally reached by using a specific combination of these techniques, i.e., a dropout with a probability of 75 %, an AdamW optimizer with a weight decay value of 10^{-3}, batch normalization, Xavier initialization, and a learning rate of 10^{-5}.

The Balancing Problem

The PyTorch BCE implementation used in this project is able to take into account the dataset unbalance, by calculating the ratio of negative examples to positive examples and passing it to the function using the `pos_weight` argument. This feature of the `BCEWithLogitsLoss` function was particularly useful for the imbalance problem addressed in this project, allowing the loss to act as if the dataset was balanced.

To further help the model to correctly focus on positive examples, an addition manual balancing was carried out, removing proteins with an amount of LIP regions less than or equal to 10 %, which was the only way to guarantee an effective undersampling, being input whole protein sequences. In this way, it was possible to apply a strong undersampling process that brought to the unbalance magnitude of the dataset from 7.1 % to 34.11 % (in terms of non-LIP amino acid positions out of LIP amino acid positions), while excluding 4693 protein sequences from the total training dataset (i.e., 61.25 % of the sequences). The comparison between this pruned dataset and the whole dataset brought to improvements in model performance, reducing overfitting and increasing the positive examples prediction performance. Indeed, for both cases a the same grid-search with the same parameters were performed, with a significant improvement in performance on both training and validation datasets for the undersampled case.

Metrics and Evaluation

This section presents the metrics and plots used to evaluate the performance of the model during the course of the training epochs (on both training and validation datasets) and on the test datasets in the final evaluation phase, fundamental to catch strengths and weakness of the model. Such metrics take into account the the rates of true positives (TPR), true negatives (TNR), false positives (FPR) and false negatives (FNR), providing a comprehensive measure of performance for the type of problem addressed in this project [95]:

- accuracy score, which is the simplest way to measure the performance of the model, representing the proportion of correct predictions made by the model out of all predictions;
- precision score, that is the number of true positive predictions out of the total number of positive predictions (i.e., true positives and false positives); this metric is usually important when the cost of false positives is high;
- recall score, which is the proportion of the correctly identified actual positives out of all positives (i.e., true positives and false negatives); this metric is usually important when the cost of false negatives is high;
- F1-score, that is the harmonic mean of precision and recall, that provides a single metric to evaluate the balance between precision and recall; this metric represents a useful way to evaluate a model over an unbalanced dataset, when the accuracy score can be misleading;
- Receiver Operating Characteristic (ROC) curve, which is a curve plot that shows the TPR against the FPR at various threshold settings, providing a valuable tool for find the right trade-off between true positives and false positives;
- Precision-Recall (PR) curve, i.e., a curve plot which shows the precision against the recall for different thresholds, providing an excellent way to evaluate the performance of a model over an unbalanced dataset;
- Area Under the Curve (AUC) score, i.e., the measure of the areas under the ROC or PR curves, useful to quantify the overall ability of the model to distinguish between classes, with higher values that determine better performances.

During the learning phase, at the end of each epoch, losses values and several other metrics (i.e., precision score, recall score, and F1-score) were calculated for the CNN model for both training and validation datasets and for each fold. Also, the trained CNN model was tested on both TE440 and CAID Binding datasets, in order to evaluate the model's predictive capabilities on real data outside the training dataset used for the learning phase. In this case, the predictions were evaluated for a range of decision thresholds, so as to maximize the performance metrics.

Finally, ROC and PR curves (with respective calculation of AUC scores) and a histogram of score distribution (highlighting TPR, FPR, TNR, FNR) were plotted for visualizing the model performance.

The CNN model was finally evaluated in terms of computational time required for predicting both the test sets. This calculation was performed by executing the model on both the test sets, using the results for each single input for extracting the most accurate estimate possible. This was possible by determining the time required for every batch, dividing it by the sum of all the amino acid positions contained in such batch (i.e., by excluding the padded positions). This estimation was averaged for all the batches of both the tests sets. To this calculation time must be added the embedding calculation time (calculated in a similar way to how the CNN execution time was calculated) and the time needed to load the model into memory, which together add 0.2 seconds to the model execution time, for a total time required of 0.3 seconds.

During the model evaluation phase on both TE440 dataset and CAID Binding dataset, the best decision threshold for predicting positive and negative examples was chosen in order to maximize the F1-score over 1000 equidistant threshold values in the range $[0, 1]_{\mathbb{R}}$.

3 Results and Discussion

This final section is fully dedicated to results obtained from the model on the training, validation, and test sets, by showing all the performance results calculated using several metrics and by presenting various plots for easing the results' visualization. These results will be presented and explained in a first section and then discussed and commented in a second section, in an attempt to highlight the strengths and limitations of the model through an interpretation that exploits the point of view of the technique (both in terms of data processing and manipulation and in terms of model creation and development) and the strictly biochemical point of view. The direction of this chapter is to provide a proper interpretation of the results, suggesting ideas for future studies aimed at a better interpretation of the problem. All these concepts are presented in the context of the problem, which is known to be very complex and still lacking an efficient and consistent predictive state-of-the-art solution.

3.1 Results

This section is devoted to the presentation and description of the results obtained, which are shown and commented in detail, through the use of tables and plots derived from the measurement of specific metrics, useful for assessing the predictive abilities of the model. The section starts with presenting and describing the results related to the learning phase of the model on both training and validation datasets. The final part of this section is dedicated to expose the results related to both the test datasets used for the final evaluation, i.e., the TE440 dataset and the CAID Binding dataset.

3.1.1 Training and Validation Datasets Performance

Here the results of the model after and during the training are presented, showing a series of metrics and plots useful for understanding the final performance and behavior of the CNN during the training phases. In particular, such information is presented both in global terms in a first part (as an averaged measure of the behavior of each fold) and local terms in a second part (showing the performance data for each fold, individually). All results are described in detailed for a better understanding of what the model did during the training and what has been achieved after it. In particular, the values of losses, precision scores, recall scores and F1-scores were calculated for both the training and validation sets for each training epoch and for each fold.

Averaged metrics

This part shows the global behaviour of the model over the training epochs, quantified through the calculation of specific metrics averaged over the five folds used in the cross-validation. Plots are presented and described in detail. here, the global behavior of the model can be accurately estimated by averaging loss, precision score, recall score, and F1-score of each fold calculated on both the training set and validation set for each epoch.

In Figure 3.1 it is possible to see loss and metrics trends for both training and validation sets, calculated for each epoch and averaged over the five folds of the cross validation. The loss shows a regular and smoothed monotonically decreasing trend, with a training loss that has an error which is lower than the error of validation loss; this is a canonical behavior, since the model tends to better adapt to the training set, providing a certain amount of overfitting, that must be taken low. In this specific case, overfitting was kept low by the hybrid regularization system of the model. The extent of this overfitting can be quantified using a direct measure of the relative distance between training and validation losses. The simple calculation consists in computing the ratio of the distance of the two curves to the value of the validation loss, which returns a relative distance of 4.31 % at the end of the training step, which suggests an acceptable level of overfitting.

The growth slope of the curves related to the metrics shown in Figure 3.1 can be seen as a first simple quantita-

	Negatives	Positives	Global
Precision score	0.9901	0.5679	0.7790
Recall score	0.9709	0.7984	0.8846
F1-score	0.9804	0.6637	0.8221
Loss			0.8334

Table 3.1: Averaged metrics for the training dataset.

tive indication to determine how much the model learned during the training phase. Looking at the plots in figure it is possible to see that, although the matrix curves do not show major differences in height, it is useful to note that the percentage performance of the related metrics shows relatively high values from the outset, maintaining them throughout the training phase. In particular, precision shows a smaller relative growth difference during training, but it continues to slightly increase throughout all training epochs in an upward trend; in contrast, recall has a greater relative difference in its training growth, but quickly reaches a plateau, keeping it throughout the rest of training. As a final result, the F1-score, which is the harmonic mean of precision and recall, continues to show a slight upward trend throughout the training, which is an index that highlights the learning effectiveness of the model.

This general analysis of the metrics shows a good overall performance, which, however, needs further study, to fully understand the separate contribution of positives and negatives to the total values. To do this, precision score, recall score and F1-score were plotted separately. This plots allowed to get two distinct trends, one for the subset of positive examples and one for the subset of negative examples, for for both training and validation sets. These separated metrics are shown in Figure 3.2, in which a very important detail is highlighted. In fact, although the model shows an overall good performance, as seen in Figure 3.1, it is possible to observe in this case, according to the difficulty of the predictive problem of positives (already discussed on several occasions), the presence of a significant distance between the learning curves of the subset of negative examples, whose performance is, in general, very high, and the learning curves of negative examples, whose curves are not able to achieve sufficiently high performances. This aspect is crucial for a correct evaluation of the predicting ability of the model and it is addressed in detail in the next **Discussion**, which is dedicated to discussing the results presented here.

The final metrics measured after training of the model are shown in Table 3.1 and Table 3.2, which show the separated values for negative and positive examples along with the global values derived from the average of these two groups. Here, the gap between the performance achieved for the negative examples and the performance achieved for the positive examples is numerically presented.

	Negatives	Positives	Global
Precision score	0.9894	0.5281	0.7588
Recall score	0.9675	0.7783	0.8729
F1-score	0.9783	0.6293	0.8038
Loss			0.871

Table 3.2: Averaged metrics for the validation dataset.

