# NEW METHODS FOR PROTEIN STRUCTURE PREDICTION USING MACHINE LEARNING AND DEEP LEARNING

---

A Dissertation

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

---

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

JUNLIN WANG

Professor Yi Shang, Dissertation Advisor

December 2022

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

NEW METHODS FOR PROTEIN STRUCTURE PREDICTION USING MACHINE LEARNING AND DEEP LEARNING

presented by Junlin Wang,

a candidate for the degree of Doctor of Philosophy

and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Yi Shang

Dr. Dong Xu

Dr. Jianlin Cheng

Dr. Ioan Kosztin

# DEDICATION

*To my wife Yizhen and my parents.*

# ACKNOWLEDGEMENTS

I would like to take the opportunity to thank the following people who have supported me through my PhD program. Without the support of them, this thesis would not have been possible.

First and foremost, I am extremely grateful to my supervisors, Dr. Yi Shang for his invaluable advice, continuous support, and patience during my PhD study. He has been my advisor since my senior year. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to thank my committee members, Dr. Dong Xu, Dr. Jianlin Cheng, and Dr. Ioan Kosztin. They have been giving me constructive suggestions and advices to my dissertation and helping me solve problems in my research. Their support is essential, and I appreciate their time and efforts very much.

I would like to thank my friends, lab mates, and colleagues for a cherished time spent together in the lab, and in social settings. Special thanks to Wenbo Wang, Zhaoyu Li, Chao Fang, and Son Nguyen, who were always there for me.

Last but not least, I would like to thank my family, Yizhen Wang, Jianyu Wang, Lan Wang, Li Wang, and Guiju Cui. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

This work was supported in part by National Institutes of Health grant R01GM100701. The high-performance computing infrastructure is supported by the National Science Foundation under grant number CNS-1429294.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

**NEW METHODS FOR PROTEIN STRUCTURE PREDICTION USING MACHINE LEARNING AND DEEP LEARNING**

Junlin Wang

Dr. Yi Shang, Dissertation Advisor

ABSTRACT

Computational protein structure prediction is one of the most challenging problems in bioinformatics area. Due to the widespread use of sampling-and-selection strategy, protein model quality assessment became important. In this dissertation, new machine learning and deep learning methods have been proposed for protein model quality assessment, protein contact prediction, protein model refinement, and loop modeling.

The goal of model quality assessment (QA) is to estimate the quality of predicted protein models. First, two new single-model QA methods based on Residual Neural Networks, called PDRN and VDRN, were proposed to achieve state-of-the-art performance. They used a comprehensive set of structure features to predict a quality score in the range of [0, 1]. Next, three single-model QA methods, MMQA-1 MMQA-2 and MMQA-HE, were proposed based on ideas of two-stage learning and hierarchical ensembles. MMQA-1 and MMQA-2 divided the entire feature set into two different sets and used different feature sets and training data in each stage of learning. In addition, MMQA-HE created ensembles of models in the first stage of learning for improved performance. In CASP14, MMQA-1 ranked NO. 2 in terms of average GDT-TS difference. MMQA-2 and MMQA-HE outperformed MMQA-1 consistently across different QA performance metrics in our experiments. Furthermore, a quasi-single-model QA method called INC-QA was proposed using a new method that trained a deep neural network as a QA predictor for each protein

target based on template structure information generated from the target sequence. Experimental results using CASP data showed that INC-QA achieved state-of-the-art results, outperforming existing methods on CASP QA stage 2 category on CASP 13 targets. With the release of groundbreaking protein structure prediction software AlphaFold2 and RosettaFold, many research teams start using them to generate highly accurate protein models. We evaluated the performance of different QA methods on models generated by them with random modification by 3DRobot and found that multi-model QA methods were still better than single-model QA methods on these kind of high-performance model pools. Finally, in terms of the prediction of overall folding accuracy and overall interface accuracy for protein complexes in CASP15, we found a strong correlation between the predicted folding accuracy and predicted interface accuracy of protein models.

Loop modeling tries to predict the conformation of a relatively short stretch of protein backbone and sidechain. It is a difficult problem due to conformational variability. AlphaFold2 achieved outstanding results in 3-D protein structure prediction and was expected to perform well on loop modeling. We investigated the performances of AlphaFold2 variants on loop modeling benchmark datasets and proposed an efficient constant-time method of using AlphaFold2 for loop modeling, called IAFLoop. To predict the structure of a loop region, IAFLoop ran a fast version of AlphaFold2 with a reduced database without ensembling on an extended segment of the target loop region, and used RMSD based consensus scores to select the top models. Our experimental results showed that IAFLoop generated highly accurate loop models, outperforming basic AlphaFold2 by up to 17% in RMSD error, while using less than half of the time. Compared to the previous best method, IAFLoop reduces the RMSD error by more than half.

Contact map prediction is to predict whether the Euclidean distance between two $C_\beta$ atoms ($C_\alpha$ for Glycine) in a protein structure is less than 8 angstroms. Contacts information can act as a powerful constraint for determining the overall structural and assist the protein 3D structure prediction process. Based on MUFold-Contact, a new two-stage multi-branch deep neural network based on Residual Network and Inception V3 Network was proposed to improve the performance of MUFold-Contact. In the first stage, distance maps of short-range, medium-range and long-range residue pairs were predicted, respectively, and the predicted distance along with other features were used as input to predict a binary contact map in the second stage.

The role of protein structure refinement is to take models generated by protein structure prediction process and bring them closer to the true native structure. Inspired by AlphaFold in CASP13, a new protein structure refinement process MUFOLD-REFINE based on distance distribution of template pool was developed and achieve improved performance over the MUFOLD refinement method used in CASP13

# CHAPTER 1. INTRODUCTION

## 1.1 Protein and Protein Structure Prediction

In biology, protein are large macromolecules comprised of chains of amino acid residues and can perform a vast array of functions within organisms. The study of protein 3D structure is essential on studying its functions (Baker & Sali, 2001). So far, the structures of about 100,000 unique proteins have been determined, but this represents only a small part of the billions of known protein sequences. Unfortunately, experimental methods used for determining protein structures, such as electron microscopy, X-ray crystallography and nuclear magnetic resonance (NMR) are expensive and time consuming (Johnson, Srinivasan, Sowdhamini, & Blundell, 1994). Computational methods can generate numerous alternative decoys for a given protein sequence in limit amount of time (Eisenhaber, Persson, & Argos, 1995), but accurately predicting the 3D structure of proteins has been a challenging problem for decades (Kuhlman & Bradley, 2019).

Proteins have different levels of structural organization: protein sequence, secondary structure, tertiary structure, and quaternary structure. A protein sequence is a sequence of amino acid residues, and each amino acid is represented by a letter. Secondary structure is a form of local segments of proteins. It can be divided into alpha helix, beta sheet and coil. The tertiary structure of a protein is the three-dimensional structure of a complete protein. And the quaternary structure is a complex of multiple proteins.

Protein structure prediction methods are usually divided into two categories: template-based modeling (TBM) and free modeling (FM). Template-based modeling means an approximate three-dimensional model for a given target is built on the basis of a related

protein of known structure, and free modeling represents ab initio, template-free methods (Fiser, 2010). The Critical Assessment of protein Structure Prediction (CASP) is a biennial worldwide contest aimed at establishing the current state of the art in protein structure prediction (Moult, 1999). Before CASP11, improvements were possible due to the exploration of powerful homology searching tools (Moult, 2014). After the accuracy of template based methods reached a bottleneck, most significant improvements were seen in harder targets, where only ab initio, template-free methods could be applied. Through the advancement in protein secondary structure prediction, skeleton torsion angle prediction and contact prediction and the usage of deep learning methods, there is a significant increase in model accuracy for harder targets. In CASP13, hard targets were modeled with an average GDT_TS of 70% by AlphaFold (AF) from DeepMind (Kryshtafovych, 2019). And in CASP14, there is a further leap in model accuracy, with the best model for targets at different difficulty level reaching a GDT_TS above 90% (Senior, Evans, Jumper et al., 2020; Jumper, Evans, Pritzel et al., 2021).

## 1.2 Protein Quality Assessment

The most common used strategy by protein structure prediction methods is sampling-and-selection, which means large numbers of alternative models will be generated during the prediction process, then how to evaluate the performance of the generated models and pick up the best model from the pool becomes one of the most important problem (Kihara, Chen, & Yang, 2009).

Protein quality assessment problem is formulated as following: given the animal acid sequence of a target protein and a predicted decoy structure or a predicted decoy pool, return the quality score that reflects the similarity between the decoy structure and the

native protein structure of the target sequence (Cristobal, Zemla, Fischer et al., 2001). One of the most commonly used measure to compare the results of protein structure prediction to the experimentally determined structure is Global Distance Test Total Score (GDT-TS), which can be calculated as follow:

$$GDT - TS(s_i, s_j) = \frac{(P_1 + P_2 + P_3 + P_4)}{4}$$

Where $s_i$ and $s_j$ are two protein 3D structures and $P_n$ represents the percentage of C-alpha atoms within the threshold of nÅ (n=1, 2, 4, 8) after superimposing one structure over the other structure. GDT-TS score has range from 0 to 1, higher value indicating better accuracy (Zemla, 2003; Zemla, Venclovas, Moult et al., 2001). With values above 0.7 denotes that local and global details are mostly modeled accurately and values below 0.3 means mostly a random structure. If the value is 1, the structures being compared are identical.

Different QA methods generated different style of QA score, with the development of the field of protein structure prediction, the goal of the QA algorithms has also changed. From the judging criteria of QA category used in recent CASP competition, we can know that the current protein model QA field is most concerned about three points: 1) Prediction, given a predicted protein structure, evaluating how good the structure is. 2) Selection, given a pool of predicted models, best model selection is another most import goal for different QA methods. 3) Ranking, given a pool of predicted models, the correlation between predicted GDT-TS scores and true GDT-TS scores is still important (Pereira, Simpkin, Hartmann et al., 2021). But as the evaluation results from different QA methods become more and more accurate, the high performance of prediction itself already guarantees a

high correlation between predicted GDT-TS scores and true GDT-TS scores of the target pool.

The community-wide initiative on the Critical Assessment of Predicted Interactions (CAPRI), established in 2001, has played a central role in fostering the development of better docking algorithms and closely monitoring their performance (Janin, 2003). State-of-the-art techniques for assessing the structural quality of docking models are currently based on three related but independent quality measures: Fnat, LRMS, and iRMS proposed and normalized by CAPRI. These quality metrics quantify different aspects of the quality of a particular docking model and need to be viewed together to reveal true quality, e.g., a model with relatively poor LRMS (>10Å) might still qualify as 'acceptable' with a descent Fnat (>0.50) and iRMS ( <3.0Å). The CAPRI criteria for assessing the quality of docking models are defined by applying various interim cutoffs to these metrics to classify docking models into four categories: incorrect, acceptable, moderate, or high quality. The criteria for evaluating the quality of the predicted complexes are summarized in simplified form in Figure 1.1.

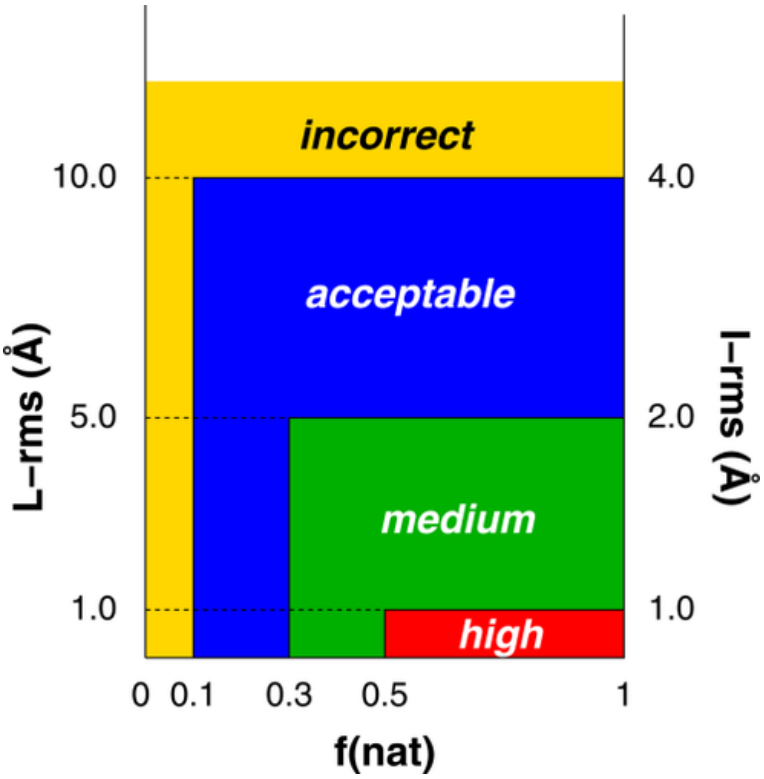


*Figure 1.1 The criteria for evaluating the quality of the predicted complexes in CAPRI.*

## 1.3 Protein Loop Modeling

In protein loop modeling problem, "loop" means the regions with insertions or deletions in the target sequence or templates (Levefelt & Lundh, 2006). In the template-based homology protein modeling problem, the template generally cannot completely cover the target sequence, then the predicted structure may have gaps, which is a region that the amino acid has no atomic coordinates. In this case, it is important to reconstruct these missing regions for protein functions and dynamics studies (Ginalski, 2006). Protein loop modeling is a small-scale protein structure prediction problem since we know the protein sequence of the missing part and we are trying to predict the structure from its sequence and its surrounding known structures. If we can fill in the missing regions very well, the performance protein structure prediction can also be improved.

The following Figure 1.2 shows an example of loop modeling. The model on the left side is the initial model with a missing region, and the three models on the right side are the complete models with the missing region filled in by three different structures.

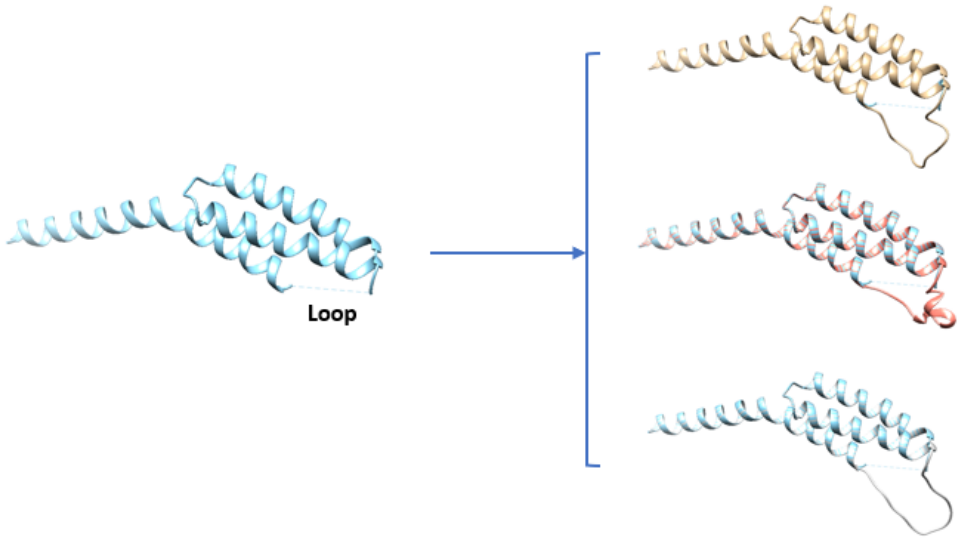


*Figure 1.2 An example of loop modeling problem. The loop region can be filled in in different ways.*

## 1.4 Protein Contact Map Prediction

A protein contact map is a binary matrix representing the presence or absence of spatial contact between all pairs of amino acid residues of a protein. For a protein with sequence length $L$, the corresponding contact map is a binary L×L matrix C, in which each element $C_{ij}$ in the matrix is defined as:

$$C_{ij} = \begin{cases} 0, & if\ D < 8\ \text{Å} \\ 1, & otherwise \end{cases}$$

where $D$ is the Euclidean distance between $C_\beta$ atoms ($C_\alpha$ for glycine) of residue $i$ and $j$.

The contact in contact map is split into different ranges according to the sequence separation domain. The sequence separation domain for long, medium, and short-range

distances, and local distances are [24+], [12, 23], [6, 11], and [0, 5], respectively. Figure 1.3 shows separation domain for different ranges. The labels in the x and y axis refer to the residue index in the corresponding protein sequence (Vendruscolo, Kussell, & Domany, 1997).

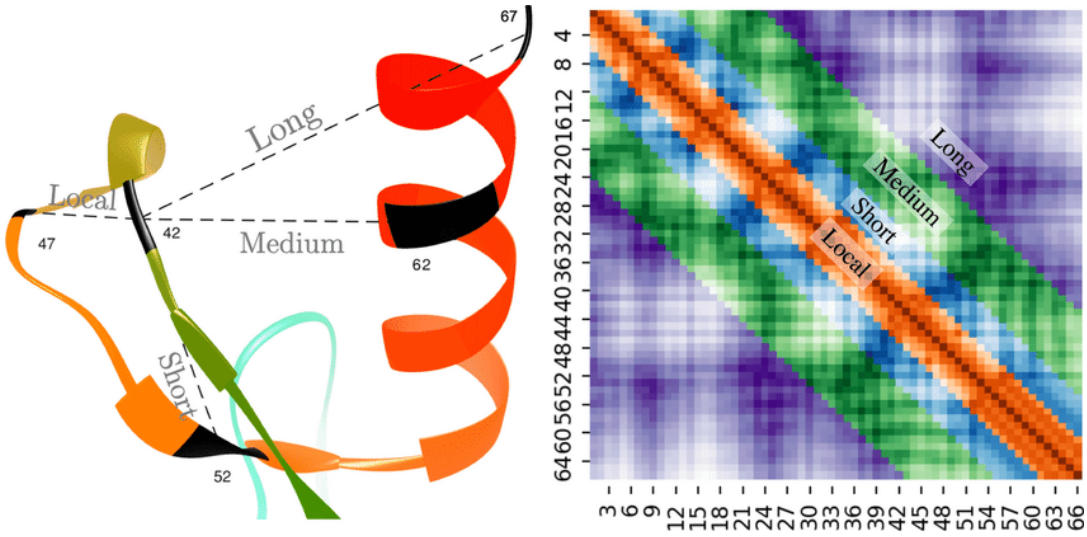


*Figure 1.3 Separation domain for different ranges*

In our research, another definition is used called distance matrix. For a protein with sequence length $L$, its distance matrix is also an L×L matrix C but contains real distance values as each element $C_{ij}$ in the matrix. The following Figure 1.4 shows an example of the contact map (left) and the distance map (right) of protein *6EMN-A*.

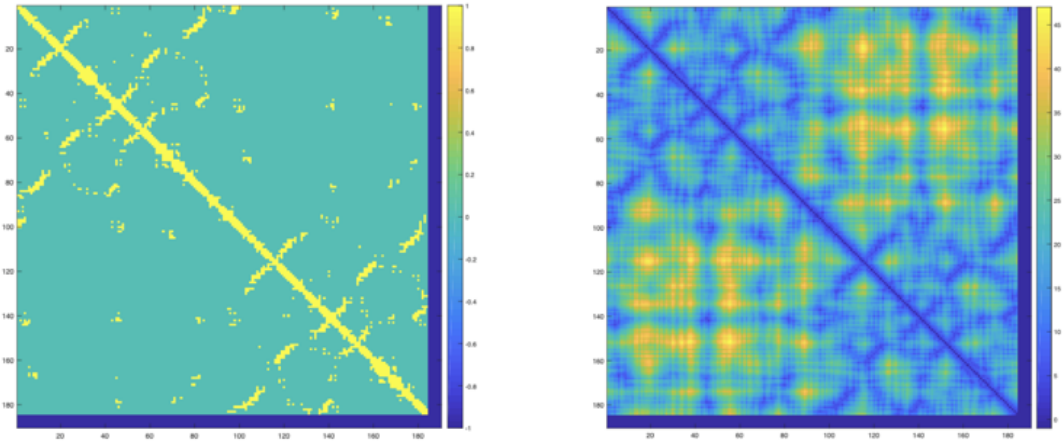

*Figure 1.4 An example of the contact map (left) and the distance map (right) of protein 6EMN-A*

## 1.5 Protein Structure Refinement

Homology-modeling methods can in some cases produce models with high performance, but in many cases starting models still contain significant errors. Protein structure refinement aims to bring moderately accurate template-based protein models closer to the native state. However, guiding the refine process towards the native state by effectively using restraints remains a major issue in structure refinement problem.

Structural averaging of molecular dynamics (MD) simulation trajectories (Mirjalili, Noyes, & Feig, 2014) and modeling with strong restraints to a high-resolution reference model have been shown to consistently improve model accuracy when starting models are close to the native structure. However, when starting model contains significant errors, the conformational phase space exceeds by orders of magnitude what can be explored using such methods, and little or no accuracy increase is observed. In contrast, coarse-grained conformation search and unrestricted simulation can sample more extensively (Stumpff-Kane, Maksimiak, Lee et al., 2008) but are affected by the inaccuracy of the energy

function, and therefore generally reduce the quality of the model rather than improve it. Due to the conflicting requirements for energy function accuracy and extensive sampling, protein structure refinement is still an outstanding challenge.

## 1.6 Contributions

This thesis makes the following contributions:

1. For single model QA problem, two extra deep residual neural network structures are implemented, trained, and tested, these two networks are based on a comprehensive set of model structure features, such as predicted and real secondary structure and solvent accessibility, distance matrix and $\phi$, $\psi$ angle. For each model, using these features as inputs, trained deep neural networks are able to predict a quality score in the range of [0, 1]. The deep learning methods already got comparable result compared with existing state of art single-model QA methods. To the best of our knowledge, it is the first time ultra-deep neural network is applied to this problem.
2. For quasi-single mode QA problem, a deep inception networks are developed for the model quality assessment (QA) problem based on templates of the target sequence and the model to be evaluated. The networks take an extensive set of 1-D and 2-D features of a model similar with the above deep residual neural network and output its predicted quality score in the range of [0, 1]. In this research we tested three training strategies and compared their performance. Experimental results using CASP targets show that the new method achieved state-of-the-art results, comparable with the best existing QA methods and outperformed existing methods on CASP QA stage 2 category on CASP 13 targets.

3. In addition to the deep learning related methods mentioned above, we also developed three new single-model QA methods, MMQA-1 MMQA-2 and MMQA-HE, with the combination of two heuristic methods, two stage machine learning and hierarchical ensemble. MMQA-1 and MMQA-2 applied two stage machine learning strategy, the entire feature set is divided into two different groups and uses completely different feature sets and training data in each stage of machine learning. MMQA-HE applied ensemble strategy not only on tree level but also on forest level in stage 1 of the two stage machine learning process, and further improved the performance. In CASP14, MMQA-1 ranked NO.2 in terms of average GDT-TS difference, and after we further improved the method and applied ensemble idea, MMQA-2 and MMQA-HE show improved consistently performance across different QA performance metrics.
4. With the code and model released by AlphaFold, we carefully analyzed the changes and challenges that AlphaFold will bring to the protein quality assessment problem. We simulated a new high performance decoy pool with the code and models released by AlphaFold2 , trRosetta and 3DRobot and tested the performance of different QA methods on this dataset, We also analyze the effective amount of information that different structural features and score functions can provide in the high performance model pool quality assessment problem based on this dataset.
5. For protein loop modeling, an efficient constant-time method of using AlphaFold2 for loop modeling called IAFLoop has been proposed. To predict the structure of a loop region, IAFLoop gives a moderately extended segment of the target loop region as input to AlphaFold2, runs a fast version of AlphaFold2 using a reduced database without ensembling, and uses RMSD based consensus scores to select the final output

models. Our experimental results on benchmark datasets show that IAFLoop outperformed naïve applications of AlphaFold2 by up to 17% in terms of RMSD error, while only using half of the time. Compared to the best previous methods, IAFLoop reduced RMSD error by at least 4 times.

6. For contact map prediction problem, a new two-stages multi-branch network based on fully convolutional neural network and inception network has been proposed. The first stage predicts the distance maps for different sequence separation domain. The second stage combines the original input feature set for stage one and the output distance maps from first stage as input and predicts contact map. The new method further improves the performance of MUFOLD contact prediction on the CASP13 dataset.
7. For protein structure refinement, we design and developed a refinement process MUFOLD-REFINE and achieved excellent results in the CASP13 competition. After that, we borrowed the inspiration from AlphaFold in CASP13, and further updated our refinement process with the distance distribution of the template pool as constraint. The updated refinement process has achieved better performance and has improved the performance of the start model on many targets.

## 1.7 Thesis Organization

This thesis is organized into the following sections:

1. Chapter 1 describes the introduction of computational biology and an overview of problems that this work focuses on.
2. Chapter 2 describes the background and related work of model quality assessment, protein loop modeling, protein contact map prediction and protein structure refinement.

The basic background of deep learning, the applications of deep learning in computational biology and the two versions of AlphaFold are also reviewed.

3. Chapter 3 introduces our work related to protein structure QA problem. Including single model QA method based on two extra deep residual neural network structures, quasi-single model QA method based on deep inception network, single model QA methods using two heuristic ideas, two stage machine learning and hierarchical ensemble, new simulated high performance dataset and related analysis, and the QA methods used to evaluate overall folding accuracy and overall interface accuracy for protein complexes we used in CASP15.
4. Chapter 4 introduces our new efficient constant-time AlphaFold2 based method used to solve the Loop Modeling problem and compares and analyzes the performance of generated loop region structure using different prediction process of AlphaFold2 and different length of sequence as input.
5. Chapter 5 introduces our deep neural network for protein contact prediction. The network consists of 4 inception networks in total divided into two stages. The stage 1 focuses on distance map prediction of different sequence separation domain and the stage 2 focus on contact map prediction.
6. Chapter 6 introduces our new refinement process with the heuristic potential function designed bases on distance distribution of the template pool and some case study of the new refinement process.
7. Chapter 7 summarizes all the previous work and describe the future work.

# CHAPTER 2. BACKGROUND AND RELATED WORK

## 2.1 Protein Quality Assessment

In general, we divide QA methods into two categories based on the information used in the evaluation process: single-model QA and multi-model QA. Single Model QA method only uses the information of the target model itself, and the multi-model approach requires a model pool and uses other models in the pool to predict the quality of any one model. If the method takes a single model as input and generate a reference pool by method itself, we call the method a quasi-single QA method.

### *Multi-Model QA Methods*

Usually, the performance of multi-model approaches is better than that of single model QA methods, many teams have done a lot of research on optimizing the naive consensus algorithm like MULTICOM (Cao, Bhattacharya, Adhikari, et al., 2015; Chen, Liu, Guo et al., 2021), Wallner (Elofsson, Joo, Keasar et al., 2018) and MUfoldQA_C (Wang, Wang, Xu, et al., 2018; Wang, Li, Wang et al., 2019), etc. However, there are also limitations for the multi-model QA methods. First of all, for same input model, the result of the multi-model QA method not only depends on the input model, but also depends on the distribution of the model pool. Different model pools will have a great impact on the evaluation results. Secondly, the optimization direction of the multi-model QA method is relatively single and difficult to be optimized stably. In CASP13, many excellent multi-model QA methods successfully beat naive consensus algorithm but in CASP14, the unoptimized naive consensus has achieved exceptional results again.

### *Single-Model QA Methods*

Different from multi-model approaches, single-model approaches can design and optimize algorithms based on multiple directions including energy, structure, angle, etc., or can comprehensively consider multiple aspects and use more complex algorithms for integrated analysis. Pure single-model approaches have fixed results for target model as the results only depends on the target model itself. In terms of single-model methods, we commonly divide them into potential functions and machine learning based methods. The former uses physics-based or knowledge-based potential functions to evaluate the quality of the model. Well-known ones include SBROD (Karasikov, Pagès, & Grudinin, 2019), Fisher (Rykunov & Fiser, 2007), RWplus (Zhang & Zhang, 2010) and GOAP (Zhou & Skolnick, 2011), etc. Methods based on machine learning usually collect the information from model itself and the output from different potential functions, then use it as input for further comprehensive evaluation using machine learning algorithms. The most famous machine learning based methods of is the Proq series of algorithms. Proq method released in 2003 uses a two-layer neural network to combine 12 input features (Wallner & Elofsson, 2003), Proq2 and Proq3 use support vector machine to process more complete and diversified input feature set (Ray, Lindahl, & Wallner, 2012; Uziela, Shu, Wallner et al., 2016), while Proq3D and Proq4 try to use Deep Learning related technologies (Uziela, Menendez, Shu, et al., 2017; Hurtado, Uziela, & Elofsson, 2018). Proq team have achieved excellent results in multiple CASP competitions, and they constantly update the mechanical learning algorithms used and ensure that they are always one of the state of art tools.

### *Quasi-Single QA Methods*

Quasi-single QA method tries to integrate the advantages of single model QA method and Multi-model QA method. It uses reference model information to improve performance. At the same time, since the reference model is generated by the method itself for the target sequence, it will not change with the different model pool, which enhances the stability of the method itself.

### *Compound Multi-Model QA Methods*

With the continuous update of machine learning algorithms and the wide application of deep learning algorithms in the QA field, more teams try to use complex machine learning algorithms and more comprehensive input information to improve the performance of QA algorithms. Using the consensus information used by the multi-model approaches as one of the input features of the mechanical learning algorithms, and the attempt to combine with the model's own information and improve performance also provides a new idea for the optimization and update of the multiple model QA algorithms. The use of consensus information has greatly improved the performance of the compound QA algorithm, and the features extracted from the target model itself makes the generated QA score pay more attention to the target model instead of relying on the distribution of the model pool. Many groups in CASP14 like MULTICOM-CONSTRUCT, MULTICOM-AI, MULTICOM-CLUSTER (Chen, Liu, Guo et al., 2021) and MESHI_CONSENSUS used this idea and achieved good performance.

## 2.2 Protein Loop Modeling

For the loop modeling problem, there are two main strategies: *ab initio* method and template-based method.

### *Ab-Initio Methods*

The ab-initio method is also considered a miniature protein folding problem (Fiser & Do, 2000). It usually includes initial sampling and ring conformation selection. It first generates a conformation through some statistical methods under geometric constraints and fits the ring conformation into the gap through a closed-loop algorithm. Then use some selection algorithms to select the final conformation in terms of minimizing the energy function.

