Graph Neural Networks for Protein Interface Prediction

Chengkai Xiao
School of Computing and Communications
Lancaster University

September, 2022

Abstract

Computational modelling of biomolecular structures is of great importance in biology and medicine. Proteins are large biomolecules with complex structures that are essential to the function of life. Advancements in the understanding of protein-protein interaction often leads to new treatments for difficult conditions, especially in Oncology and Immunology. Accurately predicting the interface contacts between proteins can improve the performance of other bioinformatics tools and pipelines. Recent developments in data mining on the Protein Data Bank and graph neural networks have opened possibilities in deep learning on protein graphs. In this project, we tackle the task of modelling partner-specific protein-protein residue contacts given the 3D tertiary structures of two proteins as input. We represent proteins as complete spatial graphs and use Transformer-based GNN with learnable structural and positional representations to extract node features for downstream tasks. We improve[1] upon the state-of-the-art graph neural network performance for protein interface contact prediction on three different challenging benchmarks: DIPS-Plus, CASP-CAPRI 13 & 14 and Docking Benchmark 5. Our model achieves more than twice the current top-k precision and recall on CASP-CAPRI 13 & 14. Our work contributes to validating the effectiveness of modelling proteins as graphs and Transformer-based graph neural networks.

[1]Code and model checkpoint available at `https://github.com/ChengKaiXiao/DeepInteract`.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Protein Interface Prediction

Proteins are the essential building blocks of life, there are just over 20,000 proteins in humans and over 200 million known proteins in all life. Proteins are made from a long chain of organic compounds called amino acids (i.e. residues). Under normal conditions in the body, each protein chain folds into a tertiary structure that gives rise to the protein's function. Determining the structure of proteins by experimentation is a very costly and lengthy process, this drove research interests in predicting the structure computationally. Protein folding is one of the greatest challenges in biology. For decades, there has been little progress made in this prediction problem. In 2021, the deep learning system AlphaFold2 (AF2) (Jumper et al., 2021) made huge breakthroughs in making very accurate structural predictions. The protein folding problem is now considered solved as the errors in the prediction from AF2 are on the same level as experimental errors. AF2 proved once again that deep learning is capable of solving complex prediction problems with the following characteristics: large combinatorial search space of solutions, clear objective function to optimise against and large amounts of data. AF2 models proteins as **spatial graphs**, where the nodes are residues and edges are connections between residues (DeepMind, 2020). The Evoformer is the core component in AF2, it is a Transformer-based neural network. Evoformer refines this graph representation of the protein at each step and the final graph is used to produce structural predictions. The insights and methods developed in AF2 open the door to tackling many problems in protein science with deep learning, including protein-protein interaction.

Protein-protein interactions (PPI) are some of the most important processes in biology, PPI plays a fundamental role in the immune system, cellular functions and dysfunctions (i.e. cancers). Proteins can bind together to form a protein complex (i.e. quaternary structure, dimer). One of the major difficulties in modelling PPI is the

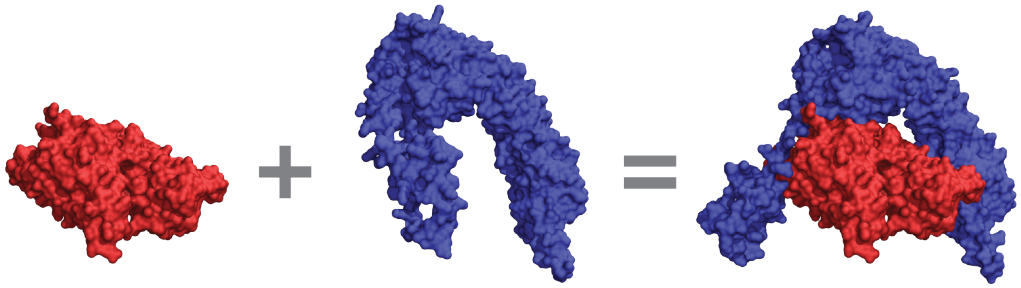


Figure 1.1: Protein Binding. The BNI1 protein (blue) opens up to bind to actin (red). (Townshend et al., 2018)

conformational changes during protein binding, see Figure 1.1. The proteins involved in binding react to each other, and their individual structures can twist and change in order to form a stable complex. The study of PPI can be framed in several ways, with varying levels of resolution, difficulty and applications. PPI can be modelled on the atomic level or the residual level, the former is a higher dimensional problem and so it is more challenging and computationally demanding. Protein-protein docking is the prediction of the structure of the complex, given the structures of the individual proteins (Vakser, 2014). Protein-protein docking commonly models proteins on the atomic level and recent efforts aims to account for the flexible protein backbone. Recent developments in geometric deep learning and graph neural networks have enabled methods for *rigid* protein docking (Ganea et al., 2021), EquiDock predict the structure of the bound protein complex from protein graph inputs. However, modelling conformational change explicitly is still out of the reach of deep learning. There are also studies on binding affinity, the strength of binding measured by energy between proteins and between proteins and smaller molecules (i.e. ligand). Binding affinity prediction is useful for studying drug target interaction in order to understand drugs' mechanism of action.

As we can see, PPI is an immensely complex problem and many ways to model. In this report, we narrow our scope and focus on the framing of PPI as **protein interface prediction** on the residual level. In other words, given two proteins, we are trying to predict which pairs of residues are in contact when their proteins bind together. This is a multi-label classification problem in essence. In terms of resolution, this task sits between protein docking and binding energy prediction. Previous machine learning methods on this problem have used techniques such as

SVMs, boosted trees with hand-crafted features. Recent works including AF2 have demonstrated the promises of modelling proteins as graphs with an end-to-end neural network. Starting with the structures of two proteins, we need to preprocess these into their respective graph representations. Then use GNNs to extract graph features used to generated a prediction. Each step of modelling is very involved, therefore we must rely on existing pipelines for much of our project. The main goal of this project is to explore GNNs which may be able to improve the performance of existing pipelines.

1.2 Project Aims and objectives

Etcembly Ltd is a biotechnology research company that specialises in immunotherapies and personalised medicines. Etcembly have assembled a large health and disease immune database and by leveraging machine learning, the company is looking to develop insights into T Cell Receptors (TCRs) therapeutics. Etcembly has developed a proprietary machine learning platform named EMLy™. The platform is trained to identify the highest potential TCR candidates that have the highest affinity and lowest probability of cross-reactivity, off-target or toxicity events.

Recent GNNs were able to learn and predict properties of chemical molecular graphs used for drug discovery such as antibacterial effect and toxicity. GNNs are already making an impact, a message-passing type GNN discovered the antibiotic halicin (Stokes et al., 2020). This is highly significant as new antibiotics costs billions and 10-15 years to develop. Halicin was data-mined from known chemicals and repurposed as an antibiotic, more ambitious research in generative GNNs may lead to completely new therapeutic compounds and proteins. Etcembly is interested in graph neural networks (GNNs) to help develop biological agents that can be used to develop medicines for Oncology and Infectious Disease. Etcembly is working on modelling of multichain protein complexes, this project is based on modelling simpler biomolecules. It is hoped that insights from this project will be used to drive Etcembly's discovery process. The aim of the project is to explore existing novel architectures of graph neural networks for modelling biochemical compounds. The project concerns the evaluation of the existing techniques, but can involve research into improving machine learning techniques subject to the time available for the project.

Summary of project goals of GNNs for protein interface prediction:

- Selection of baseline models/pipelines to be improved upon.
- Evaluation of existing models.
- Investigate newly published models.

1.3 Report Summary

The introduction have highlighted the importance of understanding protein-protein interactions and the potentials of modelling biomolecular graphs with GNNs to develop medicines. We step away from PPI in the background chapter and introduce required material to understand methods of machine learning with graphs and GNNs. The next chapter comes back to the main topic, defining the protein interface prediction problem, examining relevant protein structure datasets and reviewing related machine learning methods. We propose a method for protein interface prediction based on the state-of-the-art pipeline DeepInteract. We use a simpler edge representation for protein graphs and propose a modified Graphormer model. Graphormer is a Transformer based GNN designed for atomic graphs of chemicals. We show that the replacement of Geometric Transformer architecture with Graphormer helps improve the performance on two widely known datasets for protein-protein interaction. Graphormer also improves the transfer learning performance on a third dataset.

Chapter 2

Background

This chapter covers the background required to understand graph neural networks: machine learning, deep learning, modelling relational systems as graphs. We introduce models and methods specific to this project such as the Transformer, Transformer-based GNNs, Graphormer and transfer learning.

2.1 Supervised Learning

Supervised learning is machine learning on annotated data (Müller and Guido, 2016). Suppose we have training data X and labels Y. We want our model to learn a function to learn a mapping between X and Y. This usually involves minimising a cost function of predicted values $\hat{Y}$ and Y. Our goal is to produce good predictions on new or unseen data. This means that when given unseen data point x, our model produces prediction $\hat{y}$ that is close to the true label y. To put this into context, we start with the raw structural data of proteins and protein complexes and create our data and labels for supervised learning. Based on the protein residue contact model of PPI, we need to preprocess these into graph representations of proteins and interaction labels accordingly. We expand on protein graph preprocessing in Chapter 3 and provide the exact problem formulation in Chapter 4. Next, we introduce deep learning which has been very successful on complex supervised learning tasks.

2.2 Deep Learning

Artificial neural networks have been around for a surprisingly long time, a single layered neural network called the Perceptron (Rosenblatt, 1958) have been proposed in the 1950s. These early neural networks were very limited in their capacity to learn. Subsequent development of the backpropagation algorithm (Rumelhart,

Hinton, and Williams, 1986) and multilayer perceptron (feedforward neural networks) saw moderate successes. The increase in the availability of data and computation drove architectural developments, the most prominent examples are convolutional neural networks (CNN) for vision and recurrent neural networks (RNN) for language and speech.