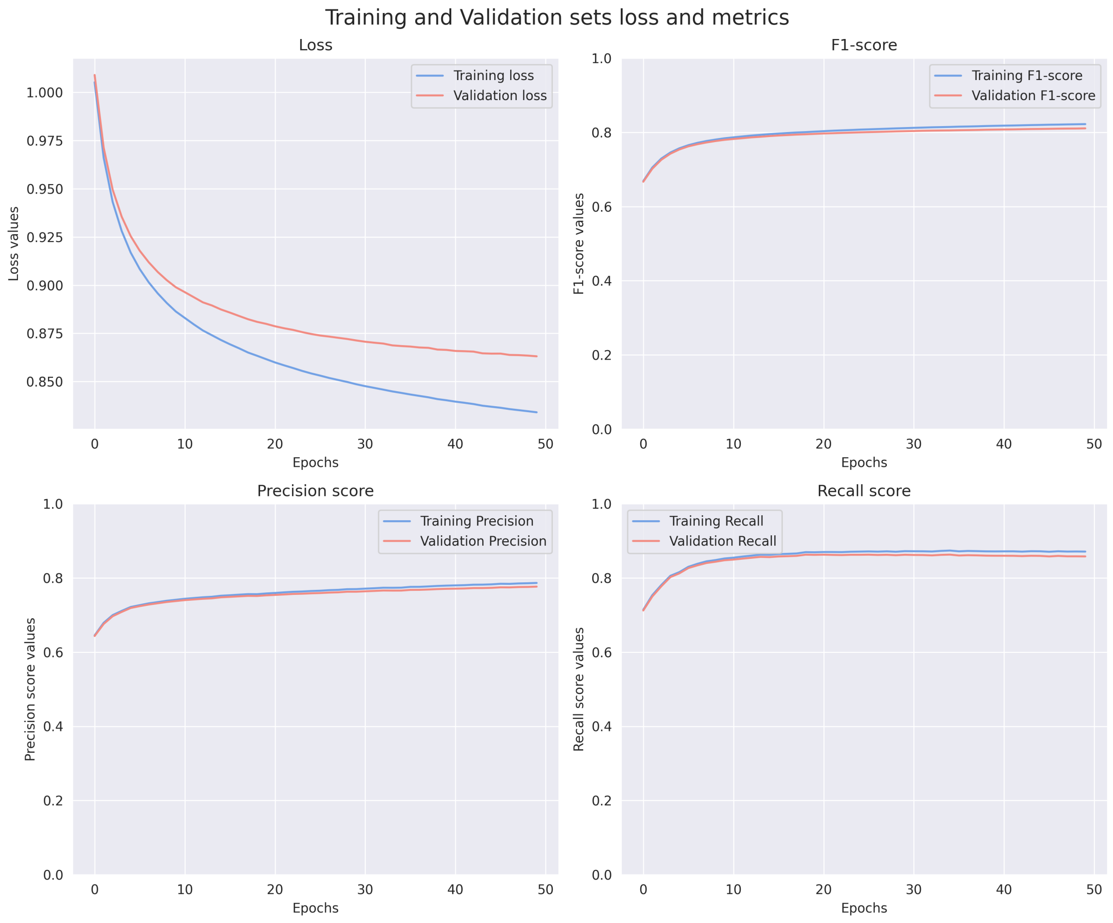


Figure 3.1: Loss and metrics trends for both training and validation sets over the course of the training epochs. All these plots show the mean trend averaged over all five folds of the stratified cross-validation. In the top-left corner, the loss plots, with a 4.31 % distance between the two losses with respect to the absolute size of the validation loss. In the top-right corner, The F-1 score plots, showing a value of 82.21 % for the training set and a value of 80.38 % for the validation set at the end of the training. In the bottom-left corner, the precision score plots, showing a value of 77.90 % for the training set and a value of 75.88 % for the validation set at the end of the training. In the bottom-right corner, recall score plots, showing a value of 88.46 % for the training set and a final value of 87.29 % for the validation set at the end of the training.

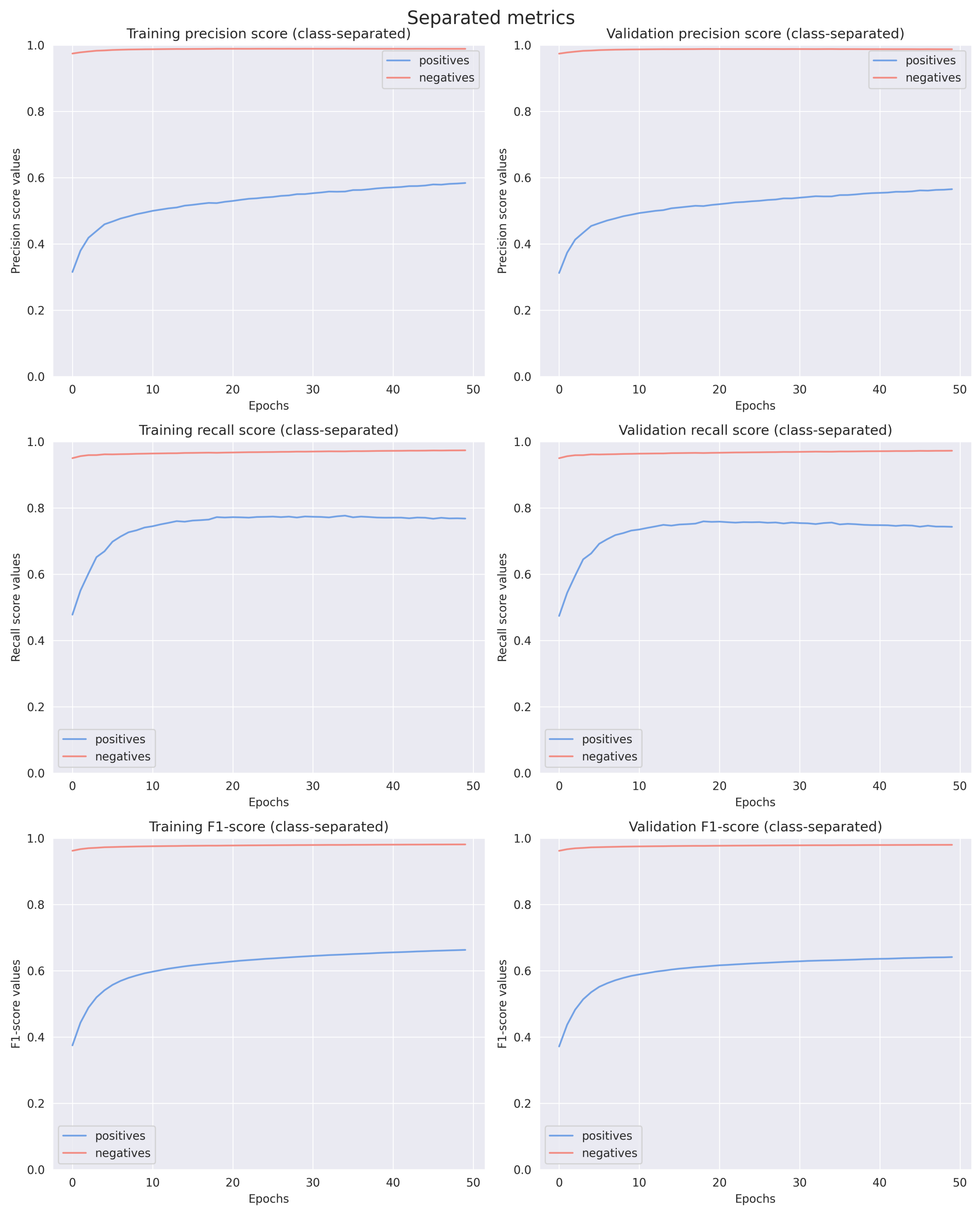


Figure 3.2: Precision, recall, and F1-scores trends for both training and validations sets, separated for positive and negative examples. In the top-left corner, training precision score. In the top-right corner, validation precision score. In the middle-left, training recall score. In the middle-right, validation recall score. In the bottom-left corner, training F1-score. In the bottom-right corner, validation F1-score. The separation in terms of performance between positive and negative examples affects all metrics on each of the two datasets used for training the model.

Fold-specific metrics

This part shows again the results presented above, this time comparing the fold-specific plots with the averaged plots, in order to show the 5 fold-split results from which the averaged plots come from.

In particular, fold-specific loss plots show a stable situation, with a consensus in the behavior of the model on each of them, even if the training losses are slightly more stable than the validation losses Figure 3.3. In general, all plots of the losses show a stable and relatively uniform learning trend from the model on each fold. Analyzing the results in terms of metrics, by looking at the plots of precision score, recall score and F1-score, it is possible to see an even greater magnitude of consensus in the behavior of the curves for each fold, both for training and validation sets (Figures 3.4, 3.5, and 3.6). In this case the behaviour of the CNN model is very consistent on all the folds, which is a suggestion of intrinsic stability, allowed by the implementation of the hybrid and grid-searched regularization described in the previous chapter. The small variation observed among the fold-specific metrics could be due to intrinsic specificities of the dataset or to slightly different data distributions within each fold. In general, this small variations can be marked as minimal and not problematic, considering them in the context of a training dataset and a problem that are particularly complex.

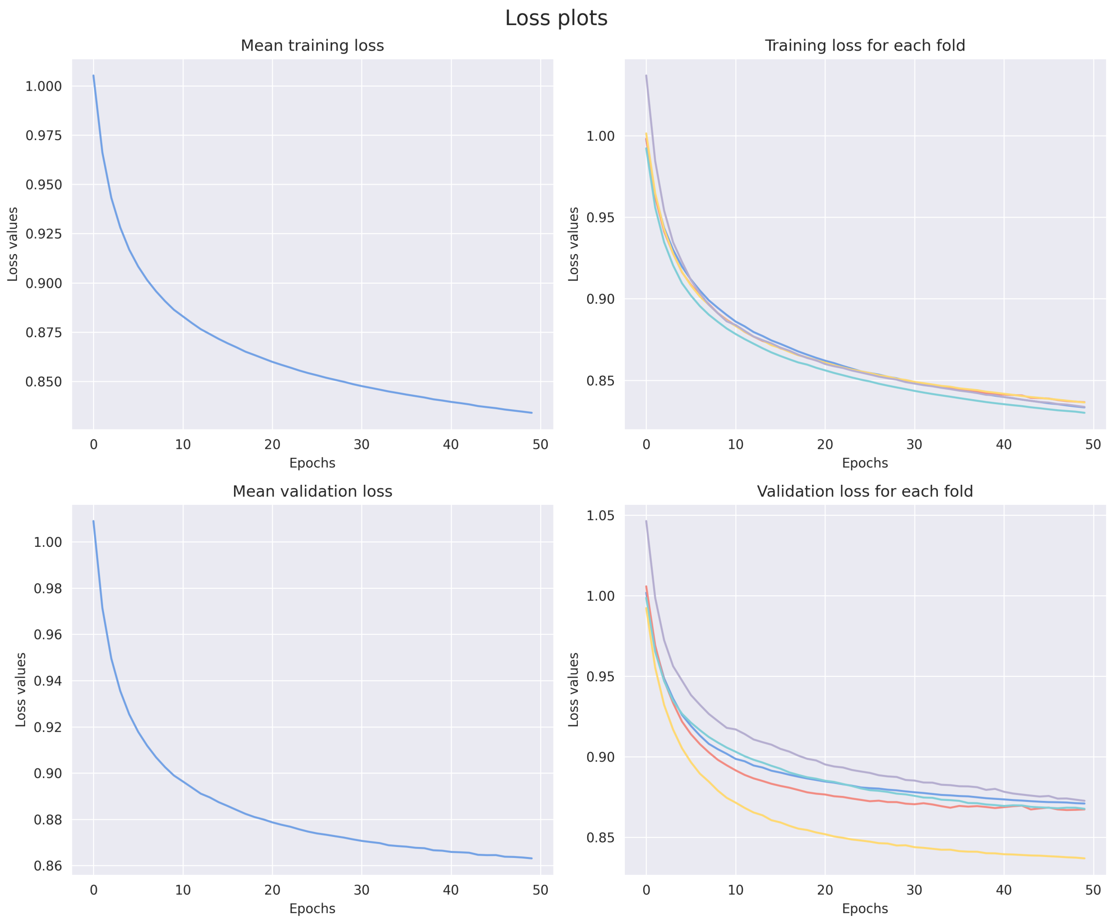


Figure 3.3: Loss trends for all the five folds of the cross-validation and their average behavior over the course of the training epochs. In the top-left corner, the averaged loss for the training set. In the top-right corner, the losses for each fold for the training set. In the bottom-left corner, the averaged loss for the validation set. In the bottom-right corner, the losses for each fold for the validation set.

