University of Reading

Improvements to methods for the quality assessment of three-dimensional models of proteins

A thesis submitted for the degree of
Doctor of Philosophy

School of Biological Sciences
University of Reading

Ali Hassan Ahmed Maghrabi

September 2019

Declaration

I confirm that this is my own work and to the best of my knowledge, does not breach copyright law, and has not been taken from other sources except where such work has been cited and acknowledged within the text.

Ali Hassan Ahmed Maghrabi: 

Date: 19/5/2020

Abstract

After water, proteins are the most abundant substances in the human body, forming around 80% of its dry mass. Understanding protein function is beneficial for life needs, such as finding medicines, producing healthy foods and combating infectious diseases. Each protein molecule has its own unique sequence which is comprised of linear chains of amino acids. These amino acid chains fold to form tertiary structures, which confer the protein's function. It is important that we can characterise protein structures in order to better understand their functions. Several experimental methods such as X-ray Crystallography and Nuclear Magnetic Resonance have been applied to solve protein structures. However, such methods are costly and time consuming, and some proteins are also problematic or impossible to solve using these methods. Consequently, the process of growing protein structure data is relatively slow in comparison to the speed of sequencing genomes and their encoded proteins, which has kept increasing especially after breakthroughs in the genetic sequencing technology. As a result, a gap has grown between known protein sequences and their resolved structures and it has been necessary to find other solutions. Computational methods for predicting the structures of proteins directly from own sequences have become fast and effective alternatives to experimental methods. Over the past 20 years there has been an emergence of different types of protein structure predicting methods, the most accurate type being the comparative modelling method, which consists of a number of steps including: template recognition, alignment, quality assessment, and ending with refinement. Each of these steps contribute to successful modelling pipelines, but perhaps the most critical step for the wider acceptance of 3D models of proteins has been the quality assessment step, where the predicted models are evaluated in terms of their likely accuracy, prior to the availability of an experimental structure. Numerous approaches to the quality estimation problem have been developed over the years including the use of statistical potentials, stereochemistry checks and machine learning techniques. Such methods have traditionally been referred to as Model Quality Assessment Programs (MQAPs). One of the leading MQAPs has been the ModFOLD method which has been developed by our group. Since its inception, ModFOLD has been continuously improved, going through many upgrades until its latest version, ModFOLD7. This study was conducted during a major development cycle, beginning with the benchmarking of ModFOLD6, the most powerful MQAP method compared to its other competitors at that time. The study starts with the investigation of the integration of ten MQAP scoring methods in an attempt to enhance performance. The study also explores the implementation of deep neural networks on the MQAP

method's pipeline, and how this technique can be used to improve the MQAP scoring approach. In the later stage of our research, we managed to improve our method significantly leading to the latest upgrade, ModFOLD7. During this project, we also participated in a number of independent blind experiments and competitions to verify our improvements. We also undertook several collaborations in order to apply our methods in practical contexts. The overall results have shown incremental but significant improvements in ModFOLD performance during this study, with an approximate 5% improvement over previous versions. However, there are still plenty of room for ModFOLD to improve further and a number of suggestions for further developments will be addressed throughout this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ANNs	Artificial Neural Networks
ASE	Accuracy of Self-Estimates
AUC	Area Under the Curve
BNNs	Biological Neural Networks
CAD(AA)	Contact Area Difference (All Atom)
cAMP	Cyclic Adenosine Monophosphate
CASP	Critical Assessment of Structure Prediction
CDA	Contact Distance Agreement
Cryo-EM	Cryogenic Electron Microscope
DANNs	Deep Artificial Neural Neworks
DBA	Disorder B-factor Agreement
DOIs	Digital Object Identifiers
DSSP	Dictionary of Secondary Structures of Proteins
EMA	Estimate of Model Accuracy
FM	Free Modelling
GDT	Global Distance Test
GDT-HA	Global Distance Test-High Accuracy
GDT-TS	Global Distance Test-Total Score
GPGPUs	General-Purpose Graphics Processing Units
IEEE	Institute of Electrical and Electronic Engineer
LDDT	Local Distance Difference Test
MCC	Matthews correlation coefficient
MLP	Multilayer Perceptron
MNIST	Modified National Institute of Standards and Technology
MSA	Multiple Sequence Alignment
MSE	Mean Squared Error
MTM	Multiple Template Modelling
MQAPs	Model Quality Assessment Programs
NMR	Nuclear Magnetic Resonance
NN	Neural Network
OMS	Optimal Mean Score
PDB	Protein Data Bank
PKA	Protein Kinase A
PR	Precision and Recall
PSSM	Position-Specific Scoring Matrix
QA	Quality Assessment
QE	Quality Estimation
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic
RSNNS	R Stuttgart Neural Network Simulator
SCE	Sigmoid Cross-Entropy
SG	SphereGrinder
SNNS	Stuttgart Neural Network Simulator
SSA	Secondary Structure Agreement

SSEA	Secondary Structure Element Alignment
StdErr	Standard Error
TBM	Template-Based Modelling
TM-score	Template Modeling Score
VASP	Vasodilator-Stimulated Phospho-Protein
WPGMC	Weighted Pair Group Method Centroid

Acknowledgment

First, and most of all, I would like to acknowledge the source of everything, and the cause behind all reasons. The one which without nothing would have been existed. All praises be to the almighty God.

I would like to express the deepest appreciation to Dr Liam McGuffin, this amazing person who has the attitude and the substance of a genius. He continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without Dr McGuffin's guidance and persistent help, this thesis would not have been possible.

I would like to extend my sincere gratitude to the exceptional Bioinformatics department in the School of Biological Sciences at the University of Reading. I would like to give a special thanks to Professor David Leake in sharing the effort and knowledge to make this research a reality. I also thank Dr Bajuna Salehe, Jennifer Atkins, Dr Naqib Shuid, Recep Adiyaman, John Nealon, Limcy Philomina, Ben Livesey, Filipe Jesus, and all the other colleagues who were continually cooperative and supportive to our study and research.

I would like also to give my deepest gratitude to the other colleagues from the other departments, thanks to Dr Khaled Sahli, Dr Gagan Flora and all the team in Professor Jon Gibbin's lab for the great collaborative study. I also thank the other teams from different international universities and organisations for the great collaboration during my Ph.D. study.

Finally, I would like to give special thanks to my happy family who has provided me a continuous spiritual and material support. Never-ending love from me to my beloved mother for her support. It is a great gift from God to be her son. Thanks are also given for my father the one who always had confidence in me and knew that I can do it, and to my brothers, sisters, relatives and friends who have given me cheerful and joyful days. I would like also to thank my government for supporting me morally and financially throughout my entire years of studies.

Chapter 1
Introduction

1.1. Proteins

The central dogma of molecular biology states that genetic information is stored in the DNA and transcribed into RNA, which in turn is translated into the most versatile macromolecules that govern the very basis of life - proteins. Proteins comprise the second largest percentage of material (after water) in a cell and they play a key role in virtually every cellular process within living organisms (Chauhan and Varma, 2013). Knowing the function of proteins is critical in many fields within the life sciences - from the development of better drugs, improvement of crop yields, and even the development to synthetic biofuels. In a single experiment using *E. coli*, for instance, 2300 proteins have been identified to be representing approximately 88% of the estimated expressed proteome of *E. coli* (Soufi et al., 2015). Determining and characterising every protein molecule of that bacterial cell can lead to the knowledge of how this prokaryotic cell is regulated. Normal functioning proteins usually have one or more continuous polymer chains comprising various combinations of multiple different amino acid residues. These residues interact with one another forming covalent, hydrogen and ionic bonds and are subject to van der Waals interactions and hydrophobic forces, which give rise to the specific 3D shapes and functions of proteins.

1.2. Amino acids

The tremendous abundance and variety of proteins with their enormous functional complexity are all achieved by their polymeric nature, which emerges from a limited alphabet of building blocks with varying properties - the amino acids. The discovery of the first few of the amino acids occurred the early 19th century, when Vauquelin and Robiquet isolated a compound in Asparagus (subsequently named Asparagine) to find the first protein amino acid (Vauquelin and Robiquet, 1806). The 20 common (or called the standard) amino acids then were discovered one after another until the last amino acid, Threonine, was found in 1935 by William Cumming Rose (Simoni et al., 2002).

The long unbranching chain of amino acids defines the primary structure of proteins, they are linked to one another through a covalent peptide bond. Each amino acid consists of a central carbon atom attached to an amino group (-NH2), a carboxyl group (-COOH) and a side chain group (-R). There are 20 commonly known amino acids which make protein chains each with different side chains. Each are specified by codons in the universal genetic code (Table 1.1). The different side

chains give each amino acid its own chemical structure and properties, allowing us to classify them differently, for example by polarity and charge (Figure 1.1). The 20 amino acids are typically labelled with a one-letter as well as three-letter abbreviations in order to simplify the way of recording, processing and understanding protein sequences. The one letter amino acid code can also help us to more easily identify any specific mutations or binding sites occurred in a sequence (Lodish et al. 2000), via e.g. sequence alignments and motif searching. Moreover, there are two other "non-standard" amino acids (Selenocysteine and Pyrrolysine) which do not have a dedicated codon, but are added in place of a stop codon when a specific sequence is present, UGA codon and SECIS element for Selenocysteine (Bo¨ck et al., 1991), UAG PYLIS downstream sequence for Pyrrolysine (Théobald-Dietrich et al., 2005). There are also other amino acids which are not naturally encoded or found in the genetic code of any organism but rather they occur in nature or be synthesised in the laboratory, these types of amino acids are termed as "non-proteinogenic" (Filip and Iancu, 2018).

Amino acid	Abbreviation		Side chain Polarity	Side chain charge	Class
	Three-letter	One-letter			
Aspartic acid ($C_4H_7NO_4$)	Asp	D	Polar	Negative	Acidic
Glutamic acid ($C_5H_9NO_4$)	Glu	E	Polar	Negative	Acidic
Arginine ($C_6H_{14}N_4O_2$)	Arg	R	Polar	Positive	Basic
Lysine ($C_6H_{14}N_2O_2$)	Lys	K	Polar	Positive	Basic
Histidine ($C_6H_9N_3O_2$)	His	H	Polar	Positive	Basic
Asparagine ($C_4H_8N_2O_3$)	Asn	N	Uncharged polar	Neutral	Acidic
Glutamine ($C_5H_{10}N_2O_3$)	Gln	Q	Uncharged polar	Neutral	Acidic
Serine ($C_3H_7NO_3$)	Ser	S	Uncharged polar	Neutral	Hydroxyl
Threonine ($C_4H_9NO_3$)	Thr	T	Uncharged polar	Neutral	Hydroxyl
Tyrosine ($C_9H_{11}NO_3$)	Tyr	Y	Uncharged polar	Neutral	Aromatic
Alanine ($C_3H_7NO_2$)	Ala	A	Nonpolar	Neutral	Aliphatic
Cysteine ($C_3H_7NO_2S$)	Cys	C	Nonpolar	Neutral	Hydroxyl
Glycine ($C_2H_5NO_2$)	Gly	G	Nonpolar	Neutral	Aliphatic
Isoleucine ($C_6H_{13}NO_2$)	Ile	I	Nonpolar	Neutral	Aliphatic
Leucine ($C_6H_{13}NO_2$)	Leu	L	Nonpolar	Neutral	Aliphatic
Methionine ($C_5H_{11}NO_2S$)	Met	M	Nonpolar	Neutral	Hydroxyl
Phenylalanine($C_9H_{11}NO_2$)	Phe	F	Nonpolar	Neutral	Aromatic
Proline ($C_5H_9NO_2$)	Pro	P	Nonpolar	Neutral	Cyclic
Tryptophan ($C_{11}H_{12}N_2O_2$)	Trp	W	Nonpolar	Neutral	Aromatic
Valine ($C_5H_{11}NO_2$)	Val	V	Nonpolar	Neutral	Aliphatic
Alanine ($C_3H_7NO_2$)	Ala	A	Nonpolar	Neutral	Aliphatic

Table 1.1. **List of the 20 amino acids found in proteins.** The list includes the names formula, three-letter abbreviation, One-letter abbreviation, side chain polarity, side chain charge and class for the. Adapted from Williamson, (2011).

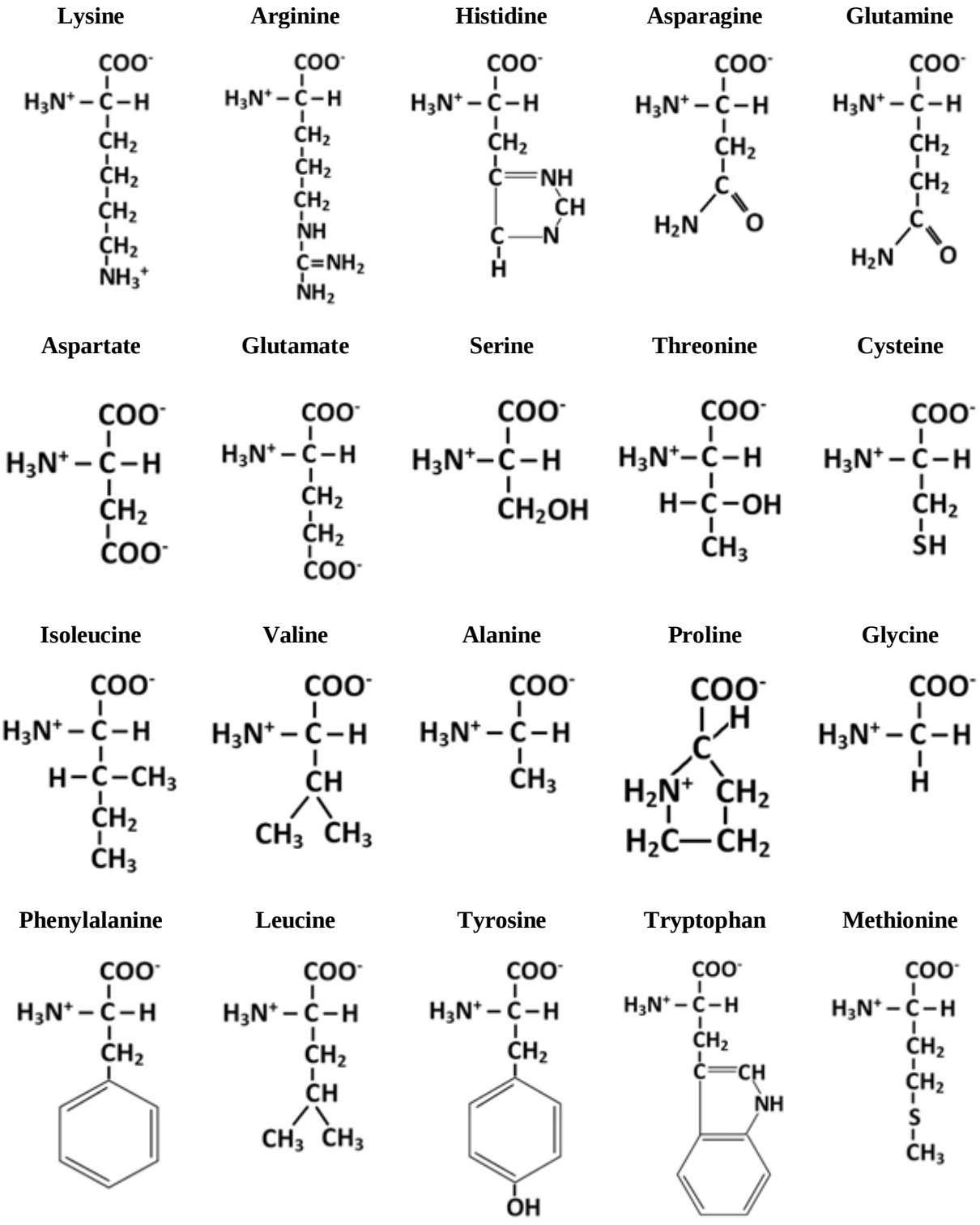

Figure 1.1. The structure of the 20 amino acids found in proteins. They are encoded by the universal genetic code for the construction of protein sequences. Each amino acid is represented with its own side chain. Adapted from Williamson (2011).

1.3. From Primary to Quaternary Structure

Proteins exist in a plethora of different shapes and sizes which confers a vast range of biochemical and cellular functions, as well as the different phenotypes they produce in a cell or an organism. The structural diversity arises from the genetic code, specified in the DNA sequences, which in turn specifies the linear chains of various lengths made up from different amino acid combinations. The linear chains of amino acids subsequently fold into a diverse range of specific three-dimensional structures which allows them to perform an incredible array of functions (Petsko and Ringe, 2008). Four principal levels of protein structure have been characterised in order to help us to better understand the diversity of protein structures that results from the sequence complexity (Figure 1.2).

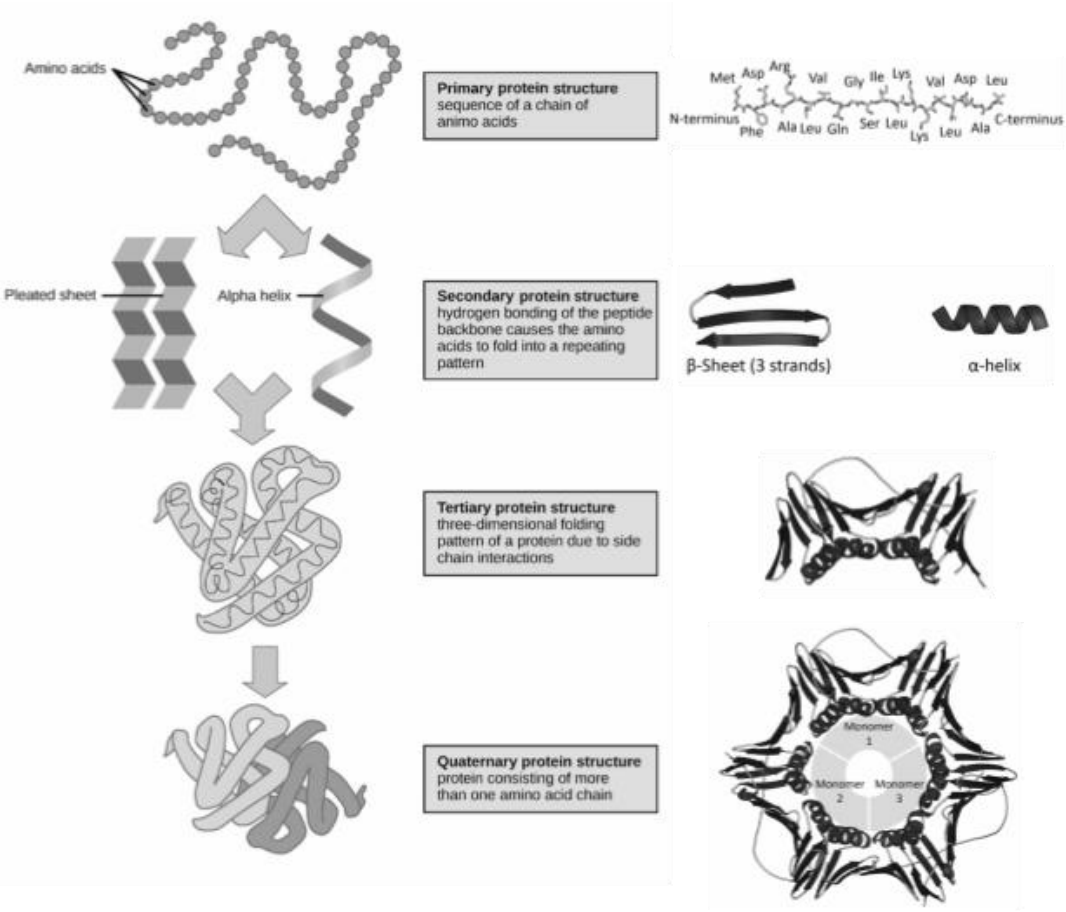

Figure 1.2. Summary of the protein structure. In the left side, a simplified cartoon representation of the four levels of protein structures, and in the right side, a 3D computational cartoon representing the four degrees of the protein structures using the example of PCNA (Mudavath and Pittu, 2013) (PDB: 1AXC).

1.3.1. Primary structure

The primary structure is the linear sequence chain of polymer that is built from the 20 amino acids described in Section 1.2. The chain is formed by a condensation (dehydration) reaction between the carbonyl group (C=O) of one amino acid and the amino group (H-N) of the next leading to the formation of a covalent (peptide) bond. The repeated process leads to the formation of an unbranched polypeptide chain with two ends: the carboxyl terminus (C-terminus) and the amino terminus (N-terminus) based on the nature of the free group on each end. If the chain is short (less than ~40 amino acids) we generally refer to the chain as peptides rather that a protein. When writing out the sequence of a protein, we begin with the first encoded amino acid from the N-terminus and then continue until the last amino acid, which will be the one at the C-terminus end (Petsko and Ringe 2008).

1.3.2. Secondary structure

The secondary structure level describes the locally ordered structure which is stabilised by hydrogen bonding. There are two major types of secondary structures observed in proteins. The first major type is called the alpha (α) helix, this structure resembles a coiled spring, and it is stabilised by local hydrogen bonding located between the NH and CO groups in the polypeptide chain. The second major type is called the beta (β) strand. Extended beta strands come together to form beta-pleated sheets via more distant hydrogen bonding between amino acids linking the folded chain so that strands lie adjacent to one another. The beta strands in a sheet can either be described as being parallel, where the strands are running in the same direction from N-terminus to C-terminus, or the pairing strands can run in opposite directions making them an anti-parallel β-sheet. The elements of the regular secondary structure are usually connected by loop regions, also referred to as coils.

A standardised vocabulary of secondary structure types was published by Kabsch and Sander in their DSSP program (Kabsch and Sander, 1983). The program processes the atomic coordinate data contained within PDB files and assigns secondary structures states to each residue in the protein based on hydrogen bonding patterns. DSSP defines secondary structure states such as the π-helix (’I’), three-ten helix (’G’), turn (’T’), β-bridge (’B’), bend (’S’) and coil (’C’). However, commonly residues are grouped into a simplified three-state scheme - either helical (H), extended

strand (E) and coil (C) states – for processing by structure prediction and classification programs (Jones, 1999) (Zhang and Skolnick, 2005).

1.3.3. Tertiary structure

When multiple secondary structural elements start to get condensed, this level of structure arises. The tertiary structure refers to the overall three-dimensional shape of a protein, once all the secondary structure elements have folded together among each other. In this stage, the protein backbone topology or mutual orientation of secondary structures are specified, and the full 3D arrangement of all atoms is created through interactions between polar, nonpolar, acidic, and basic R group within the polypeptide chain. The folding in this level are driven by several forces such as hydrophobic effects, van der Waals forces, ionic interactions and hydrogen bonds (Lodish et al., 2000). Once folded, the hydrophobic R groups of nonpolar amino acids will mostly lie in the interior of the structure. In contrast, the hydrophilic R groups will mostly lie on the outside. Cysteine side chains will form disulphide linkages in the presence of oxygen which is the only covalent bond forming during protein folding. Such interactions are the main causes of the final 3D shape of a protein. Without these types of bonding and forces, the protein three-dimensional shape will be lost, and the protein will no longer has its function (Baldwin, 2007).

1.3.4. Quaternary structure

At the highest level of organisation, the quaternary structure combines two or more folded chains into a multi-subunit structure. The multiple polypeptide chains arrange into stable and semi-stable complexes based on the hydrophobic effect or electrostatic interactions between residues. The hydrophobic interaction among nonpolar side chains at the contact regions of the subunits is the major force which stabilise the quaternary structure. Whereas, the interactions between side chains of the subunits including electrostatic interactions such as hydrogen and disulphide bonds are functioning as additional stabilisers. Therefore, Hydrogen bonding, van der Waals interactions, ionic bonding and disulphide bonding are the factors that keep the quaternary structure of proteins held together. An example of a protein with quaternary structure is haemoglobin, an oxygen transport protein which is translated as a tetrameric (four unit) protein consisting of two α and two β subunits. Other examples include the DNA, polymerase, and ion channels as well. If there are

multiple polypeptide chains with the same sequence, then the protein complex is referred to as a homo-oligomer or homomer, whereas protein complexes having at least two different polypeptide sequences are referred to as hetero-oligomers or heteromers. In the living cell, oligomers can be found abundantly, and they serve in many different parts of the living body with a multitude of functions (Berg et al., 2002).

1.4. Why determine the structure of proteins?

The motivation behind protein structure determination is based on the assumption that the function of a protein is representative in its own morphology. When proteins fold into specific structural conformations they also perform a specific biological function based on that fold. By determining the shape of a protein, we can understand its role within the body and how it works, enabling scientists to design new, effective cures for diseases more efficiently. Determining protein structure can also help in finding how proteins interact with each other as well as with other molecules such as ligands. There are several diseases which were caused because of misfolded proteins. Such these diseases include cystic fibrosis (Fraser-Pitt and O'Neil, 2015) and Alzheimer's (Ashraf et al., 2014).

Protein folding is also important in improving our understanding about the proteins themselves. Acquiring more knowledge about the proteins shape and the way they are operated through simulations and models can open new potentials within drug discovery in a far reduced costs association in experiments. The results of such research would ultimately improve the quality of life for millions of patients around the world.

Protein design is another field which can be tremendously benefited from protein structure determination. Many advances such as designing biodegradable enzymes which could help managing pollutants like plastic and oil for the purpose of breaking down waste in an environmentally friendly way can be boosted by knowing how to fold these protein enzymes (Halton, 2018).

The ongoing increase in structural information of proteins has been providing continual insights into their functions. However, revealing protein function from sequence is still an unsolved problem as they differ in their biochemical properties, structures, and interactions between one another. Having the knowledge that the protein has the ability to add a phosphate group to serine residues, for example, is not going to be sufficient for understanding the exact function of that

protein. Therefore, a number of molecular determinants have been considered crucial for elucidating function, such as: protein structure; genetic approaches; protein homology and cell location; protein-protein interactions, each of which has been carried out using a number of powerful techniques and methods in order to characterise the precise structure, which means, the precise function of a protein (Patthy, 2008) (Alberts et al., 2014).

1.5. Experimental methods

The precise 3D shape of individual proteins cannot be directly observed due to their relative size which lies below the limit of detection. To overcome this issue, a number of methods have been utilised in order to determine protein structures experimentally.

1.5.1 X-ray Crystallography

The most dominant experimental structure determination method is X-ray crystallography, a method used for observing protein molecules by dealing with them in the crystalline state. The technique exploits the properties of highly ordered crystals for the purpose of obtaining structural information of biological macromolecules at atomic resolution. Determining protein structure using the X-ray crystallography method are proceeded in 3 steps. Firstly, crystallising the protein target, and that is by expressing and purifying a large amount of proteins in order to grow their crystals. Secondly, collecting the diffraction patterns which are caused by the interaction between the electrons of the sample and the X-ray wave as described by Braggs law (Bragg William Henry, 1913). Thirdly, the output data is combined computationally with complementary chemical information to produce and refine a model of the arrangement of atoms within the crystal. The final refined model is called a crystal structure and is stored usually in the PDB (Berman et al., 2000).

X-ray crystallography is a very useful experimental method in determining protein structure. It can provide very detailed atomic information, showing every atom in a protein or nucleic acid along with atomic details of ligands, inhibitors, ions, and other molecules that are incorporated into the crystal. However, the crystallisation process is challenging and can impose limitations on the types of proteins that may be studied by the method. For instance, the method can determine the structures of rigid proteins excellently if only they formed nice, ordered crystals. Flexible proteins are far more difficult to study through the process of this method because crystallography relies on

having the exact same orientation of alignments for the protein molecules, similarly as a repeated pattern in wallpaper. Having flexible portions of protein will often be invisible in crystallographic electron density maps, since their electron density will be smeared over a large space (Guinier, 2013).

Recently, a new tool termed serial femtosecond crystallography has been revolutionasing the X-ray crystallography method. The idea behind it lies in making very short pulses of radiation which last only femtoseconds, these pulses are created by using a free electron X-ray laster (XFEL), and this radiation is extremely bright. Afterwards, a stream of tiny crystals which do not exceed the nano to the micrometers in size is passed through the beam, and each X-ray pulse produces a diffraction pattern from a crystal, often burning it up in the process. A full data set is compiled from as many as tens of thousands of these individual diffraction patterns. The method is very powerful because it allows scientists to study molecular processes that occur over very short time scales, such as the absorption of light by biological chromophores (Bergman et al., 2017).

1.5.2 Nuclear Magnetic Resonance

Nuclear Magnetic Resonance (NMR) is also used for proteins structure observation. This spectroscopic technique is preferable for small protein molecules, and it determines the structure in solution (Branden and Tooze, 1998). The method of this technique relies on the energy differences between spin states of nuclei with the uneven number of protons and neutrons in a magnetic field. State transitions between the low and high energy spin state are induced using a radio pulse, the magnetic field perceived by nuclei differ due to chemical shielding by electrons, these differences can be detected in the spectrum.

NMR has a major advantage that is missing in the X-ray crystallography, NMR can provide information on proteins in solution, as opposed to those locked in crystal or bound to a microscope grid. This feature makes NMR the premier method for studying the atomic structures of flexible proteins. The method can give indications that help spotting the flexible parts of the molecule by giving less or weaker signals in the experiment than the harder parts (Williams et al., 2016).

According to the PDB resource (www.pdb.org), over 11% of the deposited protein structures have been solved using NMR. In such a database, two types of coordinate entries for NMR structures can be found. The first type of entry includes the full ensemble from the structural determination,

with each structure designated as a separate model. The second is a minimised average structure. These files attempt to capture the average properties of the molecule based on the different observations in the ensemble. A list of restraints that were determined by the NMR experiment can also be found in PDB. These include a number of factors and other information such as hydrogen bonds and disulphide linkages, distances between hydrogen atoms that are close to one another, and restraints on the local conformation and stereochemistry of the chain (Wawer and Diehl, 2017).

1.5.3 Cryogenic Electron Microscopy

Another technique that has been used for structural analysis is the Cryogenic Electron Microscope (Cryo-EM), this technique is a branch of the commonly used technique 3D Electron Microscopy (3DEM). Unlike X-ray crystallography and NMR spectroscopy, Cryo-EM does not require 3D crystals or a soluble medium. Molecules can be observed directly in multiple conformations in their native environment (Murata and Wolf, 2018). The technique analyses the molecular structures through their dynamics, this analysis is proceeded using advanced image-processing algorithms. Providing these views show the molecule in myriad different orientations, a computational approach akin to that used for computerised axial tomography (CAT) scans in medicine will yield a 3D mass density map. The 3D map can then be interpreted by fitting an atomic model of the macromolecule into the map when there are enough single particles. This process is similar to the macromolecular crystallographers interpret their electron density maps, electron diffraction from 2D or 3D crystals or helical assemblies of biomolecules can also be used in a restricted number of cases in order to determine 3D structures with an electron microscope using an approach very similar to that of X-ray crystallography.

Cryo-EM is favoured by many structural biologists to solve proteins tertiary structure at cryogenic temperature which is below -180°C (Schmidt and Urlaub, 2017). The 3DEM techniques are gaining prominence in studying biological assemblies inside cryo-preserved cells and tissues using electron tomography. This method involves recording images at different tilt angles and averaging the images across multiple copies of the biological assembly in situ. The single particle 3DEM as well as the electron diffraction, both methods can yield structures with high molecular and atomic detail at resolution limits comparable to macromolecular crystallography, it can enable visualising amino acid sidechains, surface water molecules, and non-covalently bound ligands. Cryo-electron tomography can also provide structural information at slightly lower resolution such as protein

domains and secondary structural elements. In 2016, PDB reported that the number of 3DEM structures that was deposited in their bank exceeded those which were produced by NMR spectroscopy for the first time. However, likewise the previous protein structure observing methods, Cryo-EM has a number of limitations which makes it difficult in use. The main limitation of this technique is the need for a very low thickness of the samples. Most cells are structurally thick, that makes the method fails in excellently imaging the targeted structure as it depends on the thickness of the sample (Cabra and Samsó, 2015). Such a limitation is due to the very low signal to noise ratio that Cryo-EM has, and it leads to having a low contrast of the resulting images which makes it difficult to detect features of a given sample when viewing a few samples.

1.6. Resources for experimental protein structures

Protein structures which have been determined experimentally are deposited in a computational archive called the Protein Data Bank (PDB) (Berman et al., 2000). The PDB was established in the early seventies in order to gather a small but growing number of solved protein structures in one place and make it accessible to the scientific community (Bernstein et al., 1977). The atomic coordinates together with the information associated with the crystallised polymer of every protein experimental structure are deposited in the form of a PDB file (Figure 1.3). This information includes oligomeric state, references and experimental details, such as unit cell size, and refinement parameters. The stored structures are then labelled by assigning them with PDB identifiers, which are unique four-character codes for each structure, as well as DOIs.

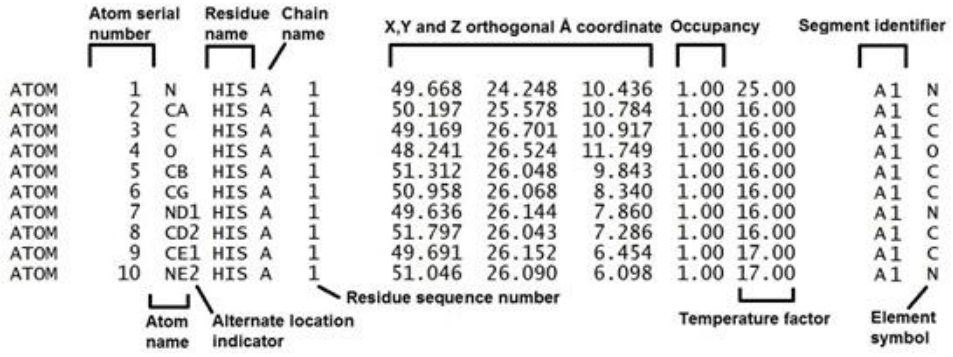

Figure 1.3. ATOM records within a PDB file format. The example was labelled for description purposes. Adapted from https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/tutorials/pdbintro.html.

1.7 Challenges facing experimental methods

Although such methods and experimental techniques help in determining higher resolution structures, they are still not reliable as not all proteins can be successfully determined under their conditions. A large number of proteins cannot be crystallised; particularly problematic are membrane bound proteins. Other types of proteins such as the small-sized insoluble ones are unable to be determined by NMR. In addition, the currently available experimental methods used to identify protein 3D structure like NMR and X-ray are relatively time-consuming and cost-ineffective (Zhu and Azar, 2015). The difficulties in these techniques have led the structural biological community to look for another substitute to rely on. Cryo-EM has become increasingly important due to the recent advances that was developed in its techniques. Such improvements in Cryo-EM have made it possible for structural biologists to determine the 3D structure of proteins to near-atomic resolution (Callaway, 2015). However, as with the other experimental methods, Cryo-EM still has many limitations. The Cryo-EM map is still limited in resolution and cannot be used directly for building models (Cheng, 2015).

1.8 The sequence-structure gap

Breakthroughs in genetic sequencing techniques have led to obtaining a huge amount of protein sequence information that has far outstripped the rate at which we are able to experimentally characterise each protein structure. Genomic sequences of many organisms have nowadays been easily and rapidly determined. This sequencing revolution is owing to the significant improvement in genome sequencing technologies and efforts. Over the recent decades, over 1 billion sequences from 420,000 formally described species have been deposited in the comprehensive public sequences data banks, GenBank databases (Sayers et al., 2019). More than 150 million sequences among them have been translated into protein amino acid sequences and stored in the UniprotKB/trEMBL database (UniProt Consortium, 2015). Contrarily, despite all the tremendous progress in the experimental structure determination techniques, the number of experimentally determined structures deposited in the PDB database increased slower as reports showed only 147,508 structures at the end of 2018 (Figure 1.4). Therefore, it was urgent for biologists to find a solution for this rapid widening in the sequence-structure gap.

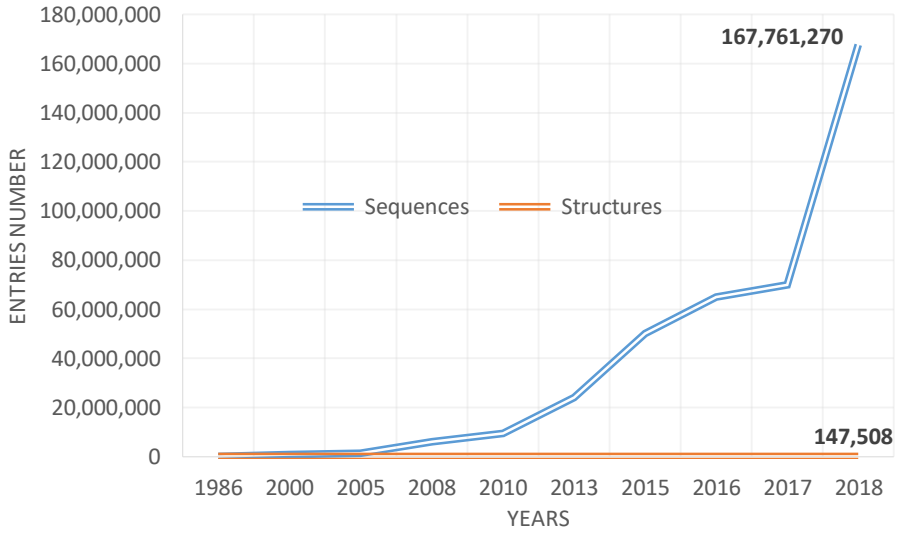

Figure 1.4. Line graph representing the sequence-structure gap. The number of entries in the trEMBL sequence database (UniProt Consortium, 2015) is growing exponentially, while the number of the protein structure entries is jammed making the protein structure gap between sequence and structures widening dramatically.

1.9 The intervention of Bioinformatics

To close this gap issue, the development of computational approaches was exploited to grow the field of what is called the protein structure prediction. Numerous studies have been focusing on determining the protein 3D structure directly from the sequence data using a wide variety of computational methods.

1.9.1 Sequence-structure relationship

A grand challenge that molecular biologists face in determining the protein three-dimensional structure is unravelling the relationship between the amino acid sequence and the protein 3D conformation. The work of Anfinsen in 1973 revealed that the major determinant of the 3D structure of a protein was its own primary sequence (Anfinsen, 1973). Although the primary sequence of proteins is subjected to changes as a direct consequence of evolution, protein structure has proven to be astonishingly robust towards mutations. Such a robustness is due to the proteins ability in adopting a limited ensemble of native conformations since those conformers have lower energy than unfolded and mis-folded states (Taverna and Goldstein, 2002) (Tokuriki et al., 2007).

This process is achieved by a distributed, internal network of cooperative interactions such as the hydrophobic, polar and covalent (Shakhnovich et al., 2005). Stability in structural similarity can even be found with the distantly related proteins.

Later and after a decade of achieving a huge amount of protein structural information, homologous proteins which share detectable sequence similarity was seen to have similar 3D structures, and their structural diversity is increasing with evolutionary distance as well. Such a finding was shown by Chathia and Lesk in their seminal paper, "The Relation between the Divergence of Sequence and Structure in Proteins" (Chothia and Lesk, 1986) (Figure 1.5).

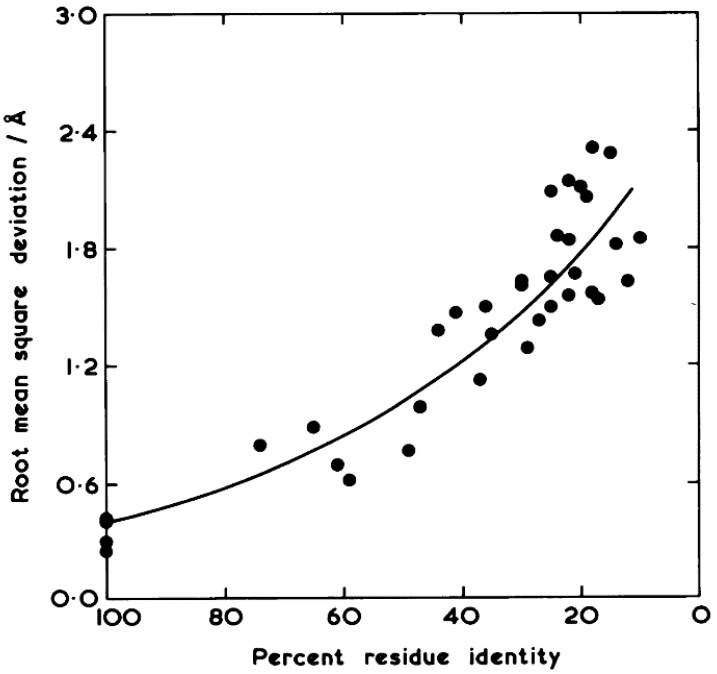

Figure 1.5. Scatter plot representing the relationship between sequence identity and structural similarity of core residues. The fitted curve shows that similar sequences imply similar structures. Adapted from Chothia and Lesk, (1986).

1.9.2. Sequence alignments

The fact that sequence similarity implies structural similarity has been validated via many bioinformatics and computational biology studies and methodologies, which compare the relationships between protein sequences (Eddy, 2011). Sequence similarities are typically found through comparing protein sequences using sequence alignment techniques. Computational approaches to sequence alignment generally fall into two algorithmic categories. The first category

involves the global alignment methods and these approaches force the alignment to span the entire length of all query sequences. The second category involves the local alignment methods, and these approaches identify regions of similarity within long sequences that are often widely divergent overall. Both types of algorithms are within a complexity of O(nm), where n and m represent the lengths of the first and second sequences respectively. Global alignments are useful when the sequences in the query set are similar and of roughly equal size. An example of a global alignment approach is the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970). However, local alignments approach sometimes are preferable when the aligned sequences are not directly related to produce high-quality global alignments. A general example of local alignment method is the Smith-Waterman algorithm (Smith and Waterman, 1981) (Polyanovsky et al., 2011).

1.9.2.1 Pairwise sequence alignments

Sequences are usually aligned in pairs to find the best-matching piecewise alignments of only two query sequences at a time. The pairwise alignment approach is efficient and is often used for methods which do not require extreme precision (such as searching a database for sequences with high similarity to a query). There are three common techniques that produce pairwise alignments (Mount, 2004): (1) Dot-matrix, this method implicitly produces a family of alignments for individual sequence regions. The technique is qualitative and conceptually simple, though time-consuming to analyse on a large scale. (2) Dynamic programming, this method is applied to produce global alignments via Needleman-Wunsch algorithm, and local alignments via the Smith-Waterman algorithm. The approach is useful for finding an optimal alignment given a particular scoring function, though identifying a good scoring function is often an empirical rather than a theoretical matter. (3) Word, also known as k-tuple, and this method is heuristic, which does not guarantee finding an optimal alignment solution. However, Word technique is more efficient than dynamic programming, especially in large-scale database searches where large proportion of the candidate sequences will have essentially no significant match with the query sequence.

The efficiency of word or k-tuple based techniques, led to the development of a number of widely and used programs such as FASTA (Lipman and Pearson, 1985) and BLAST (Altschul et al., 1990). Since its introduction until today, BLAST is still one of the most widely used programs for carrying out efficient sequence database searches. Its heuristic algorithm makes the program much faster than other approaches. Prior to FASTA and BLAST, the procedure of searching for sequence

alignments used to be by using the full alignment algorithms such as Smith-Waterman, which were time consuming.

1.9.2.2 Multiple sequence alignments

More than two sequences can also be incorporated at a time using multiple sequence alignment (MSA) approach which is an extension of pairwise alignment. MSA methods are often used to identify conserved sequences regions across a group of sequences hypothesised to be evolutionarily related (Rosenberg, 2009). Such approaches are useful in conjunction with structural and mechanistic information to locate, for instance, the catalytic active sites of enzymes. They can also be used to aid in establishing evolutionary relationships by constructing phylogenetic trees. Although MSA approaches are computationally difficult to produce and sometimes they lead to NP-complete combinatorial optimisation problems, the utility of these alignments has been able to develop a variety of methods which helped in advancing bioinformatics. Such these methods are Clustal Omega (Sievers and Higgins, 2014), Kalign (Lassmann and Sonnhammer, 2005), MUSCLE (Edgar, 2004)… etc), these methods are suitable for aligning three or more sequences (Wang and Jiang, 1994) (Elias, 2006).

Sequence alignments are usually stored in a format based on several specific alignment programs or implementations. Many of these programs provide web-based tools that allow storing sequence alignments using a limited number of input and output formats, such as FASTA format (Goldstein et al., 2014) and GenBank format.

1.9.2.3. From sequence alignments to structure modelling

The fact that evolutionarily related proteins have similar structures has encouraged researchers to develop methods for predicting the structure of proteins from their sequences (Kaczanowski and Zielenkiewicz, 2010). One way of modelling a protein structure is by aligning the sequence to those of already experimentally observed protein structures and then using those structures as templates in order to map the 3D coordinates of each aligned residue. This procedure has been termed homology modelling or comparative modelling (Martí-Renom et al., 2000). However, sometimes structurally homologous proteins can have a very low sequence identity, and in these cases homology modelling methods fail to identify a suitable template structures or produce poor

alignments. This issue led to another way of determining protein structure called Threading or Fold recognition (Rost et al., 1997). This modelling method does not use the homologous proteins with known structures, but rather uses statistical knowledge of the relationship between the structures which have been deposited in the PDB database and the targeted sequence. In recent years fold recognition and homology modelling techniques have somewhat merged, with the ability to detect ever more distant evolutionarily relationships using profile-profile searching methods and HMM-HMM methods, such as the popular HHpred method (Hildebrand et al., 2009). The general concept of modelling based on existing structures is now classified as Template-Based Modelling (TBM), and the success of such methods relies on the availability and accurate detection of suitable templates.

As the amount of detectable similarity between target protein and template structures decreases, the accuracy of template-based techniques will start to be insufficient and such methods become unreliable. In this case, another structure prediction technique, traditionally called de novo or ab initio protein structure prediction is the only remaining option. The technique is based on predicting the structure of proteins without the need of a template and is therefore known as template Free Modelling or FM (Jothi, 2012). FM methods are not nearly as accurate as TBM methods when templates are available (Moult et al., 2005). However, the concept of such techniques is somewhat simpler comprising of only two elements: firstly an algorithm to search the space of possible protein configurations for cost function minimisation and secondly various restraints, which are the composition of the cost function itself, being either derived from physical laws and structural features predicted by machine learning or other types of statistical systems (Larrañaga et al., 2006). FM techniques have been incrementally improving and can provide us with valuable information on how novel domains may fold (Kihara et al., 2001).

Most recent breakthroughs have arisen with the onset of deep learning. New approaches built using Artificial Intelligence (AI) have been accelerating the structure prediction field by far. A method called AlphaFold was developed by the DeepMind AI company has shown a significant progress on generating 3D models of proteins in the worldwide protein prediction competition, CASP (more details about this competition will be addressed in the later sections). The method was placed first in rankings among the teams that entered in protein modelling competitions (details about the protein structure prediction competition community will be addressed later). The reason behind this success lies to the integration of the Deep Artificial Neural Neworks (DANNs) approach,

which is a system of many learning algorithms which can be trained in order to be able to search the protein landscape thoroughly, and as a result, it generates highly accurate structures. Such a success has drawn the attention from all structural biologists to start studying this field in depth.

1.10. Protein structure prediction software

The field of computational protein prediction is evolving constantly, following the increase in computational power of machines and the development of intelligent algorithms. Such an evolution made the classification and categorisation of these methods hard to conserve. In this section, some of the most popular programs and servers for protein structure prediction will be listed (Table 1.2) according to the organisations which are interested on protein structure predictions, CASP (Moult et al., 2005) and CAMEO (Haas et al., 2018). Some programs/servers can have more than one edition/approach so that they can fulfil different methods.

Program	Type	Short Description	Reference
IntFOLD	FM/TBM	An approach that works via iterative multi-template-based modelling, using the target-template alignments from 14 alternative methods and eight alternative threading methods, and a powerful model quality assessment component called ModFOLD.	(Roche et al., 2011) (McGuffin et al., 2019)
I-TASSER	FM/TBM	A hierarchical approach that MetaServer which combines various TASSER-based approaches.	(Roy et al., 2010) (Yang and Zhang, 2015)
MODELLER	TBM	A homology or comparative modelling approach which calculates a model containing all non-hydrogen atoms through the satisfaction of spatial restraints.	(Fiser et al., 2000) (Webb and Sali, 2016)
PconsFold2	FM	The program uses contact predictions from PconsC3, the CONFOLD folding algorithm and model quality estimations to predict the structure of a protein.	(Michel et al., 2017) (Bassot et al., 2019)
QUARK	FM	A computer algorithm for ab initio protein modelling that uses replica-exchange Monte Carlo simulation under the guide of an atomic-level knowledge-based force field.	(Xu and Zhang, 2012)
RaptorX-Contact	FM	An approach that integrates evolutionary and physical constraints using machine learning (Random Forests) and integer linear programming.	(Wang et al., 2017)
RBO_aleph	FM	A machine learning method that uses graph-based features of contact.	(Schneider and Brock, 2014)
Rosetta	FM	A distributed computing project for protein structure prediction on the Berkeley Open Infrastructure for Network Computing (BOINC) platform, run by the Baker laboratory at the University of Washington.	(Das and Baker, 2008) (Ó Conchúir et al., 2015)

Table 1.2. List of a few of the common successful TBM and FM programs and servers. The list was sorted in alphabetical order starting with our method which was developed by Dr Liam McGuffin, IntFOLD.

1.11. Critical Assessment of Structure Prediction

Such techniques are a few of several other modelling techniques which have been developed and utilised through the last decades in order to solve the protein sequence-structure gap dilemma. The importance and far-reaching implications of having the ability to predict protein structures from their amino acid is manifested by the ongoing biennial competition on "Critical Assessment of Structure Prediction" (CASP).

The Critical Assessment of Techniques for Protein Structure Prediction or CASP is a global community-wide experiment that has started taking place every other year since 1994 (Moult *et al*., 1995). Protein structure modellers in more than a hundred research centres around the world dedicate their late spring and summer to preparing their methods to be independently tested in this centre. CASP is designed as a blind prediction experiment. A set of protein sequences is selected by the organisers in order to assess the performance of the structural protein methods that will predict it and help advance them. In the first CASP, the experiment was quite basic consisting of just three parts: collecting targets experimentally, collecting tertiary structure predictions, and assessing and discussing the results (Moult *et al*., 1995). The centre has since become popular, and its participants with their experiments have been increasing continuously over the years until it has taken the form of a competition which can be thought of as the "World Protein Structure Prediction Championships". Thirteen CASP experiments have been performed during the last 20 years, with the last one completed in late 2018. The competition has evolved over the years and is now carried out dividing its experiments into slightly more complicated sub categories, including tertiary structure prediction; disorder prediction; contact prediction; model quality assessment or (QA) which recently was called Estimate of Model Accuracy (EMA); binding site prediction; protein-protein interactions; oligomerisation state; protein model refinement (Roche & McGuffin, 2016). Each category represents an important part of the experiment that needs improvements in the predictive power of algorithms, which will lead to having a high level of accuracy and consistency in producing models close in quality to the experimentally derived protein structures.

1.12. Estimate of Model Accuracy (Model Quality Assessment)

Protein structure modeling is far less accurate in terms of the credibility than deriving protein structures from experiments. Models are typically left un-annotated with quality estimates and can span a broad range of the accuracy spectrum, whereas the accuracy of observed protein structures

can be estimated from experiments and falls within a narrow range (Martí-Renom *et al*., 2000). Therefore, a number of unbiased evaluation methods have been developed by modelers using some techniques such as statistical potentials, molecular mechanics energy-based functions, stereochemistry checks, and machine learning in order to analyse the correctness of protein structures and models, and to tackle their problems (Kryshtafovych & Fidelis, 2009). Examples of the early quality assessment tools are WHAT-CHECK (Hooft *et al*., 1996), PROCHECK (Laskowski *et al*., 1996) and, more recently, MolProbity (Lovell *et al*., 2003). These tools use basic stereochemical checks, and they are very useful in identifying unusual geometric features in a model. However, such early quality assessment tools are not able to produce a single score for ranking alternative models. Other examples of early quality assessment tools, that use a variety of different methods, are ProSA (Sippl, 1993) and DFIRE (Zhou & Zhou, 2002), which have been used along with VERIFY3D (Lüthy *et al*., 1992) in order to provide single scores that relate to the global quality of protein models. Machine learning-based quality assessment programs have also been utilised to provide a higher value of prediction accuracy. ProQ (Wallner & Elofsson, 2003), the first ModFOLD method (McGuffin, 2007) and QMEAN (Benkert *et al*., 2008) are examples of machine learning-based QA method, which helped programmers to use various combinations of structural features and individual energy potentials in order to increase the accuracy of predicted global model quality.

Because of the necessity of quality assessment methods, the CASP7 (2006) organisers had decided to include a section for evaluating Model Quality Assessment Programs (MQAPs) as a separate category in CASP (Fasnacht *et al*., 2007). In this category, CASP assessors will be testing the methods that assess the quality of protein models. How accurate are MQAP methods in distinguishing the similarity between model and native protein structures is the most important goal in this category. Initially in CASP7, MQAP participants started using their QA tools to evaluate each model individually and produce predicted global quality scores for a single model at a time, except some other candidates, such as the Swedish meta server called Pcons, used a clustering approach. In Pcons, it assumes that if the majority of the joined servers agree on a similar model, then that model is more likely to match with its observed protein structure than models proposed by one or a few servers. In other words, instead of considering a protein model from one server in isolation, Pcons considers models from many prediction servers, and each server uses a different modelling method from the other. The Pcons method works by carrying out all against all pairwise

structural comparisons to assign a confidence score to each model. By using this technique, Pcons had outperformed all the single-model servers by generating nearly 10% more correct predictions (Lundström *et al*., 2001). Thereafter, this technique has been classified as a consensus-based or clustering model quality assessment method, which is contrastable to single-model measurements.

1.13. Project objectives

The main aim of the study is to gain an improvement in the performance of our protein model quality assessment method, ModFOLD6. For the initial aims of the study, an investigation of the potential state of the art model quality assessment programs was undertaken. A review of where our method stands between the other MQAPs is searched. As the first step of the project, a number of the top ranked MQAPs are selected and revised in order to reflect the actual performance of ModFOLD6 in comparison with the top performing MQAPs.

The initial part of analysis includes a measurement of correlations between predicted quality scores from the methods selected for this study and four standard observed scores. These correlations were performed using the Pearson's R, Spearman's Rho and Kendall's tau B correlation coefficients (more details about these measurements will be addressed in the Materials and Methods section of this chapter) in a try to optimise the current scoring method of ModFOLD6. The optimised ModFOLD6 was benchmarked and cross validated among the top model quality assessment programs using data from the official worldwide competition, CASP11. The second chapter shows the outcome of the benchmarking with a number of suggestions that can improve our EMA method.

For the second objective of the study, two different Deep Neural Network techniques (RSNNS and TensorFlow) were utilised to find a new way to combine the ten top performing model quality assessment methods in order to improve local as well as the global quality score. Both techniques are featured with the multilayer perceptron (MLP) class of feedforwarding artificial neural network. The differences between them rests in their complexity. The techniques were tested for comparisons showing different results. Several neural networks architectures were trained, and a number of scoring measures were tested for a try to feature our EMA method with a deep learning technique that can improve the accuracy of model estimations.

In the second year of this project, we participated with our updated version of ModFOLD6 in one of the most popular competitions in protein structure prediction field, CASP12. The method was independently benchmarked among the most powerful participating EMA methods. Chapter 4 addresses a detailed description about our updated method and the successful results which ModFOLD6 achieved from the participation in CASP12.

After CASP12, EMA researchers started to focus on integrating the Deep Neural Networks as they found it to be a useful tool in strengthening their methods. Several were showing a huge difference between including and excluding DANNs to their methods. This technique has started to be utilised in a number of novel MQAP methods such as the Wang methods (Liu *et al*., 2016) and DL-pro method (Nguyen *et al*., 2014). However, the benefit of deep learning technique has been emerged in ProQ3D, a recently developed EMA method that could not participate in CASP12 (Uziela et al. 2016). After evaluating it, ProQ3D was proven to show a high improvement when the developers utilised the theano backend as the deep learning strategy. The third objective involves Deep Artificial Neural Networks parameterisation analysis for the purpose of gaining more optimisations. The TensorFlow python software library was utilised in order to determine the hyperparameters for a rank-optimised network and a correlation-optimised network.

For the fifth objective, a number of the pioneering pure-single and quasi-single model approaches as well as some scoring functions were integrated to our EMA method. Such an integration plus the previous updates gave our method the strengths to accurately score and rank predicted models with higher consistency. The method was benchmarked, cross validated, and tested to be ready to participate in the coming CASP as the 7th version of our program. In 2018, ModFOLD7 was ready to participate in CASP13, and the competition showed excellent results for the method. Such a success was described in detail in Chapter 6.

After CASP12 and CASP13, both upgraded ModFOLD methods have become available for public use. The methods have been used in many experimental biological works and has been cited in several publications. Alongside with the main study, a number of experimental applications of the updated ModFOLD6 and ModFOLD7 were carried out by a collaboration between our team and others in the biological school. For the last chapter of this thesis, a number of applications of ModFOLD6 and ModFOLD7 which have been carried out during the project study will be described.

Chapter 2

Benchmarking ModFOLD6 among Ten MQAP Methods for Local and Global score Optimisations

2.1. Background

2.1.1. ModFOLD

In the two years following CASP7, performance of protein structural QA servers were observed to be considerably increasing. MQAP tools have become the cornerstone of many protein structure modeling methods. More than a dozen papers were published in the area of QA between CASP7 and CASP8, and 45 methods were submitted for evaluation to CASP8 in that category. ModFOLDclust2 (McGuffin, 2009), MQAPRank (Jing and Dong, 2017), MULTICOM_cluster (Wang et al., 2010), ProQ2 (Ray et al., 2012), VoroMQA (Olechnovič and Venclovas, 2017), ZHOU-SPARKS-X (Yang et al., 2011) are all examples of QA methods. Some methods have been developed originally by their predictors using their own approach to measurements. Other methods U meta-server/consensus approach that was rated as a highly effective MQAP was ModFOLD (McGuffin, 2007).

ModFOLD is a machine learning-based QA program that was developed at the University of Reading by Dr McGuffin's group (McGuffin, 2007). The original ModFOLD method was developed based on the nFOLD protocol (Jones *et al*., 2005), which is a combination of the new GenTHREADER protocol (McGuffin & Jones, 2003) and a number of extra inputs into the underlying neural network. The idea behind GenTHREADER is implied in three stages (Jones, 1999). Firstly, the sequence alignment using BLASTP program (Altschul *et al*., 1990) (the later versions of GenTHREADER used PSI-BLAST (Altschul *et al*., 1997)) to scan the template sequence against non-redundant dataset of proteins. Secondly, the pair potential and solvation calculation terms based on a set of pairwise potentials of mean force (Hendlich *et al*., 1990), the last one is determined by resolved protein X-Ray crystal structures and the application of the inverse Boltzmann equation (Jones *et al*., 1992), plus using a solvation potential (Jones *et al*., 1992). Thirdly, the alignment evaluation by training a neural network in order to combine sequence alignment score, length information and energy potentials from threading into a single score. In the original GenTHREADER, the neural network was simply trained using a binary classification system called CATH (Dawson et al., 2017). This publicly available online resource was created in the middle of 1990s by Professor Christine Orengo and colleagues (Orengo *et al*., 1997). CATH database can provide information on the evolutionary relationships of protein domains. CATH resource can be accessed freely at: https://www.cathdb.info/. In later versions of GenTHREADER, the neural network was modified to be trained to learn a proteins similarity measurement termed

FSSP Z-scores (Holm and Sander, 1996) which has improved the program. The nFOLD protocol was developed afterwards by feeding three additional inputs into the neural network, which include the SSEA score (McGuffin & Jones, 2003), a new functional site detection score (MetSite) (Sodhi *et al*., 2004), and a simple model quality checking algorithm, MODCHECK (Jones & McGuffin, 2003).

Initially, ModFOLD was developed in two editions: ModFOLD, designed to be fast and used for the global assessment of either single or multiple models, and ModFOLDclust, a more intensive method that carries out clustering of multiple models and provides a per-residue local quality assessment. ModFOLDclust has shown to significantly outperform all of its clustering/multiple MQAP competitors, while ModFOLD has competed well against some of the best "true" single model MQAP methods (McGuffin, 2007). Since CASP ranking relies on the prediction accuracy regardless of the method used, clustering- or consensus-based MQAPs were ranked as the most accurate methods for predicting 3D model quality, outperforming the single model methods.

2.1.2. Q-score in ProQ2 and ModFOLDclustQ for Speed, Accuracy and Consistency

Despite their accuracy, it was noticed that a number of advantages of the single model-based methods were missing in the clustering methods. One missing feature was the speed. Like Pcons and other consensus-based approaches, ModFOLDclust carries out pairwise comparisons of numerous models by using multiple structural alignments, and that makes it often CPU intensive (McGuffin, 2008). Another difficulty found in QA programs including ModFOLDclust was the requirement of a large pool of diverse models, having a small number of models can minimise the efficiency of MQAPs (Cao, Wang, & Cheng, 2014). To overcome such problems, Roche designed an upgraded version of the same method, they called it ModFOLDclustQ (Roche et al., 2014). The initial 'Q' labeled in the upgraded version name is referred to a score called Q-score has been utilised in ModFOLDclustQ, while also standing for Quick. The Q-score is derived from the Q measure that was developed by the Wolynes group (Eastwood *et al*., 2001). The Q-score has the ability to efficiently estimate structural relations between two proteins based on their residue distances. This method has been suggested by the CASP8 assessors as an alternative to the other scoring methods such as the GDT-TS (Ben-David *et al*., 2009). By importing Q-score, ModFOLDclustQ has shown to compete with the leading consensus MQAPs, but that was not the eventual state of ModFOLDclust. When taking the mean of ModFOLDclustQ scores and its older

ModFOLDclust, it showed a significant increase in prediction accuracy, with little computational overhead. That led Dr McGuffin's group to combine both scoring methods to form a new method named ModFOLDclust2 (Roche et al., 2014). There are a number of other MQAPs that also used Q-score to assess each individual residue in a model pertaining to the per-residue accuracy. A successful per-residue consensus-based method was Pcons method, which was superseded then by one of the leading consensus single model per-residue programs, known as ProQ (Wallner & Elofsson, 2003). The method was then upgraded by updating its structural and predicted feature, this upgrade to be as the second top ranking MQAP, ProQ2 (Wallner & Elofsson, 2007).

Although upgrading ModFOLDclust to ModFOLDclustQ and combining their scores have shown a high improvement in the quality assessment speed and accuracy level, McGuffin's group also noticed the potential of using ModFOLDclust2 to guide 3D modelling using multiple templates. In the process of modelling, using more than one-fold template is helpful in assessing models more accurately. However, it was noticed that such a technique is not preferable in many cases as it may result in poorer model quality. Besides the speed and the accuracy of an MQAP, there has to be the consistency as well. To solve such a problem, Dr McGuffin and colleagues have started to investigate the use of local as well as global model quality prediction scores that are produced by ModFOLDclust2. This led to improvements in the selection of target-template alignments for the construction of multiple-template models. After the investigation, it was found that the most accurate and consistent way in improving models is to use accurate local model quality scores to guide alignment selection while using accurate global model quality before selection for re-ranking alignments. Applying this technique has made significant performance improvements to the IntFOLD server (Buenavista *et al.*, 2012).

2.1.3. "Quasi-single-model mode" algorithm

Another important feature that was missing in the clustering base approaches is addressing the real-life needs of protein researchers when often only a single or few models for each protein target are available for evaluation. In that case, clustering methods will provide a very poor result in performance. McGuffin's group was aware of this problem and they found a way to solve it. Instead of proceeding a direct clustering to the submitted model/s, a tertiary structure prediction method (IntFOLD2-TS, Roche *et al.*, 2011) is used at the beginning as the first stage of the quality assessment procedure to generate an initial reference set of template-based models. The submitted

model/s of the target with the generated models are then clustered using ModFOLDclust2 as the second stage of the process. By this algorithm, if the server received multiple models then the procedure will go with the full clustering approach, whereas if only single or few models are submitted, then the operation will take the so called quasi single-model approach, which operates with comparable accuracy. This method has been implemented initially with the ModFOLD v3.0: a server developed using ModFOLDclust2 integrated with the IntFOLD-QA tertiary structure prediction pipeline (McGuffin & Roche, 2010). The algorithm has since been independently tested for confidence and published with the term "quasi-single-model mode" in the fourth version of ModFOLD, when ModFOLDclust2 was integrated with IntFOLD2-TS (McGuffin *et al*., 2013).

CASP assessments of QA methods were more concerned about the quality scoring results rather than other practical considerations, such as the researcher accessibility, until the assessment was updated following the eighth and ninth experiment (Kryshtafovych *et al*., 2011). In CASP10, the criteria were modified to rebalance the quality assessment. This modification was implied by using smaller bespoke data sets rather than allowing large sets of models, which some said unfairly favoured clustering approaches. ModFOLD4 was the first beneficiary of this change of focus having been benchmarked independently at CASP10 where it was ranked among the top performing methods in the quality assessment category. The ModFOLD4 server provides a free service for accurate prediction of global and local QA of 3D protein models. ModFOLD4 has a comparable performance to clustering-based methods but retains the capability of making predictions for a single model at a time, which is what has made it such a powerful MQAP (McGuffin *et al*., 2013).

In 2015, the 5th version of ModFOLD has been released. This version was integrated with the upgraded tertiary structure prediction IntFOLD3-TS pipeline which has given ModFOLD5 the ability to generate greater number and variety of reference models (McGuffin *et al*., 2015).

2.2. Objective

The main aim of study is to gain an improvement in the performance of the protein model quality assessment method, ModFOLD6. For this part of the study, an investigation of the potential state of the art model quality assessment programs is undertaken. A number of the top-ranked MQAPs are selected and revised in order to reflect the actual performance of ModFOLD6 in comparison

with the top performing MQAPs. The initial part of analysis includes a measurement of correlations between predicted quality scores from the methods selected for this study and four standard observed scores using the Pearson's R, Spearman's Rho and Kendall's tau B correlation coefficients.

2.3. Materials and Methods

In this section, we describe the initial study in our research. A number of methods were used for benchmarking ModFOLD6 in order to evaluate its performance and the possibilities of improving its pipeline. Such evaluating methods include linear and non-linear regressions, ranking and correlation methods individually and in combinations. Before getting into details, Figure 2.1 can show an overall flow of the process and steps which are addressed in this section.