ModLoop from MODELLER (Fiser & Do, 2000) first build a straight line in the loop region as the initial conformation of the loop, then use conjugate gradient minimization and simulated annealing to get the conformation with the lowest energy. Similar to ModLoop, another loop modeling program Loopy (Xiang, Soto, & Honig, 2002) first generates loop conformations by sampling torsion angle pairs. Then uses random tweak loop closure algorithm to close the loop. Later, RAPPER (de Bakker, DePristo, Burke et al., 2003) program generates the loop conformation by combining different fragments sampled by residue specific $\phi/\psi$ angles from one end of the loop towards the other end. Then they use SCWRL clash energy, and Samudrala-Moult potentials for final selections. RCD+ (López-Blanco, Canosa-Valls, Li et al., 2016) program uses the random coordinate descent algorithm to generate initial loop conformations, then the conformations are ranked by a distance-orientation dependent energy filter.

### *Template-Based Methods*

Template-based methods are also called database based or knowledge based method. These methods will search existing loop structures in a database like Loops in Protein (LIP) (Michalsky, Goede, & Preissner, 2003), Loop in Membrane Proteins (LIMP) (Hildebrand, Goede, Bauer et al., 2009), or other self-generated database by superimposing the stem region of the gap and then selects ones with low RMSD or high sequence similarity.

SuperLooper (Hildebrand, Goede, Bauer et al., 2009) is a program that searches LIP and LIMP databases to find loop candidates. Then all the candidates are scored based on sequence criteria and the RMSD of overlap regions. A more advanced database based tool FREAD (Choi & Deane, 2010) uses four main filters in database search phase: anchor $C_\alpha$ separations, sequence similarities, statistical energy function, and anchor RMSD.

### *Hybrid Methods*

Some other tools use the combination of ab-initio and template based methods to solve the loop modeling problem and achieve improved performance. For instance, NGK (Stein, Kortemme, 2013), Galaxy PS1 (Park & Seok, 2012) and Galaxy PS2 (Park, Lee, Heo et al., 2014). Among those methods, NGK performs sampling the loop structures based on the idea of combining intensification of torsion and parameter annealing strategies. Both Galaxy PS1 and Galaxy PS2 are loop refinement methods that starts with an inaccurate loop structure. The energy of Galaxy PS1 is optimized for application to the refinement of template-based models, while Galaxy PS2 is developed for higher performance for the near native models.

## 2.3 Protein Contact Map Prediction

When solving protein structure prediction, information about the contacts between pairs of residues can be a powerful constraint for determining the overall structural topology (Vendruscolo, Kussell, & Domany, 1997). Therefore, researchers have been studying contact map prediction methods for a long time. With the research of contact prediction problem, several different mainstream methods have emerged, and the performance of contact prediction is also improving. In the early days, researchers tried to apply purely statistical methods to predict contact maps. There are two types of statistical methods: the local statistical methods and global statistical methods.

Local statistical methods treat each pair of two residues in a sequence that are statistically independent of each other. It is not widely used because it is affected by the transfer effect between multiple residue pairs, which makes it impossible to distinguish directly and indirectly related signals (Weigt, White, Szurmant et al., 2009).

Different from local statistical methods, global statistical methods consider all other residual pairs when predicting a pair. The global statistical model commonly used to describe this joint probability distribution is the Pott model (Wu, 1982). Different types of methods have been proposed to infer the parameters of the Pott model. In addition to the traditional log-likelihood maximization method, other methods based on direct coupled analysis (DCA) have also been proposed. So far, pseudo-likelihood has proven to be the most successful contact prediction approximation (Ekeberg, Lövkvist, Lan et al., 2013), and it is widely used in different tools, such as GREMLIN (Kamisetty, Ovchinnikov & Baker, 2013), CCMPred (Seemayer, Gruber, & Soding, 2014) and plmDCA.

The invention of DCA in 2009 was a breakthrough to do contact map prediction from sequence (Weigt, White, Szurmant et al., 2009), one disadvantage of DCA methods is that they require a large number of multiple sequence alignments to provide accurate contact prediction. This problem has been overcome by using machine learning methods to refine the initial DCA prediction.

### *Machine Learning Methods*

Many traditional machine learning methods have been used to learn the mapping between features and contact maps, such as support vector machines (SVM) used by SVMCon (Cheng & Baldi, 2007) and SVM-SEQ (Wu & Zhang, 2008), and random forest used by PhyCMap (Wang & Xu, 2013). Many neural network or deep learning based methods have also been proposed and they have improved the performance significantly, such as NNCon (Tegge, Wang, Eickholt et al., 2009), DNCon (Eickholt & Cheng, 2012), and RaptorX (Wang, Sun, Li et al., 2017), etc.

In addition to the application of algorithms related to mechanical learning, another idea of making predictions based on the prediction results of other predictors is also used by many new predictors. This type of predictor is called a meta-predictor. Since each predictor has its own algorithms and advantages, if they are combined, we can take advantage of the characteristics of each predictor. For example, PconsC (Skwark, Abdel-Rehim, & Elofsson, 2013) combines the sequence features and predictions of PSICOV (Jones, Singh, Kosciolek et al., 2015) and plmDCA. Both metaPSICOV and RaptorX combine sequence features and predictions from PSICOV, mfDCA, and CCMPred. EPSILON-CP (Stahl, Schneider, & Brock, 2017) and NeBcon (He, Mortuza, Wang et al., 2017) combine the results of five and eight other predictors, respectively.

***Evaluation Metrics***

According to the purpose of using the contact diagram, there are multiple evaluation indicators. In this work, we follow the evaluation indicators in the CASP community (Monastyrskyy, Fidelis, Tramontano, & Kryshtafovych, 2011; Monastyrskyy, D'Andrea, Fidelis, Tramontano, & Kryshtafovych, 2014, 2016). According to the CASP definition, contact means that the distance between the $C_\beta$ atoms ($C_\alpha$ for glycine) of two residues is less than 8 Å.

The main performance indicator is accuracy, which is the ratio of the number of real contacts to the predicted contact with the highest score, as shown in the following equation:

$$precision = \frac{TP}{TP + FP}$$

where the *TP* is the true positive contacts and *FP* is the false positive contacts.

The contact in contact map is split into different ranges according to the sequence separation domain. The sequence separation domain for long, medium, and short-range distances, and local distances are [24+], [12, 23], [6, 11], and [0, 5], respectively. The definition of different contact ranges is shown in the following equation, and we usually evaluate three ranges, short, medium, and long separately, and ignore the local range as residual pairs in this range is considered always in contact.

$$Contact\ Range = \begin{cases} short, & 6 \le separation \le 11 \\ medium, & 12 \le separation \le 23 \\ long, & separation \ge 24 \end{cases}$$

In each contact range, we select the top $N$ scoring predicted contacts to evaluate and $N$ depends on the length of the protein. Generally, we choose $N$ as the following values: $L/2, L/5,$ and $10$.

There is another evaluation metric called F1 score used by CASP for ranking by default. F1 score is the harmonic mean of precision and recall in range of 0 and 1. It is defined by the following equation:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Figure 2.1 below shows an example of contact prediction output used for CASP evaluation and an overview of all the scores to be evaluated. In the sample prediction, we can see that each row has five values separated by spaces. The first two values are the residue IDs in the sequence. The next two values are the distance definition of in contact. If the Euclidean distance of $C_\beta$ atoms of these two residues is between 0 and 8 angstroms, the two residues are considered to be in contact. The last value is the predicted probability of these two residues are in contact. The predicted contact pairs are ranked using this probability score highest to lowest.

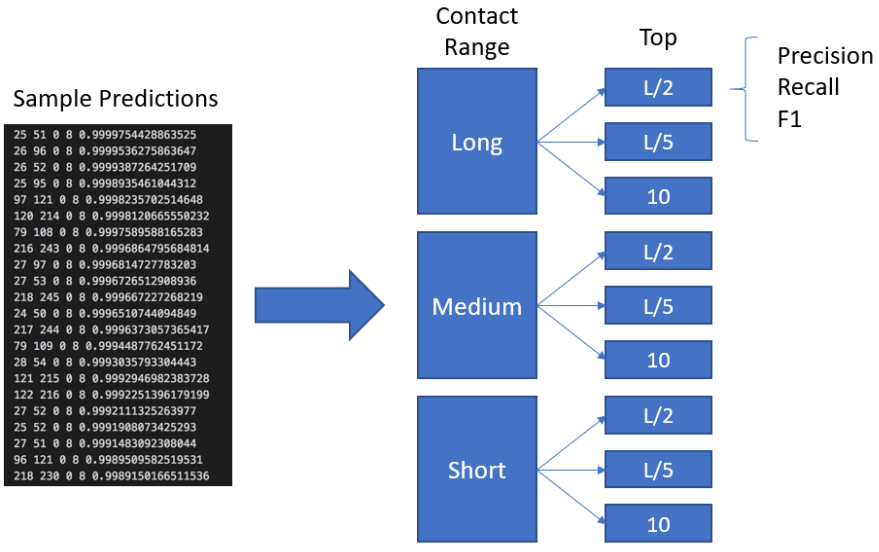


*Figure 2.1 Example of contact map prediction output and the evaluation scores.*

## 2.4 Protein Structure Refinement

The role of protein structure refinement is to take predicted template-based models and bring them closer to the native structure. Current methods for computational structure refinement rely on molecular dynamics simulations, related sampling methods, or iterative structure optimization protocols.

Currently most successful refinement method uses molecular dynamics (MD) simulation sampling, and energy minimization of physics-based and/or knowledge-based force fields. Thanks to the improved force field by combining physics and knowledge-based scoring terminology, enhanced sampling on longer time scales, and the use of overall averaging, protein structure refinement has made great progress.

The key issues revolve around the accuracy of the energy function, the inability to reliably rank multiple models, and the tradeoff between degree of refinement and consistency. On one hand, aggressive and unrestrained sampling around the starting structure that has the ability to produce large degree of refinement often deviates away from the native structure rather than towards it. On the other hand, in the refinement process, when the performance of the start structure is not that good, the model structure may need to be modified significantly, but applying the restrictions directly derived from the starting structure may prevent the occurrence of major changes in the structure.

## 2.5 Random Forest

The random forest algorithm is developed by Leo Breiman and Adele Cutler (Quinlan, 1986), it combines the idea of “bagging” and the random selection of features, it is an ensemble learning method for classification, regression, and other tasks, that operate by constructing a large number of decision trees at training time and outputting the class that

is the mode of the classes for classification case or of the individual trees mean prediction for regression case (Breiman, 2001). Random forests correct the overfitting problem of decision tree, and it provides the importance information of each input variable, which is suitable for information retrieving from a dataset of high dimension with noise, with these advantages, random forest became a state-of-the-art machine learning method and widely used so solve different biological problems (Manavalan, Lee, & Lee, 2014; Šikić, Tomić, & Vlahoviček, 2009; Wang, Yang, & Yang, 2009).

Given a training dataset $D$ of size $n$, random forest use bootstrap sample to generate different sample used to grow large number of unpruned regression tree. For growing m different decision trees, bootstrap sample will generate m new sample datasets with size $n$ by sampling from $D$ uniformly and with replacement. And as mentioned before, for each node, $t$ features will be randomly chosen and used to find the best split, which will which maximizes the information gain measure by Gini impurity (In ID3 it is the largest standard deviation). Decision trees will stop growing when reach the termination criteria. After repeat $m$ times, a forest with $m$ trees are generated.

The data not used for each tree in growing decision tree is called out of bag samples, which is used to estimate the error rate of the tree as well as the importance of each variable. When used for prediction, all trees in the forest will be tested and give their own outputs, the average of outputs from all the trees will be used as the final result.

## 2.6 Deep Neural Networks

In recent years, deep learning has had a huge impact on computer science and has achieved unprecedented performance on many machine learning problems, such as image classification (Krizhevsky, Sutskever, & Hinton, 2012), semantic segmentation (Garcia-

Garcia, Orts-Escolano, Oprea et al., 2017), and natural language processing (Collobert & Weston, 2008).

The development of the graphics processing unit (GPU) has made it possible to train very large neural networks faster. Generally, each deep neural network contains an input layer, hidden layers, and an output layer. The number of hidden layers can be very large, so a large amount of data can be used to train deep neural networks with tens of millions of parameters. These hidden layers can automatically extract complex abstract representations of data, rather than artificially designed representations. With the development of the field of deep learning, many advanced deep neural network architectures have been proposed, such as Convolutional Neural Network (Krizhevsky, Sutskever, & Hinton, 2012), Recurrent Neural Network (Mikolov, Karafiát, Burget et al., 2010), Residual Neural Network (He, Zhang, Ren et al., 2016), and Inception Network (Szegedy, Vanhoucke, Ioffe et al., 2016).

In recent years, deep learning technology has been increasingly used to solve computational bioinformatic problems like loop modeling (Li, Nguyen, Xu et al., 2017), contact prediction (Wang, Sun, Li et al., 2017), secondary structure prediction (Wang, Peng, Ma et al., 2016) and protein quality assessment (Wang, Li, & Shang, 2017), and some good results have been achieved. As AlphaFold has achieved breakthrough success in the field of protein structure prediction, it has further established the position of deep learning technology in solving problems in the field of bioinformatic (Senior, Evans, Jumper et al., 2020; Jumper, Evans, Pritzel et al.,2021).

### *Convolutional Neural Network*

Convolutional Neural Network (CNN) is a type of artificial neural network that dominates various computer vision tasks and is attracting interest in various fields. CNN is inspired by the organization of animal visual cortex. It aims to learn spatial hierarchies of features automatically and adaptively through backpropagation by using multiple building blocks (such as convolutional layer, pooling layer, and fully connected layer). The convolutional layer extracts feature from the input data based on a shared weight architecture of convolution kernels or filters that slide along the input features and provides a translational equivariant response known as feature maps. Local connection and parameter sharing are common methods to reduce the number of parameters. The pooling layer is used to perform non-linear downsampling. After several convolutional layers and pooling layers, there is a fully connected layer to make the final decision. The architecture of the convolutional neural network allows the network to learn more and more abstract features at a higher level. The following figure 2.2 shows the basic operation of using a 3×3 kernel convolution.

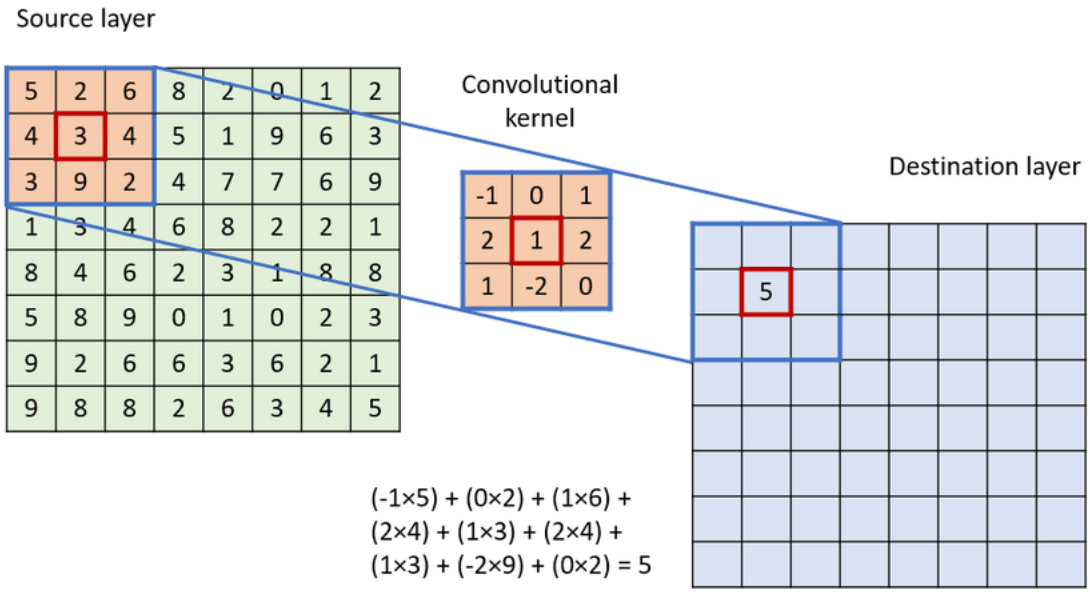


*Figure 2.2 Convolutional operation with a 3 by 3 kernel.*

***Fully Convolutional Network (FCN)***

A Fully Convolutional Network (FCN) is a neural network that only performs convolution (and downsampling or upsampling) operations (Long, Shelhamer, & Darrell, 2015). Equivalently, FCN is a CNN without a fully connected layer. Fully Convolutional Network (FCN) is proposed for image semantic segmentation. It can accept input of any size and produce output of corresponding size. The typical convolution neural network (CNN) is not fully convolutional because it often contains fully connected layers which do not perform the convolution operation, fully connected layers are parameter-rich, in the sense that they have much more parameters compared to their equivalent convolution layers. In addition, the fully connected layer maps the feature map from the convolutional layer to a fixed-length feature vector. Therefore, the input size must be fixed because the output length is fixed. To solve this problem, FCN was proposed, and it transforms the fully connected layers into convolution layers with kernel size 1×1 since the convolution

operation doesn't care about the input size and a 1×1 kernel can do an operation similar to fully connected layer for each pixel in the feature depth axis. The following figure 2.3 shows an example of using FCN to solve semantic segmentation problem.

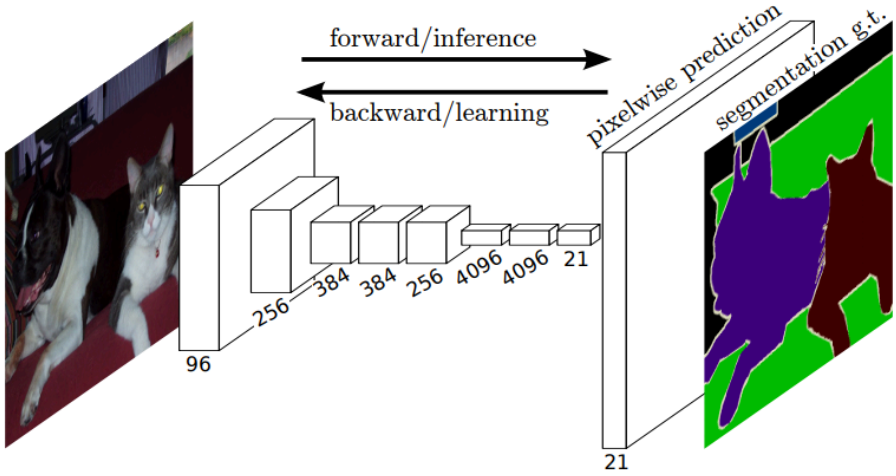


*Figure 2.3 An example of using FCN to solve semantic segmentation problem.*

***Residual Neural Network (ResNet)***

In the ImageNet competition, as the deep neural network used becomes deeper and more complex, a problem arises. People noticed that adding more layers to the neural network can make it more robust to image-related tasks but keep adding more Layers can also cause them to lose accuracy. This is not due to overfitting, because in that case, dropout and regularization techniques can be used to completely solve the problem. It appears mainly because in deeper layers the problem of vanishing gradients is more likely to occur. With the proposal of Residual Neural Network, the problem of training very deep networks has been alleviated (He, Zhang, Ren et al., 2016).

The most important point of ResNet is the 'Skip Connection', identity mapping. This identity mapping does not have any parameters and is just there to add the output from the

previous layer to the layer ahead. If the input of a building block is X, then the output of this block is X+X', in which X' is the non-linear transformation of X. However, sometimes x and F(x) will not have the same dimension, then the identity mapping is multiplied by a linear projection W to expand the channels of shortcut to match the residual, it can be implemented with 1×1 convolutions. This allows for the input x and F(x) to be combined as input to the next layer. The design of ResNet block can handle vanishing and exploding gradients when adding more layers to an already deep neural network very well, which makes ResNet become quite popular. The following figure 2.4 shows an example of a ResNet block.

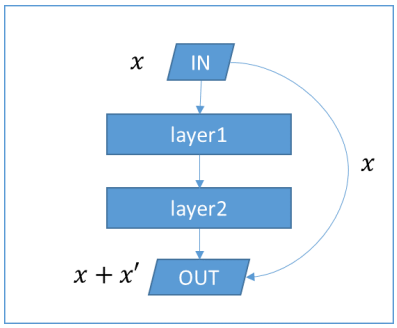


*Figure 2.4 Residual neural network building block structure.*

Compared with the image processing and recognition problem, computational bioinformatic problems is more complicated and has more complex input information, so it needs deeper neural network and a very large number of parameters to extract the connection between the input features. Resnet can perfectly solve the very deep network training problem, so it is widely used in solving various bioinformatic problems.

***Inception Neural Network***

The Inception network is an important milestone in the development of CNN classifiers. Before it was proposed, most popular CNN network structures simply stacked the

convolutional layers deeper and deeper, hoping to obtain better performance. But inception network (Szegedy, Vanhoucke, Ioffe et al., 2016) was designed to improve the performance by using different size of kernels in the inception block, in order to make the network "wider" instead of "deeper". It uses many techniques to improve performance. Its continuous evolution has led to the creation of multiple versions of the network .

The below figure 2.5 shows the "naive" inception module. It performs convolution on an input, with 3 different sizes of filters (1×1, 3×3, 5×5). Additionally, max pooling is also performed. The outputs are concatenated and sent to the next inception module.

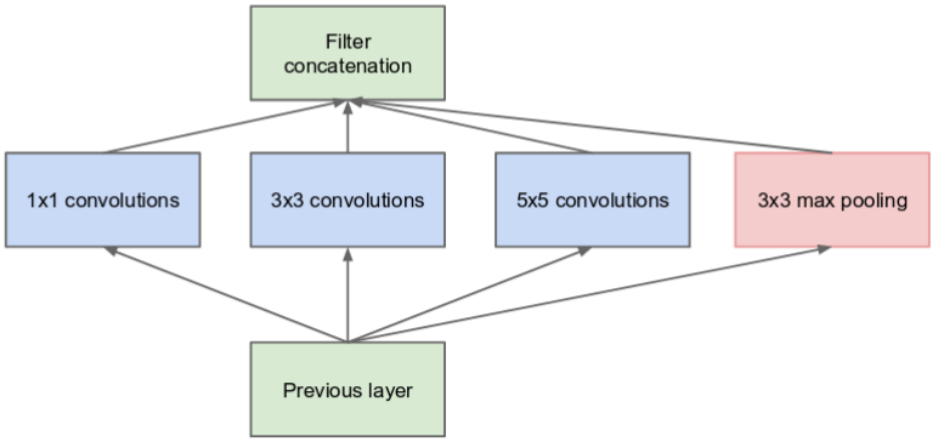


*Figure 2.5 An example of "naive" inception block.*

Inception v2 and Inception v3 were presented in the same paper. The authors proposed a number of upgrades which increased the accuracy and reduced the computational complexity. First, it factorizes 5×5 convolution to two 3×3 convolution operations to improve computational speed. Moreover, they factorize convolutions of filter size n×n to a combination of 1×n and n×1 convolutions. For example, a 3×3 convolution is equivalent to first performing a 1×3 convolution, and then performing a 3×1 convolution on its output.

This method is 33% more cheaper than the single 3×3 convolution. The below figure 2.6 shows the two inception blocks mentioned above.

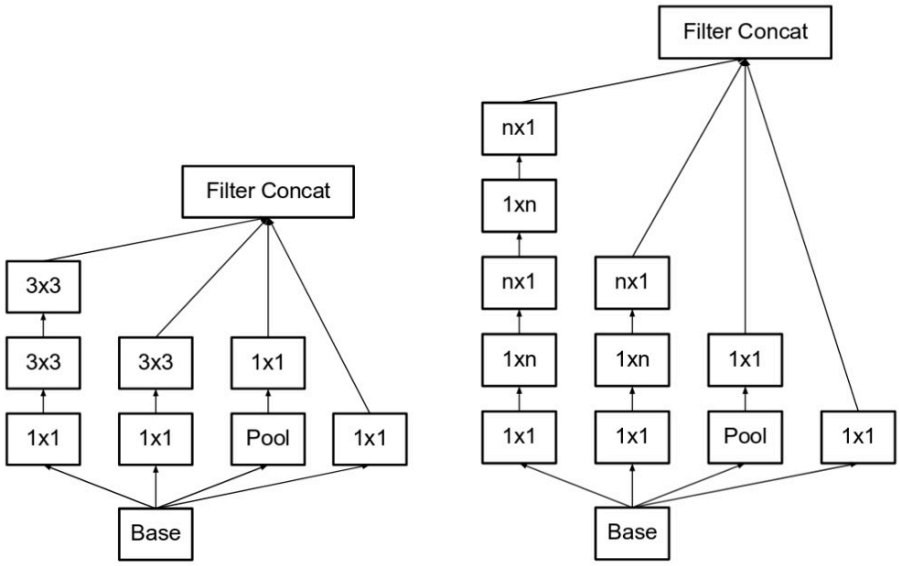


*Figure 2.6 Two version of Inception V2 blocks.*

## 2.7 AlphaFold from DeepMind

AlphaFold from DeepMind can accurately predict 3D models of protein structures and has the potential to accelerate research in every field of biology. In CASP13, AlphaFold first time predict hard targets with an average GDT_TS of 70%. And in CASP14, AlphaFold brings a further leap in protein structure accuracy, with the best model for targets at different difficulty level reaching an average GDT_TS above 90%.

### *AlphaFold in CASP13*

The central component of AlphaFold in CASP13 (Senior, Evans, Jumper et al., 2020) is a convolutional neural network that is trained on PDB structures to predict the distances $d_{ij}$ in between the $C_\beta$ atoms of pairs, i and j are index of residues of a protein. The model

predicts a discrete probability distribution $P(d_{ij}|S, MSA(S))$ for every ij pair in any 64 × 64 region of the L × L distance matrix. The full set of distance distribution predictions constructed by combining such predictions that covers the entire distance map is termed a distance histogram. To generate structures that conform to the distance predictions, AlphaFold constructed a smooth potential $V_{distance}$ by fitting a spline to the negative log probabilities and summing across all of the residue pairs. In our work, we borrowed this idea and generate a discrete probability distribution for every ij pair base on the searched template pool, and then constructed a potential function used for refinement process.

***AlphaFold in CASP14***

AlphaFold v2.0 provide the first computational method that can regularly predict protein structures with atomic accuracy even in cases in which no similar structure is known (Jumper, Evans, Pritzel et al.,2021). It is a novel machine learning approach that incorporates physical and biological knowledge about protein structure, leveraging multi-sequence alignments, into the design of the deep learning algorithm.

The AlphaFold v2.0 network comprises two main stages. First stage is the trunk of the network which processes the inputs through repeated layers of a novel neural network block that called Evoformer to produce an Nseq × Nres array (Nseq, number of sequences; Nres, number of residues) that represents a processed MSA and an Nres × Nres array that represents residue pairs. The Evoformer blocks contain a number of attention-based and non-attention-based components. Stage 2 followed Stage 1 is the structure module that introduces an explicit 3D structure in the form of a rotation and translation for each residue of the protein.

Multiple versions of AlphaFold have been released including the full version and two light version which utilize smaller database, the resources and time required for different versions of AlphaFold to run are also significantly different. In our work we tested and analyzed the performance of AlphaFold on solving loop modeling problem and simulated a high performance decoy pool with the code and models released by multiple versions of AlphaFold.

# CHAPTER 3. APPLYING MACHINE LEARNING AND DEEPLEARNING TO PROTEIN MODEL QUALITY ASSESSMENT PROBLEM

## 3.1 Motivations

Initially, QA methods were designed to try to estimate the quality of protein structures from different aspects, including statistical analysis, generating different energy scores, analyzing angels, secondary structures, etc. With the increasing application of machine learning algorithms in the field of bioinformatics, researchers began to design different machine learning methods to combine the results of different basic scoring functions and structural information to further improve the performance of QA methods.

After CASP12, deep learning began to be gradually applied to solve the problem of protein structure quality assessment, and we designed and implemented the first extra deep neural network (26-layer residual neural network) that tried to solve QA problems.

Subsequently, we try to apply the template information obtained by searching the database based on the target sequence to solve the QA problem. We designed and implemented the first target based deep learning QA method MUFOLD-INC, due to the usage of template structure information, we believe that MUFOLD-INC should be classified as a quasi-single model QA method. The performance of MUFOLD-INC net has reached the level of state of art quasi-single model QA method.

While constantly exploring how to use deep learning technology to solve the protein structure QA problem, we also try to use random forest with multiple stage machine learning and hierarchical ensemble to solve the protein structure QA problem, the single model QA method MMQA designed based on random forest and two stage training

achieved state of art performance in CASP14 among many single model QA methods, after replacing naive random forest with hierarchical ensemble model, the performance of MMQA-HE has been further improved.

With the development and progress of protein structure prediction methods, protein structure quality assessment has also attracted the attention of many researchers, and this attention has also reached its peak at CASP14. In CASP14, more than 70 groups participated in the competition, and various state-of-the-art machine learning and deep learning techniques were applied to QA problems. But with the success of AlphaFold2 in CASP14 and the release of AlphaFold2 source code and models, the problems faced by the entire QA field have changed a lot. We simulate a new high-performance decoy pool using AlphaFold2 and trRosetta, we test the performance of different QA methods on this dataset and analyze the information that commonly used features can provide on QA on this dataset.

In CASP15, estimating the accuracy of single protein models has been removed as CASP believes that these methods cannot compete with modeling method specific estimates. And accuracy estimation for protein complexes has been added to the competition as a new category. In the accuracy estimation of protein complexes, the global score that CASP focuses on contains two accuracy scores for a model, one is used to predict the overall folding accuracy and the other one is used to estimate the overall interface accuracy. In response to this change, we also designed new QA methods based on US-align and DockQ that can be used on protein complexes to estimate the related accuracy.

## 3.2 Problem Formulation

Protein quality assessment problem is formulated as following: given the animal acid sequence of a target protein and a predicted decoy structure, return a quality score that

reflects the similarity between the decoy structure and the native protein structure of the target sequence. In this work, one of the most commonly used measure GDT-TS is used to evaluate the similarity between decoys and native structure. GDT-T score ranging from 0 to 1, higher value indicating more similar. If the value is 1, the structures being com-pared are identical.

