PROTEIN-DNA INTERACTION PREDICTION AND PROTEIN STRUCTURE MODELING BY MACHINE LEARNING

A Dissertation presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

CHEN CHEN

Dr. Jianlin Cheng, Dissertation Supervisor

JULY 2022

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

PROTEIN-DNA INTERACTION PREDICTION AND PROTEIN STRUCTURE MODELING BY MACHINE LEARNING

presented by Chen Chen,

a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Jianlin Cheng

Dr. Dong Xu

Dr. James A. Birchler

Dr. Jeffrey Uhlmann

ACKNOWLEDGMENTS

First of all, I would like to give my appreciation to my advisor and committee chair Dr. Jianlin Cheng and my committee members: Dr. Dong Xu, Dr. James A. Birchler, and Dr. Jeffrey Uhlmann. Thank you for your support and guidance in my Ph. D. study. Dr. Cheng not only provides valuable research directions and advice to me though out these years, but also demonstrate what characteristics are required for conducting scientific research. Without his kindness and considerable mentoring, it would be impossible for me to finish the work that I have done during my study. Also, I wish to especially express my gratitude to Dr. James A. Birchler, who have brought great insights from his field to my work, and led many successful collaborations between our labs.

Second, I wish to thank people who are currently working together with me in the lab, including Tianqi Wu, Zhiye Guo, Alex Morehead, Xiao Chen, Jian Liu, Farhan Quadir, Raj Roy, Ashwin Dhakal, Sajid Mahmud, Nabin Giri, Frimpong Boadu, and Yanli Wang. I learned a great deal from many things where they are better than me, and their help and support always inspired me throughout my study. Also, I wish to thank people who have collaborated with me during the years, including previously graduated students from our lab: Dr. Jie Hou and Dr. Meshal Alfarhood, and researchers from Dr. Birchler's lab: Dr. Xiaowen Shi and Dr. Hua Yang. I have a great experience working with them. In addition, I wish to thank people who worked in Oak Ridge National Laboratory: Dr. Ada. Sedova and Dr. Mu Gao, who have offered extremely valuable help for us to fully utilize the computing resources from the Summit Supercomputer.

Finally, I would like to thank my family and friends for supporting me throughout my Ph.D. study. It would not have been possible for everything without the company of you during the years!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Proteins are large, complex molecules that perform most essential functions within organisms. In this work, we mainly focus on two important aspects that determine their functional properties: the tertiary structure of the proteins and their interaction patterns with the genome. Understanding these properties brings valuable insights on the fundamentals of biology and result in new applications in areas such as agriculture, precision medicine, and drug discovery.

The recent developments of bioinformatics and structural biology, machine learning, in particular deep learning has proven to be extremely powerful in inference and interpretation of experimental observations by taking advantage of the large amount data publicly available today. We aim to propose novel machine learning frameworks that can both extract information from higher-level features, and provide explainability for meaningful insights beyond the predictions as well. However, due to the volatility of biology phenomena, the design of data processing and modeling need to be extensive for features from the the proteins. Also, the different geophysical measurements (1D, 2D and 3D) of the protein properties bring new challenges for the selection of model architectures that can effectively leverage different forms of data structure.

In this dissertation, four major contributions are described. First, DeepGRN, is a method for transcription binding site prediction using 1D transformer-based network. Second, GNET2, is a data-assisted method to infer the interactions between proteins and genes from gene expression data using decision tree and information theory. Third, ATTContact, is a tool for protein contact prediction based on 2D residual neural networks with attention mechanism. Finally, EnQA, a method based on 3D equivariant graph networks for protein model quality assessment and selection of the most accurate model as the final protein structure prediction. All the methods

described have been released as open source software, and are freely available to the scientific community.

Chapter 1

Introduction

Proteins are essential molecules that participate in various biological functions and processes in different organisms. For example, transcription factors (TFs) are proteins that bind to genome sequences with unique motifs in the genome to regulate (activate or repress) transcription of target genes [1]. However, it is usually costly and time consuming to acquire accurate functional properties of proteins from experiments. For example, X-ray crystallography and nuclear magnetic resonance (NMR) are golden standard to determine the 3D structure of proteins. Chromatin immunoprecipitation followed by sequencing (ChIP-seq) is the most widely used technique to profile the protein-DNA binding signatures on the genome scale. Due to the constraints of experimental validation methods, computational prediction are drawing much attention as an alternative and less costly approach in these areas.

In recent years, machine learning, especially the deep learning techniques, have proved to be highly capable to large data processing and pattern recognition, and has brought many successful applications in areas such as computer vision [2], natural language processing [3], search/recommendation [4], and many other fields. Machine learning and deep learning have also been exploited in many bioinformatics problems, such as biomedical imaging, biomedical signal processing, and literature mining [5,

6]. These techniques improved the state-of-the-art performance in a variety of tasks previously dominated by traditional methods, and provide insight from data for both academia and industry.

In this dissertation, I mainly focus on my research in the following two topics: the interactions between proteins and genes/genome (Chapter 2 and 3), and the structural modelling of proteins (Chapter 4 and 5).

1.1 Transcription factors binding sites prediction

Transcription factors (TFs) are proteins that can bind to specific genomic sequences and activate or repress the rates of transcriptional activities of downstream genes. Chromatin immunoprecipitation-sequencing (ChIP-Seq) is the most commonly used technique to find all potential binding regions on genomic sequences for TFs. However, it requires specific reagents, such as antibodies for different TFs, thus limits both its availability and scalability. As a result, use of computational methods to predict the binding sites of different TFs is considered as alternative solutions and we have seen many of these methods designed based on different machine learning algorithms [7, 8, 9, 10, 11, 12]. On the other hand, methods developed using deep neural networks, including DeepBind [13], TFImpute [14], and DeepSEA [15], have achieved better performances than traditional machine learning models. These approaches often rely on techniques such as convolutional and recurrent neural networks. Both architectures have the capability to recognize sequential patterns from input features. The convolutional layers are usually used for extracting information from local patterns, while the recurrent layers are better at capturing the long range information. Several recently developed methods, including DanQ [16], DeeperBind [17] and FactorNet [18] are the combinations of both architectures. It is another interesting research topic to interrogate the casual relationship between the input and

prediction results in machine learning,of which the attention mechanism has achieved great success [19, 20]. It increases the capability of existing RNN models to learn the long-range dependencies from the features, and enhances the interpretability of existing deep learning architectures.

In Chapter 2, we introduce a TF binding site prediction tool (DeepGRN) that uses attention mechanism. The experimental results demonstrate that our approach is achieved state-of-the-art performance by comparing with other tools. Our results also demonstrated the pattern of informative features in both DNase-Seq and DNA sequences. The work is published in the following paper:

***Chen Chen**, Jie Hou, Xiaowen Shi, Hua Yang, James A Birchler, and Jianlin Cheng. Deepgrn: prediction of transcription factor binding site across cell-types using attention-based deep neural networks. BMC bioinformatics, 22(1):1–18, 2021* [21]

1.2 Prediction of interactions between transcription factors and genes

In the previous section, we introduced DeepGRN, which utilizes DNase-Seq data to predict the binding site of transcription factors on the genome. However, DNase-Seq results are not feasible to acquire in many experiments, and usually the exact location of binding sites is sufficient to unravel the numerous direct and indirect interactions of transcription factors and genes. A gene regulatory network (GRN) is a set of genes and their regulatory relationship with each other, and plays an important role to control various biological processes and molecular functions [22]. Also, compared with DNase-Seq, the gene expression profile from RNA-Seq experiments are much easier to acquire and more informative in different organisms. As a result, many computational methods have been developed to identify the GRNs from the gene expression profile. They include the approaches modeling the GRNs using similarity

metrics, such as correlation or mutual information, or other confidence scores using information theory [23, 24], Bayesian networks [25] and Gaussian Graphical Model [26, 27].

In Chapter 3, we introduce GNET2 (The Gene Network Estimation Tool), which is developed based on the original GNET algorithm described [27, 28]. GNET uses a decision tree algorithm combined with Gaussian Graphical Model to construct GRN, and has been applied to several previous studies, including regulatory of estrogens [29] and nodulation [27]. We improved the initialization process with the gradient boosting-based approach, and extended the modeling to different conditions by utilizing the similarity in regulatory relationship, which allow the users to acquire more accurate results from replicates of different experiment conditions. The work is published in the following paper:

***Chen Chen**, Jie Hou, Xiaowen Shi, Hua Yang, James A Birchler, and Jianlin Cheng. Gnet2: an r package for constructing gene regulatory networks from transcriptomic data. Bioinformatics, 37(14):2068–2069, 2021.* [30]

1.3 Protein contact maps prediction

In computational modeling of the 3D structures of proteins, the accuracy of the residue-residue contacts plays a critical role, as many software, including AlphaFold [31], and MULTICOM [32] use predicted contact map for computational reconstruction of 3D protein structure. The development of inter-residue coevolutionary analysis provide essential clues for contact prediction from correlated mutation-based analysis [33, 34]. Recent development of deep learning-based methods incorporated with coevolutionary features have brought significant improvements in protein structure prediction [35, 31, 36, 37]. The features of the deep learning models often derives from the direct coupling analysis (DCA). DCA methods use information from multi-

ple sequence alignments (MSAs) to generate correlated mutation information between residues. The information from DCA can be transformed into pairwise 2D feature maps, and then be learned by 2D convolutional neural networks. Several early methods that first adapted this paradigm are RaptorX-Contact [38], DNCON2 [39] and MetaPSICOV [40]. Chapter3 describes a deep learning approach for protein contact prediction, which is equipped with two different architectures based on the attention mechanism. Our results demonstrate that by applying attention mechanisms on the general deep learning-based contact predictors, we are able to improve the accuracy of existing methods. In addition, the attention weights learned by the model make it possible to explain the relationship between position-wise information and the later contact predictions. The work is published in the following paper:

***Chen Chen**, Tianqi Wu, Zhiye Guo, and Jianlin Cheng. Combination of deep neural network with attention mechanism enhances the explainability of protein contact prediction. Proteins: Structure, Function, and Bioinformatics, 89(6):697–707, 2021.* [41]

1.4 Protein model quality assessment

The recent breakthroughs in the end-to-end deep learning method for protein structure prediction - AlphaFold2 [42] and RoseTTAFold [43] present notable improvements in the capability of generating highly confident 3D structures using deep learning methods. However, there are still discrepancies between many predicted structures and the corresponding true structure. For example, while AlphaFold2 is able to predict the structure of the full protein, its results often contain segments with uncertainty, which can cause incorrect identifications of local structural information, such as folds or pockets, which may even result in poor model building [44]. In addition, it is often expected to acquire multiple candidates for one input sequence from deep

learning models. Thus, it is still a challenging task to estimate the accuracy of the predicted tertiary structural models, and provide information of their similarity or discrepancy of the unknown native structure in both local and global level. If accurate estimation of model quality can be acquired, it would contribute to the selection of the final predictions from all candidates, and identify regions with high uncertainty for further refine processes. Chapter 4 describes a model quality assessment predictor based on equivariant graph neural networks, which exploits the geometric properties of the protein models (rotation and translation equivariance). The work is from the following deposited paper:

***Chen Chen**, Xiao Chen, Alex Morehead, Tianqi Wu, and Jianlin Cheng. 3d-equivariant graph neural networks for protein model quality assessment. bioRxiv, 2022.* [45]

Chapter 2

DeepGRN: prediction of transcription factor binding site across cell-types using attention-based deep neural networks

2.1 Abstract

Due to the complexity of the biological systems, the prediction of the potential DNA binding sites for transcription factors remains a difficult problem in computational biology. Genomic DNA sequences and experimental results from parallel sequencing provide available information about the affinity and accessibility of genome and are commonly used features in binding sites prediction. The attention mechanism in deep learning has shown its capability to learn long-range dependencies from sequential data, such as sentences and voices. Until now, no study has applied this approach in binding site inference from massively parallel sequencing data. The successful applications of attention mechanism in similar input contexts motivate us to build

and test new methods that can accurately determine the binding sites of transcription factors.

In this study, we propose a novel tool (named DeepGRN) for transcription factor binding site prediction based on the combination of two components: single attention module and pairwise attention modules. The performance of our methods is evaluated on the ENCODE-DREAM in vivo Transcription Factor Binding Site Prediction Challenge datasets. The results show that DeepGRN achieves higher unified scores in 6 of 13 targets than any of the top four methods in the DREAM challenge. We also demonstrate that the attention weights learned by the model are correlated with potential informative inputs, such as DNase-Seq coverage and motifs, which provide possible explanations for the predictive improvements in DeepGRN.

DeepGRN can automatically and effectively predict transcription factor binding sites from DNA sequences and DNase-Seq coverage. Furthermore, the visualization techniques we developed for the attention modules help to interpret how critical patterns from different types of input features are recognized by our model. The source code of DeepGRN is available at https://github.com/jianlin-cheng/DeepGRN.

2.2 Introduction

Transcription factors (TFs) are proteins that bind to specific genomic sequences and affect numerous cellular processes. They regulate the rates of transcriptional activities of downstream genes through such binding events, thus acting as activators or repressors in the gene regulatory networks by controlling the expression level and the protein abundance of their targeted genes [46]. Chromatin immunoprecipitation-sequencing (ChIP-Seq) is the golden standard to determine the interactions of a TF and all its potential binding regions on genomic sequences. However, ChIP-Seq experiments usually require reagents and materials that are infeasible to acquire, such

as antibodies targeting specific TF of interest. Thus, predictions of potential binding sites through computational methods are considered as alternative solutions. Also, the prediction of binding sites of TFs would facilitate many biological studies by providing resources as reference for experimental validation.

Many algorithms have been developed to infer the potential binding sites of different TFs, including hidden Markov models [7, 8], hierarchical mixture models [9], support vector machines [10, 11], discriminative maximum conditional likelihood [12] and random forest [47, 48]. These methods usually rely on prior knowledge about sequence preference, such as position weight matrix [48]. However, these features may be less reliable if they are generated from inference based methods (such as de-novo motif discovery) when no prior knowledge is available [12].

More recently, methods based on deep neural networks (DNNs), such as DeepBind [13], TFImpute [14], and DeepSEA [15], have shown performances superior to traditional models. Compared with the conventional methods, deep learning models have their advantages at learning high-level features from data with huge sizes. This property makes them ideal for the binding site prediction task since a genome-wide binding profile of a TF can be generated from each ChIP-Seq experiment. Unlike many existing models that rely on the quality of the input data and labor-intensive feature engineering, deep learning requires less domain knowledge or data pre-processing and is more powerful when there is little or no prior knowledge of potential binding regions. Current studies in the protein binding site prediction tasks usually involve the combination of two deep learning architectures: convolutional neural networks (CNN) and recurrent neural networks (RNN). The convolutional layer has the potential to extract local features from different genomic signals and regions [49], while the recurrent layer is better at utilizing useful information across the entire sequences of data. Several popular methods for binding prediction, such as DanQ [16], DeeperBind [17], and FactorNet [18], are built on such model architecture.

Recently, the concept of attention mechanism has achieved great success in neural machine translation [50] and sentiment analysis [19]. It enhances the ability of DNNs by focusing on the information that is highly valuable to successful prediction. Combining with RNNs, it allows models to learn the high-level representations of input sequences with long-range dependencies. For example, long short-term memory (LSTM) models with attention mechanism have been proposed in relation classification [20] and sentence compression [51]. Because of the input context similarities between language processing (sentences) and the DNA binding site prediction (sequences and results from massively parallel sequencing), similar approaches can be applied improve the performance of existing methods [52, 53, 54].

Interrogating the input–output relationships for complex models is another important task in machine learning. The weights of a deep neural network are usually difficult to interpret directly due to their redundancy and nonlinear relationship with the output. Saliency maps and feature importance scores are conventional approaches for model interpretation in machine learning involving genomics data [55]. With the application of attention mechanism, we are also interested in testing its ability to enhance the interpretability of existing CNN-RNN architecture models.

In this paper, we develop a TF binding prediction tool (DeepGRN) that is based on deep learning with attention mechanism. The experimental results demonstrate that our approach is competitive among the current state-of-the-art methods. Also, our work can be extended to explain the input–output relationships through the learning process. We show that the utilization of informative patterns in both DNase-Seq and DNA sequences is important for accurate prediction.

2.3 Materials and Methods

2.3.1 Datasets from ENCODE-DREAM challenge

The datasets used for model training and benchmarking are from the 2016 ENCODE-DREAM in vivo Transcription Factor Binding Site Prediction Challenge with Synapse ID: SYN6131484.

For all TF and cell-types provided in the challenge datasets, the label of the binding status of the TFs is generated from ChIP-Seq experiments and used as ground truth. Chromatin accessibility information (DNase-Seq data), and RNA-Seq data are provided as input features for model training.

For model training, we follow the rules and restrictions of the DREAM challenge: the models are trained on all chromosomes except 1, 8, and 21, and chromosome 11 is used as validation. The model with the best performance in validation data is used for final prediction if no "leaderboard" dataset is provided by the challenge. The leaderboard data are available for some TFs for benchmarking, and each participant can test the performance on these TFs with up to ten submissions. Thus, if such data are provided, we pick the top 10 best models from the first step as an optional model selection step. The final performance of our models is reported based on the final test data that are used to determine the rank of the submissions in the challenge, and is described in the Table 2.1, 2.2 and Figure 2.1.

We use the similar organization of input features introduced by FactorNet: DNA Primary sequence, Chromatin accessibility information (DNase-Seq data) are transformed into sequential features and become the input of the convolution layers at the first part of the models. Gene expression and annotations are transformed into non-sequential features and feed into the intermediate dense layers of the model.

We also collected DNase and ChIP profiles for additional cell lines from the Encode Project (https://www.encodeproject.org) and Roadmap Epigenomics databases

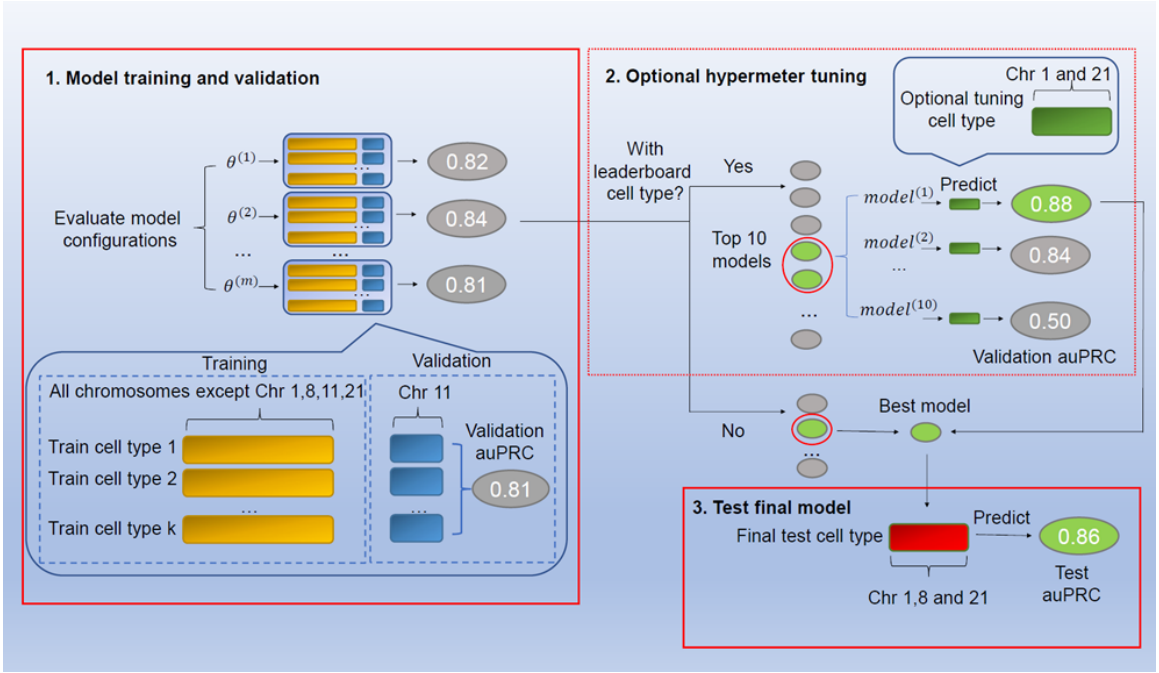

Figure 2.1: The training, hyperparameter tuning, and testing procedure used throughout the Challenge dataset. The models are trained on the data split described in the methods section. If the leaderboard data is available for a TF, then an additional model selection is performed. The final performance evaluation in this study is performed in the dataset for final ranking determination in the challenge. The models are first trained without epigenomic/conservation scores and external datasets. After training, additional models using epigenomic or conservation score trained using the same configuration as the final model of each TF on all training data. Thus, for each TF there are 6 models in total: pair, single, pair+epigenomic, single+epigenomic, pair+conservation, single+conservation. The results are the combination of two basic models and additional models with auPRC in validation data higher than either of the base models while the downstream analysis are based on two base models.

TF	Training Cell types
CTCF	A549, H1-hESC, HeLa-S3, HepG2, IMR90, K562, MCF-7, GM23338*, HL-60*
E2F1	GM12878, HeLa-S3
EGR1	GM12878, H1-hESC, HCT116, MCF-7
FOXA1	HepG2
FOXA2	HepG2
GABPA	GM12878, H1-hESC, HeLa-S3, HepG2, MCF-7, HL-60*
HNF4A	HepG2
JUND	HCT116, HeLa-S3, HepG2, K562, MCF-7
MAX	A549, GM12878, H1-hESC, HCT116, HeLa-S3, HepG2, K562, NB4*
NANOG	H1-hESC, GM23338*
REST	H1-hESC, HeLa-S3, HepG2, MCF-7, Panc1
TAF1	GM12878, H1-hESC, HeLa-S3, K562, GM12891*

Table 2.1: The cell types used for training

TF	Optional Model Tuning	Evaluation
CTCF	GM12878	PC-3, iPSC
E2F1		K562
EGR1	K562	liver
FOXA1	MCF-7	liver
FOXA2		liver
GABPA	K562	liver
HNF4A		liver
JUND	H1-hESC	liver
MAX	MCF-7	liver
NANOG		iPSC
REST	K562	liver
TAF1	HepG2	liver

Table 2.2: The cell types used for optional model tuning and evaluation

(http://www.roadmapepigenomics.org/data/) to improve the capability of generalization of our model. The performance of models trained with and without external datasets are evaluated separately.

2.3.2 Transcription factor binding data

Transcription factor binding data from ChIP-Seq experiments is the target for our prediction. The whole genome is divided into bins of 200 bp with a sliding step size of 50 bp (i.e., 250-450 bp, 300-500 bp). Each bin falls into one of the three types: bound, unbound, or ambiguous, which is determined from the ChIP-Seq results. Bins overlapping with peaks and passing the Irreproducible Discovery Rate (IDR) check with a threshold of 5% [56] are labeled as bound. Bins that overlap with peaks but fail to pass the reproducibility threshold are labeled as ambiguous. All other bins are labeled as unbound. We do not use any ambiguous bins during the training or validation process according to the common practice. Therefore, each bin in the genomic sequence will either be a positive site (bounded) or a negative site (unbounded).

2.3.3 DNA primary sequence

Human genome release hg19/GRCh37 is used as the reference genome. In concordance with the common practice of algorithms that perform feature extraction from chromatin profile, such as FactorNet [18], DeepSea [15], and DanQ [16], we expand each bin by 400 bp in both upstream and downstream, resulting in a 1000 bp input region. In addition, we have evaluated the performance of different selections of input ranges and showed that range above 600 bp is sufficient to acquire stable prediction performance as shown in Figure 2.2. The sequence of this region is represented by a 1000×4 bit matrix by 1-hot encoding, with each row represented a nucleotide. Since low mappability sequences may introduce bias in parallel sequencing experiments, sequence uniqueness (also known as "mappability") is closely related to the quality of sequencing data [57]. Thus, we select Duke 35 bp uniqueness score (https://genome.ucsc.edu/cgi-bin/hgFileUi?db=hg19&g=wgEncodeMapability) as an extra feature. Scores ranging from 0 to 1 are assigned to each position as the inverse of occurrences of a sequence with the exceptions that the scores of unique sequences are 1 and scores of sequences occurring more than four times are 0 [58]. As a result, the sequence uniqueness is represented by a 1000×1 vector for each input bin. The ENCODE Project Consortium has provided a blacklist of genomic regions that produce artifact signals in NGS experiments [59]. We exclude input bins overlapping with these regions from training data and set their prediction scores to 0 automatically if they are in target regions of prediction.

2.3.4 DNase-Seq data

Chromatin accessibility refers to the accessibility of regions on a chromosome and is highly correlated with TF binding events [9]. DNase-Seq experiment can be used to obtain genome-wide maps of chromatin accessibility information as chromatin accessi-

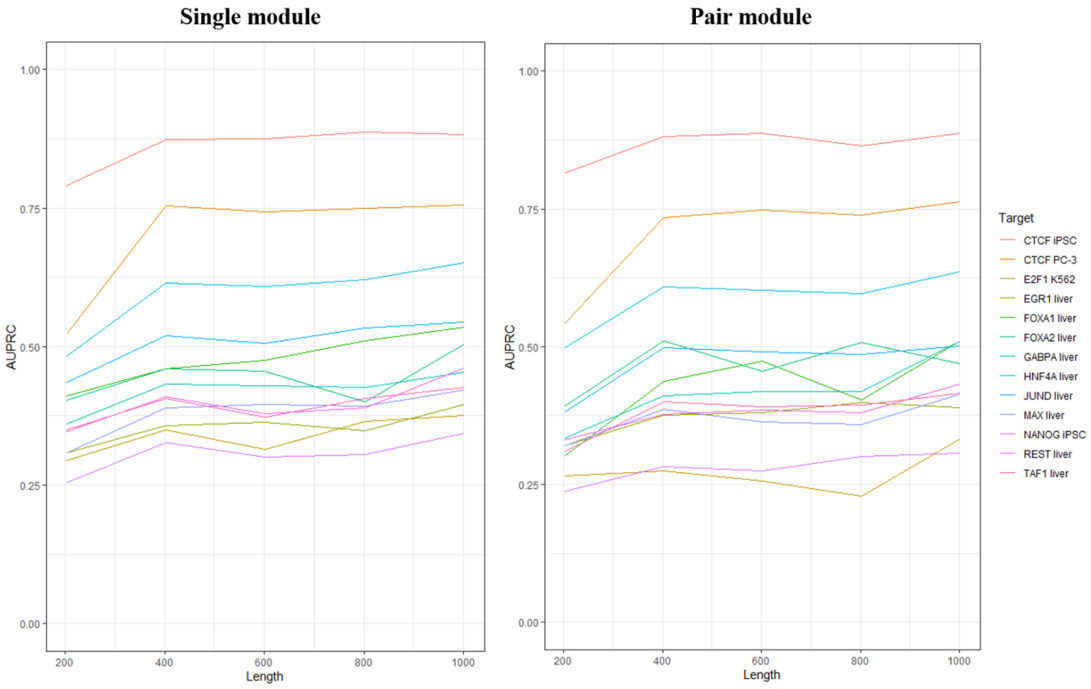

Figure 2.2: Performance of DeepGRN with different selections of input ranges.

ble regions are usually more sensitive to the endonuclease DNase-I than non-accessible regions [60]. DNase-Seq results for all cell-types are provided in the Challenge datasets in the BigWig format. Normalized $1\times$ coverage score is generated from the BAM files using deepTools [61] with bin size $= 1$ and is represented by a 1000×1 vector for each input bin.

2.3.5 Gene expression and annotation

The annotation feature for each bin is encoded as a binary vector of length 6, with each value represent if there is an overlap between the input bin and each of the six genomic features (coding regions, intron, promoter, 5'/3'-UTR, and CpG island). We also include RNA-Seq data since they can be used to characterize the differences in gene expression levels among different cell-types. Principal Component Analysis (PCA) is performed on the Transcripts per Million (TPM) normalized counts from RNA-Seq data of all cell-types provided by the Challenge. The first

eight principal components of a cell-type are used as expression scores for all inputs from that cell-type, generating a vector of length 8. The processed data files for these features are provided in the FactorNet Repository (https://github.com/uci-cbcl/FactorNet/tree/master/resources). These non-sequential features are fused into the first dense layer in the model.

2.3.6 PhastCons genome conservation tracks

We use the 100-way PhastCons conservation tracks [62] as a feature for additional models. The PhastCons scores are represented as base-by-base conservation scores generated from multiple alignments of 99 vertebrates to the human genome. Conserved elements along the genome are recognized from phylogenetic models, and the conservation score for each base is computed as the probability that it locates in such conserved regions. For each input bin, the PhastCons scores are represented as a vector of 1000×1 with a range from 0 to 1.

2.3.7 CpG island feature profiling

We use the CGI score derived from Mocap [63] to profile the epigenomic environment for each input region. The CGI score can be calculated as:

$$CGI\left(N_{CpG}, N_C, N_G, L\right) = \begin{cases} 1 \; if \frac{N_{CpG} L}{((N_C+N_G)/2)^2} > 0.6 \; and \; \frac{N_C+N_G}{L} > 0.5 \\ 0 \; otherwise \end{cases} \tag{2.1}$$

For each input bin, the CGI scores are represented as a vector of 1000×1 with binary values of 0 or 1.

2.3.8 Deep neural network models with attention modules

The shape of each sequential input is $L \times (4 + 1 + 1)$ for each region with length L after combining all sequential features (DNA sequence, sequence uniqueness, and Chromatin accessibility). Sequential inputs are generated for both the forward strand and the reverse complement strand. The weights in all layers of the model are shared between both inputs to form a "Siamese" architecture [18, 14, 64]. Vectors of non-sequential features from gene expression data and genomic annotations are fused into the model at the first dense layer. The overall architecture of our model is shown in Figure 2.3. The model is built with two major modules: single attention and pairwise attention. They use the same input and architecture except for their internal attention mechanism. The final result of our model is the average of the output of two modules.

