Semestral work

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Deep learning for protein docking

Anatolii Filkin

Supervisor: Ing. Anton Bushuiev
January 2024

Acknowledgements

Declaration

Abstract

Protein docking, a cornerstone in modern biological and medical research, is pivotal for innovations ranging from vaccine development to stroke treatments. In this semester project, I focused on enhancing PPIformer, a cutting-edge machine learning model for designing protein-protein interactions[2]. Building on PPIformer, I adapted the model's architecture, loss functions, and training procedures for the specific task of protein-protein docking. I demonstrated that this problem can be effectively solved with fine-tuning of the PPIformer. The evaluation of the model's performance showed state-of-the-art on two standard metrics and second-best on the third one.

Keywords:

Supervisor: Ing. Anton Bushuiev

Abstrakt

Proteinové dokování, klíčový prvek v moderním biologickém a lékařském výzkumu, je klíčové pro inovace od vývoje vakcín po léčbu mrtvice. V tomto semestrálním projektu jsme se zaměřili na zlepšení PPIformer, špičkového strojového učícího modelu pro návrh proteinových interakcí[2]. Na základě PPIformeru jsme upravili architekturu modelu, ztrátové funkce a postupy tréninku pro konkrétní úkol proteinového dokování. Dokazuji, že tento problém lze efektivně řešit jemným laděním PPIformeru. Hodnocení výkonnosti modelu prokázalo jeho konkurenceschopnost ve srovnání s existujícími přístupy hlubokého učení a tradičními metodami dokování.

Klíčová slova:

Překlad názvu: Hluboké učení pro dokování proteinů

Contents

Figures

Tables

Chapter 1
Introduction

The COVID-19 pandemic, caused by the SARS-CoV-2 virus, has emerged as a global health crisis, highlighting the critical need for advanced understanding and manipulation of viral mechanisms. Central to the virus's ability to infect human cells is its spike protein, a structure that facilitates entry into host cells. This spike protein operates by binding to the ACE2 receptor in human cells, a process that has become a focal point for therapeutic interventions.[13] The intricate molecular dance between the virus's spike protein and the human cell receptor underscores the significance of protein-protein interactions (PPIs) in the biological world.

At the heart of understanding and potentially disrupting such interactions is the field of protein docking, which aims to predict how two proteins interact. In the quest to unravel these complex interactions, PPIformer[2] has emerged as a groundbreaking tool. Developed using advanced machine learning techniques, the PPIformer is created to predict and design protein-protein interactions with high accuracy. However, like any model in a rapidly evolving field, it requires continual refinement and adaptation to meet specific challenges, such as the docking scoring problem.

In this context, the primary aim of our work is to develop a robust method for solving the docking scoring problem, with a specific focus on predicting the FNAT (Fraction of Native Contacts) value that measures the quality of generated docking poses. This metric is crucial in assessing the accuracy of protein docking models. To validate our approach, I will employ the CAPRI (Critical Assessment of Predicted Interactions) dataset, a widely recognized benchmark in the protein docking community[6]. Our methodology draws inspiration from successful fine-tuning practices in other domains, such as computer vision and large language models, where such approaches have led to significant improvements in model performance.

By harnessing the power of PPIformer and fine-tuning it for our specific task, I anticipate significant advancements in our ability to accurately predict protein-protein interactions. Such progress not only has implications for combating diseases like COVID-19 but also extends to a broader range of biomedical applications, potentially leading to breakthroughs in drug design and disease treatment.

Chapter 2

Background

In this chapter, I focus on an introduction to the key concepts and terminology necessary for understanding this work. Let's start with the fundamentals of biochemistry and then continue with a straightforward overview of the core principles of modern deep learning.

2.1 Biochemistry

2.1.1 Proteins

Proteins, the fundamental building blocks of life, perform a myriad of functions within biological systems. Their roles range from catalysis of metabolic reactions to the structure of tissues, nerve transmission, muscle contraction, blood clotting, immunologic defenses, hormone function, and regulatory molecules. Understanding the intricate details of proteins' structure and function is crucial for grasping their role in biological processes and for advancements in fields like medicine and biochemistry.[14]

Structure of Proteins:

- Primary Structure: Proteins are synthesized as a sequence of amino acids, linked in a linear polyamide structure known as a polypeptide. Each amino acid consists of an alpha-carbon attached to an amino group, a carboxyl group, a hydrogen atom, and a unique side chain (R group). The sequence and number of these amino acids determine the protein's unique structure and function.

- Secondary Structure: The polypeptide chain folds into alpha-helices or beta-pleated sheets, forming the protein's secondary structure. This is influenced by hydrogen bond interactions between the protein backbone atoms.

- Tertiary and Quaternary Structures: Further folding leads to the tertiary structure, the three-dimensional shape of the protein. Some proteins also have a quaternary structure, formed by the assembly of multiple polypeptide subunits.[14]

PDB file

A PDB (Protein Data Bank) file is a standard file format used in structural biology and bioinformatics to store and exchange three-dimensional structural information of biological macromolecules, primarily proteins and nucleic acids. The PDB file format was developed to facilitate the sharing and dissemination of structural data among researchers and institutions.

2.1.2 Protein docking

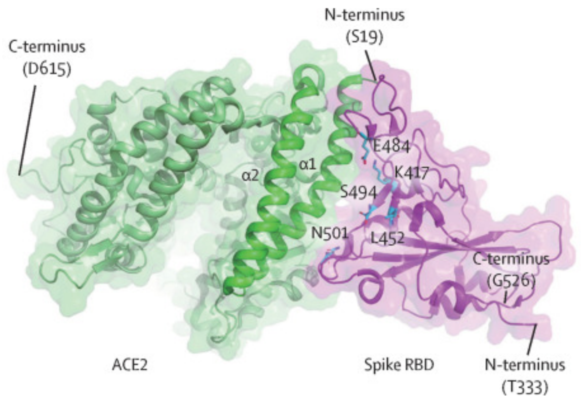


Figure 2.1: Protein-protein interaction between the ACE2 human receptor and SARS-CoV-2 spike.[12]

Protein docking is a computational method used to predict how two proteins interact with each other. Essentially, it involves simulating how proteins, which are complex molecules crucial to many biological functions, fit together like pieces of a puzzle. This process is vital for understanding various biochemical processes and for drug development. Protein docking uses algorithms to simulate and predict the orientation and position where two proteins bind together. This involves assessing shape complementarity and other interactions at the binding site.