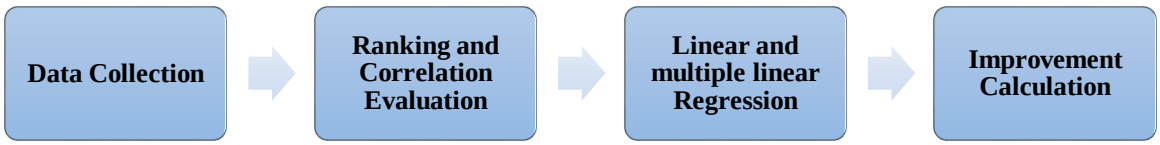

Figure 2.1. Flowchart summarising the overall process of section 2.3.

2.3.1. Ten MQAPs

A selection of ten high performing model quality assessment methods were benchmarked using models that submitted by servers in CASP experiments. Six of these methods that were related to ModFOLD were benchmarked against four other single-model MQAPs using four observed global scoring measures. The ten MQAPs producing the predicted quality scores were ModFOLD5_single_orig_global (M5so), ModFOLDclustQ_single_orig_global (Mcqso), ModFOLDclust2_single_orig_global (Mc2s), ModFOLD5_single_res_global (M5sr), ModFOLDclustQ_single_res_global (Mcqsr), ProQ2_res_global (P), CDA_res_global (C), DBA_res_global (D), SSA_res_global (S) and ModFOLD6_single_res_global (M6), and the four observed quality scores were GDT-HA, GDT, MaxSub and TM-score. This analysis was implemented using the statistical computing software, R v3.2.3.

2.3.1.1. ModFOLD5_single_orig_global (M5so)

The global scores were taken from our quasi-single model method that was developed for CASP11. The global score was calculated from the comparison of each model with the reference set of models built by IntFOLD3 server. Individual models were compared against the reference IntFOLD3 set using the TM-score (Zhang & Skolnick, 2004). The equation for $TMscore = \text{Max}\left[\frac{1}{L_N}\sum_{i=1}^{L_T}\frac{1}{1+(\frac{d_i}{d_0})^2}\right]$ where L_N and L_T were the lengths of the native structure and the residues to the template structure respectively. d_i was the distance between i the pair of aligned residues and d_0 was a scale to normalise the match difference. The ModFOLD5_single_orig_global (M5so) global score was calculated as $M5so = \frac{1}{N-1}\sum_{m \in M} Tm$ where $M5so$ was the global quality score for a model, N was the number of models for the target, $N-1$ was the number of pairwise structural alignments carried out for each model (i.e. models were not aligned with themselves), M was the set of alignments and Tm was the TM-score for each pairwise alignment of models. A TM-score cut-off was implemented so that alignments with scores < 0.2 were not included in the calculation. Therefore, the size of set M was equal to the number of alignments with TM-scores ≥ 0.2.

2.3.1.2. ModFOLDclustQ_single_orig_global (Mcqso)

This quasi-single model QA score was calculated from the comparison of each model with the reference set of models built by IntFOLD3 server in a similar way to the ModFOLD5_single_orig_global score described above, however here individual models were compared against the reference IntFOLD3 set using the Q score (McGuffin & Roche, 2010) (Eastwood *et al.*, 2001).

In each pairwise comparison, the matrix of internal distances for 'model a' was designated as r_{ij}^a and the matrix for 'model b' was designated r_{ij}^b , where r_{ij} was the distance between the *Ca* atom of residue i and the *Ca* atom of residue j in the same model. For each pair of equivalent residues in a pairwise comparison of models, a score Q_{ij} was calculated as $Q_{ij} = exp\left[-(r_{ij}^a - r_{ij}^b)\right]$ Thus, for a perfect match between residues $Q_{ij} = 1$ and for poor match $Q_{ij} \approx 0$. In the paper by Ben-David *et al.*, two measures, Q_{long} and Q_{short} were described to indicate the observed quality of the

tertiary structure and secondary structure prediction, respectively, for a given model. The Q_{short} score for a given model was calculated by averaging Q_{ij} , considering the best pair and 20, 40, 60, 80 and 100% of the ranked pairs that satisfied $|i-j| \leq 20$. Conversely, Q_{long} was calculated by considering those that satisfied $|i-j| > 20$. For this study, both the Q_{long} and Q_{short} measures were attempted separately; however, it was found that the best results were obtained by including both the long- and short-range internal distances by averaging Q_{ij} for all ranked pairs ($|i-j|>0$) to give a score, Q_{tot} . For a given model, the Q_{tot} scores were calculated for each pairwise comparison and the final ModFOLDclustQ global (QMODE1) prediction of model quality was calculated as $Mcqso = \frac{1}{N-1}\sum_{m \in M} Q_{tot}$ where $Mcqso_{global}$ was the global model quality score for a model, N was the number of models for the target, $N-1$ was the number of pairwise comparisons carried out for each model, M was the set of comparisons and Q_{tot} was the pairwise similarity between models.

2.3.1.3. ModFOLDclust2_single_orig_global (Mc2s)

This quasi-single model QA score was equivalent to the mean of the ModFOLD5_single_orig_global (ModFOLDclust_single_orig_global) and the ModFOLDclustQ_single_orig_global scores described above. Thus, $Mc2s = (M5so + Mcqso)/2$.

2.3.1.4. ModFOLD5_single_res_global (M5sr)

The sum of the predicted per-residue errors calculated using the ModFOLDclust_single quasi-single model QA method was divided by the original target sequence length. For a residue in a pairwise superposition the *S*-score (McGuffin *et al.*, 2013) was defined as $S_i = \frac{1}{1+(\frac{d_i}{d_0})^2}$ where S_i represents the *S*-score for residue i in a model, whereas d_i represents the distance between aligned residues according to the TM-score superposition, and d_0 represents the distance threshold (3.9Å). That means the S_i scores will be given only if d_i was above 3.9Å. For each residue, the *S*-scores were summed, and the mean score is calculated. The equation is $S_r = \frac{1}{N-1}\sum_{a \in A} S_{ia}$ where S_r provides the accuracy of the predicted residue for the model, N shows the models number for the

target, A is the set of alignments and S_{ia} is the S_i score for a residue in a structural alignment a. The final global score is the $(\sum Sr)/\mathrm{L}$ where, L is the original target sequence length.

2.3.1.5. ModFOLDclustQ_single_res_global (Mcqsr)

The sum of the predicted per-residue errors using the ModFOLDclustQ_single QA method was divided by the original target sequence length.

2.3.1.6. ProQ2_res_global (P)

The sum of local scores are taken from the ProQ2 (Wallner & Elofsson, 2007) pure single model method was divided by the original target sequence length. Thus, $P = (\sum ProQ2)/L$ where, $ProQ2$ is the per-residue score and L is the original target sequence length.

2.3.1.7. CDA_res_global (C)

This score was based on our new pure single model QA method from ModFOLD6. The Contact Distance Agreement (CDA) score relates to the agreement between the predicted residue contacts according to the MetaPSICOV (Jones *et al.*, 2015) sequence-based method and the measured Euclidean distance (in Angstroms) between residues in the model. All pairs of residues in a model that were measured to be 8 Angstroms apart or less were considered and the CDA score for each residue (*i*) was calculated by the mean MetaPSICOV p-value for that residue. Thus, $C = \sum p/numC$ where, p was the probability of the two residues being in contact according to MetaPSICOV, and $numC$ wass the number of contacts <= 8 Angstroms for the residue in the model where a value for p exists. The C score was the sum of the per-residues CDA scores divided by the original target sequence length. Thus, $C = \sum CDA/L$.

2.3.1.8. DBA_res_global (D)

This score was based on our new quasi-single model QA method from ModFOLD6. The Disorder B-factor Agreement (DBA) score relates to the agreement between the predicted disordered residues in the sequence according to DISOPRED3 (Jones & Cozzetto, 2015) and the

ModFOLDclust_single predicted per-residue error in the model (as it appears in the B-factor column). Thus, $D = 1 - |S_r - (1 - P_d)|$ where, S_r was the accuracy of the predicted residue for the model, as described in the *S*-scores equation above, and P_d was the probability of disorder according to DISOPRED3. The DBA_res_global score was the sum of the DBA scores for each residue divided by the original target sequence length. Thus, $D = \sum DBA/L$.

2.3.1.9. SSA_res_global (S)

This score was based on our new pure single model QA method from ModFOLD6. The Secondary Structure Agreement (SSA) score relates to the agreement between the predicted secondary structure of each residue according to PSIPRED (Buchan *et al*., 2013) and the secondary structure state of the residue in the model according to DSSP (Kabsch & Sander, 1983). Thus, $S = P_{CHE}$ where, P_{CHE} was simply the p-value from PSIPRED for the secondary structure state of the residue, coil (C), helix (H) or strand (E), in the model according to DSSP. The SSA_res_global score was the sum of the SSA scores for each residue divided by the original target sequence length. Thus, $S = \sum SSA/L$.

2.3.1.10. ModFOLD6_single_res_global (M6)

ModFOLD6 is our new neural network based quasi-single model method that takes as its input a sliding window of per-residue scores from the ModFOLD5_single, ModFOLDclustQ_single, ProQ2, CDA, DBA & SSA methods described above and outputs a single quality score for each residue in the model. The ModFOLD6_single_res_global score was the sum of the ModFOLD6 local scores for each residue divided by the original target sequence length.

2.3.2. Observed Model Quality Measurements

In order to evaluate predicted model quality scores, four observed scoring measures were used. The measures were GDT-HA, GDT, MaxSub and TM-score. These scores were used to measure the observed model quality for each individual model by comparing them to the native (solved experimental) structures. The term GDT stands for “global distance test”, in both the GDT and GDT-HA scores. These two scores represent the measurement of similarity between two protein

structures that both have identical amino acid sequences but may have different tertiary structures i.e. a predicted model and the observed crystal structure (Zhang & Skolnick, 2004). The difference between GDT and GDT-HA is that GDT-HA is "High Accuracy" and uses smaller cut off distances, which makes it more rigorous and, as a result, is more stringent than GDT (Read & Chavali, 2007). MaxSub is a measure that identifies in a model the largest subset of Cα atoms that superimpose over the experimental structure, producing a single normalised score that represents the quality of that model. And finally, TM-score stands for "template modelling" score. Again, this measure is for calculating the similarity between two models with the same sequence, but with different tertiary structure. The TM-score is arguably more accurate than GDT and GDT-HA in comparing the similarity of structures with full-length protein chains, rather than domains (Zhang & Skolnick, 2004). Each of these scores indicate the difference between two protein structures (predicted versus observed) by providing a score between 0 and 1, where 1 is a perfect match between the two compared structures (i.e. identical relative atom coordinates) and 0 is a non-matched structure (Siew et al., 2000). The predicted output scores produced were tested by comparing them to the observed scores.

2.3.3. Data Collection

Predicted QA scores (from the 10x MQAPs) and observed QA scores (from GDT-HA, GDT, MaxSub and TM-score) were collected by evaluating the 16483 models produced for the CASP11 QA assessment category. Scores for 2383 models were removed because they did not have native structures available, which means that those models could not form part of our benchmark, so we ended up with a net amount of 14100 models. The individual QA scores for each model were then collated, separated and distributed into columns, each column was named by its related method.

2.3.4. Ranking/Selection and Correlation evaluation

The collected scores were evaluated for all MQAP methods. Cross-validation tests were carried out using the collected data from CASP11 to measure the ability of each MQAP method individually as well as in different combinations. This measurement was evaluated in terms of local/per-residue and global scores which are produced by the MQAP methods using two different ways of scaling. Firstly, the Ranking/Selection approach which evaluates the ability of a method

in ranking models correctly so that they can select the right top model more accurately. This scale of measurement is calculated by measuring the cumulative GDT-HA, GDT, MaxSub and TM-score scores (e.g. ΣGDT-HA). Secondly, the Correlation coefficient approach which evaluates the ability of a method in how consistent its prediction can be, and how this method can achieve the highest correlation between its predicted scores and the observed scores. This second scale of evaluation was carried out using Pearson's (R), Spearman's rank (Rho) and Kendall's rank (Tau). Each of the three method has its own properties. Spearman's rank correlation is a non-parametric test which measures association between two variables without making assumptions on bivariate relationships (McDonald, 2014). Kendall's rank correlation is also a non-parametric test but measures the strength of dependence between the two variables by quantifying the difference between the percentage of concordant and discordant pairs among all possible pairwise events (Legendre, 2005). Pearson's correlation coefficient is different in that it is a parametric test, which measures the degree of relationship between the linearly related variables (McDonald, 2014).

2.3.5. Linear Regression for MQAPs Individually

Following the collection of the target scores for each individual model, correlations were performed. Each score of the ten MQAPs was correlated individually with its native observed score. This scoring correlation is carried out in order to investigate the relationship between the predicted and observed scores. The correlation has been implemented for this study by using three methods, Pearson's R, Spearman's Rho and Kendall's tau B correlation coefficients. Each correlation method has its property in scoring the relation between the predicted and the observed quality assessments. Pearson's correlation coefficient is a linear correlation measure whereas Spearman's Rho and Kendall's tau B correlation coefficients are non-parametric measures.

2.3.6. Linear Regression for MQAPs in Combinations

After scoring the ten MQAPs by correlating them individually, an optimisation is performed. The ten MQAP methods were combined using all the combination odds (1012 combinations) for all collected scores. The combined MQAP methods were then correlated using linear regression to determine the most correlated combination of methods that shows the highest positive correlation. This analysis was implemented using the three types of correlation coefficients, Pearson's R,

Spearman's Rho and Kendall's tau B. Four of the most correlated combinations were highlighted for further analysis to find the resolution at which the best correlation coefficients of the optimum combination occur.

2.3.7. Multiple Linear Regression for MQAPs in Combinations

The top four combinations of predictive methods, according to the correlations with observed scores, were taken forward for multiple linear regression analysis. Each combination was correlated against the other three combinations using the observed scores as measures. The analysis was performed using Pearson's R, Spearman's Rho and Kendall's tau B correlation coefficients.

2.3.8. Improvement Calculation

All scoring results were put together for comparison, and calculations were performed in order to analyse any improvement could be achieved. Firstly, the individual as well as the combination performance for the ten MQAP methods were compared. Such a comparison allowed us to gauge the improvement that could be obtained from combining these MQAP methods i.e. the maximum individual score was subtracted from the maximum correlation achieved through the linear combinations. Secondly, the linear regression performance was similarly compared with the results obtained by the multiple linear regression in order to determine if any further improvements were gained. This last step of analysis was proceeded by finding the maximum value of score of the multiple linear regression of the combination methods and subtracting it from the maximum value of the linear regression MQAP combination scores.

2.4. Results and Discussion

In this study, 10 MQAP methods including the latest version of our method, ModFOLD6, were benchmarked using two scales of measurements, Ranking/Selection and Correlation.

2.4.1. Ranking/Selection benchmarking

For benchmarking using the ranking/selection scale, the cumulative scoring technique was carried out to find the top optimised MQAP pipeline in selecting the best model through the 10 selected MQAP methods. Firstly, each method was benchmarked individually against the four observed scores. The results showed that ModFOLD6_single_res_global was performing well compared to the other MQAP methods (Table 2.1).

MQAP Methods	GDT-HA	GDT	MaxSub	TM-score
ModFOLD5_single_orig_global	29.0719	40.0590	36.4405	42.4259
ModFOLDclustQ_single_orig_global	28.9016	39.9194	36.3461	42.2751
ModFOLDclust2_single_orig_global	29.0935	40.0328	36.5134	42.4060
ModFOLD5_single_res_global	29.0167	39.9715	36.3934	42.3664
ModFOLDclustQ_single_res_global	28.9016	39.9194	36.3461	42.2751
ProQ2_res_global	30.7222	42.9578	39.5015	45.7958
CDA_res_global	28.6485	40.4575	37.2034	43.1661
DBA_res_global	28.9930	40.0457	36.4816	42.4484
SSA_res_global	27.5855	39.3166	35.7842	41.9146
ModFOLD6_single_res_global	31.3937	43.1859	39.8614	45.8227

Table 2.1. Global score benchmarks of the 10 MQAPs individually using CASP11 data. ModFOLD6_single_res_global was benchmarked against the component of the global scoring MQAP methods, representing the cumulative scores from GDT-HA, GDT, MaxSub and TM-score.

In the second part, the cumulative evaluation was carried out with every combination from the 10 MQAP methods. After completing the test, the optimum combinations which produced the highest QA scores were selected for further analysis (Table 2.2).

MQAP combination	Observed measure	Cumulative Score
CDA_res_global SSA_res_global ModFOLD6_single_res_global	GDT-HA	32.34630537
CDA_res_global SSA_res_global ModFOLD6_single_res_global	GDT-TS	44.53946127
CDA_res_global SSA_res_global ModFOLD6_single_res_global	MaxSub	41.53627352
ProQ2_res_global CDA_res_global SSA_res_global ModFOLD6_single_res_global	TM-score	47.24946607

Table 2.2. Global score benchmarks of the 10 MQAPs in combinations using CASP11 data. ModFOLD6_single_res_global was benchmarked against the component of the global scoring MQAP methods in combinations, representing the cumulative scores from GDT-HA, GDT, MaxSub and TM-score.

2.4.2. Correlation benchmarking

For the correlation scale evaluation, the ten methods were benchmarked using Pearson's, Spearman's and Kendall's. These correlation coefficients were investigated between predicted and observed model quality scores to show which of the ten MQAPs either individually or in combinations can produce the largest amount of model quality scores that are highly close to their observed scores.

Firstly, the modelling scores produced from the 10 MQAP methods were correlated with the observed scores individually. The results from Table 2.3 shows that all the 10 methods performed well with all the correlation coefficients in general. The highest values and most consistency were shown with method Mc2s, it performed well when was correlated with GDT-HA and GDT using Spearman's rho (≈ 0.925, ≈ 0.924 respectively) and Kendall's tau (≈ 0.762, ≈ 0.768 respectively) measures. The ModFOLD5_single-res_global (M5sr) method performed well, being ranked as the second-best method compared to the 10 MQAPs with MaxSub and TM-score, when using Spearman (≈ 0.92, ≈ 0.928 respectively) and Kendall (≈ 0.771, ≈ 0.768) correlations. When Pearson

was used as a correlation coefficient measure, the DBA method (D) provided the highest result (≈ 0.924) among all the ten methods, except with GDT-HA, where was outperformed by ModFOLDclustQ_single_orig_global (Mcqso), ModFOLDclust2_single (Mc2s) and ModFOLDclustQ_single_res_global (Mcqsr) (≈ 0.901 versus ≈ 0.897 for DBA). Such results indicated the high performance of the quasi-single model based ModFOLD methods compared to the pure-single model MQAPs in terms of assigning absolute global accuracy values. However, the table illustrated a low level of consistency in the order of performance when using different observed scores.

	GDT-HA			GDT			MaxSub			TM-score		
	R	**Rho**	**Tau**	**R**	**Rho**	**Tau**	**R**	**Rho**	**Tau**	**R**	**Rho**	**Tau**
M5so	0.888	0.917	0.749	0.911	0.917	0.756	0.917	0.924	0.764	0.916	0.918	0.759
Mcqso	0.901	0.922	0.755	0.907	0.918	0.756	0.911	0.917	0.75	0.902	0.914	0.751
Mc2s	0.901	0.925	0.762	0.917	0.924	0.768	0.922	0.927	0.767	0.917	0.923	0.767
M5sr	0.892	0.922	0.757	0.912	0.922	0.765	0.92	0.928	0.771	0.915	0.923	0.768
Mcqsr	0.901	0.922	0.755	0.907	0.918	0.756	0.911	0.917	0.75	0.902	0.914	0.751
P	0.688	0.732	0.527	0.718	0.741	0.54	0.717	0.742	0.54	0.723	0.744	0.545
C	0.637	0.72	0.528	0.672	0.732	0.542	0.674	0.731	0.538	0.677	0.734	0.544
D	0.897	0.917	0.742	0.919	0.921	0.754	0.924	0.925	0.76	0.923	0.922	0.76
S	0.518	0.561	0.385	0.535	0.56	0.385	0.532	0.561	0.387	0.533	0.555	0.382
M6	0.881	0.916	0.748	0.914	0.92	0.762	0.914	0.924	0.763	0.919	0.922	0.767

Table 2.3. List of the top ranked individual MQAP methods based on predicted versus observed scores using Pearson's (R), Spearman's (Rho) and Kendall's (Tau) correlation coefficients. The models for each target were pooled together and each correlation was measured separately for each method and then the overall mean score was calculated. The observed model quality score was also calculated for each individual model. The highest scores are highlighted in grey. Abbreviations are defined in the List of Abbreviations section.

The DBA method was noticed to outperform all the ten MQAPs by providing the highest Pearson correlation using three out of four observed scoring methods (the highest one is ≈ 0.924). However, DBA lost its top spot according to the Spearman's rho and Kendall's tau correlations. These varying results with the Pearson correlation coefficient highlighted the non-linear nature of some of the high performing MQAP methods. The Pearson's correlation coefficient is always used for

measuring the strength of the linear relationship between two variables, whereas, Spearman and Kendall are distribution-free correlation coefficients, which can be used for assessing non-linear relationships between two variables. Typically, Spearman's and Kendall's methods showed how well an arbitrary monotonic function can describe the relationship between two variables without making any assumption about the frequency distribution of the variables (Hauke & Kossowski, 2011). Since, the relationship between predicted and observed model assessments was not always parametric and their target scores were freely distributed, Spearman and Kendall were more suitable method for measuring the correlation coefficient. Although there were strong correlations for some of the methods according to some measures, the sub optimal results shown here confirmed the necessity to improve the consistency of ModFOLD6 global scoring across the board. One attempt to fulfil such a need is by combining the strengths of many MQAPs to achieve a better score (McGuffin, 2007).

In the second part of the evaluation, the ten selected MQAPs were combined together in all permutations (where the order was not important, without repetition), and a linear regression for all the combination of the ten methods was performed. Each combination was compared with the four observed scores using the three correlation coefficients. After going through the permutations, improvements in correlation scores were shown (Table 2.4). The optimal combination was the combination of three methods, ModFOLDclustQ_single_orig_global, DBA_res_global and ModFOLD6_single_res_global. This simple combination (the mean of three scores) provided the highest score improvement for most of the correlations, and even the lower scores were close to those which were benchmarked highest in the other combinations. The results from the first attempt of this optimisation showed that the correlation was slightly increased when the new version of ModFOLD6 was combined with the approach based on the older ModFOLD method, and the new scoring method, DBA. This increase was measured as ≈ 0.01 over approximately 75% of the target scores. The second, third and fourth optimal combinations were also taken into account for further optimisation analysis as they provided close results comparing to the first optimal linear combination.

Correlation Coefficient	Observed Measure	Combination	Correlation Score	Improvement
R	GDT-HA	Mcqso+D	0.91	0.009
Rho		Mcqso+Mcqsr+D+M6	0.931	0.006
Tau		Mcqso+Mcqsr+D+M6	0.771	0.009
R	GDT	Mcqso+D+M6	0.926	0.007
Rho		Mcqso+D+M6	0.932	0.008
Tau		Mcqso+D+M6	0.782	0.014
R	MaxSub	Mcqso+D+M6	0.93	0.006
Rho		Mcqso+D+M6	0.935	0.007
Tau		Mcqso+D+M6	0.781	0.01
R	TM-score	D+M6	0.93	0.007
Rho		Mcqso+D+M6	0.932	0.009
Tau		Mcqso+D+M6	0.784	0.016

Table 2.4. List of the top ranked combinations for the ten MQAP methods based on predicted versus observed scores using linear regression. The combined methods were measured using Pearson's (R), Spearman's (Rho) and Kendall's (Tau) correlation coefficients, and the topmost correlated combinations were listed with their improvement over the scores of the methods individually. The highest improvement is highlighted in grey.

Results showed that the combination of Mcqso+D+M6 gave the highest correlation score, providing an approximate total of 66.67% among all the highly ranked scores. The second top was for the combination of four methods (Mcqso+Mcqsr+D+M6) providing ≈ 16.67% of the highly ranked scores. The third and fourth combinations were for the combinations of two methods (Mcqso+D and D+M6), giving a ≈ 8.33% each.

The four ranked combinations were then tested using multiple linear regression in an attempt to achieve additional improvement. However, the results of this test showed an insignificant increase in some of the selected optimal combinations scores (Appendix 1). the highest correlated combinations using multiple linear regression being produced by the combination of four methods, (ModFOLDclustQ_single_orig_global, ModFOLDclustQ_single_res_global, DBA_res_global and ModFOLD6_single_res_global) with a maximum score of ≈ 0.931 using Pearson's R, ≈ 0.935 using Spearman's Rho, and ≈ 0.784 using Kendall's Taue, while the variable was GDT. Some scores of the correlated optimal combinations were shown to be decreased when comparing with

the scores of the mean linear regression which showed a better correlation than the multiple linear regression. From the results, it can be seen that multiple linear regression provided an unconsidered increase to the QA scores neither for the MQAP methods individually nor in combinations.

However, the scores produced by combining MQAP methods appeared to be increased by approximately 2% as compared to the best scores from any individual of the ten MQAP methods. To demonstrate such an improvement, the results from both best individual and combined MQAP methods were plotted to visually compared their scores. In Figure 2.1, we can see a comparison of two scatterplots, one plot is representing the highest improvement in terms of Kendall's tau score achieved by the top optimal combination method Mcqso+D+M6, and the second plot is for the scores achieved by the top performing individual MQAP, Mc2s using the same correlation coefficient. The plot consists of dots, each dot represents a model that was collected from CASP11 participants servers. Both scores were compared against the TM-score observed measurement. The optimal combination output scores of Mcqso+D+M6 were shown to give a slightly tighter scatter of dots, leading to an enhanced correlation in comparison to the one achieved by Mc2s individually.

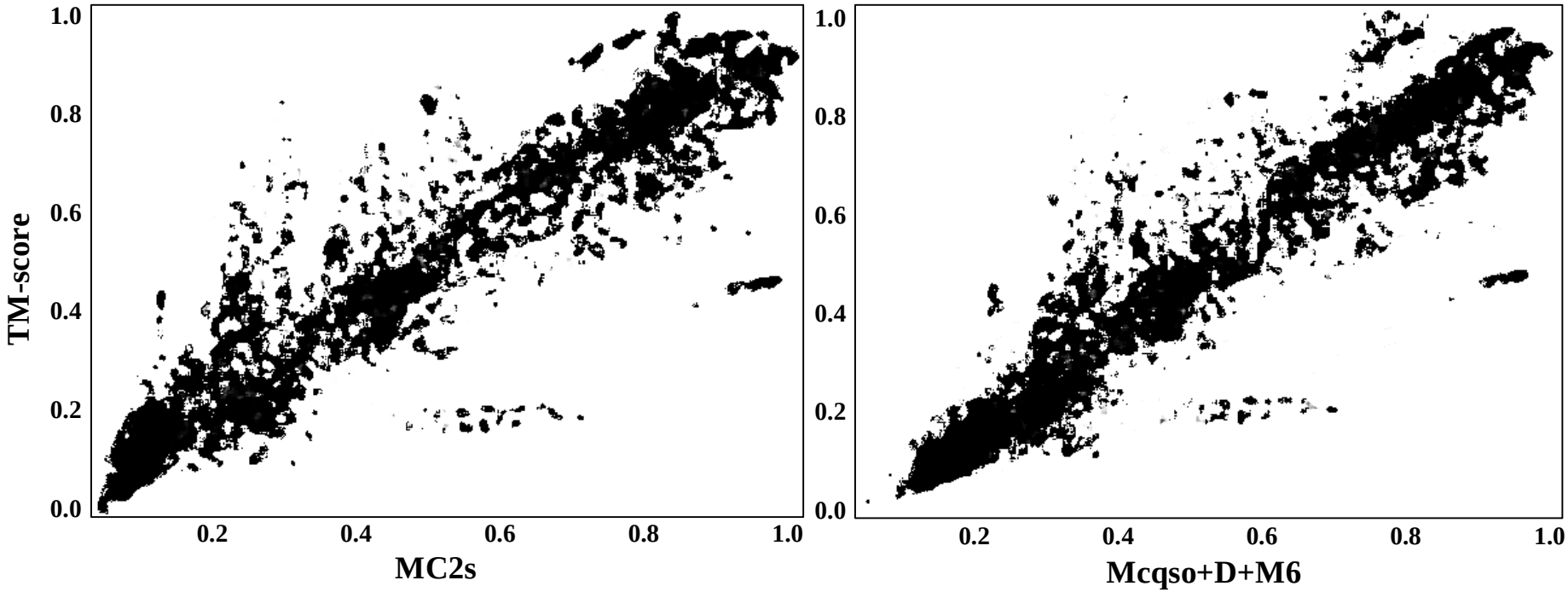

Figure 2.2. Predicted model quality scores versus observed model quality scores. A comparison between two scatterplots. The plot on the left is for the top performing individual MQAP, ModFOLDclust2_single_orig_global (Mc2s). The plot on the right is for the optimal MQAP combination ModFOLDclustQ_single_orig_global + DBA_res_global + ModFOLD6_single_res_global (Mcqso+D+M6). Both methods predicted global scores were plotted against TM-score observed score. The plots are presented in brush paint style for more clarity.

ModFOLD6 then was also benchmarked to gauge local quality performance versus component methods (Figure 2.2). The three pure-single scoring methods (ProQ2, CDA, SSA) and the three quasi-single scoring methods (DBA, Mcs, Mcqs) have been compared against ModFOLD6 in an evaluation of the target sequence and 3D model in terms of the ROC curve that shows the true positive and negative rate. From the line graph it can be noticed that ModFOLD6 is outperforming all the comprised scoring methods.

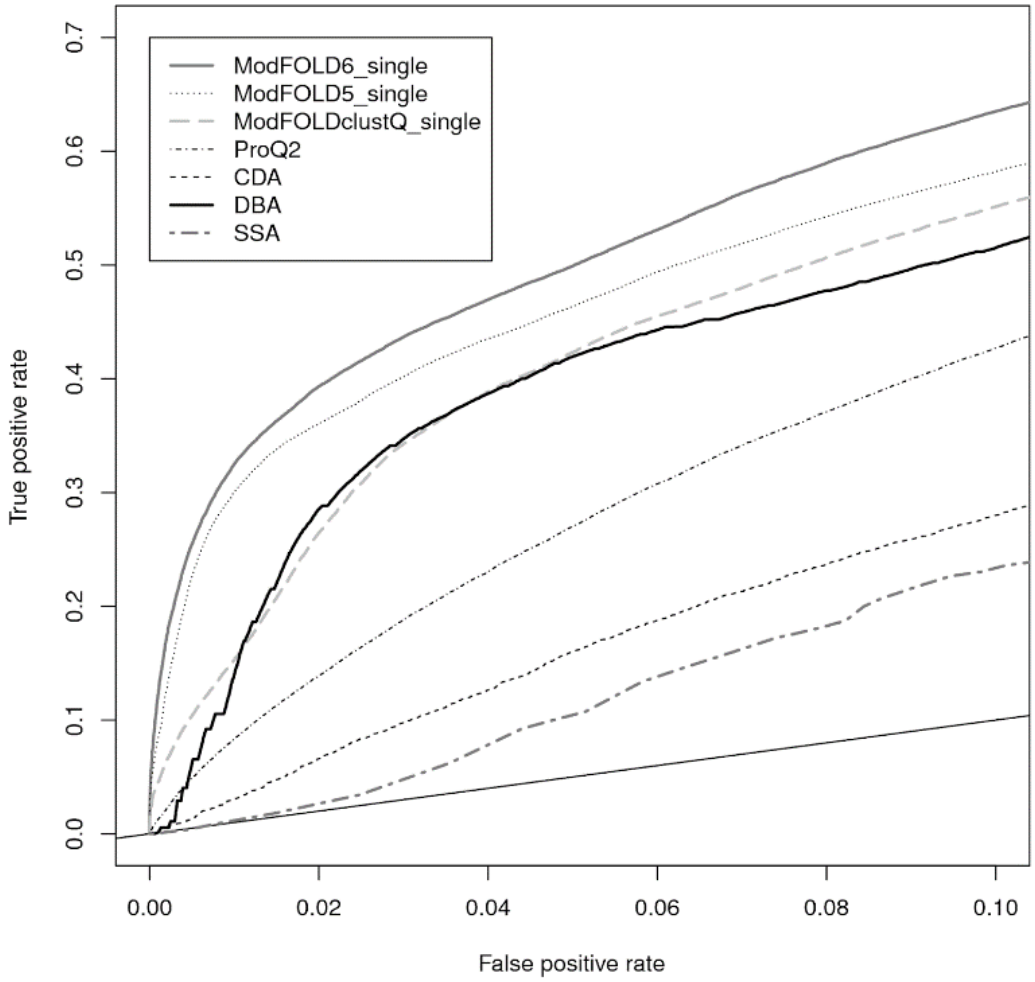

Figure 2.3. Line graph representing a benchmark of ModFOLD6 local scores versus its component methods using CASP11 data. The local QA scoring ModFOLD6 method was benchmarked with the constituent methods using the CASP11 data set. This comparison is performed continuously and can be checked anytime at: http://cameo3d.org/qe/.

Furthermore, the server is continuously benchmarked for local quality estimation (QE) performance using the CAMEO resource (Haas et al., 2018) (Figure 2.3). Our internal benchmarks and the independent CAMEO data shows that ModFOLD6 and another QE method called QMEANDisCo are currently the leading public QA method for producing local scores.

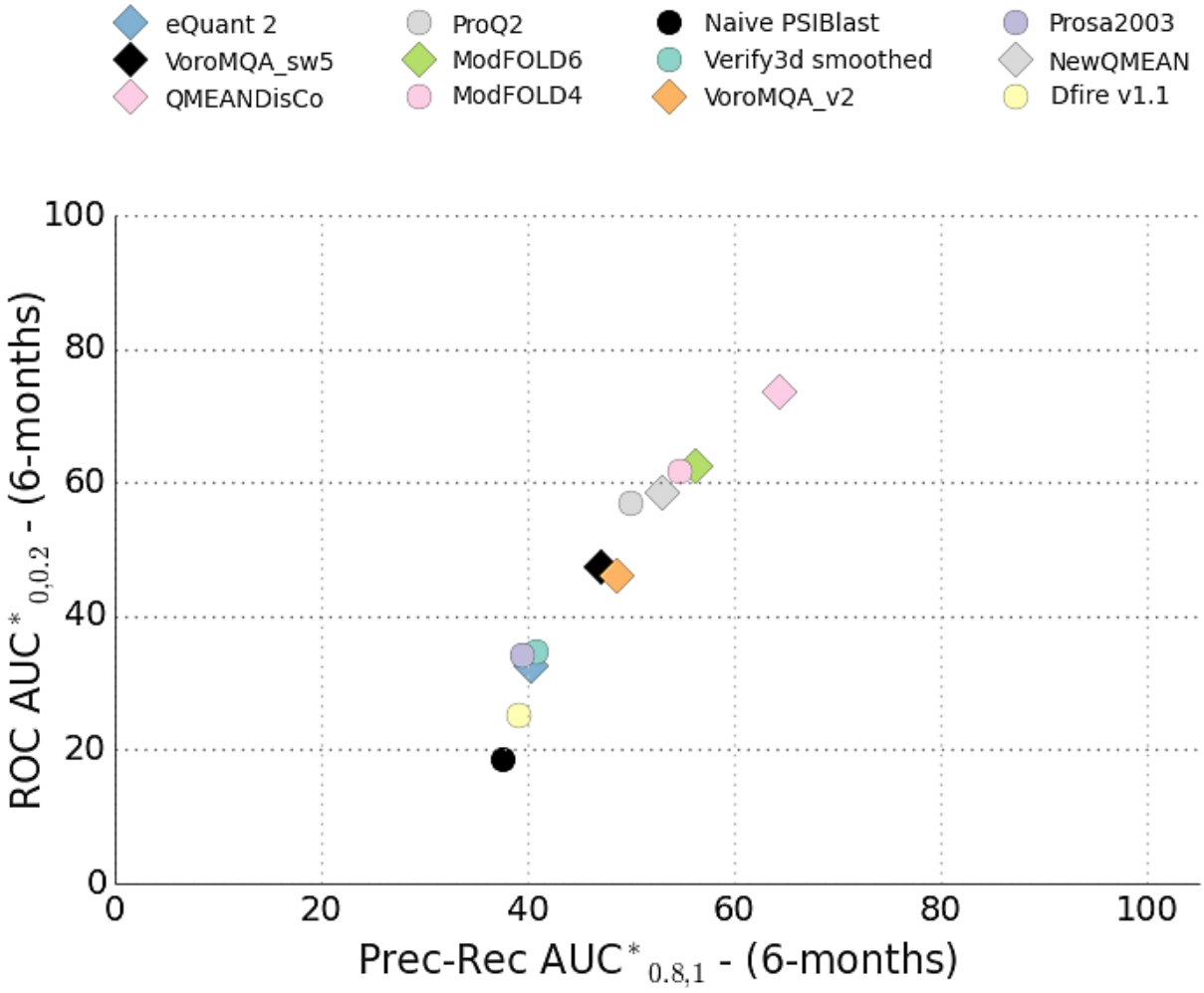

Figure 2.4. Dot plot demonstrating a six-month performance summary for competitive local QA programs including the previous version of our program (ModFOLD4) and ModFOLD6. The evaluation was carried out using the receiver operating characteristic (ROC) and Precision vs Recall analysis. This comparison is performed continuously and can be checked anytime at: http://cameo3d.org/qe/.

Such results indicate that the correlation scores of the selected combinations have reached their maximum optimisation in terms of both linear and multiple linear regression. However, this is still the beginning of this research study. Further investigation and analysis will be performed in order to find the way to improve the accuracy and consistency of ModFOLD6. One suggested way is by using machine learning techniques with the help of the advanced deep artificial neural network. Alongside with energy/scoring functions and consensus methods, DANNs has been addressed to be another supportive method for improving the QA of model prediction (Nguyen *et al*., 2014). Deep learning is a branch of machine learning techniques that has made major advances in solving many problems including the ones in image recognition (LeCun et al., 2015).

2.4.3. New approach to update ModFOLD6

Since CASP7 first included the Quality Assessment (currently termed as Estimate of Model Accuracy or EMA) category, predictors have been persevering to develop the best Model Quality Assessment Programs (MQAPs) for estimating the absolute quality of protein 3D models with the highest accuracy. Numerous strategies and methods have been devised in order to increase the accuracy of the best model selection as well as the consistency of MQAP output scores with observed scores. Accuracy of selection was often found to be better with the methods that follow the pure-single approaches whereas consistency of scores has historically been greater to be with the consensus/clustering methods. Since its development, ModFOLD6 was used as our latest version of EMA method in every MQAP competition. The program was designed to balance both model selection accuracy and score consistency. Such a balance can be achieved by combining the pure-single model with consensus/clustering-based methods. However, in this study, the current version of ModFOLD6 can be updated with a suggested hybrid pure-single/quasi-single strategy that can be adopted and swapped with the clustering method.

2.4.3.1. Suggested component of per-residue/local similarity scoring methods for ModFOLD6

Our initial emphasis was on increasing the accuracy of per-residue assessments for single models. Three pure-single model methods were suggested to be parts of ModFOLD6 components, these methods include ProQ2 (Wallner & Elofsson, 2007) and 2 newly developed methods, the Contact Distance Agreement (CDA) score using MetaPSICOV (Jones *et al*., 2015) and the Secondary

Structure Agreement (SSA) score using PSIPRED (Buchan *et al*., 2013). Additionally, a set of 130 reference 3D models (generated by IntFOLD4) were also suggested to score models using three alternative quasi-single model methods, the newly developed Disorder B-factor Agreement (DBA) score using DISOPRED3 (Jones & Cozzetto, 2015), the ModFOLDclust_single_res score (Mcs) and the ModFOLDclustQ_single_res score (McQs). A simple neural network was then used to combine the component per-residue/local quality scores from each of the six alternative scoring methods, resulting in a final consensus of per-residue quality scores for each model (Figure 2.4).

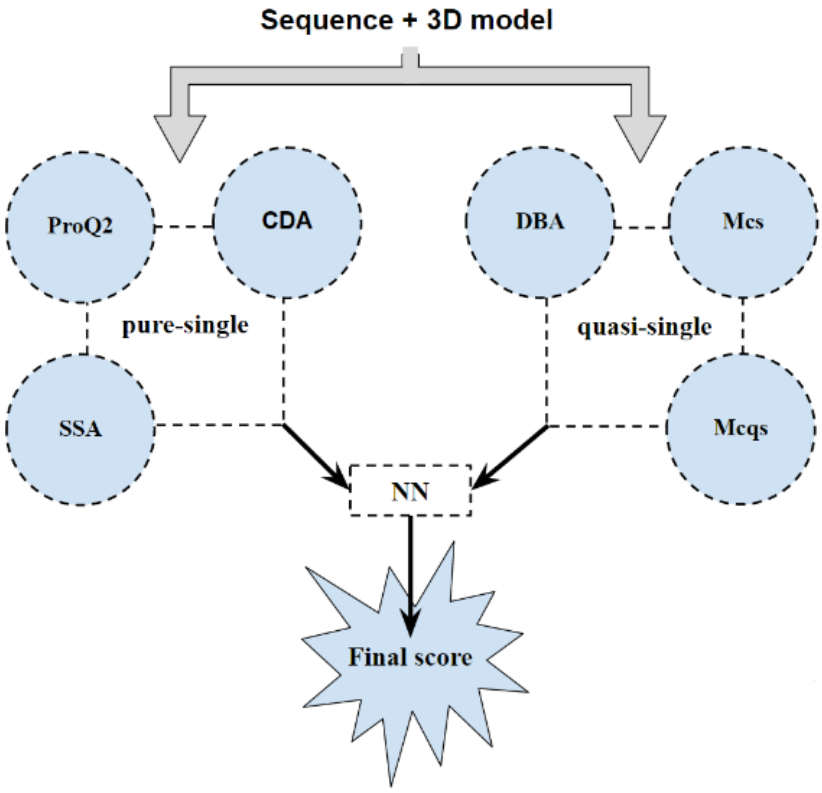

Figure 2.5. Flowchart simplifying the procedure of the local/per-residue similarity scoring method suggested for ModFOLD6. The process used three pure single model methods and three quasi-single model methods (130 reference models are generated from sequence using IntFOLD4).

2.4.3.2. Suggested component of global scoring methods for ModFOLD6

Global scores were calculated by taking the mean per-residue scores (the sum of the per-residue similarity scores divided by sequence lengths) for each of the 6 individual component methods, described before, and the consensus output (ModFOLD6). Furthermore, 3 additional quasi-single

global model quality scores were generated for each model based on the original ModFOLDclust, ModFOLDclustQ and ModFOLDclust2 global scoring methods (in a similar vein to the ModFOLD4_single and ModFOLD5_single global scores, tested in CASP10 and CASP11 respectively). Thus, we ended up with a total of 10 alternative global QA scores, which could be combined in various ways in order to optimise for the different aspects of quality estimation.

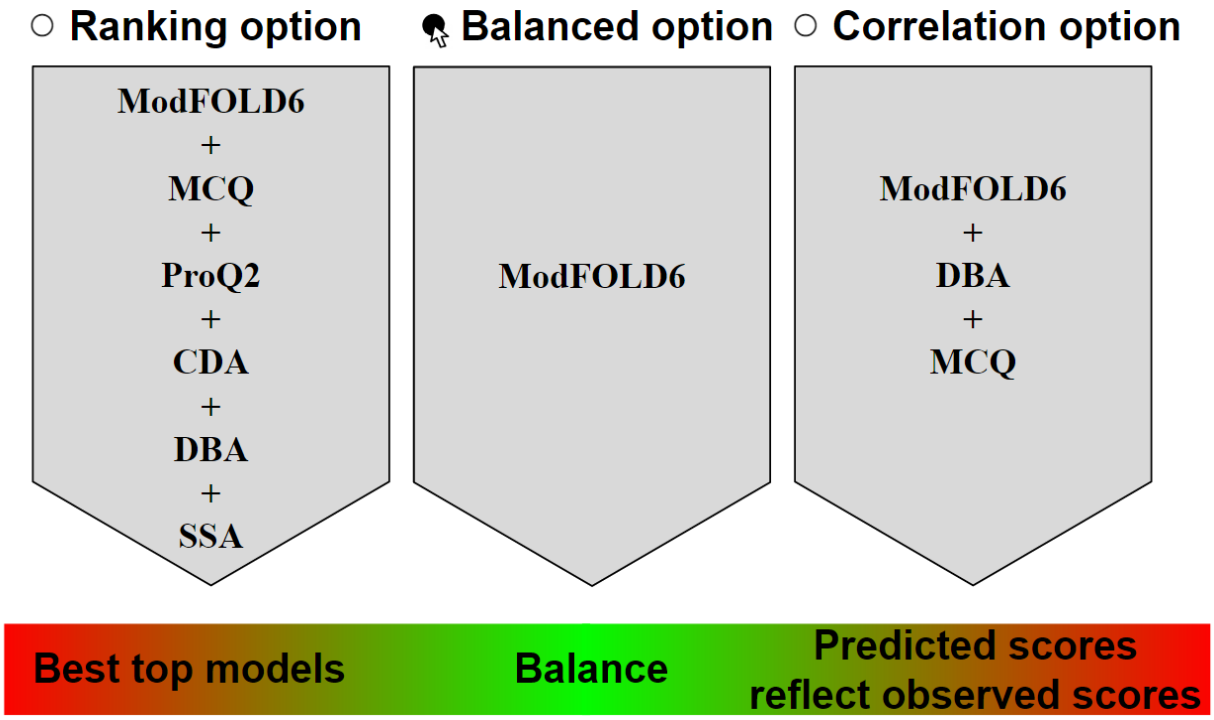

Figure 2.6. Diagram representing the three suggested options of ModFOLD6 global scoring variants.

Three ModFOLD6 global scoring variants were suggested for evaluation (Figure 2.5). (1) Balanced, The ModFOLD6 mean local scores considered alone which have a good balance of performance based on correlations of predicted and observed scores and rankings of the top models; (2) Correlation, The ModFOLD6_cor global score variant was found to be an optimal combination for producing consistent correlations with the observed scores, i.e. the predicted global quality scores should produce closer to linear correlations with the observed global quality scores; (3) Ranking, The ModFOLD6_rank global score variant was found to be an optimal combination for ranking, i.e. the top ranked models (top 1) should be closer to the highest accuracy, but the relationship between predicted and observed scores may not be so linear.

2.5. Conclusion

An initial study was undertaken to evaluate ten performing model quality assessment methods including ModFOLD6. Statistical analysis showed that some methods like ModFOLD6 were giving high scores individually.

However, it was seen that some combinations of the ten MQAP methods were producing better scoring results than single methods. Thus, the methods were benchmarked as combinations, and the results showed that ModFOLD6 method were able to provide some improvement in terms of accuracy as well as consistency when combining the three pure-single methods (ProQ2, CDA and SSA) as well as the three quasi-single methods (DBA, ModFOLDclust_single_res and ModFOLDclustQ_single_res) to the component of per-residue/local similarity scoring methods. It was also noticed that the method can be improved by using different weightings. Combining some methods can improve ModFOLD6 in selecting the top 1 model and combining some other methods can improve ModFOLD6 in correlating predictions with observed scores. Therefore, it was suggested that we provide 3 alternatives (ModFOLD6, ModFOLD6_rank and ModFOLD6_cor) for ModFOLD6 accordingly.

Finally, we found that some ranking techniques, such as regressions, have helped us in finding some increase in the scores of our methods. We managed to gain ~ 2% improvement from our method by combining and then benchmarking them using linear regressions. Such an improvement led us to suggest a new strategy for updating ModFOLD6 which is to hybridise it to become a pure-single/quasi-single EMA method.

However, as different weightings were found to be effective, we can then look for a way to calculate scores more accurately. Instead of calculating only the mean scores from all combinations together, we need to find an approach that can take the needed number of scores from some methods and neglect another number of scores from others. We need ModFOLD6 to be able to calculate its input scores based on which of the component MQAP methods are more important in certain times in order to output some more accurate scores. One suggested technique that can apply this approach is the Deep Artificial Neural Network, which was our focus in the next parts of this research project.

Chapter 3

Integrating Two Deep Artificial Neural Networks (RSNNS & TensorFlow) for Optimising the Local and Global score of ModFOLD6

3.1. Background

The invention of aeroplanes was an inspiration of birds, Velcro was also invented by the inspiration of burdock plants, and many other innovations have been inspired by nature. When researchers looked at drawing inspiration from the workings of the brain nervous systems, novel machine learning algorithms were designed and termed as Artificial Neural Networks (ANNs) (Hecht-Nielsen, 1988). However, although some researchers argued that we should drop exactly the whole biological analogy altogether, inspiring something does not mean copying the exact same thing, for example the invented airplanes do not have to flap their wings (Cao, 2014).

The simplest definition of Artificial Neural Networks (ANNs) was provided by the inventor of one of the first neurocomputers, Dr Robert Hecht-Nielsen. He defines ANNs as: *"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs"* (Caudill, 1987). ANNs can be identified as computing systems in their very core of Deep Learning which have been versified and became a very powerful and scalable technique that can make them ideal in tackling large and highly complex Machine Learning tasks, such as powering speech recognition services like Google Assistant and Apple Siri, classifying billions of images like Google images, recommending the best videos to watch to millions of users like YouTube, or learning a machine to be able to beat the world champion at a game like the GO champion who lost against the DeepMind's AlphaGo machine.

3.1.1. History

The development of these computing systems began in the 1940s, when Artificial Neural Networks emerged by McCulloch and Pitts who came up with the idea after their analysis of how human brains works (McCulloch and Pitts, 1943). Since that time, researchers in this field started to mimic the mechanism of neurons in the brain by modelling simple neural networks using electrical circuits. After McCulloch and Pitts hypothesis by 7 years, a work was published by Donald Hebb pointing out the fact that neural pathways are strengthened when they are in the usage status by the human body (Hebb, 1949). His argument was stating that having 2 nerves fire at the same time can enhance the connection between them. Such a concept was fundamentally essential as it gave an insight to the ways in which humans learn.

In the 1950's, computers became more advanced, and it was finally possible for scientists to test the Artificial Neural Networks (ANNs) (Priddy and Keller, 2005). The first attempt was conducted by Nathaniel Rochester from the IBM research laboratories. Nathanial's trial was stepping towards building a simulated human brain, but unfortunately, that attempt was failed in doing so. Later in 1959, two models were developed by Bernard Widrow and Marcian Hoff, they called them "MADALINE" and "ADALINE" relating to the use of the Multiple Adaptive Linear elements (Widrow, 1960). These two models were the first ANNs being applied to solve real-world problems. "MADALINE" was developed for using an adaptive filter which eliminates echoes on phone lines, while "ADALINE" was developed for recognising binary patterns so that it can predict the next parts of a read streaming section from phone lines.

In 1962, another model was developed by Windrow and Hoff, it was a learning procedure technique which examines the value before the weight adjusts it (i.e. 0 or 1) according to the rule: Weight Change = (Pre-Weight line value) * (Error / (Number of Inputs)). The model was developed based on the assumption that while one active perceptron may receive a big error, one can adjust the weight values to distribute it across the network, or at least to adjacent perceptrons. The results of this equation still show an error whenever the line before the weight is 0, and eventually the error gets corrected automatically. However, the error gets eliminated when it is conserved which means it is distributed to all the weights. Although these models were developed using old fashion techniques, they are still in commercial use.

After such revolutionary achievements, the success in applying the ANNs technique started to expand more and more until the beginning of the 1970s, when it was confronted with the traditional von Neumann architecture which took over the computing scene leaving the neural networks technique behind (D'Addona, 2016). Ironically, one of the suggestions that John von Neumann gave it himself was the imitation of neural functions by using telegraph relays or vacuum tubes. In the same period, a number of other works reported that there could not be an extension from the single layered neural network to a multiple layered neural network. Such suggestions at that time have led to a sharp decrease in funding the ANNs research.

After funding flew elsewhere, ANNs entered a long dark era until the early 1980s when new network architectures were invented, and better training techniques were developed. A paper was presented to the National Academy of Sciences by John Hopfield showing a renewal interest in the neural networks field (Hopfield, 1982). The approach was focusing on creating more useful

machines by using bidirectional lines rather than having only one way in the connections between neurons as the previous methods. Following that approach, a "Hybrid network" was utilised by Reilly and Cooper with the feature having multiple layers and each layer uses a different problem-solving strategy (Reilly et al., 1982). In the same year also, a great conference on Cooperative/Competitive Neural Networks was held by USA and Japan (Amari and Arbib, 1982) announcing a new fifth generation effort on neural networks as this technique went through 4 states before. The first state when NNs used switches and wires, the second state used the transistor, the third generation used solid-state technology such as integrated circuits and higher-level programming languages, and the fourth state was the use of code generators. The fifth generation announced in the joint US-Japan conference was from Japan, and it involved the artificial intelligence. This announcement from Japan made US worries about being left behind, and therefore, funding started to flow once again. In 1985, an American institute in Physics began an annual meeting in Neural Networks for Computing, and in the third time the meeting was held in the IEEE as the first international conference on NNs drawing more than 1800 attendees.

During the following years, the multiple layered NNs concept was spread, and the main issue at that time was in finding the way to extend the Windrow-Hoff rule to multiple layers. In 1986, a group of researchers from Stanford's psychology school came up with similar ideas which are now called propagation networks as it distributes pattern recognition errors throughout the network (Tanaka et al., 1986). In contrast to Hybrid networks which used only two layers, the back-propagation networks were able to use more than two layers. However, at that time, back-propagation networks were found to be a "slow learner" which needed possibly thousands of iterations to learn, and that was one of the problems which NNs faced during that time.

By the 1990s, other powerful machine learning techniques such as Support Vector Machines started to show up, making researchers changing their minds in using NNs as they seemed to offer better results and stronger theoretical foundations with other techniques (Gholami and Fakhari, 2017). However, another wave of interest in ANNs has been witnessed recently, and it seems that this wave is not going to disappeared like the previous waves. Few reasons to believe that Artificial Neural Networks are not going to put down again. Firstly, the huge quantity of available data that can be used for training neural networks making ANNs frequently outperform all other ML techniques on very large and complex problems. Secondly, the tremendous increase in computing power that can now make it possible for computers to train large NNs in a reasonable amount of

time. Thirdly, the improvement in the training algorithms which has a huge positive impact. Fourthly, solving many ANNs limitations such as the worries of getting stuck in local optima when it turns out that this is rather rare in practice. Fifthly, the huge amount of funding and progress that this technique is having in the many different areas. Up until now, the NNs technique has been evolving through various applications which have been practicing it thoroughly (Géron, 2017).

3.1.2. Biological Neurons

Before discussing artificial neurons, we should have a look at the biological structure of neuron itself (Figure 3.1). The biological neuron is an unusual-looking cell which is found in animal cerebral cortexes composing of a cell body which contains the nucleus and most of the cell's complex components, and many branching extensions called dendrites, plus a longer extension called the axon. The axon splits off into branches called telodendria, the tips of these branches are to a minuscule structure called synaptic terminals, these are connected to the dendrites of other neurons. When a biological neuron receives a sufficient number of an electrical impulse (called a signal) from another neuron via these synapses within a few milliseconds, the neuron fires its own signals as well (Seikel et al., 2018).

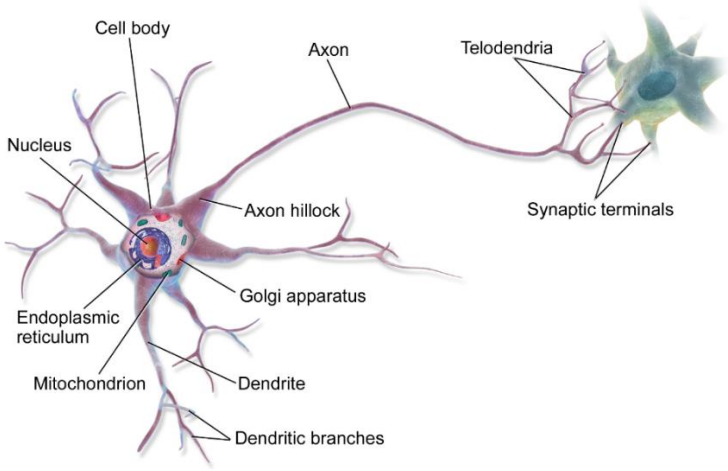

Figure 3.1. 3D drawing of the biological structure of a neuron. Adapted from Géron, (2017).

The behaviours of an individual biological neuron seem to be simple. However, biological neurons are organised in a vast network of billions of neurons connecting with each other building an architecture of biological neural networks (BNNs) that can solve highly complex computations. The architecture of BNNs is still the subject of active research. However, studies have managed to map some parts of the brain showing that neurons are often organised in consecutive layers (Figure 3.2) (Géron, 2017).

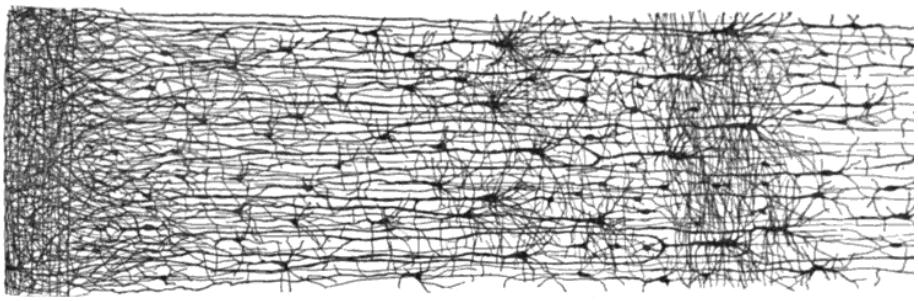

Figure 3.2. Drawing represents the consecutive layers construction of neurons in the brain. Adapted from Géron, (2017).

3.1.3. Artificial Neurons

The first simple model of the biological neuron was proposed by Warren McCulloch and Walter Pitts, the model has one or more binary (on/off) inputs and one binary output. When more than a certain number of the model's inputs are active, the model simply activates its output. This model was seen to mimic the biological neuron (Figure 3.3), and therefore, it was later termed as an artificial neuron. By this simple architecture of the artificial neuron, McCulloch and Pitts showed that it is possible to build a complex Artificial Neural Networks which can compute any logical proposition we want (Browne, 1997).

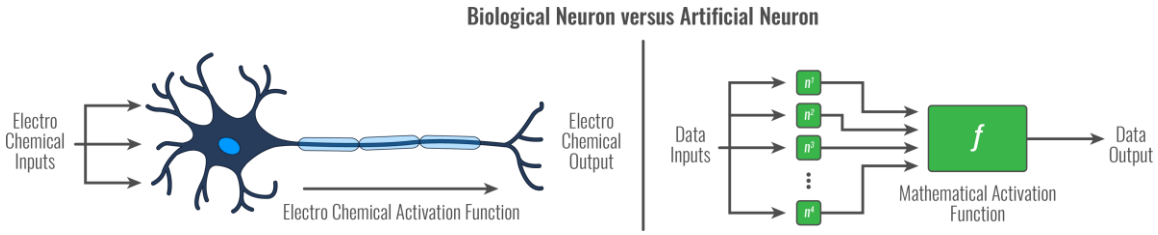

Figure 3.3. Schematic drawing representing an analogy of Biological Neuron and Artificial Neuron. Adapted from Géron, (2017).

The building block unit and the first implementation of an ANNs was known as the perceptron (Rosenblatt, 1957), a single layer of linear threshold units with nodes (neurons) connected to all the inputs. Each node contains an activation function turns to be activated once that node receives enough number of signals from the previous interconnected nodes. In a perceptron network, each neuron is a linear threshold unit, and the inputs and output are considered as numbers rather than on/off binary values. Each input is connected with other outputs of previously located neurons (except the first inputs). Between these neurons there are connections associated with weights. The linear threshold unit computes the weighted sum of its inputs as $z = w_1x_1 + w_2x_2 + \cdots + w_nx_n = w^T.x$ and then applies a step function for that sum to be outputted as $h_w(x) = step(z) = step(w^T.x)$. An example in Figure 3.4 clarifies the architecture of this feedforward technique. This example has 3 inputs, each of which has its own weight. These inputs are summed by the linear combiner ∑ and then put through a function.

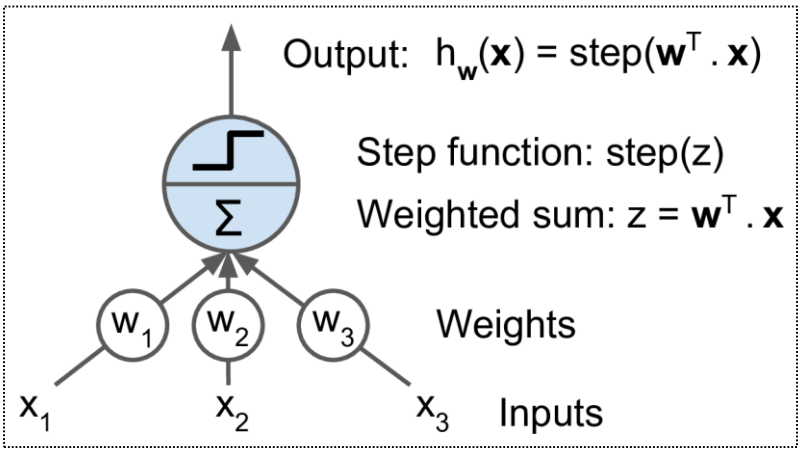

Figure 3.4. A diagram representing a linear threshold unit.

There are several step functions that can be used in Perceptrons, the most common ones are the sgn step function and the Heaviside step function, they are calculated as:

$$sgn\,(z) = \begin{cases} -1\ if\ z < 0 \\ 0\quad if\ z = 0 \\ +1\ if\ z > 0 \end{cases} \qquad heaviside\,(z) = \begin{cases} 0\ if\ z < 0 \\ 1\ if\ z \geq 0 \end{cases}$$

3.1.4. Deep Neural Networks

A single linear threshold unit can be used for simple linear binary classification. It can compute a linear combination of the inputs and if the result exceeds a threshold, it outputs the positive class or else outputs the negative class as similar as a Logistic Regression classifier or a linear SVM.

However, processing multiple linear threshold units in many layers of perceptrons constructing what we may call it Deep Artificial Neural Networks. DANNs can compute more advanced problems with several techniques and architectures to be formulated, the multilayer perceptron (MLP) has been the best feedforward neural network class that has the ability to map a set of inputs which pass it through hidden layers and send the calculated data to an output unit (Rosenblatt, 1962). MLP networks have been considered as a powerful technique in a large number of applications from different fields of research. The benefits of MLPs come from the appropriateness in dealing with most of the problems involving function approximation, pattern classification, process control and time series forecasting (Efendigil et al., 2009).

Intuitively, we can expect that having more hidden layers would make our networks more powerful. The single layer can be changed to multiple-layered artificial neural networks. This approach was built up and found to have more complex intermediate layers which can have multiple layers of abstraction (Ba and Caruana, 2014). Having multiple layers can give neural networks the ability to solve more advanced challenges such as visualising pattern recognition. Eventually, this type of NNs was referred as the Deep Artificial Neural Networks (DANNs).

The DANNs was found to spot the correct mathematical manipulation more accurately and turn the input into the output whether in a linear or a non-linear relationship. Unlike ANNs, the multiple layers in the DANNs give it the ability to process more complicated problems (Toth et al., 1996). By testing the visualising pattern recognition example using DANNs, the neurons in the first layer could learn recognising edges, then the neurons in the second layer would learn recognising more shapes like triangles or rectangles which are built up from edges which already been learnt in the first layer. The third layer could then recognize static more complex shapes, and the fourth learns animatic shapes, and so on. This reminds us by how children start to learn basic shapes around

them when their brains which contain multiple layers of neurons give them a compelling advantage in starting to learn complexes pattern.

The differences between ANNs and DANNs lies in the depth of the model. The phrase Deep learning is a term that has been used for more advanced artificial neural networks which contain multiple processing layers. With this level of layers, networks will be able to create more space for processing a huge amount of data. Figure 3.5 can show an example of how DANNs architectures are highly complicated compared to ANNs. Such a complexity in DANNs is attributed by elaborate patterns of how data can flow throughout the model.

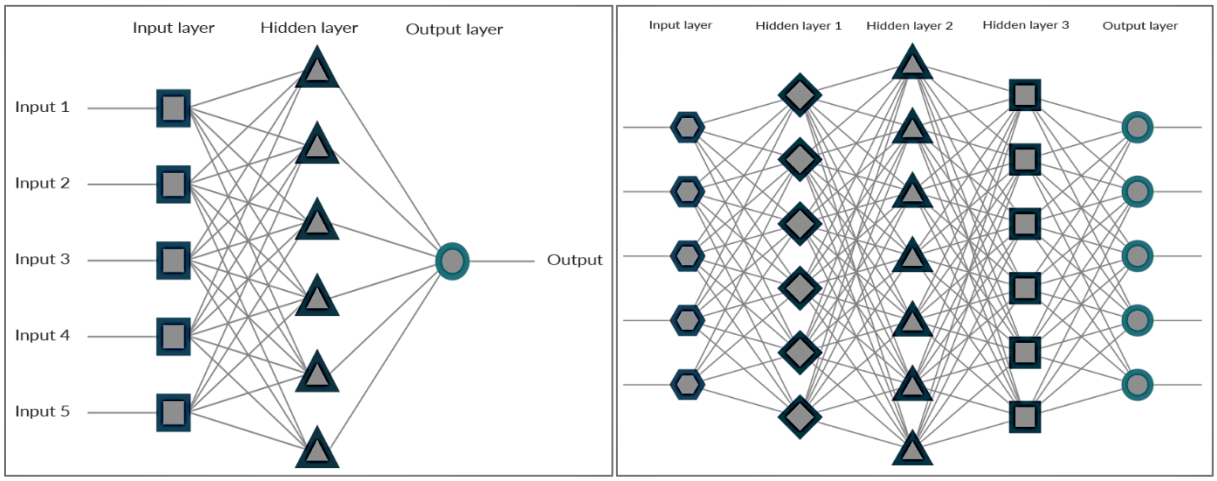

Figure 3.5. Two diagrams illustrating the differences between ANNs (left panel) and DANNs (right panel).

3.2. Objectives

The 6th version of ModFOLD described in the previous chapter is powered with different model quality assessment programs (MQAPs) which produce a combination of model quality scores. These scores are usually averaged to assign an Optimal Mean Score (OMS) for evaluating the quality of a model. The OMS is a simple equal weighting consensus approach, which means that ModFOLD6 assumes that all EMA scores used in the combination are equal under all circumstances, however this can seem practically inaccurate. Sometimes, some EMA methods scores can be relied on more than other scores. In the free modelling situation for example, the scores produced by the pure-single methods such as CDA, SSA and ProQ2 should be more considered than the scores produced from the quasi-single methods. Therefore, it was important to

look for a way to overcome such an issue.