Preprocessing layers

The first part of our model is a 1D convolutional layer, which is a common practice for feature extraction in deep learning models involving genomics data [13, 18]. We use Bidirectional Long Short-term Memory (Bi-LSTM) nodes as recurrent units in our model. The computation steps in an LSTM unit can be written as:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2.2}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2.3}$$

$$\widetilde{C}_t = tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{2.4}$$

$$\widetilde{C}_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{2.5}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{2.6}$$

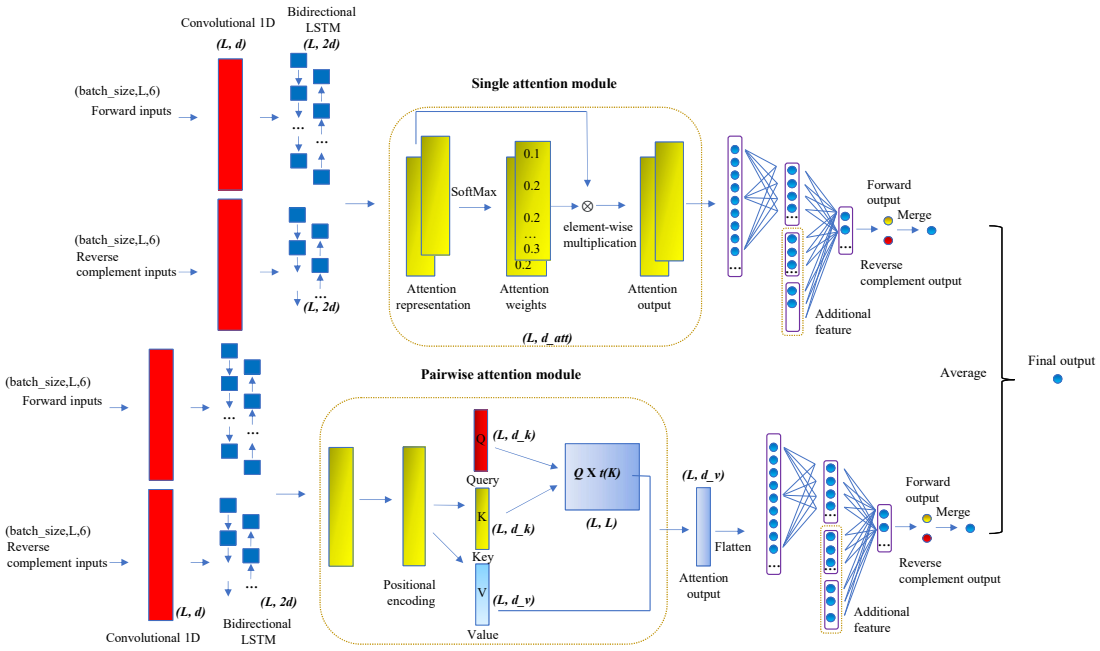

Figure 2.3: The general framework of the two attention modules of DeepGRN. The diagram of the deep neural network architecture. Convolutional and bidirectional LSTM layers use both forward and reverse complement features as inputs. In the single attention module, attention weights are computed from hidden outputs of LSTM and are used to generate the weighted representation through an element-wise multiplication. In the pairwise attention module, three components: Q(query), K(key), and V(value) are computed from LSTM output. The multiplication of Q and transpose of K are used to calculate the attention weights for each position of V. The multiplication of V and attention scores is the output of the pairwise attention module. Outputs from attention layers are flattened and fused with non-sequential features (genomic annotation and gene expression). The final score is computed through dense layers with sigmoid activation and merging of both forward and reverse complement inputs. The dimensions of each layer are shown beside each component

$$h_t = o_t * tanh\left(\widetilde{C}_t\right) \tag{2.7}$$

where f_t, i_t, and o_t are the forget gate, input gate, and output gate. h_{t-1} and h_t are the hidden state vectors at position $t-1$ and t. x_t is the input vector at position t. [h_{t-1},x_t] stands for vector concatenation operation. C_{t-1}, $\widetilde{C}_t$ and C_t are output cell state at position $t-1$, new cell state at position t, and output cell state at position t, respectively. W_f, W_i, W_C, and W_o are learned weight matrices. b_f, b_i, b_C, and b_o are learned bias vector parameters for each gate. σ and $tanh$ are sigmoid function and hyperbolic tangent function, respectively.

In Bi-LSTM layers, two copies of the inputs of LSTM are rearranged into two directions: one for the forward direction and one for the backward direction, and they go into the LSTM unit separately. The outputs from two directions are concatenated at the last dimension. Thus, the last dimension of the Bi-LSTM output is two times of the last dimension of the input.

Single attention

In the single attention module, suppose its input vector h has shape l by r, we first computed the unnormalized attention score $e = M \times h$ where M is a weight matrix with shape l by l, and e has shape l by r. A learned bias of shape l by r is added to e after the multiplication. This can be summarized as a dense layer operation $f_{att,r}$ on input h. Then, we apply the Softmax function along the first dimension of e in order to get the normalized attention score α. Finally, the weighted output Z will be computed based on the attention weight α. At dimension r of input h, these steps can be written as follows:

$$e_r = f_{att,r}\left(h_{1,r}, h_{2,r}, ..., h_{N,r}\right) \tag{2.8}$$

$$\alpha_{i,r} = exp\left(e_{i,r}\right) / \sum_{k=1}^{N} exp\left(e_{k,r}\right) \tag{2.9}$$

$$\alpha_i = (\sum_{r=1}^{R} \alpha_{i,r})/D \tag{2.10}$$

$$z_{i,r} = h_{i,r} * \alpha_i \tag{2.11}$$

Here, e_r is the unnormalized attention score at dimension r. Vector α_i,r represents attention weight at dimension r of position i and is normalized by Softmax function. The attention dimension r in our model will stay unchanged during the transformations. The dimension of the attention weights can be reduced from $N \times r$ to $N \times 1$ by averaging at each position. The final output z_i,r is computed based on the corresponding attention score. After the attention layers, the prediction scores are computed from dense layers with sigmoid activation function and merged from both forward and reverse complement inputs.

Pairwise attention

In the pairwise attention module, there are three components: Q(query), K(key) and V(value). Their values are computed from LSTM output from three different trainable weight matrices. The dimension of the trained weights for Q, K and V are l by d_k, l by d_k and l by d_v where d_k and d_v are set as 64 as the default setup described in [65]. The multiplication of Q and transpose of K are used to compute the attention weights for each position of V after Softmax conversion and dimension normalization. The multiplication of V and attention weights are the output of the pairwise attention module. The output of the pairwise attention module is computed as:

$$Z = Softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V \tag{2.12}$$

Positional encodings

Since each position in the sequential features simultaneously flows through the pairwise attention module, the pairwise attention module itself is not able to sense the position and order from the sequential input. To address this, we add the positional encodings to the input of the pairwise attention. We expect this additional encoding will enhance the ability of the model to make use of the order of the sequence. The positional encodings have the same dimension d as the input of the pairwise attention module. In this work, we choose different frequencies sine and cosine functions [65] to encode the positional information:

$$PE_{(pos,2i)} = sin\left(pos/10000^{2i/d}\right) \tag{2.13}$$

$$PE_{(pos,2i+1)} = cos\left(pos/10000^{2i/d}\right) \tag{2.14}$$

where pos is the position in the sequential input, and i is the index of the last dimension of the model. The resulting positional encodings vector is added to its input. Through such encoding technique, the relative position information can be learned by the model since for any fixed offset k, PE_{pos+k} can be represented as $PE_{(pos,2i)}cos\left(10000^{2k/d}\right) + PE_{(pos,2i+1)}sin\left(10000^{2k/d}\right)$, which is the linear combination of PE_{pos}. Similarly, this also applies to dimensions of $2i+1$ as well.

The single attention module is designed to represent the importance of different regions along with the sequential input, while the pairwise attention module seeks to attend the importance between each pair of positions across the sequential input. We expect this difference in architecture will help to improve the learning ability of the model in a complementary manner.

	Convolution layers				LSTM layers		
Learning rate	Layers	Kernel Size	Dimension	Dropout rate	Layers	Dimension	Dropout rate
0.01	0	15	64	0.1	1	32	0
0.001	1	20	128	0.5	2	64	0.1
0.0005	2	30					0.5
0.0001		34					

Table 2.3: Hyperparameters selection for model training. Due to the limitation of computing capacity 50 sets of the combination of these hyperparameters are sampled without replacement for evaluating the configurations. The single and pairwise modules always share the same configuration.

Training configurations

We tested different configurations for typical hyperparameters (learning rate, network depth, dropout rates) and the hyperparameters specific to our model (the dimension of attention weights, merging function the two output scores) during training. The complete description of hyperparameters and their possible options are summarized in Table 2.3. We train one model for each TF, resulting in 12 models in total. The single and pairwise attention modules will always use the same configuration rather than train separately.

There are 51,676,736 bins in total on training chromosomes in the labels, resulting in $51676736 \times n$ potential training samples for each TF, where n is the number of available cell-types for training. Due to limited computing capacity, we use the iterative training process. During training, the training data is the mixture of all positives (labeled as "B") with downsampled negatives (labeled as "U") [18]. In the traditional model training in deep learning, all input data are used to update the model weights exactly once for each epoch. However, this is not applicable in our task since the negative samples (regions do not bind to TFs) are much more abundant than the positive samples (regions bind to TFs), and use all negative samples for training in one epoch is not practical since the number of them is extremely huge (as they cover most of the human genome). Thus, in each epoch during model training, we first sample negative samples with numbers proportional to the number of

all positive samples, and combine these negative samples with all positive samples for training. We will re-sample the negative bins and start another round of model training (next epoch). To make the training process more effective, we use a different strategy to generate positive training samples for transcription factors that have a large number of positive labels (CTCF, FOXA1, HNF4A, MAX, REST and JUND). For these TFs, we randomly sample a 200-bp region from each ChIP-Seq peak in the narrow peak data as positive instances for training instead of using all positive samples in each epoch. We use the Adam [66] optimizer with binary cross-entropy as the loss function. The default number of epochs is set to 60, but the training will be early stopped if there are no improvements in validation auPRC for five consecutive epochs.

Attention weights and saliency scores

Both attention weights and saliency scores are compted from trained models of corresponding TFs. For saliency scores, the non-sequential input features are considered as weights that are not trainable and only the partial derivative of output values to the sequential features are computed using the tf.gradients() function in Tensorflow. Since both attention weights and saliency scores are high dimensional vectors and generally consist of only noisy values in many of their channels. We only use the channel with the highest correlation with the fold change value for both attention scores and saliency scores for a fair comparison of their correlation. For attention scores, since it has a shorter length than the fold change and DNase values due to the convolution layers in the model, we downsampled those longer sequences to fit the length of the original input sequence in order to calculate the correlation.

TF Name	Cell-type	auROC	auPRC	Recall at 50% FDR	Recall at 10% FDR
CTCF	PC-3	0.987	0.767	0.766	0.603
CTCF	iPSC	0.998	0.902	0.945	0.744
E2F1	K562	0.989	0.404	0.388	0.100
EGR1	liver	0.993	0.405	0.318	0.021
FOXA1	liver	0.985	0.546	0.584	0.164
FOXA2	liver	0.984	0.548	0.588	0.143
GABPA	liver	0.991	0.516	0.488	0.154
HNF4A	liver	0.971	0.636	0.700	0.263
JUND	liver	0.983	0.535	0.585	0.027
MAX	liver	0.990	0.425	0.349	0.004
NANOG	iPSC	0.996	0.499	0.515	0.035
REST	liver	0.986	0.482	0.527	0.030
TAF1	liver	0.989	0.424	0.393	0.000

Table 2.4: The performance of DeepGRN with four metrics used in the DREAM Challenge.

2.4 Results

2.4.1 Overall benchmarking on evaluation data

We list the performance of our model as four metrics used in the DREAM Challenge (Table 2.4) and compare them with the unified score from the top four teams in the final leaderboard of the ENCODE-DREAM Challenge (Table 2.5). The unified score for each TF and cell-type is based on the rank of each metric and is computed as: $\sum ln\left(r/\left(6\right)\right)$ where r is the rank of the method for one specific performance measure (auROC, auPRC, Recall at 50% FDR and Recall at 10% FDR). Thus, smaller scores indicate better performance. The TFs, chromosomes, and cell-types for evaluation are the same as those used for the final rankings. DeepGRN typically achieves auROC scores above 98% for most of the TF/cell type pairs, reaching as low as 97.1% for HNF4A/liver. The scores of auPRC have a more extensive range of values, from 40.4% for E2F1/ K562 to 90.2% for CTCF/iPSC.

For each TF and cell-type combination, our attention model has better perfor-

TF	cell	Anchor	FactorNet	Cheburashka	Catchitt	DeepGRN
CTCF	PC-3	0.67	0.17	0.83	0.5	0.33
CTCF	iPSC	0.83	0.33	0.67	0.5	0.17
E2F1	K562	0.5	0.83	0.67	0.17	0.33
EGR1	liver	0.17	0.83	0.67	0.33	0.5
FOXA1	liver	0.67	0.33	0.83	0.5	0.17
FOXA2	liver	0.33	0.83	0.67	0.5	0.17
GABPA	liver	0.33	0.83	0.67	0.5	0.17
HNF4A	liver	0.67	0.33	0.83	0.5	0.17
JUND	liver	0.17	0.83	0.67	0.5	0.33
MAX	liver	0.17	0.83	0.33	0.67	0.5
NANOG	iPSC	0.33	0.5	0.83	0.67	0.17
REST	liver	0.67	0.33	0.83	0.5	0.17
TAF1	liver	0.17	0.5	0.67	0.33	0.83

Table 2.5: The unified scores of DeepGRN and the top four algorithms in the DREAM Challenge.

mance on 69% (9/13) of the prediction targets than Anchor [67], 85% (11/13) than FactorNet [18], 85% (11/13) than Cheburashka [12], and 77% (10/13) than Catchitt [68]. Among all methods benchmarked, our method has the highest ranking in 7 out of 13 targets (CTCF/iPSC, FOXA1/liver, FOXA2/liver, GABPA/liver, HNF4A/liver, NANOG/iPSC, and REST/liver), with the best average score (0.31) across all TF/ cell-types pairs (Table 2.5).

To precisely evaluate the capability of deepGRN under the restrictions of the ENCODE DREAM Challenge, we also compared the performance of deepGRN trained using datasets provided by the challenge with four available features: Genomic sequence features, DNase-Seq and RNA-Seq data. The results are summarized in Table 2.6 and 2.7. DeepGRN still achieves the highest ranking in 6 out of 13 targets, with the best average unified score (0.33) across all targets. We also compared our results with models without the attention component using the four challenge features. We built these models using the same architecture as deepGRN models, except for the attention component and trained them with the same hyperparameter selection process. The results are shown in Figure 2.4. DeepGRN with attention mechanism

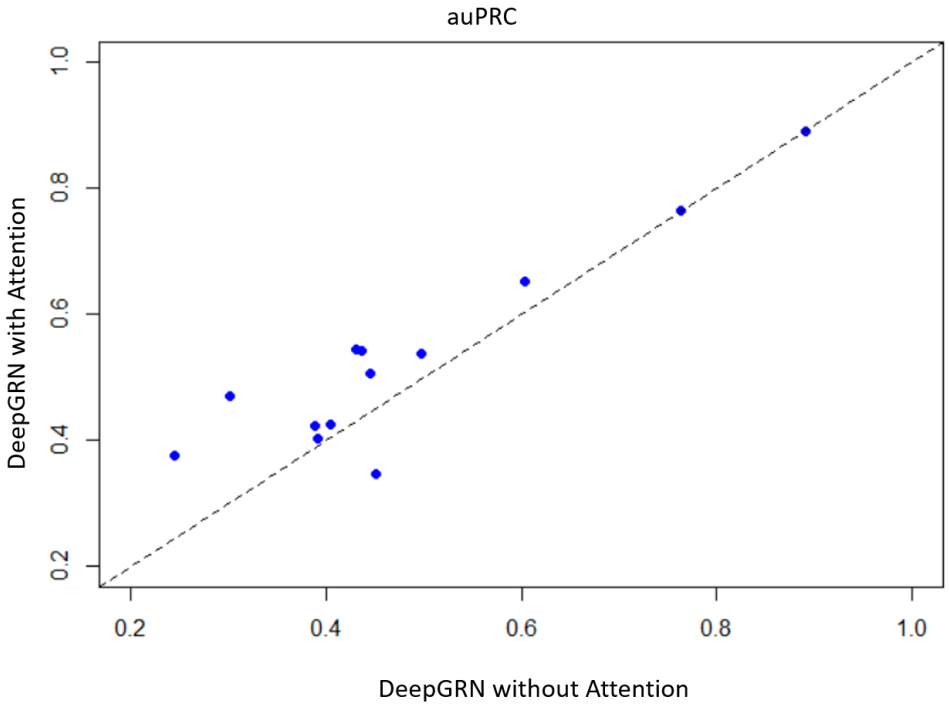

Figure 2.4: Comparision of the deep learning models with and without attention mechanism.

outperforms the models without attention in 11 out of 13 targets by the auPRC metric, with the largest difference from target REST (0.168).

2.4.2 Performance comparison between two attention modules

In addition to the comparisons with the top 4 methods in the challenge, we also benchmarked the individual performance of the single and pairwise attention modules (Table 2.8, 2.9). In general, the results extracted from the single attention module have similar performances. For all 13 TF and cell-type pairs, the single attention module has higher auROC in 6 targets while the pairwise attention module has higher auROC in 3 targets. The rest of the targets are tied. The final output of the model is the ensemble of these two modules by averaging, and it outperforms any of the individual attention modules in 10 of 13 targets (Table 2.4). The largest im-

TF Name	Cell-type	auROC	auPRC	Recall at 50% FDR	Recall at 10% FDR	DeepGRN
CTCF	PC-3	0.987	0.764	0.764	0.595	0.33
CTCF	iPSC	0.998	0.891	0.93	0.733	0.17
E2F1	K562	0.989	0.376	0.338	0	0.33
EGR1	liver	0.993	0.404	0.307	0.019	0.5
FOXA1	liver	0.987	0.544	0.579	0.155	0.17
FOXA2	liver	0.985	0.539	0.573	0.117	0.17
GABPA	liver	0.99	0.506	0.468	0.148	0.17
HNF4A	liver	0.978	0.652	0.69	0.303	0.17
JUND	liver	0.983	0.542	0.605	0.001	0.33
MAX	liver	0.99	0.424	0.34	0.003	0.5
NANOG	iPSC	0.989	0.347	0.314	0.003	0.17
REST	liver	0.986	0.47	0.51	0.025	0.17
TAF1	liver	0.99	0.425	0.39	0	0.83

Table 2.6: The performance of DeepGRN trained with challenge datasets only with four metrics used in the DREAM Challenge.

TF	cell	Anchor	FactorNet	Cheburashka	Catchitt	DeepGRN
CTCF	PC-3	0.67	0.17	0.83	0.5	0.33
CTCF	iPSC	0.83	0.33	0.67	0.5	0.17
E2F1	K562	0.33	0.83	0.67	0.17	0.5
EGR1	liver	0.17	0.83	0.67	0.33	0.5
FOXA1	liver	0.67	0.33	0.83	0.5	0.17
FOXA2	liver	0.33	0.83	0.67	0.5	0.17
GABPA	liver	0.33	0.83	0.67	0.5	0.17
HNF4A	liver	0.67	0.33	0.83	0.5	0.17
JUND	liver	0.17	0.83	0.67	0.5	0.33
MAX	liver	0.17	0.83	0.33	0.5	0.67
NANOG	iPSC	0.17	0.5	0.83	0.67	0.33
REST	liver	0.67	0.33	0.83	0.5	0.17
TAF1	liver	0.17	0.5	0.83	0.33	0.67

Table 2.7: The unified scores of DeepGRN trained with challenge datasets only and the top four algorithms in the DREAM Challenge.

TF Name	Cell type	Single attention			
		auROC	auPRC	Re@0.50 FDR	Re@0.10 FDR
CTCF	PC-3	0.985	0.756	0.762	0.583
CTCF	iPSC	0.998	0.883	0.923	0.712
E2F1	K562	0.99	0.375	0.378	0
EGR1	liver	0.992	0.396	0.309	0.026
FOXA1	liver	0.989	0.535	0.565	0.115
FOXA2	liver	0.986	0.504	0.527	0.052
GABPA	liver	0.987	0.454	0.382	0.134
HNF4A	liver	0.98	0.652	0.699	0.286
JUND	liver	0.981	0.545	0.615	0.013
MAX	liver	0.991	0.422	0.322	0.001
NANOG	iPSC	0.99	0.343	0.292	0
REST	liver	0.986	0.461	0.469	0.009
TAF1	liver	0.989	0.426	0.391	0

Table 2.8: Individual performance of single attention module.

provements from ensemble (as auPRC) come from FOXA2 (0.34), REST (0.09) and FOXA1 (0.09). We also found that the performance of the two attention modules have the same trend across all TF and cell-types in all four performance measures (Figure 2.5), suggesting that the capability of learning from features are coherent between the two modules.

We evaluated the importance of each feature between single and pairwise attention mechanisms. For the prediction of each target, we set the values of each sequential feature (DNase-Seq, sequence, or uniqueness) to zero, or randomly switched the order of the vector for a non-sequential feature (genomic elements or RNA-Seq). The decrease of auPRC from these new predictions is used as the importance score of each feature (Figure 2.6). We found that across all TF and cell-types, the sequential features have the largest average importance scores: DNase-Seq (0.36), DNA sequence (0.21), and 35 bp uniqueness (0.21) while the scores for other features are much smaller. Similar trends have also been found using individual single and pair attention modules.

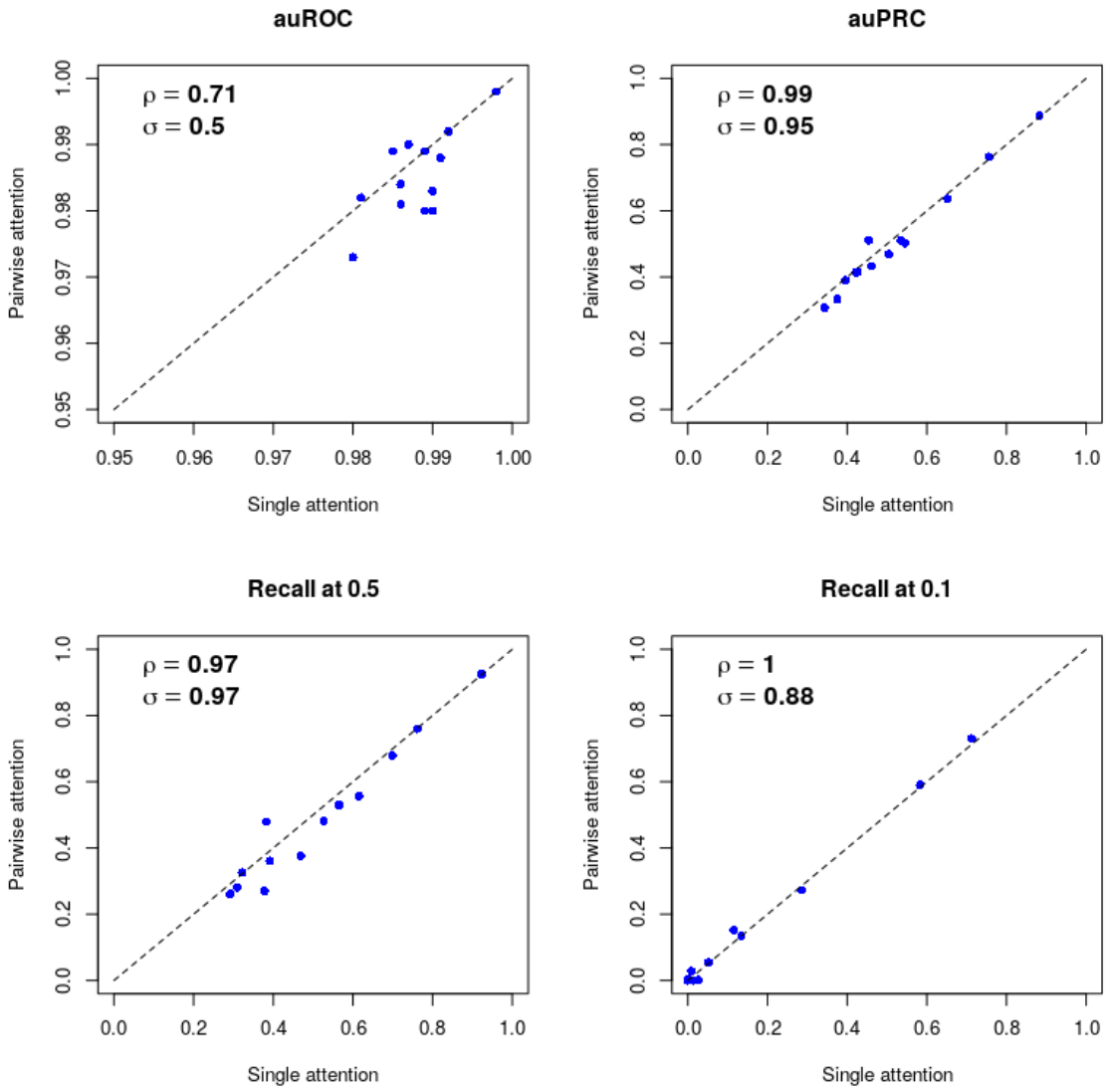

Figure 2.5: Performance comparison between single and pairwise attention mechanisms. The performance of each TF and cell-type pairs of the output of the individual module are shown in four measures: (auROC, auPRC, recall at 50% FDR and Recall at 10% FDR). ρ: Pearson Correlation Coefficient, σ: Spearman Correlation Coefficient

TF Name	Cell type	Pairwise attention			
		auROC	auPRC	Re@0.50 FDR	Re@0.10 FDR
CTCF	PC-3	0.989	0.763	0.76	0.59
CTCF	iPSC	0.998	0.888	0.925	0.73
E2F1	K562	0.983	0.333	0.27	0
EGR1	liver	0.992	0.39	0.281	0.002
FOXA1	liver	0.98	0.511	0.53	0.152
FOXA2	liver	0.981	0.469	0.481	0.055
GABPA	liver	0.99	0.511	0.479	0.135
HNF4A	liver	0.973	0.636	0.679	0.273
JUND	liver	0.982	0.502	0.557	0
MAX	liver	0.988	0.414	0.326	0.003
NANOG	iPSC	0.98	0.307	0.261	0.001
REST	liver	0.984	0.433	0.376	0.028
TAF1	liver	0.989	0.416	0.361	0.002

Table 2.9: Individual performance of pairwise attention module.

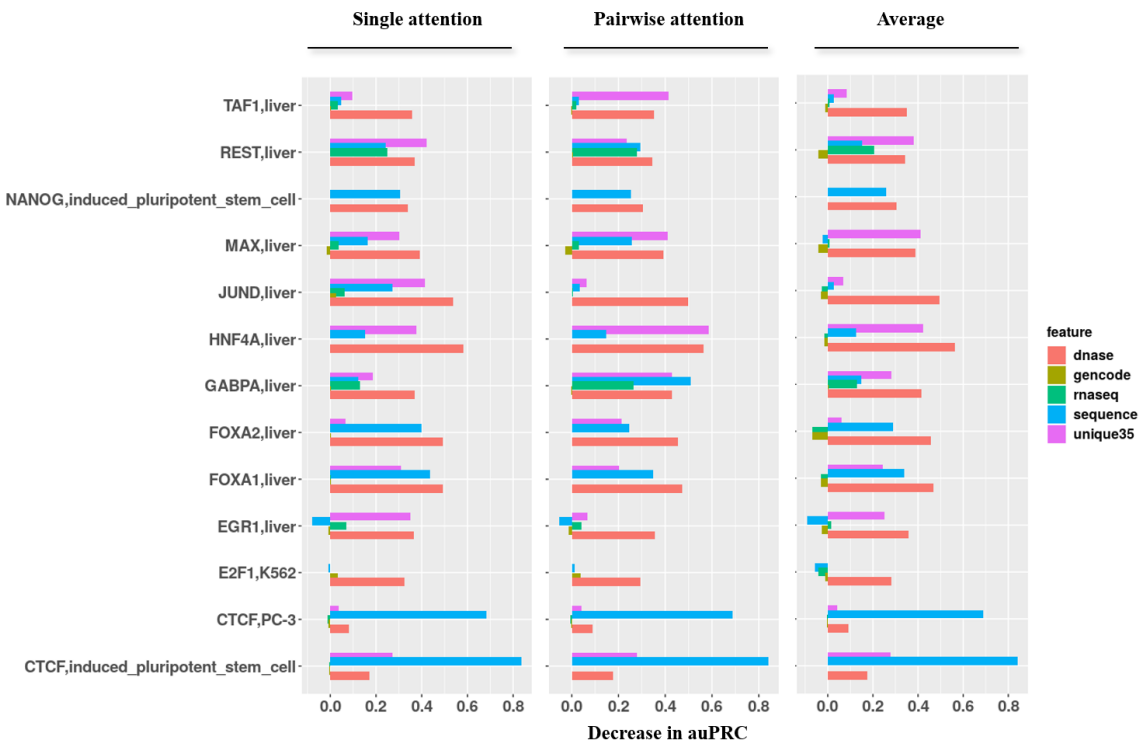

Figure 2.6: Importance score of features between single and pairwise attention mechanisms. The values represented as the decrease of auPRC without using the specific feature for prediction. The negative value represents an increase of auPRC.

2.4.3 Interpretation of attention scores

In the single attention module, the output is a weighted sum of the input from the attention layer, and the attention scores are used as weights. These scores characterize a unified mapping between the importance of input feature with its relative position in the sequential input. To analyze the relationship between attention weights and the position of TF binding events, we extract the attention scores from the single attention module for both forward strand and reverse complement strand and compare them with the corresponding normalized ChIP-Seq fold changes in the same region that are predicted as positive (score > 0.5). Similarly, we computed the saliency scores for the same input regions. We found that the attention scores on the two DNA strands have a higher correlation ($\rho = 0.90$, $\sigma = 0.79$) than the saliency scores ($\rho = 0.78$, $\sigma = 0.51$) (Figure 2.7a, b). Across all TF and cell-type pairs, we found that there is a positive correlation between the attention weights and normalized ChIP-Seq Fold (Figure 2.7c), and such relationship is not detected globally in saliency scores (Figure 2.7d). For all TF and cell-types in the benchmark datasets, we select at least four different genomic regions that have a clear ChIP-Seq peak signal in each target for demonstration. We show that the averaged attention weights put more focus on the actual binding region for each cell-type and these focusing points shift along with the shift of TF binding signals (Figure 2.8).

