Deep Learning for Protein-Protein Interaction Prediction and Protein Design

Junzheng Wu

Thesis submitted to the University of Ottawa
In partial fulfillment of the requirements
For the Ph.D. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Dedication

To the ones who choose to believe,

Abstract

Protein–protein interactions (PPI) play a fundamental role in many biochemical functions such as signal transduction, cellular organization, and cell cycle progression. Laboratory experiments are expensive and time-consuming, limiting their applicability to a handful of cases, which severely impacts the speed of research innovation. Meanwhile, computational experiments can explore and screen a vast number of possibilities leaving the researchers with the most promising cases. For this reason, there is an urgent need to develop computational solutions for accurately determining whether proteins interact. Specifically, developing deep learning solutions to predict PPI constitutes an emerging area of research with much practical application. This thesis focuses on advancing the research in this area.

In our first contribution, the thesis introduces a novel deep learning algorithm for predicting PPI solely from their amino acid sequences. The novelty of our work is that we consider self-binding and folding properties rather than only focusing on the sequences per se, as is commonplace in the state of the art. Our Siamese pyramid network (SPNet) architecture comprises a multilevel Siamese neural network with an attention mechanism and a trainable probability prediction network. Our experimental evaluation indicates that SPNet outperforms the state of the art against strict data sets, that is, data sets with no data leakage, leading to accurate and timely predictions.

Our second contribution centers on the observation that the sizes and compositions of PPI data sets vary considerably, which may impact the learnability for deep learning solutions. For instance, small data sets are commonplace for rare and new diseases, while current deep learning implementations usually rely on large PPI data sets to produce accurate predictions. To this end, we introduce an adaptive and learnable pyramid network architecture in which the depth and the complexity of the network are directly learned from the data. Our experimental evaluation shows that these characteristics make our solution suitable for both small and large PPI data sets, varying in sizes from a few thousand to seventeen million pairs.

In addition, the number of negative (non-interacting) protein pairs is generally scarce, resulting in highly imbalanced data sets that may impede neural network training. There-

fore, our third contribution centers on a novel methodology, based on graphs and geodesic distances, that extracts non-interacting proteins from the graph associated with their interactions and binding strength. A new balanced B-STRING data set is created that consists of 17,591,832 protein pairs. Our comparative experimental evaluation not only confirms the value of our learnable pyramid network architecture but also shows the value of providing a novel benchmark data set for future use by the community.

In our final contribution, we introduce our architecture for designing protein binders based on primary sequences through a transformer deep learning model that adopts the mechanism of attention, differentially weighing the significance of each part of the input sequence. Predicting protein binders is a multivalued problem, which implies that there is more than one solution, since there may be more than one binder for a specific protein. To solve a such a learning task, we utilize two types of constraints in our deep learning solution. These are based, respectively, on the binding score which is related to the strength of interactions, and the Bayesian prior, where we assume that a small portion of the ligands' amino acids are known. Our experimental evaluation confirms the strengths of this novel approach.

Acknowledgements

I cannot express enough appreciation to Dr. Herna L. Viktor. As a mere student with no name, I was deeply absorbed in your intriguing class. Your kindness and generosity offered me an opportunity, which marked the beginning of all my dreams.

I would also like to extend my sincere gratitude to Dr. Eric Paquet. Throughout the past four years, you have been a hero to me. Whenever I was lost, you were always there to guide and support me. The completion of my dissertation would not have been possible without your unwavering assistance.

To my parents, Huaigang Wu and Zuonan Zhang, I am proud to be your son. Without your love and support, I would not have been able to achieve this milestone.

Additionally, I am deeply grateful to my friends Julian, Boyang, Johnathon, Rui, Reese, and many others who have acted as my therapists and provided me with emotional support without expecting anything in return.

Finally, I would like to acknowledge the financial support from the National Research Council (NRC) and the University of Ottawa, which has been instrumental in enabling me to pursue my research goals.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Proteins perform many functions at the cellular level, including metabolic reaction catalysis, DNA replication, stimulus response, molecular transportation, and organism structure [11]. Proteins are made of linear chains of amino acid residues called polypeptides. The sequence of amino acids in a protein, determined by genetics, distinguishes one protein from another. The amino acid sequence allows a protein to fall into a specific 3-D structure, called a native conformation that determines its activity. Proteins may shift into different conformations to perform their functions, making some structures available for a reaction and hiding others. The transitions between shape states are called conformational changes.

PPIs are responsible for various cellular functions such as signal transduction, cellular organization, and cell progression [12, 13]. Furthermore, PPIs play a central role in both the prevention and treatment of diseases [14]. Prevention may be achieved, for instance, with prophylactic vaccines, whereas treatment may consist of targeted therapeutic proteins such as antibodies [15]. Due to the astonishing number of possibilities, finding the most suitable interacting protein is similar to locating a needle in a haystack. High throughput experimental techniques have been created to address the complexity of the search space, such as chip-based techniques [16], microarray [17], and interactomics [18]. Unfortunately, these in vitro experiments are both time-consuming and expensive, often resulting in inadequate performances.

Meanwhile, computational methods, especially those based on machine learning and artificial intelligence, can explore and screen a vast number of possibilities, providing the experimentalists with the most promising cases. As early as 2005, the first machine learning approach, based on Bayesian classifiers, was proposed by Jansen *et al.* . [19]. Since then, machine learning techniques have been applied for the prediction of PPI utilizing support vector machine (SVM) [10, 20–25], random forest (RF) [26, 27], and boosting [28, 29].

Shortly after its introduction, deep learning superseded traditional machine learning techniques in many fields such as computer vision and natural language processing (NLP) [30, 31]. This is particularly true when large amounts of data are available; in such cases, deep learning generally outperforms traditional machine learning techniques [32, 33]. This statement is also true for PPI. For instance, Sun *et al.* employed a stacked autoencoder (SAE) for protein classification [34]. Proteins were encoded either with the autocovariance method, as proposed earlier by Guo *et al.* , or with the triad method, as introduced by Shen *et al.* ; the former performs better than the latter. Guo's approach outperformed the state-of-the-art conventional machine learning approaches [21, 23, 35] on Pan's benchmark data set [35]. Hashemifar *et al.* [36] combined a Siamese Neural Network with a position-specific iterative basic local alignment search tool (PSIBLAST) [37] and outperformed several SVM-based methods, including [10, 24, 25]. In another work, Li *et al.* [38] created an architecture consisting of an embedding layer, convolutional layers, and long short-term memory (LSTM) layer in order to aid PPI classification.

With the powerful expression ability of deep learning approaches, it is possible to rapidly screen many binders with a PPI predictor. However the screening process is limited to known proteins and may not necessarily be suitable for all applications, e.g., the design of a therapeutic antibody for a new or orphan disease [39]. As a result, it is not possible to directly generate new proteins with a screening method. Therefore, there is a great need for solutions, especially those based on deep learning, that can predict non-existent binders in addition to known ones. More specifically, it is proposed, given the amino acid sequence of a target protein, to determine the amino acid sequence of the corresponding binder without recourse to domain knowledge such as structural information. The above requirements may be realised by utilizing the seq2seq family of algorithms which usually employ deep learning to convert one sequence into another [40], as will be discussed in this

work.

The research contributions presented in this thesis center around two main themes, namely PPI predictions and PPI binder designs, as will be detailed in the next section.

1.1 Research Problems

Five research problems are addressed in this study:

I Predicting PPI with DL from amino acid sequences while considering self-binding and folding.

II Addressing the scarcity of data about non-interacting proteins with a graph of interacting proteins.

III Automatically learning the deep neural network complexity and depth in order to obtain high prediction accuracy for both very large and very small (e.g. rare and new diseases) PPI data sets.

IV Automatically designing a therapeutic protein (binder) that binds to the receptor, given the amino acid sequence of a pathogen.

In the following, a description, motivation and methodology are provided for each research problem.

1.1.1 Research Problem I: Protein-Protein Interactions with Folding and Self-Binding Primary Sequences

PPIs play a fundamental role in drug design, gene therapy and vaccine development. The study of PPIs relies heavily on complex and time-consuming experiments, which severely impacts research throughputs. Thus, it is important to provide the researcher with the most promising cases by screening rapidly through many potential candidates. We introduce a novel deep neural network architecture that allows the binding probability for

two proteins to be predicted instantly based solely on their amino acid sequences. Subsequently, screenings are performed based on the binding probabilities. The novelty of our approach lies in the fact that we consider self-binding and folding amino acid sequences (as illustrated in Fig. 1.1) rather than just looking at these sequences per se. The structure of the recently identified 2019-nCov (Covid-19) spike [1] exhibits the complexity associated with folding and self-binding amino acid sequences. These aspects are important because they determine a protein's conformation (shape), which, in turn, strongly constrains its function. Chapter 3 presents our novel Siamese pyramid network (SPNet) architecture inspired by feature pyramid networks and consists of a multilevel Siamese neural network with an attention mechanism and a multilevel, trainable binding probability prediction network. We detail our experimental evaluation performed on a strict data set (no data leakage), and it is shown that SPNet outperforms the state-of-the-art architectures.

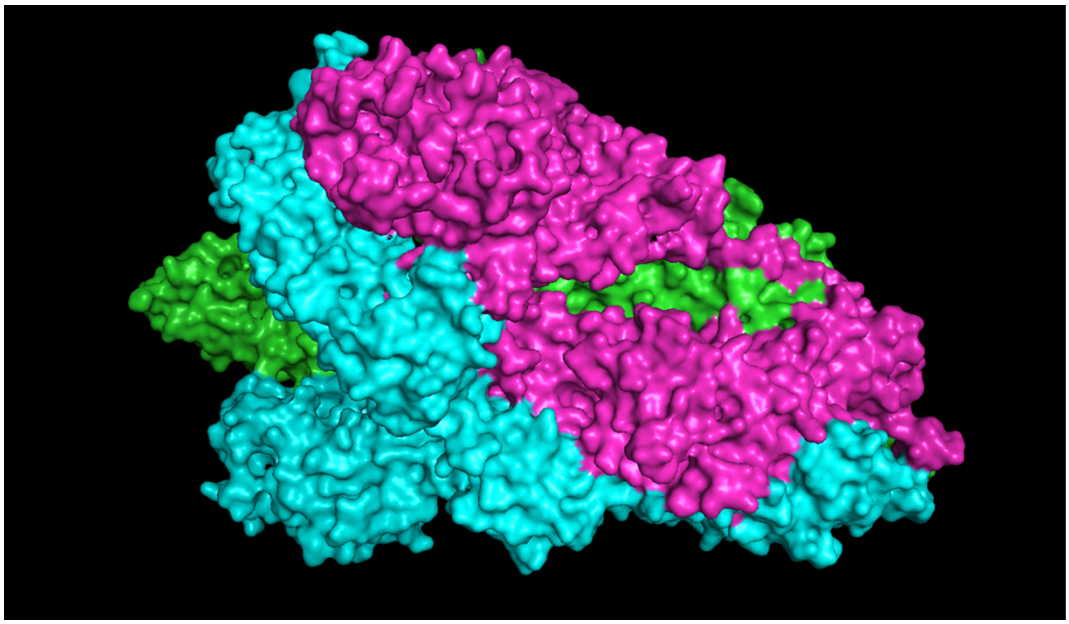


Figure 1.1: Structure of the 2019-nCov spike in the prefusion conformation: a key target for therapeutic antibodies [1]; each colour corresponds to an amino acid sequence.

The architecture, the contributions, and the experimental results are fully reported in Chapter 3.

1.1.2 Research Problem II and III: Adapting to Complexity: Data Generation and Deep Learnable Architecture for Protein-protein Interaction Predictions

As stated above, PPIs play an important role in developing new therapeutic treatments and, specifically, prophylactic vaccines. For instance, the efficacy of a vaccine strongly depends on the extent to which an antibody may form a stable bond with an antigen. In-laboratory (in vitro) experiments are both time-consuming and expensive, limiting their scope to only the most relevant interactions. As already noted, computational experiments (in silico) have the potential to explore and screen a considerable number of possibilities, thus vastly speeding up the process of identifying candidates for future experimentation. PPIs may be learned by deep learning networks, such as our SPNet architecture introduced above. However, the training of these networks typically requires many instances that may not be readily available, as in the case of rare or new diseases. Furthermore, the learning process is made more complex by the scarcity of data about non-interacting proteins, making networks prone to overfitting.

These two shortcomings are addressed in Chapter 4 of this thesis. A new method, based on graphs and geodesic distances, extracts non-interacting proteins from the interaction graph associated with interacting proteins: the most distant pairs, in terms of binding strength, corresponding to the non-interacting ones. A new balanced data set is created from STRING: B-STRING, which consists of 17,591,832 protein pairs. In addition, a new learnable pyramid network architecture is proposed in which the depth and the complexity of the network are directly learned from the data; such adaptability makes the network suitable for both small and large data sets. As will be shown in Chapter 4, our resultant architecture outperforms state-of-the-art neural networks for PPI predictions. It can adapt its architecture to data sets whose size varies from a few thousand to seventeen million. Notably, a classification accuracy of over 96% is achieved for all benchmark data sets. The approach, the contributions, and the experimental results are detailed in Chapter 4.

1.1.3 Research Problem IV: Therapeutic and Prophylactic Protein Design with Deep Learning

PPI prediction allows exploring and screening a vast number of possibilities providing the researchers with the most promising cases. Although systems may be trained once and can screen millions of proteins in a fraction of a second, the solution is still restricted to known proteins. Ligands may be optimized, for instance, by performing amino acid substitutions and macromolecular docking, but these transformations are limited to refinements and fine-tuning [41]. In addition, new diseases are more likely to be cured by new proteins than by existing ones [42–44]. Therefore, a new protein must be designed from scratch considering the interaction with the pathogen, a problem known as de novo design [45–50].

The design of binder proteins for specific target proteins using deep learning is a challenging task that has a wide range of applications in both designing therapeutic antibodies and creating new drugs. Machine learning-based solutions, as opposed to laboratory design, streamline the design process and enable the design of new proteins that may be required to address new and orphan diseases. Most techniques proposed in the literature necessitate either domain knowledge or some appraisal of the target protein's 3-D structure. We introduce an approach for designing binder proteins based solely on the amino acid sequence of the target protein and without recourse to domain knowledge or structural information. Another problem specific to proteins is that there is usually more than one possible binder for a given target protein [51], which implies that training involves learning a multivalued function which associates more than one value to each input. Details of the research are presented in Chapter 5.

1.2 Research Scope

Thesis Concentration - The concentration of this work is on deep learning architectures (i) to predict protein-protein interactions and (ii) to facilitate protein design.

Experimental evaluations – We employ the following data sets in our experiments:

- Richoux *et al.* data set [52]: This data set consists of 16,210 unique proteins with a maximum length of 1166 amino acids composing of 104,262 pairs in total, among which 39,412 are binding proteins, that is, proteins with the ability to form either transient or long-lived complexes. This data set is a strict data set in the sense that there is no data leakage [53] in between the training set, the validation set and the test set.

- Guo's benchmark data set [23]: This data set was employed by Guo *et al.* It consists of 20,971 interactive protein pairs (positive examples) and 31,496 non-interacting pairs (negative examples), in which each protein consists of, at most, 1,200 amino acids.

- Database of Interacting Proteins (DIP) data set [54]: This data set was first introduced by Xenerios *et al.* in 2002 and is also a benchmark data set. Multiple releases have followed since. Version 20160430 is employed for this study. It consists of a strict data set of 3,242 protein pairs.

- Human Integrated Protein-Protein Interaction Reference (HIPPIE) [55]: This dataset, which is a benchmark data set, consists of 52,467 proteins pairs. A confidence score is associated to each pair.

- INWEB dataset [56]: The INWEB dataset was employed in Li's study, and it is one of the largest PPI data sets available. As with the HIPPIE dataset, the pairs are divided between low quality and high quality; the quality threshold was fixed at one. The high-quality dataset consisted of 8,672 distinct proteins, whereas the low-quality dataset contained 15,288 proteins, resulting in 141,354 protein-protein interactions for the former and 380,318 interactions for the latter.

- STRING dataset [57]: The STRING database is a large-scale repository used for studying protein interactions. It only consists of positive examples, meaning that non-interacting proteins are absent. It contains a subset of 19,356 *Homo sapiens* proteins involved in 5,879,727 PPIs.

- B-STRING dataset: The B-STRING database is constructed based on the known PPIs in the STRING dataset together with 5,879,727 negative examples (non-interacting

proteins) in order to create a balanced dataset. The detailed procedure for producing the B-STRING dataset is presented in 4.

Evaluations - Experimental results related to PPIs are evaluated with the following metrics: accuracy, F1 score, ROC curves, and precision and recall curves, as is standard in the literature. For our future work in protein design, the correctness of the predicted amino acid sequence will be evaluated with the sequence accuracy, the BLEU score (a metric from NLP) and metrics such as the Smith–Waterman and the Needleman–Wunsch similarity measures. These bioinformatics measures will enable us to find the optimal global alignment between the ground-truth sequence and the predicted sequence and find an optimal local alignment.

1.3 Research Methodology

Because we attempt to achieve better performance than existing methods, we follow a quantitative methodology that is commonplace in machine learning. We initiated our work with a literature review and data collection. We have designed a first deep neural network for PPI predictions that takes folding and self-binding into account. Considering that parameter selection is often challenging and that the size and the complexity of PPI data sets may vary considerably, we next designed a second deep network that automatically adapts its depth and complexity to the size and complexity of the dataset employed for training. In addition, we have developed an approach based on transformer architectures for protein design, utilizing binding scores and priors during learning.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 introduce essential background concepts regarding proteins and their interactions, as well as the use of conventional machine learning techniques and deep learning approaches in this domain. Chapter 3 is devoted to protein-protein interactions with folding and self-binding primary sequences.

Chapter 4 discussed the adaptation to complexity; particularly data generation and deep learnable architecture for PPI predictions. Chapter 5 described our solutions for the multivalued protein binder design problem. Chapter 6 concludes this thesis and highlights our future work.

Chapter 2

Background and Related Work

Recall that this thesis focuses on PPI prediction and PPI design, through the development of novel deep learning solutions. Predicting PPI concentrates on obtaining the binary binding result given by a pair of proteins. The main motivation for predicting PPI is that PPI can be used for predicting protein function [58–60] and developing drug targets [61,62]. In contrast, the design of PPI has the purpose of generating a protein which binds to a target protein. As for PPI design, it may be utilized to design novel binders and optimize existing binding interactions [63]. This chapter introduces essential background concepts to address these two research challenges.

We begin the chapter with a discussion on proteins, PPI prediction and PPI design in Section 2.1. Following that, we introduce conventional computational studies associated with the prediction of PPI and the design of PPI. Specifically, in Section 2.2, we introduce state-of-the-art computational methods used in both types of tasks.

Although remarkable success has been achieved by conventional computational methods, they have several shortcomings. Indeed, conventional computational methods often lack precision [63] requiring intensive experiments to optimize the result. In contrast, deep learning methods have outperformed traditional algorithms in many domains. To this end, we review the most relevant deep learning studies in Section 2.3.

2.1 Biological Background of Proteins

2.1.1 Protein: A chain of amino acids

To illustrate proteins, we need to explain the basic elements composing the proteins which are amino acids. Amino acids are organic compounds that comprise a carboxylate, an amino, and a side chain [64]. The side chain determines each amino acid. To date, more than 500 naturally existing amino acids have been identified [65] which seems to be a small fraction of the total. According to Kitano *et al.* , in homo sapiens alone, the total number of amino acids is estimated to be about 80,000 [66]. So far, 20 amino acids have been identified as standard amino acids which are repeatedly found in human protein structures. These 20 amino acids can be classified into two major categories: essential and non-essential amino acids. The former category includes valine, leucine, isoleucine, lysine, threonine, phenylalanine, methionine, histidine, and tryptophan. The latter group comprises glutamine, asparagine, glutamic acid, arginine, alanine, proline, cysteine, aspartic acid, serine, glycine, and tyrosine. Beyond the standard amino acids, selenocysteine has been found to be the 21st amino acid in homo sapiens [67]. Each amino acid mentioned above possesses a unique one-letter abbreviation used to represent it in sequence format. There are also several special letter codes in use. In some cases, it is not possible to differentiate aspartic acid and asparagine, or glutamic acid and glutamine. Therefore, code B is used for the amino acid that cannot be determined between aspartic acid and asparagine, and code Z is used for the amino acid that cannot be determined between glutamic acid and glutamine. One more special code, X, can refer to any amino acid and is only used when the exact amino acid cannot be determined. The amino acid code charts are shown in Table 2.1.

Amino acids can be linked through chemical bonds. Two major reactions in which this happens are the condensation reaction and cysteine oxidation. In the condensation reaction, the first amino acid loses its hydroxide (-OH) and the second loses its hydrogen (-H) to form a molecule of water and connect the two amino acids. This connection is known as an amide or peptide bond. Besides the peptide bond, another major bond that links amino acids is the disulfide bond that forms during cysteine oxidation. In this reaction,

Amino Acid	Code	Abbreviation
Alanine	A	Ala
Cysteine	C	Cys
Aspartic acid	D	Asp
Glutamic acid	E	Glu
Phenylalanine	F	Phe
Glycine	G	Gly
Histidine	H	His
Isoleucine	I	Ile
Lysine	K	Lys
Leucine	L	Leu
Methionine	M	Met
Asparagine	N	Asn
Proline	P	Pro
Glutamine	Q	Gln
Arginine	R	Arg
Serine	S	Ser
Threonine	T	Thr
Selenocysteine	U	Sec
Valine	V	Val
Tryptophan	W	Trp
Tyrosine	Y	Tyr
Aspartic acid or Asparagine	B	Asx
Glutamic acid or Glutamine	Z	Glx
Any amino acid	X	Xaa

Table 2.1: Code and abbreviation for homo sapiens amino acids

two molecules of cysteine lose their hydrogens and become linked by the disulfide bond. This reaction leaves a connected single molecule, called a cystine.

Through the two reactions described above, amino acids can be linked into a long chain, which is a protein sequence. Since the amino acids in this sequence have lost their hydroxide and hydrogen, they are now referred to as amino acid residues. Therefore, the amino acid sequence for each type of protein is unique. This sequence determines the shape, biochemical properties, and cellular function of the stand-alone protein [68]. This characteristic of protein is of primary importance and justifies the foundation of our work, which is to use the amino acid sequence as our only information source.

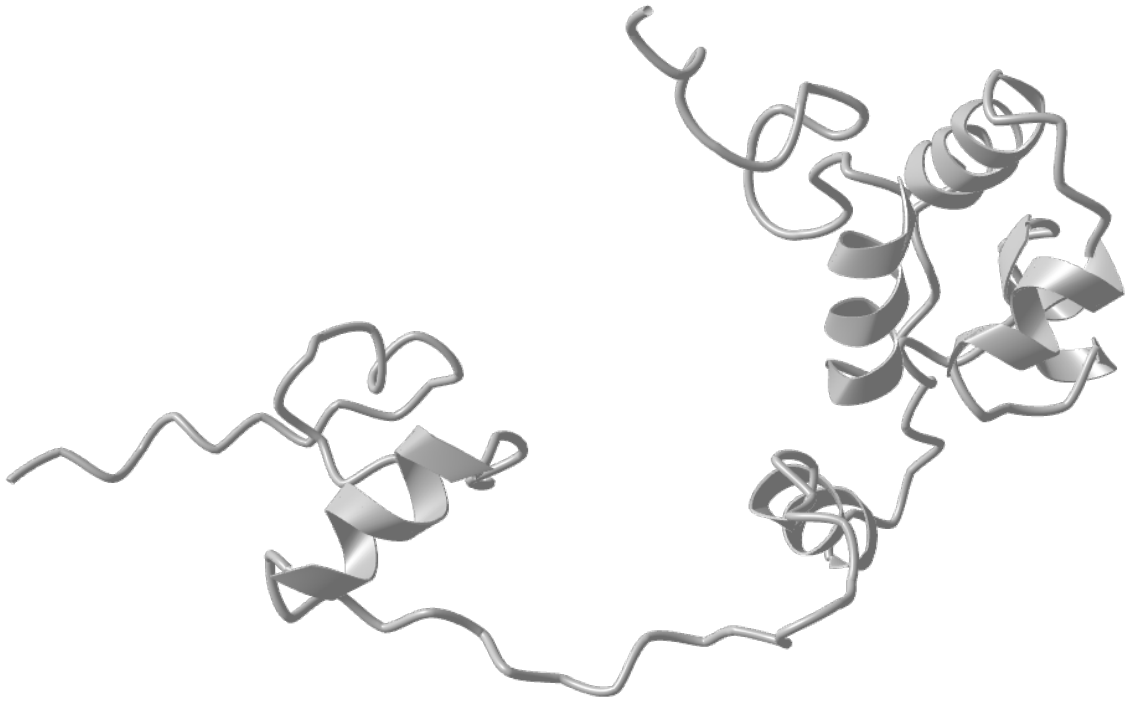


Figure 2.1: 3D Structure of the protein *multiprotein-bridging factor 1* acquired from iCn3D. [2].

Protein is present in all life forms on earth. It provides nutrition and structural support, and it catalyzes most of the biochemical processes in the cell [69]. To perform these

functions, proteins must fold into unique three-dimensional shapes known as their native states [70]. For instance, in Fig. 2.1, the protein *multiprotein-bridging factor 1* folds into a three-dimensional structure from a linear amino acid sequence. The structure of a protein can be categorized into four levels: primary structure, secondary structure, tertiary structure, and quaternary structure. The primary structure, also known as primary sequence, refers to the amino acid sequence of a protein which determines the shape and function of the protein [68]. The secondary structure of a protein is formed by stable folding patterns that happen only in neighbouring regions. These folding patterns are caused by hydrogen bonding between amino groups and carboxyl groups, known as $\alpha - helix$, $\beta - sheet$, and turns. The main characteristic of the secondary structures is that they can repeatedly occur in the same protein since secondary structures are local structures. The tertiary structure is obtained from a higher perspective and refers to the general shape of a single protein molecule. The tertiary structure controls the basic function of the protein. The next level of protein structure is the quaternary structure, which is made up of several protein chains that function as a single protein complex.

Another critical condition for protein functions is that most of them require cooperation between two or more proteins. This cooperation is steered by protein-protein interactions.

2.1.2 Protein-Protein Interaction

Protein-protein interactions (PPI) refer to physical contacts between two or more proteins and are responsible for most cellular processes [71–74]. According to Phizicky *et al.* [75], the properties of PPIs include modifying the structural features of enzymes, enabling substrate channelling, creating novel binding sites for small ligands, and suppressing proteins.

Understanding PPIs is a phenomenon in inferring the functions of a protein because the functionality of a protein can be analyzed using evidence of its interactions with other proteins. The study of PPIs also improves the comprehension of cellular biochemistry [76].

To categorize the PPIs, there are various standards regarding distinct aspects. The first rule is to differentiate interactions by whether they are constituted by identical proteins. For this, the quaternary structures of protein complexes provide the best perspective.

Here, we can determine the type of a given PPI from the complex formed by the protein subunits. If the protein subunits are from one single type, we recognize the complexes as homo-oligomers. Conversely, protein complexes that are constructed of various types of subunits are classified as hetero-oligomers [77].

The other classification method is the duration time of a complex formed by protein subunits. If the complex lasts for only a short period, this indicates that the proteins only interact briefly. These brief interactions are usually reversible and are recognized as transient interactions. For instance, chemokines are small proteins which act as chemoattractants. They guide the direction of cell movement. The cell migrates to the source of chemokines by the ascension of its concentration. This signal can only be identified by chemokine receptors located on the surface of cellscitemurphy2000international, murphy2002international. Therefore, cell migration is directed by the interactions between chemokines and their receptors, known as transient interactions [78]. In comparison, the interactions among protein subunits are referred to as stable interactions, since these permanent protein complexes, such as cytochrome c [79] and ATPase [80], last a long time.

Many in-lab methods for PPI detection have been developed. These can be divided into two categories: in vivo method and in vitro method. The best-known study using in vivo methods might be the Yeast two-hybrid (Y2H) technique, which was first proposed by Field *et al.* in 1989 [81]. The Y2H approach utilizes the biological characteristic of yeast that if a protein A in its Gal4 DNA-binding domain interacts with a protein B in its Gal4 activation domain, a transcription of the reporter gene will appear. With the presence of this gene, we can imply whether the protein pair is interactive or not. The development of this method has undoubtedly accelerated the progress of screening protein interaction. However, since the Y2H technique employs yeast as the host system, it encounters the problem of a high false negative rate when applicated to mammalian proteins [82].

For in vitro approaches, one of the earliest attempts is Tandem affinity purification–mass spectroscopy (TAP-MS), first carried out in 1999 [83], which is mostly used in detecting stable interactions. The advantage of this approach is that besides identifying protein interactions, it can be used to test the activeness of the protein complexes formed by the interactions [22]. However, since TAP-MS contains a purification process, interactions may be washed out and missed. Also, TAP-MS is unable to detect transient interactions [84].

Another widely used in vitro approach is DNA microarray [85]. The primary idea underlying DNA microarray is that the subunits in a protein complex are highly related to the gene expression of these subunits. Therefore, we can compare the gene expressions of target protein pairs with those of random noninteracting protein pairs. As shown in the study [86], with this approach, we can use the most obvious coexpression to identify permanent protein complexes. The DNA microarray method does have the drawback that, like TAP-MS, it lacks the ability to detect transient interactions. In contrast with the above methods, which target stable and permanent interactions, the affinity chromatography approach has the advantage of high sensitivity and can be used to detect even the weakest interactions, according to Rao *et al.* [22]. However, this advantage is associated with its drawback, that this high specificity among proteins causes a high false positive rate.

2.2 Conventional Computational Approaches

As illustrated above, the research into detecting PPI has been very successful. However, the experimental studies remain prone to errors [87, 88]. Moreover, although these approaches have improved efficiency and obtained high-throughput performance, we are still unable to go through all possible PPIs in even a single species using only experimental approaches. According to Hosur [89], a species with 10,000 genes can form a set of possible PPI containing more than 50 million interactions. Therefore, the study of a computational approach to PPI is not only for the purpose of improving efficiency but also because the enormous number of possible interactions makes it an impossible problem for experimental methods.

In this section, we will review the computational studies of PPI from two perspectives: the prediction of PPI and the design of PPI. The prediction of PPI is to determine whether a protein pair is interactive. The design of PPI is to create a protein de novo or modify an existing protein to enable it to interact with a target protein. PPI prediction allows large-scale screening of PPI in a short time and can help with identifying whether two arbitrary proteins can interact. PPI design is mainly used for creating novel proteins to interact

with a target protein, which has significant application in drug target development.

Before our review of computational approaches, we will review some conventional machine learning techniques, since they are widely used in these computational methods. Moreover, these machine learning techniques usually act as the core classifier in the computational approaches, so knowledge of them is required to fully understand these computational methods. Background information about machine learning will be provided in the first subsection, followed by the computational approaches.

2.2.1 Conventional Machine Learning Techniques

2.2.1.1 Decision Trees and Random Forests

Many computational PPI prediction studies have employed random forest approaches as their backbone machine learning models [90–92]. There are a few possible reasons for this. First, the performance of random forest is generally good and stable. Second, unlike other black-box machine learning methods, random forest brings interpretability to the machine learning models obtained. Also, as a bootstrap aggregation approach, random forest increases the stability of the model and reduces the variance. Therefore, a random forest model is less likely to overfit.

To better demonstrate how the random forest achieves these advantages, we should explain the decision tree, which is a basic unit of the random forest. As described by Bishop [4], a decision tree can be viewed as "a sequential decision-making process corresponding to the traversal of a binary tree". A binary tree is composed of its root node, intermediate nodes, leaf nodes, and decision rules. The root represents the initial state of the tree. By adding new decision rules, we obtain leaf nodes and intermediate nodes. Leaf nodes are the nodes where the tree stops growing and makes the final predictions. The intermediate node, on the other hand, can be further divided and grow other nodes. Optimizing a tree is usually done using a greedy strategy, which is to make one split at a time. At every step, the optimal split can be found with an exhaustive search. However, a problem remains: which criteria to use to determine an optimal split. For one of the most popular decision tree algorithms *classification and regression trees (CART)* [3], the metric utilized is the

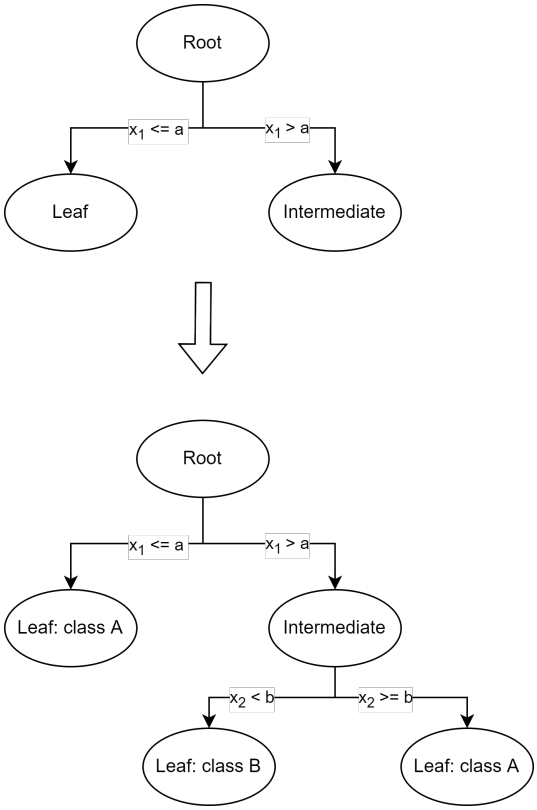


Figure 2.2: Illustration of a decision tree and its corresponding growing process [3].