## 3.3 MUFOLD-DRN

### 3.3.1 Motivations

In terms of the single-model methods, before we design this method, the commonly used ones are potential functions and machine learning methods, but most of the QA methods are using traditional machine learning algorithms like linear regression and SVM, some more complex methods like random forest have been tested but still limited by the traditional machine learning idea. Most of the machine learning methods on quality assessment are not using lots of parameters and pairwise features, which contains the most information not used as input. They only focus on how to combine other basic QA tool's results. During CASP12, more and more teams already focus on deep learning methods, especially Convolutional Neural Networks. Deep learning already became one of the most popular directions for researchers to solve computational bioinformatics problems. For problems like protein secondary structure prediction and protein contact prediction, not only traditional Neural Networks methods, some complex extra deep networks also been tested and got amazing result, extra deep Neural Networks like Deep Residual Networks generate much better result compared with earlier state-of-art methods. Back to QA problem, some of the deep learning methods are tried to solve QA problem, but they are

not directly working on the problem, some are used to classify the models into different quality levels, some are used to measure how many structure errors the model may have. Some methods tried to use deep learning, but the network structure is very simple and shallow. There is still no successful example of using deep learning algorithms directly solve the QA problems.

This work borrows the idea of Deep Residual Neural Network from image recognition. Residual network gives extra layers to a network structure without a large increase on training time and reached a high performance in several computer vision challenges such as image recognition and object recognition. When evaluate the quality of a 3D model, the distance matrix will become one of the most important feature that contains lots of information, the information can be used to connected to many other structural properties like secondary structure, solvent accessibility and contact information. As the 3D structure of a protein can be rebuilt based on its distance matrix, and distance matrix can be treated as an image, when we evaluate the quality of a 3D structure, it is also evaluating the quality distance of the structure.

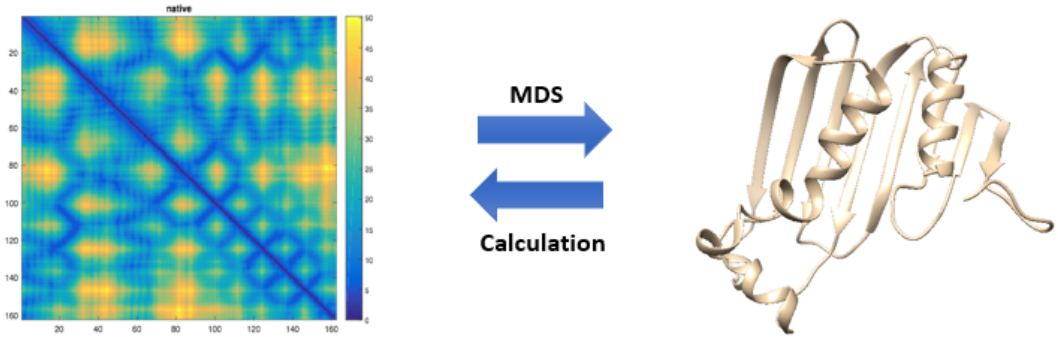


*Figure 3.1 Protein 2D distance map of $C_\alpha$ atoms (left) and the corresponding 3D structure (right) can be converted to each other.*

### 3.3.2 Pipeline and Method

Limited to the resource, the pipeline and the network is trained for structures with fixed length of 50. For different model with different length, if the length is less than 50, the input to the network will be padded with 0. If the length is larger than 50, all the features will be prepared for the whole model and then cut into fixed length, take 90 as an example, the model will be cut into two part 1-50 and 41-90, the final QA result of the model is calculated by average GDT-TS score of different parts. Figure 3.2 shows the prediction pipeline of MUFOLD-DRN.

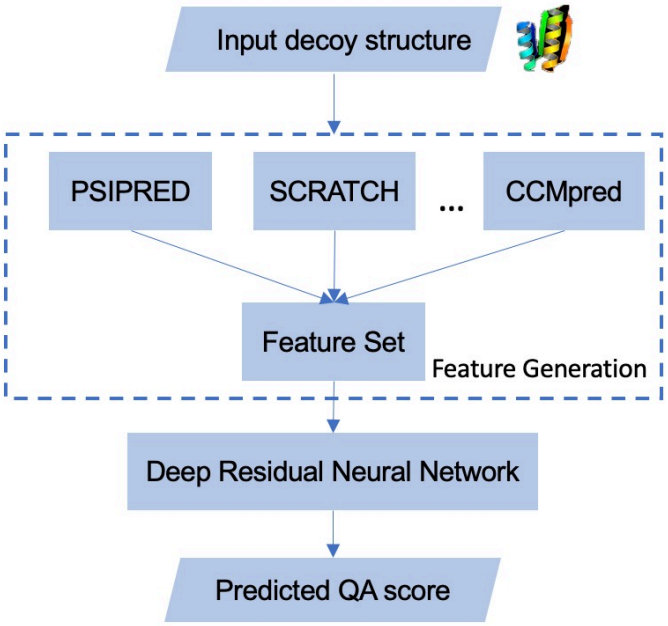


*Figure 3.2 Prediction pipeline of MUFOLD-DRP*

***The Residual Block***

Both of the two deep neural network structures of this work are consisting of some residual blocks. Figure 3.3 shows an example of a residual block consisting of 2 parts in its residual function, each part consists of 1 batch normalization, 1 convolution layer and 1 activation layer. For the shortcut, there will be a convolution layer if the input X needs

to match the shape of output X+1. The activation layer conducts a nonlinear transformation of its input without using any parameters. ReLU activation function is used here in our residual block.

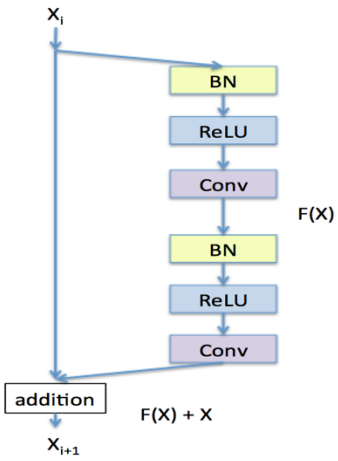


*Figure 3.3 Example of a residual block*

***Feature Extraction***

In this work, both sequential features and pairwise features of 3D structure is used, all the structures are generated base on the sequence and the 3D coordinate of the structure. The sequential features include protein sequence profile, predicted 3-state secondary structure and 3-state solvent accessibility base on the sequence, real 3-state secondary structure and 3-state solvent accessibility base on the 3D structure, and $\phi/\psi$ angle of each residual. For 3-state secondary structure and 3-state solvent accessibility, the predicted features are using the probability value, and real features are using one-hot representation. Pairwise features include direct co-evolutionary information generated by CCMpred, pairwise potential, distance matrix of the model, predicted contact map base on sequence and real contact map generated by the 3D coordinates.

### 3.3.3 Deep Neural Network Structure

In this work, two extra deep residual neural network structures are implemented, trained, and tested, following is the details of the structure of the two networks.

***Parallel Deep Residual Network (PDRN)***

The Parallel Deep Residual Network includes two modules, the first module is a 1D Residual Network with the depth (i.e., the number of convolution layers) 8, which means there are 4 residual blocks. The first module is taken sequential features as input, as the length of the input is fixed to 50, the input of the network is $34 \times 50$. The filter size (i.e., window size) used by a 1D convolution layer is 17, with the 8 convolution layers, it can cover all the information of length 50, the filter number will be fixed to 32, after the 8 convolution layers and flatten, the first module will generate 1600 features which will be combined with the output of the second module. The second module of The Parallel Deep Residual Network will take pairwise features as input; the input size of the network will be $50 \times 50 \times 5$. 8 Residual blocks with 16 convolution layers will be stacked in this module and an ending convolution layer will be added after 8 residual blocks. The number of the filters used in 2D residual network is increased as 8, 16, 16, 32, 32, and the size of the filter is $3 \times 3$. After flatten operation this module will generate 2500 features. There will be two fully connected layers used to generate one output base on the combined 4100 features. Figure 3.4 shows the illustration of the structure of the Parallel Deep Residual Network.

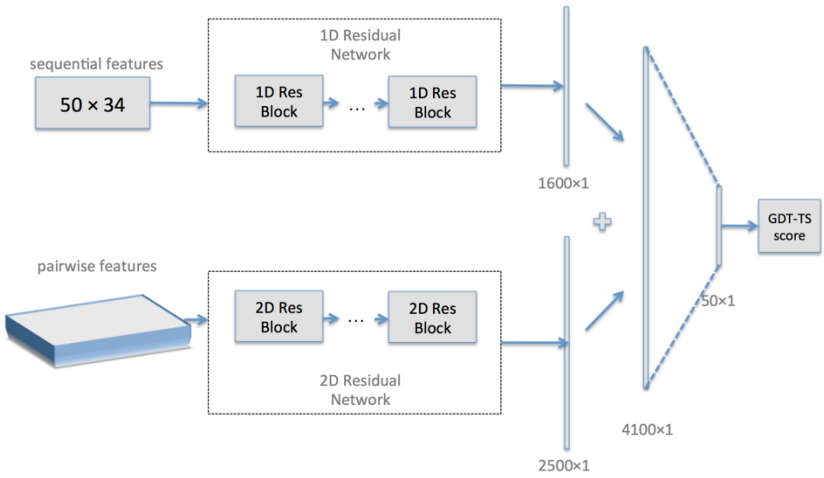


*Figure 3.4 The illustration of the structure of the Parallel Deep Residual Network*

***Vertical Deep Residual Network (VDRN)***

The Vertical Deep Residual Network also includes 1D Residual Network module and 2D Residual Network module, the structure of each module is not changed. The different between two network structures is for Vertical Deep Residual Network the sequential features generated by 1D Residual Network module will be converted to pairwise features and combined with other 5 pairwise features and fed into the 2D Residual Network module. The conversion process is similar to outer product. Let $v_i$ be the feature vector storing the output information for residue *i*, for a pair of residues *(i, j)*, we concatenate $v_i$ and $v_j$ to a single vector and use it as one input pairwise feature of this residue pair. After combined with the original 5 pairwise features, the final input size for 2D Residual Network module will be $50 \times 50 \times (32 \times 2 + 5)$. The number of the filters used in 2D residual network is fixed as 32, and the size of the filter is $3 \times 3$. Also, two fully connected layers will be added after the 2D Residual Network to generate one output score. Figure 3.5 shows the illustration of the structure of the Vertical Deep Residual Network.

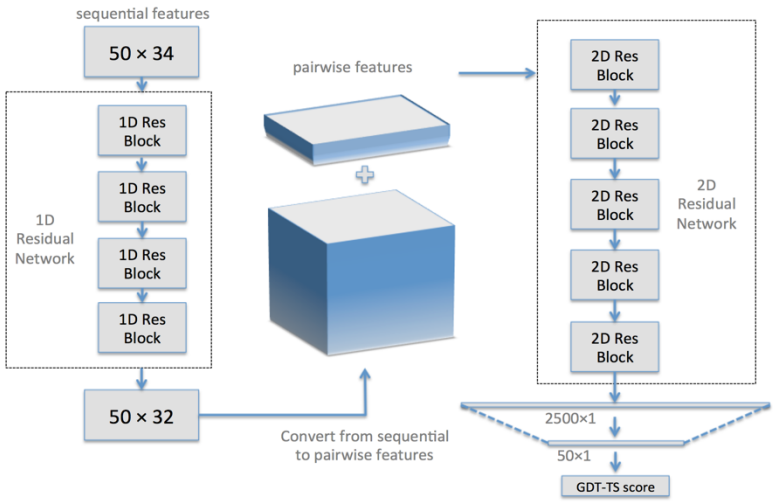


*Figure 3.5 the illustration of the structure of the Vertical Deep Residual Network*

### 3.3.4 Evaluation

***Benchmark Dataset***

The two networks are trained and validated using CASP 10 and CASP 11 dataset and tested on CASP 12 dataset. For CASP 10 and CASP 11 data, 90% of the dataset are used as training dataset and left 10% are used as validate dataset, as this will be a supervised learning problem, so when prepare the dataset, all decoys are cut based on the native protein released, removed the part without corresponding native structure. Then all the targets are cut into fixed length 50, 778 small targets are generated with corresponding decoy dataset. Among these targets, 700 targets and corresponding decoy dataset are used as training dataset and 78 targets are used as validation dataset used to validate the model.

Since CASP10, the quality assessment (QA) task has been divided into two different stages. Stage 1 prediction pool consists of the 20 prediction models with a large performance range. Stage 2 prediction pool consists of the 150 selected good prediction models. For the test dataset, two datasets are used, the first one is CASP 12 set150 and the second one is CASP12 set20. The features are generated for the whole decoy and then cut

based on the decoys. The absolute difference between predicted GDT-TS and true GDT-TS score is used to check the performance.

***Absolute Difference (Predicted vs. Observed)***

In this work, we chose one of the judging criteria of QA category used in recent CASP competition. The criteria "differences (predicted vs observed)" used in CASP is calculating the average of difference between predicted QA scores and corresponding GDT-TS values between the predicted models and the native structure of the target protein, which reflects the prediction ability of a QA method. It can be defined as:

$$Diff_{average} = \frac{1}{N}\sum_{i=1}^{N} ||G - G'||$$

Where *N* is the number of decoys in the pool, *G* is the real GDT-TS score between decoy and native structure and *G'* is the predicted GDT-TS score.

***Training Process Analysis***

The program is tested with batch size equals 5, learning rate set as 0.001, and totally trained for 5 epochs. At the beginning of training, all the GDT-TS scores of the models of one target are generated very similar, as the models of same target share the same sequence and predicted sequential features, but as the training process going, the models can specify the quality of different models better and better. Figure 3.6 shows the result of validate dataset on different saved models of Vertical Deep Residual Network.

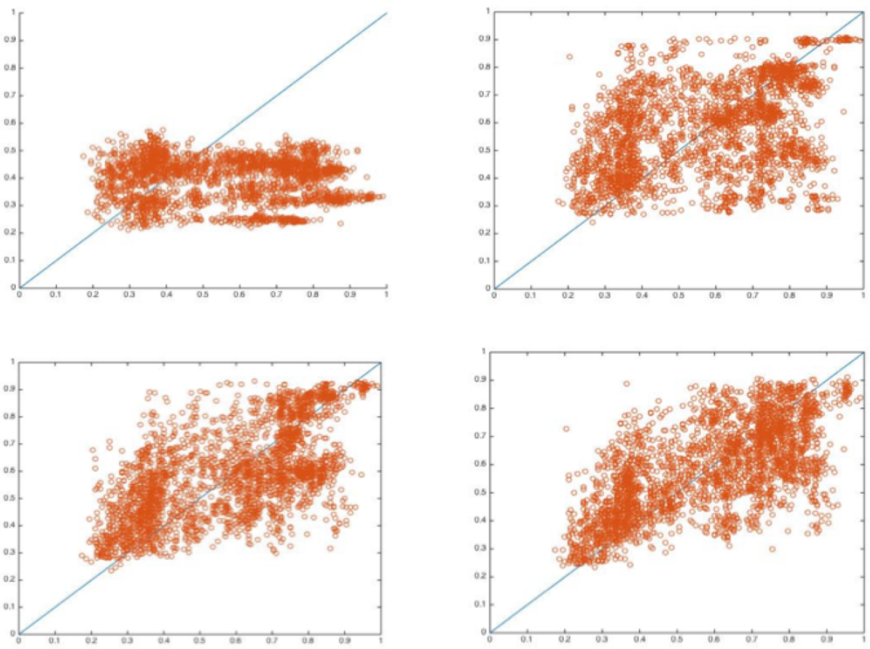

*Figure 3.6 Training process of VDRN*

***Results on CASP12 dataset***

Both PDRN and VDRN are validated and saved after each 500 steps, the validate dataset is used to select the best model and test with the two test datasets. Figure 3.7 shows the performance on two different test datasets.

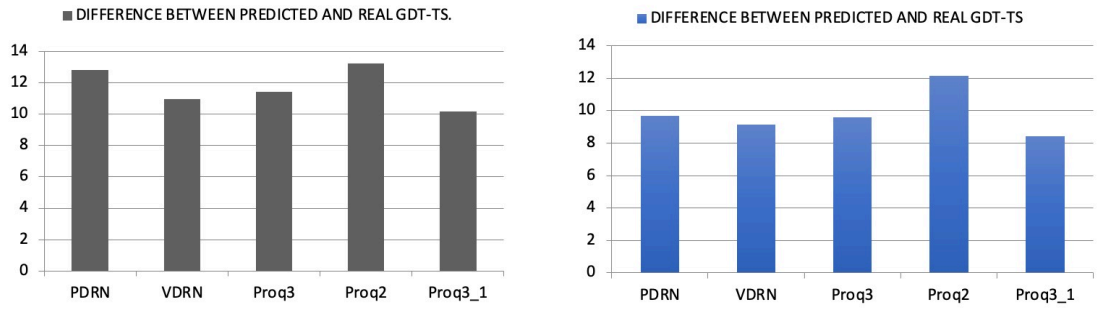


*Figure 3.7 Difference between predicted and real GDT-TS for set150 (left) and set20 (right)*

From the result we can see the two deep learning methods already be able to generate result comparable with other top single model QA methods, but with limited training

dataset and quality of the features, it is still worse than the best result handed in by Proq3_1. Also, for the structure of the network, the Vertical Deep Residual Network shows better result than the Parallel structure for both set150 dataset and set20 dataset.

### 3.3.5 Conclusion

The idea of using deep neural network on single model QA problem still needs more test and parameters evaluation to improve the performance as there is still not a very success network structure can achieve outstanding result compare with other methods. For many of the predicted features used in this work, the accuracy is not achieved state-of-art, which means the quality of these features may limit the performance of the result. Based on the result of two test datasets, the Vertical Deep Residual Network shows better result than the Parallel structure, which means doing conversion, connect the sequence features and pairwise features and doing the learning is better than split the two representation and just connect the sequence features and pairwise features through the last fully connect layers.

The test for how to use deep neural network to solve QA problem still need more working and has large space to improve. The first aspect is the way to generate the final GDT-TS result, the way used to cut the decoys and then use the average GDT-TS score as the final result in this work is only one of the simplest methods, use slides window with different strides can be tried to cut the decoys and use weighted methods to combine the GDT-TS scores generated for each part. The ideas of the deep residual network structures are tried, but the structure details still need more test, if residual neural network is used, more layers with different configuration can be tested, also there are many parameters evaluation works can be worked on.

## 3.4 MUFOLD-INC

### 3.4.1 Motivations

In efforts to combine the best of both worlds, a new group of algorithms called quasi-single model QA methods has been proposed. They borrowed the “consensus” idea from multi-model QA methods but generate their own reference pool to avoid the shortcomings attached to using the external model pool. However, such approaches also create another problem that is to generate a high quality reference model pool. In order to avoid this limitation, people have found that the consensus idea is not used directly, and other similar alignment structural information can also be used in other ways.

MUFOLD-INC explores a new direction of solving QA problem with deep learning by proposing a per-target trained deep learning QA strategy, that is, for each target sequence, a deep neural network will be trained using the same processing pipeline and network structure. This work borrows the idea form Deep Inception Learning from computer vision. As the 3D structure of a protein can be reconstructed based on its distance matrix, and distance matrix can be treated as an image, so the problem of evaluating the quality of a 3D structure can also be treated as evaluating the quality of a 2D distance matrix image. The distance matrix can be connected with other structural features and information extracted from the sequence, to evaluate if the structure is matched the target sequence. In this work one deep inception network structure has been designed and tested with three different training strategies, the training dataset is prepared by MUFOLD server used in CASP13. The network takes in the 1D features extracted from the sequence and converts them to 2D, then combine the converted features with 2D features. The output is a predict quality score in range of [0, 1].

### 3.4.2 Pipeline and Method

The whole pipeline of MUFOLD-INC includes training dataset preparation and network training. The basic idea is to extract useful information from the similar target sequence and use the dataset built by these similar sequence to train deep inception network for quality assessment. This network will generate a quality score in range [0,1], figure 3.8 is an overview of the method.

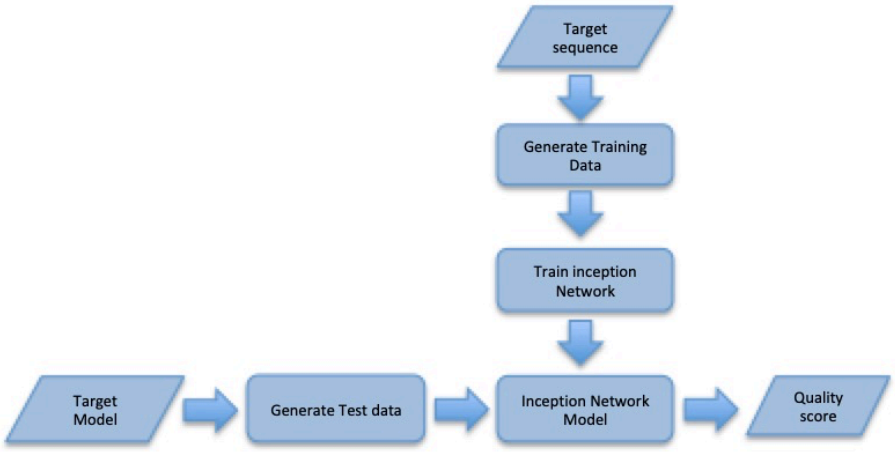


*Figure 3.8 An overview of MUFOLD-INC pipeline.*

***Inception Neural Network***

The Inception network was an important milestone in the development of CNN based deep neural networks. Prior to its inception idea, new CNN based network structures are mainly just stacking convolution layers deeper and deeper, in order to get better performance. Instead of choosing what filter size to use in the convolution layer and if pooling layer will be used, inception will do them together. The idea of the inception layer used in this work is from inception V3. In this version, the filter banks in the module were expanded by made wider to remove the representational bottleneck. If the module was

made deeper instead, there would be excessive reduction in dimensions, and hence loss of information.

### *Training Dataset Preparation*

Training dataset preparation process includes three stages, stage one will generate template set based on the input target sequence. Stage two will generate structure predictions for each template in the template set generated in stage one as the training decoy set. The third stage will generate different features for all the decoys generated in stage two to generate the final training dataset as the input of the deep inception network. Figure 3.9 shows the entire process of training dataset preparation.

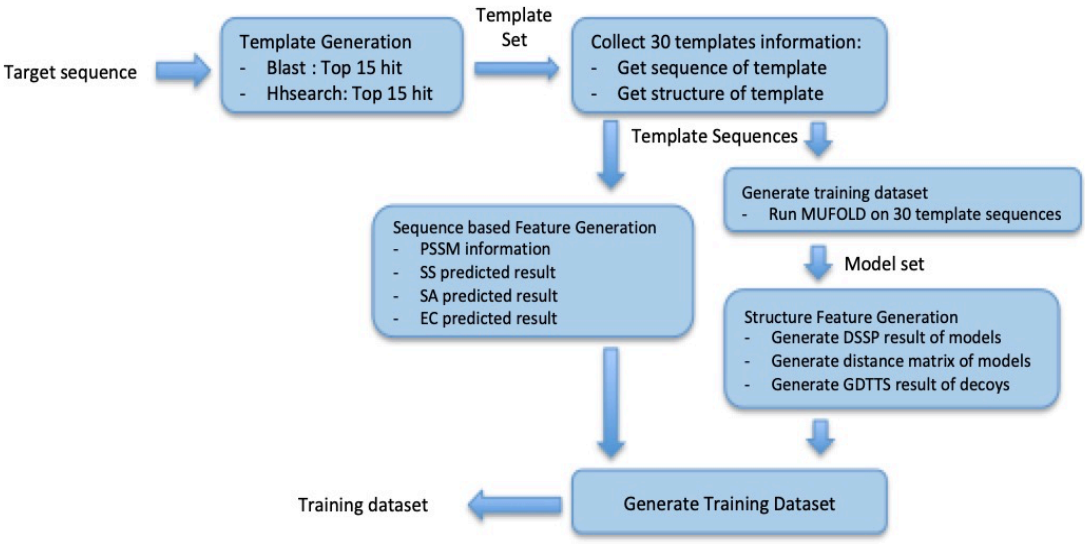


*Figure 3.9 Entire process of training dataset preparation.*

Template set generation stage will take the target sequence as input. Usually target sequence is the sequence of the target model or the model set. The sequence will be used to search for template similar to itself using Blast and Hhsearch, top 15 results from each

tool will be kept and used to compose the template set. The template set will be filtered to remove the redundant hit by calculating the pairwise GDT-TS and using the threshold 0.98.

Second stage will use the sequence of the templates in the template set generated in stage one. The sequence will be given to MUFOLD server to generate structure predictions and these decoys will be used as the training decoy set. In order to generate more predictions with varies accuracy, MUFOLD will use Blast and Hhsearch to search template, and generate decoys using all the hits by fully extending the templates. The native structure of templates in the template set generated in stage one will be used to generate GDT-TS value with the decoys generated by MUFOLD server, which will be used as the training label.

***Two Training Dataset Preparation Strategies***

It is usually hard to find templates that can cover the whole target sequence for some hard targets, then after using MUFOLD server to generate the prediction, the model in the model set usually cover only a part of the original target sequence. After convert to 2D, a model covers 70% of the original target sequence can only cover 49% of the 2D input, and all other parts are padded with 0, this will lead to training problems and likely to cause the training process crash. So, when generating the template set, two strategies will be considered. The first strategy will generate template set includes only the matched range, and for the second strategy, the template will be fully extended on both head and tail. The fully extended operation will increase the coverage of the training dataset, but also include noise. MUFOLD server will generate structure predictions based on both template sequence sets, and the model will be used to generate two training datasets. In this work

we will compare the performance using different training datasets generated by different strategies.

***Feature Extraction***

After the training decoy set preparation, it will be filtered by the coverage of the decoy. The decoys covering less than 60% of the targets sequence will be removed from the decoy set. For the remaining decoys, both predicted features based on the sequence and the features based on the structure of the decoys will be generated. The sequential features include protein sequence profile generated by Blast, predicted 3-state secondary structure generated by Psipred and solvent accessibility generated using Solvpred based on the sequence of the templates in the template set. The real 3-state secondary structure and solvent accessibility are generated by DSSP based on the 3D structure of the decoys in the decoy set. Before being fed into the deep inception network, all the 1D sequential features will be converted into 2D by stacking the data of the amino acid on position *i* on the data of the amino acid on position *j*. Besides the 1D features mentioned, 2D features including co-evolutionary information generated by CCMpred, PSICOV and EVfold, and the distance matrix of the decoy will also be calculated and used to train the network.

### 3.4.3 Deep Inception Network Structure

This section will show the design of the inception block and the structure of the whole deep inception network used in this research.

***The Inception Block***

The inception block design in this work borrows the idea from inception V2. Instead of using filter size n×n in basic inception block design, we used combination of 1×n and n×1

convolutions as this design will reduce the computation by 33% compared to the single 3x3 convolution.

Instead of simply stacking the 1×n and n×1 filter, we choose to parallel them to prevent excessive reduction in dimensions from causing loss of information. Also, we compared the performance of basic inception block, and this upgraded version, and finally we choose to use this design as it shows better performance.

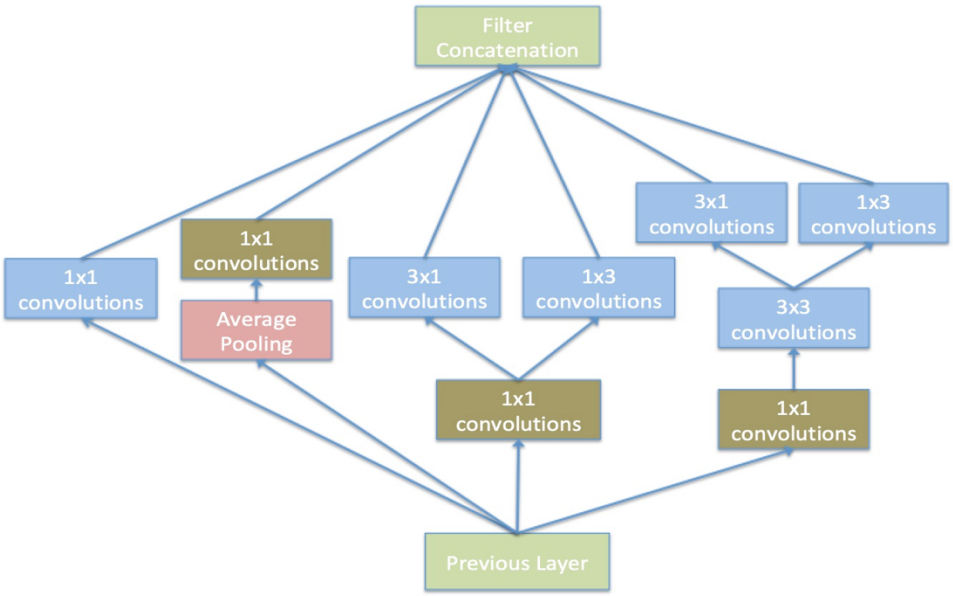


*Figure 3.10 Inception Block design.*

***Deep Inception Network***

Different from normal computer vision problem, computational bioinformatics problems usually got larger depth on input data. In this problem, 1D protein sequence profile has the shape of $l \times 20$, in which $l$ is the length of the sequence. After being converted to 2D, the shape changes to $l \times l \times 40$. Combined with secondary structure and solvent accessibility related features, co-evolutionary related features and distance matrix, the input data shape will achieve $l \times l \times 60$. As inception network will make the network

wider, we added more traditional convolutional layers to reduce the depth in order to avoid the rapid reduction of dimension.

The deep inception network takes the input with shape 1 × 1 × 60, first using two convolutional layers with 64 channels, then followed by five inception blocks. Based on the design of the inception block showed earlier, the output shape from the blocks will be 1 × 1 × 384. Four convolutional layers are added after the inception part, the number of the channels is reduced to 64, 16, 4 and 1. After the last convolutional layer fixed the shape to 1 × 1 × 1, two fully connected layers are added to generate the final output.

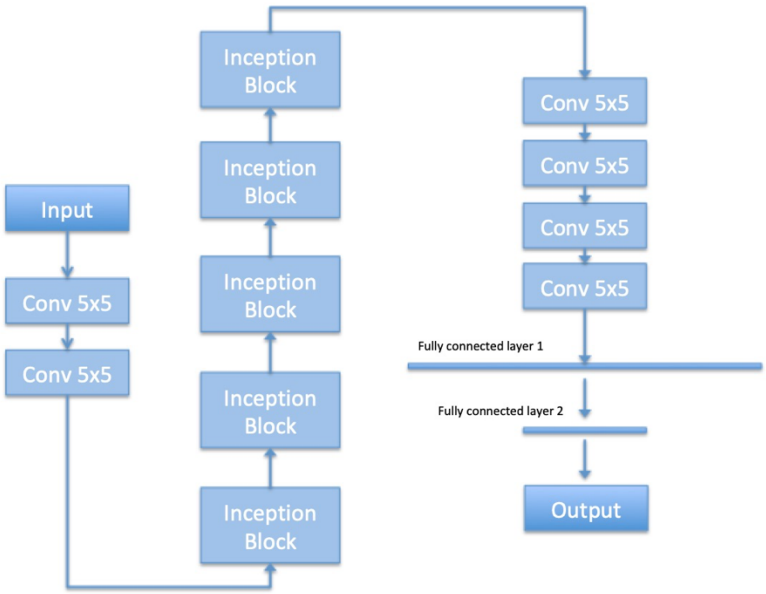


*Figure 3.11 The illustration of the structure of the deep Inception Network*

***Three Different Training Strategies***

The whole training dataset preparation process will search templates for two times, the template hit usually can't cover the whole target sequence, so the decoys used for training usually can cover only less than 70% of the target sequence. In order to fix the input size

to $l \times l \times 60$, all the missing parts are padding as 0. But later we noticed that when training data includes too many zeros, sometimes the training process crashed. Even though the training still trains good models if the process goes well, this coverage problem will still cause the time consuming problem more time is used on training. In order to fix this problem, we tried different training strategies with different training datasets.