In the context of SARS-CoV-2, the virus responsible for COVID-19, a significant example of protein docking involves the interaction between the virus's spike glycoprotein and the human angiotensin-converting enzyme 2 (ACE2) receptor on human cells as shown at 2.1. This intricate molecular binding process serves as a crucial step for the virus to enter host cells during its infection cycle.[13]

2.2 Machine Learning

2.2.1 Deep Learning

Deep feedforward networks, also known as feedforward neural networks or multilayer perceptrons (MLPs), are designed to approximate a function f^*.[5] These networks establish a mapping $y = f(x; \theta)$ and optimize the parameters

θ for the best function approximation. Mathematically, a feedforward network with n layers is expressed as:

$$f(x) = f^{(n)}\left(f^{(n-1)}\left(\dots f^{(1)}(x;\theta_1);\theta_{n-1}\right);\theta_n\right)$$

where $f^{(i)}$ represents the function computed by the i-th layer, and θ_i are the parameters (weights and biases) of that layer.

The hidden layers in a feedforward network provide outputs not directly observable in the training data. The output of the i-th hidden layer is given by:

$$h^{(i)} = \sigma\left(W^{(i)}h^{(i-1)} + b^{(i)}\right)$$

where $h^{(i)}$ is the output, $W^{(i)}$ and $b^{(i)}$ are the weights and biases, respectively, and σ is a nonlinear activation function, such as ReLU defined by $\sigma(x) = \max(0, x)$. The transformation in each layer, involving weights (W) and biases (b), is represented as:

$$f_i(x) = \sigma(W_i x + b_i)$$

The back-propagation algorithm computes gradients of the loss function L concerning the parameters θ. For a network with n layers, the gradient of L with respect to the weights in layer i is:

$$\frac{\partial L}{\partial W^{(i)}} = \frac{\partial L}{\partial h^{(i)}}\frac{\partial h^{(i)}}{\partial W^{(i)}}$$

where $\frac{\partial h^{(i)}}{\partial W^{(i)}}$ is the gradient of the layer output concerning its weights, and $\frac{\partial L}{\partial h^{(i)}}$ is the gradient of the loss for the layer output.

2.2.2 Graph Deep Learning

Graphs

Before we start with machine learning on graphs, we need to define graph data. In math, Graph $G = (V, E)$ can be defined as a pair of a set of nodes, called V, and a set of edges connecting these nodes called E. We define an edge that connects node $u \in V$ to node $v \in V$ as $(u, v) \in E$. Also, every node could have its features. We represent it using a real-valued matrix $X \in \mathbb{R}^{|V|\times m}$.[7]

Graph Neural Networks

Graph Neural Networks operate on graph-structured data. The goal of GNNs is to learn a function on this graph structure that can output a feature vector for each node, capturing both its features and the structure of its neighborhood. This is typically achieved through a neighborhood aggregation or message-passing scheme, where each node aggregates features from its neighbors, followed by a transformation. Mathematically, this can be expressed as:

$$h_v(k) = \mathrm{COMBINE}(k)\left(h_v(k-1), \mathrm{AGGREGATE}(k)\left(\{h_u(k-1) : u \in N(v)\}\right)\right)$$

where $h_v(k)$ is the feature vector of node v at the k-th iteration, $N(v)$ denotes the neighbors of v, and COMBINE and AGGREGATE are specific functions defining the GNN architecture.[1]

2.2.3 Fine-tuning

Fine-tuning in the context of a deep learning model refers to the process of adjusting the parameters of a pre-trained network to improve its performance on a specific task.[5] This is often done when the original task and the new task are similar but not identical. Mathematically, this involves modifying the weights θ of the network.

Assuming the pre-trained network has parameters θ_{pre}, fine-tuning adjusts these parameters to θ_{fine} to better fit the specific task. This is achieved through continued training, often with a smaller learning rate compared to the initial training. The process can be expressed as:

$$\theta_{\text{fine}} = \theta_{\text{pre}} - \eta \nabla_\theta L(\theta_{\text{pre}}, D_{\text{new}})$$

where η is the learning rate, $\nabla_\theta L$ is the gradient of the loss function L with respect to the parameters θ, and D_{new} is the new dataset specific to the task. The loss function L is typically a measure of the difference between the network's predictions and the actual outcomes in D_{new}.

Chapter 3

Problem Definition

3.1 Datasets

3.1.1 Training and validation data: BM5

For our study, I utilize the BM5 dataset, which is a collection of protein-protein interaction (PPI) structures. This dataset has been acquired through the data publication of the DeepRank[8] project. The BM5 dataset comprises a diverse array of protein complexes, each represented by PDB (Protein Data Bank) files. These files contain detailed information about the atomic structure of each protein complex, offering insights into the molecular composition and spatial arrangement of the interacting proteins.

Rationale for Dataset Selection

The BM5 dataset was selected as the training set for this study, primarily due to its prior use in the Deeprank-GNN[9] model, which serves as the baseline for comparison in our analysis. Employing BM5 holds strategic importance as it allows us to isolate the impact of model architecture and design improvements on the enhanced performance observed in the fine-tuned model. By using the same dataset as the baseline model, I can ensure that any observed improvements in the results are attributable to the modifications in the model itself, rather than variations in data quality or dataset characteristics. This approach provides a fair and direct comparison, ensuring that the enhancements in the model's performance are genuinely due to the architectural and design changes implemented during the fine-tuning process.

Structure

The Benchmark 5 (BM5) dataset, crafted for the purpose of docking studies, encompasses a non-redundant assembly of 231 complexes.[9] Each of these complexes provides insights into the structures of interacting proteins in both their bound and unbound forms. In a focused effort to hone the dataset for more specific docking analysis, 56 antibody-antigen complexes and others involving more than two protein chains were excluded. This refinement led to a concentration of 142 dimer complexes. Our setup contains 127 complexes as a train set and 15 others as validation.