Since the accessibility of the genome plays an important role in TF binding, it is expected to find high DNase coverage for those openly accessible areas that can explain the binding event detected by the ChIP-Seq experiment. We run a genome-wide analysis on regions with high DNase-Seq peaks in the single attention module for transcription factor JUND, which is one of the most susceptible targets to DNase-Seq. We illustrate the distribution of normalized DNase coverage values from both the true positives that are false negatives without attention and true negatives that are false positives without attention (Figure 2.9). The results show that the true positives

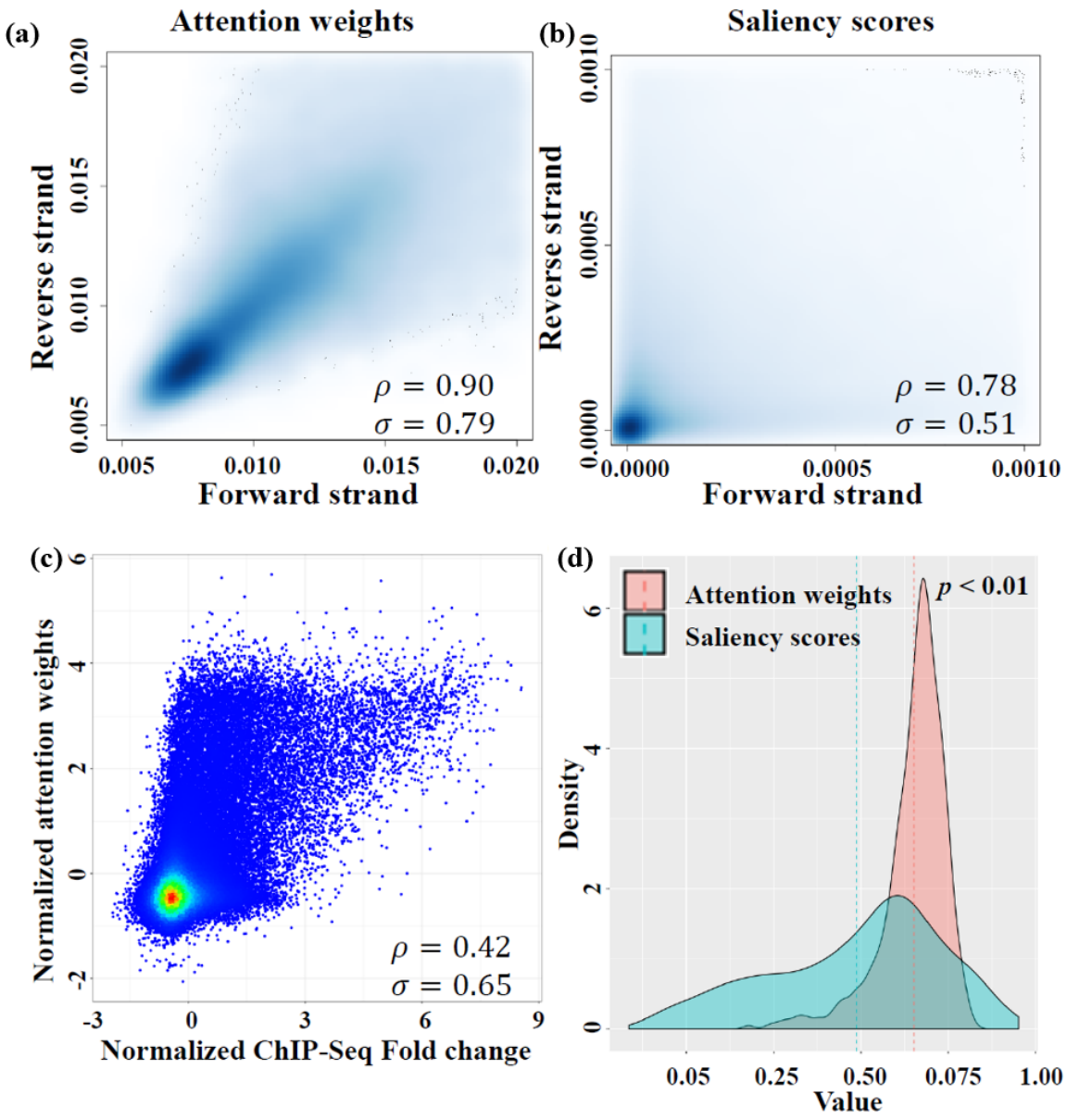

Figure 2.7: Analysis of attention weights and saliency scores. (a) Scatterplot of attention weights from positive strand and reverse strand. (b) Scatterplot of saliency scores from positive strand and reverse strand. (c) Scatterplot of ChIP-Seq fold change and mean attention weights from both strands. Z-score transformation is applied to both axes. (d) Distribution of the correlation between attention weights/saliency scores and ChIP-Seq fold change. The dashed line represents the mean of each group. The p-value is calculated using the Wilcoxon signed-rank test. The attention weights and saliency scores on the reverse complement strand are reversed before plotting. ρ: Spearman Correlation Coefficient, σ: Pearson Correlation Coefficient. The correlation between normalized ChIP-Seq Fold change and normalized saliency scores is 0.40 (Spearman) and 0.49 (Pearson)

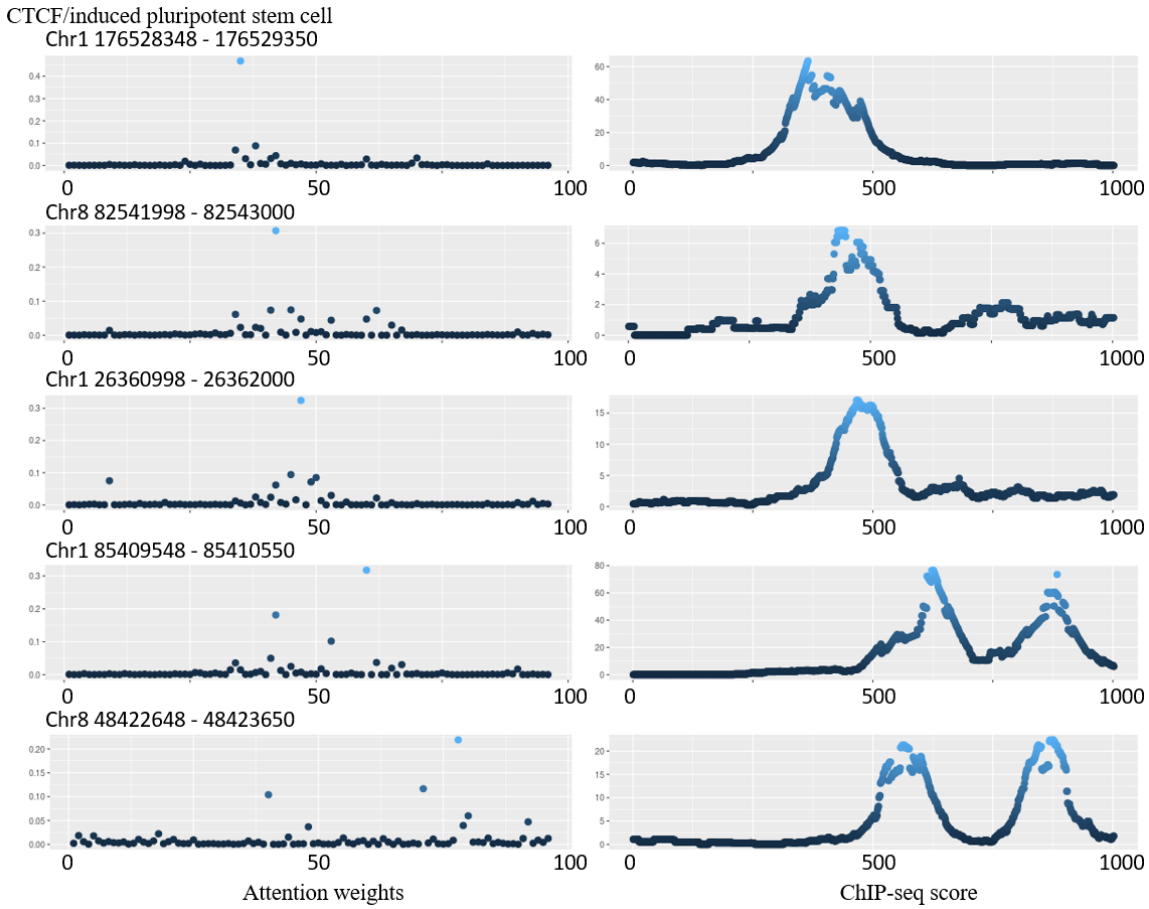

Figure 2.8: For each genomic region, the figure on the left represents the attention weights and the figure on the right represents the enrichment of fold changes in ChIP-seq BigWig file in the same region. Since the lengths of attention weights are reduced by the convolution and pooling layers, their lengths are less than the fold change values. Thus, the plots are aligned on the X-axis to represent the relative position of fold change and averaged attention weights. The attention scores are extracted from both single and pairwise models. For single module, the dimension of attention scores is $L \times d_{model}$ channel that contains the highest attention weight is used. Similarly, for pairwise module, the dimension of attention scores is $L \times L$, and the row sum of these weights are always 1. As a result, we use the row that contains the highest attention weight.

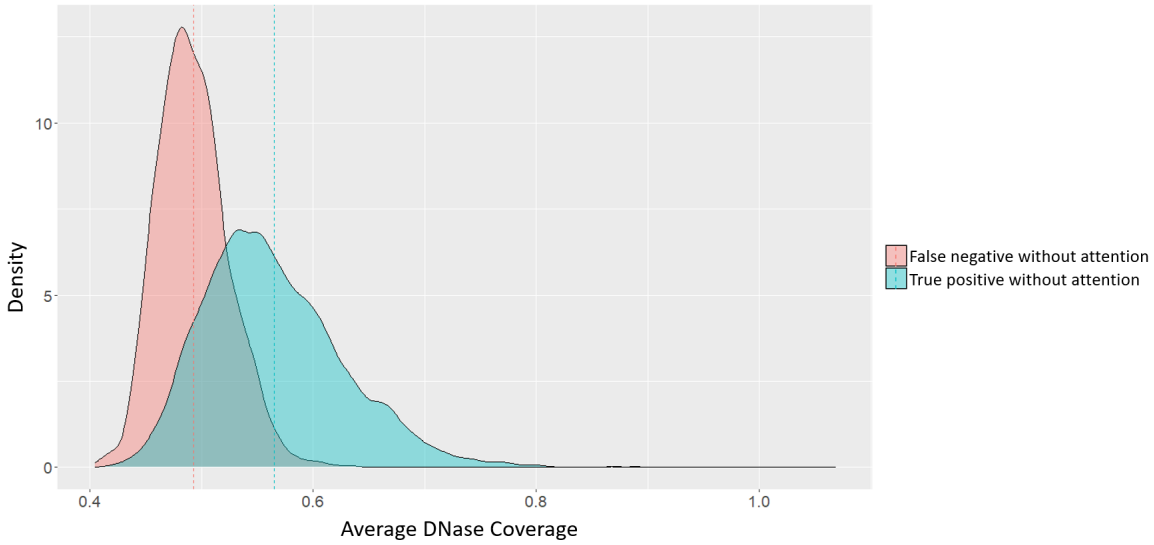

Figure 2.9: Distribution of average normalized DNase coverage values of different regions with the inputs of JUND. The predictions from both models with and without attention from our training are evaluated by the true positive labels. Then the average normalized DNase coverage is calculated based on bins classified differently by the two models.

that are only recognized by attention models generally have a smaller DNase coverage than those recognized by both models. This observation indicates that the predictive improvements of attention models may result from focusing on more informative DNase-Seq coverage values while ignoring irrelevant regions in negative samples.

2.4.4 Motif detection over high attention scores regions

For those positive samples without distinct DNase-Seq peaks, the patterns of genomic sequences are critical information for successful prediction. To test the ability of attention weights to recognize motifs that contribute to binding events from the genomic sequences, we use an approach similar to DeepBind [13]. For the model trained for each TF, we first acquire the coordinates on the relative positions of maximum column sum of the attention weights from all positive bins in test datasets and extract a subsequence with a length of 20 bp around each coordinate. To exclude samples that can be easily classified from patterns of DNase-Seq signal, we only select positive bins

that have no significant coverage peaks (ratio between the highest score and average scores < 15). Then we run FIMO [69] to detect known motifs relevant to the TF of the model in the JASPAR database [70]. From the extracted subsequences, we discover motif MA0139.1 (CTCF) in the prediction for CTCF/induced pluripotent cell and MA0148.4 (FOXA1) in the prediction for FOXA1/liver cell. Figure 7a and b show the comparison between the sequence logo of the motif rebuilt from the subsequences and the actual known motifs. We also plot the attention scores of the samples that contain these subsequences (Figure 2.10c, f) and the relative location of the regions with detected motifs in FIMO (Figure 2.10d, g). Furthermore, we show that these maximum attention weights do not come from the DNase-Seq peaks near the motif regions by coincidence since no similar pattern is detected from the normalized DNase scores in the same regions (Figure 2.10e, h). We illustrate the similar trends found in the single attention module in Figure 2.11.

2.5 Conclusions

In this study, we propose a new tool (DeepGRN) that incorporates the attention mechanism with the CNNs-RNNs based architecture. The result shows that the performances of our models are competitive with the top 4 methods in the Challenge leaderboard. We demonstrate that the attention modules in our model help to interpret how critical patterns from different types of input features are recognized. The usage of DeepGRN are described in Appendix A.

The attention mechanism is attractive in various machine learning studies and has achieved superior performance in image caption generation and natural language processing tasks [65, 71]. Recurrent neural network models with attention mechanism are particularly good at tasks with long-range dependency in input data. Inspired by these works, we introduce the attention mechanism to DNN models for TF binding

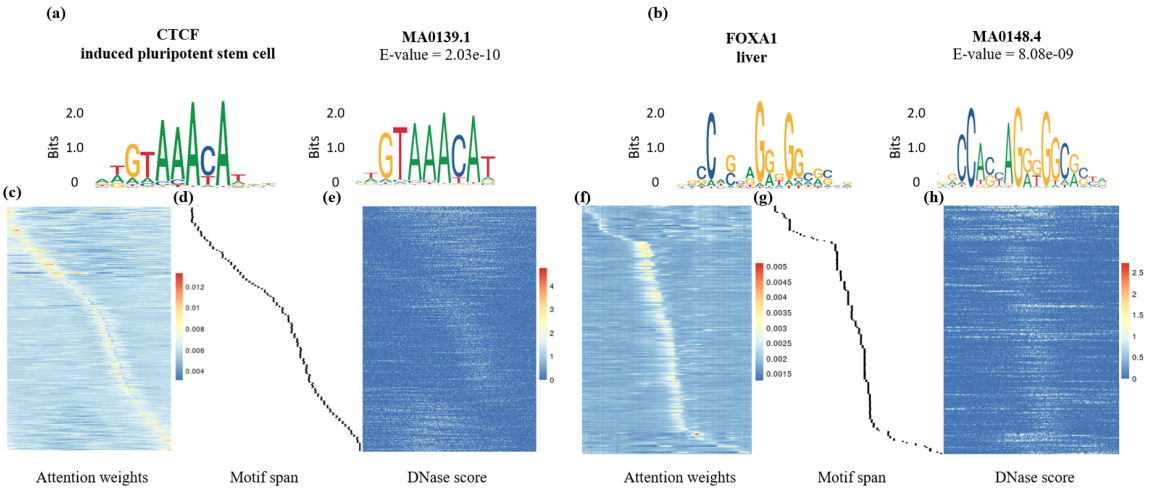

Figure 2.10: Comparisons of known motifs and matching motifs learned by pairwise attention module in CTCF and FOXA1. (a) Sequence logo built from subsequences detected in CTCF/induced pluripotent cell prediction (left) and motif MA0139.1/ CTCF (right). (b) The attention scores of the samples selected from CTCF/induced pluripotent cell prediction with hits of MA0139.1/ CTCF in FIMO. (c)The relative positions of the detected motifs in the same region of (b). (d) The normalized DNase-Seq scores in the same region of (b). (e) Sequence logo built from subsequences detected in FOXA1/liver cell prediction (left) and motif MA0148.4/ FOXA1 (right). (f) The attention scores of the samples selected from FOXA1/liver cell prediction with hits of MA0148.4/ FOXA1 in FIMO. (g) The relative positions of the detected motifs in the same region of (f). (h) The normalized DNase-Seq scores in the same region of (f)

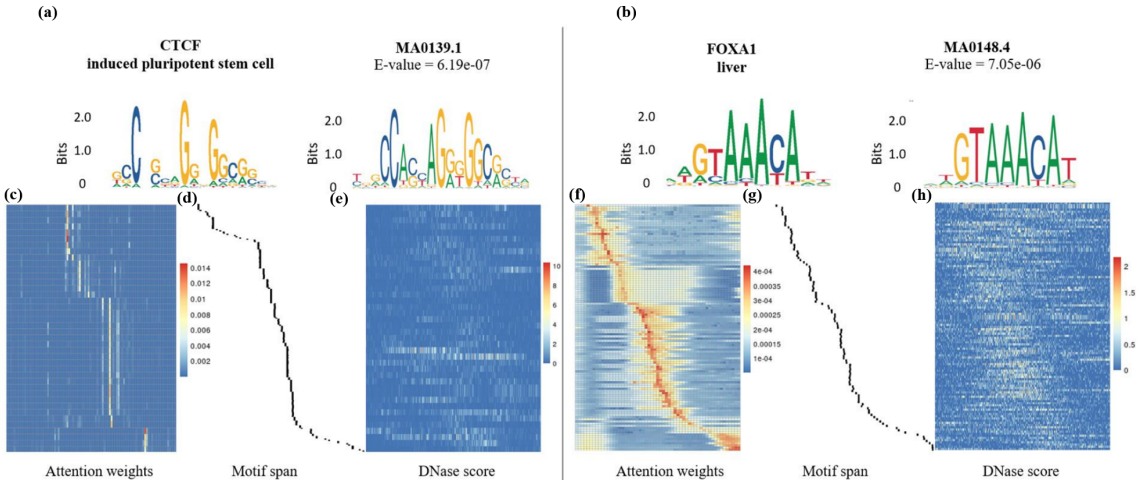

Figure 2.11: Comparisons of known motifs and matching motifs learned by pairwise attention module in CTCF and FOXA1. (a) Sequence logo built from subsequences detected in CTCF/induced pluripotent cell prediction (left) and motif MA0139.1/ CTCF (right). (b) The attention scores of the samples selected from CTCF/induced pluripotent cell prediction with hits of MA0139.1/ CTCF in FIMO. (c)The relative positions of the detected motifs in the same region of (b). (d) The normalized DNase-Seq scores in the same region of (b). (e) Sequence logo built from subsequences detected in FOXA1/liver cell prediction (left) and motif MA0148.4/ FOXA1 (right). (f) The attention scores of the samples selected from FOXA1/liver cell prediction with hits of MA0148.4/ FOXA1 in FIMO. (g) The relative positions of the detected motifs in the same region of (f). (h) The normalized DNase-Seq scores in the same region of (f)

site prediction.

The benchmark result using ENCODE-DREAM Challenge datasets shows that the performances of our model are competitive with the current state-of-the-art methods. It is worth mentioning that the DNase-Seq scores are the most critical feature in the attention mechanism from our experiments according to the feature importance analysis. Many prediction tools for binding site prediction before the challenge, such as DeepBind or TFImpute, are not able to utilize the DNase-Seq data and are not as suitable as the four methods that we used for benchmarking in this study. However, the methods we benchmarked in this study share the similar concepts with these existing tools (For example, FactorNet is built with similar architecture as the TFImpute with additional support for the DNase-Seq data) and may reflect the potential of them using the same set of features.

The attention weights learned by the models provide an alternative approach to exploring the dependencies between input and output other than saliency maps. By comparing true ChIP-Seq fold change peaks with attention weights, we show how attention weights shift when the fold change peaks move along the DNA sequence. We also demonstrate that our attention model has the ability to learn from known motifs related to specific TFs.

Due to the rules of the DREAM Challenge, we only use very limited types of features in this work. However, if more types of features (such as sequence conservation or epigenetic modifications) are available, they can possibly be transformed into sequential formats and may further improve the prediction performance through our attention architecture. The attention mechanism itself is also evolving rapidly. For example, the multi-head attention introduced by Transformer [65] showed that high-level features could be learned by attention without relying on any recurrent or convolution layers. We expect that better prediction for the TF binding may also be benefited from these novel deep learning architectures in both accuracy and efficacy.

Chapter 3

GNET2: an R package for constructing gene regulatory networks from transcriptomic data

3.1 Abstract

The GNET software is designed to build the gene regulatory network from transcriptomics gene expression profiles. It reconstructs gene regulatory modules consisting of transcription factors and their target genes based on a probabilistic graphical model. The data preprocessing, model construction, and visualization function modules for the original GNET software are developed on different programming platforms, which makes it inconvenient for application. Also, its interface of data exchange between different modules relies on constantly saving and loading from disk storage, resulting in low-performance efficiency. These issues motivate us to develop a fully integrated and automated framework with better performance and operability.

In this chapter, we present the new implementation of GNET2 as an integrated R package. It achieves a significant improvement in computational performance over the previous version. Also, the new implementation provides more flexibility for

parameter initialization and regulatory module construction while preserving the core iterative modeling process of the original algorithm. The data exchange interface of the package is now all handled within an R session, with little intervention required during the analysis process. Given the growing demand for regulatory network module inference methods from transcriptome data, GNET2 provides a convenient option for gene regulatory network inference in large datasets. The source code of GNET2 is available at https://github.com/jianlin-cheng/GNET2.

3.2 Introduction

Understanding the interactions between transcription factors (TFs) and their target genes is a crucial step to reconstruct the gene regulatory networks (GRNs) systematically. Currently, there are several common types of representation for GRNs. In directed sparse graphs, all transcription factors and their targets are connected by interaction edges [72]. In module networks, different sets of genes that are regulated by a shared regulation program form different regulatory modules [73]. Many algorithms have been implemented to infer GRNs from gene expression data, including regression-based method [74, 75] and models based on information theoretic framework [23, 24, 76]. The Gene Network Estimation Tool (GNET) [27, 28] is a module-based network method for GRN reconstruction and has been applied to infer the regulatory interactions among genes involved in various biological processes, such as regulations of estrogens [29] and nodulation [27]. With the recent development of Next-Generation Sequencing technologies, transcriptomics gene expression data become much feasible to acquire. The growing need for the downstream analysis of transcriptomics data demands more efficient and accurate tools for GRN construction.

In this work, we introduce a new implementation of the GNET algorithm as an R package. Furthermore, we add support for a gradient boosting based module

initialization, as a complement to the previous K-means version. We benchmark the performance of the new implementation (GNET2) together with the previous version of GNET on a large Aneuploidy Arabidopsis RNA-Seq datasets. We developed the scoring and visualization of network confidence based on the coherence among different experiment conditions, which allow users to acquire more accurate results from customized data.

3.3 Materials and Methods

In GNET2, a gene regulatory module consists of two components: a regulatory tree built from the gene expression profiles and a set of target genes regulated by the tree. The regulatory tree is a binary decision tree with each internal node representing a regulator (i.e. transcription factor). Different samples from the input data are divided into different branches based on the expression levels of the regulators in the tree nodes. Target genes in samples assigned to the same leaf node are expected to share the similar regulatory pattern and are controlled by the regulators in the tree.

3.3.1 Initialization based on gradient boosting decision tree

To build the decision tree for each module, a subset of target genes is required to decide the best split of the branches. The split with the highest increase in the maximum likelihood estimation for the target genes is selected to be the next divide. Thus, the initial clusters of genes need to be determined before the tree construction. The gradient boosting decision tree(GBDT) approach initialization contains the following steps: First, a GBDT model is constructed for every regulator using all genes other than the regulators as predictors. Then we compute the fractional contribution of each feature to the model based on the total gain (reduction of loss function) of the splits of this predictor and use them as information gained from each predictor.

The contribution of each target gene to each regulator can be represented as a n by p matrix where n is the number of all genes in the expression data expect the regulators and p is the number of the regulators. Then we calculate the Euclidean distances between each row of the contribution matrix and these distances are used to group the target genes by K-means clustering. This approach is better at capturing the interaction between the regulators and their targets than the original K-means method, since the clustering is based on the similarities between the capabilities of the expression of target genes to predict the expression of a regulator, rather than the gene expression level itself.

We use R package XGBoost [77] to implement the GBDT-based initialization in GNET2. During the initialization step, a GBDT model is constructed for each target gene using the expression level of all regulatory genes as potential predictors. For a target gene, suppose its expression level at condition i is y_i, then the expression level of all regulatory genes can be represented as a vector x_i with length equal to the total number of regulatory genes. The predict value of the GBDT model is computed as:

$$\widehat{y_i} = \sum_{k=1}^{K} f_k\left(x_i\right) \tag{3.1}$$

Here $f_k \in F$, is one configuration from the space of all regression trees and K is the total number of trees in the model. Suppose $l\left(y_i, \widehat{y_i}\right)$ is the loss function between true expression y_i and predicted value $\widehat{y_i}$ and $\Omega\left(f_k\right)$ is the regularization function to penalize the complexity of tree f_k, then the objective function for all n conditions can be written as:

$$Obj \;=\; \sum_{i=1}^{n} l\left(y_i, \widehat{y_i}\right) \;+\; \sum_{k=1}^{K} \Omega\left(f_k\right) \tag{3.2}$$

We define the regularization function $\Omega\left(f_k\right)$ for a tree f_k as:

$$\Omega\left(f_k\right) \;=\; \gamma T \;+\; \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.3}$$

Here T is the total number of leaves for all trees in the model, w_j is the value of leaf j. γ and λ are predefined coefficients.

At the first step, a tree that gives constant (e.g., 0.5) prediction values is used as the first tree. During each iteration, a new tree is trained and added to the model. Since $\widehat{y_i} = \sum_{k=1}^{K} f_k(x_i)$, so at step t we have:

$$\widehat{y_i}^{(0)} = 0\widehat{y_i}^{(t)} = \widehat{y_i}^{(t-1)} + f_t(x_i) \tag{3.4}$$

We use the square loss for the regression trees, and the objective function at step t can be rewritten as:

$$Obj = \sum_{i=1}^{n} \left[\left(\widehat{y_i}^{(t)} - y_i \right)^2 \right] + \Omega(f_t) + constant \tag{3.5}$$

Here, the constant is the regularization penalty for all trees before t, which is independent of tree f_t. If we define a single term in the beginning summation part of the objective function as a function of $\widehat{y_i}^{(t)}$, it will become:

$$F_i\left(\widehat{y_i}^{(t)}\right) = \left(\widehat{y_i}^{(t)} - y_i \right)^2 \tag{3.6}$$

It can be approximated as the second-order Tyler expansion near $F_i\left(\widehat{y_i}^{(t-1)}\right)$

$$F_i\left(\widehat{y_i}^{(t)}\right) = F_i\left(\widehat{y_i}^{(t-1)} + f_t(x_i)\right) \cong F_i\left(\widehat{y_i}^{(t-1)}\right) + F_i'\left(\widehat{y_i}^{(t-1)}\right) f_t(x_i) + \frac{F_i''\left(\widehat{y_i}^{(t-1)}\right)}{2!} {f_t}^2(x_i) \tag{3.7}$$

Let $g_i = F_i'\left(\widehat{y_i}^{(t-1)}\right) = 2\left(\widehat{y_i}^{(t-1)} - y_i\right)$, $h_i = F_i''\left(\widehat{y_i}^{(t-1)}\right) = 2$. Since $F_i\left(\widehat{y_i}^{(t-1)}\right)$ is independent of tree f_t and can be treated as a constant, then the objective function at step t can be approximated as:

$$Obj^{(t)} \cong \sum_{i=1}^{n} \left(g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \Omega(f_t) + constant \tag{3.8}$$

$$= \sum_{i=1}^{n} \left(g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 + constant \quad (3.9)$$

$$= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T + constant \quad (3.10)$$

Here, I_j is the indicator if a gene has been assigned to leaf j. Let $G_j = \sum_{i \in I_j} g_i$, $H_j = \sum_{i \in I_j} h_i$, we can find the w_j that minimize the objective function by letting the partial derivative $\frac{\partial Obj}{\partial w_j} = 0$:

$$Obj^{(t)} = \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T + constant \quad (3.11)$$

$$w_j^* = \text{argmax } Obj^{(t)} = -\frac{G_j}{H_j + \lambda} \quad (3.12)$$

$$Obj^{(t)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T + constant \quad (3.13)$$

When growing a tree, the algorithm performs a linear scan on the sorted expression values of each condition for every regulatory gene. The best split/feature is determined by the split that brings the largest reduction of loss when adding that split to the tree:

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \quad (3.14)$$

Here L, R represent the left and right leaf, respectively. For target gene i, the importance score is represented as the fractional contribution of the regulatory genes to the model based on the total gain from all splits with this regulatory gene. Thus, the importance score for all target genes can be represented as a n by p matrix where n is the number of target genes, and p is the number of regulator genes. These scores are then used for determining the initial clustering of the target genes through K-means clustering.

3.3.2 Iterative module inference process

After assigning the genes to different modules, an iterative gene regulatory tree inference and target gene re-assignment process are performed to construct the regulatory tree and update the target genes for each regulatory module. Figure 3.1a and b illustrates one regulatory module obtained by GNET2.

Acquiring potential upstream regulators

GNET2 requires a list of regulatory genes to infer the regulatory pattern from gene expression profile. In our analysis on the Aneuploidy Arabidopsis RNA-Seq dataset, the regulatory genes are obtained from the transcription factors (TFs) curated in the PlantTFDB database [78]. For most common organisms, the currently identified transcription factors can be acquired in publicly available databases. For example, the AnimalTFDB [79] contains 80,060 TFs genes from 97 animal genomes. The PlantTFDB 4.0 contains 320,370 TFs from 165 species. If no available information about TF genes for the input gene expression dataset, the regulatory genes can be inferred from software that identifies transcription factors. For example, the binding analysis for regulation of transcription (BART) (https://faculty.virginia.edu/zanglab/bart/index.htm) can identify TFs whose genomic binding profile correlates with a query cis-regulatory profile derived from ChIP-seq datasets available in the public domain [80].