2.2.1 Artificial Neurons

Artificial neurons (i.e. neurons, units, nodes or weights) as loosely inspired by biological neurons (I. Goodfellow, Bengio, and Courville, 2016). Each unit is a function defined by their weights $\boldsymbol{w}$, bias b and activation function g. Let $\boldsymbol{x} \in \mathbb{R}^n$ be the input, we have the output $h_i = g(\boldsymbol{w}_i^T \boldsymbol{x} + c_i)$. The layers of a MLP and filters in CNN are composed of such units. The rectified linear unit (ReLU) (Nair and Hinton, 2010) is the most widely used activation function, $f(x) = \max(0, x)$. ReLU solves the vanishing gradient problem, allowing the training of larger deeper networks. The computation is also cheaper compared to the classical sigmoid activation function. The Gaussian error linear unit (GELU) (Hendrycks and Gimpel, 2016) is a smoothed version of ReLU, for input values close to zero the function is smooth, GELU converges to ReLU as the input deviates away from zero. GELU offers slight improvements to training speed and accuracy.

2.2.2 Training Deep Models

During training, the forward pass produces a scalar cost $J(\boldsymbol{\theta})$. The weights of the network are updated with backpropagation by computing gradients from the cost, this requires the cost to be differentiable. The selection of cost function varies depending on the task and labels, cross-entropy is a common choice. In popular deep learning libraries such as PyTorch (Paszke et al., 2019), backprop is implemented with a computational graph. Optimisers are algorithms that update the weights in a model to minimise the cost function. Some examples are stochastic gradient descent, Adam (Kingma and Ba, 2014) and Adam with weight decay (AdamW) (Loshchilov and Hutter, 2017). Regularisation is another important technique to reduce overfitting and improve generalisation error. Dropout (Srivastava et al., 2014) randomly disables units during training, it has been proven to be a cheap and effective regularisation method.

2.3 Transfer Learning

Transfer learning is a machine learning method where a model trained on a dataset is used as the starting model for training on another dataset. This second dataset

should be related to the first and is often smaller than the first. The model achieves better performance on the second dataset than models that were trained only on the second dataset. In the inductive transfer approach, it is thought that the model learns features that may only be present in the first dataset helps the model to generalise better on the second dataset.

Fine-tuning of pre-trained models is a method of transfer learning. A common application of fine-tuning is in computer vision where a model pre-trained on large datasets such as ImageNet is then fine-tuned on a smaller dataset for classification (Gopalakrishnan et al., 2017). The vision model has learned to detect features such as edges, shapes and textures on ImageNet which were also useful for the classification task. Today, fine-tuning pre-trained models has become the de-facto method for image classification and image segmentation.

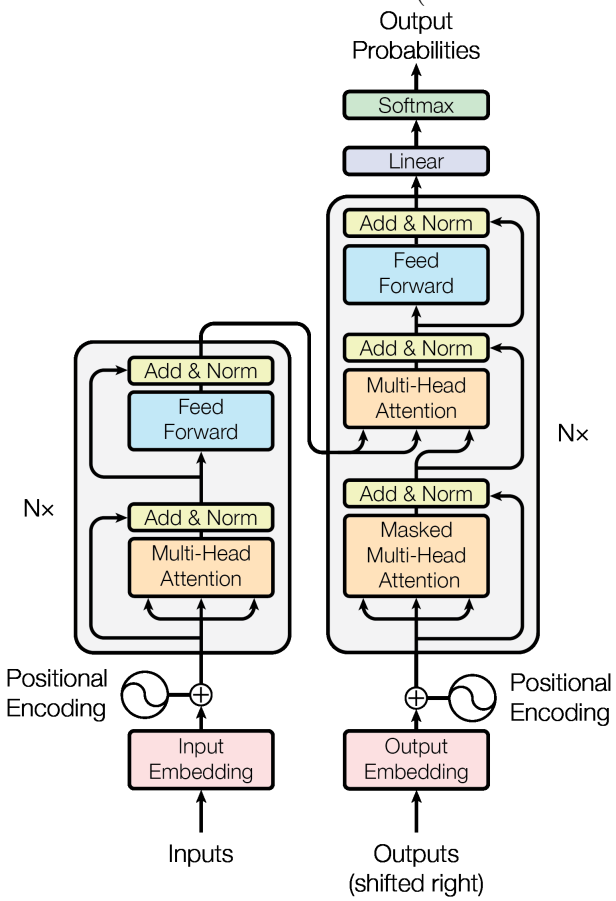
Catastrophic forgetting is a well-known problem in transfer learning and fine-tuning of deep neural networks (McCloskey and Cohen, 1989; I. J. Goodfellow et al., 2013). If a network is trained for too long on a new task, it forgets what was learned from the previous task that was useful for the new task. The performance will keep degrading the longer it is trained and generally never reaches the performance reached before forgetting. The aim of fine-tuning is to train the network to a good level of performance before catastrophic forgetting begins. Regularisation techniques that keep the model close to the original is often employed. Hyperparameters such as batch size, learning rate and momentum all affects fine-tuning. A simple solution is to train the model very slowly by using a small learning rate while keeping all other hyperparameters unchanged. Given enough domain knowledge about particular architectures and datasets, fine-tuning can target only the relevant parts of the model. Other parts of the model can be trained on a more reduced learning rate or frozen, meaning prevented from updating during training.

2.4 Attention and the Transformer model

The transformer (Vaswani et al., 2017) model was proposed for natural language processing (NLP) tasks on long sequences of data. Previous state-of-the-art recurrent neural networks (RNN) process each token sequentially, this causes the positional information of the tokens to be lost as each token need to traverse a long distance through the network to the output. As a result of this architecture, RNNs were limited for NLP tasks as they can only express relationships between terms that are close to one another in the sequence. A second problem with RNNs is that they cannot be trained in parallel, so RNNs are not able to scale into large language models like those based on transformers, such as BERT (Devlin et al., 2018) and GPT-3 (Brown et al., 2020). Transformers solve the problems faced by RNNs. Transformers use attention mechanisms, more specifically a form of attention called self-attention that is capable

to model dependencies between arbitrarily distant terms in the sequence. Since then, attention have been adapted into a number of models and application domains.

Figure 2.1: The Transformer Model (Vaswani et al., 2017)



2.4.1 Scaled Dot-Product Attention

First we look at the simplest form of self-attention. Let x be the input and W^Q, W^K, W^V the learnable weights. The *query, key* and *value* are $Q = xW^Q, K = xW^K$ and $V = xW^V$, each have dimension d_k. Attention is defined as

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

For vector v softmax compute a vector of the same length with entry i as

$$\text{softmax}(v)_i = \exp(v_i) / \sum_j \exp(v_j)$$

The value of the entries of the matrix QK^T indicates the similarity between each query (rows of Q) and key (columns of K^T). The matrix QK^T is scaled by $1/\sqrt{d_k}$, this prevents diminishing gradient problems caused by large values produced by softmax. The softmax function is applied to each row of QK^T to compute a probability distribution. The columns of V are combined according to the computed distributions.

2.4.2 Multi-Head Attention

We can apply attention to the input multiple times in parallel with h heads, this allows each head to attend to the input sequence differently. All the outputs are then concatenated and projected with weights $W^O \in \mathbb{R}^{hd_k \times d_{\text{embed}}}$.

$$\text{MultiHead}(Q, K, V) = (\Big\|_{i=1}^{h} \text{Attention}(xW_i^Q, xW_i^K, xW_i^V))W^O,$$

where $\Big\|$ is the concatenation operation along features. When selecting model hyperparameters, we need d_{model} the overall embedding dimension of the input to be divisible by h the number of attention heads.

2.4.3 Positional Encoding

The self-attention operation is *permutation invariant*, which means the model cannot distinguish between two sequences with the same set of tokens reordered. This is very problematic for the model on datasets where the order of the sequence is important. For example, in NLP the order of words in a sentence can only be arranged so much before all meaning is lost. There is a need to inject the positional information of each token in the sequence into the model, this can be done through positional encoding (PE). Positional encoding can be absolute or relative, fixed or learned. In the original transformer, PE is applied with a matrix with the same dimension as the input added to the input matrix. This is defined by sine and cosine functions, where t is the position of the token in the sequence, $0 \leq i \leq d$ are the embedding dimensions:

$$PE_{(t,2i)} = sin(t/10000^{2i/d_{model}})$$
$$PE_{(t,2i+1)} = cos(t/10000^{2i/d_{model}})$$

This particular PE is relative and fixed, the authors report no improvement when these were learned. This approach is not restricted to sequences, we see later how the structural information of a graph can also be encoded.

2.5 Graph data

In many areas of science, graphs are used to model systems of relations and interactions. For example, computer networks, transportation networks, social networks, brain connectomes, chemical compounds and proteins. There are many types of useful graphs: directed graphs, trees, DAGs and hypergraphs. In this report we only work with *simple graphs*, these are sufficient to model 3D structures such as molecules and proteins. Figure 2.2 is an example of a graph G with two labelled nodes i and j and edge e_{ij}.

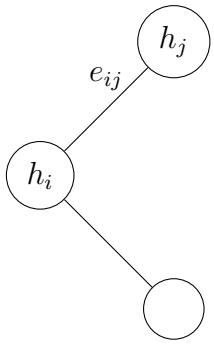


Figure 2.2: Example graph $G(V, E, A)$

Formally, graphs are defined by:

- Vertices V
- Edges E
- Adjacency matrix A

In machine learning there are additional features associated with individual nodes, edges and whole graphs:

- Node features $h_i, h_j \in \mathbb{R}^{d_v}$
- Edge features $e_{ij} \in \mathbb{R}^{d_e}$
- Graph features $g \in \mathbb{R}^{d_g}$

There are many predictive tasks on graphs: at the node, edge and graph-level. In recommendation systems, customers and products are modelled as nodes. We want to predict edges between them to generate recommendations.

2.6 Graph Neural Networks

Deep learning on graphs have its own unique challenges. Starting with a very general definition of a graph neural network: a GNN is an optimisable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances) (Sanchez-Lengeling et al., 2021). GNNs adopt a graph in graph out architecture, while preserving the connections between nodes and edges, these embeddings are progressively transformed into useful features, see Figure 2.3. General GNNs have to scale to arbitrarily large graphs, the update for each node v is restricted to take information from the neighbourhood $\mathcal{N}(v)$ of that node. In a given layer of message passing GNNs, the update for each node can be characterised by an AGGREGATE-COMBINE step. AGGREGATE is a function that collects information from the neighbourhood, it performs an operation on collected feature vectors. The ideal aggregator function is permutation invariant to the input while still being trainable and maintaining high representational capacity (Hamilton, Z. Ying, and Leskovec, 2017). Common aggregator functions are MEAN, MAX and SUM. COMBINE combines the aggregate information with the previous node features to generate a new node representation.