DeepRank[8] employed the integrative modeling software HADDOCK[3] to generate a substantial number of models for each of these selected dimers. This endeavor resulted in about 25,300 models for each complex, cumulatively amounting to approximately 3.59 million models across the dataset.

Imbalance of Classes

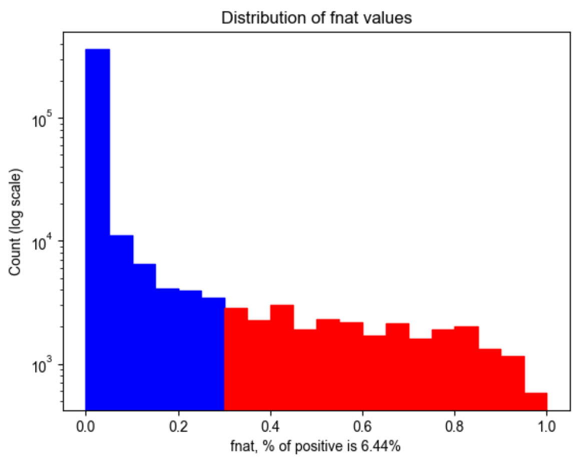


Figure 3.1: Fnat distribution in BM5 dataset.

As shown in Figure 3.1, the BM5 dataset presents a significant class imbalance with only 6.44% of the models classified as positive, or near-native. This small proportion of positive classes in the dataset underscores the rarity of biologically accurate docking configurations compared to the possible incorrect models. This imbalance is a critical factor in the dataset's complexity and poses unique challenges in the model training and evaluation process.

The distribution of classes within the BM5 dataset is not arbitrary but is designed to reflect the distribution of CAPRI[6] classes. It ensures that the distribution of model qualities in the dataset mirrors realistic scenarios encountered in protein docking research.

Redundancy of BM5 dataset

A notable characteristic of this dataset is the presence of a high degree of redundancy. It manifests as a large number of very similar conformations for each protein-protein interaction (PPI) present within the dataset. It poses a challenge for machine learning models, as it can lead to overfitting and reduced generalizability.

To address this issue, the DeepRank[8] project team selected a subset of the BM5 dataset. This subset was defined to reduce redundancy while still providing a comprehensive representation of the data. This meant selecting approximately 420,000 models from the dataset, equating to roughly 3,000 models per complex. This number was determined to be sufficient to accurately represent the diversity of the data, balancing the need for comprehensiveness with the desire to minimize redundancy.

3.1.2 Test data: CAPRI

The CAPRI (Critical Assessment of Predicted Interactions) dataset is a standard and extensively utilized resource in the field of protein-protein interaction (PPI) studies.[6] This dataset has been referenced and employed in various high-profile projects, including DeepRank, DeepRank-GNN, and the Piston study, underscoring its significance in advancing our understanding of protein docking.

Structure

The CAPRI dataset consists of 13 carefully selected protein complexes, with 17k models.

3.2 Interfaces

In the context of our work, the interface is the region or surfaces where two proteins come into contact, potentially interacting with each other. In this project, a residue is considered an interface residue if it is within a distance cutoff (10 Å by default, adjustable) of any heavy (non-hydrogen) atom on the other chain.

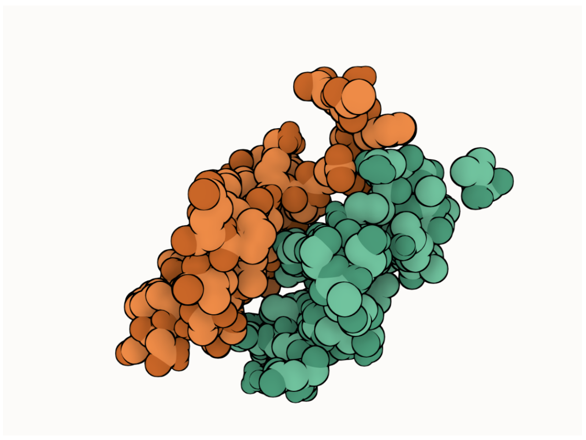


Figure 3.2: Sample of interface from BM5 dataset

3.3 Targets

3.3.1 Training and validation target: Fnat

The FNAT value is used as a target on the train and validation set. The fraction of native contacts (fnat) is defined as the fraction of reference interface contacts preserved in the interface of the docking model. Here the interface is identified by any pair of heavy atoms from two chains within a distance of 5Å. Specifically, fnat is a measure of how well the interface contacts of a docking model match those of a reference structure.

The formula for calculating fnat is as follows:

$$\text{fnat} = \frac{\text{Number of preserved native contacts}}{\text{Total number of reference interface contacts}} \tag{3.1}$$

3.3.2 Test target:CAPRI Class

The CAPRI Classin this paper is used as the test target . It is based on the interface RMSD (iRMSD) and categorized into different classes:

- Class 1: iRMSD $<$ 1Å
- Class 2: iRMSD $<$ 2Å
- Class 3: iRMSD $<$ 4Å
- Class 4: iRMSD $<$ 6Å

The iRMSD measures the RMSD between superimposed interface residues. It is a critical metric for evaluating the accuracy of protein-protein docking models. DockQ (Basu and Wallner, 2016) further provides a binary classification based on iRMSD, with class 0 for iRMSD $\geq$ 4Åand class 1 for iRMSD $<$ 4Å.

3.4 Metrics

To compare the results of the model Machine Learning I use the metrics:

3.4.1 Basic Metrics

Accuracy

Accuracy is the proportion of right-classified instances among the total predictions made by a model. It furnishes a holistic assessment of the model's correctness.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{False Positive} + \text{False Negative} + \text{True Positive} + \text{True Negative}} \tag{3.2}$$

Precision

Precision is a metric of positive predictions, that shows the ratio of true positives to the sum of all positives. It is important in maintaining a model's ability to minimize false positive occurrences.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3.3}$$

Recall

The recall is a metric of the true positive rate. It quantified the model's ability to accurately identify all positive instances. Recall is the ratio of true positives to the sum of true positives and false negatives.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.4}$$

F1 Score

The F1 Score is a balanced metric, serving as the harmonic mean of precision and recall. It offers an equilibrium measure of a model's effectiveness, especially valuable in scenarios characterized by class imbalance.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.5}$$

3.4.2 ROC AUC

The Receiver Operating Characteristic Area Under Curve (ROC AUC) metric is tailored for binary classification models. It quantifies a model's capability to discriminate between positive and negative classes by evaluating the area beneath the ROC curve. The ROC curve itself is constructed by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold values.