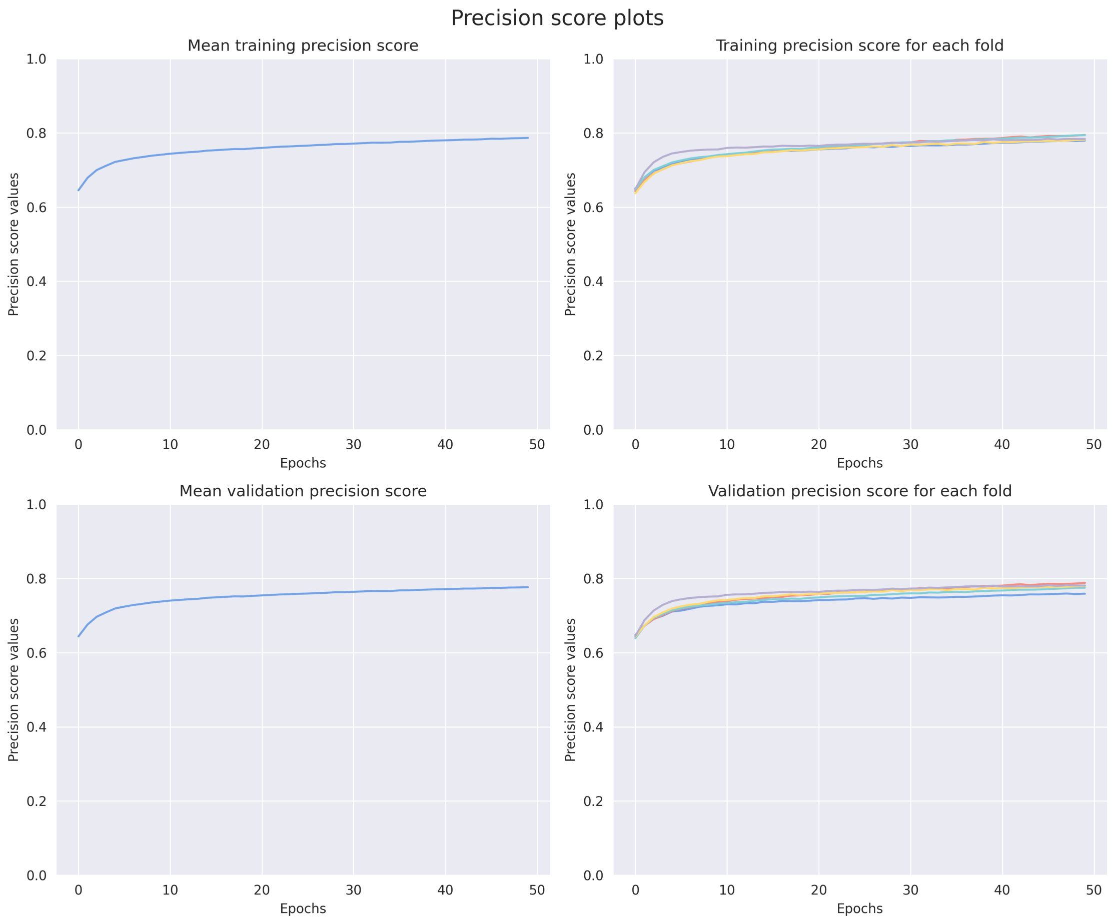


Figure 3.4: Precision score trends for all five folds of the cross-validation and their average behavior over the course of the training epochs. In the top-left corner, the averaged precision score for the training set. In the top-right corner, the precision score for each fold for the training set. In the bottom-left corner, the averaged precision score for the validation set. In the bottom-right corner, the precision score for each fold for the validation set.

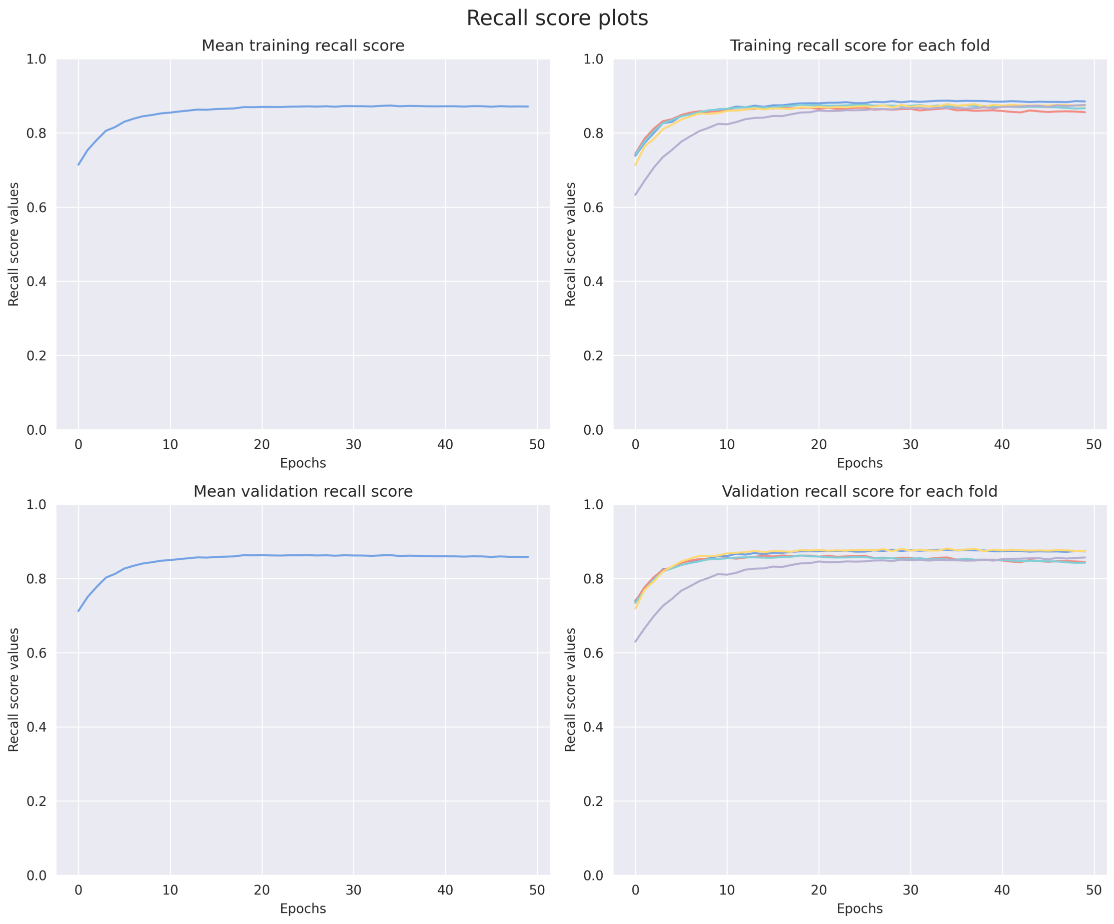


Figure 3.5: Recall score trends for all five folds of the cross-validation and their average behavior over the course of the training epochs. In the top-left corner, the averaged recall score for the training set. In the top-right corner, the recall score for each fold for the training set. In the bottom-left corner, the averaged recall score for the validation set. In the bottom-right corner, the recall score for each fold for the validation set.

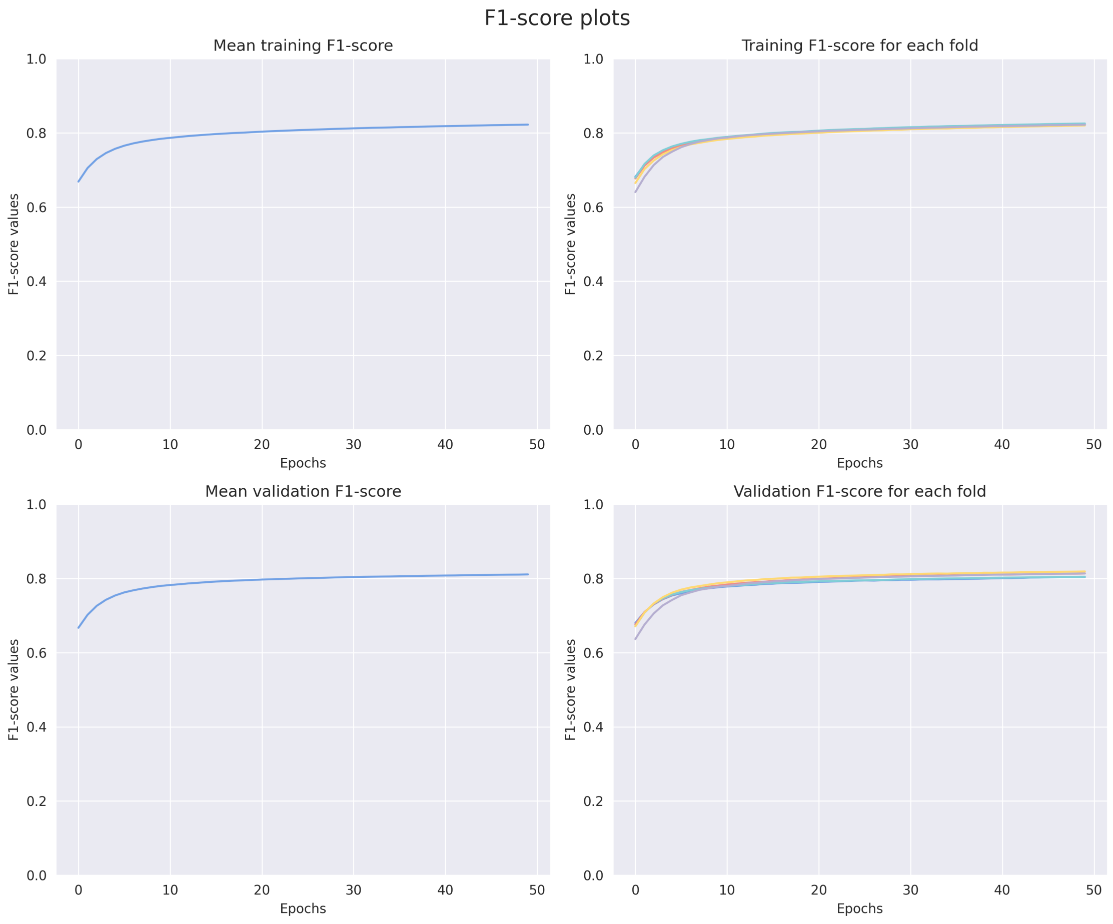


Figure 3.6: F1-score trends for all five folds of the cross-validation and their average behavior over the course of the training epochs. In the top-left corner, the averaged F1-loss for the training set. In the top-right corner, the F1-loss for each fold for the training set. In the bottom-left corner, the averaged F1-loss for the validation set. In the bottom-right corner, the F1-loss for each fold for the validation set.

3.1.2 Test Datasets Performance

After the 5-fold stratified cross-validation, the CNN model was tested on TE440 and CAID Binding datasets. In this section the test set results are presented, showing the final precision score, recall score and F1-score for both positive and negative examples, while bringing also their final aggregated values, providing the best threshold for optimizing these metrics on both test sets. Also, several plots for each of the two sets are shown, i.e., ROC and PR curves, with the calculation of underlying areas of both the curves, and a histogram for showing the distribution of the scores.

The model showed a general prediction speed (calculated on both datasets) with an average of about 0.1 second for outputting the labels of a protein sequence of 1000 amino acids, which can be considered a good result, if compared with the results shown by other predictive models in CAID Figure 1.12.