This chapter demonstrates an introduction to the Deep Artificial Neural Networks for a try to combine scores more accurately than OMS by using different weightings for different score combinations. The training process was automated in a way which allowed training with multiple combinations and neural networks parameters in a short amount of time. Two packages were utilised for constructing DANNs. For a "shallow" DANNs, we used the RSNNS package from the statistical computing software, R v3.2.3 (Bergmeir and Benítez, 2012). While for a more advanced DANNs, we used Google's TensorFlow v1.0 package in Python v2.7.5 (Abadi et al., 2016). The results from this study show that both DANN packages are useful at different levels, RSNNS derived networks were found to be the best method when testing for correlations with observed scores while the TensorFlow deep neural networks outperformed all other techniques when testing the method's ability to pick the top ranked model.

Indeed, like all predicted scores, the Quality Assessment (QA) scores vary in accuracy when compared to the true score of the predicted model. The OMS technique does not take this into account as it assumes that all scores being combined are of the same accuracy. The work in this chapter aims to create and train a deep neural network that can produce global scores which are more accurate at model ranking/selection and produce more consistent output scores than the standard ModFOLD6 global score. This will be achieved by exploring different feed forward deep neural network architectures for combining scores using various optimal weightings. This Chapter also aims to compare the performance of the deep learning NNs to the standard NNs in order to discover which technique can produce the optimum combined QA scores.

3.3. Materials and Methods

3.3.1 Inputs and Outputs

Ten individual MQAPs scores were chosen to be the MLPs inputs while the output data were the GDT-HA (Mirjalili and Feig, 2013) scores. Additionally, for benchmarking purposes, the GDT (Zemla et al., 1999), MaxSub (Siew et al., 2000) or TM-scores (Zhang and Skolnick, 2004), were also chosen as the observed quality measures. The initial MLP construction was basic, consisting of 3 layers; an input layer, a hidden layer and an output layer. More complex systems can have multiple hidden layers. The number of nodes (similar to neurons in biology) within a layer can also

vary. Much like in the nervous system, NNs worked by feeding information into the input layer which can be seen as the sensory neurons (Rosenblatt, 1962). The inputted data was then fed into the hidden layers in which the information was interpreted; this is where weights were assigned and could be seen as the interneurons. The interpreted information was then moved on and collected at the output layer (which could be analogous to a motor neuron) where all the data was then interpreted to be ready for outputting as results. MLPs also utilise a supervised learning technique called backpropagation for the training of the network. Learning occurs through the changing of the weights after each piece of data is processed in accordance with the amount of error in the output compared to the observed scores (Rumelhart, et al., 1986).

In this section, two alternative NN packages were investigated. Both were MLPs however, they varied in complexity. The first was an MLP from the RSNNS package in R (Appendix 2) (Bergmeir & Sánchez, 2012) while the second was a shallow networks version of MLP which was created for the specific task of QA score combination using tools from Google's TensorFlow package (Appendix 3). While the TensorFlow DANNs itself is built in python, the python script also utilises two R scripts for data management and analysis (Appendix 4).

3.3.2. RSNNS

The Stuttgart Neural Network Simulator (SNNS) library (Zell, et al., 1994) with its many standard implementations of neural networks contained was brought through the RSNNS package into R script. The RSNNS MLP is a basic single hidden layer MLP (Bergmeir & Sánchez, 2012). It takes multiple parameters; training inputs, training outputs (GDT-HA), size (which defines the number of nodes in the hidden layer), iteration number, initialisation function, learning function, test inputs and test outputs. These parameters were varied during this project except for the initialisation function as it was unnecessary to change this value from its default when performing a high-level function.

3.3.3. TensorFlow

For implementing Deep Artificial Neural Networks, Google's TensorFlow was utilised. TensorFlow is a software library which contains tools for numerical computations using data flow graphs. TensorFlow was developed with the aim of creating advanced tools for pattern prediction and correlation (Abadi, et al., 2016). TensorFlow uses nodes or tensors which are formed in multidimensional arrays in its DANNs. Our TensorFlow based MLP was designed with 2 hidden layers, however, more can be used. In addition, multiple neural networks parameters were configurable including training/testing inputs (number), training/testing outputs (e.g. GDT-HA), the learning rate, iterations, size of the first hidden layer, size of the second hidden layer. These parameters were varied and evaluated. With the utility of TensorFlow we used optimisation algorithms and activation functions in order to mitigate several DANNs problems including hidden layers and units separation and activation. One optimising algorithm used in our DANNs was the AdaGrad (Duchi et al., 2011), an algorithm which adaptively scales the learning rate for each dimension. The equation for the parameter update which have been used in practice was: $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\varepsilon I + \mathrm{diag}(G_t)}} . g_t$, where θ is the parameter to be updated, η is the initial learning rate, ε is a small quantity which are used to avoid the division of zero, I is the identity matrix, G_t is the gradient estimate in time-step t that can be solved in further equations. The dropout function was also used in our DANNs. This regularising tool is often very effective at reducing overfitting. It works by dropping neurons based on the probability, which is defined by the user (a 90% keep probability was used in this study), that a neuron's output is kept during dropout. This allows for dropout to be turned on during training and turned off during testing.

3.3.4. Neural Networks insertion using Multi-Layer Perceptron machine learning method

After obtaining results from all the ten baseline MQAP methods, the Multi-Layer Perceptron techniques were then inserted. Two types of MLPs were utilised, a basic MLP which was obtained from the RSNNS package in R, and a more advanced MLP using TensorFlow. Both techniques were evaluated and parameterised in order to find a suitable NNs pipeline for EMA scoring.

3.3.4.1. Neural Networks Setup

The NN scripts were written using the programming language, Python v2.7.5. The initial scripts were able to run the NNs with manually adjustable parameters. Subsequently, the parameterisation was automated using an updated DANNs script (Appendix 5) for the purpose of accelerating the research speed.

3.3.4.2. Neural Networks Parameterisations

When we managed to automate our DANN script, it became possible for us then to parameterising the iterations and number of hidden nodes per layer. Each of the selected combinations from the previous step was then run through a neural network parametrisation stage. The DANNs parameters were changed, hidden node numbers ranged from 1 to 20 in RSNNS and 1 to 5 in TensorFlow (due to time constraints) while iterations ranged from 50 to 950, all of these adjustments were proceeded with each run (thorough investigations on DANNs parameterisation will be studied in Chapter 5). After the DANNs parameterisation stage completed, we analysed our results to look for the best combination with its optimal DANNs adjustment for both ranking and correlation scoring methods.

3.3.4.3. Data searching

After DANNs parameterisation, the script was then used to run through every possible combination for the ten MQAPs using six different sets of iterations and numbers of hidden nodes (Table 3.1). After running several DANN scripts of the six different sets for both MLPs, we ended up having a huge amount of data which was outputted as tables showing the resulted top 10 ranking and correlation methods scores of each run. However, the top 10 scores of the EMA combinations produced from these DANNs runs were varied. Some scores of methods appeared to take the 1st rank in some runs but lower ranks in some others. To exploit such an output, all tables were analysed to look for the best methods in most of the resulted data. The methods which appeared to produce ranking as well as correlation scores within the top 10 more often was chosen. The best-chosen combinations with their optimal parameters for each of the 6 sets were then run through a looped NN script (Appendix 6) aiming to achieve the highest possible model quality score. In this part of study, GDT-HA was used as the observed scoring measurement control for most of the conducted tests as it was showing better results in correlations.

Set Number	RSNNS		TensorFlow		
	Hidden neurons	Iterations	1st layer – number of hidden neurons	2st layer – number of hidden neurons	Iterations
1	3	100	2	3	550
2	Input number	100	3	2	550
3	Half Input number	100	Half input number	2	550
4	2x input number	100	3	2	100
5	3	300	2	1	100
6	2x input number	300	6	2	100

Table 3.1. The parameters used in each set for both RSNNS and TensorFlow. There are 6 sets of parameters. There are 2 parameters to vary for RSNNS as it only has a single hidden layer, while TensorFlow has 3 parameters to vary as it has two hidden layers.

3.3.4.4. Data analysis

All the best outputted data of scores resulted from the combinations of the 10 MQAP methods through RSNNS and TensorFlow using the correlation and ranking scales were collected and analysed in order to look for any possible improvement to our EMA score.

3.4. Results and Discussion

In this study, the effect of implementing RSNNS and TensorFlow on our MQAP method was analysed. RSNNS and TensorFlow were applied separately to our selected combinations of MQAP methods, and an evaluation was carried out to search for any improvement through NNs process. GDT-HA, GDT, MaxSub and TM-score were the observed scoring measurements which were used as controls for all the following benchmarking analysis. Both benchmarking tests provided a number of interesting results in the local as well as the global scoring level.

3.4.1. MQAP score optimisation using RSNNS and TensorFlow

Similarly as in the study done in Section 2.3.4, all combinations of the original 10 MQAPs were considered for benchmarking individually and in combinations using the same measurements but with the integration of the MLP technique. The resulted scores from MLP integration were then

compared with the previously yielded scores from Chapter 2 in order to look for any improvement. For guidance, the following key numbers will be used to describe each of the ten EMA methods as described in Table 3.2.

Key	MQAP method
3	ModFOLD5_single_orig_global
4	ModFOLDclustQ_single_orig_global
5	ModFOLDclust2_single_orig_global
6	ModFOLD5_single_res_global
7	ModFOLDclustQ_single_res_global
8	ProQ2_res_global
9	CDA_res_global
10	DBA_res_global
11	SSA_res_global
12	ModFOLD6_single_res_global

Table 3.2. List of key numbers used to label the 10 MQAP methods.

3.4.1.1. Correlation benchmarking through RSNNS and TensorFlow

Firstly, the 6 sets of neural network parameters (Table 3.1) from RSNNS and TensorFlow were adapted to benchmark the ten MQAP methods following the correlation scale. The results of this benchmarking were then analysed to rank the best combination of MQAP methods which can produce the optimum QA score. The analysis was carried out based on the performance consistency between the ranked combinations throughout the 6 sets of NNs parameters. This means that with the RSNNS inclusion, the combined methods which produced the highest scores in most of the six parameters were selected. These methods were ranked and denoted as the highest combination of methods in appearance (Figure 3.6).

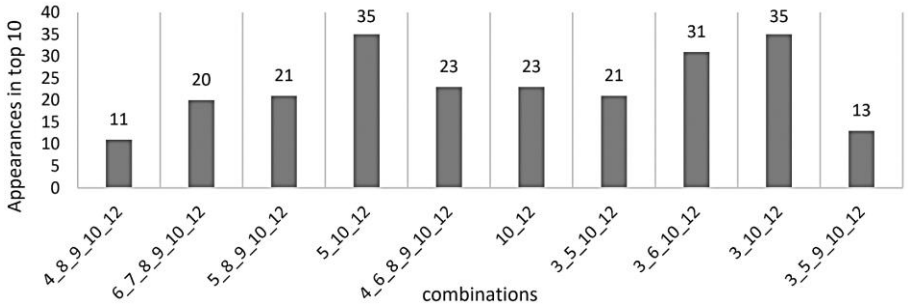

Figure 3.6. Bar chart representing the top 10 MQAP combinations for correlation through RSNNS. The ranking was carried out based on the appearances in each set of the NNs**.**

Further, all combinations were subjected to the same statistical analysis again, but this time for finding only the combination of MQAP methods which produce the highest correlation scores. Table 3.3 represents the output of this analysis showing differences in the correlation scores. From these results, it can be noticed that the top combinations were diversified due to the variations of the used correlation coefficients as well as our measuring controls. However, such a diversity did not occur when the predicted scores were correlated against GDT-HA. This can indicate the likelihood that the neural network was trained on the GDT-HA scores, thus making it a more reliable measurement for correlation. Therefore, further evaluations in this section focused more on regarding GDT-HA as a control, but also considering the other observed measurements.

Correlation Coefficients	Observed Measure	Correlation Score	Combination
R	GDT-HA	0.9223	4_8_9_10_12
Rho		0.9374	4_8_9_10_12
Tau		0.7769	4_8_9_10_12
R	GDT	0.9305	6_7_8_9_10_12
Rho		0.9391	3_5_9_10_12
Tau		0.7847	3_5_9_10_12
R	MaxSub	0.9343	5_8_9_10_12
Rho		0.9393	6_8_9_10_12
Tau		0.7841	5_10_12
R	TM-score	0.9289	5_8_9_10_12
Rho		0.9375	3_5_9_10_12
Tau		0.7861	5_10_12

Table 3.3. The top combinations and their scores for each testing method in the combination stage in RSNNS.

Additionally, by looking at the results, we can notice that one of the top combinations (6_8_9_10_12) did not appear within the selected highly performing combination methods in most of the six sets of the RSNNS parameters demonstrated in Figure 3.6. However, 6_8_9_10_12 was still one of the combinations which gave the highest correlation scores, thus it will be taken with the selected methods to be tested in the NNs parameterisation stage.

In the parameterisation stage, the selected combinations of MQAP methods were tested through several RSNNS runs to look for the most suitable NNs parameters that can give the highest correlation scores. As can be seen in Table 3.4, 12 MQAP combinations were chosen as the highest RSNNS correlation scoring methods. The optimum combination was for 6_8_9_10_12 with it achieving the top score in 6 of the 12 selected methods, while 4_8_9_10_12 correlated best with GDT-HA achieving the top score in all 3 correlation coefficients. For combination 6_8_9_10_12 the parameters of 2 hidden units and 800 iterations were chosen as it was the top performer in 5 of the 6 times in which 6_8_9_10_12 was the highest scoring combination. For the 4_8_9_10_12 combination, the parameters of 2 hidden units and 100 iterations were chosen due to the large increase in Pearson's R. These selected combinations with their suitable NNs parameters were then taken forward for a further analysis.

Correlation Coefficient	**Observed Measure**	**Correlation Score**	**Combination**	**Parameters**	
				Hidden units	**Iterations**
R	GDT-HA	0.9234691	4_8_9_10_12	2	100
Rho		0.937887	4_8_9_10_12	6	100
Tau		0.7773133	4_8_9_10_12	6	100
R	GDT	0.9312651	4_8_9_10_12	2	100
Rho		0.9410867	6_8_9_10_12	2	800
Tau		0.7867924	6_8_9_10_12	2	800
R	MaxSub	0.9340363	5_8_9_10_12	2	150
Rho		0.9425893	6_8_9_10_12	2	800
Tau		0.7861622	6_8_9_10_12	2	800
R	TM-score	0.9317801	6_8_9_10_12	2	250
Rho		0.9408387	6_8_9_10_12	2	800
Tau		0.7871806	5_10_12	4	500

Table 3.4. The top combinations and the respective parameters for each correlation testing methods for RSNNS.

For TensorFlow, the same steps of NNs runs using the 6 sets of parameters were used in the combination stage. As with RSNNS, the top 10 combinations were benchmarked based on the methods performance consistency which led them toward the appearances in the top 10 scores in most of the results, the top 10 combinations can be seen in Figure 3.7.

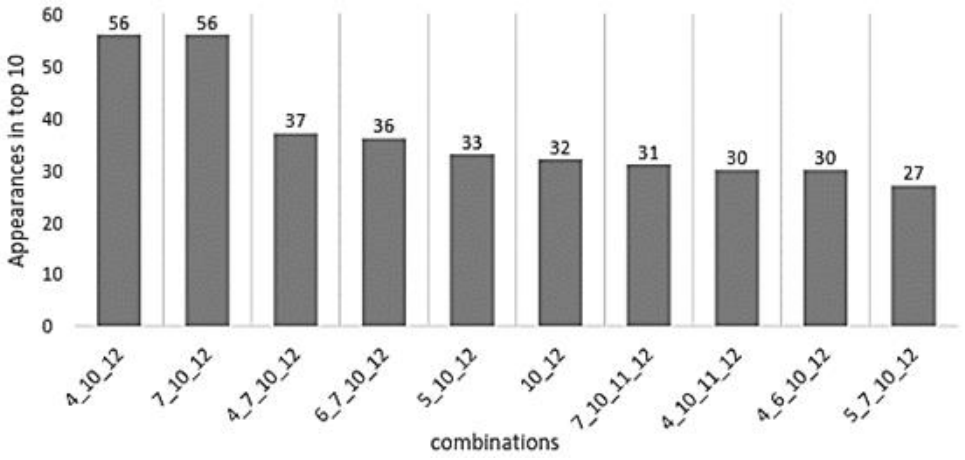

Figure 3.7. Bar chart representing the top 10 MQAP combinations for correlation through TensorFlow. The ranking was carried out based on the appearances in each set of the NNs**.**

As the RSNNS procedure, statistical analysis was applied for all combinations to look for the optimal EMA combinations with can produce the highest scores. The top correlation scores for each of the selected combinations are represented in Table 3.5. Similarly as in the RSNNS results, one combination (4_10) from the TensorFlow ranked methods was found to disappear in the top correlated TensorFlow combinations. Yet, the combination method was also used in the parameterisation stage along with the top correlated methods.

Correlation Coefficient	Observed Measure	Correlation Score	Combination
R	GDT-HA	0.9103466	4_10
Rho		0.9307373	4_7_10_12
Tau		0.7708717	4_7_10_12
R	GDT	0.9260703	7_10_12
Rho		0.9321401	7_10_12
Tau		0.7820457	7_10_12
R	MaxSub	0.9295573	7_10_12
Rho		0.9347715	7_10_12
Tau		0.7807776	7_10_12
R	TM-score	0.9302162	10_12
Rho		0.9319469	10_12
Tau		0.7839458	7_10_12

Table 3.5. The top combinations and their scores for each testing method in the combination stage in TensorFlow.

The top combinations were benchmarked using several parameters of TensorFlow DANNs this time. The benchmarked MQAP combinations were then subjected to the correlation analysis to find the optimal correlated methods. By including the TensorFlow, two combinations were found to be of interest as can be seen in Table 3.6. The combination of 4_7_10_12 MQAP methods had the best performance with GDT-HA (it came second in Pearson's Rank) but in the majority of the other observed measures 7_10_12 had the top correlation score. For this reason, both combinations were taken into the data analysis stage. For 4_7_10_12 combination, the parameter of 2 hidden units in the first layer, 2 units in the second and 50 iterations was chosen as it was the top performer in most of the times in which 4_7_10_12 was showing the highest scoring combination. For combination 7_10_12, the parameter of 4 hidden units in the first layer, 3 units in the second and 50 iterations was chosen as it was one of the top performing parameters for 7_10_12.

Correlation Coefficient	Observed Measure	Correlation Score	Combination	Parameters		
				Hidden units		Iterations
				Layer 1	Layer 2	
Pearson	GDT-HA	0.91435	4_10	5	1	50
Spearman		0.9330322	4_7_10_12	2	2	50
Kendal		0.7742101	4_7_10_12	2	3	50
Pearson	GDT	0.9299396	7_10_12	4	3	50
Spearman		0.9343262	7_10_12	4	3	50
Kendal		0.7847976	4_10_12	1	4	50
Pearson	MaxSub	0.9321558	7_10_12	1	1	100
Spearman		0.9363999	7_10_12	1	1	100
Kendal		0.7826315	10_12	2	2	50
Pearson	TM-score	0.9322891	10_12	5	1	50
Spearman		0.933866	7_10_12	1	1	100
Kendal		0.7857452	10_12	2	2	50

Table 3.6. The top MQAP combinations and the respective parameters for each correlation testing methods for TensorFlow.

It can be seen from the data presented in Table 3.7 that a “shallow” single layer MLP from RSNNS was clearly the superior method when attempting to get the optimal correlations from a combination of scores. The two RSNNS combinations had between them all the top correlation scores with each getting 6 of the possible 12 top scores. 6_8_9_10_12 was selected as the top combination as it had an average improvement of 0.76% when considering all 12 forms of testing correlation compared to 4_8_9_10_12’s 0.63%. However, 4_8_9_10_12 showed an improvement of 2% in Pearson’s R when correlated with GDT-HA, and an average of 1% improvement when using all techniques to correlate to GDT-HA, these are the largest improvements found in the study for correlation.

Correlation Coefficient	Observed Score	ModFOLD6	RSNNS		TensorFlow	
			6_8_9_10_12	4_8_9_10_12	4_7_10_12	7_10_12
Pearson	GDT-HA	0.9058867	0.9186586	0.9253479	0.9122147	0.9112993
Spearman		0.9305616	0.9366413	0.9375623	0.9328477	0.9330121
Kendal		0.7700707	0.7750816	0.7755898	0.7734703	0.7734843
Pearson	GDT	0.9263458	0.9309587	0.9323503	0.927651	0.9300441
Spearman		0.9321835	0.9406859	0.9395187	0.9331065	0.9344475
Kendal		0.7819467	0.7854926	0.7821959	0.7812008	0.7833844
Pearson	MaxSub	0.9297004	0.9338626	0.9344752	0.9298453	0.9319266
Spearman		0.9349359	0.9421471	0.9397281	0.9341264	0.9360434
Kendal		0.7810229	0.7849646	0.7796613	0.7778905	0.7811618
Pearson	TM-score	0.9279442	0.9303009	0.9291652	0.9267494	0.930639
Spearman		0.9320759	0.9405161	0.9378111	0.9313664	0.9335525
Kendal		0.7841708	0.7858688	0.7801108	0.7800309	0.7835955

Table 3.7. Results of the data searching stage for correlation along with the ModFOLD6 scores. Green represents the best score and red the worst score for each respective testing methods.

On the other hand, TensorFlow showed improvements in comparison to ModFOLD6 when testing using GDT and GDT-HA. However, the pipeline failed to perform well in the majority of tests when MaxSub and TM-score were used for comparisons. When looking at the regression plots in Figure 3.8 it can be noticed that RSNNS provided a better correlation as its formed trendlines were linear and almost intersects (0,0) and (1,1). Whereas both plots for TensorFlow show a similar shape to ModFOLD6 and they seem to better fit an exponential trendline other than a linear line.

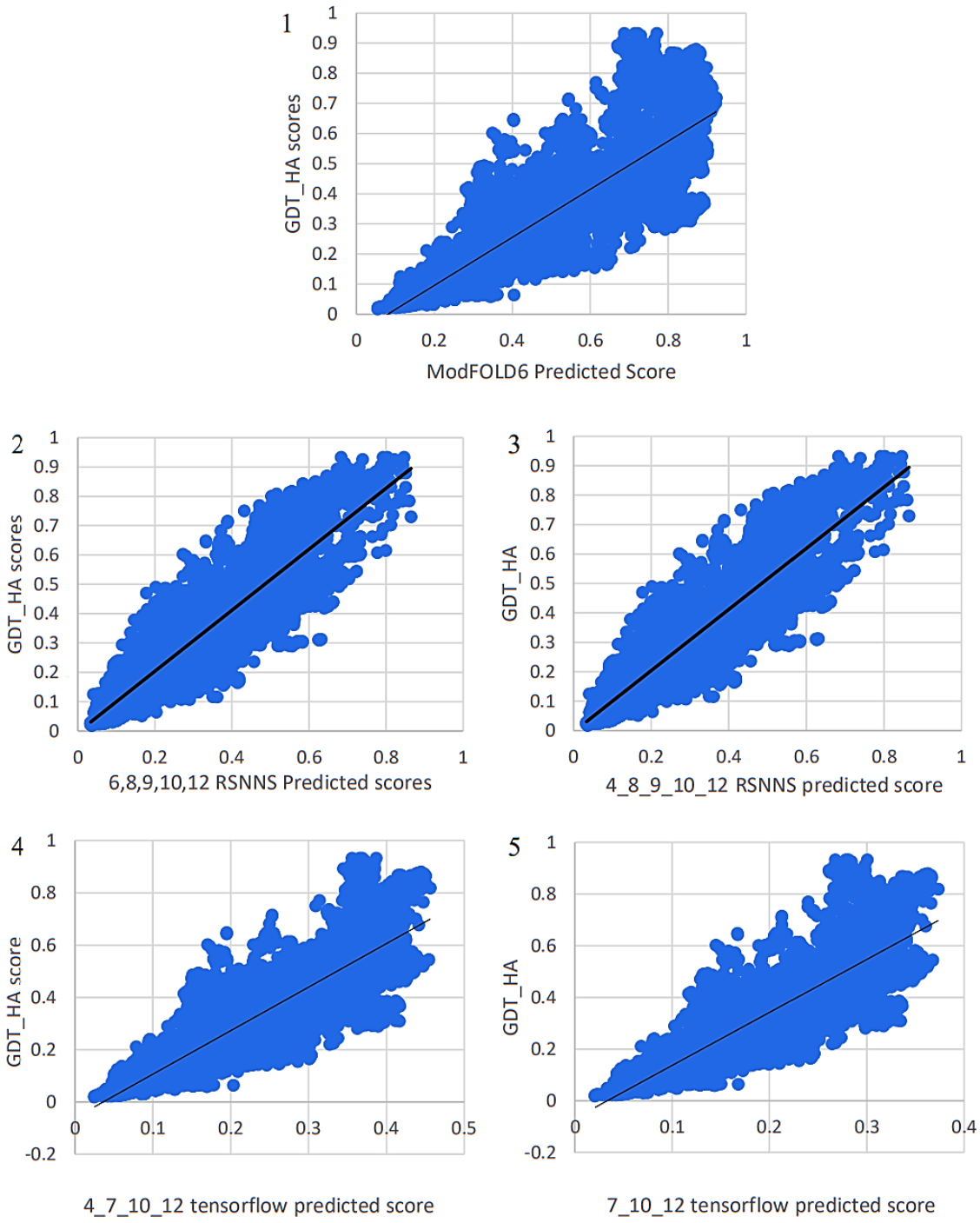

Figure 3.8. Regression plots comparing ModFOLD6 with the ranked RSNNS and TensorFlow MQAP combinations. Plot 1 shows the regression plot for ModFOLD6, Plot 2 shows the regression plot for 6_8_9_10_12 in RSNNS, Plot 3 shows the regression plot for 4_8_9_10_12 in RSNNS, Plot 4 shows the regression plot for 4_7_10_12 in TensorFlow and Plot 5 shows the regression plot for 7_10_12 in TensorFlow. All predicted scores were compared to GDT-HA.

3.4.1.2. Ranking/Selection benchmarking through RSNNS and TensorFlow

The same 6 sets of neural network parameters were implied through RSNNS and TensorFlow when benchmarking the MQAP methods using the ranking/selection scale.

Firstly, the selected combinations were benchmarked with the inclusion of the RSNNS networks. The outputted scores of this benchmarking were analysed, and the results were ranked based on the performance consistency through the 6 sets of NNs parameters. Unlike the correlation section, all the top-ranking scores came from the combination 8_9 as seen in Table 3.8, and thus only the 10 combinations found in Figure 3.9 will be used in the parameterisation step.

Observed Measures	Cumulative Score	Combination
GDT-HA	31.76354231	8_9
GDT	44.01783111	8_9
MaxSub	40.69864599	8_9
TM-score	46.77266494	8_9

Table 3.8. The top combinations and their scores for each Observed method in the combination stage in RSNNS.

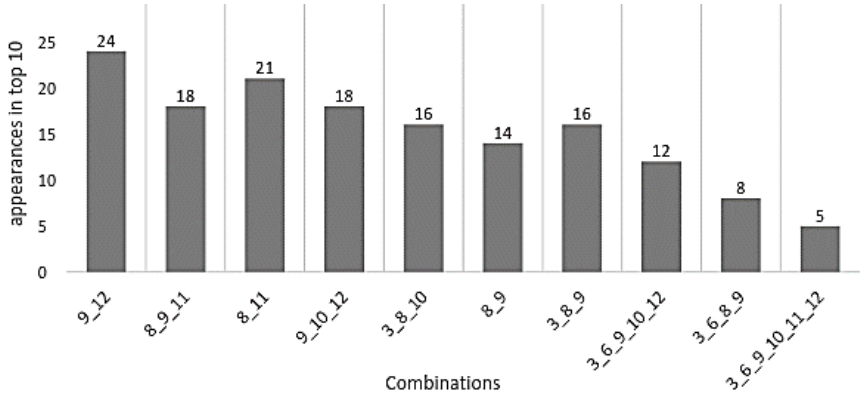

Figure 3.9. Bar chart representing the top 10 MQAP combinations for correlation through RSNNS. The ranking was carried out based on the appearances in each set of the NNs**.**

The parameterisation stage showed clear results as to which combination was best for ranking scores. As seen in Table 3.9, the combination 8_9_11 produced the top score in each observed score other than TM-score where it placed second. The top combination for ranking in TM-score was 8_9 however this exact score was achieved 19 times by 8_9 during the parameterisation stage

and is thus likely to be an upper limit of this combination. For this reason, the combination 8_9_11 was selected and used in the further optimisation stage. The parameters chosen in the data searching stage was 2 hidden units and 550 iterations as these parameters worked best when selecting the top model using GDT-HA.

Observed score	Sum of Top Model Scores	Combination	Parameters	
			Hidden units	Iterations
GDT-HA	31.98587951	8_9_11	2	550
GDT	44.05480416	8_9_11	2	550
MaxSub	40.88442186	8_9_11	2	1000
TM-score	46.77266494	8_9	Multiple	Multiple
	46.75706769	8_9_11	2	300

Table 3.9. The top combinations and the respective parameters for each Observed score in RSNNS.

The 6 sets of parameters used for ranking in TensorFlow were the same as those used in correlation. All the combinations shown in Figure 3.10 were used in the next stage, parameterisation. The results in Table 3.10 showed that 9_11_12 outperformed all other combinations except for 8_9_11_12 according to the TM-score. The top scores shown already outperformed the results at the end of the parameterisation step of RSNNS. As all the top-ranking combinations, shown in Table 3.10, were included in the combinations in Figure 3.10, no extra combinations passed to the parameterisation step.

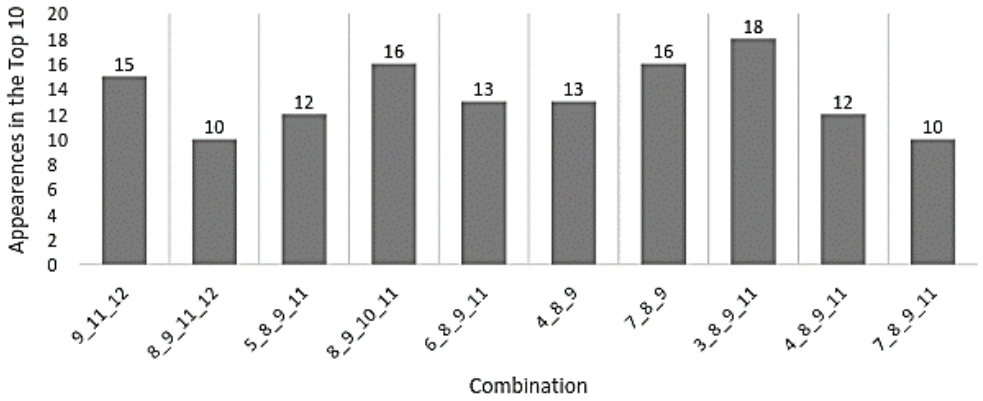

Figure 3.10. Bar chart representing the top 10 MQAP combinations for correlation through TensorFlow. The ranking was carried out based on the appearances in each set of the NNs**.**

Observed Measure	Cumulative Score	MQAP Combination
GDT-HA	32.40589837	9_11_12
GDT	44.54382173	9_11_12
MaxSub	41.55411428	9_11_12
TM-score	47.25694338	8_9_11_12

Table 3.10. The top combinations and their scores for each observed method in the combination stage in TensorFlow.

Table 3.11 shows the top scoring combinations for each observed score in TensorFlow along with the respective parameters used. 9_11_12 was clearly the best combination at ranking as it performed well according to all the observation scores, except the scores from TM-score where 8_9_11_12 was the top performing combination. However, 8_9_11_12 scored the same score of 47.28648181 using various parameters, and thus meaning this score is likely the upper limit of this combination. This made picking the best combination easy, however, the results in Table 3.11 made it difficult to pick the parameters to use along with the combination 9_11_12. 4 hidden units in layer 1, 2 hidden units in layer 2 and 300 iterations was the model parameter as it was best for both GDT and TM-score. While 5 hidden units in layer 1, 4 hidden units in layer 2 and 100 iterations only achieved a top score in GDT-HA and MaxSub. Ultimately, 5 hidden units in layer 1, 4 hidden units in layer 2 and 100 iterations was chosen to be used in the data searching stage. This set of parameters was chosen for two reasons: firstly, the NN is trained using GDT-HA thus making it the primary focus of this study and secondly the methodology behind data searching uses a set GDT-HA to loop over the NN.

Observed Measure	Cumulative Score	Combination	Parameters		
			Hidden units in layer 1	Hidden units in layer 2	Iterations
GDT-HA	32.42876369	9_11_12	5	4	100
GDT	44.60932707	9_11_12	4	2	300
MaxSub	41.57506947	9_11_12	5	4	100
TM-score	47.28648181	8_9_11_12	various*	various*	various*
	47.2269774	9_11_12	4	2	300

Table 3.11. The top combinations and their respective parameters for each Observed score in TensorFlow. *The various parameters were 2, 3, 5 hidden units in layer 1, 3, 5 in layer 2, and 100, 400 iterations.

3.4.2. Data Analysis

After running the 10 MQAP methods through the 6 sets of parameters of NN pipelines with the built-in RSNNS and TensorFlow and then benchmarking them using the ranking/selection and correlation scales, a massive amount of data was collected. The data was analysed, and impressive improvements were found when comparison. As can be seen in Figure 3.11, all differences in scores between the new RSNNS and TensorFlow techniques were whatever outside the margin of errors thus showing that the improvements are likely to be reliable.

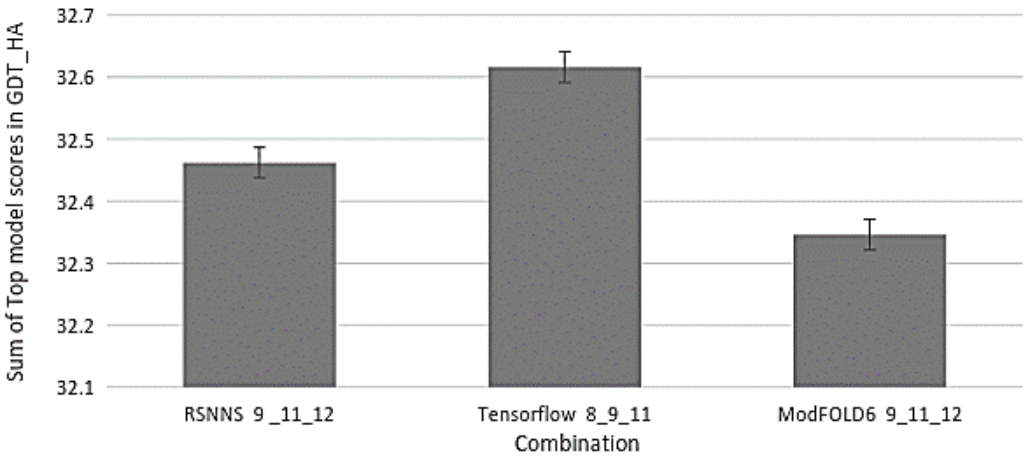

Figure 3.11. Bar chart representing the top-ranking combination score for each technique using GDT-HA.

The results of analysed from the data searching stage were also shown in Table 3.12. While RSNNS technique scores were an improvement to that of ModFOLD6, RSNNS underperformed when compared to TensorFlow as well. However, TensorFlow technique achieved the highest score for every observational method. The largest improvement was in GDT-HA with an overall improvement of 0.88% compared to ModFOLD6.

Observed Score	Cumulative Score		
	ModFOLD6	RSNNS	TensorFlow
GDT-HA	32.34630537	32.4621495	32.62412141
GDT	44.53946127	44.65405679	44.67576979
MaxSub	41.53627352	41.30589405	41.63982609
TM-score	47.18034881	47.28185714	47.26613235

Table 3.12. The results of the data searching stage for ranking along with the ModFOLD6 scores. Green represents the best score for each Observed score while red represents the worst score for the respective Observed score.

The study of both Correlation and Ranking scales gave opposing results. The correlation study tells us that RSNNS using 6_8_9_10_12 is the superior method as TensorFlow was not even able to outperform MODFOLD6 in certain tests. However, the ranking scores showed that TensorFlow was the superior technique. In the Ranking scale, RSNNS also showed an improvement compared to ModFOLD6. However, when statistical analyses were conducted using the Wilcoxon sign test, the improvements were not statistically significant (p-value = 0.4836, N = 84), thus RSNNS could not give any improvement when combining multiple global scores. On the other hand, TensorFlow was found to be slightly better with a p-value of 0.0000000000000004064 (N = 84).

During the entirety of this study it became apparent that combinations were either good at ranking or correlation, this may seem counterintuitive as the expected efficient MQAP method would be a combination of methods which are accurate at assigning scores and also good at picking out the top-ranking model. This may be due to the negatively skewed data set (Figure 3.12) thus meaning combinations which were accurate at assigning scores to bad models score higher in the correlation analysis than combinations which were only accurate at scoring good models. To resolve this issue, future projects could use positively skewed data, this would mean that the NN is more accurate at assigning scores to good models and worse at assigning scores to bad models. Being inaccurate at assigning scores to bad models is not as much of a problem as the models themselves are inaccurate.

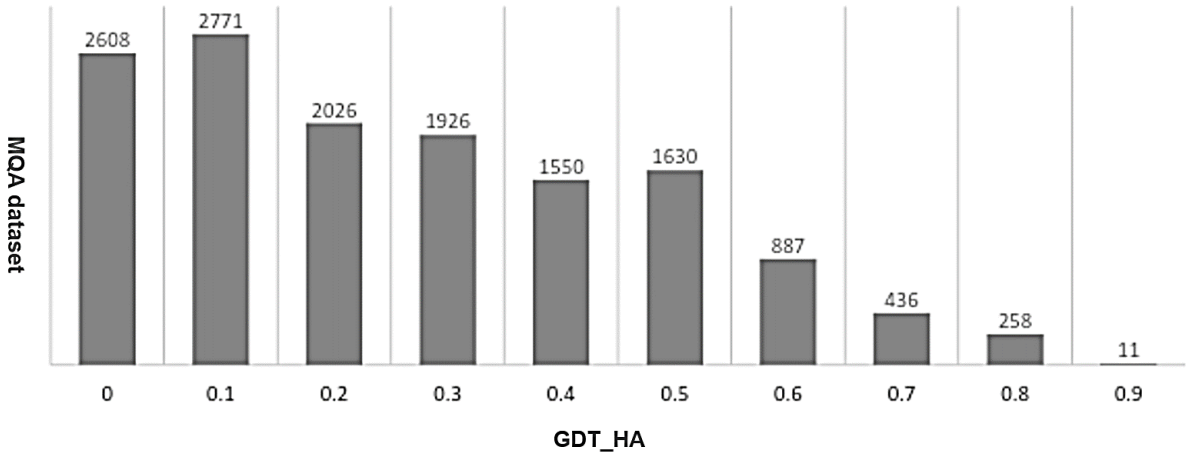

Figure 3.12. Distribution of Model Quality in the data set, measured using GDT-HA.

The poor correlation for TensorFlow could be due to an error in the training methodology or a lack of a specialised activation function. Indeed, this could be the case because as can be seen in the regression plots in Figure 3.8, the highest prediction score made by TensorFlow was 0.46 which is half the value of the highest true score. A change in methodology or a different activation function, such as Rectified linear unit (ReLU) may solve this issue (Abadi, et al., 2016). ReLU uses a ramp function to add linearity to the model which allows it to improve the training of the NN especially when presented with a large data set (Arora, et al., 2016). Other activation functions can also be considered such as sigmoid, softplus and tanh, however these functions seem to be of less use for ModFOLD compared to the potential of ReLU Relu due to their non-linearity (Abadi, et al., 2016).

There are further improvements that can be made in TensorFlow, which may improve the accuracy for both correlation and ranking. One improvement would be to use biases when training the NN. Biases are used to manipulate the prediction scores and thus can be used to make the prediction scores more linear. While this won't improve correlation scores, it will improve the regression plots. Batch training can also be implemented. In batch training, the training step for the NN uses smaller batches of the training data, and the models in the batch are changed with each iteration. Batch training could potentially allow for the increase in learning rate and iteration number without overfitting the NN, thus this technique could improve accuracy of prediction.

3.5. Conclusion

In this study we have sought to optimise combinations of the ten global model quality assessment methods, including ModFOLD6. The methods were benchmarked firstly through different regressions, individually as well as in combinations, using 2 scales of measurements, Ranking/Selection and Correlation. The second benchmarking was conducted after the inclusion of a simple MLPs (RSNNS) and a deep MLPs (TensorFlow) to the MQAP methods pipeline separately. After benchmarking, the outputted data were analysed to find the optimum scores that was achieved from the including these two neural networks. It was found that both NNs were useful for different reasons. The MLPs from RSNNS outperformed the other techniques when testing for correlation between the predicted score and the true score while the deep MLPs using TensorFlow outperformed the others when testing for the ability to pick out the highest ranked models. This study shows the potential in using deep learning techniques for combining scores from MQAPs and offers suggestions as to how deep learning methodology could be modified (e.g. using different activation functions and positively skewing training data) to improve the predictive ability of the neural networks.

Chapter 4

Independent Benchmarking for an Updated Version of ModFOLD6 with the Top EMA Methods in CASP12

Work presented in this chapter has been published in the following papers:

Maghrabi, A.H.A., McGuffin, L.J., 2017. ModFOLD6: an accurate web server for the global and local quality estimation of 3D protein models. Nucleic Acids Res 45, W416–W421. https://doi.org/10.1093/nar/gkx332 (Both authors contributed equally to the paper as first authors. Figures and tables are adapted from Maghrabi & McGuffin 2017, unless otherwise indicated).

Elofsson, A., Joo, K., Keasar, C., Lee, J., **Maghrabi, A.H.A.**, Manavalan, B., McGuffin, L.J., Hurtado, D.M., Mirabello, C., Pilstål, R., Sidi, T., Uziela, K., Wallner, B., 2018. Methods for estimation of model accuracy in CASP12. Proteins: Structure, Function, and Bioinformatics 86, 361–373. https://doi.org/10.1002/prot.25395 (All authors contributed equally to this study and the list is sorted alphabetically.)

4.1. Background

Methods which predict the three-dimensional (3D) models of proteins are now routinely relied upon to drive research across the life sciences. The reason behind that lies in the expense of the protein structures determination experiments, and also their time limitations. Predicting a 3D model is comparatively quick and can often be of sufficiently high quality. However, with all predictions there is some level of uncertainty, and therefore accurate methods for model quality assessment have become necessary for driving the acceptance of structure prediction methods. Essentially, relying on a 3D model of a protein without an estimate of its accuracy is tantamount to relying on a sequence alignment without an E-value. Thus, the development of 3D model Quality Assessment (QA) tools has become an important area of research in itself. Numerous methods have been developed over the years in an attempt to provide users with scores that will give them confidence in their 3D models and allow them to identify any potentially suspect regions.

The model quality assessment field has its roots in early structure validation tools (Laskowski et al., 1996) (Hooft et al., 1996) (Wiederstein and Sippl, 2007). Such tools can be used to perform basic stereochemical checks, and they are very useful in identifying unusual geometric features in a model. However, such methods are not able to produce a single global score that can be used for ranking alternative models or discriminating good models from bad (often bad models will still have good stereochemistry). Modern methods for QA can be classified into three broad categories: pure-single model methods, which consider only information within an individual model (Eisenberg et al., 1997) (Wiederstein and Sippl, 2007) (Zhou and Zhou, 2002) (McGuffin, 2007) (Uziela and Wallner, 2016) (Uziela et al., 2017) (Benkert et al., 2008) (McGuffin, 2008), clustering/consensus approaches (McGuffin, 2009) (Larsson et al., 2009) (Benkert et al., 2009) (Cheng et al., 2009) (McGuffin and Roche, 2010), which can only be used if you have multiple alternative models built for the same protein target, and quasi-single model methods (McGuffin et al., 2013) (Roche et al., 2014), which can score an individual model against a pool of alternative models generated from the target sequence. Each approach has its advantages and disadvantages. Clustering methods have been far more accurate than pure single-model methods but are more computationally intensive and do not work when very few similar models are available, which is often the case in real life research scenarios. Pure-single model methods are less accurate overall, but they are more rapid, they produce consistent scores for single or few models at a time and they often perform better at model ranking and selection.

Quasi-single model methods attempt to provide comparable accuracy to clustering methods, while addressing real-life needs of researchers with few/single models. We initially implemented a quasi-single model approach with our ModFOLD3 method (Roche et al., 2014), which generated reference sets of models from the target sequence, using IntFOLD-TS (McGuffin and Roche, 2011), for comparison with the submitted model using ModFOLDclust2 (McGuffin and Roche, 2010). The method has since undergone a number of updates: ModFOLD4 (McGuffin et al. 2013), which makes use of IntFOLD2-TS (Buenavista et al., 2012) models, and ModFOLD5, which makes use of IntFOLD3-TS (McGuffin et al., 2015) models. Each of these quasi-single model versions of ModFOLD have been ranked among the top performing methods in the quality assessment categories of the recent CASP experiments (Kryshtafovych et al., 2014) (Kryshtafovych et al., 2016) and have undergone incremental improvements in accuracy. By some measures, the quasi-single model methods have been competitive with the predictive power offered by clustering-based methods, as well as being capable of making predictions for a single model at a time. While the ModFOLD server has been a pioneer of the quasi-single model approach and a leader in terms of prediction performance, it has fallen short in some aspects, such as model selection. Furthermore, there is still significant room for improvement in many aspects of quality assessment.

In this chapter, we describe significant major updates to the ModFOLD server. The server has been popular with modellers around the world, having completed ~200 000 quality assessment jobs for ~9000 unique users. The latest version, ModFOLD6, operates solely in single model mode, deploying a novel hybrid pure/quasi-single model QA algorithm. In addition to interface updates, in this chapter we will also briefly describe the major modifications to the prediction algorithm, which have led to significant performance gains in both local and global model quality predictions, allowing us to maintain our position as a leading prediction group. The main changes under the hood have been the addition of several new local scoring inputs, a new neural network (NN) architecture and alternative optimized global scores for different use cases. On the front end submission page, users are now given three alternative choices for optimized global model quality scoring, depending on whether their preference is for optimal model selection (the best models are ranked at the very top), predicting absolute values (the predicted scores closely reflect the observed scores) or more balanced performance for the two use cases. We also report on the independent benchmarking of the server for the recent CASP12 experiment and ongoing CAMEO project.

4.2. Materials and methods

4.2.1. Architecture and pipeline of the optimised ModFOLD6

The ModFOLD6 server combines a pure-single and quasi-single model strategy for improved accuracy, which was originally developed for the CASP12 experiment. For ModFOLD version 6, our initial emphasis was on increasing the accuracy of per-residue assessments for single models. Each model was considered individually using three pure-single model methods, ProQ2 (Uziela and Wallner 2016) and two newly developed methods: The Contact Distance Agreement (CDA) score and the Secondary Structure Agreement (SSA) score. Additionally, a set of 130 reference 3D models (generated using the latest version of IntFOLD (McGuffin and Roche, 2011) (Buenavista et al., 2012) (McGuffin et al., 2015)) was used to score models using three alternative quasi-single model methods: the Disorder B-factor Agreement (DBA) score, the ModFOLD5_single residue score and the ModFOLDclustQ_single residue score (Figure 4.1). An NN was then used to combine the component per-residue quality scores from each of the six alternative scoring methods, resulting in a final consensus of per-residue quality scores for each model.

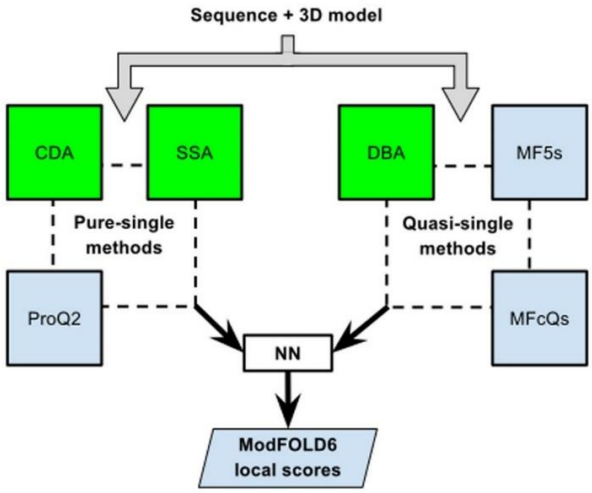

Figure 4.1. Flow of data for local quality assessment scoring in ModFOLD6. The target sequence and 3D model were evaluated with three pure-single model scoring methods (Secondary Structure Agreement (SSA), Contact Distance Agreement (CDA) and ProQ2) and three quasi-single model methods (Disorder B-factor Agreement (DBA), ModFOLD5_single (MF5s) and ModFOLDclustQ_single (MFcQs)). The new methods developed for ModFOLD6 are highlighted in green. The per-residue scores from all six methods were combined into a single residue score using an artificial neural network. Adapted from Maghrabi and McGuffin, (2017).

We used a standard neural network architecture (a Multi-Layer Perceptron, or MLP) for ModFOLD6 in order to strengthen the accuracy of local quality assessment scoring (Figure 4.2). Scores for each residue in the model were fed into the input layer, taken from the 6 local scoring methods using a sliding window of 5 residues (30 inputs). The hidden layer was made up of 15 hidden neurons and the network was trained to learn the output S_i score of the residue in the model compared to the native structure according to the TM-score structural superposition: $S_i = 1/(1 + (d_i/d_0)^2$, where S_i ranges from 0 to 1, d_i is the distance between structurally aligned residues and d_0 is the distance threshold = 3.9. Using 30 inputs and 15 hidden neurons was found to be an optimal architecture. No significant improvement was gained by further increasing the number of hidden neurons. The MLP function from RSNNS was used to build and train the network in R (https://cran.r-project.org/web/packages/RSNNS/). This part of the research was taken into consideration, and further studies were carried out later with the focus on the implementation of neural network and deep neural network into our method.

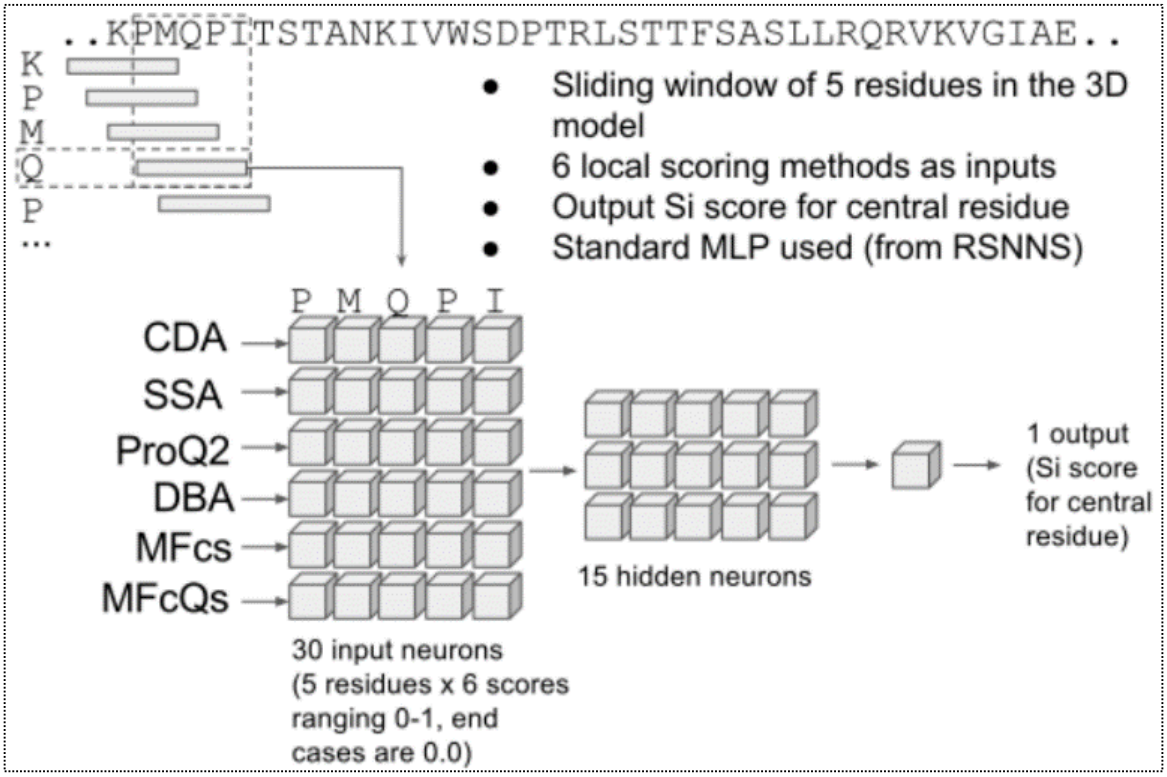

Figure 4.2. P**ipeline showing details of neural network architecture and flow of data for local quality assessment scoring in ModFOLD6.** Scores for each residue in the model are fed into the input layer, taken from the 6 local scoring methods using a sliding window of 5 residues (30 inputs). The hidden layer was made up of 15 hidden neurons and the network was trained to learn the output S_i score of the residue in the model compared to the native structure according to the TM-score structural superposition ($S_i = 1/(1 + (d_i/d_0)^2)$, where S_i ranges from 0 to 1, d_i is the distance between structurally aligned residues and d_0 is the distance threshold (3.9). 30 inputs and 15 hidden neurons was found to be an optimal architecture. No significant improvement was gained by further increasing the number of hidden neurons. The multilayer perceptron (MLP) function from RSNNS was used to build and train the network in R (https://cran.r-project.org/web/packages/RSNNS/). Adapted from Maghrabi and McGuffin, (2017).

The ModFOLD6 component per-residue/local quality scoring methods were as follows: (1) CDA is new pure-single model local QA method that relates to the agreement between the predicted residue contacts according to MetaPSICOV (Jones et al., 2015) and the model contacts, which are measured by the Euclidean distance (in Å) between residues in the 3D model. All pairs of residues in a model that were measured to be 8 Å apart or less were considered to be in contact and the CDA score for each residue was calculated by the mean MetaPSICOV score for those model contacts. In other words, if residue *i* was measured to be in contact with both residue *j* and residue *k* in the

model, and MetaPSICOV scores also existed for *ij* and *ik*, then the CDA score for residue *i* was taken as the mean MetaPSICOV score for *ij* and *ik*. Thus, $\mathrm{CDA} = (\sum \mathrm{p})/\mathrm{c}$, where p is the MetaPSICOV score and c is simply the number of contacts for the residue in the model where a value for p also exists. (2) SSA is a simple new pure-single model local QA method that relates to the agreement between the predicted secondary structure of each residue according to PSIPRED (Buchan et al. 2013) and the secondary structure state of the residue in the model according to DSSP (Kabsch and Sander 1983). Thus, $\mathrm{SSA} = \mathrm{P_{CHE}}$, where, $\mathrm{P_{CHE}}$ is simply the p-value from PSIPRED for the secondary structure state — coil (C), helix (H) or strand (E) — of the residue in the model according to DSSP. The eight DSSP states (H, I, G, E, B, S, T, -) were reduced to three states such that E (strand) and H (helix) were preserved and all other states were treated as C (coil). (3) The local scores were also taken from the ProQ2 (Uziela and Wallner 2016) method. (4) The ModFOLD5_single local QA scores were calculated from the comparison of each model with the reference set of 130 models built by IntFOLD version 4, in a similar way to the ModFOLD4 (McGuffin et al. 2013) method acting in quasi-single model mode, with the predicted distances *d* converted back into residue similarities S_r, thus: $\mathrm{S_r} = 1/(1 + (\mathrm{d}/3.9)^2)$. (5) The ModFOLDclustQ_single local QA scores were calculated in a similar way to ModFOLD5_single, however, in this case individual models were compared against the reference IntFOLD4 set using the local Q-score approach (McGuffin and Roche, 2010) (Ben-David et al., 2009). (6) DBA is a new quasi-single model QA method that relates to the agreement between the predicted disordered residues in the sequence according to DISOPRED3 (Jones and Cozzetto, 2015) and the ModFOLD5_single predicted per-residue error. Thus, $DBA = 1 - |S_r - (1 - P_d)|$, where, S_r is the ModFOLD5_single accuracy of the predicted residue for the model and P_d is the probability of disorder according to DISOPRED3.

Global scores were then calculated by taking the mean per-residue scores (the sum of the per-residue similarity scores divided by the target sequence lengths) for each of the six individual component methods described above and the NN consensus output (ModFOLD6). Furthermore, three additional quasi-single global model quality scores were generated (Figure 4.3) for each model based on the original ModFOLDclust, ModFOLDclustQ and ModFOLDclust2 global scoring methods (McGuffin and Roche, 2010) (in a similar vein to the ModFOLD4_single and ModFOLD5_single *global* scores, which were previously tested in CASP10 (Kryshtafovych et al., 2014) and CASP11 (Kryshtafovych et al. 2016) respectively).

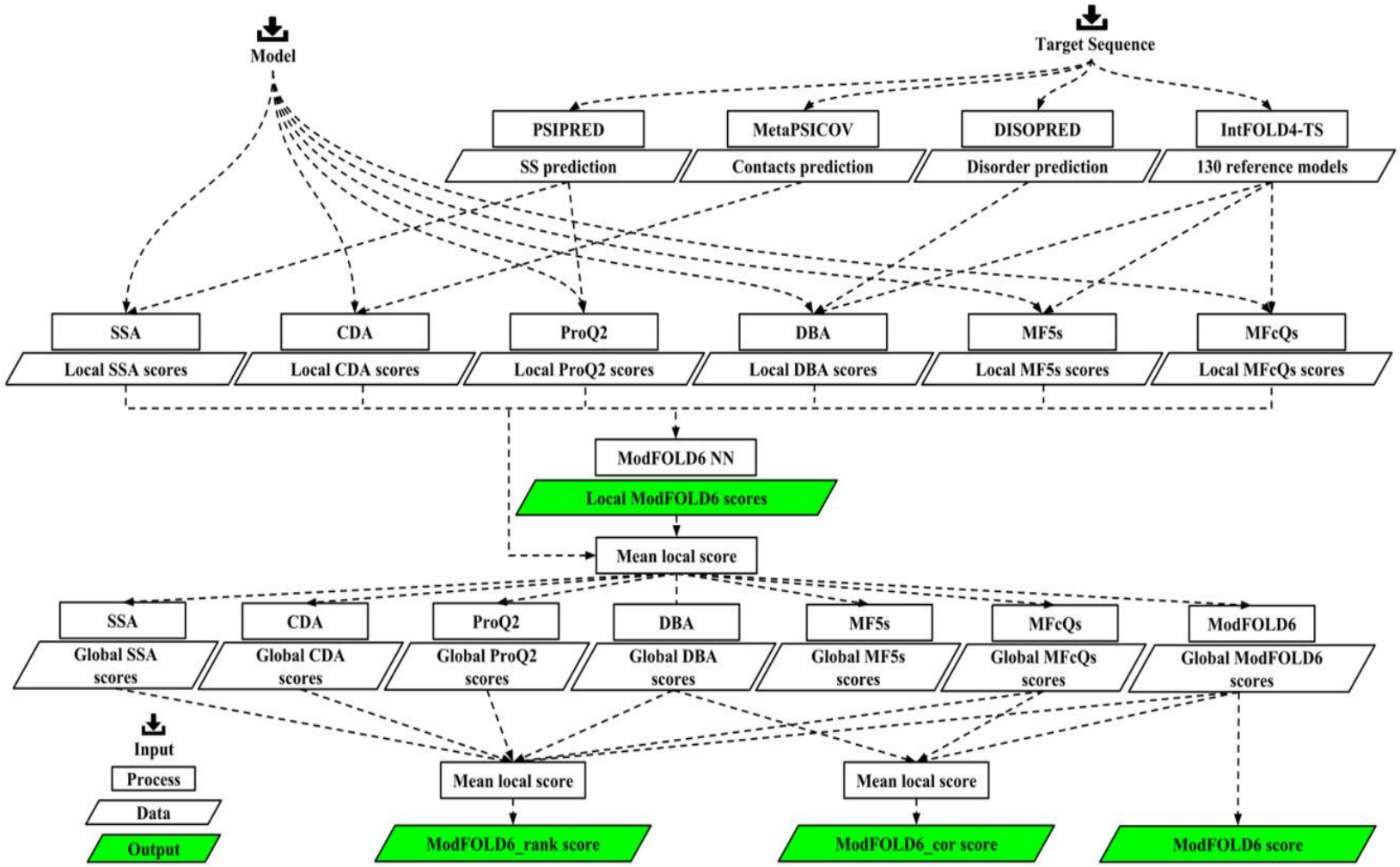

Figure 4.3. Flowchart outlining the principal stages of the ModFOLD6 server prediction pipeline. The initial input data are the target sequence and a single 3D model. The output data are the local/per-residue scores from the ModFOLD6 NN and the global score variants—ModFOLD6, ModFOLD6_rank, and ModFOLD6_cor. The ModFOLD6 pipeline is dependent on the following methods PSIPRED,31 DISOPRED,36 and MetaPSICOV37. Adapted from Elofsson et al., (2018).

4.2.2. ModFOLD6 variants

Thus, we ended up with 10 alternative global QA scores, which could be combined in various ways in order to optimise for the different aspects of quality estimation (QE) (Figure 4.4). The ModFOLD6 global score (the mean per-residue NN output score) considered alone was found to have a good balance of performance based on correlations of predicted and observed scores and rankings of the top models. The ModFOLD6_cor global score variant (calculated as: (ModFOLDclustQ_single_global + DBA_global + ModFOLD6_global)/3) was found to be an optimal combination for producing good correlations with the observed scores, i.e. the predicted global quality scores produced should produce closer to linear correlations with the observed global quality scores. The ModFOLD6_rank global score variant (calculated as: ModFOLDclustQ_single_global + ProQ2_global + CDA_global + DBA_global + SSA_global + ModFOLD6_global)/6) was found to be an optimal combination for ranking, i.e. the top ranked models (top 1) should be closer to the highest accuracy, but the relationship between predicted and observed scores may not be linear.

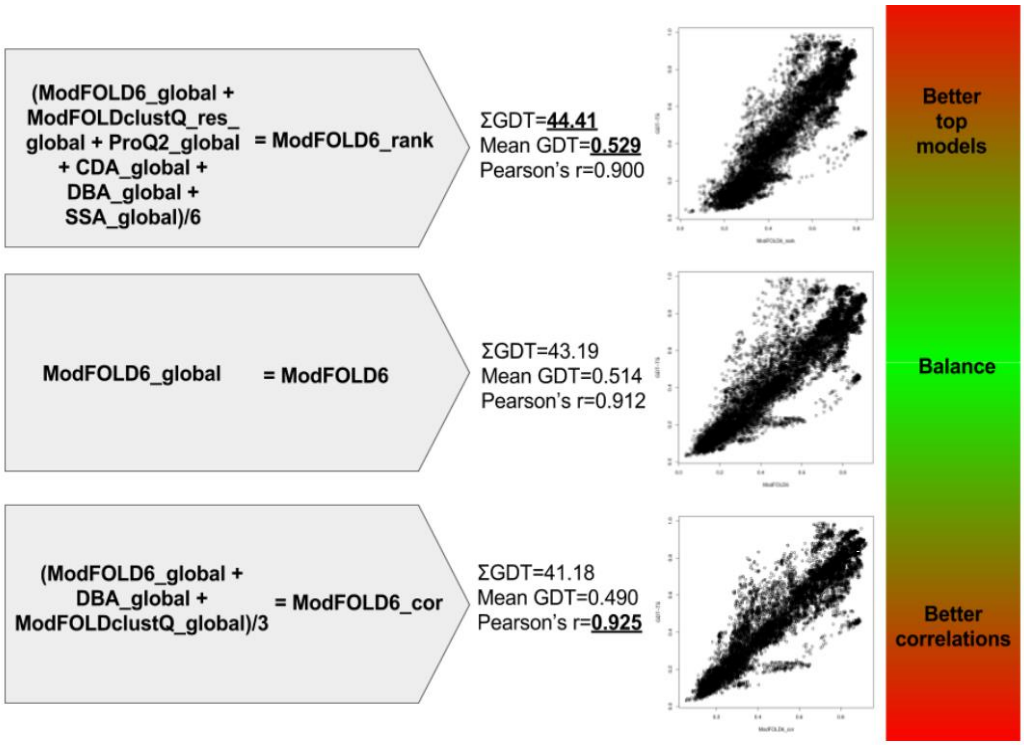

Figure 4.4. Summary of global score benchmarks for the 3 ModFOLD6 alternatives using CASP11 data. ModFOLD6_rank, ModFOLD6 and ModFOLD6_cor are the three optimised global accuracy scores

that may be selected by users on the ModFOLD6 server submission page. They were benchmarked and visualised for comparison. Adapted from Maghrabi and McGuffin, (2017).

4.3. Results and Discussion

4.3.1. Server inputs and outputs

The only required inputs to the ModFOLD6 server are the amino acid sequence for the target protein and a single 3D model (in PDB format) for evaluation. However, users may optionally upload multiple alternative models (as a compressed archive of PDB files), a name for their protein sequence and their email address. The server provides a clean and simple interface so that results can be easily interpreted by non-experts at a glance. The results page consists of a single table summarising the quality assessment scores for each submitted model (Figure 4.5a). The prediction data in the table are represented graphically, with thumbnail images of the local error plots and annotated 3D models. Users can click through the images in the table in order to drill down into individual results and visualize annotated 3D models interactively in using the JSmol/HTML5 framework (Figure 4.5b and c). No plugins are required and, conveniently, interactive results may also be viewed on mobile devices.

Each row in the results table includes: a global score for the model, a P-value indicating the likelihood that the observed similarity between the model and native structure is random (TM-score < 0.2) and a plot of the local errors in the model (the predicted distance in Ångströms of each residue from the native structure) (Figure 4.5a). Conveniently, the server also inserts the predicted local quality scores into the B-factor column of the ATOM records for each submitted model and makes them available to download, either individually or as a compressed archive. The results table also includes a graphical view of each model coloured by predicted B-factors using the temperature scheme (Figure 4.5a and b). The raw machine-readable data files for each set of predictions are also provided for developers, which comply with the CASP data standards.