***First Training Dataset***

The first training dataset will generate template set including only the matching range, this strategy ensures the information in the training is more reasonable as they have similar sequence to the target sequence. But the related problem is for some hard targets with fewer templates, the coverage of the whole training dataset can be quite small, most of the decoys cover less than 70% of the target sequence, after covert into 2D input, more than 50% of the input will be padded with 0.

***Second Training Dataset***

The second training dataset will fully extend the templates on both head and tail when generating the template set in stage one. This will make most of the templates cover almost the whole target sequence, and the coverage of the whole training dataset of hard targets enjoy substantial growth. But this also causes some hidden troubles. The fully extended parts are not real template, which means they are not similar to the target sequence and may become noise information when training the model.

***Three Training Strategies***

In this work, we tested three training strategies. For the first two strategies, we only use the first training data set and the second training data set for training. For the third training

strategy, we use the second training data set to pre-train the model for 5 epochs. Then continue to use the first training data set to train the network.

### 3.4.4 Evaluation

This section will first describe the dataset we work with, and then training details and hyperparameter used. The performance of three strategies will be compared with each other first and then be compared with the state of art single and quasi-single QA methods in CASP13.

#### *Dataset Information*

Due to the fully connected layers used before generating the output of the network greatly increased the number of parameters and the limitation of the resource, the deep inception network still cannot be used on targets longer than 250. Also limited to the native structures released by CASP13, 10 targets from CASP13 with length less than 250 are tested and compared the performance with the state of art single and quasi-single QA methods. For each target, two datasets are used, the first one is CASP 13 QA stage1 and the second one is CASP13 QA stage2. QA stage 1 dataset consists of 20 decoys with different performance, while QA stage 2 dataset is the best 150 decoys submitted by different structure prediction servers. The absolute difference between predicted and true GDT-TS value is used to check the performance.

#### *Network Training Information*

As this QA method is target based, different targets are trained based on their own training dataset. But the training process is shared. For first two training strategies, the network will be trained for 20 epochs with learning rate 0.00005 using Adam optimizer.

As for the third training strategy, the network will be trained with fully extended training dataset for 5 epochs, and then continue be trained with the non-fully extended training dataset. Early stop with difference 0.001, patient 30 is also used to avoid over-fitting.

***Comparison of different training strategies***

When comparing the performance between three training strategies, it is clear that on both two datasets, the third training strategy achieved the best performance. From the result for each target, we can notice that even though for most of the targets, strategy two received worse result than strategy one, it still improved the performance for some hard targets. Also, training strategy two on fully extended training dataset saves lots of training time by avoiding the training crash caused by the zeros in the input. Figure 3.12 and figure 3.13 shows the detailed performance comparison on two different test datasets, label X lists CASP target name and label Y is the absolute difference between predicted QA score and real GDTTS score.

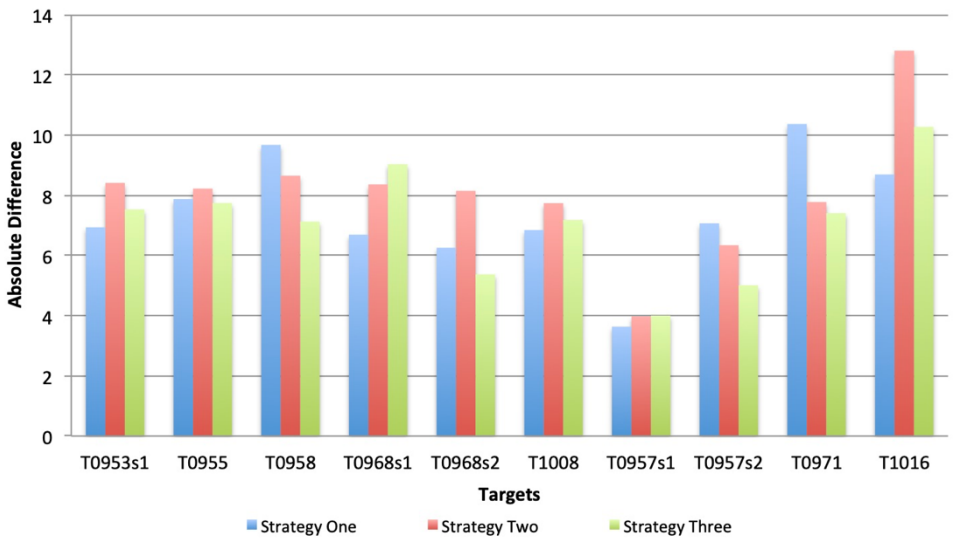


*Figure 3.12 Performance comparison of different strategies on QA stage 1 dataset.*

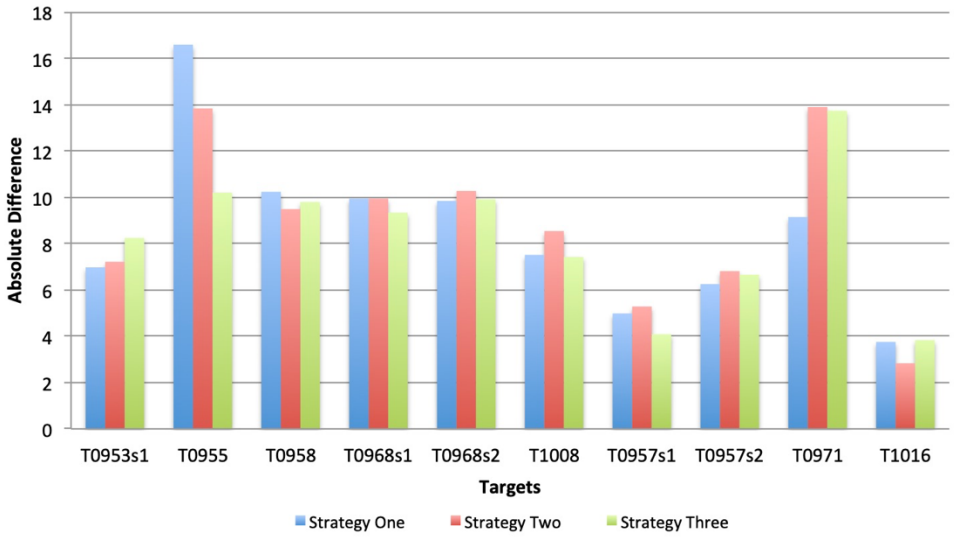


*Figure 3.13 Performance comparison of different strategies on QA stage 2 dataset.*

The third training strategy with two training stages retains the advantage of training strategy two, saving time by training on fully extended training dataset and improved the performance on some hard targets. Table 3.1 is the comparison of absolute difference between predicted QA score and real GDTTS score of three training strategies on two test datasets.

*Table 3.1 Difference between predicted and real GDT-TS for three training strategies.*

| | **Stage1** | **Stage2** |
|---|---|---|
| Training Strategy 1 | 7.403 | 8.521 |
| Training Strategy 2 | 8.042 | 8.877 |
| Training Strategy 3 | 7.027 | 8.323 |

***Comparison of different methods***

MUfoldQA_S2 and Modfold7 are the best quasi-single QA methods in CASP13. Proq3D and MUFold_server are two typical single model QA methods, also achieved good performance among different single QA methods in CASP13. By comparing the average difference between predicted QA score and real GDT-TS, it is easy to see that deep

inception network already achieved much better result than two single model QA methods. Even though on QA stage 1 the deep inception net is still worse than two state of art quasi-single model QA methods, it has shown better performance on QA stage 2 category. Table 3.2 shows the performance comparison between five methods on two different test datasets.

*Table 3.2 Difference between predicted and real GDT-TS*

| | **Stage1** | **Stage2** |
|---|---|---|
| MUfoldQA_S2 | 4.876 | 9.979 |
| Modfold7 | 5.7169 | 9.438 |
| Proq3D | 10.005 | 12.682 |
| MUFold_server | 10.927 | 12.567 |
| MUFOLD-INC | 7.027 | 8.323 |

### 3.4.5 Conclusion

This work proposes a new idea on how to apply deep learning methods to solve the QA problem and achieved some good result on a dataset with sequence length less than 250. It also provides a new idea for how to use the structural information of the reference model pool to assist in improving the performance of model quality evaluation. Instead of training a deep neural network to directly solve cross targets QA problem, this MUFOLD-INC proposed trains different models for each target, so the template pool can be used to build the training dataset. The results have shown that deep inception network can already perform on par with state of art single and quasi-single QA methods, and even outperformed other methods on some categories.

Computational bioinformatics problems usually got complex input. Inception net idea is more suitable to solve these problems by making the network wider and using multiple filters together to capture different information. The operation extending the templates to full size during the training dataset generation can improve the coverage of the whole

training dataset to avoid the training crash caused by padding too many zeros in the training input. But it also introduces noise information and makes the performance worse. A short pre-train on fully extended dataset before training on normal dataset perfectly combines the advantages of two basic strategies and achieved the best performance.

Target based deep learning methods can be a new option to apply the deep learning knowledge on QA problem. But still has its own aspects to be improved. The training dataset and the network can still be compressed in order to make the method widely used. For now, the complexity of the network makes it hard to work on long targets with length larger than 250. The size of training dataset limited the usage of other QA methods' result as part of input as generate the QA result for thousands of models still spends lots of time. The current way to select templates to be used and the methods used to generate the training dataset can still be refined to improve the performance and reduce the size of the training dataset. Generating a smaller decoy set with higher performance can not only save time, but also give more chance to add more features of the decoys set.

## 3.5 MMQA

### 3.5.1 Motivations

With the continuous update of machine learning algorithms and the wide application of deep learning algorithms in the QA field, more teams try to use more complex machine learning algorithms and more comprehensive input information to improve the performance of QA algorithms. As more complex algorithms are applied in the QA field and have achieved reasonable results, new bottlenecks have also appeared. From the results of CASP14, we found that more complex algorithms have achieved better performance on

some different QA targets. The performance of many hard QA targets has been improved, which proves that these algorithms have indeed obtained new reasonable information and can improve QA performance. But one problem is that more comprehensive input information and more parameters do not improve the overall performance. The new method including ensemble methods, deep neural network and even graph neural network, have achieved better performance on some targets, but on average many of them are not as good as some basic tools like Proq2. This shows that simply increasing the complexity of the QA algorithm is not the only direction to improve QA performance, the strategy of balancing the training and testing process and solving the over-fitting problem caused by the use of massive parameter models is also an important direction.

In this work, based on different machine learning based single-model algorithms, we developed three new heuristic single-model QA methods applying different strategies in order to improve the QA performance. MMQA-1 and MMQA-2 applied two stage machine learning strategy with different training datasets without intersection for each stage. In addition to the data used for training, we also split the whole feature set into different sets base on the information contained in different features and used as input features in each stage. After MMQA-1 got excellent performance in CASP14, we collected more new features and applied same strategy and developed MMQA-2, which is the improved version of MMQA-1. Based on these two methods, we further applied hierarchical ensemble strategy based on MMQA-2 and implemented MMQA-HE which improved the performance of MMQA-2 across different QA performance metrics.

### 3.5.2 Evaluation Metrics

Different QA methods generated different style of QA score, with the development of the field of protein structure prediction, the goal of the QA algorithms has also changed. From the judging criteria of QA category used in recent CASP competition, we can know that the current protein model QA field is most concerned about two points: 1) Prediction, given a predicted protein structure, evaluating how good the structure is. 2) Selection, given a pool of predicted models, best model selection is another most import goal for different QA methods. The mature "differences (predicted vs observed)" used in CASP is calculating the average of difference between predicted QA scores and corresponding GDT-TS values between the predicted models and the native structure of the target protein, which reflects the prediction ability of a QA method. It can be defined as:

$$Diff_{average} = \frac{1}{N}\sum_{i=1}^{N} ||G - G'||$$

Where *N* is the number of decoys in the pool, *G* is the real GDT-TS score between decoy and native structure and *G'* is the predicted GDT-TS score. The mature "difference from the best" used in CASP calculate the difference of GDTTS between the best model selected by the QA method and the real best model in a predict-ed model pool, which shows the selection ability of a QA method. It can be defined as:

$$Diff_{from\ best} = GDT - TS_{best} - GDT - TS_{selected}$$

Where $GDT\text{-}TS_{best}$ is the GDT-TS score of the best model in the pool, and $GDT\text{-}TS_{selected}$ is the GDT-TS score of the model selected by the QA method.

### 3.5.3 Feature extraction

For each protein model, we capture two different kinds of features: structural features, energy/ potential score features.

For each protein model structure, we generated the following structural features: for each protein sequence, we first capture: SS3 prediction [3-state secondary structure predicted by PSIPRED 4.02.] and ACC prediction [solvent accessibility predicted by SCRATCH-1D 1.2], and for each protein model structure, we derive secondary structure (SS3) and relative solvent accessibility (RSA) calculated by DSSP. From the predicted SS3 and ACC of the sequence and the related information generated by DSSP, we totally generate 10 structural features. We also collect 5 angle-based scores generated by hoppscore 2.02, totally 15 structural features are used, we also added the length of the model and the confidence score of the secondary structure prediction as two input features.

For each protein model structure, we generated the following energy and potential score features: ddfire score and Dfire score by dDFIRE1.1, RW and RWplus score, DOPE score by MODELLER, three Opus_psp scores [total energy, orientation-dependent energy, LJ repulsive energy], Fisher score generated with potential file RF_CB_SRS_OD, sbrod score and three score generated by GOAP. These 13 scores are used as input features in stage 1 training.

We also generate Proq2 and Proq3 score using released Proq3 software, and two energy scores generated using Rosetta 2016.15 bundle. Table 3.3 shows the information of all 34 features we used in this work.

Among the 34 features mentioned above, 29 features are used in MMQA-1, Fisher score, sbrod score and GOAP score are added to the feature set when we try to improve the method and develop MMQA-2 and MMQA-HE.

*Table 3.3 Feature list used for MMQA*

| ***Feature ID*** | ***Feature Name*** | ***Stage used*** | ***Tool*** |
|---|---|---|---|
| 1 | Length of target model | 1 | --- |
| 2 | Sum of confidence score | 1 | PSIPRED |
| 3 | Matching of 3-state secondary structure | 1 | PSIPRED, DSSP |
| 4 | Matching of helix | 1 | |
| 5 | Matching of sheet | 1 | |
| 6 | Matching of coil | 1 | |
| 7 | Matching of solvent accessibility (threshold = 0.2) | 1 | SCRATCH-1D, DSSP |
| 8 | Matching of Bury Amino Acid (threshold = 0.2) | 1 | |
| 9 | Matching of expose Amino Acid (threshold = 0.2) | 1 | |
| 10 | Matching of Solvent Accessibility (threshold = 0.25) | 1 | |
| 11 | Matching of Bury Amino Acid (threshold = 0.25) | 1 | |
| 12 | Matching of expose Amino Acid (threshold = 0.25) | 1 | |
| 13~17 | HOPPscore (Pair = 1,2,3,4,5) | 1 | HOPPscore v2.02 |
| 18 | ddfire | 1 | dDFIRE1.1 |
| 19 | Dfire | 1 | |
| 20 | RW | 1 | calRW |
| 21 | RWplus | 1 | calRWplus |
| 22 | DOPE | 1 | MODELLER |
| 23 | Opus_psp (total energy) | 1 | OPUS-PSP v1.0 |
| 24 | Opus_psp (orientation-dependent energy) | 1 | |
| 25 | Opus_psp (LJ repulsive energy) | 1 | |
| 26 | Fisher score (RF_CB_SRS_OD) | 1 | Fisher |
| 27 | sbrod score | 1 | Sbord |
| 28 | goap score | 1 | GOAP |
| 29 | goap_dfire score | 1 | |
| 30 | goap_ag score | 1 | |
| 31 | Proq2 | 2 | Proq3 |
| 32 | Proq3 | 2 | Proq3 |
| 33 | ProQRosCen | 2 | Proq3, Rosetta bundle |
| 34 | ProQRosFA | 2 | |

### 3.5.4 Methods

***Two stage machine learning***

Among all the features we prepared, we split them into two different feature sets. If a feature is directly generated from the information of the sequence and the structure of the target model, it will be added to the first feature set, and used as the input feature set of stage 1 training. Otherwise, if the generation of a feature not only used information from of the sequence and the structure of the target model, but also used information of other features in the feature list, it will be added to the second feature set and used for stage 2 training. For totally 34 features listed in Table 1, feature 1 to 30 form the first feature set and feature 31 to 34 belong to the second feature set.

Different with most of the machine learning and deep learning based methods take all features in and directly generate QA score, our new methods designed a two-stage machine learning based method without interaction in both training dataset and input feature set. The first stage is using half of the whole training dataset and the first feature set contains 25 features (30 features for MMQA-2 and MMQA-HE as we add Fisher score, sbrod score and three scores generated by GOAP into the stage 1 feature set) to train a random forest model. This random forest model will be used to generate a QA score, which will become the input feature of stage 2 together with other features in the second feature set to train a SVM model. Figure 3.14 shows the pseudocode of MMQA-1 and MMQA-2.

***Algorithm MMQA-1***
***Input***: $S_1$, $S_2$, $S_t$
***Output***: Predicted GDTTS score $P$
$R, S \leftarrow$ TRAIN ($S_1$, $S_2$)
$F_1 \leftarrow$ Stage1 Feature Extraction ($S_t$)
$f_r \leftarrow$ Run Random Forest Model R with ($F_1$)
$F_2 \leftarrow$ Stage2 Feature Extraction ($S_t$)
$F_2 \leftarrow F_2 \cup \{f_r\}$
$P \leftarrow$ Run SVM model S with ($F_2$)
***End***

***Function*** TRAIN ($S_1$, $S_2$)
    $R \leftarrow$ STAGE1_TRAIN ($S_1$)
    $F_1 \leftarrow$ Stage1 Feature Extraction ($S_2$)
    $f_r \leftarrow$ Run Random Forest Model R with ($F_1$)
    $S \leftarrow$ STAGE2_TRAIN ($S_2$, $f_r$)
    ***return*** $R, S$
***end function***

***Function*** STAGE1_TRAIN ($S_1$)
    $F_1 \leftarrow$ Stage1 Feature Extraction ($S_1$)
    $R \leftarrow$ Random Forest Train ($F_1$)
    ***return*** $R$
***end function***

***Function*** STAGE2_TRAIN ($S_2$, $f_r$)
    $F_2 \leftarrow$ Stage2 Feature Extraction ($S_2$)
    $F_2 \leftarrow F_2 \cup \{f_r\}$
    $S \leftarrow$ SVM Learn with ($S_2$, $F_2$)
    ***return*** $S$
***end function***

*Figure 3.14 pseudocode of MMQA-1 and MMQA-2 S1 = Training set for stage1, S2 = Training set for stage2, St = Test set, R=Random Forest Model Trained in stage1, F1= feature set used as stage1 input, S=SVM Model Trained in stage2, F2= feature set used as stage2 input.*

When doing training for different stages, we split the training dataset into two datasets without intersection. The comparation between using all training da-ta to train two stages

and this strategy shows that use different training data for stages helps avoid overfit-ting and can further improve the test performance.

***Hierarchical Ensemble***

The random forest algorithm combines the idea of "bagging" and the random selection of features. It is an ensemble learning method for multiple tasks. In MMQA-HE, we further applied hierarchical ensemble idea and try ensemble not only on tree level but also on forest level. The hierarchical ensemble algorithm also applied bagging idea. We set the total forest number as *n*, and for first *n-1* random forest model we will random sampling 80% of the training dataset with replacement. For the last random forest model, the whole training dataset for stage 1 will be used, Figure 3.15 shows pseudocode of stage 1 in MMQA-HE.

***Algorithm MMQA-HE Stage1***
***Function*** MMQA-HE_S1_TRAIN (***S***)
    R ← ∅
    ***for*** I ∈ 1, …, n-1 ***do***
        $S^{(i)}$ ← A bootstrap sample from ***S***
        $F_1^{(i)}$ ← Stage1 Feature Extraction ($S^{(i)}$)
        $r_i$ ← Random Forest Train ($F_1^{(i)}$)
        $R \leftarrow R \cup \{r_i\}$
    ***end for***
    ***return R***
***end function***

*Figure 3.15 pseudocode of stage 1 in MMQA-HE S= Training set for MMQA-HE stage1, R=Random Forest Model set, n= number of random forests will be trained, F1= feature set used as stage1 input.*

### 3.5.5 Evaluation

This section will first describe the dataset we work with, and then training details and hyperparameter used. The performance of three QA methods on CASP12, 13 and 14 will also be showed and compared with other state-of-art single model QA methods.

***Dataset Information***

When choose the training data of our MMQA-1, considering the performance distribution of server sub-mission before CASP8 may be different with the recent CASP competition, we chose all server prediction from CASP9 to CASP11 as our training dataset, and all server predictions from CASP12 as validation dataset to do finetuning of our parameters. When do testing on CASP12 dataset, we retrain our model with CASP9, CASP10, and half of CASP11 dataset, and used another half CASP11 dataset as validation dataset. The whole training dataset we used for the model we use to test CASP13 and CASP14 dataset includes 76136 decoys from 304 targets. We randomly selected 60% of the training data, 45212 decoys from 183 targets are used to train the random forest model in stage 1, and 40% if the training data, 30924 decoys from 121 targets are used to train the SVM model in stage 2. And the validation dataset we used includes 6893 decoys from 40 targets in CASP12. For the model we trained to test CASP12 test dataset, 41012 decoys from 158 targets are used to train random forest model in stage 1, and 27412 decoys from 105 targets are used to train the SVM model in stage 2. And the validation dataset includes 7712 decoys from 41 CASP11 targets.

***Training Information***

Both random forest model for stage 1 and SVM model for stage 2 are trained using sklearn python library. For MMQA-1 we used in CASP14 competition, the stage 1 input

feature size is 24, and the stage 2 input feature size is 5, the features we used as input of stage 2 is the output from stage 1 mode, Proq2, Proq3 and two Rosetta energy score. For MMQA-2, we collected five more features from three tools (Fisher score generated with potential file RF_CB_SRS_OD, sbrod score and three score generated by GOAP) and add the length of the decoy into the input feature set, the final stage 1 input feature size of is 30, and the inputs for stage 2 are not changed.

For random forest model, there are several statistical parameters can be fine-tuned to improve the performance. We mainly focus on the number of the trees in the forest mtree, the number of features random chosen tchosen, and the max depth of the tree in forest dmax. In this study, we optimized the parameters in the following sets: mtree $\in$ {50, 100, 300, 500, 1000,3000}, tchosen $\in$ {1, 5, 10,$\cdots$ ,30, auto} and dmax $\in$ {5, 10, 15, None}. Finally random forest model with 500 trees, auto feature number and max depth 10 shows best performance.

For MMQA-HE, after testing with n $\in$ {2, 3, 5, 7, 9}, we found for this problem hierarchical ensemble can further improve the performance of the algorithm base on the fine-tuned random forest model, and *n=3* achieve the best performance across different QA performance metrics.

### 3.5.6 Results

This work has been tested using different CASP datasets. During the development, we tested the methods on CASP13 dataset. Then we picked most stable version MMQA-1 and participated CASP14 for blind test. After CASP14, we re-developed our work with more features and ensemble idea, two updated version MMQA-2 and MMQA-HE shows improved performance on CASP13 and CASP14 dataset. In order to test the robustness our

methods, we also modified the training and validation dataset and tested it on CASP12 dataset.

Since CASP10, the quality assessment (QA) task has been divided into two different stages. Stage 1 prediction pool consists of the 20 prediction models with a large performance range. Stage 2 prediction pool consists of the 150 selected good prediction models. Both stage 1 and stage 2 dataset will be used to estimate the performance of our experimental result. Two matures used in CASP QA competition will be used to estimate the performance of different methods in this section. “Differences (predicted vs observed)” used in CASP is calculating the average of difference (AD) between predicted QA scores and corresponding GDT-TS values between the predicted models and the native structure of the target protein, which reflects the prediction ability of a QA method. “Difference from the best” (DB) used in CASP calculate the difference of GDTTS between the best model selected by the QA method and the real best model in a predicted model pool, which shows the selection ability of a QA method.

***CASP12 Results***

During CASP 12, a total of 85 QA targets were re-leased, of which 13 targets were cancelled, and finally native structure of 40 targets were released. For these 40 targets, there are 1, 3, 1, 1 target result pages not existing for “Stage 1: Differences (predicted vs observed)”, “Stage 1: Difference from the best”, for “Stage 2: Differences (predicted vs observed)” and “Stage 2: Difference from the best”, respectively.

Table 3.4 shows that MMQA-1 achieves smallest stage 1 DB compare with other top teams, and MMQA-2 got improvement in all four different QA performance metrics we used and already shows best performance in Stage 1 DB, Stage 2 AD and Stage 2 DB.

Based on MMQA-2, MMQA-HE using hierarchical ensemble further improved the performance in various performance metrics and got top performance on all matures.

*Table 3.4 Differences (predicted vs observed) and Difference from the best in CASP12 QA Stage1 and Stage 2*

| Method Name | S1 AD | S1 DB | S2 AD | S2 DB |
|---|---|---|---|---|
| Proq2 | 9.48 | 5.24 | 11.09 | 6.85 |
| Proq3 | 12.44 | 9.41 | 13.06 | 7.98 |
| Wang4 | 7.65 | 7.36 | 12.58 | 13.61 |
| MESHI_SERVER | 11.51 | 7.48 | 13.83 | 6.22 |
| MMQA-1 | 11.33 | 4.71 | 10.89 | 7.27 |
| MMQA-2 | 10.12 | 1.58 | 10.42 | 5.41 |
| MMQA-HE | 7.22 | 1.24 | 8.93 | 3.23 |

***CASP13 Results***

Testing on the CASP13 dataset is challenging because after the competition only the real structure of 20 targets were released by official. And there are 6 target result pages not existing for “Stage 1: Difference from the best” and 4 target result pages not existing for “Stage 2: Difference from the best”. Also, as a representative group using deep learning algorithms to solve QA problem, 3DCNN only submitted prediction results for 6 targets out of these 20 targets. From Table 3.5 shows that MMQA-1 only got comparable result on four categories compare with other top groups, but MMQA-2 again improvement in all four different QA performance metrics and already ranked No.2 on stage1 DB and stage2 AD, then MMQA-HE using hierarchical ensemble got even better result and got smallest stage1 AD and ranked No.2 on both stage1 DB and stage2 AD. After analysis the result we noticed that Proq3, Proq3D, and our MMQA-2, MMQA-HE all picked up the best model from 13/14 targets we considered about stage1 DB and only one targets missed.

*Table 3.5 Differences (predicted vs observed) and Difference from the best in CASP13 QA*

*Stage1 and Stage 2*

| Method Name | S1 AD | S1 DB | S2 AD | S2 DB |
|---|---|---|---|---|
| Proq3 | 10.80 | 2.10 | 10.75 | 8.28 |
| Proq4 | 14.23 | 5.70 | 14.52 | 9.91 |
| ProQ3D | 9.57 | 2.10 | 9.71 | 7.09 |
| 3DCNN | 23.30 | 6.27 | 16.21 | 4.57 |
| Bhattacharya-S | 12.50 | 6.43 | 17.48 | 10.64 |
| MMQA-1 | 12.07 | 6.09 | 12.32 | 13.88 |
| MMQA-2 | 11.84 | 3.94 | 10.59 | 13.12 |
| MMQA-HE | 8.63 | 3.94 | 10.11 | 10.91 |

***CASP14 Results***

The most recent CASP14 competition is an ultimate blind test, our MMQA-1 participated in CASP 14 as group MUFold_server and ranked No.2 in stage2 AD. For CASP14 dataset, totally 83 targets are re-leased and finally the native structures of 70 targets are released. When capture the official result for CASP website, we noticed there is totally no information released for stage1 DB information. In order to compare with other state-of-art methods in this category, we downloaded the prediction file submit-ted by these groups and calculated the number of stage1 DB ourselves with the submission and GDTTS score calculated with LGA and the released native structures.

Table 3.6 shows the performance comparation of our three methods and top ranked single-model QA methods in CASP14. Our MMQA-1 shows comparable performance on all categories and ranked No.3 in stage2 AD among all methods on these 70 targets. Its improved version MMQA-2 again got improvement in all performance matures and with the improvement MMQA-2 already ranked No.1 in stage2 AD, and got top performance in stage1 AD and stage1 DB. When tested on CASP14 dataset, MMQA-HE didn't improve on "prediction" aspect but show better performance on "selection" part. It significantly im-proved performance on stage1 DB and makes MMQA-HE ranked No.2 in stage1 DB category

*Table 3.6 Differences (predicted vs observed) and Difference from the best in CASP14 QA Stage1 and Stage 2*

| Method Name | S1 AD | S1 DB | S2 AD | S2 DB |
|---|---|---|---|---|
| Proq3D | 12.91 | 6.42 | 13.20 | 10.79 |
| Proq4 | 12.57 | 10.43 | 15.46 | 15.06 |
| AngleQA | 8.39 | 4.48 | 13.69 | 14.01 |
| SASHAN | 9.96 | 10.19 | 13.66 | 14.74 |
| 3DCNN_prof | 10.88 | 4.67 | 13.95 | 10.66 |
| GraphQA | 7.70 | 7.66 | 13.61 | 9.93 |
| RaptorX-QA | 7.32 | 7.26 | 12.26 | 15.66 |
| ROSETTASERVER | 11.70 | 2.23 | 12.17 | 8.53 |
| MMQA-1 | 10.32 | 7.31 | 12.48 | 11.97 |
| MMQA-2 | 7.98 | 5.83 | 12.05 | 9.83 |
| MMQA-HE | 8.66 | 4.27 | 13.04 | 9.80 |

### 3.5.7 Conclusion

In this work, we propose three new machine learning-based algorithms MMQA-1, MMQA-2, and MMQA-HE, and two new strategies for applying complex machine learning algorithms to solve QA problems. Different from using a more comprehensive and complete feature set directly as the input information of the complex mechanical learning algorithm and directly making an evaluation. MMQA-1 and MMQA-2 split the feature set into two parts based on the information level of the feature and use different features in different stages. And use datasets without intersection to train the machine learning models of different stages. Result from CASP12, 13 and blind test result from CASP14 shows this training strategy achieve perfect state of art performance. Based on the Ensemble algorithm used in MMQA-1 and MMQA-2, we further proposed the MMQA-HE with the Hierarchical ensemble strategy. On the CASP12 and 13 test datasets, MMQA-HE has an over-all improvement in different QA performance metrics compared to

MMQA-2. And on the more complex CASP14 test dataset, MMQA-HE also got improvement on most performance metrics compared to MMQA-2.