TE440 Dataset Results

The TE440 test dataset results are presented and described in this section, through proper tables and plots.

The metrics obtained after the evaluation of the model on this dataset are shown in Table 3.3. While performances for negatives examples are very high, performances for positives examples show an important drop, both in precision and recall. So, the problem regarding the reduced performance on the subset of positive examples, which is well highlighted in the plots for training and validation datasets (as already seen in the previously), predictably also appeared in the test set evaluation phases, although in a more pronounced manner. This higher pronunciation is a symptom of overfitting on the training and validation sets for this specific prediction sub-task, due to a problem with the data, with the model, or both.

The ROC curve derived from the prediction scores of the TE440 set (Figure 3.7) shows an area under the curve (AUC) of 73.92 %, which is moderately good. The black dashed line shown in the figure is a reference for the ROC curve. It shows, indeed, how the model differs in performance from a random prediction context. In particular, a ROC curve above this line indicates a model with the ability of predicting a set of samples that is better in comparison with the prediction ability of a hypothetical random predictor, which does not bring any value in terms of predictive learning. The ROC curve is not so close to 1 (which represents the ideal situation), but it is also significantly better than a random model (which would have a AUC of 0.5), suggesting a model with a fair ability of correctly identifying true positive examples. The curve is smooth and with a steady increasing plot of the true positives rate as the false positives rate increases too, indicating a consistent improvement of the model at different decision thresholds. However, the PR curve (Figure 3.8) shows an area of 15.40 %, which is quite low, result that is in line with the high amount of false positive examples compared to false negative examples. The high recall maintained suggests a good ability to identify most of the true positive examples, however the precision drop is very steep in relation to the increase in recall. This is a common situation in datasets with a significant imbalance, where true positive examples are rare compared with the false positive examples.

The distribution histogram for the scores of each amino acid position evaluated on the TE440 test dataset Figure 3.9 provides a better visualization of this situation, showing a maximized decision threshold of 94 %. Indeed, this plot shows a very good ability to find negative examples, with a very low number of false negatives. In particular, the model was able to identify 247557 true negative examples (i.e., 96.86 % of all negative predicted examples), which are shown at the left of the red line, against 8024 false negative examples (i.e., 3.14 % of all negative predicted examples), which are shown at the right of the red line. On the other hand, as already shown above, the problem of false positives is considerable, since, in trying to effectively identify positive examples, the model was able to only predict 2663 true negative examples (i.e., 18.79 % of all positive predicted examples), while the other 11506 examples were false positives (i.e., 81.21 % of all positive predicted examples). The scores distribution histogram shows, also, a clear reduction in candle heights as it proceeds from score 0 to score 0.5, after which the candle heights start to increase again, even if with a smaller absolute value of the slope. This behaviour is consistent with a binary prediction task, although the reduced steepness of the slope indicates a presence of too many scores around 0.5. This fact is a symptom of difficulties of the model in predicting certain examples.

In summary, the results obtained for this test dataset show a general good classification ability of the model, but with an important problem in correctly identifying positive examples, both in term of precision and recall. (Table 3.3).

Figure 3.10 shows the fraction of true positives, true negatives, false positives, and false negatives for several chosen thresholds on the whole TE440 test set. Predictions are done by classifying as positives all the example with a score which is greater or equal than the considered threshold. By increasing the threshold, the number of false positives and true positives decrease, while the number of false negatives and true negatives increase, until no positives are predicted for a decision threshold of 1.

	Negatives	**Positives**	**Global**
Precision score	0.9686	0.1879	0.5783
Recall score	0.9556	0.2492	0.6024
F1-score	0.9621	0.2143	0.5882
Accuracy score			0.9276
ROC-AUC score			0.7392
PR-AUC score			0.1540

Table 3.3: Evaluation metrics for the TE440 test set.

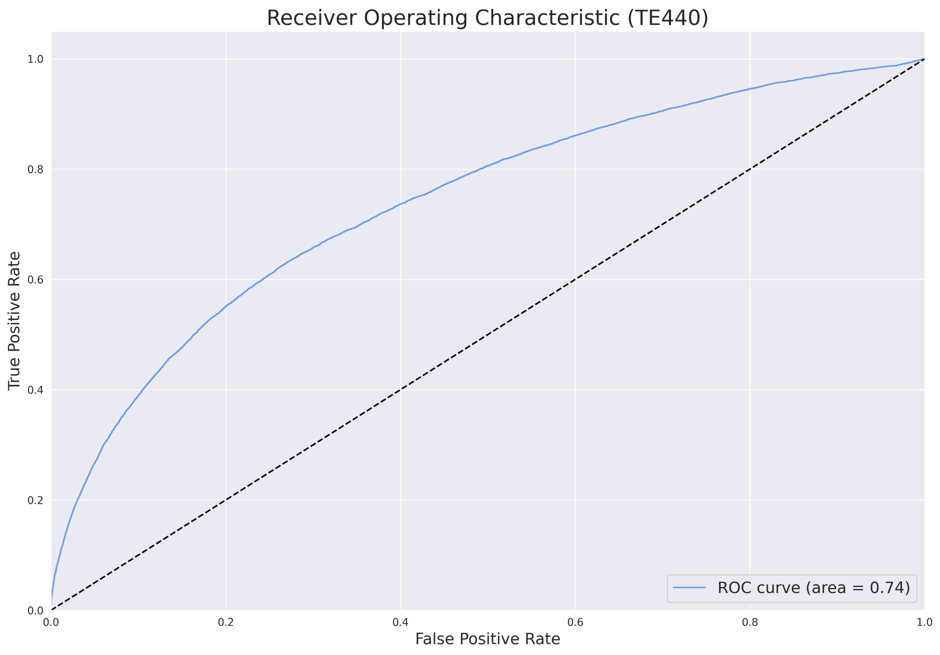


Figure 3.7: ROC curve for the TE440 test set, showing a ROC-AUC score of 73.92 %.

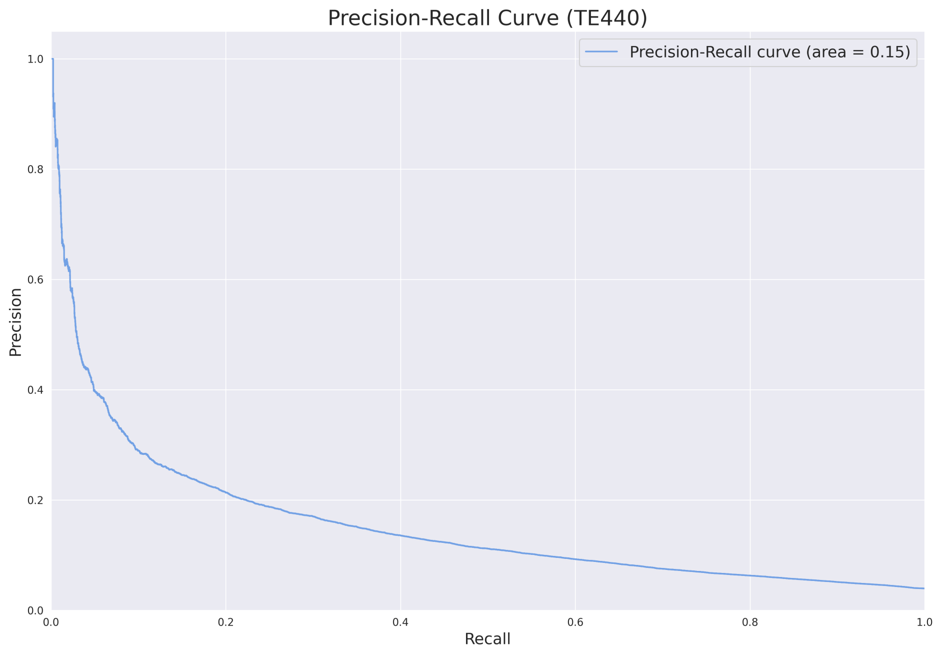


Figure 3.8: PR curve for the TE440 test set, showing a PR-AUC score of 15.40 %.

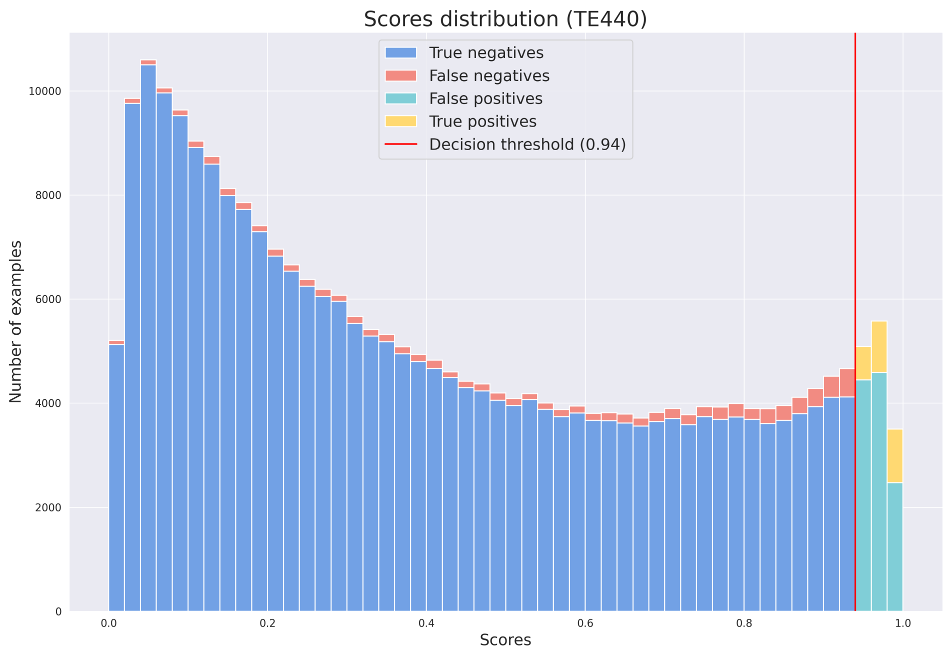


Figure 3.9: Histogram plot showing a distribution for the TE 440 test set for the maximized chosen threshold of 0.94, with a split of each bar that shows the four categories of true negatives (light blue bars), false negatives (pink bars), false positives (light green bars) and true positives (yellow bars). The red line corresponds to the decision threshold and divide the predictions of the model.