Gini index, which is calculated in Eq. 2.1. In this, $p_{\tau k}$ is the proportion of data points classified as category k given by region R_τ.

$$Q_t(T) = \sum_{k=1}^{K} p_{\tau k}(1 - p_{\tau k}) \tag{2.1}$$

Random forest, as suggested by its name, comprises a set of decision trees. It was created by Ho *et al.* in 1995 [93]. During the training time, the training data are sampled with replacement into a set of datasets by bootstrapping. Since each dataset obtained is slightly different from the rest, each trained decision tree is also distinct. In addition, the trees utilize a subset of features and are not pruned [94]. In this way, the over-fitting phenomenon occurring in decision trees is mitigated. The training process is shown in Fig.

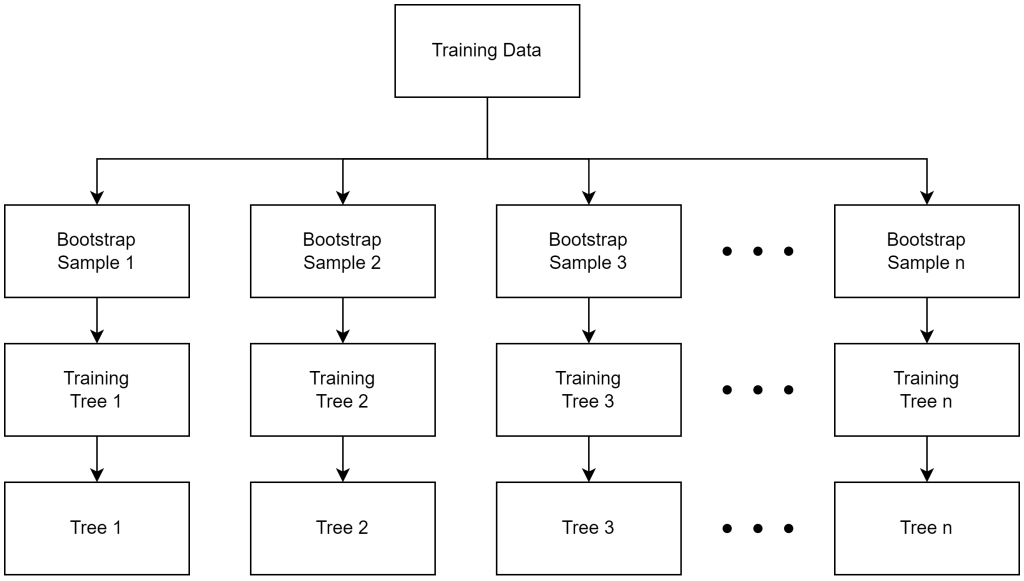


Figure 2.3: Illustration of the training process of a random forest [4].

2.3. In the prediction stage, the decision trees in the random forest work individually first. Then, the prediction of each tree is collected, and the final result is obtained by majority vote, as shown in Fig. 2.4.

2.2.1.2 Naive Bayes

In Liu *et al.* 's computational PPI prediction study [95], the likelihood of both interactive pairs and non-interactive pairs was calculated based on their co-evolutionary divergence information. Subsequently, the likelihood was used to train a Bayesian classifier. The major advantage of a Bayesian classifier is that it does not rely on large amounts of training data, which is particularly useful during the early development of computational PPI prediction because few large datasets are available.

The Bayesian classifier is constructed from the Bayes theorem, named after Thomas Bayes, which describes the probability of an event given the probability of a related condition, as shown in Eq. 2.2. Here, $P(B)$ is the evidence, while $P(A)$ is the prior knowledge. The term $P(B|A)$ is the likelihood, which indicates the probability of observing B given

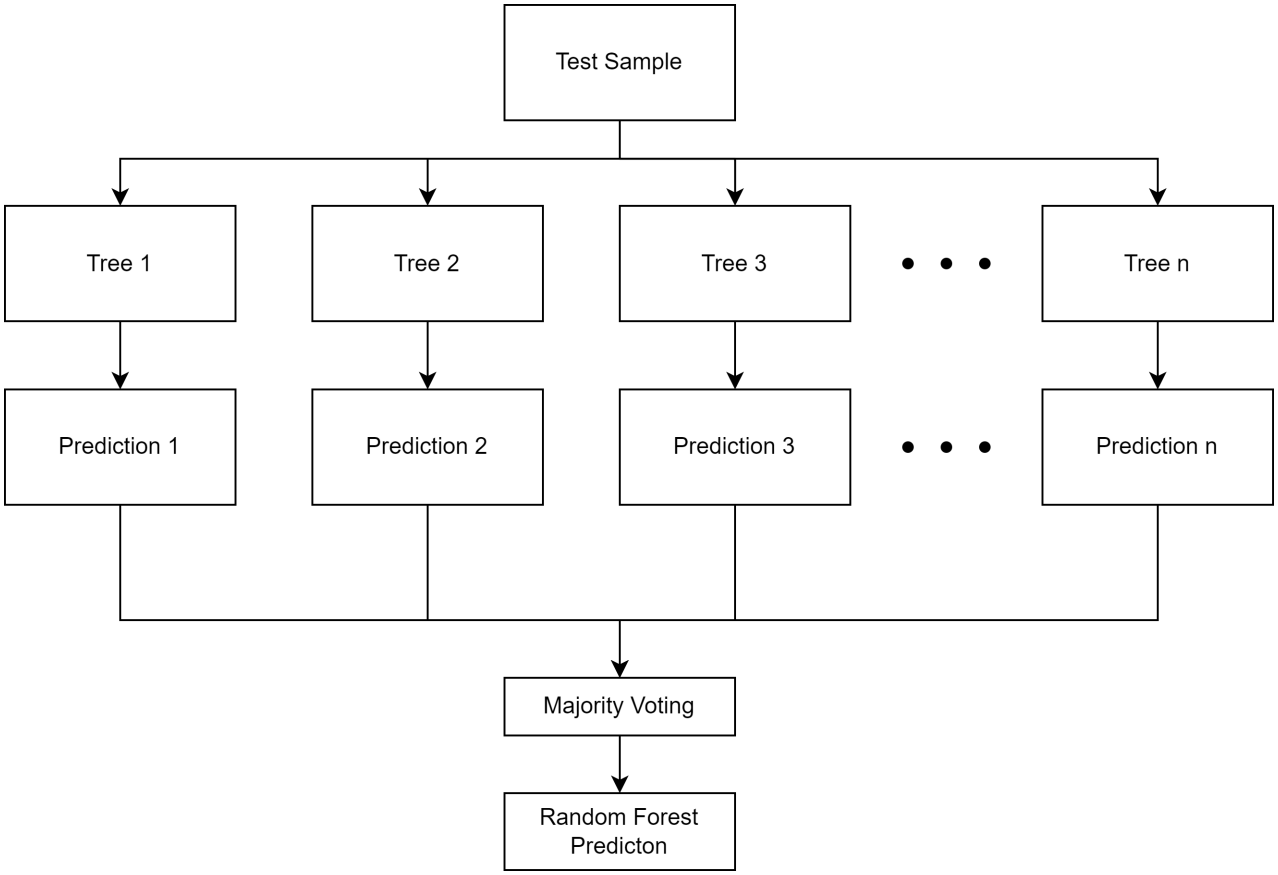


Figure 2.4: Illustration of the prediction of a random forest [4].

the condition of A.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.2}$$

This theorem is utilized in the classification problem. Given $\mathbf{x}$ as an instance to be classified, which is also the evidence, an instance $\mathbf{x}$ as a vector holds n attributes. This can be represented as: $\mathbf{x} = \{x_1, x_2, ..., x_n\}$. We would like to obtain the probability of this instance belonging to a certain class c_j in all possible K categories, which is represented as $P(c_j|\mathbf{x})$. Then, we can express the problem by the Bayes theorem as Eq. 2.3.

$$P(c_j|\mathbf{x}) = \frac{P(\mathbf{x}|c_j)P(c_j)}{P(\mathbf{x})} \tag{2.3}$$

By statistically analyzing the dataset, we can obtain the probability of each class c_j, referred to as prior knowledge. Another component in the equation that needs to be solved is $P(\mathbf{x})$. Since $P(\mathbf{x})$ is independent from any class and from the values of its attributes, the term $P(\mathbf{x})$ is constant, and in practice, it is ignored. The last component to be solved is the likelihood function $P(\mathbf{x}|c_j)$. First, we combine the two components in the numerator as Eq. 2.4.

$$P(\mathbf{x}|c_j)P(c_j) = P(\mathbf{x}c_j) = P(c_j, x_1, x_2, ..., x_n) \tag{2.4}$$

Eq. 2.4 can be extended into Eq. 2.5 by repeatedly applying the conditional probability.

$$\begin{aligned} P(c_j, x_1, x_2, ..., x_n) &= P(x_1|x_2, ..., x_n, c_j)P(x_2, ..., x_n, c_j) \\ &= P(x_1|x_2, ..., x_n, c_j)P(x_2|x_3, ..., x_n, c_j)P(x_3, ..., x_n, c_j) \\ &= \cdots \\ &= P(x_1|x_2, ..., x_n, c_j) \cdots P(x_{n-1}|x_n, c_j)P(x_n|c_j)P(c_j) \end{aligned} \tag{2.5}$$

Due to the assumption that all attributes in the vector $\mathbf{x}$ are mutually independent and only conditional on category c_j, each term in Eq. 2.5 can be simplified as Eq. 2.6.

$$P(x_i|x_{i+1}, ..., x_n, c_j) = p(x_i|c_j) \tag{2.6}$$

If we apply this process to every term in Eq.2.5, we can obtain Eq.2.7.

$$P(c_j, x_1, x_2, ..., x_n) = P(c_j)\prod_{i=1}^{n} P(x_i|c_j) \tag{2.7}$$

Bring Eq.2.3 to Eq.2.7 together we can have the proportional substitution of Bayes equation as Eq.2.8.

$$P(c_j|x_1, x_2, ..., x_n) \propto P(c_j) \prod_{i=1}^{n} P(x_i|c_j) \tag{2.8}$$

To predict the category that an instance belongs to, we select the category c_j, with which we obtain the largest $P(c_j) \prod_{i=1}^{n} P(x_i|c_j)$ which can be expressed as Eq. 2.9.

$$\hat{y} = \underset{j \in \{1,2,...,K\}}{\operatorname{argmax}} P(c_j) \prod_{i=1}^{n} P(x_i|c_j) \tag{2.9}$$

2.2.1.3 Support Vector Machines

PPI prediction is equivalent to binary classification in that the given pairs are divided into interactive pairs and non-interactive pairs. For binary classification tasks, a support vector machine (SVM) [96] is one of the most dominant supervised machine learning algorithms, which explains its many applications of it in PPI prediction [10, 20, 97]. The major idea of an SVM is that it regards the training data as points in a space, then learns a hyperplane to separate the two categories. Generally, our target is a model which can be generalized outside the training data. For an enhanced generalization ability, the final optimizing goal for an SVM is to obtain a hyperplane that maximizes its distance from the nearest data point. The obtained hyperplane is also known as a maximum-margin hyperplane. With this, we would expect that given an arbitrary new data point, it can be classified with more confidence.

The above concepts can be used to describe the optimization of an SVM. First, if the labels for a binary classification problem are specified as $+1$ and -1, and b as a bias term, the constraint of the optimization is presented as Eq. 2.10.

$$\begin{aligned} w^T x - b \geq 1, \text{if } y_i = 1 \\ w^T x - b \geq 1, \text{if } y_i = -1 \end{aligned} \tag{2.10}$$

Then the distance D between two hyperplanes is calculated as Eq.2.11.

$$D = \frac{2}{\|w\|} \tag{2.11}$$

To maximize D, the optimization goal is then to minimize w subject to Eq.2.10.

As a linear classifier, the SVM encounters certain problems. The first one is that for some of the data, there exist outliers or errors in labelling. Therefore, in reality, an SVM sometimes cannot perfectly separate the given data points. Because of this, a hard margin is not appropriate, so a soft margin is introduced. Instead of optimizing the problem with a constraint, the constraint is replaced by a hinge loss, as shown in Eq. 2.12.

$$l_{hinge} = \max(0, 1 - y_i(w^T x_i - b)) \tag{2.12}$$

The hinge loss for the entry, classified correctly, equals 0. In the optimization of an SVM with soft margins, the goal becomes a combination of two tasks: maximize the margin and misclassify as few samples as possible. A parameter λ is introduced to control the balance. Finally, the optimization goal is described as Eq. 2.13.

$$\lambda \|w\|^2 + \left[\frac{1}{n}\sum_{i=1}^{n} \max(0, y_i(w^T x_i - b))\right] \tag{2.13}$$

The other problem for a linear classifier is that some of the classes may not be linearly separable. To solve this problem, in 1992 Boser *et al.* [98] suggested introducing the kernel trick to leverage the SVM into a nonlinear classifier. The procedure is to project every data point to a higher dimensional space using a nonlinear kernel function. One of the most powerful and commonly used kernels is the Gaussian radial basis function, Eq.2.14, while σ is a free parameter.

$$K(x_1, x_2) = \exp(-\frac{|x_1 - x_2|^2}{2\sigma^2}) \tag{2.14}$$

2.2.1.4 Gradient Boosting

The other conventional machine learning technique that has been widely used in computational PPI prediction is gradient boosting [99]. Similar to a random forest, gradient

boosting is an ensemble learning method. What differentiates the gradient boosting approach from the random forest is that it is a boosting method rather than a bagging method. Random forest, as a bagging method, trains multiple decision trees or regression trees simultaneously and obtains the final results by majority voting. In contrast, the boosting method trains sequentially to learn the residual errors of the previous predictor. This process can be expressed by Eq. 2.15, in which f_m represents the ensembled model at iteration m and t_m indicates the last-trained learner, while η is the learning rate.

$$f_m = f_{m-1} + \eta t_m \tag{2.15}$$

We aim to fit the latest added learner into the fastest direction to lower the loss, which is given by the negative derivative of the loss calculated in the previous iteration. The computation of t_m is presented by Eq. 2.16.

$$t_m = -\frac{\partial loss}{\partial f(m-1))} \tag{2.16}$$

Empirically, the boosting approaches tend to surpass bagging methods by their prediction power. Boosting methods also have the advantage of being easy to interpret by their sequential alignment with learners. However, gradient boosting has the flaw of being sensitive to outliers because of its sequential optimization process, in which each learner added is dedicated to compensating for the errors left by its predecessors. Therefore, gradient boosting models might end with fitting the outliers.

2.2.2 Computational Prediction of PPI

2.2.2.1 Three Dimension Structure Based Approaches

This category of PPI prediction approaches follows an intuitive idea that the interactions between pairwise proteins are caused by the three-dimensional structure of a pair of subunits that interact with each other (also known as interface residues) [100]. Therefore, different pairs of proteins with a similar binding site containing the same or similar interface residues should also interact [22]. A compelling study using this approach was carried

out by Lu *et al.* in 2002 [101], taking protein-protein interfacial energy into account. The interfacial pairwise potentials derived were used to construct a dataset containing multimeric protein structures. This benchmark set included 40 homodimers, 15 heterodimers and 69 monomers. The interfacial knowledge obtained from this set was utilized, along with the novel threading algorithm, to scan through all possible pairs of yeast proteins. They obtained 2865 PPIs in total and were able to confirm 1215 of them in the DIP dataset [54]. Lu *et al.* 's approach is a promising attempt at using existing protein structure templates for screening possible candidates. However, since the screening can only be done with the knowledge of binding sites, it can only search PPIs with this information and cannot identify PPIs with novel binding interfaces.

Another critical research study utilizing a three-dimensional structure was proposed by Joan *et al.* in 2013 [90]. The novelty of this study is that, as a three-dimensional computational method in PPI, it also takes the non-interacting region into consideration. Joan *et al.* first constructed a balanced dataset with interactive pairs and non-interactive pairs. The interactive pairs were retrieved by Biological Interactions and Network Analysis [102], which integrates multiple sources of PPIs including HPRD [103], MINT [104], BioGrid [105], IntAct [106], and MIPS [107]. The negative pairs were collected from the Negatome database. Next, three different structures were utilized to obtain protein signatures: domains, loops, and loops belonging to the same domain. A combination of protein signatures, one from each partner, was used to construct each interaction signature. If an interaction signature was observed in the positive reference set, it was noted as positive. Based on this, a set of statistical features were calculated and employed as classifiers for interactive or non-interactive.

To evaluate each type of interaction signature, Joan *et al.* also trained a set of random forest classifiers equipped with different interaction signatures. The best performance was obtained by joining all interaction signatures. Joan *et al.* 's study provides a unique perspective for PPI prediction using a set of well-designed statistical features which separates interactive and non-interactive pairs. This study also reveals significant domain knowledge for PPI. As claimed, the interacting region is not the only factor that determines an interaction; the rest of the protein is also important [90]. This finding supports the computational approaches which utilize entire protein information. For instance, the primary

sequence–based approaches, which will be reviewed in the next subsection, use the entire primary sequence of proteins as input.

2.2.2.2 Primary Sequence Based Approaches

According to Anfinsen *et al.* , the protein sequence also known as the primary structure provides sufficient information for determining the likelihood of interaction between a pair of proteins [108]. Anfinsen *et al.* 's study provides a theoretical foundation for computational methods that utilize only primary sequence information. An early attempt at using primary sequence to predict PPI was conducted by Bock *et al.* [109] in 2000. This study addressed the research question of whether the PPI prediction problem can be directly solved using the primary sequence and associated data such as physicochemical properties. The solution provided by Bock was to construct a feature representation for each protein sequence in a pair by encoding every residue in the sequence into a physicochemical property vector, including charge, hydrophobicity, and surface tension. Since proteins have widely varying native lengths, the study also normalized the length of the obtained representations by mapping them onto a fixed-length interval. Next, for each protein pair, the representation was assembled by concatenating the representation for every partner of the pair. A support vector machine was trained on the obtained representations of the protein pairs.

Bock *et al.* 's method showed promise, performing with 80.96% accuracy. This approach is a remarkable milestone in computational PPI prediction which provides a novel and elegant solution. However, it may have a drawback in that the amino acid sequences in the negative dataset are generated randomly. These artificial proteins possess an inherent bias, and exactly based on which, the non-interactive pairs were designed. This bias may not be the authentic reason for the non-interacting phenomenon. Worse, this inherent bias could be learned by an SVM and used as the hyperplane to separate the interactive and non-interactive pairs.

In 2012, Liu *et al.* [95] carried out a novel sequence-based co-evolution algorithm to predict human PPI. This algorithm, known as co-evolutionary divergence (CD), is built on the assumption that protein pairs with similar substitution rates have higher probabilities

for interaction. Liu *et al.* first retrieved 16,299 proteins from Homo sapiens and their orthologous proteins from other 13 vertebrates, for a total of 172,338 protein sequences. Next, the software ClustalWe was employed to conduct pairwise alignment among these protein sequences. Using these alignments, the number of amino acid substitutions per site was calculated and used as the substitution rate. The absolute value of the substitution rate difference between protein pairs was used to measure the CD of the interactive pairs and non-interactive pairs. Then, the CD values were used to calculate the likelihood of protein interaction given by a CD value, which was done by dividing the proportion of interactive protein pairs within a bin of CD values by the proportion of non-interactive pairs within the same bin of CD values. Finally, the likelihood was used to obtain a naive Bayes classifier. Liu *et al.* 's approach successfully utilized the protein sequence to achieve a CD-based Bayesian classifier and outperformed its predecessor. However, its performance was not precise enough; the highest AUC score obtained was 0.783. Also, this approach cannot be applied to an arbitrary protein if there is no orthologous protein. Although they tried to solve this problem by setting the likelihood of a missing orthologous protein at 1.0, this remedy does not provide a meaningful approximation.

2.2.2.3 Gene Neighboring Based Approaches

The history of the study of a gene neighbouring approach for PPI can be traced back to 1998 [110] and one of the earliest genomic context–based PPI prediction attempts. The general concept of this approach is that when the genes responsible for certain proteins are located close to each other in the genome, this implies that the corresponding proteins are highly related and likely to interact. For instance, Yamada [111] pointed out that the adjacent bidirectional genes have a functional link. The inference of PPI using gene neighbouring can be strengthened when the neighbourhood relationship appears repeatedly across various genomes. Therefore, the quality of prediction using the gene neighbouring method depends on the number of genomes provided [112], and this approach is not applicable without knowledge of the related genomes. Another limitation is that the gene neighbouring in eukaryotes genomes is evident for PPI prediction, which prohibits its application in eukaryotes. In addition, although the simplicity of gene neighbouring is preferred, this characteristic causes a false negative rate because it frequently fails to recognize distantly

located genes [113]. Recently, Muley *et al.* found another drawback [114], that the choice of reference of genomes has a strong effect on the prediction result of gene neighbouring methods.

2.2.2.4 Phylogenetic Similarity Based Approaches

The phylogenetic similarity approach, also known as phylogenetic profile, can be regarded as an extension of gene neighbouring. The phylogenetic profile approach constructs a profile for each protein containing information about its presence across multiple organisms. The profiles are compared with each other to compute similarity scores. The higher the similarity score associated with a protein pair, the more likely it is that this pair would interact [115, 116]. The theoretical support for this approach is that proteins that are linked together by their functions tend to coexist during evolution [117]. Therefore, if the phylogenetic profiles of proteins are similar, it implies that the two proteins are likely to be linked in functions. The phylogenetic similarity approach can be used to detect the distant interactions that gene neighbouring fails to detect. It also has an additional ability, which is to discover the function of a protein.

However, this approach also possesses shortcomings. First, in contrast to gene neighbouring, a phylogenetic similarity approach requires full genome sequences, which are more difficult to access. Second, its performance depends strongly on the number of genomes provided [114]. Moreover, some of the essential proteins appear in most of the organisms with no detectible difference in their phylogenetic profiles, in which case this approach cannot be applied.

2.2.3 Computational Design of PPI

Although various computational methods have been developed for predicting PPI, they lack the ability to generate novel proteins and are thus limited to known proteins. Given a protein without such information, it is impossible for any PPI prediction approach to obtain a binder. Therefore, the computational PPI design comes into application. The existing approaches for computational PPI design can be roughly categorized into two

kinds: 1) PPI redesign, and 2) de novo PPI design. The first refers to the approaches that create variants based on known protein structures, including primary structures, while the second means designing a protein from scratch.

2.2.3.1 PPI Redesign

PPI redesign approaches can be further classified into three categories. The first optimizes the existing binding sites. One study attempting to modify the PPI was carried out by Chen *et al.* [118]. In this study, Chen *et al.* developed a computational approach which re-engineers a specific protein $Bcl-X_L$ to reduce its interactions and turn it into a high-affinity and selective binder of the BH3 region of proapoptotic protein Bad. First, manual structure inspection and modelling software Rosetta [119] are used to predict desired sequences features, which include the residues responsible for 1) maintaining the binding relationship with the target, which in this case is the BH3 region of protein Bad, and 2) imparting the specificity over an alternative, which is the BH3 region of protein Bim. The selected proteins are optimized as an integer linear program (ILP) [120] and generate a focused combinatorial library. The limitation that excludes this method from a general application is that it is designed with prior knowledge that is highly specific to the single protein region BH3.

The second category of approach in PPI redesign is to graft one side of a natural binding site to increase the affinity. One such study was conducted by Azoitei *et al.* [121], whose approach integrates the computational techniques with experimental selection to graft the epitope of HIV gp120, enabling it to bind the antibody b12 onto an unrelated scaffold protein. This method first searches the suitable scaffolds in the Protein Data Bank (PDB) [122]. Then, the scaffolds are modified and optimized by replacing native scaffold backbones with motif backbone segments. Next, in the stage of computational-guided library design, small libraries are devised based on the sampling of low-energy structures. The best models selected by Rosetta are screened. Similar to the approaches that optimize existing binding sites, this category of PPI redesign requires sufficient prior knowledge and relies on experimental selection procedures.

The third approach in PPI redesign is to graft both sides of an existing natural binding

site [121]. Potapov *et al.* carried out a study in 2008 based on the interaction between $TEM_1\beta - lactamase$ and its inhibitor protein. This study replaces a complete module of the interface between the two proteins with structural fragments from non-related proteins, achieving both high affinity and specificity. A possible drawback of this approach is that the replacing interfaces are ready-made templates retrieved from PDB, which limits it to existing PPI interfaces.

For our research, we intend to design a computational approach in which the designs generated are free from such restrictions, including protein structures and interface templates. Therefore, these approaches are not suitable for our research topic.

2.2.3.2 De Novo PPI Design

De novo PPI design has three categories. The first category is to design a cluster of residue interactions and search for the protein based on it. One general computational method based on residue cluster design was proposed by Fleishman [123] in 2011. This approach was used to produce the influenza hemagglutinin with high affinities. The core strategy behind this approach is to form high-affinity interactions at the interface between the proteins. The first step is to construct an interaction hotspot region with the single-residue docking function provided by RosettaDock software [124]. By doing this, the interactions of a larger number of surfaces on the target protein can be computed beforehand. This information allows all compatible rotamers for each residue in the hotspot region to be retrieved. At the same time, the high-shape-complementary configurations of the designed protein according to the target protein surface are computed by the coarse-grained docking technique. The two results obtained, the rotamers and the configurations, are combined. The rotamers are incorporated into the scaffold protein and then examined by both structural and energetic filters. The remainder of the residues in the scaffold at the interaction surface is designed and refined.

The other category of de novo PPI design is to search the binder with high affinity and, based on this, to obtain novel ones by optimization. Jha *et al.* carried out research within this category in 2010 [125], designing a computational protocol known as Dock, Design, and Minimize Interface (DDMI). In this approach, given a docked conformation for the

scaffold and the target backbones, a low-energy sequence for the docked conformation is generated by searching the side chain sequences and conformations. The whole process is accomplished iteratively. At the beginning stage of DDMI, the docking of the two proteins is enacted using randomization, including the position, orientation, and docking of the scaffold protein. During the iteration, the designs are optimized using a gradient-based approach and eventually settle in a low-energy state.

The study by Der *et al.* [126] belongs to the third category of de novo PPI design, which is to utilize the metal-binding sites and optimize the affinity around these sites. First, the two-residue cysteine/histidine zinc-binding sites are searched through the monomer scaffold surfaces. Then, the scaffold surfaces on the monomers are grafted with the obtained residues. Next, the two proteins are rotated to make a symmetric complex. A grid search is carried out for the second chain to obtain a proper zinc coordination geometry. The design is utilized by Monte Carlo simulated annealing. Similar to other computational methods, the obtained results are filtered by binding energy to obtain high-quality results.

The above methods all require sufficient prior knowledge of the backbone structure of the target proteins and the scaffolds. For instance, Fleishman *et al.* 's approach depends on searching the library of scaffolds and retrieving the complements of the target proteins. In addition, some of the approaches are highly specific to a certain type of PPI. The metal-binding method, even as a more general approach, is still restricted to nonpolar proteins [126].

2.2.4 Discussion

In subsection 2.2, we illustrated the widely used conventional computational approaches in PPI prediction and design. We first talked about the conventional machine learning techniques that have been employed in these approaches. Their advantages and limitations are summarized in Table 2.2. Then, we introduced the conventional computational PPI prediction methods, which are summarized in Table 2.3. Finally, we discuss the PPI designs with computational approaches which are summarized in Table 2.4.

Table 2.2: Comparison of conventional machine learning approaches

Algorithm	Advantages	Limitations
Decision tree	- High interpretability - Supports multiple types of data	- Limited prediction power compared to other methods - Depends on order of attributes selected for branching
Random forest	- Lower variance and risk of overfitting - High interpretability - Easily scales up to a large dataset - High time efficiency obtained by parallel training - Better performance than decision tree	- Number of trees must be empirically defined - Bias in estimating importance of attributes
Naive Bayes	- Good scalability - Requires less training data - Suitable for both binary and multi-class classification problems - Suitable for both continuous and discrete data	- Assumption of normal distribution - Prediction performance impaired by presence of dependency between attributes
Support vector machine	- Performs well in semi-structured and unstructured data	- Computationally expensive - Prone to noise - Low interpretability - Vanilla SVM can only classify binary problems
Gradient boosting	- Strong prediction power - High interpretability - Implicit feature selection	- Computationally expensive - Prone to noise - Low scalability

Table 2.3: Comparison of conventional computational PPI prediction studies

Study	Methodology	Advantages	Limitations
Lu *et al.*	3D structure	- Capability of large-scale screening	- Requires knowledge of existing binding sites - Unable to recognize unknown interactions
Joan *et al.*	3D structure	- Well-designed feature engineering - Balanced dataset - Supports proteins with variable lengths - Combining physicochemical properties	- Requires knowledge of existing binding sites
Bock *et al.*	Primary sequence	- Competitive performance compared to other methods based on 3D structure information - Computational efficiency - Easily accessed data format	- Artificial process of generating negative samples provides bias
Liu *et al.*	Primary sequence	- Computational efficiency - Supports proteins with variable lengths - Easily accessed data format	- Less compelling prediction performance than other studies
Dandekar *et al.*	Gene neighbourhood	- Heuristically indicates possible PPI	- Not applicable when knowledge of related genomes is absent - Fails to recognize distantly located genes - Choice of reference of genomes has strong effect on prediction result
Marcotte *et al.*	Phylogenetic similarity	- Able to detect distant interactions that gene neighbouring approaches do not - Discovers function of a protein	- Performance depends on number of genomes provided - Not applicable for cross-original proteins

Table 2.4: Comparison of conventional computational PPI design

Study	Methodology	Advantages	Limitations
Chen *et al.*	- PPI redesign - Optimize existing binding sites	- High affinity - Selective binding	- Requires strong prior knowledge - Only applicable to protein region BH3
Azoitei *et al.*	- PPI redesign - Optimize existing binding sites	- Integrates computational techniques with experimental selection to provide practical results	- Requires strong prior knowledge - Requires in-lab experiments
Potapov *et al.*	- PPI redesign - Graft both sides of existing natural binding site	- High affinity - Selective binding	- Depends on ready-made templates and thus limited to existing PPI interfaces
Fleishman *et al.*	- De novo design - Design a cluster of residue interactions then search	- High affinity	- Expensive computation - Requires strong prior knowledge
Jha *et al.*	- De novo design - Search then design	- High affinity - Can be continuously optimized by iteration	- Expensive computation - Depends on binder library - Not applicable for novel proteins
Der *et al.*	- De novo design - Metal-binding	- Supports most proteins	- Requires strong prior knowledge - Not applicable for polar proteins

2.3 Deep Learning Approaches

In recent years, both the computer vision and natural language processing industries, amongst others, have witnessed a tremendous surge of deep learning-based solutions [30, 31, 127]. Indeed, deep learning architectures have been successfully employed in numerous domains. For instance, Mesbaha *et al.* developed a novel approach for lip reading [128] in which the discrete Hahn polynomial [129] is adopted and combined with a convolutional neural network. Also, in the domain of the Internet of Things (IOT), Zhang *et al.* proposed an approach utilizing deep reinforcement learning to achieve lower energy consumption, while witnessing less delays and leading to lower monetary costs, when making optimal offloading decisions [130]. Deep learning architectures are also adopted in health care, including a solution where several hidden convolution layers is proposed for solving the seizure detection problem and successfully detected 94.6 percent of the 198 tested seizure records [131].

Given enough high quality data, deep learning approaches are often able to surpass conventional algorithms and machine learning approaches [32, 33]. As a result, attempts in the applications of PPI prediction and design have been carried out to benefit from the advantages of deep learning approaches [34, 36, 38, 47–49, 132–136]. With this motivation, we focused on applying deep learning algorithms to PPI prediction and design problems. Next, we review advanced deep learning techniques relevant for our research.

2.3.1 Advanced Deep Learning Techniques

2.3.1.1 Siamese Neural Network

The Siamese neural network is a specific category of deep learning neural network that contains two individual branches to process two different input vectors by shared weights. The results are computed based on the comparison or combination of the inputs. Fig. 2.5 shows the general structure of a Siamese neural network. This type of network was first proposed by Bromley *et al.* [5] in 1993 and was originally designed for recognizing handwritten checks.

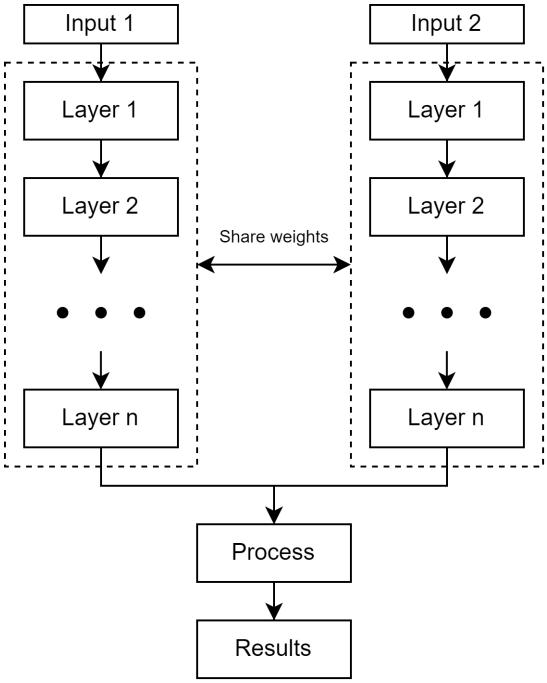


Figure 2.5: General structure of Siamese neural network [5].