## 3.6 Simulated High Performance Decoy Pool

### 3.6.1 Motivations

Usually, we divide QA algorithms into two categories: Single model QA and Multiple model QA, although the performance of single model QA is not as good as multiple model QA in most cases, many groups still focus on improve the performance of single model QA methods. One of the reasons is that the two types of methods have their own strengths. Multiple model QA performs better in accuracy (difference between predicted and real GDT-TS is smaller), and Single model QA methods perform better in top model selection. Another reason is that compared to multiple model QA methods that are susceptible to reference models, single model QA is more reliable in terms of stability. A more obvious example is that with the efforts of many groups, many multiple model QA methods in CASP13 beat the naive consensus. Before the arrival of CASP14, many groups further improved performance, but it was found that naive consensus has once again become the best performing multiple model QA method. But look at the single model QA methods. Although Proq2 is already a very old QA method, its performance has been very stable since many CASP competitions, and it has always performed well on the best model selection problem.

With the development of protein structure prediction, the distribution of server prediction pool in each CASP competition has changed greatly. In CASP13, many excellent multiple model QA methods beat the naive consensus, which has been squeezed

out of the top five. But in CASP14, the naive consensus is back at number one again. This is proves that the performance changes in the model pool greatly affect the performance of QA algorithms, and it also reflects the lack of stability of the current QA methods.

It is not difficult to see from the abstracts of the major groups in CASP13 that the predictions of many groups are generated based on the Rosetta toolbox released by the Rosetta Group. Therefore, it is foreseeable that with the release of code and model of AlphaFold2 and RosettaFold, more groups will design their protein structure prediction pipeline on this basis. And the average performance of the server prediction pool will be greatly improved, the distribution performance distribution will also change significantly. QA problem will also become an evaluation problem for high performance, high similarity decoy pool.

### 3.6.2 Simulated High Performance Decoy Pool

In order to face this challenge, we simulated a high performance decoy pool based on AlphaFold and trRosetta, the target we used is the 70 targets in CASP14 with native structure released.

In order to generate this dataset, we run AlphaFold2 with three different settings, trRosetta, and 3DRobot. For each target sequence, we will first run trRosetta to generate 5 predictions, then we run AlphaFold2 with three different settings:

- AF -- the default, complete AlphaFold2 prediction process,
- AF_fast -- no ensembling, and
- AF_mini -- reduced version of database and no ensembling.

we also collected the official submission of AlphaFold2 in CASP14, this process will generate 25 decoys for each target sequence. We noticed that when run AlphaFold2

multiple times, the prediction is different, so we run complete AlphaFold2 again and generated another five decoys.

For each of 70 targets, take the 10 decoys generated by running AlphaFold2 and the five official submissions in CASP14 from AlphaFold2 as start structure, we run 3DRobot on these 15 decoys with the following settings:

-hour = 6 (max time used to run MC simulation),

-nd = 10 (number of output decoys is 10),

-cut = 5 (RMSD cutoff for output decoys is 5, make sure the generated decoy is not much different from the start structure).

3D Robot will totally generate 150 models for each target. For the 4 targets AlphaFold doesn't have submission, we run AlphaFold third time to make sure each target will have 15 start decoy for 3DRobot.

For the 70 targets, 3DRobot totally run 1050 jobs, generated 10469 decoys (31 models' generation failed), as we may stop the MC simulation of 3Drobot in the halfway, so some generated decoys have format problem, totally 159 decoys are reformatted. Among the 1050 jobs, 192 jobs generated new models with better performance than the start structure pool, 36 jobs generated the best model with the same performance as the template, and the performance of the new model generated by the remaining 822 jobs has decreased compared with the template. The following figure 3.16 shows the performance balance for model generation.

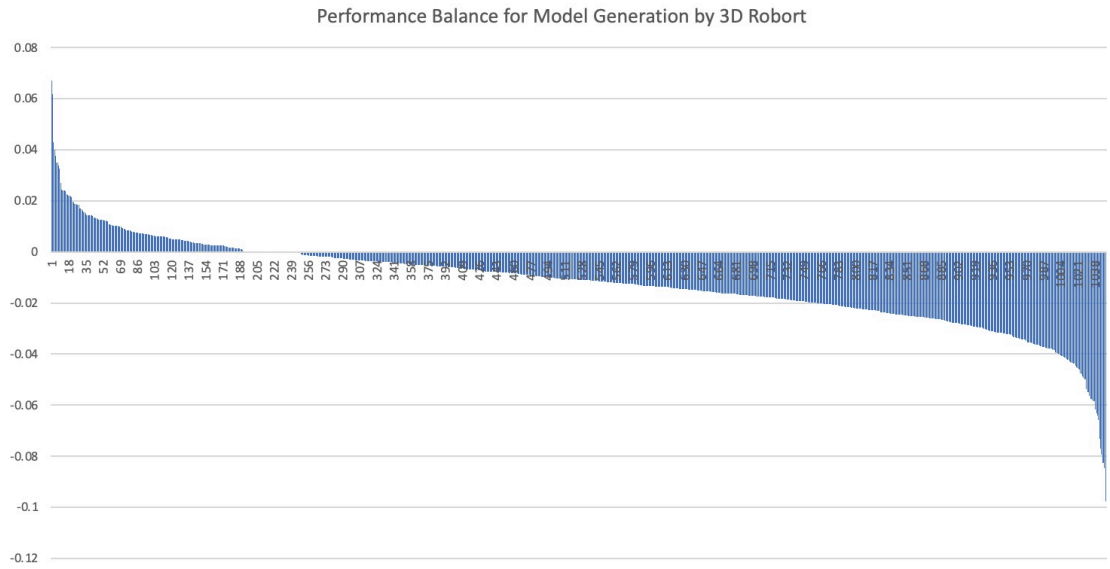


*Figure 3.16 Performance balance for model generation*

We think this is a reasonable performance distribution, because get improvement base on AlphaFold's prediction is not easy, and some groups may choose to run light version of AlphaFold with smaller database due to insufficient resources, and the performance of prediction will also be affected accordingly. At the same time, there will be some excellent groups that have successfully made improvements on the basis of AlphaFold's prediction.

For each target, with the 180 models as initial pool, we first directly took out the 10 decoys generated by running complete AlphaFold2 twice and the five decoy submitted by AlphaFold2's group in the CASP14 competition, then for the remaining 150 models, we used naive consensus selects 135 models and forms a "set150" decoy pool together with the 15 models previously taken out. For totally 70 targets, the mean GDT-TS score of the best model in the pool achieve 0.8403, while the mean value of the average GDT-TS score of the decoy pools achieves 0.7503, which means that the new generated pool does have a high performance.

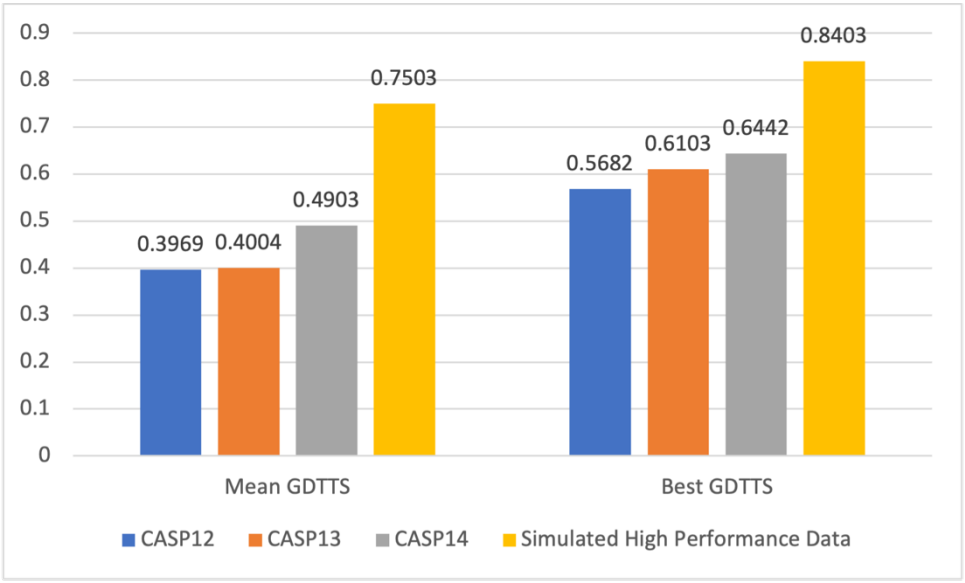


*Figure 3.17 GDT-TS analysis for the new generated decoy pool with Set150 dataset of CASP12~14.*

The Above figure shows the performance of Set150 decoy pool of CASP12~14 and our new simulated decoy set. Form the figure we can see, although the difficulty of CASP targets is increasing year by year, with the maturity of protein structure prediction technology, the performance of Set150 dataset is still rising steadily. With the release and open source of AlphaFold2, what the protein quality assessment will face is obviously the performance-level model pool of the new simulated dataset.

### 3.6.3 Analysis QA Performance on New Dataset

In CASP15, estimates of the accuracy of single protein models have been removed as CASP believes that these QA methods cannot compete with specific estimates from modeling methods. Although estimates of the accuracy of individual protein models are no longer in the CASP competition, we still did some analysis of single protein model QA with this new simulated decoy pool.

***Single Model QA vs. Naïve Consensus***

In CASP14, for set150, MUFOLD single model QA ranks No.2 among all single model QA methods in the absolute difference category, while naive consensus ranks No.1 among all QA methods. Therefore, we try to analyze what changes the new performance distribution brings to different QA methods by observing the performance of MUFOLD single model QA and Naive consensus on the newly simulated decoy pool. Figure 3.18 shows the relationship between GDTTS score and consensus score (left) and MUFOLD QA score (right),

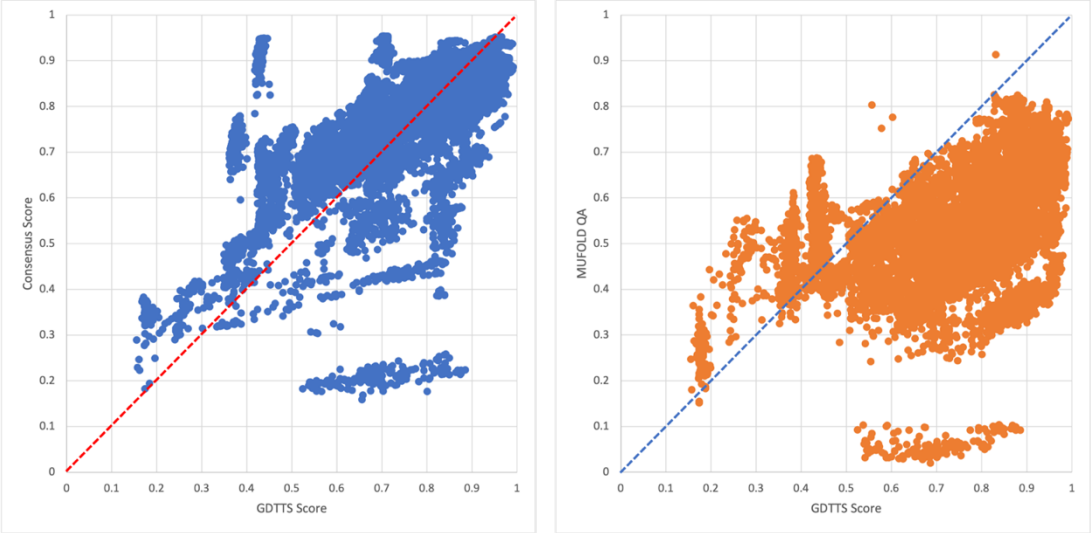


*Figure 3.18 Relationship between GDTTS score and consensus score (left) and MUFOLD QA score (right).*

From the above figure, we can see that the QA score given by MUFOLD single model QA has obvious underestimate problem. For most models, the predicted GDTTS score given by MUFOLD single model QA is much lower than the real GDTTS score. The performance of naive consensus score is relatively better, but there is also a slight overestimate problem, and because the model structure in the pool is still different, the

upper limit of the naive consensus score is limited. For models with a GDTTS score of 0.95 or more , the QA score given by naive consensus is still lower than ground truth.

The average correlation of MUFOLD single QA on totally 70 targets is 0.5860, while the naive consensus is 0.7033. It can be seen from the figure 3.19 below that for totally 70 targets, the performance consistency of MUFOLD single QA and naive consensus is not very strong. Both MUFOLD single QA and naive consensus have targets that each perform much better than the other.

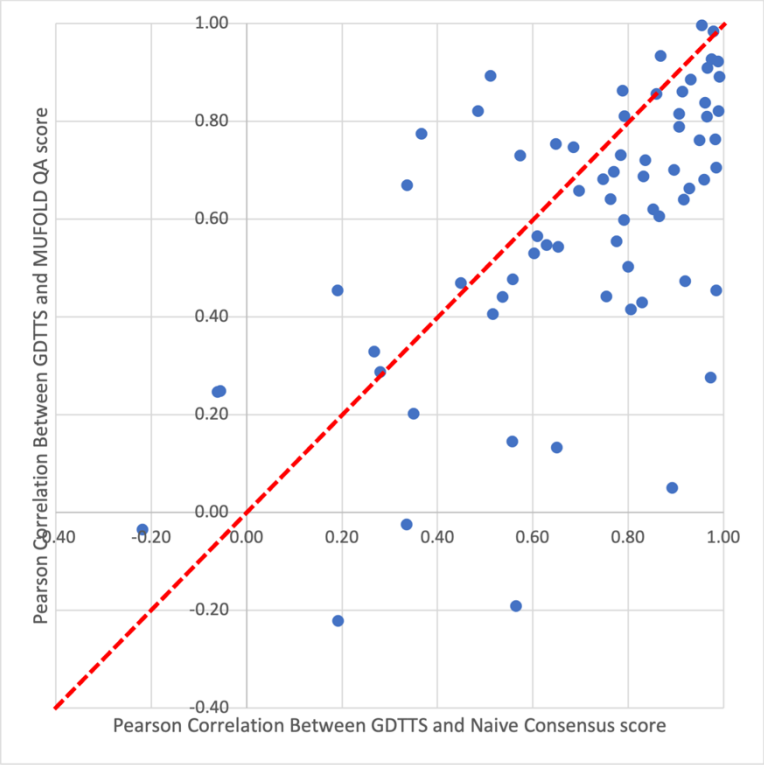


*Figure 3.19 Comparison between MUFOLD single model QA and Naïve Consensus on Pearson Correlation with GDTTS score.*

***Analysis on Features***

The most commonly used features of QA methods can be divided into two categories, structural features and score functions. Usually, we measure the information contribution

of a feature to model QA by calculating the Pearson correlation between a feature and true GDTTS.

For some common features of QA methods, we calculated their Pearson correlation performance on CASP12~14 datasets and new simulated dataset. We found that with the advancement of protein structure prediction technology and the difference in model pool performance distribution, the information contribution of many features has changed greatly. Figure 3.20 shows four commonly used structural features with large variation in Pearson Correlation on different datasets, including secondary structure match between predict secondary structure and models, coils match between predict coils and models, solvent accessibility match of exposed and buried amino acids. Although the difficulty of CASP targets and different targets themselves will have a certain impact on the Pearson Correlation of the structural feature, but we can still see that with the advancement of protein structure prediction technology, structural errors in predicted models are gradually decreasing, the structural accuracy of many models themselves even exceeds the accuracy of secondary structure and solve accessibility predictions predicted by common used secondary structure and solve accessibility prediction tools, which means that when do QA on high performance model pool generated by tools like AlphaFold2, the effective information provided by structural feature will also become less.

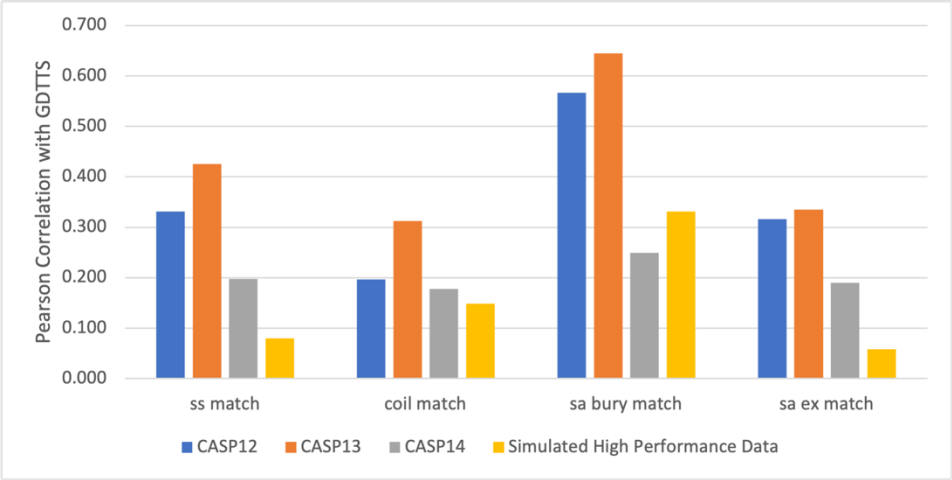


*Figure 3.20 Pearson Correlation between commonly used structural features and GDTTS score on CASP12~14 dataset and new simulated dataset.*

Different from the structural feature, as the structure of the predicted model becomes more accurate and reasonable, the Pearson correlation of many score functions and GDTTS is significantly improved. The following figure 3.21 shows the Pearson Correlation of some score functions and GDTTS score.

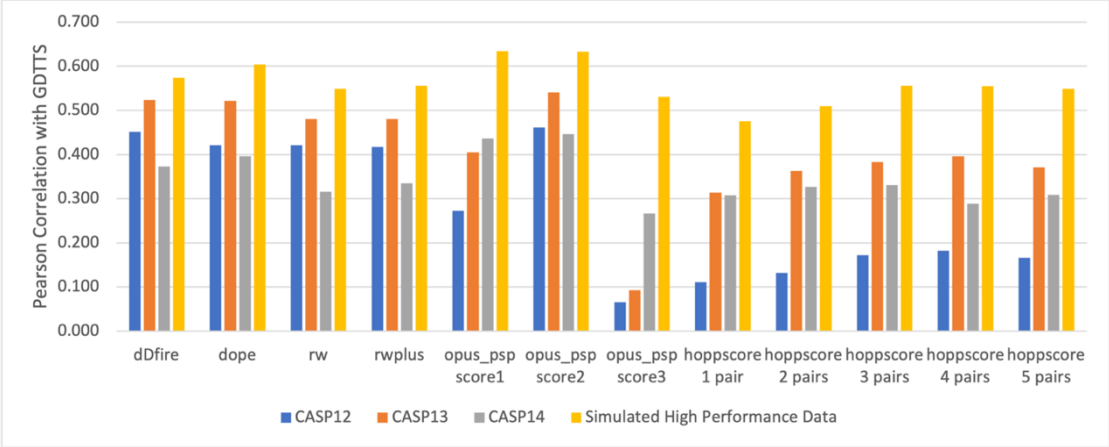


*Figure 3.21 Pearson Correlation between commonly used score functions and GDTTS score on CASP12~14 dataset and new simulated dataset.*

We can find that for some commonly used score functions like dDfire, Dope, and RW, although the Pearson Correlation fluctuates on different CASP datasets, their performance on the simulated high performance dataset is the best, and there is

significant improvement compared to the performance on CASP14 stage2 dataset. Some other features that provide little effective information for model QA earlier, such as "total energy", "orientation-dependent energy", and "LJ repulsive energy" generated by OPUS_PSP, and the five scores generated by HOPPscore based on phi-psi pairs, the performance on the simulated high performance dataset is almost qualitatively improved, which also shows that these scores can provide more effective information on the QA of the high performance model pool.

### 3.6.4 Conclusion

Based on the above analysis, we can know that when we do QA on the high performance model pool generated by good predictors like AlphaFold2, the performance of the multiple model QA methods is still better than the single model QA methods. The problem faced by single model QA methods is obvious underestimate. With the change of model pool performance distribution, the effective information provided by many originally important structural features dropped significantly, while many score functions shows better performance, some features with low correlation earlier may provide much more information now. It is clear that the models trained on the old dataset are no longer applicable.

## 3.7 QA for protein complexes in CASP15

### 3.7.1 Motivations

CASP14 (2020) made a huge leap in the accuracy of individual protein and domain models, making many models competitive with experiments. This advancement is largely the result of the successful application of deep learning methods, especially AlphaFold and

RosettaFold. With the open source of these excellent tools, the calculated protein structures are being used in an increasingly wide range of applications. In response to this new situation, CASP revised a set of modeling categories. In the field of accuracy estimation, estimating the accuracy of single protein models has been removed as CASP believes that these methods cannot compete with modeling method specific estimates. And accuracy estimation for protein complexes has been added to the competition as a new category. In the accuracy estimation of protein complexes, the global score that CASP focuses on contains two accuracy scores for a model. The accuracy scores are real numbers in range [0.0, 1.0], predicting overall folding accuracy and overall interface accuracy.

In order to adapt to the changes of CASP15 in the QA field, we also try to develop corresponding QA methods for model QA and selection. Corresponding to the two aspects of CASP QA focus on, we developed two methods corresponding to predicted overall folding accuracy and overall interface accuracy, respectively. The QA method based on US-align is used to predict the overall folding accuracy, while the QA method based on DockQ is used to estimate the overall interface accuracy.

### 3.7.2 Methods

Both methods we have developed are based on the idea of consensus. The main difference between the two methods is that the tools used for calculating similarity are US-align and DockQ respectively.

#### *Folding QA based on US-align Consensus*

US-align (Universal Structural alignment) is a unified protocol to compare 3D structures of different macromolecules (proteins, RNAs and DNAs) in different forms (monomers, oligomers and heterocomplexes) for both pairwise and multiple structure alignments. The

core algorithm of US-align is extended from TM-align and generates optimal structural alignments by maximizing TM-score of compared structures through heuristic dynamic programming iterations.

Same as TM-score, The naive consensus algorithm can be directly applied on US-align to calculate a pair-wise similarity score for protein complexes. The TM-score from US-align will be used to measures the similarity of three-dimensional structures of the Cα atoms. TM-score has values in (0,1] with 1 indicating an identical structure match, where a TM-score $\geq 0.5$ means the structures share the same global topology for proteins. We calculate pairwise TM-score scores between all candidate models and compute the average similarity score for each model as the final folding QA score. When calculating the similarity of two models *i* and *j*, considering that the TM-scores we get when we use model *i* and model *j* as the reference structure may be different, we finally use the average of US-align(*i, j*) and US-align(*j, i*) as the similarity score of model *i* and *j*. The five model with the largest average similarity score is selected as top five models.

***Docking QA based on DockQ Consensus***

State-of-the-art techniques for assessing the structural quality of docking models are currently based on three related but independent quality measures: $F_{nat}$, LRMS, and iRMS proposed and normalized by CAPRI. CAPRI's criteria for assessing the quality of docking models are defined by applying various interim cutoffs to these metrics to classify docking models into four categories: Incorrect, Acceptable, Medium, or High quality. This classification is useful in CAPRI, but if all models are just divided into four bins, it is difficult to rank different models, or correlate with other scoring functions, and if we want

to use it in a machine learning algorithm, the information that One Hot Encoding can provide is also very limited.

DockQ is a continuous protein-protein docking model quality measure derived by combining Fnat, LRMS, and iRMS to a single score in the range [0, 1] that can be used to assess the quality of protein docking models. By using DockQ on the CAPRI model, the original CAPRI classification can be almost completely reproduced as incorrect, acceptable, medium, and high quality, with an average PPV of 94% at 90% recall. Since DockQ generalizes the CAPRI classification almost perfectly, it can be considered as a higher-resolution version of the CAPRI classification, allowing estimation of model docking quality.

The only problem when using DockQ to calculate the similarity of Docking is that DockQ is designed to compare the docking for only two sub-structures of a model, which means, for targets with more than two interacting chains. We need to clarify which chains to group together and also in which order to combine them. The good news is that when we are not sure in which order to combine multiple chains, there are options to try all possible chain combinations (-perm1 and -perm2), this is important if for instance a homo oligomer is interacting asymmetrically with a third partner, or if there are symmetries that make multiple solution possibly correct. With DockQ covering all chain combinations, we only need to care which chains should be grouped together. In our approach, when computing the docking similarity between two models with more than two chains, we first decompose the multimeric docking into pairs-wise docking of any two contacting units. The average evaluation score was then calculated from the pairwise docking evaluation tool. Take T1184 with three chains as an example, the way to calculate the docking similarity with DockQ is showed in Figure 3.22.

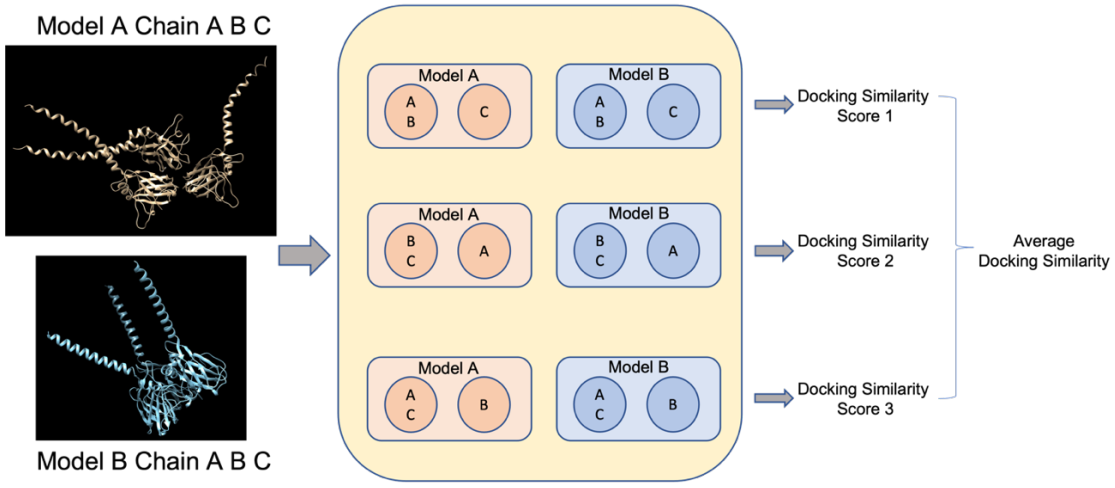


*Figure 3.22 Method used to calculate the docking similarity with DockQ for targets with more than two chains.*

***CASP15 Dataset***

Since CASP updated the units of the QA category from Angstrom to pLDDT, and it is unclear how to calculate the ground truth for the overall folding accuracy and overall interface accuracy of protein complexes, we are temporarily unable to intuitively evaluate the performance of the new method. But we still did some analysis of these two methods based on the data of MUFOLD group in CASP15.

In CASP15, totally 44 targets with multiple chains are released. The distribution of target with different chain numbers is showed in figure 3.23. For each CASP target, we collected the predictions from Duffman group, our own MUFOLD group and another AlphaFold2 based tool ColabFold's prediction. All three group are doing prediction with AlphaFold-Multimer, for each target, there will be 50 to 55 decoys (25 from Duffman server, 25 from MUFOLD server and 5 from Colab server).

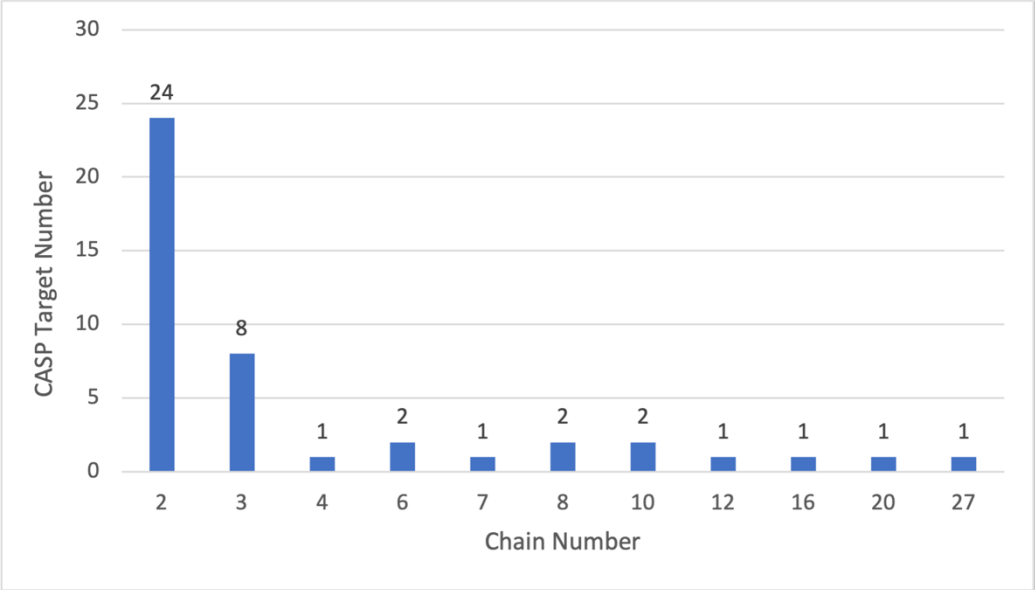


*Figure 3.23 distribution of target with different chain numbers.*

***Analysis of two QA methods***

During the CASP we found that since DockQ will try all possible chain combinations in each group, which means for two groups with $m$ and $n$ chains, there will be totally $n! \times m!$ combinations. And this is only for a combination. For a target with $k$ chains, when we compare the docking similarity between two models, the time we need to run DockQ $R$ will be:

$$R = \sum_{ceil\left(\frac{k}{2}\right)}^{i=1} C_k^i \times i! \times (k-i)!$$

When $k = 3$, run number will be 6, and when $k = 4$, run number will be 48.