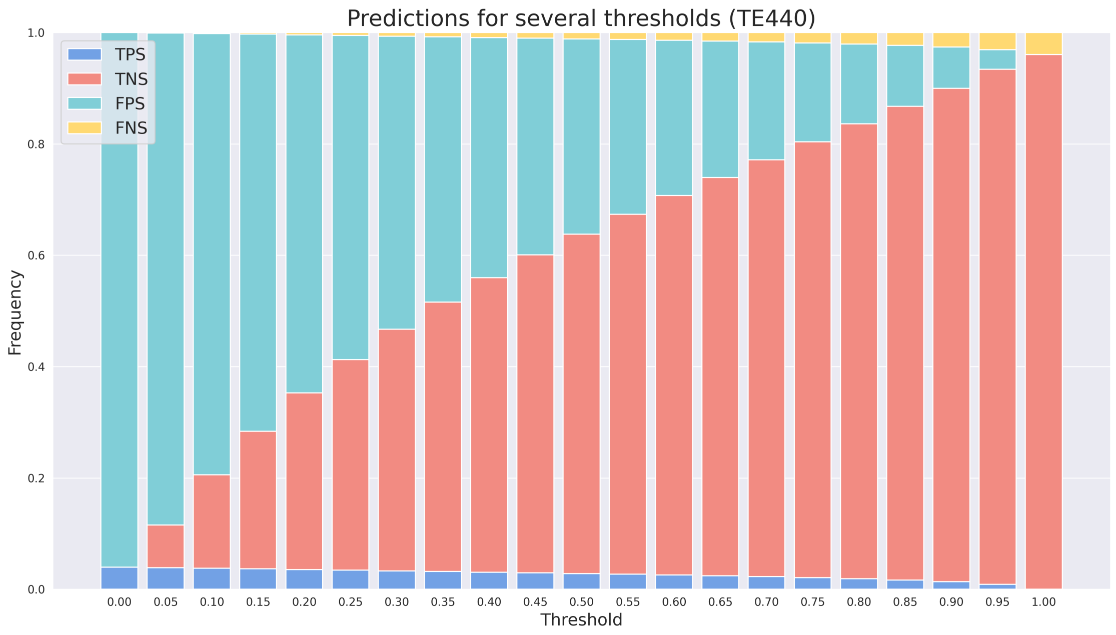


Figure 3.10: 21 bins histogram showing the fraction of true positives (in blue), true negatives (in pink), false positives (in green), and false negatives (in yellow) for several chosen thresholds on the whole TE440 test set.

CAID Binding Dataset Results

The results for the CAID Binding test dataset showed that the model is less performing on this data than the data obtained from the TE440 test dataset, since the task related to the former dataset is not perfectly in line with the task proposed by this project, as mentioned in the previous chapter. However, the results are comparable, showing similar strengths and problems, while the evaluation is important to give a great context to the problem addressed in this project.

In this case, the ROC curve shows a ROC-AUC score of 62.95 % (Figure 3.11), while the PR curve shows a PR-AUC score of 22.59 % (Figure 3.12). The former score is lower than the one found on the TE440 test set , while the latter is higher than it. Even in this case, the prediction of negative examples shows a better score in comparison with the prediction of positive examples (Table 3.4), although , for this dataset, the prediction performance of negative examples is slightly worse than the one seen for the TE440 dataset, while the prediction performance of positive examples is slightly better.

Even the distribution histogram for the scores of each amino acid position, shown in Figure 3.13, confirms this interesting aspect. With a maximized decision threshold of 86 %, it is possible to see a slightly bad performance on CAID Binding negative examples (on the left of the red line) in comparison with TE440 negative examples, while providing a better performance on CAID Binding positive examples (at the right of the line) negative examples compared with TE440 negative examples. In general, the ability to predict negative examples remains high, while the problem of false positives, although to a lesser extent, persists. Therefore, the model evaluated on the CAID Binding dataset was, indeed, able to identify 49157 true negative examples (i.e., 89.94 % of all negative predicted examples) against 6110 false negative examples (i.e., 11.06 % of all the negative predicted examples), while predicting 2086 true negative examples (i.e., 27.82 % of all the negative predicted examples), against 5411 examples that were incorrectly predicted (i.e., 72.18 % of all the negative predicted examples). As for the TE440 plot, the scores distribution histogram shows here a reduction in candle heights as it proceeds from score 0 to score 0.5, although, it is, in this case, a less pronounced slope, which, after the value of 0.5, struggles to rise again, but rather reaches a sort of plateau. This fact indicates, as discussed above, a predictably greater difficulty of the CNN model in predicting the CAID Binding dataset, which deviates from the typical test dataset for the specific task of this project.

Finally, Figure 3.15 shows a comparison of the results between the developed CNN model and the other models hosted in the CAID results page. In particular, the comparison takes into account the PR ad ROC curves derived from the evaluation of the CNN on the CAID Binding dataset. As already mentioned many times in this paper, it is once again reminded that the prediction task of this project is different from the prediction task involving the CAID Binding dataset, despite the two tasks show some overlap and allow a proper contextualization of what has been done here, highlighting the fact that the difficulties addressed here affect also all the models that try

	Negatives	**Positives**	**Global**
Precision score	0.8894	0.2782	0.5838
Recall score	0.9008	0.2545	0.5777
F1-score	0.8951	0.2659	0.5805
Accuracy score			0.8164
ROC-AUC score			0.6295
PR-AUC score			0.2259

Table 3.4: Evaluation metrics for the CAID Binding test set.

to solve problems related with protein interactions and disorder regions. Figure 3.14 shows the fraction of true positives, true negatives, false positives, and false negatives for several chosen thresholds on the whole CAID Binding test set. Even in this case, predictions are done by classifying as positives all the examples with a score which is greater or equal than the considered threshold. Exactly the same as the TE440 test set, increasing the threshold brought to a number of false positives and true positives which decreased, while a number of false negatives and true negatives which increased, until no positives are predicted for a decision threshold of 1.

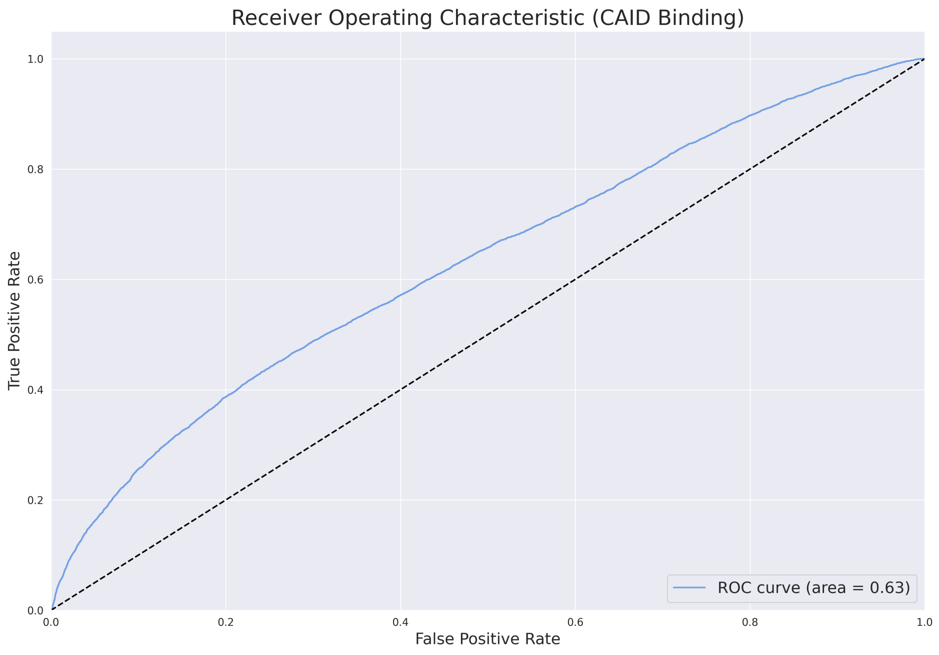


Figure 3.11: ROC curve for the CAID Binding test set, showing a ROC-AUC score of 62.95 %.

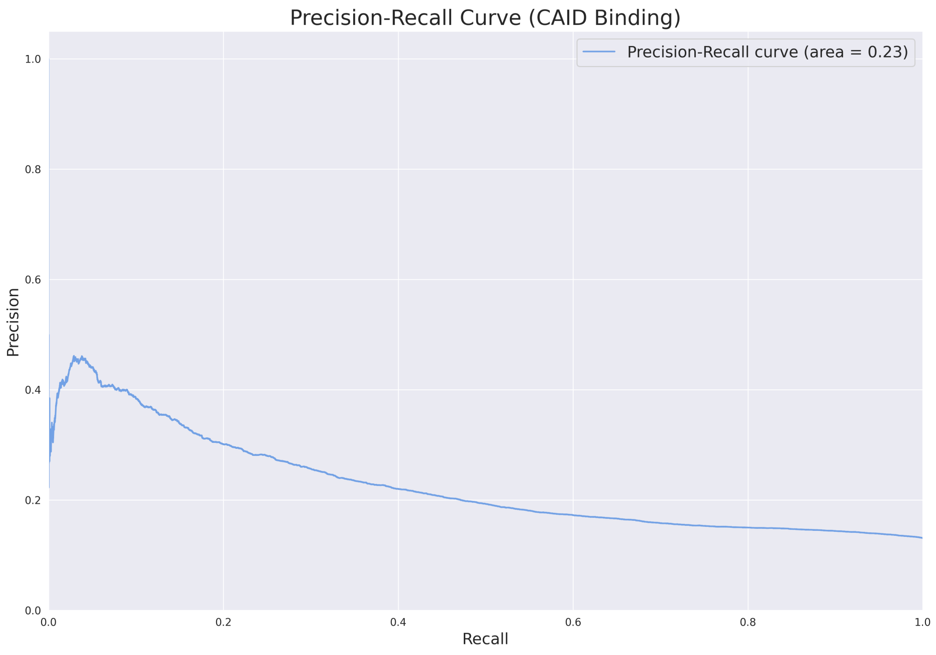


Figure 3.12: PR curve for the CAID Binding test set, showing a PR-AUC score of 22.59 %.

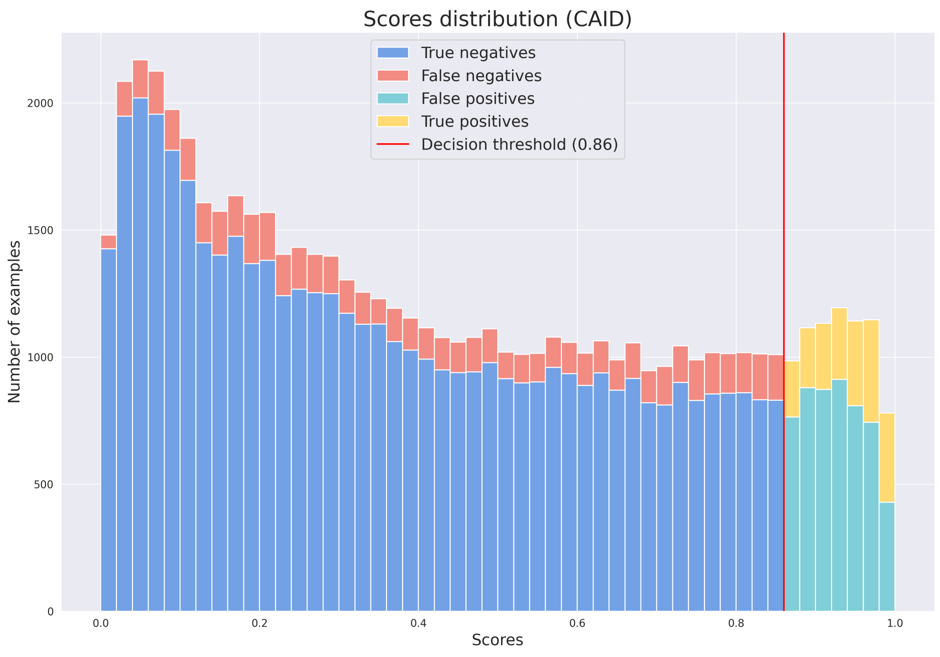


Figure 3.13: Histogram plot showing a 25 bins distribution for the CAID Binding test set for the maximized chosen threshold of 0.96, with a split of each bar that shows the four categories of true negatives (light blue bars), false negatives (pink bars), false positives (light green bars) and true positives (yellow bars). The red line corresponds to the decision threshold and divide the predictions of the model.