3.4.3 AUPRC

Area Under Precision-Recall Curve (AUPRC) is another metric designed for binary classification models. It scrutinizes the trade-off between precision and recall, measured by the area under the precision-recall curve. The precision-recall curve is constructed by plotting precision against recall at various threshold values.

3.4.4 Success Rate

The Success Rate[8] metric is employed to evaluate the model's proficiency in meeting pre-defined success criteria or thresholds. It finds utility in settings where binary outcomes hinge on specific criteria.

$$\text{Success Rate} = \frac{n_{\text{successful cases}}(K)}{N} \tag{3.6}$$

where:

- $n_{\text{successful cases}}(K)$ is the number of cases with at least one near-native model among the top K models.
- N is the total number of cases.

3.5 State of the art

3.5.1 DeepRank

DeepRankleverages 3D CNNs to assess protein-protein docking models[8]. The framework includes feature calculation, 3D grid mapping, and neural network training. It maps atomic and residue features onto a 3D grid using Gaussian functions and trains a customizable neural network on these grids. DeepRank demonstrates robust scoring, especially in rigid-body docking models, and shows less sensitivity to interface energetics compared to original HADDOCK scores.

3.5.2 DeepRank-GNN

DeepRank-GNN uses GNNs to analyze protein-protein interfaces, transforming PDB 3D coordinates into graphs[9]. This framework addresses the limitations of CNNs and handles diverse PPIs more naturally. DeepRank-GNN's architecture, including Graph Interaction Networks (GINet), extracts information from both interface and individual protein structures. It shows competitive performance in scoring docking models against traditional methods, confirmed by metrics like AUC, hit rate, and success rate.

3.5.3 PIsToN

PIsToN represents a novel approach in protein-protein docking interface evaluation, combining deep learning with transformer networks[15]. It transforms interfaces into 2D multi-channel images, processed through a Vision Transformer Network, focusing on spatial aspects and features. PIsToN merges empirical energy terms with surface features for enhanced prediction. It demonstrates superior classification and ranking capabilities, outperforming traditional scoring functions in various metrics.

3.5.4 Other baselines

- **ISCORE**: An energy-based scoring function that leverages graph kernels to evaluate docking models, emphasizing the structural properties and interactions within protein complexes.[4]

- **DOVE**: Utilizes deep learning with 3D convolutional neural networks to assess the quality of protein docking models, focusing on the intricate interaction patterns at protein interfaces.[10]
- **DOVE_GNN**: An advancement of DOVE that integrates Graph Neural Networks to enhance the model's ability to capture complex relational data between amino acids in protein complexes.[11]
- **HADDOCK**: An algorithm suite for integrative modeling that combines experimental data with computational predictions, offering a robust approach to constructing detailed models of protein assemblies.[3]

Chapter 4
Methods

4.1 Fine-tuning

The fine-tuning of the PPIformer model for docking scoring involves enhancing the model's ability to interpret and predict protein-protein interactions docking stability.[2][5]

4.1.1 PPIformer

At its core, PPiformer is a machine learning model that leverages the power of SE(3)-equivariant GNN architecture. This means that it can effectively handle and interpret the three-dimensional structures of protein complexes, respecting the spatial orientations and permutations inherent in these biological molecules. The model is trained to recognize and predict the likelihood of amino acids occurring at specific positions in a protein complex, considering their structural context. This is particularly important because the function and interaction of proteins are highly dependent on their 3D structure and the spatial arrangement of amino acids.

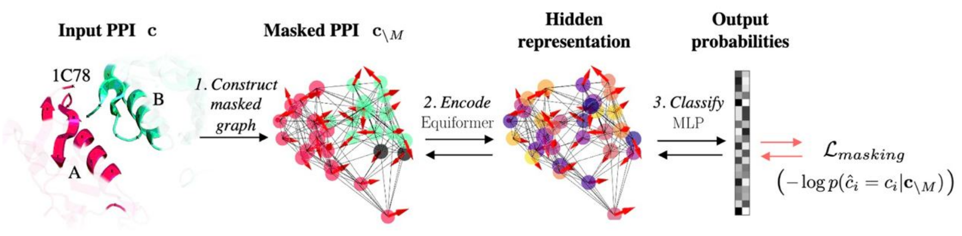


Figure 4.1: PPIformer achitecture[2]

Model description:

The core of PPIformer comprises Equiformer graph attention blocks, denoted as $f^{(l)}$. These blocks update the equivariant features $H_k^{(l)}$ of amino acids via message passing. The process is mathematically represented as:

$$H_0^{(l+1)}, H_1^{(l+1)}, \ldots, H_{\text{deg}}^{(l+1)} = f^{(l)}(G, X, E, H_0^{(l)}, H_1^{(l)}, \ldots, H_{\text{deg}}^{(l)}) \tag{4.1}$$

where $H_k^{(l)}$ are the node feature matrices of different equivariance types at layer l, G is the graph representing the protein complex, X contains the coordinates, and E represents edge features.

The encoder f is a composition of Equiformer blocks that process the input and output invariant amino acid embeddings H. The classifier g then predicts the probability matrix P based on these embeddings:

$$H := H_0^{(L-1)} \tag{4.2}$$

$$P = g(H) \tag{4.3}$$

where H represents the final invariant embeddings, and P is the probability matrix for amino acid types at each position.