For a Siamese neural network denoted as S, S contains twin networks S_r, S_l. During both training and inference, the weights are shared by the twin networks as $w(S_r) = w(S_l)$. Therefore, the computations in the two information flows are identical, which can be further denoted as $f_1(x_l)$ and $f_1(x_r)$, while the entire Siamese network can be aggregated as Eq. 2.17, which can be written into Eq. 2.18.

$$S(x_l, x_r) = f_2(S_l(x_l), S_r(x_r)) \tag{2.17}$$

$$S(x_l, x_r) = f_2(f_1(x_l), f_1(x_r)) \tag{2.18}$$

From Eq. 2.18, it is obvious that if a symmetric procedure is used as the final process function f_2 such as distance metrics, a Siamese neural network will also possess the symmetry. Therefore, it has been widely applied to applications related to comparisons, including handwriting recognition, face recognition, and fingerprint recognition.

When the final process function is replaced, a Siamese neural network can be used to

solve the PPI prediction problem [36, 38, 52]. There are two major advantages. First, the Siamese neural network provides an end-to-end structure for the task, with two parallel inputs associated with a single output, which suits the PPI prediction problem. Second, constructing a dataset for PPI prediction is more expensive than a regular dataset because of the requirements for highly specific knowledge and intense experiments, so deep learning applications in PPI prediction are sometimes limited to only small datasets. The Siamese neural network structure shares the weight in its branches, greatly reducing the number of parameters and alleviating the risk of overfitting.

2.3.1.2 Convolutional Neural Network

One of the most essential neural network structures employed in this thesis, especially for PPI predictions, is the convolutional neural network, which has drastically shaped modern techniques in computer vision. A variant, AlexNet [137], is one of the most influential studies and started the widespread trend of deep learning. By dividing the calculation into two and assigning them to different GPUs, Krizhevsky *et al.* made computation of a large-scale neural network feasible and achieved the best performance in the 2012 ISLVR challenge [138]. Since then, many other papers have been published based on convolutional neural networks including VGGNet [139], GoogLeNet [140], ResNet [141], DenseNet [142], and ConvNeXt [143]. The convolutional-based neural networks have been applied to various fields such as lesion detection [144], image segmentation [145, 146], and protein structure prediction [47]. A convolutional neural network is usually composed of two layers: convolutional layers and pooling layers.

In this thesis, the study target is the primary structure of protein only, which is a one-dimensional amino acid sequence. Therefore, the convolution layers employed in our studies are primarily one-dimensional convolution, which is explained below. The operation involved in the convolution layer is kernel convolution, which applies a sliding window to the input and conducts element-wise production with a summation. Denoting the input as s and the kernel as k, for an arbitrary position t in the input sequence, the convolution results can be calculated as Eq. 2.19.

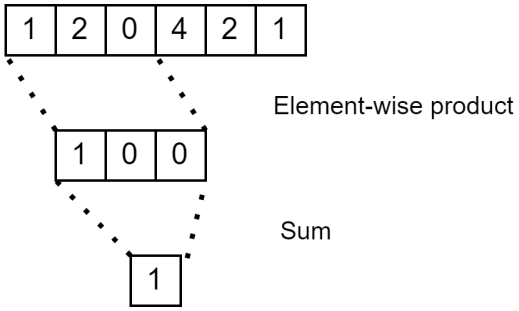


Figure 2.6: An example of one-dimensional convolution.

$$R[t] = \sum_{i}^{L} k[i]s[t+i] \tag{2.19}$$

With this operation, the convolution layer is able to extract features and provide translation invariance [147].

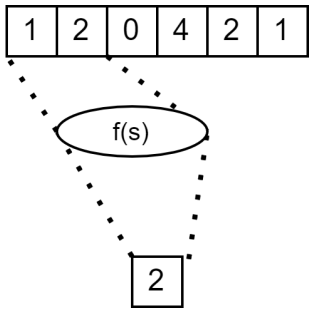


Figure 2.7: An example of the one-dimensional max pooling operation.

The other major component employed in a convolutional neural network is the pooling layer. A pooling layer functions similarly to a convolution operation by applying a sliding window. In contrast with a convolution layer, it usually applies a function to the submatrix rather than utilizing a kernel. There are two major types of pooling layers: the max pooling layer and the average pooling layer. The functions applied to the submatrix for the two categories are the max function and the average function, respectively.

2.3.1.3 Residual Neural Network

The design of a residual neural network was first proposed as ResNet by He *et al.* in 2016 [141]. This is an artificial neural network structure with skip connections. Its general structure is presented in Fig. 3.2. An improved version of ResNet is DenseNet, which contains multiple parallel skip connections [142]. Neural networks with skip connections are able to learn the residual between the output and the ground truth label which stabilizes the gradient during propagation through multiple layers. Denoting the input of layer i as x_i and the layer i as $f_i(x)$, we can calculate the input x_{i+1} for layer $i+1$ as Eq. 2.20.

$$x_{i+1} = x_i + f_i(x_i) \tag{2.20}$$

Similarly, the input x_{i+2} for layer $i+1$ is calculated as Eq.2.21.

$$x_{i+2} = x_{i+1} + f_{i+1}(x_{i+1}) \tag{2.21}$$

Utilizing Eq.2.21 to further expand Eq.2.20, we can have Eq.2.22.

$$x_{i+2} = x_i + f_i(x_i) + f_{i+1}(x_{i+1}) \tag{2.22}$$

Following this, we are able to generalize it to arbitrary layers j, k in the residual neural network as Eq. 2.23.

$$x_k = x_j + \sum_{i=j}^{k-1} F_i(x_i) \tag{2.23}$$

Then, we calculate the backward propagation of Eq. 2.23 as shown by Eq. 2.24.

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial x_k}(1 + \frac{\partial}{\partial x_j} \sum_{i=j}^{k-1} F_i(x_i)) \tag{2.24}$$

From the final Eq. 2.24, we can see that because of the existence of constant number 1, the loss is directly propagated from layer k to layer i without any intervention, which prevents the gradients from vanishing. On the other hand, the calculation of Eq. 2.24 is additive, which is smoother than the multiplicative calculation in other neural networks.

2.3.1.4 Recurrent Neural Network

The recurrent neural network (RNN) was designed by Rumelhart *et al.* in 1986 to process temporal sequences [6]. It augments conventional neural networks by introducing connections between adjacent time steps, which are implemented by maintaining an internal state. The internal state of RNN also called its memory, provides the neural network with a concept of time and enables it to anticipate future time steps [148]. A paradigm for the structure of RNN is presented in Fig. 2.8.

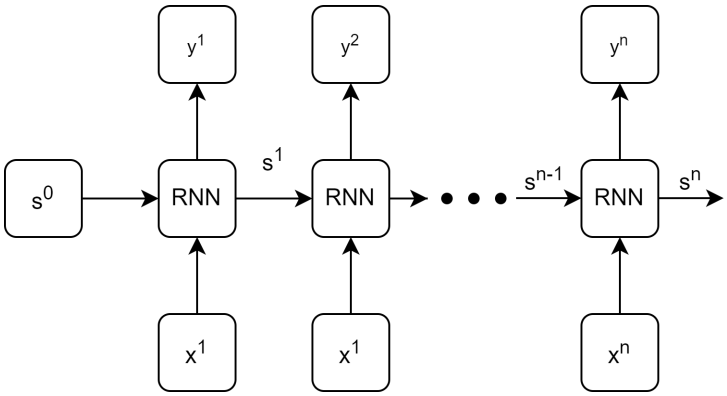


Figure 2.8: General structure of a recurrent neural network [6].

The calculation of an RNN can be described using the following steps. Denote input as x, output as y, internal state as s, and three learnable weights associated with them respectively as w_x, w_y, w_s. For an arbitrary time step t, the activation function a, and the bias as b, the internal state s^t is first expressed as Eq. 2.25.

$$s^t = a_1(w_s s^{t-1} + w_x x^t + b_1) \tag{2.25}$$

And the output at the current time step is calculated as Eq.2.26.

$$y^t = a_2(w_y s^t + b_2) \tag{2.26}$$

In RNN, the calculation in each time step is based on the previous time step. This is shown by Eq. 2.27, which is obtained by introducing Eq. 2.25 to Eq. 2.26.

$$y^t = a_2(w_y a_1(w_s s^{t-1} + w_x x^t + b_1) + b_2) \tag{2.27}$$

As proposed by Pascanu *et al.* in 2013 [149], the multiplicative process could cause the exploding and vanishing gradient problem in the backpropagation through time (BPTT) algorithm when optimizing RNN. To address this problem, Hochreiter *et al.* [7] carried out a variant of the RNN, which is the long short-term memory (LSTM). The LSTM primarily differs from a conventional RNN with its additional gates, which are responsible for controlling the flow of information. It also includes two internal states, the cell state and the history state. The general workflow of an LSTM cell is presented in Fig. 2.9.

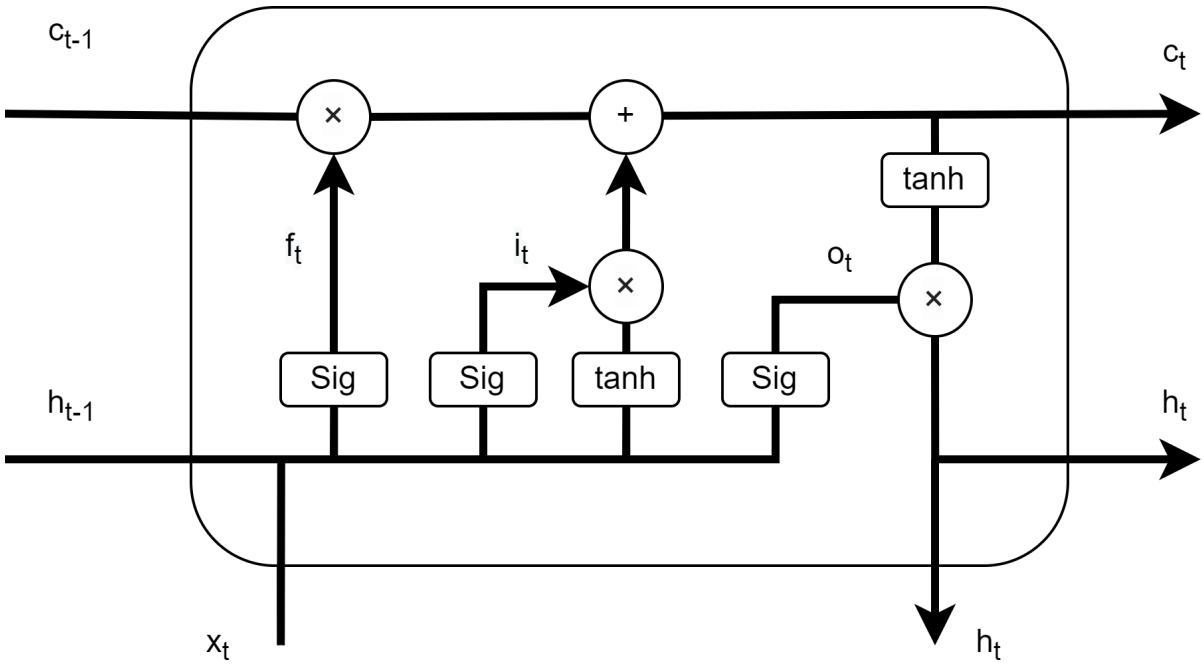


Figure 2.9: A long short-term memory cell [7].

The first gate that comes in the workflow is the forget gate, which decides what content in the previous history state h_{t-1} to add to its long-term memory, which is known as the cell state c based on its previous history state h_{t-1} and the current input x_t.

$$f_t = \text{sigmoid}(w_f[h_{t-1}, x_t] + b_f) \tag{2.28}$$

The forget gate f_t is expressed as Eq. 2.28, in which w_f is the learnable weights for the layer, b_f stands for the bias, and sigmoid is the sigmoid activation function. The output of Eq. 2.28 is bounded in $(0, 1)$, which is the proportion of the content to be preserved. Then another gate is calculated as Eq. 2.29 in a similar manner. This is the input gate responsible for controlling the information flow from the input to its long-term memory.

$$i_t = \text{sigmoid}(w_i[h_{t-1}, x_t] + b_i) \tag{2.29}$$

The updated long-term memory is composed of two parts, the reserved parts of the previous memory and the current input. The mathematical expression is Eq. 2.30. The *tanh* activation utilized in this equation is because its second derivative can be sustained for a larger range of input.

$$c_t = f_t * c_{t-1} + i_t * \tanh(w_c[h_{t-1}, x_t] + b_c) \tag{2.30}$$

After this, the output gate for regulating the information flow from its cell state to output is calculated as Eq. 2.31.

$$o_t = \text{sigmoid}(w_o[h_{t-1}, x_t] + b_i) \tag{2.31}$$

Finally, the history state h, which is also used as output, is updated from the long-term memory cell state c_t as Eq. 2.32.

$$h_t = o_t * \tanh(c_t) \tag{2.32}$$

2.3.1.5 Attention

The attention mechanism is a remarkable breakthrough invention in deep learning from the last decade. It is inspired by the biological nature of humans, based on the ability to top-down consciously concentrate on a part of useful information [150]. This enables humans to efficiently select high-value key points from massive resources. Many attention mechanisms in deep learning are designed based on this concept. One of the most influential studies of

the attention mechanism was carried out by Bahdanau *et al.* [8], who originally applied it to neural machine translation. It was soon introduced to other fields, including image caption generation [151, 152], recommendations [153, 154], and graph neural networks [155, 156].

The attention mechanism from Bahdanau *et al.* is shown in Fig. 2.10. The basic context is that the input for the attention layer is from a bidirectional RNN which is a sequence of hidden states $h_1, h_2, \cdots, h_T$. The goal of the attention layer is to generate the probability of the next word y_i, which can be described by Eq. 2.33.

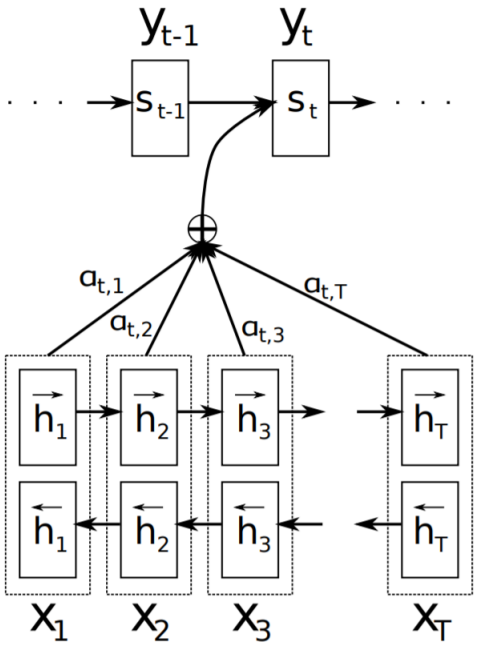


Figure 2.10: The graphical illustration of the attention used to generate the t-th target word y_t given by a source sentence $(x_1, x_2, \cdots, x_T)$ [8].

$$p(y_i|y_1, y_2, \cdots, y_{i-1}, \mathbf{x}) = \text{RNN}(\mathbf{c_t}) \tag{2.33}$$

The calculation of the context vector $\mathbf{c_t}$ is done by summing all the weighted hidden states

by associated attention weights α_j as Eq. 2.34.

$$\mathbf{c_t} = \sum_{j=1}^{T} \alpha_j h_j \tag{2.34}$$

The attention weights are obtained by a trainable feed-forward neural network ffn and a softmax activation function shown as Eq.2.35 and Eq.2.36.

$$e_j = \text{ffn}(s_{t-1}, h_j) \tag{2.35}$$

$$\alpha_j = \frac{\exp(e_j)}{\sum_{k=1}^{T} \exp(e_k)} \tag{2.36}$$

With this attention design, the model is able to focus on a hidden state at an arbitrary time step without memorizing the entire input sequence.

2.3.2 Deep Learning Based PPI Prediction

During the past decade, a number of studies have been carried out to apply deep learning algorithms to the prediction of PPI. Based on the network structure backbone, these approaches may be roughly classified into two categories: singular networks and Siamese networks. Sun *et al.* 's study is one of the earliest attempts to introduce the deep learning algorithm to PPI prediction [34], which belongs to the first category. In this, a stacked autoencoder (SAE) [157] is employed as the backbone model. The SAE is a stack of autoencoders trained layer by layer. Each autoencoder layer learns to reconstruct the input from the previous layer. The entire stacked autoencoder is finely tuned based on the specific task. In this approach, the protein pairs are first encoded by encoders and then concatenated as a single vector. Two methods for encoding the protein sequence are employed and tested separately: the autocovariance method [158] and the conjoint triad method [21]. The concatenated vector is then used to train the SAE in an unsupervised manner. A softmax activation is attached to the trained model, and the parameters in it are finetuned by backpropagation. A possible drawback of this method is that the SAE training process might cause the loss of important information. Because the pretraining stage of an SAE is based on the goal of reconstruction rather than the specific task, the autoencoder learns

that a representation should carry as much information as possible rather than carry the most relevant information associated with the task. The above discussion is one possible reason that the most powerful algorithm is the convolutional neural network rather than SAE.

A representative study in the Siamese network category was proposed by Li *et al.* in 2018 [38]. The study employs a more modern design of deep learning models, including convolutional layers, pooling layers, and LSTM layers. The general structure of the model follows the Siamese network model: each protein in its pair, it is processed individually by a subnetwork in the Siamese network. For both subnetworks, the parameters are shared and synchronized during training and inference, which reduces the number of parameters by half. The protein is first encoded by the embedding layer. Next, a sequence of convolutional layers and pooling layers is utilized to extract the features. The automatically extracted feature is then fed into an LSTM layer to capture the pseudo-time dependencies. The outputs from the LSTM layers in the two subnetworks for the protein pair are concatenated into one single vector as the input of a fully connected network. The binary classification result is obtained by the network using a sigmoid activation function.

With the sophisticated design of the deep learning model, Li *et al.* achieved state-of-the-art results. However, there is room for improvement in that the interaction between the two proteins in a pair is only computed once during the entire information flow of the neural network. During feature extraction, it is inevitable for some information to be permanently lost, even if it might be crucial for the following fully concatenated network to accomplish the classification result. Therefore, based on this assumption, we carried out our studies.

2.3.3 Deep Learning Based Protein Design

One of our goals in this thesis is to deliver a deep learning algorithm that, given the amino acid sequence of a target protein, is able to generate the sequence of its protein binder. Our background research found two studies with the highest relevance. The first is by Saka *et al.* [133] from 2021. It proposes a workflow for generating de novo antibodies for a given antigen utilizing deep learning. The first step in the workflow is to construct a dataset.

Phage display panning is performed against an antigen, and then the next-generation sequencing technique [159] is employed to generate a large set of antibody codons. The codons are translated into amino acid sequences. A beginning of sequence (BOS) token is added to every sequence in the dataset, then encoded into a vector using one-hot encoding. The result is then used to train an LSTM model to predict the probability of the next amino acid based on the past sequence. The trained model is then used to generate protein sequences based on only BOS tokens. Theoretically, if the sampling strategy is to select the amino acid with the highest probability, this model is only able to generate a single antibody. To solve this problem, a temperature factor–controlled softmax is introduced to the model to increase the diversity. The limitation that prevents us from employing this workflow is its high cost in both time and computation power. Applying this workflow to PPI design requires a unique dataset to be constructed and a new model to be trained from scratch in order to generate the binder of any arbitrary protein, which leaves the solution impractical.

The other relevant study is by Wu *et al.* [132] that utilizes the previously introduced seq2seq algorithm. This design choice is able to address the high-cost issue in Saka *et al.* 's method, since the model learns the mapping from the input sequence to the desired sequence instead. Wu *et al.* 's approach uses the signal peptide (SP) design, which is to generate a short chain of amino acids located at the termini of a given expressed protein. Similar to other methods based on machine learning techniques, a dataset from Swiss-Prot [160] is first collected for training the model. Next, the amino acid sequences are encoded with one-hot encoding. The backbone model for Wu *et al.* 's approach is one of the latest deep learning models, Transformer [161]. This approach achieved promising results and inspired our study into PPI design. However, there is one drawback remaining to be solved, which is that, unlike the unique SP-protein mappings, the interaction between proteins belongs to multivalued problems, which require extra constraints and cannot be solved by a conventional Seq2seq algorithm. Our work addresses this shortcoming.

2.3.4 Discussion

In section 2.3, we explained the most relevant deep learning techniques for PPI prediction and design. Then, we illustrated the most recent deep learning applications in both fields. For deep learning techniques, they are compatible components that are usually combined to form a model rather than competing with one another. For instance, the Siamese network is a higher-level structure that can be used with any deep learning layers including convolutional layers, recurrent layers, attention layers, and fully connected layers. Similarly, a residual neural network refers to a group of deep learning models with skip connections that enable the propagation of gradient to pass by certain layers in the model. The type of layers employed in the model will not conflict with the definition of residual neural network. Therefore, rather than simply comparing them, we should discuss the most suitable application scenario for each type. The convolutional layer and the pooling layer possess the ability to automatically extract features and compress the size of vectors, which suggests that they should be employed at the early stage of a model. The recursive layers, including LSTM layers loop as many times as the length of an input vector. To reduce the computation and the risk of accumulating errors, a recursive layer should be placed at the late stage of a model, with a feature map compressed by the convolutional neural network as input. As for attention layers, according to Cordonnier *et al.* [162], *a self-attention layer can express any convolutional filters given enough heads and using relative positional encoding*. Therefore, in a model containing self-attention layers, such as the transformer model, there are usually neither convolutional layers nor recursive layers, since their functions can be fulfilled by attention layers.

For applications in both PPI prediction and design, we summarize their advantages and limitations in Table 2.5 and Table 2.6.

2.4 Summary

In this chapter, we first introduced background knowledge about protein and protein interaction. Proteins are the foundation of all kinds of life on earth. They provide nutrition, support the structure of cells, and catalyse biochemical processes. Proteins are sequentially

Table 2.5: Comparison of deep learning–based PPI prediction approaches

Study	Methodology	Advantages	Limitations
Sun *et al.*	- Stack autoencoder	- Given enough data, prediction performance of the model can be improved by increasing capacity of pretraining	- Require large-scale datasets - Fully connected network prone to overfitting - Performance less compelling than competitors
Li *et al.*	- Siamese network - Convolutional neural network - LSTM	- Efficiency in computation - Less prone to overfitting	- Parameters of model must be modified according to size of training dataset - Ignores lower-level information

Table 2.6: Comparison of deep learning–based PPI design approaches

Study	Methodology	Advantages	Limitations
Saka *et al.*	- LSTM	- A temperature factor–controlled softmax is introduced to the model to increase the diversity - Latest advanced deep learning model	- The model must be retrained when being applied to every other protein
Wu *et al.*	- self-attention - transformer	- Employ Seq2seq to predict a binder based on the target protein - Free from retraining	- Not applicable for multivalued problems

formed by a specific type of organic compound, amino acids. In homo sapiens, there are 20 standard amino acids. Proteins as chains of amino acids need to fold into three-dimensional shapes to exert their functions. The process of folding and the resulting three-dimensional structure are determined by the primary structure of the protein, which is the sequence of amino acids.

The majority of protein functions are related to the physical contacts between proteins, which are referred to as protein-protein interactions (PPIs). The study of PPIs is essential for the comprehension of cellular biochemistry. To explore PPIs, many in-lab methods have been developed and have obtained high-throughput performance. However, they are still prone to errors. Compared to silico approaches, in-lab approaches are expensive, labour-intense, and time-consuming. Moreover, there exist 50 million possible PPIs for a single species, which is impossible to solve using in-lab experiments. The above reasons motivate the studies of computational PPI approaches.

Next, we introduced the computational approaches. We have two different standards to classify the studies. First, using the methodology, we separate these approaches into two kinds: conventional computational approaches and deep learning-based approaches. Second, we can also select the studies for the two kinds of applications which interest us most: PPI prediction and PPI design. PPI prediction refers to determining whether a pair of proteins is interactive with each other, and PPI design refers to attempting to design a protein which binds to a target given protein.

Since it is common to involve conventional machine learning approaches in conventional computational studies, we explained the most frequently used conventional machine learning techniques: decision tree, random forest, naive Bayes, a support vector machine, and gradient boosting. These conventional approaches tend to share the drawbacks of expensive computation, strong dependency on prior knowledge, dependency on the existing binder library, and the limiting of applications to a specific protein.

Next, we introduced the approaches based on deep learning solutions. Their advantages compared to conventional machine learning approaches are that they require less expert knowledge of the problem and generally have better prediction performance. However, they also have their limitations and flaws. PPI prediction applications usually require a larger

scale of dataset than conventional machine learning–based approaches. Because most of them employ the Siamese network structure to abstract the feature from the protein pairs separately, the detailed information crucial to PPI might be discarded. As for PPI design, it is not applicable to multivalued problems. We would like to utilize the advantages of deep learning–based approaches and also overcome the current drawbacks. With this motivation, we focus our work in this thesis on the topic of applying deep learning to PPI prediction and design.

Chapter 3

Paying Attention: Siamese Pyramid Network Architecture

The work presented in this chapter is based on our published paper: *Paying Attention: Using a Siamese Pyramid Network for the Prediction of Protein-Protein Interactions with Folding and Self-Binding Primary Sequences* [163].

Recall from Chapter 1 that one of our main aims in this thesis is to design deep learning solutions to predict the binding probabilities of interacting protein pairs, by considering self-binding and folding amino acid sequences rather than only relying on the sequences. In this chapter, we introduce our SPNet architecture that was designed and implemented to accomplish this task.

This chapter is organized as follows. Section 3.1 highlights related work in PPI prediction, while Section 3.2 details the SPNet architecture. This is followed in Section 3.3 with an experimental evaluation, while Section 3.4 concludes the chapter.

3.1 Related Work

Recently, significant efforts have been undertaken to design efficient and high-throughput methods for the experimental study of PPIs [18]. Nonetheless, a drawback of current

approaches is their inability to screen millions of cases in a reasonable amount of time and at a realistic cost. As mentioned above, machine learning has the potential to address these two limitations [97,164]. For example, a support vector machine (SVM) approach was proposed by [10] in which proteins with different amino acid lengths were transformed into common matrix representations that allowed neighbouring effects to be considered. This method was further improved by [20], who encoded such sequences with binary descriptors related to the distances between the amino acid pairs. However, SVM methods cannot predict the binding probabilities required to characterize PPIs properly. Ensemble-based methods have also been applied to PPI prediction. To this end, a random forest algorithm based on Minimum Redundancy Maximal Relevance and incremental feature selection was introduced in [92] and was shown to be superior to SVMs. However, when considering large data sets, deep learning networks often outperform their traditional machine learning counterparts [32, 33] due to their complex hierarchical architecture and the expressing abilities associated with their large latent spaces. Deep learning methods also have the capability to extract features automatically without human intervention.

In a key study, Hashemifar developed a deep learning framework based on amino acid sequences [36] that employs a Siamese network, where the sequences are encoded with PSI-BLAST [37], thus allowing for apparently dissimilar, but nonetheless homologous, proteins [165] to be characterized by similar representations. Here, the outputs generated by the two subnetworks are projected into a common subspace with a non-trainable reverse random projection, making the network robust against feature ordering. Competitive results were obtained by adding an embedding layer inspired by word2vec [166] to evaluate the semantic quality of individual sequences [38].

Several recent PPI deep learning architectures have utilized Siamese neural networks [36,38,52]. This choice can be justified as follows: 1) the twin subnetworks share common parameters (weights, biases, and so forth) and hyperparameters; indeed, the same network is applied twice in tandem, thus reducing the number of parameters to be learned while reducing the risk of overfitting; 2) the input features for both subnetworks are the same, although each subnetwork is associated with a distinct member of the protein pair; 3) The architecture is suitable for end-to-end learning, meaning that the system, as a whole, can be trained with gradient-based techniques [167]. In current Siamese networks, the

two subnetworks interact only once and usually via some projection mechanism, implying implicitly that the amino acid sequences are analysed on a single level. As stated above, both local and non-local interactions at the sequence level play a critical role in PPIs [168], and such behaviours are difficult to capture if the subnetworks only interact once. Therefore, we propose a novel architecture in which the subnetworks interact repeatedly at various levels to capture the numerous interaction mechanisms between the amino acids. An interaction probability is calculated at each level. The probabilities from all levels, along with the probability generated by the attention layer, are then combined into a single score via a trainable linear layer. This architecture is illustrated in Fig. 3.1. Our most important contributions are as follows: a self-attention mechanism that addresses protein folding and self-interaction; multilevel interactions between substructures in the Siamese network that capture the various interaction levels; and preventing information leaks with a strict dataset.

3.2 Architecture of SPNet Network

Fig. 3.1. Illustrates the general architecture of our SPNet network. Recall that a Siamese network consists of two identical subnetworks $\{S_1, S_2\}$ with the same parameters. Given a pair of proteins $\{P_i, P_j\}$, each subnetwork takes, as an input, the amino acid sequence associated with one member of the pair. The network consists of various convolutional neural networks (CNN) and long short-term memory (LSTM) layers. The LSTM allows the time series nature of the sequences to be exploited by capturing local (short-term) and non-local (long-term) interactions. The CNN reduces the number of parameters required considerably due to the relatively small size of the kernel involved in the calculation of the convolution (as opposed to a dense layer). This reduction is a highly desirable characteristic when aiming to avoid overfitting. Further, the CNN component allows us to achieve translation invariance, an important property for PPIs, as the position of an amino acid may shift due to mutation and evolution [165]. In addition, convolution neural networks tend to reduce the spectral bandwidth (low-pass filter) associated with their input tensor, which means that, when feeding a LSTM, they provide the latter with smoother (less noisy) time series, which are more suitable for learning.

Our architecture consists of four components: the convolution block, identity block, attention block, and random projection [36]. Pyramid Feature modules are built from these four elements. These blocks will be described in detail in the next two subsections. The Pyramid Feature module is inspired by the Feature Pyramid Network [169], which consists of twin branches, each containing a convolution block followed by two identity blocks, as first proposed in ResNet [141]. The corresponding blocks in the two branches have the same parameters, and the outputs of both branches are combined with a non-trainable random projection. The network consists of n Pyramid Feature modules, where n was set to five via inspection. Each Pyramid Feature module generates a binding probability. These binding probabilities, as well as the one resulting from the attention block, are combined into a single score by a trainable linear layer, a feature that distinguishes the Pyramid Feature module from its Feature Pyramid Network counterpart. This is, as far as the authors are aware, the first time that this novel structure has been employed in a PPI context.

The network input consists of a pair of one-hot encoded tensors that represent the amino acid sequences. These tensors are followed by a convolutional layer, a batch normalization layer, an ReLU activation function, a maximum pooling layer and the five Pyramid Feature modules. The ReLU function was selected because it was less computationally expensive than other activation functions, while yielding, in our case, better results. The network can be fine-tuned for specific applications by varying the size of the kernels (currently at 16 with 32 filters, as determined by inspection) and by adding more Pyramid Feature modules, if necessary. The outcomes of all the Pyramid Feature modules and the Attention Pyramid Feature module are combined with a trainable linear layer, thus generating a unique binding probability. Recall that the networks have two inputs, one for each member of the protein pair $\{P_i, P_j\}$. For each input, the amino acid sequence is represented by means of one-hot encoding, since amino acids are categorical. Each one-hot vector has twenty-four elements corresponding to the twenty-two standard amino acids plus two codes marking an ambiguous amino acid, that is, one that cannot be entirely determined experimentally due to, for instance, the limitations associated with X-ray crystallography techniques. Therefore, no specific features are calculated from the amino acid sequence, leaving the deep neural network completely free to decide which features to extract. Each one-hot tensor is connected to a convolutional layer, followed by a batch normalization layer, an

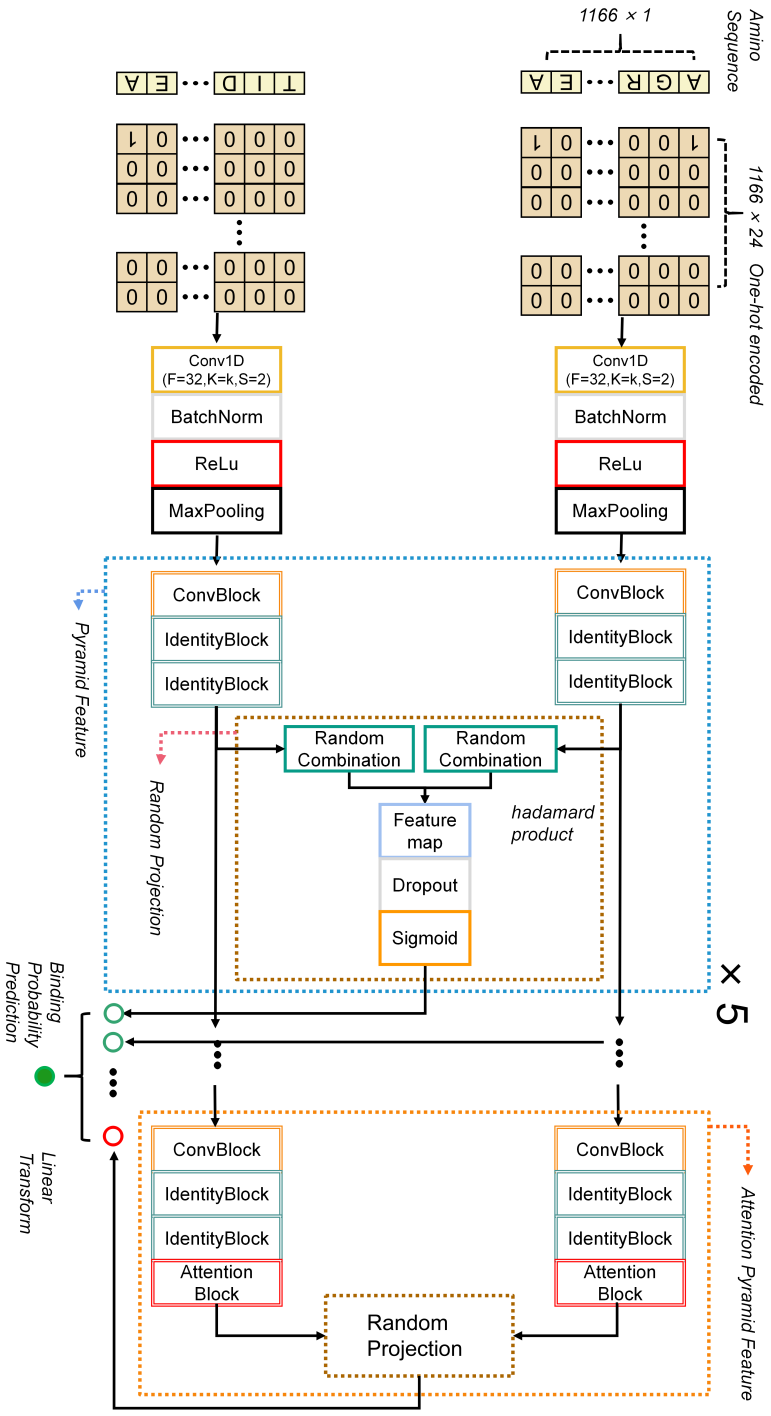


Figure 3.1: Siamese Pyramid Network (SPNet) architecture

ReLU activation function, and a maximum pooling layer. The two branches are then connected to a sequence of five Pyramid Feature blocks, as determined by inspection. Each Pyramid Feature network consists of a Siamese network formed by a convolutional block and two identity blocks. The outputs of the Siamese network are combined with a non-trainable random projection, which is followed by a feature map layer (Hadamard product), a dropout layer and a sigmoid function employed to generate a binding probability. As a result, a binding probability is associated with each one of the five Pyramid Feature networks. These binding probabilities, as well as the one generated by the Attention Pyramid Feature network, are combined by a learnable linear layer to calculate a unique score.