Gene regulatory tree inference

The expression level of each regulator gene is assigned into three categories (low, neutral, and high). Whenever a new split is going to be added into the regulatory tree, we tested the two possible splits (low+neutral vs. high, or low vs. neutral+high) for all remaining regulator genes. The Maximum Likelihood Estimation to estimate parameters of a Gaussian distribution for each possible split is computed for both left

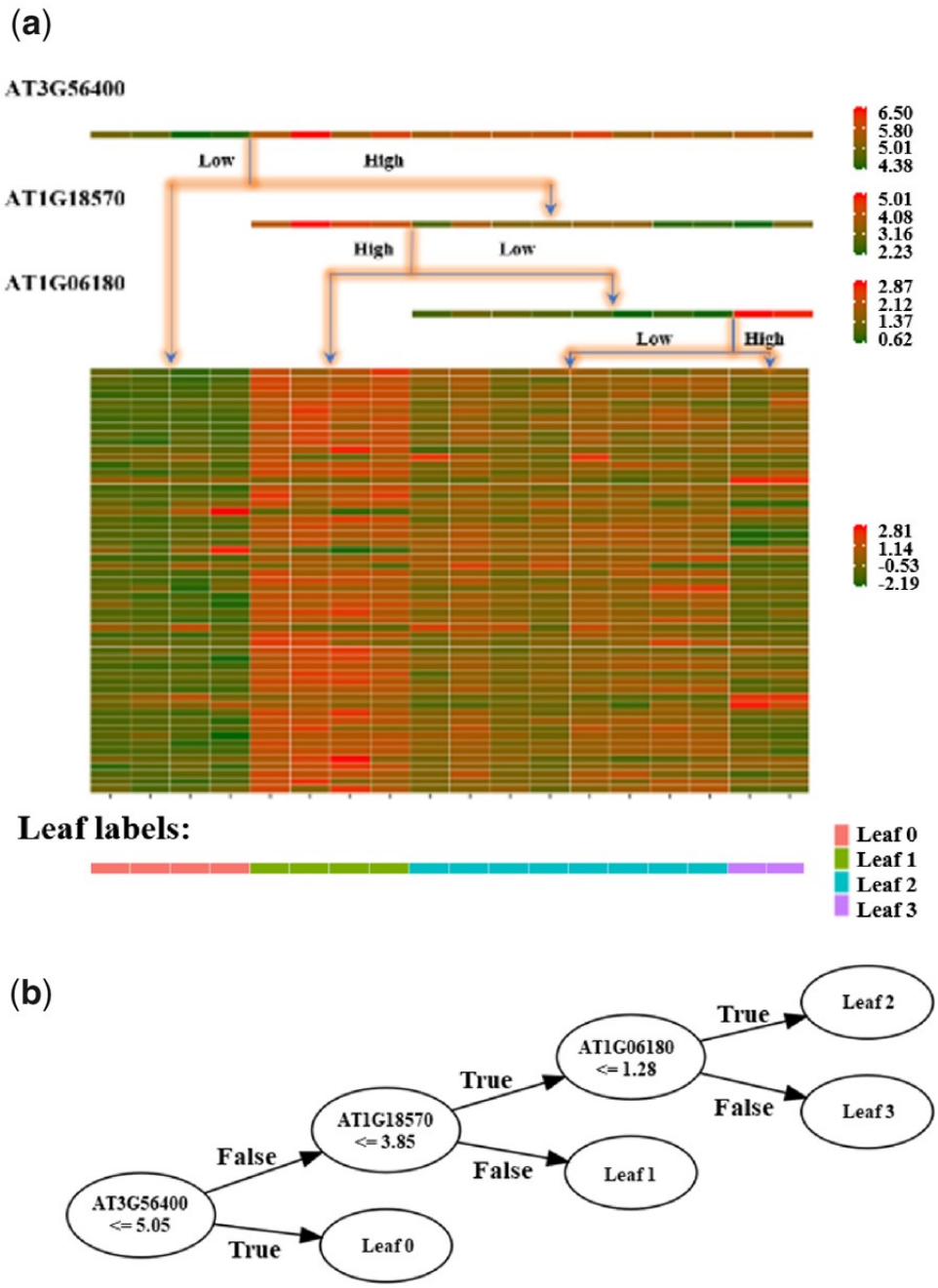

Figure 3.1: Illustration of the results generated by GNET2. (a) One of the regulatory modules generated from the Arabidopsis dataset. The top color bars indicate how samples are separated according to the expression levels of the regulators. The heatmap indicates the expression pattern for the target genes in the module. The bottom color bar indicates the groups of samples that are in the same leaf node of the regulatory tree. (b) Plot of the regulatory tree shown in (a). The unit of numbers in the figure is log_2 RPKM

and right child nodes and the parent node, which can be computed as:

$$L_m\left(I_L, I_R\right) = \sum_{n=1}^{N}\left(l\left(g_n, I_L\right) + l\left(g_n, I_R\right) - l\left(g_n, I_{L+R}\right)\right) \tag{3.15}$$

Here N is the number of genes that belong to module m. I_L, I_R are the indicators for conditions that belong to the left and right of the split. $l\left(g_n, I\right)$ is the sum of Gaussian log-likelihood for gene n of conditions in I, and can be computed as:

$$l\left(g_n, I\right) = \sum_{i \in I}\left(-\frac{\left(g_{n,i} - \mu_{n,I}\right)^2}{2\pi\sigma_{n,I}^2} - ln\sqrt{2\pi}\sigma_{n,I}\right) \tag{3.16}$$

Here $\mu_{n,I}$ and $\sigma_{n,I}$ are the mean and standard deviation for expression the level of g_n in all conditions in I. The regulator gene that has the highest likelihood gain will be chosen for the cutoff for the next split. The tree will keep splitting until the numbers of conditions in all leaf nodes are smaller than 3, or the predefined maximum depth of the regulatory tree is reached.

Target genes re-assignment

After a gene regulatory tree is built for every target gene group, a gene re-assignment procedure is performed to assign all target genes into different groups based on the updated regulatory tree of the group. The Gaussian log-likelihood is used to determine the best group for each target gene.

$$Score_T\left(g\right) = \sum_{l=1}^{L}\sum_{i \in I} -\frac{\left(g_i - \mu_L\right)^2}{2\pi\sigma_L^2} - ln\sqrt{2\pi}\sigma_L \tag{3.17}$$

Where L is the number of leaf nodes for regulatory tree T, I is the indicator for conditions that belong to leaf n. g_i is the gene expression level under condition i. μ_L and σ_L are the mean and standard deviation for the gene expression levels belong to leaf node l, respectively. Each gene is then assigned to the regulatory tree that

gives the highest likelihood score. The process of gene regulatory tree inference and target genes re-assignment above is performed iteratively until the assignment of genes remains unchanged for two consecutive iterations, or the maximum number of iterations has been reached.

Missing values imputation

When missing values are present in the input data, we recommend the users to apply missing value imputation techniques before running GNET2. Common common missing value imputation algorithms, such as k-nearest neighbors (KNN) [81] and quantile regression imputation of left-censored data (QRILC) [82] can easily handle data with larges sizes.

3.3.3 Module scoring functions

Module and interaction score evaluation

The quality of regulatory module is evaluated from the weighted average of within-group correlations:

$$Q_i \;=\; \frac{1}{N}\sum_{m=1}^{M} n_m p_m \tag{3.18}$$

Where Q_i is the quality score for module i, N is the total number of conditions, M is the total number of leaf nodes for module i, n_m is the number of conditions that belong to leaf node m, and p_m is the average Pearson correlation coefficient between every pair of conditions that belong to leaf node m. For the score of interactions between regulatory gene i and target gene j, we define an empirical score as:

$$P_{i,j} \;=\; Q_i - \;2 * \max(0, \ln\left(1 - {p_{i,j}}^2\right)) \tag{3.19}$$

Where $p_{i,j}$is the Spearman correlation coefficient between the expression values of i and j.

3.3.4 Incorporate experimental setup information

When additional information about samples is available (such as different experiment conditions), GNET2 can utilize the information from user-defined sample labels to measure the coherence between known sample information and the regulatory pattern in prediction. Suppose a gene expression dataset contains measurements from multiple conditions, and the user is interested in discovering the change of regulatory pattern caused by different experimental treatments. Since samples that are clustered in the same leaf node of the regulatory tree are predicted to share the same regulatory patterns, those modules that have a clustering of samples that shares a similar structure of the sample conditions are considered to be more relevant to the experimental conditions. For example, modules with most biological replicates under the same leaf node indicate high relevance between the changes in the expression level of regulators and experimental conditions. Thus, we introduced the following two scoring functions for in GNET2 to characterize the similarity between the clusters based on sample information and the clusters from each functional module. For both scoring functions, higher scores indicate better coherence between known sample information and the regulatory pattern in each module GNET2 predicted.

Categorical labels

For categorical labels, suppose there are n samples in the input, and they can be grouped as $X = X_1, X_2, \ldots, X_r$ with each cluster X_i represents samples of the same condition (e.g. replicates). If the regulatory tree for module m split the samples into clusters $Y = Y_1, Y_2, \ldots, Y_r$ with each cluster Y_i represents samples in the same

leaf node. The similarity between sample information and regulatory patterns can be characterized by the Adjusted Rand index (Rand, 1971).

Ordinal labels

For ordinal labels such as the dosage of drug intervention or different time points, it is important to measure the pairwise distance between different labels as well as the replicates proposed by the user. For example, suppose there are n samples in the input, and they can be grouped as $X = X_1, X_2, \ldots, X_r$ where each label represents a time point after some treatment. We expect the neighboring conditions of X_i (e.g. X_{i-1} and X_{i+1}) should have a higher similarity to X_i than other samples. In addition, we assume that replicates should always have the shortest distance between each other, regardless of the type of experimental conditions. To acquire the similarity between clusters and ordinal labels, we design a similarity score inspired by the Stochastic Neighbor Embedding (SNE), which can be computed by the following steps:

1. Rank the ordinal labels in sequential order, use the ranks as the numeric label for each sample. The index of leaf nodes (from left to right) is used to represent the clusters of under the same leaves in the regulatory tree of each module. The order of labels is trivial since we are only interested in the relative pairwise distance between samples.

2. Compute the Euclidean distance between samples using the ranks of the ordinal labels, and the Euclidian distance between samples using the index of leaf nodes.

3. Convert the two distances with Gaussian Probability density. Suppose x_{ij} is the distance between sample i and j, the normalized distance is computed as:

$$d_{ij} = \frac{\exp\left(-||x_{ij}||^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-||x_{ik}||^2/2\sigma_i^2\right)} \tag{3.20}$$

4. The Kullback–Leibler (K-L) divergence between the normalized distance computed from ranks of the ordinal labels of the samples (p_{ij}) and the normalized distance computed from the index of leaf nodes of the samples (q_{ij}): $KL(P||Q) = \sum_{i \neq j} p_{ij} log \frac{p_{ij}}{q_{ij}}$. The final similarity score is defined as the negative z score normalized K-L divergence.

3.3.5 Benchmark experiments setup

Synthetic transcriptional networks dataset

We benchmark the algorithms using synthetic transcriptional networks proposed by the SynTReN (Synthetic Transcriptional Regulatory Networks) algorithm [83]. In brief, network topologies are generated by selecting subnetworks from previously described *E. coli* [84] and *S. cerevisiae* [85] regulatory networks with cluster addition approach. Gene expression profile is then inferenced based on the interaction kinetics are modeled by Michaelis-Menten and Hill kinetics. We use SynTReN to generate networks consist of 100 genes and corresponding gene expression datasets with 100 experiments for both *E. coli* and *S. cerevisiae* with default parameters. The list of genes that are external inputs to the network in SynTReN can be treated as regulatory genes and the performance of the predictions are evaluated based on the interactions between these regulatory genes and all other genes in the network.

Aneuploidy Arabidopsis RNA-Seq dataset

We benchmarked the performance of GNET2 in terms of speed and quality in the Aneuploidy Arabidopsis RNA-Seq dataset (GSE79676) [86]. The details for sequencing, mapping and RPKM calculations are described in the "RNA-Seq Library Construction" section of the paper (https://www.pnas.org/content/115/48/E11321). For benchmarking purposes, we select genes with the top 20% highest variances, resulting

in a total of 6712 genes for regulatory module construction. We use a list of 342 transcription factors that both exist in the filtered expression data and the list obtained from PlantTFDB 4.0 [78] as input regulators.

Network inference algorithm in comparison

We select four popular network inference algorithm that are available as R package in Bioconductor for performance comparison:

The Maximum Relevance/Minimum Redundancy network (MRNET) algorithm [76] is a network inference strategy based on feature selection using the maximum relevance/minimum redundancy approach. For gene i and its potential target j, this algorithm aims to maximize the difference between the mutual information between the two genes (maximum relevance) and the average mutual information with all interactions previously introduced to gene i (minimum redundancy).

The Context Likelihood of Relatedness (CLR) algorithm [23] computes a modified mutual information $I(X_i; X_j)$ for each pair of genes and derives a score related to the empirical distribution of the MI values. Both the sample mean and standard deviation of the empirical distribution of the mutual information between all pair of genes.

The Algorithm for the Reconstruction of Accurate Cellular Networks (ARACNE) software [24] takes Data Processing Inequality into consideration during the network pruning procedure. After the pairwise interaction score is retrieved from the mutual information, the edge with lowest score of each triplet is interpreted as an indirect interaction if its difference with the second largest interaction in the triplet is larger than a threshold.

The GENIE3 (GEne Network Inference with Ensemble of trees) software [87], unlike the three relevance network-based approach described above, perform variable selection with ensembles of regression trees. In each of the regression tasks, the expression pattern of the target gene is predicted from the expression patterns of all the

other potential regulatory genes. The importance of each feature (regulatory gene) is taken as an indication of a putative regulatory relation and is then aggregated to generate a ranking of interactions for network reconstruction. The initial functional group clustering with gradient boosting in GNET2 is inspired from this feature selection strategy in GENIE3.

We use ROC (Receiver Operating Characteristic) curves to characterize the performance of different algorithms. The ROC curve is the TPR (true positive rate) vs. FPR (Number of false positives/Number of all positives) for a binary classifier system along a series of thresholds. Points above the diagonal line represent predictions with high TPR and low FPR, which indicate accurate classification results. Thus, the area under the ROC curve (aucROC) can be used to access the overall quality of different methods without setting a fixed cutoff. Better inference algorithms will have higher the aucROC in general. A total random guess will have aucROC of 0.5 while the perfect prediction will result in aucROC of 1.

3.4 Results

3.4.1 Assessment of prediction quality

The accuracy for the prediction of the synthetic transcriptional networks with different software are listed in Table 3.1 as aucROC. The benchmark results indicate that the performance of GNET2 with GBDT-based initialization is consistently better than the K-means-based initialization. Also, the inference with GBDT-based initialization is comparable to other widely used methods as it ranks the third place on the *E. coli* dataset and the first place on the *S. cerevisiae* dataset. The plots of the ROC curves in the two synthetic transcriptional regulatory networks are shown in Figure 3.2.

Network	ARACNe	MRNET	CLR	GENIE3	GNET2-K	GNET2-G
E. coli	0.8089	0.8407	0.9145	**0. 9593**	0. 7664	0. 8817
S. cerevisiae	0.8138	0.6759	0.7529	0. 8285	0. 8545	**0. 8739**
Arabidopsis	0.5664	0.5976	0.595	0.5874	0. 6187	**0. 6721**

Table 3.1: Evaluation of network inference results. The values are the aucROC scores of the different approaches (bold: the best results)

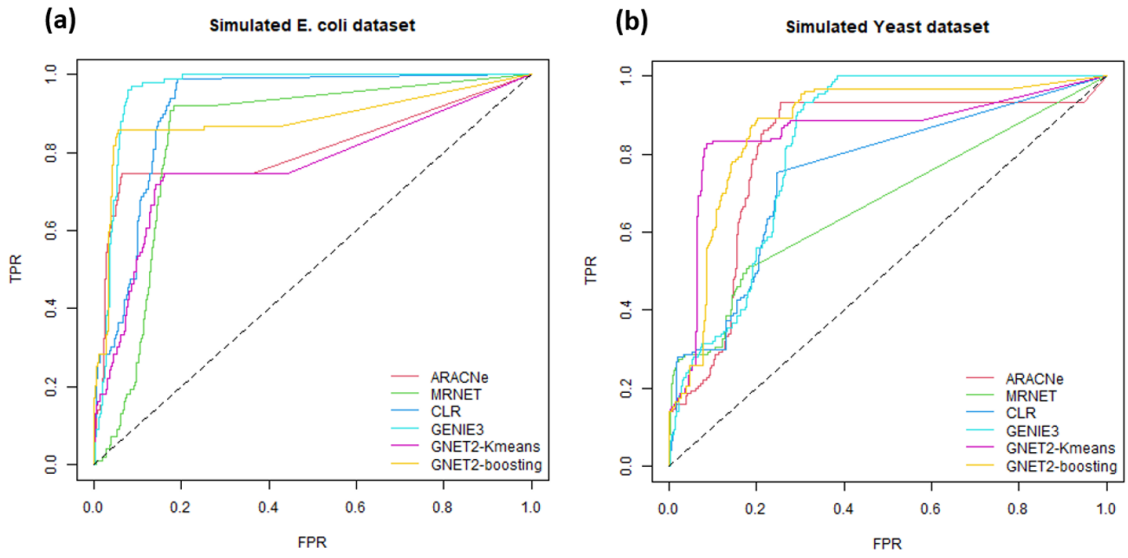

Figure 3.2: ROC plots for 100 samples generated from SynTReN. (a) Performance with gene expression simulated on the *E. coli* regulatory network. (b) Performance with gene expression simulated on the *S. cerevisiae* regulatory network.

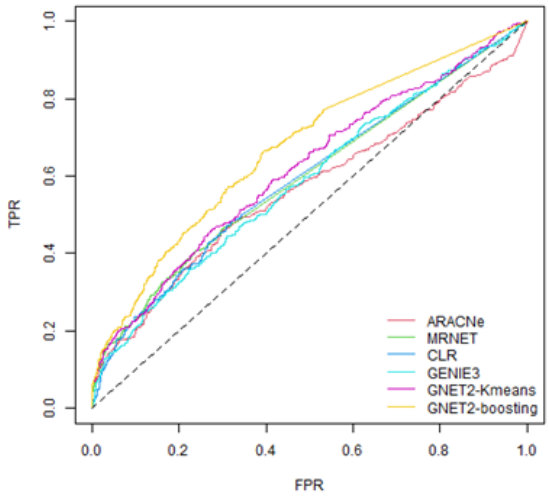

Figure 3.3: ROC plots for 18 samples generated on the Aneuploidy Arabidopsis RNA-Seq dataset.

For Aneuploidy Arabidopsis RNA-Seq dataset, we use the first 10 genes that exist in both of the filtered gene expression data and the TF list as regulator genes. Then we retrieve all interactions between the regulator genes and all other genes in the expression data from the STRING database [88] with the cutoff of the combined score set to 400. This result in a network consists of 448 interactions and is used as ground truth for evaluation, which are the genes kept in the input data for evaluation. The results for transcriptional networks derived from experimental data are shown in Table 3.1. GNET2 with GBDT-based initialization achieves the highest aucROC among all benchmarked methods. However, the performances of all predictions are decreased probably due to the lack of complete information for the entire gene regulatory network. The plot of the ROC curves and shown in Figure 3.3.

3.4.2 Impact of sample numbers and initial group sizes

To evaluate how well does the tool perform for different number of samples, we performed benchmark on simulated *E. coli* and *S. cerevisiae* data with different numbers

of samples (20,40,60,80,100). We found that 40 samples will be sufficient to achieve the optimal performance with aucROC (Figure 3.4a). While the performance in the experiment with fewer samples is decreased, GNET2 can still output predictions with reasonable predictions. However, when there are too few samples provided (<10), the construction of regulatory trees may fail because of lacking meaningful branch splits.

To benchmark the impact of setting different numbers of clusters during group initialization, we run GNET2 on simulated expression data synthesized from *E. coli* networks with different sizes. (node number =100 or 500). Our strategy is to choose initial group number according to predefined average groups sizes (10,20,30,40,50). For example, by choosing average size of 50, the initial group number is 2 for networks with 100 nodes and 10 for networks with 500 nodes. We found that an average size of $leq35$ works well on the small network, while the performance of GNET2 is relatively insensitive to the average size in large networks. (Figure 3.4b) It is worth mentioning that for both K-means and GBDT-based initialization in GNET2, the initial group size is set exactly as the initial group parameter. Instead, it performs clustering with sizes from 2 to the initial group parameter and the cluster number with highest correlation of expression for genes in the same group is used. This can explain the stable performance when using small initial group size (equivalent to set large cluster numbers). However, setting too large cluster numbers may heavily impact the running time for initialization. In practice we use average group size of 25 for synthesized data and 40 for Aneuploidy Arabidopsis data.

3.4.3 Systematic validation on the predicted functional modules

We used the same evaluation strategy in the original GNET benchmarking [27] to systematic interpret the functional modules predicted by GNET2. We focused on assessing the validity of 52 gene regulatory modules whose correlation coefficients were

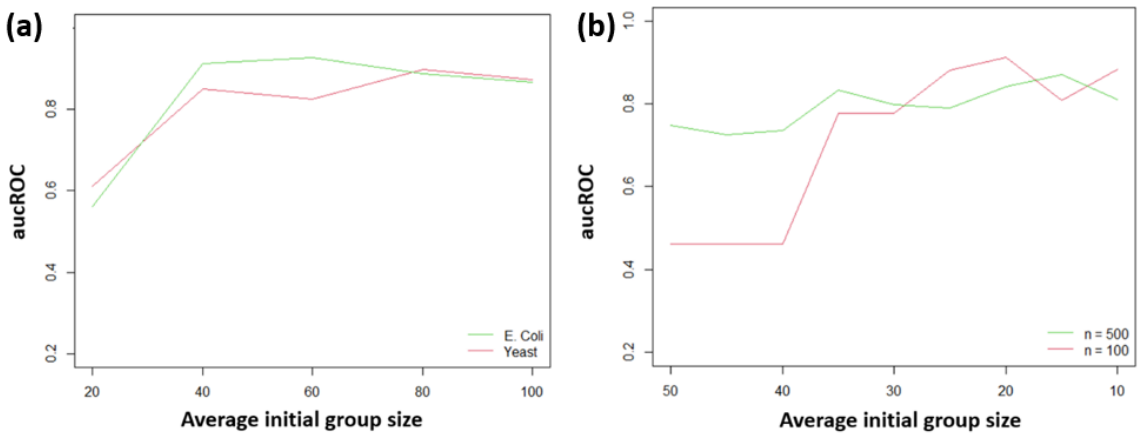

Figure 3.4: Evaluation of the impact of inital group size. a) evaluation with different sample sizes in *E. coli* and *S. cerevisiae* dataset. b) evaluation with different sample sizes in the *E. coli* with different number of total genes (n=100 and 500).

greater than 0.3 from evidence of interaction potential between TFs and predicted downstream genes by STRING database, and literature confirmation of the regulatory function of TFs and the genes in corresponding experimental conditions.

We first examined the validity of the predicted function modules was furthered confirmed by the potential interactions among the predicted TFs and the target genes within the same module from evidence curated in the STRING database. In Table 3.2 and 3.3 we listed the 28 gene regulatory modules that had at least one matching supporting evidence in the STRING database using R package STRINGdb [89]. The types of interactions we selected are co-occurrence, homology, co-expression, experiments, database, and text mining. In the last column we also listed the PubMed ID for the publications in which any regulators and target genes in the module are co-mentioned as literature support.

Index	Co-occurrence	Homology	Co-expression	Experiments
1	0	0	8	0
2	0	1	5	0
3	0	0	3	0
4	0	3	9	0
5	0	0	4	0
6	4	2	63	2
7	0	3	31	0
8	1	1	106	0
9	0	0	8	0
10	0	1	7	0
11	0	11	49	0
12	0	0	5	0
13	1	0	4	0
14	0	0	34	0
15	2	2	217	0
16	0	1	17	0
17	0	2	366	0
18	0	0	13	0
19	2	0	75	0
20	0	0	7	0
21	1	3	57	0
22	0	0	2	0
23	1	3	94	0
24	0	0	45	0
25	0	0	1	0
26	0	0	19	0
27	0	0	6	0
28	0	0	14	0

Table 3.2: Occurrences of each type of interaction between the transcription factors and predicted downstream genes.

Index	Database	Textmining	Literature support
1	0	0	
2	0	4	PMID:22639632
3	0	1	PMID:20463887;PMID:16679424
4	0	2	PMID:21642548;PMID:19529824
5	0	1	
6	0	10	
7	0	0	
8	0	3	PMID:24966866;PMID:21487097;PMID:18805951
9	0	2	PMID:24999812; PMID:24228871
10	0	4	PMID:24688486;PMID:22238427;PMID:16121258
11	1	15	PMID:23390424;PMID:20307264;PMID:19843695
12	0	0	
13	1	2	PMID:22346763;PMID:21733976;PMID:19654206; PMID:16513813;PMID:11500563
14	1	5	PMID:21258004
15	1	22	PMID:21258004
16	0	5	PMID:24148294;PMID:24795732;PMID:24563199; PMID:23185464;PMID:21733976;PMID:19247443; PMID:12068110
17	0	21	PMID:22252389
18	0	3	PMID:23390424;PMID:22544736; PMID:21733976;PMID:19239350
19	2	6	PMID:24106755;PMID:21464308;PMID:18502975; PMID:17420173;PMID:15310842
20	1	2	PMID:24148786;PMID:21897874; PMID:18989364;PMID:15561727
21	0	9	PMID:23185391;PMID:21803941
22	0	1	PMID:19247443;PMID:18684336;PMID:21733976; PMID:20876338;PMID:19482972;PMID:16778081
23	0	18	PMID:23390424;PMID:23300166; PMID:22212120;PMID:21841124; PMID:21586684;PMID:21150289; PMID:20663954;PMID:19948955
24	0	8	
25	0	0	PMID:22927830;PMID:20307264;PMID:19717544; PMID:16126837;PMID:15901827;PMID:15561727
26	0	1	
27	0	3	
28	0	0	

Table 3.3: Evidences from databases and text mining

Experiment	Samples	Total genes	Regulators	Initial modules
1	20,40,60,80,100	200	20	10
2	20	200,400,600,800,1000	10% of total genes	10
3	20	1000	100	10,20,30,40,50
4	20	200	10,20,30,40,50	10

Table 3.4: Input data for running time benchmark. Experiment 1, 2, 3 and 4 are the benchmark for different number of samples sizes, number genes, number of given regulators and initial number of modules, respectively.

3.4.4 Representation of the functional modules predicted by GNET2

3.4.5 Evaluation of running efficiency for GNET2

Since the module inference step is the most time-consuming step in GNET2, we evaluate the running time during module inference with different number of samples sizes, number of genes, initial number of modules, and number of given regulators with GNET, GNET2-Kmeans, GNET2-GBDT. All tests are performed on a platform with Intel Core 6700K (4.0 GHz) and Windows 10 build 19041. The results are measured as the average running time across 5 separate runs. The experiment design of input data for running time benchmark is shown in Table 3.4.

We compared the time consumption (averaged by five separate runs) between GNET, GNET2-Kmeans, and GNET2-GBDT. The results are summarized in Table 3.5. in most cases, the running time requirements increase linearly with the size of the input. However, this trend is not stringently scaled, and this can be explained by the data-dependent termination time for the two main steps during module inference: initial group clustering and regulatory tree reconstruction. Both initialization approaches in GNET2 (K-means and GBDT) do not have a deterministic running time for a fixed size of input data since they may converge before the maximum number of iterations has been reached. The same happens to the regulatory tree reconstruction, as it will early stop when no changes happen during the assignment of target genes

Experiment		Running time (mins)		
		GNET	**GNET2-G**	**GNET2-K**
Samples	20	111.39	8.16	9.00
	40	201.39	16.04	15.55
	60	278.66	21.38	17.32
	80	335.67	28.29	22.87
	100	304.15	33.10	28.30
	200	68.32	8.46	7.87
	400	172.2	25.16	11.73
Number of total genes	600	413.72	39.14	34.06
	800	624.01	53.88	41.85
	1000	674.16	63.55	48.12
	10	88.63	10.35	7.05
	20	137.27	10.88	10.26
Number of regulators	30	136.61	20.28	10.84
	40	256.48	23.54	20.55
	50	285.20	26.35	22.61
	10	75.38	1.34	4.53
	20	19.84	1.84	1.44
Number of initial modules	30	35.69	3.72	3.49
	40	43.01	2.44	3.07
	50	38.11	4.10	2.83

Table 3.5: Comparison of time consumption for GNET, GNET2-Kmeans and GNET2-GBDT.

to different modules.

3.5 Conclusions

In this chapter, we introduce the GNET2 as a powerful tool for inference and visualization of the relationships between regulatory genes and their targets. The re-implementation of GNET2 as an R package makes it much easier for users to deploy and use. Furtermore, GNET2 reconstructs more accurate regulatory networks with the new GBDT-based initialization than GNET. The usage of GNET2 are described in Appendix B.

However, it is worth mentioning that decision tree for a target gene may not be

fully reconstructed if they have complicated regulatory patterns that interfere with different regulators, which may require more than one regulatory tree to characterize. While we expect the regulatory tree can capture those interactions with the most control over the expression of target genes, it is difficult to identify all regulatory paths, especially for those genes with entirely different sets of regulators under different conditions. Our recommendations for such situations is to run the module inference with different subsets of the data under different conditions, as well as run all samples together. The best modules can then be determined by the scoring functions based on the average correlation among the target genes under the same leaf.

Chapter 4

Combination of deep neural network with attention mechanism enhances the explainability of protein contact prediction

4.1 Abstract

Deep learning has emerged as a revolutionary technology for protein residue-residue contact prediction since the 2012 CASP10 competition. Considerable advancements in the predictive power of the deep learning-based contact predictions have been achieved since then. However, little effort has been put into interpreting the black-box deep learning methods. Algorithms that can interpret the relationship between predicted contact maps and the internal mechanism of the deep learning architectures are needed to explore the essential components of contact inference and improve their explainability. In this study, we present an attention-based convolutional neural network for protein contact prediction, which consists of two attention mechanism-based modules: sequence attention and regional attention. Our benchmark results on the

CASP13 free-modeling targets demonstrate that the two attention modules added on top of existing typical deep learning models exhibit a complementary effect that contributes to prediction improvements. More importantly, the inclusion of the attention mechanism provides interpretable patterns that contain useful insights into the key fold-determining residues in proteins. We expect the attention-based model can provide a reliable and practically interpretable technique that helps break the current bottlenecks in explaining deep neural networks for contact prediction. The source code of ATTContact is available at https://github.com/jianlin-cheng/InterpretContactMap.

4.2 Introduction

Prediction of residue-residue contacts in proteins plays a vital role in the computational reconstruction of protein tertiary structure. Recently, advancements in the mathematical and statistical techniques for inter-residue coevolutionary analysis provide essential insights for correlated mutation-based contact prediction, which is now becoming a critical component to generate input features for machine learning contact prediction algorithms. For instance, in the recent 13th Community-Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction (CASP13) contact prediction challenge, significant improvements have been achieved due to the integration of both inter-residue coevolutionary analysis and novel deep learning architectures [35, 31, 36, 37].

A variety of deep learning-based models have been proposed to improve the accuracy of protein contact prediction since deep learning was applied to the problem in 2012 CASP10 experiment [90]. Many of these methods leverage the contact signals derived from the direct coupling analysis (DCA). Most DCA algorithms [91, 92, 34, 33] generate correlated mutation information between residues from multiple sequence alignments (MSAs), which is utilized by the deep convolutional neural networks in

the format of 2D input feature maps. For example, RaptorX-Contact [38], DNCON2 [39], and MetaPSICOV [93] are a few early methods that apply the deep neural network architectures with one or more DCA approaches for contact prediction. The connection between the two techniques underscores the importance of explaining the contribution of patterns in coevolutionary-based features to the deep learning-based predictors.