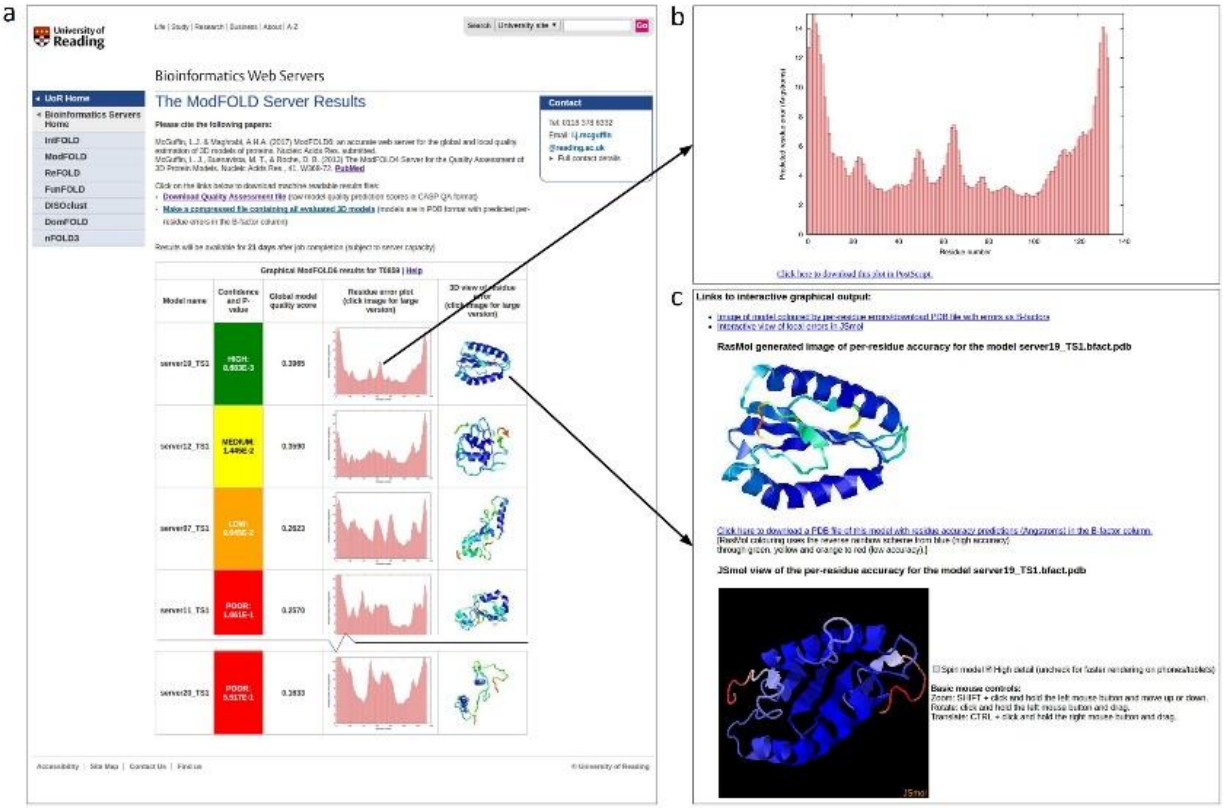

Figure 4.5. ModFOLD6 server results for models submitted to CASP12 generated for target T0859 (PDB ID: 5jzr). (a) An example of the graphical output from the server showing the main results page with a summary of the results from each method (truncated here to fit page). Clicking on the thumbnail images in the main table allows results to be visualized in more detail. (b) A histogram of the local or per-residue errors for the top ranked model, with the residue number on the x-axis and the predicted residue error (distance of the Cα atom from the native structure in Å) on the y-axis, which may be downloaded. (c) Interactive views of models, which can be manipulated in 3D using the JSmol/HTML5 framework and/or downloaded for local viewing. Adapted from Maghrabi and McGuffin, (2017).

4.3.2. Independent benchmarking of global scoring with official CAMEO and CASP12 data

The ModFOLD6 server is continuously independently benchmarked for local QE performance using the CAMEO resource (Haas et al., 2013). At the time that the original work was published (July 2017), the CAMEO public QE data (http://www.cameo3d.org/) showed that ModFOLD6, and another unpublished method (QMEANDisCo), were the leading public QA methods for producing local (per-residue) quality scores, according to the lDDT (Mariani et al., 2013) measure over 6 months. Our common subset analysis using 6 months of CAMEO data prior to CASP12,

verifies that the ModFOLD6 server is a significant improvement on our previous leading public ModFOLD4 method (McGuffin et al., 2013). Furthermore, these data show that ModFOLD6 also outperforms the top publicly available published methods in terms of local quality (Table 4.1 & 4.2 and Figure 4.6).

Method	AUC	StdErr	AUC 0–0.1	AUC 0–0.1 rescaled
ModFOLD6 (server18)	0.8748	0.00096	0.0508	0.5081
ModFOLD4 (server7)	0.8638	0.00099	0.0467	0.4669
ProQ2 (server 8)	0.8374	0.00107	0.0428	0.4283
Verify3d (server0)	0.7020	0.00134	0.0208	0.2081
Dfire v1.1 (server1)	0.6606	0.00138	0.0168	0.1675

Table 4.1. Independent benchmarking of local scoring publicly available published EMA methods with CAMEO comparing. A 26-weeks data was collected between 29 April 2016 and 21 October 2016 from http://www.cameo3d.org/. AUC = Area Under the ROC Curve. StdErr = Standard Error in AUC score. AUC 0-0.1 = Area Under the ROC curve with False Positive Rate ≤ 0.1. The table is sorted by the AUC score. Adapted from Maghrabi and McGuffin, (2017).

Independent benchmarking of local scoring with CAMEO using 6 months of common data comparing five publicly available published methods (177 025 common residues, 725 common models, 113 650 high quality residues, 63 375 low quality residues).

Method	AUC	StdErr	AUC 0-0.1	AUC 0-0.1 rescaled
ModFOLD6 (server18)	0.8921	0.002	0.0525	0.5249
ModFOLD4 (server7)	0.883	0.00207	0.0519	0.5189
ProQ2 (server 8)	0.8552	0.00229	0.0437	0.4369
VoroMQA_v2 (server17)	0.7925	0.00267	0.0247	0.2472
VoroMQA_sw5 (server15)	0.7657	0.0028	0.0304	0.3036
Verify3d (server0)	0.7157	0.003	0.02	0.2003
EQuant 2 (server16)	0.7014	0.00305	0.0185	0.1848
Prosa2003 (server2)	0.7007	0.00305	0.0215	0.2148
Naive PSIBlast (server3)	0.6769	0.00312	0.0171	0.1712
Dfire v1.1 (server1)	0.6332	0.00322	0.0156	0.1564

Table 4.2. Independent benchmarking of the top local scoring EMA methods. The data was collected from CAMEO using 6 months including all 10 publicly available published methods (126 common models, 31076 common residues, 17984 high quality residues, 13092 low quality residues). 26 weeks of data between 2016-04-29 and 2016-10-21 downloaded from http://www.cameo3d.org/. AUC = Area Under ROC

curve. StdErr = Standard Error in AUC score. AUC 0-0.1 = Area Under the ROC curve with False Positive Rate <= 0.1. Table is sorted by the AUC score. Adapted from Maghrabi and McGuffin, (2017).

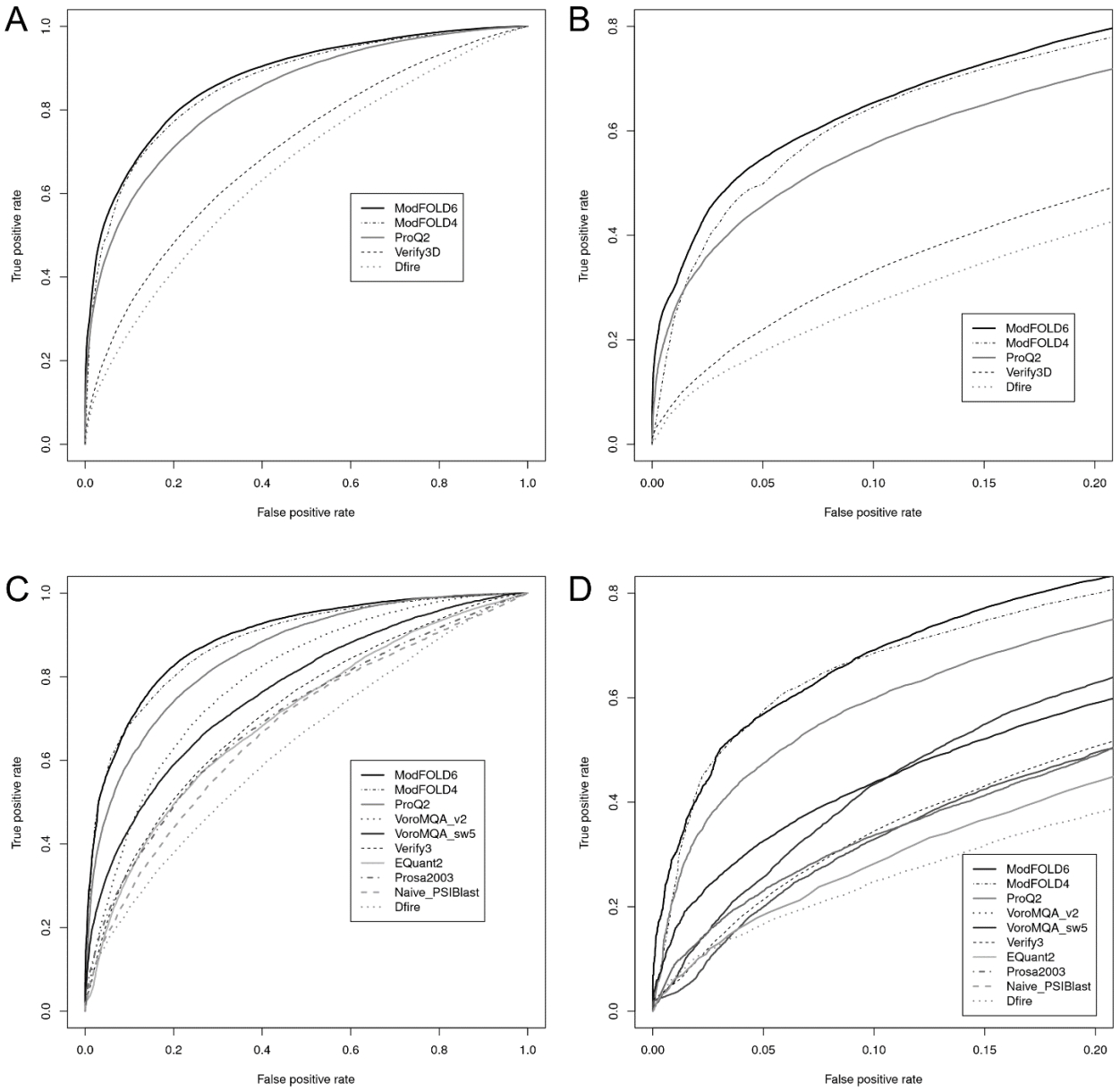

Figure 4.6. Line graph representing independent benchmarking of local scoring EMA methods. The data was collected from CAMEO using 6 months - ROC plots for data shown in Table 4.2 and Table 4.3. A true positive is defined as a residue correctly identified to be low quality, with local LDDT <= 60. (A) Full ROC plot for common subset with 5 publicly available published methods. (B) ROC plot with False Positive Rate (FPR) <= 0.1 for common subset including 5 publicly available published methods. (C) Full ROC plot for common subset including all 10 publicly available published methods. (D) ROC plot with FPR <= 0.1 for common subset including all 10 publicly available published methods. Adapted from Maghrabi and McGuffin, (2017).

The ModFOLD6 server was also subjected to independent blind testing during the CASP12 experiment in 2016. We were invited to speak at the CASP12 meeting in Gaeta as one of the leading groups in the Estimation of Model Accuracy category. The ModFOLD6 server performed particularly well in terms of differentiating between good and bad models (Table 4.3), local scoring (Table 4.4 & 2.5) and assigning absolute global accuracy values (Tables 4.6 – 4.9). The CASP12 data indicates that: ModFOLD6 ranks in top 10 in every benchmark of local score performance, it is the overall leading single model approach, it is competitive with the consensus/clustering approaches and it outperforms all pure-single model methods (Table 4.2 – 4.5). In terms of global scores, the ModFOLD6 variants were ranked within the top three for nearly every global benchmark using LDDT and CAD (Olechnovič et al., 2013) scores, as well as ranking within the top 10 according to other scores. (Table 4.1 and 4.6 – 4.9). The server was also a key factor contributing to our success in the Template Based Modelling category, where our group ranked in second position according to the assessors formula (http://www.predictioncenter.org/casp12/).

			GDT-TS	LDDT	CAD(AA)	SG
Rank	**Gr.Name**	**Gr.Model**	**AUC**	**AUC**	**AUC**	**AUC**
1	ModFOLD6_rank	QA072_1	0.993	0.99	0.926	0.962
2	ModFOLD6_cor	QA360_1	0.995	0.988	0.885	0.949
3	ModFOLD6	QA201_1	0.994	0.988	0.878	0.944
4	qSVMQA	QA120_1	0.982	0.983	0.862	0.937
5	ProQ3	QA213_1	0.985	0.978	0.892	0.916
6	ProQ3_1_diso	QA095_1	0.982	0.978	0.891	0.922
7	ProQ3_1	QA302_1	0.981	0.977	0.889	0.917
8	ProQ2	QA203_1	0.944	0.971	0.921	0.932
9	MUfoldQA_S	QA334_1	0.977	0.968	0.898	0.913
10	MULTICOM-CLUSTER	QA287_1	0.956	0.968	0.893	0.921

Table 4.3. Independent benchmarking of global scoring EMA methods in CASP12. The ability of methods to separate good models (accuracy score ≥ 50) from bad (<50) according to GDT-TS (Li et al., 2016), LDDT, CAD and SG scores is evaluated using the Areas Under the Curve (AUC) (see http://predictioncenter.org/casp12/doc/presentations/CASP12_QA_AK.pdf). Only the top 10 methods are shown, and the table is sorted using LDDT scores. The scores are calculated over all models for all targets (QA stage 1–select 20). The table is sorted by the LDDT AUC score. Data are from http://predictioncenter.org/casp12/qa_aucmcc.cgi. Adapted from Maghrabi and McGuffin, (2017).

Rank	Gr.Name	Gr.Model	MCC(3.8)	MCC(5.0)	Corr.
1	ModFOLD6_rank	QA072_1	0.536	0.54	0.513
2	ModFOLD6	QA201_1	0.536	0.54	0.513
3	ModFOLD6_cor	QA360_1	0.536	0.54	0.513
4	Pcons-net	QA432_1	0.457	0.511	0.381
5	Wallner	QA073_1	0.27	0.379	0.301
6	Pcons	QA089_1	0.22	0.307	0.275
7	ProQ3	QA213_1	0.332	0.313	0.216
8	ProQ3_1	QA302_1	0.342	0.34	0.204
9	ProQ3_1_diso	QA095_1	0.341	0.34	0.203
10	Wang1	QA132_1	0.188	0.164	0.198

Rank	Gr.Name	Gr.Model	MCC(3.8)	MCC(5.0)	Corr.	AUC(3.8)
1	Wallner	QA073_2	0.756	0.745	0.697	0.946
2	Pcons	QA089_2	0.755	0.76	0.688	0.945
3	Pcons-net	QA432_2	0.738	0.737	0.67	0.935
4	ModFOLDclust2	QA214_2	0.748	0.785	0.669	0.949
5	ModFOLD6_cor	QA360_2	0.73	0.719	0.657	0.938
6	ModFOLD6_rank	QA072_2	0.73	0.719	0.657	0.938
7	ModFOLD6	QA201_2	0.73	0.719	0.657	0.938
8	DavisEMAconsensus	QA034_2	0.717	0.773	0.635	0.948
9	Pcomb-domain	QA411_2	0.717	0.744	0.642	0.939
10	ProQ3_1_diso	QA095_2	0.601	0.579	0.512	0.885

Table 4.4. Independent benchmarking of Corr. local scoring EMA methods in CASP12. The top 10 groups are shown. Table is sorted by the Corr. score. Data are from http://predictioncenter.org/casp12/qa2_aucmcccorr.cgi. Adapted from Maghrabi and McGuffin, (2017).

Rank	Gr.Name	Gr.Model	ASE
1	DavisEMAconsensus	QA034_1	85.831
2	ModFOLDclust2	QA214_1	84.808
3	Pcons-net	QA432_1	84.805
4	Pcons	QA089_1	84.182
5	ModFOLD6	QA201_1	83.475
6	ModFOLD6_cor	QA360_1	83.473
7	ModFOLD6_rank	QA072_1	83.473
8	Pcomb-domain	QA411_1	83.146
9	Wallner	QA073_1	81.845
10	ZHOU-SPARKS-X	QA452_1	80.188

Rank	Gr.Name	Gr.Model	ASE
1	ModFOLDclust2	QA214_2	87.032
2	DavisEMAconsensus	QA034_2	86.832
3	Pcons	QA089_2	86.679
4	Pcons-net	QA432_2	84.178
5	Wallner	QA073_2	84.17
6	ModFOLD6	QA201_2	83.852
7	ModFOLD6_cor	QA360_2	83.851
8	ModFOLD6_rank	QA072_2	83.851
9	Pcomb-domain	QA411_2	83.634
10	ProQ3_1_diso	QA095_2	79.158

Table 4.5. Independent benchmarking of ASE local scoring EMA methods in CASP12. The top 10 groups are shown. Table is sorted by the ASE score. Data are from http://predictioncenter.org/casp12/qa2_ase.cgi. Adapted from Maghrabi and McGuffin, (2017).

Rank	Gr.Name	Gr.Model	GDT-TS	LDDT	CAD(AA)	SG
1	ModFOLD6_cor	QA360_1	6.697	4.249	17.461	9.931
2	ModFOLD6_rank	QA072_1	10.578	4.877	13.368	12.2
3	Pcomb-domain	QA411_1	8.56	4.987	17.031	10.413
4	QASproCL	QA267_1	9.107	5.432	16.59	11.624
5	ModFOLD6	QA201_1	5.883	5.813	19.241	9.005
6	MULTICOMCLUSTER	QA287_1	10.222	5.835	14.943	12.013
7	Wang2	QA206_1	8.021	5.874	18.213	9.836
8	Deepfold-Contact	QA219_1	8.507	6.313	20.421	10.464
9	naive	QA109_1	8.507	6.313	20.421	10.464
10	DeepFold-Boom	QA223_1	8.507	6.313	20.421	10.464

Rank	Gr.Name	Gr.Model	GDT-TS	LDDT	CAD(AA)	SG
1	ModFOLD6_rank	QA072_2	9.754	6.019	9.751	12.732
2	ModFOLD6_cor	QA360_2	6.748	8.248	14.423	12.319
3	MULTICOMCLUSTER	QA287_2	11.445	8.472	12.658	13.161
4	ModFOLD6	QA201_2	7.087	8.565	14.35	12.292
5	Pcomb-domain	QA411_2	9.839	8.842	11.312	13.275
6	qSVMQA	QA120_2	11.608	8.879	12.336	13.642
7	ProQ3_1	QA302_2	10.155	8.91	14.32	12.213
8	ProQ3_1_diso	QA095_2	10.159	8.931	14.778	12.088
9	MUfoldQA_S	QA334_2	8.898	9.053	16.343	12.268
10	ProQ3	QA213_2	11.418	9.14	15.006	12.622

Table 4.6. Independent benchmarking of LDDT global scoring EMA methods in CASP12. The top 10 groups are shown. Table is sorted by the LDDT score. Data are from http://predictioncenter.org/casp12/qa_diff_mqas.cgi. Adapted from Maghrabi and McGuffin, (2017).

			GDT-TS			LDDT			CAD(AA)			SG		
Rank	Gr.Name	Gr.Model	MCC (40)	MCC (50)	AUC	MCC (40)	MCC (50)	AUC	MCC (40)	MCC (50)	AUC	MCC (40)	MCC (50)	AUC
1	ModFOLD6_rank	QA072_1	0.613	0.814	0.993	0.685	0.686	0.99	0.382	0.502	0.926	0.572	0.523	0.962
2	ModFOLD6_cor	QA360_1	0.694	0.863	0.995	0.668	0.686	0.988	0.314	0.472	0.885	0.538	0.483	0.949
3	ModFOLD6	QA201_1	0.676	0.708	0.994	0.665	0.578	0.988	0.343	0.489	0.878	0.551	0.504	0.944
4	qSVMQA	QA120_1	0.521	0.579	0.982	0.587	0.562	0.983	0.399	0.541	0.862	0.544	0.525	0.937
5	ProQ3	QA213_1	0.579	0.625	0.985	0.611	0.53	0.978	0.314	0.524	0.892	0.491	0.524	0.916
6	ProQ3_1_diso	QA095_1	0.516	0.572	0.982	0.556	0.509	0.978	0.337	0.503	0.891	0.481	0.476	0.922
7	ProQ3_1	QA302_1	0.522	0.584	0.981	0.557	0.526	0.977	0.343	0.511	0.889	0.473	0.483	0.917
8	ProQ2	QA203_1	0.366	0.455	0.944	0.456	0.459	0.971	0.48	0.522	0.921	0.451	0.443	0.932
9	MUfoldQA_S	QA334_1	0.716	0.764	0.977	0.561	0.568	0.968	0.228	0.432	0.898	0.43	0.416	0.913
10	MULTICOMCLUSTER	QA287_1	0.45	0.465	0.956	0.504	0.457	0.968	0.348	0.456	0.893	0.407	0.423	0.921

Table 4.7. Independent benchmarking of stage 1 global scoring EMA methods in CASP12. The top 10 groups are shown. Table is sorted by the LDDT-AUC score. Data are from http://predictioncenter.org/casp12/qa_aucmcc.cgi. Adapted from Maghrabi and McGuffin, (2017).

			GDT-TS			LDDT			CAD(AA)			SG		
Rank	**Gr.Name**	**Gr.Model**	**MCC (40)**	**MCC (50)**	**AUC**	**MCC (40)**	**MCC (50)**	**AUC**	**MCC (40)**	**MCC (50)**	**AUC**	**MCC (40)**	**MCC (50)**	**AUC**
1	Wallner	QA073_2	0.721	0.734	0.988	0.707	0.745	0.966	0.351	0.592	0.923	0.665	0.665	0.936
2	Pcomb-domain	QA411_2	0.735	0.801	0.984	0.717	0.692	0.963	0.531	0.668	0.925	0.632	0.654	0.932
3	ModFOLD6_rank	QA072_2	0.763	0.843	0.983	0.74	0.773	0.962	0.486	0.679	0.925	0.639	0.675	0.929
4	QASproCL	QA267_2	0.788	0.783	0.987	0.733	0.676	0.958	0.493	0.635	0.906	0.629	0.635	0.928
5	MUfoldQA_C	QA318_2	0.812	0.844	0.982	0.746	0.727	0.958	0.442	0.65	0.902	0.652	0.654	0.927
6	Pcons	QA089_2	0.662	0.703	0.985	0.644	0.712	0.957	0.322	0.567	0.903	0.642	0.643	0.928
7	FDUBio	QA237_2	0.835	0.872	0.984	0.773	0.741	0.957	0.437	0.66	0.91	0.673	0.664	0.928
8	ModFOLD6	QA201_2	0.774	0.835	0.983	0.723	0.735	0.955	0.471	0.664	0.903	0.614	0.657	0.919
9	ModFOLDclust2	QA214_2	0.761	0.851	0.985	0.72	0.751	0.954	0.393	0.644	0.901	0.639	0.677	0.924
10	Pcons-net	QA432_2	0.66	0.66	0.979	0.633	0.662	0.954	0.324	0.537	0.923	0.612	0.616	0.925

Table 4.8. Independent benchmarking of stage 2 global scoring EMA methods in CASP12. The top 10 groups are shown. Table is sorted by the LDDT-AUC score. Data are from http://predictioncenter.org/casp12/qa_aucmcc.cgi. Adapted from Maghrabi and McGuffin, (2017).

			GDT-TS		LDDT		CAD(AA)		SG	
Rank	**Gr.Name**	**Gr.Model**	**No. Targets**	**Score**	**No. Targets**	**Score**	**No. Targets**	**Score**	**No. Targets**	**Score**
1	MUfoldQA_C	QA318_1	47	0.82	45	0.129	66	0.304	43	0
2	ModFOLD6_rank	QA072_1	47	1.077	45	0.129	66	0.39	43	0.25
3	qSVMQA	QA120_1	47	1.186	45	0.129	65	0.487	43	0.325
4	ModFOLD6_cor	QA360_1	47	1.279	45	0.434	66	0.806	43	0.871
5	ModFOLD6	QA201_1	47	1.279	45	0.434	66	0.557	43	0.871
6	MUfoldQA_S	QA334_1	47	2.558	45	1.417	66	0.903	43	2.755
7	Pcons-net	QA432_1	42	2.949	41	1.434	57	0.767	40	1.973
8	QASproCL	QA267_1	47	3.644	45	1.619	66	1.502	43	1.938
9	SVMQA	QA208_1	47	3.557	45	1.723	65	0.739	43	2.93
10	ProQ3	QA213_1	47	4.244	45	2.148	66	1.113	43	3.103

Table 4.9. Independent benchmarking of global scoring EMA methods using specific targets from CASP12. For each score, only the targets with the best model scoring above the threshold (GDT-TS, SG: 40.0; LDDT, CAD(AA): 0.4) were considered. The top 10 groups shown. Table is sorted by the LDDT score. Data are from http://predictioncenter.org/casp12/qa_diff2best.cgi. Adapted from Maghrabi and McGuffin, (2017).

4.3.3. Further benchmarking and cross-validation with official CASP11 data

Prior to CASP12, the ModFOLD6 methods were also cross-validated using the CASP11 data to gauge performance versus the component methods, in terms of local (Tables 4.10 – 4.12) and global scores (Table 4.13 and 4.14). In all target categories, the ModFOLD6 local scores significantly outperform the component methods. Similarly, significant performance gains can be made from combining component global scores, both in terms of cumulative GDT-TS of the top ranked models (with ModFOLD_rank) and in terms of assigning absolute accuracy values (with ModFOLD6_cor) (Figure 4.7).

Method	Pearson	Spearman	AUC	AUC 0-0.1	StdErr
ModFOLD6	0.6657	0.5478	0.856	0.0505	0.00064
ModFOLD5_single	0.6472	0.5285	0.8357	0.0479	0.00067
ModFOLDclustQ_single	0.6062	0.5225	0.7975	0.045	0.00072
DBA	0.5543	0.3709	0.7856	0.0408	0.00074
ProQ2	0.3848	0.3642	0.7512	0.025	0.00077
SSA	0.1571	0.1506	0.6242	0.0109	0.00084
CDA	0.1769	0.2012	0.6187	0.0114	0.00084

Table 4.10. FM Cross-validation of ModFOLD6 versus its component methods using CASP11 data. local scores evaluated on stage1 and stage 2 models for targets with FM domains (861605 residues, 147428 high quality, 714177 low quality). A 3.5Å CA atom cut-off was used to define high quality residues (<=3.5Å are high quality, >3.5 Å low quality). Pearson = Pearson's r. Spearman - Spearman's rho. AUC = Area Under ROC Curve. AUC 0-0.1 = Area Under the ROC curve with False Positive Rate <= 0.1. StdErr = Standard Error in AUC score. Table is sorted by the AUC score. Adapted from Maghrabi and McGuffin, (2017).

Method	Pearson	Spearman	AUC	AUC 0-0.1	StdErr
ModFOLD6	0.7111	0.6782	0.8736	0.0482	0.00088
ModFOLD5_single	0.6664	0.6704	0.8512	0.0444	0.00094
ModFOLDclustQ_single	0.6212	0.6694	0.815	0.0383	0.00102
DBA	0.5929	0.5839	0.8122	0.0374	0.00103
ProQ2	0.4653	0.4514	0.7888	0.0257	0.00107
CDA	0.2599	0.2808	0.6825	0.0154	0.0012
SSA	0.1697	0.1671	0.6226	0.011	0.00123

Table 4.11. TBM Cross-validation of ModFOLD6 versus its component methods using CASP11 data. - local scores evaluated on stage1 and stage 2 models for targets with TBM hard domains (344169 residues, 70689 high quality, 273480 low quality). The same criteria in Table 4.10 was applied. Adapted from Maghrabi and McGuffin, (2017).

Method	Pearson	Spearman	AUC	AUC 0-0.1	StdErr
ModFOLD6	0.8337	0.7975	0.934	0.0645	0.00017
ModFOLD5_single	0.8192	0.7982	0.9282	0.063	0.00017
DBA	0.7674	0.7411	0.9037	0.0524	0.0002
ModFOLDclustQ_single	0.7687	0.7494	0.9032	0.0526	0.0002
ProQ2	0.6467	0.6342	0.8496	0.0374	0.00026
CDA	0.3848	0.4024	0.7186	0.022	0.00036
SSA	0.1835	0.1625	0.604	0.0097	0.00041

Table 4.12. Cross-validation of ModFOLD6 versus its component methods using CASP11 data. Local scores evaluated on stage1 and stage 2 models for targets without FM or TBM hard domains (2036888 residues, 1367703 high quality, 669185 low quality). A 3.5 Å cut-off was used to define high quality residues (<=3.5Å are high quality, >3.5 Å low quality). Pearson = Pearson's r. Spearman - Spearman's rho. AUC = Area Under ROC Curve. AUC 0-0.1 = Area Under the ROC curve with False Positive Rate <= 0.1. StdErr = Standard Error in AUC score. Table is sorted by the AUC score. Adapted from Maghrabi and McGuffin, (2017).

QA method used for model ranking	ΣGDT-TS	StdErr in GDT-TS
Maximum possible GDT-TS	48.4655	0.0273
ModFOLD6_rank	44.4149	0.0277
ModFOLD6	43.1859	0.028
ProQ2	42.9578	0.027
ModFOLDclust2	42.6768	0.0294
CDA	40.4575	0.0281
ModFOLD5_single	40.059	0.0291
DBA	40.0457	0.029
ModFOLDclust2_single	40.0328	0.0292
ModFOLDclustQ_single	39.9194	0.0291
SSA	39.3166	0.0268
Random	37.87	0.0284

Table 4.13. Ranking/selection global score benchmarks using CASP11 data. ModFOLD6_rank versus component global scoring methods. Cumulative GDT scores and standard error. 84 targets with structures, models from QA round1 and round2 combined. The maximum possible GDT-TS is the cumulative score obtained by selecting the best model available for every target. The StdErr in GDT-TS is $\sigma/\sqrt{n}$, where σ is the standard deviation and n is the number of targets (84). Table is sorted by the ΣGDT-TS. Adapted from Maghrabi and McGuffin, (2017).

	GDT-HA (Mirjalili and Feig, 2013)			GDT (Zemla et al., 1999)			MaxSub (Siew et al., 2000)			TM-score (Zhang and Skolnick, 2004)		
Method	**R**	**Rho**	**Tau**	**R**	**Rho**	**Tau**	**R**	**Rho**	**Tau**	**R**	**Rho**	**Tau**
ModFOLD6_cor	0.9045	0.9288	0.7675	0.925	0.9303	0.7793	0.9285	0.9335	0.7789	0.9266	0.9302	0.7816
DBA_res_global	0.8962	0.9157	0.7405	0.9177	0.9192	0.7526	0.923	0.9245	0.7585	0.9216	0.9212	0.7586
ModFOLD6_single_res_global	0.8793	0.9143	0.7451	0.9121	0.9181	0.7591	0.9133	0.9221	0.7608	0.9178	0.9196	0.7644
ModFOLDclust2_single_orig_global	0.899	0.9234	0.7595	0.9152	0.922	0.7649	0.9205	0.9256	0.7652	0.9157	0.9209	0.7647
ModFOLD5_single_orig_global	0.886	0.9155	0.7469	0.9098	0.9155	0.7539	0.916	0.9223	0.7618	0.9144	0.9162	0.757
ModFOLD5_single_res_global	0.8905	0.9198	0.754	0.91	0.9203	0.7625	0.9186	0.9263	0.7682	0.9137	0.9211	0.765
ModFOLDclustQ_single_res_global	0.8996	0.92	0.7524	0.9054	0.9165	0.7536	0.9094	0.9161	0.7481	0.9003	0.9122	0.749
ModFOLDclustQ_single_orig_global	0.8995	0.92	0.7524	0.9053	0.9165	0.7536	0.9094	0.9161	0.748	0.9003	0.9122	0.749
ProQ2_res_global	0.6878	0.7319	0.5272	0.7182	0.7417	0.5404	0.7174	0.7427	0.5404	0.7239	0.7446	0.5452
CDA_res_global	0.6354	0.7192	0.527	0.6703	0.731	0.5407	0.6727	0.73	0.5369	0.6746	0.7333	0.5431
SSA_res_global	0.517	0.5595	0.3838	0.5348	0.5585	0.384	0.5318	0.5601	0.386	0.5324	0.5539	0.3816

Table 4.14. Correlation global score benchmarks using CASP11 data. Correlations between predicted and observed global scores. ModFOLD6_cor versus component global scoring methods. R = Pearson's r. Rho - Spearman's rho. Tau = Kendall's tau. The analysis is carried out on all of the 84 targets with known structures. The models from QA stage 1 and stage 2 were combined and all duplicate models (models from stage 1 occurring also in stage 2) were removed. The table is sorted by the TM-score R value. Adapted from Maghrabi and McGuffin, (2017).

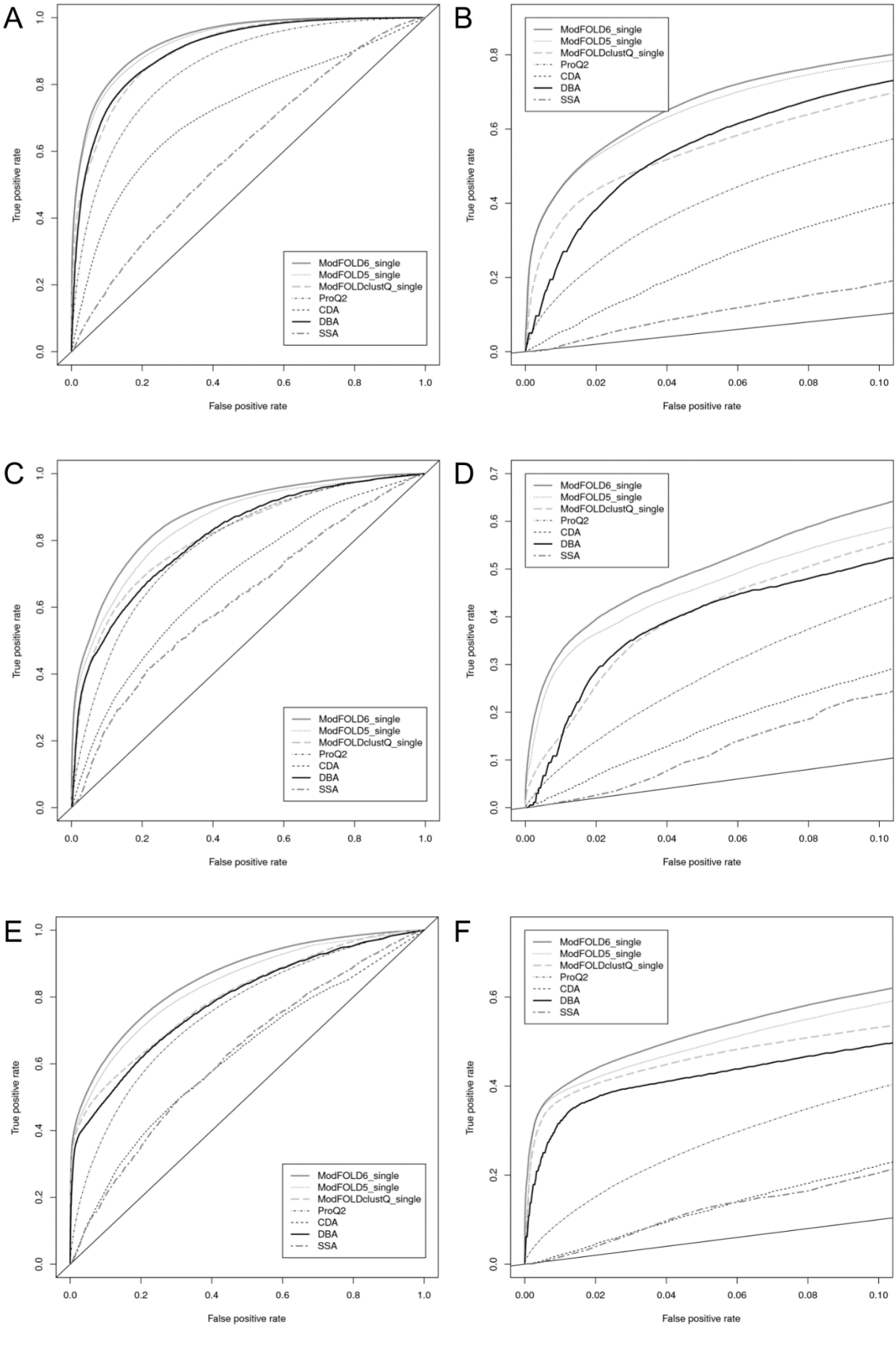
A
B
C
D
E
F
True positive rate
False positive rate
ModFOLD6_single
ModFOLD5_single
ModFOLDclustQ_single
ProQ2
CDA
DBA
SSA

Figure 4.7. Line graphs representing cross-validation of ModFOLD6 local scores versus its component methods using CASP11 data. ROC plots for the data shown in Tables 12 - 14. A true positive is defined as a residue correctly identified to be of low quality (> 3.5Å from the native structure). The full-length chains were used for the official CASP11 QA analysis, and so they contain multiple domains of varying difficulty, with each domain being officially classified as either FM, TBM-hard or just TBM (easy). In order to demonstrate the performance of ModFOLD6 on easy, medium and hard CASP11 targets, we compare ROC plots for 3 different subsets of full length models: 1. Targets without any TBM-hard or FM domains (i.e. the models for easy targets), 2. Targets with TBM-hard domains (i.e. models for medium/hard targets) and 3. Targets with FM domains (i.e. models for hard targets). The analysis is carried out on all of the 84 targets with known structures. The targets with FM domains are: T0761, T0763, T0767, T0771, T0775, T0777, T0781, T0785, T0789, T0790, T0791, T0793, T0794, T0799, T0802, T0804, T0806, T0808, T0810, T0814, T0820, T0824, T0826, T0827, T0831, T0832, T0834, T0836, T0837, T0855. The targets with TBM-hard domains are: T0774, T0781, T0793, T0799, T0800, T0812, T0814, T0830, T0831, T0848. The targets without FM or TBM-hard domains include all remaining targets. Domain definitions are from: http://www.predictioncenter.org/casp11/domains_summary.cgi. The models from QA stage 1 and stage 2 were combined and all duplicate models (models from stage 1 occurring also in stage 2) were removed. (A) The full ROC plot for targets without FM or TBM-hard domains. (B) ROC plot with FPR <= 0.1 for targets without FM or TBM domains. (C) Full ROC plot for targets with TBM-hard domains. (D) ROC plot with FPR <= 0.1 for targets with TBM-hard domains. (E) Full ROC plot for targets with FM domains. (F) ROC plot with FPR <= 0.1 for targets with FM domains. Adapted from Maghrabi and McGuffin, (2017).

4.3.4. Comparisons between the top CASP12 EMA methods

An independent detailed analysis of the CASP12 EMA methods is provided in the official EMA assessment paper (Kryshtafovych et al., 2018). In this section, we refer to the results provided pertaining to our methods and we also show an additional analysis based on the correlation between different scores for different types of competing methods. A summary of the top EMA methods in CASP12 are shown in Table 4.15.

Methods	Type	Comment about global performance	Comment about local performance
MESHI (Amir et al., 2008)	Single	Top model selection	N/A
MESHI_con (Amir et al., 2008)	Singlea	Top model selection	N/A
ProQ2 (Ray et al., 2012)	Single	Good model selection	Acceptable local scores
ProQ3 (Uziela et al., 2016)	Single	Top model selection	Good local scores
SVMQA (Manavalan and Lee, 2017)	Single	Top model selection	N/A
ModFOLD6 (Maghrabi and McGuffin, 2017)	Quasi-single	Balanced performance	Good assignment of local scores
ModFOLD6_rank (Maghrabi and McGuffin, 2017)	Quasi-single	Acceptable model selection	Identical to ModFOLD6
ModFOLD6_cor (Maghrabi and McGuffin, 2017)	Quasi-single	Best absolute but suboptimal model selection	Identical to ModFOLD6
qSVMQA (Manavalan and Lee, 2017)	Quasi-single	Assignment of the absolute score is not accurate.	N/A
ModFOLDclust2 (McGuffin and Roche, 2010)	Clustering	Good assignment of absolute global scores but suboptimal model selection	Top assignment of local scores
Pcons (Lundström et al., 2001)	Clustering	Good assignment of absolute global scores	Top assignment of local scores
Pcomb-domain (Lundström et al., 2001)	Combined	Good assignment of absolute global scores, requires good domain prediction	Top assignment of local scores
Wallner (Wallner and Elofsson, 2007)	Combined	Good assignment of absolute global scores	Top assignment of local scores

Table 4.15. Summary of the best performing QA methods in CASP12 and comments about their strength and weaknesses. Note: MESHI_con is not pure single methods but requires multiple models to average the predictions. Adapted from Elofsson et al., (2018).

4.3.4.1. Estimation of global accuracy

The results above showed that 3 single EMA methods were ranked highest when identifying the best model. These methods were ProQ3, SVMQ and MESHI. The average error which means the difference between the GDT-TS of the selected model and the best GDT-TS was around 5 GDT-TS units. Each EMA method was ranked individually depending on the evaluation criteria. According to the assessment results, there was not a noticeable difference between the top methods (Kryshtafovych et al., 2018). When using these criteria, the best consensus and quasi-single methods were only marginally worse than the pure single methods. However, since CASP11, these results interpret a significant progress in single model method performance.

4.3.4.2. Distinguishing good models from bad

The results also showed that the best EMA approaches according to GDT-TS usually use consensus or quasi-single methods and combine them with single model approaches. The top ranking 3 EMA methods were Wallner, Pcomb-domain and ModFOLD6_rank. They all use the single model method ProQ2 as part of their scoring. The Wallner and Pcomb-domain scores are weighted sums of ProQ2 and Pcons scores, while our ModFOLD6_rank method uses ProQ2 together with many other scores. Although such methods are statistically better (Kryshtafovych et al., 2018), the much simpler pure consensus methods Pcons and ModFOLDclust2 were not far behind, ranked sixth and ninth respectively when using lDDT (Table 4.8).

4.3.4.3. Ranking of models

All targets were evaluated individually in order to test the methods ability in ranking the top models for each target. This evaluation was carried out using the per target correlation (i.e. the correlation of predicted and observed accuracy for each target). By looking at Figure 4.8, we can see the distribution of per target correlations for all of the top CASP12 methods with the 3 different model accuracy estimation measures (lDDT, CAD, and GDT-TS) as well. This ranking was sorted according to the median correlations. The results showed that individual rankings of the methods are quite different depending on the accuracy measure which was used. I can also be seen that consensus and quasi-single based methods clearly outperformed the single model accuracy estimation methods when using GDT-TS (Zemla, 2003). Contrarily, we can see that the best

correlation was obtained with ProQ3 when using CAD (Olechnovič et al., 2013) or lDDT (Mariani et al., 2013), and all the top methods were single model accuracty estimations. Similar differences in ranking could be found in the AUC analysis on the CASP homepage (http://predictioncenter.org/casp12/qa_aucmcc.cgi). Moreover, when using GDT-TS we can see that ProQ3 was ranked 20th, but it jumped up to the 7th when CAD was used instead of GDT-TS. On the other side, Pcons was ranked the 4th using GDT-TS but the 12th when using CAD. Such results are interesting as they showed that "pure" consensus methods such Pcons and ModFOLDclust2 only show a modest per target correlation with CAD and lDDT.

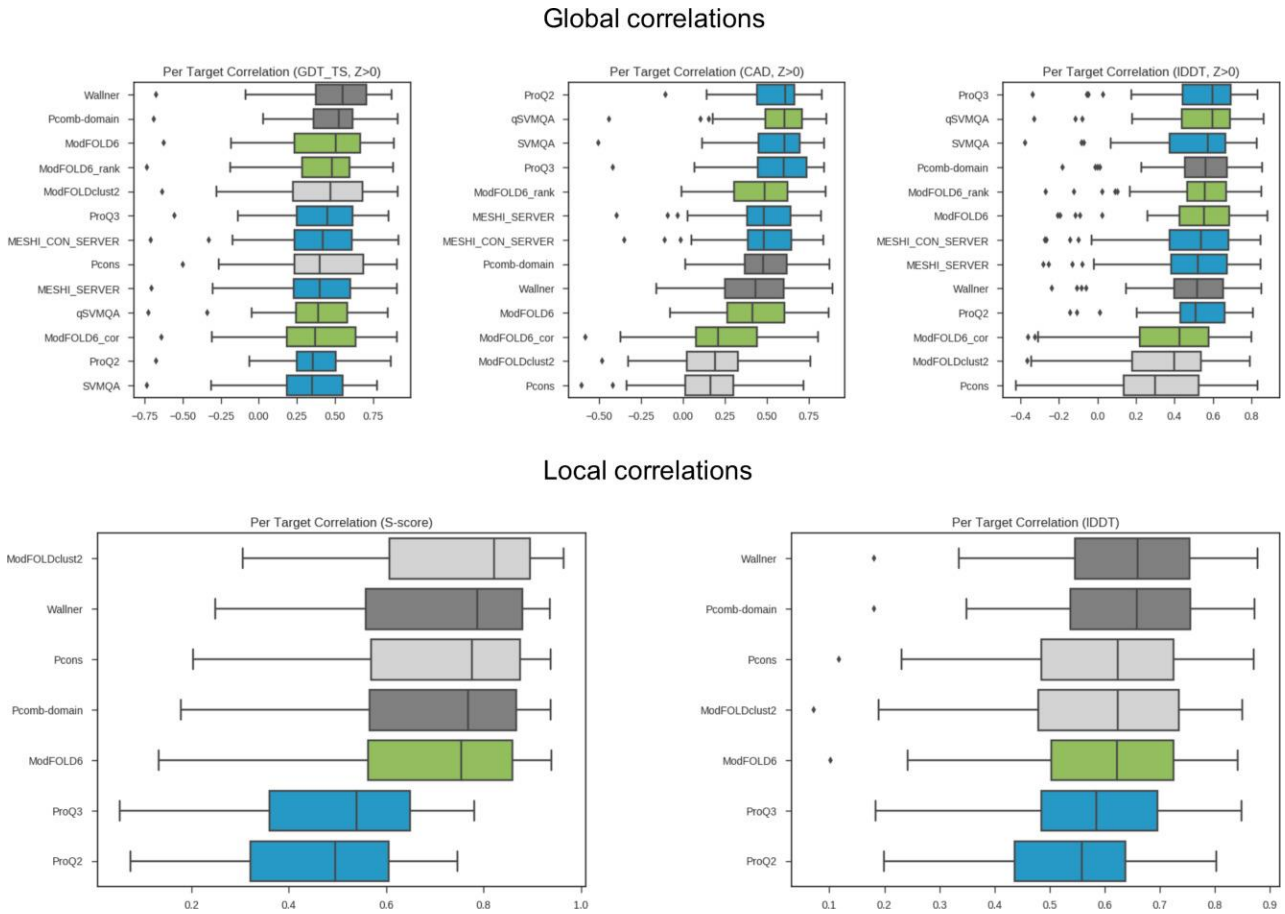

Figure 4.8. Boxplots of per target correlation for the top CASP12 EMA method versus GDT-TS, CAD, and lDDT, (A-C) global evaluations, (D, E) local evaluations. To avoid bias from bad models only models with $Z > 0$ are included in the global analysis. For local correlation CAD values were not available so only the distances, turned into S-scores, and lDDT values are compared. Single-model methods are colored blue, quasi green, clustering light gray and combination models dark gray. Using GDT-TS the clustering-based methods are slightly better than the single-model predictors, while this is not the case using the alternative measures CAD and lDDT. Clustering methods benefit from having low-quality models in the pool while the single model methods appear better at ranking higher quality models. For both local measures the single-model evaluation methods have lower correlation than the superposition-based ones, but the difference in correlation is smaller when using lDDT. Adapted from Elofsson et al., (2018).

4.3.4.4. Similarities in model accuracy estimation scores

It was also important to know how different EMA methods produced similar model accuracy estimation scores. Figure 4.9 shows the correlations between predicted accuracy estimates from all the EMA servers. The methods were then clustered using WPGMC with the median correlation as linkage. The results showed that all methods which use some sort of consensus (quasi-single or consensus) were clustered except qSVMQA. The separation within this group was not between quasi-single and consensus methods, but rather between the methods that primarily use consensus and those which combine the consensus score with ProQ2 (Pcomb-domain, ModFOLD6_rank, Wallner, and ModFOLD6). It can be seen that ModFOLD6_cor was more similar to the pure consensus methods (Pcons and ModFOLDclust2) than the other combined methods. The method did not use ProQ2 global scores directly in its classification. The combined EMA methods results are also closer to all the single methods than the pure consensus methods, that may be due to including single methods to their pipelines as well.

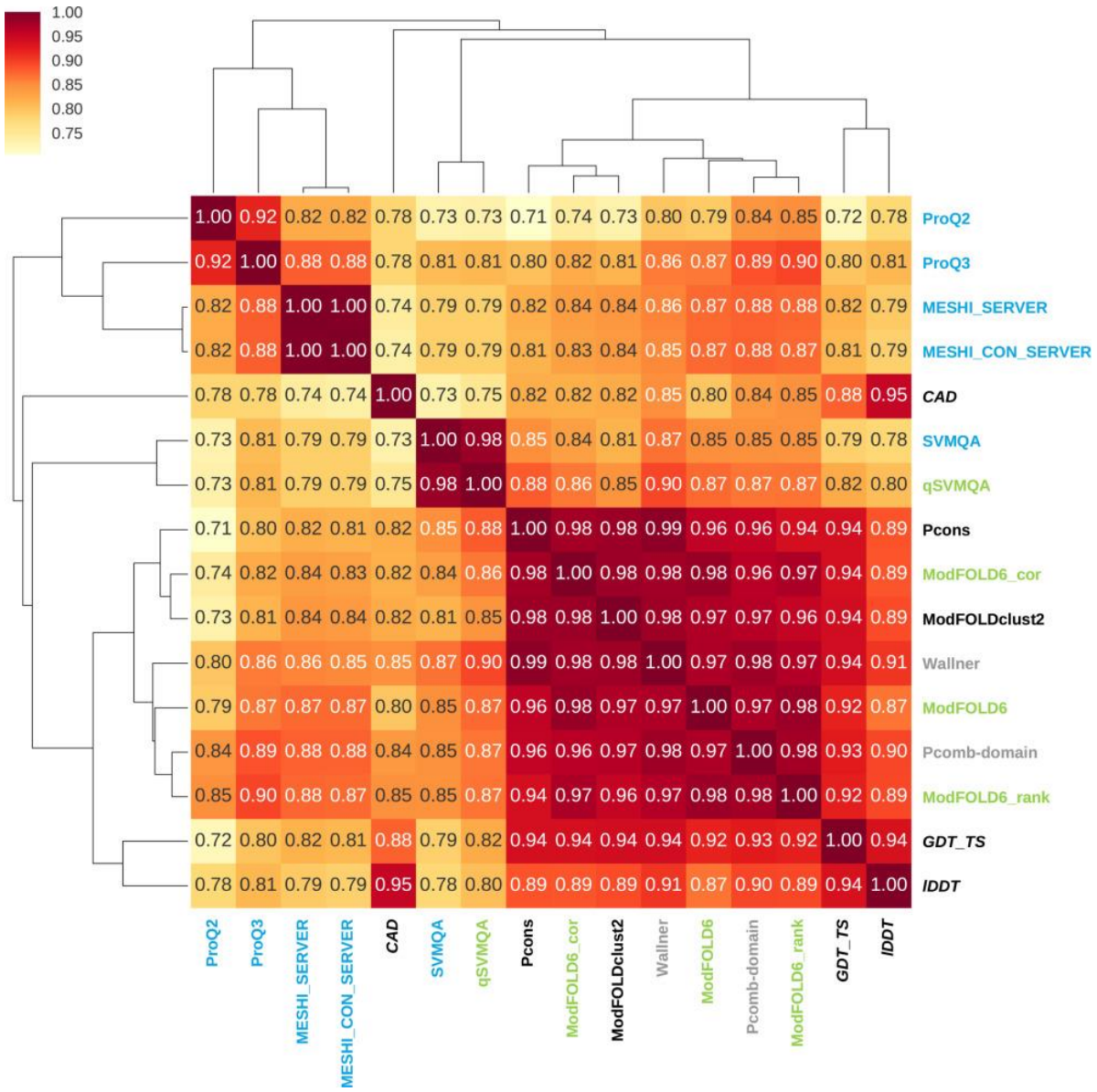

Figure 4.9. Pairwise correlations between predicted global accuracy scores from different methods and actual accuracy scores according to 3 measures. The methods are clustered hierarchically using WPGMC algorithm with the median correlation as similarity measure. Methods are coloured as follows. Dark grey, pure consensus methods; light grey, combined single/consensus methods; green, quasi-single methods; and blue pure single methods. It can be noted that both quasi, pure, and combined consensus methods are very similar (cc > 0.94), while the single model quality methods are more different (cc < 0.90 between the groups). ProQ2 is the real outlier having a cc < 0.82 to most methods. Interestingly ProQ2 and ProQ3 are less similar to each other than any pair of consensus-based methods. It can also be noted that the combined methods are more similar to the single-model methods than the pure consensus methods (Pcons, ModFOLDClust2). Adapted from Elofsson et al., (2018).

When looking at the performance of the single model accuracy estimation methods we found that they have the largest performance diversity. The least method in showing similarities to the others was SVMQA, it showed a more similar scores to the consensus methods than to any other single model accuracy estimation method. The other 3 methods were more correlated, with the newer methods ProQ3 and MESHI showing the highest correlation. It can also be noted that in general ProQ2 showed the lowest correlation with the consensus methods, being by that the outlier.

The 3 different quality measurements (GDT-TS, CAD, and lDDT) were also compared together, showing that they do not correlate with each other better than the consensus methods with GDT-TS as seen in Figure 4.9. The correlation between the quality measures CAD and GDT-TS was 0.88; while the correlation of the predicted values from the consensus methods to GDT-TS is 0.92 or higher. It was clear that the accuracy of model quality estimation is getting close to a point where they challenge the notion of measuring the quality of model given a known native structure even when some of the problems might origin from domain division, as mentioned in Wallner sections in Elofsson et al., 2018.

4.3.4.5. Comparison of local accuracy estimations

With regards to the estimation of local accuracy, we saw that the pure consensus methods were the best performers, followed by quasi-single model approaches (Kryshtafovych et al., 2018). A heat map in Figure 4.10 showed the correlation between all local predictions by the EMA methods discussed in this section. Unfortunately, ProQ2 and ProQ3 from all the evaluated single predictors were the only methods which produced local predictions, nevertheless the trend was similar as for the global methods. All the consensus and quasi-single methods provided very similar accuracy estimates, while the 2 single model methods were outliers. It was clear from this analysis that the consensus methods correlated better with S-score (cc ~0.85) than with lDDT (cc ~0.77). As the consensus methods are based on superposition algorithms, similar to those used when calculating the S-score, this might not come as a surprise. Interestingly both ProQ2 and ProQ3 correlated better with lDDT (cc ~0.71) than with S-score (cc ~0.65). It can also be noted that ProQ3 correlated better than ProQ2 with both lDDT and S-score. This highlights the improvements achieved in single model quality estimates since CASP11.

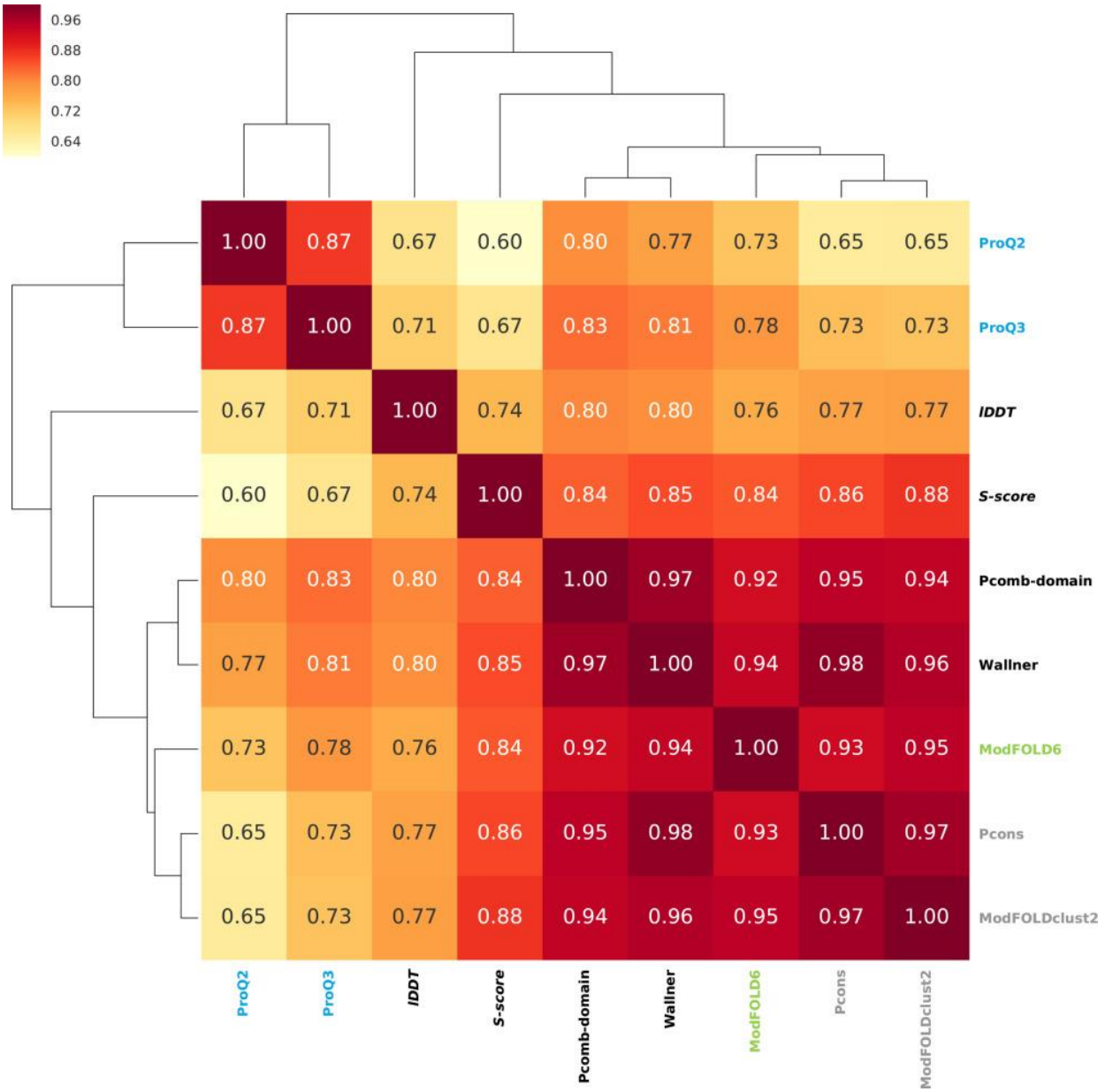

Figure 4.10. Pairwise correlation between local predicted S-scores. The correlation was calculated using the predicted distance using S-score formula (see above) with $d_0 = 5$ and local lDDT values (unfortunately local CAD scores were not available). Only methods that predicted local quality are included. As the ModFOLD6 methods only differ in their global scores and provide identical local estimates they were all represented by the ModFOLD6 method. Methods are coloured as follows. Dark grey, pure consensus methods; light grey, combined single/consensus methods; green, quasi-single methods; and blue pure single methods. Adapted from Elofsson et al., (2018).

4.4. Conclusions

The ModFOLD6 series of methods (ModFOLD6, ModFOLD6_rank and ModFOLD6_cor) perform particularly well in terms of assigning absolute global accuracy values. As expected, the ModFOLD6_cor variant is the best of these as it was optimized for this task. The ModFOLD6 series of methods also perform competitively with clustering approaches for differentiating between good and bad models; the ModFOLD6_rank method being the best of these, which is only outperformed by 2 clustering groups (Wallner and Pcomb-domain). Furthermore, as we anticipated, the ModFOLD6_rank variant is better at selecting the top models than the ModFOLD6 and ModFOLD6_cor variants; however, it is outperformed by the latest pure-single model methods. Overall, in terms of global scores, the ModFOLD6 variants rank within the top 3 methods for nearly every global benchmark according to lDDT and CAD scores, as well as ranking within the top 10 according to other scores. The ModFOLD6 server provides users with intuitively presented, high accuracy estimates of local and global quality of 3D protein models and it implements each of the methods tested in the CASP12 experiment. The ModFOLD6 server has also been independently verified, via the CAMEO project, showing a significant improvement on our previous published server as well as taking the lead over other public published methods, in terms of local accuracy estimates.

It is gratifying to see progress in CASP12 from many groups in both pure- and quasi-single model approaches to estimate model accuracy. However, it is also clear there is still room for improvement of our methods. For instance, we are outperformed in terms of model selection by the newer pure single model methods. Further integration of methods is probably needed. Different methods are clearly better suited for different aspects of model accuracy estimation, therefore all approaches to the problem are still important to pursue. Perhaps the most difficult problem faced by all groups is how to optimize a global score for all aspects of model accuracy estimation, as there seems to be no one-size-fits-all solution presently. One potential solution to this might be to use a deep learning approach that outputs multiple scores depending on the intended use case. A global score for ranking models on a per-target basis, irrespective of the observed model-target similarity scores, is clearly very useful, if it can consistently select the better models. On the other hand a global score that can produce a near 1:1 mapping between predicted and observed scores, that is consistent across all targets, will allow us to assign accurate confidence scores to individual models (which is arguably more useful to an experimentalist than a top ranked, but nevertheless

poor quality, model). As model accuracy estimation methods continue to improve and approach perfect optimisation for each use case, eventually the scores may possibly converge on a single answer.

It is clear from the CASP12 results that there has been progress in single model accuracy estimations since CASP11. 3 new methods, SVMQA, MESHI, and ProQ3, are all better than the best single model method in CASP11 (ProQ2). These methods are also better at selecting the top-ranked model compared to consensus-based methods. However, quasi-single model method and consensus methods are still superior when it comes to distinguishing correct and incorrect models as well as for local predictions. In those targets that have a wide spread of quality there is a clear distinction between the correlations of single and consensus methods with the later performing better. These are typically subunit of protein complexes, for which templates are available. Here, estimating the accuracy of a single model might not make sense without taking the entire complex into account. In CASP12 this is most dramatic for target T0865, where correlations for consensus-based methods are high and correlations for all single model methods are negative. By comparing the predictions to each other it is seen that all consensus and quasi-single methods actually are very similar, while there is larger variation between the single methods, hence combining them may provide additional value in the future.

During this evaluation we noted issues for multi-domain targets where the individual domains are correct but not their relative arrangement. Here, the GDT-TS score (and any superposition-based score) is based on the superposition of the largest domain. This causes problems when the evaluation is not domain based. For model quality estimations the problem is most notable when evaluating local QAs. It could therefore be useful, in future CASPs, to also use CAD or lDDT in order to evaluate the quality of a model without using domain division. We do also notice that single model estimation methods perform better when assessed with CAD or lDDT.

Training to lDDT scores, rather than the S-scores as the target function is one option to improvement in both local and global scores if lDDT scores are to be used for evaluation (more details about lDDT impacts will be described in Chapter 6). However, other optimisation strategies can also be pursued to gain more improvement in terms of global and local accuracy, one of which is to go beyond the simple OMS by combining scores using Deep Artificial Neural Networks. This technique has been testified in recent years to dominate several areas of studies making significant improvements in many researches, which attracted our interest towards it in the later studies.

Chapter 5

Deep Artificial Neural Network Parameterisation

5.1. Background

In Chapter 3, we had a glance at Deep Artificial Neural Networks technique. We saw how this learning model can process information via networks in order to solve problems, the same way that a biological brain does. Such networks have been remarkably useful in solving problems in different fields. They have been used in classifying handwritten digits with better than 98% accuracy (McDonnell et al., 2015). However, there are more difficult computing problems that need more advanced DANNs.

The revolution in Deep Artificial Neural Networks began in the last decade when deep learning was considered as the key component. Its popularity started in the beginning of 2006 (Hinton et al., 2006) (Hinton and Salakhutdinov, 2006). The first major breakthrough in deep learning was achieved in speech recognition, when the DANNs designed model outperformed a technique called HMM-GMM used to dominate this field for many years (Hinton et al., 2012).

The success behind this technology was due to the rapid improvement of hardware resources such as GPGPUs, as well as the improved theory, starting with unsupervised pre-training and deep belief nets. Nowadays, DANNs have become popularly used with impressive results in many areas such as pattern recognition, image analysis, and object detection to name a few areas (Wang, 2016).

Although the impressive results that DANNs have achieved in several important application areas, the parameterisation step of this technique is still challenging. Deciding what number of layers to choose, how many neurons per layer, and how much time should the training iterate, all of these parameters and others need training and testing which consume computing time and effort.

DANNs excel at classification problems, such as with the famous MNIST dataset of handwritten digits (Deng, 2012), where the inputs are placed into one of several categories. The other usage is regression, where the output is a number on a continuous scale.

5.2. Objectives

For the purposes of this experiment, DANNs were applied for scoring protein model quality generated by different methods and comparing the output to the observed model quality scores (measured by GDT-HA, GDT-TS, MaxSub or TM-score). The primary aim of this experiment is to determine optimal hyperparameters to get the best possible network for ranking the top models, and the best correlation between the network output and the observed model quality score.

Examples of the various hyperparameters which were used within this experiment are given in Table 5.1.

Hyperparameter	Datatype	Range	Notes
Number of Neurons per Hidden Layer	Integer	1 -> ∞	Each layer in the network can have different numbers of neurons. Adding neurons slows down the network but increases processing power.
Number of Hidden Layers	Integer	1 -> ∞	Adding layers should increase the processing power, but considerably slows down the network.
Number of Training Cycles	Integer	1 -> ∞	Training loss is reduced with higher numbers of cycles unless overfitting occurs, but training time is also increased.
Learning Rate	Real Number	0 -> 1	This value controls how much weights and biases change during training.
Dropout	Real Number	0 -> 1	Represents the probability for each neuron to be "dropped" during training. This prevents overfitting of the data.
Regularisation Parameter	Real Number	0 -> ∞	Helps prevent overfitting by preventing weights from growing too large.
Identity of Inputs	List of input scores	N/A	While using all 10 inputs gives the network access to the most information, large number of internal parameters and conflicting information may result in a loss of accuracy.
Identity of Training Target Score	Output score identity string	N/A	Learning via backpropagation of a different score could improve the generalisation of the network.