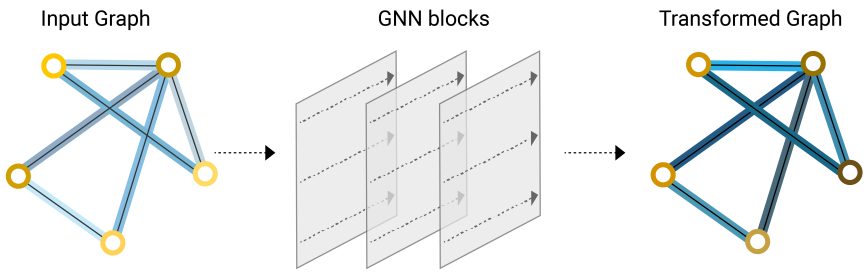


Figure 2.3: Graph Neural Networks (Sanchez-Lengeling et al., 2021)

2.7 Graphormer

The attention mechanism has already been incorporated into a number of GNNs with some success. It was hoped that transformers would revolutionise deep learning on graphs as it did with sequential data. However it was not clear that graph transformers

are better than existing GNNs with significance on many graph learning tasks. For example, the general graph transformer (Dwivedi and Bresson, 2020) performs worse than GatedGCN on ZINC (Irwin et al., 2012). Graphormer (C. Ying et al., 2021) is a modification of the general transformer model for learning on graphs. What makes Graphormer stand out from other models is through its use of inductive biases that is specific to the type of graph in the dataset. Graphormer has seen success across a number of datasets modelling chemical molecular graphs: Open Graph Benchmark PCQM4M quantum chemistry (Hu et al., 2021) and more recently the open catalyst (OC20) challenge (Chanussot et al., 2021). In OC20, chemical molecules (adsorbate) and catalysts are represented as graphs, atoms as nodes and chemical bonds as edges. Interestingly, 3D coordinates of each atom is also included in the dataset. Graphormer have been adapted to molecular graphs with 3D coordinates (Shi et al., 2022), using only node features and node 3D coordinates. The original edges representing chemical bonds is discarded, it is replaced with fully connected edges of euclidean distance between nodes. The molecule and catalyst system is modelled as a fully connected graph of atoms. Following the positional encoding in the original transformer, a structural encoding scheme inject structural information into the attention layers. The structural encoding models euclidean distance between nodes with as set of Gaussian basis functions (Shuaibi et al., 2021).

2.7.1 Gaussian Basis Function

Gaussian basis function (GBF) is a type of radial basis function (RBF). RBFs are designed to be prototype detectors, the output is maximised when the input matches the prototype exactly. In deep learning, GBF is a neuron with the activation function:

$$h_j(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2}\frac{||x-\mu_j||^2}{\sigma^2})$$

2.7.2 Structural Encoding

Spatial Encoding: The distance encoding between two nodes t and s is encoded by a GBF network G, it consists of one layer of GBFs followed by a dense layer. Gain $v_{a_s a_t}$ and offset $u_{a_s a_t}$ are learned features between atom types to account for size differences. Then A_{st} the attention between the nodes is offset by b_{st}.

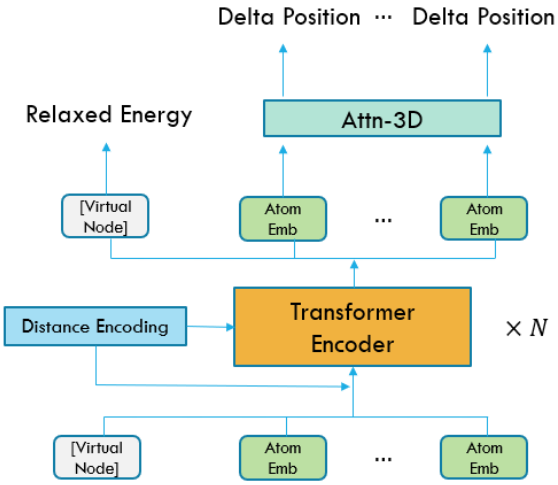
$$b_{st} = G(v_{a_s a_t} d_{st} + u_{a_s a_t} - \mu, \sigma)$$

$$A_{st} = \frac{(x'_s W_Q)(x'_t W_K)^T}{\sqrt{d}} + b_{st}$$

Centrality Encoding: Encodes information between a node and its neighbours, the sum of all distance encoding is added to the node feature.

$$x'_t = x_t + \sum_s b_{st}$$

Figure 2.4: The architecture of Graphormer for 3D molecular graphs (Shi et al., 2022)



Chapter 3

Data and Related Works

This chapter introduces the protein structure datasets suitable for modelling protein interface prediction. We examine the raw data in the Protein Data Base format and additional residue features provided by DIPS-Plus. We define the task of predicting residue pair-wise interaction and review previous machine learning methods on this exact problem.

3.1 Residue contact prediction

As stated in the introduction, we are studying PPI as protein interface contact between residues of two proteins. In the literature, this approach is classified as *partner-specific*. This is contrary to more common partner-independent methods that aim to predict binding sites by identifying residues that are likely to interact with another protein. We adopt a common definition of residue contact (Ahmad and Mizuguchi, 2011), a pair of residues from different chains of proteins is labelled the positive class if the distance between any heavy atoms in the residue is less than 6Å.

3.2 Protein Structural Data

Protein structures are determined experimentally by X-ray crystallography and NMR. These methods are very time consuming, costly and complex. This means that the number of data available is limited, currently there are just over 190,000 known structures available publicly on the Protein Data Bank. We are interested in protein-protein binding structural data, this leaves us with a very small subset of the available data. Recent efforts have led to the creation of a closely related dataset with potential for transfer learning. Its large size makes deep learning for PPI more feasible than ever before.

3.2.1 Raw Data

The Protein Data Bank (PDB) is the single global archive of experimentally determined three-dimensional (3D) structure data of biological macromolecules (wwPDB consortium, 2019). The structural data of individual molecules are stored in the PDB file format. These are text files with a header section containing information about the authors, experiment and publication, the main body contains rows describing each atom in the molecule. Below are a few rows from a typical PDB file displaying the following features: atom, atom number, element, amino acid, chain name, sequence number and X, Y, Z coordinates.

```
ATOM      1  N   GLY A   1      40.425  -3.870  28.102
ATOM      2  CA  GLY A   1      41.321  -5.010  28.055
ATOM      3  C   GLY A   1      42.196  -4.995  26.821
ATOM      4  O   GLY A   1      42.264  -3.993  26.122
ATOM      5  N   ARG A   2      42.868  -6.105  26.547
```

3.2.2 Docking Benchmark 5

Proteins can bind to other proteins to form a protein complex. When binding occurs, the structure of either or both proteins can change (i.e. deform). This is conformational change, a major challenge in protein modelling. Docking Benchmark 5 (DB5) is a collection of protein structures before binding and after binding where the conformational structure changes take place and results in the bound complex (Vreven et al., 2015). DB5 and previous versions of the dataset is the gold standard dataset for PPI. These have been used extensively and for many publications the only training data. This type of data is still very limited, the latest version contains 230 pairs (i.e. dimers).

3.2.3 CASP-CAPRI 13 & 14

Critical Assessment of protein Structure Prediction (CASP) is a competition on protein structure prediction, it inspired AlphaFold2. A related project is the Critical Assessment of Prediction of Interaction (CAPRI) where is main interest is in the prediction of protein-protein interactions. Because only a few structures is released at every CAPRI meeting, the targets for CASP-CAPRI 13 & 14 is suitable as a test set, there are 19 pairs in total.

3.2.4 DIPS & DIPS-Plus

Driven by the lack of data on PPI, the Database of Interacting Protein Structures (DIPS) (Townshend et al., 2018) was created. DIPS targets the problem of paired

protein interface prediction. DIPS was data mined from PDB, searching for binary protein complexes where the structure of individual proteins was not available. The individual protein structures were then retrieved from the complex. DIPS contains significant biases as it is without information on conformational deformations after binding. This may cause models to only generalise for proteins that fit each other almost perfectly before binding. The authors provide evidence that deep learning models can learn representations that encode the flexibility of proteins without training on this type of data.

The Enhanced Database of Interacting Protein Structures for Interface Prediction (Morehead, Chen, Sedova, et al., 2021) is the largest and feature-rich public dataset for the task of protein interface prediction. DIPS-Plus contains 42,112 binary protein complexes and provides additional residue-level features on top of existing features in DIPS. Morehead et al. provide a short summation of available residue features:"(1) an 8-state one-hot encoding of the secondary structure in which the residue is found; (2) a scalar solvent accessibility; (3) a scalar residue depth; (4) a 1×6 vector detailing each residue's protrusion concerning its side chain; (5) a 1×42 vector describing the composition of amino acids towards and away from each residue's side chain; (6) each residue's coordinate number conveying how many residues to which the residue meets a significance threshold, (7) a 1×27 vector giving residues' emission and transition probabilities derived from HH-suite3 (Steinegger et al., 2019) profile hidden Markov models constructed using MSAs; and (8) amide plane normal vectors for downstream calculation of the angle between each intra-chain residue pair's amide planes."

3.3 Classical ML methods

We summarise some earlier classical machine learning methods that did not represent proteins as graphs, these typically involve manual feature extraction from residue sequences combined with classifiers. PAIRpred (Amir Afsar Minhas, Geiss, and Ben-Hur, 2013) is a SVM classifier and uses feature extraction based on domain knowledge, using both sequence and structural features. BIPSPI (Sanchez-Garcia et al., 2018) is a heavily feature engineered, extreme gradient boosted tree model with a novel scoring method.

Moving onto deep learning based methods. ComplexContact (Zeng et al., 2018) preprocesses a pair of protein sequences with multiple sequence alignment (MSA) tools, producing two sequences containing information on functional and evolutionary relationships. Each sequence is then transformed by a 1D ConvNet with residual connections. The resulting sequences are then combined and converted into a matrix, feeding into a 2D ConvNet to produce the final pairwise predictions.