Despite the great success of deep learning-based models in a variety of tasks, this approach is often treated as black-box function approximators that generate classification results from input features. Since the number of parameters in a network is somewhat proportional to its depth, it is infeasible to extract human-understandable justifications from the inner mechanisms of deep learning without proper strategies. Saliency maps and feature importance scores are widely used approaches for model interpretation in machine learning. However, due to the unique characteristic of contact prediction, these methods involve additional analysis procedures that require far more computational resources than other typical applications. For example, the saliency map for a protein with length L requires $L \times L$ times of deconvolution operations in a trained convolutional neural network since the output dimension of contact prediction is always the same as its input. While this number can be reduced by choosing only positive labels for analysis, the whole saliency map is still much harder to determine since the many DCA features fed to the network have higher dimensions than the traditional image data. For example, RaptorX-Contact [38], one of the state-of-the-art contact predictors, takes 2D input with a size of $L \times L \times 153$. The recent contact/distance predictor DeepDist [94] takes input with size up to $L \times L \times 484$. The very large input size for contact prediction makes it difficult to use these model interpretation techniques.

Recently, the attention mechanism has been applied in natural language processing (NLP) [65, 71], image recognition [95], and bioinformatics [96, 21]. The attention

mechanism assigns different importance scores to individual positions in its input or intermediate layer so that the model can focus on the most relevant information anywhere within the input. In 2D image analysis, the attention weights for any individual positions on an image allow the visualization of critical regions that contribute to the final predictions. In addition, these weights are generated during the inference step, without requiring additional computation procedures after the prediction of a contact map. Hence, the attention mechanism is a suitable technique to facilitate the interpretation of protein contact prediction models.

In this article, we propose an attention-equipped deep learning method for protein contact prediction (ATTContact), which adopts two different architectures of the attention targeted for interpreting 2D and 1D input features, respectively. The regional attention utilizes the $n \times n$ region around each position of its input 2D map while the sequence attention utilizes the whole range of its 1D input. The regional attention module is implemented with a specially designed 3D convolutional layer so that training and prediction on large datasets can be performed with high efficiency. The sequence attention is similar to the multi-headed attention mechanism applied in the NLP tasks. We show that by applying attention mechanisms on the general deep learning predictors, we can acquire models that are able to explain how position-wise information anywhere in input or hidden features are transferred to later contact predictions, and this can be done without significant extra computational cost and decrease of the prediction accuracy.

4.3 Materials and Methods

4.3.1 Overview

The overall workflow of this study is shown in Figure 4.1. We use the combined predictions from two neural network modules of different attention mechanisms (sequence attention and regional attention) to predict the contact map for a protein target. Both modules take two types of features as inputs: the pseudolikelihood maximization matrix (PLM) [34] generated from multiple sequence alignment as a coevolution-based 2D feature and the position-specific scoring matrix (PSSM) which provides the representation of the sequence profile for the input protein sequence. The outputs of the two modules are both $L \times L$ contact maps with scores ranging from 0 to 1, where L represents the length of the target protein. The final prediction is produced by the ensemble of two attention modules. We implemented our model with Keras (https://keras.io). For the evaluation of the predicted contacted contact map, we primarily focus on long-range contacts (sequence separation between two residues: $n \geq 24$).

4.3.2 Datasets

We select targets from the training protein list used in DMPfold [97] and extract their true structures from the Protein Data Bank (PDB) to create a training dataset. After removing the redundant proteins that may have >25% sequence identity with any protein in the validation dataset and test dataset, 6463 targets are left in the training dataset. The validation set contains 144 proteins used to validate DNCON2 [39]. The blind test dataset is built from 31 CASP13 free modeling (FM) domains. The CASP13 test dataset contains new proteins that have no sequence similarity with both the training and test datasets at all.

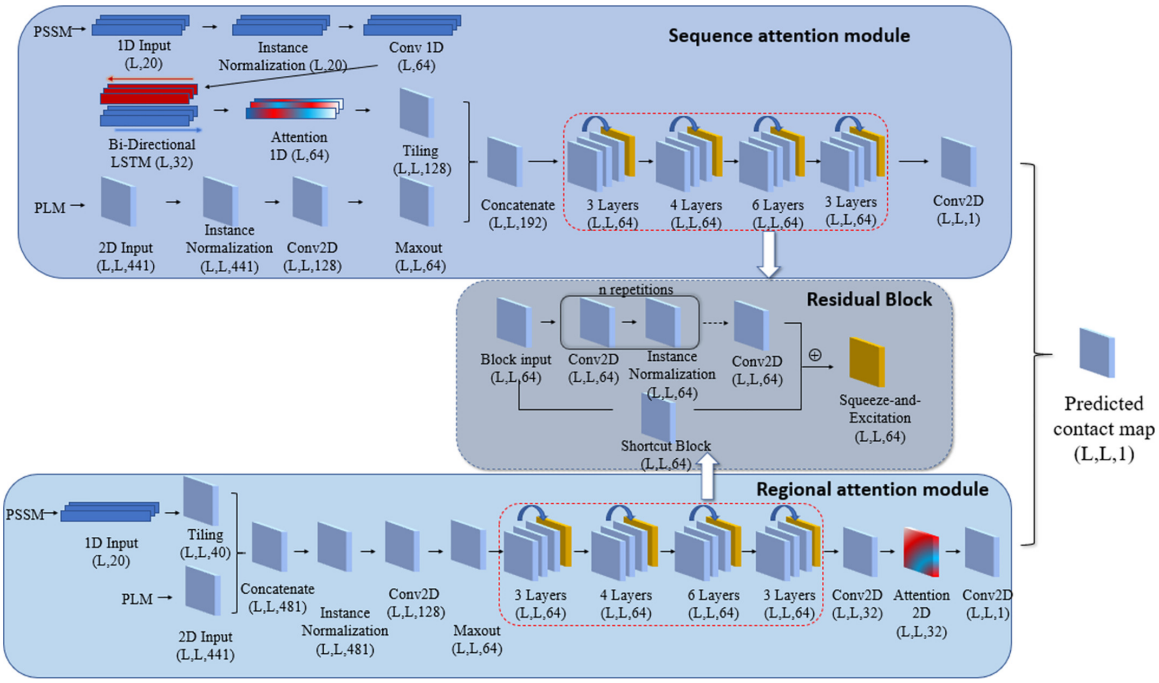

Figure 4.1: An overview of the proposed attention mechanism protein contact predictor framework. The architecture of the deep neural network employed with two attention modules: In the sequence attention module, the 1D input (PSSM) first goes through the 1D convolution network and bidirectional long- and short-term memory network (LSTM). Then the attention mechanism is applied to the LSTM output. The 2D input (PLM) is first processed with the 2D convolutional neural network and the Maxout layer. The 1D input is then tiled to 2D format so that it can be combined with the 2D input. The concatenated inputs then go through a residual network with four residual blocks consist of 3, 4, 6, 3 repetitions of 2D convolution layers, respectively. In regional attention networks, the 1D inputs are first tiled to 2D format and concatenated with the 2D input. The combined inputs are then processed similarly with the sequence attention module, except for the additional 2D attention layer before the last convolution layer. The average of the outputs from the two modules is used as the final predicted contact map.

4.3.3 Input feature generation

For each protein sequence, we use two features as inputs for the deep learning model: PLM and PSSM. The PLM is generated from MSAs produced by DeepMSA [98]. The sequence databases used in the DeepMSA homologous sequences search include Uniclust30 (2017-10), Uniref90 (2018-04) [99], and Metaclust50 (2018-01) [100], our in-house customized database which combines Uniref100 (2018-04) and metagenomics sequence databases (2018-04), and NR90 database (2016). All of the sequence databases used for feature generation were constructed before the CASP13 experiment (eg, before the CASP13 test dataset was created). DeepMSA combines iteratively homologous sequence search of HHblits [101] and Jackhmmer [102] on the sequence databases to compute MSAs for feature generation. It performs trimming on the sequence hits from Jackhmmer with a sequence clustering strategy, which reduces the search time of the HHblits database construction for the next round of search. The final input of the model consists of two major conponents: 1D features (PSSM) of dimension $L \times 20$ and 2D features (PLM) of dimension $L \times L \times 441$.

4.3.4 Deep network architectures

Our model consists of two primary components, the regional attention module, and the sequence attention module (Figure 4.1). The two modules include the attention layers, normalization layers, convolution layers and residual blocks. The outputs of these two modules are combined to generate the final prediction. Below are the detailed descriptions of each module with an emphasis on the attention layers.

Sequence attention module

In the sequence attention module , the 1D PSSM feature first goes through an instance normalization layer [103] and a 1D convolution operation, which is followed by a

Bi-Directional long- and short-term memory network (LSTM) in which the LSTM operations are applied on both forward and reverse directions of the inputs. The output vectors on both directions are concatenated. The outputs are then fed into a multi-headed scaled dot product attention layer (Figure 4.2a). Three vectors required for the attention mechanism: Q(Query), K(Key), and V(Value) are generated from different linear transformations of the input of the attention layer. The attention output Z is computed as:

$$Z = Softmax\left(\frac{Q \times K^T}{\sqrt{d_{att}}}\right) \times V \tag{4.1}$$

The 2D feature PLM first goes into the instance normalization and a ReLU activation [104]. It is then processed by a convolutional layer with 128 kernels of size 1×1 and a Maxout layer [105] to reduce the input dimension from 128 to 64. The 2D inputs are concatenated with the tiled attention output and go into the residual network component. The final output of the sequence attention module is generated from a 2D convolution layer with a filter of size 1×1 and Sigmoid activation, resulting in output of size $L \times L$.

Regional attention module

The regional attention module (Figure 4.2b) takes inputs from the PLM matrix and the tiled 2D PSSM feature. The two features are concatenated at the beginning of the module and are processed in the same way as the 2D PLM input of the sequence attention module. The residual network component with the same configuration as in the sequence attention module is also applied. The last residual block is followed by a convolutional layer with 32 filters, and the results are used as the input of the attention 2D layer.

The input shape of the attention 2D layer is $L \times L \times 32$. It is converted by a

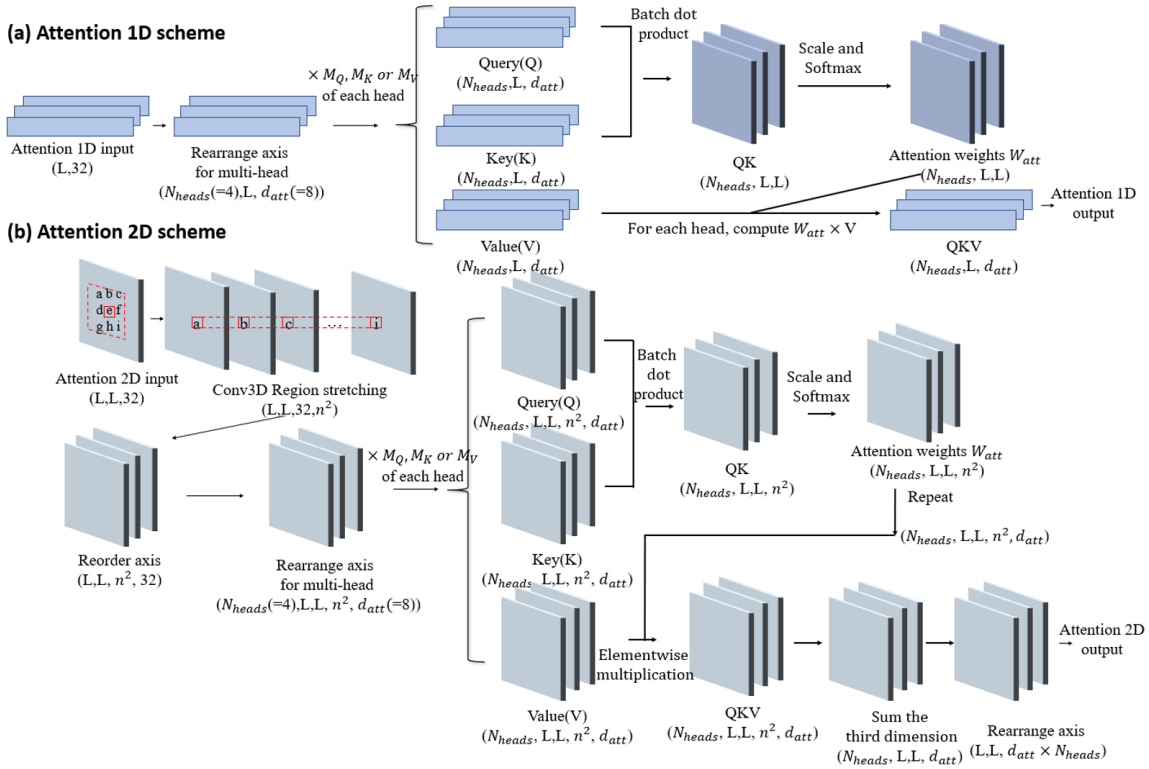

Figure 4.2: Schematic illustration of 1D and 2D attention mechanism. a, The scheme for 1D attention mechanism. The input is first transformed into a vector of size (N_{heads}, L, d_{att}) for the efficient multi-headed attention implementation. For each head, the vector of size (L, d_{att}) is multiplied to three different trainable matrices of size (d_{att}, d_{att}) to generate Query(Q), Key(K), and Value(V). Different heads have their own transformation matrices for Q, K and V. Q and K first go through a batch dot product operation, resulting in a new vector QK^T with size (N_{heads}, L, L). QK^T is then scaled and normalized with Softmax function on the last axis, which becomes the attention score Watt. The product of $W_{att} \times V$ for each head becomes the 1D attention output. b, The scheme for 2D attention mechanism. The 2D input is first transformed with a 3D convolution and becomes a stretched vector of size $(L, L, 32, n^2)$. It is then computed with the similar attention operation as the 1D attention scheme on the last axis.

3D convolution layer (Region Stretching layer) with specially designed filters so that the output has shape $L \times L \times 32 \times n^2$, where n is the dimension of the attention region for each position in the 2D input. The purpose of this layer is to make the last dimension of its output represent the flattened n by n region around each element of the original input (in our model n is set to 5). Thus, each position in the $L \times L$ output are determined by the weighted sum of the n by n square window around itself. The Region Stretching layer has n^2 filters with shape $n \times n$. For the i-th filter of the layer, the weight of the i-th element (flatten in row-major order) in the $n \times n$ area is always set to 1 with all other positions set to 0. We repeat these filters 32 times so that the stretching operation is applied to all dimensions of the input. The weights of these filters will not be changed during training. This operation can leverage the highly optimized convolution implementation in Keras and is much more efficient than the explicit implementation. The corresponding Q, K, and V vectors for the attention mechanism are computed from the transformed output of 3D convolution. The scaling and Softmax normalization are applied to the last dimension for the products of Q and K so that different attention weights can be assigned to the n×n surrounding area for each position on the $L \times L$ map. As a result, the output of each position on the feature map will be a weighted sum of their surrounding regions. After the 2D attention layer, the output of the regional attention module is generated from a 2D convolutional layer with a filter of size 1×1 and the Sigmoid activation.

Residual network architecture

Both attention modules have the same residual network component consisting of four residual blocks differing in the number of internal layers (Figure 4.1). Each residual block is composed of several consecutive instance normalization layers and convolutional layers with 64 kernels of size 3×3. The number of layers showed in each block represents the number of 2D convolution layers in the corresponding

component. The final values of the last convolutional layer are added to the output of a shortcut block, which is a convolutional layer with 64 kernels of size 1×1. A squeeze-and-excitation (SE) block [106] is added at the end of each residual block. The SE operation weights each of its channels differently by a trainable 2-layer dense network when creating the output feature maps, so that channel-wise feature responses can be adaptively recalibrated.

4.3.5 Model Training Configuration

The training of the deep network is performed with the customized Keras data generators to reduce the memory requirement. The batch size is set to 1 due to the large size of feature data produced from long protein sequences. A normal initializer [107] is used to initialize the weights of the layers in the network. Adam optimizer [66] is used for training, with the initial learning rate set to 0.001. For epochs (complete passes through the entire training data) ≥ 30, the optimizer is switched to stochastic gradient descent, with learning rate and momentum set to 0.01 and 0.9, respectivly. The learning rate determines the scale for model parameters update at each iteration and the momentum30 is used to compute the next update of the weights as a linear combination of the current gradient and the update of corresponding weights in the previous iteration. At the end of each epoch, the current weights are saved, and the precision of top L/2 long-range contact predictions (predicted contacts with sequence separation ≥ 24) on the validation dataset is evaluated. The training process is terminated at epoch 60, and the epoch with the best performance on the validation dataset is chosen for the final blind test.

Type	Metric	Sequence attention	Regional attention	Combined model	Baseline model
Short-range	Top-L/5	58.26	**61.08**	60.94	59
	Top-L/2	41.51	41.95	42.69	**42.73**
	Top-L	27.95	28.38	**28.83**	28.17
Medium-range	Top-L/5	63.1	65.46	**66.45**	64.29
	Top-L/2	45.87	48	**48.45**	48.01
	Top-L	32.51	33.65	**34.19**	33.05
Long-range	Top-L/5	64.46	67.32	**70.73**	66.31
	Top-L/2	52.13	54.15	**55.88**	49.42
	Top-L	39.82	40.96	**42.64**	36.4

Table 4.1: Precision (%) of the top L/5, L/2 and L predicted long-range contacts on the CASP13 dataset

4.4 Results

4.4.1 Benchmark ATTContact with state-of-the-art methods

We evaluate our models on 31 CASP13 FM targets based on the average of the per-target performance on them. According to the definition from CASP13, a pair of residues are considered to be in contact if the distance between their C_β atoms in the native structure is less than 8.0 Å. By convention, long-range contacts are defined as contact pairs in which the sequence separations between the two residues of the contacts are larger than or equal to 24 residues. The sequence separation for medium-range is between 12 and 23 and short-range between 6 and 11 residues. Following a common standard in the field,1 we evaluate the precision of top L/n (n = 1, 2, 5) predicted long-range contacts. In addition to evaluating the overall performance of the combined model, we benchmarked the predictions from the two independent attention modules. The evaluation results are shown in Table 4.1.

The combined model outperforms each individual attention model and model without attention mechanism for top L/5, top L/2, and top L predicted contacts in medium and long-range. For instance, the top L/5 long-range precision of the

Method Name	**Top-L/5**	**Top-L/2**	**Top-L**
RaptorX-Contact	71.70	59.02	45.58
ATTContact(ours)	70.73	55.88	42.64
TripletRes	65.97	55.34	42.65
ResTriplet	65.36	54.81	41.84
DMP	62.76	48.90	37.69
TripletRes_AT	60.77	52.02	40.13
RRMD	60.29	49.60	38.50
ZHOU-Contact	59.66	49.42	38.16
RRMD-plus	58.63	47.86	36.98
ResTriplet_AT	58.22	49.18	38.48
Zhang_Contact	58.07	49.58	39.21

Table 4.2: Comparison of the performance of the combined attention model with top 10 CASP13 methods

combined model is 70.73%, higher than both the sequence attention module (64.46%) and the regional attention module (67.32%) as well as the baseline model that without either of the attention mechanisms. According to the pair t test, the combined model performance is significantly better than the sequence model in all ranges ($P < 0.05$), while no significant difference is observed when compared with the baseline or regional attention model. We also compare the performance of our method with the top 10 methods in CASP13 on the FM targets (Table 4.2) and show that it achieves the overall performance comparable to the top-ranked CASP13 methods. Specifically, the accuracy of top L/5 or top L/2 predictions of our method ("Combined Attention") is ranked second out of the 11 methods.

We also find that the predictive improvements in combining the two attention modules are from the predictions with high confidence scores. Figure 4.3a and b illustrates the receiver operating curve (ROC) and Precision-Recall curves (PR curve) of the three models on targets for evaluation. The area under the curve (AUC) for ROC curve and PR curve of all three models has similar trends. Figure 4.3c and d shows the ROC and PR curves of the union of residue pairs from top-L/5 scores in any of the three models. For AUC of both curves, the combined results have a higher

ID	Model type	heads	Region	Top-L/5	Top-L/2	Top/L
1	Sequence	1	-	0.6195	0.4604	0.3455
2	Sequence	2	-	0.5961	0.4540	0.3445
3	Sequence	4	-	**0.6356**	**0.4740**	**0.3556**
4	Regional	1	5	0.6589	0.4798	0.3523
5	Regional	2	5	0.6646	0.4780	0.3548
6	Regional	4	5	**0.6708**	**0.4925**	**0.3675**
7	Regional	4	3	0.6607	0.4830	0.3581

Table 4.3: Comparison of the performance of different attention configurations. Bold scores denote the highest.

score (0.7888 for ROC curve and 0.8031 for PR curve) than the sequence attention model (0.7614 for ROC curve and 0.7935 for PR curve) and the regional attention model (0.7769 for ROC curve and 0.7907 for PR curve). The improved performance of combining the two attention models indicates that the ensemble of two different attention architectures enhances the final prediction.

For both models, we have also benchmarked the impact of different combinations of attention configuration (number of attention heads and size of attention regions) with the maximum scale of the architectures allowed by our GPU memory capacity (Nvidia GeForce 1080Ti 11G). The results are showed in Table 4.3.

4.4.2 Comparison of two attention modules

We compare the performance of the two attention modules for each target in Figure 4.4a and b. The results show that the precision scores of the two attention modules have a strong correlation (Pearson Correlation Coefficient = 0.78) among all targets. As expected, most of the targets with high prediction precision in the combined model are those with high precision scores in both attention modules. Interestingly, there are cases in which the combined predictions acquire an improved performance when the two attention modules perform very differently. For example, the top-L/5 precision score of T1008_D1 reaches 93.33% in the combined model, higher than the

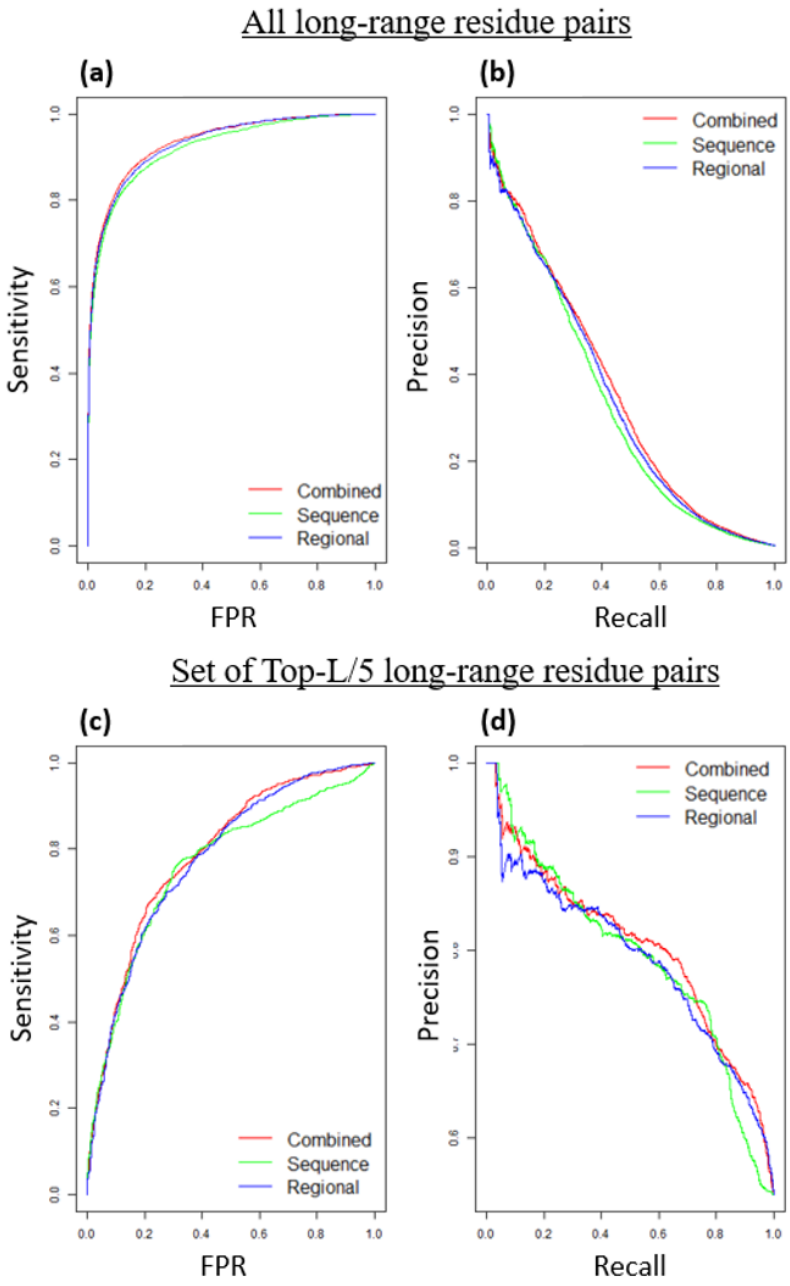

Figure 4.3: Prediction performance curves of the sequence attention model, regional attention model, and combined model. (a) Receiver operating curve (ROC) curve for all long-range contact predictions. (b) Precision-Recall curve for all long-range contact predictions. (c) ROC curve for all residue pairs that appear in the union of residue pairs from top-L/5 scores in any of the three models. (d) Precision-Recall curve for all residue pairs that appear in the top-L/5 scores in any of the three models.

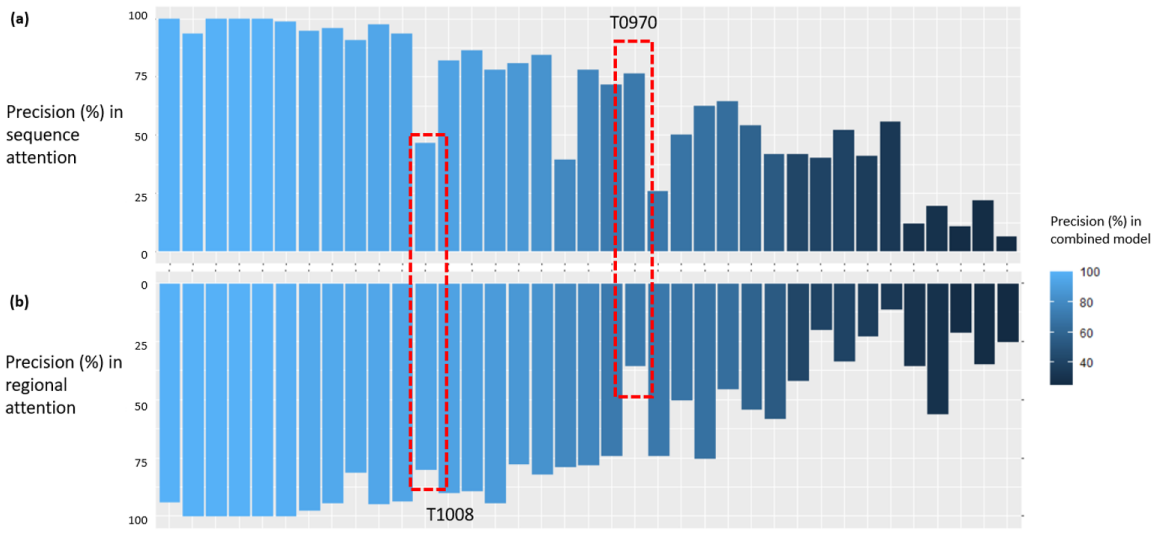

Figure 4.4: Comparison of the top-L/5 precision between sequence and regional attention module. The targets are arranged in the descending order of the top-L/5 precision in the combined model. (a) Precision scores from the sequence attention module. (b) Precision scores from the regional attention module. T1008 and T970 are two examples in which the two modules perform very differently.

sequence module (46.67%) and the regional module (80.00%). Similarly, the top-L/5 precision score of T0957s2 reaches 64.52% in the combined model, which is equal to the sequence module and higher the regional module (45.16%). These results further confirm that the difference in the architecture of two attention mechanisms provides a complementary effect that can contribute to performance improvement.

4.4.3 Visualization of attention scores from the sequence model

Our sequence attention model is similar to the neural translation model proposed in the Transformer [65], in which the attention weights are visualized through case studies of the importance of each word in the source language for a sentence to each word in the target language. While it may be infeasible to directly understand the importance of each residue in a protein in the folding process through human observations, we included several proteins (2PTL,1IDY and 1SHG) [108, 109, 110, 111] that have been studied through experimentaly determined Φ-values. The Φ-values

are the ratio of the change in stability of the transition-state ensemble (TSE) to that of the native state during folding due to the mutation of each residue, and represent important information about residue interactions present within the TSE [112].

Next, we demonstrate how position-wise information in sequence attention model is transferred to later-stage contact predictions in our attention mechanisms for 1D features. Since the sequence attention score is a $L \times L$ matrix, in which element (i,j) represents the importance of the j-th residue to the i-th residue, and the sum of each row is normalized to 1. Thus, the column sum of the attention weights can represent the overall importance of each residue according to the 1D input. We first checked the column sums of attention scores of these three proteins and compared the density of scores from regions of the highest Φ-value peak and scores from the rest regions. The results are shown in Figure 4.5. We show that the scores of Φ-value peak are significantly higher than other regions (P-value < 0.01, Wilcoxon test).