The node features are represented as matrices X, F_0, and F_1, with X containing the coordinates of alpha-carbons and F_0 representing one-hot encodings of amino acids. F_1 are type-1 vectors representing virtual beta-carbon orientations:

$$X \in \mathbb{R}^{N\times 3}, \quad F_0 \in \mathbb{R}^{N\times 20\times 1}, \quad F_1 \in \mathbb{R}^{N\times 1\times 3} \tag{4.4}$$

The protein complex c is represented as a k-NN graph G with nodes as residues and edges based on proximity. The graph is enhanced with node-level and pairwise features X, E, F_0, F_1:

$$G = \text{k-NN}(X), \quad E \in \{0,1\}^{N\times N} \tag{4.5}$$

Equivariant feature processing ensures that geometric relationships are maintained:

$$f_{PPIformer}(G, X, E, F_0, F_1) = f(G, XR + 1t^T, E, F_0, F_1) \tag{4.6}$$

where $R \in SO(3)$ and $t \in \mathbb{R}^3$ are rotation and translation operations, respectively, ensuring SE(3)-equivariance.

To estimate the probabilities of masked amino acids, a 1-layer classifier with the softmax activation $g : \mathbb{R}^{N\times d_0\times 1} \to [0,1]^{N\times|A|}$ is applied on top of node embeddings H to obtain the probability matrix P.

4.1.2 PPIformer for Docking Scoring

In this project, to fine-tune PPIformer for docking scoring, two different approaches are utilized.

Embeddings solution

Instead of using the existing FC-layer after Equiformer, this solution aims to extract full information directly from embeddings. The mathematical representation of this process is as follows:

Let h represent the output from the forward pass of the PPIformer, without the existing logits layer:

$$h = f_{\text{PPIformer}}(\text{data}) \tag{4.7}$$

The pooling operation is applied to h:

$$h_{\text{pooled}} = \text{Pooling}(h, \text{data.batch}) \tag{4.8}$$

The regression is then performed using a multilayer perceptron (MLP):

$$h_{\text{regressed}} = \text{MLP}(h_{\text{pooled}}) \tag{4.9}$$

The MLP is defined with parameters as follows:

- Input channels: Embedding dimension of the PPIformer.
- Hidden channels: Defined by `regressor_hidden_channels`.
- Output channels: 1 (for regression).
- Number of layers: Defined by `regressor_num_layers`.

Logits solution

In this approach, the PPIformer's training pipeline is retained, and a new fully connected layer is added after the classifier layer. The classifier layer outputs a matrix of logits for each node in the interface. The aggregation of these logits is performed as follows:

Let h represent the output from the forward pass of the PPIformer:

$$h = f_{\text{PPIformer}}(\text{data}) \tag{4.10}$$

The wild-type probability predictions are obtained from h:

$$h_{\text{wt}} = h_i[\text{data.y}_i] \quad \forall i \in \{1, \ldots, n\} \tag{4.11}$$

where n is the number of nodes in the interface.

Let N be the number of nodes (amino acids) in a training graph representing a sampled protein-protein interaction. The following pooling operations are applied:

For mean pooling:

$$h_{\text{mean}} = \frac{1}{\text{N}} \sum_{i=1}^{N} h_{\text{wt},i} \tag{4.12}$$

For addition pooling:

$$h_{\text{add}} = \sum_{i=1}^{N} h_{\text{wt},i} \tag{4.13}$$

For max pooling:

$$h_{\max} = \max_{i \in \{1,\ldots,N\}} h_{\text{wt},i} \tag{4.14}$$

Those pooled values proceed with the regressor as above.

4.2 Losses

In machine learning, the concept of 'loss' quantifies how well a model's predictions match the actual target values. Different types of loss functions are suitable for various kinds of problems and models. Here, I discuss three suitable loss functions used for fine-tuning in this project: Mean Squared Error (MSE), Binary Cross Entropy (BCE), and Focal Loss.[5]

4.2.1 Mean Squared Error (MSE)

Mean Squared Error (MSE) is a widely used loss function, especially suitable for regression problems. It measures the average squared difference between the estimated values and the actual value. For a set of predictions $\hat{y}$ and true values y, the MSE is calculated as:

$$\text{MSE} = \frac{1}{m} \sum_{i} (\hat{y}_i - y_i)^2$$

Where m is the number of instances in the test set.

4.2.2 Weighted Mean Squared Error

The weighted Mean Squared Error is an extension of the standard MSE that incorporates class weights into the loss calculation. This modification is particularly useful for datasets with imbalanced classes. The mathematical formulation, adapted from the provided Python code, is:

$$MSE_{weighted} = \frac{1}{m} \sum_{i} w_i (\hat{y}_i - y_i)^2$$

Here, $\hat{y}$ and y are the predicted and actual values, respectively, m is the number of samples, and w_i represents the weight associated with each bin, derived from the target distribution in the dataset.

4.2.3 Binary Cross Entropy (BCE)

Binary Cross Entropy (BCE) is a loss function commonly used in binary classification tasks. It measures the performance of a classification model whose output is a probability value between 0 and 1. BCE loss increases as the predicted probability diverges from the actual label. It is defined as:

$$\mathrm{BCE}(p, y) = -\left[y \log(p) + (1 - y)\log(1 - p)\right]$$

Here, y is the true probability in the interval [0, 1] given by fnat, and p is the predicted probability of the class labeled as 1.

4.2.4 Focal Loss

Focal Loss is particularly designed for scenarios where there is a significant imbalance between classes, such as in one-stage object detection tasks.[16] It modifies the BCE loss by adding a factor that reduces the loss contribution from easy examples and focuses more on correcting misclassified examples. The Focal Loss is defined as:

$$\mathrm{FL}(p_t) = -\alpha_t(1 - p_t)^{\gamma}\log(p_t)$$

Where α_t is a weighting factor for balancing class 1 and 0, p_t is the model's estimated probability for the class with label $y = 1$, and γ is a focusing parameter that adjusts the rate at which easy examples are down-weighted.

4.2.5 Rationale Behind Loss Function Selection

Initially, I employed the Mean Squared Error (MSE) as our loss function for predicting FNAT in a regression framework. However, due to the imbalanced nature of our dataset, this approach led to a model bias, predominantly predicting the majority class (in this case, 0). This issue rendered MSE ineffective for our purposes.

To address this, I switched to Binary Cross Entropy (BCE) loss. BCE interprets FNAT as the probability of belonging to the positive or negative class, rather than a direct regression prediction. This shift in perspective yielded better results, as BCE is more adept at handling classification problems and provided a probabilistic understanding of FNAT.