The Attention Pyramid Feature network is also a Siamese network. It is formed by a convolutional block, two identity blocks and an attention block; the outputs are combined with a non-trainable random projection. The convolution and identity blocks are described next.

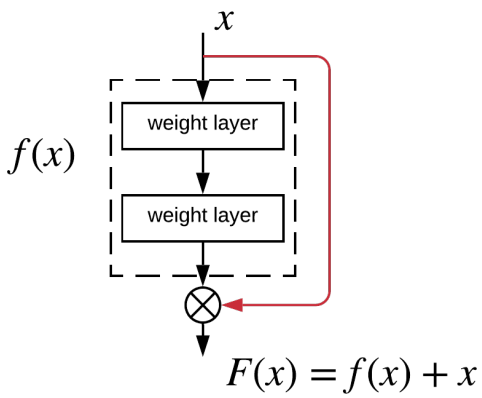


Figure 3.2: Structure of a residual network. The red line indicates the skip connection.

Convolution and Identity Blocks. Residual networks were first proposed by He in 2016 [141]. The structure of a residual block is shown in Fig. 3.2. A residual block is designed to ensure that a model learns the residual rather than the real output [141]. The

loss function of a residual block, such as the one illustrated in Fig. 3.2, is given by

$$L \hat{=} T(x) - F(x) = T(x) - f(x) - x \tag{3.1}$$

where x is the input, $f(x)$ is the output, $T(x)$ is the true output and $F(x)$ is the difference between the output and the input. The residual is defined as

$$R(x) \hat{=} T(x) - x \tag{3.2}$$

It follows immediately that

$$L = R(x) - f(x) \tag{3.3}$$

which shows that the loss associated with a residual network is the difference between the residual and the output. Residual blocks alleviate the problems associated with either the disappearance or the explosion of the gradient during training and improve the convergence rate [141]. The reason for this behaviour is rooted in the concept of a skip connection. *Skip connections are extra connections between nodes in different layers of a neural network that skip one of more layers of nonlinear processing* [170]. Without skip connections, while propagating, the error can accumulate or dissipate, thus leading to a gradient explosion or disappearance, respectively. Nevertheless, by skipping a few nonlinear layers, these problems can be avoided while improving the convergence rate [141].

Our convolutional and identity blocks are shown in Fig. 3.3. The convolutional block is a residual network consisting of three batch normalization layers, three convolutional layers and three ReLu activation function layers for the hierarchical decomposition of the input. The skip connection consists of a batch normalization layer and a convolutional layer. The identity block is a residual block formed by three batch normalization layers, three convolutional layers and three ReLu activation function layers. The skip connection in this case is the identity function.

Attention Mechanism. Our attention block is shown in Fig. 3.3. The concept of an attention mechanism was first proposed by Vinyals in 2015 [171]. Since then, it has been used extensively in numerous fields. Let us consider an amino acid sequence. First, recall that the amino acids in the sequence are not only in contact with their immediate

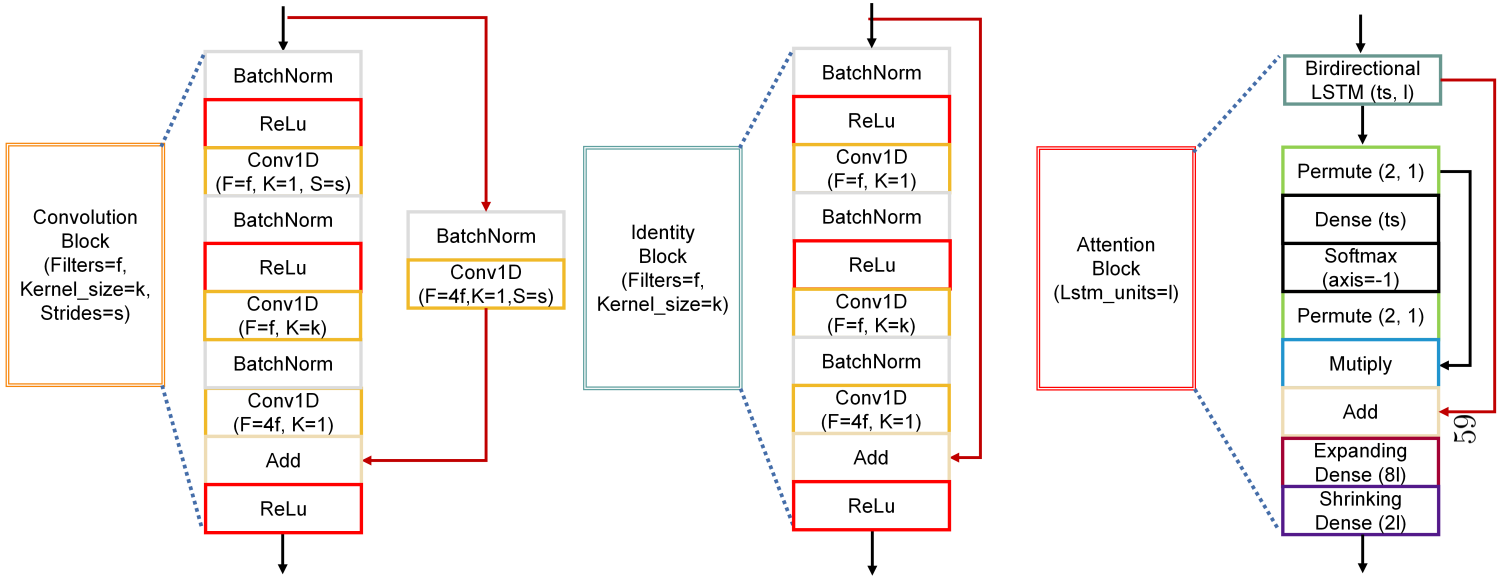


Figure 3.3: Convolutional, identity and attention blocks. The red lines indicate the skip connections.

neighbours but also with more distant amino acids due to folding [168]. Folding is of paramount importance, as it determines the protein's shape (also known as its conformation or molecular surface) ,which, in turn, determines its interactions with other proteins. As stated above, this important fact has been overlooked in previous works on PPIs, making it a novel aspect of our contribution, which is then implemented as follows. In our network, the folding process is represented abstractly by a self-attention mechanism that allows the various amino acids to interact and fold. In our implementation, we employ a self-attention mechanism consisting of a bidirectional LSTM followed by a transpose layer, a dense layer, a softmax layer and another transpose layer. Amino acid sequences are highly suitable for LSTM. Indeed, they have a beginning that is determined by an amino terminus [168]. Therefore, they can be assimilated into time series, where discrete pseudo-times mark the positions of specific amino acids within the sequence. As a result, local and distant interactions can be captured by the short- and long-term memories associated with the LSTM. The transpose layer that follows permutes the order of the dimensions in the output of the LSTM layer. To be more specific, it switches the time-step dimension and the feature dimension in order for the attached dense layer and softmax layer to generate the probability distribution on the last dimension of the permuted output, i.e., the time-step dimension. Then the order of this probability distribution is recovered and an element-wise multiplication is conducted between the time dimension probability distribution and the original LSTM output. The network is completed by a dense expansion layer (by a factor α), a dense shrinking layer (by a factor β) and a global maximum pooling layer. The factors were determined by inspection and set to eight and four, respectively. This last substructure is called a BOOM layer and was first proposed by Merity [172] to create a bottleneck that regularizes the latent space, thus mitigating overfitting and improving accuracy [172]. The proposed attention block is also a residual network since it possesses an identity skip connection [141]. The random projection mechanism is described in the next section.

Random Projection. The role of the random projection is to combine the information flowing from each branch of the Siamese network; this concept was first introduced by Hashemifar [36]. More specifically, a random projection consists of two constant and non-trainable random matrices, $\mathbf{w_1}$ and $\mathbf{w_2}$, that are generated by a random Gaussian

distribution. $\mathbf{i_1}$ and $\mathbf{i_2}$ represent the feature maps associated with each branch of the Siamese network. The output of each branch of the Siamese network is projected as follows:

$$\begin{aligned} p_1 &= \mathbf{W}_1 i_1 \\ p_2 &= \mathbf{W}_2 i_2, \end{aligned} \tag{3.4}$$

where the projection matrices are defined as

$$\begin{aligned} \mathbf{W}_1 &= [\mathbf{w}_1 | \mathbf{w}_2] \\ \mathbf{W}_2 &= [\mathbf{w}_2 | \mathbf{w}_1] \end{aligned} \tag{3.5}$$

This definition ensures invariance against input ordering [36]. The combination of the two outputs is achieved with a Hadamard or element-wise product:

$$O = p_1 \odot p_2 \tag{3.6}$$

Our approach differs from that in [36] in several ways. In [36], the random projection is followed by an ReLU activation function. But, as stressed by [173], in the framework for MobileNet, when the ReLU collapses the channel, it inevitably loses information in that channel; for that reason, the activation function was discarded. In addition, in [36], the projection was performed only once, but, in our case, the projection is performed at every stage: five times for the five Pyramid Feature networks and once for the Attention Pyramid Feature network. Next, we describe our experimental evaluation of the SPNet architecture with a strict dataset and detail our results with regards to the binding probabilities of the proteins tested against the 2019-nCov spike.

3.3 Experimental Evaluation

Recall that overfitting may occur when amino acid sequences in a training set are also present in the test set. A strict dataset avoids such information leaks by ensuring that no proteins are present both in the training and test sets. In our work, we used the strict

dataset created by Richoux *et al.* [52]. All the amino acid sequences in this dataset were retrieved from the UniProt repository on 18 June 2019, and all proteins in the dataset are from the homo sapiens (human) species [52]. The dataset consists of 16,210 unique proteins with a maximum length of 1166 amino acids composing 104,262 pairs in total. Of these pairs, 91,036 pairs constitute the training set, and 12,506 pairs form the validation set. Inside these two sets, 33,318 pairs and 6,094 pairs, respectively, are binding proteins, i.e., proteins with the ability to form either transient or long-lived complexes. Two test sets are used. Test-460 is a balanced strict test set with 230 true positive and 230 true negative instances. Test-720 is an extension of Test-460 and has 260 true positive and 460 true negative instances.

Each amino acid is represented by a 24-bin one-hot indicator. The first 22 bins correspond to the 20 standard amino acids and the two stop codons, while the last two bins indicate unknown or ambiguous amino acids. Therefore, the dimension of the one-hot tensor corresponding to a protein is 1166 x 24. If a protein has less than 1166 amino acids, the remaining one-hot indicator is set to zero.

As far as the authors are aware, Richoux et al [52]. conducted the only study on PPIs utilizing a strict dataset. For this reason, we compare our approach to their work. Two networks were considered in their research, namely, a fully-connected dense Siamese network (FC) and a recurrent network (LCDS). The LCDS architecture was formed by several convolutional blocks, followed by an LSTM and a fully connected layer. Each convolutional block consisted of a convolutional layer, a maximum pooling layer and a batch normalization layer. The FC corresponds to *FC2_20_2Dense* in Richoux's original paper, while the LCDS corresponds to *LSTM32_3Conv3_2Dense_S*. More details about this implementation can be found in [52]. All networks were optimized with the ADAM algorithm with an initial learning rate of 0.001. A reducing factor of 0.9 was applied to the learning rate each time the validation loss remained unchanged across five epochs until it reached a minimum of 0.0008. The learning hyperparameters are reported in Table 3.1 ,and our experimental results are presented in the next section.

An evaluation approach similar to that used in [52] was followed, where the dataset was divided into training, validation, and test sets. Initially, the network was trained and validated at the end of each epoch until convergence. The epoch with the lowest validation

loss was recorded. A new training set was subsequently generated by combining the original training set and the validation set. A network was trained once more against this combined set until it reached the original lowest validation loss epoch. We subsequently report the results for the two test sets Test-460 and Test-720.

3.3.1 Experimental Results

Table 3.1: Learning hyperparameters.

Model	LCDS / FC	SPNet
Batch Size	2048	256
Optimizer	Adam	Adam
Loss	Binary Cross-entropy	Binary Cross-entropy
Learning Rate	0.001	0.001
Learning Rate Decay	True	False
Reducing Factor	0.9	-
Patience	5	-
Minimum Learning Rate	0.0008	-

Table 3.2: Accuracy, F1-score and AUC.

	Test-460			Test-720		
Model	SPNet	LCDS	FC	SPNet	LCDS	FC
Accuracy	**78.26%**	70.00%	72.17%	**79.58%**	75.14%	76.81%
F1-score	**0.7685**	0.6368	0.6484	**0.7168**	0.6049	0.6178
AUC	**0.8385**	0.7615	0.8261	**0.8370**	0.7656	0.8076

Our SPNet network was compared to FC and LCDS using the strict dataset described above. Four well-known metrics were employed for the comparison, namely, accuracy, the F1-score (the harmonic mean of precision and recall), the receiver operating characteristic curve (ROC) and the area under the ROC curve (AUC) [174]. Our results for the first

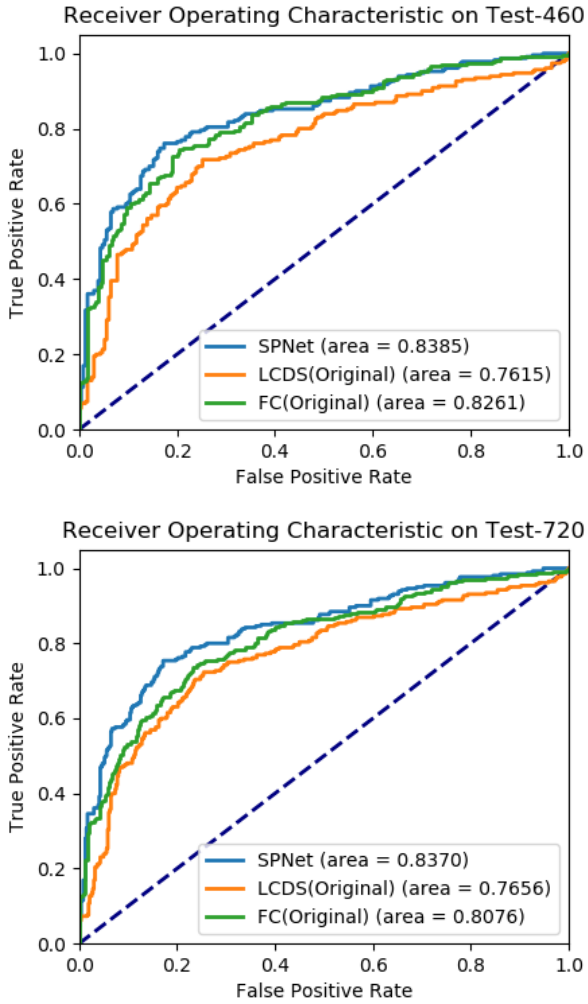


Figure 3.4: ROC curves for SPNet, LCDS and FC for Test-460 (Upper) and Test-720 (Lower).

three metrics are reported in Table 3.2, whereas Fig. 3.4 displays the ROC curves. Our SPNet network outperforms both FC and LCDS for all four metrics and for both strict test sets. For instance, using the balanced test set (Test-460), the F1-score for our SPNet is 0.77, while the scores for the LCDS and FC of 0.64 and 0.65, respectively. The results are even more striking for the imbalanced dataset (Test-720), with our SPNet achieving an

Table 3.3: Five proteins with the highest binding probabilities for the 2019-nCov spike.

UniProt Code	Binding Probability
P60410	0.9589
P60409	0.9588
Q969F0	0.9587
Q7Z3S9	0.9586
Q99750	0.9577

F1-score of 0.72, while the scores for the LCDS and FC were 0.60 and 0.62, respectively. With the balanced test set, our SPNet achieves an accuracy of 78.2%, as opposed to 70.0% for LCDS and 72.17% for FC.

We next turn our attention to the Covid-19 pandemic. A recent study by Wrapp *et al.* [1] has revealed the importance of the 2019-nCov spike in the prefusion conformation as a key target for vaccines, therapeutic antibodies, and diagnostics. The sequence of the spike was extracted, and the binding probabilities between the 2019-nCov spike and all the proteins in our dataset were calculated to assess which proteins would be the most likely to bind with this spike. Our intention was to find potential candidates for prophylactic (preventative) and therapeutic vaccines. Some of our results are reported in Table 3.3, and the best 500 results are available upon request, while information about the proteins on the list can be found at www.uniprot.org. For instance, the reader may notice that the P60409 and P60410 proteins are located on the roots of human hair shafts. Q969F0, on the other hand, is expressed predominantly in testes and is highly expressed in certain types of cancer tissues, such as hepatocellular carcinoma and colon and gastric cancer. We trust that these results will aid experimentalists in exploring promising avenues, saving precious time in our race towards a vaccine, while strengthening our goal of protecting vulnerable individuals.

3.4 Summary

The chapter begins in Section 3.1 with a discussion of related work, focusing on the strengths and limitations of traditional machine learning techniques and deep learning approaches. Next, in Section 3.2, we introduce the SPNet architecture for learning to predict interactions between protein pairs. The novelty of our approach lies in the fact that we focus the learning on the self-binding and folding characteristics of proteins, through the implementation of attention mechanisms. Our experimental results detailed in Section 3.3 illustrate the value of our technique, when compared to the state of the art, against a strict dataset. We also discuss some preliminary findings when considering the 19-nCov spike.

In the next chapter, we turn our attention to data generation and class imbalance. In addition, we introduce our deep learnable architecture for adaptive learning from a variety of dataset sizes.

Chapter 4

Adapting to Complexity: Deep Learnable Architecture and Data Generation

The work presented in this chapter is based on our published paper *Adapting to Complexity: Data Generation and Deep Learnable Architecture for Protein-Protein Interaction Predictions* [175].

In Chapter 3, we introduced the SDPPI architecture to predict protein interaction when considering strict data sets. In this chapter, we focus our attention on data realities when considering protein data sets. First, it should be noted that deep learning algorithms often require large data sets that are not always readily available when studying PPI. Second, in this domain, information about non-interacting proteins is scarce. We introduce a deep learnable architecture that can adapt the data set sizes and properties to address these issues. In addition, we introduce a graph-based algorithm to extract non-interacting proteins, resulting in a balanced data set of more than seventeen million pairs.

This chapter is organized as follows. Section 4.1 discusses related work, focusing on the impact of data realities such as scarcity of non-interacting pairs and the risks of overfitting on current solutions. In Section 4.2, we detail our graph-based algorithm for generating non-interacting pairs. Section 4.3 provides a detailed discussion of our deep learnable

architecture that dynamically adapts between deep and shallow learning. Section 4.4 contains our extensive experimental evaluation, while Section 4.5 concludes the paper.

4.1 Related Work

In the past few years, deep learning has mostly superseded traditional machine learning approaches both in computer vision and natural language processing (NLP) [30, 31]. This is particularly true when a large amount of data is available; in which case deep learning outperforms traditional machine learning techniques [32, 33]; including PPI. For example, Sun *et al.* employed a stacked autoencoder (SAE) for protein classification [34]. Proteins were encoded either with the autocovariance method, as proposed earlier by Guo *et al.* , or with the triad method, as introduced by Shen *et al.* ; the former performed better than the latter. Guo's approach outperformed other approaches [21, 23, 35] on Pan's benchmark dataset [35]. Improved performance was achieved by Li *et al.* [38] by employing Siamese neural networks in which the twin arms shared common parameters: each arm corresponded to a particular member of a pair. An embedding layer was used to map the sparse protein amino acid sequence into a dense structure. This was followed by multiple convolutional layers intended to perform a hierarchical analysis. Trans-hierarchical features were extracted using a long short-term memory (LSTM) network. A dense layer with a sigmoid activation function provided a binary classification. This approach may be considered the state of the art, as it outperforms not only on Pan's dataset [34], but also on the Human Protein Reference Database (HRPD) [103], the Database of Interacting Proteins (DIP) [54], the Human Integrated Protein-Protein Interaction Reference (HIPPIE) [55], and InWeb_InBioMap (INWEB) [56]. Siamese neural networks were also employed by Hashemifar *et al.* [36] but, instead of using raw amino acid sequences, sequences were rather encoded with the position-specific iterative basic local alignment search tool (PSI-BLAST) [37]. Hashemifar's *et al.* approach outperformed several SVM-based methods, including [10, 24, 25].

The strength of Siamese neural networks is that they involve a limited number of parameters, in contrast to the more complex architectures used in image classification and

NLP applications [142, 176–178]. In addition, as shown by [179], small datasets strongly constrain model complexity. As an example, and to introduce the idea, a dense network with two layers involving $2n + d$ parameters has the ability to fit a dataset consisting of n instances of size d. If this rule is applied to Guo's dataset, the data may be modeled with only 122,542 parameters. Beyond this number, there is a serious risk of overfitting. Consequently, most networks proposed in the literature are not suitable for large datasets, as overfitting may seriously impair their predictive capability [180].

Most deep learning techniques employed in PPI originate from other fields: convolutional neural networks were introduced in computer vision, whereas LSTM neural networks were initially intended for time series. In contrast to the datasets used in these fields, which are large, PPI datasets are usually quite small. For instance, the ImageNet dataset [181] consists of 1,281,167 images, whereas Guo's dataset, which is the most commonly employed dataset in PPI, consists of only 60,671 protein pairs. A small dataset imposes strong limitations on deep learning algorithms, as there is a serious risk of overfitting when a complex model is employed. As a result, models were intentionally kept simple and the number of parameters was kept small to prevent overfitting. Depending on the dataset and the application, the number of protein pairs may vary from small to large and, irrespectively of the dataset size, the amino acidic complexity may range from low to high. This issue must be addressed by selecting the right architecture and depth for the neural network. This is usually done by inspection, which is a time-consuming and ad hoc approach. In addition, most protein interaction datasets, especially large ones, lack negative examples, i.e., non-interacting proteins. This is an important issue, as the algorithm must not only learn if there is an interaction, but also be able to detect the nonexistence of such an interaction. Indeed, for therapeutic or prophylactic vaccines, an antibody that interacts with an antigen may constitute a good candidate for a vaccine whereas a non-interacting antibody is not.

4.2 Graph-based algorithm for generating negative examples

The STRING database is a large-scale repository used for studying protein interactions. Unfortunately, one drawback of the STRING database is that it only consists of positive examples, meaning that non-interacting proteins are absent. Negative examples are available from the Negatome database [182] but there are not enough: they number just a few thousand. As such, this is an example of a highly skewed or so-called class-imbalanced setting. Class imbalance has been the subject of numerous studies in the machine learning community. In essence, researchers treat such cases by employing either data-driven, algorithmic, or hybrid approaches [183]. Typically, data-driven approaches modify the proportion of classes by employing sampling techniques such as undersampling and/or oversampling [182, 183]. Algorithmic approaches, however, adapt existing algorithms to pay more attention to the minority class, i.e., in our case, negative interactions or non-binding proteins. Hybrid methods proceed by combining both data-driven and algorithmic methods.

A drawback of oversampling methods is that they typically introduce synthetic samples. In our domain, such samples would not correspond to actual non-binding pairs and would not be suitable for use. Oversampling using negative data derived by randomly selecting proteins from different cellular locations may lead to biased results [182]. Undersampling, on the other hand, may lead to important interacting pairs being discarded. Algorithmic approaches, such as cost-sensitive algorithms, focus on difficult-to-learn examples, i.e., regions in the learning space with difficult-to-learn interactions. This may potentially lead to overfitting. One-class learners are another example of algorithmic learner, which concentrates only on a single set of objects. Focusing only on a subset of protein pairs may lead to important knowledge being left undiscovered [183].

Therefore, we propose generating negative examples directly from the STRING database. To demonstrate the scalability of our approach, 5,879,727 PPIs involving 19,356 Homo sapiens proteins were extracted from the STRING database, as follows. For each interacting protein pair, the STRING database provides a score, between 0 and 1,000, for the binding

strength between the two proteins. This score is first converted into a distance using a Cauchy M-estimator [184]:

$$s \to d(s) = \alpha \frac{1}{1 + \left(\frac{s}{\alpha^2}\right)}, \quad \alpha = 1 \tag{4.1}$$

This distance is equal to 1 if the score is 0, and 0.000999001 if the score is equal to 1000; the stronger the interaction, the shorter the distance. An interaction graph is then created in which each vertex represents a particular protein, and edges account for the interaction between proteins. Given an interaction, the length of an edge is equal to the distance associated with the binding score. Such a graph is called a weighted graph [185]. Edges are attributed only to protein pairs for which there is a known interaction, otherwise, none is attributed. We postulate that the non-interacting proteins correspond to the **most distant** proteins in terms of their geodesic distance (graph distance). Recall that, in our formulation, interacting proteins are close to each other in the graph. The geodesic distance is defined as the length of the shortest path between two proteins, for a path entirely embedded in the graph [186]. Thus, the greater the distance, the more distant the proteins are in terms of their ability to bind. In our ProGraph algorithm, this distance is evaluated using the Floyd–Warshall algorithm [120] for every protein pair, resulting in 187,278,981 geodesic distances. The pseudo-code for our ProGraph algorithm is reported in Algorithm 1.

The 5,879,727 protein pairs with the largest geodesic distance were selected to generate a dataset of 5,879,727 non-interacting protein pairs, which were then merged with STRING to generate a balanced dataset of almost twelve million pairs, namely the B-STRING Dataset.

It is impossible to visualize the entire graph because of the large number of proteins involved. Therefore, as an illustration, we create a graph from the first 30 proteins. This graph is shown in Fig.4.1. Because there are 30 proteins, there are 30 vertices in the graph. The interactions are represented by edges; the stronger the interaction, the shorter the edge. For instance, the interaction between proteins 14 and 15 is much stronger than between proteins 3 and 29. Proteins 1 and 2 do not interact with the other ones, as they are not linked (they have no edge) to any particular protein.

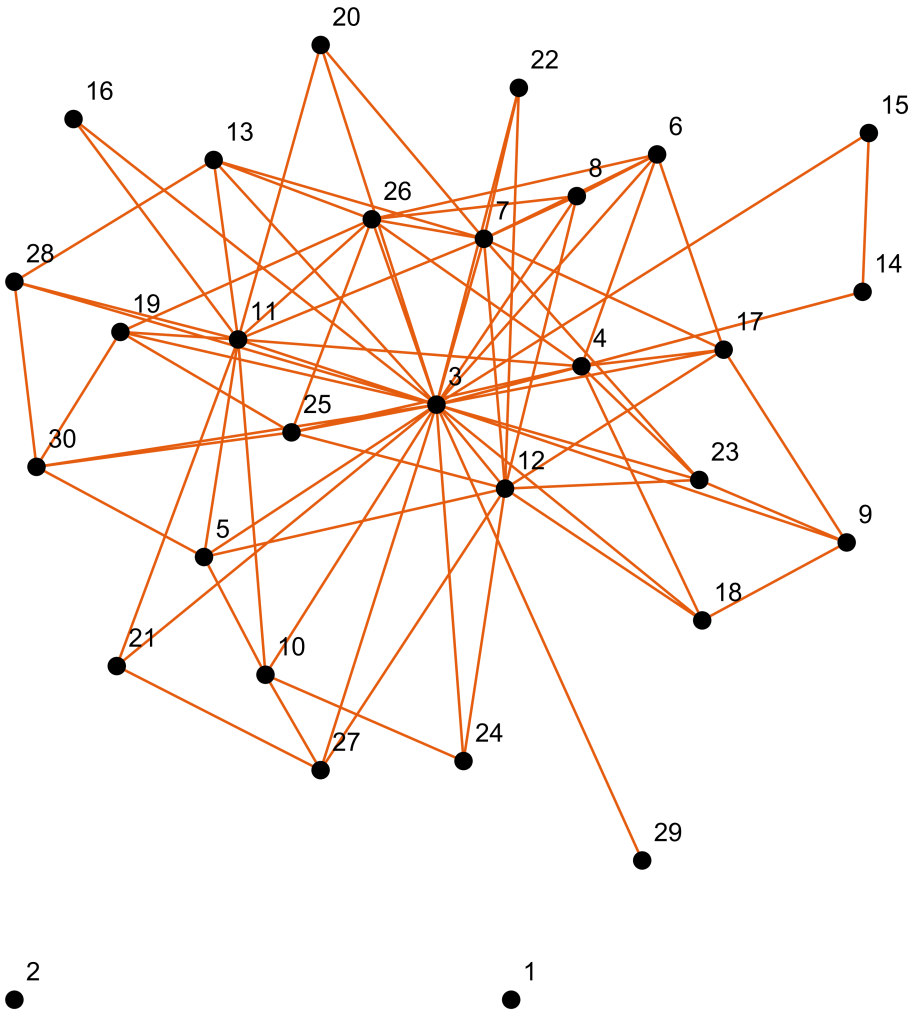


Figure 4.1: Interaction graph for the first thirty proteins of STRING Homo sapiens.

Algorithm 1: The ProGraph algorithm: D being the geodesic distance matrix, v_i a vertex, e_{ij} an edge, and $|V|$ the number of vertices.

Initialization: $D = [d_{ij}] \in \mathbb{R}_+^{|v||v|}$ $d_{ij} \equiv d(v_i, v_j)$ $e_{ij} = \overline{v_i v_j} \in E$ **for** $e_{ij} \in E$ **do**
  $d_{ij} \leftarrow \|e_{ij}\|$
end
for $v_i \in V$ **do**
  $d_{ii} \leftarrow 0$
end
for $k \leftarrow 1$ **to** $|V|$ **do**
  for $i \leftarrow 1$ **to** $|V|$ **do**
    for $j \leftarrow 1$ **to** $|V|$ **do**
      $d_{ij} > d_{ik} + d_{kj} \Rightarrow d_{ij} \leftarrow d_{ik} + d_{kj}$
    end
  end
end

As stated earlier, negative examples correspond to the most distant protein pairs, in terms of their geodesic distance, on the graph. For instance, the geodesic path between proteins 16 and 29 is reported in Fig.4.2, the geodesic path being displayed as a red line.

The geodesic distances between all protein pairs are reported in Fig.4.3. As self-interactions are irrelevant for the purpose of generating negative examples, the latter are ignored and appear as white squares in the distance matrix. This matrix is symmetrical: indeed, if protein A interacts with protein B, protein B necessarily interacts with protein A. Non-interacting pairs are assigned a beige square. All the interactions involving either protein 1 or 2 are beige because these proteins do not interact at all. The geodesic distances are encoded in purple: the larger the distance, the darker the color. The most distant protein pairs correspond to non-interacting proteins. This is done to increase the non-interaction likeliness, as not all interactions are known.

Naturally, the complexity of Fig.4.1, 4.2 and 4.3 are considerably higher when all 5,879,727 interactions are considered. Extremely large protein datasets are of paramount importance for protein design. Indeed, after a large PPI dataset has been trained and

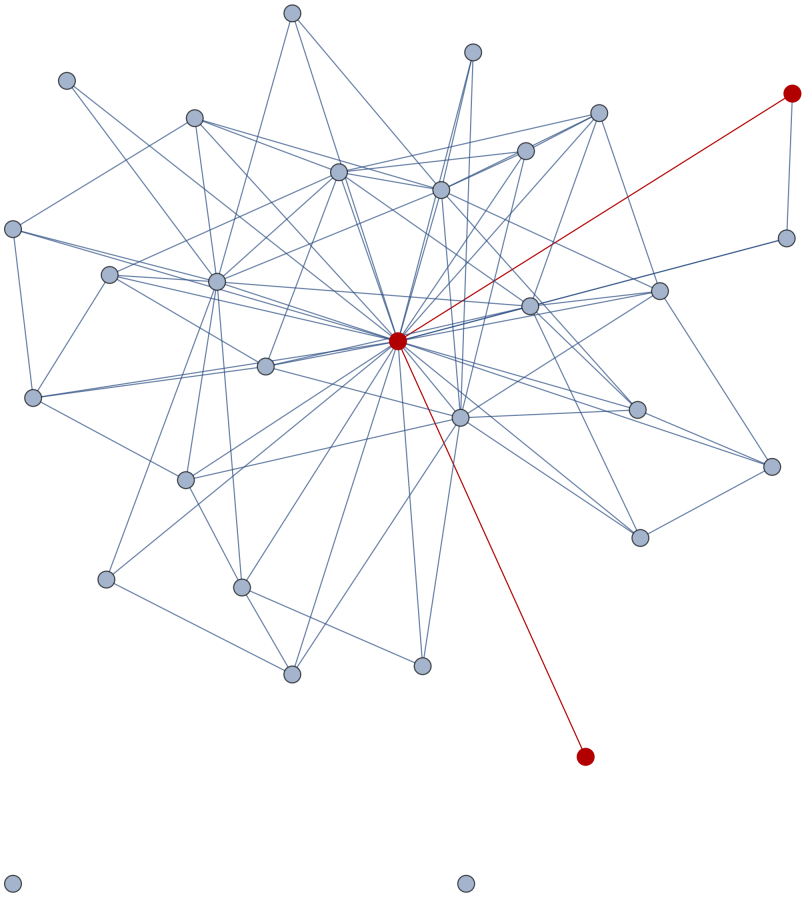


Figure 4.2: Geodesic path between proteins 16 and 29.