Table 5.1. List of the various hyperparameters within our neural network. The data type for each parameter, their range of values and a brief description of what aspect of the network they control are presented.

To determine the hyperparameters of DANNs, several broad strategies can be conducted. One of the simplest available strategies is the grid search where all possible combinations of values for the network hyperparameters are explored, and then a comparison is performed over the network accuracies in order to determine the optimal combinations (Hsu et al., 2003). This strategy has a disadvantage with the amount of time required to run the network on all possible combinations. Assuming that the network will take 10 minutes to run, by looking at only 10 values for each hyperparameter listed in Table 5.1 a grid search would take over 1900 years to be completed! A far more time-efficient method is called the random search. This is similar to a grid search, except a far smaller number of values for each hyperparameter are chosen using random or quasi-random

methods. Random searches work effectively as long as there are enough datapoints to provide good coverage of the range of values for each hyperparameter. The main advantage is that it is considerably faster than a full grid-search (Bergstra and Bengio, 2012). Hand-tuning hyperparameters is another option, which makes use of the human ability to analyse and learn from results but is also the least automated option. Hand-tuning can also be combined with grid searches and random searches to make a semi-automated method which takes advantage of the human analytical ability and the good coverage of the automated methods (Bergstra and Bengio, 2012). Bayesian optimisation is a more advanced method of hyperparameter determination. A Gaussian process model is used to predict new values for hyperparameters. These models assume that similar inputs give similar outputs, they are particularly useful for predicting relationships where little prior knowledge is available (Snoek et al., 2012).

5.3. Materials and Methods

5.3.1. Raw Data

For DANNs training and testing, raw data were obtained by assessing 16483 models from the QA category of CASP11 using the QA programs listed in Table 5.2 to obtain 10 quality scores for each model. Observed quality scores were also included in the collected data, they were used as training targets for the network. After removing models for which there was no native structure, 14103 models remained, of which there were 84 unique protein targets.

5.3.2. Neural Network Inputs

The inputs for the DANNs were protein global quality scores generated by 10 different prediction servers. For the purpose of simplicity during the project, each input was assigned a short key which are summarised in Table 5.2. The DANNs inputs could be changed so that different numbers and combinations of inputs could be utilised.

Input Program Name	Program Description	Short Name
ModFOLD5_single_orig_global (McGuffin et al., 2015)	Quasi-single model technique. Submitted models are compared to those predicted by the IntFOLD3 server.	V3
ModFOLDclustQ_single_orig_global (McGuffin and Roche, 2010) (Eastwood et al., 2001)	Another quasi-single model program which uses the Q score for model comparisons.	V4
ModFOLDclust2_single_orig_global (McGuffin and Roche, 2010)	Integrates the scores of V3 and V4 into a single score.	V5
ModFOLD5_single_res_global (McGuffin et al., 2015)	The per-residue errors, predicted by ModFOLDclust are summed and divided by sequence length.	V6
ModFOLDclustQ_single_res_global (McGuffin and Roche, 2010)	The per-residue errors, predicted by ModFOLDclustQ are summed and divided by the sequence length.	V7
ProQ_res_global (Uziela and Wallner, 2016)	ProQ2 uses a support vector machine to predict model accuracy by combining different features of the model.	V8
CDA_res_global (Jones et al., 2015)	Predictions are made by the MetaPSICOV method based on contact distance agreement.	V9
DBA_res_global (Jones and Cozzetto, 2015) (Maghrabi and McGuffin, 2017)	Disorder B-factor agreement compares the predicted disordered regions between DISOPRED3 and ModFOLDclust.	V10
SSA_res_global (Maghrabi and McGuffin, 2017) (Buchan et al., 2013) (Kabsch and Sander, 1983)	Secondary Structure Agreement is a pure-single model method which compares the secondary structures predicted by PSIPED to those in the Dictionary of Secondary Structures of Proteins.	V11
ModFOLD6_single_res_global (Maghrabi and McGuffin, 2017)	A neural network which combines scores from V3, V4, V8, V9, V10 and V11 to generate local scores. The sum of the scores is divided by the sequence length.	V12

Table 5.2. Summary of the ten protein QA programs used as inputs during the experiments along. The short names V3-V12 refer to the default column vector names used in R data.

5.3.3. DANNs Training Targets.

The DANNs were built to output one of four different model quality scores. These were the possible training targets for the networks. The first measure is Global Distance Test – Total Score (GDT-TS), it is one of the most common measures of protein model quality. GDT-TS is calculated by finding the percentage of superimposed residues in the model and true structure within specified threshold distances of 1, 2, 4 and 8 Å (Li et al., 2011). Secondly, Global Distance Test – High Accuracy (GDT-HA), this is a more stringent version of GDT-TS which has been used in CASP experiments since CASP7 (Moult et al., 2007). The threshold distances for GDT-HA are half the size of those for GDT-TS, being 0.5, 1, 2 and 4 Å. The third measure is MaxSub, which is a scoring method which looks at how well the α-carbon atoms of the predicted model superimpose over the experimentally determined structure (Siew et al., 2000). Fourthly, Template Modeling Score (TM-score), this measure was developed as an extension to GDT and MaxSub. The scoring function uses a scale to eliminate the effect of protein size, and rather than using threshold cut-offs, all superimposed residue pairs are included in the score (Zhang and Skolnick, 2004).

5.3.4. Three-Fold Cross-Validation.

In order to maximise the amount of training and test data available, cross-validation was used. The Rscript program ParaPart1.R divided the raw data up into three training and testing sets, then randomised the sample order. Each testing set contains data for unique models not included in the testing set of the other two cross-validation sets. The overall result is that over the three validation sets all of the data were used for testing and all were used for training, but the same data was never used for training and testing on the same cross-validation set. All three sets were used to train and test independent DANNs and the results were re-integrated by ParaPart2.R in order to calculate the correlation and sum of top model scores (rank) for the network as a whole. Both Rscript programs were identical to those used in Chapter 2 and can be found in Appendix 2.

5.3.5. Neural Network Parameters

The DANNs software used for this experiment was TensorFlow 1.0 (Rampasek and Goldenberg, 2016), an open source software library which provides tools for the construction of flexible deep artificial neural networks in Python v2.7.5. The network architecture being used was that of a

multilayer perceptron feedforward neural network (Ruck et al., 1990). The version of the DANNs program used for parameterisation was designed to allow easy iteration over multiple values for a single hyperparameter in sequence.

The inputs and training targets of the DANNs can be defined by changing the "inputs" list variable and the "output" list variable within the program. Adding multiple training targets to the list will cause the program to iterate over each target in turn. Other standard parameters (hidden neuron number in each layer, initial training rate, number of training epochs, dropout probability and L2 regularisation parameter) can be changed by modifying their specific variables. Alternatively, the program can iterate over multiple values of any standard parameter by replacing the x, y and z variables with the parameter variable name and changing their range to either a range or list of desired values to test. Using this method, it was also possible to test every combination of values from two or three variables, although doing so was extremely time-consuming. This is equivalent to a grid search using one, two or three parameters at a time.

Advanced parameters (number of layers, loss function, optimiser and activation function) cannot be altered by simply changing a variable and involve making changes to the program. These parameters cannot be iterated over and must be changed manually each time.

5.3.6. Solutions to Overfitting

Overfitting can be an issue when the network learns the unique features of the dataset itself rather than the general pattern. When the training set is predicted extremely well by the DANNs, but the testing set is predicted very poorly, that is a typical sign of overfitting.

Several ways can be used to solve this issue, one way was by using the "Dropout" activation function. In TensorFlow, Dropout is a variable between 0 and 1, which indicates the probability that any neuron in the hidden layers will be temporarily removed during any single round of training (but not during testing). The same Dropout variable was applied to all hidden layers of the network. The random removal of neurons in the network is designed to prevent the neurons from co-adapting to complex patterns in the data and encourage generalisation instead (Srivastava et al., 2014). The Dropout activation function tool was implemented in the para.py (Appendix 3) program using tensorflow's tf.nn.Dropout method, and can be easily iterated over multiple values.

Another potential solution to overfitting, which is implemented in para.py was L2 regularisation (Schmidhuber, 2015). This is defined by the following formula: $\boldsymbol{C = C_0 + \frac{\lambda}{2n} \sum_w w^2}$ Regularisation alters the loss or cost function, where $\boldsymbol{C_0}$ is the unregularised cost, and $\boldsymbol{w}$ are the weights in the model. $\boldsymbol{n}$ is the number of samples in the training set, and $\boldsymbol{\lambda}$ is the regularisation factor, a variable which determines the strength of the regularisation. The purpose of L2 regularisation is to prevent the model from generating excessively large weights, which are not significantly changed by feeding the network conflicting data. A high value of $\boldsymbol{\lambda}$ places more emphasis on retaining low weights in the cost function while a high value of $\boldsymbol{\lambda}$ puts more emphasis on minimising the original cost function. $\boldsymbol{\lambda}$ is a simple variable within the para.py program, and can be iterated over multiple values, a value of 0 disables L2 regularisation.

5.3.7. Outcome Metrics

The two ways (correlation and ranking) of optimisation evaluation were conducted to assess and compare the performance of the hyperparameterised DANNs. Correlation was measured by Pearson's Correlation Coefficient between the network's predicted quality scores for the test set and the true quality score. The sum of top model scores is a metric to assess how well the network ranks the top models of the same protein structure to their observed quality score ranking. It is calculated by taking the sum of the GDT-HA score of each top predicted model and will herein be referred to as the "Rank Score". Ideally there should be a close relationship between the ranking score and correlation, but this is not necessarily the case. ModFOLD6, for example, uses a completely different set of parameters to get a good correlation and a good top model score, then uses another set to obtain a balanced result (Maghrabi and McGuffin, 2017).

5.4. Results and Discussion

5.4.1. Deep Artificial Neural Networks for Correlation (DANNs C)

5.4.1.1. Inputs and Training Targets

Most of the network hyperparameters were given placeholder values to start with. We began by looking at the different combinations of inputs which can be used for the networks as well as alternative training targets. The earlier studies indicated that using all the 10 MQAP methods together as a combined input does not give the best results for rank or correlation. We tested the top 10 combinations of inputs which were identified in Chapter 2 as producing the highest Pearson correlations (Figure 5.1). The combination which produced the highest correlation was V4+V10, with scores which were generally lower than the scores produced by our current optimum correlation scoring method (ModFOLD6_cor). The identity of the training target score also had an effect on the correlation for most input/target combinations. For most combinations, using GDT-HA as the training target score produced the best correlation. However, when using the V4+V10 input combination, MaxSub gives the highest correlation. The results in Figure 5.1 have clearly showed that using all 10 inputs is inferior to using a smaller selection despite the network having much more information available to it when using more inputs. This could be due to the different input scores providing conflicting information, making the predicted scores less reliable. Alternatively, co-adaptations between multiple scores could be occurring during training, resulting in the network identifying inappropriate patterns and applying this to the testing set (Hinton et al., 2012). While less information being input into the network means less processing is required, it also makes it harder to improve the network in the future when more quality-scoring methods are available as inputs. Instead of adding a new score into the network, it will have to be tested in all possible combinations with the other scores to identify any benefit of using it.

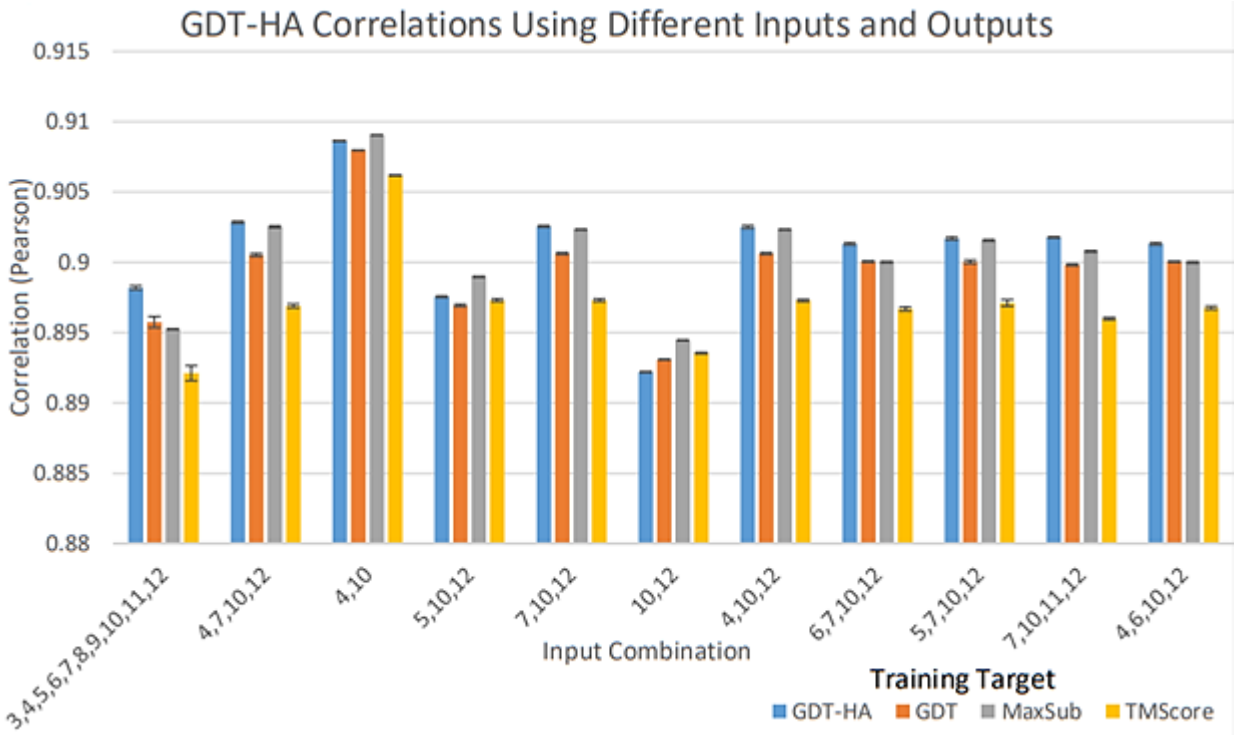

Figure 5.1. The effect of using different combinations of inputs scores and training targets scores on the results of the neural networks. Histogram showing the Pearson correlations produced by DANNs C for the top 10 input combinations. The results of using all 10 inputs is also shown for comparison. The clustered bars represent different network training targets. Error bars are calculated by re-sampling each network ten times with the same parameters and taking the standard deviation.

5.4.1.2. Optimiser and Loss-Function

The effect of the DANNs optimiser algorithms and loss-function were tested on the networks correlations as the next stage of the chapter project. The optimisation algorithm is what enables the neural network to learn by calculating the updates to the weights and biases of all nodes within the network in order to minimise loss. Adagrad was used repeatedly as in Chapter 3, a variant on the simple gradient descent optimiser, which varies the size of weight updates based on the sparsity of the data. Additionally, in this chapter, we tested the Adadelta optimiser (Zeiler, 2012), an extension of Adagrad, which does not decay the learning rate as aggressively as Adagrad. Adam optimiser, which takes into account past gradients (Kingma and Ba, 2014) in a similar way to a momentum optimiser was also used. Each of the three optimisers were tested at initial learning rates of 0.1, 0.01 and 0.001, and the results of the comparison can be seen in Figure 5.2a. There was relatively little difference in the correlations achieved except for Adadelta and also little difference between

using different initial learning rates. The highest correlation was from Adagrad at an initial learning rate of 0.001 (0.9088 ± 6.88E-5). The difference between using a learning rate of 0.01 and 0.001 was not quite statistically significant ($p = 0.0567$ in an unpaired two-tailed t-test, $N = 10$). However, the difference between using Adagrad with a learning rate of 0.001 and using Adam at 0.001 was statistically significant ($p < 0.001$, $N=10$).

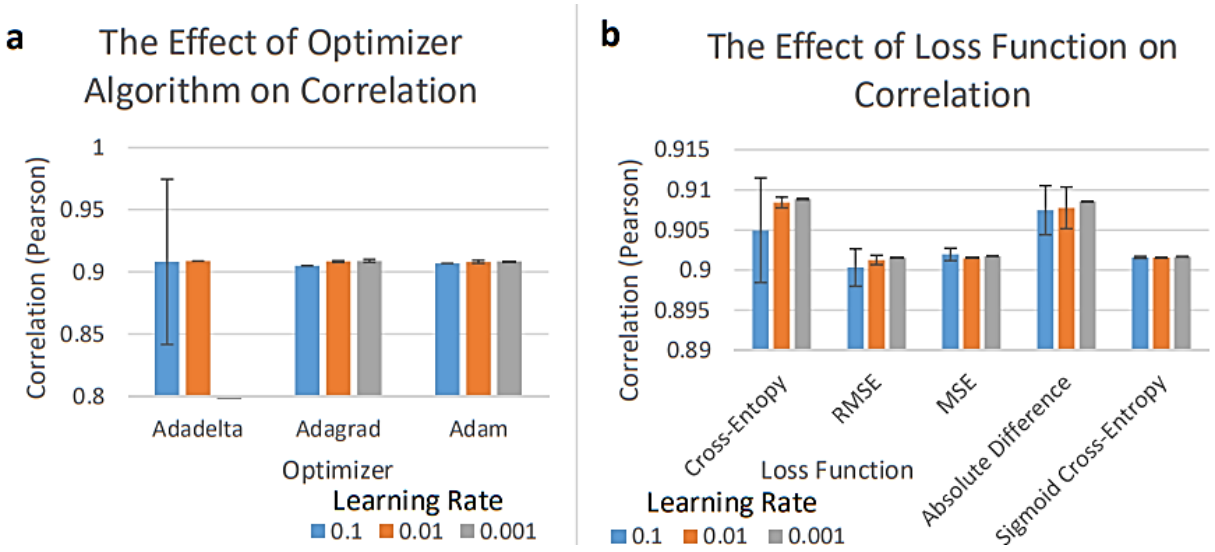

Figure 5.2. The effects of using different optimiser algorithms and loss functions on the performance of the networks. All error bars are derived from the standard deviation of 10 repetitions of each experiment. a) A comparison of the effect of the three different optimiser algorithms on correlation in DANNs C with different learning rates. The result for Adadelta with a learning rate of 0.001 is not shown because the correlation is significantly below that of the other optimisers (0.166 ± 0.066). b) A comparison of the effect of using different loss functions on the mean correlation in DANNs C. Each loss function was tested at three different learning rates.

The loss function of a neural network determines how far the neural network's predictions are from the observed values during training. The optimiser algorithms make updates to the network variables in order to minimise loss as defined by the loss function. In Chapter 3, we made use of a cross-entropy loss function throughout (used as a default). We also investigated the effect of using RMSE, MSE, absolute difference (abs. diff.) and SCE. Similar to the optimiser algorithm, each of these loss functions were tested at learning rates of 0.1, 0.01 and 0.001, the results are shown in Figure 5.2b. The highest correlation observed was when using the original cross-entropy loss function with a learning rate of 0.001 (0.9088 ± 6.88E-5), which was the same result observed in

the previous test. RMSE, MSE and SCE all produced results lower than 0.902, Abs. Diff. came close with a Pearson correlation of 0.9086 ± 1.24E-5. However, the difference between the two mean correlations was still significant ($p < 0.0001$).

It turned out that both the optimiser and loss-function used throughout the experiments in Chapter 3 proved superior to the other ones tested (at least with the placeholder parameters used in these experiments). Learning rate was varied in these experiments in order to gauge whether its effect on the different optimisers was significant or not. The lower learning rate produced a better result with almost all of the optimisers and loss-functions tested, the only exception being with MSE. All of the optimisers were designed to dynamically modify the learning rate (typically decaying the learning rate as training proceeds) which should make the initial learning rate less important, despite this, some difference is still seen between the optimisers at different learning rates. All the later experiments on DANNs C made use of the Adagrad optimiser with the cross-entropy loss function.

5.4.1.3. Learning Rate and Training Cycles

For our DANNs, several optimising parameters such as the learning rate and training cycles have been evaluated in this study. The learning rate acts as a multiplier to the scale of updates, which the optimiser applies to the network variables. By minimising the scale of learning rate, the risk of not updating the weights and biases enough will be high. Contrarily, a large learning rate risks over-compensating for small "quirks" of the training data. From the previous experiments, we have already determined that a smaller training rate slightly improves the correlation of DANNs C. Figure 5.3a shows the results of varying the training rate between 0.000001 and 1.0. There was a little observable change in the correlation between a rate of 0.1 and 0.00001, beyond these points there was a marked decrease in correlation and increase in standard deviation. The highest mean correlation was 0.9088 ± 6.54E-5 at a learning rate of 0.0001. Further testing between rates of 0.0005 and 0.00005 resulted in a mean correlation of 0.9089 ± 1.37E-4 at a rate of 0.00006 (Appendix 7).

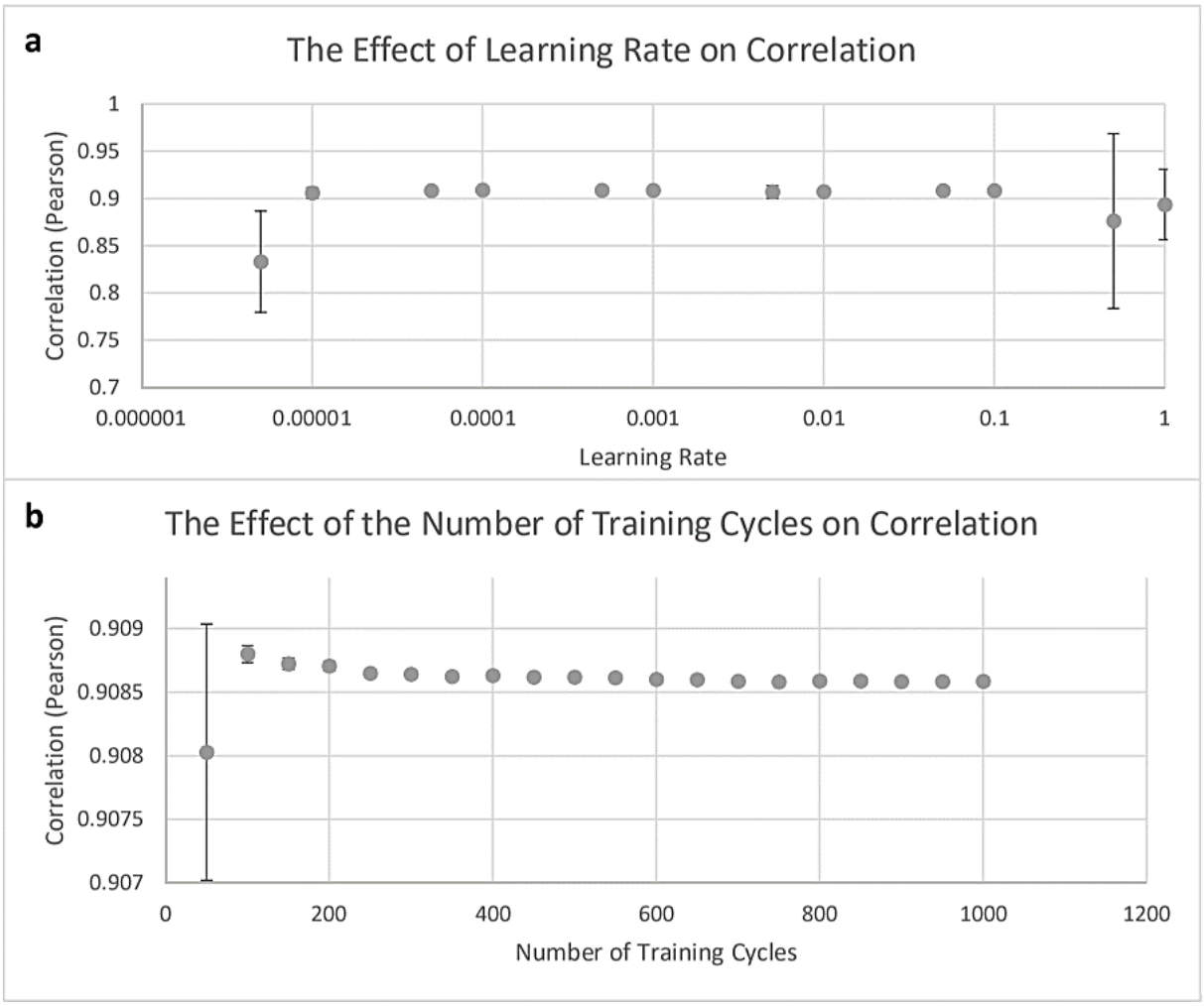

Figure 5.3. The effects of changing the number of training cycles and the learning rate on the results of the neural networks. Error bars represent the standard deviation of 10 repetitions of the network using the same parameters. a) Plot showing how the correlation of DANNs C varies as the learning rate is changed. The result for 0.000001 is excluded from the chart because it is significantly lower than the other results (0.316). b) Plot showing how the correlation of DANNs C varies as the number of training cycles change.

The other DANNs optimising parameter which was evaluated in our study was the number of training cycles. Results showed that there was a slight effect when varying the number of training cycles on the correlation between 50 and 1000 cycles (Figure 5.3b). The peak occurs at 100 training cycles and then gradually drops as the number of cycles increases (although the standard deviation decreases). The peak correlation was 0.9088 ± 6.65E-5.

The lack of variation between the different learning rates used, except at the high and low extremes, is probably once again due to the dynamic updates which the optimiser makes to the rate which

makes the initial rate far less important than if using a simple gradient descent optimiser. The gradual drop-off in correlation as the number of training cycles increase was somewhat more unexpected. Logically one would expect a network, which is provided with more opportunities to

analyse the training data and update its weights, to make better predictions. The fact that this is not the case is an indicator that the network is over-fitting the training data to an extent.

5.4.1.4. Regularisation

For DANNs regularisation, the hyperparameters, Dropout, and L2 regularisation were included in our evaluation. For the Dropout, the outputted data showed that implementing varying levels of this activation function on the Pearson correlation and during training has different effects (Figure 5.4a). The indicated level of Dropout was applied evenly to both hidden layers of the network throughout training with no dynamic updates. The results indicated that using any level of Dropout on the network produces lower correlations, the highest correlation was found when there was 0 chance of Dropout.

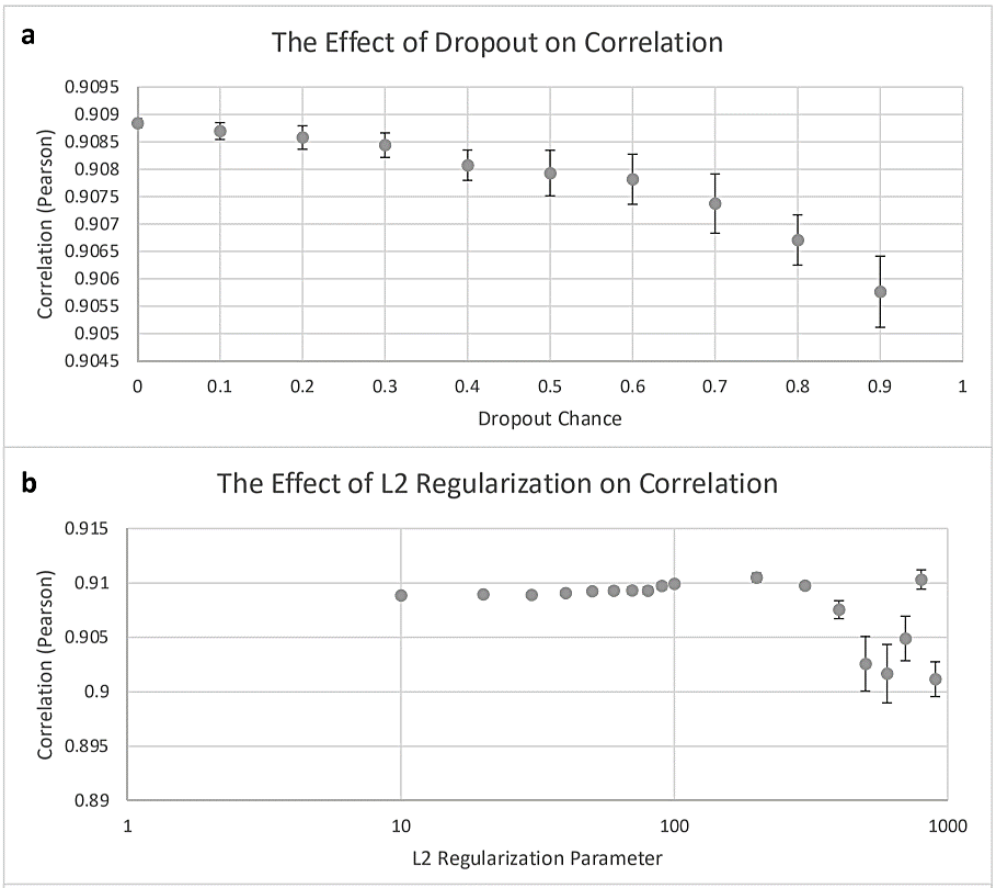

Figure 5.4. The effect of Dropout and L2 regularisation on the results of the networks. All error bars represent the standard deviation of 10 repetitions of the neural network using the same parameters. a) A scatter chart showing how different levels of Dropout probability effect the correlations generated by

DANNs C. b) Plot showing how different levels of L2 regularisation effect the correlations generated by DANNs C.

On the other hand, L2 regularisation was found to be able to improve the correlation as shown in Figure 5.4b. The highest correlation of an initial test (results not shown) of L2 hyperparameters between 100 and 0.0000001 found to be when L2 = 100. The results in Figure 5.4b shows the second test with L2 values between 10 and 1000. The highest correlation in this figure is 0.9105 ± 3.94E-4. Further testing found a more precise optimal L2 value of 170 where the correlation was 0.9107 ± 2.05E-4 (Appendix 8). L2 values less than that resulted in a gradual drop-off in the Pearson correlation while using greater L2 values resulted in a steeper drop-off and increase in result variance. Compared to the best correlation obtained from testing the number of training cycles, the increase was 0.209% with a p-value of > 0.0001.

The fact that Dropout did not improve the results was surprising, since it is a commonly used and effective method of regularisation (Srivastava et al., 2014). The preliminary tests indicated that Dropout had a beneficial effect on rank score when applied at low levels (results not shown), but evidently this does not carry over to correlation. Applying Dropout to DANNs with higher numbers of training cycles may have a more beneficial effect since overfitting is more likely. Another option was to introduce an individual Dropout probability for each node in the network which are updated during training, just like the weights and biases. Another possibility for future research is to use multiple methods of regularisation, this approach has been shown to improve results in other networks (Phaisangittisagul, 2016).

5.4.1.5. Architecture

The architecture of a neural network refers to the number of layers in the network and the number of neurons in each layer. The network in Chapter 3 used two layers for all tests, which is also our starting point. Figure 5.5a is a heatmap showing how the correlation of the network varies as the number of nodes in hidden layers 1 and 2 are changed between 10 and 100. The highest mean correlation was found when there were 20 nodes in both hidden layers (0.9128 ± 6.69E-4) while using 10 nodes in both layers was the least correlated combinations. Scores of 0.910 and higher form a “stripe” across the heatmap, favouring low numbers of nodes in one layer. Furthermore, a

total node number of around 150 is associated with lower correlations, forming a secondary stripe across the heatmap.

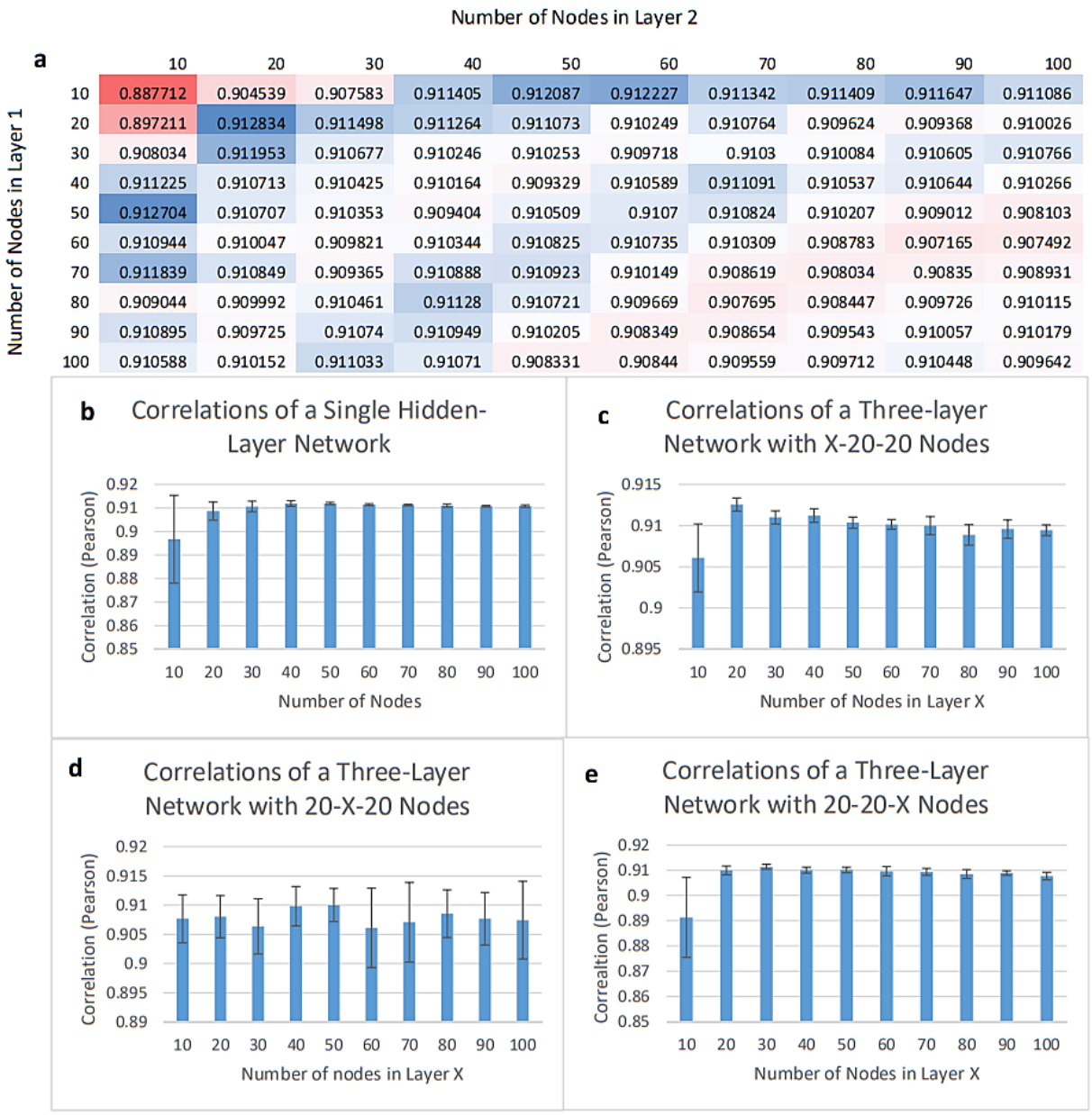

Figure 5.5. Results from testing different network architectures of DANNs C. All error bars represent the standard deviation of 10 repeats of the neural network using the same parameters. a) A heatmap showing the effect of different numbers of nodes in a 2-hidden layer network on the correlation. Results are colour-coded on a scale from low (red) to intermediate (white) to high (blue). Results are all the mean of 10 network runs with the same hyperparameters. b) A bar chart showing how the correlation varies with the number of nodes in a single-layer network. c, d and e) Bar charts showing how correlation varies in a three hidden-

layer network using 20 nodes in two layers and then a varying number in a third layer before, between and after the previous two.

Figure 5.5b shows how the number of nodes in a single-hidden layer network affects the correlation, it did not manage to improve on the score of the two-layered network. Figure 5.5c, 4.6d and 4.6e show the effect of using a three-layer network where one of the layers, designated layer X in the figures, has a variable number of neurons. The other two layers of the network both had 20 neurons, which was shown to be the optimal configuration of a 2-layer network. None of the three-layer networks make an improvement upon the optimal correlation of the 2-layer network.

Most preliminary tests up till this point used 200 nodes in each layer, a small number of nodes is the optimal configuration is probably because only two inputs are used, therefore additional network complexity is not required and only leads to overfitting. Using less than the optimal number of nodes means that the networks processing capacity is impaired while learning, leading to poor correlation. Using more nodes introduces needless complexity and leads to greater overfitting. Tests of the three-layer network were far from comprehensive, largely due to the time required to run a grid-search with three variables is an order of magnitude greater than a grid-search with two variables. However, given that introducing additional complexity to a two-layer network reduced the correlation, using a more complex three-layer network will probably never out-perform the optimal 2-layer network.

5.4.2. Deep Artificial Neural Networks for Ranking (DANNs R)

5.4.2.1. Inputs and Training Targets

The starting point for determination of network inputs and training targets is identical to what has been proceeded in Chapter 3, which identified 10 combinations of inputs that were partially better at ranking models than using all 10 inputs. The results of using these 10 combinations compared with using all 10 inputs while changing the network's training target are shown in Figure 5.6. All 10 combinations tested produced slightly higher rank scores than using all 10 inputs. Compared to the input/output testing on DANNs C, the identity of the output had much less effect on the rank score than on the correlation. The highest rank score was achieved by the combination: V9, V11, and V12 while using the GDT-HA output (32.409 ± 0.019).

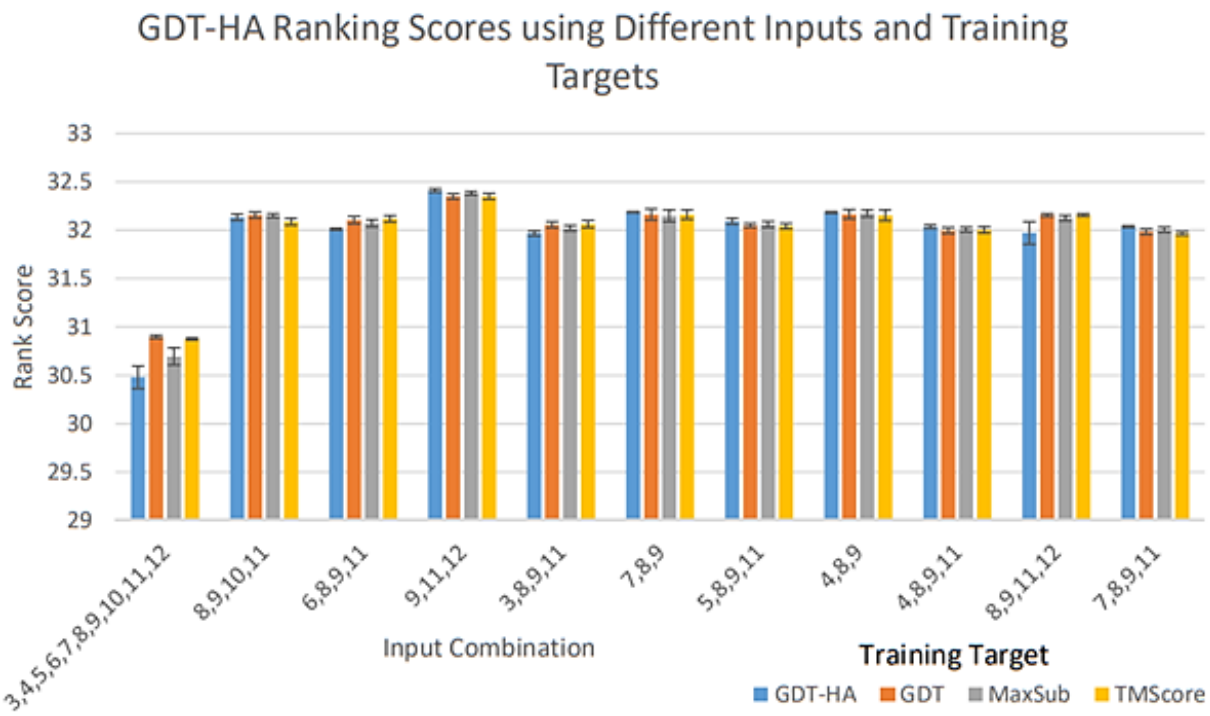

Figure 5.6. Histogram showing the rank scores produced by DANNs R for the top 10 input combinations. The results of using all 10 inputs is also shown for comparison.

Once again, using all 10 inputs was shown to be inferior than using a smaller number, certain inputs were seen far more frequently than others during this test, specifically V9 (occurs in all 10 combinations) and V11 (occurs in 8 of the top 10 combinations). This demonstrates that certain inputs were of considerably more value than others when the objective was model ranking.

5.4.2.2. Optimiser, Loss Function and Learning Rate

An initial test was done using the three optimiser functions similar to the one done on DANNs C with learning rates from 0.1 to 0.0001 (results not shown). In order to get a better idea of how the correlation and rank score both change with the learning rate, a further test was performed, looking at a more precise range of learning rates where the correlation began to fall below the target of 0.8. The results of this test are shown in Figure 5.7. For Adagrad and Adadelta, the optimal rank scores occur when the correlation dropped below the target value. However, when using the Adam optimiser algorithm, the peak rank score (32.407 ± 0.028) occurred while the correlation was still 0.8231 at a learning rate of 0.0006.

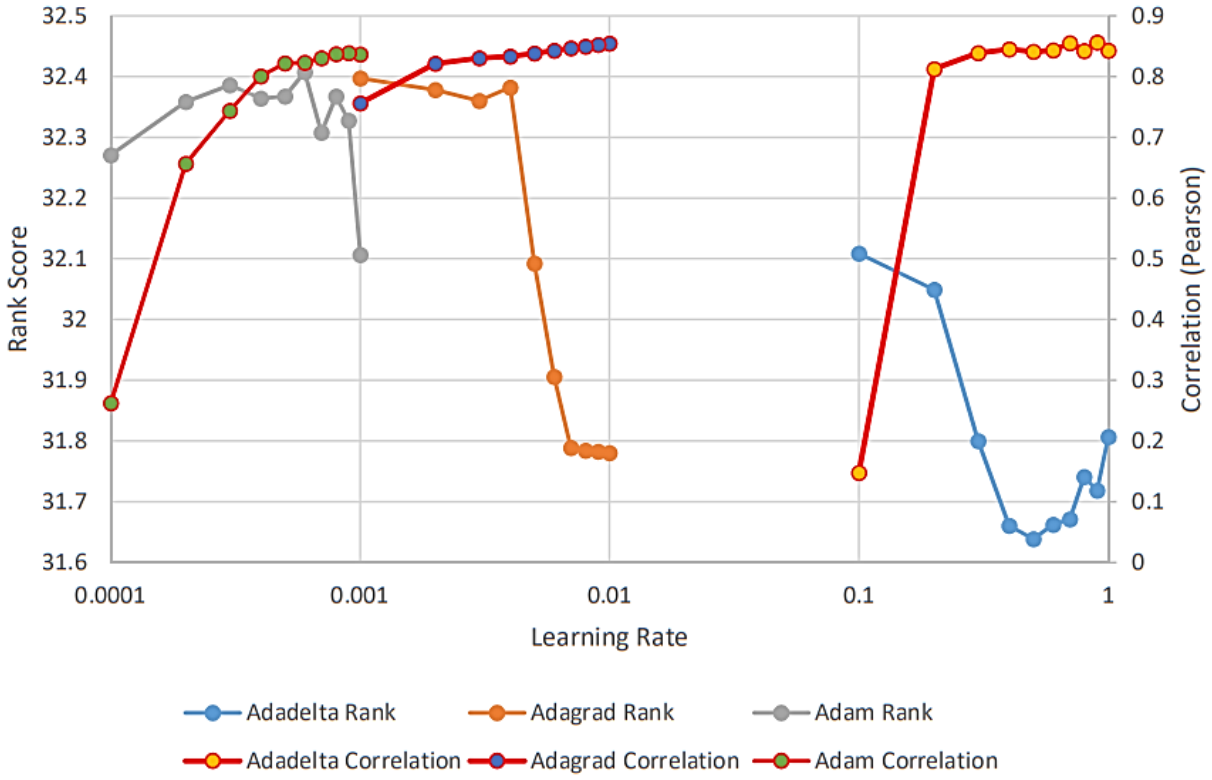

Figure 5.7. Plot comparing the rank scores for three optimiser algorithms over different learning rates in DANNs R. The corresponding Pearson correlations for each network are plotted on the secondary Y-axis. Error bars were excluded from this chart for clarity.

The five different loss functions were also tested over a more precise range of learning rates than in DANNs C. However, although the correlation scores still dropped off as the learning rate was 13 reduced, none of the top rank scores were associated with a correlation of less than 0.80. The top rank score from each loss function are shown in Figure 5.8. There was no significant difference between the top mean rank scores for Cross-entropy, Abs. Diff., MSE and RMSE. The top score was achieved by Cross-entropy (32.414 ± 0.029) at a learning rate of 0.006.

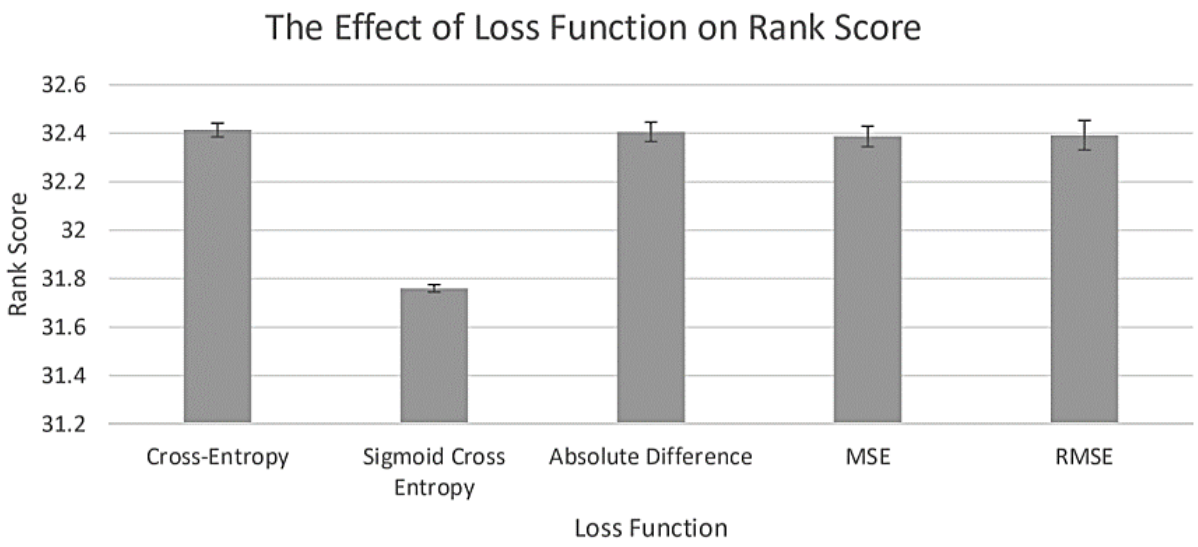

Figure 5.8. A comparison of the different loss functions on the rank scores. Only the top score loss function of each method is plotted.

From the results of these experiments, it seems that the cause of the issue with correlation is a combination between optimiser and learning rate. Although the correlation also dropped off while testing the loss-functions at different learning rates (data not shown), it did so in a pattern consistent with that observed while testing the Adam optimiser which was also used for the loss function experiment. This shows that the Adam optimiser and learning rates were the main factors in determining correlation drop-off during the loss-function experiments. Future experiments with this network will use the Adam optimiser in combination with cross-entropy as the loss function with a learning rate of 0.0006.

5.4.2.3. Training Cycles and Regularisation

Since learning rate was changed extensively while investigating the optimiser and loss function, further experimentation with this hyperparameter was deemed unnecessary. Figure 5.9 shows the effect of changing the number of training cycles between 50 and 1000 on the rank score. The optimal rank score was achieved at 100 training cycles (32.408 ± 0.081) and dropped off as the number of training cycles increased. Once the lowest rank score was achieved at 550 training cycles, it began to improve again incrementally as the number of training cycles increased further.

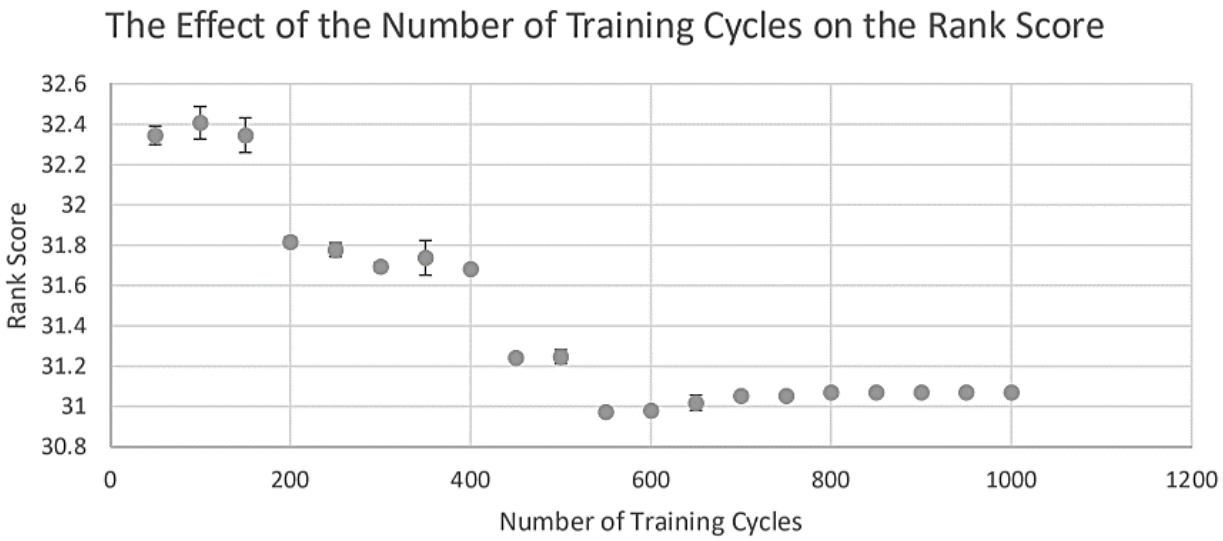

Figure 5.9. Plot showing how the rank score of DANNs R varies as the number of training cycles are changed.

In order to identify any potential effect of regularisation on overfitting within the network, different levels of Dropout between 0 and 0.9 were tested using the 100-cycle network, 500 cycle network and 1000 cycle network. The results of this test are shown in Figure 5.10a. There was no significant effect of using any level of Dropout on the 100-cycle network. In the 500-cycle network, the usage of any level of Dropout caused a slight reduction in the rank score. The 1000-cycle network proved to be the exception and showed a slight improvement in rank score when the Dropout probability reached 0.5 or higher. Overall, the application of Dropout did not manage to get any improvement upon the rank score.

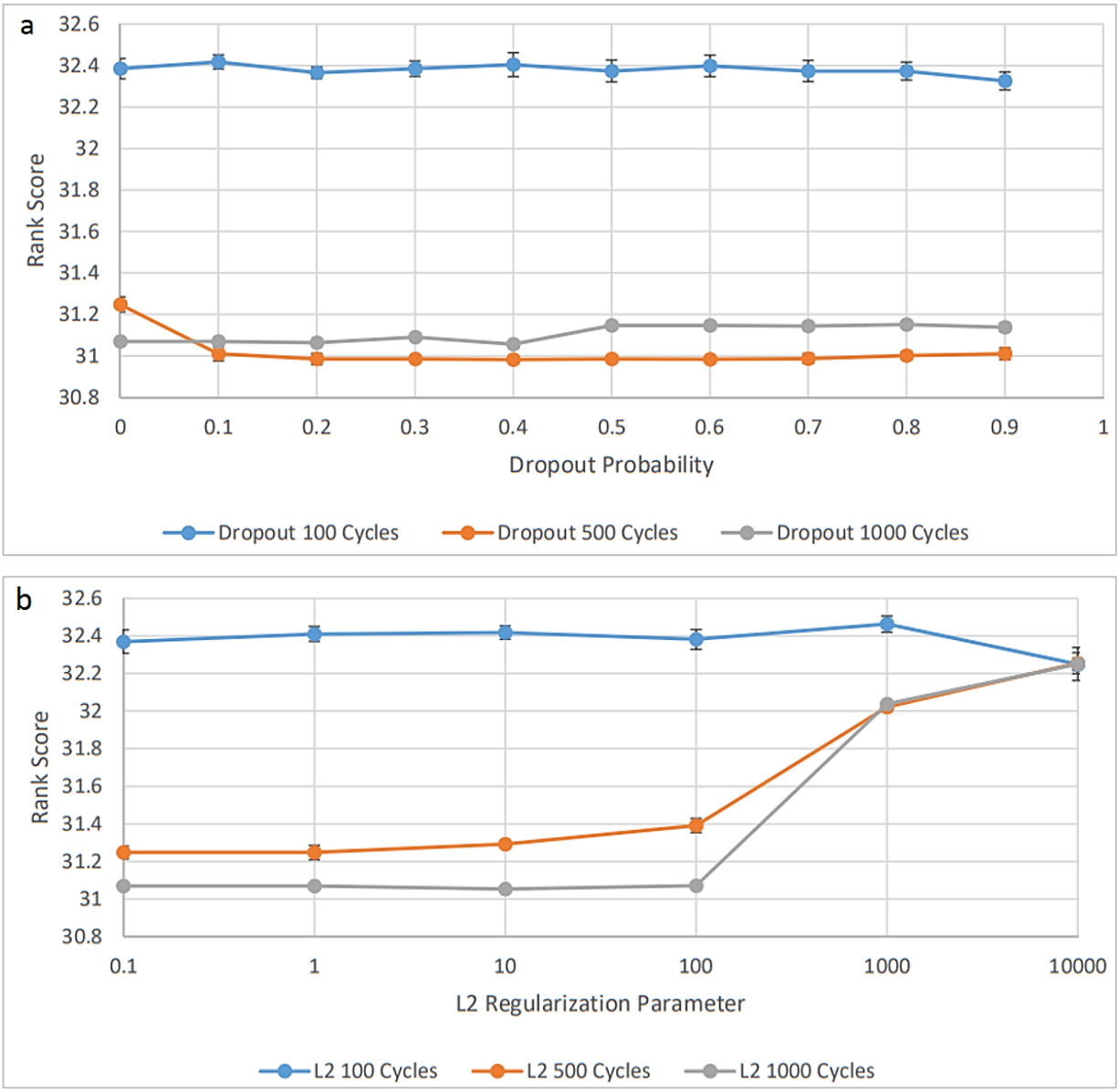

Figure 5.10. The effect of Dropout and L2 regularisation on the results of the networks. All error bars represent the standard deviation of 10 repetitions of the neural network using the same parameters. a) The effect of Dropout on the rank score of DANNs R. Results of Dropout are shown on a 100, 500 and 1000 training cycle network. b) The effect of L2 regularisation on the rank score of DANNs R. Results of L2 are shown for 100, 500 and 1000 training cycle networks.

Different levels of L2 regularisation were also tested on the same three network sizes. The results of the experiment are shown in Figure 5.10b. The effect of L2 regularisation was least apparent on the 100-cycle network although it still managed to improve upon the unregularised network's score with an L2 hyperparameter of 1000 (32.462 ± 0.043). Unfortunately, at this point, the correlation

also dropped below the 0.8 threshold (0.758). Although the 500-cycle and 1000 cycle networks did not manage to improve upon this rank score, applying an L2 hyperparameter of 1000 + resulted in a steep increase in the rank score although the correlation also dropped below the threshold in both cases. Further testing with levels of L2 regularisation between 100 and 1000 (Appendix 9) found that using an L2 regularisation hyperparameter of 200 produced the highest rank score without violating the correlation threshold (32.415 ± 0.011). Using an L2 hyperparameter of 500 or greater resulted in the correlation dropping below 0.80.

Dropout did cause an improvement; however, it only improved the performance of the 100-cycle network causing it to surpass the performance of the 500-cycle network. The effect on the 100-cycle network was negligible potentially because it was not over fitting the data. Introducing L2 regularisation slightly improved the performance of the 500 and 1000-cycle networks, but it also dropped the correlation below the threshold. The results of all three networks converged when the L2 hyperparameter reaches 10000. This is because the L2 hyperparameter acts as a balancing 14 factor within the loss function between the original loss function, and the size of the weights in the matrix. A very large value of L2 skewed the balance so that most of the final loss function result was composed of the weights matrix, and the original loss function became irrelevant. Overall a slight improvement in the rank score was obtained when an L2 hyperparameter of 200 was used.

5.4.2.4. Architecture

Figure 5.11a shows the results of a grid-search using two hidden layers with between 10 and 100 nodes in each layer. The highest rank scores were achieved with less than around 100 nodes total within the network. Using more than 100 total nodes resulted in the rank score degrading. The highest score was with 40 nodes in the first hidden layer and 30 nodes in the second (32.422 ± 0.039).

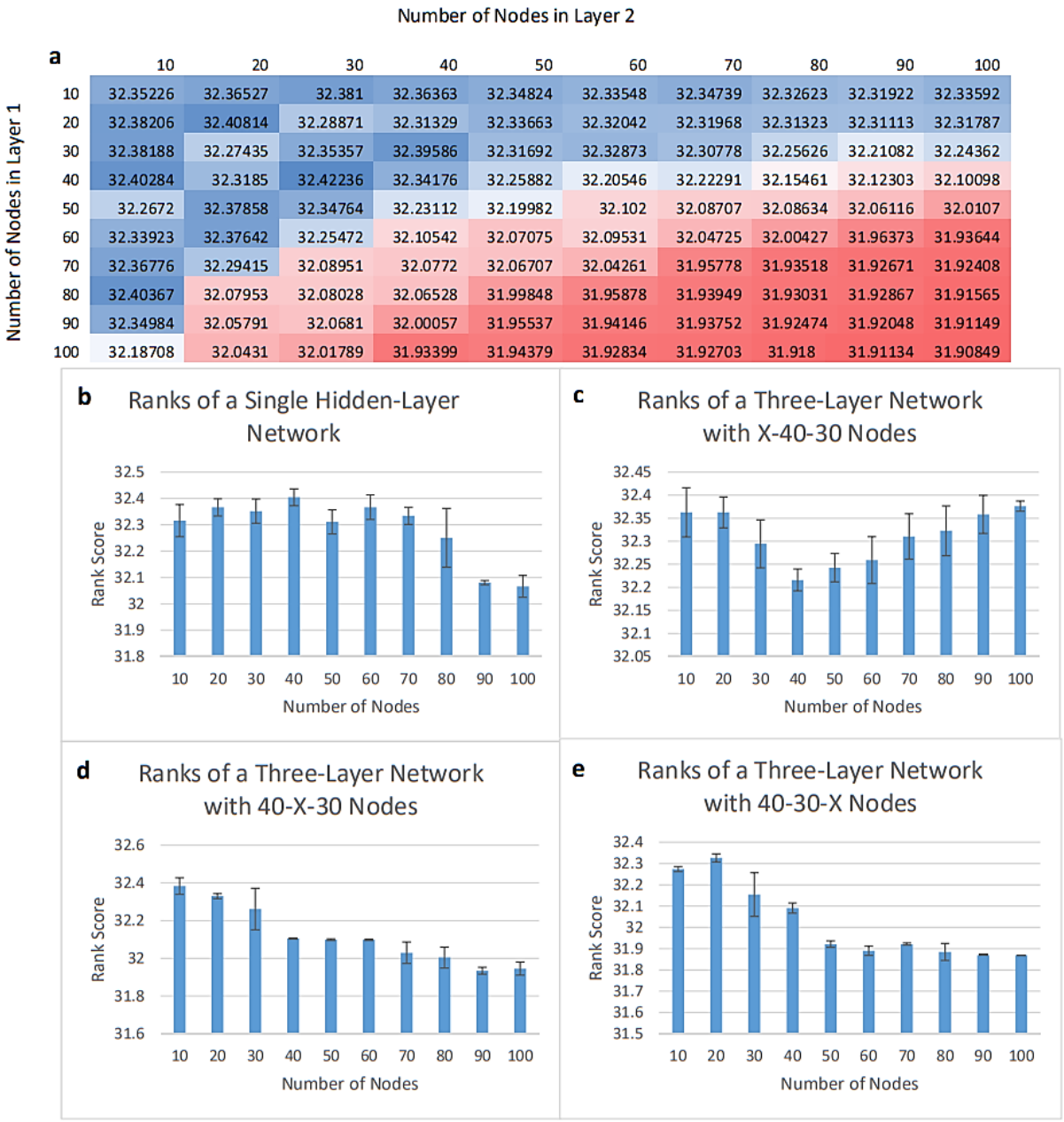

Figure 5.11. Results from testing different network architectures of DANNs R. All error bars represent the standard deviation of 10 repeats of the neural network using the same parameters. a) A heatmap showing the effect of different numbers of nodes in a 2-hidden layer network on the rank score. Results are colour coded on a scale from red (low) to intermediate (white) to high (blue). Results are the mean of 10 network runs with the same hyperparameters. b) A bar chart showing how the rank scores varies with the number of nodes in a single-layer network. c, d and e) Bar charts showing how the rank scores vary in a three hidden-layer network using 30 nodes in two layers and then adding a varying number in a third layer before, between and after the previous two.

Using only a single hidden layer in the network did not improve the rank score (Figure 5.11b). A third layer was added to the network by using the optimal combination for a two hidden-layer network from Figure 5.11a and adding a third layer between 10 and 100 nodes. The results from these DANNs are shown in Figures 5.11c, d and e. Using three layers did not improve upon the optimal results from the two layered networks. The optimal number of nodes in DANNs R was higher than in DANNs C, this was not surprising since the amount of input data was greater.

5.4.3. Significance of Results

Both DANNs were run a further 100 times with the optimal parameters in order to compare the performance to existing EMA methods. The final hyperparameters for both networks are shown in Table 5.3.

Hyperparameter	DANNs C	DANNs R
Input Combination	V4, V10	V9, V11, V12
Training Target	MaxSub	GDT-HA
Optimiser	Adagrad	Adagrad
Loss Function	Cross-Entropy	Cross-Entropy
Learning Rate	0.00006	0.0006
Number of Training Cycles	100	100
Dropout	0	0
L2 Regularisation	170	200
Number of Layers	2	2
Number of Nodes per Layer	20/20	40/30

Table 5.3. A summary of the final hyperparameters for DANNs C and DANNs R.

Figure 5.12a shows the network predictions plotted against the observed GDT-HA scores. Figure 5.12b shows the mean of V4 and V10 plotted against the observed GDT-HA scores. DANNs C produced a top correlation of 0.913971 using the V4 and V10 inputs. Taking the average of the two inputs gave a correlation of 0.910036. The network performed marginally better (0.43%) than simply taking the average of the network inputs. The results from the network described a clearer relationship with fewer outliers, however the networks results mostly lied between 0.3 and 0.6, leading to a more compressed scatter. This compressed scale of network output may limit its performance in other CASP benchmarks, e.g. absolute differences in predicted versus observed score.

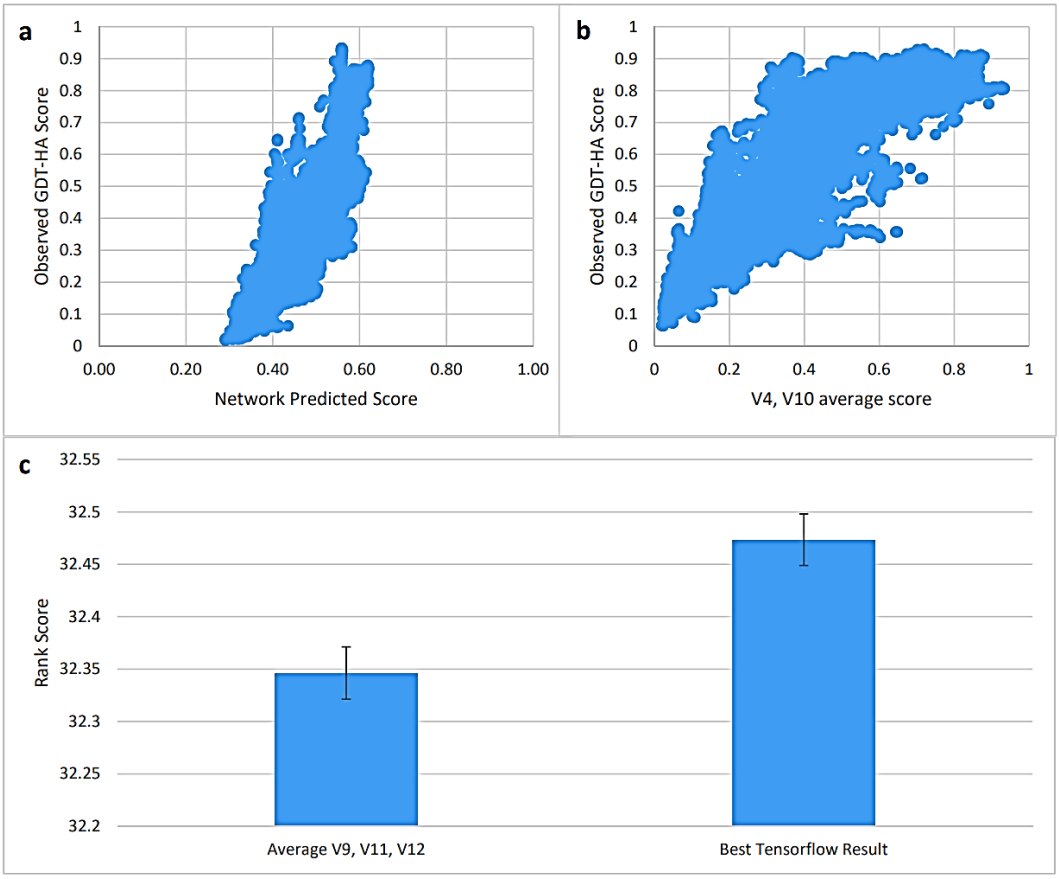

Figure 5.12. The final results of DANNs C and DANNs R compared to using an average of the input scores. a) Plot with the predicted scores of DANNs C plotted against the observed GDT-HA scores. b) Plot with the average of the V4 and V10 inputs plotted against the observed GDT-HA scores. c) A bar chart comparing the results of DANNs R to the average of Inputs V9, V11 and V12 for ranking models. Error bars are standard error.