4.4.4 Regional attention scores identify key residue pairs in folding

We first consider the importance of the area with the high attention scores in contact prediction. To demonstrate this, we permute the input features around the positions that have high or low attention scores and use this permuted feature for prediction. Here we choose locations that have the highest and lowest k attention scores as centers for permutated regions, where k is the number of true positives of each target. Our results show that the number of true positive predictions will decrease most drastically (Figure 4.6), indicating that they contain important information related to protein fold. Also, the level of decrease remains similar when the region of permutated data grows from 1×1 to 5×5 in areas with high attention scores. In contrast, the level of decrease in areas with low attention score is much smaller and increases with the expansion of the permuted area. These results indicate the existence of potential

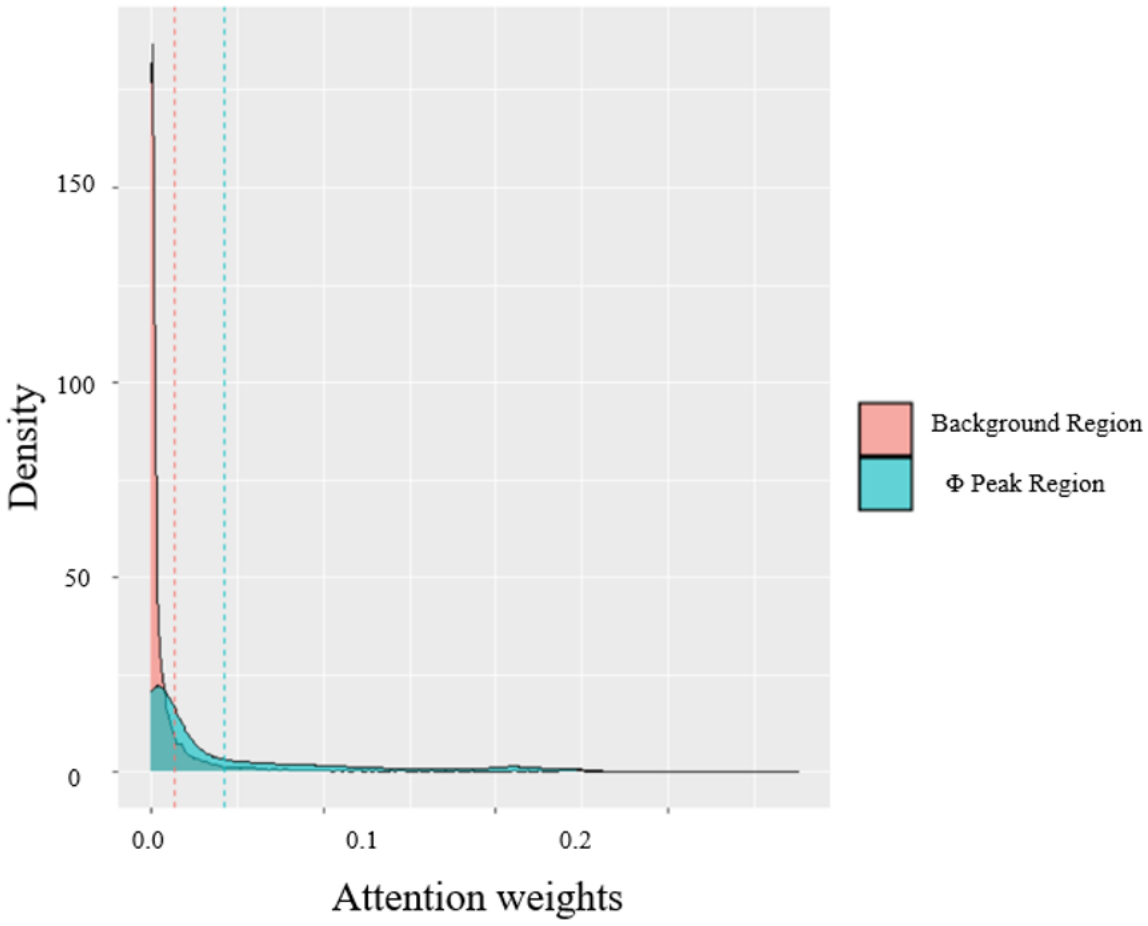

Figure 4.5: Comparison of attention scores from regions of the highest Φ-value peak and scores from the rest regions. The attention scores are averaged across all attention heads.

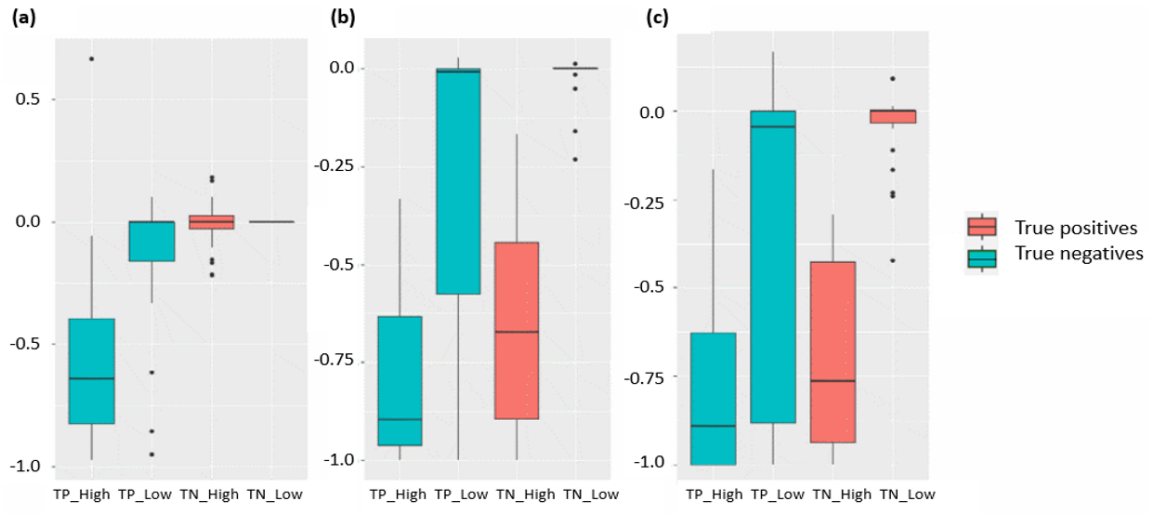

Figure 4.6: Performance after permutation of different locations of the input. The Y-axis indicates the increase or decrease of top-L/5 precision scores after permutation. (a) Impact of permutated regions with size 1×1. (b) Impact of permutated regions with size 3×3. (c) Impact of permutated regions with size 5×5. TP_High, true positive predictions with high scores; TP_Low, true positive predictions with low scores; TN_High, true negative predictions with high scores; TP_Low, true negative predictions with low scores.

protein folding-related key information in small areas with high attention scores.

To further explore the interpretability of our method, we analyze the model on a protein whose folding mechanism has been well studied: Human common-type acylphosphatase (AcP). The structure and sequence information of AcP is obtained from PDB (https://www.rcsb.org/structure/2W4C), which has been identified with three key residues (Y11, P54, and F94) that can form a critical contact network and result in the folding of a polypeptide chain to its unique native-state structure [113]. The 3D structure model and three key residues are shown in Figure 4.7a.

We use the regional attention module to predict the contact map of the protein. The precisions of the top-L/5, L/2, and L prediction are 100%, 95.74%, and 75.79%, respectively. We then extract the 2D attention score matrix from the model and combine the normalized row sums and column sums to reformat its dimension to $L \times 1$. The attention score mapped to the protein 3D structure spot two key residues:

Y11 and F94, where large regions of high attention weights are located (Figure 4.7b). Furthermore, we apply the same strategy with the experimentally determined Φ-values on the 3D structure of AcP (Figure 4.7c). The comparison (Figure 4.7d and e) shows that the 4.7-values and normalized attention scores have similar trends along the peptide sequence (Pearson correlation coefficient = 0.4) with three peaks for Y11, P54, and F94 appeared in neighboring regions of the curves determined by both the experimental method and the attention method. Also, we find that the true contract map does not provide the same level of information about the three key residues (Figure 4.7f). These results indicate that the attention scores can be applied to identify the critical components of the input feature. However, we also find that the co-evolutionary input scores calculated by PSICOV can also be used to identify some Φ-value peaks of AcP. Therefore, the 2D regional attention weights can be either a new way to identify folding-related new residues or summarization of the input. This situation is different from the 1D sequence attention, where the 1D attention weights can definitely identify Φ-value peaks (folding-related residues) that cannot be recognized from 1D inputs at all. Therefore, attention mechanisms can improve the explainability of contact prediction models, but the effects are not guaranteed and may depend on their architecture and inputs.

4.5 Conclusions

Attention mechanisms have two valuable properties that are useful for protein structure prediction. First, attention mechanisms can identify important input or hidden features that are important for structure prediction, and therefore they have the potentials to explain how predictions are made and even increase our understanding of how proteins may be folded. However, the knowledge gained from the attention mechanisms depends on how they are designed and the input information used with them.

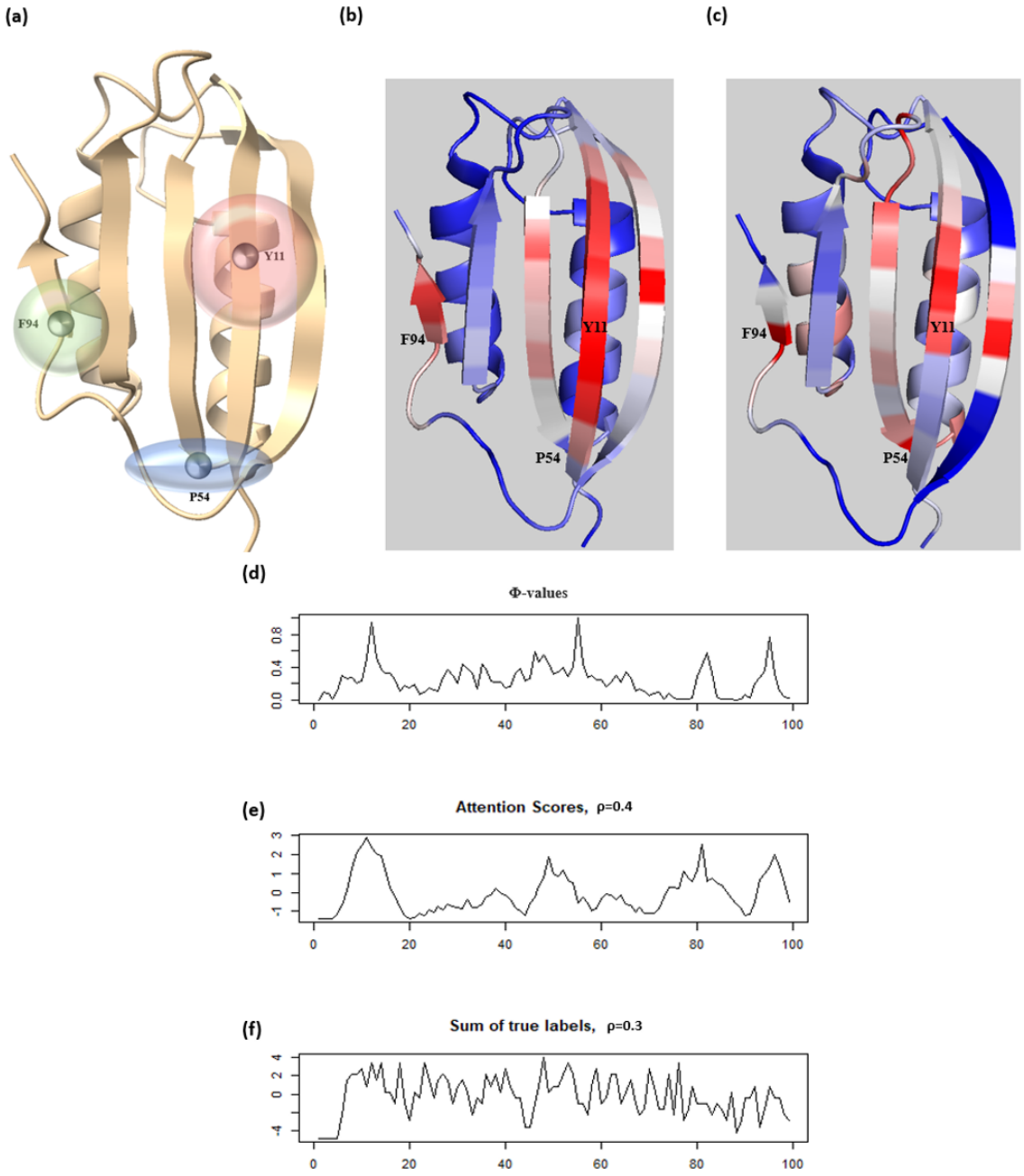

Figure 4.7: Visualization and interpretation contact predictions of Human common-type acylphosphatase from the regional attention module. (a) The 3D model of acylphosphatase (AcP) with the three highlighted critical residues in protein folding. The transparent spheres around the residues indicate their corresponding scopes in the contact networks. (b) The heatmap of regional attention scores shown on the 3D structure of AcP. (c) The heatmap of Φ-values shown on the 3D structure of AcP. (d)-(f), The Φ-values, attention scores and the count of true contacts for each reside plotted along the protein sequence. ρ: Pearson Correlation Coefficient.

Second, attention mechanisms can pick up useful signals relevant to protein structure prediction anywhere in the input, which is much more flexible than other deep neural network architectures such as sequential information propagation in recurrent neural networks and spatial information propagation in convolutional neural networks. As protein folding depends on residue-residue interactions that may occur anywhere in a protein, the attention mechanisms can be a natural tool to recognize the interaction patterns relevant to protein structure prediction or folding more effectively.

Interrogating the input-output relationships for complex deep neural networks is an important task in machine learning. It is usually infeasible to interpret the weights of a deep neural network directly due to their redundancy and complex nonlinear relationships encoded in the intermediate layers. In this study, we show how to use attention mechanisms to improve the interpretability of deep learning contact prediction models without compromising prediction accuracy. More interestingly, patterns relevant to key fold-determining residues can be extracted with the attention scores. These results suggest that the integration of attention mechanisms with existing deep learning contact predictors can provide a reliable and interpretable tool that can potentially bring more insights into the understanding of contact prediction and protein folding. The usage of ATTContact are described in Appendix C.

Chapter 5

3D-equivariant graph neural networks for protein model quality assessment

5.1 Abstract

Quality assessment of predicted protein tertiary structure models plays an important role in ranking and using them. With the recent development of deep learning end-to-end protein structure prediction techniques of generating highly confident tertiary structures for most proteins, it is important to explore corresponding quality assessment strategies to evaluate and select the structural models predicted by them since these models have better quality and different properties than the models predicted by traditional tertiary structure prediction methods.

In this chapter, we describe EnQA, a novel graph-based 3D-equivariant neural network method that is equivariant to rotation and translation of 3D objects to estimate the accuracy of protein structural models by leveraging the structural features acquired from the state-of-the-art tertiary structure prediction method - AlphaFold2. We train and test the method on both traditional model datasets (e.g.,

the datasets of the Critical Assessment of Techniques for Protein Structure Prediction (CASP)) and a new dataset of high-quality structural models predicted only by AlphaFold2 for the proteins whose experimental structures were released recently. Our approach achieves state-of-the-art performance on protein structural models predicted by both traditional protein structure prediction methods and the latest end-to-end deep learning method - AlphaFold2. It performs even better than the model quality assessment scores provided by AlphaFold2 itself. The results illustrate the 3D-equivariant graph neural network is a promising approach to the evaluation of protein structural models. AlphaFold2 features are important for improving protein model quality assessment and are complimentary with the geometric property features extracted from structural models. The source code of EnQA is available at https://github.com/BioinfoMachineLearning/EnQA.

5.2 Introduction

Predicting the structures of proteins from their sequences is crucial for understanding their roles in various biological processes. Various computational methods have been developed to predict protein structure from sequence information [114, 43, 36, 42, 31, 115, 116]. However, some predicted structures are still far from the true structure, especially for some proteins lacking critical information such as homologous structural templates or residue-residue co-evolution information in their multiple sequence alignments. Besides, many computational methods produce multiple outputs for one input sequence. Thus, it is important to estimate the accuracy of the predicted tertiary structural models, i.e., their similarity or discrepancy with the native but unknown structure. Such estimation can help select the best models from the predicted candidates and identify erroneous regions in the models for further refinement.

Many methods for model quality assessment (QA) have been developed. For ex-

ample, PCONS [117] and ModFOLDclustQ [118] uses the comparison between 3D models to evaluate their quality. VoroMQA [119] computes confidence scores based on the statistical potential of the frequencies of observed atom contacts. SBROD [120] uses a smooth orientation-dependent scoring function with a ridge regression model. Deep learning-based QA methods have been reported. DeepQA [121] uses a deep belief network and different agreement metrics. ProQ4 [122] uses the partial entropy of the sequence characteristics with a Siamese network configuration. GraphQA [123] tackles the QA protein with graph convolutional networks based on geometric invariance modeling. Ornate [124] and DeepAccNet [125] are based on voxelized spatial information of the predicted models and 2D/3D convolution networks. DeepAccNet is one of the best-performing methods in the QA category of the CASP14 competition [126].

The pioneering development of the end-to-end deep learning method for protein structure prediction - AlphaFold2 [42] that generated highly confident 3D structures for most protein targets in CASP14 as well as the recent release of a similar approach - RoseTTAFold [43] presents notable improvements in structure prediction and brings new challenges for the model quality assessment task because traditional QA methods developed for evaluating structural models predicted by traditional methods likely do not work well for the models predicted by the new methods such as AlphaFold2. Since the software of the end-to-end approach, such as AlphaFold2 has been publicly released and is becoming the primary tool for tertiary structure prediction, it is important to develop corresponding quality assessment methods to evaluate their models. Furthermore, since AlphaFold2 generates structural models with a self-reported per-residue local distance difference test (lDDT) [127] quality score, new QA methods should outperform (1) the consensus evaluation of a predicted model by comparing it with the reference models predicted by AlphaFold2 and (2) the self-reported per-residue lDDT score for models provided by AlphaFold2. And it would be interesting

to investigate if and how various information extracted from AlphaFold2 predictions can be used to enhance the quality assessment of 3D tertiary structural models. Finally, it is important to leverage the latest deep learning techniques of analyzing 3D objects.

The concept of rotation and translation equivariance in neural networks is useful for the analysis of rotation/translation-invariant properties of 2D and 3D objects in multiple domains, including 2D images [128, 129], quantum interactions [130], and 3D point clouds [131, 132, 133]. For equivariant networks, applying rotation and translation to the input results in a corresponding equivalent transformation to the output of the network. Invariance is a special case of equivariance, in which the same output is generated from the networks when such transformations are applied. Because the quality of a protein structural model is invariant to rotation and translation, it is desirable to use equivariant networks to predict model quality. As the locations of residues in a protein model can be represented as point clouds in 3D space, it is natural to represent a protein model as a graph, which can be equivariant to its rotation and translation. For example, the refinement step in RoseTTAFold [43] uses an equivariant SE(3)-Transformer architecture to update the 3D coordinates. GNNRefine uses a graph convolution network with invariant features for protein model refinement.

In this work, we present EnQA, a 3D equivariant graph network architecture for protein model QA. We evaluate the performance of our method on three different test datasets: the CASP14 stage2 models, the models of the Continuous Automated Model EvaluatiOn (CAMEO), and a collection of AlphaFold2 predictions for recently released protein structures in the Protein Data Bank (PDB). EnQA achieves state-of-the-art performance on all three datasets. It is able to distinguish the high-quality structural models from other models and performs better than the self-reported lDDT score from AlphaFold2. To the best of our knowledge, our method is the first 3D-

equivariant network approach to the problem of model quality assessment. It can effectively evaluate the quality of the models predicted by the current high-quality protein structure prediction methods such as AlphaFold2 that previous QA methods cannot.

5.3 Materials and Methods

In this section, we first describe the training and test datasets and data processing procedure. Then we define the input features to represent protein tertiary structures. Finally, we introduce the EnQA architecture and the implementation details.

5.3.1 Datasets

CASP model quality assessment dataset

We use structural models from server predictions for CASP 8-14 protein targets (Stage two models if available) (Kwon, et al., 2021; Moult, et al., 1995) as one dataset, which can be downloaded from https://predictioncenter.org/download_area/. Models are first filtered by removing those with missing or inconsistent residues with respect to the corresponding experimental structure. The models from CASP 8-12 are used for training. The models from CASP13 are used to validate the neural network and select its hyperparameters. The models from CASP14 are used as the benchmark/test dataset. As a result, there are 109,318 models of 477 CASP8-12 targets used for training, 12,118 models of 82 CASP13 targets used for validation, and 9,501 models of 64 CASP14 targets for the final benchmark/test, respectively. The models in the CASP dataset were generated by traditional protein structure prediction methods during the CASP experiments between 2008 and 2020. The average quality of the models is much lower than the models predicted by the state-of-the-art method –

AlphaFold2.

AlphaFold2 model quality assessment dataset

To create a QA dataset containing protein structural models predicted by the latest end-to-end prediction method - AlphaFold2, we first collect protein structures in the AlphaFold Protein Structure Database [134] with corresponding experimental structures in Protein Data Bank (https://www.rcsb.org/) [135] released after the cutoff date (04/30/2018) of the structures on which AlphaFold2 was trained. In total, there are 4676 protein targets collected after filtering out identical ones. We divide these targets into training and test/benchmark sets with a 9:1 ratio. The AlphaFold2 models of the targets selected for training are combined with the training dataset from CASP 8-12 as the final training data. The targets for the final AlphaFold2 benchmark/test dataset are selected by two criteria: (1) released after the start date of CASP14 (05/14/2020) and (2) having sequence identity ¡30% with any sequence in the training data, which is filtered by MMseqs2 [136]. In total, 178 test targets are selected for the AlphaFold2 benchmark/test data after filtering. For each of these targets above, we generate 5 AlphaFold2 models using the model preset of "casp14" restricting templates only to structures available before CASP14 (i.e., max_template_date = "2020-05-14") to make sure the AlphaFold2 models of the targets are generated with only the information available before their experimental structures were released. The AlphaFold2 models generate for the training targets are added into the training data. The AlphaFold2 models for the 178 test target form the final AlphaFold2 test/benchmark dataset.

CAMEO model quality assessment dataset

To create an additional benchmark dataset, we use the recent models from Continuous Automated Model EvaluatiOn [137]. We downloaded the protein structural

models registered between 9/04/2021 to 11/27/2021, which include predictions from the latest predictors from different groups, such as RoseTTAFold. Models are filtered by removing the ones with inconsistent sequences with the corresponding reference structure. In total, 38 targets with 945 structural models are selected for benchmarking.

5.3.2 Model filtering

CASP14 benchmark dataset protocol

We first select CASP14 target list from the QA results (68 in total), then remove targets that are not evaluated in global QA benchmark as is described in the official assessment: https://onlinelibrary.wiley.com/doi/full/10.1002/prot.26192

The removed targets are: T1048, T1072s1, T1062, T1070, T1080, T1077 (66 remaining).

Then we remove two targets without publicly available native structure The removed targets are: T1085 and T1086 (64 remainings).

"T1098 MESHI_SERVER_TS4" is excluded due to incomplete prediction compared to the reference structure (486/538 residues).

AlphaFold2 predictions dataset

We use 5 models from our AlphaFold predictions and the models from AlphaFold database for training, for testing we use 5 models from our AlphaFold predictions. The configuration we used for running AlphaFold2 is setting the parameters to "casp14", "full_db" and max_template_date is set to "2020-05-14".

For selecting targets for training and benchmark, we first sample 10% (468) of the count of all targets from those targets released after 05/14/2020. The rest are combined with the training datasets created using targets in CASP 8-12. Then we

use MMseqs2 to filter out any targets that have sequence identity ¡30% with any sequence in the training data. Finally, the remaining 178 targets are used for further analysis in the benchmark.

Targets from CAMEO dataset

We first filter the models by removing those predictions with inconsistent sequences with the corresponding reference structure. Targets are excluded for further analysis if less than three models are left after filtering.

5.3.3 Features

We use a graph to represent a protein structural model, which contains node features and edge features. The 1D node feature has a shape (L, d), and the 2D edge feature has a shape (L, L, d) in which L is the number of residues in the model and d is the number of dimensions. The node feature describes the information of each residue, which the edge feature describes the information for each pair of residues. We briefly describe each type of features below.

Node features

We use the 20-number one-hot representation to encode 20 amino-acid types of each residue. Following the spherical convolutions on molecular graphs [138], we use three types of features to characterize the geometric property for each residue: the solvent-accessible surface area, the size of Voronoi cell [139], and the shortest topological distance to nearby solvent-accessible residue (also known as "buriedness"). In addition, we leverage the information from AlphaFold2 predictions made for the protein sequence of each model to generate the quality features for the model. AlphaFold2 predictions used for feature generation are made with the template database curated

before the release date of the experimental structure of any target in the CASP14, CAMEO and AlphaFold2 test datasets. The lDDT score of each residue in a structural model to be evaluated with respect to an AlphaFold2 prediction for the same target (called a reference model) is used as a feature for the residue. The AlphaFold2 self-reported lDDT score for each residue in the reference model is also used as a feature measuring the confidence of the reference model. Here five AlphaFold2 reference models are used for generating features for the structural models of each target, 10 lDDT features are generated for each residue in each structural model. The final shape of the node features for each residue is (L, 33).

Edge distance features

We first extract the logits from the distogram representation of the Al-phafold2 predictions for a protein target, which represents the probability of the beta carbon (C_β) distance between two residues falling into pre-defined 64 distance bins, which has a shape (L, L, 64). From the 64-bin distogram, we then compute the probability of the distance error between two residues in a structural model falling into the 9 distance bins defined by lDDT as follows.

$$d^i_{error} = (d^i_{upper} + d^i_{lower})/2 - d_{model} \tag{5.1}$$

$$P_n = \sum_{i=1}^{64} P^i_{disto} I_{d^i_{error} \in bin_n} \tag{5.2}$$

Here d^i_{error} is the distance error (difference) between the AlphaFold2 predicted distance and an input model for the i-th distance bin of AlphaFold2 and d^i_{upper} and d^i_{lower} are the upper and lower bound of the i-th bin of the distogram, respectively. d_{model} is the distance between any two residues in the input model. P^n is the probability of the distance error between two residues falling into the n-th distance bin defined by lDDT [127]. P^i_{disto} is the softmax-normalized probability of the i-th dis-

tance bin from AlphaFold2 distogram. I is an indicator function that equals 1 if d^i_{error} falls into the range of the n-th bin defined by lDDT and 0 otherwise. Since we use 5 AlphaFold2 distogram predictions for each target and 9 distance bins according to the definition of lDDT, this results in the pairwise edge features with a shape (L, L, 45) for each pair of residues in a structural model. We also create additional binary contact maps by summing up all probabilities in AlphaFold2 distograms that fall into the bins with middle point *leq* 15 Å. The final binary contact map is the average from all five AlphaFold2 predictions to produce an additional edge feature with a shape (L, L, 1).

Spherical graph embedding edge features

We generate rotation-invariant graph embeddings following the Spherical Graph Convolutions Network [138] to use spatial information as spatial edge features. We first build the local coordinate frame for each residue in a structural model. We define the normalized C_α–N vector as the x-axis, the unit vector on the C–C_α–N plane and orthogonal to the C_α–N vector as the y-axis. The direction of the y-axis is determined by the one that has a positive dot product with the C_α–C vector. Naturally, the z-axis is the cross-product of x and y. We compute the spherical angles θ and ϕ of the vector between the C_α of each residue and that of any other residues with respect to this local spherical coordinate system. Figure 5.1 illustrates the local spherical coordinate system used in this work. The spherical angles θ and ϕ are transformed into real spherical harmonics with the formula desribed in [138].

5.3.4 3D-equivariant model architecture

The overall architecture of our method is depicted in Figure 5.2. The processed 1D features (node features) are first processed with 1D convolutions to generate hidden

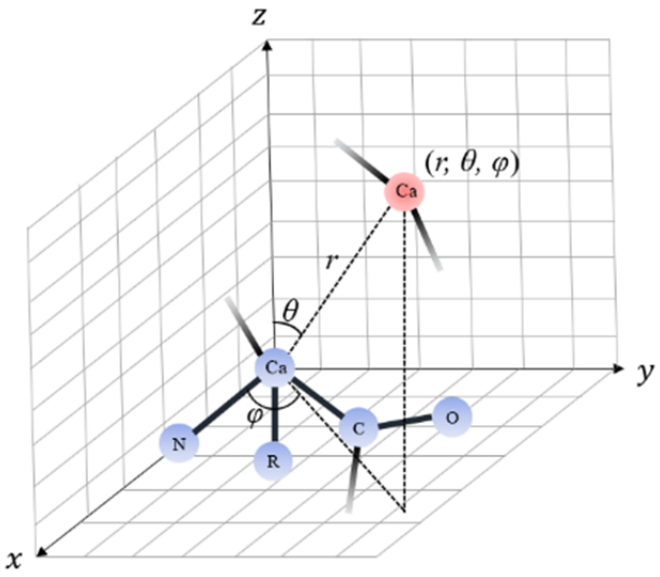

Figure 5.1: The illustration of the local spherical coordinate system. Different colors indicate atoms from different residues. Here θ, ϕ and r are spherical angles and the radial distance for the vector between the alpha carbons (C_α) of two residues (blue and red).

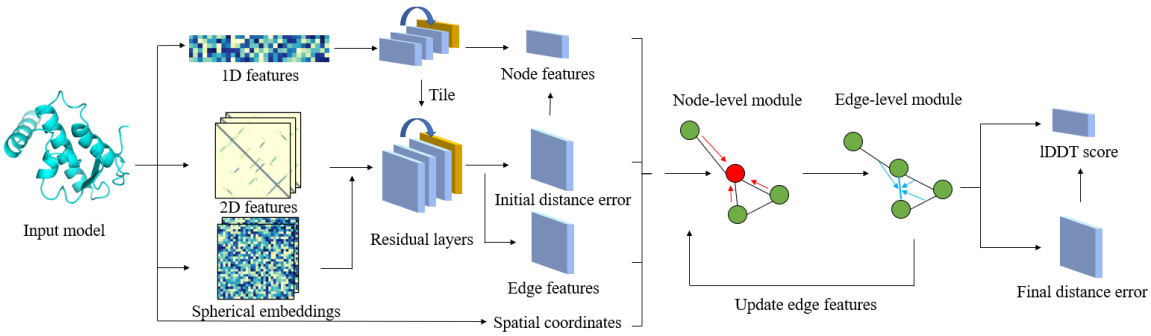

Figure 5.2: The illustration of the overall architecture of EnQA. The 1D/2D features from the input model are first converted into hidden node and edge features for the 3D-equivarant graph module. The spatial coordinates of C_α atoms of the residues are also used as an extra feature. The node and edge network modules update the graph features iteratively. In the end, the final per-residue lDDT score and distance errors of residue pairs are predicted from the updated node/edge features and spatial coordinates by the 3D-equivariant network.

node features. Then 2D features (both distance and graph embedding edge features) and the 2D tiling of the 1D hidden features are processed with a residual architecture with 5 blocks and 32 channels similar to the DeepAccNet [125]. The goal is to predict an initial distance error as a classification task with 9 bins. The distance error is converted into an initial quality estimation using the following formula:

$$s = \sum_{i=1}^{L} p_{ni}(p_{0.5\ \text{Å}} + p_{1\ \text{Å}} + p_{2\ \text{Å}} + p_{4\ \text{Å}})/4 \tag{5.3}$$

Here p_{ni} is the probability of the beta carbon distance between n-th and i-th residue in the binary contact map. $p_{0.5\ \text{Å}}, p_{1\ \text{Å}}, p_{2\ \text{Å}}, p_{4\ \text{Å}}$ are the sum of the probabilities of the multi-class error prediction from the residual layers below different distance cutoffs. This score is combined with the other 1D node features as the node features for the following 3D-equivariant graph network. The spatial coordinates of Ca atom of each residue from the input model are used as one additional feature, which is processed by the graph network in the 3D-equivariant way. The input graph for the 3D-equivariant graph network is constructed by connecting any residue pairs with distance ≤ 15 Å with an edge. The edge features for the graph network are the concatenation of the multi-class error prediction and a separate output of the residual layers for the pairs of the residues.