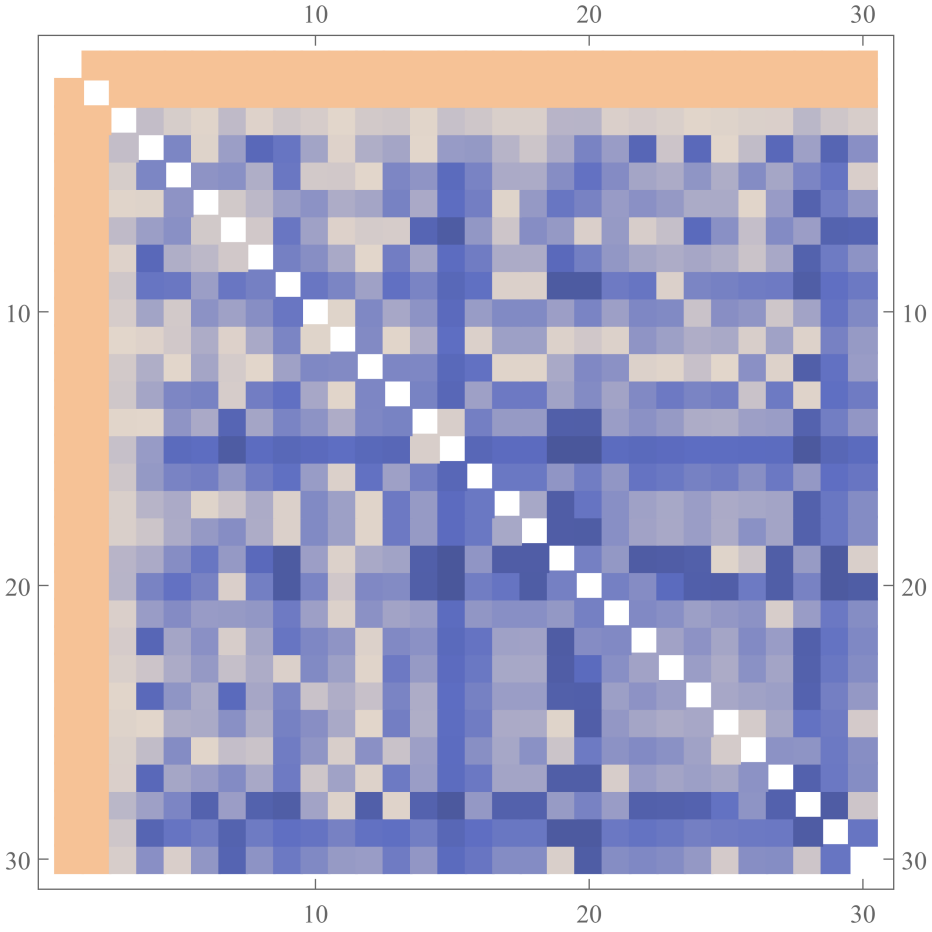


Figure 4.3: Geodesic distance between protein pairs.

given a receptor, our network has the ability to screen millions of potential ligands in a matter of seconds and to attribute to them a binding probability. The ligands with the highest binding probabilities are the most likely candidates for the therapeutic. Next, we discuss the architecture of our neural network.

4.3 Shallow-deep protein-protein interaction network architecture

4.3.1 General architecture

The architecture of our SDPPI neural network is reported in Fig.4.4. Our network belongs to the Siamese network category, as both the top and the bottom branch share the same parameters and the same architecture. However, it differs from traditional Siamese networks in a number of ways.

That is, as opposed to Siamese networks, both branches interact periodically. Such a network is called a pyramidal network. These couplings allow the network to analyze the incoming information at various complexity levels. For each complexity level i, the feature maps from both the top L_i (ligand) and the bottom R_i (receptor) branches are combined or concatenated into a single tensor V_i, as described by Eq.4.2.

$$V_i = L_i \oplus R_i, \quad i = 0, 1, \ldots, n \tag{4.2}$$

Given a complexity level i, the subnetwork associated with this particular level may be assimilated to a nonlinear function f_i, which acts recursively on the previous levels $L_i = f_i(L_{i-1})$. As a result, the resulting concatenated tensor at complexity level j is given by Eq.4.3.

$$V_j = \bigoplus_{i=0}^{j-1} V_i = \bigoplus_{i=0}^{j-1} [L_i \oplus R_i], \quad j = 1, 2, \ldots, n \tag{4.3}$$

Therefore, the feature map tensors are stacked together from shallow to deep levels. *Shallow neural networks process features directly, whereas deep networks extract features*

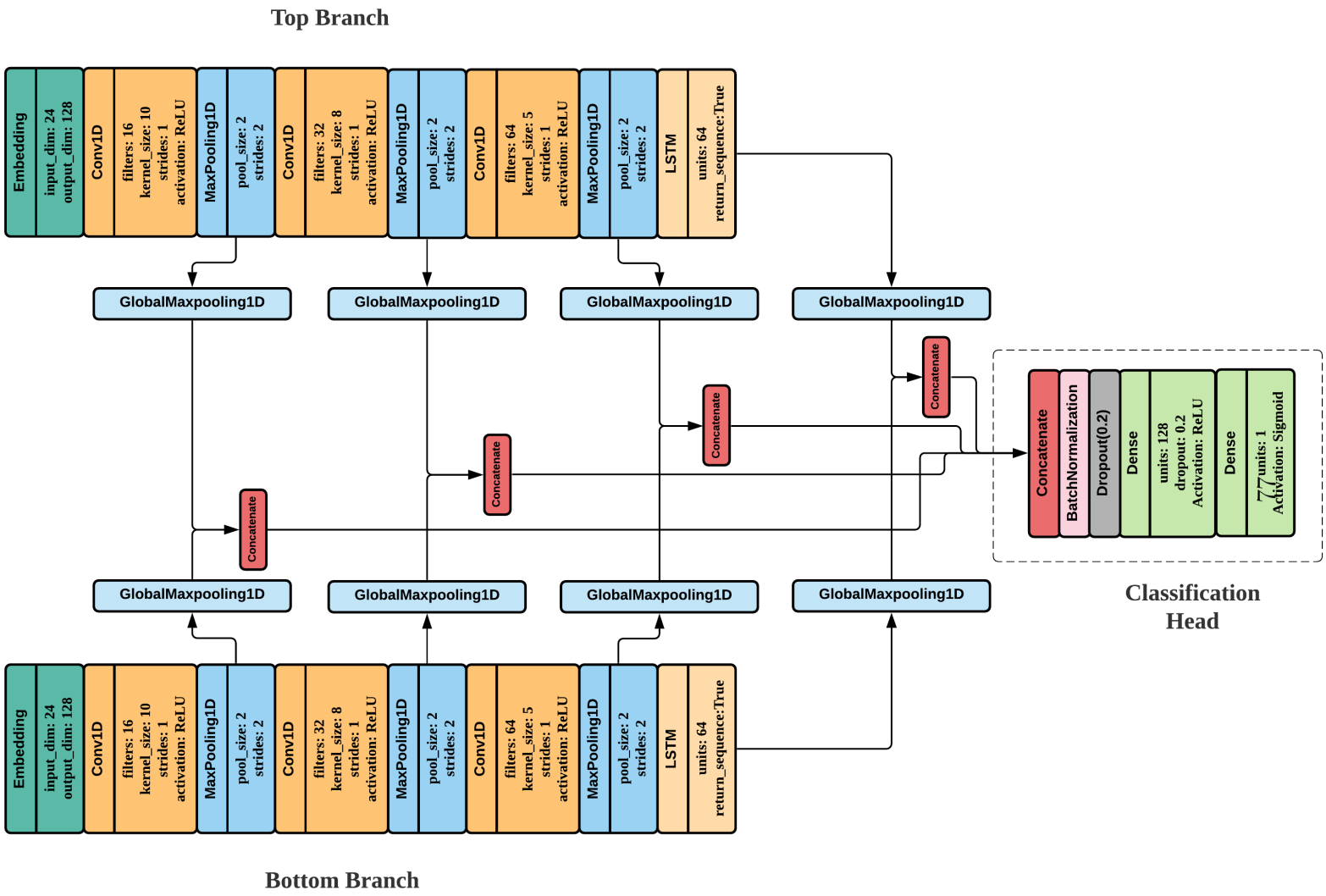


Figure 4.4: Architecture of our shallow deep protein-protein interaction (SDPPI) neural network.

automatically during the training [180]. The resulting concatenated feature maps are processed by the classification head, which consists of a batch normalization layer, a dropout layer, and dense layers. The normalization as well as the randomness introduced by the dropout layer serve as a regularization mechanism that aims to minimize overfitting. These layers do not affect the order in which the various complexity levels are concatenated.

The first dense layer, which is called the selecting dense layer, contains a layer of neurons that is activated using a ReLU function to obtain positive weights for the weighting of the various layers. If a shallow network is learned, the weights corresponding to the deeper layers will be small, whereas if a deeper network is learned, most weights will be non-negligible. The last layer consists of a single neuron having a sigmoid activation function that predicts the binding probability. The sigmoid function is employed to generate a probability between zero and one. The latter is converted to a binary classification: the proteins are binding if the probability is greater than 0.5, and non-binding otherwise. The selecting dense layer adaptively attributes weights to the concatenated layers according to Eq.4.4.

$$Z_j = \mathrm{ReLU}\left(W_j \cdot V_j + b_j\right) \tag{4.4}$$

In other words, the hierarchical and periodical interaction between the two branches allows the selecting dense layer to discard certain layers, if deemed appropriate by the loss function, thus resulting in a shallower network. Furthermore, the selecting dense layer also constrains the gradient of the loss function. Indeed, when the weights associated with a particular complexity level increase as a result of the action of the selecting dense layer, their contribution to the back propagation gradient increases as well. Therefore, the selecting dense layer attributes the most weight to the most accurate levels. For instance, if the deepest level (V_4)has the highest accuracy, the weights attributed to the shallower layers are reduced by the selecting dense layer accordingly, along with their gradients (remember that the deepest level includes all the previous results, as specified by Eq.4.3).

In general, if the number of instances is small, a shallower network should be chosen by the selecting dense layer. Indeed, a shallower network performs better than a deep network because the latter has a large number of parameters, which makes it prone to

overfitting [180]. Comparatively, if the number of instances is large, a deeper network is selected, which is more suitable for complex datasets due to its large latent space. As pointed out by [179], a deep learning model with p parameters trained with instances of dimension d is quite likely to be subject to overfitting if the number of instances is greater than $(p-d)/2$. Therefore, the complexity and the depth of the model is strongly constrained by the size of the underlying dataset. This is why the model must adapt its architecture accordingly.

Therefore, our SDPPI employs a shallower network when the dataset is too small to train a deeper network properly. The selecting dense layer assigns greater importance to the shallower networks by increasing their weights while reducing the weights associated with the deeper networks. As a result, most of the gradient originates from the shallower networks. On the other hand, a deeper network is employed by the selecting dense layer if the number of instances is large. The selection is performed by attributing more weight to the deeper networks and less to the shallower networks, as well as their respective gradients. Therefore, SDPPI may simulate networks of various depth and determines the network architectures that are the most suitable for a given dataset irrespective of its size and without recourse to arbitrary rules or human intervention. According to the complexity of the dataset, a superposition of shallow and deep networks may be generated.

The action of selecting the dense layer may be explained mathematically as follows. Let us consider an SDPPI network with interaction levels. The input of the selecting dense layer is given by Eq.4.3. Therefore, its input is the direct sum of each interaction level. The direct sum may be expressed as Eq.4.5.

$$V = [V_0, V_1, \dots V_n] \tag{4.5}$$

It follows that the weight matrix, associated with the dense selecting layer, may also be partitioned into submatrices; one for each level:

$$W = [W_0, W_1, \dots, W_n] \tag{4.6}$$

As a result, the output of the selecting dense layer becomes:

$$O = \eta\left(V \cdot W\right) = \mathrm{ReLU}\left(\sum_{i=o}^{n} V_i W_i + b_i\right) \tag{4.7}$$

It follows that the depth of an SDPPI network is determined by the weight of its respective submatrices$\{W_i\}_{i=0}^{n}$: all their elements become extremely small when certain levels are not required by the network. When backpropagation is performed, each submatrix W_i is updated according to Eq.4.8:

$$\begin{aligned} W_i &\leftarrow W_i - \alpha \frac{\partial \mathcal{L}}{\partial O} \frac{\partial O}{\partial \left(V \cdot W\right)} \frac{\partial \left(V \cdot W\right)}{\partial W_i} \\ &= W_i - \alpha \frac{\partial \mathcal{L}}{\partial O} \frac{\partial O}{\partial \left(V \cdot W\right)} V_i \end{aligned} \tag{4.8}$$

where α is the learning rate and $\mathcal{L}$is the cost or loss function. Therefore, it is apparent from Eq.4.8 , that the weights at a given level are determined by the input of the selecting dense layer at the very same level V_i. The weights are updated in accordance with their effect on the cost function: they are increased if they reduce the cost function and decreased otherwise. This, in turn, determines which levels and depth are chosen by the selecting dense layer, thereby allowing the best architecture for the training dataset to be learned. As a result, the gradient assigned to each level becomes:

$$\delta V_i = \frac{\partial \mathcal{L}}{\partial O} \frac{\partial O}{\partial \left(V \cdot W\right)} \frac{\partial \left(V \cdot W\right)}{\partial V_i} \tag{4.9}$$

where the last member of Eq.4.9 is given by Eq.4.10.

$$\frac{\partial \left(V \cdot W\right)}{\partial V_i} = W_i + \frac{\partial V_{i+1}}{\partial V_i} W_{i+1} + \ldots + \frac{\partial V_n}{\partial V_i} W_n \tag{4.10}$$

Let us define:

$$\begin{aligned} L_i &\triangleq L\left(V_i\right) \\ R_i &\triangleq R\left(V_i\right) \end{aligned} \tag{4.11}$$

and the recursive function:

$$V_{i+1} = g\left(V_i\right) = f_i\left(L\left(V_i\right)\right) \oplus f_i\left(R\left(V_i\right)\right) \tag{4.12}$$

Then, Eq.4.10 may be written as Eq.4.13.

$$\frac{\partial\left(V \cdot W\right)}{\partial V_i} = W_i + \sum_{j=i+1}^{n} \frac{\partial g_{j-1}\left(\ldots g_i\left(V_i\right)\right)}{\partial V_i} W_j \tag{4.13}$$

which means that greater gradient is attributed to V_i, implying that the network tends toward learning an architecture of depth i.

The ability of SDPPI to prevent overfitting may be understood by taking inspiration from ResNet [141]. Indeed, as stated earlier, each layer may be assimilated to a function. As a result, the output of a given branch at level j is:

$$\begin{cases} L_j = f_{j-1}\left(\ldots f_1\left(f_0\left(L_0\right)\right)\ldots\right) \\ R_j = f_{j-1}\left(\ldots f_1\left(f_0\left(R_0\right)\right)\ldots\right) \end{cases} \tag{4.14}$$

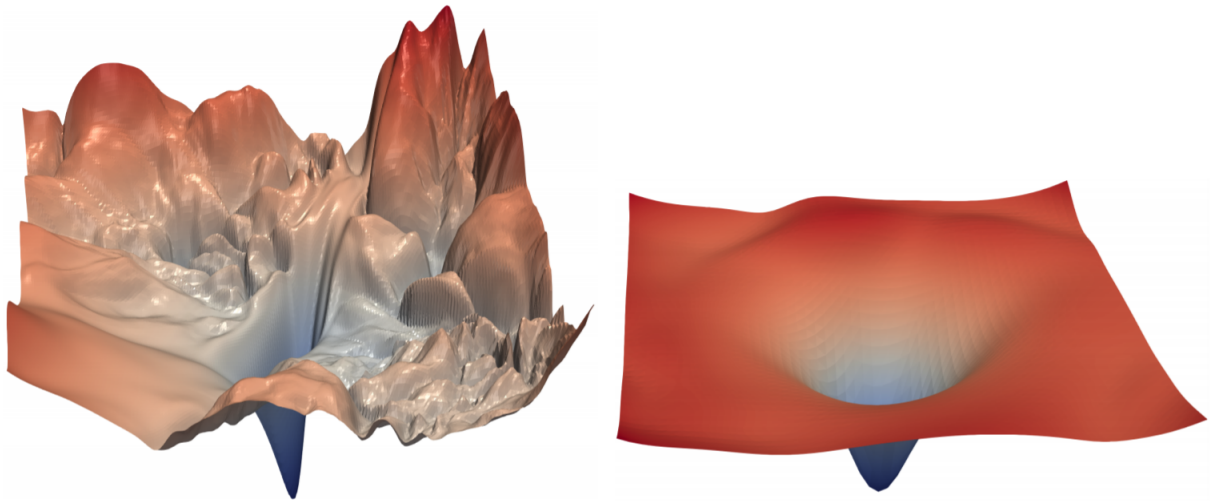
Then, the concatenated tensor becomes:

$$\begin{aligned} V_j &= \bigoplus_{i=0}^{j-1} \left[L_i \oplus R_i\right] \\ &= \bigoplus_{i=0}^{j-1} \left[f_i\left(\ldots f_0\left(L_0\right)\ldots\right) \oplus f_i\left(\ldots f_0\left(R_0\right)\ldots\right)\right] \end{aligned} \tag{4.15}$$

This means that the network input(L_0, R_0) appears at each complexity level, which implies that the gradient propagates through multiple paths, one for each complexity level. The periodical connections between the two branches may be assimilated to skip connections, as *Skip connections are extra connections between nodes in different layers of a neural network that skip one of more layers of nonlinear processing* [170].

These connections constitute additional paths for the gradient to propagate along. As a result, error accumulation is alleviated, convergence is accelerated, and the number of parameters is reduced, thus mitigating overfitting. The skip connections improve feature reusability while facilitating the training process (parameter optimization). Indeed, operations such as maximum pooling and convolutions discard information as a result of

downsampling. As it is lost, the information cannot be propagated to subsequent layers. The skip connections present in SDPPI allow re-inputting the discarded information into the network at each resolution level. The space associated with the loss function has a very large number of dimensions, which is equal to the number of parameters of the network. These spaces tend to have multiple local minima that make optimization of the network more difficult. Recently, Li demonstrated *et al.* [9] that skip connections contribute to the elimination of these detrimental minima, which makes the search space smoother, thus facilitating the optimization process. This is illustrated in Fig.4.5.



(a) without skip connections (b) with skip connections

Figure 4.5: Loss function parameter space with and without skip connections (with permission of the author [9])

4.3.2 Architecture of the various components

Our network is constructed of six components: the input and embedding layers, the convolutional and maximum pooling layers, the LSTM layer, the global maximum pooling layer, and the dense layers.

Input and embedding layers: There are 23 standard amino acids, namely alanine, arginine, asparagine, aspartic acid, cysteine, glutamine, glutamic acid, glycine, histidine, isoleucine, leucine, lysine, methionine, phenylalanine, proline, serine, threonine, tryptophan, tyrosine, valine, selenocysteine, and pyrrolysine. Each categorical amino acid is encoded with a number between 1 and 23, whereas ambiguous or unknown amino acids are attributed a value of 24. As mentioned earlier, sequences are restricted to 1,200 amino acids. Smaller amino acid sequences are padded with zeros. Therefore, the dimension of the input vector is fixed at 1,200. Naturally, the names (categories) neither reflect the similarity between amino acids nor their physicochemical properties. To solve this problem, a trainable embedding layer is added, as first proposed by [187]. The categorical input is first transformed to a set of one-hot vectors. Given a vocabulary of n amino acids and an embedding size of m, the embedding layer learns a $n \times m$ projection matrix. This layer learns, from the amino acids, a more informative and dense latent representation, which is better suited for machine learning.

Convolutional and maximum pooling layers: Convolutional networks were initially introduced in computer vision [188]. They are known for their ability to extract local features and for providing translation invariance [147]. As opposed to dense layers, they involve far fewer parameters, and therefore are less subject to overfitting. They evaluate the convolution between an input and a set of filters (also known as sliding windows or kernels). For each position of the filter, the outcome of the convolution operation is summarized or downsampled using a maximum pooling layer, which extracts the maximum value from the convolutional window. As illustrated in Fig.4.4, each branch of SDPPI consists of three convolutional networks. The number of filters as well as their sizes were determined, for each layer, by inspection. The first convolutional network consists of 16 filters of size 10, the second one is formed of 32 filters of size 8, and the last one has 64 filters of size 5. The filters perform a multiresolution analysis from low to high resolution in the forward direction. The first convolutional network performs a low-resolution analysis, as its filters are larger and less numerous, whereas the last convolutional network performs a high-resolution analysis, as the filters are smaller and more numerous. This approach is reminiscent of the wavelet transform [189] in the sense that a multiresolution analysis is performed, noise is reduced, and feature maps are compressed [190].

Long short-term memory network: The LSTM is a recurrent neural network with long and short-term memory that solves the gradient vanishing problem by adding control gates [7]. The LSTM is located just after the last convolutional network. Its role is essentially to introduce a long and short-term memory mechanism within the network after the multiresolution analysis is performed by the three convolutional neural networks and to reduce the number of output parameters.

Global maximum pooling layer: In addition to the local maximum pooling layers, our SDPPI network employs multiple global maximum pooling layers, as illustrated in Fig.4.4. These layers are involved in the periodical and hierarchical interaction between the two branches. The global layer determines the maximum for the entire feature map at each resolution level, each resolution level being determined by the corresponding convolutional networks. Therefore, the strength of the interaction between the two branches is based on the most salient feature—that is, the global maximum at a particular level. This interaction is repeated three times, once for each resolution level.

Batch normalization and dropout layers: The batch normalization layer performs a batch normalization, both in origin and scale, which improves the training convergence while mitigating overfitting [191]. The dropout layer randomly sets to zero a certain number of neurons. In addition to introducing some randomness into the network, the dropout layer regularizes the loss function. Therefore, the occurrence of over-trusted neurons is less likely, thus contributing to the overall network robustness [192].

Dense layer: Our network consists of two dense networks, both of which are located in the classification head, as illustrated in Fig.4.4. The first network, which is called the selecting dense layer, consists of a single layer that is activated by a ReLU function.

As explained earlier, the selecting dense layer role is essentially to determine the depth of the network according to the size and the complexity of the training set. The last layer, which is a decision layer, consists of a single neuron with a sigmoid activation function that evaluates the binding probability.

4.4 Experimental Evaluation

Our experimental evaluation consists of three sets of experiments. In the first set, our results are compared against the state of the art for benchmark datasets, namely DIP, HIPPIE, and INWEB, which are described below. In the second set, the performances and the scalability of our system are evaluated against a very large dataset, namely B-STRING, as introduced in Section II. The last set provides an empirical demonstration of the ability of our network to adapt its architecture to the size and complexity of the dataset. The calculations were performed with TensorFlow on a Compute Canada Beluga supercomputer with two 2.4 GHz Intel Gold 6148 Skylake CPUs, 186 GB of RAM, and four NVIDIA 16 GB V100 SXM2 GPUs.

Five datasets were employed in this study, namely Guo's benchmark dataset, the DIP dataset, the HIPPIE dataset, the INWEB dataset, and the B-STRING dataset. These datasets are described in the following subsections, and their characteristics are reported in Table 4.1. Our research focuses on Homo sapiens proteins, as they are the most suitable for therapeutic treatments [193]. Indeed, non-human proteins must be submitted to a humanization process intended to produce proteins that do not elicit an immune response and are safe for human usage without compromising efficacy [194]: a complex trial-and-error experimental process.

4.4.1 Guo's benchmark dataset

This dataset was employed by Guo *et al.* in their study [23]. It consists of 20,971 interactive protein pairs (positive examples) and 31,496 non-interacting pairs (negative examples), in which each protein consists of, at most, 1,200 amino acids. A validation set, consisting of 2943 interacting pairs and 3057 non-interacting pairs, was randomly selected, and the remaining pairs, 26,128 positive examples and 28,439 negative examples, were employed in five-fold cross-validation. Consequently, the dataset was essentially balanced.

Table 4.1: Characteristics of the datasets.

Dataset	Benchmark	STRING	DIP	HIPPIE HQ	HIPPIE LQ	INWEB HQ	INWEB LQ
Positive Pairs	29071	9366352	3547	34152	18315	141354	380318
Negative Pairs	31496	8225481	-	-	-	-	-
Total Pairs	61197	17591832	3547	34152	18315	141354	380318

4.4.2 DIP dataset

The Database of Interacting Proteins (DIP) dataset was first introduced by Xenerios *et al.* in 2002 [54]. Multiple releases have followed since. Version 20160430 was employed for this study. In order for the results to be comparable between datasets, and to avoid information leakage (a protein being in both the training and the testing set) [53], proteins with more than 1,200 amino acids were excluded, as well as proteins already appearing in Guo's dataset. Indeed, the DIP dataset was employed for testing, whereas Guo's dataset was employed both for training and testing. This is the reason proteins appearing in Guo's dataset were not considered—to guarantee that the results achieved were the result of a prediction and not of a regression. The filtering procedure resulted in 3,242 protein pairs.

4.4.3 HIPPIE dataset

The Human Integrated Protein-Protein Interaction Reference (HIPPIE) dataset [55] consists of 52,467 protein pairs. A confidence score is associated with each one of them. The dataset was divided into two subsets: a high-quality (HQ) dataset, for which the score was greater than 0.73, and the remaining pairs were attributed to a low-quality (LQ) dataset. To compare the results between datasets and to prevent information leakage [53], proteins with more than 1,200 amino acids, as well as proteins appearing in Guo's dataset were excluded.

4.4.4 INWEB dataset

The INWEB dataset was employed in Li's study [56], and it is one of the largest PPI datasets available. As with the HIPPIE dataset, the pairs are divided between low quality and high quality; the quality threshold was fixed at one. The high-quality dataset consisted of 8,672 distinct proteins, whereas the low-quality dataset contained 15,288 proteins, resulting in 141,354 PPI for the former and 380,318 interactions for the latter. The filtering process was similar to the process employed for the DIP and HIPPIE datasets.

Table 4.2: Five-fold cross-validation result for Guo's benchmark dataset.

Fold	0	1	2	3	4	Average
Accuracy	98.95%	98.80%	98.95%	98.94%	98.94%	98.92%

4.4.5 Comparison against the state of the art for protein-protein interactions

The performances of our SDPPI network were evaluated against Guo's dataset and the DIP, HIPPIE, and INWEB datasets. Recall that the accuracy of our network, against Guo's benchmark hold-out training set, was evaluating using five-fold cross-validation [195]. The accuracy obtained for each fold, as well as the average five-fold accuracy are reported in Table 4.2. Our results were compared against the state of the art for PPI, namely Li's [38] and Sun's [34], using the best model obtained from five-fold cross-validation (fold 2). The classification accuracy, precision, recall, and F1 scores [196] for SDPPI, Li, and Sun are reported in Table 4.3. The results confirm that our system consistently outperformed the others and yielded superior results. The receiver operating characteristic (ROC) curve, as well as the area under the curve (AUC) [197] associated with our network are reported in Fig.4.6: both indicators clearly demonstrate the quality of the results (a perfect classifier has an AUC of one). In addition, our approach was compared to the approach of Li [38] and Sun [34] against the following datasets: DIP, HIPPIE HQ, HIPPIE LQ, INWEB HQ, INWEB LQ, and Guo's hold-out testing sets. Their respective accuracies are reported in Table 4.4. Our network outperformed the state-of-the-art networks on all datasets except for HIPPIE HQ, for which it was *ex aequo* with Li [38]. To determine the significance of the results, a paired t-test (Student's t-test) [198] was performed for SDPPI against Li, and SDPPI against Sun. In the first case, the p-value was 0.02949, and in the second case, it was 0.00542, as reported in Table 4.5. Both p-values were lower than the significance threshold, which was fixed at 0.05. Therefore, the null hypothesis that the results were comparable was rejected, which implies that our approach surpasses the others.

Table 4.3: Comparison of SDPPI to the state of the art for Guo's hold-out testing set

Model	Accuracy	Precision	Recall	F1-score
SDPPI	**98.82%**	0.9881	**0.9878**	**0.9879**
Li *et al.*	98.78%	0.9891	0.9861	0.9876
Sun *et al.*	96.82%	-	-	-

4.4.6 Large-scale experiment

The performances of SDPPI against the B-STRING dataset were evaluated again using five-fold cross-validation. Recall that the dataset consisted of 19,356 Homo sapiens proteins participating in 5,879,727 PPIs. All the interactions were obtained from the STRING database [57]. In addition, 5,879,727 negative examples (non-interacting proteins) were generated with the method described in Section II. Therefore, nearly twelve million instances were employed to train and test the network. The results are reported in Table.4.6. The results were consistent from fold to fold, and an average accuracy of 96.08 % was achieved despite the large number of instances. These results clearly demonstrate the ability of our network to adapt to large datasets through the selecting dense layer, in addition to demonstrating the scalability of our approach. Indeed, a deeper network was automatically selected by the selecting dense layer to reflect the complexity and the size of the dataset. The corresponding ROC curve and the AUC are reported in Fig.4.7. They further illustrate the performance of our system.

4.4.7 Selecting dense layer: an empirical analysis

As explained in Section III, the depth of the network is automatically determined according to the size and complexity of the dataset. The outcome may be a shallower network, a deeper network, or any combination thereof; the architecture of the resulting network is determined by the selecting dense layer. In this section, we propose an empirical analysis of this mechanism. Small and large datasets were selected, namely Guo's dataset and the B-STRING dataset. For each dataset, the element-wise gradient with respect to the

Table 4.4: Comparison of SDPPI to the state of the art for six benchmark datasets.

Model	DIP	HIPPIE HQ	HIPPIE LQ	INWEB HQ	INWEB LQ	Benchmark Hold-Out Test
SDPPI	**97.19%**	95.99%	**95.33%**	**95.25%**	**95.89%**	**98.82%**
Li *et al.*	94.33%	96.08%	93.40%	93.07%	92.80%	98.78%
Sun *et al.*	93.77%	92.24%	87.04%	91.14%	87.99%	96.82%

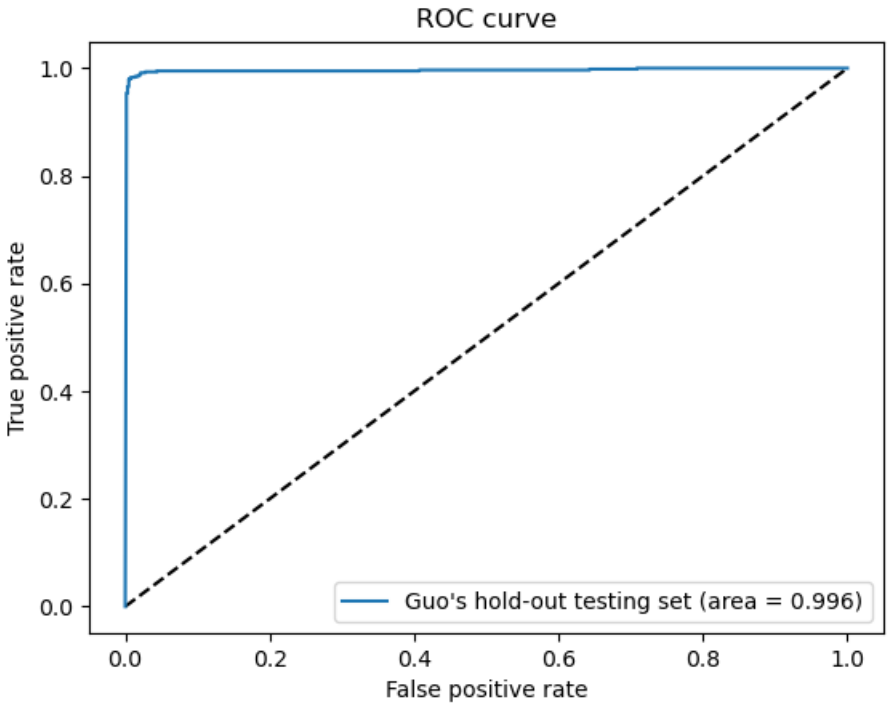


Figure 4.6: ROC curve and AUC for Guo's hold-out testing set [10]

selecting dense layer was evaluated. These gradients are represented as heat maps: a cold color corresponds to a low gradient and a warm color corresponds to a higher one. Our results are reported in Fig.4.8 and 4.9, for Guo's dataset and B-STRING, respectively. The left side of these figures corresponds to the shallower networks, while the right side corresponds to deeper networks. In Fig.4.8, more weight and gradient is attributed to the shallower layers (warmer colors) whereas in Fig.4.9, the deeper layers clearly dominate the shallower layers. Therefore, a shallower network was automatically created for the small

Table 4.5: Validation of the results of Table 4.4 with a Student's t-test.