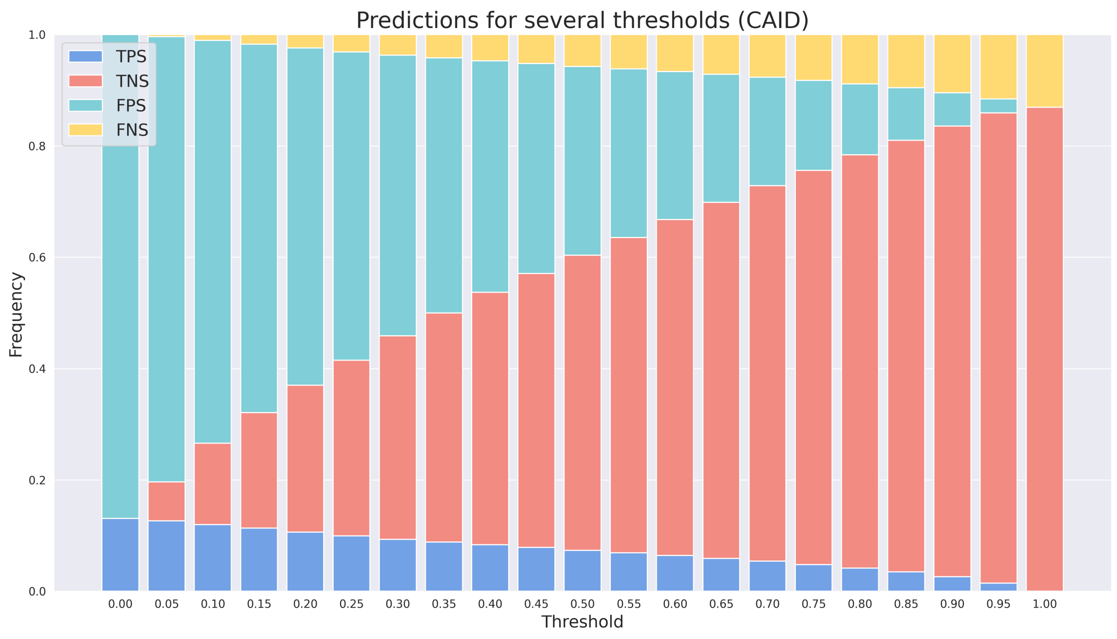


Figure 3.14: 21 bins histogram showing the fraction of true positives (in blue), true negatives (in pink), false positives (in green), and false negatives (in yellow) for several chosen thresholds on the whole CAID Binding test set.

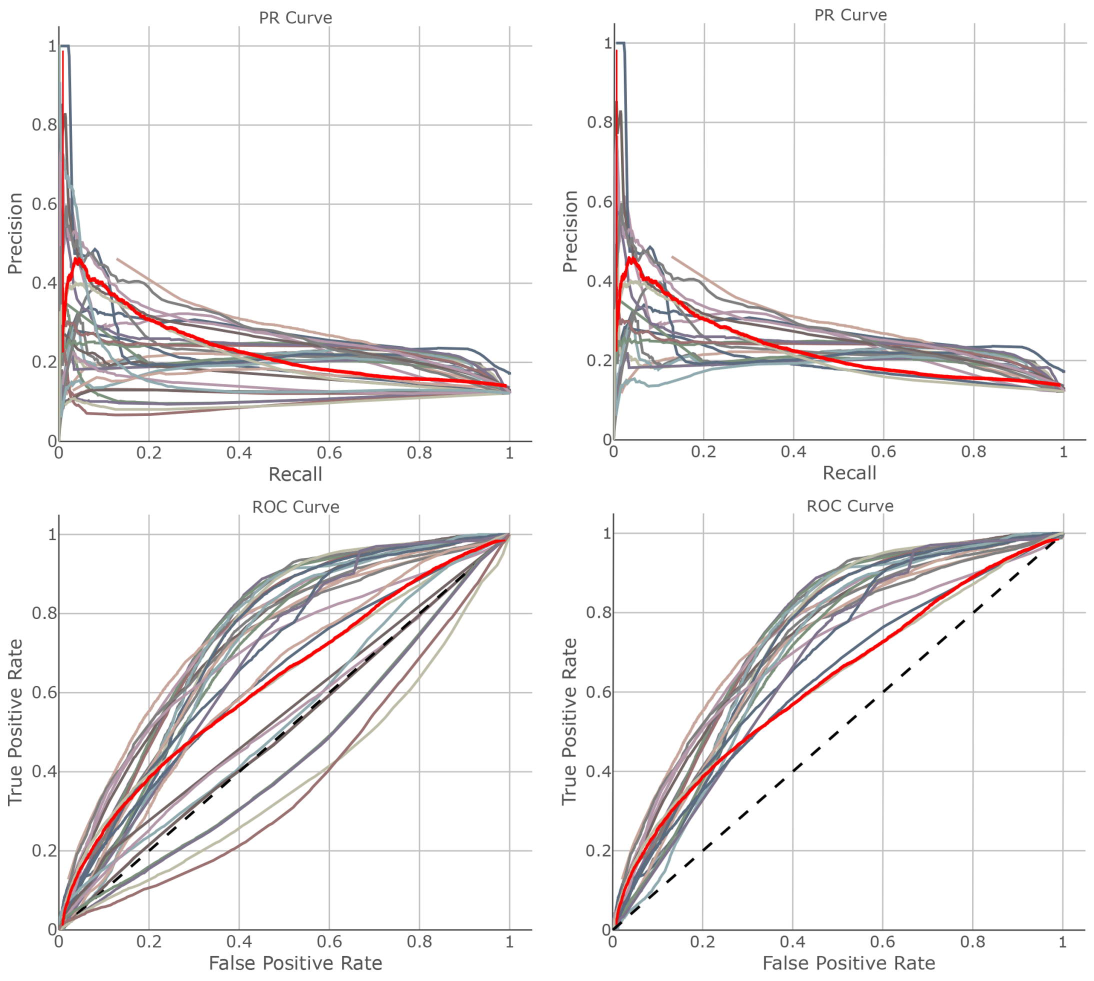


Figure 3.15: Comparison of PR and ROC curves for the CNN model developed for this project (evaluated on the CAID Binding dataset) with PR and ROC curves for CAID models developed with the specific task of predicting the CAID Binding dataset; the curves for these models underwent a 75 % desaturation, while the curve for the CNN was thickened and colored bright red, for a better look. In the top-left corner, PR curves comparison against all CAID models. In the bottom-left corner, ROC curves comparison against all CAID models. In the top-left corner, PR curves comparison against the top 20 CAID models. In the bottom-right corner, ROC curves comparison against the top 20 CAID models.

3.2 Discussion

This final section is dedicated to discussing results and choices presented thus far, offering interpretations and insights aimed at filling the results with biological significance. In this way, the reader can be able to get a more comprehensive understanding of this project. Furthermore, this sections hopes to provide some contributes to research in this field, by proposing additional insights for future investigations, with the common goal of bringing the state-of-the-art of research closer to the solution of the problem. For this reason, it is crucial for this discussion to adopt a cross-disciplinary approach, which seeks to merge together the domain-specific point of view, using the tools of molecular biology and structural biochemistry, and the Data Science point of view.

In the previous sections, by analyzing the results on all training and test datasets, a major identified issue was the inability of the model to correctly identifying positive examples, a problem highlighted in multiple occasions. In situations like this, the first thought that comes to mind might be to focus on the model, thinking that the problem stems from some implementation inadequacy. However, while the idea of reevaluating the model is a logical step for addressing results' issues, it is also crucial to contextualize the problem itself, by investigating the context in which the task is placed, considering the complexity of the problem itself and the inherent issues provided by the data used to train the model, while also taking into account the limitations of the tools employed to perform the prediction task.

3.2.1 The Context of the problem

In evaluating the predictive abilities of the model, it is important to consider the scientific context of the problem. For this purpose, the comparison provided by the CAID Binding dataset evaluation serves as a reference to put into context the task of this project and the model used to perform it. CAID predictions shows indeed how disordered regions binding is difficult to predict, even for all the models shown, which were specifically trained to solve this problem. Considering the fact that the majority of LIPs is contained in disordered regions, the problems related to the model can be put into perspective by looking at the comparison of the CNN with the models specifically used to predict interactions in disordered regions (Figure 3.15). Thus, although the test sets' results of the CNN seem not so high in absolute terms, such comparison shows an actually good contextualized performance of the model, which ranks among the top 20 models, in terms of ROC-AUC score, although the slightly different tasks. As already discussed in the **Introduction** chapter, the problem of predicting disordered binding starting from the protein sequence is an extremely complex and still open problem, in a context in which any predictor shows difficulties. Such contextually good results obtained with this first model on the CAID task can be taken as a validation of it, making it possible to candidate it, after further refinements and additional studies, to aspire to an improved predictive power and thus, to a potential more competitive comparison with other models.

3.2.2 Datasets and Tools Limitations

Another very important discussion must be dedicated to limitations in tools and data. Model performances shown in Figure 3.15 must be indeed considered within a strongly limited context, both in terms of data availability and reliability, due to the lack of an extensive and informative data background from which to start [12, 13, 103]. In fact, when dealing with the problem of predicting binding within disordered regions, it is difficult to find large amounts of reliable data, due to a fundamental major technical problem in building such datasets; this difficulty stems from the inherent structural complexity of intrinsically disordered regions, which results into an evident problem in finding large, manually curated datasets, since performing experiments in this area is prevented by the inherent instability of such regions [104, 54]. Due to this situation, experimental tasks are negatively influenced by the presence of high-dynamic structures, ultimately affecting the research attempt to experimentally verify the presence of protein interactions, as well as the attempt to purify structures, by using, for instance, biocrystallography or cryo-EM [54]. As an immediate consequence of these challenges, there is a major limit in generating high-dimensional, reliable datasets, introducing, in this way, a considerable bias when attempting to perform a predictive task. This is a crucial aspect, that must be considered when trying to develop a predictive model, since it can contributes to the degradation of predictive performances, especially when dealing with predicting the minority class of positive examples. This phenomenon would be independent from the inherent capabilities of the model, placing an upper limit on the prediction power that could only be crossed providing more complete and reliable datasets.

Another aspect of great importance is to contextualize the dataset used, which is derived from FLIPPER's predictions on the entire PDB, which, as already seen in the **Introduction** chapter, contains experimental protein structures derived from very specific conformational situations (especially what concerns cryo-EM and biocrystallography data). PDB files show very specific fixed structural situations, while, according to the definition of disorder, a very dynamic protein conformational ensemble is to be expected, in which, a disordered situation can occur in precise situations that may depend on binding events, on the presence of the protein in a specific compartment of the cell, or on the presence of an enzymatic activity; this facts highlights the fact that a PDB file cannot be considered as an adequate holotype of what is the conformational behavior of a given protein [105, 106, 107]. It is important to considering this fact when performing a prediction task involving interactions, especially in disordered regions. In the case of this project, the FLIPPER prediction dataset used for training the model could bring data that are not sufficiently complete or accurate, imposing an upper limit on the maximum performance of the model.