We design a variant of the E(n) Equivariant Graph Neural Networks (EGNN) [132] to process the node and edge features from the input graph and predict the final model quality score. Given a graph $G = (V, E)$ with nodes $v_i \in V$ and edges $e_{ij} \in E$. Our 3D-equvariant network has a node-level module and an edge-level module. In the node-level module, the hidden node features h_i and alpha carbon (C_α) coordinates x_i associated with each of the residues are considered. The equation of the EGNN layers is the following:

$$m_{ij} = \phi_e(h_i^l, h_j^l, ||x_i^l - x_j^l||^2, a_{ij}) \tag{5.4}$$

$$x_i^{l+1} = x_i^l + \frac{1}{N} \sum_{j \in N(i)} (x_i^l - x_j^l) \phi_x(m_{ij}) \tag{5.5}$$

$$m_i = \sum_{j \in N(i)} m_{ij} \tag{5.6}$$

$$h_i^{l+1} = \phi_h(h_i^l, m_i) \tag{5.7}$$

Here h_i^l andh_j^l are the node features at layer l, a_{ij} is the edge feature, x_i^l and x_j^l is the alpha carbon coordinates. φ_e, φ_x and φ_h are multi-layer perceptron operations. m_{ij} and m_i are the intermediate messages for edges and nodes, respectively.

For the edge-level EGNN module, inspired by the Geometric Transformer [140], we use edges in the original graph as nodes, and define the new node features as the original edge features. Unlike the edges in the node-level module, we use the k-nearest neighbors approach to define the edges in the edge-level module with k set to 3 to accommodate the memory limit for edge-level graphs. The coordinates of the edges are the midpoint of two ends and are always determined by node coordinates rather than updates from the edge-level module. Finally, we use the distances between the midpoints as the new edge attributes. The whole architecture can be trained end-to-end from the input features to the final lDDT score prediction. In addition to the EGNN based graph layer, we also implemented a variant of the network by replacing the EGNN layers with a graph convolution network with kernels regularized by spherical harmonics function as described in the SE(3)-Transformer [131] for comparison. We use 6 Nvidia Tesla V100 32G GPUs on the Summit supercomputer and Horovod/Pytorch to train the method. The batch size is set to 1 for each GPU, resulting in an effective batch size of 6. We use the stochastic gradient descent (SGD) optimizer with learning rate 1e-6, momentum 0.9 and weight decay 5e-5. We use the categorical cross-entropy as the loss function for initial distance error and the MSE loss for predicted lDDT scores as well as the final distance errors. The weight of the

loss for predicted lDDT set to 5, while the weight of the other two errors is set to 1. We set the number of training epochs to 60 with the early stopping when there is no improvements in validation loss for five consecutive epochs.

5.4 Results

5.4.1 Model quality assessment on CASP14 and CAMEO datasets

To compare the performance of EnQA with other state-of-the-art QA methods, we first evaluate it on the CASP14 stage 2 models (Table 5.1, 5.2). We compare it with DeepAccNet [125], VoroMQA [119] and ProQ4 [122], which are all publicly available. We also use five AlphaFold models predicted for each CASP14 target as reference to evaluate the CASP14 stage 2 models. The average lDDT score between a CASP14 model and the five AlphaFold2 models is used as the predicted quality score of the model. This method is called AF2Consensus. The evaluation metrics used include residue and model-level mean squared error (MSE), mean absolute error (MAE) and Pearson Correlation Coefficient between the predicted lDDT scores and ground truth lDDT scores of the models. The per-residue metrics are first computed for each model and are then averaged across all models. Finally, the ranking losses in terms of lDDT and GDT-TS scores are used to evaluate the model ranking capability of the QA methods. The average of the predicted per-residue lDDT scores for each model is calculated as the predicted global quality score of the model. The predicted global quality scores for all the models for a target are used to rank them. The difference between the true GDT-TS (or true average lDDT score) of the best model and that of the top 1 ranked model of the target is the loss.

Our method trained on the combination of CASP 8-12 models and AlphaFold2

Method	Per-residue			Per-model		
	MSE	**MAE**	**Cor**	**MSE**	**MAE**	**Cor**
AF2Consensus	0.0057	**0.0439**	0.8596	0.0018	0.0244	0.9612
DeepAccNet	0.0254	0.1249	0.5725	0.0137	0.0945	0.7459
VoroMQA	0.0686	0.2115	0.3929	0.0466	0.184	0.462
ProQ4	0.0296	0.1331	0.4493	0.0113	0.0806	0.7292
EnQA-Full	**0.0049**	0.0451	**0.8676**	**0.0015**	**0.0227**	**0.9648**
EnQA-Reduced	0.0477	0.1859	0.647	0.0376	0.179	0.8296
EnQA-SE(3)	0.007	0.0607	0.7903	**0.0015**	0.0228	0.9611

Table 5.1: The benchmark of QA results on the CASP14 model dataset. Bold scores denote the highest.

Method	Ranking loss	
	lDDT	**GDT-TS**
AF2Consensus	0.0092	**0.0328**
DeepAccNet	0.0444	0.0933
VoroMQA	0.0614	0.1175
ProQ4	0.057	0.1021
EnQA-Full	**0.0088**	0.0331
EnQA-Reduced	0.0408	0.082
EnQA-SE(3)	0.0116	0.0323

Table 5.2: The ranking loss of QA results on the CAMEO model dataset. Bold scores denote the highest.

models (EnQA-Full) outperforms all the other methods on both residue and model-level metrics , except its per-residue MAE and ranking loss of GDT-TS is slightly worse than the consensus of AlphaFold2 (AF2Consensus). EnQA-Full is performing better than AF2Consensus in terms of most metrics, demonstrating our 3D-equivariant QA method can add value on top of AlpahFold2 predictions in model quality assessment. EnQA-Full and AF2Consensus perform substantially better than the existing methods DeepAccNet, VoroMQA and ProQ4, indicating the importance of incorporating AlphaFold2 features into QA and the value of the 3D-equivariant architecture for QA. The variant of EnQA that uses the SE(3)-Transformer (EnQA-SE(3)) performs slightly worse than EnQA-Full, indicating the 3D-equivariant network (a variant of EGNN) in ENQA-Full may be slightly more effective. The model trained solely on AlphaFold2 models (EnQA-Reduced) yields the worse performance on the CASP14 test dataset than EnQA trained on both CASP8-12 and AphaFold2 models, which is expected since its training dataset contains only the models from AlphaFold2, which is not a good representative of the CASP14 server models generated by the traditional protein structure prediction methods. The quality of the former is generally much better than the latter.

We then evaluate all methods on the CAMEO dataset (Table 5.3, 5.4). Similar to the results from the CASP14 benchmark dataset, EnQA-Full shows the best performances in all metrics, except the ranking loss, which falls behind AF2Consensus by a small margin.

5.4.2 Model quality assessment on AlphaFold2 dataset

To further assess the performance of our method on generally high-quality models, we perform the evaluation on our AlphaFold2 test dataset (Table 5.5, 5.6). We also include the self-reported lDDT scores of the models from AlphaFold2 as the baseline method for comparison (AF2-plddt). In this test, EnQA-Reduced, which

Method	Per-residue			Per-model		
	MSE	**MAE**	**Cor**	**MSE**	**MAE**	**Cor**
AF2Consensus	0.0084	**0.0535**	0.8529	0.0036	0.0353	0.9191
DeepAccNet	0.0245	0.1215	0.6636	0.0146	0.1006	0.7250
VoroMQA	0.1297	0.3175	0.4561	0.1094	0.3125	0.5512
ProQ4	0.0684	0.2163	0.4498	0.0508	0.1961	0.5374
EnQA-Full	**0.0061**	0.0508	**0.8602**	**0.0017**	**0.0272**	**0.9517**
EnQA-Reduced	0.0265	0.1267	0.7250	0.0182	0.1159	0.8322
EnQA-SE(3)	0.0085	0.0681	0.7764	**0.0021**	0.0340	0.9335

Table 5.3: The benchmark of QA results on the CAMEO model dataset. Bold scores denote the highest.

Method	Ranking loss	
	lDDT	**GDT-TS**
AF2Consensus	0.0054	**0.0105**
DeepAccNet	0.0144	0.0193
VoroMQA	0.0470	0.0537
ProQ4	0.0656	0.0673
EnQA-Full	**0.0068**	0.0132
EnQA-Reduced	0.0177	0.0342
EnQA-SE(3)	0.0115	0.0190

Table 5.4: The ranking loss of QA results on the CAMEO model dataset. Bold scores denote the highest.

Method	Per-residue			Per-model		
	MSE	**MAE**	**Cor**	**MSE**	**MAE**	**Cor**
AF2-plddt	0.0232	**0.1119**	0.6651	0.0148	0.1011	0.7113
AF2Consensus	0.0417	0.1579	0.5549	0.0314	0.1562	0.6125
DeepAccNet	0.0394	0.1518	0.4907	0.0269	0.1426	0.5255
VoroMQA	0.1887	0.3899	0.3892	0.1644	0.3856	0.2386
ProQ4	**0.0983**	0.2690	**0.4111**	**0.0791**	**0.2565**	**0.2857**
EnQA-Full	0.0132	0.0803	0.6994	0.0058	0.0533	0.7439
EnQA-Reduced	0.0118	0.0768	0.7090	**0.0043**	0.0455	0.7814
EnQA-SE(3)	0.0127	0.0840	0.6556	0.0045	0.0511	0.7540

Table 5.5: The benchmark of QA results on the AlphaFold2 predicted model dataset. Bold scores denote the highest.

trained solely on AlphaFold2 models, outperforms all other methods on all residue- and model-level metrics, indicating the importance of ensuring the consistency between the training models and test models. Its better performance than AF2-plddt shows that our method performs better in evaluating AlphaFold2 models than AlphaFold2's own quality scores. Furthermore, EnQA-full also outperforms all other methods except EnQA-Reduced in all metrics, including AF2-plddt, indicating combining AlphaFold2 models with traditional protein structural models for training the deep learning method can work well on both new AlphaFold2 test models and non-AlphaFold test models. All our three methods perform substantially better than the previous QA methods (DeepAccNet, VoroMQA, and ProQ4) on this dataset, clearly demonstrating the need of developing new QA methods for evaluating AlphaFold2 models.

5.4.3 Analysis of the performance on AlphaFold2 predicted models

We examine the distribution of model quality of the models in the all benchmark dataset (Figure 5.3c). The average true lDDT score for all models is 0.8034, with 79.82% above 0.7, which has much higher average quality than models used for bench-

Method	Ranking loss	
	lDDT	GDT-TS
AF2-plddt	0.0052	**0.0139**
AF2Consensus	0.0098	0.0235
DeepAccNet	0.0086	0.0226
VoroMQA	0.0090	0.0233
ProQ4	**0.0103**	0.0246
EnQA-Full	0.0049	0.0123
EnQA-Reduced	0.0046	0.0109
EnQA-SE(3)	0.0073	0.0178

Table 5.6: The ranking loss of QA results on the AlphaFold2 predicted model dataset. Bold scores denote the highest.

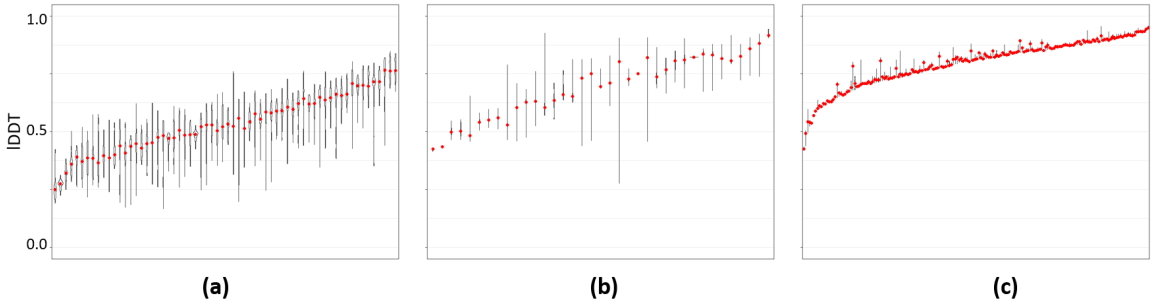

Figure 5.3: The distribution of lDDT scores of benchmark models. (a) CASP14 dataset. (b) CAMEO dataset. (c) AlphaFold2 predicted models. X axis denotes the targets ordered by the mean lDDT of their models in increasing order. The red dots indicate the position of the median.

mark from the CASP (0.5907) and CAMEO (0.6932) dataset (Figure 5.3a and b).

We further investigate the characteristics of the predictions of EnQA-Full and the self-reported lDDT score from AlphaFold2 predictions on the AlphaFold2 test models (Figure 5.4). The predicted scores of EnQA-Full have higher correlation with the true lDDT scores than AlphaFold2 self-reported quality scores. At both residue and global-level, the AlphaFold2 reported score tends to overestimate the quality of the models more than EnQA-Full, which explain one improvement made by EnQA-Full.

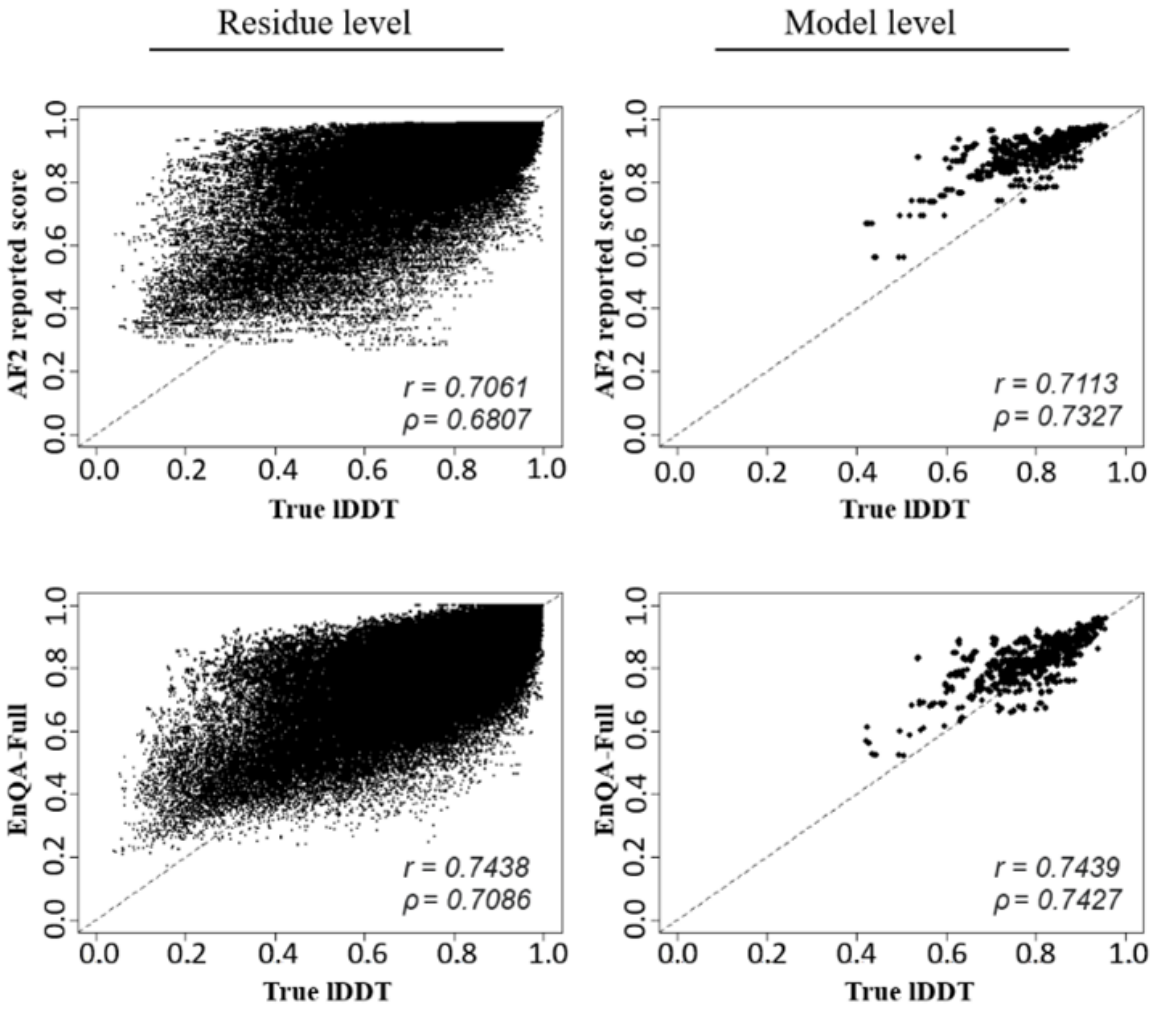

Figure 5.4: The comparison between the predicted and true lDDT scores for AlphaFold models. The residue-level correlation is computed for all residue at once, which is different from the average of the residue-level correlation in each model. r: Pearson Correlation Coefficient. ρ: Spearman Correlation Coefficient.

5.4.4 Analysis of the impact of features

We examine the impact of different input features on the prediction performance of EnQA. We calculate the residue-level Pearson's Correlation Coefficient between predicted lDDT score and true lDDT score when each type of feature is replaced with random number on different benchmark datasets (Figure 5.5). We use EnQA-Full as the baseline model and report the prediction performance when each feature is randomly permuted in its value range during prediction. A larger change of Pearson Correlation Coefficient indicates a higher impact. The lDDT score feature of a model with respect to AlphaFold reference models is the most important feature on AlphaFold2 predicted models (Figure 5.5c) as its permutation causes the largest drop of the Pearson's Correlation Coefficient. The geometric property features for each node, distograms and the confidence score of AlphaFold also have a noticeable impact on the predictive capability. The similar trend can also be observed on both CASP14 and CAMEO dataset (Figure 5.5a, b).

Furthermore, we show that the geometric property feature is critical for those models on which EnQA-full makes large improvements over AF2Consesus. We pick the top 10% models for which it has the largest improvements in residue-level correlation over AF2Consensus and bottom 10% models for which it has the least improvement. In Figure 5.6, the importance of the geometric property feature (i.e., the change of the correlation) in top 10% models is significantly higher than the bottom 10% models (p value ¡ 0.01 according to Mann-Whitney test), suggesting the orthogonal, synergistic effect of the geometric property feature and the features extracted from AlphaFold2 predictions.

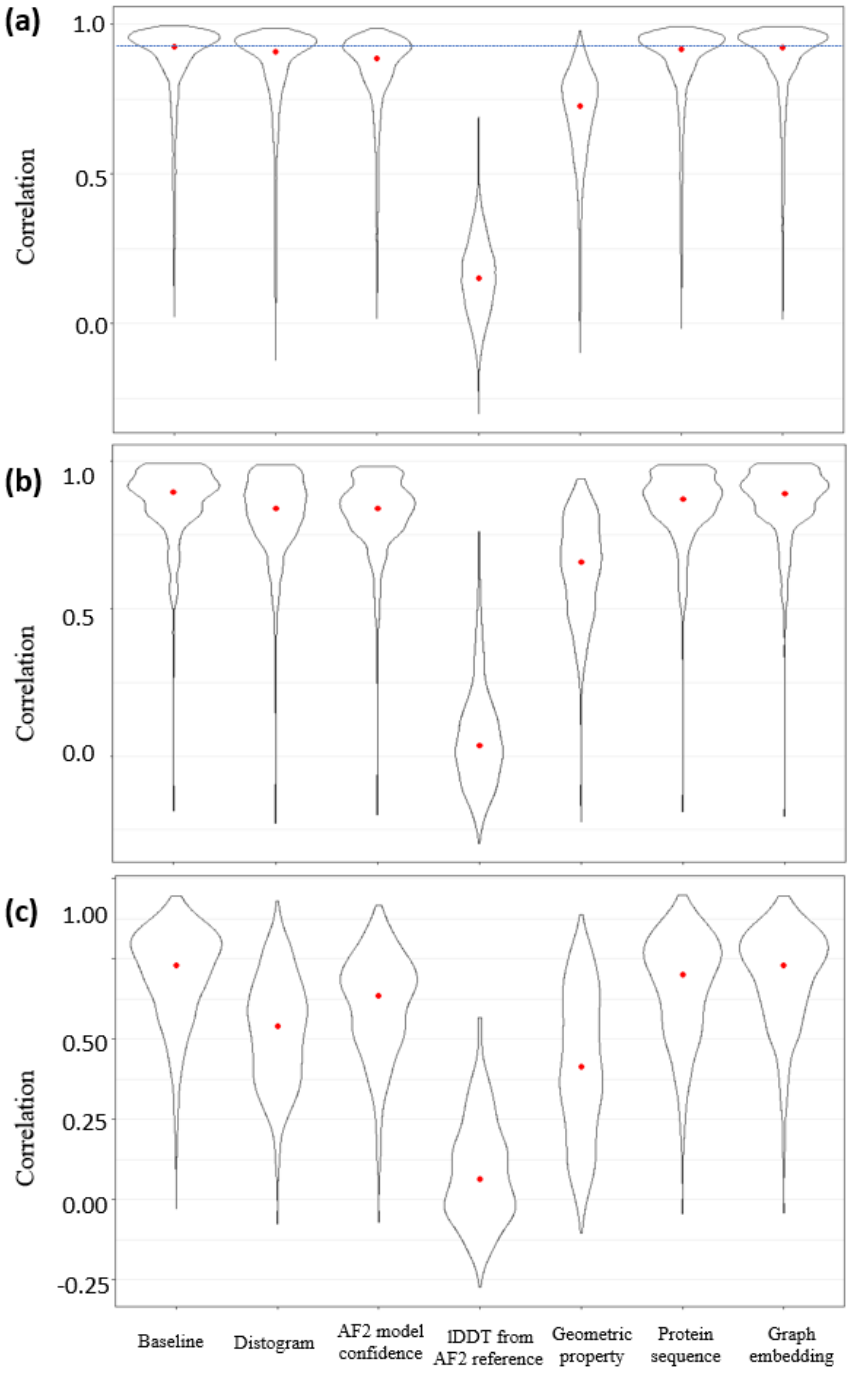

Figure 5.5: The comparison of residue-level Pearson's Correlation Coefficient when different features are randomly permuted for model quality assessment. The red dots indicate the position of the median. (a) CASP14 dataset. (b) CAMEO dataset. (c) AlphaFold2 dataset.

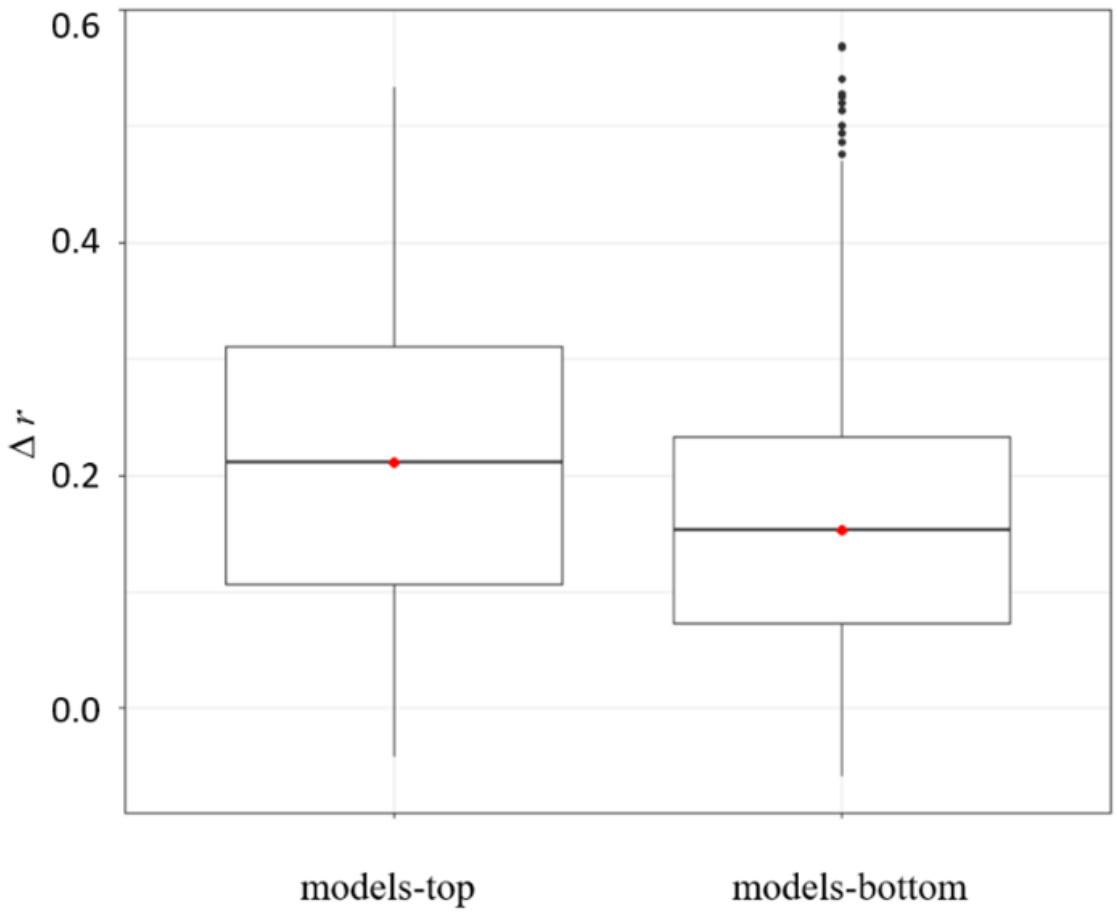

Figure 5.6: The comparison of the importance of the geometric property features. The importance is measured as the decrease in residue-level correlation for models that have most (models-top) and least (models-bottom) improvements in EnQA-Full over AF2Consensus. The change in the correlation in the former is higher than in the latter ($p < 0.01$, Mann-Whitney test).

5.5 Conclusions

In this paper, we introduce EnQA, a novel 3D-equivariant network method for protein quality assessment. Our approach utilizes both the geometric structural features of an input model and the features extracted from AlphaFold2 predictions. The network is developed as an equivariant framework with the node and edge features passing through the node and edge-level graph networks. Performed computational experiments on diverse structural model datasets prove EnQA achieves the state-of-the-art performance of protein quality assessment. More precisely, on both CASP14 and recent CAMEO protein structures, EnQA outperforms all other methods on most evaluation metrics, including using AlphaFold2 predictions as reference to evaluate models. Furthermore, our method performs better than the self-reported lDDT score of AlphaFold2 in evaluating high-quality AlphaFold2 models. On all the test datasets, EnQA performs substantially better than the previous QA methods, demonstrating the value of using 3D-equivarnant architecture and AlphaFold2-based features. Also, we show that the input features extracted from structural models have a complementary effect with the information extracted from AlphaFold2 predictions, especially for those models on which EnQA performs better.

To the best of our knowledge, the method is the first 3D-equivariant network approach to leveraging information from AlphaFold2 predictions to improve model quality assessment. It may be further expanded for protein model refinement by adding additional graph layers to update the coordinates of the nodes (residues) and other protein structure analysis tasks. The usage of EnQA are described in Appendix D.

Chapter 6

Summary and concluding remarks

In this dissertation, four contributions to the functional and structural modelling of proteins are described. Chapter 2 describes DeepGRN, which is a method to predict the binding sites of transcription factors. Chapter 3 describes GNET2, which uses decision tree algorithm and Gaussian Graphical Model to predict the interactions between transcription factors and genes in the gene regulatory network. Chapter3 describes ATTContact, which uses deep learning approach with two different architectures based on the attention mechanism to predict the contact maps of proteins. Chapter 4 describes a novel equivariant graph neural network named EnQA, which takes advantage of the rotations and translations equivariance properties of the protein structures to predict the model quality.

Previous protein/gene interactions modelling methods are usually based on DNA or proteins sequence or gene expression profile, and does not take protein structure information into consideration as the limited availability of protein structures. However, with the release of AlphaFold2 and AlphaFold database, the number of high accuracy of predicted structures is rapidly growing. In the future, it will also be interesting to explore the possibility to use predicted 3D structural models of the proteins for more functional modeling tasks. Last but not least, this paradigm will

also be beneficial to future studies in protein engineering, drug discovery, and many other areas where the properties of proteins are relevant.

Appendix A

Predicting transcription binding sites with DeepGRN

The source code and detailed instructions of generating input data in correct format can be found at: https://github.com/jianlin-cheng/DeepGRN.

A.1 Required software

`Python 3.6.6`

`bedtools 2.26.0`

A.2 Required Python library

`pandas 0.24.2`

`numpy 1.16.2`

`tensorflow 1.11.0`

`pyfaidx 0.5.5.2`

`pyfasta 0.5.2`

```
pybedtools 0.8.0
pyBigWig 0.3.14
Keras 2.2.4
h5py 2.9.0
deepTools 3.2.1
```

Optional(for training with GPUs):

```
cuda 10
tensorflow-gpu 1.11.0
```

A.3 Usage

After cloning the DeepGRN repository, users can run predict.py with the following parameters for prediction:

```
Arguments of predict.py:
--data_dir              Path to the input data
--model_file            Path to model file
--cell_name             Cell ID
--predict_region_file   The bed file contains region to predict
--output_predict_path   Prediction output path
--bigwig_file_unique35  Optional. 35bp uniqueness file
--rnaseq_data_file      Optional. RNA-Seq PCA results
--gencode_file          Optional. Genomic annotation file
--batch_size            Optional. Batch size (default: 512)
--blacklist_file        Optional. The bed file contains regions will be
                        assigned as non binding.
```

Appendix B

Build functional gene modules with GNET2

B.1 Build module networks

To install this package, start R (version "4.2") and enter:

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("GNET2")
```

We generate random expression data and a list of regulator gene names. The input is typically a p by n matrix of expression data of p genes and n samples, for example log2 RPKM from RNA-Seq.

```
set.seed(2)
init_group_num = 8
init_method = 'boosting'
exp_data <- matrix(rnorm(300*12),300,12)
reg_names <- paste0('TF',1:20)
```

```
rownames(exp_data) <- c(reg_names,paste0('gene',
                        1:(nrow(exp_data)-length(reg_names))))
colnames(exp_data) <- paste0('condition_',1:ncol(exp_data))
```

For the list of potential regulator genes, they are usually available from databases. For example, the PlantTFDB (http://planttfdb.gao-lab.org/download.php) curated lists of transcription factors in 156 species, and this information can be imported by the follow steps:

```
url<-"http://planttfdb.gao-lab.org/download/TF_list/Ath_TF_list.txt.gz"
dest_file <- './Ath_TF_list.txt.gz'
download.file(url, destfile)
reg_names <- read.table(gzfile(destfile),sep="\t",header = T,as.is = T)
reg_names = reg_names$Gene_ID
```

The module construction process make take a while, depending on the size of data and maximum iterations allowed.