Paired T-Test	SDPPI - Li *et al.*	SDPPI - Sun *et al.*
p-value	0.02949	0.00543

Table 4.6: Five-fold cross-validation results for the B-STRING dataset.

Fold	0	1	2	3	5	Average
Testing Accuracy	95.88%	96.15%	96.11%	96.15%	96.12%	96.08%

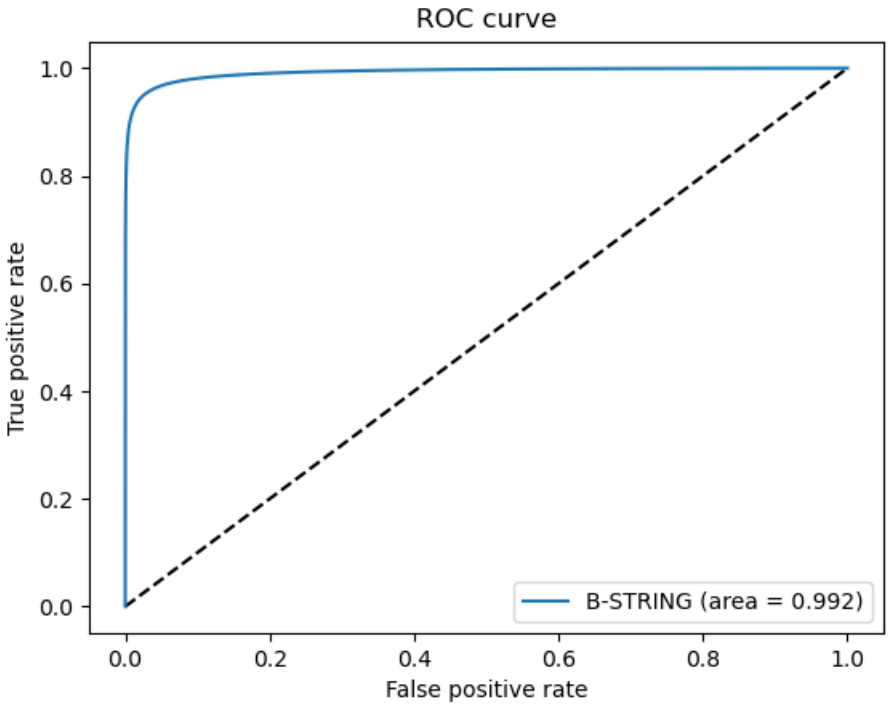


Figure 4.7: ROC curve and AUC for the B-STRING dataset.

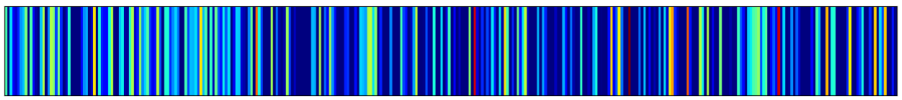


Figure 4.8: Gradient associated with the selecting dense layer for Guo's dataset.

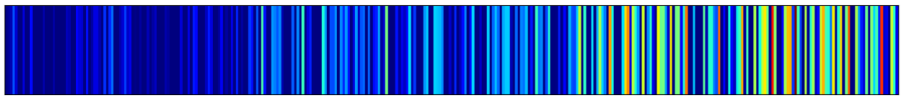


Figure 4.9: Gradient associated with the selecting dense layer for the B-STRING dataset.

dataset, and a denser network was generated for the larger and more complex one.

4.5 Summary

In this chapter, we address the so-called data realities that deep learning architectures for PPI face. We begin the chapter with a discussion on related work, notably focusing our attention on overfitting and lack of non-interacting proteins. Section 4.2 details our graph-based algorithm for generating a balanced version of the STRING data set. Section 4.3 details our deep learnable network architecture, while Section 4.4 presents our extensive experimental evaluation, discuss our results, and offer reflections.

In the next Chapter, we introduce a novel primary sequence-based approach for protein design.

Chapter 5

Primary Sequence Based Protein-protein Interaction Binder Generation with Transformers

The work presented in this chapter is based on our paper: *Primary Sequence Based Protein-protein Interaction Binder Generation with Transformers* [199] submitted for publication.

Amino acid sequences may be assimilated to pseudo-time series, i.e., the time corresponding to the position of the amino acid in the chain [200]. As such, transformer-based architectures that utilize the mechanism of self-attention may be utilized for protein design [161].

In this chapter, we introduced a primary sequence-based approach to design PPIs with two different transformers namely the AppendFormer and MergeFormer. We also address the multivalued problem in PPI design by using the binding score as well as a prior. A prior consists of the first n amino acids of the binder to ensure the uniqueness of the solution. The proposed system is evaluated against the B-STRING data set. During the training phase, information about the binding probability is employed concurrently with amino acid sequences to further constrain the problem. Given 10% of the binder as a prior, both of the transformers are able to achieve a Needleman–Wunsch score of 0.86. which indicates that our solutions significantly outperform a seq2seq baseline model.

This chapter is organized as follows: related work is covered in Section 5.1, while the new transformer-based deep learning architectures employed for binder prediction are described in Section 5.2. To the best of our knowledge, this is the first work that employs transformers for PPI design. The dataset and the preprocessing techniques are presented in Section 5.3 and the transformer training procedure is described in Section 5.4. The inference, the evaluation techniques, and the prior are addressed in Section 5.5, while the experimental results are reported in Section 5.6. Section 5.7 summarizes this chapter.

5.1 Related Work

Protein design is the rational design of new protein molecules to design novel activity, behaviour, or purpose and advance the basic understanding of protein function. Proteins may be designed from scratch (de novo design) or by making calculated variants of a known protein structure and its sequence (protein redesign). The most common methods for redesign are residue interactions clustering followed by high-affinity binding protein searching [123, 201], searching for an existing binding protein with the high affinity that is subsequently optimized [125, 202, 203], and incorporating metal-binding sites while optimizing their periphery [126]. As for de novo design, the most common approaches are optimizing existing binding sites [118, 204] and grafting either one [121, 205, 206] or two sides [207] of an existing site. All these techniques rely on domain knowledge such as the ternary structures (3-D shape) and information about residue interactions and complexes. For instance, metal-binding sites are restricted to nonpolar proteins [126]. Frequently, information such as the ternary structure is unavailable, thereby considerably reducing the effectiveness and generalizability of these methods.

There are relatively few studies about protein design based on deep learning. Based on the criterion that if the study be related to sequence-based protein design, we select a few studies. Among the most important are signal peptide generation by attention-based neural networks [132], antibody design using a long short-term memory (LSTM)-based deep generative model from a phage display library for affinity maturation [133], sequence-based deep learning antibody design for in silico antibody affinity maturation

[134,135], improving the accuracy of neural network-based computational protein sequence design with DenseNet [47] using backbone structures, computational protein design with deep learning neural networks [49], and the generation of functional protein variants with variational autoencoders [136]. Wu *et al.* approach lacks the ability to solve mutivalued problems, and all these methods require either knowledge of the 3-D structure [47] or domain-specific knowledge [134, 135].

An approach for solving the antibody-antigen binding problem is proposed in [134]. Given an antibody-antigen pair, a one-hot embedded matrix and an adjacent matrix are first evaluated. Then, the two matrices are fed to a neighborhood aggregation layer and a graph neural network (GNN) to obtain a binary classification. Being confined to binary classification, the Kang *et al.* method is unsuitable for determining the sequences of binding proteins.

A single-domain antibody (sdAb), also known as a nanobody, is an antibody fragment consisting of a single monomeric variable antibody domain. Like a whole antibody, it can bind selectively to a specific antigen. Shin *et al.* employ an autoregressive dilated convolutional model for predicting nanobody sequences [135]. As the model is autoregressive, the amino acids are predicted sequentially, with the prediction of a new amino acid being based on the previous ones. Wang *et al.* propose a framework, consisting of three cooperating deep learning models, that predicts the amino acid sequence of the binder based on the backbone structure (3-D) of the target protein [49]. The applicability of this framework is limited to target proteins for which the backbone structure is known.

In one study, a variational autoencoder is employed to generate new proteins [136]. In this work, by modulating the latent representation, it is possible to generate new proteins. Unfortunately, the autoencoder does not account for the target protein, making it unsuitable for interacting proteins.

The approach by [47] is based on DenseNet [142], which is a convolutional neural network with several parallel skip connections that was originally introduced in computer vision for diagnostics and identification [208, 209] and more recently was used to diagnose Covid-19 infections from computed tomography scans [210]. In the proposed architecture, DenseNet is extended to volumetric data [47]. The latter consists of regular volumetric grids

in which each volume element is related to the atomic density, while each grid corresponds to a particular atomic type, thus focusing on the 3-D structure. The amino acid sequence of the target residues is predicted from the distribution of backbone atoms. Qi *et al.* achieved an accuracy of 53.2% with a 5-fold cross-validation, which constitutes an improvement of more than 10% compared to the state of the art. Unfortunately, because of the one-to-one correspondence between each amino acid and the backbone, the target protein and the binder must have the same number of amino acids, which strongly limits the applicability of this approach. Indeed, most of the time, the binder and the target protein do not have the same number of amino acids [211].

This limitation may be circumvented by employing an autoregressive model [212] in which the new amino acid is predicted recursively from the previous one. The recursive prediction process ends when an end-of-sequence (EOS) token is predicted. Therefore, the model can predict sequences with variable lengths, thus allowing for binders and targets not having the same number of amino acids, which is by far the most common scenario. Such a strategy was followed by Sake *et al.* [133] for antibody design against kynurenine, a metabolite used in the production of niacin. The autoregressive model consisted of an LSTM [7]. An extended dataset was obtained with the phage display laboratory technique [213] and an antigen-binding (Fab) library [214]. Sake *et al.* model can predict sequences with arbitrary lengths. Nonetheless, the predictions are based on a specific probability distribution, the one associated with the maturation of antibodies against kynurenine, which means that the approach cannot be easily generalized. A similar problem occurs in [135].

The study by Wu *et al.* is the one closest to our work [132]. The authors employ an autoregressive model to predict binders of variable length. Their training dataset consist of signal peptides (short amino acid sequences) and their corresponding target proteins, and their architecture is based on a seq2seq transformer. Recall that a transformer is a deep learning model that adopts the mechanism of self-attention [161]. As such, seq2seq transforms the target protein amino acid sequence into the binder amino acid sequence. The main components are the encoder and the decoder. The encoder converts each input sequence into a more informative and compact latent (hidden) representation. The decoder reverses this process, converting the latent representation into an output sequence using the

previous output as the input context. The network includes two optimization mechanisms: attention and beam search. The input to the decoder is a single vector that stores the entire context. Attention allows the decoder to look at the input sequence selectively. As for beam search, instead of picking a single output (amino acid) as the output, multiple highly probable choices are retained while being represented as a data structure. For testing, de novo signal peptides were generated with the trained model. It was determined experimentally that 48% of the peptides bound effectively. These peptides shared on average 73% of their sequence with the corresponding closest known signal peptides.

Yet, there are some limitations to the method of Wu *et al.* [132]. First, it is essentially limited to small molecules, whereas our main concern is PPIs. Second, as opposed to signal peptides, PPIs are less specific. As a result, a given target protein may interact with multiple binders. Indeed, only 0.13% of proteins studied by Wu *et al.* interacted with more than one binder. The situation is completely different for PPIs. For instance, in the Search Tool for the Retrieval of Interacting Genes/Proteins (STRING) dataset [57], most proteins interact with at least four binders. The distribution of the number of interactions per protein for the STRING dataset is reported in Fig. 5.1. Therefore, protein design is made more complex by the fact that there are multiple solutions for a given target protein (multivalued problem). This is to be contrasted with signal peptides, for which the solution is essentially unique, which means that the problem is more constrained.

As indicated earlier, all state-of-the-art methods either require knowledge of the 3-D structure of the target protein or domain-specific knowledge. Therefore, they may encounter difficulties when applied to new problems, meaning that their generalizability is rather limited. In addition, very often, the 3-D structure is not readily available. On the other hand, amino acid sequences may be easily obtained. For these reasons, we introduce a solution for designing binder proteins that is based on amino acid sequences. Our approach aims at designing target proteins solely from amino acid sequences while addressing the multivalued problem inherent to PPIs. The architectures of our models are presented in Section 5.2.

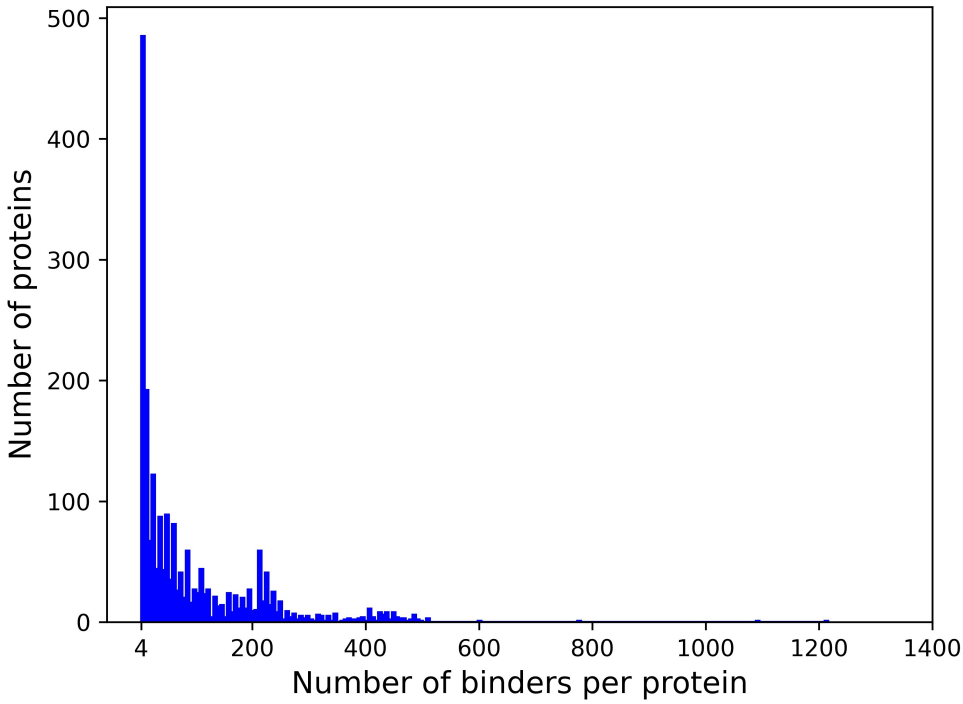


Figure 5.1: Distribution of the number of interactions per protein in the STRING dataset.

5.2 Transformer for Binding Protein Prediction Given a Known Target Protein

Recall that a transformer is a deep learning architecture that transforms one sequence into another and that it is based on seq2seq. It is an autoregressive model with an attention mechanism that was first proposed for neural machine translation [161]. As opposed to recurrent neural networks [215], no sequential treatment of the input vector is required. In addition, the scaled dot product attention mechanism densely connects the input and output. Therefore, the context vector is linked to the entire input, thus overcoming the problem associated with long-term dependencies [216]. As stated earlier, PPI is a multivalued problem since more than one binder may be associated to a target protein, as reported in Fig. 5.1. To obtain a particular solution, some prior knowledge must be introduced in order to constrain the solution to a particular binder. To achieve a unique solution, it is proposed to employ the binding scores between target proteins and binders in addition to prior knowledge of the amino acid sequence of the binder. Here, the prior refers to the first amino acids of the target protein that are known.

In our study, two transformer architectures are introduced: one is optimized for short priors while the other is optimized for longer priors. The first AppendFormer architecture extends the original transformer as introduced by Vaswani *et al.* [161] with the objective of learning in the presence of short priors. The input AppendFormer transformer consists of the amino acid sequence of the target protein and the discretized binding score that is appended in between the padding and the EOS tokens. The EOS token is employed to determine the localization of the discretized binding score within the input sequence. The architecture of the AppendFormer is reported in Fig. 5.3. The transformer consists of an encoder and a decoder, where the encoder transforms the target protein amino acid sequence into a latent representation, while the decoder generates the amino acid sequence of the corresponding binder protein from the latent representation. The decoder has a recurrent architecture, which means that its output is reinjected into its input as illustrated in Fig. 5.3. The output of the decoder is initialized with a beginning-of-sequence (BOS) token as well as with a prior about the binder amino acid sequence that consists of its first n amino acids. The Bayesian prior is employed to constrain the solution to a unique

binder. Indeed, as stated earlier, because PPI is essentially a multivalued problem, it is necessary to introduce some constraints to obtain a unique solution. For both the encoder and the decoder, the amino acid sequences are processed with an embedding layer [161] that ensures that both sequences have a continuous representation while mapping amino acids with similar properties to the same region. As the attention mechanism does not account for the order in which the amino acids appear in the sequences, a positional encoder [161] is added after each embedding layer, thus allowing the transformer to account for the order in which the amino acids appear without jeopardizing parallel processing. The positional layer is followed by a multi-head self-attention layer. The aim of this layer is to capture the interrelations among the various amino acids such as self-folding and self-interaction either for the target protein or the binder, as demonstrated in [163]. Then, the latent representation is generated by a multi-head attention layer. For this layer, the query keys are associated with the decoder, while the memory and value keys are associated with the encoder, thus allowing the capture of the interrelations between the amino acids of the target protein and those of the target proteins. The multi-head attention layer is followed by a classifier that consists of a feed-forward neural network, a linear, and a softmax activation function. The classes correspond to the twenty standard amino acids as well as to a potentially unknown amino acid (multi-class problem). During the training phase, the classifier allows for determining if the amino acid generated by the decoder is the right one. Various skip connections [141] are added along the way to reduce the complexity of the loss function landscape, therefore facilitating its optimization [9].

We also introduce the MergeFormer architecture, designed for longer priors, i.e., a scenario in which a larger portion of the target protein is known. In the second architecture, a merge layer is introduced between the encoder and the decoder to inject the binding score deeper into the encoder just after the feed-forward neural network. Therefore, the binding score is decoupled from the amino acid sequence as opposed to the first architecture. The MergeFormer architecture is described in Fig. 5.4. The input of the encoder consists of the amino acid sequence of the target protein. The network is designed in such a way that the effect of the binding score may be learned independently for each amino acid. Therefore, a constant vector is created, the constant being the binding score, which has the same dimension as the latent or hidden representation. This vector is concatenated to

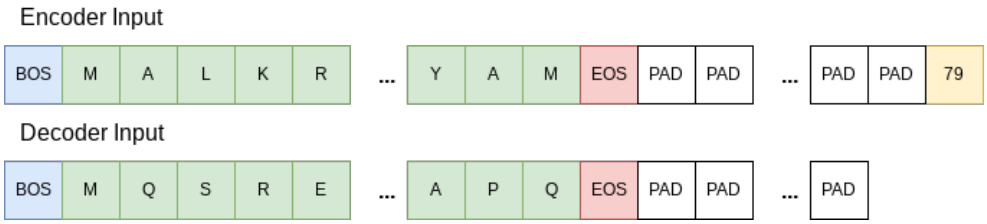


Figure 5.2: Example input for encoder and decoder in AppendFormer.

the latent representation, the resulting vector is transposed, and a linear transformation is applied, thus allowing for the learning of the interrelations between the binding score and the amino acids. Finally, the linearly transformed representation is transposed to restore the original topology. The main advantages of MergeFormer are:

- No vocabulary is required for the binding affinity, thus reducing the number of tokens that must be learned by the network.
- For AppendFormer, the discretized binding score is appended at the end of the protein sequence. Since the amino acid sequence may have an arbitrary length, the position of the binding score is unknown and therefore must be learned by the network. In the case of MergeFormer, the binding score is injected directly into the latent space while its relation to each amino acid is learned.
- The fact that the binding score is independent of the amino acid sequence allows for initializing the encoder weights with techniques such as masked language modeling (MLM), which has been shown to substantially improve performances in NLP when employed in conjunction with the bidirectional encoder representations from transformers (BERT) [217].

The hyperparameters employed for both transformers are reported in Table 5.1. The table consists of four transformers, namely one MergeFormer architecture and three AppendFormer architectures, with the difference in the three AppendFormer architectures being the number of neurons in the feed-forward neural network.

To further evaluate the proposed transformers, their performances are compared against a seq2seq baseline with LSTM layers [40]. We employ this model as a baseline since

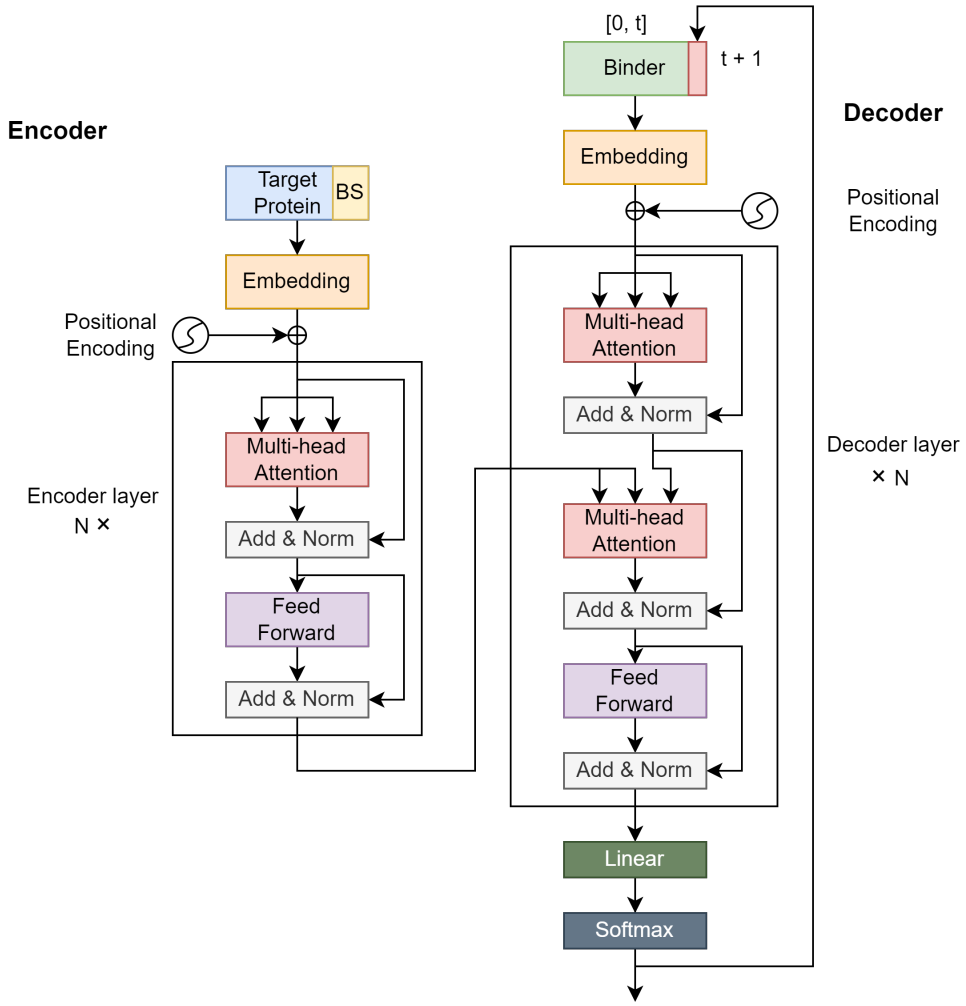


Figure 5.3: Architecture of AppendFormer.The digitized binding score BS is inserted at the end of the sequence. The number N of encoder layers was 3 in this study.

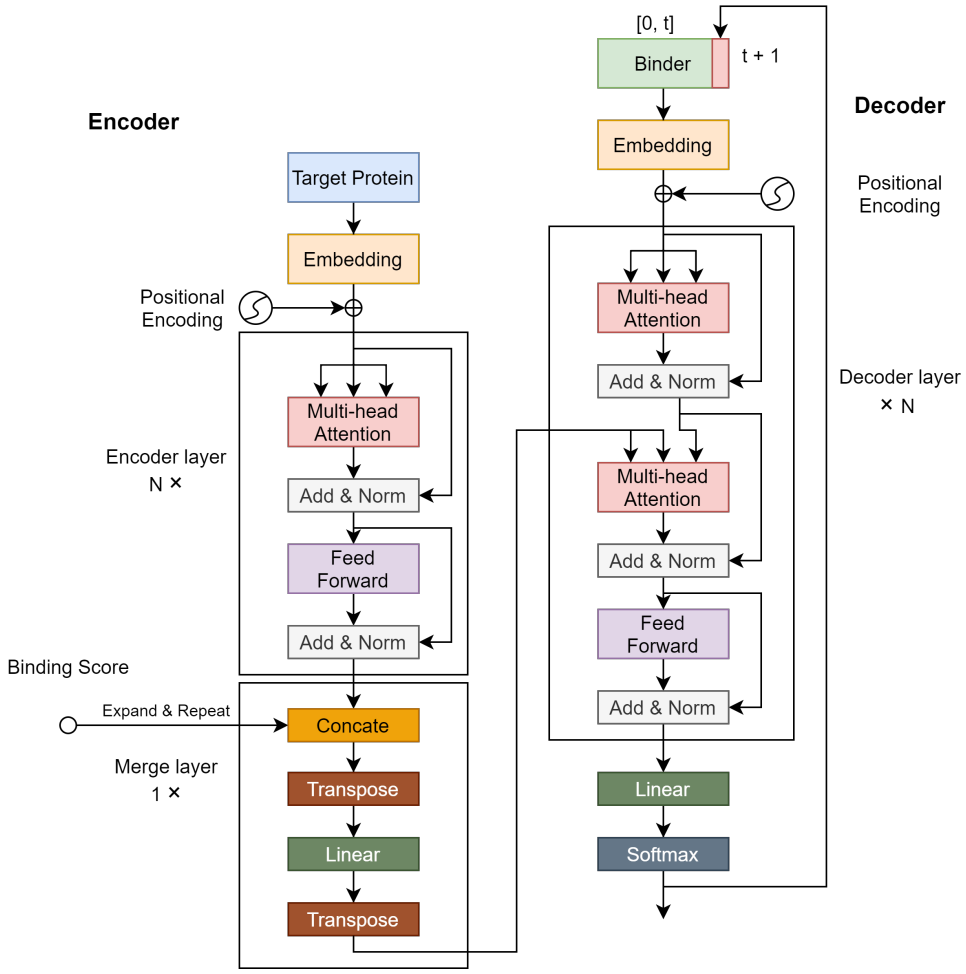


Figure 5.4: Architecture of MergeFormer. The input is the amino acid sequence of the target protein, while the binding score is concatenated with the latent representation. $N = 3$ in this study.

Model	MergeF	AppenF-s	AppenF-m	AppenF-l
$n_{encoder}$	3	3	3	3
$n_{decoder}$	3	3	3	3
P_{drop}	0.1	0.2	0.2	0.2
d_{ffn}	1024	1024	1280	1792
d_{model}	256	256	256	256
n_{head}	8	8	8	8
v_{input}	31	131	131	131
v_{output}	31	31	31	31

Table 5.1: Hyperparameters associated with the transformers: $n_{encoder}$ and $n_{decoder}$ represent the number of layers in the encoder and the decoder, respectively, d_{model} and d_{ffn} are the dimensions of the embedding and feed-forward neural network, respectively, and P_{drop} is the dropout probability. The number of heads in the multi-head attention mechanism is given by n_{head}, while the size of the input and output vocabulary is represented by v_{input} and v_{output}, respectively. For MergeFormer, these are equal as they correspond to the sum of the number of amino acids, the number of special amino acids, and the number of special tokens. For AppendFormer, v_{input} is larger by the number of tokens.

it famously surpassed the state of the art for machine translation in the Conference on Machine Translation (WMT'14) English to French Challenge [218]. To date, it remains one of the most widely used seq2seq models and has been successful in a wide range of applications [219–221]. Specifically, it has also been successfully adopted in protein design applications [222–224].

Its architecture is described in Fig. 5.5. As for AppendFormer and MergeFormer, baseline model consists of an encoder and a decoder. The key difference is that the attention layers are replaced by LSTM layers. The number of neurons in the LSTM layers were set to 512 by inspection. The baseline model in AppendFormer and MergeFormer possesses embedding layers that convert the amino acid sequences into hidden or latent representations that are subsequently processed by the LSTM layers. The LSTM layer associated with the decoder is initialized from the internal states of the encoder's LSTM layer. These internal states consist of the cell state and the hidden state. The decoder's LTSM autoregressively generates the amino acid sequence for the binder, given the target protein. In AppendFormer and MergeFormer, the amino acids are predicted with a linear layer employing a softmax activation function.

5.3 Dataset and Preprocessing

The dataset employed for training and testing is the STRING dataset that was originally proposed by Szklarczyk *et al.* in 2019 [57]. For the species Homo sapiens, the dataset consists of 11,759,455 interacting protein pairs, each of which is characterized by a binding score. Amino acid sequences consisting of more than 256 amino acids are excluded because the computational requirements for the self-attention mechanism grow quadratically with the number of amino acids [225]. In addition, since we are interested in binding proteins with high affinities, only proteins for which the binding score is between 0.9 and 1.0 are retained [226–229].

Recall that a transformer architecture is asymmetric in the sense that the binder may become the target protein and vice versa. Therefore, the dataset was extended by inverting the binder and the target protein. As a results, 96,744 interacting pairs involving 2091

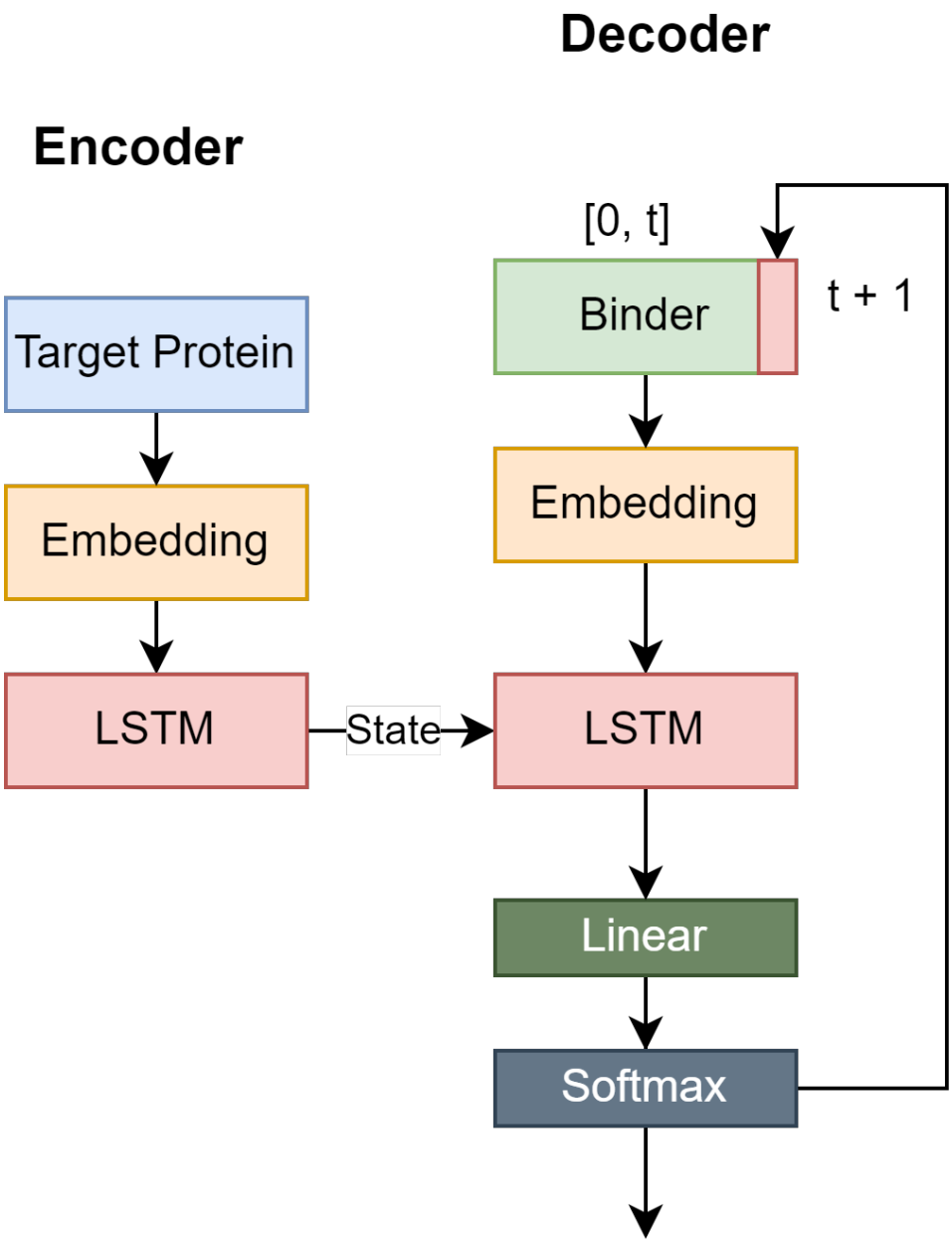


Figure 5.5: Architecture of the seq2seq baseline model.

distinct proteins were obtained in conjunction with their binding scores. Since a seq2seq model is employed, the amino acids are assimilated to tokens, while the amino acid sequences are assimilated to sentences. As in NLP, each sequence starts with a BOS token and ends with an EOS token. The BOS token is employed to initialize the recursive prediction at the inference stage, while the EOS token marks the end of the sequence. Therefore, a sequence of arbitrary length may be predicted. Sequences with fewer than 256 amino acids are padded with padding tokens to be able to predict smaller amino acid sequences.