The authors of DIPS (Townshend et al., 2018) have proposed SASNet in the same paper, SASNet is a Siamese-like 3D CNN. Each protein is represented with a voxelised

3D grid, plus a fourth dimension to encode the chemical element of atoms. This method is limited due to 3D convolutions being computationally expensive. Training on only 3% of DIPS, SASNet is able to be competitive with the performance of BIPSPI on DIPS and DB5.

3.4 Graph methods

The first work to model protein residue structure as graphs for protein interface prediction is by (Fout et al., 2017). The concepts and model architecture introduced here have set a trend in the studies following it. DB5 structures are preprocessed to produce graphs with node features representing the properties of the residue. The edge features represent the relationships between residues with distance and angles. The edges are created between a residue and its k closest neighbours. Graph convolutional networks are used to learn node features used later for interface prediction as shown in Figure 3.1. The transformed residual representations are merged by concatenation and passed to a fully connected layer for classification.

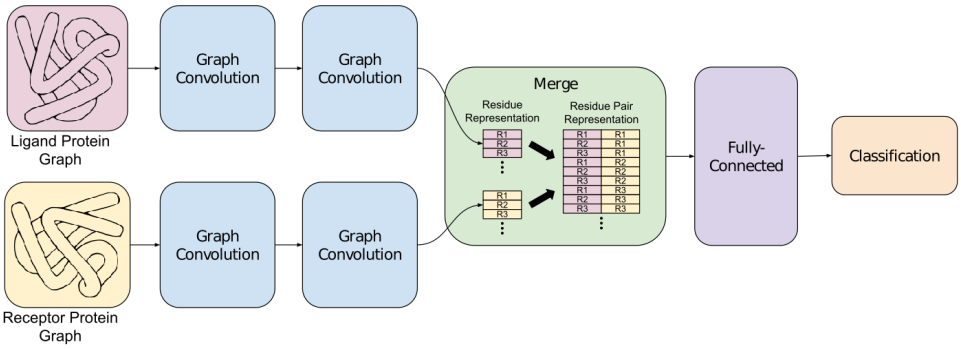


Figure 3.1: GCN for protein interface prediction (Fout et al., 2017)

The work of (Liu et al., 2020) investigated message-passing GNNs for protein interface prediction as shown in Figure 3.2. Both structural and sequential information have been known to be important for protein modelling. Some of the sequential information could be lost when the protein graph is transformed by a GNN. Lie et al. proposed a method to inject sequential information back into the graph. The ordering set of the original residue chain is first recorded, the graph output from the GNN is transformed into a sequence and reordered based on the ordering set. A novel transformation of the prediction problem was introduced, enabling the application of powerful computer vision models. Previously the prediction of each node pairing is done independently, here the sequential node representations of both proteins are merged into a 3D tensor so that CNNs could be employed. The intuition is that the

neighbourhood of a pairing also contains information on interaction, this could be exploited by the various types of filters in CNNs where they aggregate information from the neighbourhood of a node. This allows higher-order interaction patterns to be learned by the interaction predictor. Interestingly, the GNN module was trained on pairs of subgraphs of the parent protein complex, this eventually builds up to the whole interaction prediction tensor. Although no explicit reasoning was given, this was likely a solution to account for the high variability in the protein graph size. This would have made memory management of the implementation much simpler. For handling extra large images in CNNs, an image could be divided into several subsections or tiles. This means subdivision of the interaction tensor is also a valid method, as was the case in DeepInteract.

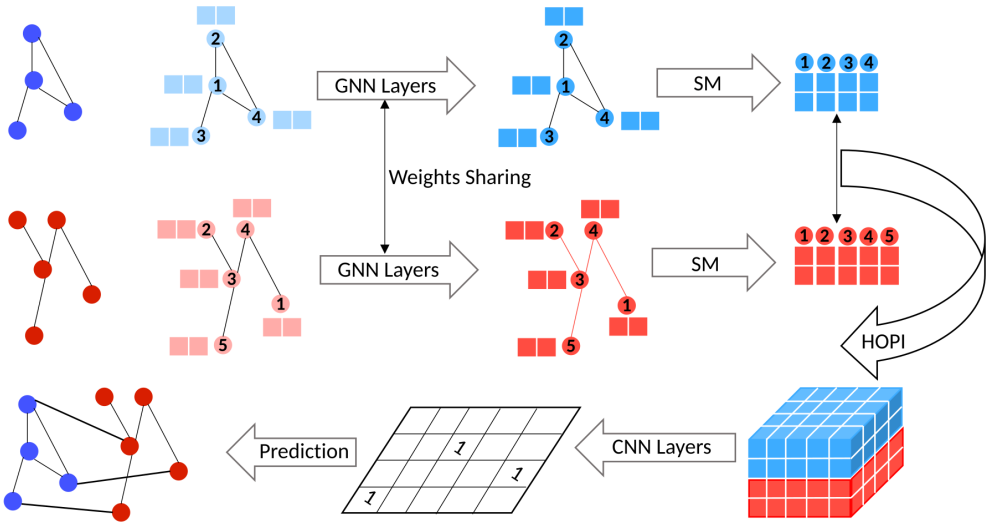


Figure 3.2: The model of (Liu et al., 2020)

DeepInteract (DI) is an end-to-end pipeline for protein interface prediction (Morehead, Chen, and Cheng, 2022). Unlike previous works, the complete source code and data are available publicly for reproduction. The overall architecture of the model is similar to previous work as shown in Figure 3.3. The GNN module extracts node features from a pair of protein graphs then the two node representations are combined to form an interaction tensor. Finally, a vision based interaction module generates interaction predictions.

DI is trained on DIPS-Plus, the pipeline preprocess PDB structures into graphs of residues based on the alpha carbon and edges defined by $k = 20$ nearest neighbours. Extremely long proteins were excluded to limit memory usage. Protein chains that were over 30% sequence identity were also removed according to common bioinformatics practice. DI proposed a novel Geometric Transformer (GeoTran)

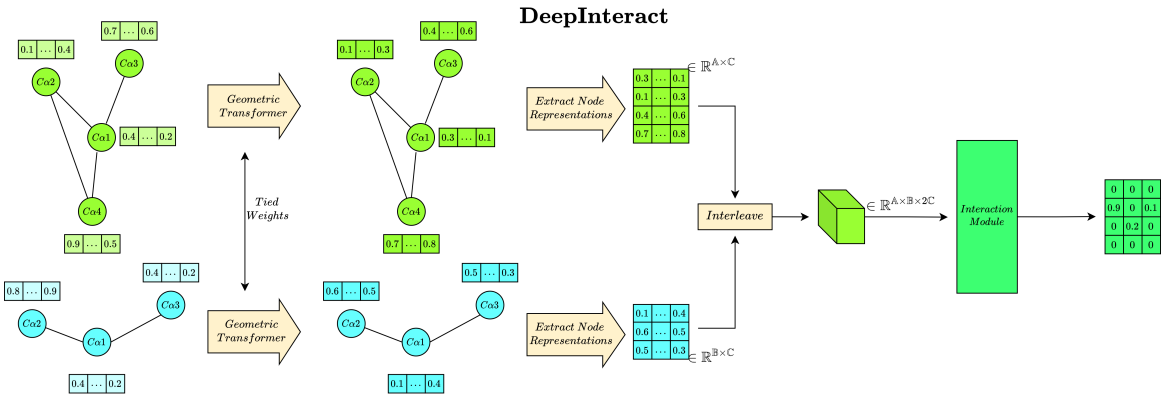


Figure 3.3: DeepInteract (Morehead, Chen, and Cheng, 2022)

shown in Figure 3.4 which is based on a general Graph Transformer by (Dwivedi and Bresson, 2020). The architecture of GeoTran is quite complicated, both node and edge features are updated at each layer. Extensive ablation studies show that each of the components contributes to the final model performance. DI represents protein geometric information through edge features: position of the node in the chain, sinusoidal positional encodings, Euclidean distance and protein specific geometric features proposed by (Ingraham et al., 2019). The edge initialisation module provides the starting edge feature embeddings with gated (Gu et al., 2019) MLPs. GeoTran has a submodule called the conformation module which evolves the edge embeddings at the beginning of each layer of the transformer. GeoTran is rotation and translation invariant to the input, a desirable property since these molecules are floating in the body and interaction is caused by random motions. DI demonstrate the effectiveness of Transformer-based GNNs for learning representations of protein geometries by achieving state-of-the-art performance on CAPRI 13 & 14 and DB5 test sets.

Reading previous works chronologically, it is easy to see that the advancements in protein interface prediction closely follow developments in machine learning. From SVMs to boosted trees, CNNs, GNNs, Geometric deep learning and Transformer-based GNNs. So far, these models achieves poor performance when measured by common metrics such as precision and recall. There is still a great need to develop machine learning models for this task.

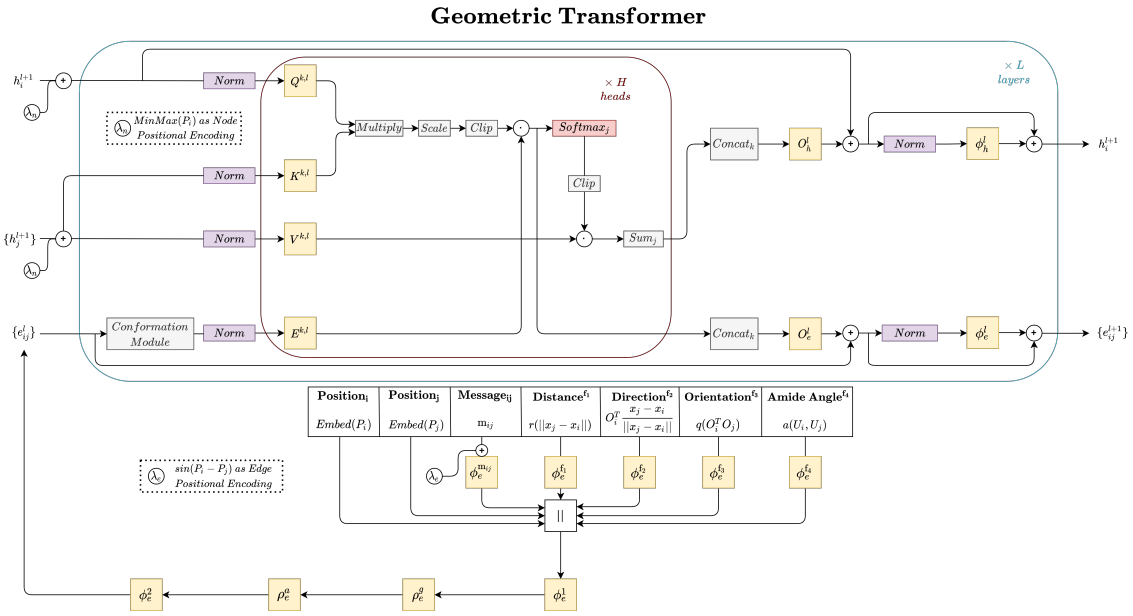


Figure 3.4: Geometric Transformer (Morehead, Chen, and Cheng, 2022)

Chapter 4

Methodology

We base much of our pipeline on DeepInteract due to its state-of-the-art performance and high reproducibility with publicly available code and trained model weights. Our focus is to develop the methods for a graph representation of proteins and suitable graph neural networks. We detail the exact model architecture, giving justification for our model design and model validation method. We take into account engineering considerations when working with limited computational resources such as distributed GPU training and memory management.