The following figure 3.24 show the relationship between total sequence length and the time it takes to run one DockQ between two model of targets with two chains (left) and three chains (right). If a target has 55 decoys, then generate pairwise similarity matrix needs to run 2970 times. For longest two-chain target H1157, DockQ needs 7.4 hours to generate the similarity matrix, and for the longest three-chain target T1181, DockQ takes 73.12 hours to finish the

work. In CASP15, we only calculated docking accuracy for targets whose chain number is less than or equal to 4, because although the only four-chain target H1185 in CASP15 has only 35 decoy (needs 1190 times of comparison), the running time of DockQ has reached 109.78 hours. As a reference, when we generate similarity matrix with US-align, for H1135 with 12 chains, the time used for one comparison is only 15.23s, and the whole matrix generation for 55 decoys takes 12.57 hours.

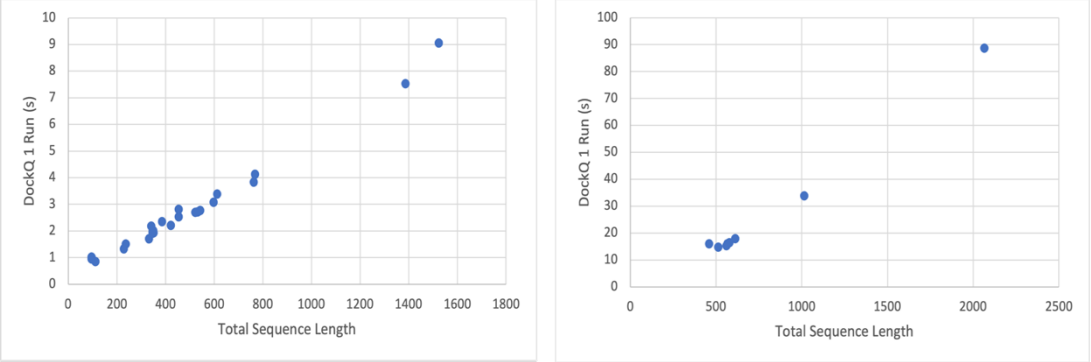


*Figure 3.24 The relationship between total sequence length and the time it takes to run one DockQ between two model of targets with two chains (left) and three chains (right)*

In addition to the time analysis of DockQ based consensus method, we also tried to analyze the correlation between DockQ based consensus score and US-align based consensus score, following figure shows the correlation between DockQ based consensus score and US-align based consensus score on 1760 models from 32 targets with 2 or 3 chains.

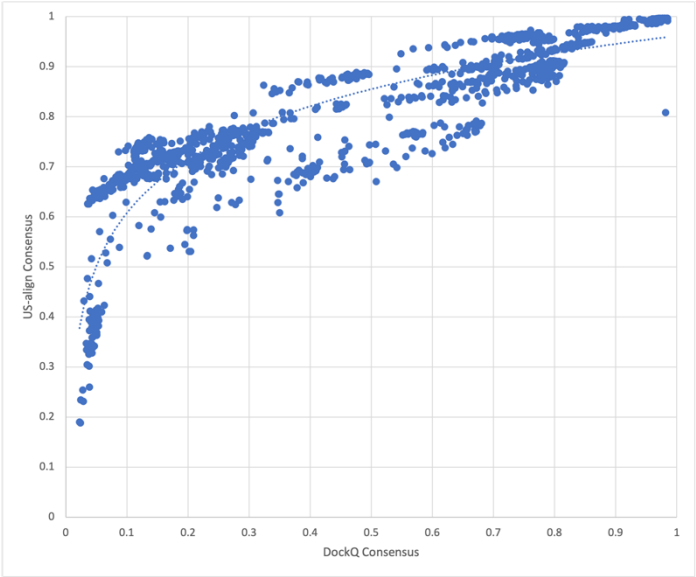


*Figure 3.25 The correlation between DockQ based consensus score and US-align based consensus score on 1760 models from 32 targets with 2 or 3 chains.*

There is a strong correlation between the consensus score base on DockQ and the consensus score base on UA-align, with all models together, the Pearson correlation between two score is 0.8824, and the target average Pearson correlation between two score is 0.7894. After CASP publishes the native structure and clarifies how to calculate the ground truth of the two QA scores, we will further evaluate the performance of the two methods on both ranking and selection.

# CHAPTER 4. IAFLOOP: PROTEIN LOOP MODELING USING ALPHAFOLD2

## 4.1 Motivations

AlphaFold2 has achieved outstanding results in 3-D protein structure prediction and can generate highly accurate models for moderate size proteins. The accuracy of its prediction on many targets even directly surpasses the accuracy of many intermediate steps of protein structure prediction, such as contact prediction and secondary structure prediction. At the same time, AlphaFold's success has also brought positive changes to many other problems, such as loop modeling. Since loop modeling is a small-scale structure prediction problem, AlphaFold2 is expected to work well on this problem.

In loop modeling problems, the loop regions are usually short compared to the full protein. A question when running AlphaFold2 to predict the loop regions is whether it is necessary to run Al-phaFold2 on the full protein sequence. When the protein is very long, AlphaFold2 may fail due to resource constraints. Running AlphaFold2 on the loop region sequence only will not work because the context of the loop region is essential in determining its unique structure. If AlphaFold2 is just run on a window of sequence around the loop region, then we need to determine appropriate window size and understand solution quality and computation time tradeoffs. In addition to this, AlphaFold2's prediction process can be split into a feature generation stage and a modeling stage. The efficiency and running time of AlphaFold2 heavily depend on whether a complete or reduced database is used to generate features during the feature generation stage and whether the ensembling process is used in the modeling stage. Therefore, whether a

complete database and ensembling process are needed is also an important topic when we try to use AlphaFold2 to solve loop modeling problems.

## 4.2 Problem Formulation

The loop modeling problem can be formulated as following: Given a protein sequence S containing a continuous segment R (called loop region), where the 3-D structure of S is known except for the R region, predict the 3-D structure of the R region, i.e., generate the 3-D coordinates of the residues in the R region as output.

A commonly used evaluation metric is the root mean square deviation (RMSD). Assume the true 3-D structure of a loop region is L and a predicted 3-D structure L'. The RMSD value between L and L' is calculated using the coordinates of the corresponding main chain atoms (*N, Cα, C* and *O*) between *L* and *L'*, as shown in the following formula:

$$RMSD_{loop} = \sqrt{\frac{1}{|L|}\sum_{i=1}^{|L|}\|L - L'\|^2}$$

*L'* need to be superimposed to *L* before calculation of RMSD.

## 4.3 Applying AlphaFold2 to Loop Modeling

### 4.3.1 Dataset

The benchmark dataset we used to compare the performance between different settings of Al-phaFold2 contains 40 backbone perturbed targets from Park's paper. The dataset is divided into two 20-target subsets: 1) a subset of 8-residue instances (20 targets, each containing an 8-residue missing structure region) and 2) a subset of 12-residue in-stances (20 targets, each containing a 12-residue missing structure region).

AlphaFold2 ran successfully on our computer on all except one target in the dataset. A 12-residue in-stance, 1ms9, has 1245 sequence length. When we ran AlphaFold2 on the full-length target, the program failed due to insufficient resources.

### 4.3.2 Naïve AlphaFold2 for Loop Modeling

As the best protein structure prediction tool, Al-phaFold2 can be directly used to loop modeling by predicting the structure of the missing region using the default parameter settings, given the full sequence of the target protein as input. This method is called Naive AlphaFold2 loop modeling method in this work.

In this work, the naïve AlphaFold2 method is compared with AlphaFold2 running with non-default settings, previous loop modeling methods, and the new method IAFLoop, on various test instances.

### 4.3.3 Effects of input sequence length on AlphaFold2

we compare the performance of AlphaFold2 on various window sizes around the loop regions. Specifically, we tried window size 50, 100, 200, and 300. For a window size $W$, if the loop region is of length $L$, then the window will contain a continuous segment of the protein sequence that includes (W-L)/2 residues before the loop region, the loop region, and (W-L)/2 residues after the loop region. So, the loop region is at the center of the window segment. For example, for an 8-residue loop region, a window of size 100 will contain 46 residues before the loop region, the 8-residue loop region, and 46 residues after the loop region, in the target protein sequence. When a loop region is located near the beginning or end of a protein sequence, the window may be truncated at the beginning or end.

Figures 4.1 show the performance of Al-phaFold2 running on various input segment window sizes around the loop regions on the 8-residue and 12-residue loop dataset. The loop modeling quality in terms of the RMSD of model 1 generated by Al-phaFold2 is reported. AF_FULL denotes running AlphaFold2 using the full protein sequence as input. AF_50, AF_100, AF_200, and AF_300 denote running AlphaFold2 with window sizes 50, 100, 200, and 300 around the loop regions as input, respectively.

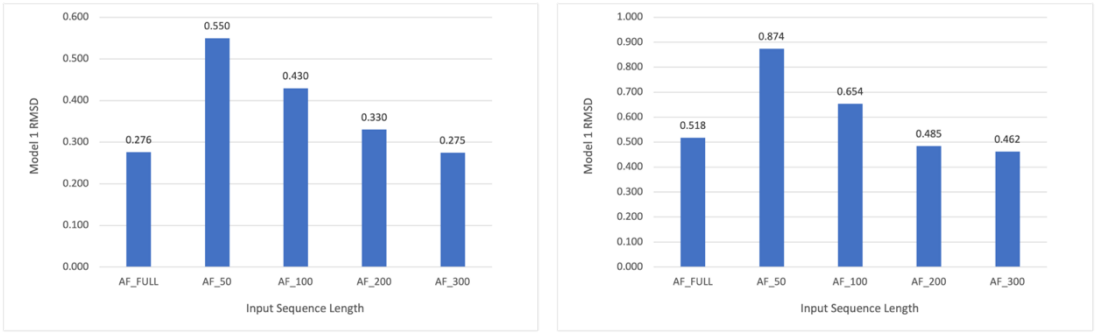


*Figure 4.1 Performance comparison of RMSD scores of loop models generated by AlphaFold2 on the 8-residue loop dataset (left) and 12-residual loop dataset (right).*

The results show that window sizes 50 and 100 are too small and do not provide enough context information to generate good loop models. Window sizes 200 and 300 led to significantly better models, signaling sufficient context information for the 8-residue or 12-residue loop modelling problems. With window size 300, the models generated by AlphaFold2 are as good or better than those generated by AlphaFold2 with the full protein sequences as input.

When the protein of a loop modeling problem is short, the segment of window size 300 around its loop region will be about the length of the whole sequence. Next, we focus on performance comparison on longer proteins in the test dataset.

Figure 4.2 shows the performance of AlphaFold2 running on various input segment window sizes around the loop regions on 14 8-residue and 12-residue instances that are

longer than 350 residues. Again, the results from window size 50 and 100 are much worse, whereas the results from window size 200 and 300 are very good, even much better than those using full protein sequences as input. Window size 300 got the best results, signaling that sufficient context information exists in about 300 residues around a small loop region to produce near optimal predictive models.

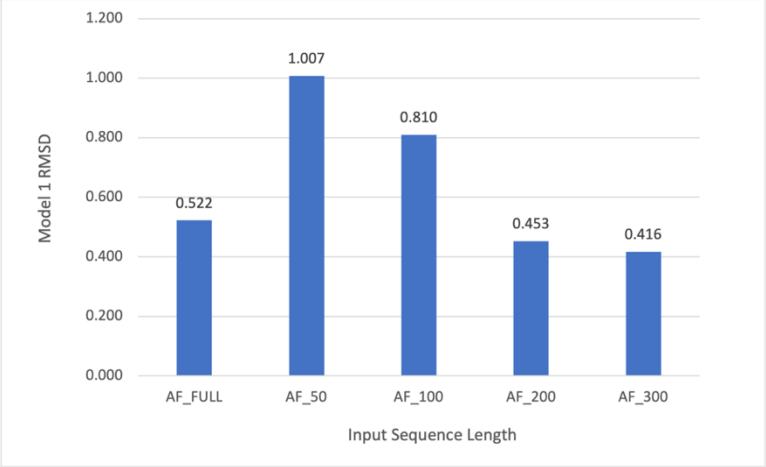


*Figure 4.2 Performance comparison of RMSD scores of loop models generated by AlphaFold2 on the 8-residue and 12-residue loop modeling proteins longer than 350 residues.*

### 4.3.4 Effects of Simplified Prediction Process on AlphaFold2

As the AlphaFold2's prediction process can be split into a feature generation stage and a modeling stage. The efficiency and running time of AlphaFold2 heavily depend on whether a complete or reduced database is used to generate features during the feature generation stage and whether the ensembling process is used in the modeling stage. We tested three versions of AlphaFold2 in our experiments:

1) AF -- the default, complete AlphaFold2 prediction process,
2) AF_fast -- no ensembling, and
3) AF_mini -- reduced version of database and no ensembling.

Figure 4.3 shows the computational times of three versions of AlphaFold2 on 8-residue and 12-residue problems. The times are divided into those for the feature generation.

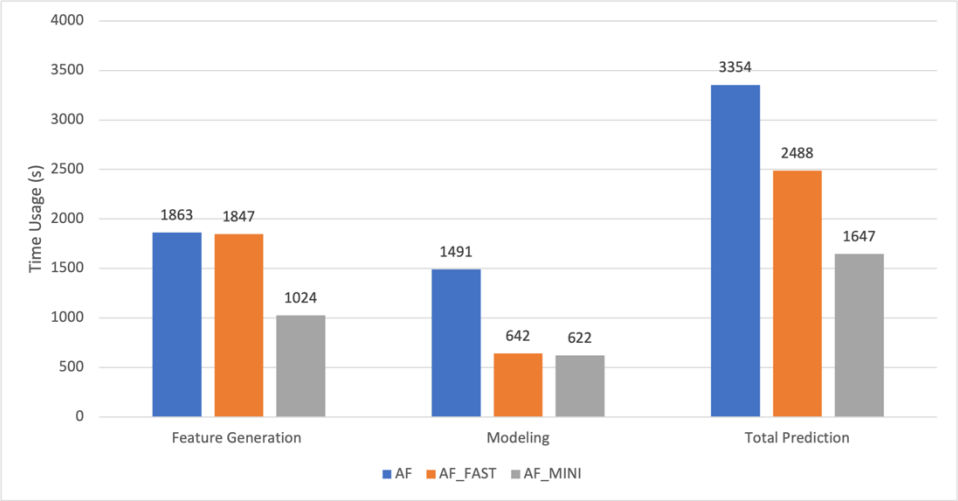


*Figure 4.3 Average execution times used by the feature generation, modeling and total prediction in three versions of AlphaFold2 (AF, AF_fast, and AF_mini) on 8-residue and 12-residue dataset.*

From figure 4.3 we can see that using reduced version of the database in AF_mini brought down the execution time in the feature generation stage to about half. Removing ensembling in AF_mini and AF_fast saved more than half of the execution time in the modeling stage.

Figure 4.4 show the performance comparison between three versions of AlphaFold2 (AF, AF_fast, and AF_mini) on the benchmark dataset. The RMSD of the model 1 generated by AlphaFold2 is used as the performance metric. The results show that models of similar quality were generated by the three versions of AlphaFold2. AF_mini is slightly better than the other two versions, which is a little bit of counter intuitive. One explanation is that the loop modeling problem is relatively simple compared to predicting structures of longer proteins, for which AlphaFold2 was designed and trained. The full version of AlphaFold2 prediction may be overtrained and overfitting for loop modeling. The simpler

AF_mini version is more robust. The good news is that AF_mini is not only fast, but also generates models as good as the full version of AlphaFold2 does.

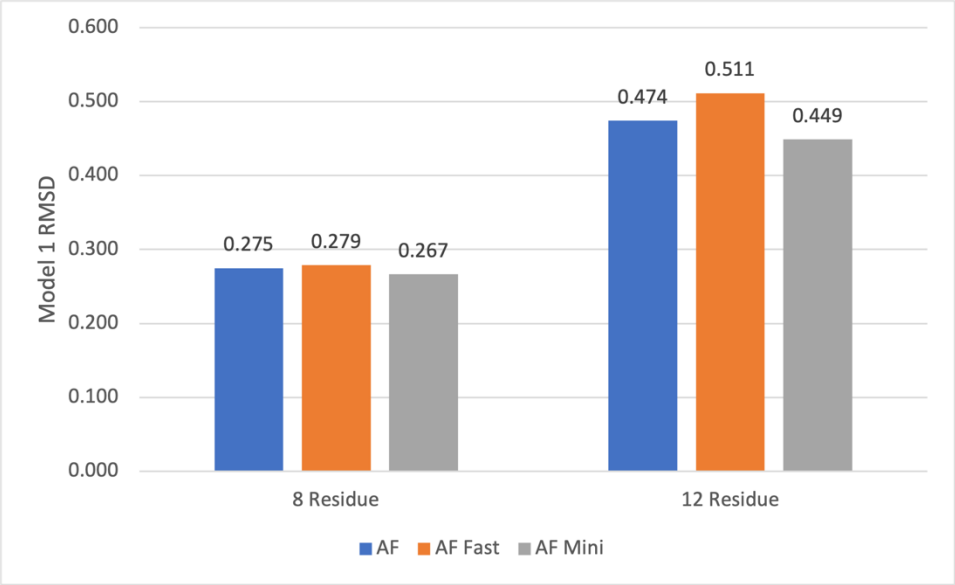


*Figure 4.4 Performance comparison of RMSD scores of loop models generated by three versions of AlphaFold2 (AF, AF_fast, and AF_mini) on the 8-residue loop dataset and 12-residue loop dataset.*

## 4.4 IAFLoop – a New Sublinear-Time AlphaFold2 protocol

### 4.4.1 Method

Based on the performance analysis of AlphaFold2 reported in the last section, we proposed a new efficient AlphaFold2 protocol called IAFLoop for loop modelling. It has a sublinear computation time related to the length of the target protein. IAFLoop, as shown in figure 4.5, consists of three steps:

1) loop region extension,
2) structure prediction using AF_mini and
3) model selection based on consensus.

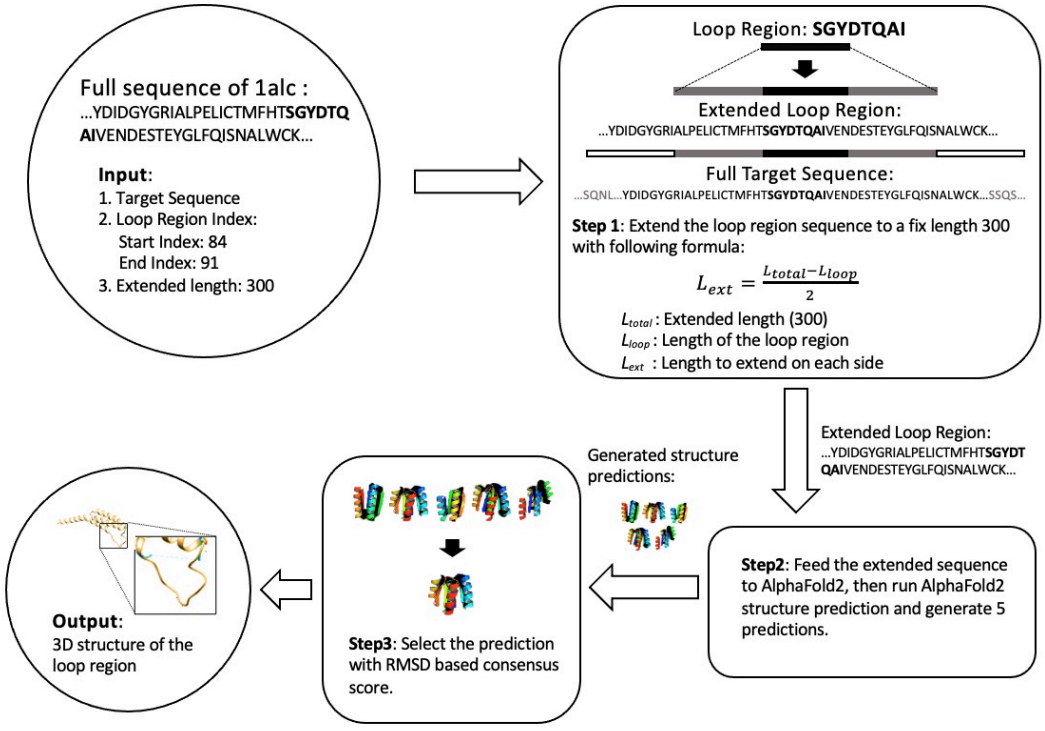


*Figure 4.5 The flowchart of a new method IAFLoop.*

### *Loop Region Extension*

Given a protein sequence with loop region indices, we first extend the sequence of the loop region on both side to a pre-set length, such as 300. The loop region will be at the middle of this extended segment. For example, if the length of the loop region is 8 and the extended length is 300, then there will be 146 residues on each side of the loop region in the extended segment. We used 300 as the extended length in our program.

When the loop region is near the head or tail of the protein, we only extend to the first or last amino acid of the protein. In these cases, the loop regions may not be in the center of the extended segments and the lengths of the extended segments are less than the pre-set length.

***Structure Prediction using AF_mini***

The protein sequence of the extended regiment is given to AF_mini (AlphaFold2 running with reduced database and no ensembling) to generate five predicted models.

***Model Selection***

A consensus method is used to select the final model from the five candidates generated. RMSD is used to calculate the average distance between the $C_\alpha$ atoms of two protein models, which measures the similarity of three-dimensional structures of the $C_\alpha$ atoms. Low RMSD score means the $C_\alpha$ atom structures are similar. We calculate pairwise RMSD scores between all candidate models generated by AlphaFold2 and compute the average similarity score for each model. The model with the smallest average similarity score is selected. The loop region of the selected model is the final output.

### 4.4.2 Evaluation

In this section, we compare the new method IAFLoop with existing loop modeling tools and naïve AlphaFold2 Loop modeling method on commonly used benchmark datasets and a new dataset we created from CASP14 targets.

*Table 4.1 Performance comparison of IAFLoop with state-of-the-art loop modeling tools. The performance metric is RMSD in unit Å between a predicted model and the native structure. Dataset: 8-residue loop dataset.*

| PDB* | NGK | Galaxy PS1 | Galaxy PS2 | Exp_GAN | AlphaFold | IAFLoop |
|---|---|---|---|---|---|---|
| **135L** | 0.3 | 1.5 | 0.5 | 1.2 | 0.225 | 0.193 |
| **1ALC** | 0.3 | 0.4 | 0.4 | 0.3 | 0.172 | 0.176 |
| **1BTL** | 0.4 | 1.5 | 1.2 | 1.2 | 0.195 | 0.24 |
| **1CEX** | 0.3 | 0.7 | 1.1 | 1.1 | 0.148 | 0.171 |
| **1CLC** | 0.4 | 0.4 | 0.3 | 0.6 | 0.119 | 0.111 |
| **1DDT** | 1.0 | 1.1 | 1.0 | 1.5 | 0.182 | 0.193 |
| **1EZM** | 0.3 | 2.3 | 0.6 | 2.2 | 0.623 | 0.401 |
| **1HFC** | 0.5 | 0.9 | 0.6 | 0.9 | 0.145 | 0.132 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **1IAB** | 0.5 | 2.1 | 1.9 | 1.3 | 0.143 | 0.16 |
| **1IVD** | 0.8 | 3.5 | 2.9 | 1.7 | 0.434 | 0.423 |
| **1LST** | 0.5 | 0.6 | 0.5 | 1.2 | 0.132 | 0.138 |
| **1NAR** | 1.3 | 2.6 | 2.0 | 2.7 | 0.16 | 0.191 |
| **1OYC** | 0.3 | 0.6 | 0.5 | 0.7 | 0.161 | 0.151 |
| **1PRN** | 0.3 | 1.6 | 0.5 | 1.0 | 1.375 | 1.385 |
| **1SBP** | 0.3 | 0.6 | 0.3 | 1.0 | 0.205 | 0.179 |
| **1TML** | 0.5 | 1.6 | 0.9 | 2.1 | 0.102 | 0.12 |
| **2CMD** | 0.4 | 0.6 | 0.5 | 0.9 | 0.264 | 0.244 |
| **2EXO** | 0.3 | 0.9 | 0.5 | 0.6 | 0.182 | 0.198 |
| **2SGA** | 1.3 | 1.1 | 0.9 | - | 0.346 | 0.282 |
| **5P21** | 0.3 | 0.8 | 0.3 | 0.5 | 0.208 | 0.227 |
| **AVG.** | 0.515 | 1.27 | 0.87 | 1.163 | 0.276 | 0.266 |

*Table 4.2 Performance comparison of IAFLoop with state-of-the-art loop modeling tools. The performance metric is RMSD in unit Å between a predicted model and the native structure. Dataset: 12-residue loop dataset.*

| **PDB*** | **NGK** | **Galaxy PS1** | **Galaxy PS2** | **Exp_GAN** | **AlphaFold** | **IAFLoop** |
|---|---|---|---|---|---|---|
| **1A8D** | 5.2 | 2.8 | 0.3 | 2.7 | 3.06 | 1.514 |
| **1ARB** | 0.4 | 3.7 | 2.0 | 2.1 | 0.638 | 0.689 |
| **1BHE** | 0.4 | 0.9 | 1.0 | 3.2 | 0.213 | 0.189 |
| **1BN8** | 1.1 | 1.3 | 0.7 | 1.7 | 0.222 | 0.204 |
| **1C5E** | 0.4 | 2.8 | 1.6 | - | 0.298 | 0.314 |
| **1CB0** | 0.6 | 0.5 | 0.5 | 1.7 | 0.267 | 0.247 |
| **1CNV** | 2.0 | 3.3 | 2.5 | 2.0 | 0.324 | 0.377 |
| **1CS6** | 2.5 | 3.7 | 3.6 | 4.0 | 0.597 | 0.607 |
| **1DQZ** | 0.6 | 1.1 | 0.7 | 2.6 | 0.35 | 0.293 |
| **1EXM** | 1.0 | 2.9 | 1.2 | 0.6 | 0.31 | 0.33 |
| **1F46** | 2.1 | 1.4 | 1.6 | 2.7 | 0.978 | 0.898 |
| **1I7P** | 0.4 | 2.9 | 0.3 | 1.3 | 0.272 | 0.274 |
| **1M3S** | 6.4 | 5.4 | 6.0 | 2.0 | 0.232 | 0.215 |
| **1MY7** | 0.6 | 2.2 | 2.6 | 2.2 | 0.291 | 0.285 |
| **1OTH** | 0.4 | 0.9 | 0.5 | 2.2 | 0.313 | 0.33 |
| **1OYC** | 0.4 | 2.1 | 2.1 | 0.4 | 0.167 | 0.164 |
| **1QLW** | 4.8 | 4.3 | 1.5 | 2.9 | 0.742 | 0.603 |
| **1T1D** | 0.7 | 3.5 | 1.6 | 2.6 | 0.289 | 0.307 |
| **2PIA** | 0.8 | 0.9 | 0.8 | 1.4 | 0.28 | 0.306 |
| **AVG.** | 1.62 | 2.45 | 1.63 | 2.09 | 0.518 | 0.429 |

***Benchmark Dataset***

The benchmark dataset includes 40 backbone perturbed targets as described in Section 4.3.1. In this dataset, 20 targets have 8-residue loop regions with missing structures, and another 20 targets have 12-residue loop regions with missing structures. One 12-residue instance, 1ms9, is too long (length 1245) for AlphaFold2 to run successfully.

***Experimental Results***

We compared the new method IAFLoop with many state-of-the-art loop modeling tools, including NGK, Galaxy PS1, Galaxy PS2, Exp_GAN, HyLooper, and Naïve AlphaFold2 for loop modeling with the whole protein sequence as input [16][18]. The results on 8-residue test cases are shown in Table 1, while the results on 12-residue test cases are in Table 2. The average RMSD scores in unit Å are reported.

The results show that AlphaFold2 greatly outperformed all existing loop modeling tools. IAFLoop slightly improved naïve AlphaFold2 for loop modeling that predicts the structure of the whole target protein. When compared with naïve AlphaFold2 for loop modeling, IAFLoop has an improvement of 3.6% and 17.1% on the two test sets.

Figure 4.6 shows the time usage comparison between naïve AlphaFold2 for loop modeling and IAFLoop. IAFLoop runs much faster, on average, using only around 33% of the time on both 8-residue and 12-residue cases.

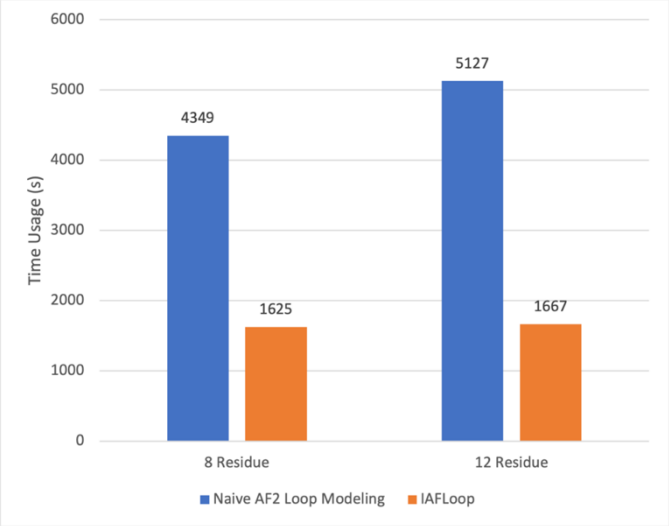


*Figure 4.6 Time usage comparison between naïve AlphaFold2 for loop modeling and IAFLoop on 8-residue and 12-residue test sets*

Figures 4.7 show the running time of naïve AlphaFold2 for loop modeling and IAFLoop against the length of the target protein on the 8-residue and 12-residue dataset, respectively. The running time of naïve AlphaFold2 for loop modeling grows linearly with respect to the length of the target protein, whereas IAFLoop has a sublinear computation time related to the length of the target protein. The advantage of IAFLoop over naïve AlphaFold2 method are significant for long proteins. When the length of the target sequence is greater than 300, the running time of naïve AlphaFold2 exceeded one hour and was growing, where IAFLoop just took about 20 minutes.