For AppendFormer, the binding score is discretized into 100 bins ranging from 0.9 to 1.0, with each bin corresponding to a particular binding score. The number of bins was determined through experimentation. As a result, the vocabulary needed to be increased by 100 tokens in addition to those associated with the amino acids. For MergeFormer, since the binding score is injected directly at the level of the latent space, no additional tokens are required. Network training and the priors are addressed in Section 5.4.

5.4 Network Training

Teacher forcing, which is an essential part of many seq2seq architectures [230], is employed at the training stage. Being an autoregressive model, a transformer predicts the next amino acid from the latent representation of the input sequence (target protein) and the previous predictions of the decoder. In teacher forcing, during training, the predicted value at position t is replaced by the true value if the prediction is erroneous. As indicated earlier, given a target protein, the solution in terms of binders is multivalued. Therefore, due to the multiplicity of solutions, an error may be made when predicting an amino acid. The model being autoregressive, this error may jeopardize subsequent predictions in addition to causing error accumulation. Teacher forcing avoids the accumulation of errors by providing the decoder with the true input, resulting in a faster convergence during the training phase.

With teacher forcing, the training may be drastically accelerated. Without teacher forcing, the predictions have to be made recursively because the decoder's output is reinjected into the decoder's input. This means that the prediction for the current position is always based on the previous one. The recursive process eliminates any possibility for parallel

training. However, with teacher forcing, the decoder is provided with ground truth labels rather than the model's previous predictions. Therefore, the sequence prediction may be partitioned into multiple independent sub-tasks (parallelization). These sub-tasks can be distributed to many nodes and calculated concurrently, thus accelerating the training process. The loss function is optimized with the Adaptive Moment Estimation (ADAM) stochastic optimizer technique [231].

For the loss function, with teacher forcing training, each amino acid prediction on the chain may be performed independently, therefore resulting in independent multiclass classification problems, with the classes corresponding to the amino acid types. It follows that the loss function is a sparse categorical cross entropy [174] that is described by Eq. 5.1. w is the weights of the model. The output of the model and the ground-truth label are noted as $\widehat{y}$ and y respectively. As in [161], the padding tokens are ignored during training.

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\widehat{y}_i) \tag{5.1}$$

As for the original transformer, a dynamic learning rate with warm-up and decay is employed as it improves convergence during training. We use the same for our AppendFormer model since it is similar to the original transformer architecture as shown in Eq. 5.2. For our MergeFormer model, we design a dynamic learning rate with spiking followed by a constant learning rate, as described by Eq. 5.3.

$$l_{\text{appe}} = d_{\text{model}}^{-0.5} \cdot \min(n_{\text{s}}^{-0.5}, n_{\text{s}} \cdot n_{\text{ws}}^{-1.5}) \tag{5.2}$$

$$l_{\text{merg}} = \begin{cases} l_{\text{init}} \cdot \frac{n_{\text{s}}}{n_{\text{ws}}} & n_{\text{s}} \leqslant n_{\text{ws}} \\ \max(l_{\text{init}} \cdot \frac{1}{1+\alpha\cdot(n_{\text{s}}-n_{\text{ws}})}, \epsilon) & n_{\text{s}} > n_{\text{ws}} \end{cases} \tag{5.3}$$

where n_{s} represents the number of training steps, $n_{\text{ws}}$the number of warming steps, ϵ the lower boundary, α the learning rate decay speed, and l_{init} the initial learning rate.

The learning rate for MergeFormer is slightly different. To avoid local minima, both the lower boundary and the initial learning rate must be smaller [190]. The learning rates for both AppendFormer and MergeFormer are reported in Fig. 5.6.

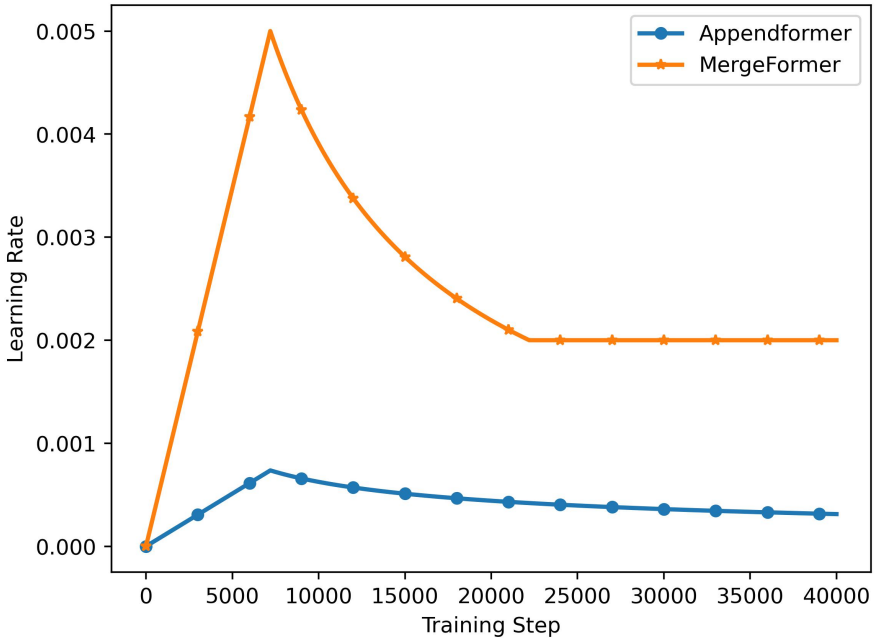


Figure 5.6: Dynamic learning rates for AppendFormer and MergeFormer.

5.5 Inference, Evaluation, and Prior

Recall that both AppendFormer and MergeFormer are autoregressive models, which means that the prediction of a new amino acid is based on the previous one. The encoder generates a latent representation from the target amino acid sequence and the binding score, the latter being appended to the sequence in AppendFormer or injected into the latent space in the case of MergeFormer. The latent representation acts as a key and a value for the multi-head attention mechanism associated with the decoder.

Initially, when the amino acid sequence of the binder is completely unknown, the sequence is initialized with a BOS. Then, the decoder predicts the first amino acid. Being a multiclass classifier, the decoder predicts an occurrence probability for each of the twenty amino acid types or classes. The amino acid with the highest probability is selected with a greedy search strategy [232]. The output sequence is updated with the outcome of the greedy search, and the recursion is repeated until an EOS is generated by the discriminator, allowing for the prediction of sequences of arbitrary length.

The greedy search algorithm only retains the amino acid with the highest classification probability. Because there is more than one binder for a given target, many correct solutions are possible, at least at the beginning of the prediction process. The real and unique solution may only be inferred when more amino acids are predicted along the way. This problem may be overcome with beam search [233]. Instead of considering only the first b_w predicted amino acid, beam search considers the first predicted amino acid, with the last one being conditional on the first $b_w - 1$. For instance, for $b_w = 2$, the probabilities associated with both amino acids (bi-gram) are multiplied to obtain the conditional probability of the bi-gram, with the probability of the second amino acid being conditional upon the first one. Then, the bi-gram with the highest conditional probability is selected and the process is repeated for each new prediction. As a result, the results are improved, and the issues associated with the multivalued problem are mitigated.

To further constrain the multivalued problem, a prior is attributed to the binder. Such priors are common in protein design, especially when mutation-based techniques are employed [118, 121, 204–207]. Their application is based on the observation that, without the prior, the transformer would not have enough information, or constraints, to converge

onto a particular solution [234]. In our work, a prior consists of the first n amino acids forming the binder for both AppendFormer and MergeFormer. In the first case, the prior is appended to the input of the AppendFormer, while, in the second case, the prior is injected directly into the latent space between the encoder and the decoder. Depending on its length, the prior may be assimilated to a peptide or a polypeptide. The prior may be assimilated as a form of teacher forcing. Indeed, for the first n amino acids of the binder, the amino acids generated by the decoder are replaced by the corresponding amino acids of the prior. From the $n+1$ prediction, the amino acid is directly predicted via the decoder with beam search. Additionally, during the predicting stage, teacher forcing is only applied within the section of prior. During training, in contrast, teacher forcing is engaged in the entire process.

Because the ground truth is known for the prior, it is not involved in the evaluation. Therefore, all the evaluation metrics ignore the prior and only focus on the amino acid directly generated by the decoder, thus avoiding the introduction of bias in the evaluation procedure. Four metrics are employed for the evaluation, namely the token matching rate (TMR), bilingual evaluation understudy (BLEU), Needleman–Wunsch, and Smith–Waterman metrics.

- Token matching rate: The token matching rate is evaluated over the whole test set. It corresponds to the ratio between the total number of tokens (amino acids) correctly predicted to the total number of tokens. The metric ranges between 0 and 1.

$$TMR = \frac{n_{\text{token\_match}}}{n_{\text{token}}} \tag{5.4}$$

- BLEU metric: The BLEU score [235] is a metric that was originally developed for evaluating the quality of texts that have been machine-translated from one natural language to another. In this study, the BLEU score is the logarithm of the geometric mean of the modified n-gram precisions for the amino acid prediction, with an n-gram corresponding to a contiguous sequence of amino acids of length n. Short n-grams are penalized with a brevity penalty score [235]. Here, the value of n ranges between 1 and 4, while the BLEU score lies between 0 and 1. The metric depends only on the n-grams and not on their positions within the sequence. Therefore, the

BLEU score determines the similarity between the true and predicted sequence in terms of their n-grams. The BLEU score is also related to the protein structure quality. Indeed, the number of amino acid substitutions is significantly correlated with neighborhood preference among amino acids [236]. As reported by [237], the neighborhood preference may be employed to assess structural quality.

- Needleman–Wunsch metric: The Needleman–Wunsch algorithm determines the best global alignment and similarity measure between two protein sequences with dynamic programming [238]. Given an amino acid sequence, each correctly predicted amino acid is attributed a reward of 1, while a penalty of 0 is attributed otherwise. The similarity measure is the ratio between the sum of the rewards and penalties and the total number of amino acids. This is a number between 0 and 1 whose value is independent of the number of amino acids forming the sequence.

- Smith–Waterman metric: The Smith–Waterman algorithm [239] is based on local alignment, which compares amino acid segments of all possible lengths and optimizes the similarity measure with dynamic programming. The measure itself is evaluated in the same way as for the Needleman–Wunsch metric. It is a number between 0 and 1 whose value is independent of the number of amino acids forming the sequence.

To further ascertain the performances of the transformers, the classification accuracy and the top-5 classification accuracy are evaluated. Absorbed from classification tasks, Top-5 accuracy means that the predicted amino acid is among the five best results in terms of prediction probability.

The hyperparameters of the transformers were determined by evaluating and comparing the performances of the models for various values of the hyperparameters. As stated earlier, the sequences are evaluated recursively, meaning that the training process cannot be parallelized making the evaluation of the hyperparameters computationally prohibitive. For this reason, as explained in Section 5.4, teacher forcing was employed to train each model. Indeed, teacher forcing allows the prediction of each amino acid independently of the others, thus allowing for parallelization of the calculations which reduces the time required for training [240].

Model	Trainable Parameters
AppendFormer-s	5,579,039
AppendFormer-m	6,367,007
AppendFormer-l	7,942,942
Baseline	3,173,663
MergeFormer	5,620,519

Table 5.2: Number of trainable parameters for each transformer.

Model	Accuracy	Top-5 Accuracy
AppendFormer-s	$93.07\% \pm 0.20\%$	$98.00\% \pm 0.04\%$
AppendFormer-m	$94.03\% \pm 0.11\%$	$98.51\% \pm 0.01\%$
AppendFormer-l	$95.81\% \pm 0.42\%$	$99.02\% \pm 0.09\%$
Baseline	$96.53\% \pm 0.29\%$	$98.41\% \pm 0.16\%$
MergeFormer	$96.41\% \pm 0.48\%$	$98.38\% \pm 0.31\%$

Table 5.3: 5-fold cross-validation accuracy and top-5 accuracy with teacher forcing.

Our experimental results are presented in the next section.

5.6 Experimental Results

The best models and hyperparameters were determined from the accuracy and the top-5 accuracy as obtained by 5-fold cross-validation with teacher forcing. Recall that teacher forcing was employed to parallelize amino acid predictions, thus allowing for faster convergence. The best performances for AppendFormer (AppendFormer-l) were obtained with a feed-forward neural network consisting of 1792 neurons. The accuracy and the top-5 accuracy for each model are reported in Table 5.3, while the corresponding number of parameters associated with the best models appear in Table 5.2. While MergeFormer and AppendFormer-l have comparable performances in terms of accuracy, MergeFormer has much fewer parameters than AppendFormer-l, thus reducing the risk of overfitting while facilitating convergence during the training phase. AppendFormer-l has slightly better per-

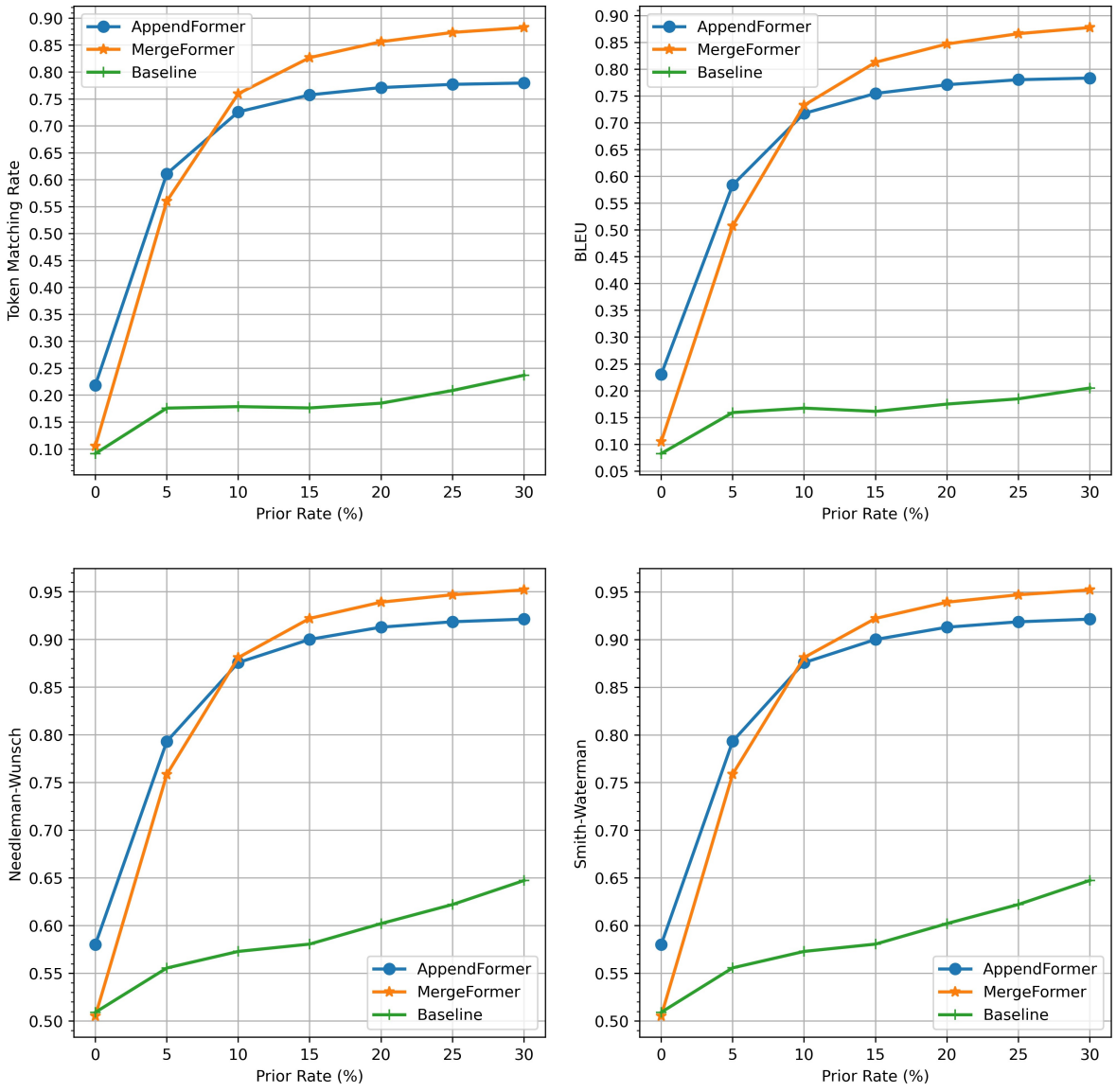


Figure 5.7: Evaluation of the performances of AppendFormer, MergeFormer, and the seq2seq baseline model according to the importance of the prior for four metrics, namely the token matching rate, BLEU, Needleman–Wunsch and Smith–Waterman.

formance in terms of top-5 accuracy than MergeFormer, while the situation is the opposite

for accuracy.

The models that obtain the best-three accuracy, Append-Former-l and MergeFormer, as well as the baseline seq2seq model were evaluated with the token matching rate and the BLEU, Needleman–Wunsch, and Smith–Waterman metrics. It should be noted that teacher forcing was not employed during the testing phase as it would bias the results by providing the network with the correct solution. These metrics were evaluated for different priors on the binding proteins. The priors were obtained by providing the decoder with a certain percentage of the first amino acids forming the binder. This number was restricted in between 0% and 30% of the total length. As stated in Section 5.2, this prior is required in order to constrain the multivalued problem as multiple binders may bind to a given target protein. Therefore, the prior ensures that the right binder is found for a particular interaction. The results for the four metrics are reported in Fig. 5.7. First, we turn our attention to the Needleman–Wunsch and Smith–Waterman metrics since they are widely used in bioinformatics. The best performances in terms of the Needleman–Wunsch and Smith–Waterman metrics are obtained with AppendFormer-l when no prior is employed, the metrics being equal to 0.58. This value increases to 0.61 for AppendFormer-l with a 5% prior, 0.86 with a 10% prior (both for AppendFormer-l and MergeFormer) and up to 0.95 with a 30% prior with MergeFormer. AppendFormer-l slightly outperforms MergeFormer when the prior is inferior to 10%, while MergeFormer strongly outperforms AppendFormer-l otherwise, improving performances by up to 40% when a 30% prior is employed. In comparison, as illustrated by Fig. 5.7, the seq2seq baseline model has relatively poor performances with the Needleman–Wunsch and Smith–Waterman metrics, barely reaching 0.65 when a 30% prior is employed. These results show that AppendFormer is suited for short priors (less than 10% of the entire sequence), while MergeFormer performs better with longer priors (more than 10%), thus confirming the validity of our architectures. Both transformers outperform the baseline seq2seq model. The performances in terms of Needleman–Wunsch and Smith–Waterman metrics are quite similar, meaning that no local alignment is required, and thus, that the sequence is properly predicted up to a global alignment.

The same general behaviour is observed for the token matching rate and BLEU results. Although these two metrics were reported for the sake of completeness, as stated above,

the Needleman–Wunsch and Smith–Waterman metrics are the most relevant from a bioinformatics perspective; token matching and BLEU scores are utilized more by researchers familiar with NLP evaluation metrics. Indeed, these results further highlight the successes of AppendFormer and MergeFormer.

5.7 Summary

We introduced novel deep learning architectures for therapeutic antibody and drug design based solely on amino acid sequences. A new primary sequence-based approach for predicting the amino acid sequence of the corresponding binder from a given target protein amino acid sequence was introduced. As a target protein may interact with more than one protein, the problem was constrained by employing the binding score and a prior on the binder. The binder was predicted using the transformer deep learning approach. Two architectures were created: a transformer in which the discretized binding score is appended to the target amino acid sequence and a transformer for which the binding score is injected directly into the latent space between the encoder and the decoder, resulting in fewer parameters and a learnable relation between the binding score and the latent space. To improve convergence and parallelize the calculations, training was performed with teacher forcing. Our results are promising and shows that transformer architectures outperform the state-of-the-art.

Next, in Chapter 6, we present our conclusions and details of our future work.

Chapter 6

Conclusions and Future Work

The contributions of this thesis are as follows.

6.1 Contributions

We presented a novel architecture for PPI testing that focused on amino acid sequence time series and incorporated an attention mechanism for learning from the folding and self-binding properties inherent in proteins. Our SPNet networks are characterized by the use of residual networks to mitigate gradient explosion and disappearance, a multilevel pyramid feature structure encompassing various PPI mechanisms, Siamese networks connected via a random projection and a self-attention mechanism that takes self-binding and folding into account. The experimental results demonstrated the effectiveness of our approach, which outperforms the state-of-the-art architectures when a strict data set is employed.

Large repositories of PPIs mostly contain interacting proteins. Consequently, they are highly imbalanced, which needs to be addressed before learning. A new approach was proposed for determining non-interacting pairs. A graph was constructed in which the nodes refer to the proteins, and the edges correspond to their binding strength: the shorter the length of an edge, the stronger the bond. Non-interacting proteins correspond to the most distant nodes in terms of their geodesic distance in the graph. A large balanced

data set was thus created from the STRING database, called B-STRING. A new deep pyramidal network that learns both its architecture and depth from the data set was proposed. This allows the network to adapt to the structural complexity and size of the data set while preventing overfitting. This is of great importance as, for rare or new diseases, the size of the corresponding PPI data sets may be small, whereas the size of data sets for common and older diseases is much larger. The adaptive nature of our architecture also considerably reduces the number of parameters that must be determined by inspection, thereby streamlining the design of the network while removing arbitrariness and biases that result from human intervention. Given the high classification accuracy for both the benchmark data sets and the B-STRING, the network could be employed for screening a large number of interactions to provide researchers with the most promising candidates.

In addition to the above-mentioned approaches for PPI prediction, we also delivered a protein binder design approach. This research was conducted to address the challenge of discovering novel binders for target proteins, an essential task for therapeutic antibody and drug design. We introduced two transformer-based architectures for designing the corresponding binder from a given target protein. Since a target protein may interact with more than one protein, these novel architectures are designed to involve the binding score and a prior on the binder to obtain a constraint. Our work has wide applications in protein design, such as in targeting viruses that are prone to mutations. In such scenarios, if the unknown binder is a variant of the older one, the amino acid sequences of the two strains overlap strongly. As such, the concept of a prior may be fruitfully used to design a new therapeutic for emerging variants. Such an approach has obvious positive implications in terms of development speed and reduced financial costs.

Next, we detail our future research directions in Section 6.2.

6.2 Future Work

6.2.1 Protein-Protein Interaction Prediction with Multimodal Learning

Recall from Chapter 1 that, although amino acid sequences are the most widely available information about proteins, physicochemical properties, such as hydrophobicity, may further be inferred directly from the amino acid sequences. Thus, complementary information may be obtained by analyzing the various correlations between proteins. The utilization of such properties and evolutionary information may thus potentially lead to more informative PPI predictions. In addition, 3-D information about the macromolecular structures, available from the Protein Data Bank [241], has the potential to provide complementary insights into the structures and conformations of proteins. As noted in Chapter 1, only a limited number of curated 3-D protein structures may be obtained from the Protein Data Bank [241], whereas the B-STRING data set contains more than seventeen million protein pairs. This data mismatch hinders the wider applicability of this potential source of information.

Recently, accurate 3-D protein structure prediction, solely from the amino acid sequence, has been achieved with deep learning systems such as AlphaFold [242]. As a proposed future avenue of research, we will investigate the possibility of generating a large 3-D data set from B-STRING by employing AlphaFold. In addition, we may create a multimodal deep neural network for PPI prediction that employs physicochemical properties, evolutionary information, and 3-D structures.

6.2.2 GAN-based Protein Binder Design

Our protein design approach requires some prior knowledge about the ligand. With the aim of overcoming this restriction, it is proposed to generate binders with a generative adversarial network (GAN) [243] in which the generator would be a transformer and the discriminator would be a pyramid neural network [163]. The discriminator would determine if the target and binder proteins interact or not. In this extended framework, the

transformer would not require any prior information. Rather, all the binders would be generated directly by the GAN, thus overcoming the problems associated with multivalued solutions in addition to providing the most realistic solutions.

6.2.3 Large Amino Acid Chain Design

Most of our architectures involve attention mechanisms. These have quadratic complexity. Because of GPU memory requirements, our approach is limited to sequences not longer than 105 amino acids. This constitutes a problem for Homo sapiens proteins since the median length is 250 [135]. Therefore, we propose to replace our current attention mechanisms with the new Linformer [244], which has linear complexity. That would allow us to handle chains as long as 4096 with the same hardware.

6.2.4 Diffusion in Protein Design

Diffusion models were initially proposed by Sohl-Dickstein *et al.* [245] and further improved by Ho *et al.* [246]. The diffusion model consists of a forward process in which noise is progressively added to the data and a denoising (reverse) process which generated data from noise. The forward process has a closed-form solution while the reverse process must be learned with a neural network such as an equivariant graph neural network [247]. Diffusion models already outperform GAN for image generation [248]. Therefore, it is proposed to employ diffusion models for protein generation. We propose to use guided diffusion processes [248,249] in order to generate stable proteins with specific physicochemical properties such as aromaticity [250] and instability index [251].

References

[1] Daniel Wrapp, Nianshuang Wang, Kizzmekia S Corbett, Jory A Goldsmith, Ching-Lin Hsieh, Olubukola Abiona, Barney S Graham, and Jason S McLellan. Cryo-em structure of the 2019-ncov spike in the prefusion conformation. *Science*, 367(6483):1260–1263, 2020.

[2] Jiyao Wang, Philippe Youkharibache, Dachuan Zhang, Christopher J Lanczycki, Renata C Geer, Thomas Madej, Lon Phan, Minghong Ward, Shennan Lu, Gabriele H Marchler, et al. icn3d, a web-based 3d viewer for sharing 1d/2d/3d representations of biomolecular structures. *Bioinformatics*, 36(1):131–135, 2020.

[3] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

[4] Christopher M Bishop. Pattern recognition. *Machine learning*, 128(9), 2006.

[5] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in Neural Information Processing Systems*, 6, 1993.

[6] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[9] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[10] Yanzhi Guo, Lezheng Yu, Zhining Wen, and Menglong Li. Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. *Nucleic Acids Res.*, 36(9):3025–3030, 2008.

[11] Jeremy M Berg, John L Tymoczko, Lubert Stryer, et al. Protein structure and function. In *Biochemistry*, chapter 3. New York : W.H. Freeman, New York, 2002.

[12] Bruce Alberts. The cell as a collection of protein machines: preparing the next generation of molecular biologists. *Cell*, 92(3):291–294, 1998.

[13] WJ Cui, XJ Gong, H Yu, and XC Zhang. Mining topological structures of protein-protein interaction networks for human brain-specific genes. *Genet. Mol. Res*, 14:12437–12445, 2015.

[14] Yanghe Feng, Qi Wang, and Tengjiao Wang. Drug target protein-protein interaction networks: a systematic perspective. *BioMed Research International*, 2017, 2017.

[15] David E Gordon, Gwendolyn M Jang, Mehdi Bouhaddou, Jiewei Xu, Kirsten Obernier, Kris M White, Matthew J O'Meara, Veronica V Rezelj, Jeffrey Z Guo, Danielle L Swaney, et al. A sars-cov-2 protein interaction map reveals targets for drug repurposing. *Nature*, 583(7816):459–468, 2020.

[16] Heng Zhu, Metin Bilgin, Rhonda Bangham, David Hall, Antonio Casamayor, Paul Bertone, Ning Lan, Ronald Jansen, Scott Bidlingmaier, Thomas Houfek, et al. Global analysis of protein activities using proteome chips. *Science*, 293(5537):2101–2105, 2001.

[17] Johnathan Neiswinger, Ijeoma Uzoma, Eric Cox, HeeSool Rho, Guang Song, Corry Paul, Jun Seop Jeong, Kuan-Yi Lu, Chien-Sheng Chen, and Heng Zhu. Protein

microarrays: flexible tools for scientific innovation. *Cold Spring Harbor Protoc.*, 2016(10):pdb–top081471, 2016.

[18] Kevin Titeca, Irma Lemmens, Jan Tavernier, and Sven Eyckerman. Discovering cellular protein-protein interactions: Technological strategies and opportunities. *Mass Spectrom. Rev.*, 38(1):79–111, 2019.

[19] Ronald Jansen, Haiyuan Yu, Dov Greenbaum, Yuval Kluger, Nevan J Krogan, Sambath Chung, Andrew Emili, Michael Snyder, Jack F Greenblatt, and Mark Gerstein. A bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, 302(5644):449–453, 2003.

[20] Zhuhong You, Zhong Ming, Ben Niu, Suping Deng, and Zexuan Zhu. A SVM-based system for predicting protein-protein interactions using a novel representation of protein sequences. In *Int. Conf. Intell. Comput.*, pages 629–637, Berlin, Heidelberg, 2013.

[21] Juwen Shen, Jian Zhang, Xiaomin Luo, Weiliang Zhu, Kunqian Yu, Kaixian Chen, Yixue Li, and Hualiang Jiang. Predicting protein–protein interactions based only on sequences information. *Proc. Natl. Acad. Sci.*, 104(11):4337–4341, 2007.

[22] V Srinivasa Rao, K Srinivas, GN Sujini, and GN Kumar. Protein-protein interaction detection: methods and analysis. *Int. J. Proteomics*, 2014:1–13, 2014.

[23] Yanzhi Guo, Menglong Li, Xuemei Pu, Gongbin Li, Xuanmin Guang, Wenjia Xiong, and Juan Li. PRED_PPI: a server for predicting protein-protein interactions based on sequence data with probability assignment. *BMC Res. Notes*, 3(1):1–7, 2010.

[24] Shawn Martin, Diana Roe, and Jean-Loup Faulon. Predicting protein–protein interactions using signature products. *Bioinformatics*, 21(2):218–226, 2005.

[25] Tobias Hamp and Burkhard Rost. Evolutionary profiles improve protein–protein interaction prediction from sequence. *Bioinformatics*, 31(12):1945–1950, 2015.

[26] Oznur Tastan, Yanjun Qi, Jaime G Carbonell, and Judith Klein-Seetharaman. Prediction of interactions between HIV-1 and human proteins by information integration. In *Pac. Symp. Biocomputing*, pages 516–527. World Scientific, Hawaii, USA, 2009.

[27] Stefan R Maetschke, Martin Simonsen, Melissa J Davis, and Mark A Ragan. Gene ontology-driven inference of protein–protein interactions using inducers. *Bioinformatics*, 28(1):69–75, 2012.

[28] Lei Deng, Jihong Guan, Xiaoming Wei, Yuan Yi, Qiangfeng Cliff Zhang, and Shuigeng Zhou. Boosting prediction performance of protein–protein interaction hot spots by using structural neighborhood properties. *Journal of Computational Biology*, 20(11):878–891, 2013.

[29] Kuan-Hsi Chen, Tsai-Feng Wang, and Yuh-Jyh Hu. Protein-protein interaction prediction using a hybrid feature representation and a stacked generalization scheme. *BMC bioinformatics*, 20(1):308, 2019.

[30] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Comput. Vision Conf.*, pages 128–144, Las Vegas, USA, 2019.

[31] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.*, 13(3):55–75, 2018.

[32] Md Zahangir Alom, Tarek M Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal, and Vijayan K Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292–358, 2019.

[33] Nitin Kumar Chauhan and Krishna Singh. A review on conventional machine learning vs deep learning. In *Int. Conf. Comput. Power Commun. Technol.*, pages 347–352, Greater Noida, India, 2018.

[34] Tanlin Sun, Bo Zhou, Luhua Lai, and Jianfeng Pei. Sequence-based prediction of protein protein interaction using a deep-learning algorithm. *BMC Bioinf.*, 18(1):1–8, 2017.

[35] Xiao-Yong Pan, Ya-Nan Zhang, and Hong-Bin Shen. Large-scale prediction of human protein- protein interactions from amino acid sequence based on latent topic features. *J. Proteome Res.*, 9(10):4992–5001, 2010.

[36] Somaye Hashemifar, Behnam Neyshabur, Aly A Khan, and Jinbo Xu. Predicting protein–protein interactions through sequence-based deep learning. *Bioinformatics*, 34(17):i802–i810, 2018.

[37] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25(17):3389–3402, 1997.

[38] Hang Li, Xiu-Jun Gong, Hua Yu, and Chang Zhou. Deep neural network based predictions of protein interactions using primary sequences. *Molecules*, 23(8):1923–1939, 2018.

[39] Nariko Ikemura, Atsushi Hoshino, Yusuke Higuchi, Shunta Taminishi, Tohru Inaba, and Satoaki Matoba. Sars-cov-2 omicron variant escapes neutralization by vaccinated and convalescent sera and therapeutic monoclonal antibodies. *MedRxiv*, 2021.

[40] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[41] Wei Yang and Luhua Lai. Computational design of ligand-binding proteins. *Current Opinion in Structural Biology*, 45:67–73, 2017.

[42] Sravanth K Hindupur, Marco Colombi, Stephen R Fuhs, Matthias S Matter, Yakir Guri, Kevin Adam, Marion Cornu, Salvatore Piscuoglio, Charlotte KY Ng, Charles Betz, et al. The protein histidine phosphatase lhpp is a tumour suppressor. *Nature*, 555(7698):678–682, 2018.

[43] Simone Parn, Gabriel Jabbour, Vincent Nguyenkhoa, and Sivanesan Dakshanamurthy. Design of peptide vaccine for covid19: Cd8+ and cd4+ t cell epitopes from sars-cov-2 open-reading-frame protein variants. *bioRxiv*, 2021.

[44] Ruben Molina, Baldo Oliva, and Narcis Fernandez-Fuentes. A collection of designed peptides to target sars-cov-2–ace2 interaction: Pepi-covid19 database. *bioRxiv*, 2020.

[45] Yi Liu and Brian Kuhlman. Rosettadesign server for protein design. *Nucleic Acids Research*, 34:W235–W238, 2006.

[46] Jianyi Yang, Ivan Anishchenko, Hahnbeom Park, Zhenling Peng, Sergey Ovchinnikov, and David Baker. Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences*, 117(3):1496–1503, 2020.