4.1 Problem Formulation

We start with a pair of graphs representing two proteins, $h_A \in \mathbb{R}^{A \times C}$ and $h_B \in \mathbb{R}^{B \times C}$. Where C is the number of node features, which is the same in both graphs. The prediction target is the interaction between each pair of amino acids $I \in \mathbb{R}^{A \times B}$. Interactions are labelled with binaries, 1 indicating presence of interaction and 0 otherwise, $I_{ij} \in \{0, 1\}$. We take a different approach from previous works and model proteins as *complete* graphs with edges defined by node 3D coordinates. Previously, edges were defined by k-nearest neighbours of its nodes and edge features are preprocessed accordingly. We discard these features and compute new features used to represent the relative spatial positions of the nodes.

4.2 Data Preprocessing

DB5 and CASP-CAPRI 13 & 14 structures are preprocessed in the same way as DIPS-Plus, additional node and edge features are generate for the structural graph.

4.3 Model Overview

We give an overview of the entire model from input to prediction. Following the architecture of previous works, the overall model is essentially a Siamese network with two weight sharing GNN legs. The two preprocessed protein graphs are passed through our graph transformer to extract node level representations used for contact prediction. Interleaving is the replication and extension of a tensor along a dimension. The selection of computer vision models for the interaction module has been thoroughly explored, including Vision Transformers and MLP-based models. We use the dilated ResNet proposed by DI for the interaction module.

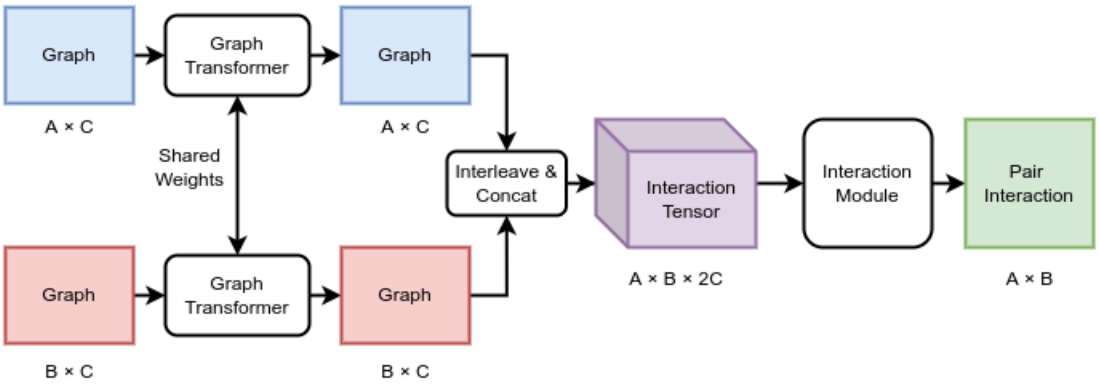


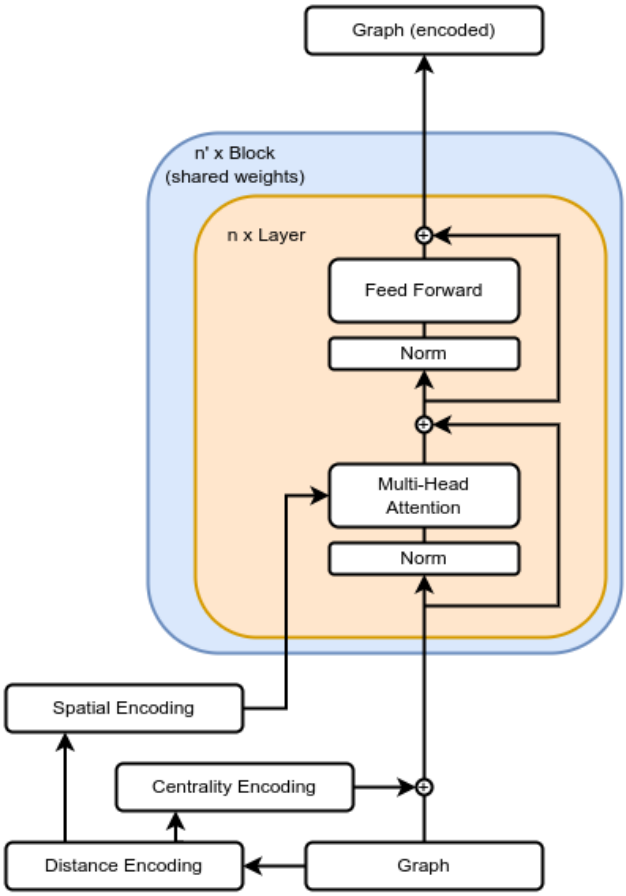
Figure 4.1: Proposed Model

4.4 Graph Transformer Architecture

Transformers have been proven to be powerful general models, even in protein modelling. This is evidenced by the significant architectural change from AlphaFold1 (Senior et al., 2020) to AlphaFold2, the ConvNet-based backbone is replaced with the Transformer-based backbone. Theoretically, there is no upper limit for the length of input for transformers, however current computing limitations restrict inputs to be 512 or 1024 tokens. Fortunately for modelling protein residues, human proteins are on average 375 residues long with almost all below 2000 in length (Brocchieri, 2005). This means that is it is practical to model proteins as fully connected graphs with Transformers. Adapting parts of AF2 would be an interesting direction, but it is out of the question due to its size, complexity and hyper specialised architecture. We require a general Transformer-based GNN that is compatible with DI. Taking the

above considerations into account, we base our graph transformer on Graphormer, as shown in Figure 4.2. Graphormer achieves state-of-the-art results on tangentially related benchmarks of 2D and 3D atomic graphs of chemicals. The learnable spatial encoding used for the 3D atoms OC20 dataset is especially relevant to our problem, both are graphs of similar size with node 3D coordinates.

Figure 4.2: Graph Transformer based on Graphormer (Shi et al., 2022)



Now we begin a walk through of the architecture of our proposed model. The input for our model is the node features and node 3D coordinates, these are stored in separate matrices. First, from the node 3D coordinates, the full node-node Euclidean distance matrix is computed. This is the input into a set of Gaussian basis functions to produce the distance encoding. The output is connected to two separate linear embedding layers to produced the centrality encoding and spatial encoding.

$$b_{st} = G(d_{st} - \mu, \sigma)$$

Node features are embedded with a linear embedding layer with no bias, the number of units in this layer determines d_{embed} the embedding dimension of our transformer. The node feature is added with the centrality encoding and then passed into the transformer. The architecture of our transformer is identical to the original transformer with a few exceptions, see Figure 2.1. All ReLU activations are replaced with GELU. Layer normalisation is applied before multi-head attention (MHA) and feed forward (Xiong et al., 2020). This change has been universally adapted to current transformers as it leads to more effective training (Narang et al., 2021). The same spatial encoding is passed into the MHA in each transformer layer. In each block, the graph is passed through the same n transformer layers. The weight sharing blocks were inspired by the Evoformer in AF2, where the graph representation is recycled 3 additional times through the 48 layer transformer. With each pass through the transformer, the graph features produced are refined.

It is important to maintain the original residue chain ordering of nodes in the construction of the interaction tensor (Liu et al., 2020). Our Graph Transformer takes a sequence of nodes as input and produces an encoded sequence as output. The ordering of the nodes is not changed by any operation. This is an simplification over message-passing GNNs where a reordering operation is required.

4.5 Model validation

In essence, our task is a multi-label binary classification problem. Due to highly imbalanced classes, we have to use metrics like precision and recall. We use top k precision (i.e. precision at k, P@k) and recall (i.e. recall at k, R@k). This metric is commonly used in information retrieval. For the value of k, we used a proportion of the length of the shortest protein chain L. For $P@k$, we choose $k = 10, L/10, L/5$ and for $R@k$, $k = L, L/2, L/5$.

$$P@k = \frac{T_{pos_k}}{k}$$

$$R@k = \frac{T_{pos_k}}{T_{pos}}$$

Cross validation requires a model to be trained from scratch for each fold, this is not always feasible due to resource constraints. We use the hold-out method, splitting the DIPS-Plus and DB5 into train and test sets. We train and fine-tune our models on the train set and test performance on the test set.

4.6 Hyperparameter tuning

In machine learning, the training of models is mostly non-convex optimisation problems. Hyperparameters are parameters that are used to control the training process. There are optimiser specific hyperparameters such as learning rate and momentum. Model related dropout rate and temperature. Data related batch size and mini-batch size. Different settings in hyperparameters can affect training stability, training time and model performance. By tuning these parameters we aim to find a set of values that maximises some of these training objectives. Certain combinations of classes of models and data are more sensitive to hyperparameter tuning than others. We choose to use simple manual tuning strategies as opposed to systematic ones (e.g. grid search, random search).

4.6.1 Basic strategies

If we are limited by computational resources, our model is large and takes a long time to train. We should train models serially the select the best one. During the training of each model, the loss curve should be monitored. Depending on the characteristics of training performance in recent epochs, we can choose to intervene by stopping training and manually adjusting some hyperparameters. If training is very slow we can increase the learning rate.