Further experimentation led us to the Focal Loss. The motivation here was to focus the training more intensively on the worst-predicted classes. Focal Loss is designed to prioritize learning from challenging, misclassified examples, making it suitable for our scenario where class imbalance was a significant issue.

Lastly, I explored the Weighted Mean Squared Error , as described above. This variant of MSE, which accounts for class imbalances by applying distinct weights to different classes, demonstrated promising results, particularly in enhancing recall. By weighting the loss according to class distribution, weighted MSE effectively addressed the overfitting issue observed with standard MSE.

4.3 Optimizer

4.3.1 Adam

The Adam optimization algorithm, short for "Adaptive Moment Estimation," was introduced by Kingma and Ba in 2014. It is a versatile optimization method that combines elements from both the momentum and RMSProp algorithms, offering improved performance in training deep neural networks. [5]

Momentum

Momentum is a widely used optimization technique designed to expedite the convergence of gradient-based optimization. It proves particularly useful when dealing with challenges such as high curvature, small but consistent gradients, or noisy gradient estimates. The central idea behind momentum is to maintain an exponentially weighted moving average of past gradients, which guides the optimization process by providing direction and speed information.

The momentum algorithm updates the parameters θ as follows:

$$v \leftarrow \alpha v - \eta \nabla_\theta \mathcal{L}(f(x(i);\theta), y(i)) \tag{4.15}$$

$$\theta \leftarrow \theta + v \tag{4.16}$$

Here, α is a user-defined hyperparameter in the range $[0, 1)$ that controls the rate at which previous gradients influence the current direction. The learning rate is denoted by η.

Conceptually, We can liken the momentum algorithm to simulating the behavior of a particle subject to continuous-time Newtonian dynamics. The velocity vector $v(t)$ represents the particle's velocity at time t, and it experiences forces due to the cost function's gradient and a form of viscous drag, facilitating convergence to a local minimum.

RMSProp

RMSProp is an optimization method that builds upon AdaGrad but adjusts it to perform better in non-convex optimization scenarios. Unlike AdaGrad, which accumulates squared gradients without any forgetting mechanism, RMSProp employs an exponentially weighted moving average to adapt the learning rate.

The updated rules for RMSProp are as follows:

$$r \leftarrow \rho r + (1-\rho) g \odot g \tag{4.17}$$

$$\Delta\theta = -\frac{\eta}{\sqrt{r+\epsilon}} \odot g \tag{4.18}$$

$$\theta \leftarrow \theta + \Delta\theta \tag{4.19}$$

In these equations, ρ denotes a decay rate, r is an accumulation variable for squared gradients, ϵ is a small constant ensuring numerical stability, and g represents the gradient.

Adam Optimization

Adam, short for "Adaptive Moment Estimation," incorporates elements of both momentum and RMSProp, offering a robust and efficient optimization algorithm for neural network training.

The update rules for Adam are as follows:

$$m \leftarrow \beta_1 m + (1-\beta_1) g \tag{4.20}$$

$$v \leftarrow \beta_2 v + (1-\beta_2) g \odot g \tag{4.21}$$

$$\hat{m} = \frac{m}{1-\beta_1^t} \tag{4.22}$$

$$\hat{v} = \frac{v}{1-\beta_2^t} \tag{4.23}$$

$$\Delta\theta = -\frac{\eta}{\sqrt{\hat{v}+\epsilon}} \odot \hat{m} \tag{4.24}$$

$$\theta \leftarrow \theta + \Delta\theta \tag{4.25}$$

In the context of Adam, β_1 and β_2 represent exponential decay rates for the first and second moments, m and v are biased estimates of these moments, $\hat{m}$ and $\hat{v}$ are bias-corrected estimates, and t denotes the current iteration.

Adam's incorporation of momentum as an estimate of the first-order moment, along with its adaptive learning rate strategy, makes it a powerful optimization algorithm for deep learning tasks.

Chapter 5
Ablations

This chapter details a series of experiments conducted to optimize the performance of the PPIformer model in docking scoring[2]. These experiments were crucial in determining the most effective learning rate, loss functions, and other hyperparameters. A systematic approach was adopted, testing various configurations and analyzing their impact on the model's accuracy and efficiency. The findings from these experiments are crucial for fine-tuning the model to achieve optimal results in predicting protein-protein interactions.

5.1 Learning Rate

In my experiments, I evaluated the performance of the PPIformer model under different learning rates. The learning rate is a critical hyperparameter in training neural networks, as it determines the step size at each iteration while moving towards a minimum of the loss function. After extensive testing, I identified the most effective learning rate for our model, which significantly improved its performance. I tested it with different loss functions and got the optimal learning rate. As shown in the example for Focal Loss **??** It is set to 10^{-4}.

5.2 Losses

We experimented with various loss functions to understand their impact on the model's learning and predictive accuracy. Each loss function was analyzed based on its ability to handle the intricacies of protein docking scoring.

5.3 Conclusions

The experiments conducted provided invaluable insights into the optimal configuration of the PPIformer model for docking scoring. The optimal loss for this task is *weighted MSE*, and the optimal Learning Rate is 10^{-4}. The identification of the most effective learning rate, along with the analysis of various loss functions, has played a crucial role in enhancing the model's performance. The findings from these experiments will be instrumental in further refining and applying the model in real-world protein-protein interaction predictions.

Chapter 6
Implementation Details

6.1 Pytorch

The PPIformer model was implemented using PyTorch, a popular deep-learning library known for its flexibility, efficiency, and ease of use. PyTorch's dynamic computation graph (eager execution) and strong GPU acceleration capabilities make it an ideal choice for developing sophisticated models like PPIformer. To fine-tune the model, I used the same framework and implemented an additional class exactly for graph-level tasks - docking scoring.

6.1.1 Evaluation and Visualization

Post-training, I conducted a thorough evaluation of the model's performance using PyTorch's comprehensive metric system. This included accuracy, precision, recall, and area under the ROC curve (AUROC) analyses. Additionally, I implemented custom plotting functions using Plotly for detailed visualization of model predictions and performance metrics. These visualizations were crucial in interpreting the model's performance and understanding the nuances of protein-protein interactions.