[47] Yifei Qi and John ZH Zhang. Densecpd: improving the accuracy of neural-network-based computational protein sequence design with densenet. *Journal of chemical information and modeling*, 60(3):1245–1252, 2020.

[48] Christoffer Norn, Basile IM Wicky, David Juergens, Sirui Liu, David Kim, Doug Tischer, Brian Koepnick, Ivan Anishchenko, David Baker, and Sergey Ovchinnikov. Protein sequence design by conformational landscape optimization. *Proceedings of the National Academy of Sciences*, 118(11), 2021.

[49] Jingxue Wang, Huali Cao, John ZH Zhang, and Yifei Qi. Computational protein design with deep learning neural networks. *Scientific Reports*, 8(1):1–9, 2018.

[50] Zachary Wu, Kadina E Johnston, Frances H Arnold, and Kevin K Yang. Protein sequence design with deep generative models. *Current Opinion in Chemical Biology*, 65:18–27, 2021.

[51] Javier De Las Rivas and Celia Fontanillo. Protein–protein interactions essentials: key concepts to building and analyzing interactome networks. *PLoS Computational Biology*, 6(6):e1000807, 2010.

[52] Florian Richoux, Charlène Servantie, Cynthia Borès, and Stéphane Téletchéa. Comparing two deep learning sequence-based models for protein-protein interaction prediction. *arXiv preprint arXiv:1901.06268*, 2019.

[53] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Trans. Knowl. Discovery Data*, 6(4):1–21, 2012.

[54] Ioannis Xenarios, Lukasz Salwinski, Xiaoqun Joyce Duan, Patrick Higney, Sul-Min Kim, and David Eisenberg. DIP, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res.*, 30(1):303–305, 2002.

[55] Martin H Schaefer, Jean-Fred Fontaine, Arunachalam Vinayagam, Pablo Porras, Erich E Wanker, and Miguel A Andrade-Navarro. HIPPIE: Integrating protein interaction networks with experiment based quality scores. *Public Lib. Sci. ONE*, 7(2):1–8, 2012.

[56] Taibo Li, Rasmus Wernersson, Rasmus B Hansen, Heiko Horn, Johnathan Mercer, Greg Slodkowicz, Christopher T Workman, Olga Rigina, Kristoffer Rapacki, Hans H Stærfeldt, et al. A scored human protein–protein interaction network to catalyze genomic interpretation. *Nat. Methods*, 14(1):61–78, 2017.

[57] Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, et al. STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Res.*, 47(D1):D607–D613, 2019.

[58] Troy Hawkins and Daisuke Kihara. Function prediction of uncharacterized proteins. *Journal of Bioinformatics and Computational Biology*, 5(01):1–30, 2007.

[59] Troy Hawkins, Meghana Chitale, and Daisuke Kihara. New paradigm in protein function prediction for large scale omics analysis. *Molecular BioSystems*, 4(3):223–231, 2008.

[60] Ishita K Khan and Daisuke Kihara. Genome-scale prediction of moonlighting proteins using diverse protein association information. *Bioinformatics*, 32(15):2281–2288, 2016.

[61] Woong-Hee Shin, Charles W Christoffer, and Daisuke Kihara. In silico structure-based approaches to discover protein-protein interaction-targeting drugs. *Methods*, 131:22–32, 2017.

[62] Michael D Wendt. *Protein-Protein Interactions*. Springer Berlin Heidelberg, 2012.

[63] Gideon Schreiber and Sarel J Fleishman. Computational design of protein–protein interactions. *Current Opinion in Structural Biology*, 23(6):903–910, 2013.

[64] Laurence A Moran, H Robert Horton, K Gray Scrimgeour, Marc D Perry, and D Rawn. *Principles of biochemistry*. Pearson London:, 2012.

[65] Areski Flissi, Emma Ricart, Clémentine Campart, Mickael Chevalier, Yoann Dufresne, Juraj Michalik, Philippe Jacques, Christophe Flahaut, Frederique Lisacek, Valérie Leclère, et al. Norine: update of the nonribosomal peptide resource. *Nucleic Acids Research*, 48(D1):D465–D469, 2020.

[66] Takashi Kitano, Yu-Hua Liu, Shintaroh Ueda, and Naruya Saitou. Human-Specific Amino Acid Changes Found in 103 Protein-Coding Genes. *Molecular Biology and Evolution*, 21(5):936–944, 05 2004.

[67] Jonathan N Gonzalez-Flores, Sumangala P Shetty, Aditi Dubey, and Paul R Copeland. The molecular biology of selenocysteine. *Biomolecular Concepts*, 4(4):349–365, 2013.

[68] Keith Roberts, Bruce Alberts, Alexander Johnson, Peter Walter, and Tim Hunt. Molecular biology of the cell. *New York: Garland Science*, 32(2), 2002.

[69] Amit Kessel and Nir Ben-Tal. *Introduction to proteins: structure, function, and motion*. Chapman and Hall/CRC, 2018.

[70] K Murray, Victor Rodwell, David Bender, Kathleen M Botham, P Anthony Weil, and Peter J Kennelly. Harper's illustrated biochemistry. 28. *Citeseer, New York, United States*, 2009.

[71] Loic Giot, Joel S Bader, C Brouwer, Amitabha Chaudhuri, Bing Kuang, Y Li, YL Hao, CE Ooi, Brian Godwin, E Vitols, et al. A protein interaction map of drosophila melanogaster. *Science*, 302(5651):1727–1736, 2003.

[72] Siming Li, Christopher M Armstrong, Nicolas Bertin, Hui Ge, Stuart Milstein, Mike Boxem, Pierre-Olivier Vidalain, Jing-Dong J Han, Alban Chesneau, Tong Hao, et al. A map of the interactome network of the metazoan c. elegans. *Science*, 303(5657):540–543, 2004.

[73] Nevan J Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P Tikuisis, et al. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 440(7084):637–643, 2006.

[74] Anne-Claude Gavin, Patrick Aloy, Paola Grandi, Roland Krause, Markus Boesche, Martina Marzioch, Christina Rau, Lars Juhl Jensen, Sonja Bastuck, Birgit Dümpelfeld, et al. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631–636, 2006.

[75] Eric M Phizicky and Stanley Fields. Protein-protein interactions: methods for detection and analysis. *Microbiological reviews*, 59(1):94–123, 1995.

[76] Aidong Zhang. *Protein interaction networks: computational analysis*. Cambridge University Press, 2009.

[77] Kosuke Hashimoto, Hafumi Nishi, Stephen Bryant, and Anna R Panchenko. Caught in self-interaction: evolutionary and functional mechanisms of protein homooligomerization. *Physical biology*, 8(3):035007, 2011.

[78] Laura Martínez-Muñoz, Ricardo Villares, José Luis Rodríguez-Fernández, José Miguel Rodríguez-Frade, and Mario Mellado. Remodeling our concept of

chemokine receptor function: from monomers to oligomers. *Journal of leukocyte biology*, 104(2):323–331, 2018.

[79] Emanuel Margoliash. Primary structure and evolution of cytochrome c. *Proceedings of the National Academy of Sciences of the United States of America*, 50(4):672, 1963.

[80] MV Riley and MI Peters. The localization of the anion-sensitive atpase activity in corneal endothelium. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 644(2):251–256, 1981.

[81] Stanley Fields and Ok-kyu Song. A novel genetic system to detect protein–protein interactions. *Nature*, 340(6230):245–246, 1989.

[82] Seesandra V Rajagopala, Patricia Sikorski, J Harry Caufield, Andrey Tovchigrechko, and Peter Uetz. Studying protein complexes by the yeast two-hybrid system. *Methods*, 58(4):392–399, 2012.

[83] Guillaume Rigaut, Anna Shevchenko, Berthold Rutz, Matthias Wilm, Matthias Mann, and Bertrand Séraphin. A generic protein purification method for protein complex characterization and proteome exploration. *Nature Biotechnology*, 17(10):1030–1032, 1999.

[84] Benjamin A Shoemaker and Anna R Panchenko. Deciphering protein–protein interactions. part i. experimental techniques and databases. *PLoS computational biology*, 3(3):e42, 2007.

[85] Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[86] Ronald Jansen, Dov Greenbaum, and Mark Gerstein. Relating whole-genome expression data with protein-protein interactions. *Genome Research*, 12(1):37–46, 2002.

[87] Åsa K Björklund, Sara Light, Linnea Hedin, and Arne Elofsson. Quantitative assessment of the structural bias in protein–protein interaction assays. *Proteomics*, 8(22):4657–4667, 2008.

[88] David Sontag, Rohit Singh, and Bonnie Berger. Probabilistic modeling of systematic errors in two-hybrid experiments. In *Biocomputing 2007*, pages 445–457. World Scientific, 2007.

[89] Raghavendra Hosur, Jinbo Xu, Jadwiga Bienkowska, and Bonnie Berger. iwrap: an interface threading approach with application to prediction of cancer-related protein–protein interactions. *Journal of molecular biology*, 405(5):1295–1310, 2011.

[90] Joan Planas-Iglesias, Jaume Bonet, Javier García-García, Manuel A. Marín-López, Elisenda Feliu, and Baldo Oliva. Understanding protein–protein interactions using local structural features. *Journal of Molecular Biology*, 425(7):1210–1224, 2013.

[91] Bi-Qing Li, Kai-Yan Feng, Lei Chen, Tao Huang, and Yu-Dong Cai. Prediction of protein-protein interaction sites by random forest algorithm with mrmr and ifs. *PloS one*, 7(8), 2012.

[92] Yanjun Qi. Random forest for bioinformatics. In *Ensemble Machine Learning*, pages 307–323. Springer, 2012.

[93] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[94] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[95] Chia Hsin Liu, Ker-Chau Li, and Shinsheng Yuan. Human protein–protein interaction prediction by a novel sequence-based co-evolution method: co-evolutionary divergence. *Bioinformatics*, 29(1):92–98, 2013.

[96] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[97] Debasree Sarkar and Sudipto Saha. Machine-learning techniques for the prediction of protein–protein interactions. *Journal of Biosciences*, 44(4):1–12, 2019.

[98] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[99] Chang Zhou, Hua Yu, Yijie Ding, Fei Guo, and Xiu-Jun Gong. Multi-scale encoding of amino acid sequences for predicting protein interactions using gradient boosting decision tree. *PLoS One*, 12(8):e0181426, 2017.

[100] Fei Guo, Shuai Cheng Li, Lusheng Wang, and Daming Zhu. Protein-protein binding site identification by enumerating the configurations. *BMC bioinformatics*, 13(1):1–25, 2012.

[101] Long Lu, Hui Lu, and Jeffrey Skolnick. Multiprospector: an algorithm for the prediction of protein–protein interactions by multimeric threading. *Proteins: Structure, Function, and Bioinformatics*, 49(3):350–364, 2002.

[102] Javier Garcia-Garcia, Emre Guney, Ramon Aragues, Joan Planas-Iglesias, and Baldo Oliva. Biana: a software framework for compiling biological interactions and analyzing networks. *BMC Bioinformatics*, 11(1):1–12, 2010.

[103] Suraj Peri, J Daniel Navarro, Troels Z Kristiansen, Ramars Amanchy, Vineeth Surendranath, Babylakshmi Muthusamy, TKB Gandhi, KN Chandrika, Nandan Deshpande, Shubha Suresh, et al. Human protein reference database as a discovery resource for proteomics. *Nucleic Acids Res.*, 32:D497–D501, 2004.

[104] Andrew Chatr-Aryamontri, Arnaud Ceol, Luisa Montecchi Palazzi, Giuliano Nardelli, Maria Victoria Schneider, Luisa Castagnoli, and Gianni Cesareni. Mint: the molecular interaction database. *Nucleic Acids Research*, 35:D572–D574, 2007.

[105] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic Acids Research*, 34:D535–D539, 2006.

[106] Samuel Kerrien, Yasmin Alam-Faruque, Bruno Aranda, Iain Bancarz, Alan Bridge, Cathy Derow, Emily Dimmer, Marc Feuermann, Anja Friedrichsen, Rachael Huntley, et al. Intact—open source resource for molecular interaction data. *Nucleic Acids Research*, 35:D561–D565, 2007.

[107] Philipp Pagel, Stefan Kovac, Matthias Oesterheld, Barbara Brauner, Irmtraud Dunger-Kaltenbach, Goar Frishman, Corinna Montrone, Pekka Mark, Volker Stümpflen, Hans-Werner Mewes, et al. The mips mammalian protein–protein interaction database. *Bioinformatics*, 21(6):832–834, 2005.

[108] Christian B Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

[109] Joel R Bock and David A Gough. Predicting protein–protein interactions from primary structure. *Bioinformatics*, 17(5):455–460, 2001.

[110] Thomas Dandekar, Berend Snel, Martijn Huynen, and Peer Bork. Conservation of gene order: a fingerprint of proteins that physically interact. *Trends in Biochemical Sciences*, 23(9):324–328, 1998.

[111] Mamoru Yamada, Md Shahinur Kabir, and Ryouichi Tsunedomi. Divergent promoter organization may be a preferred structure for gene control in escherichia coli. *Journal of Molecular Microbiology and Biotechnology*, 6(3-4):206–210, 2003.

[112] Anna Panchenko and Teresa M Przytycka. *Protein-protein interactions and networks: identification, computer analysis, and prediction*, volume 9. Springer, 2010.

[113] Javad Zahiri, Joseph Hannon Bozorgmehr, and Ali Masoudi-Nejad. Computational prediction of protein–protein interaction networks: algorithms and resources. *Current Genomics*, 14(6):397–414, 2013.

[114] Vijaykumar Yogesh Muley and Akash Ranjan. Effect of reference genome selection on the performance of computational methods for genome-wide protein-protein interaction prediction. *PLOS ONE*, 7, 07 2012.

[115] Edward M Marcotte, Matteo Pellegrini, Ho-Leung Ng, Danny W Rice, Todd O Yeates, and David Eisenberg. Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285(5428):751–753, 1999.

[116] Matteo Pellegrini, Edward M Marcotte, Michael J Thompson, David Eisenberg, and Todd O Yeates. Assigning protein functions by comparative genome analysis: protein

phylogenetic profiles. *Proceedings of the National Academy of Sciences*, 96(8):4285–4288, 1999.

[117] K Srinivas, A Appa Rao, GR Sridhar, and S Gedela. Methodology for phylogenetic tree construction. *Journal of Proteomics & Bioinformatics*, 1:S005–S011, 2008.

[118] T Scott Chen, Hector Palacios, and Amy E Keating. Structure-based redesign of the binding specificity of anti-apoptotic bcl-xl. *Journal of Molecular Biology*, 425(1):171–185, 2013.

[119] Rhiju Das and David Baker. Macromolecular modeling with rosetta. *Annu. Rev. Biochem.*, 77:363–382, 2008.

[120] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[121] Mihai L Azoitei, Bruno E Correia, Yih-En Andrew Ban, Chris Carrico, Oleksandr Kalyuzhniy, Lei Chen, Alexandria Schroeter, Po-Ssu Huang, Jason S McLellan, Peter D Kwong, et al. Computation-guided backbone grafting of a discontinuous motif onto a protein scaffold. *Science*, 334(6054):373–376, 2011.

[122] Protein Data Bank. Protein data bank. *Nature New Biol*, 233:223, 1971.

[123] Sarel J Fleishman, Jacob E Corn, Eva-Maria Strauch, Timothy A Whitehead, John Karanicolas, and David Baker. Hotspot-centric de novo design of protein binders. *Journal of Molecular Biology*, 413(5):1047–1062, 2011.

[124] Jeffrey J Gray, Stewart Moughon, Chu Wang, Ora Schueler-Furman, Brian Kuhlman, Carol A Rohl, and David Baker. Protein–protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. *Journal of Molecular Bbiology*, 331(1):281–299, 2003.

[125] Ramesh K Jha, Andrew Leaver-Fay, Shuangye Yin, Yibing Wu, Glenn L Butterfoss, Thomas Szyperski, Nikolay V Dokholyan, and Brian Kuhlman. Computational design of a pak1 binding protein. *Journal of Molecular Biology*, 400(2):257–270, 2010.

[126] Bryan S Der, Mischa Machius, Michael J Miley, Jeffrey L Mills, Thomas Szyperski, and Brian Kuhlman. Metal-mediated affinity and orientation specificity in a computationally designed protein homodimer. *Journal of the American Chemical Society*, 134(1):375–385, 2012.

[127] Mohammad Jamshidi, Ali Lalbakhsh, Jakub Talla, Zdeněk Peroutka, Farimah Hadjilooei, Pedram Lalbakhsh, Morteza Jamshidi, Luigi La Spada, Mirhamed Mirmozafari, Mojgan Dehghani, et al. Artificial intelligence and covid-19: deep learning approaches for diagnosis and treatment. *IEEE Access*, 8:109581–109595, 2020.

[128] Abderrahim Mesbah, Aissam Berrahou, Hicham Hammouchi, Hassan Berbia, Hassan Qjidaa, and Mohamed Daoudi. Lip reading with hahn convolutional neural networks. *Image and Vision Computing*, 88:76–83, 2019.

[129] Jian Zhou, Huazhong Shu, Hongqing Zhu, Christine Toumoulin, and Limin Luo. Image analysis by discrete orthogonal hahn moments. In *International Conference Image Analysis and Recognition*, pages 524–531. Springer, 2005.

[130] Kaiyi Zhang and Nancy Samaan. Optimized look-ahead offloading decisions using deep reinforcement learning for battery constrained mobile iot devices. In *2020 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 181–186, 2020.

[131] William Sukaria, James Malasa, Shiu Kumar, Rahul Kumar, Mansour H. Assaf, Voicu Groza, Emil M. Petriu, and Sunil R. Das. Epileptic seizure detection using convolution neural networks. In *2022 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, pages 1–5, 2022.

[132] Zachary Wu, Kevin K Yang, Michael J Liszka, Alycia Lee, Alina Batzilla, David Wernick, David P Weiner, and Frances H Arnold. Signal peptides generated by attention-based neural networks. *ACS Synthetic Biology*, 9(8):2154–2161, 2020.

[133] Koichiro Saka, Taro Kakuzaki, Shoichi Metsugi, Daiki Kashiwagi, Kenji Yoshida, Manabu Wada, Hiroyuki Tsunoda, and Reiji Teramoto. Antibody design using lstm based deep generative model from phage display library for affinity maturation. *Scientific reports*, 11(1):1–13, 2021.

[134] Yue Kang, Dawei Leng, Jinjiang Guo, and Lurong Pan. Sequence-based deep learning antibody design for in silico antibody affinity maturation. *arXiv preprint arXiv:2103.03724*, 2021.

[135] Jung-Eun Shin, Adam J Riesselman, Aaron W Kollasch, Conor McMahon, Elana Simon, Chris Sander, Aashish Manglik, Andrew C Kruse, and Debora S Marks. Protein design and variant prediction using autoregressive generative models. *Nature Communications*, 12(1):1–11, 2021.

[136] Alex Hawkins-Hooker, Florence Depardieu, Sebastien Baur, Guillaume Couairon, Arthur Chen, and David Bikard. Generating functional protein variants with variational autoencoders. *PLoS Computational Biology*, 17(2):e1008736, 2021.

[137] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[138] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[139] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[140] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[141] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vision Pattern Recognit.*, pages 770–778, Las Vegas, NV, USA, 2016.

[142] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conf. Comput. Vision Pattern Recognit.*, pages 2261–2269, Honolulu, HI, USA, 2017.

[143] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[144] Paras Lakhani and Baskaran Sundaram. Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks. *Radiology*, 284(2):574–582, 2017.

[145] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8818–8826, 2019.

[146] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv:2005.10821*, 2020.

[147] O. Semih Kayhan and J. C. van Gemert. On translation invariance in CNNs: Convolutional layers can exploit absolute spatial location. In *IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, pages 14262–14273, Seattle, WA, USA, 2020.

[148] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[149] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[150] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[151] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image

caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057. PMLR, 2015.

[152] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 375–383, 2017.

[153] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[154] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. Sequential recommender system based on hierarchical attention network. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[155] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[156] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*, 2018.

[157] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(12), 2010.

[158] Yan-Bin Wang, Zhu-Hong You, Xiao Li, Tong-Hai Jiang, Xing Chen, Xi Zhou, and Lei Wang. Predicting protein–protein interactions from protein sequences by a stacked sparse autoencoder deep neural network. *Molecular BioSystems*, 13(7):1336–1344, 2017.

[159] Frederick Sanger, Steven Nicklen, and Alan R Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, 1977.

[160] UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515, 2019.

[161] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[162] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019.

[163] Junzheng Wu, Paquet Eric, Herna Viktor, and Wojtek Michalowski. Paying attention: Using a siamese pyramid network for the prediction of protein-protein interactions with folding and self-binding primary sequences. *The International Joint Conference on Neural Networks (IJCNN*, 2021.

[164] Mengying Zhang, Qiang Su, Yi Lu, Manman Zhao, and Bing Niu. Application of machine learning approaches for protein-protein interactions prediction. *Medicinal Chemistry*, 13(6):506–514, 2017.

[165] William R Pearson. An introduction to sequence similarity ("homology") searching. *Current Protocols in Bioinformatics*, 42(1):3–1, 2013.

[166] Tomas Mikolov, Kai Chen, Gregory S Corrado, and Jeffrey A Dean. Computing numeric representations of words in a high-dimensional space, May 19 2015. US Patent 9,037,464.

[167] Tobias Glasmachers. Limits of end-to-end learning. *arXiv preprint arXiv:1704.08305*, 2017.

[168] Jing-Dong J Han, Denis Dupuy, Nicolas Bertin, Michael E Cusick, and Marc Vidal. Effect of sampling on topology predictions of protein-protein interaction networks. *Nature Biotechnology*, 23(7):839–844, 2005.

[169] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[170] A Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017.

[171] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[172] Stephen Merity. Single headed attention rnn: Stop thinking with your head. *arXiv preprint arXiv:1911.11423*, 2019.

[173] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[174] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[175] Junzheng Wu, Paquet Eric, Herna Viktor, and Wojtek Michalowski. Adapting to complexity: Data generation and deep learnable architecture for protein-protein interaction predictions. In *8^{th} International Conference on Machine Learning, Optimization, and Data Science (LOD 2022)*. Springer, 2022. In Press.

[176] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE Int. Conf. Comput. Vision*, pages 2999–3007, Los Alamitos, CA, USA, 2017.

[177] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[178] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Adv. Neural Inf. Process. Syst.*, volume 33, pages 1877–1901, 2020.

[179] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115, 2021.

[180] Mohammad Mahdi Bejani and Mehdi Ghatee. A systematic review on overfitting control in shallow and deep neural networks. *Artif. Intell. Rev.*, pages 1–48, 2021.

[181] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vision Pattern Recognit.*, pages 248–255, Miami, FL, USA, 2009.

[182] Pawel Smialowski, Philipp Pagel, Philip Wong, Barbara Brauner, Irmtraud Dunger, Gisela Fobo, Goar Frishman, Corinna Montrone, Thomas Rattei, Dmitrij Frishman, et al. The Negatome database: a reference set of non-interacting protein pairs. *Nucleic Acids Res.*, 38:D540–D544, 2010.

[183] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Prog. Artif. Intell.*, 5(4):221–232, 2016.

[184] Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust statistics: the approach based on influence functions*. John Wiley & Sons, 2011.

[185] F. Meyer. *Topographical Tools for Filtering and Segmentation 1: Watersheds on Node- or Edge-weighted Graphs*. John Wiley & Sons, 2019.

[186] Jérémie Bouttier, Philippe Di Francesco, and Emmanuel Guitter. Geodesic distance in planar graphs. *Nucl. Phys. B*, 663(3):535–567, 2003.

[187] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

[188] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and Cooperation in Neural Nets*, pages 267–285, Berlin, Heidelberg, 1982.

[189] Yves Meyer. *Wavelets and Operators: Volume 1*. Cambridge University Press, 1992.

[190] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT press Cambridge, 2016.

[191] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating deep network training by reducing internal covariate shift. In *Int. Conf. Mach. Learn.*, volume 37, pages 448–456, Lille, France, 2015.

[192] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[193] Shide Liang and Chi Zhang. Prediction of immunogenicity for humanized and full human therapeutic antibodies. *Public Lib. Sci. ONE*, 15:1–14, 2020.

[194] Mark Chin, Claire Marks, and Charlotte M Deane. Humanization of antibodies using a machine learning approach on large-scale repertoire data. *bioRxiv*, 2021.

[195] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification and scene analysis*. Wiley New York, 1973.

[196] Nancy Chinchor and Beth Sundheim. MUC-5 evaluation metrics. In *Proc. 5th Conf. Message Understanding*, page 69–78, Baltimore, Maryland, 1993.

[197] Charles E. Metz. Basic principles of ROC analysis. *Semin. Nucl. Med.*, 8(4):283–298, 1978.

[198] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[199] Junzheng Wu, Paquet Eric, Herna Viktor, and Wojtek Michalowski. Primary sequence based protein-protein interaction binder generation with transformers. Submitted for publication, 2022.

[200] Mallela MG Krishna and S Walter Englander. The n-terminal to c-terminal motif in protein folding and function. *Proceedings of the National Academy of Sciences*, 102(4):1053–1058, 2005.

[201] Sarel J Fleishman, Timothy A Whitehead, Damian C Ekiert, Cyrille Dreyfus, Jacob E Corn, Eva-Maria Strauch, Ian A Wilson, and David Baker. Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. *Science*, 332(6031):816–821, 2011.

[202] Maren Butz, Peter Kast, and Donald Hilvert. Affinity maturation of a computationally designed binding protein affords a functional but disordered polypeptide. *Journal of Structural Biology*, 185(2):168–177, 2014.

[203] Erik Procko, Rickard Hedman, Keith Hamilton, Jayaraman Seetharaman, Sarel J Fleishman, Min Su, James Aramini, Gregory Kornhaber, John F Hunt, Liang Tong, et al. Computational design of a protein-based enzyme inhibitor. *Journal of Molecular Biology*, 425(18):3563–3575, 2013.

[204] Mickey Kosloff, Amanda M Travis, Dustin E Bosch, David P Siderovski, and Vadim Y Arshavsky. Integrating energy calculations with functional assays to decipher the specificity of g protein–rgs protein interactions. *Nature Structural & Molecular Biology*, 18(7):846–853, 2011.

[205] Mihai L Azoitei, Yih-En Andrew Ban, Jean-Philippe Julien, Steve Bryson, Alexandria Schroeter, Oleksandr Kalyuzhniy, Justin R Porter, Yumiko Adachi, David Baker, Emil F Pai, et al. Computational design of high-affinity epitope scaffolds by backbone grafting of a linear epitope. *Journal of Molecular Biology*, 415(1):175–192, 2012.

[206] Sen Liu, Shiyong Liu, Xiaolei Zhu, Huanhuan Liang, Aoneng Cao, Zhijie Chang, and Luhua Lai. Nonnatural protein–protein interaction-pair design by key residues grafting. *Proceedings of the National Academy of Sciences*, 104(13):5330–5335, 2007.

[207] Vladimir Potapov, D Reichmann, Renne Abramovich, D Filchtinski, Nir Zohar, D Ben Halevy, Marvin Edelman, Vladimir Sobolev, and Gideon Schreiber. Computational redesign of a protein–protein interface for high affinity and binding specificity using modular architecture and naturally occurring template fragments. *Journal of Molecular Biology*, 384(1):109–119, 2008.

[208] Zheng Huang, Yiwen Zhao, Xin Li, Xingang Zhao, Yunhui Liu, Guoli Song, and Yang Luo. Application of innovative image processing methods and adabound-se-densenet to optimize the diagnosis performance of meningiomas and gliomas. *Biomedical Signal Processing and Control*, 59:101926, 2020.

[209] Abdul Muntakim Rafi, Uday Kamal, Rakibul Hoque, Abid Abrar, Sowmitra Das, Robert Laganiere, Md Kamrul Hasan, et al. Application of densenet in camera model identification and post-processing detection. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 19–28, 2019.

[210] Najmul Hasan, Yukun Bao, Ashadullah Shawon, and Yanmei Huang. Densenet convolutional neural networks application for predicting covid-19 using ct image. *SN Computer Science*, 2(5):1–11, 2021.

[211] Sankaran Sandhya, Saane Sudha Rani, Barah Pankaj, Madabosse Kande Govind, Bernard Offmann, Narayanaswamy Srinivasan, and Ramanathan Sowdhamini. Length variations amongst protein domain superfamilies and consequences on structure and function. *PLoS One*, 4(3):e4981, 2009.

[212] Quan Le, Luis Miralles-Pechuán, Shridhar Kulkarni, Jing Su, and Oisín Boydell. An overview of deep learning in industry. *Data Analytics and AI*, pages 65–98, 2020.

[213] André Frenzel, Thomas Schirrmann, and Michael Hust. Phage display-derived human antibodies in clinical development and therapy. In *MAbs*, volume 8, pages 1177–1194. Taylor & Francis, 2016.

[214] Lars-Inge Larsson. *Immunocytochemistry: theory and practice*. CRC press, 2020.

[215] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[216] Yanxia Qin, Jingjing Ding, Yiping Sun, and Xiangwu Ding. A transformer-based model for low-resource event detection. In *International Conference on Neural Information Processing*, pages 452–463. Springer, 2021.

[217] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[218] Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, 2014.

[219] Zhongrun Xiang, Jun Yan, and Ibrahim Demir. A rainfall-runoff model with lstm-based sequence-to-sequence learning. *Water Resources Research*, 56(1):e2019WR025326, 2020.

[220] Yongchao Cui, Bo Yin, Ruixue Li, Zehua Du, and Mingquan Ding. Short-time series load forecasting by seq2seq-lstm model. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 9, pages 517–521, 2020.

[221] Yunjie Li, Mengtao Zhu, Yihao Ma, and Jian Yang. Work modes recognition and boundary identification of mfr pulse sequences with a hierarchical seq2seq lstm. *IET Radar, Sonar & Navigation*, 14(9):1343–1353, 2020.

[222] Mostafa Karimi, Di Wu, Zhangyang Wang, and Yang Shen. Deepaffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks. *Bioinformatics*, 35(18):3329–3338, 2019.

[223] Keisuke Kawano, Satoshi Koide, and Chie Imamura. Seq2seq fingerprint with byte-pair encoding for predicting changes in protein stability upon single point mutation. *IEEE/ACM transactions on computational biology and bioinformatics*, 17(5):1762–1772, 2019.

[224] Daniel S Berman, Craig Howser, Thomas Mehoke, and Jared D Evans. Mutagan: A seq2seq gan framework to predict mutations of evolving protein populations. *arXiv preprint arXiv:2008.11790*, 2020.

[225] Niko Moritz, Takaaki Hori, and Jonathan Le Roux. Capturing multi-resolution context by dilated self-attention. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5869–5873. IEEE, 2021.

[226] Christian Von Mering, Lars J Jensen, Berend Snel, Sean D Hooper, Markus Krupp, Mathilde Foglierini, Nelly Jouffre, Martijn A Huynen, and Peer Bork. String: known and predicted protein–protein associations, integrated and transferred across organisms. *Nucleic Acids Research*, 33:D433–D437, 2005.

[227] Gaston K Mazandu and Nicola J Mulder. Scoring protein relationships in functional interaction networks predicted from sequence data. *PLoS One*, 6(4):e18607, 2011.

[228] Linh Tran, Tobias Hamp, and Burkhard Rost. Profppidb: Pairs of physical protein-protein interactions predicted for entire proteomes. *Plos One*, 13(7):e0199988, 2018.

[229] Lyuba V Bozhilova, Alan V Whitmore, Jonny Wray, Gesine Reinert, and Charlotte M Deane. Measuring rank robustness in scored protein interaction networks. *BMC Bioinformatics*, 20(1):1–14, 2019.

[230] Ronald J. Williams and David Zipser. A learning algorithm for continually running rully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[231] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[232] Dieter Jungnickel. *The greedy algorithm*, pages 129–153. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[233] Bruce T Lowerre. *The harpy speech recognition system*. Carnegie Mellon University, 1976.

[234] Tingyu Xia, Yue Wang, Yuan Tian, and Yi Chang. Using prior knowledge to guide bert's attention in semantic textual matching tasks. In *Proceedings of the Web Conference 2021*, pages 2466–2475, 2021.

[235] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[236] Xuhua Xia and Zheng Xie. Protein structure, neighbor effect, and a new index of amino acid dissimilarities. *Molecular Biology and Evolution*, 19(1):58–67, 2002.

[237] Siyuan Liu, Xilun Xiang, Xiang Gao, and Haiguang Liu. Neighborhood preference of amino acids in protein structures and its applications in protein structure assessment. *Scientific Reports*, 10(1):1–11, 2020.

[238] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[239] Temple F Smith, Michael S Waterman, et al. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[240] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[241] Stephen K Burley, Charmi Bhikadiya, Chunxiao Bi, Sebastian Bittrich, Li Chen, Gregg V Crichlow, Cole H Christie, Kenneth Dalenberg, Luigi Di Costanzo, Jose M

Duarte, et al. Rcsb protein data bank: powerful new tools for exploring 3d structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. *Nucleic acids research*, 49(D1):D437–D451, 2021.

[242] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[243] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[244] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[245] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[246] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[247] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[248] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[249] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[250] JR Lobry and Christian Gautier. Hydrophobicity, expressivity and aromaticity are the major trends of amino-acid usage in 999 escherichia coli chromosome-encoded genes. *Nucleic acids research*, 22(15):3174–3180, 1994.

[251] Kunchur Guruprasad, BV Bhasker Reddy, and Madhusudan W Pandit. Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence. *Protein Engineering, Design and Selection*, 4(2):155–161, 1990.