4.7 Transfer Learning on DB5

DeepInteract have demonstrated transfer learning between DIPS-Plus and DB5 can produce good results. This is somewhat surprising since DIPS-Plus contains no conformational structural changes, and DB5 only contains complexes with conformational change. We take the model trained on DIPS-Plus and *finetune* it on DB5, that is training the model with small learning rates from $1e^{-5}$ to $1e^{-4}$. Empirically, regularisation (e.g. dropout) and certain activation functions (e.g. ReLU) reduce the effects of forgetting (I. J. Goodfellow et al., 2013). Fortunately, these methods are already used in our model. Dropout is used throughout and GELU is very close to ReLU.

4.8 Baselines

Reproducing baseline results for this task is challenging due to the variety of software and technology used in previous works. There is a lack of source code or web servers available to generate predictions. We will use baseline performance

published in DeepInteract. Our baselines are BIPSPI (BI), ComplexContact (CC) and DeepInteract (DI).

4.9 Distributed Training

Distributed training allows for models to be trained in less time and therefore speeding up model design and iteration. Depending on the model architecture, the training of large models with lots of parameters is made possible by distributed computing. It is well known that transformers scale well with model size, in both depth and width, giving rise to successful applications such as large language models (Chowdhery et al., 2022). Distributed training can be done on a single machine with multiple GPUs or on a cluster of such GPU nodes. Generally, computational efficiency decreases as the number of GPUs used in parallel increases. For example, training on 64 GPUs does not increase training speed by 64 times that of a single GPU. There is a further distinction between data-level and model-level parallelism where a trade-off needs to be made. In training, data parallel replicates the model and optimiser in all GPUs, the training batch is distributed among each GPU (Wolf, 2020). During the forward pass, each GPU computes the loss on its own sub-batch, and then all the losses are aggregated by the CPU. The aggregate loss is used to compute gradients and update model parameters in the backward pass. When models get too large for a single GPU, the model parallel is used, and parts or all components during training need to be divided among GPUs: model parameter, gradients and optimiser states (Rajbhandari et al., 2019). A lot more inter-GPU communication is needed which will reduce GPU efficiency. We should select the appropriate level of parallelism based on available hardware and model size.

Chapter 5

Experiments & Results

We train two models with different parameterisations and slightly different architectures, we name them the main model and the alternative model. The main model is the better performing of the two. We trained the main model for a few epochs before training the alternative model. As the performance of the alternative model was poor, we resume training the main model and then used the trained model for transfer learning on DB5.

5.1 Main model

The main model has 4 transformer layers, 12 recycling blocks, 128 embedding dimension and 8 attention heads. We used the Adam optimiser with learning rate of 1e-3, weight decay 1e-2, batch size 1, dropout probability 0.1. The full hyperparameter configuration of our main model is shown in Table A.1. The model implementation is trainable on a multiple GPU machine and is capable of operating on a single GPU machine with at least 40GB of VRAM for testing and inference. The The training was done on cloud instances of 4 A100 or 4 A6000 GPUs based on availability, using *ddp_sharded* strategy which is based on the ZeRO-2 (DeepSpeed, 2020) class of algorithms. We implement our model with the PyTorch Lightning (Falcon et al., 2019) library. We use wandb for experiment tracking and ease of reproducibility (Biewald, 2020). Some fine-tuning and testing were on a single A6000 GPU, this was a cost-saving method as there were lots of downtime between stopping training to update code and reloading the model to resume training. We train the model on DIPS-Plus, stopping after the validation cross-entropy increased after 1 epoch of training, the test result is shown in Table 5.1. Each training epoch takes about 40 minutes. The trained model is then tested on the CASP-CAPRI dataset, see Table 5.2.

Method	P@10	P@L/10	P@L/5	R@L	R@L/2	R@L/5
BI	0.01	0.01	0.01	0.01	0.004	0.003
DI	0.20	0.19	0.17	0.15	0.09	0.04
Graphormer	**0.33**	**0.32**	**0.28**	**0.24**	**0.15**	**0.07**

Table 5.1: The average top-k precision and recall on DIPS-Plus test targets.

Method	P@10	P@L/10	P@L/5	R@L	R@L/2	R@L/5
BI	0.01	0	0.01	0.02	0.01	0.001
DI	0.21	0.19	0.14	0.13	0.08	0.04
Graphormer	**0.55**	**0.53**	**0.44**	**0.39**	**0.27**	**0.14**

Table 5.2: The average top-k precision and recall on CASP-CAPRI 13 & 14 test targets.

Transfer Learning. The model is then fine-tuned on the DB5 training set, we started fine-tuning with a conservative learning rate of 1e-5 and kept other hyperparameters unchanged. Fine-tuning behaviour on this type of data and model is unknown so it is good practice to start slow. After noticing the training loss continues to decrease linearly we increased the learning rate to 5e-5 and then to 1e-4. Near the end of training we reduced the learning rate down to 1e-5 until early stopping on validation cross-entropy kicked in with patience 5 and minimum improvement 5e-6. As recommended by the literature in transformer fine-tuning (Lee, Tang, and Lin, 2019), we started fine-tuning with the body of the model frozen. The body is the part of of the model closer to the input, here it is GNN module. When the validation cross-entropy stops decreasing, the entire model is unfrozen to continue fine-tuning. The DB5 test set results is shown in Table 5.3.

Method	P@10	P@L/10	P@L/5	R@L	R@L/2	R@L/5
BI	0	0.002	0.001	0.003	0.001	0.0004
CC	0.002	0.003	0.003	0.007	0.003	0.001
DI	0.013	0.009	0.011	0.018	0.010	0.0034
Graphormer(finetune)	**0.0164**	**0.0150**	**0.0168**	**0.0288**	**0.0143**	**0.0055**

Table 5.3: The average top-k precision and recall on DB5 test targets.

Our results show that when measured by top-k precision and recall, our graph transformer performs better than the Geometric Transformer proposed by DI on all three datasets. Both models perform poorly on DB5 as it was the target of transfer learning and is a very small dataset. Ideally, there would be cross validation, but our proposed method consistently beats DI across metrics and datasets. We are confident that the current model's performance would generalise even after cross validation.

Given that only two models have been trained due to time and resource constraints, the effects of hyperparameters such as the number of layers, blocks and embedding dimensions have not been explored.

We provide visualisations of predictions generated by our model as shown in Figure 5.1. A prominent feature visible on the softmax plots for both our model and DI is the checkerboard like artefacts. This is to be expected since both models use the same CNN based interaction module. The convolution operation may be causing false positives and negatively impacting model performance. This is somewhat alleviated by thresholding the final positive predictions by a value of 0.5. The dilated ResNet probably produce less such false positives than other types of CNN and perhaps this was partly why DI chose it over alternatives.

5.2 Alternative architecture

To explore the effects of scaling up the Graph Transformer, we train an alternative model with a modified architecture. We increase the model width, the alternative model have increased the number of embedding dimensions from 128 to 864 and attention heads from 8 to 54. A linear layer is added between the GNN module and interaction module to reduce the node embedding dimension. This layer reduces the dimension of node features from 864 to 128. The alternative model configuration is shown in Table A.2. We abandon the alternative model as the training validation loss is much worse than the main model as shown in Figure 5.2, we did not benchmark performance on any of the test sets.

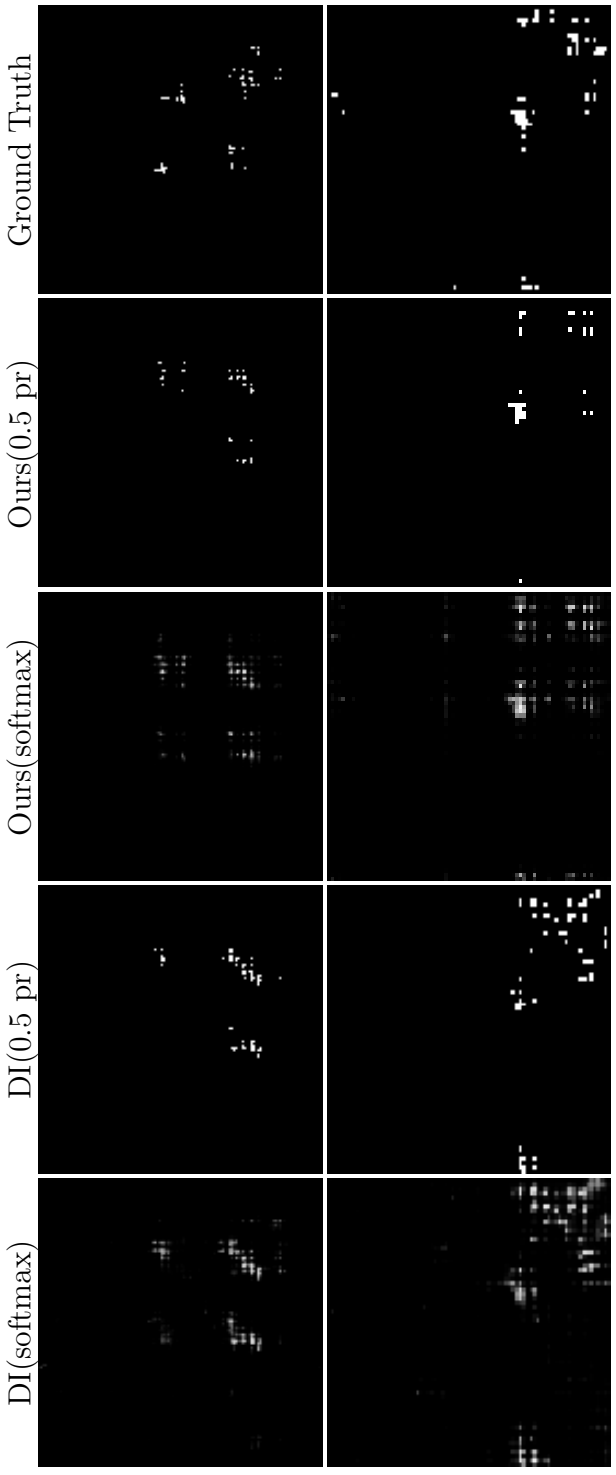


Figure 5.1: Comparison of DI and our model. Visualisations of softmax contact probabilities, 0.5 positive probability threshold predictions, and ground-truth labels. PDB ID:4HEQ (first row) and 6TRI (second row)