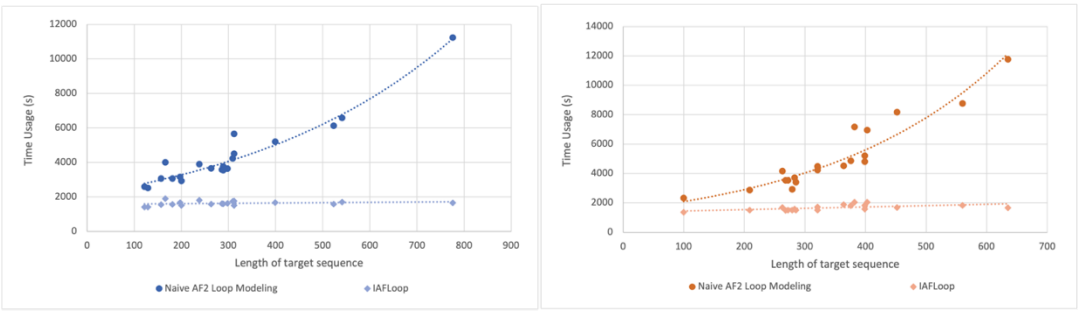


*Figure 4.7 Running time of naïve AlphaFold2 for loop modeling and IAFLoop against the length of the target protein on the 8-residue dataset (left) and 12-residue dataset (right).*

## 4.5 A New Loop Modeling Dataset Created from CASP14 Targets

AlphaFold2 was tremendously successful in CASP14, outperforming the other participants on average by a large margin. Although AlphaFold2 generated highly accurate models on most CASP14 targets, there are still some targets that are hard for AlphaFold2, and some loop regions of the targets were predicted poorly. We collected some of these difficult loop regions and created a new loop modeling dataset. The results of AlphaFold2 based loop modeling methods, including IAFLoop, are much worse on this dataset than those on the previous benchmark datasets. This new dataset may help researchers develop new methods in the future to improve over AlphaFold2.

The content of this dataset is shown in Table 4.3. It contains 23 loop regions from 15 CASP14 targets. The average RMSD error of naïve AlphaFold2 and IAFLoop are both 3.6 Å, 13 times higher than the RMSD error 0.266 Å of IAFLoop in Table 1 and 8 times higher than the RMSD error 0.429 Å of IAFLoop in Table 4.2.

The new dataset is made available to the public at https://wormlin.com/dataset

*Table 4.3 A new loop modelling dataset created based on CASP14 targets*

| CASP Target | Length | Start Index | End Index |
|---|---|---|---|
| T1027 | 11 | 1 | 11 |
| | 9 | 28 | 36 |
| | 25 | 73 | 97 |
| T1029 | 11 | 18 | 28 |
| T1033 | 9 | 1 | 9 |
| | 13 | 31 | 43 |
| T1039 | 11 | 92 | 102 |
| T1040 | 9 | 78 | 86 |
| T1043 | 9 | 10 | 18 |
| | 9 | 27 | 35 |
| | 10 | 41 | 50 |
| | 15 | 101 | 115 |

| | | | |
|---|---|---|---|
| T1050 | 11 | 669 | 679 |
| T1055 | 9 | 3 | 11 |
| T1061 | 13 | 229 | 241 |
| | 8 | 255 | 262 |
| T1064 | 9 | 18 | 26 |
| | 13 | 52 | 64 |
| T1067 | 17 | 122 | 138 |
| T1083 | 9 | 52 | 60 |
| T1093 | 17 | 33 | 49 |
| T1098 | 15 | 340 | 354 |
| T1101 | 8 | 266 | 273 |

## 4.6 Conclusion

In this work, we evaluated the performance of AlphaFold2 in loop modeling. Using commonly used loop modelling benchmark datasets, we showed that AlphaFold2 generated highly accurate loop models, outperforming previous methods by a large margin. By experimenting with limited local segments of various lengths around loop regions as input to AlphaFold2, we discovered that a AlphaFold2 of length 300 contains sufficient context information that led to high precision loop models, which are usually similar to or better than the ones generated by AlphaFold2 using the whole protein sequence as input. A main benefit of using a fixed size local segment to run AlphaFold2 is that the computational time grows very little as the length of the input protein sequence grows, instead of growing linearly with respect to the length of the input protein sequence.

In addition to running AlphaFold2 using the default parameter settings, we experimented with other faster versions of AlphaFold2 and found that the loop models generated by the faster versions are as good. Specifically, the fastest version with a reduced database and no ensembling produced slightly better loop models than the full version did.

Since the loop modeling problem is simpler than generic 3-D protein structure prediction, a reduced version of AlphaFold2 seems to be robust again overfitting.

Finally, armed with the understanding of AlphaFold2 properties, we proposed the new loop modeling method IAFLoop, which runs a reduced version of AlphaFold2 on an input sequence of length 300 covering the target loop region. Experimental results show that IAFLoop improved over naïve application of AlphaFold2 and reduced loop modelling errors by an order of magnitude over existing loop modelling methods. IAFLoop is efficient and runs in constant time.

Although AlphaFold2 works well on loop modeling benchmark datasets in general, there are still some test cases that are difficult for it and the loop models generated were of poor quality. We created a challenging dataset containing loop regions from targets in CASP14. This new dataset reveals some of the challenges facing AlphaFold2 and could be helpful to researchers in developing new methods to advance the loop modeling area.

# CHAPTER 5. MUFOLD-CONTACT: PROTEIN RESIDUE-RESIDUE CONTACT MAP PREDICTION WITH TWO STAGES DEEP LEARNING

## 5.1 Motivations

Protein contact map is a binary matrix. If we treat it as a binary image with one channel, then contact map prediction problem can be treated as a pixel-wise image classification problem. The most common pixel-wise image classification problem is image segmentation, in which the input is an image, and the output is a same size matrix with each pixel being classified into different categories. For example, if there are a desk and a chair in an image as input, then all pixels belong to the desk will be classified into one category and all the pixels belong to the chair will be classified into another category. The following figure 5.1 shows the concept of image semantic segmentation.

*Figure 5.1 Example of image semantic segmentation*

Inspired by image segmentation problem, our network is designed to do pixel-wise classification for protein contact map prediction. As Fully Convolutional Network (FCN) is always used on solving pixel-wise classification problem, we designed our network mainly based on “Fully Convolutional”, which also helped us to solve each protein has different length problem.

## 5.2 Problem Formulation

The protein residue-residue contact map prediction problem is addressed as follows. Given a protein sequence $S = \{R_1, R_2, \dots, R_L\}$. In the sequence, $R_i$ represents the $i$th $C_\beta$ atoms ($C_\alpha$ for glycine) of amino acid residue and the length of the sequence is $L$. Our goal is to predict a $L \times L$ matrix $C$, in which each $C_{ij}$ represents whether the distance of $R_i$ and $R_i$ are within a threshold. If the distance is in the threshold, $C_{ij}$ is 1, otherwise $C_{ij}$ is 0. This matrix $C$ is called the predicted contact map of protein sequence $S$.

## 5.3 Dataset and Features

### *Training Dataset*

The training dataset used in this work is a subset of CullPDB, it is generated with the following parameters:

*Table 5.1 Training Dataset PDB25 generation parameters*

| | |
|---|---|
| Database date | 2018-09-18* |
| Maximum percentage identity | 25% |
| Minimum resolution | 0.0 |
| Maximum resolution | 3.0 |
| Maximum R-value | 1 |
| Minimum chain length | 40 |
| Maximum chain length | 700 |
| Skip entries | non-X-ray, CA-only |

* To be able to test on CASP13 targets, we later manually remove all entries that are released after 2018-05-01.

Since the maximum percentage identity is 25%, this dataset is called PDB25. Totally there are 13116 entries in this dataset version 1. Then the following filtering are applied to the dataset:

1. Manually remove all entries that are released after 2018-05-01 so that the training dataset won't have any homology or native structure of CASP13 targets. In this way, the CASP13 targets can be used to compare the performance.
2. Remove all entries with Not a Number (NaN) values in the generated features or with missing features.

After the filters above, the version 2 dataset has 10896 entries. Then we noticed training samples with too many missing values in it or the sample itself is too short will lead to the training curve fluctuated sharply. So, we applied two more filters:

1. Limit the length of training sample in range 50 and 500.
2. Calculate the percentage of missing residues in the protein sequence. If the percentage is equal or greater to 5%, the training sample will be removed.

After the above filters, there are 5250 samples left in the version 3 dataset. The length distribution is shown in the following:

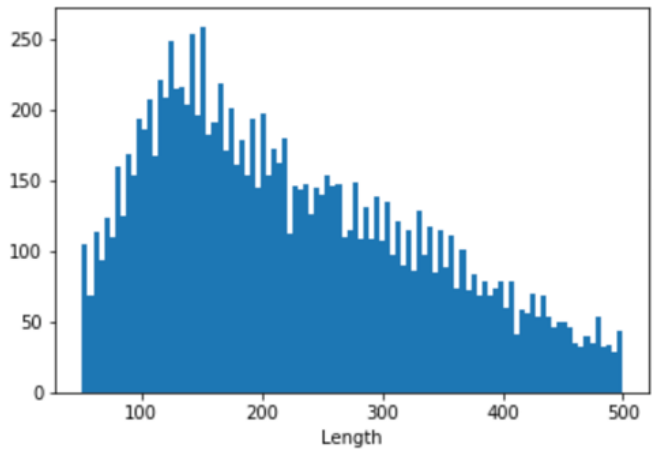


*Figure 5.2 Length distribution of all samples in filtered dataset version 3.*

***Feature List***

The features are all generated from the target sequence and can be categories into two kinds: the 1D features and the 2D features. The 1D feature means the feature vector is generated for each amino acid and the 2D feature means the feature vector is for each amino acid pair in the sequence. Table 5.2 summarized all the features that are used in this work together with the dimension of the feature, the software used to generate the feature, and the database if needed in the feature generation.

*Table 5.2 Overview of all features used in this work.*

| # | **Feature Name** | **Dimension** | **Software** | **Database (if any)** |
|---|---|---|---|---|
| 1 | PSSM | 20 | BLAST-2.7.1+ | uniref90_042016 |
| 2 | SS | 3 | Psipred_4.0 | Profile from 1 |
| 3 | SA | 1 | metaPSICOV 2.0.3 | Profile from 1 |
| 4 | Multiple Sequence Alignment (MSA) | - | HHsuite-2.0.16 | uniprot20_2016_02 |
| 5 | ALN Stats | L × L × 3 | metaPSICOV 2.0.3 → alnstats | MSA from 4 |
| 6 | HHM Profile | 30 | Hhsuite-2.0.16 → hhmake | MSA from 4 |
| 7 | CCMPred | L × L | CCMPred | MSA from 4 |
| 8 | EVFold | L × L | freecontact-1.0.21 | MSA from 4 |
| 9 | PSICOV | L × L | PSICOV | MSA from 4 |
| 10 | metaPSICOV | L × L | metaPSICOV 2.0.3 | From 2, 3, 5, 7, 8, 9 |
| 11 | shapeString | 8 | frag1d | PSSM from 1 |
| 12 | Raw Co-evolution Statistics | L × L × 441 | cov21stats | MSA from 4 |
| 13 | Physicochemical Features | 5 | Fixed values | - |

In order to make the model learning process more efficient, for real continuous value features, all values are normalized to the range between -1 and 1. Experiments show that

the network converges better and faster after normalization. We tried and compared a variety of normalization methods. This minimum and maximum scaler normalization achieved the best performance.

## 5.4 Deep Neural Network Structure

This work is based on the existing MUFOLD-CONTACT to try to replace part of Dilated Resnet with Dilated Inception net and compare whether the results have been improved accordingly.

***Dilated Convolution Layers***

The dilated convolution is also known as àtrous algorithm. It can greatly increase the size of receptive field in the convolutional layers. The basic idea is to adding spaces between points in the kernel. The following figure shows the kernel points and the corresponding receptive fields in dilated kernel. From left to right, the dilation is 0, 1, and 3 respectively.

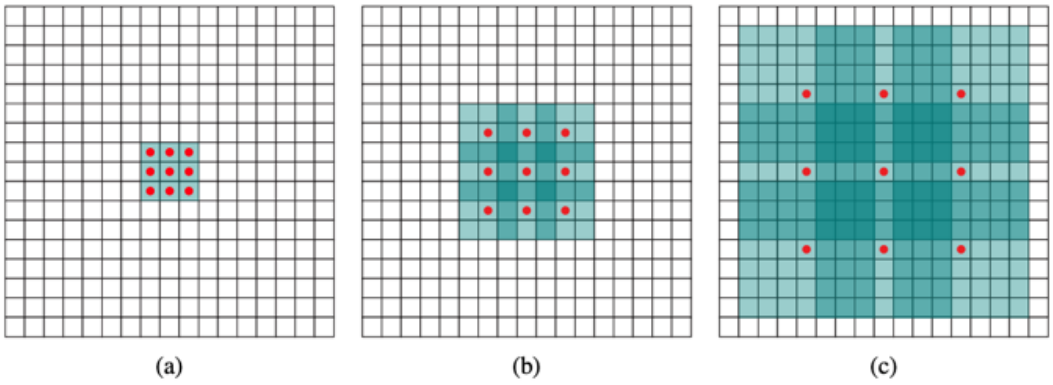


*Figure 5.3 Dilated kernels and the receptive fields.*

### *Two-Stage Multi-branch Network*

The distance map prediction is informative but not very accurate. While the binary contact map prediction may lose the distance distribution information since the final classes are only 0 and 1. Based on these intuitions, a two-stage multi-branch network to combine the distance map and binary contact map prediction has been designed and developed.

In the first stage, three networks with same structure predict the distance map for short, medium, and long range separately. Those predicted distance maps are then combined with our initial feature set in the second stage to predict the binary contact map. Which means the predicted distance map are used as the intermediate results for contact prediction in the whole process.

The following figure shows the overall structure of the two-stages multi-branch network.

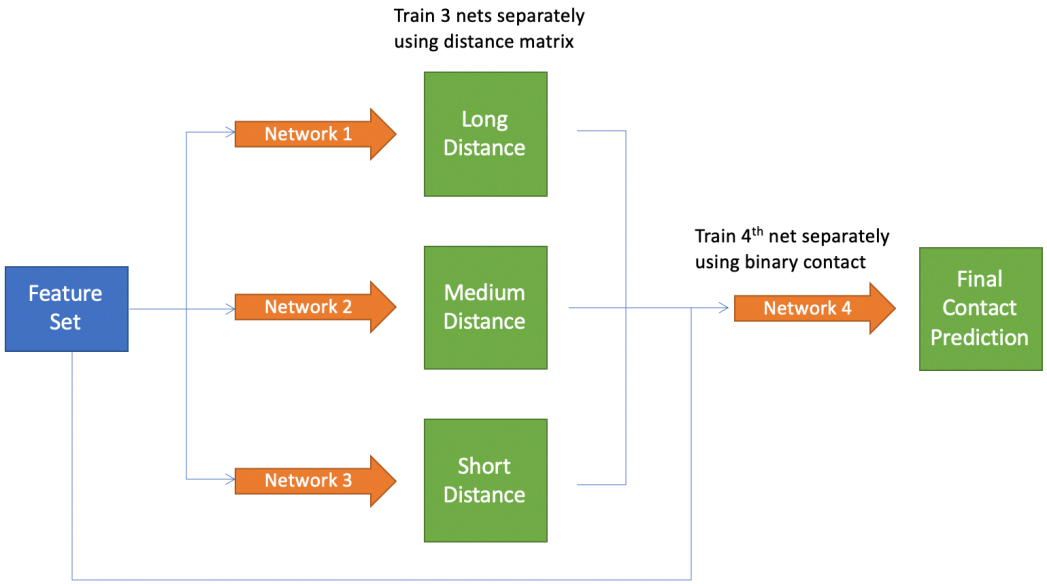


*Figure 5.4 Network structure of the two-stages multi-branch network.*

***Input Features***

For this network, the following features are generated from the sequence as the input, and each sequence has the real distance map as the ground truth.

1. Features for each residue, the features include the Position-Specific Scoring Matrix (PSSM), secondary structure prediction (3 states), solvent accessibility prediction (1 state), and the physicochemical features with dimension of 5. Therefore, for each residue, the feature vector length is 29. We call this set of features 1D feature.
2. Features for each residue pair, includes three evolutionary coupling scores from EVFold, PSICOV, and CCMPred respectively, and the mutual information, the normalized mutual information, and the contact potential, the dimension of those features are all 1 for each. Therefore, for each residue pair, the feature vector length is 6 and for a protein sequence with length L, the feature dimension is $L \times L \times 6$. We call this set of features 2D feature.

Three more features are added. In addition to all the features mentioned above, the metaPSICOV output, the Hidden Markov Model (HHM) profile and Shape String are added. The HHM profile and Shape String are for each residue, and they have the dimension of 30 and 8 respectively. The metaPSICOV output is for each residue pair and the dimension is $L \times L \times 1$. The final size for the 1D features is $L \times 67$ and for the 2D features is $L \times L \times 7$.

***Stage 1: Distance Map Prediction***

The basic structure of this network is like previous networks but replace two different part's residual block with Inception block V3. There are 1D and 2D features, 1D features will be processed first, but instead of combining the output with 2D features directly, the

2D features will be processed before the combination. The following figure 5.5, 5.6 and 5.7 shows the structure of the original MUFOLD-CONTACT network and two new network structures separately.

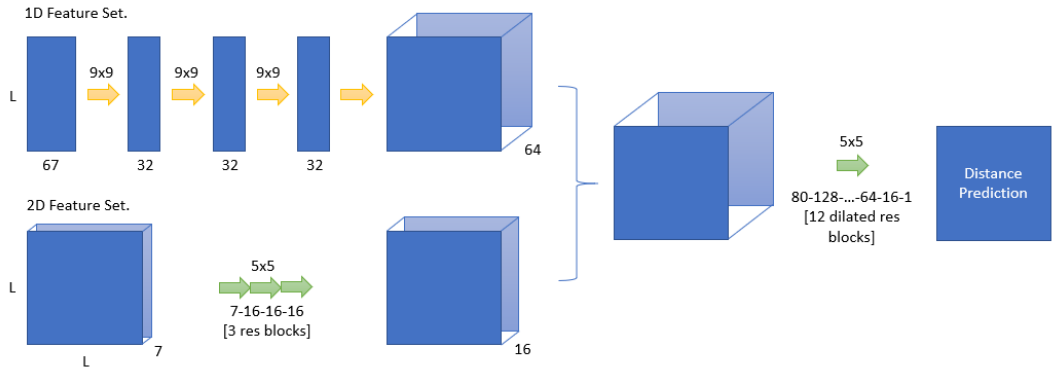


*Figure 5.5 Original Dilated ResNet structure for distance prediction in the first stage in the two-stages multi-branch network.*

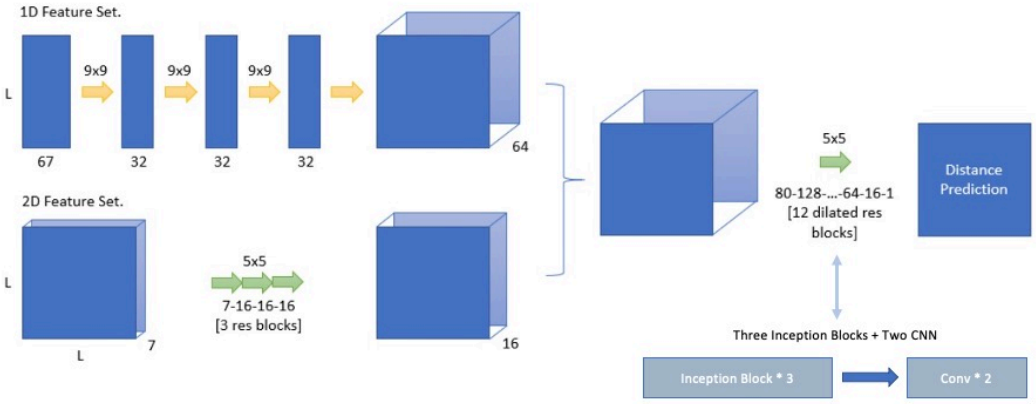


*Figure 5.6 ResNet + Inception Net structure V1 for distance prediction in the first stage in the two-stages multi-branch network.*

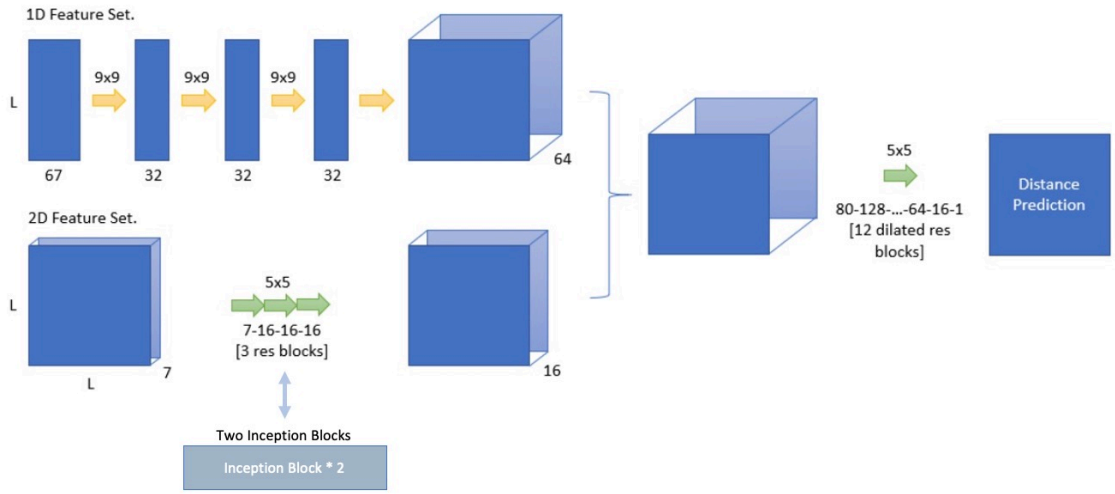


*Figure 5.7 ResNet + Inception Net structure V2 for distance prediction in the first stage in the two-stages multi-branch network.*

Three networks of two versions of ResNet + Inception Net with the same network structure will be trained for short, medium, and long-range prediction respectively. The intuition behind this is to let each network focus on each range to utilize the feature information.

***Stage 2: Binary Contact Map Prediction***

The output from stage 1 will be combined with the original feature set as the input to the network in stage 2. There are three distance map predictions in stage 1, original MUFOLD-CONTACT use all three distance map predictions as the additional features to stage 2. But here we changed the way to use these intermedia results, we will combine the three predicted distance map by take the short, medium, and long-range from different predicted distance maps respectively and generate a new predicted distance map, so the additional features to stage 2 reduce from dimension $L \times L \times 3$ to $L \times L \times 1$. For the network structure used in stage 2, after some test and comparison we finally choose the Original Dilated ResNet.

## 5.5 Evaluation Results

In this work we choose CASP13 targets as the test dataset to evaluate the performance. The training set is dated before the start date of CASP13 so there are no native structures contained in our training set. However, not all CASP13 targets have native structures, only 19 out of all targets are used because CASP has officially released the native structures for those 19 targets.

four widely used downloadable tools are chosen to compare the performance without proposed methods. All tools are downloaded and installed on our server to test. Online servers are excluded since we don't know if their training set contains natives of our testing set. The four tools are: FreeContact, PSICOV, CCMPred and metaPSICOV2. Except these four tools, we also added the original MUFOLD-CONTACT as a tool to compare with. The following table shows the performance of the comparison.

*Table 5.3 Contact prediction precision comparison with other tools using CASP13 targets.*

| **Tools** | **Long** | | | **Medium** | | | **Short** | | |
|---|---|---|---|---|---|---|---|---|---|
| | ***L/2*** | ***L/5*** | ***L/10*** | ***L/2*** | ***L/5*** | ***L/10*** | ***L/2*** | ***L/5*** | ***L/10*** |
| FreeContact | 0.071 | 0.092 | 0.147 | 0.077 | 0.105 | 0.165 | 0.085 | 0.093 | 0.124 |
| PSICOV | 0.224 | 0.279 | 0.365 | 0.155 | 0.233 | 0.365 | 0.155 | 0.225 | 0.365 |
| CCMPred | 0.227 | 0.277 | 0.395 | 0.174 | 0.266 | 0.411 | 0.175 | 0.225 | 0.395 |
| metaPSICOV | 0.337 | 0.450 | 0.526 | 0.404 | 0.562 | 0.747 | 0.409 | 0.576 | 0.784 |
| MUFOLD Contact_D | 0.282 | 0.408 | 0.482 | 0.295 | 0.384 | 0.482 | 0.204 | 0.258 | 0.335 |
| MUFOLD Contact_C | 0.388 | 0.492 | 0.574 | 0.411 | 0.578 | 0.763 | 0.382 | 0.536 | 0.742 |
| MUFOLD Contact | 0.524 | 0.630 | 0.687 | 0.451 | 0.625 | 0.706 | 0.449 | 0.661 | 0.753 |
| MUFOLD Contact V2 | 0.496 | 0.603 | 0.645 | 0.454 | 0.657 | 0.737 | 0.447 | 0.662 | 0.752 |
| MUFOLD Contact V3 | 0.530 | 0.649 | 0.708 | 0.488 | 0.662 | 0.766 | 0.485 | 0.688 | 0.788 |

From the above table, MUFold-Contact_V3 performed the best among all categories, and MUFold-Contact_V2 for comparable result with MUFOLD Contact, only a little worse than metaPSICOV on short and medium L/10.

## 5.6 Summary

The result of this work shows that using Inception block instead of Residual block on some part can improve the whole performance of the contact prediction process. And combine the three predicted distance maps is better than use all three of them. As when the network focus on one range of the distance map, other parts may become inaccuracy and includes more noise information.

# CHAPTER 6. MUFOLD REFINEMENT

## 6.1 Motivations

The appearance of AlphaFold in CASP13 surprised everyone. After the AlphaFold paper was published, one of its core ideas predicts the discrete probability distribution of the residual pair's distance, which also inspired more group. This discrete probability distribution can be fitted to a Cubic Spline and used to design a new potential score.

Inspired by this, we designed a new optimization process with new potential score based on discrete probability distribution, and an updated version of MUFOLD-REFINEMENT process It was designed and implemented.

## 6.2 Pipeline and Method

### 6.2.1 Refinement Pipeline

The updated version of MUFOLD-REFINEMENT pipeline includes Model optimization, structure mixing and structure selection, following is the overview of MUFOLD-Refinement pipeline:

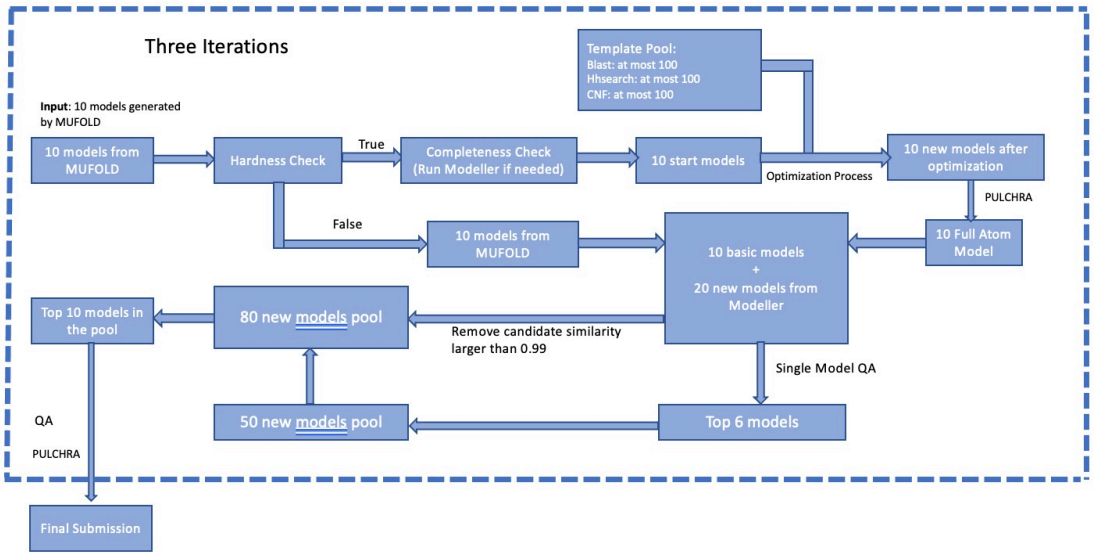


*Figure 6.1 Overview of MUFOLD-Refinement pipeline*

### 6.2.2 Potential Score Based on Discrete Probability Distribution

The design of Potential Score borrows the idea form AlphaFold's Discrete Probability Distribution idea, first of all, template search tool Blast, HHsearch and CNF will be run to generate a reference template pool for the target sequence, in order to ensure the performance of template pool, we only take at most top 100 template results from each tool.

Take 1OOK_G.pdb with length 25 as a simple example, after alignment search, we got a total of 248 templates. After removed diagonal line (residual pair distance is 0) 1OOK_G.pdb with length 25 contains 600 residual pairs in total. As not all alignment covers the whole target sequence, after converting all alignments into distance matrix representation, for each residual pair, there will be at most 248 distance matrixes cover each position.

For each residual pair ij, we do the following operation:

1. From all the template distance matrixes cover this position, find the max distance number $d_{max}$, then the number of distance bins can be calculated as:

$$N_{bins} = (d_{max} - 2.3)/0.3$$

Here we used the same lower bound as AlphaFold, and in order not to lose distance information, we did not set an upper limit like AlphaFold.

2. For each distance bin, calculate the probability that the $d_{ij}$ of template is in this range.
3. Take x as the medium distance of a distance bin and y as the probability that the $d_{ij}$ of template falls in this bin, fit all (distance, probability) to a Cubic Spline.