DANNs R produced a top rank score of 32.473 using the V9, V11 and V12 combination of inputs. Taking the average predictions of servers V9, V11 and V12 using the same data, gave a rank score of 32.346, meaning that DANNs R achieved an improvement of 0.39% over averaging the inputs. Comparing the top-ranked GDT-HA results for each model reveals that out of the 84 different modelling targets, only 17 had a different model picked by the two methods. Figure 5.12c shows both rank scores with the standard error included as error bars. Testing for significance by Wilcoxon's signed rank test (2-tailed) gives a p-value of 0.246, indicating that the improvement is not significant.

Grid search combined with hand-tuning has proved to be an effective, time-saving method of hyperparameter determination. However, the time investment in this project was still considerable.

Furthermore, only a small number of all possible combinations of hyperparameters were tested, using a random grid search would allow a greater number of combinations to be explored. Bayesian optimisation on the other hand uses the results from previous attempts to inform as to potential parameters for the next attempt. Using this method should save a considerable amount of time.

While both the model ranking and correlation of the results have been shown to be improved by employing a neural network, what does this mean in terms of real-world applications? A network which produces good correlations makes good estimations as to the quality of the models. This is useful in the case where a quality estimate on one or a few models are required. A network which produces good rank scores is capable of picking out the top model from a group of alternate models. This is most useful when you must discriminate between multiple models of the same protein target.

5.5. Conclusion

Using Deep Artificial Neural Networks, we have made marginal improvements in correlation and ranking ability. DANNs show promise in improving the area of protein structure quality assessment, but the amount of time and effort required to find the optimal hyperparameters is prohibitive. The use of random grid searches or Bayesian optimisation could improve this in the future. Another goal is to create a DANNs which is capable of getting high correlations and effectively ranking the models without the necessity for two separate networks. In theory a network which produces extremely high-quality predictions should produce good correlation and rank scores.

With the large amount of protein structural data available, and the CASP experiment acting as a catalyst, more and more software is being written to predict protein structure. The need for useful EMA programs is increasing as more protein models are generated. DANNs have the potential to learn how to combine scores from QA programs in ways that maximise their usefulness. The ability of DANNs to learn, means that even very situational QA scores have the potential to be used as inputs for any model, given that the DANNs can learn the situations where each individual score is of most use.

Chapter 6

Independent Benchmarking for the Upgraded ModFOLD7 with the Top EMA Methods in CASP13

Work presented in this chapter has been published in the following papers:

Maghrabi, A.H.A., McGuffin, L.J., 2019. Estimating the quality of 3D protein models using the ModFOLD7 server. Submitted to Springer.

Cheng, J., Choe, M.-H., Elofsson, A., Han, K.-S., Hou, J., **Maghrabi, A.H.A.**, McGuffin, L.J., Menéndez-Hurtado, D., Olechnovič, K., Schwede, T., Studer, G., Uziela, K., Venclovas, Č., Wallner, B., 2019. Estimation of model accuracy in CASP13. Proteins: Structure, Function, and Bioinformatics. https://doi.org/10.1002/prot.25767

6.1. Background

Since researchers from different fields of biological sciences started relying on the three-dimension structural models of proteins, prediction programs have been improving rapidly. One of the major components of structure prediction pipelines is the evaluation or assessment of the predicted model accuracy. It is possible to generate many hundreds of alternative 3D models for any give protein target using many different algorithms. Often the best modelling method is not always the most accurate for a given target, so it is problematic to choose rank and select the models that are most likely to be the closest to the native structure. Furthermore, local regions of models may differ in quality and so it may help a biologist to know whether their specific regions of interest are accurately modelled e.g. predicted interface/interacting residues. Such problems have been recognised by the field of structural bioinformatics and many developers have focused their attention towards improving methods for model QA that support their prediction pipelines. Such tools and servers are also currently referred to as the Estimates of Model Accuracy (EMA) methods.

The EMA (a.k.a. MQA) methods and servers were included for evaluation as a category in two major worldwide organisations that are specialised in the protein structure prediction field. The first organisation conducts independent blind testing with the Critical Assessment of Techniques for Protein Structure Prediction (CASP) (Moult et al., 2014) experiments, which are held every other year. The second organisation is the continuously automatic model evaluation project called CAMEO (Haas et al., 2018). Both organisations have highlighted the importance of the EMA development for the improvement of protein structure prediction and have helped to encourage progress in the field.

Modern methods of EMA can be classified into three broad categories. (1) The pure-single model methods, which can score the data from the information of an individual model - they are featured by their rapid processing and their strong performance at model ranking and selection, but they often produce less consistent global scores. (2) The clustering/consensus approaches, which use multiple alternative models build for the same protein target - these types of methods have the opposite features of the single-model methods, they have been far more accurate but are more computationally intensive and do not work when very few similar models are available. (3) The quasi-single model methods, which can score an individual model against a pool of reference alternative models that are generated from the same target sequence. Quasi-single model methods

attempt to provide comparable accuracy to clustering methods, while addressing real-life needs of researchers with few/single models.

ModFOLD (McGuffin, 2007) is our EMA protocol and various successive versions have been competing with the top leading model quality assessment programs throughout the past 10 years. ModFOLD was built in the beginning as two separate methods. The original single-model method was called by its own original name, ModFOLD. Additionally, we developed a clustering-based method, called ModFOLDclust (McGuffin and Roche, 2010). Over the years, both methods have been merged with the adoption of a number of other methods to develop a new ModFOLD program which was a pioneer of the quasi-single model approach.

The quasi-single model approach was firstly implemented with the 3rd version of ModFOLD (Roche et al., 2014). By using this approach, ModFOLD3 was able to generate reference sets of models from the target sequence, using the IntFOLD-TS (McGuffin and Roche, 2011) method which were used for comparison with the submitted model using ModFOLDclust2 (McGuffin and Roche, 2010). ModFOLD has since undergone a number of updates through versions 4 (McGuffin et al., 2013), 5 (McGuffin et al., 2015) and 6 (Maghrabi and McGuffin, 2017), which have maintained the use of a quasi-single model approach. Each successive version has been ranked among the top performing EMA methods of the recent CASP experiments. The implementation of quasi-single method has helped our ModFOLD pipeline keep its competitiveness using the predictive power offered by clustering-based methods, as well as being capable of making predictions for a single model at a time. While we have made significant progress in performance over the years with our ModFOLD methods, there is still room for improvement in many aspects of EMA.

Here we describe significant major updates to the ModFOLD server. The server has been popular with modellers around the world, having completed hundreds of thousands of EMA jobs for thousands of unique users over the past decade. In 2018, the ModFOLD7 server variant methods participated in the latest world-wide Critical Assessment of Techniques for Protein Structure Prediction competition (CASP13). The goal of this competition was to help advance the methods which identify protein structure from sequence by testing them objectively via the process of blind prediction. The competition includes many subcategories, one of them is the EMA where our ModFOLD7 methods are independently evaluated. The CASP assessors provide sequences of proteins whose structures have never been observed before. Participants uses their prediction

servers in order to generate the 3D models of the target structures. Once server models have been generated for a given target, they are then used for the EMA category; participants use their model quality assessment methods in order to estimate the accuracy of the predicted models for each target.

6.2. Objectives

In this chapter, we describe our latest upgrade of ModFOLD. ModFOLD7 is our leading resource for EMA, which has been upgraded by integrating a number of the pioneering pure-single and quasi-single model approaches. Such an integration has given our latest version the strengths to accurately score and rank predicted models, with higher consistency compared to older EMA methods. Additionally, the server provides three options for producing global score estimates, depending on the requirements of the user: (i) ModFOLD7_rank, which is optimised for ranking/selection, (ii) ModFOLD7_cor, which is optimised for correlations of predicted and observed scores and (iii) ModFOLD7 global for balanced performance. ModFOLD7 has been ranked among the top few EMA methods according to independent blind testing by the CASP13 assessors. Another evaluation resource for ModFOLD7 is the CAMEO project, where the method is continuously automatically evaluated, showing a significant improvement compared to our previous versions. The ModFOLD7 server is freely available at: http://www.reading.ac.uk/bioinf/ModFOLD/.

We will also compare the improvement in our methods with all improvements gained from CASP12 to CASP13 in the field of EMA as seen from the progress of the most successful methods in CASP13. We show small but clear progress, i.e. several methods perform better than the best methods from CASP12. Some progress is driven by applying deep learning and residue-residue contacts to model quality prediction. We show that there has been measurable progress since CASP12. Although direct comparisons are difficult, as the targets and underlying methods that generate the targets change between CASP seasons, it is clear that progress has been made as novel methods outperform the best methods in CASP12. Further, we show that the best EMA methods marginally outperform the best servers when it comes to selecting one model per target.

6.3. Materials and Methods

The latest version of our server, ModFOLD7, uses a new quality assessment technique which combines the strengths of multiple pure-single and quasi-single model methods for the improvement of prediction accuracy. The server comprises a single model approach which combines 10 scoring methods. Six of the methods are pure-single model inputs methods, these include: 1- Contact Distance Agreement (CDA) which uses MetaPSICOV (Jones et al., 2015) to relate to the agreement between the predicted residue contacts and the contacts in model; 2- Secondary Structure Agreement (SSA) which uses PSIPRED (Buchan et al., 2013) to relate to the agreement between the predicted secondary structure of each residue and the secondary structure state of the residue in model according to Dictionary of Secondary Structures of Proteins (DSSP); 3- ProQ2 (Uziela and Wallner, 2016); 4- ProQ2D (Uziela et al., 2017); 5- ProQ3D (Uziela et al., 2017); and 6- VoroMQA (Olechnovič and Venclovas, 2017). The remaining four methods are quasi-single model input methods, these are: 1- ModFOLDclust_single (MFcs) which uses input model against the 130 IntFOLD5 reference models; 2- Disorder “B-factor” Agreement (DBA) which compares DISOPRED (Jones and Cozzetto, 2015) scores against the MFcs score; 3- ModFOLDclustQ_single (MFcQs) (McGuffin and Roche, 2010) which uses input model against the IntFOLD5 reference models; and 4- ResQ (Yang et al., 2016) which estimates the residue-specific quality and B-factor, and it compares the input model against LOMETS (Wu and Zhang, 2007) models. The combination of the component per-residue/local quality scores from each of the 10 methods are processed using Neural Networks resulting in a final consensus of per-residue

quality scores for each model. A flowchart of the data and processes used in the ModFOLD7 server is shown in Figure 6.1.

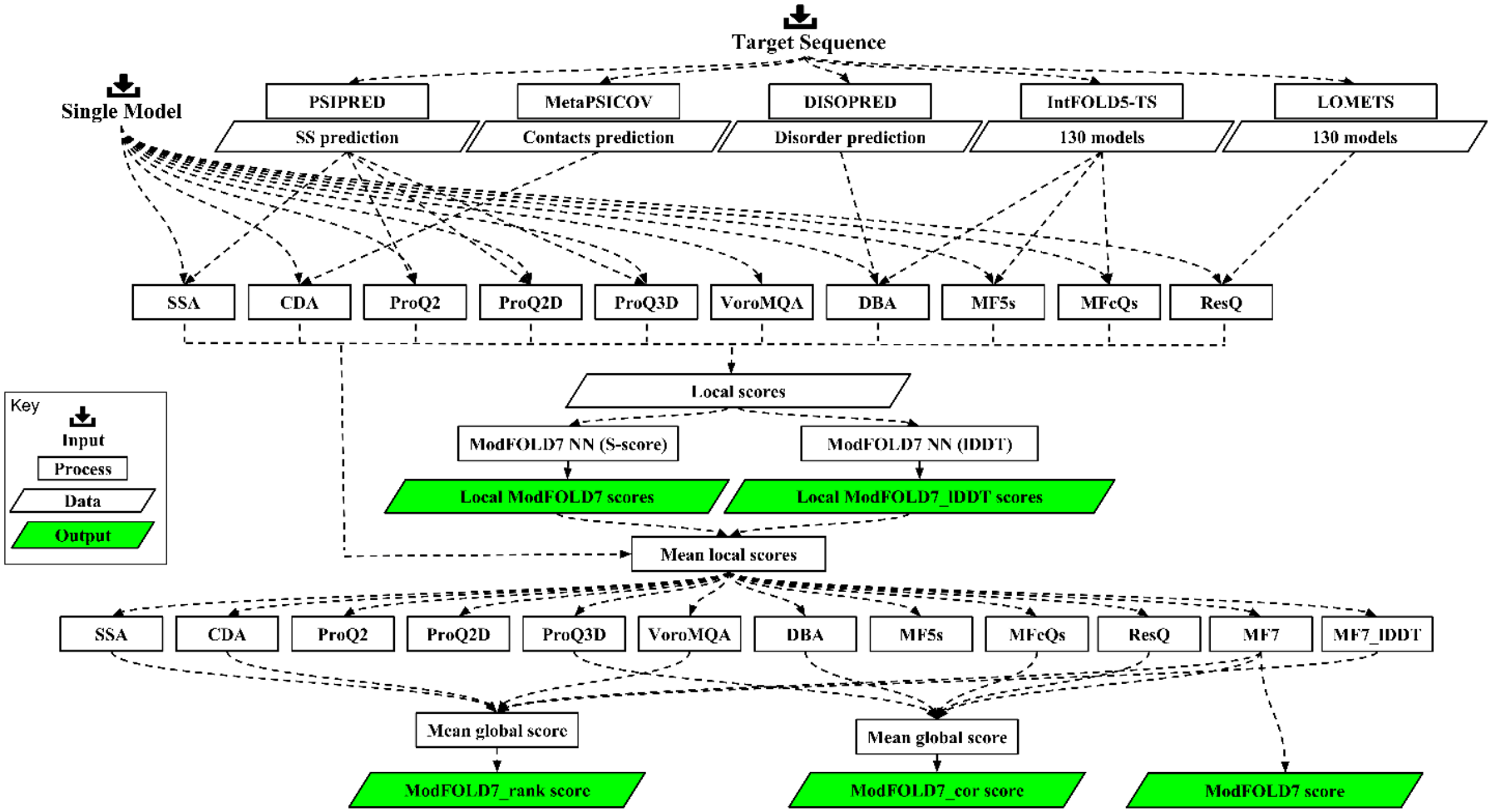

Figure 6.1. Flow of data illustrating the local and global estimates of model accuracy in ModFOLD7. The method pipeline starts with 2 inputs, the target sequence and a single model. The target sequence is evaluated with 5 pre-processing methods. The resulting data from the pre-processing methods with the input single model then are evaluated with 10 scoring methods resulting in local score input data. Next, the local scores are processed using 2 neural networks trained to two target functions, the S-score and the lDDT score, resulting in the final local score outputs. Lastly, the mean local scores from each method are used to form 12 global scores, which are then optimally combined in the different ways indicated to form the 3 variants of ModFOLD7. Figure from Cheng et al., (2019).

6.3.1. The ModFOLD7 component per-residue/local quality scoring methods

The ModFOLD7 NNs were trained using two separate target functions for each residue in a model: the residue contacts based lDDT score and the superposition-based S-score which has been used in previous versions of ModFOLD. The RSNNS package for R was used to construct the NNs, which were trained using data derived from the evaluation of CASP11 & 12 server models versus native structures. The per-residue similarity scores were calculated using a simple multilayer perceptron (MLP). For the method trained using the lDDT score (ModFOLD7_res_lddt), the MLP input consisted of a sliding window (size=5) of per-residue scores from all 10 of the methods described above, and the output was a single quality score for each residue in the model (50 inputs, 25 hidden, 1 output). For the method trained using the S-score (ModFOLD7_res), this time only 7 of the 10 methods were used as inputs - all apart from the ProQ2, CDA and SSA scores - with a sliding window (size=5), therefore 35 inputs, 18 hidden, 1 output. For both of the per-residue scoring methods, the similarity scores, s, for each residue were converted back to distances, d, with $d = 3.5\sqrt{((1/s) - 1)}$.

6.3.2. The ModFOLD7 global scoring methods

Global scores were calculated by taking the mean per-residue scores (the sum of the per-residue similarity scores divided by sequence lengths) for each of the 10 individual component methods, described above, plus the NN output from ModFOLD7_res and ModFOLD7_res_lddt. Furthermore, 3 additional quasi-single global model quality scores were generated for each model based on the original ModFOLDclust, ModFOLDclustQ and ModFOLDclust2 global scoring methods (in a similar vein to the ModFOLD4_single and ModFOLD5_single global scores, tested in CASP10 and CASP11 respectively). Thus, we ended up with 15 alternative global QA scores, which could be combined in various ways in order to optimize for the different facets of the quality estimation problem. For the CASP13 experiment, we registered three ModFOLD7 global scoring variants: 1. The ModFOLD7 global score, which used the mean per-residue NN output score from ModFOLD7_res, this score considered alone was found to have a good balance of performance both for correlations of predicted versus observed scores and rankings of the top models. 2. The ModFOLD7_cor global score variant ((*MFcQs + DBA + ProQ3D + ResQ + ModFOLD7_res)/5*) was found to be an optimal combination for producing good correlations with the observed scores, i.e. the predicted global quality scores produced should produce closer to linear correlations with

the observed global quality scores. 3. The ModFOLD7_rank global score variant ((*CDA* + *SSA* + *VoroMQA* + *ModFOLD7_res* + *ModFOLD7res_lDDT)/5*) was found to be an optimal combination for ranking, i.e. the top ranked models (top 1) should be closer to the highest accuracy, but the relationship between predicted and observed scores may not be linear. The local scores of the ModFOLD7 and ModFOLD_rank variants used the output from the ModFOLD7_res NN, whereas the ModFOLD_cor variant used the local scores from the ModFOLD7_res_lddt NN.

6.3.3. Server inputs

Like the previous versions, the ModFOLD server version 7.0 requires the amino acid sequence (Figure 6.2) of your target protein and either a single 3D model file in PDB format or a tarball containing a directory of multiple separate files in PDB format. To produce a tarball file for your own 3D models, for Linux/OSX/other Unix users: (a) Tar up the directory containing your PDB files e.g. type the following at the command line: tar cvf my_models.tar my_models/ (b) Gzip the tar file e.g. gzip my_models.tar (c) Upload the gzipped tar file (e.g. my_models.tar.gz) to the ModFOLD server, for Windows users: (a) Download a file archiver application such as 7-zip (b) Select the directory (folder) of model files to add to the .tar file, click "Add", select the "tar" option as the "Archive format:" and save the file as something memorable e.g. my_models.tar (c) Select the tar file, click "Add" and then select the "GZip" option as the "Archive format:" - the file should then be saved as my_models.tar.gz (d) Upload the the gzipped tar file (e.g. my_models.tar.gz) to the ModFOLD server. Providing the email address will give the permission to send a link with the graphical results and machine-readable results directly after the predictions are completed. However, if the user does not provide the email address then s/he must bookmark the results page in order to view and refer to it when it is available. In the text box labelled “Input sequence of protein target”, users should carefully paste in the full amino acid for the interested target protein in single letter format.

It is important that the user provides the full sequence that corresponds to the sequence of residue coordinates in the model file. If the model does not contain numbering which corresponds directly to the order of residues in the sequence file, then the server will attempt to renumber the residues in the model files accordingly. However, submitting a model file with residues that are not contained in the provided sequence will not complete the prediction for that model. Users must ensure that each PDB file contains the coordinates for one model only. The coordinates for multiple

models should always be uploaded as a tarred and gzipped directory of separate files. Assigning a short memorable name to user's prediction jobs is useful for identifying and distinguishing them, because ModFOLD will not necessarily return the results in the order the user submitted them.

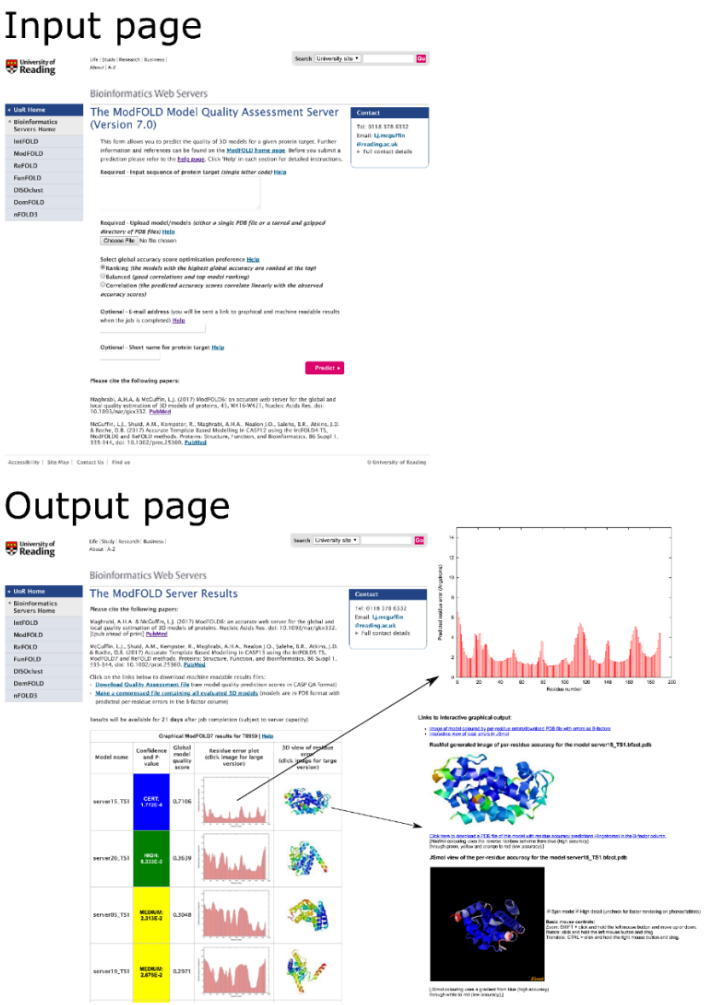

Figure 6.2. ModFOLD7 server inputs and outputs pages. *Inputs page:* containing a text box to paste the amino acid sequence of protein target in single letter code, a push button to upload model/models (either a single PDB file or a tarred and gzipped directory of PDB files) of the protein target, three options to select the global accuracy score optimisation preference, and two optional text boxes to input the user E-mail address and to give a short name for protein target. *Outputs page:* showing the result page for models submitted to CASP13 generated for target T0959. The main output page is shown with summary tables of the results for each model. Results can also be visualised in more detail by clicking on the thumbnail images in the main table.

6.3.4. Server outputs

The outputted results are provided in a clean and simple user interface so that it can be interpreted easily by non-experts at a glance. Once the prediction process is complete a results page is generated containing a single table summarising the quality assessment scores for each submitted model. Each assessed model is represented in the table graphically, with thumbnail images of the local error plots and annotated 3D models. Images in the table are clickable for detailed 3D visualisation using the JSmol/HTML5 framework. Conveniently, interactive 3D results can also be viewed on mobile devices without any plugin requirement. The results table shows a global score for each model, a P-value indicating the likelihood that the model is incorrectly folded and a plot of the local errors in the model in Ångströms. Users can also download the models annotated with the ModFOLD7 predicted local quality scores, which have been inserted into the B-factor column of the ATOM records for each submitted model. The raw machine-readable data files for each set of predictions, which comply with the CASP data standards, are also provided for developers and more advanced users. An overview of the ModFOLD7 interface is shown in Figure 6.2.

The results table is ranked according to decreasing global model quality score. The global model quality scores range between 0 and 1. In general, scores less than 0.2 indicate there may be incorrectly modelled domains and scores greater than 0.4 generally indicate more complete and confident models, which are highly similar to the native structure. If the global model quality scores are low, then the per-residue scores can give you an idea of specific domains or regions in your protein that might be correctly modelled. From the global scores, the p-value which represents the probability that each model is incorrect can be calculated. In other words, for a given predicted model quality score, the p-value is the proportion of models with that score that do not share any similarity with the native structure (TM-score < 0.2). Each model is also assigned a colour coded confidence level depending on the p-value: $p < 0.001$ = blue = CERT = Less than a 1/1000 chance that the model is incorrect, $p < 0.01$ = green = HIGH = Less than a 1/100 chance that the model is incorrect, $p < 0.05$ = yellow = MEDIUM = Less than a 1/20 chance that the model is incorrect, $p < 0.1$ = orange = LOW = Less than a 1/10 chance that the model is incorrect, $p > 0.1$ = red = POOR = Likely to be a poor model with little or no similarity to the native structure.

The per-residue scores indicate the predicted distance (in Angstroms) between the CA atom of the residue in the model and the CA atom of the equivalent residue in the native structure. Thumbnail images of plots depicting the per-residue error versus residue number are included in each row in

the results table. Each of the thumbnails links to a page that displays a larger view of the plot and contains a further link to download a PostScript version. Each row in the table also displays a thumbnail of the 3D cartoon view of the model which is colour coded with the residue error according to the RasMol temperature colouring scheme. Each small image also links to a page that shows a larger image of the 3D view and contains a link to download a PDB file of the model with residue accuracy predictions (Angstroms) in the B-factor column. The model is also loaded into JSmol for convenient interactive viewing of per-residue errors within the browser. The time taken for a prediction will depend on the length of sequence, the number of models submitted and the load on the server. For a new run on single model the user should typically receive his/her results back within 24 hours, once the job is running. Large batches of models (several hundred) for a single target may take several days to process. If the user has already submitted a model for the same target sequence within the same week, then the reference model library for that sequence will already be available to the server (the results will be cached) and so s/he will receive the results back much more quickly (within a few hours).

6.3.5 Benchmarking ModFOLD7 within the top ranked EMA methods in CASP13

ModFOLD7 has been evaluated with a number of EMA predictors from six top-performing labs in CASP13. Each EMA method has its own input and output approach which features it from the other EMAs. In terms of input, EMA methods can be classified as single-model methods (Wallner and Elofsson, 2003) (Olechnovič and Venclovas, 2017) (Ray et al., 2012) (Sippl, 1990) and multi-model (or consensus) methods (Ginalski et al., 2003) (Lundström et al., 2001). The former takes a single structural model as input to predict its accuracy, while the latter uses multiple structural models of a protein as input to estimate their accuracy, often leveraging the similarity between the models. In terms of output, EMA methods can be categorised as global accuracy assessment methods (Zhang and Skolnick, 2005) (Zemla, 2006) and local accuracy assessment methods (Wallner and Elofsson, 2006), see Table 6.1. The global methods predict a single global score (eg, GDT-TS score) measuring the global accuracy of a whole model, whereas the local methods estimate the local accuracy (e.g., the distance deviation from the native position) for each residue in a model. The vast majority of local accuracy methods also produce a global estimate of the accuracy. This is often done by using the average local accuracy.

Group Name	Method	Local/Global	Inputs	Sequence features	Structure features	Predicted Features	Target Function	Machine Learning method
Studer (Waterhouse et al., 2018)	FaeNNz	Local (Global is avg. Local)	Model and full-length target sequence		Statistical Potentials of Mean Force + Distance Constraints from Templates + Solvent Acc.	Sec. Str and Surface Area	LDDT (local)	Multi-Layer Perceptron
McGuffin (Maghrabi and McGuffin, 2019) (McGuffin et al., 2019)	ModFOLD7	Local (Global is avg. of local)	Model and full-length target sequence	PSSM	Pairwise comparisons of generated reference models, residue contacts	Contacts, Sec. Str and Disorder	S-score (local)	Multi-Layer Perceptron
	ModFOLD7_cor	Local and optimised composite global score	Model and full-length target sequence	PSSM	Pairwise comparisons of generated reference models, residue contacts	Contacts, Sec. Str and Disorder	lDDT (local) GDT-TS (global)	Multi-Layer Perceptron
	ModFOLD7_rank	Local and optimised composite global score	Model and full-length target sequence	PSSM	Pairwise comparisons of generated reference models, residue contacts	Contacts, Sec. Str and Disorder	S-score (local) GDT-TS (global)	Multi-Layer Perceptron
Elofsson (Wallner and Elofsson, 2003)	ProQ2	Local (Global is sum of local)	Profile + model + predictions	PSSM	Atom contacts, residue contacts	Sec. Str and Surface Area	S-score (local)	Linear SVM
	ProQ3	Local (Global is sum of local)	Profile + model + predictions + energies	PSSM	Atom contacts, residue contacts + Energy terms	Sec. Str and Surface Area	S-score (local)	Linear SVM
	ProQ3D	Local (Global is sum of local)	Profile + structure + predictions + energies	PSSM	See Atom contacts, residue contacts + Energy terms	Sec. Str and Surface Area	S-score (local)	Multi-Layer Perceptron
	ProQ3D-TM	Local (Global is sum of local)	Profile + model + predictions + energies	PSSM	Atom contacts, residue contacts + Energy terms	Sec. Str and Surface Area	TM-score (local)	Multi-Layer Perceptron
	ProQ3D-lDDT	Local (Global is sum of local)	Profile + model + predictions + energies	PSSM	Atom contacts, residue contacts + Energy terms	Sec. Str and Surface Area	lDDT(local)	Multi-Layer Perceptron
	ProQ3D-CAD	Local (Global is sum of local)	Profile + model + predictions + energies	PSSM	Atom contacts, residue contacts + Energy terms	Sec. Str and Surface Area	CAD-score (local)	Multi-Layer Perceptron

(Continues)

Group Name	Method	Local/Global	Inputs	Sequence features	Structure features	Predicted Features	Target Function	Machine Learning method
Elofsson (Wallner and Elofsson, 2003)	ProQ4 (ProQ4)	Local (Global is sum of local)	Profile + DSSP	PSSM	DSSP (sec. Str and surface area)	Internally DSSP.	lDDT (local)	Deep Network
Han (Cheng et al., 2005)	SART_G	Global	Model + predictions + energies		Statistical Potentials + Solvent Acc + Sec. Str + Residue Contact	Sec. Str, Solvent Acc and Residue Contact	GDT-TS	Linear Regression
	SART_L	Local	Model + predictions + energies		Statistical Potentials + Solvent Acc + Sec. Str + Residue Contact	Sec. Str, Solvent Acc and Residue Contact	S-score	Linear Regression
	SARTclust_G	Global	Model + predictions + energies		Statistical Potentials + Solvent Acc + Sec. Str + Residue Contact	Sec. Str, solvent acc and residue contact	GDT-TS	Linear Regression
Venclovas (Olechnovič and Venclovas, 2014)	VoroMQA-A, VoroMQA-B	Local and global	Model	Not used	Voronoi tessellation-based contact areas.	Not used	Not used	Statistical potential
Cheng (Wang et al., 2010)	MULTICOM-CLUSTER	Global	Model and full-length sequence	Not used	Secondary structure, Solvent accessibility, residue contacts	Contacts, Sec. Str, surface area and structural scores	GDT-TS (global)	Deep network + ensemble
	MULTICOM-CONSTRUCT	Global	Model and full-length sequence	Not used	Secondary structure, solvent accessibility, residue contacts	Contacts, Sec. Str, surface area and structural scores	GDT-TS (global)	Deep network + ensemble
	MULTICOM NOVEL	Local (Global is sum of local)	Model and full-length sequence	PSSM, Amino acid encoding	Secondary structure, Solvent accessibility, Energy terms	Disorder, Sec. Str and surface area	S-score (local) and GDT-TS (global)	Deep network

Table 6.1. Overview of EMA methods discussed in this study and the way they were developed. Adapted from Cheng et al., (2019).

Different EMA methods utilise different descriptions of the models. Historically, EMA methods were often divided into single and consensus methods. Here, single methods only use a single model and predict the accuracy of that model (or regions of that model), while consensus methods compared a set of models and (often) assumed that the more similar they were the more likely they were to be correct. In earlier CASPs a category of "quasi-single" methods also existed. These methods do not require a set of models, as for the consensus methods, instead they compare the model with a set of internally generated models, assuming that the more similar the model is to the internally generated models the better it is. Now, many methods combine many of the methods making it hard to exactly classify each method, but we have tried to describe the most important features of all EMA methods in Table 6.1.

6.3.6. Relative performance of EMA methods depending on evaluation metric

Using different reference-based scores (evaluation metrics) may lead to different rankings of models and different best models. Some EMA methods are trained to predict specific reference-based scores, for example, GDT-TS or TM-score. Therefore, it might be expected that the relative performance of EMA methods may depend on the use of specific evaluation scores. To test whether this is the case, we asked how successful different EMA methods are in selecting models according to four different scores: two superposition-based scores (GDT-TS and TM-score) and two superposition-free scores (lDDT and CAD-score). To make the comparison straightforward, for every reference-based score we used Z-scores instead of raw values. For every CASP13 target, we derived z-score values using the procedure typically used in CASP assessments: calculate z-scores for all models; exclude models with z-scores lower than −2 and recalculate Z-scores; assign −2 to every Z-score lower than −2. For each EMA method, we then summed Z-scores of selected models for all CASP13 targets. The evaluation was done separately for GDT-TS, TM-score, lDDT and CAD-score. If a given EMA method is equally successful in selecting models according to each of the four reference-based scores, then the contribution of each type of z-score would be approximately the same, or ~25% of the total sum of z-scores for GDT-TS, TM-score, lDDT, and CAD-score (100%). We tested whether this is the case by computing the actual deviation from 25% for each type of z-scores. The positive and negative values indicate correspondingly that the

EMA method is either relatively more or less successful according to that score, but not its absolute performance.

6.4. Results and Discussion

The value and potential of EMA methods can be seen when selecting the top model for each target, see Figure 6.3. Here a small improvement can be obtained when using the best EMA methods compared with using the best server alone. The average GDT-TS for the best server on the 80 full-length targets used in the evaluation of the EMA methods is 56.3. When the best EMA method is used to select the best model the average GDT-TS score is 57.6. Moreover, in total nine EMA methods select models better than the best individual server. However, the potential for improvement is quite significant. If the best model for each target were selected, the average GDT-TS would increase by 10% to 63.3%. Using any other measure, similar numbers appear. Unfortunately, no EMA method is close to always identifying the best model yet. The value of EMA methods seems slightly bigger for hard targets (2.5%-6.0%) compared with easier targets (0.8%-3.5%). Also, as expected there is more room for improvement for the harder targets, see Figure 6.3.

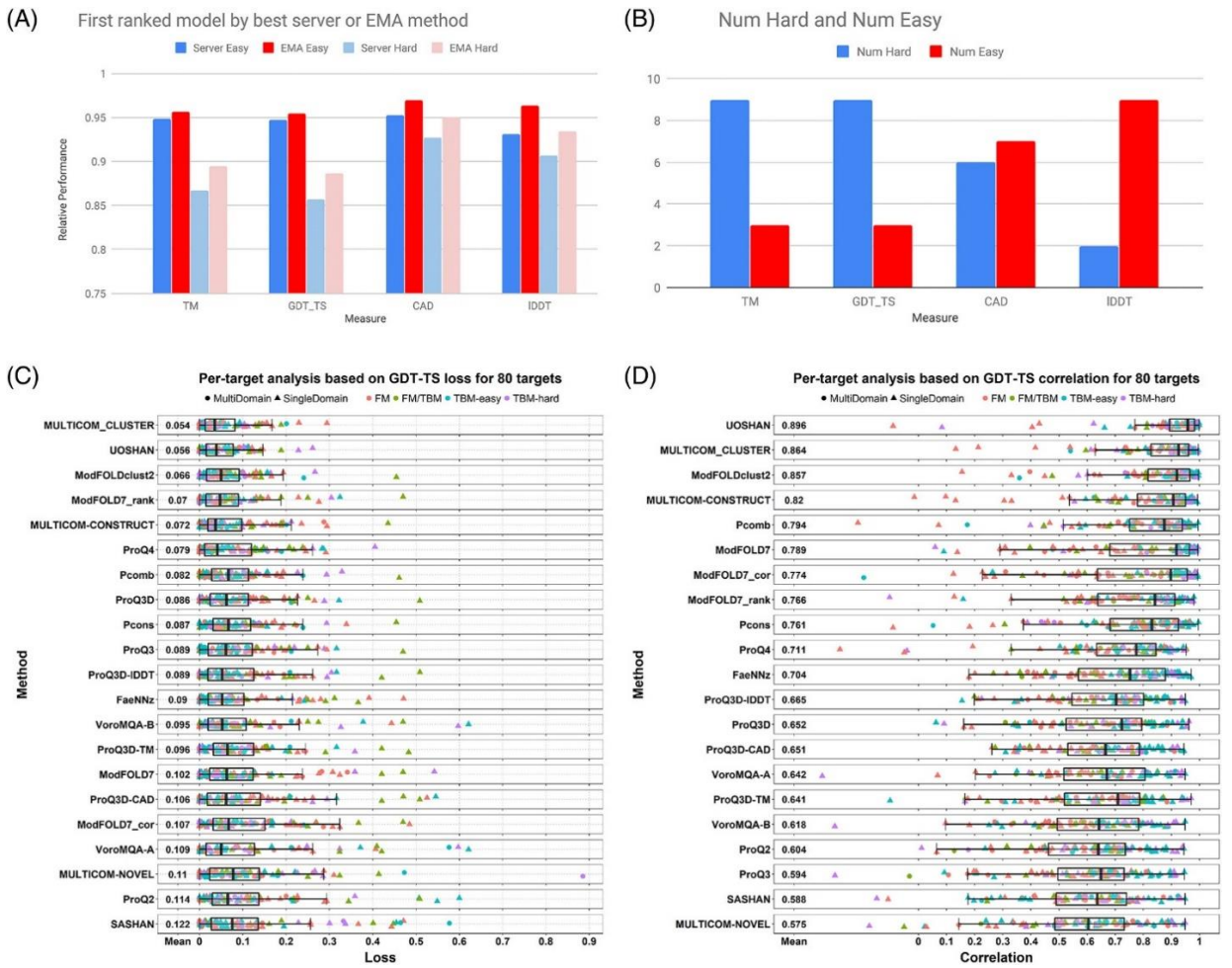

Figure 6.3. A new approach of evaluation for benchmarking the top ranked EMA methods in CASP13 including ModFOLD7. A, Comparison of average score of the first ranked model for each target in relationship to the score of the best model made by any server using different evaluation measures. In blue the best server and in red the model selected by the best EMA method. In darker colours easy targets (average GDT-TS > 0.5) and in lighter colours the harder targets. In (B) the number of EMA methods that are better than the best server is shown. C, Boxplot of per target loss for the top group methods based on the GDT-TS score. The rectangular box shows the median, 25% percentile, 75% percentile of the loss on 80 targets. Dots of different shapes/colours denote the loss of individual targets of different types (MultiDomain, SingleDomain, FM, FM/TBM, TBM-easy, and TBM-hard). The mean of the loss is also listed next to the name of each method. D, Boxplot of per target correlation for the top group methods based on the GDT-TS score. Adapted from Cheng et al., (2019).

6.4.1. Evaluation metric analysis

Results of the four different reference-based evaluation are presented in Figure 6.4. Several inferences can be drawn from these results. First, the relative success of most EMA methods indeed depends on the evaluation metric. Only some consensus-based methods were relatively balanced in this regard. Strikingly, all the ModFOLD7 alternatives and the absolute majority of EMA methods showed relatively better performance according to the superposition-free scores, lDDT, and CAD-score (the latter in particular). It is interesting that even an EMA method trained using TM-score as a target function (ProQ3D-TM) was still relatively more successful according to the superposition-free scores. The results suggest that for single-model EMA methods, it is generally easier to predict superposition-free scores than the superposition-based scores. In turn, this might be interpreted as the ability of superposition-free scores to provide a more objective definition of model accuracy.

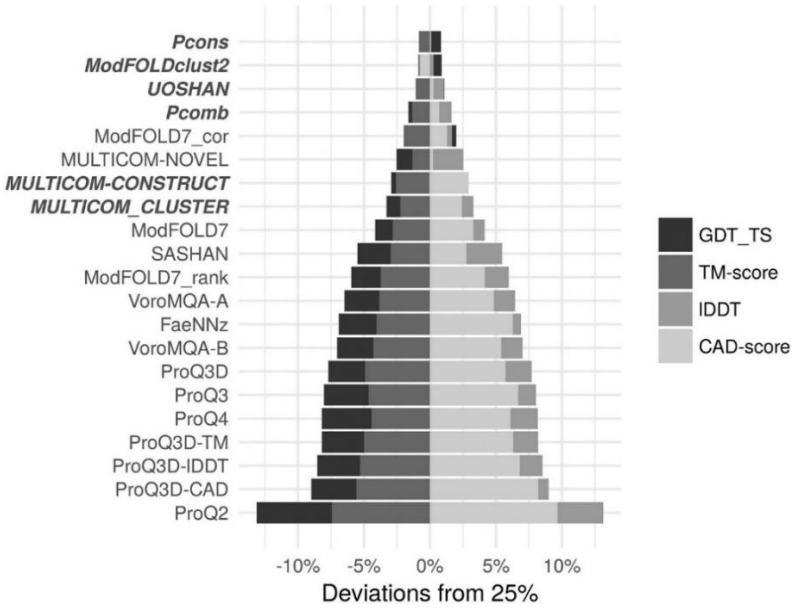

Figure 6.4. Relative success of different EMA methods in predicting four reference-based evaluation scores. The relative success according to each of the four scores is expressed as the difference between the actual percentage and 25%. Positive values indicate relatively higher success, negative values indicate relatively lower success. For each method positive values balance out negative ones (their sum is zero). EMA methods are ordered by increasing disbalance, which is unrelated to the absolute performance. The methods that are not classified as single-model are indicated with the bold italic font. Adapted from Cheng et al., (2019).

6.4.2 Correlation of top N models

When choosing an evaluation metric for EMA methods, it is essential that this metric rates the methods based on whether they accurately estimate the correctness of high-quality models, but it is less important to rate them based on whether they accurately estimate the correctness of low-quality models. For that reason, it has been argued that the correlation between the predicted and real scores of models is not a useful metric when evaluating EMA methods, as it gives equal importance to all models. As a result, one of the evaluation metrics that are currently most employed is the first-ranked score loss, as it takes into account only the best ranked model for each target, so gives more importance on how the EMA methods evaluate the high-quality models. However, the first ranked score loss has its disadvantages, because it might be somewhat noisy when the differences between the predicted scores are tiny.

Here, we suggest a novel way to evaluate the EMA methods, see Figure 6.5. We calculate the average per target Pearson correlation and first ranked lDDT loss for Top N models, where Top N models are selected based on their lDDT scores. In such a way we evaluate how the EMA methods perform when all the models are high quality, but also when they are of varying qualities.

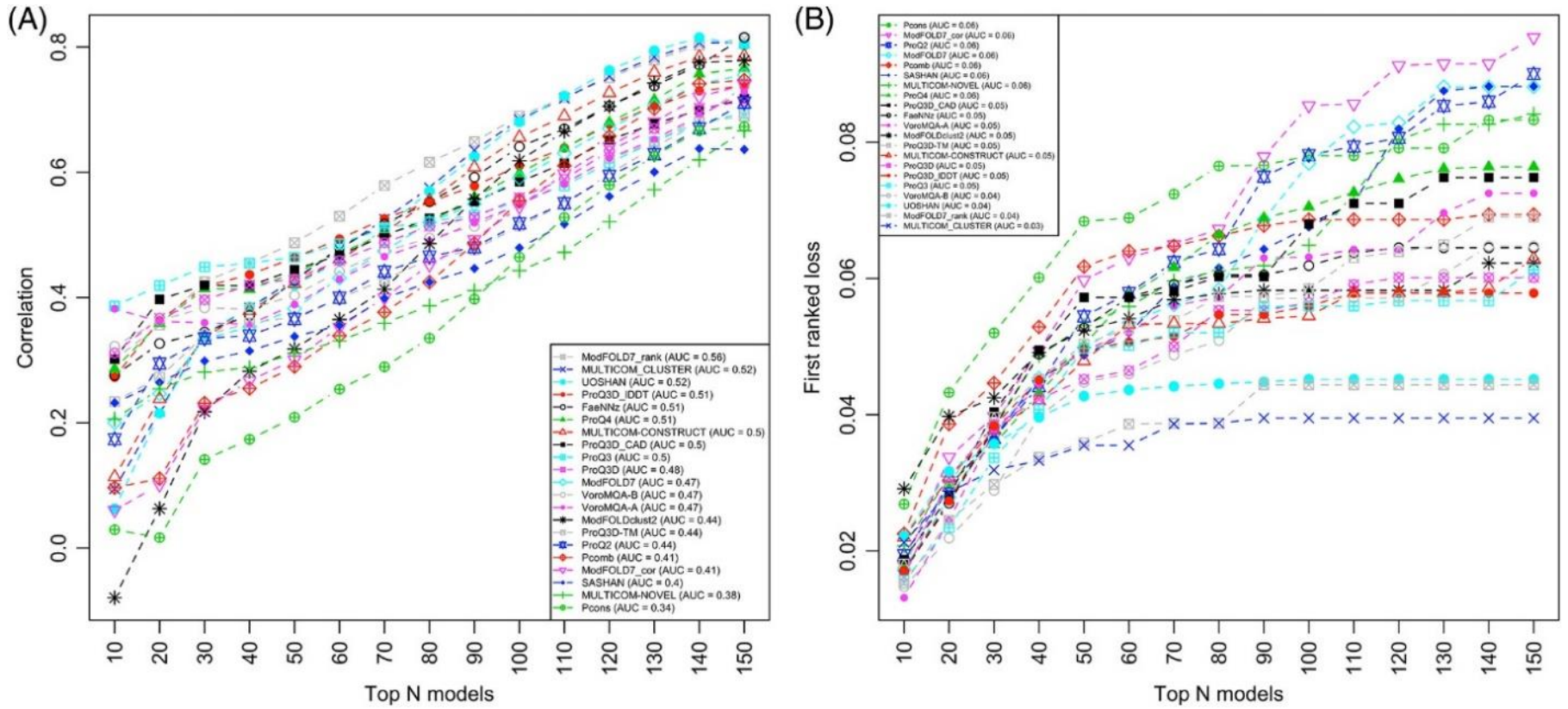

Figure 6.5. Line charts representing the top ranking EMA methods based on the top N models evaluation. (A) Average per target Pearson correlations between lDDT and the predicted accuracy scores of our EMA methods for top N models. (B) First ranked lDDT loss for top N models. Top N models are selected based on lDDT scores. For example, top 10 models are the 10 models that have the best lDDT scores. The methods in the legend are sorted according to Area Under the Curve (AUC) values. Adapted from Cheng et al., (2019)

The output of the top N models correlation is shown in Figure 6.5. The results show that ModFOLD7_rank had the highest correlation with an Area Under the Curve (AUC) of 0.56. The method also showed the lowest first ranked loss after MULTICOM_CLUSTER with an AUC of 0.04.

One important thing that we learn from this analysis is that the performance of different EMA methods depends a lot on how many of the top models we choose as the evaluation data set. Recently it has been a standard in CASP to evaluate all the methods on 150 models per target that are selected by an arbitrary consensus method (ie, the "stage 2" evaluations). We believe that the evaluation would be more independent if we evaluate the methods on a range of different data set sizes.

6.4.3 ModFOLD7 variants

Looking at global scoring evaluations on the CASP13 data, as expected the ModFOLD7_rank method was the best variant at ranking or selecting the best models and the ModFOLD7_cor variant was better at reflecting observed accuracy scores or estimating the absolute error, while the ModFOLD7 method was more balanced in terms of performance. For local scoring, the ModFOLD7_rank and ModFOLD7 variants performed better according to S-score and ModFOLD7_cor method according to lDDT.

ModFOLD7 is also one of the EMA servers that are continuously independently benchmarked for local EMA performance by the evaluating organisation, CAMEO (identified as server 28). The method has been independently verified to be an improvement on our previous methods (ModFOLD4 and ModFOLD6). At the time of writing, the ModFOLD7_(res)_lDDT method ranks among the top few QE servers on CAMEO. For the last year, the CAMEO public EMA data (https://www.cameo3d.org/) shows that ModFOLD7 is one of the leading public EMA methods for producing local (per-residue) quality scores. The results from CAMEO also show that ModFOLD7 is performing significantly better than its previous versions, ModFOLD6 and ModFOLD4 (Maghrabi and McGuffin, 2017) (McGuffin et al., 2013) (Table 6.2).

Server	Structural models			ROC		ROC normalised		PR		PR normalised	
	Submitted	Received	%	AUC 0,1	AUC 0,0.2	AUC 0,1	AUC 0,0.2	AUC 0,1	AUC 0.8,1	AUC 0,1	AUC 0.8,1
QMEANDisCo	10388	10329	99.4	0.94	0.8	0.94	0.79	0.91	0.69	0.91	0.69
ModFOLD7_lDDT	10388	8442	81.3	0.91	0.71	0.74	0.58	0.87	0.6	0.71	0.49
ModFOLD6	10388	7375	71.0	0.89	0.65	0.63	0.46	0.84	0.57	0.59	0.41
QMEAN 3	10388	9544	91.9	0.87	0.61	0.8	0.56	0.81	0.53	0.74	0.49
ProQ2	10388	9695	93.3	0.86	0.59	0.8	0.55	0.8	0.51	0.74	0.47
ModFOLD4	10388	6009	57.8	0.85	0.58	0.49	0.34	0.78	0.5	0.45	0.29

Table 6.2. Top EMA methods in CAMEO. 1 year of data downloaded from http://www.cameo3d.org/. 1-year – (2018-07-20 - 2019-07-13) - "All" dataset. The table is sorted by the ROC AUC score. ROC = Receiver Operating Characteristic. AUC = Area Under the ROC Curve. PR = Precision and Recall. Adapted from Cheng et al., (2019).

6.4.4 ModFOLD7 vs ModFOLD6

Specific improvements over ModFOLD6 from our in-house analysis using CASP11, CASP12, and CASP13 data were calculated for global and local scoring, and a summary of selected key results are shown in Figure 6.6. The ModFOLD7 variants showed small but significant improvements in both local scoring and selection of best models across all three datasets (CASP11-13), compared with the equivalent ModFOLD6 variants. The plots on top panels of the figure demonstrate the 3 alternative optimised scoring methods of the ModFOLD7 server being benchmarked against their respective previous versions from the ModFOLD6 server. For the cumulative GDT-TS of top ranked model, ModFOLD6_rank method was giving a score below 44.5 as their highest, whereas ModFOLD7_rank was able to cross the 45 and go higher. For the Pearson correlation comparing the predicted score versus the observed score (GDT-TS), ModFOLD6_cor achieved a correlation 0.9250 while for ModFOLD7_cor the correlation was found to be over 0.9300. For the evaluation of local model quality prediction accuracy using the Area Under the ROC Curve (AUC) (where, residues with lDDT scores <= 0.6 = 0), ModFOLD6 could not reach an AUC score of 0.93, whereas ModFOLD7 was closer to 0.95.

The plots on left panels of Figure 6.6 also showed that ModFOLD7 rank outperformed ModFOLD6_rank in terms of selecting the best models measured by cumulative GDT-TS; a

significant improvement on all three data sets. In the middle panels, the ModFOLD7_cor method outperforms ModFOLD6_cor in terms of the correlation of the global output score vs the GDT-TS score on some data sets. However, no consistent improvement in global correlations was observed for ModFOLD7_cor over ModFOLD6_cor across all data sets, and any improvements seen were dependent on the chosen data set and/or the observed score (eg, ModFOLD7 outperforms ModFOLD6_cor according to the lDDT score on the CASP13 set, but not by GDT-TS). Finally, in terms of local accuracy estimates, based on both the lDDT scores (Figure 6.6, right panels) and S-scores, we also observed a significant improvement with the newer ModFOLD7 variants vs our older ModFOLD6 method.

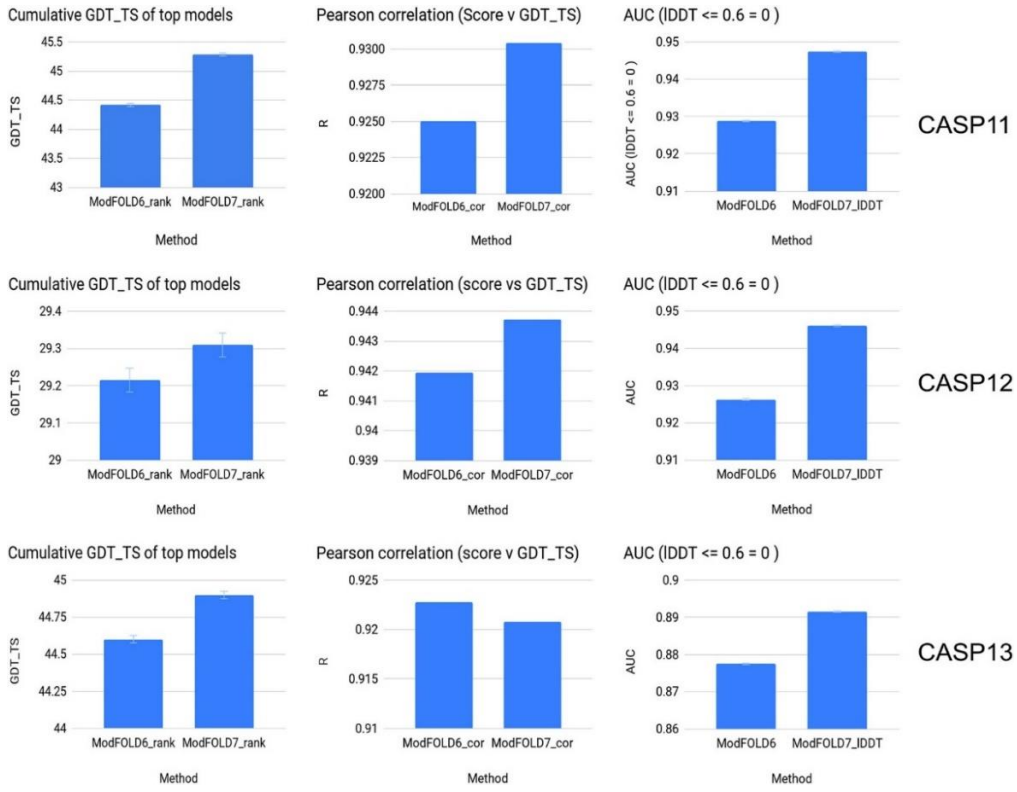

Figure 6.6. Histograms summarising the improvements in ModFOLD7 variants vs ModFOLD6 variants on CASP11-13 datasets. Model data from EMA stages 1 and 2 are combined with duplicate models removed. Left panels show the ranking/model selection performance measures by cumulative GDT-TS scores of the top selected models by each method. Middle panels show Pearson correlation coefficients of global predicted accuracy vs observed accuracy according to GDT-TS. Right panels show performance of local accuracy estimates as measured by the area under the curve (AUC) scores from ROC analysis using the lDDT observed local scores. Adapted from Cheng et al., (2019).

Such results indicate that our latest version, ModFOLD7 has demonstrated progress in performance compared to ModFOLD6 and according to many measures the improvements are significant. The consistent performance improvements of ModFOLD7 variants over ModFOLD6 were due to; (a) The addition of more input scores and correspondingly more input and hidden layer neurons to the neural network, (b) Training to different local target functions (the lDDT score and the S-score), and (c) Optimising for different evaluation metrics using a higher number of global scoring metrics.

6.6. Conclusions

We show that there has been a marginal but significant improvement since CASP12 in ModFOLD7 and the other EMA methods over the previous versions of methods. It can be noted that many of the improved methods use deep learning, but in different ways. The rapid development of deep learning models as exemplified here might indicate that the best way to use machine learning for model accuracy evaluations is still not developed. We also notice that on average the best EMA methods select models that are better than those provided by the best server. However, still, much more significant improvements could be achieved if there were possible ways to always select the best model for each target. Finally, we do notice systematic differences when using different model evaluation methods. Single model methods perform relatively better when using superposition free evaluation methods.

Chapter 7
ModFOLD Applications

Work presented in this chapter has been published in the following papers:

McGuffin, L.J., Shuid, A.N., Kempster, R., **Maghrabi, A.H.A**., Nealon, J.O., Salehe, B.R., Atkins, J.D., Roche, D.B., 2018. Accurate template-based modeling in CASP12 using the IntFOLD4-TS, ModFOLD6, and ReFOLD methods. Proteins: Structure, Function, and Bioinformatics 86, 335–344. https://doi.org/10.1002/prot.25360

McGuffin, L.J., Adiyaman, R., **Maghrabi, A.H.A.**, Shuid, A.N., Brackenridge, D.A., Nealon, J.O., Philomina, L.S., 2019. IntFOLD: an integrated web resource for high performance protein structure and function prediction. Nucleic Acids Res 47, W408–W413. https://doi.org/10.1093/nar/gkz322

Keasar, C., McGuffin, L.J., Wallner, B., Chopra, G., Adhikari, B., Bhattacharya, D., Blake, L., Bortot, L.O., Cao, R., Dhanasekaran, B.K., Dimas, I., Faccioli, R.A., Faraggi, E., Ganzynkowicz, R., Ghosh, Sambit, Ghosh, Soma, Giełdoń, A., Golon, L., He, Y., Heo, L., Hou, J., Khan, M., Khatib, F., Khoury, G.A., Kieslich, C., Kim, D.E., Krupa, P., Lee, G.R., Li, H., Li, J., Lipska, A., Liwo, A., **Maghrabi, A.H.A**., Mirdita, M., Mirzaei, S., Mozolewska, M.A., Onel, M., Ovchinnikov, S., Shah, A., Shah, U., Sidi, T., Sieradzan, A.K., Ślusarz, M., Ślusarz, R., Smadbeck, J., Tamamis, P., Trieber, N., Wirecki, T., Yin, Y., Zhang, Y., Bacardit, J., Baranowski, M., Chapman, N., Cooper, S., Defelicibus, A., Flatten, J., Koepnick, B., Popović, Z., Zaborowski, B., Baker, D., Cheng, J., Czaplewski, C., Delbem, A.C.B., Floudas, C., Kloczkowski, A., Ołdziej, S., Levitt, M., Scheraga, H., Seok, C., Söding, J., Vishveshwara, S., Xu, D., Crivelli, S.N., 2018. An analysis and evaluation of the WeFold collaborative for protein structure prediction and its pipelines in CASP11 and CASP12. Scientific Reports 8, 9939. https://doi.org/10.1038/s41598-018-26812-8

Khaled A. Sahli, Gagan D. Flora, Parvathy Sasikumar, **Ali H. Maghrabi**, Lisa-Marie Holbrook, Sarah K. AlOuda, Tanya Sage, Alexander R. Stainer, Recep Adiyaman, Mohammad AboHassan, Marilena Crescente, Alexander P. Bye, Liam J. McGuffin, Jonathan M. Gibbins., 2019. Structural, Functional and Mechanistic Insights Uncover the Fundamental Role of Orphan Connexin 62 in Platelets. Submitted to Cell.

7.1. Background

The application of accurate methods for producing EMA has become a principal focus for researchers who need to establish the utility of their 3D protein models. Having any percentage of improvement in the EMA scoring methods can lead directly to improved quality and higher accuracy of 3D modeling. Therefore, relying on the model quality assessment has always been at the core of our modeling strategy in both CASP and real-world use cases. IntFOLD is our main server of predicting 3D protein models. The method has developed tremendously over the past years, and that is due to the relative development of the integrated MQA which has been inspired by previous research with the nFOLD (Jones, 1999) and GenTHREADER (McGuffin and Jones, 2003) methods. Focused development on methods for EMA was initiated from the 7th round of CASP when the QA category was introduced. In CASP7, the ModFOLD method (McGuffin, 2007) was developed purely to tackle to the QA problem, and it was utilised in parallel with the third version of nFOLD. A further improvement in scoring 3D models was achieved after integrating the clustering-based variant (ModFOLDclust) (McGuffin, 2008). The improvement was independently verified when the method was used subsequently in CASP8 for the predictions in the quality assessment category (Buenavista et al., 2012), and also for ranking of server models for the manual predictions of the Tertiary Structure (TS) category.

In the 9th season of CASP, assessors requested that predictors should include error estimates for every submitted 3D model, which were scaled in Angstroms in place of the temperature factor (B-factor) field for each atom record. After applying that request in CASP9, a change of performance was noticed. The assessors then began to emphasize the value of the “B-factor errors” or what has now been termed the ASE scores. The importance of assessing the quality of a 3D predicted protein model has become as important as considering the *E*-value when using BLAST.

The first integration of ModFOLD with the original IntFOLD server method, was in 2011, when ModFOLDclust2 was integrated for ranking and providing the ASE scores for single template modeling (Roche et al., 2011). The output from ModFOLDclust2 included ASE scores for each generated IntFOLD model which were included in the “B-factor” columns of all atom coordinate files. The high performance of the IntFOLD-TS method (McGuffin and Roche, 2011) in CASP9 gained attention in the consideration of model reliability, and the “B-factor” scores were independently evaluated in the TBM category (McGuffin and Roche, 2011).

In CASP10, our group aimed to exploit the strengths in ASE scoring by using our QA methods in multiple-template modeling protocols rather than just in single-template modeling (Buenavista et al., 2012). Some extra sequence-structure alignment methods were added for IntFOLD3-TS (McGuffin et al., 2015), and it was evaluated in CASP11 while using the same multiple-template modeling ranking and ASE scoring protocols of IntFOLD2.

In every CASP experiment since CASP8, the McGuffin group has used the ModFOLD variants for both ranking server models and adding ASE scores for all of their manual target model submissions. The IntFOLD server has shown incremental improvements cumulative GDT-TS scores and consistent success in the server category, however the main strengths gained have been through developments in the ASE scores achieved by ModFOLD.

In this chapter, we describe each of the major applications of the ModFOLD versions and variants throughout the period of this study; we describe the projects where ModFOLD has been involved, the way it has been integrated, and the improvements that ModFOLD has made to our understanding of protein structures.

Finally, in this chapter we describe the application of our method in the investigation of the Connexin62 complex, which is a new orphan hexametric hemichannel protein that has been found to have a fundamental role in the thrombi mechanism in platelets. The protein was uncovered using several techniques including ModFOLD.

7.2. IntFOLD

Over the past 20 years, the community of structure prediction has achieved great advances with several major improvements in the TBM, FM and EMA coming in the last few CASP experiments (Kryshtafovych et al., 2018) (Abriata et al., 2018). The IntFOLD server components along with their upgraded versions have been independently benchmarked in every CASP from CASP9 to the latest CASP13 experiments. The methods have also been benchmarked continually by the CAMEO project (Haas et al., 2018). Over the years, the improvements of the ModFOLD methods specifically have led to our own advances in the IntFOLD server performance, and particularly in the ranking of 3D models and ASE scoring improvements (Kryshtafovych et al., 2018) (McGuffin et al., 2018).

Successive IntFOLD versions have been described and benchmarked in several publications. The initial versions were described in the Nucleic Acid Research journal (Roche et al., 2011) and 2015 (McGuffin et al., 2015). Over the years the IntFOLD server has served more than 15k unique users, and it has completed more than 200k predictions since its inception. The component methods of IntFOLD server have been developed in order to model the 3D structure of proteins as well as the interactions for a diverse range of specialisations across the life sciences. Numerous studies have been carried out by researchers using our tools to help them investigate their own proteins of interest. For example, modelling novel proteins in the *Drosophila melanogaster* genome (Dunwell et al., 2013); revealing new interactions and mechanisms for the regulation of mammalian GCKIII kinases (Fuller et al., 2012) (Sugden et al., 2013), explaining the evolutionary resurrection of flagellar motility in *Pseudomonas fluorescens* (Taylor et al., 2015), annotating the proteome of barley powdery mildew structurally and functionally (*Blumeria graminis* f. sp. *hordei*) (Bindschedler et al., 2011) and understanding the effect of the missense mutation associated with dermatosparaxis (Monteagudo et al., 2015).

There are six component methods that are integrated and can be accessed through the single interface of the IntFOLD server. Firstly, IntFOLD-TS, the main tool which predicts the tertiary structure of proteins (McGuffin et al., 2018) (Roche et al., 2011) (McGuffin et al., 2015) (McGuffin and Roche, 2011) (Buenavista et al., 2012). Secondly, ModFOLD, and this is the key stone for 3D model selection and ASE scoring (McGuffin et al., 2018) (Maghrabi and McGuffin, 2017). Thirdly, ReFOLD, the tool which refines the proteins after being predicted and quality assessed by ModFOLD (Shuid et al., 2017) (Adiyaman and McGuffin, 2019). Fourthly, DISOclust - this tool is for predicting the disordered regions in the modelled proteins (McGuffin, 2008) (Atkins et al., 2015). Fifthly, DomFOLD - this tool predicts the structural domains of the predicted protein models (Roche et al., 2011) (McGuffin et al., 2015). Finally, FunFOLD, a tool for ligand binding site prediction (Roche et al., 2011) (Roche et al., 2013). Each component method has its own category in CASP and has been tested independently with the other competitors of its kind.

Since its inception, IntFOLD has been through many enhancements to the server methodology, but the foundation has always been the TS prediction algorithm with integrated model quality assessment at its core (McGuffin et al., 2015). The world leading quality self-estimates and ranking have been the key contributing factors to the historical success of the component methods (Kryshtafovych et al., 2018) (Kryshtafovych et al., 2018) (McGuffin et al., 2018) (McGuffin and

Roche, 2011) (Noivirt-Brik et al., 2009) (Schmidt et al., 2011) (Kryshtafovych et al., 2014) (McGuffin, 2009) (Kryshtafovych et al., 2016).

7.2.1. ModFOLD6 in IntFOLD4

The IntFOLD method (Version 4) integrates the ModFOLD6 (Maghrabi and McGuffin, 2017) variant, ModFOLD6_rank, for improved selection and ASE scoring. Such an integration has given IntFOLD4 significant improvements to be able to compete with and rank among the best other servers in the protein prediction field.

7.2.1.1. Methods

In 2016, the McGuffin group participated in CASP12 with an updated version of IntFOLD, version 4. This version was developed with the feature of producing results for the TS prediction category (including the now mandatory built–in ASE scores). For the local quality assessment predictions, each TS model file included predicted distances in the B-factor column. These ASE scores were produced by the ModFOLD6 (Maghrabi and McGuffin, 2017) EMA method. (N.B. predictions in the EMA/QA category of CASP12 were all also returned by our ModFOLD6 and ModFOLDclust2 servers, see Chapter 4 for more details).