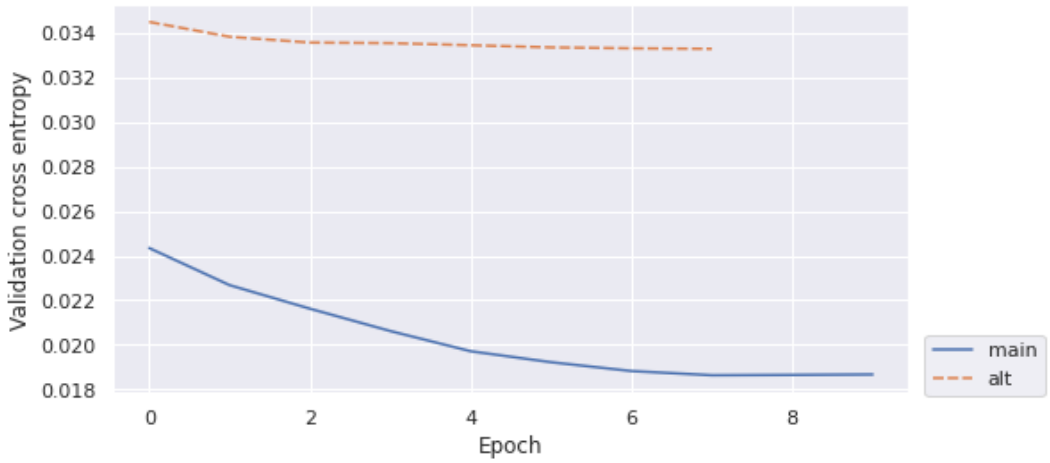


Figure 5.2: Training validation cross entropy

Chapter 6

Discussion

6.1 Comparison with DeepInteract

Comparing our work to DeepInteract, while both make use of Transformer-based GNNs, there is a major difference in the way proteins are represented as graphs. Conceptually, we were inspired by AF2's Transformer architecture and its representation of proteins as full spatial graphs. On the other hand, DI represents proteins as 20-NN sparse graphs, this is the same representation used in works before DI. We chose Graphormer3D (Shi et al., 2022) which is suitable for such fully connected graphs. Compared to message-passing GNNs such as GT (Dwivedi and Bresson, 2020) used by DI, Graphormer enjoys two unique characteristics: the global receptive field from Transformer and adaptive aggregation strategy as a result of using learnable spatial encodings. (C. Ying et al., 2021) proves mathematically that many popular GNNs are special cases of Graphormer and Graphormer has more expressive power than these GNNs. In graph theory, *expressive power* is the capability of a model to learn a representation of graphs (Xu et al., 2018). GNNs are often compared with the k-Weisfeiler-Lehman graph isomorphism test (Weisfeiler and Leman, 1968). (Shi et al., 2022) argues that a global receptive field qualitatively improves the expressive power of Graphormer.

Although not mentioned in DI, the Geometric Transformer might not be scalable in depth. This is evidenced by DI only reporting the 2 layer configuration and making no comments on deeper configurations despite having adequate resources. Conversely, we have only tested a shallow 4 layer configuration, Graphormer has shown to be scaled to 12 and 24 layers. There is still room for improvement by scaling deeper. DI might also have stability issues during training as methods for dealing with training on difficult tasks were used, such as the gate mechanism and gradient clipping. Both the training and fine-tuning of our model is stable.

Our model has a much simplified architecture. We have used a simpler edge

representation of Euclidean distance between nodes and do not update the edge representation for each Transformer layer. This puts in question the necessity of seemingly hand crafted edge features used in DI: direction, orientation and amide angle. The obvious benefits of having a simpler model in production as it simplifies version control and hyperparameter optimisation. As it is closely based on the original Transformer, any new improvements in the architecture, training or inference in Transformers could be directly applied.

6.2 General Comments

We have taken an iterative approach as opposed to parallel search in model building and hyperparameter selection. Multiple models could have been trained for one epoch instead of training the much larger alternative model to convergence. Overall this has resulted in the lack of exploring the effects of combining hyperparameters. There are other methods for hyperparameter tuning on large datasets, we could have used a subset of data to tune parameters before moving on to the full training data. The alternative model used a linear layer for dimensionality reduction, as this is the only architecture difference from the main model, it might be the cause of poor performance. A better approach for dimensionality reduction would be performed on the interaction tensor through the interaction module as it was proposed by DI. Where a convolutional layer with 1×1 filters reduced the number of channels.

On reflection, the choice to have 12 blocks recycling the same 4 layers may be excessive and could negatively affect model performance. Justifying the selection of methods is a common problem in deep learning since the combinatorial search space for architecture is so large and experiments are costly. In the literature, the selection of methods is often based on popularity and trends in research without theoretical guarantees or explanations. We were no exception to this.

Etcembly is interested in modelling large multiple protein chain complexes. Although the interaction of multi-chain complexes could be transformed into a series of binary structural interaction prediction, this may affect our model generalisation. Our training data implicitly assumes that a stable structure is formed when two proteins bind together, this may not generalise to complexes which require 3 or more proteins to bind together at the same time. Even if transformation as series of binary interaction is valid, the training data may not contain sufficiently many large complexes for the model to generalise. Another limitation of modelling larger proteins is the size of the graph which means that the model have to accept very long sequences as input. We have filtered out very large complexes for memory management reasons as Transformers scale poorly at $\mathcal{O}(n^2 \cdot d)$ cost complexity. In order to process very long sequences much more memory is required.

Ablation study is a popular method to study the effects of individual components

in models with many components. However, our Graph Transformer is rather simple, therefore ablation study is not suitable. It may be that some residue features used may not be so useful, for example feature (8) amide plane normal vectors was intended to be used by GeoTran to compute geometric edge features. Feature selection methods could have been used to determine if certain residue features were redundant, removing these features would streamline the pipeline and increase model resource efficiency.

6.3 Further work

During the project, DeepMind released the predicted structures of over 200 million proteins, that is nearly all proteins known to science. While these are structures for single proteins, this immediately opens the possibility for a new dataset for PPI based on DIPS, by keeping the protein complexes as the target and replacing individual proteins with structures generated by AlphaFold2.

There were also good reasons to use sparse graphs, (Dwivedi and Bresson, 2020) have found that for datasets with arbitrary graph structures, sparse graph connectivity outperforms full graphs. Since our graph transformer uses learnable spatial encodings, it may be the case that the model learned to enforce a sparse structure over the full graph. If it turns out that there is an optimal level of sparsity for protein graphs, it could mean that GT will become preferable on a dataset preprocessed with this special sparsity. This is because our we have only used a single edge feature, the distance between residues. GT can take advantage of the richer edge features which we did not utilise.

We have almost exclusively studied Transformer-based GNNs for this project. Since Graphormer, spatial encoding has been shown to improve the performance of other types of GNNs. Alternatives such as the recently proposed MP-GNNs with learnable structural and positional representations (Dwivedi, Luu, et al., 2021) could for a sparse graph representation of proteins. Development in this direction could yield highly efficient and powerful architectures for protein structure representation learning. It might be feasible to model proteins at the atomic level, a large increase in resolution than the residue representation used so far.

Graph matching networks could be explored as an alternative to the current method of interaction prediction via computer vision models. These types of models take two graphs as inputs and output relational information between them. The way that the interaction tensor is constructed is inherited from a sequential view of proteins. It would be better if the interaction module utilised graphs which represent the structure of proteins.

Chapter 7

Conclusions

This project aimed to explore the potential of modelling proteins with graph neural networks for drug discovery. We have examined using Transformer-based GNNs for protein interface prediction and proposed a model based on Graphormer which have improved the state-of-the-art performance of DeepInteract. We have discussed the limitations of our model, particularly the cost complexity of full attention. This project have contributed to the understanding of modelling proteins as graphs, further demonstrating the potential of deep learning for computational biology.

Much time was spent on the implementation and training of models and valuable experiences were gained from that. It was interesting to dig deep into research code and libraries in order to understand how they work. It was impressive to see the engineering and open source tools in data science and deep learning that made this project possible. Computational resources were a major challenge throughout this project, machine learning research will become more productive and fruitful with the inevitable development in hardware and software.

Appendix A

Experiments & Results

A.1 Final model hyperparameters

	Hyperparameter	Value
Interaction Module	interact_module_type	dil_resnet
	use_interact_attention	False
	num_interact_hidden_channels	128
GNN Module	layers	4
	blocks	12
	embed-dim	128
	ffn_embed_dim	128
	attention-heads	8
	dropout	0.1
	attention-dropout	0.1
	activation-dropout	0.0
	input-dropout	0.0
	num-kernel	128
	batch_size	1
	lr	1e-3
	weight_decay	1e-2
	weight_classes	False

Table A.1: Final model hyperparameters

A.2 Alternative model hyperparameters

	Hyperparameter	Value
Interaction Module	interact_module_type	dil_resnet
	use_interact_attention	False
	num_interact_hidden_channels	128
GNN Module	layers	4
	blocks	12
	embed-dim	864
	ffn_embed_dim	864
	attention-heads	54
	dropout	0.1
	attention-dropout	0.1
	activation-dropout	0.0
	input-dropout	0.0
	num-kernel	128
	batch_size	1
	lr	1e-3
	weight_decay	1e-2
	weight_classes	False

Table A.2: Alternative model hyperparameters

References

Ahmad, Shandar and Kenji Mizuguchi (Dec. 2011). "Partner-Aware Prediction of Interacting Residues in Protein-Protein Complexes from Sequence Data". In: *PLoS ONE* 6.12. Ed. by Charlotte M. Deane, e29104. DOI: 10.1371/journal.pone.0029104 (cit. on p. 14).

Amir Afsar Minhas, Fayyaz ul, Brian J. Geiss, and Asa Ben-Hur (Dec. 2013). "PAIRpred: Partner-specific prediction of interacting residues from sequence and structure". In: *Proteins: Structure, Function, and Bioinformatics* 82.7, pp. 1142–1155. DOI: 10.1002/prot.24479 (cit. on p. 16).

Biewald, Lukas (2020). *Experiment Tracking with Weights and Biases*. Software available from wandb.com. URL: https://www.wandb.com/ (cit. on p. 27).

Brocchieri, L. (June 2005). "Protein length in eukaryotic and prokaryotic proteomes". In: *Nucleic Acids Research* 33.10, pp. 3390–3400. DOI: 10.1093/nar/gki615 (cit. on p. 22).