```
gnet_result <- gnet(exp_data,reg_names,init_method,init_group_num,
                    heuristic = TRUE)
#> Determining initial group number...
#> Building module networks...
#> Iteration 1
#> Iteration 2
#> Iteration 3
#> Iteration 4
#> Iteration 5
#> Converged.
#> Generating final network modules...
#> Done.
```

B.2 Plot the modules and trees

Plot the regulators module and heatmap of the expression inferred downstream genes for each sample. It can be interpreted as two parts: the bars at the top show how samples are split by the regression tree and the heatmap at the bottom shows how downstream genes are regulated by each subgroup determined by the regulators.

```
plot_gene_group(gnet_result,group_idx = 1,plot_leaf_labels = T)
```

It is also possible to compare the clustering of GNET2 with experimental conditions by providing the labels of conditions

```
exp_labels = rep(paste0('Exp_',1:4),each = 3)
plot_gene_group(gnet_result,group_idx = 1,group_labels = exp_labels)
```

The similarity between the clusters from each module of GNET2 and experimental conditions can be quantified by Adjuster Rand Index (for categorical labels) or the inverse of K-L Divergence (for ordinal labels, e.g. dosage,time points). For both cases, significant P-values suggest high similarity between the grouping of the modules and the label information provided by the user.

```
exp_labels_factor = as.numeric(factor(exp_labels))
# Similarity to categorical experimental conditions of each module
print(similarity_score(gnet_result,exp_labels_factor))
#> $score
#> [1]  0.06201550 -0.10852713  0.02531646  0.11814346  0.40310078
#> [6] -0.06751055 -0.16033755
#>
#> $p_value
#> [1] 0.276 0.793 0.328 0.138 0.001 0.636 0.920
```

```
# Similarity to ordinal experimental conditions of each module
print(similarity_score(gnet_result,exp_labels_factor),ranked=TRUE)
#> $score
#> [1]  0.06201550 -0.10852713  0.02531646  0.11814346  0.40310078
#> [6] -0.06751055 -0.16033755
#>
#> $p_value
#> [1] 0.264 0.783 0.319 0.144 0.003 0.644 0.904
```

Plot the tree of the first group

```
plot_tree(gnet_result,group_idx = 1)
```

Appendix C

ATTContact for protein contact maps prediction

The source code and detailed instructions of generating input data in correct format can be found at: https://github.com/jianlin-cheng/InterpretContactMap.

C.1 Required software

`PSI-BLAST 2.2.26 (For generating PSSM sequence profile)`

`CCMpred (For generating pseudo-likelihood maximization)`

`Python 3.6`

C.2 Required Python libraries

`numpy 1.18.1`

`pandas 1.1.2`

`tensorflow-gpu/tensorflow 1.15.2`

`keras 2.1.6`

C.3 Feature gerarationn

Two types of features are required: PSSM sequence profile, which can be generated from PSI-BLAST, and PLM (pseudo-likelihood maximization), which can be generated from CCMpred from multiple sequence alignments (MSAs) produced by DeepMSA. The details of how to acquire both features are described in the DeepDist (https://www.biorxiv.org/content/10.1101/2020.03.17.995910v1), which is also developed by the BDM lab.

The sequence databases used in the DeepMSA homologous sequences search include Uniclust30 (2017-10), Uniref90 (2018-04) and Metaclust50 (2018-01), our inhouse customized database which combines Uniref100 (2018-04) and metagenomics sequence databases (2018-04), and NR90 database (2016). Sample features can be found under the example folder, and users can build both features from their own customized sequence databases.

C.4 Usage

After cloning the repository, users can run predict.py with the following parameters for prediction:

```
-h, --help show this help message and exit
-m, --model_type Type of model, can be one of "sequence", "regional",
                 or "combine"
-l, --plm_data   Path to PLM data. Should be a numpy array flatten
                 from (441,L,L), where L is the length of the input
                 sequence. It should be saved as .npy format.
-s, --pssm_data  Path to PSSM data. Should be a text file start
                 with " # PSSM" as the first line, and the
```

```
                    following contents should each contains L values.

-o, --out_file      Path to output contact map. An L by L numeric matrix
                    saved as TSV format.
-w, --weights       Should attention weights be extracted.
```

Appendix D

EnQA for protein structure accuracy estimation

The source code and detailed instructions of generating input data in correct format can be found at: https://github.com/BioinfoMachineLearning/EnQA.

D.1 Required software

`Python 3.6`

D.2 Required Python libraries

`biopandas 0.2.9`

`biopython 1.79`

`numpy 1.21.3`

`pandas 1.3.4`

`scipy 1.7.1`

`torch 1.10.0`

```
equivariant_attention (Optional, used by models based on
                       SE(3)-Transformer only)
pdb-tools (Optional, used by models with multiple chains only)
```

You may also need to set execution permission for utils/lddt and files under utils/SGCN/bin. Note: Currently, the dependencies support AMD/Intel based system with Ubuntu 21.10 (Impish Indri). Other Linux-based system may be also supported but not guaranteed.

D.3 Usage

After cloning the repository, users can run EnQA.py with the following parameters for prediction:

```
--input                   Path to input pdb file.
--output                  Path to output folder.
--method                  Prediction method, can be "ensemble",
                          "EGNN_Full", "se3_Full", "EGNN_esto9"
                          or "EGNN_covariance".
                          Ensemble can be done listing multiple
                          models separated by comma.
--alphafold_prediction    Path to alphafold prediction results.
--alphafold_feature_cache Optional. Can cache AlphaFold features
                          for models of the same sequence.
--af2_pdb                 Optional. PDBs from AlphaFold
                          predcition for index correction with
                          input pdb when input PDB only
                          contains partial sequence of the
```

```
                              AlphaFold results.

  --cpu                       Optional. Force to use CPU.
```

Bibliography

[1] David S Latchman. Transcription factors: an overview. *The international journal of biochemistry & cell biology*, 29(12):1305–1312, 1997.

[2] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[3] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[4] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[5] Pedro Larranaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Inaki Inza, José A Lozano, Rubén Armananzas, Guzmán Santafé, Aritz Pérez, et al. Machine learning in bioinformatics. *Briefings in bioinformatics*, 7(1):86–112, 2006.

[6] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in bioinformatics*, 18(5):851–869, 2017.

[7] Pankaj Mehta, David J Schwab, and Anirvan M Sengupta. Statistical mechanics of transcription-factor binding site discovery using hidden markov models. *Journal of statistical physics*, 142(6):1187–1205, 2011.

[8] Anthony Mathelier and Wyeth W Wasserman. The next generation of transcription factor binding site prediction. *PLoS computational biology*, 9(9):e1003214, 2013.

[9] Roger Pique-Regi, Jacob F Degner, Athma A Pai, Daniel J Gaffney, Yoav Gilad, and Jonathan K Pritchard. Accurate inference of transcription factor binding from dna sequence and chromatin accessibility data. *Genome research*, 21(3):447–455, 2011.

[10] Tianyin Zhou, Ning Shen, Lin Yang, Namiko Abe, John Horton, Richard S Mann, Harmen J Bussemaker, Raluca Gordân, and Remo Rohs. Quantitative modeling of transcription factor binding specificities using dna shape. *Proceedings of the National Academy of Sciences*, 112(15):4654–4659, 2015.

[11] Marko Djordjevic, Anirvan M Sengupta, and Boris I Shraiman. A biophysical approach to transcription factor binding site discovery. *Genome research*, 13(11):2381–2390, 2003.

[12] Jens Keilwagen, Stefan Posch, and Jan Grau. Accurate prediction of cell type-specific transcription factor binding. *Genome biology*, 20(1):1–17, 2019.

[13] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

[14] Qian Qin and Jianxing Feng. Imputation for transcription factor binding predictions based on deep learning. *PLoS computational biology*, 13(2):e1005403, 2017.

[15] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931–934, 2015.

[16] Daniel Quang and Xiaohui Xie. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic acids research*, 44(11):e107–e107, 2016.

[17] Hamid Reza Hassanzadeh and May D Wang. Deeperbind: Enhancing prediction of sequence specificities of dna binding proteins. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 178–183. IEEE, 2016.

[18] Daniel Quang and Xiaohui Xie. Factornet: a deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data. *Methods*, 166:40–47, 2019.

[19] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

[20] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 207–212, 2016.

[21] Chen Chen, Jie Hou, Xiaowen Shi, Hua Yang, James A Birchler, and Jianlin Cheng. Deepgrn: prediction of transcription factor binding site across cell-types using attention-based deep neural networks. *BMC bioinformatics*, 22(1):1–18, 2021.

[22] Christopher T Harbison, D Benjamin Gordon, Tong Ihn Lee, Nicola J Rinaldi, Kenzie D Macisaac, Timothy W Danford, Nancy M Hannett, Jean-Bosco Tagne, David B Reynolds, Jane Yoo, et al. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431(7004):99–104, 2004.

[23] Jeremiah J Faith, Boris Hayete, Joshua T Thaden, Ilaria Mogno, Jamey Wierzbowski, Guillaume Cottarel, Simon Kasif, James J Collins, and Timothy S Gardner. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS biology*, 5(1):e8, 2007.

[24] Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. In *BMC bioinformatics*, volume 7, pages 1–15. BioMed Central, 2006.

[25] Alexander Statnikov and Constantin F Aliferis. Analysis and computational dissection of molecular signature multiplicity. *PLoS computational biology*, 6(5):e1000790, 2010.

[26] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[27] Mingzhu Zhu, Xin Deng, Trupti Joshi, Dong Xu, Gary Stacey, and Jianlin Cheng. Reconstructing differentially co-expressed gene modules and regulatory networks of soybean cells. *BMC genomics*, 13(1):1–13, 2012.

[28] Mingzhu Zhu, Jeremy L Dahmen, Gary Stacey, and Jianlin Cheng. Predicting gene regulatory networks of soybean nodulation from rna-seq transcriptome data. *BMC bioinformatics*, 14(1):1–13, 2013.

[29] Ping Gong, Zeynep Madak-Erdogan, Jilong Li, Jianlin Cheng, C Michael Greenlief, William Helferich, John A Katzenellenbogen, and Benita S Katzenellenbogen. Transcriptomic analysis identifies gene networks regulated by estrogen receptor α (erα) and erβ that control distinct effects of different botanical estrogens. *Nuclear receptor signaling*, 12(1):nrs–12001, 2014.

[30] Chen Chen, Jie Hou, Xiaowen Shi, Hua Yang, James A Birchler, and Jianlin Cheng. Gnet2: an r package for constructing gene regulatory networks from transcriptomic data. *Bioinformatics*, 37(14):2068–2069, 2021.

[31] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[32] Jilong Li, Debswapna Bhattacharya, Renzhi Cao, Badri Adhikari, Xin Deng, Jesse Eickholt, and Jianlin Cheng. The multicom protein tertiary structure prediction system. In *Protein Structure Prediction*, pages 29–41. Springer, 2014.

[33] David T Jones, Daniel WA Buchan, Domenico Cozzetto, and Massimiliano Pontil. Psicov: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, 28(2):184–190, 2012.

[34] Stefan Seemayer, Markus Gruber, and Johannes Söding. Ccmpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. *Bioinformatics*, 30(21):3128–3130, 2014.

[35] Rojan Shrestha, Eduardo Fajardo, Nelson Gil, Krzysztof Fidelis, Andriy Kryshtafovych, Bohdan Monastyrskyy, and Andras Fiser. Assessing the ac-

curacy of contact predictions in casp13. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1058–1068, 2019.

[36] Jie Hou, Tianqi Wu, Renzhi Cao, and Jianlin Cheng. Protein tertiary structure modeling driven by deep learning and contact distance prediction in casp13. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1165–1178, 2019.

[37] Wei Zheng, Yang Li, Chengxin Zhang, Robin Pearce, SM Mortuza, and Yang Zhang. Deep-learning contact-map guided protein structure prediction in casp13. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1149–1164, 2019.

[38] Sheng Wang, Siqi Sun, and Jinbo Xu. Analysis of deep learning methods for blind protein contact prediction in casp12. *Proteins: Structure, Function, and Bioinformatics*, 86:67–77, 2018.

[39] Badri Adhikari, Jie Hou, and Jianlin Cheng. Dncon2: improved protein contact prediction using two-level deep convolutional neural networks. *Bioinformatics*, 34(9):1466–1472, 2018.

[40] David T Jones, Tanya Singh, Tomasz Kosciolek, and Stuart Tetchner. Metapsicov: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. *Bioinformatics*, 31(7):999–1006, 2015.

[41] Chen Chen, Tianqi Wu, Zhiye Guo, and Jianlin Cheng. Combination of deep neural network with attention mechanism enhances the explainability of protein contact prediction. *Proteins: Structure, Function, and Bioinformatics*, 89(6):697–707, 2021.

[42] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek,

Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[43] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[44] Mehmet Akdel, Douglas EV Pires, Eduard Porta Pardo, Jürgen Jänes, Arthur O Zalevsky, Bálint Mészáros, Patrick Bryant, Lydia L Good, Roman A Laskowski, Gabriele Pozzati, et al. A structural biology community assessment of alphafold 2 applications. *BioRxiv*, 2021.

[45] Chen Chen, Xiao Chen, Alex Morehead, Tianqi Wu, and Jianlin Cheng. 3d-equivariant graph neural networks for protein model quality assessment. *bioRxiv*, 2022.

[46] Oliver Hobert. Gene regulation by transcription factors and micrornas. *Science*, 319(5871):1785–1786, 2008.

[47] Yuanyuan Xiao and Mark R Segal. Identification of yeast transcriptional regulation networks using multivariate random forests. *PLoS computational biology*, 5(6):e1000414, 2009.

[48] Bart Hooghe, Stefan Broos, Frans Van Roy, and Pieter De Bleser. A flexible integrative approach based on random forest improves prediction of transcription factor binding sites. *Nucleic acids research*, 40(14):e106–e106, 2012.

[49] Manal Kalkatawi, Arturo Magana-Mora, Boris Jankovic, and Vladimir B Bajic. Deepgsr: an optimized deep-learning structure for the recognition of genomic signals and regions. *Bioinformatics*, 35(7):1125–1132, 2019.

[50] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[51] Nhi-Thao Tran, Viet-Thang Luong, Ngan Luu-Thuy Nguyen, and Minh-Quoc Nghiem. Effective attention-based neural architectures for sentence compression with bidirectional long short-term memory. In *Proceedings of the Seventh Symposium on Information and Communication Technology*, pages 123–130, 2016.

[52] Ritambhara Singh, Jack Lanchantin, Arshdeep Sekhon, and Yanjun Qi. Attend and predict: Understanding gene regulation by selective attention on chromatin. *Advances in neural information processing systems*, 30, 2017.

[53] Zhen Shen, Wenzheng Bao, and De-Shuang Huang. Recurrent neural network for predicting transcription factor binding sites. *Scientific reports*, 8(1):1–10, 2018.

[54] Sungjoon Park, Yookyung Koh, Hwisang Jeon, Hyunjae Kim, Yoonsun Yeo, and Jaewoo Kang. Enhancing the interpretability of transcription factor binding site prediction using attention mechanism. *Scientific reports*, 10(1):1–10, 2020.

[55] Gökcen Eraslan, Žiga Avsec, Julien Gagneur, and Fabian J Theis. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403, 2019.

[56] Qunhua Li, James B Brown, Haiyan Huang, and Peter J Bickel. Measuring reproducibility of high-throughput experiments. *The annals of applied statistics*, 5(3):1752–1779, 2011.

[57] Samuel J Sholtis and James P Noonan. Gene regulation and the origins of human biological uniqueness. *Trends in genetics*, 26(3):110–118, 2010.

[58] Thomas Derrien, Jordi Estellé, Santiago Marco Sola, David G Knowles, Emanuele Raineri, Roderic Guigó, and Paolo Ribeca. Fast computation and applications of genome mappability. *PloS one*, 7(1):e30377, 2012.

[59] ENCODE Project Consortium et al. An integrated encyclopedia of dna elements in the human genome. *Nature*, 489(7414):57, 2012.

[60] Pedro Madrigal and Paweł Krajewski. Current bioinformatic approaches to identify dnase i hypersensitive sites and genomic footprints from dnase-seq data, 2012.

[61] Fidel Ramírez, Friederike Dündar, Sarah Diehl, Björn A Grüning, and Thomas Manke. deeptools: a flexible platform for exploring deep-sequencing data. *Nucleic acids research*, 42(W1):W187–W191, 2014.

[62] Katherine S Pollard, Melissa J Hubisz, Kate R Rosenbloom, and Adam Siepel. Detection of nonneutral substitution rates on mammalian phylogenies. *Genome research*, 20(1):110–121, 2010.

[63] Xi Chen, Bowen Yu, Nicholas Carriero, Claudio Silva, and Richard Bonneau. Mocap: large-scale inference of transcription factor binding sites from chromatin accessibility. *Nucleic acids research*, 45(8):4315–4329, 2017.

[64] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[67] Hongyang Li, Daniel Quang, and Yuanfang Guan. Anchor: trans-cell type prediction of transcription factor binding sites. *Genome research*, 29(2):281–292, 2019.

[68] A Lando, IE Vorontsov, V Boeva, GV Sapunov, IA Eliseeva, VJ Makeev, and IV Kulakovskiy. Preselection of training cell types improves prediction of transcription factor binding sites. *2016*, 2016.

[69] Timothy L Bailey, Charles Elkan, et al. Fitting a mixture model by expectation maximization to discover motifs in bipolymers. 1994.

[70] Aziz Khan, Oriol Fornes, Arnaud Stigliani, Marius Gheorghe, Jaime A Castro-Mondragon, Robin Van Der Lee, Adrien Bessy, Jeanne Cheneby, Shubhada R Kulkarni, Ge Tan, et al. Jaspar 2018: update of the open-access database of transcription factor binding profiles and its web framework. *Nucleic acids research*, 46(D1):D260–D266, 2018.

[71] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

[72] Guy Karlebach and Ron Shamir. Modelling and analysis of gene regulatory networks. *Nature reviews Molecular cell biology*, 9(10):770–780, 2008.

[73] Eran Segal, Michael Shapira, Aviv Regev, Dana Pe'er, David Botstein, Daphne Koller, and Nir Friedman. Module networks: identifying regulatory modules and

their condition-specific regulators from gene expression data. *Nature genetics*, 34(2):166–176, 2003.

[74] Alex Greenfield, Christoph Hafemeister, and Richard Bonneau. Robust data-driven incorporation of prior knowledge into the inference of dynamic regulatory networks. *Bioinformatics*, 29(8):1060–1067, 2013.

[75] Simon Rogers and Mark Girolami. A bayesian regression approach to the inference of regulatory networks from gene expression data. *Bioinformatics*, 21(14):3131–3137, 2005.

[76] Patrick E Meyer, Kevin Kontos, Frederic Lafitte, and Gianluca Bontempi. Information-theoretic inference of large transcriptional regulatory networks. *EURASIP journal on bioinformatics and systems biology*, 2007:1–9, 2007.

[77] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[78] Jinpu Jin, Feng Tian, De-Chang Yang, Yu-Qi Meng, Lei Kong, Jingchu Luo, and Ge Gao. Planttfdb 4.0: toward a central hub for transcription factors and regulatory interactions in plants. *Nucleic acids research*, page gkw982, 2016.

[79] Hui Hu, Ya-Ru Miao, Long-Hao Jia, Qing-Yang Yu, Qiong Zhang, and An-Yuan Guo. Animaltfdb 3.0: a comprehensive resource for annotation and prediction of animal transcription factors. *Nucleic acids research*, 47(D1):D33–D38, 2019.

[80] Zhenjia Wang, Mete Civelek, Clint L Miller, Nathan C Sheffield, Michael J Guertin, and Chongzhi Zang. Bart: a transcription factor prediction tool with query gene sets or epigenomic profiles. *Bioinformatics*, 34(16):2867–2869, 2018.

[81] Pietro Di Lena, Claudia Sala, Andrea Prodi, and Christine Nardini. Missing value estimation methods for dna methylation data. *Bioinformatics*, 35(19):3786–3793, 2019.

[82] Cosmin Lazar, Laurent Gatto, Myriam Ferro, Christophe Bruley, and Thomas Burger. Accounting for the multiple natures of missing values in label-free quantitative proteomics data sets to compare imputation strategies. *Journal of proteome research*, 15(4):1116–1125, 2016.

[83] Tim Van den Bulcke, Koenraad Van Leemput, Bart Naudts, Piet van Remortel, Hongwu Ma, Alain Verschoren, Bart De Moor, and Kathleen Marchal. Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC bioinformatics*, 7(1):1–12, 2006.

[84] Shai S Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, 31(1):64–68, 2002.

[85] Nabil Guelzim, Samuele Bottani, Paul Bourgine, and François Képès. Topological and causal structure of the yeast transcriptional regulatory network. *Nature genetics*, 31(1):60–63, 2002.

[86] Jie Hou, Xiaowen Shi, Chen Chen, Md Soliman Islam, Adam F Johnson, Tatsuo Kanno, Bruno Huettel, Ming-Ren Yen, Fei-Man Hsu, Tieming Ji, et al. Global impacts of chromosomal imbalance on gene expression in arabidopsis and other taxa. *Proceedings of the National Academy of Sciences*, 115(48):E11321–E11330, 2018.

[87] Vân Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. *PloS one*, 5(9):e12776, 2010.

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

[89] Andrea Franceschini et al. Stringdb package vignette. *Nucleic Acids Res*, 2013.

[90] Jesse Eickholt and Jianlin Cheng. Predicting protein residue–residue contacts using deep networks and boosting. *Bioinformatics*, 28(23):3066–3072, 2012.

[91] Faruck Morcos, Andrea Pagnani, Bryan Lunt, Arianna Bertolino, Debora S Marks, Chris Sander, Riccardo Zecchina, José N Onuchic, Terence Hwa, and Martin Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, 108(49):E1293–E1301, 2011.

[92] Magnus Ekeberg, Cecilia Lövkvist, Yueheng Lan, Martin Weigt, and Erik Aurell. Improved contact prediction in proteins: using pseudolikelihoods to infer potts models. *Physical Review E*, 87(1):012707, 2013.

[93] Dan Agranoff, Delmiro Fernandez-Reyes, Marios C Papadopoulos, Sergio A Rojas, Mark Herbster, Alison Loosemore, Edward Tarelli, Jo Sheldon, Achim Schwenk, Richard Pollok, et al. Identification of diagnostic markers for tuberculosis by proteomic fingerprinting of serum. *The Lancet*, 368(9540):1012–1021, 2006.

[94] Tianqi Wu, Zhiye Guo, Jie Hou, and Jianlin Cheng. Deepdist: real-value inter-residue distance prediction with deep residual convolutional network. *BMC bioinformatics*, 22(1):1–17, 2021.

[95] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[96] Yan Hu, Ziqiang Wang, Hailin Hu, Fangping Wan, Lin Chen, Yuanpeng Xiong, Xiaoxia Wang, Dan Zhao, Weiren Huang, and Jianyang Zeng. Acme: pan-specific peptide–mhc class i binding prediction through attention-based deep neural networks. *Bioinformatics*, 35(23):4946–4954, 2019.

[97] Joe G Greener, Shaun M Kandathil, and David T Jones. Deep learning extends de novo protein modelling coverage of genomes using iteratively predicted structural constraints. *Nature communications*, 10(1):1–13, 2019.

[98] Chengxin Zhang, Wei Zheng, SM Mortuza, Yang Li, and Yang Zhang. Deepmsa: constructing deep multiple sequence alignment to improve contact prediction and fold-recognition for distant-homology proteins. *Bioinformatics*, 36(7):2105–2112, 2020.

[99] Milot Mirdita, Lars Von Den Driesch, Clovis Galiez, Maria J Martin, Johannes Söding, and Martin Steinegger. Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic acids research*, 45(D1):D170–D176, 2017.

[100] Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nature communications*, 9(1):1–8, 2018.

[101] Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. Hhblits: lightning-fast iterative protein sequence searching by hmm-hmm alignment. *Nature methods*, 9(2):173–175, 2012.

[102] Jaina Mistry, Robert D Finn, Sean R Eddy, Alex Bateman, and Marco Punta. Challenges in homology search: Hmmer3 and convergent evolution of coiled-coil regions. *Nucleic acids research*, 41(12):e121–e121, 2013.

[103] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[104] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[105] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR, 2013.

[106] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[107] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[108] David E Kim, Cindy Fisher, and David Baker. A breakdown of symmetry in the folding transition state of protein l. *Journal of molecular biology*, 298(5):971–984, 2000.

[109] Stefano Gianni, Nicholas R Guydosh, Faaizah Khan, Teresa D Caldas, Ugo Mayor, George WN White, Mari L DeMarco, Valerie Daggett, and Alan R Fersht. Unifying features in protein-folding mechanisms. *Proceedings of the National Academy of Sciences*, 100(23):13286–13291, 2003.

[110] Jose C Martínez and Luis Serrano. The folding transition state between sh3 domains is conformationally restricted and evolutionarily conserved. *Nature structural biology*, 6(11):1010–1016, 1999.

[111] Hiroshi Wako and Haruo Abe. Characterization of protein folding by a ϕ-value calculation with a statistical-mechanical model. *Biophysics and physicobiology*, 13:263–279, 2016.

[112] Kresten Lindorff-Larsen, Emanuele Paci, Luis Serrano, Christopher M Dobson, and Michele Vendruscolo. Calculation of mutational free energy changes in transition states for protein folding. *Biophysical journal*, 85(2):1207–1214, 2003.

[113] Michele Vendruscolo, Emanuele Paci, Christopher M Dobson, and Martin Karplus. Three key residues form a critical contact network in a protein folding transition state. *Nature*, 409(6820):641–645, 2001.

[114] Konstantin Arnold, Lorenza Bordoli, Jürgen Kopp, and Torsten Schwede. The swiss-model workspace: a web-based environment for protein structure homology modelling. *Bioinformatics*, 22(2):195–201, 2006.

[115] Jinbo Xu. Distance-based protein folding powered by deep learning. *Proceedings of the National Academy of Sciences*, 116(34):16856–16865, 2019.

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

[117] Björn Wallner and Arne Elofsson. Prediction of global and local model quality in casp7 using pcons and proq. *Proteins: Structure, Function, and Bioinformatics*, 69(S8):184–193, 2007.

[118] Liam J McGuffin and Daniel B Roche. Rapid model quality assessment for protein structure predictions using the comparison of multiple models without structural alignments. *Bioinformatics*, 26(2):182–188, 2010.

[119] Kliment Olechnovič and Česlovas Venclovas. Voromqa: Assessment of protein structure quality using interatomic contact areas. *Proteins: Structure, Function, and Bioinformatics*, 85(6):1131–1145, 2017.

[120] Mikhail Karasikov, Guillaume Pagès, and Sergei Grudinin. Smooth orientation-dependent scoring function for coarse-grained protein quality assessment. *Bioinformatics*, 35(16):2801–2808, 2019.

[121] Renzhi Cao, Debswapna Bhattacharya, Jie Hou, and Jianlin Cheng. Deepqa: improving the estimation of single protein model quality with deep belief networks. *BMC bioinformatics*, 17(1):1–9, 2016.

[122] David Menéndez Hurtado, Karolis Uziela, and Arne Elofsson. Deep transfer learning in the assessment of the quality of protein models. *arXiv preprint arXiv:1804.06281*, 2018.

[123] Federico Baldassarre, David Menéndez Hurtado, Arne Elofsson, and Hossein Azizpour. Graphqa: protein model quality assessment using graph convolutional networks. *Bioinformatics*, 37(3):360–366, 2021.

[124] Guillaume Pagès, Benoit Charmettant, and Sergei Grudinin. Protein model quality assessment using 3d oriented convolutional neural networks. *Bioinformatics*, 35(18):3313–3319, 2019.

[125] Naozumi Hiranuma, Hahnbeom Park, Minkyung Baek, Ivan Anishchenko, Justas Dauparas, and David Baker. Improved protein structure refinement guided by deep learning based accuracy estimation. *Nature communications*, 12(1):1–11, 2021.

[126] Sohee Kwon, Jonghun Won, Andriy Kryshtafovych, and Chaok Seok. Assessment of protein model structure accuracy estimation in casp14: Old and new challenges. *Proteins: Structure, Function, and Bioinformatics*, 89(12):1940–1948, 2021.

[127] Valerio Mariani, Marco Biasini, Alessandro Barbato, and Torsten Schwede. lddt: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics*, 29(21):2722–2728, 2013.

[128] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[129] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

[130] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017.

[131] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

[132] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[133] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and

translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[134] Kathryn Tunyasuvunakool, Jonas Adler, Zachary Wu, Tim Green, Michal Zielinski, Augustin Žídek, Alex Bridgland, Andrew Cowie, Clemens Meyer, Agata Laydon, et al. Highly accurate protein structure prediction for the human proteome. *Nature*, 596(7873):590–596, 2021.

[135] David S Goodsell, Christine Zardecki, Luigi Di Costanzo, Jose M Duarte, Brian P Hudson, Irina Persikova, Joan Segura, Chenghua Shao, Maria Voigt, John D Westbrook, et al. Rcsb protein data bank: Enabling biomedical research and drug discovery. *Protein Science*, 29(1):52–65, 2020.

[136] Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.

[137] Xavier Robin, Juergen Haas, Rafal Gumienny, Anna Smolinski, Gerardo Tauriello, and Torsten Schwede. Continuous automated model evaluation (cameo)—perspectives on the future of fully automated evaluation of structure prediction methods. *Proteins: Structure, Function, and Bioinformatics*, 89(12):1977–1986, 2021.

[138] Ilia Igashov, Nikita Pavlichenko, and Sergei Grudinin. Spherical convolutions on molecular graphs for protein model quality assessment. *Machine Learning: Science and Technology*, 2(4):045005, 2021.

[139] Kliment Olechnovič and Česlovas Venclovas. Voronota: a fast and reliable tool for computing the vertices of the voronoi diagram of atomic balls. *Journal of computational chemistry*, 35(8):672–681, 2014.

[140] Alex Morehead, Chen Chen, and Jianlin Cheng. Geometric transformers for protein interface contact prediction. *arXiv preprint arXiv:2110.02423*, 2021.

VITA

Chen Chen received his Bachelor's degree from Nanjing University at 2011, and Masters degree from Northeastern University at 2016. He started his Ph.D studies in the Department of Computer Science at University of Missouri-Columbia at fall of 2017. He is interested in developing and applying machine learning and deep learning techniques to address the various bioinformatics problems. He worked on gene regulatory network prediction as the first topic in Ph.D study, he is also interested in other research topics, such as protein structural modeling and exploit the geometric properties of the proteins in deep learning models.