Another important consideration concerns the limitations due to the tools used. Of particular interest for this project is, indeed, the use of protein embeddings, which represent a necessary step and a potential bottleneck when it comes to deal with the loss of important biochemical information. A recent paper by Li *et. al.* (2024) [108]

addressed the problem related to the limitation of protein embeddings derived from PLMs. Here, the authors highlight the fact that, although protein embeddings were able to bring great innovation in the field of protein structure and feature prediction, such objects are still mostly unknown. Furthermore, as the authors warns, very little is known about what kind of information provided by embeddings can useful to downstream prediction tasks or how this information is used. The paper highlights also some potential issues related to the use of embeddings for specific tasks. So, considering this important aspect, although this aspect is not addressed in this project, due to space and time constraints, it may be of great future interest to conduct analyses aimed at a greater understanding of the interaction existing between these innovative vector objects and proteins (upstream) and models (downstream), in relation to the specific task of characterizing and predicting LIP regions.

Despite the inherent problems shown above, better performance results can be provided by other processing strategies for the training set, further studies of the model architecture and further analysis to find the optimal parameters for it. Performance improvements could also result from the use of different models or ensemble learning techniques (for instance, an idea could be to try an ensemble approach, by merging some other CAID Binding prediction model with the CNN trained for this project). Despite these proposals, it is of fundamental importance to always take into account the premises discussed above, since an ideal performance would require a change in the quality of the information brought as input to the model.

References

[1] D. L. Nelson, A. L. Lehninger, and M. M. Cox, *Lehninger principles of biochemistry*. Macmillan, 2008.

[2] J. B. Reece, N. Meyers, L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, and R. Orr, *Campbell Biology*. Pearson, 2021.

[3] D. Szklarczyk, A. Franceschini, S. Wyder, K. Forslund, D. Heller, J. Huerta-Cepas, M. Simonovic, A. Roth, A. Santos, K. P. Tsafou *et al.*, "String v10: protein–protein interaction networks, integrated over the tree of life," *Nucleic acids research*, vol. 43, no. D1, pp. D447–D452, 2015.

[4] B. Alberts, *Molecular biology of the cell*. Garland science, 2017.

[5] K. M. Poluri, K. Gulati, K. M. Poluri, and K. Gulati, "Biotechnological and biomedical applications of protein engineering methods," *Protein Engineering Techniques: Gateways to Synthetic Protein Universe*, pp. 103–134, 2017.

[6] C. Li, R. Zhang, J. Wang, L. M. Wilson, and Y. Yan, "Protein engineering for improving and diversifying natural product biosynthesis," *Trends in biotechnology*, vol. 38, no. 7, pp. 729–744, 2020.

[7] U. T. Bornscheuer, B. Hauer, K. E. Jaeger, and U. Schwaneberg, "Directed evolution empowered redesign of natural proteins for the sustainable production of chemicals and pharmaceuticals," *Angewandte Chemie International Edition*, vol. 58, no. 1, pp. 36–40, 2019.

[8] B. Peng, X.-Y. Tian, Z.-Y. Liu, J. Peng, M.-Y. Zheng, X.-H. Song, Y.-Q. Huang, A. Xin-Xiang, Y.-M. Zhang, F. Yang *et al.*, "The application of modern biotechnology in protein interaction research-a review," *Journal of Agriculture and Crops*, vol. 7, no. 2, pp. 39–47, 2021.

[9] A. K. Dunker, J. D. Lawson, C. J. Brown, R. M. Williams, P. Romero, J. S. Oh, C. J. Oldfield, A. M. Campen, C. M. Ratliff, K. W. Hipps *et al.*, "Intrinsically disordered protein," *Journal of molecular graphics and modelling*, vol. 19, no. 1, pp. 26–59, 2001.

[10] H. J. Dyson and P. E. Wright, "Intrinsically unstructured proteins and their functions," *Nature reviews Molecular cell biology*, vol. 6, no. 3, pp. 197–208, 2005.

[11] A. K. Dunker, I. Silman, V. N. Uversky, and J. L. Sussman, "Function and structure of inherently disordered proteins," *Current opinion in structural biology*, vol. 18, no. 6, pp. 756–764, 2008.

[12] V. N. Uversky, A. K. Dunker *et al.*, *Intrinsically Disordered Protein Analysis: Volume 1, Methods and Experimental Tools*. Springer, 2012.

[13] B. Kragelund and K. Skriver, *Intrinsically Disordered Proteins*. Springer, 2020.

[14] T. Berggård, S. Linse, and P. James, "Methods for the detection and analysis of protein–protein interactions," *Proteomics*, vol. 7, no. 16, pp. 2833–2842, 2007.

[15] F. Diella, N. Haslam, C. Chica, A. Budd, S. Michael, N. P. Brown, G. Travé, and T. J. Gibson, "Understanding eukaryotic linear motifs and their role in cell signaling and regulation," *Front Biosci*, vol. 13, no. 6580, p. 603, 2008.

[16] A. M. Monzon, P. Bonato, M. Necci, S. C. Tosatto, and D. Piovesan, "Flipper: predicting and characterizing linear interacting peptides in the protein data bank," *Journal of Molecular Biology*, vol. 433, no. 9, p. 166900, 2021.

[17] Z. Peng, Z. Li, Q. Meng, B. Zhao, and L. Kurgan, "Clip: accurate prediction of disordered linear interacting peptides from protein sequences using co-evolutionary information," *Briefings in Bioinformatics*, vol. 24, no. 1, p. bbac502, 2023.

[18] M. Kanehisa and P. Bork, "Bioinformatics in the post-sequence era," *Nature genetics*, vol. 33, no. 3, pp. 305–310, 2003.

[19] K. Karasavvas, R. Baldock, and A. Burger, "Bioinformatics integration and agent technology," *Journal of biomedical informatics*, vol. 37, no. 3, pp. 205–219, 2004.

[20] P. Hogeweg, "The roots of bioinformatics in theoretical biology," *PLoS computational biology*, vol. 7, no. 3, p. e1002021, 2011.

[21] P. Romano, R. Giugno, and A. Pulvirenti, "Tools and collaborative environments for bioinformatics research," *Briefings in bioinformatics*, vol. 12, no. 6, pp. 549–561, 2011.

[22] M. W. Covert, E. M. Knight, J. L. Reed, M. J. Herrgard, and B. O. Palsson, "Integrating high-throughput and computational data elucidates bacterial networks," *Nature*, vol. 429, no. 6987, pp. 92–96, 2004.

[23] W. W. Soon, M. Hariharan, and M. P. Snyder, "High-throughput sequencing for biology and medicine," *Molecular systems biology*, vol. 9, no. 1, p. 640, 2013.

[24] T. U. Consortium, "Uniprot: the universal protein knowledgebase in 2023," *Nucleic Acids Research*, vol. 51, no. D1, pp. D523–D531, 2023.

[25] C. H. Wu, R. Apweiler, A. Bairoch, D. A. Natale, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. Martin, R. Mazudmer, C. O'Donovan, N. Redaschi, and B. Suzek, "The universal protein resource (uniprot): an expanding universe of protein information," *Nucleic acids research*, vol. 34, no. suppl_1, pp. D187–D191, 2006.

[26] S. Poux, M. Magrane, C. N. Arighi, A. Bridge, C. O'Donovan, K. Laiho, and U. Consortium, "Expert curation in uniprotkb: a case study on dealing with conflicting and erroneous data," *Database*, vol. 2014, p. bau016, 2014.

[27] M. Magrane and U. Consortium, "Uniprot knowledgebase: a hub of integrated protein data," *Database*, vol. 2011, p. bar009, 2011.

[28] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic Acids Research*, vol. 28, no. 1, pp. 235–242, 2000.

[29] ——, "The protein data bank," *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000.

[30] S. K. Burley, H. M. Berman, G. J. Kleywegt, J. L. Markley, H. Nakamura, and S. Velankar, "Protein data bank (pdb): the single global macromolecular structure archive," *Protein crystallography: methods and protocols*, pp. 627–641, 2017.

[31] "Rcsb pdb main page." [Online]. Available: https://www.rcsb.org/

[32] J. Westbrook, Z. Feng, L. Chen, H. Yang, and H. M. Berman, "The protein data bank and structural genomics," *Nucleic acids research*, vol. 31, no. 1, pp. 489–491, 2003.

[33] R. P. Joosten, T. A. Te Beek, E. Krieger, M. L. Hekkelman, R. W. Hooft, R. Schneider, C. Sander, and G. Vriend, "A series of pdb related databases for everyday needs," *Nucleic acids research*, vol. 39, no. suppl_1, pp. D411–D419, 2010.

[34] S. Velankar, J. M. Dana, J. Jacobsen, G. Van Ginkel, P. J. Gane, J. Luo, T. J. Oldfield, C. O'Donovan, M.-J. Martin, and G. J. Kleywegt, "Sifts: structure integration with function, taxonomy and sequences resource," *Nucleic acids research*, vol. 41, no. D1, pp. D483–D489, 2012.

[35] T. Di Domenico, I. Walsh, A. J. Martin, and S. C. Tosatto, "Mobidb: a comprehensive database of intrinsic protein disorder annotations," *Bioinformatics*, vol. 28, no. 15, pp. 2080–2081, 2012.

[36] D. Piovesan, A. Del Conte, D. Clementel, A. M. Monzon, M. Bevilacqua, M. C. Aspromonte, J. A. Iserte, F. E. Orti, C. Marino-Buslje, and S. C. Tosatto, "Mobidb: 10 years of intrinsically disordered proteins," *Nucleic acids research*, vol. 51, no. D1, pp. D438–D444, 2023.

[37] E. Potenza, T. D. Domenico, I. Walsh, and S. C. Tosatto, "Mobidb 2.0: an improved database of intrinsically disordered and mobile proteins," *Nucleic acids research*, vol. 43, no. D1, pp. D315–D320, 2015.

[38] D. Piovesan, F. Tabaro, L. Paladin, M. Necci, I. Mičetić, C. Camilloni, N. Davey, Z. Dosztányi, B. Mészáros, A. M. Monzon, G. Parisi, E. Schad, P. Sormanni, P. Tompa, M. Vendruscolo, W. F. Vranken, and S. C. Tosatto, "Mobidb 3.0: more annotations for intrinsic disorder, conformational diversity and interactions in proteins," *Nucleic acids research*, vol. 46, no. D1, pp. D471–D476, 2018.

[39] M. C. Aspromonte, M. V. Nugnes, F. Quaglia, A. Bouharoua, S. C. Tosatto, and D. Piovesan, "Disprot in 2024: improving function annotation of intrinsically disordered proteins," *Nucleic Acids Research*, vol. 52, no. D1, pp. D434–D441, 2024.