Brown, Tom B. et al. (2020). *Language Models are Few-Shot Learners*. DOI: 10.48550/ARXIV.2005.14165 (cit. on p. 7).

Chanussot, Lowik et al. (2021). "Open Catalyst 2020 (OC20) Dataset and Community Challenges". In: *ACS Catalysis* 11.10, pp. 6059–6072. DOI: 10.1021/acscatal.0c04525 (cit. on p. 12).

Chowdhery, Aakanksha et al. (2022). *PaLM: Scaling Language Modeling with Pathways*. DOI: 10.48550/ARXIV.2204.02311 (cit. on p. 26).

DeepMind (2020). *AlphaFold: a solution to a 50-year-old grand challenge in biology*. URL: https://www.deepmind.com/blog/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology (cit. on p. 1).

DeepSpeed (2020). *DeepSpeed & Zero-2: Shattering barriers of Deep Learning Speed & Scale*. URL: https://www.microsoft.com/en-us/research/blog/zero-2-deepspeed-shattering-barriers-of-deep-learning-speed-scale/ (cit. on p. 27).

Devlin, Jacob et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (cit. on p. 7).

Dwivedi, Vijay Prakash and Xavier Bresson (2020). *A Generalization of Transformer Networks to Graphs*. DOI: 10.48550/ARXIV.2012.09699 (cit. on pp. 12, 19, 32, 34).

Dwivedi, Vijay Prakash, Anh Tuan Luu, et al. (2021). "Graph Neural Networks with Learnable Structural and Positional Representations". In: DOI: 10.48550/ARXIV.2110.07875 (cit. on p. 34).

Falcon et al. (Mar. 2019). *PyTorch Lightning*. Version 1.4. DOI: 10.5281/zenodo.3828935. URL: https://github.com/Lightning-AI/lightning (cit. on p. 27).

Fout, Alex et al. (2017). "Protein Interface Prediction using Graph Convolutional Networks". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/f507783927f2ec2737ba40afbd17efb5-Paper.pdf (cit. on p. 17).

Ganea, Octavian-Eugen et al. (2021). "Independent SE(3)-Equivariant Models for End-to-End Rigid Protein Docking". In: DOI: 10.48550/ARXIV.2111.07786 (cit. on p. 2).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press (cit. on p. 6).

Goodfellow, Ian J. et al. (2013). *An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks*. DOI: 10.48550/ARXIV.1312.6211 (cit. on pp. 7, 25).

Gopalakrishnan, Kasthurirangan et al. (2017). "Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection". In: *Construction and building materials* 157, pp. 322–330 (cit. on p. 7).

Gu, Albert et al. (2019). *Improving the Gating Mechanism of Recurrent Neural Networks*. DOI: 10.48550/ARXIV.1910.09890 (cit. on p. 19).

Hamilton, Will, Zhitao Ying, and Jure Leskovec (2017). "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf (cit. on p. 11).

Hendrycks, Dan and Kevin Gimpel (2016). *Gaussian Error Linear Units (GELUs)*. DOI: 10.48550/ARXIV.1606.08415 (cit. on p. 6).

Hu, Weihua et al. (2021). "OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs". In: *CoRR* abs/2103.09430. URL: https://arxiv.org/abs/2103.09430 (cit. on p. 12).

Ingraham, John et al. (2019). "Generative models for graph-based protein design". In: *Advances in neural information processing systems* 32 (cit. on p. 19).

Irwin, John J et al. (2012). "ZINC: a free tool to discover chemistry for biology". In: *Journal of chemical information and modeling* 52.7, pp. 1757–1768 (cit. on p. 12).

Jumper, John et al. (July 2021). "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873, pp. 583–589. DOI: 10.1038/s41586-021-03819-2 (cit. on p. 1).

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization*. DOI: 10.48550/ARXIV.1412.6980 (cit. on p. 6).

Lee, Jaejun, Raphael Tang, and Jimmy Lin (2019). *What Would Elsa Do? Freezing Layers During Transformer Fine-Tuning*. DOI: 10.48550/ARXIV.1911.03090 (cit. on p. 28).

Liu, Yi et al. (2020). "Deep Learning of High-Order Interactions for Protein Interface Prediction". In: KDD '20. Virtual Event, CA, USA: Association for Computing Machinery, pp. 679–687. ISBN: 9781450379984. DOI: 10.1145/3394486.3403110 (cit. on pp. 17, 18, 24).

Loshchilov, Ilya and Frank Hutter (2017). *Decoupled Weight Decay Regularization*. DOI: 10.48550/ARXIV.1711.05101 (cit. on p. 6).

McCloskey, Michael and Neal J. Cohen (1989). "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem". In: ed. by Gordon H. Bower. Vol. 24. Psychology of Learning and Motivation. Academic Press, pp. 109–165. DOI: https://doi.org/10.1016/S0079-7421(08)60536-8 (cit. on p. 7).

Morehead, Alex, Chen Chen, and Jianlin Cheng (2022). "Geometric Transformers for Protein Interface Contact Prediction". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=CS4463zx6Hi (cit. on pp. 18–20).

Morehead, Alex, Chen Chen, Ada Sedova, et al. (2021). *DIPS-Plus: The Enhanced Database of Interacting Protein Structures for Interface Prediction*. DOI: 10.48550/ARXIV.2106.04362 (cit. on p. 16).

Müller, A.C. and S. Guido (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media. ISBN: 9781449369897. URL: https://books.google.co.uk/books?id=vbQlDQAAQBAJ (cit. on p. 5).

Nair, Vinod and Geoffrey E. Hinton (2010). "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *ICML*, pp. 807–814. URL: https://icml.cc/Conferences/2010/papers/432.pdf (cit. on p. 6).

Narang, Sharan et al. (2021). *Do Transformer Modifications Transfer Across Implementations and Applications?* DOI: 10.48550/ARXIV.2102.11972 (cit. on p. 24).

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on p. 6).

Rajbhandari, Samyam et al. (2019). *ZeRO: Memory Optimizations Toward Training Trillion Parameter Models*. DOI: 10.48550/ARXIV.1910.02054 (cit. on p. 26).

Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain". In: *Psychological Review*, pp. 65–386 (cit. on p. 5).

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323, pp. 533–536 (cit. on p. 5).

Sanchez-Garcia, Ruben et al. (July 2018). "BIPSPI: a method for the prediction of partner-specific protein–protein interfaces". In: *Bioinformatics* 35.3, pp. 470–477. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty647 (cit. on p. 16).

Sanchez-Lengeling, Benjamin et al. (2021). "A Gentle Introduction to Graph Neural Networks". In: *Distill*. https://distill.pub/2021/gnn-intro. DOI: 10.23915/distill.00033 (cit. on p. 11).

Senior, Andrew W et al. (2020). "Improved protein structure prediction using potentials from deep learning". In: *Nature* 577.7792, pp. 706–710 (cit. on p. 22).

Shi, Yu et al. (2022). *Benchmarking Graphormer on Large-Scale Molecular Modeling Datasets*. DOI: 10.48550/ARXIV.2203.04810 (cit. on pp. 12, 13, 23, 32).

Shuaibi, Muhammed et al. (2021). *Rotation Invariant Graph Neural Networks using Spin Convolutions*. DOI: 10.48550/ARXIV.2106.09575 (cit. on p. 12).

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56, pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (cit. on p. 6).

Steinegger, Martin et al. (2019). "HH-suite3 for fast remote homology detection and deep protein annotation". In: *BMC bioinformatics* 20.1, pp. 1–15 (cit. on p. 16).

Stokes, Jonathan M. et al. (2020). "A Deep Learning Approach to Antibiotic Discovery". In: *Cell* 180.4, 688–702.e13. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2020.01.021 (cit. on p. 3).

Townshend, Raphael J. L. et al. (2018). *End-to-End Learning on 3D Protein Structure for Interface Prediction*. DOI: 10.48550/ARXIV.1807.01297 (cit. on pp. 2, 15, 16).

Vakser, Ilya A. (Oct. 2014). "Protein-Protein Docking: From Interaction to Interactome". In: *Biophysical Journal* 107.8, pp. 1785–1793. DOI: 10.1016/j.bpj.2014.08.033 (cit. on p. 2).

Vaswani, Ashish et al. (2017). *Attention Is All You Need*. DOI: 10.48550/ARXIV.1706.03762 (cit. on pp. 7, 8).

Vreven, Thom et al. (2015). "Updates to the Integrated Protein–Protein Interaction Benchmarks: Docking Benchmark Version 5 and Affinity Benchmark Version 2". In: *Journal of Molecular Biology* 427.19, pp. 3031–3041. ISSN: 0022-2836. DOI: https://doi.org/10.1016/j.jmb.2015.07.016 (cit. on p. 15).

Weisfeiler, Boris and Andrei Leman (1968). "The reduction of a graph to canonical form and the algebra which appears therein". In: *NTI, Series* 2.9, pp. 12–16 (cit. on p. 32).

Wolf, Thomas (2020). *training neural nets on larger batches: Practical tips for 1-GPU, Multi-GPU & Distributed Setups*. URL: https://medium.com/huggingface/training-larger-batches-practical-tips-on-1-gpu-multi-gpu-distributed-setups-ec88c3e51255 (cit. on p. 26).

wwPDB consortium (Jan. 2019). "Protein Data Bank: the single global archive for 3D macromolecular structure data". en. In: *Nucleic Acids Res.* 47.D1, pp. D520–D528 (cit. on p. 15).

Xiong, Ruibin et al. (2020). "On Layer Normalization in the Transformer Architecture". In: DOI: 10.48550/ARXIV.2002.04745 (cit. on p. 24).

Xu, Keyulu et al. (2018). *How Powerful are Graph Neural Networks?* DOI: 10.48550/ARXIV.1810.00826 (cit. on p. 32).

Ying, Chengxuan et al. (2021). "Do Transformers Really Perform Bad for Graph Representation?" In: DOI: 10.48550/ARXIV.2106.05234 (cit. on pp. 12, 32).

Zeng, Hong et al. (2018). "ComplexContact: a web server for inter-protein contact prediction using deep learning". In: *Nucleic Acids Research* 46.W1, W432–W437. ISSN: 0305-1048. DOI: 10.1093/nar/gky420 (cit. on p. 16).