6.1.2 Distributed Training and Experiment Tracking

Given the complexity and scale of the data, distributed training was employed to expedite the training process. PyTorch's distributed computing capabilities allowed us to train the model efficiently across multiple GPUs, significantly reducing the training time.

For experiment tracking and logging, I integrated Weights & Biases (wandb), a powerful tool for tracking and visualizing machine learning experiments. This integration facilitated real-time monitoring of training progress and comparison of different model versions and configurations.

6.2 Dataset Preparation

For this project, I undertook the task of preparing large datasets BM5 and CAPRI, comprising, an impressive 3.59 million PDB (Protein Data Bank) files.

This dataset size far exceeded those typically used in previous applications of PPIformer, posing unique challenges in terms of data handling and processing efficiency.

To address these challenges, I developed an advanced interface extraction method specifically tailored for handling such a large volume of PDB files. This method harnessed the power of concurrency and multi-processing techniques, which were instrumental in significantly accelerating the data extraction process. This method not only maximized CPU utilization but also drastically reduced the overall processing time, making it feasible to handle the dataset. By adopting these technique, I were able to maintain a high level of efficiency and speed, crucial for the successful application of PPIformer to such an extensive dataset..

Chapter 7
Results

In a comprehensive evaluation of performance metrics, our models, PPIformer and PPIformer_pretrained, have demonstrated good results, attaining state-of-the-art scores in ROC-AUC and ROC-PR metrics, and securing the second-best position in the Success Rate metric. These achievements underscore the robustness and precision of our models in predicting protein-protein interactions.

7.1 ROC-AUC

The ROC-AUC is a critical metric for assessing the performance of binary classification models. In this metric, the pre-trained and fine-tuned version of PPIformer has attained an AUC of 82.74, affirming the effectiveness of pretraining in enhancing model accuracy.

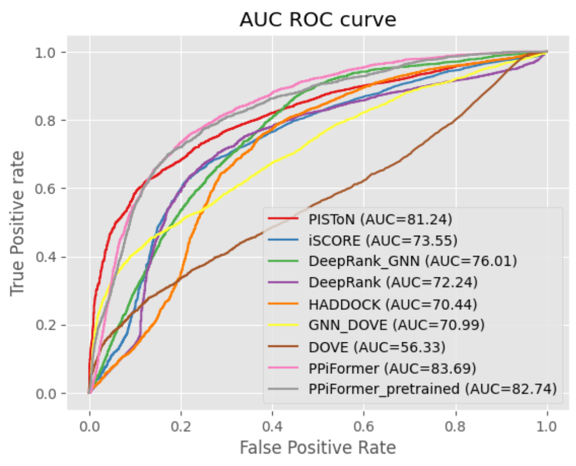


Figure 7.1: AUROC on CAPRI test set compared for different models.

7.2 AUPRC

The Precision-Recall curve (AUPRC) is another vital measure, especially for imbalanced datasets. Our PPIformer model excelled in this metric as well, with an Average Precision (AP) score of 38.28. The pre-trained and fine-tuned variant of PPIformer further improved the performance, achieving an AP score of 41.65. These results indicate that both models can identify positive samples with high precision amidst a large number of negative samples.

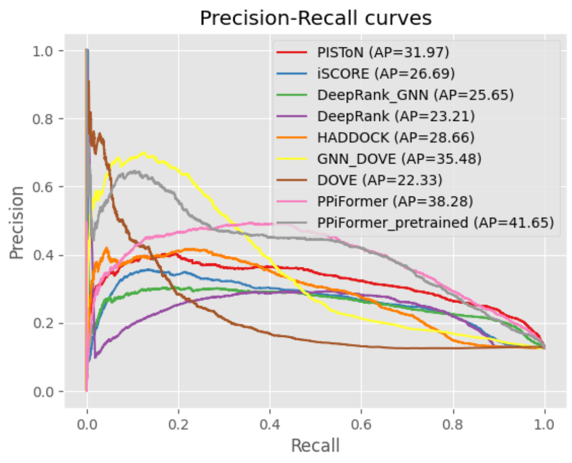
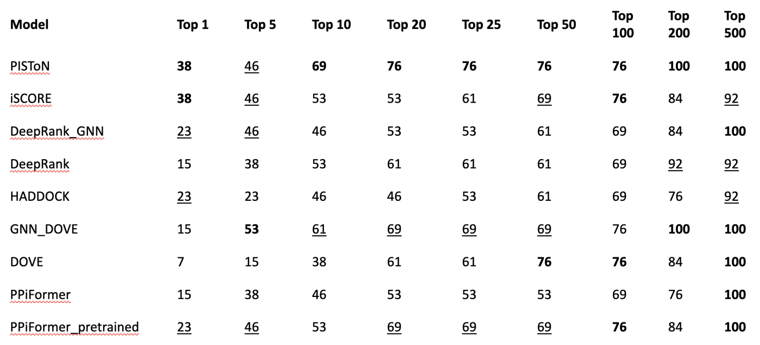


Figure 7.2: AUPRC on CAPRI test set compared for different models.

7.3 Success Rate

The Success Rate metric gauges the model's ability to rank true positive samples highly among a set of predictions. While PPIformer showed competitive results, the pre-trained version outperformed the randomly initialized version of the model across various thresholds. Notably, at the top 25, PPIformer scored 53, and PPIformer_pretrained scored 69, highlighting the significant improvement brought about by pretraining.



Model	Top 1	Top 5	Top 10	Top 20	Top 25	Top 50	Top 100	Top 200	Top 500
PISToN	**38**	46	**69**	**76**	**76**	**76**	**76**	**100**	**100**
iSCORE	**38**	46	53	53	61	69	**76**	84	92
DeepRank_GNN	23	46	46	53	53	61	69	84	**100**
DeepRank	15	38	53	61	61	61	69	92	92
HADDOCK	23	23	46	46	53	61	69	76	92
GNN_DOVE	15	**53**	61	69	69	69	76	**100**	**100**
DOVE	7	15	38	61	61	**76**	**76**	84	**100**
PPiFormer	15	38	46	53	53	53	69	76	**100**
PPiFormer_pretrained	23	46	53	69	69	69	**76**	84	**100**

Figure 7.3: Success rate on CAPRI test set compared for different models.