Take a residual pair (9, 23) as example, the max distance covers this position is 39.9886, so the number of distance bins is (39.9886 – 2.3)/0.3 = 126. After we fit the Cubic Spline, we notice that the curve is too complicated and needs smooth, after some test we finally choose smooth parameter = 0.03, figure 4.2 shows the example fitted Cubic Spline for residual pair (9, 23) of 1OOK_G:

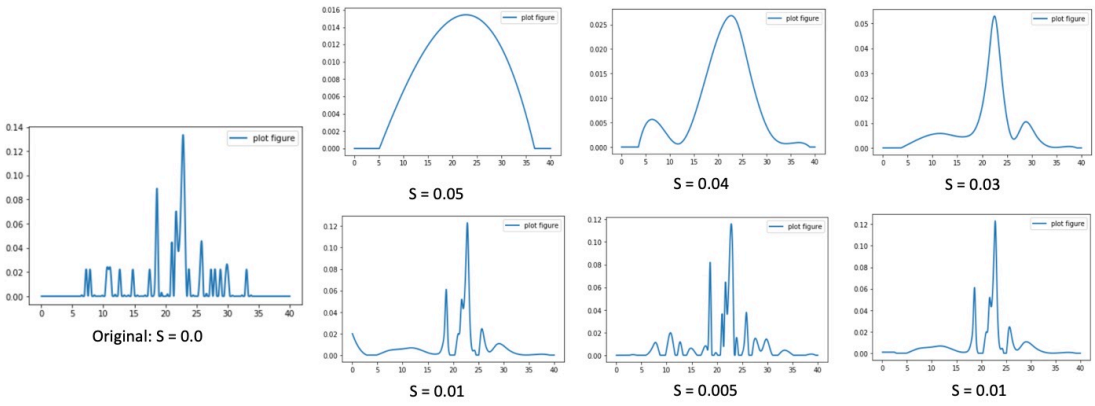


*Figure 6.2 Smooth parameter choose for fitting Cubic Spline*

The potential function used for doing optimization is defined as following:

$$Potential = -W_{t1} \sum P(d_{ij}) * W_{ij} + W_{t2} \sum (d_{i,i+1} - 3.8)^2$$

Where $Potential$ means potential score, $W_{t1}$ and $W_{t2}$ are the weight for two terms in the formula, $P$ is the corresponding value for distance $d_{ij}$ in cubic spline, and weight $W_{ij}$ can be defined as following:

$$W_{ij} = \frac{Number\ of\ template\ cover\ AA\ pair(i,j)}{Total\ number\ of\ template}$$

Which means the more templates cover this position, the more weight we will give to the potential score of this position.

For optimization function, we chose L-BFGS and for the hyperparameter we set epsilon (step size)=1e-03, pgtol (converge threshold)=1e-12, maxiter (max iteration)=15. After the optimization process, we will run MODELLER to make the prediction more protein like.

Take the MUFOLD predicted 1OOK_G TS1 as an example, the following figure shows the structure change after each step:


*Figure 6.3 Model structure after each step*

### 6.2.3 Hardness Check Module

After the testing on 10 targets of CASP13, we noticed that in some cases it is not suitable for run optimization process, so we designed a hardness check module to filter out these cases. If any of the following situations occur, we will not run the optimization process:

3. If the sequence length of the model is larger than 400, the optimization process may need more than 4 hours to converge, so we don't want to run optimization process in this case.
4. We will run our QA method on the input model, is the performance is larger than 0.8, we will not run optimization process, as when the performance is really good, the information from the template pool may become noisy and the optimization process will generally reduce the quality of the model rather than improve it.

The following figure shows the design of our hardness checking module.

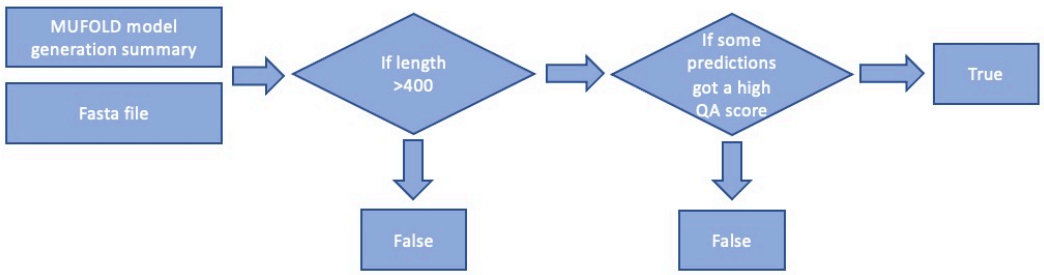


*Figure 6.4 Hardness checking module design*

### 6.2.4 Structure Mixing Module

After optimization process generated 10 new structures, they will be feed into MODELLER and generate another 20 modes, we will run our single model QA method on this small pool and select top 6 models for structure mixing.

In structure mixing module, we will take the model ranked No.1 as template, and first mix the template model with model ranked No.2 and generate 10 new models, then mix the template model together with model ranked No.2 and No.3 to generate another 10 new models… The following figure shows the design of the structure mixing module.

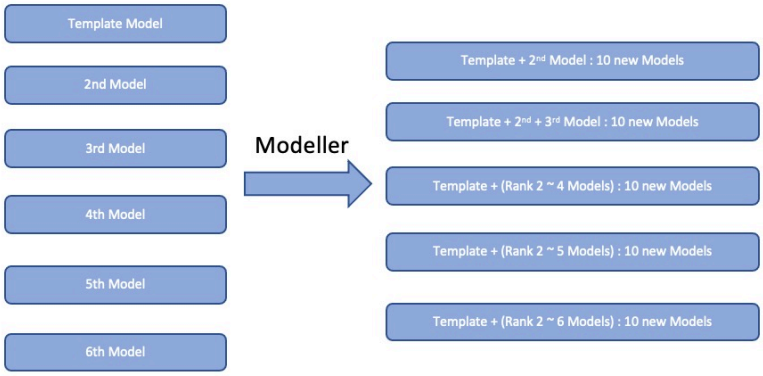


*Figure 6.5 Design of the structure mixing module*

## 6.3 Results

### 6.3.1 Test on CASP13 Dataset

After achieving success on target 1OOK_G, we tested our Optimization Process on 10 targets of CASP13 dataset with sequence length < 350 (T0950 sequence length 353). We used TS1 of MUFOLD protein structure prediction as the start model. The result is shown in the table below.

*Table 6.1 Optimization process test on 10 targets of CASP13*

| Target | Length | Start Model GDTTS | Optimized GDTTS | Time Information | Template number |
|---|---|---|---|---|---|
| T0950 | 353 | 0.144 | 0.1462 | 11732s | 82 |
| T0953s1 | 72 | 0.2118 | 0.2222 | 702s | 102 |
| T0955 | 41 | 0.3902 | 0.4024 | 361s | 134 |
| T0957s1 | 163 | 0.1728 | 0.1775 | 3192s | 39 |
| T0957s2 | 164 | 0.3703 | 0.3956 | 3185s | 86 |
| T0958 | 96 | 0.3929 | 0.3799 | 1094s | 90 |
| T0968s1 | 126 | 0.2034 | 0.2055 | 1871s | 80 |
| T0968s2 | 116 | 0.2217 | 0.2196 | 1540s | 63 |
| T0971 | 186 | 0.9015 | 0.8617 | 4235s | 370 |
| T1008 | 80 | 0.3734 | 0.3571 | 760s | 422 |

Form the result table above we can notice that the optimization process is not that stable as it only improved 6 targets among 10 totally, and when the sequence length is longer than 350, the running time may become quite long.

### 6.3.2 Test on CASP13 Refinement Dataset

In this work, we take 9 refinement targets form CASP13 to test the performance of our complete refinement process, we will use the start model given by CASP competition and check the performance of the final selected model.

During the refinement process, we will use single model QA method to do selection and ranking, but at the last selection step which is used to generate the final output structure, we tried multiple QA methods and the comparison is showed in the following table:

*Table 6.2 Performance on 9 refinement targets from CASP13*

| Target | Start Model GDTTS | MUFOLD_REF + MUFOLD_QA | MUFOLD_REF + consensus set150 | MUFOLD_REF + proq3 | MUFOLD_REF + consensus all server models |
|---|---|---|---|---|---|
| TR862 | 0.5887 | 0.4973 | 0.4677 | 0.4973 | 0.5887 |
| TR870 | 0.3780 | 0.3780 | 0.3394 | 0.378 | 0.3211 |
| TR872 | 0.7415 | 0.8153 | 0.7727 | 0.821 | 0.821 |
| TR879 | 0.7898 | 0.8000 | 0.7955 | 0.7909 | 0.7977 |
| TR893 | 0.8728 | 0.8550 | 0.855 | 0.8624 | 0.8491 |
| TR920 | 0.7945 | 0.7979 | 0.7945 | 0.7979 | 0.7888 |
| TR922 | 0.8311 | 0.8412 | 0.8446 | 0.8446 | 0.8446 |
| TR944 | 0.7401 | 0.7510 | 0.7520 | 0.7421 | 0.7480 |
| TR947 | 0.6614 | 0.6729 | 0.6457 | 0.6786 | 0.6586 |
| Average | 0.7109 | 0.7121 | 0.6963 | 0.7125 | 0.7131 |

From the result we can see after three iteration we can see the refinement process only improve the performance very tiny, and using MUFOLD_QA, Proq3 and consensus with all server models got quite similar performance on average.

## 6.4 Case Study

In order to further evaluate the performance of our refinement process and analysis the advantages and disadvantages, we kept some key performance data in the refinement performance, includes the best model in the pool after optimization, best model in the pool after structure mixing and the final selected model No.1 after each iteration.

### 6.4.1 Case Study – T0922

T0922 is a very representative target, which reflects one aspect of the refinement process, the performance of our QA method is still not strong enough. We can see in first iteration after the optimization process and structure mixing process, good structure s has been generated and added into the pool, but after selection process these good models are not selected, which lead to almost no significant performance improvement in the next two iterations.

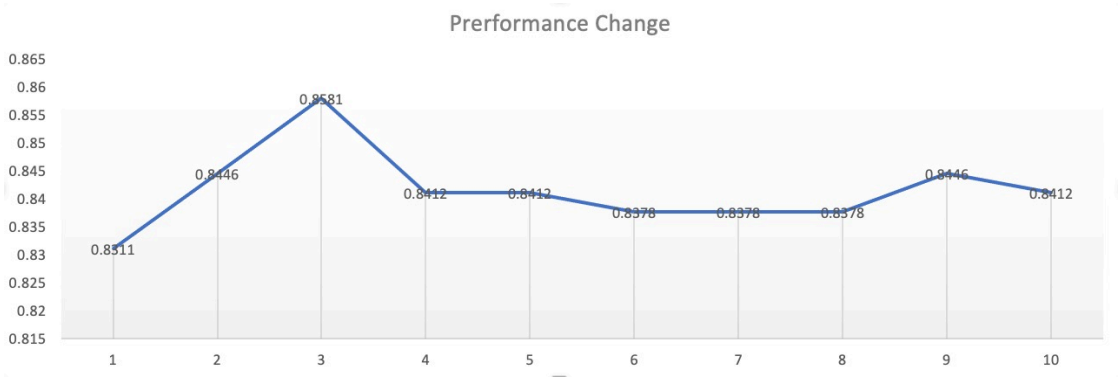


*Figure 6.6 T0922 performance change in each step*

### 6.4.2 Case Study – T0947

T0947 is actually a failed target as the output structure from the refinement process is worse than the input structure. The problem points to the selection step, and it also reflects

that the optimization and hybrid modules do not generate many good structures, which also increases the difficulty of the QA step.

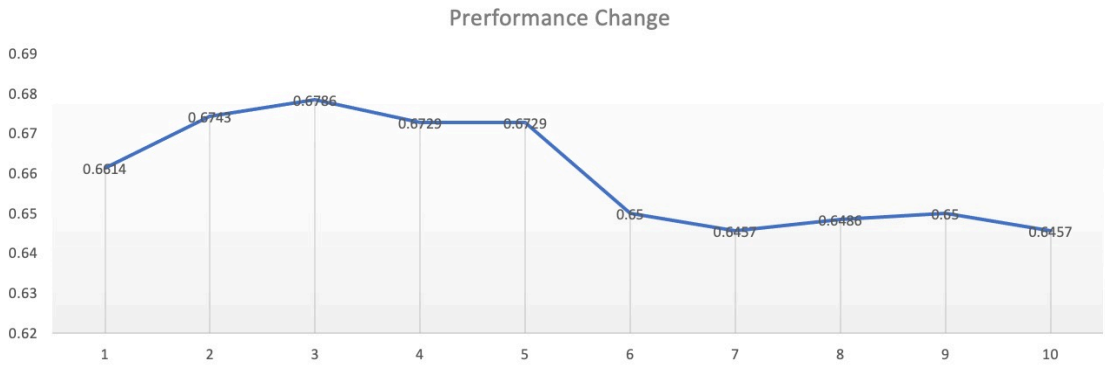


*Figure 6.7 T0947 performance change in each step*

## 6.5 Conclusion

In this work, we borrowed the discrete probability distribution idea from AlphaFold and designed our potential score function and optimization process, then applied the process in to MUFOLD-REFINEMENT pipeline, form the experiential result we can see the performance of the output model compared to start The model has only a small improvement, and the biggest problem of the entire refinement process is still QA method not strong enough, sometimes the good structures are generated by the optimization process or mixing process, but they cannot be selected and continue into the next iteration.

# CHAPTER 7. SUMMARY AND FUTURE WORK

## 7.1 Summary

In this thesis, two single model QA methods based on extra deep Residual neural network, one quasi-single model QA method based on Inception neural network, and three new single model QA methods using multi-stage machine learning and hierarchical ensemble has been proposed. A new simulated high performance decoy pool with the code and models released by AlphaFold2 and trRosetta has been prepared and used to analyze the impact of large improvements in performance distribution on different QA methods and commonly used features. For protein contact prediction, an updated version of MUFOLD-CONTACT with new neural network combines the Resnet and inception network has been proposed. For protein structure refinement, a refinement pipeline with new designed potential score function has been development.

For single model QA problem, totally two methods based on extra deep residual neural network and three machine learning based methods have been proposed, PDRN and VDRN are the first methods try to apply extra deep neural network on QA problem, they already got comparable result compared with state of art single-model QA methods on CASP 12 dataset, it proves that extra deep residual neural network can be applied to solve the protein QA problem. Three MMQA methods achieved perfect performance on QA problem with traditional machine learning methods by using heuristic training strategies. The two stage machine learning strategy, with entire feature set divided into two different groups and uses completely different feature sets and training data in each stage of machine learning

successfully overcome the overfitting problem of using complex machine learning and deep learning algorithms.

For the quasi-single-mode QA problem, MUFOLD-INC tried the goal-based deep learning method for the first time. It achieved performance comparable to other quasi-single mode QA methods and showed a new direction to use the template structures to improve the performance of the QA problem. Instead of simply using them as a reference structure pool.

With the code and model released by AlphaFold2 and trRosetta, a new high performance decoy pool has been prepared. After testing we noticed that when we do QA on the high performance decoy pool generated by good predictors like AlphaFold2, the performance of the multiple model QA methods is still better than the single model QA methods and single model QA methods may lead to obvious underestimate problem. On feature level, the effective information provided by many originally important structural features dropped significantly, while score functions shows better Pearson Correlation with true GDTTS score on the high performance decoy pool. Some features with low correlation earlier like OPUS_PSP and HOPPscore may provide much more information now. It is clear that the models trained on the old dataset are no longer applicable.

For contact map prediction problem, a new two-stages multi-branch network based on fully convolutional neural network and inception network has been proposed. This updated version further improve the performance of MUFOLD-CONTACT on all categories. In addition to the new network structure, this work also modified the way to deal with the intermedia result by combine the three predicted distance map into one. The improved

performance shows this operation successfully decrease the noise and improved the performance of the input features od second stage.

For protein structure refinement, we design and developed a refinement process MUFOLD-REFINE and borrowed the discrete probability distribution idea from AlphaFold in CASP13, and further updated our refinement process with the distance distribution of the template pool as constraint.

For loop modeling, by experimenting with limited local segments of various lengths around loop regions as input to AlphaFold2, we discovered that a AlphaFold2 of length 300 contains sufficient context information that led to high precision loop models, which are usually similar to or better than the ones generated by AlphaFold2 using the whole protein sequence as input. We also find that the fastest version of AlphaFold2 with a reduced database and no ensembling produced slightly better loop models than the full version did. Finally, we proposed the new loop modeling method IAFLoop, which runs a reduced version of AlphaFold2 on an input sequence of length 300 covering the target loop region. Experimental results show that IAFLoop improved over naïve application of AlphaFold2 and reduced loop modelling errors by an order of magnitude over existing loop modelling methods. IAFLoop is efficient and runs in sublinear time.

Finally, here is the list of publications related to the work in this research:

1. Wenbo Wang, Zhaoyu Li, Junlin Wang, Dong Xu and Yi Shang, "PSICA: a fast and accurate web-based platform for protein model quality analysis,", Nuclear acid research web server issue, 2019.

2. Junlin Wang, Zhaoyu Li, and Yi Shang. "New Deep Neural Networks for Protein Model Evaluation." 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2017.
3. Junlin Wang, Wenbo Wang, and Yi Shang. "A New Approach Of Applying Deep Learning To Protein Model Quality Assessment." 2019 IEEE International Conference on Bioinformatics and Biomedicine. IEEE, 2019.
4. Wenbo Wang, Junlin Wang, Dong Xu, and Yi Shang, "Two New Heuristic Methods for Protein Model Quality Assessment," IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2018, 17(4): 1430-1439.
5. Wenbo Wang, **Junlin Wang**, Zhaoyu Li, Dong Xu and Yi Shang. "MUfoldQA_G: High-Accuracy Protein Model QA via Retraining and Transformation". Computational and Structural Biotechnology Journal, 2021.
6. Junlin Wang, Wenbo Wang, Dong Xu, and Yi Shang, "New Heuristic Methods for Protein Model Quality Assessment via Two Stage Machine Learning and Hierarchical Ensemble." submitted, 2022.
7. Junlin Wang, Wenbo Wang, and Yi Shang, " Protein Loop Modeling Using AlphaFold2" submitted, 2022.

## 7.2 Future Work

There is some future work in terms of protein quality assessment. For the newly added QA category for protein complexes in CASP15, there is no native structure released yet, and it is not clear how to calculate the true overall folding accuracy and overall interface accuracy. Therefore, the QA method we used in CASP15 based on US-align and DockQ cannot be evaluated yet. Also, we are not sure if the method we used to calculate docking

similarity between to structures is correct with the target chain number larger than 2. After CASP publishes how to calculate true overall folding accuracy and overall interface accuracy, we can further verify the correctness and performance of our new method.

# REFERENCE


Baker D, Sali A. Protein structure prediction and structural genomics[J]. Science, 2001, 294(5540): 93-96.

Breiman L (2001) Random forests. Machine learning 45: 5-32.

Cao R, Bhattacharya D, Adhikari B, et al. Large-scale model quality assessment for improving protein tertiary structure prediction[J]. Bioinformatics, 2015, 31(12): i116-i123.

Chen X, Liu J, Guo Z, et al. Protein model accuracy estimation empowered by deep learning and inter-residue distance prediction in CASP14[J]. Scientific Reports, 2021, 11(1): 1-12.

Cheng J, Baldi P. Improved residue contact prediction using support vector machines and a large feature set[J]. BMC bioinformatics, 2007, 8(1): 1-9.

Choi Y, Deane C M. FREAD revisited: accurate loop structure prediction using a database search algorithm[J]. Proteins: Structure, Function, and Bioinformatics, 2010, 78(6): 1431-1440.

Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning[C]//Proceedings of the 25th international conference on Machine learning. 2008: 160-167.

Cristobal S, Zemla A, Fischer D, et al. A study of quality measures for protein threading models[J]. BMC bioinformatics, 2001, 2(1): 1-15.

de Bakker P I W, DePristo M A, Burke D F, et al. Ab initio construction of polypeptide fragments: Accuracy of loop decoy discrimination by an all-atom statistical potential and the AMBER force field with the Generalized Born solvation model[J]. Proteins: Structure, Function, and Bioinformatics, 2003, 51(1): 21-40.

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

Eisenhaber F, Persson B, Argos P. Protein structure prediction: recognition of primary, secondary, and tertiary structural features from amino acid sequence[J]. Critical reviews in biochemistry and molecular biology, 1995, 30(1): 1-94.

Ekeberg M, Lövkvist C, Lan Y, et al. Improved contact prediction in proteins: using pseudolikelihoods to infer Potts models[J]. Physical Review E, 2013, 87(1): 012707.

Elofsson A, Joo K, Keasar C, et al. Methods for estimation of model accuracy in CASP12[J]. Proteins: Structure, Function, and Bioinformatics, 2018, 86: 361-373.

Fiser A, Do R K G. Modeling of loops in protein structures[J]. Protein science, 2000, 9(9): 1753-1773.

Fiser A. Template-based protein structure modeling[J]. Computational biology, 2010: 73-94.

Garcia-Garcia A, Orts-Escolano S, Oprea S, et al. A review on deep learning techniques applied to semantic segmentation[J]. arXiv preprint arXiv:1704.06857, 2017.

Ginalski K. Comparative modeling for protein structure prediction[J]. Current opinion in structural biology, 2006, 16(2): 172-177.

He B, Mortuza S M, Wang Y, et al. NeBcon: protein contact map prediction using neural network training coupled with naïve Bayes classifiers[J]. Bioinformatics, 2017, 33(15): 2296-2306.
He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
Hildebrand P W, Goede A, Bauer R A, et al. SuperLooper—a prediction server for the modeling of loops in globular and membrane proteins[J]. Nucleic acids research, 2009, 37(suppl_2): W571-W574.
Hurtado D M, Uziela K, Elofsson A. Deep transfer learning in the assessment of the quality of protein models[J]. arXiv preprint arXiv:1804.06281, 2018.
Janin J, Henrick K, Moult J, et al. CAPRI: a critical assessment of predicted interactions[J]. Proteins: Structure, Function, and Bioinformatics, 2003, 52(1): 2-9.
Johnson M S, Srinivasan N, Sowdhamini R, et al. Knowledge-based protein modeling[J]. Critical reviews in biochemistry and molecular biology, 1994, 29(1): 1-68.
Jones D T, Singh T, Kosciolek T, et al. MetaPSICOV: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins[J]. Bioinformatics, 2015, 31(7): 999-1006.
Jumper J, Evans R, Pritzel A, et al. Highly accurate protein structure prediction with AlphaFold[J]. Nature, 2021, 596(7873): 583-589.
Karasikov M, Pagès G, Grudinin S. Smooth orientation-dependent scoring function for coarse-grained protein quality assessment[J]. Bioinformatics, 2019, 35(16): 2801-2808.
Kamisetty H, Ovchinnikov S, Baker D. Assessing the utility of coevolution-based residue–residue contact predictions in a sequence-and structure-rich era[J]. Proceedings of the National Academy of Sciences, 2013, 110(39): 15674-15679.
Kihara D, Chen H, Yang Y D. Quality assessment of protein structure models[J]. Current Protein and Peptide Science, 2009, 10(3): 216-228.
Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25: 1097-1105.
Kryshtafovych A, Schwede T, Topf M, et al. Critical assessment of methods of protein structure prediction (CASP)—Round XIII[J]. Proteins: Structure, Function, and Bioinformatics, 2019, 87(12): 1011-1020.
Kuhlman B, Bradley P. Advances in protein structure prediction and design[J]. Nature Reviews Molecular Cell Biology, 2019, 20(11): 681-697.
Levefelt C, Lundh D. A fold-recognition approach to loop modeling[J]. Journal of molecular modeling, 2006, 12(2): 125-139.
Li Z, Nguyen S P, Xu D, et al. Protein loop modeling using deep generative adversarial network[C]//2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2017: 1085-1091.
Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431-3440.
López-Blanco J R, Canosa-Valls A J, Li Y, et al. RCD+: Fast loop modeling server[J]. Nucleic acids research, 2016, 44(W1): W395-W400.

Manavalan B, Lee J, Lee J. Random forest-based protein model quality assessment (RFMQA) using structural features and potential energy terms[J]. PloS one, 2014, 9(9): e106542.

Michalsky E, Goede A, Preissner R. Loops In Proteins (LIP)—a comprehensive loop database for homology modelling[J]. Protein engineering, 2003, 16(12): 979-985.

Mikolov T, Karafiát M, Burget L, et al. Recurrent neural network based language model[C]//Interspeech. 2010, 2(3): 1045-1048.

Mirjalili V, Noyes K, Feig M. Physics-based protein structure refinement through multiple molecular dynamics trajectories and structure averaging[J]. Proteins: Structure, Function, and Bioinformatics, 2014, 82: 196-207.

Monastyrskyy B, Fidelis K, Tramontano A, et al. Evaluation of residue–residue contact predictions in CASP9[J]. Proteins: Structure, Function, and Bioinformatics, 2011, 79(S10): 119-125.

Monastyrskyy B, d'Andrea D, Fidelis K, et al. Evaluation of residue–residue contact prediction in CASP10[J]. Proteins: Structure, Function, and Bioinformatics, 2014, 82: 138-153.

Monastyrskyy B, Fidelis K, Tramontano A, et al. Evaluation of residue–residue contact predictions in CASP9[J]. Proteins: Structure, Function, and Bioinformatics, 2011, 79(S10): 119-125.

Moult J, Hubbard T, Fidelis K, et al. Critical assessment of methods of protein structure prediction (CASP): round III[J]. Proteins, 1999, 37(S3): 2-6.

Moult J, Fidelis K, Kryshtafovych A, et al. Critical assessment of methods of protein structure prediction (CASP)—round x[J]. Proteins: Structure, Function, and Bioinformatics, 2014, 82: 1-6.

Park H, Seok C. Refinement of unreliable local regions in template-based protein models[J]. Proteins: Structure, Function, and Bioinformatics, 2012, 80(8): 1974-1986.

Park H, Lee G R, Heo L, et al. Protein loop modeling using a new hybrid energy function and its application to modeling in inaccurate structural environments[J]. PloS one, 2014, 9(11): e113811.

Pereira J, Simpkin A J, Hartmann M D, et al. High-accuracy protein structure prediction in CASP14[J]. Proteins: Structure, Function, and Bioinformatics, 2021, 89(12): 1687-1699.

Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106

Ray A, Lindahl E, Wallner B. Improved model quality assessment using ProQ2[J]. BMC bioinformatics, 2012, 13(1): 1-12.

Rykunov D, Fiser A. Effects of amino acid composition, finite size of proteins, and sparse statistics on distance-dependent statistical pair potentials[J]. Proteins: Structure, Function, and Bioinformatics, 2007, 67(3): 559-568.

Seemayer S, Gruber M, Söding J. CCMpred—fast and precise prediction of protein residue–residue contacts from correlated mutations[J]. Bioinformatics, 2014, 30(21): 3128-3130.

Senior A W, Evans R, Jumper J, et al. Improved protein structure prediction using potentials from deep learning[J]. Nature, 2020, 577(7792): 706-710.

Šikić M, Tomić S, Vlahoviček K. Prediction of protein–protein interaction sites in sequences and 3D structures by random forests[J]. PLoS computational biology, 2009, 5(1): e1000278.

Skwark M J, Abdel-Rehim A, Elofsson A. PconsC: combination of direct information methods and alignments improves contact prediction[J]. Bioinformatics, 2013, 29(14): 1815-1816.

Stahl K, Schneider M, Brock O. EPSILON-CP: using deep learning to combine information from multiple sources for protein contact prediction[J]. BMC bioinformatics, 2017, 18(1): 1-11.

Stein A, Kortemme T. Improvements to robotics-inspired conformational sampling in rosetta[J]. PloS one, 2013, 8(5): e63090.

Stumpff-Kane A W, Maksimiak K, Lee M S, et al. Sampling of near-native protein conformations during protein structure refinement using a coarse-grained model, normal modes, and molecular dynamics simulations[J]. Proteins: Structure, Function, and Bioinformatics, 2008, 70(4): 1345-1356.

Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.

Tegge A N, Wang Z, Eickholt J, et al. NNcon: improved protein contact map prediction using 2D-recursive neural networks[J]. Nucleic acids research, 2009, 37(suppl_2): W515-W518.

Uziela K, Menendez Hurtado D, Shu N, et al. ProQ3D: improved model quality assessments using deep learning[J]. Bioinformatics, 2017, 33(10): 1578-1580.

Uziela K, Shu N, Wallner B, et al. ProQ3: Improved model quality assessments using Rosetta energy terms[J]. Scientific reports, 2016, 6(1): 1-10.

Vendruscolo M, Kussell E, Domany E. Recovery of protein structure from contact maps[J]. Folding and Design, 1997, 2(5): 295-306.

Wallner B, Elofsson A. Can correct protein models be identified?[J]. Protein science, 2003, 12(5): 1073-1086.

Wang J, Li Z, Shang Y. New deep neural networks for protein model evaluation[C]//2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2017: 309-313.

Wang L, Yang M Q, Yang J Y. Prediction of DNA-binding residues from protein sequence information using random forests[J]. Bmc Genomics, 2009, 10(1): 1-9.

Wang W, Li Z, Wang J, et al. PSICA: a fast and accurate web service for protein model quality analysis[J]. Nucleic acids research, 2019, 47(W1): W443-W450.

Wang S, Peng J, Ma J, et al. Protein secondary structure prediction using deep convolutional neural fields[J]. Scientific reports, 2016, 6(1): 1-11.

Wang S, Sun S, Li Z, et al. Accurate de novo prediction of protein contact map by ultra-deep learning model[J]. PLoS computational biology, 2017, 13(1): e1005324.

Wang W, Wang J, Xu D, et al. Two new heuristic methods for protein model quality assessment[J]. IEEE/ACM transactions on computational biology and bioinformatics, 2018, 17(4): 1430-1439.

Wang Z, Xu J. Predicting protein contact map using evolutionary and physical constraints by integer programming[J]. Bioinformatics, 2013, 29(13): i266-i273.

Weigt M, White R A, Szurmant H, et al. Identification of direct residue contacts in protein–protein interaction by message passing[J]. Proceedings of the National Academy of Sciences, 2009, 106(1): 67-72.

Wu F Y. The potts model[J]. Reviews of modern physics, 1982, 54(1): 235.

Wu S, Zhang Y. A comprehensive assessment of sequence-based and template-based methods for protein contact prediction[J]. Bioinformatics, 2008, 24(7): 924-931.

Xiang Z, Soto C S, Honig B. Evaluating conformational free energies: the colony energy and its application to the problem of loop prediction[J]. Proceedings of the National Academy of Sciences, 2002, 99(11): 7432-7437.

Zemla A. LGA: a method for finding 3D similarities in protein structures[J]. Nucleic acids research, 2003, 31(13): 3370-3374.

Zemla A, Venclovas Č, Moult J, et al. Processing and evaluation of predictions in CASP4[J]. 2001.

Zhang J, Zhang Y. A novel side-chain orientation dependent potential derived from random-walk reference state for protein fold selection and structure prediction[J]. PloS one, 2010, 5(10): e15386.

# VITA

Junlin Wang is currently a Ph.D. candidate in Distributed and Intelligent Computing Lab at the Electrical Engineering and Computer Science (EECS) Department, University of Missouri, under the supervision of Professor Yi Shang.

He obtained his M.S. degree in Computer Science at University of Missouri and obtained his B.S. degree in Software Engineering at Beijing Jiaotong University.

His research focuses on developing novel machine learning and deep learning methods for computational bioinformatic problems. The techniques that he actively uses in his research involve deep learning, machine learning, artificial intelligence, optimization methods, data mining, data structure and algorithms, and statistical analysis.