The aim of developing the updated IntFOLD4-TS was to gain the ability to identify and then to attempt to fix the local errors in an initial pool of single template models via iterative multi-template modeling. The technique of this method was built upon our previous CASP successful results in accurately predicting local errors in our models (McGuffin and Roche, 2011) - we took the global and local per-residue errors into consideration during the multiple template selection stage (Buenavista et al., 2012).The IntFOLD4 pipeline can be broken down into two major stages: (1) single template modeling with ASE scoring and (2) QA guided multiple template modeling with ASE scoring (Figure 7.1)

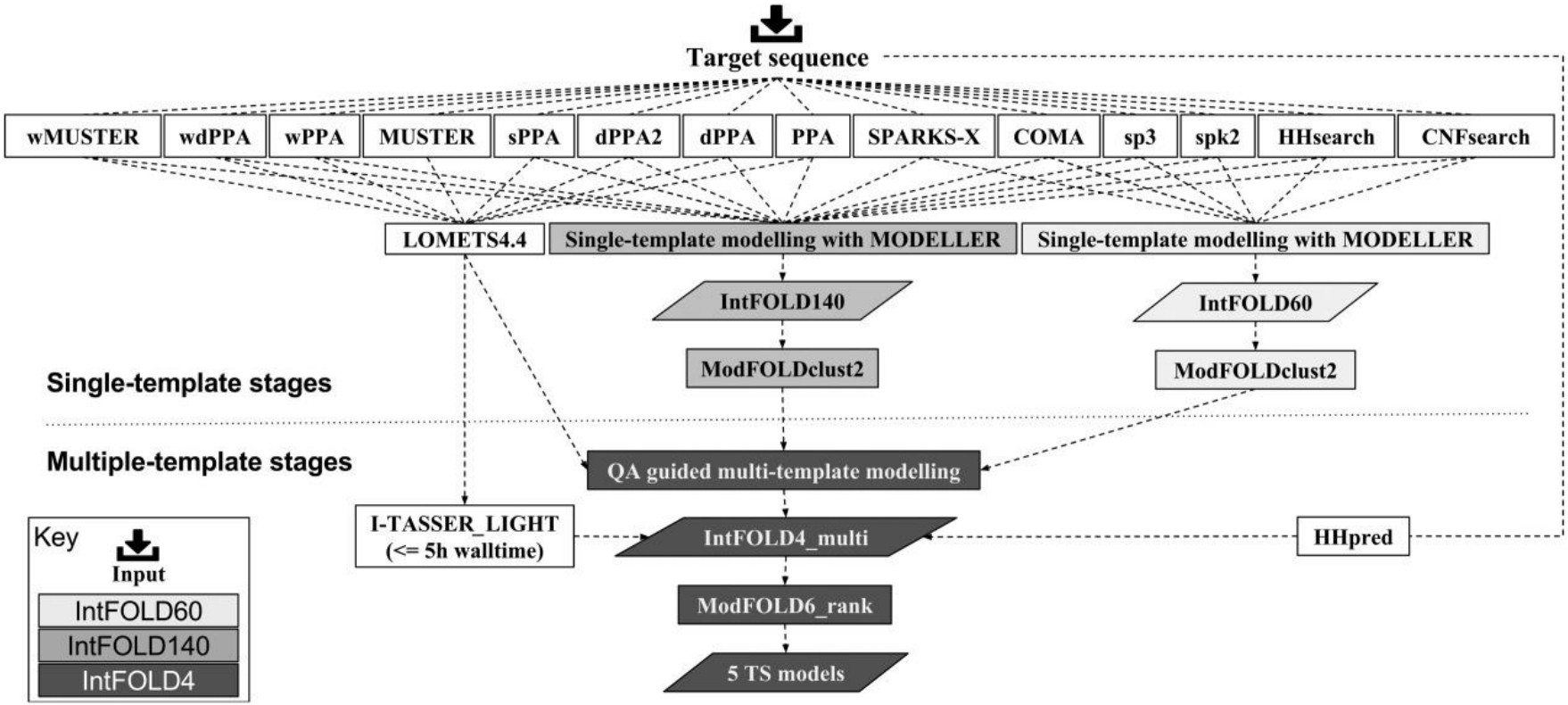

Figure 7.1. Flowchart outlining the principal stages of stages of the IntFOLD4-TS prediction pipeline. Rectangles show processes, parallelograms show datasets. The only input is the target sequence. The initial single-template modelling stages start with 14 sequence-structure alignment methods (eight from the LOMETS (Wu and Zhang, 2007) package and six others as described in the main text (Zhou and Zhou, 2005) (Söding, 2005) (Margelevicius and Venclovas, 2010) (Yang et al., 2011) (Ma et al., 2013). Single-template models are built from the various alignment methods using MODELLER (Uziela and Wallner, 2016) (creating the IntFOLD60, IntFOLD140 model datasets) and then ranked with ModFOLDclust2 (McGuffin and Roche, 2010). LOMETS4.4 is also used to rank the backbone models produced by its own component threading methods. The multiple template modelling stages include QA guided multi-template modelling (using the scores from ModFOLDclust2) in order to generate a set of multi-template models. Additionally, models from HHpred (Meier and Söding, 2015) and I-TASSER_LIGHT (Roy et al., 2010) are added to the final IntFOLD4_multi set for evaluation. The ModFOLD6_rank method (Maghrabi and McGuffin, 2017) is used for ASE and final model selection. Adapted from McGuffin et al. (2018).

In the first major stage of the single template modelling, the server ran 14 different fold recognition methods (in-house), generating up to 10 sequence-to-structure alignments for each method and resulting in up to 140 alternative single-template-based models being generated for each CASP target. The following fold recognition methods used were: SP3 (Zhou and Zhou, 2005), SPARKS2 (Zhou and Zhou, 2005), HHsearch (Söding, 2005), COMA (Margelevicius and Venclovas, 2010), SPARKSX (Yang et al., 2011), CNFsearch (Ma et al., 2013), and the eight alternative threading methods that are integrated into the current LOMETS package (Wu and Zhang, 2007) (PPA, dPPA, dPPA2, sPPA, MUSTER, wPPA, wdPPA, and wMUSTER). In order to assign global and local model quality scores at the end of the first stage, all single-template models were assessed using ModFOLDclust2 (McGuffin and Roche, 2010).

For the second major stage, sequence-structure alignments are selected using the single-template model quality scores, and other criteria involving template coverage in order to build multiple-template models (Buenavista et al., 2012). Our overall aim here is selecting appropriate target-template alignments that would minimise local errors in the final models. There are four main alternative alignment selection methods that are included in the MTM stage, they are termed as multi1, 2, 3, and 4, and worked mainly in building the 3D model. Firstly, multi1, and this method simply used the top 2 alignments according to the template ranking. Secondly, multi2, this used the top ranked alignment and any subsequent alignments if there were no less than 40 new residues covered, and 20 residues were overlapping. The third method, multi3, used the top ranked alignment and any subsequent alignment, but only if the overlapping region was predicted to increase local model quality. The fourth method is multi4, and it used the top ranked alignment and any subsequent alignments, but only if the coverage was increased by at least once residue. Additionally, four variants on these methods (multi5-multi8) repeated multi1-multi4, respectively. However, the alignments for each of the single-template methods were firstly reranked based on the ModFOLDclust2 predicted global model quality scores. These methods which uses MTM approaches were first introduced in our IntFOLD2-TS method. All the methods were described and benchmarked in our article which was published in 2012 (Buenavista et al., 2012). The result of having these alternative MTM alignment selection methods is generating a new population of up to 124 multi-template models for each target. In addition, I-TASSER_LIGHT20 (I-TASSER 4.4 run in “light mode” with wall-time restricted to 5 h; for sequences < 600 residues) and HHpred21 were used to generate three models each, which were then added into the final pool of alternative multi-template models for ranking. In the final stage, the ~130 models in the final reference set

were then evaluated using our ModFOLD6_rank1 (Maghrabi and McGuffin, 2017) QA method, and the top five ranked models were submitted as the final IntFOLD4-TS predictions.

7.2.1.2. Results

Before subjecting IntFOLD4 to blind testing in CASP12, the new method components were benchmarked locally in order to confirm the level of their performance as well as consistency. Another intention was to check if the new methods were performing better than the older versions of server TS methods: IntFOLD2-TS (Buenavista et al., 2012) and IntFOLD3-TS (McGuffin et al., 2015), and QA methods: ModFOLD4 (McGuffin et al., 2013) and ModFOLD5. In CASP11, the IntFOLD3-TS method was used for our server TS predictions and ModFOLD5 (which was similar in performance to ModFOLD4) was used in order to select the top server models for our manual submissions.

ModFOLD6_rank was evaluated via in-house benchmarking (Figure 7.2). The method was compared with other EMA methods for model selection using data collected from the previous CASP. From the results it can be seen that ModFOLD6_rank showed higher cumulative GDT-TS scores ($\sum$GDT = 44.42) for model selection than its component methods (Maghrabi and McGuffin, 2017) (Uziela and Wallner, 2016). ModFOLD6_rank also outperformed the previous versions (ModFOLD5_single with $\sum$GDT-TS = 40.06 and ModFOLDclust2 with $\sum$GDT-TS = 42.68) of ModFOLD which were used in CASP11 for EMA and model selection.

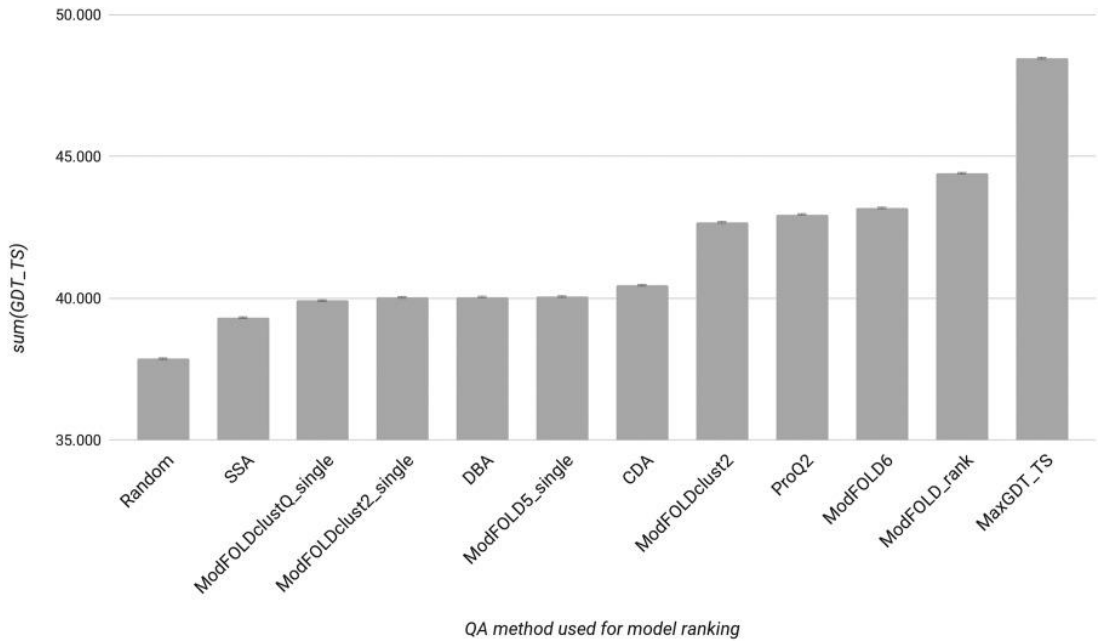

Figure 7.2. Benchmarking the performance of QA methods for model selection using CASP11 data, prior to CASP12. ModFOLD6_rank versus other global scoring methods: SSA, Secondary Structure Agreement; DBA, Disorder B-factor Agreement; CDA, Contact Distance Agreement (Maghrabi and McGuffin, 2017). Cumulative GDT scores for the top selected models from the QA targets (models from QA round1 and round2 combined, 84 targets with structure). The maximum possible GDT-TS (MaxGDT-TS) is the cumulative score obtained by selecting the best model available for every target. The error bars show the Standard Error in GDT-TS ($\sigma/\sqrt{n}$, where σ is the standard deviation and n is the number of targets (84)). Adapted from McGuffin et al. 2018.

After integrating ModFOLD6_rank method within the IntFOLD4-TS pipeline, the server was benchmarked against the other prediction methods as well as the previous versions of IntFOLD using the CAMEO resource (Haas et al., 2013). Table 7.1 shows a direct comparison of the performance between all servers including the previous IntFOLD versions. The results of 12 months of data and a common subset of 500 targets from CAMEO-3D shows that IntFOLD4-TS outperformed all servers except Robetta. (more comparisons between IntFOLD4-TS and Robetta are shown in Appendix 10. The 6 months of data from CAMEO analysis also shows the same server performance ranking (Appendix 11). Additionally, ModFOLD6 server is benchmarked separately with CAMEO-QE in terms of ASE/local score predictions. This continuous benchmarking confirms that ModFOLD6 outperformed the older versions of ModFOLD as well as most other EMAs.

	Average lDDT		Average CAD score		Average lDDT-BS	
Server Name	**Dif.**	**Ref.**	**Dif.**	**Ref.**	**Dif.**	**Ref.**
Robetta	−1.63	70.9	−0.02	0.7	2.73	68.86
IntFOLD4-TS	0	69.27	0	0.68	0	71.6
RaptorX	0.82	68.45	0	0.67	4.36	67.24
IntFOLD3-TS	1.74	67.53	0.02	0.66	3.02	68.57
IntFOLD2-TS	1.98	67.28	0.02	0.66	2.64	68.96
HHpredB	2.09	67.17	0	0.67	2.59	69.01
SWISS-MODEL	3.82	65.44	0.04	0.64	1.1	70.5
SPARKS-X	5.26	64.01	0.03	0.64	5.54	66.06
Princeton_TEMPLATE	9.36	59.91	0.09	0.59	15.14	56.46
NaiveBLAST	11.57	57.7	0.12	0.56	11.15	60.45

Table 7.1. Performance of IntFOLD4-TS versus other servers. CAMEO-3D: Common Subset Comparison, 1-year Performance (2016–05-13 to 2017–05-06) (500 targets to 10 methods). IntFOLD4-TS is the reference server. Data are from http://www.cameo3d.org/. The table is sorted by difference in Average lDDT score. Adapted from McGuffin et al. (2018).

7.2.2. ModFOLD7 in IntFOLD5

For version 5 of the IntFOLD server, the algorithms for both 3D model selection and ASE scoring have been upgraded via the integration of our new ModFOLD7_rank method.

7.2.2.1. Methods

For CASP13, the newly upgraded IntFOLD5-TS was developed and prepared to be working via iterative multi-template-based modelling (Buenavista et al., 2012) using the target-template alignments from the 14 alternative methods, SP3 (Zhou and Zhou, 2005), SPARKS2 (Zhou and Zhou, 2005), HHsearch (Söding, 2005), COMA (Margelevicius and Venclovas, 2010), SPARKSX (Yang et al., 2011), CNFsearch (Ma et al., 2013), and the eight alternative threading methods that are integrated into the current LOMETS package (Wu and Zhang, 2007), identical to the IntFOLD4 server first stage. The ASE scoring via ModFOLD7_rank method (rather than ModFOLD6_rank) then was used to select the multiple target-template alignments for 3D modelling with the aim of minimising local errors in final generated models. In addition, the HHpred method (Meier and Söding, 2015) and the template free method I-TASSER light (Roy et al., 2010) (for sequence <500 residues; run in ‘light mode’ with wall-time restricted to 5h) were utilised to contribute models for

ranking. All the final decoy models then were then pooled, scored and ranked using the ModFOLD7_rank method. The outputted data then presented to the user in descending order showing the global model quality. The produced PDB formatted model files then incorporated the ASE scores in the temperature factor column. By integrating ASE scores directly into the PDB formatted model files, users are able to view the local model quality as a temperature gradient that can be mapped onto their 3D models conveniently using their favourite molecular viewing software, such as PyMOL (http://www.pymol.org/).

The main factor behind the improvement in the IntFOLD5 prediction accuracy lies in the latest update to the QA method, ModFOLD7_rank, which combines the strengths of multiple pure-single and quasi-single model methods together - the same successful approach that led to the success of ModFOLD6 (Elofsson et al., 2018) (McGuffin et al., 2018) (Maghrabi and McGuffin, 2017). The major emphasis for the IntFOLD5 server was the noticeable increase in the performance of per-reside accuracy prediction for our own models, as well as improving our model ranking and score consistency for our models. Each IntFOLD5 model was considered individually using 6 pure-single model methods:CDA (Maghrabi and McGuffin, 2017), SSA (Maghrabi and McGuffin, 2017), ProQ2 (Uziela and Wallner, 2016), ProQ2D (Uziela et al., 2017), ProQ3D (Uziela et al., 2017) and VoroMQA (Olechnovič and Venclovas, 2017); and four alternative quasi-single model methods: DBA (Maghrabi and McGuffin, 2017), MF5s (Maghrabi and McGuffin, 2017), MFcQs (Maghrabi and McGuffin, 2017) and ResQ (Yang et al., 2016). Neural networks were then used for combining the component per-residue/local quality scores from each of the 10 alternative scoring methods. The combination resulted in a final consensus of per-residue quality scores for each model. To produce the global score outputs, several variants which combined the mean global scores from the different methods were made, and each were optimised for different aspects of the quality estimation problem. Obtaining the most accurate selection of top models produced by IntFOLD5 was the main objective. Therefore, the integration of the ModFOLD7_rank variant was applied in order to support optimisation for ranking.

Additionally, several new user interface upgrades were implemented for IntFOLD in parallel with the performance enhancement. The upgrades included a streamlined submission form, recalibrated *P*-values for confidence scoring of model quality estimates, the ability to download compressed archives of all annotated models, and the ability to interact with models and then further refine them with a few clicks via simple push buttons.

7.2.2.2. Results

By analysing the results produced from CASP9 until CASP13, it can be noticed that the performance for the major versions of the IntFOLD component methods in each of the relevant categories has remained competitive throughout (McGuffin et al., 2018) (McGuffin and Roche, 2011). The recent results showed that the component methods of IntFOLD have ranked among the top independent servers in the TS prediction category, the EMA category (Kryshtafovych et al., 2018), and historical categories of intrinsic disorder structure and function prediction (Noivirt-Brik et al., 2009) (Schmidt et al., 2009). A significant boost in performance was obtained over DISOPRED (McGuffin, 2008) by designing the DISOclust component method and integrating its latest version with the IntFOLD server. The IntFOLD5 server and its components have also been benchmarked continuously using the CAMEO resource (Haas et al., 2018), which has demonstrated high performance in each respective category (see results from the 3D, QE and LB categories at https://www.cameo3d.org/). The FunFOLD component has also been benchmarked during the most recent CAFA experiment (https://www.biofunctionprediction.org/cafa/, paper in preparation)

Principally, the CAMEO project focuses on the continuous evaluation of the TS predictions from publicly available servers. The TS predictions of the IntFOLD versions have shown a consistent ranking among the top few public servers according to lDDT_BS scores and lDDT scores. From a 3-month data for all targets represented in Table 7.2, it can be seen that IntFOLD5-TS ranked as the top publicly available method. Also, another evaluation based on pairwise comparisons using a common subset of targets over the last year (Appendix 12 and 13) showed that IntFOLD5-TS ranks as the second-best 3D server according to the lDDT scores.

	Average lDDT		Average lDDT-BS	
Server name	**All targets**	**Modelled targets**	**All targets**	**Modelled targets**
IntFOLD5-TS	68.04	68.04	70.94	70.94
RaptorX	67.38	67.38	68.45	68.45
Robetta	65.51	69.1	63.24	66.11
HHpredB	64.06	64.06	68.59	68.59
SWISS-MODEL	62.22	62.97	64.85	65.56
IntFOLD4-TS	55.02	68.1	58.12	73.25
SPARKS-X	54.63	60.7	58.07	66.78
M4T-SMOTIF-TF	54.45	60.77	62.92	65.78
IntFOLD3-TS	53.75	66.85	55.76	69.33
PRIMO	51.74	57.48	58.32	64.65
PRIMO_BST_CL	51.71	57.45	58.32	64.65
NaiveBLAST	50.34	55.69	60.08	62.11
PRIMO_BST_3D	49.83	55.86	57.99	63.51
PRIMO_HHS_3D	48.27	55.87	56.49	62.62
PRIMO_HHS_CL	46.73	56.43	55.55	61.58
Princeton_TEMPLATE	24.46	54.61	25.63	58.95
Phyre2	24.06	52.77	29.27	67.31

Table 7.2. Independent benchmarking of tertiary structure predictions with CAMEO 3D data. Performance results for 3 months of data (26 October 2018 to 19 January 2019) are shown for *all* 250 targets and *all* 17 public methods. Data are sorted by average lDDT score for all targets. The scores for the IntFOLD-TS methods are indicated in bold. Data are taken from the CAMEO 3D front page http://www.cameo3d.org/ on 19 January 2019. Adapted from McGuffin et al. (2018).

From the represented results above, we can see that IntFOLD5-TS has been improved in 3D protein modelling performance. The recent progress in the server has given it the strength to significantly perform better than all the competitive servers in the predicting field. The IntFOLD5-TS was also evaluated with its previous versions and was verified to be an improvement over both IntFOLD3-TS and IntFOLD4-TS (Table 7.3).

	Avg. lDDT		Avg. CAD-score		Avg. lDDT-BS	
Server Name	**Dif.**	**Ref.**	**Dif.**	**Ref.**	**Dif.**	**Ref.**
IntFOLD5-TS	0	67.72	0	0.67	0	71.86
IntFOLD4-TS	0.53	67.18	0	0.66	0.23	71.62
IntFOLD3-TS	2.11	65.61	0.02	0.65	1.9	69.96

Table 7.3. Independent benchmarking of IntFOLD versions with CAMEO 3D data. The data shows the sequential improvement in server performance since the last webserver paper describing IntFOLD3. Performance results for 1 year of data (26 January 2018 to 19 January 2019) are shown for a common subset of 581 targets. The reference method is IntFOLD5-TS, and the table is sorted by average lDDT. Data are downloaded from http://www.cameo3d.org/ **.** Adapted from McGuffin et al. (2018).

7.3. WeFold

A large number of web-based efforts were initiated to promote collaboration within and outside the community of CASP, so that it will be an attraction for researchers from other fields to contribute new ideas to it, one of these efforts was called WeFold. It is a coopetition (cooperation and competition) organisation found in 2012, and its goal is to build a mixed pipeline by recruiting members who have already been participating in CASP as individual teams. By attracting predictors to WeFold they can share components of their methods with other teams in one hybrid pipeline to actively contribute to this project. The organisation asserts that the scale and diversity of integrative prediction pipelines could not have been achieved by any individual lab or even by any collaboration among a few partners. All the models contributed and generated through the created pipelines from all the participating groups are publicly available at the WeFold website. Such a collaboration has created a huge amount of information and a wealth of data that remains to be tapped.

7.3.1. Methods

The main objective for WeFold is to provide a flexible infrastructure for prediction experts to be able to create hybrid pipelines, which may have different approaches of their model quality assessment, refinement and other components of methods (e.g. Figure 7.3 shows the pipelines that start with Rosetta decoys). After creating these hybrid pipelines, they participate in CASP as groups to allow the overall performance to be submitted under an objective and coherent manner of evaluation along with all the other CASP participants. Thereby, the developed methods can be

applied to a variety of inputs sources and the utility of its outcome can be tested within a variety of pipelines. Furthermore, the project aims to document the entire information flow through this infrastructure, to have a result of a data source for the development of methods, which tackle sub-problems of proteins structure prediction. The WeFold community is still growing, and its infrastructure needs a community of users in order to accomplish its goals. Up until now, the organisation pursued an inclusive approach, which brought different protein modelling groups who have already participated in CASP. This approach makes WeFold inclusive and allows predictors to reach out to raise awareness and excitement, beyond the CASP community. The WeFold project also aims to act as an incubator for new ideas. In fact, a number of non-CASP groups have been recruited to the WeFold efforts and have been co-authors of our manuscripts. Some other members have been working on more blue-sky innovative methods for the upcoming CASP exercises (Mirzaei et al., 2016) (Corcoran et al., 2018).

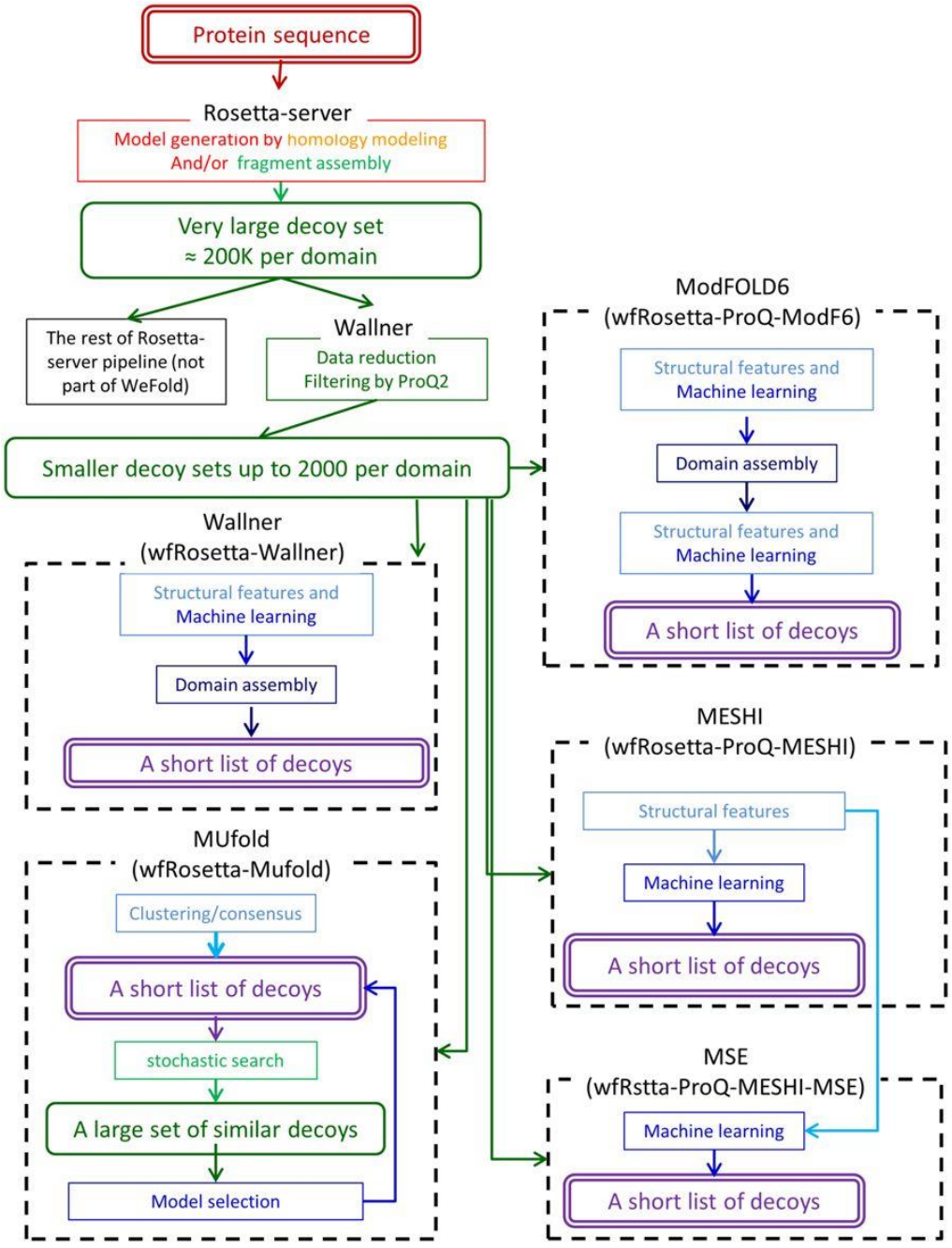

Figure 7.3. An illustration of the WeFold pipeline concept. The figure presents a schematic depiction of 5 of the WeFold3 pipelines, which share their first components and differ in the final stages. Rounded rectangles represent information and plain rectangles represent basic tasks, each of which is an open computational problem. A prediction process starts with a protein sequence, passes at least once through a set of decoys (structural models of proteins), and ends with a short list, ideally one, of high score decoys. Adapted from Keasar et al., (2018).

The protein structure prediction category in CASP was the main focus for the WeFold project. 12 different pipelines were recruited to take part in the 3rd iteration of WeFold protein structure prediction project (Figure 7.3). The groups participated with their main components which included: three major model (or decoy) generators (Rosetta, UNRES, and the CASP12 servers) (Rojas et al., 2008) (Song et al., 2013) (Bradley et al., 2005), two contact prediction methods (GREMLIN and Floudas) (Kamisetty et al., 2013), one secondary structure prediction method (conSSert) (Kieslich et al., 2016), one clustering algorithm (Murtagh, 1985), three refinement methods (Princeton_TIGRESS, GalaxyRefine, and 3Drefine) (Khoury et al., 2014) (Lee et al., 2016) (Bhattacharya and Cheng, 2013), and seven QA/selection methods (APOLLO, MESHI-score, MESHI-MSC, ModFOLD6, MUFold, ProQ2, and Seder (Mirzaei et al., 2016) (Wang et al., 2011) (McGuffin, 2008) (Ray et al., 2012) (Faraggi and Kloczkowski, 2014) (Maghrabi and McGuffin, 2017) (Zhang et al., 2011). The WeFold organisers then decided to compare QA/scoring methods fairly by applying them to the same decoys sets. Thus, *wfRosetta-MUfold*, *wfRosetta-ProQ-MESHI*, *wfRosetta-ProQ-MESHI-MSC*, *wfRosetta-ProQ-ModF6*, *wfDB_BW_SVGroup*, and *wfRosetta-Wallner* started with the same set of Rosetta decoys and *wfMESHI-Seok* and *wfMESHI-TIGRESS* started with the same subsets of server decoys selected by MESHI. Moreover, *wfRosetta-ProQ-MESHI* and *wfRosetta-ProQ-MESHI-MSC* also used the same set of decoys and features to strictly compare two scoring functions (Mirzaei et al., 2016). With regards to decoys reduction needed to reduce the large set of Rosetta decoys to a manageable size for refinement and QA, the organisers replaced the filtering and clustering procedure that we had used in WeFold2 for the Foldit decoys, by ProQ2.

7.3.2. Results

Several WeFold3 pipelines of methods were developed, built and prepared for the participation in the CASP12 experiment. Each pipeline method was then independently benchmarked against the top protein predictors in CASP12. Several pipelines were competitive in the different target categories, giving an impressive scoring compared to most of the participating groups.

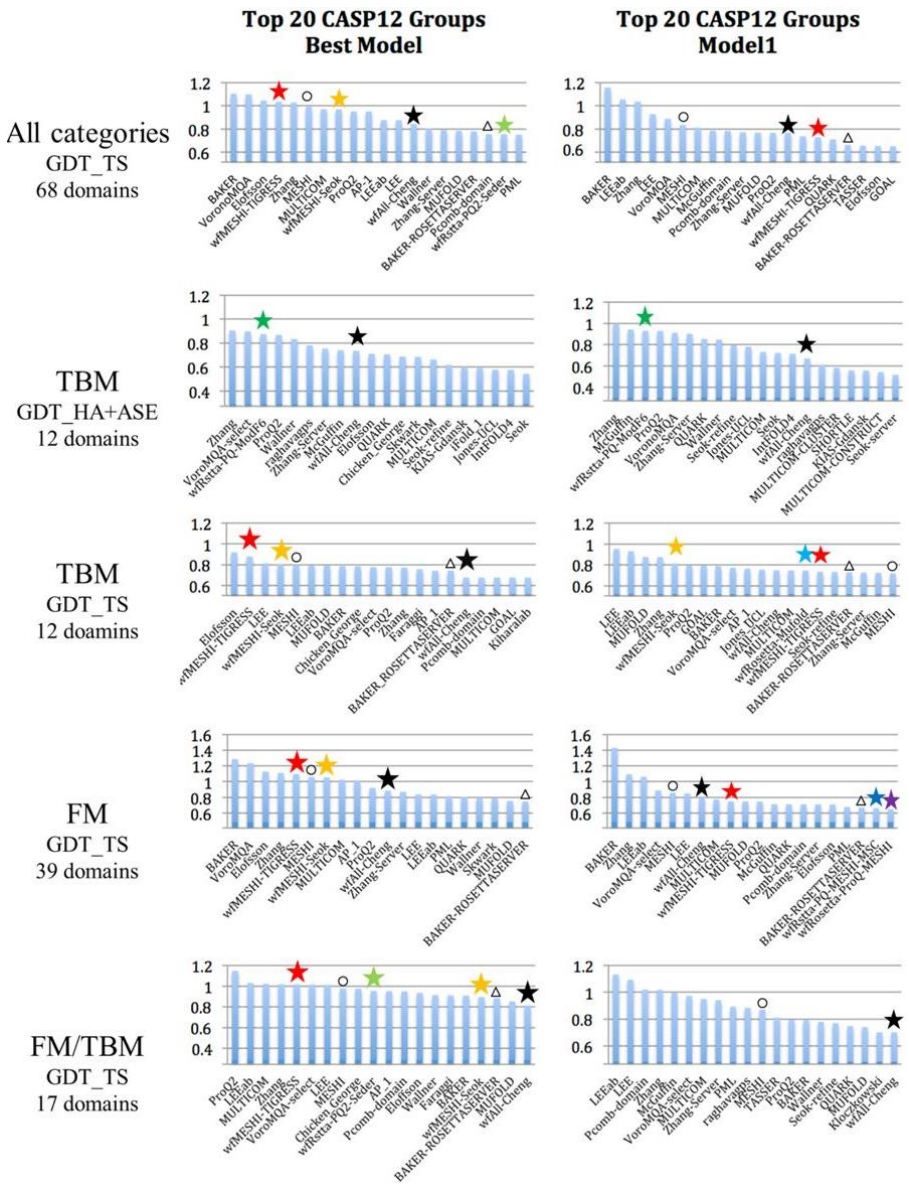

Figure 7.4. Average z-scores (>−2.0) of the 20 top CASP12 groups. WeFold pipelines are marked with asterisks (Black = wfAll-Cheng; Red = wfMESHI-TIGRESS; Orange = wfMESHI-Seok; Light green = wfRstta-PQ2-seder; Dark green = wfRstta-PQ-ModF6; Light blue = wfRosetta-MUFOLD; Dark blue = wfRstta-PQ-MESHI-MSC; Purple = wfRosetta-PQ-MESHI). The results of MESHI and BAKER-ROSETTASERVER are marked by black circle and triangle respectively. Only those groups that submitted models for at least half of the targets are considered. Chart on the left shows top 20 groups/servers when considering the best model submitted by each group for each target. Chart on the right shows top 20 groups/servers when considering Model 1 only. CASP assessors used GDT-HA + ASE only for TBM targets hence the double depicting of that category. Source: http://www.predictioncenter.org/casp12/zscores_final.cgi. Adapted from Keasar et al., (2018).

Following a benchmarking analysis, the results showed significant scores in 4 groups of WeFold among the top 20 CASP12 groups/servers, as shown in the top panel of Figure 7.4. The comparison was carried out according to average GDT-TS z-scores > -2 when considering all 3 categories (template-based modelling, template-based modelling/free modelling, and free modelling), and only those groups that submitted models for at least half of the targets. The chart on the left-hand side shows the top 20 groups/servers when considering the best model submitted by each group for each target and the chart on the right-hand side shows top 20 groups/servers when considering Model 1 only.

Many WeFOLD3 pipelines gained a better performance than the original pipelines that they were built from. Such these pipelines are those which were built based on MESHI selection, *wfMESHI-TIGRESS* and *wfMESHI-Seok*. These methods benefited from the top performance of the MESHI group and one of them (*wfMESHI-TIGRESS*) slightly outperformed MESHI when considering the best model submitted by each group. Another group called *wfAll-Cheng*, which used all the models shared by all the WeFold3 groups but usually selected models from the MESHI-based groups (as shown in Appendix 14) ranked 13th in both cases, when considering the best model and model 1 only. This method showed a significant improvement with respect to its own performance in CASP11 when it ranked 47th. Of the Rosetta-based teams, none ranked among the top 20 when considering the best model submitted. Finally, group *wfRstta-PQ2-Seder*, which uses a mix of Rosetta and server models, also ranked among the top 20. In the next sections, we analyze the performance of the WeFold3 pipelines in the 3 subcategories TBM, TBM/FM, and FM.

In the TBM category, the target proteins are those for which a relationship could be detected by distant sequence similarity searches providing one or more-fold templates. Panel 2 and 3 in Figure 7.4 shows the top 20 ranking CASP12 groups/servers when considering the average z-scores of both the assessors formula and GDT-TS, respectively. The CASP12 assessors used *GDT-HA + ASE* for the assessment of models in this category. ASE is defined as $ASE = 100.0 * (1 - Mean(|S(tf_i|d_0) - S((d_i|d_0)|)$ where tf_i is the temperature factor of *i-th* residue in the model, and d_i is the distance between *i-th* residues in lga alignment (sequence dependent mode) $S(x) = 1/(1 + x^2)$, and d_0 is the scaling factor, set $d_0 = 5.0$ (http://www.predictioncenter.org/casp12/doc/help.html#ASE).

The results from the above charts illustrate that focusing on either GDT-TS or ASE produced different results. In fact, when considering the assessors formula, two WeFold pipelines ranked

among the top 20: *wfRosetta-ProQ-ModF6* and *wfAll-Cheng*. It can be noticed that *wfRosetta-ProQ-ModF6* selected best 5 models among the models generated by the BAKER-ROSETTASERVER and neither the BAKER-ROSETTASERVER nor the BAKER group are among the top 20 in this category. The high performance of the *wfRosetta-ProQ-ModF6* group was mainly due to accurate ranking and ASE using the ModFOLD6_rank method (Maghrabi and McGuffin, 2017). On the other hand, when using GDT-TS values, the two MESHI-based groups and *wfAll-Cheng* ranked among the top 20 when considering both the best model among the 5 submitted and model 1. *wfMESHI-Seok* showed better results in TBM category than in other categories probably because the refinement method was originally trained to improve template-based models.

7.4. Modelling Connexin62 to understand the haemostasis mechanism in platelets

In numerous mammalian cells, 1536 proteins have been found to be expressed in their plasma membrane, 297 of them are oligomerising into hexameric hemichannels making what we call it "cellular gap junctions" (Giepmans and van IJzendoorn, 2009). A large family of these types of proteins is called Connexins. Connexin proteins (a.k.a. Cx) are constructed as hexameric hemichannels on adjacent cells dock together to form gap junctions (GJs). These gap junctions facilitate the direct trafficking of molecules whose size are approximately less than 1 kDa. This trafficking occurs between cells, so the molecules would travel from one cell to another through these Connexins, they also serve in allowing coordinated responses between cells in tissues. For example, the human platelets have the C37 and Cx40 from the connexin family. They are expressed in human platelets for the function of selective inhibition. Several studies have reported that the presence of connexins in platelets is essential for the formation of gap junctions within platelet thrombi as they are required for the control of clot retraction. The same species are having another function which is the selective deletion in transgenic mice attenuates platelet (Vaiyapuri et al., 2012).

In this study, the expression of an orphan connexin termed Cx62 in human and mouse platelets (Cx57, mouse homologue) was identified by our collaborators by using two techniques, the Western Blot and the Immunocytochemistry. A mimetic peptide, called ^{62}Gap27, was developed that targets the second extracellular loop of Cx62 to reduce the hemichannel permeability and GJ-mediated intercellular communication. In the study, we applied an *in silico* 3D modelling of the

hemichannel and prediction of the ^{62}Gap27 peptide interaction site. The 3D models corroborate the experimental observations and suggest a structural explanation for the observed changes in permeability caused by the mimetic peptide. Several features of agonist-induced platelet activation, including aggregation, degranulation, fibrinogen binding to integrin $\alpha_{IIb}\beta_3$, Ca^{2+} mobilisation and integrin $\alpha_{IIb}\beta_3$ outside-in signalling (which controls clot retraction and spreading), were inhibited by ^{62}Gap27 compared to scrambled peptide control. Thrombus formation (*in vitro* and *in vivo*) and tail bleeding were also significantly inhibited by ^{62}Gap27 when it was injected via femoral vein 5 minutes before 1mm of tail tip was removed using a scalpel blade, and the tail tip was placed in sterile saline at 37°C. The time to cessation of bleeding was measured up to 10 min. Anti-platelet and anti-thrombotic activity of the ^{62}Gap27 peptide was found to be associated with reduced platelet signalling events, including tyrosine phosphorylation of key platelet signalling components and inhibition of PKC activity. Analysis of VASP phosphorylation identified that treatment of ^{62}Gap27 was found to increased PKA activation in both resting and activated platelets in a cAMP-independent manner. This study identifies Cx62 and Cx57 are expressed in human and mouse platelets, respectively, where they play a fundamental role in platelet function, thrombus formation.

7.4.1. Methods

Obtaining the complete sequence of Cx62 was through the online server GenBank (Sayers et al., 2019), and ProPram (Wilkins et al., 1999) was utilised for the physio-chemical analysis. State-of-art structure prediction tools were employed due to the absence of the experimental structures. From the IntFOLD server (McGuffin et al., 2015), IntFOLD4-TS method (McGuffin et al., 2018) was utilised for the tertiary structure models prediction of the Cx62 protomer (monomeric subunit).

In addition, the new conformations of the full-length Cx62 protein model were evaluated in terms of its stereochemical quality assessment using ModFOLD6 (Maghrabi and McGuffin 2017). The evaluation and validation were performed firstly after designing the first model, and then before and after the assembling and refinement steps to check the quality of its structure giving the local and global scores. The designed Cx62 hemichannel was superposed with the reference to test the construction accuracy using TM-align algorithm (Zhang and Skolnick 2005). The docked hemichannels model output from PISA was edited by rotating the hemichannels against each other in order to correct their positions so that it matches the same design of the 12-mer structure of other studies (Maeda et al., 2009) (Nakagawa et al. 2010).

Due to the lack of an existing Cx62 targeting peptide, a mimetic peptide Gap27 was designed so that it can target the second external loop of Cx62 specifically. Moreover, the second external loop for most of the hemichannels has been commonly targeted while designing Gap27 mimetic peptide. Therefore, a multiple sequence alignment of human connexin sequences was performed using ClustalW (Larkin et al. 2007) in order to prevent any cross-reactivity of the designed mimetic peptide with other Cx molecules. Post multiple sequence alignments, a selective ^{62}Gap27 was designed with the feature of preventing the possibility in targeting any connexin member except Cx62. To exclude any false positive data, a negative control (scrambled peptide) was designed using a web-server tool based on the Mimotopes method (Geysen et al., 1986). Blast (Altschul et al., 1990) test was also performed to ensure that the designed ^{62}Gap27 is not present in proteins other than Cx62. The structure of the inhibitor ^{62}Gap27 was predicted using PEP-FOLD3 (Lamiable et al., 2016).

Subsequently, protein-ligand docking was performed to predict the most likely interactions that could occur between Cx62 and the ^{62}Gap27 inhibitor, this step was carried out using the SwissDock server (Grosdidier et al., 2011). The FullFitness and Gibbs free energy (ΔG) score of each run of the docking were evaluated and the final ranking of each cluster was based on the FullFitness scores.

The Cx62 hemichannels (2x 6-mers) were successfully modelled using the PDB entry 2zw3 as a template. The docked hemichannel assembly (12-mer) template for PDB ID 2zw3 was downloaded from PISA (Krissinel and Henrick, 2007) service at the EBI (http://www.ebi.ac.uk/pdbe/prot_int/pistart.html). For each hemichannel, the template was used to orientate six of the modelled protomers by a six-fold symmetry axis perpendicular to the membrane plane and build the complete model of the docked hemichannel (12-mer) complex. Residues in the modelled protein-protein and protein-ligand complexes were considered to be interacting if the distance between the closest heavy atoms (i.e. non-hydrogen) in the residues belonging to different chains was <= 5Å.

7.4.2. Results

The Cx62 sequence has 543 amino acid residues, with a calculated molecular mass of 62 kDa and a pI of 7.89. The instability index of Cx62 is computed to be 60.31 (ProtParam), which classifies

it as an unstable protein overall. This is due to a long C-terminal disordered region revealed by the structure prediction results presented below.

The 3D model of Cx62 predicted using the IntFOLD server reveals a protomer (monomer subunit) consisting of four transmembrane helices, two extracellular loops, a small bended N-terminal helix, cytoplasmic and C-terminus loops, forming a typical four-helical bundle in which any pair of adjacent helices are antiparallel (Figure 7.5a and 7.5b). The ModFOLD6 global 3D model quality score for the full-length protein was calculated as 0.43 ($p < 0.01$; less than a 1 in 100 chance of an incorrect model). The ModFOLD6 quality score increases to 0.57($p < 0.001$; less than a 1 in 1000 chance of an incorrect model) when the long-disordered C-terminal loop is excluded. The calculated local (or per-residue) errors from ModFOLD6 were mapped onto the model using the temperature colouring scheme ranging from blue (indicating residues modelled with high quality) to red (indicating residues with lower model quality, which are often more flexible or disordered) (Figure 7.5a). Such results indicate that the ordered regions of the Cx62 structure were generally modelled with high confidence.

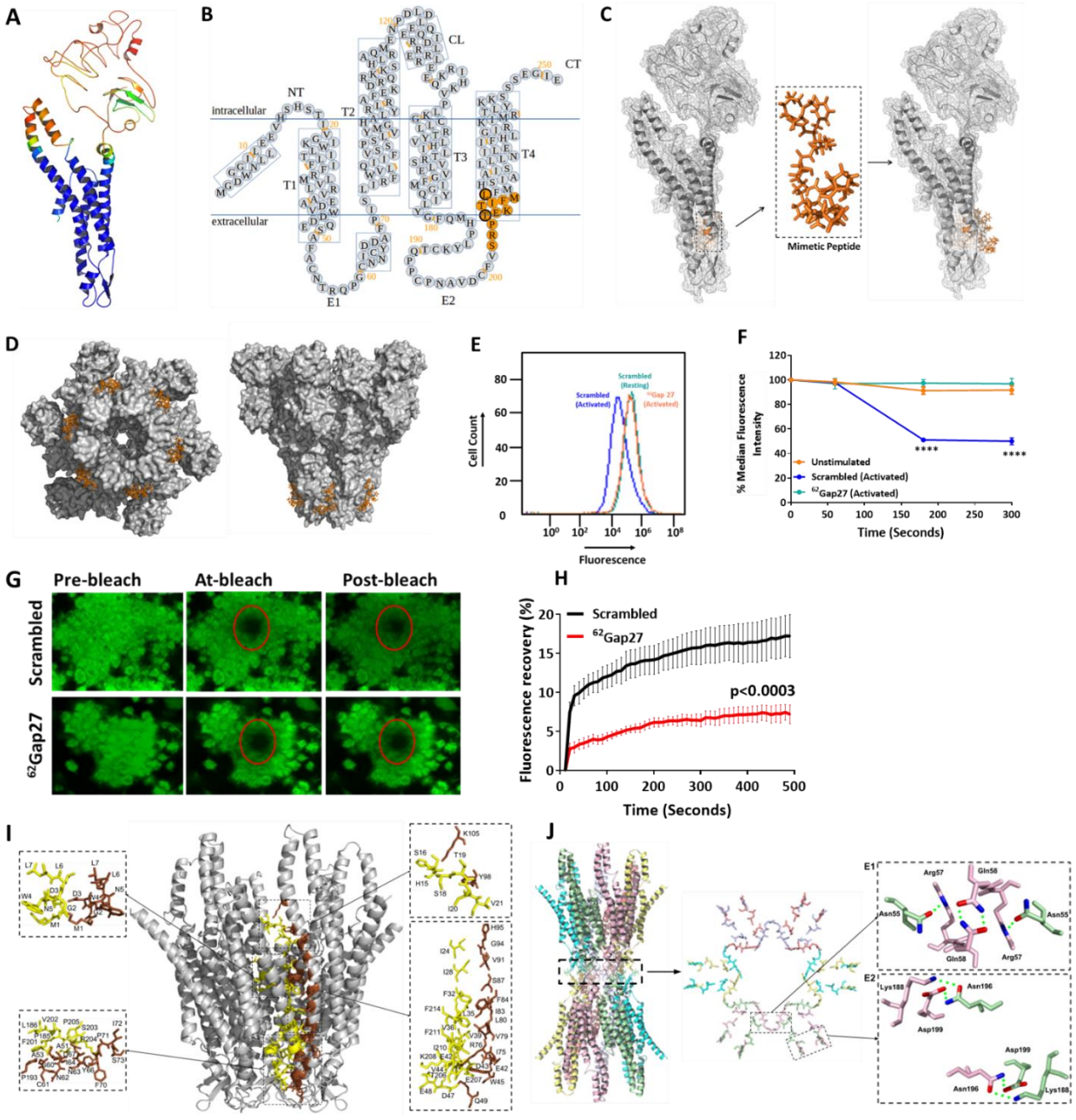
A
B
C
NT
CL
CT
intracellular
extracellular
T1
T2
T3
T4
E1
E2
Mimetic Peptide
D
E
Scrambled (Resting)
Scrambled (Activated)
62Gap 27 (Activated)
Cell Count
Fluorescence
F
% Median Fluorescence Intensity
Unstimulated
Scrambled (Activated)
62Gap27 (Activated)
****
Time (Seconds)
G
Pre-bleach
At-bleach
Post-bleach
Scrambled
62Gap27
H
Scrambled
62Gap27
p<0.0003
Fluorescence recovery (%)
Time (Seconds)
I
J
E1
E2
Arg57
Gln58
Asn55
Asp199
Lys188
Asn196

Figure 7.5. Design of the 62Gap27 mimetic peptide and its role in the regulation of intercellular communication. (A) Predicted 3D model of the Cx62 tertiary structure. The cartoon view of the structure is coloured using the temperature colouring scheme, where blue indicates ordered regions with low ModFOLD predicted per-residue errors and red indicates high per-residue errors and more flexibility. (B) Schematic representation of the designed 62Gap27 binding site on Cx62. Topological diagram of the Cx62 protomer, the predicted binding site (BS) is highlighted in orange. (C) Structural representation of the target region where the 62Gap27 mimetic peptide was designed, and the putative binding site of the inhibitor on Cx62. (D) Surface representation of Cx62 hemichannels being targeted by 62Gap27 showing the pore cross section and side views respectively. (E) The efflux of calcien was calculated using flow cytometric analysis. Calcein loaded platelets incubated with 62Gap27 or scrambled peptide (100 µg/ml) were stimulated with thrombin (0.1 U/ml). Histograms of calcein fluorescence for unstimulated (green), and thrombin-stimulated platelets in the presence of scrambled (blue) or 62Gap27 (100µg/ml) (orange) (n=4). (F) Calcein efflux following thrombin stimulation for varying time periods was measured by the rate of fluorescence reduction in platelets. Median fluorescence intensity for unstimulated and stimulated samples treated with scrambled or 62Gap27 was analysed (n=4). (G) Calcein loaded platelets were treated with scrambled or 62Gap27 (100µg/ml) for 5 minutes prior to their stimulation on fibrinogen and collagen-coated coverslips and FRAP analysis was performed. Representative images represent fluorescence recovery (Pre-bleach, At-bleach and Post-bleach) in samples treated with scrambled or 62Gap27. (H) Quantified data shows mean fluorescence recovery intensity of scrambled and 62Gap27 treated samples and normalised to the level of fluorescence at bleach point (shown in red circle) (n=5). (I) Inter-protomer interactions. The hemichannel formed by six protomers of Cx62 is shown in grey cartoon view, the sidechains in the zoomed views are shown as sticks with brown and yellow colours to differentiate between the residues of interacting protomer pairs. (J) Modelled intercellular interactions between docked hemichannels. In the left-hand panel, a Cx62 gap junction channel is shown. The region enclosed by dashed lines is sectioned perpendicular to the pore axis and is viewed from the pore axis (right-hand panel). The interactions between the 2 docked hemichannels (the first external loop (E1) and the second external loop (E2) regions) are depicted in the close-up images. In region E1, Gln58 forms symetrical hydrogen bonds with the same residue from the opposite protomer while Asn55 forms a hydrogen bond with Arg57 in the opposite protomer. In region E2, Asn196 and Asp199 form hydrogen bonds with the same residues on the opposite protomer. Data represent Mean ± SEM, ****$P<0.0001$ was calculated by two-way ANOVA.

The tertiary structure models of Cx62 were subsequently used as targets for *in silico* docking of the designed mimetic peptide and for quaternary structure assembly of the docked hemichannel complex (Figure 7.5D, 7.5I and 7.5J). To confirm the molecular interactions that occur between Cx62 and ^{62}Gap27 inhibitor, single ligand docking prediction was performed using SwissDock.

The protein-ligand docking results from SwissDock revealed a number of alternative sites for the ^{62}Gap27 inhibitor binding to Cx62. The output showed the most favourable binding locations for the ^{62}Gap27 inhibitor based on FullFitness score and cluster formation. Six clusters contained ligand poses in approximately the same location at the end of the second external loop (Figure 7.5B and 7.5C), corroborating the experimental results. For the most favourable interaction, the docking results gave a FullFitness score of -3210.54 kcal/mol and an estimated Gibbs free energy (ΔG) of -6.27 kcal/mol.

While Gap27 peptides are used widely to explore connexin function, the exact mode of action was not clearly understood. It is believed that they induce a conformational change in hemichannel (or gap junctions) and also modulate the docking of two complementary hemichannels to form a gap junction, thereby regulating permeability of the pore (Leybaert et al., 2003; Vaiyapuri et al., 2015; Vaiyapuri et al., 2012). Professor Jon Gibbin's team performed flow cytometry to investigate this in calcein-loaded platelets where efflux of calcein (anionic 0.62 kDa fluorescent dye) from the platelet cytosol was measured (Figure 7.5E, 7.5F). Upon stimulation with thrombin, calcein associated fluorescence decreased in scrambled peptide treated cells by ~50%, indicating a release of dye. The treatment of platelets with ^{62}Gap27 prevented this loss of fluorescence. This indicates a role for Cx62 hemichannels in regulating platelet permeability and/or platelet activation.

7.5. Conclusion

In this chapter, we have presented the applications of ModFOLD6 and ModFOLD7 during the project of this Ph.D. study. IntFOLD was the method which gained the largest benefits from this application. It was seen that after integrating ModFOLD6 and ModFOLD7 to our latest versions of IntFOLD, the predicting methods has been noticeably improved. When ModFOLD6 was applied in IntFOLD4, the method showed a significant improvement over the previous versions as well as its other competitive predicting methods. With IntFOLD5, the integration of ModFOLD7 has

strengthen the method making it more maintained in terms of competitiveness, and more confidence in terms of model scores and ranking.

Another application of our EMA methods was for the success in the participation with WeFold3. Official scoring exams have showed that WeFold3 pipelines were ranked among the top 10 predicting methods in the TBM category. They performed well with most of the targets, and the credits went to the quality assessment part of the methods led by ModFOLD6.

Finally, we saw that ModFOLD also helped in the huge project of Connexin62. The method with its integration to IntFOLD has managed to reveal the closest-to-native structure of the Cx62 despite that the protein structure was challenging since having too many disordered regions in it. Our methods have also succeeded in predicting the structure of the ligand ^{62}Gap27 which was used for further studies in the project showing some interesting results.

Such applications in the latest versions of IntFOLD4 and IntFOLD5, the participation in WeFold version 3, and the confirmation of the novel orphan Cx62 has also given us the opportunity to study our method more practically, knowing by that the strong and confident sides as well as the weaknesses in order find a better way of improving it. ModFOLD has also got its popularity through such applications, and therefore, became more famous for use internationally.

Chapter 8

Synthesis, conclusion and next direction

8.1. Synopsis of studies

8.1.1. ModFOLD6 optimisation and the participation in CAMEO and CASP12

This study focused on the importance of having methods that reliably estimate the likely similarity between the predicted and native structures of proteins for driving the acceptance and adoption of 3D protein models by life science community. The main aim of the initial study was to gain an improvement in the performance of the currently available model quality assessment method, ModFOLD6. It was the latest version of the leading resource for Estimates of Model Accuracy.

The ModFOLD6 method was benchmarked in the first step of the study with a number of the top ranked model quality assessments programs, which were selected and revised to reflect the actual performance of ModFOLD6. For the initial part of the study, a correlation analysis was carried out between predicted quality scores from the selected methods, including ModFOLD6, and standard observed scores from four measuring methods, and the correlation was carried out using three correlation coefficients.

Further, ModFOLD6 was optimised to use a pioneering hybrid quasi-single model approach. The server was designed to be able to integrate scores from three pure-single model methods and three quasi-single model methods using a neural network for the estimation of local quality scores. Moreover, the ModFOLD6 server interface was designed to provide three options for producing global score estimates, depending on the requirements of the users: (i) ModFOLD6_rank, which is optimised for ranking/selection, (ii) ModFOLD6_cor, which is optimised for correlations of predicted and observed scores and (iii) ModFOLD6 global for balanced performance.

When the ModFOLD6 method was optimised and was ready for testing, the server was registered for participation in the biggest independent blind testing experiment, CASP. It was the 12th season of CASP (in 2016) at the time of ModFOLD6 development. The ModFOLD6 variants were tested and ranked among the top few for EMA methods. The CASP12 experiment showed us some very interesting results, which were then used for the next parts of this study.

The ModFOLD6 server was also continuously automatically evaluated as part of a large-scale protein structure prediction project called CAMEO. The results showed significant improvements after optimisation, and performance gains were observed compared to the other EMA methods as well as our previous versions of ModFOLD. The ModFOLD6 server is freely available at: https://www.reading.ac.uk/bioinf/ModFOLD/ModFOLD6_form.html.

8.1.2. RSNNS and TensorFlow DANNs

For the third chapter of this thesis, two different deep neural network tools were trialled (each tool at a time) in order to combine the selected EMA methods for the purpose of gaining further improvements in the ModFOLD6 method. Both NN techniques used the MLPs class of feedforwarding artificial neural networks, and they differed in terms of their complexity.

Previously, the selected methods including ModFOLD6 were evaluated individually as well as in combinations using simple mean scores and multiple linear regression. Subsequently, the two different NN packages, RSNNS and TensorFlow were deployed in separate pipelines in order to explore the efficiency that could occur when integrating these different tools.

The results showed that integrating the simple MLPs effected our method when it was tested for correlation between the predicted score and the true score. ModFOLD6 with the integration of the RSNNS neural networks outperformed the other top selected methods.

However, it was noticed that the ModFOLD6 method was not showing any better performance with the deep MLPs in the correlation measurement. Contrarily, when we look at the highest ranked models a measurement, we can see that integrating TensorFlow to our method increases its performance making ModFOLD6 able to outperform most of the other EMA methods.

The study showed the potential in using standard shallow as well as deep artificial neural networks, and the effects of learning techniques in optimising model quality assessment programs. It also offered suggestions as to how the deep learning methodology could be modified in order to improve the NN ability of predictions.

8.1.3. DANNs parameterisation

In the next study, we focused on exploring the use of DANNs to combine multiple quality assessment scores. Numerous different QA programs exist which all use different methods of scoring model quality. Therefore, it was suggested to use DANNs in order to regularise these methods so that we can achieve a better prediction of model quality, and to pick the top ranked model from a group of alternatives.

A DANN was built using the TensorFlow python software library in order to determine the hyperparameters for rank-optimised as well as correlation-optimised networks. The constructed

deep network was built and underwent several modifications. The network then was subjected to alternative parameterisations in an attempt to obtain the most suitable DANNs for our EMA method.

The results show that the DANNs used can improve both the ranking (0.39% improvement) as well as the correlation (0.43% improvement) beyond that of taking the average of networks input scores. However, the improvement is marginal, and future research should focus more on alternative methods of hyperparameter optimisation.

8.1.4. ModFOLD7 upgrade and the participation in CAMEO and CASP13

For the CASP13, we managed to make some improvement to our method by combining further pure and quasi-single model methods. Such a combination has given the method further performance boost and enabled better prediction accuracy. The method was built on the successful strategy that was used in ModFOLD6, but with the additional training to an alternative target function (lDDT) for accuracy self-estimates and scoring.

The upgraded ModFOLD7 then was tested for stability having noticed that this version showed the same strengths and accuracy score in the ranking and correlation assessments, but with higher consistency compared to ModFOLD6. The server also provides the three alternative options based on the users interest whether they are looking for a ranking/selection, correlations or a balanced performance.

After testing and preparation, ModFOLD7 was ready for the participation to the 13th season of the worldwide independent blind testing experiment, CASP. The results in CASP13 showed that our method was ranked among the top few EMA methods according to several of the official benchmarks. The method also showed relatively better performance in accordance with the superposition-free scores, lDDT, and CAD-score.

Another evaluation resource for ModFOLD7 was the CAMEO project, where the method was continuously automatically evaluated, showing a significant improvement compared to the previous versions as well as the other EMA competitors. The ModFOLD7 server is also freely available at: https://www.reading.ac.uk/bioinf/ModFOLD/ModFOLD7_form.html.

8.1.5. ModFOLD6 and ModFOLD7 applications

The first application of our method was the integration of ModFOLD6 with the 4th version of IntFOLD. The ModFOLD6_rank variant was chosen for this integration for the purpose of improving the selection and ASE scoring. As a result, IntFOLD4 became more powerful, significantly outperforming IntFOLD3, and competitive with the best publicly available 3D prediction methods. Details about this work can be found in the following article: https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.25360

ModFOLD6 was also part of the web-based efforts community, WeFold. With diversified integrative prediction pipelines, our method was allocated to be included with the constructed WeFold3 pipelines, with the aim of increasing the model quality estimation of the modelled 3D structure of proteins. The prepared WeFold pipelines were benchmarked against the top performing methods in CASP12. According to several official analyses, the results showed that some WeFold3 pipelines ranked among the top 20 methods. One of these methods was *wfRosetta-ProQ-ModF6* which included ModFOLD6 as the final stage of its pipeline. Details about this work can be found in the following article: https://www.nature.com/articles/s41598-018-26812-8

ModFOLD7 was developed in the latter part of this doctoral study. The method was integrated with the fifth version of IntFOLD prediction method, leading to improved performance over previous versions. IntFOLD5 method participated in CASP13 in 2018, showing such impressive results in most of the formula in the TS category. Details about this work can be found in the following article: https://academic.oup.com/nar/article/47/W1/W408/5482507

Alongside and within the period of processing and improving ModFOLD6 and ModFOLD7, a number of experimental applications were carried out. One of the successful applications our developed methods was the collaboration with the study of identifying the expression of the orphan Connexin62 in the human and mouse platelets. It was a cooperative project which combined a number of *in vivo, in vitro,* and *in silico* methods. The study was completed, and it is currently on the process of submission to the Cell journal to be reviewed.

8.2. Conclusion

Overall, it is clear that the ModFOLD method has been incrementally but significantly improved during the course of this study. The first progressive step occurred when the method was optimised,

and a combination of pure-single and quasi-single scoring methods were added, leading to an improvement of about 2%. Subsequently, DANNs were implemented in attempt to optimise the integration of component methods in the ModFOLD pipelines. Such an implementation showed an incomplete improvement which then led us to work on parameterising these Neural Networks. The second progressive step occurred after DANNs parameterisations and when the lDDT scores were used for training instead of the S-scores, this development led to a further marginal improvement (≈ 1%). Lastly, further improvements (≈ 2%) were gained after combining additional pure-single as well as quasi-single scoring methods, making ModFOLD7 one of the leading EMA methods in the field thus far.

8.3. Future directions

Although, our newly developed ModFOLD has shown some impressive results in different areas of studies, there are still a considerable room for gaining more improvement to our method's pipeline. Here are several ways that can be considered as our future goals to improve ModFOLD method:

a) Further optimisation needs to be carried out for the ranking/selection as well as the correlation scoring methods in order to achieve more accurate measuring techniques, and to try merging the two scoring methods into one single input.
b) More studies in the Deep Artificial Neural Networks are needed for the purpose of achieving an optimal network. These studies include: the type of DANNs to be used as there are different types (e.g. Recurrent neural networks, Convolutional neural networks) of Deep Neural Networks depending on the user's interest; the use of activation functions (such as sigmoid, tanh, RdLu, ReLU6, dropout) which provide forms of non-linearities for nodes in the Neural Networks; the use of optimisers (such as Adam, Adagrad) which calculate and apply gradients to variables, and some utilise the exponentially decaying average of past gradients and past squared gradients (Walia, 2018).
c) A more focus on contact predictions as recent studies showed that contact features have improved the performance of EMA methods. In the last season of CASP, a number of new CDA scores based on the contact prediction measures were reported to provide high impacts in scoring the EMA more accurately showing by that some significant results which has been considered.

d) Integration of further pure single model methods such as the ones which have been powered with more features in their upgraded versions (e.g. ProQ4 (Hurtado et al., 2018)).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X., 2016. TensorFlow: A system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). pp. 265–283.

Abriata, L.A., Tamò, G.E., Monastyrskyy, B., Kryshtafovych, A., Dal Peraro, M., 2018. Assessment of hard target modeling in CASP12 reveals an emerging role of alignment-based contact prediction methods. Proteins 86 Suppl 1, 97–112. https://doi.org/10.1002/prot.25423

Adiyaman, R., McGuffin, L.J., 2019. Methods for the Refinement of Protein Structure 3D Models. International Journal of Molecular Sciences 20, 2301. https://doi.org/10.3390/ijms20092301

Alberts, B., Johnson, A.D., Lewis, J., Morgan, D., Raff, M., Roberts, K., Walter, P., 2014. Molecular Biology of the Cell, Sixth edition. ed. W. W. Norton & Company, New York, NY.

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J., 1990. Basic local alignment search tool. J. Mol. Biol. 215, 403–410. https://doi.org/10.1016/S0022-2836(05)80360-2

Amari, S., Arbib, M.A. (Eds.), 1982. Competition and Cooperation in Neural Nets: Proceedings of the U.S.-Japan Joint Seminar held at Kyoto, Japan February 15–19, 1982, Lecture Notes in Biomathematics. Springer-Verlag, Berlin Heidelberg.

Amir, E.-A.D., Kalisman, N., Keasar, C., 2008. Differentiable, multi-dimensional, knowledge-based energy terms for torsion angle probabilities and propensities: Energy Terms for Protein Torsion Angles. Proteins 72, 62–73. https://doi.org/10.1002/prot.21896

Anfinsen, C.B., 1973. Principles that govern the folding of protein chains. Science 181, 223–230.

Ashraf, G.M., Greig, N.H., Khan, T.A., Hassan, I., Tabrez, S., Shakil, S., Sheikh, I.A., Zaidi, S.K., Akram, M., Jabir, N.R., Firoz, C.K., Naeem, A., Alhazza, I.M., Damanhouri, G.A., Kamal, M.A., 2014. Protein misfolding and aggregation in Alzheimer's disease and type 2 diabetes mellitus. CNS Neurol Disord Drug Targets 13, 1280–1293.

Atkins, J.D., Boateng, S.Y., Sorensen, T., McGuffin, L.J., 2015. Disorder Prediction Methods, Their Applicability to Different Protein Targets and Their Usefulness for Guiding Experimental Studies. Int J Mol Sci 16, 19040–19054. https://doi.org/10.3390/ijms160819040

Ba, J., Caruana, R., 2014. Do Deep Nets Really Need to be Deep?, in: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 27. Curran Associates, Inc., pp. 2654–2662.