Chapter 8

Discussion

In this project, I have fine-tuned the PPIformer model to score the docking poses of protein-protein interactions. I have achieved state-of-the-art performance on two widely-used evaluation metrics and second-best results on the third metric. This lays a promising foundation for the future work.

8.1 Futher Steps

Building upon the successful foundation laid by the current research, the following strategies are proposed to advance the PPIformer models:

1. **Hyper-parameters:** Fine-tuning the hyper-parameters could lead to improved model performance. This includes adjusting learning rates, regularization terms, and the number of layers or hidden units in the neural network. A systematic grid search random search can be employed to identify the most effective combinations.

2. **Incorporation of ISCORE Energy Solution:** As detailed in the knowledge source, the iScore energy-based scoring system could be integrated into the training process.[4] This system, which accounts for various interaction forces between proteins, may provide additional features that could refine the model's predictive power. Understanding and utilizing these energy calculations could lead to a more nuanced and accurate model. This approach was shown to significantly boost the performance of the iScore model and I believe that incorporating the energy terms may improve the performance of PPIformer on the success rate metric.

3. **Training on Alternative Datasets:** To assess the generalizability and robustness of the PPIformer models, training them on other datasets, such as MaSIF, could be insightful[17]. MaSIF offers a unique set of protein-protein interaction data characterized by its surface-based representation of proteins. This could challenge and potentially enhance the model's capability to learn from diverse data representations. Training on the MaSIF dataset may be the crucial factor for the high success rate performance of the PiSToN model.[15]

These steps are designed to extend the frontier of research in the field of protein-protein interaction prediction. By iteratively refining the PPIformer models through these suggested strategies, there is a significant potential to achieve unprecedented accuracy and reliability in computational protein docking and scoring.[2]

Bibliography

[1] Bronstein, M. M., Bruna, J., Cohen, T., Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. arXiv preprint arXiv:2104.13478 (2021).

[2] Bushuiev, Anton, Bushuiev, Roman, Filkin, Anatolii, Kouba, Petr, Gabrielova, Marketa, Gabriel, Michal, Sedlar, Jiri, Pluskal, Tomas, Damborsky, Jiri, Mazurenko, Stanislav, et al. Learning to design protein-protein interactions with enhanced generalization. arXiv preprint arXiv:2310.18515 (2023).

[3] Dominguez, C., Boelens, R., Bonvin, A. M. J. HADDOCK: a protein-protein docking approach based on biochemical or biophysical information. Journal of the American Chemical Society 125(7), 1731–1737 (2003).

[4] Geng, C., Jung, Y., Renaud, N., Honavar, V., Bonvin, A. M., Xue, L. C. iScore: a novel graph kernel-based function for scoring protein–protein docking models. Bioinformatics 36(1), 112–121 (2020).

[5] Goodfellow, I., Bengio, Y., Courville, A. Deep learning. MIT press (2016).

[6] Janin, J., Henrick, K., Moult, J., Eyck, L. T., Sternberg, M. J., Vajda, S., Vakser, I., Wodak, S. J. CAPRI: a critical assessment of predicted interactions. Proteins: Structure, Function, and Bioinformatics 52(1), 2–9 (2003).

[7] Hamilton, W. L. Graph Representation Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. 14, No. 3, Pages 1-159 (2020).

[8] Renaud, N., Geng, C., Georgievska, S., et al. DeepRank: a deep learning framework for data mining 3D protein-protein interfaces. Nat Commun 12, 7068 (2021). `https://doi.org/10.1038/s41467-021-27396-0`

[9] Réau, M., Renaud, N., Xue, L. C., Bonvin, A. M. J. J. DeepRank-GNN: a graph neural network framework to learn patterns in protein–protein interfaces. Bioinformatics, Volume 39, Issue 1, January 2023, btac759. `https://doi.org/10.1093/bioinformatics/btac759`

[10] Wang, X., Terashi, G., Christoffer, C. W., Zhu, M., Kihara, D. Protein docking model evaluation by 3D deep convolutional neural networks. Bioinformatics 36(7), 2113–2118 (2020).

[11] Wang, X., Flannery, S. T., Kihara, D. Protein Docking Model Evaluation by Graph Neural Networks. Front. Mol. Biosci. 8:647915 (2021). `https://doi.org/10.3389/fmolb.2021.647915`

[12] Zhang, H., Lv, P., Jiang, J., Liu, Y., Yan, R., Shu, S., Hu, B., Xiao, H., Cai, K., Yuan, S., Li, Y. Advances in developing ACE2 derivatives against SARS-CoV-2. The Lancet Microbe, Volume 4, Issue 5, 2023, Pages e369-e378. `https://doi.org/10.1016/S2666-5247(23)00011-3`

[13] Mehdipour, Ahmad Reza, and Hummer, Gerhard. Dual nature of human ACE2 glycosylation in binding to SARS-CoV-2 spike. Proceedings of the National Academy of Sciences, Volume 118, Number 19, Pages e2100425118, 2021, National Acad Sciences.

[14] Alberts, B., Bray, D., Hopkin, K., Johnson, A. D., Lewis, J., Raff, M., Roberts, K., Walter, P. Essential Cell Biology. Garland Science, 2015.

[15] Stebliankin, Vitalii, Shirali, Azam, Baral, Prabin, Chapagain, Prem, Narasimhan, Giri. PIsToN: Evaluating Protein Binding Interfaces with Transformer Networks. bioRxiv, Pages 2023-01, 2023, Cold Spring Harbor Laboratory.

[16] Lin, Tsung-Yi, Goyal, Priya, Girshick, Ross, He, Kaiming, Dollár, Piotr. Focal loss for dense object detection. Proceedings of the IEEE International Conference on Computer Vision, Pages 2980-2988, 2017.

[17] Gainza, Pablo, Sverrisson, Freyr, Monti, Frederico, Rodola, Emanuele, Boscaini, D, Bronstein, MM, Correia, BE. Deciphering Interaction Fingerprints from Protein Molecular Surfaces using Geometric Deep Learning. Nature Methods, Volume 17, Number 2, Pages 184-192, 2020, Nature Publishing Group US New York.