[40] "Mobidb user guide." [Online]. Available: https://mobidb.bio.unipd.it/help

[41] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[42] R. Bonetta and G. Valentino, "Machine learning techniques for protein function prediction," *Proteins: Structure, Function, and Bioinformatics*, vol. 88, no. 3, pp. 397–413, 2020.

[43] N. T. Doncheva, K. Klein, F. S. Domingues, and M. Albrecht, "Analyzing and visualizing residue networks of protein structures," *Trends in biochemical sciences*, vol. 36, no. 4, pp. 179–182, 2011.

[44] R. K. Grewal and S. Roy, "Modeling proteins as residue interaction networks," *Protein and peptide letters*, vol. 22, no. 10, pp. 923–933, 2015.

[45] D. Piovesan, G. Minervini, and S. C. Tosatto, "The ring 2.0 web server for high quality residue interaction networks," *Nucleic acids research*, vol. 44, no. W1, pp. W367–W374, 2016.

[46] D. Piovesan, M. Necci, N. Escobedo, A. M. Monzon, A. Hatos, I. Mičetić, F. Quaglia, L. Paladin, P. Ramasamy, Z. Dosztányi *et al.*, "Mobidb: intrinsically disordered proteins in 2021," *Nucleic acids research*, vol. 49, no. D1, pp. D361–D367, 2021.

[47] M. Varadi, S. Anyango, M. Deshpande, S. Nair, C. Natassia, G. Yordanova, D. Yuan, O. Stroe, G. Wood, A. Laydon *et al.*, "Alphafold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models," *Nucleic acids research*, vol. 50, no. D1, pp. D439–D444, 2022.

[48] V. Mariani, M. Biasini, A. Barbato, and T. Schwede, "lddt: a local superposition-free score for comparing protein structures and models using distance difference tests," *Bioinformatics*, vol. 29, no. 21, pp. 2722–2728, 2013.

[49] D. Piovesan, A. M. Monzon, and S. C. Tosatto, "Intrinsic protein disorder and conditional folding in alphafolddb," *Protein Science*, vol. 31, no. 11, p. e4466, 2022.

[50] T. Bepler and B. Berger, "Learning the protein language: Evolution, structure, and function," *Cell systems*, vol. 12, no. 6, pp. 654–669, 2021.

[51] R. M. Rao, J. Liu, R. Verkuil, J. Meier, J. Canny, P. Abbeel, T. Sercu, and A. Rives, "Msa transformer," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8844–8856.

[52] N. Brandes, D. Ofer, Y. Peleg, N. Rappoport, and M. Linial, "Proteinbert: a universal deep-learning model of protein sequence and function," *Bioinformatics*, vol. 38, no. 8, pp. 2102–2110, 2022.

[53] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik, and B. Rost, "Prottrans: Toward understanding the language of life through self-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 10, pp. 7112–7127, 2021.

[54] M. Necci, D. Piovesan, and S. C. Tosatto, "Critical assessment of protein intrinsic disorder prediction," *Nature methods*, vol. 18, no. 5, pp. 472–481, 2021.

[55] A. D. Conte, M. Mehdiabadi, A. Bouhraoua, A. Miguel Monzon, S. C. Tosatto, and D. Piovesan, "Critical assessment of protein intrinsic disorder prediction (caid)-results of round 2," *Proteins: Structure, Function, and Bioinformatics*, vol. 91, no. 12, pp. 1925–1934, 2023.

[56] "Biomine kurgan lab website." [Online]. Available: http://biomine.cs.vcu.edu/

[57] "Caid datasets and results." [Online]. Available: https://caid.idpcentral.org/challenge/results

[58] M. Greenacre, P. J. Groenen, T. Hastie, A. I. d'Enza, A. Markos, and E. Tuzhilina, "Principal component analysis," *Nature Reviews Methods Primers*, vol. 2, no. 1, p. 100, 2022.

[59] I. T. Jolliffe, *Principal component analysis for special types of data*. Springer, 2002.

[60] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[61] R. Bro and A. K. Smilde, "Principal component analysis," *Analytical methods*, vol. 6, no. 9, pp. 2812–2831, 2014.

[62] G. Hinton, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, pp. 857–864, 2003.

[63] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[64] J. H. Krijthe and L. Van der Maaten, "Rtsne: T-distributed stochastic neighbor embedding using barnes-hut implementation," *R package version 0.13, URL https://github. com/jkrijthe/Rtsne*, 2015.

[65] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[66] "Scikit-learn t-sne user guide." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[67] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, "Review of image classification algorithms based on convolutional neural networks," *Remote Sensing*, vol. 13, no. 22, p. 4712, 2021.

[68] F. Yu, Q. Zhang, J. Xiao, Y. Ma, M. Wang, R. Luan, X. Liu, Y. Ping, Y. Nie, Z. Tao *et al.*, "Progress in the application of cnn-based image classification and recognition in whole crop growth cycles," *Remote Sensing*, vol. 15, no. 12, p. 2988, 2023.

[69] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[70] A. El-Sawy, H. El-Bakry, and M. Loey, "Cnn for handwritten arabic digits recognition based on lenet-5," in *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016 2*. Springer, 2017, pp. 566–575.

[71] S. Arya and R. Singh, "A comparative study of cnn and alexnet for detection of disease in potato and mango leaf," in *2019 International conference on issues and challenges in intelligent computing techniques (ICICT)*, vol. 1. IEEE, 2019, pp. 1–6.

[72] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-vgg16 cnn model for big data places image recognition," in *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)*. IEEE, 2018, pp. 169–175.

[73] P. Tang, H. Wang, and S. Kwong, "G-ms2f: Googlenet based multi-stage feature fusion of deep cnn for scene recognition," *Neurocomputing*, vol. 225, pp. 188–197, 2017.

[74] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.

[75] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[76] F. Chollet, *Deep learning with Python*. Simon and Schuster, 2021.

[77] M. Jorda, P. Valero-Lara, and A. J. Pena, "Performance evaluation of cudnn convolution algorithms on nvidia volta gpus," *IEEE Access*, vol. 7, pp. 70 461–70 473, 2019.

[78] S. Yadav and S. Shukla, "Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification," in *2016 IEEE 6th International conference on advanced computing (IACC)*. IEEE, 2016, pp. 78–83.

[79] A. Blum, A. Kalai, and J. Langford, "Beating the hold-out: Bounds for k-fold and progressive cross-validation," in *Proceedings of the twelfth annual conference on Computational learning theory*, 1999, pp. 203–208.

[80] "Scikit-learn k-fold stratified corss-validation user guide." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

[81] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018.

[82] M. Bernhofer and B. Rost, "Tmbed: transmembrane proteins predicted through language model embeddings," *BMC bioinformatics*, vol. 23, no. 1, p. 326, 2022.

[83] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[84] G. Squillero, P. Burelli *et al.*, *Applications of evolutionary computation*. Springer, 2016.

[85] S. K. Kumar, "On weight initialization in deep neural networks," *arXiv preprint arXiv:1704.08863*, 2017.

[86] "Pytorch initializers user guide." [Online]. Available: https://pytorch.org/docs/stable/nn.init.html

[87] J. Li, Y. Song, X. Song, and D. Wipf, "On the initialization of graph neural networks," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19 911–19 931.

[88] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 984–993.

[89] Z. Ye, C. Li, Q. Liu, L. Bai, and J. E. Fowler, "Computationally lightweight hyperspectral image classification using a multiscale depthwise convolutional network with channel attention," *IEEE Geoscience and Remote Sensing Letters*, 2023.

[90] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[91] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[92] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, no. 8, p. 2, 2012.

[93] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[94] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," in *International Conference on Machine Learning*. PMLR, 2023, pp. 23 803–23 828.

[95] C. M. Bishop, "Pattern recognition and machine learning," *Springer google schola*, vol. 2, 2006.

[96] "Pytorch binary cross-entropy function with integrated sigmoid user guide." [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

[97] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[98] "Pytorch bathc normalization user guide." [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm2d.html

[99] M. Dmytro, "caffenet-benchmark," GitHub repository, 2016. [Online]. Available: https://github.com/ducha-aiki/caffenet-benchmark/blob/master/batchnorm.md

[100] Y. Bai, "Relu-function and derived function review," in *SHS Web of Conferences*, vol. 144. EDP Sciences, 2022, p. 02006.

[101] S. Park and N. Kwak, "Analysis on the dropout effect in convolutional neural networks," in *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II 13*. Springer, 2017, pp. 189–204.

[102] X. Xie, M. Xie, A. J. Moshayedi, M. H. Noori Skandari *et al.*, "A hybrid improved neural networks algorithm based on l2 and dropout regularization," *Mathematical Problems in Engineering*, vol. 2022, 2022.

[103] P. Romero, Z. Obradovic, X. Li, E. C. Garner, C. J. Brown, and A. K. Dunker, "Sequence complexity of disordered protein," *Proteins: Structure, Function, and Bioinformatics*, vol. 42, no. 1, pp. 38–48, 2001.

[104] A. Szilagyi, V. Grimm, A. K. Arakaki, and J. Skolnick, "Prediction of physical protein–protein interactions," *Physical biology*, vol. 2, no. 2, p. S1, 2005.

[105] H. J. Dyson and P. E. Wright, "Coupling of folding and binding for unstructured proteins," *Current opinion in structural biology*, vol. 12, no. 1, pp. 54–60, 2002.

[106] U. Jakob, R. Kriwacki, and V. N. Uversky, "Conditionally and transiently disordered proteins: awakening cryptic disorder to regulate protein function," *Chemical reviews*, vol. 114, no. 13, pp. 6779–6805, 2014.

[107] I. Bahar, C. Chennubhotla, and D. Tobi, "Intrinsic dynamics of enzymes in the unbound state and relation to allosteric regulation," *Current opinion in structural biology*, vol. 17, no. 6, pp. 633–640, 2007.

[108] F.-Z. Li, A. P. Amini, Y. Yue, K. K. Yang, and A. X. Lu, "Feature reuse and scaling: Understanding transfer learning with protein language models," *bioRxiv*, pp. 2024–02, 2024.

Acknowledgments

The code for the project was entirely made using Python 3.8.8 and its several built-in libraries. The external libraries used are NumPy 1.23.5 and Pandas 1.5.3 for manipulating and processing all the datasets; Matplotlib 3.7.1, Seaborn 0.12.2, and NetworkX 3.2.1. for every data visualization and results plot; Scikit-learn 1.3.2 for dimensionality reduction algorithms, metrics calculation and tools for stratified k-fold cross-validation; PyTorch 2.1.2 (cu121) for developing the deep learning model and the padding system for the dataset. Molecular graphics and analyses were performed with UCSF ChimeraX 1.7.1, developed by the Resource for Biocomputing, Visualization, and Informatics at the University of California, San Francisco, with support from National Institutes of Health R01-GM129325 and the Office of Cyber Infrastructure and Computational Biology, National Institute of Allergy and Infectious Diseases. Skeletal diagrams of the amino acids were realized with ChemDoodle 6.0. Nneural network diagrams drawn using DrawIo 24.0.5. A special thanks to BioComputing UP laboratory, at University of Padua, for providing the training dataset through the MobiDB service, and, most of all, for the indispensable and valuable help, in terms of knowledge, computational and storage resources, and consulting.