Baldwin, R.L., 2007. Energetics of protein folding. J. Mol. Biol. 371, 283–301. https://doi.org/10.1016/j.jmb.2007.05.078

Bassot, C., Hurtado, D.M., Elofsson, A., 2019. Using PconsC4 and PconsFold2 to Predict Protein Structure. Current Protocols in Bioinformatics 66, e75. https://doi.org/10.1002/cpbi.75

Ben-David, M., Noivirt-Brik, O., Paz, A., Prilusky, J., Sussman, J.L., Levy, Y., 2009. Assessment of CASP8 structure predictions for template free targets. Proteins 77 Suppl 9, 50–65. https://doi.org/10.1002/prot.22591

Benkert, P., Tosatto, S.C.E., Schomburg, D., 2008. QMEAN: A comprehensive scoring function for model quality assessment. Proteins 71, 261–277. https://doi.org/10.1002/prot.21715

Benkert, P., Tosatto, S.C.E., Schwede, T., 2009. Global and local model quality estimation at CASP8 using the scoring functions QMEAN and QMEANclust. Proteins 77 Suppl 9, 173–180. https://doi.org/10.1002/prot.22532

Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Wheeler, D.L., 2003. GenBank. Nucleic Acids Res. 31, 23–27. https://doi.org/10.1093/nar/gkg057

Berg, J.M., Tymoczko, J.L., Stryer, L., Berg, J.M., Tymoczko, J.L., Stryer, L., 2002. Biochemistry, 5th ed. W H Freeman.

Bergman, U., Yachandra, V.K., Yano, J., 2017. X-Ray Free Electron Lasers: Applications in Materials, Chemistry and Biology. Royal Society of Chemistry.

Bergmeir, C., Benítez, J.M., 2012. Neural Networks in R Using the Stuttgart Neural Network Simulator: RSNNS. Journal of Statistical Software 46, 1–26. https://doi.org/10.18637/jss.v046.i07

Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, 281–305.

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E., 2000. The Protein Data Bank. Nucleic Acids Res 28, 235–242. https://doi.org/10.1093/nar/28.1.235

Bhattacharya, D., Cheng, J., 2013. 3Drefine: consistent protein structure refinement by optimizing hydrogen bonding network and atomic-level energy minimization. Proteins 81, 119–131. https://doi.org/10.1002/prot.24167

Bindschedler, L.V., McGuffin, L.J., Burgis, T.A., Spanu, P.D., Cramer, R., 2011. Proteogenomics and in silico structural and functional annotation of the barley powdery mildew Blumeria graminis f. sp. hordei. Methods 54, 432–441. https://doi.org/10.1016/j.ymeth.2011.03.006

Bo¨ck, A., Forchhammer, K., Heider, J., Baron, C., 1991. Selenoprotein synthesis: an expansion of the genetic code. Trends in Biochemical Sciences 16, 463–467. https://doi.org/10.1016/0968-0004(91)90180-4

Bradley, P., Misura, K.M.S., Baker, D., 2005. Toward high-resolution de novo structure prediction for small proteins. Science 309, 1868–1871. https://doi.org/10.1126/science.1113801

Bragg William Henry, 1913. The reflection of X-rays by crystals. (II.). Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character 89, 246–248. https://doi.org/10.1098/rspa.1913.0082

Branden, C.I., Tooze, J., 1998. Introduction to Protein Structure, 2 edition. ed. Garland Science, New York.

Browne, A., 1997. Neural Network Perspectives on Cognition and Adaptive Robotics. CRC Press.

Buchan, D.W.A., Minneci, F., Nugent, T.C.O., Bryson, K., Jones, D.T., 2013. Scalable web services for the PSIPRED Protein Analysis Workbench. Nucleic Acids Res. 41, W349-357. https://doi.org/10.1093/nar/gkt381

Buenavista, M.T., Roche, D.B., McGuffin, L.J., 2012. Improvement of 3D protein models using multiple templates guided by single-template model quality assessment. Bioinformatics 28, 1851–1857. https://doi.org/10.1093/bioinformatics/bts292

Cabra, V., Samsó, M., 2015. Do's and Don'ts of Cryo-electron Microscopy: A Primer on Sample Preparation and High Quality Data Collection for Macromolecular 3D Reconstruction. Journal of visualized experiments : JoVE. https://doi.org/10.3791/52311

Cao, R., 2014. Signaling in the Brain: In Search of Functional Units. Philosophy of Science 81, 891–901. https://doi.org/10.1086/677688

Caudill, M., 1987. Neural Networks Primer, Part I. AI Expert 2, 46–52.

Chauhan, A.K., Varma, A., 2013. A Textbook of Molecular Biotechnology. I. K. International Pvt Ltd.

Cheng, J., Choe, M.-H., Elofsson, A., Han, K.-S., Hou, J., Maghrabi, A.H.A., McGuffin, L.J., Menéndez-Hurtado, D., Olechnovič, K., Schwede, T., Studer, G., Uziela, K., Venclovas, Č., Wallner, B., 2019. Estimation of model accuracy in CASP13. Proteins: Structure, Function, and Bioinformatics.

Cheng, J., Randall, A.Z., Sweredoski, M.J., Baldi, P., 2005. SCRATCH: a protein structure and structural feature prediction server. Nucleic Acids Res. 33, W72-76. https://doi.org/10.1093/nar/gki396

Cheng, J., Wang, Z., Tegge, A.N., Eickholt, J., 2009. Prediction of global and local quality of CASP8 models by MULTICOM series. Proteins 77 Suppl 9, 181–184. https://doi.org/10.1002/prot.22487

Cheng, Y., 2015. Single-Particle Cryo-EM at Crystallographic Resolution. Cell 161, 450–457. https://doi.org/10.1016/j.cell.2015.03.049

Chothia, C., Lesk, A. m., 1986. The relation between the divergence of sequence and structure in proteins. The EMBO Journal 5, 823–826. https://doi.org/10.1002/j.1460-2075.1986.tb04288.x

Corcoran, T., Zamora-Resendiz, R., Liu, X., Crivelli, S., 2018. A Spatial Mapping Algorithm with Applications in Deep Learning-Based Structure Classification. arXiv:1802.02532 [cs].

D'Addona, D.M., 2016. Neural Network, in: The International Academy for Produ, Laperrière, L., Reinhart, G. (Eds.), CIRP Encyclopedia of Production Engineering. Springer, Berlin, Heidelberg, pp. 1–9. https://doi.org/10.1007/978-3-642-35950-7_6563-3

Das, R., Baker, D., 2008. Macromolecular modeling with rosetta. Annu. Rev. Biochem. 77, 363–382. https://doi.org/10.1146/annurev.biochem.77.062906.171838

Dawson, N.L., Lewis, T.E., Das, S., Lees, J.G., Lee, D., Ashford, P., Orengo, C.A., Sillitoe, I., 2017. CATH: an expanded resource to predict protein function through structure and sequence. Nucleic Acids Res 45, D289–D295. https://doi.org/10.1093/nar/gkw1098

Deng, L., 2012. The MNIST database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine 29, 141–142.

Duchi, J., Hazan, E., Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12, 2121–2159.

Dunwell, T.L., McGuffin, L.J., Dunwell, J.M., Pfeifer, G.P., 2013. The mysterious presence of a 5-methylcytosine oxidase in the Drosophila genome: possible explanations. Cell Cycle 12, 3357–3365. https://doi.org/10.4161/cc.26540

Eastwood, M.P., Hardin, C., Luthey-Schulten, Z., Wolynes, P.G., 2001. Evaluating protein structure-prediction schemes using energy landscape theory. IBM Journal of Research and Development 45, 475–497. https://doi.org/10.1147/rd.453.0475

Eddy, S.R., 2011. Accelerated Profile HMM Searches. PLoS Comput. Biol. 7, e1002195. https://doi.org/10.1371/journal.pcbi.1002195

Efendigil, T., Önüt, S., Kahraman, C., 2009. A decision support system for demand forecasting with artificial neural networks and neuro-fuzzy models: A comparative analysis. Expert Systems with Applications 36, 6697–6707. https://doi.org/10.1016/j.eswa.2008.08.058

Eisenberg, D., Lüthy, R., Bowie, J.U., 1997. VERIFY3D: assessment of protein models with three-dimensional profiles. Meth. Enzymol. 277, 396–404.

Elias, I., 2006. Settling the intractability of multiple alignment. J. Comput. Biol. 13, 1323–1339. https://doi.org/10.1089/cmb.2006.13.1323

Elofsson, A., Joo, K., Keasar, C., Lee, J., Maghrabi, A.H.A., Manavalan, B., McGuffin, L.J., Hurtado, D.M., Mirabello, C., Pilstål, R., Sidi, T., Uziela, K., Wallner, B., 2018. Methods for estimation of model accuracy in CASP12. Proteins: Structure, Function, and Bioinformatics 86, 361–373. https://doi.org/10.1002/prot.25395

Faraggi, E., Kloczkowski, A., 2014. A global machine learning based scoring function for protein structure prediction. Proteins 82, 752–759. https://doi.org/10.1002/prot.24454

Filip, N., Iancu, C.-E., 2018. Non-Proteinogenic Amino Acids. BoD – Books on Demand.

Fraser-Pitt, D., O'Neil, D., 2015. Cystic fibrosis – a multiorgan protein misfolding disease. Future Sci OA 1. https://doi.org/10.4155/fso.15.57

Fuller, S.J., McGuffin, L.J., Marshall, A.K., Giraldo, A., Pikkarainen, S., Clerk, A., Sugden, P.H., 2012. A novel non-canonical mechanism of regulation of MST3 (mammalian Sterile20-related kinase 3). Biochem. J. 442, 595–610. https://doi.org/10.1042/BJ20112000

Géron, A., 2017. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Inc.

Geysen, H.M., Rodda, S.J., Mason, T.J., 1986. A priori delineation of a peptide which mimics a discontinuous antigenic determinant. Molecular Immunology 23, 709–715. https://doi.org/10.1016/0161-5890(86)90081-7

Gholami, R., Fakhari, N., 2017. Chapter 27 - Support Vector Machine: Principles, Parameters, and Applications, in: Samui, P., Sekhar, S., Balas, V.E. (Eds.), Handbook of Neural Computation. Academic Press, pp. 515–535. https://doi.org/10.1016/B978-0-12-811318-9.00027-2

Giepmans, B.N.G., van IJzendoorn, S.C.D., 2009. Epithelial cell–cell junctions and plasma membrane domains. Biochimica et Biophysica Acta (BBA) - Biomembranes 1788, 820–831. https://doi.org/10.1016/j.bbamem.2008.07.015

Ginalski, K., Elofsson, A., Fischer, D., Rychlewski, L., 2003. 3D-Jury: a simple approach to improve protein structure predictions. Bioinformatics 19, 1015–1018. https://doi.org/10.1093/bioinformatics/btg124

Goldstein, T., Studer, C., Baraniuk, R., 2014. A Field Guide to Forward-Backward Splitting with a FASTA Implementation. arXiv:1411.3406 [cs].

Guinier, A., 2013. X-Ray Diffraction: In Crystals, Imperfect Crystals, and Amorphous Bodies. Courier Corporation.

Haas, J., Barbato, A., Behringer, D., Studer, G., Roth, S., Bertoni, M., Mostaguir, K., Gumienny, R., Schwede, T., 2018. Continuous Automated Model EvaluatiOn (CAMEO) complementing the critical assessment of structure prediction in CASP12. Proteins 86 Suppl 1, 387–398. https://doi.org/10.1002/prot.25431

Halton, M., 2018. Recycling hope for plastic-hungry enzyme. BBC News.

Hauke, J., Kossowski, T., 2011. Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data. Quaestiones geographicae 30, 87–93.

Hebb, D., 1949. The Organization of Behavior. New York: John Wiley and Sons, Inc.

Hecht-Nielsen, R., 1988. Neurocomputing: picking the human brain. IEEE Spectrum 25, 36–41. https://doi.org/10.1109/6.4520

Hendlich, M., Lackner, P., Weitckus, S., Floeckner, H., Froschauer, R., Gottsbacher, K., Casari, G., Sippl, M.J., 1990. Identification of native protein folds amongst a large number of incorrect models: The calculation of low energy conformations from potentials of mean force. Journal of Molecular Biology 216, 167–180. https://doi.org/10.1016/S0022-2836(05)80068-3

Hildebrand, A., Remmert, M., Biegert, A., Söding, J., 2009. Fast and accurate automatic structure prediction with HHpred. Proteins 77 Suppl 9, 128–132. https://doi.org/10.1002/prot.22499

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., Kingsbury, B., 2012. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. IEEE Signal Processing Magazine 29, 82–97. https://doi.org/10.1109/MSP.2012.2205597

Hinton, G.E., Osindero, S., Teh, Y.-W., 2006. A fast learning algorithm for deep belief nets. Neural Comput 18, 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527

Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the Dimensionality of Data with Neural Networks. Science 313, 504–507. https://doi.org/10.1126/science.1127647

Holm, L., Sander, C., 1996. The FSSP database: fold classification based on structure-structure alignment of proteins. Nucleic acids research 24, 206–209. https://doi.org/10.1093/nar/24.1.206

Hooft, R.W., Vriend, G., Sander, C., Abola, E.E., 1996. Errors in protein structures. Nature 381, 272. https://doi.org/10.1038/381272a0

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. PNAS 79, 2554–2558. https://doi.org/10.1073/pnas.79.8.2554

Hsu, C.-W., Chang, C.-C., Lin, C.-J., 2003. A practical guide to support vector classification.

Hurtado, D.M., Uziela, K., Elofsson, A., 2018. Deep transfer learning in the assessment of the quality of protein models. arXiv:1804.06281 [q-bio].

Jing, X., Dong, Q., 2017. MQAPRank: improved global protein model quality assessment by learning-to-rank. BMC Bioinformatics 18, 275. https://doi.org/10.1186/s12859-017-1691-z

Jones, D.T., 1999. Protein secondary structure prediction based on position-specific scoring matrices. J. Mol. Biol. 292, 195–202. https://doi.org/10.1006/jmbi.1999.3091

Jones, D.T., Bryson, K., Coleman, A., McGuffin, L.J., Sadowski, M.I., Sodhi, J.S., Ward, J.J., 2005. Prediction of novel and analogous folds using fragment assembly and fold recognition. Proteins 61 Suppl 7, 143–151. https://doi.org/10.1002/prot.20731

Jones, D.T., Cozzetto, D., 2015. DISOPRED3: precise disordered region predictions with annotated protein-binding activity. Bioinformatics 31, 857–863. https://doi.org/10.1093/bioinformatics/btu744

Jones, D.T., Singh, T., Kosciolek, T., Tetchner, S., 2015. MetaPSICOV: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. Bioinformatics 31, 999–1006. https://doi.org/10.1093/bioinformatics/btu791

Jones, D.T., Taylor, W.R., Thornton, J.M., 1992. The rapid generation of mutation data matrices from protein sequences. Comput. Appl. Biosci. 8, 275–282.

Jothi, A., 2012. Principles, challenges and advances in ab initio protein structure prediction. Protein Pept. Lett. 19, 1194–1204.

Kabsch, W., Sander, C., 1983. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 22, 2577–2637. https://doi.org/10.1002/bip.360221211

Kaczanowski, S., Zielenkiewicz, P., 2010. Why similar protein sequences encode similar three-dimensional structures? Theor Chem Acc 125, 643–650. https://doi.org/10.1007/s00214-009-0656-3

Kamisetty, H., Ovchinnikov, S., Baker, D., 2013. Assessing the utility of coevolution-based residue-residue contact predictions in a sequence- and structure-rich era. Proc. Natl. Acad. Sci. U.S.A. 110, 15674–15679. https://doi.org/10.1073/pnas.1314045110

Keasar, C., McGuffin, L.J., Wallner, B., Chopra, G., Adhikari, B., Bhattacharya, D., Blake, L., Bortot, L.O., Cao, R., Dhanasekaran, B.K., Dimas, I., Faccioli, R.A., Faraggi, E., Ganzynkowicz, R., Ghosh, Sambit, Ghosh, Soma, Giełdoń, A., Golon, L., He, Y., Heo, L., Hou, J., Khan, M., Khatib, F., Khoury, G.A., Kieslich, C., Kim, D.E., Krupa, P., Lee, G.R., Li, H., Li, J., Lipska, A., Liwo, A., Maghrabi, A.H.A., Mirdita, M., Mirzaei, S., Mozolewska, M.A., Onel, M., Ovchinnikov, S., Shah, A., Shah, U., Sidi, T., Sieradzan, A.K., Ślusarz, M., Ślusarz, R., Smadbeck, J., Tamamis, P., Trieber, N., Wirecki, T., Yin, Y., Zhang, Y., Bacardit, J., Baranowski, M., Chapman, N., Cooper, S., Defelicibus, A., Flatten, J., Koepnick, B., Popović, Z., Zaborowski, B., Baker, D., Cheng, J., Czaplewski, C., Delbem, A.C.B., Floudas, C., Kloczkowski, A., Ołdziej, S., Levitt, M., Scheraga, H., Seok, C., Söding, J., Vishveshwara, S., Xu, D., Crivelli, S.N., 2018. An analysis and evaluation of the WeFold collaborative for protein structure prediction and its pipelines in CASP11 and CASP12. Scientific Reports 8, 9939. https://doi.org/10.1038/s41598-018-26812-8

Khoury, G.A., Tamamis, P., Pinnaduwage, N., Smadbeck, J., Kieslich, C.A., Floudas, C.A., 2014. Princeton_TIGRESS: protein geometry refinement using simulations and support vector machines. Proteins 82, 794–814. https://doi.org/10.1002/prot.24459

Kieslich, C.A., Smadbeck, J., Khoury, G.A., Floudas, C.A., 2016. conSSert: Consensus SVM Model for Accurate Prediction of Ordered Secondary Structure. J Chem Inf Model 56, 455–461. https://doi.org/10.1021/acs.jcim.5b00566

Kihara, D., Lu, H., Kolinski, A., Skolnick, J., 2001. TOUCHSTONE: an ab initio protein structure prediction method that uses threading-based tertiary restraints. Proc. Natl. Acad. Sci. U.S.A. 98, 10125–10130. https://doi.org/10.1073/pnas.181328398

Krissinel, E., Henrick, K., 2007. Inference of macromolecular assemblies from crystalline state. Journal of molecular biology 372, 774–797.

Kryshtafovych, A., Barbato, A., Fidelis, K., Monastyrskyy, B., Schwede, T., Tramontano, A., 2014. Assessment of the assessment: evaluation of the model quality estimates in CASP10. Proteins 82 Suppl 2, 112–126. https://doi.org/10.1002/prot.24347

Kryshtafovych, A., Barbato, A., Monastyrskyy, B., Fidelis, K., Schwede, T., Tramontano, A., 2016. Methods of model accuracy estimation can help selecting the best models from decoy sets:

Assessment of model accuracy estimations in CASP11. Proteins 84 Suppl 1, 349–369. https://doi.org/10.1002/prot.24919

Kryshtafovych, A., Fidelis, K., Tramontano, A., 2011. Evaluation of model quality predictions in CASP9. Proteins 79 Suppl 10, 91–106. https://doi.org/10.1002/prot.23180

Kryshtafovych, A., Monastyrskyy, B., Fidelis, K., Moult, J., Schwede, T., Tramontano, A., 2018a. Evaluation of the template-based modeling in CASP12. Proteins 86 Suppl 1, 321–334. https://doi.org/10.1002/prot.25425

Kryshtafovych, A., Monastyrskyy, B., Fidelis, K., Schwede, T., Tramontano, A., 2018b. Assessment of model accuracy estimations in CASP12. Proteins: Structure, Function, and Bioinformatics 86, 345–360. https://doi.org/10.1002/prot.25371

Larkin, M.A., Blackshields, G., Brown, N.P., Chenna, R., McGettigan, P.A., McWilliam, H., Valentin, F., Wallace, I.M., Wilm, A., Lopez, R., Thompson, J.D., Gibson, T.J., Higgins, D.G., 2007. Clustal W and Clustal X version 2.0. Bioinformatics 23, 2947–2948. https://doi.org/10.1093/bioinformatics/btm404

Larrañaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J.A., Armañanzas, R., Santafé, G., Pérez, A., Robles, V., 2006. Machine learning in bioinformatics. Brief. Bioinformatics 7, 86–112.

Larsson, P., Skwark, M.J., Wallner, B., Elofsson, A., 2009. Assessment of global and local model quality in CASP8 using Pcons and ProQ. Proteins 77 Suppl 9, 167–172. https://doi.org/10.1002/prot.22476

Laskowski, R.A., Rullmannn, J.A., MacArthur, M.W., Kaptein, R., Thornton, J.M., 1996. AQUA and PROCHECK-NMR: programs for checking the quality of protein structures solved by NMR. J. Biomol. NMR 8, 477–486.

Lassmann, T., Sonnhammer, E.L., 2005. Kalign – an accurate and fast multiple sequence alignment algorithm. BMC Bioinformatics 6, 298. https://doi.org/10.1186/1471-2105-6-298

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444. https://doi.org/10.1038/nature14539

Lee, G.R., Heo, L., Seok, C., 2016. Effective protein model structure refinement by loop modeling and overall relaxation. Proteins 84 Suppl 1, 293–301. https://doi.org/10.1002/prot.24858

Leybaert, L., Braet, K., Vandamme, W., Cabooter, L., Martin, P.E., Evans, W.H., 2003. Connexin channels, connexin mimetic peptides and ATP release. Cell communication & adhesion 10, 251–257.

Lipman, D.J., Pearson, W.R., 1985. Rapid and sensitive protein similarity searches. Science 227, 1435–1441. https://doi.org/10.1126/science.2983426

Lovell, S.C., Davis, I.W., Arendall, W.B., de Bakker, P.I.W., Word, J.M., Prisant, M.G., Richardson, J.S., Richardson, D.C., 2003. Structure validation by Calpha geometry: phi,psi and Cbeta deviation. Proteins 50, 437–450. https://doi.org/10.1002/prot.10286

Lundström, J., Rychlewski, L., Bujnicki, J., Elofsson, A., 2001. Pcons: a neural-network-based consensus predictor that improves fold recognition. Protein Sci. 10, 2354–2362. https://doi.org/10.1110/ps.08501

Ma, J., Wang, S., Zhao, F., Xu, J., 2013. Protein threading using context-specific alignment potential. Bioinformatics 29, i257–i265. https://doi.org/10.1093/bioinformatics/btt210

Maeda, S., Nakagawa, S., Suga, M., Yamashita, E., Oshima, A., Fujiyoshi, Y., Tsukihara, T., 2009. Structure of the connexin 26 gap junction channel at 3.5 A resolution. Nature 458, 597–602. https://doi.org/10.1038/nature07869

Maghrabi, A.H.A., McGuffin, L.J., 2017. ModFOLD6: an accurate web server for the global and local quality estimation of 3D protein models. Nucleic Acids Res. 45, W416–W421. https://doi.org/10.1093/nar/gkx332

Manavalan, B., Lee, J., 2017. SVMQA: support-vector-machine-based protein single-model quality assessment. Bioinformatics 33, 2496–2503. https://doi.org/10.1093/bioinformatics/btx222

Margelevicius, M., Venclovas, C., 2010. Detection of distant evolutionary relationships between protein families using theory of sequence profile-profile comparison. BMC Bioinformatics 11, 89. https://doi.org/10.1186/1471-2105-11-89

Mariani, V., Biasini, M., Barbato, A., Schwede, T., 2013. lDDT: a local superposition-free score for comparing protein structures and models using distance difference tests. Bioinformatics 29, 2722–2728. https://doi.org/10.1093/bioinformatics/btt473

Martí-Renom, M.A., Stuart, A.C., Fiser, A., Sánchez, R., Melo, F., Sali, A., 2000. Comparative protein structure modeling of genes and genomes. Annu Rev Biophys Biomol Struct 29, 291–325. https://doi.org/10.1146/annurev.biophys.29.1.291

McCulloch, W.S., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133. https://doi.org/10.1007/BF02478259

McDonald, J.H., 2014. Handbook of Biological Statistics 3rd ed Sparky House Publishing. Baltimore, MD: Available http://www. biostathandbook. com/transformation. html.

McDonnell, M.D., Tissera, M.D., Vladusich, T., Schaik, A. van, Tapson, J., 2015. Fast, Simple and Accurate Handwritten Digit Classification by Training Shallow Neural Network Classifiers with the ‘Extreme Learning Machine’ Algorithm. PLOS ONE 10, e0134254. https://doi.org/10.1371/journal.pone.0134254

McGuffin, L.J., 2009. Prediction of global and local model quality in CASP8 using the ModFOLD server. Proteins 77 Suppl 9, 185–190. https://doi.org/10.1002/prot.22491

McGuffin, L.J., 2008a. Intrinsic disorder prediction from the analysis of multiple protein fold recognition models. Bioinformatics 24, 1798–1804. https://doi.org/10.1093/bioinformatics/btn326

McGuffin, L.J., 2008b. The ModFOLD server for the quality assessment of protein structural models. Bioinformatics 24, 586–587. https://doi.org/10.1093/bioinformatics/btn014

McGuffin, L.J., 2007. Benchmarking consensus model quality assessment for protein fold recognition. BMC Bioinformatics 8, 345. https://doi.org/10.1186/1471-2105-8-345

McGuffin, L.J., Adiyaman, R., Maghrabi, A.H.A., Shuid, A.N., Brackenridge, D.A., Nealon, J.O., Philomina, L.S., 2019. IntFOLD: an integrated web resource for high performance protein structure and function prediction. Nucleic Acids Res 47, W408–W413. https://doi.org/10.1093/nar/gkz322

McGuffin, L.J., Atkins, J.D., Salehe, B.R., Shuid, A.N., Roche, D.B., 2015. IntFOLD: an integrated server for modelling protein structures and functions from amino acid sequences. Nucleic Acids Res. 43, W169-173. https://doi.org/10.1093/nar/gkv236

McGuffin, L.J., Buenavista, M.T., Roche, D.B., 2013. The ModFOLD4 server for the quality assessment of 3D protein models. Nucleic Acids Res. 41, W368-372. https://doi.org/10.1093/nar/gkt294

McGuffin, L.J., Jones, D.T., 2003. Improvement of the GenTHREADER method for genomic fold recognition. Bioinformatics 19, 874–881. https://doi.org/10.1093/bioinformatics/btg097

McGuffin, L.J., Roche, D.B., 2011. Automated tertiary structure prediction with accurate local model quality assessment using the IntFOLD-TS method. Proteins 79 Suppl 10, 137–146. https://doi.org/10.1002/prot.23120

McGuffin, L.J., Roche, D.B., 2010. Rapid model quality assessment for protein structure predictions using the comparison of multiple models without structural alignments. Bioinformatics 26, 182–188. https://doi.org/10.1093/bioinformatics/btp629

McGuffin, L.J., Shuid, A.N., Kempster, R., Maghrabi, A.H.A., Nealon, J.O., Salehe, B.R., Atkins, J.D., Roche, D.B., 2018. Accurate template-based modeling in CASP12 using the IntFOLD4-TS, ModFOLD6, and ReFOLD methods. Proteins: Structure, Function, and Bioinformatics 86, 335–344. https://doi.org/10.1002/prot.25360

Meier, A., Söding, J., 2015. Automatic Prediction of Protein 3D Structures by Probabilistic Multi-template Homology Modeling. PLOS Computational Biology 11, e1004343. https://doi.org/10.1371/journal.pcbi.1004343

Mirjalili, V., Feig, M., 2013. Protein Structure Refinement through Structure Selection and Averaging from Molecular Dynamics Ensembles. J Chem Theory Comput 9, 1294–1303. https://doi.org/10.1021/ct300962x

Mirzaei, S., Sidi, T., Keasar, C., Crivelli, S., 2016. Purely Structural Protein Scoring Functions Using Support Vector Machine and Ensemble Learning. IEEE/ACM Transactions on

Computational Biology and Bioinformatics PP, 1–1. https://doi.org/10.1109/TCBB.2016.2602269

Monteagudo, L.V., Ferrer, L.M., Catalan-Insa, E., Savva, D., McGuffin, L.J., Tejedor, M.T., 2015. In silico identification and three-dimensional modelling of the missense mutation in ADAMTS2 in a sheep flock with dermatosparaxis. Vet. Dermatol. 26, 49–52, e15-16. https://doi.org/10.1111/vde.12178

Moult, J., Fidelis, K., Kryshtafovych, A., Rost, B., Hubbard, T., Tramontano, A., 2007. Critical assessment of methods of protein structure prediction—Round VII. Proteins: Structure, Function, and Bioinformatics 69, 3–9.

Moult, J., Fidelis, K., Kryshtafovych, A., Schwede, T., Tramontano, A., 2014. Critical assessment of methods of protein structure prediction (CASP)--round x. Proteins 82 Suppl 2, 1–6. https://doi.org/10.1002/prot.24452

Moult, J., Fidelis, K., Rost, B., Hubbard, T., Tramontano, A., 2005. Critical assessment of methods of protein structure prediction (CASP)--round 6. Proteins 61 Suppl 7, 3–7. https://doi.org/10.1002/prot.20716

Moult, J., Pedersen, J.T., Judson, R., Fidelis, K., 1995. A large-scale experiment to assess protein structure prediction methods. Proteins 23, ii–v. https://doi.org/10.1002/prot.340230303

Mount, D.W., 2004. Bioinformatics: Sequence and Genome Analysis, 2nd Revised edition edition. ed. Cold Spring Harbor Press, Cold Spring Harbor, N.Y.

Mudavath, S., Pittu, V., 2013. HPLC Method Developmentfor Proteinsand Polypeptides 266–276.

Murata, K., Wolf, M., 2018. Cryo-electron microscopy for structural analysis of dynamic biological macromolecules. Biochimica et Biophysica Acta (BBA) - General Subjects, Biophysical Exploration of Dynamical Ordering of Biomolecular Systems 1862, 324–334. https://doi.org/10.1016/j.bbagen.2017.07.020

Murtagh, F., 1985. Multidimensional clustering algorithms. Physica-Verlag.

Nakagawa, S., Maeda, S., Tsukihara, T., 2010. Structural and functional studies of gap junction channels. Current Opinion in Structural Biology, Membranes / Engineering and design 20, 423–430. https://doi.org/10.1016/j.sbi.2010.05.003

Needleman, S.B., Wunsch, C.D., 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 48, 443–453. https://doi.org/10.1016/0022-2836(70)90057-4

Neumann, J., 1945. First Draft of a Report on the EDVAC. Moore School of Electrical Engineering University of Pennsylvania.

Nguyen, S.P., Shang, Y., Xu, D., 2014. DL-PRO: A Novel Deep Learning Method for Protein Model Quality Assessment. Proc Int Jt Conf Neural Netw 2014, 2071–2078. https://doi.org/10.1109/IJCNN.2014.6889891

Noivirt-Brik, O., Prilusky, J., Sussman, J.L., 2009. Assessment of disorder predictions in CASP8. Proteins 77 Suppl 9, 210–216. https://doi.org/10.1002/prot.22586

Ó Conchúir, S., Barlow, K.A., Pache, R.A., Ollikainen, N., Kundert, K., O'Meara, M.J., Smith, C.A., Kortemme, T., 2015. A Web Resource for Standardized Benchmark Datasets, Metrics, and Rosetta Protocols for Macromolecular Modeling and Design. PLoS ONE 10, e0130433. https://doi.org/10.1371/journal.pone.0130433

Olechnovič, K., Kulberkytė, E., Venclovas, C., 2013. CAD-score: a new contact area difference-based function for evaluation of protein structural models. Proteins 81, 149–162. https://doi.org/10.1002/prot.24172

Olechnovič, K., Venclovas, Č., 2017. VoroMQA: Assessment of protein structure quality using interatomic contact areas. Proteins 85, 1131–1145. https://doi.org/10.1002/prot.25278

Olechnovič, K., Venclovas, C., 2014. Voronota: A fast and reliable tool for computing the vertices of the Voronoi diagram of atomic balls. J Comput Chem 35, 672–681. https://doi.org/10.1002/jcc.23538

Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B., Thornton, J.M., 1997. CATH--a hierarchic classification of protein domain structures. Structure 5, 1093–1108.

Patthy, L., 2008. Protein Evolution. Wiley.

Petsko, G., Ringe, D., 2008. Protein Structure and Function, Primers in Biology. Oxford University Press, Oxford, New York.

Phaisangittisagul, E., 2016. An analysis of the regularization between L2 and dropout in single hidden layer neural network, in: 2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS). IEEE, pp. 174–179.

Polyanovsky, V.O., Roytberg, M.A., Tumanyan, V.G., 2011. Comparative analysis of the quality of a global algorithm and a local algorithm for alignment of two sequences. Algorithms Mol Biol 6, 25. https://doi.org/10.1186/1748-7188-6-25

Priddy, K.L., Keller, P.E., 2005. Artificial Neural Networks: An Introduction. SPIE Press.

Rampasek, L., Goldenberg, A., 2016. Tensorflow: Biology's gateway to deep learning? Cell systems 2, 12–14.

Ray, A., Lindahl, E., Wallner, B., 2012. Improved model quality assessment using ProQ2. BMC Bioinformatics 13, 224. https://doi.org/10.1186/1471-2105-13-224

Reilly, D.L., Cooper, L.N., Elbaum, C., 1982. A Neural Model for Category Learning. Biol. Cybern. 45, 35–41. https://doi.org/10.1007/BF00387211

Roche, D.B., Buenavista, M.T., McGuffin, L.J., 2014. Assessing the quality of modelled 3D protein structures using the ModFOLD server. Methods Mol. Biol. 1137, 83–103. https://doi.org/10.1007/978-1-4939-0366-5_7

Roche, D.B., Buenavista, M.T., McGuffin, L.J., 2013. The FunFOLD2 server for the prediction of protein-ligand interactions. Nucleic Acids Res. 41, W303-307. https://doi.org/10.1093/nar/gkt498

Roche, D.B., Buenavista, M.T., Tetchner, S.J., McGuffin, L.J., 2011a. The IntFOLD server: an integrated web resource for protein fold recognition, 3D model quality assessment, intrinsic disorder prediction, domain prediction and ligand binding site prediction. Nucleic Acids Res. 39, W171-176. https://doi.org/10.1093/nar/gkr184

Roche, D.B., McGuffin, L.J., 2016. In silico identification and characterization of protein-ligand binding sites, in: Computational Design of Ligand Binding Proteins. Springer, pp. 1–21.

Roche, D.B., Tetchner, S.J., McGuffin, L.J., 2011b. FunFOLD: an improved automated method for the prediction of ligand binding residues using 3D models of proteins. BMC Bioinformatics 12, 160. https://doi.org/10.1186/1471-2105-12-160

Rojas, A., Czaplewski, C., Liwo, A., Makowski, M., O_dziej, S., Kaz¬¥mierkiewicz, R., Scheraga, H., Murarka, R., 2008. Simulation of Protein Structure and Dynamics with the Coarse-Grained UNRES Force Field. pp. 107–122. https://doi.org/10.1201/9781420059564.ch8

Rosenberg, M.S., 2009. Sequence Alignment: Methods, Models, Concepts, and Strategies. University of California Press.

Rosenblatt, F., 1962. Principles of neurodynamics: perceptrons and the theory of brain mechanisms. Spartan Books.

Rosenblatt, F., 1957. The Perceptron, a Perceiving and Recognizing Automaton Project Para. Cornell Aeronautical Laboratory.

Rost, B., Schneider, R., Sander, C., 1997. Protein fold recognition by prediction-based threading. J. Mol. Biol. 270, 471–480. https://doi.org/10.1006/jmbi.1997.1101

Roy, A., Kucukural, A., Zhang, Y., 2010. I-TASSER: a unified platform for automated protein structure and function prediction. Nat Protoc 5, 725–738. https://doi.org/10.1038/nprot.2010.5

Ruck, D.W., Rogers, S.K., Kabrisky, M., 1990. Feature selection using a multilayer perceptron. Journal of Neural Network Computing 2, 40–48.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323, 533–536. https://doi.org/10.1038/323533a0

Sayers, Eric W., Agarwala, R., Bolton, E.E., Brister, J.R., Canese, K., Clark, K., Connor, R., Fiorini, N., Funk, K., Hefferon, T., 2019. Database resources of the national center for biotechnology information. Nucleic acids research 47, D23.

Sayers, Eric W, Cavanaugh, M., Clark, K., Ostell, J., Pruitt, K.D., Karsch-Mizrachi, I., 2019. GenBank. Nucleic Acids Research 47, D94–D99. https://doi.org/10.1093/nar/gky989

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural networks 61, 85–117.

Schmidt, C., Urlaub, H., 2017. Combining cryo-electron microscopy (cryo-EM) and cross-linking mass spectrometry (CX-MS) for structural elucidation of large protein assemblies. Current Opinion in Structural Biology, Cryo electron microscopy: exciting advances in CryoEM herald a new era in structural biology • Biophysical methods: behind the scenes of the cryo-EM revolution 46, 157–168. https://doi.org/10.1016/j.sbi.2017.10.005

Schmidt, T., Haas, J., Gallo Cassarino, T., Schwede, T., 2011. Assessment of ligand-binding residue predictions in CASP9. Proteins 79 Suppl 10, 126–136. https://doi.org/10.1002/prot.23174

Schneider, M., Brock, O., 2014. Combining physicochemical and evolutionary information for protein contact prediction. PLoS ONE 9, e108438. https://doi.org/10.1371/journal.pone.0108438

Seikel, J.A., Konstantopoulos, K., Drumright, D.G., 2018. Neuroanatomy and Neurophysiology for Speech and Hearing Sciences. Plural Publishing.

Shakhnovich, B.E., Deeds, E., Delisi, C., Shakhnovich, E., 2005. Protein structure and evolutionary history determine sequence space topology. Genome Res 15, 385–392. https://doi.org/10.1101/gr.3133605

Shuid, A.N., Kempster, R., McGuffin, L.J., 2017. ReFOLD: a server for the refinement of 3D protein models guided by accurate quality estimates. Nucleic Acids Res. 45, W422–W428. https://doi.org/10.1093/nar/gkx249

Sievers, F., Higgins, D.G., 2014. Clustal Omega, accurate alignment of very large numbers of sequences. Methods Mol. Biol. 1079, 105–116. https://doi.org/10.1007/978-1-62703-646-7_6

Siew, N., Elofsson, A., Rychlewski, L., Fischer, D., 2000. MaxSub: an automated measure for the assessment of protein structure prediction quality. Bioinformatics 16, 776–785. https://doi.org/10.1093/bioinformatics/16.9.776

Simoni, R.D., Hill, R.L., Vaughan, M., 2002. The discovery of the amino acid threonine: the work of William C. Rose [classical article]. J. Biol. Chem. 277, E25.

Sippl, M.J., 1993. Recognition of errors in three-dimensional structures of proteins. Proteins: Structure, Function, and Bioinformatics 17, 355–362. https://doi.org/10.1002/prot.340170404

Sippl, M.J., 1990. Calculation of conformational ensembles from potentials of mean force. An approach to the knowledge-based prediction of local structures in globular proteins. J. Mol. Biol. 213, 859–883. https://doi.org/10.1016/s0022-2836(05)80269-4

Smith, T.F., Waterman, M.S., 1981. Identification of common molecular subsequences. J. Mol. Biol. 147, 195–197.

Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical bayesian optimization of machine learning algorithms, in: Advances in Neural Information Processing Systems. pp. 2951–2959.

Söding, J., 2005. Protein homology detection by HMM-HMM comparison. Bioinformatics 21, 951–960. https://doi.org/10.1093/bioinformatics/bti125

Song, Y., DiMaio, F., Wang, R.Y.-R., Kim, D., Miles, C., Brunette, T., Thompson, J., Baker, D., 2013. High-resolution comparative modeling with RosettaCM. Structure 21, 1735–1742. https://doi.org/10.1016/j.str.2013.08.005

Soufi, B., Krug, K., Harst, A., Macek, B., 2015. Characterization of the E. coli proteome and its modifications during growth and ethanol stress. Front Microbiol 6. https://doi.org/10.3389/fmicb.2015.00103

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15, 1929–1958.

Sugden, P.H., McGuffin, L.J., Clerk, A., 2013. SOcK, MiSTs, MASK and STicKs: the GCKIII (germinal centre kinase III) kinases and their heterologous protein-protein interactions. Biochem. J. 454, 13–30. https://doi.org/10.1042/BJ20130219

Tanaka, M., Fujiwara, M., Ikegami, H., 1986. Propagation of a Gaussian wave packet in an absorbing medium. Phys. Rev. A 34, 4851–4858. https://doi.org/10.1103/PhysRevA.34.4851

Taverna, D.M., Goldstein, R.A., 2002. Why are proteins so robust to site mutations?11Edited by J. Thornton. Journal of Molecular Biology 315, 479–484. https://doi.org/10.1006/jmbi.2001.5226

Taylor, T.B., Mulley, G., Dills, A.H., Alsohim, A.S., McGuffin, L.J., Studholme, D.J., Silby, M.W., Brockhurst, M.A., Johnson, L.J., Jackson, R.W., 2015. Evolution. Evolutionary resurrection of flagellar motility via rewiring of the nitrogen regulation system. Science 347, 1014–1017. https://doi.org/10.1126/science.1259145

Théobald-Dietrich, A., Giegé, R., Rudinger-Thirion, J., 2005. Evidence for the existence in mRNAs of a hairpin element responsible for ribosome dependent pyrrolysine insertion into proteins. Biochimie, Facets of the RNA world 87, 813–817. https://doi.org/10.1016/j.biochi.2005.03.006

Tokuriki, N., Stricher, F., Schymkowitz, J., Serrano, L., Tawfik, D.S., 2007. The Stability Effects of Protein Mutations Appear to be Universally Distributed. Journal of Molecular Biology 369, 1318–1332. https://doi.org/10.1016/j.jmb.2007.03.069

Toth, G., Lent, C.S., Tougaw, P.D., Brazhnik, Y., Weng, W., Porod, W., Liu, R.-W., Huang, Y.-F., 1996. Quantum cellular neural networks. Superlattices and Microstructures 20, 473–478. https://doi.org/10.1006/spmi.1996.0104

UniProt Consortium, 2015. UniProt: a hub for protein information. Nucleic Acids Res. 43, D204-212. https://doi.org/10.1093/nar/gku989

Uziela, K., Menéndez Hurtado, D., Shu, N., Wallner, B., Elofsson, A., 2017. ProQ3D: improved model quality assessments using deep learning. Bioinformatics 33, 1578–1580. https://doi.org/10.1093/bioinformatics/btw819

Uziela, K., Shu, N., Wallner, B., Elofsson, A., 2016. ProQ3: Improved model quality assessments using Rosetta energy terms. Sci Rep 6, 33509. https://doi.org/10.1038/srep33509

Uziela, K., Wallner, B., 2016. ProQ2: estimation of model accuracy implemented in Rosetta. Bioinformatics 32, 1411–1413. https://doi.org/10.1093/bioinformatics/btv767

Vaiyapuri, S., Flora, G.D., Gibbins, J.M., 2015. Gap junctions and connexin hemichannels in the regulation of haemostasis and thrombosis. Biochem. Soc. Trans. 43, 489–494. https://doi.org/10.1042/BST20150055

Vaiyapuri, S., Jones, C.I., Sasikumar, P., Moraes, L.A., Munger, S.J., Wright, J.R., Ali, M.S., Sage, T., Kaiser, W.J., Tucker, K.L., Stain, C.J., Bye, A.P., Jones, S., Oviedo-Orta, E., Simon, A.M., Mahaut-Smith, M.P., Gibbins, J.M., 2012. Gap junctions and connexin hemichannels underpin hemostasis and thrombosis. Circulation 125, 2479–2491. https://doi.org/10.1161/CIRCULATIONAHA.112.101246

Vauquelin, L.-N., Robiquet, P.J., 1806. The discovery of a new plant principle in Asparagus sativus. Ann Chim 57, 14.

Wallner, B., Elofsson, A., 2007. Prediction of global and local model quality in CASP7 using Pcons and ProQ. Proteins 69 Suppl 8, 184–193. https://doi.org/10.1002/prot.21774

Wallner, B., Elofsson, A., 2006. Identification of correct regions in protein models using structural, alignment, and consensus information. Protein Sci. 15, 900–913. https://doi.org/10.1110/ps.051799606

Wallner, B., Elofsson, A., 2003. Can correct protein models be identified? Protein Sci. 12, 1073–1086. https://doi.org/10.1110/ps.0236803

Wang, L., Jiang, T., 1994. On the complexity of multiple sequence alignment. J. Comput. Biol. 1, 337–348. https://doi.org/10.1089/cmb.1994.1.337

Wang, S., Sun, S., Li, Z., Zhang, R., Xu, J., 2017. Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model. PLoS Comput. Biol. 13, e1005324. https://doi.org/10.1371/journal.pcbi.1005324

Wang, X., 2016. Deep Learning in Object Recognition, Detection, and Segmentation. now.

Wang, Z., Eickholt, J., Cheng, J., 2011. APOLLO: a quality assessment service for single and multiple protein models. Bioinformatics 27, 1715–1716. https://doi.org/10.1093/bioinformatics/btr268

Wang, Z., Eickholt, J., Cheng, J., 2010a. MULTICOM: a multi-level combination approach to protein structure prediction and its assessments in CASP8. Bioinformatics 26, 882–888. https://doi.org/10.1093/bioinformatics/btq058

Wang, Z., Eickholt, J., Cheng, J., 2010b. MULTICOM: a multi-level combination approach to protein structure prediction and its assessments in CASP8. Bioinformatics 26, 882–888.

Waterhouse, A., Bertoni, M., Bienert, S., Studer, G., Tauriello, G., Gumienny, R., Heer, F.T., de Beer, T.A.P., Rempfer, C., Bordoli, L., Lepore, R., Schwede, T., 2018. SWISS-MODEL: homology modelling of protein structures and complexes. Nucleic Acids Res. 46, W296–W303. https://doi.org/10.1093/nar/gky427

Wawer, I., Diehl, B., 2017. NMR Spectroscopy in Pharmaceutical Analysis. Elsevier.

Webb, B., Sali, A., 2016. Comparative Protein Structure Modeling Using MODELLER. Curr Protoc Bioinformatics 54, 5.6.1-5.6.37. https://doi.org/10.1002/cpbi.3

Widrow, B., 1960. Adaptive “adaline” neuron using chemical “memistors.”. Stanford Electronics Laboratories Technical Report.

Wiederstein, M., Sippl, M.J., 2007. ProSA-web: interactive web service for the recognition of errors in three-dimensional structures of proteins. Nucleic Acids Res. 35, W407-410. https://doi.org/10.1093/nar/gkm290

Wilkins, M.R., Gasteiger, E., Bairoch, A., Sanchez, J.C., Williams, K.L., Appel, R.D., Hochstrasser, D.F., 1999. Protein identification and analysis tools in the ExPASy server. Methods Mol. Biol. 112, 531–552. https://doi.org/10.1385/1-59259-584-7:531

Williams, A., Martin, G., Rovnyak, D., 2016. Modern NMR Approaches to the Structure Elucidation of Natural Products: Volume 2: Data Acquisition and Applications to Compound Classes. Royal Society of Chemistry.

Williamson, M., 2011. How Proteins Work, 1 edition. ed. Routledge, New York.

Wu, S., Zhang, Y., 2007. LOMETS: A local meta-threading-server for protein structure prediction. Nucleic Acids Res 35, 3375–3382. https://doi.org/10.1093/nar/gkm251

Xu, D., Zhang, Y., 2012. Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. Proteins 80, 1715–1735. https://doi.org/10.1002/prot.24065

Yang, J., Wang, Y., Zhang, Y., 2016. ResQ: An approach to unified estimation of B-factor and residue-specific error in protein structure prediction. J Mol Biol 428, 693–701. https://doi.org/10.1016/j.jmb.2015.09.024

Yang, J., Zhang, Y., 2015. I-TASSER server: new development for protein structure and function predictions. Nucleic Acids Res 43, W174–W181. https://doi.org/10.1093/nar/gkv342

Yang, Y., Faraggi, E., Zhao, H., Zhou, Y., 2011. Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates. Bioinformatics 27, 2076–2082. https://doi.org/10.1093/bioinformatics/btr350

Zeiler, M.D., 2012. ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

Zell, A., Mache, N., Huebner, R., Mamier, G., Vogt, M., Schmalzl, M., Herrmann, K.-U., 1994. SNNS (stuttgart neural network simulator), in: Neural Network Simulation Environments. Springer, pp. 165–186.

Zemla, A., 2003. LGA: A method for finding 3D similarities in protein structures. Nucleic Acids Res. 31, 3370–3374. https://doi.org/10.1093/nar/gkg571

Zemla, A., Venclovas, Č., Moult, J., Fidelis, K., 1999. Processing and analysis of CASP3 protein structure predictions. Proteins: Structure, Function, and Bioinformatics 37, 22–29. https://doi.org/10.1002/(SICI)1097-0134(1999)37:3+<22::AID-PROT5>3.0.CO;2-W

Zemla, A.T., 2006. Protein Classification Based on Analysis of Local Sequence-Structure Correspondence (No. UCRL-TR-218946). Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States). https://doi.org/10.2172/893991

Zhang, Jingfen, Wang, Q., Vantasin, K., Zhang, Jiong, He, Z., Kosztin, I., Shang, Y., Xu, D., 2011. A multilayer evaluation approach for protein structure prediction and model quality assessment. Proteins 79 Suppl 10, 172–184. https://doi.org/10.1002/prot.23184

Zhang, Y., Skolnick, J., 2005. TM-align: a protein structure alignment algorithm based on the TM-score. Nucleic Acids Res. 33, 2302–2309. https://doi.org/10.1093/nar/gki524

Zhang, Y., Skolnick, J., 2004. Scoring function for automated assessment of protein structure template quality. Proteins 57, 702–710. https://doi.org/10.1002/prot.20264

Zhou, H., Zhou, Y., 2005. SPARKS 2 and SP3 servers in CASP6. Proteins 61 Suppl 7, 152–156. https://doi.org/10.1002/prot.20732

Zhou, H., Zhou, Y., 2002. Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction. Protein Sci. 11, 2714–2726. https://doi.org/10.1110/ps.0217002

Zhu, Q., Azar, A.T. (Eds.), 2015. Complex System Modelling and Control Through Intelligent Soft Computations, Studies in Fuzziness and Soft Computing. Springer International Publishing.

Appendices

Appendix 1

Correlation Coefficient	Observed Measure	Combination	Correlation Score	Improvement
R	GDT-HA	GDT-HA~Mcqso+Mcqsr+D+M6	0.911	0.00129
Rho		GDT-HA~Mcqso+Mcqsr+D+M6	0.931	-0.00067
Tau		GDT-HA~Mcqso+Mcqsr+D+M6	0.771	-0.00121
R	GDT	GDT~Mcqso+Mcqsr+D+M6	0.927	0.00068
Rho		GDT~Mcqso+Mcqsr+D+M6	0.932	-0.00029
Tau		GDT~Mcqso+Mcqsr+D+M6	0.782	-0.00078
R	MaxSub	MaxSub~Mcqso+Mcqsr+D+M6	0.931	0.00108
Rho		GDT~Mcqso+Mcqsr+D+M6	0.935	0.00025
Tau		GDT~Mcqso+Mcqsr+D+M6	0.781	0.00047
R	TM-score	TM-score~Mcqso+Mcqsr+D+M6	0.93	0.00025
Rho		GDT~Mcqso+D+M6	0.932	0.00032
Tau		GDT~Mcqso+Mcqsr+D+M6	0.784	0.00028

Table S1. List of the top ranked combinations for the ten MQAP methods based on predicted versus observed scores using multiple linear regression. The top correlated combined methods are measured using Pearson's (R), Spearman's (Rho) and Kendall's (Tau) correlation coefficients**.** and the topmost correlated combinations are listed with the improvement over the scores of the single linear regression optimised method.

Appendix 2

```
# RSNNS_for_R
library(RSNNS)
library(data.table)
all1 <- fread("Global_QA_round1_all.out")
all2 <- fread("Global_QA_round2_all.out")
all <- rbind( all1, all2 )#combine data from both rounds
#remove data for where no native structures are available
DT <- subset( all, V1!="T0775" & V1!="T0779" & V1!="T0793" & V1!="T0795" & V1!="T0799"
& V1!="T0802" & V1!="T0804" & V1!="T0826" & V1!="T0828" & V1!="T0839" & V1!="T0842" &
V1!="T0844" & V1!="T0846" & V1!="T0850" )

training_set1 <- subset( DT, V1!="T0834" & V1!="T0798" & V1!="T0816" & V1!="T0845" &
V1!="T0822" & V1!="T0784" & V1!="T0833" & V1!="T0857" & V1!="T0763"
& V1!="T0782" & V1!="T0820" & V1!="T0854" & V1!="T0800" & V1!="T0840" & V1!="T0832" &
V1!="T0810" & V1!="T0827" & V1!="T0766"
& V1!="T0771" & V1!="T0858" & V1!="T0765" & V1!="T0855" & V1!="T0847" & V1!="T0796" &
V1!="T0778" & V1!="T0761" & V1!="T0764"
& V1!="T0821" )
#nrow(training_set1)
#9517 patterns/residues/rows in table
testing_set1 <-subset( DT, V1=="T0834" | V1=="T0798" | V1=="T0816" | V1=="T0845" |
V1=="T0822" | V1=="T0784" | V1=="T0833" | V1=="T0857" | V1=="T0763"
| V1=="T0782" | V1=="T0820" | V1=="T0854" | V1=="T0800" | V1=="T0840" | V1=="T0832" |
V1=="T0810" | V1=="T0827" | V1=="T0766"
| V1=="T0771" | V1=="T0858" | V1=="T0765" | V1=="T0855" | V1=="T0847" | V1=="T0796" |
V1=="T0778" | V1=="T0761" | V1=="T0764"
| V1=="T0821" )
#4586 patterns/residues/rows in table
#nrow(training_set1) + nrow(testing_set1) = 14103

training_set2 <- subset( DT, V1!="T0781" & V1!="T0829" & V1!="T0769" & V1!="T0836" &
V1!="T0759" & V1!="T0777" & V1!="T0852" & V1!="T0792" & V1!="T0818"
& V1!="T0772" & V1!="T0794" & V1!="T0811" & V1!="T0787" & V1!="T0762" & V1!="T0825" &
V1!="T0773" & V1!="T0801" & V1!="T0812"
& V1!="T0831" & V1!="T0760" & V1!="T0853" & V1!="T0815" & V1!="T0856" & V1!="T0788" &
V1!="T0805" & V1!="T0808" & V1!="T0835"
& V1!="T0843")
#9344 patterns/residues/rows in table
testing_set2 <- subset( DT, V1=="T0781" | V1=="T0829" | V1=="T0769" | V1=="T0836" |
V1=="T0759" | V1=="T0777" | V1=="T0852" | V1=="T0792" | V1=="T0818"
```

```
| V1=="T0772" | V1=="T0794" | V1=="T0811" | V1=="T0787" | V1=="T0762" | V1=="T0825" |
V1=="T0773" | V1=="T0801" | V1=="T0812"
| V1=="T0831" | V1=="T0760" | V1=="T0853" | V1=="T0815" | V1=="T0856" | V1=="T0788" |
V1=="T0805" | V1=="T0808" | V1=="T0835"
| V1=="T0843")
#4759 patterns/residues/rows in table
#nrow(training_set2) + nrow(testing_set2) = 14103

training_set3 <- subset( DT, V1!="T0819" & V1!="T0851" & V1!="T0790" & V1!="T0789" &
V1!="T0823" & V1!="T0813" & V1!="T0770" & V1!="T0803" & V1!="T0841"
& V1!="T0807" & V1!="T0848" & V1!="T0768" & V1!="T0785" & V1!="T0817" & V1!="T0838" &
V1!="T0797" & V1!="T0767" & V1!="T0780"
& V1!="T0837" & V1!="T0774" & V1!="T0786" & V1!="T0824" & V1!="T0814" & V1!="T0830" &
V1!="T0783" & V1!="T0849" & V1!="T0776"
& V1!="T0806")
#9345 patterns/residues/rows in table
testing_set3 <- subset( DT, V1=="T0819" | V1=="T0851" | V1=="T0790" | V1=="T0789" |
V1=="T0823" | V1=="T0813" | V1=="T0770" | V1=="T0803" | V1=="T0841"
| V1=="T0807" | V1=="T0848" | V1=="T0768" | V1=="T0785" | V1=="T0817" | V1=="T0838" |
V1=="T0797" | V1=="T0767" | V1=="T0780"
| V1=="T0837" | V1=="T0774" | V1=="T0786" | V1=="T0824" | V1=="T0814" | V1=="T0830" |
V1=="T0783" | V1=="T0849" | V1=="T0776"
| V1=="T0806")
#4758 patterns/residues/rows in table
#nrow(training_set3) + nrow(testing_set3) = 14103
#
# #randomise each training set
training_set1_ran <- training_set1[sample(1:nrow(training_set1), replace=FALSE),]
training_set1_inputs_ran <- training_set1_ran[, 3:12, with=FALSE]
training_set1_outputs_GDT-HA <- training_set1_ran[, 13, with=FALSE]
training_set1_outputs_GDT <- training_set1_ran[, 14, with=FALSE]
training_set1_outputs_MaxSub <- training_set1_ran[, 15, with=FALSE]
training_set1_outputs_TM-score <- training_set1_ran[, 16, with=FALSE]

testing_set1_inputs <- testing_set1[, 3:12, with=FALSE]
testing_set1_outputs_GDT-HA <- testing_set1[, 13, with=FALSE]
testing_set1_outputs_GDT <- testing_set1[, 14, with=FALSE]
testing_set1_outputs_MaxSub <- testing_set1[, 15, with=FALSE]
testing_set1_outputs_TM-score <- testing_set1[, 16, with=FALSE]

training_set2_ran <- training_set2[sample(1:nrow(training_set2), replace=FALSE),]
training_set2_inputs_ran <- training_set2_ran[, 3:12, with=FALSE]
training_set2_outputs_GDT-HA <- training_set2_ran[, 13, with=FALSE]
```

```
training_set2_outputs_GDT <- training_set2_ran[, 14, with=FALSE]
training_set2_outputs_MaxSub <- training_set2_ran[, 15, with=FALSE]
training_set2_outputs_TM-score <- training_set2_ran[, 16, with=FALSE]

testing_set2_inputs <- testing_set2[, 3:12, with=FALSE]
testing_set2_outputs_GDT-HA <- testing_set2[, 13, with=FALSE]
testing_set2_outputs_GDT <- testing_set2[, 14, with=FALSE]
testing_set2_outputs_MaxSub <- testing_set2[, 15, with=FALSE]
testing_set2_outputs_TM-score <- testing_set2[, 16, with=FALSE]

training_set3_ran <- training_set3[sample(1:nrow(training_set3), replace=FALSE),]
training_set3_inputs_ran <- training_set3_ran[, 3:12, with=FALSE]
training_set3_outputs_GDT-HA <- training_set3_ran[, 13, with=FALSE]
training_set3_outputs_GDT <- training_set3_ran[, 14, with=FALSE]
training_set3_outputs_MaxSub <- training_set3_ran[, 15, with=FALSE]
training_set3_outputs_TM-score <- training_set3_ran[, 16, with=FALSE]

testing_set3_inputs <- testing_set3[, 3:12, with=FALSE]
testing_set3_outputs_GDT-HA <- testing_set3[, 13, with=FALSE]
testing_set3_outputs_GDT <- testing_set3[, 14, with=FALSE]
testing_set3_outputs_MaxSub <- testing_set3[, 15, with=FALSE]
testing_set3_outputs_TM-score <- testing_set3[, 16, with=FALSE]

GDT-HA    <-    rbind(    testing_set1_outputs_GDT-HA,    testing_set2_outputs_GDT-HA,
testing_set3_outputs_GDT-HA)
GDT      <-      rbind(      testing_set1_outputs_GDT,      testing_set2_outputs_GDT,
testing_set3_outputs_GDT)
MaxSub    <-    rbind(    testing_set1_outputs_MaxSub,    testing_set2_outputs_MaxSub,
testing_set3_outputs_MaxSub)
TM-score   <-   rbind(   testing_set1_outputs_TM-score,   testing_set2_outputs_TM-score,
testing_set3_outputs_TM-score)

#try a NN with the ModFOLD6_rank combo of global score inputs
#target_id,  actualfilename,  ModFOLDclustscore,  ModFOLDclustQscore,  ModFOLDclust2,
ModFOLDclustres, ModFOLDclustQres, ProQ2res, CDAres, DBAres, SSAres, ModFOLD6res
#mean of ModFOLDclustQres+ProQ2res+CDAres+DBAres+SSAres+ModFOLD6res gives good top model
score (for each round and FM models) and reasonable correlations
cat(             "7_8_9_10_11_12-0_5_100it_3_hidden\n",             file             =
"Global_NN_both_rounds_correlations.dat",append = TRUE)
training_set1_inputs_ran <- training_set1_ran[, c(7,8,9,10,11,12), with=FALSE]
training_set2_inputs_ran <- training_set2_ran[, c(7,8,9,10,11,12), with=FALSE]
training_set3_inputs_ran <- training_set3_ran[, c(7,8,9,10,11,12), with=FALSE]
testing_set1_inputs <- testing_set1[, c(7,8,9,10,11,12), with=FALSE]
```

```
testing_set2_inputs <- testing_set2[, c(7,8,9,10,11,12), with=FALSE]
testing_set3_inputs <- testing_set3[, c(7,8,9,10,11,12), with=FALSE]

#train to GDT-HA score
model <- mlp(training_set1_inputs_ran, training_set1_outputs_GDT-HA, size = 3,
learnFuncParams = c(0.5, 0.01), maxit = 100, inputsTest = testing_set1_inputs,
targetsTest = testing_set1_outputs_GDT-HA)
save(model, file="Global_7_8_9_10_11_12-0_5_100it_3_hidden.model.train_window_set1")
predictions_set1 <- predict(model, testing_set1_inputs)

model <- mlp(training_set2_inputs_ran, training_set2_outputs_GDT-HA, size = 3,
learnFuncParams = c(0.5, 0.01), maxit = 100, inputsTest = testing_set2_inputs,
targetsTest = testing_set2_outputs_GDT-HA)
save(model, file="Global_7_8_9_10_11_12-0_5_100it_3_hidden.model.train_window_set2")
predictions_set2 <- predict(model, testing_set2_inputs)

model <- mlp(training_set3_inputs_ran, training_set3_outputs_GDT-HA, size = 3,
learnFuncParams = c(0.5, 0.01), maxit = 100, inputsTest = testing_set3_inputs,
targetsTest = testing_set3_outputs_GDT-HA)
save(model, file="Global_7_8_9_10_11_12-0_5_100it_3_hidden.model.train_window_set3")
predictions_set3 <- predict(model, testing_set3_inputs)

predictions <- rbind(predictions_set1, predictions_set2, predictions_set3)

#test correlations pred v obs
cat( "ModFOLD7_NN_test_GDT-HA",
cor(predictions, GDT-HA, method="pearson"), cor(predictions, GDT-HA, method="spearman"),
cor(predictions, GDT-HA, method="kendall"),
cor(predictions, GDT, method="pearson"), cor(predictions, GDT, method="spearman"),
cor(predictions, GDT, method="kendall"),
cor(predictions, MaxSub, method="pearson"), cor(predictions, MaxSub, method="spearman"),
cor(predictions, MaxSub, method="kendall"),
cor(predictions, TM-score, method="pearson"), cor(predictions, TM-score,
method="spearman"), cor(predictions, TM-score, method="kendall")
, "\n", sep=" ", file = "Global_NN_both_rounds_correlations.dat",append = TRUE)

#test ranking - cumulative scores of top ranked models
cat( "7_8_9_10_11_12-0_5_100it_3_hidden\n", file =
"Global_NN_both_rounds_ranks.dat",append = TRUE)

DT2 <- rbind( testing_set1, testing_set2, testing_set3)
DT2[,V17 := predictions ]#add predictions as last column (V17)
target_ids <- unique(DT2$V1)#get all IDs in data (unique variables in column $V1)
```

```
#setup empty arrays
NNtest <- c()

for(i in 1:length(target_ids) )
{
	#print(target_ids[i])
	set1 <-subset( DT2, V1==target_ids[i])

	#mean of ModFOLDclustQ_single_res_global_all, ProQ2_res_global_all, CDA_res_global_all, DBA_res_global_all, SSA_res_global_all and ModFOLD6_single_res_global_all #<--- 3rd BEST COMBO FOR RANKING
	NNtest <- rbind( NNtest, c( set1[which.max(set1$V17), ]$V1, set1[which.max(set1$V17), ]$V2, set1[which.max(set1$V17), ]$V13, set1[which.max(set1$V17), ]$V14, set1[which.max(set1$V17), ]$V15, set1[which.max(set1$V17), ]$V16 ))
}

#standard error function for error bars
std_err <- function(x) sd(x)/sqrt(length(x))

#cumlative GDT-HA, GDT-TS, MaxSub & TM-scores of top models for each target ranked by each global QA score
cumulativescores <- c()
cumulativescores <- rbind( cumulativescores, c( "Method", "GDT-HA", "GDT-TS", "MaxSub", "TM-score", "Std_err_GDT-HA", "Std_err_GDT-TS", "Std_err_MaxSub", "Std_err_TM-score" ))
cumulativescores <- rbind( cumulativescores, c( "NNtest", sum(as.numeric(NNtest[,3])), sum(as.numeric(NNtest[,4])), sum(as.numeric(NNtest[,5])), sum(as.numeric(NNtest[,6])), std_err(as.numeric(NNtest[,3])), std_err(as.numeric(NNtest[,4])), std_err(as.numeric(NNtest[,5])), std_err(as.numeric(NNtest[,6])) ))
#output table to a file
cat( "Round1+Round2\n", file = "Global_NN_both_rounds_ranks.dat", append = TRUE)
write.table( cumulativescores, file = "Global_NN_both_rounds_ranks.dat", sep = " ", quote = FALSE, row.names = FALSE, col.names = FALSE, append = TRUE)
```

Appendix 3

```
# DANNs_for_TensorFlow
import os
import tensorflow as tf
import numpy
import pandas as pd
sess = tf.InteractiveSession()

# Create a list containing the methods which are too be combined.
# Key: ModFOLD5_single_orig_global (3), ModFOLDclustQ_single_orig_global (4),
ModFOLDclust2_single_orig_global (5), ModFOLD5_single_res_global (6),
ModFOLDclustQ_single_res_global (7), ProQ2_res_global (8), CDA_res_global (9),
DBA_res_global (10), SSA_res_global (11), ModFOLD6_single_res_global (12).
combination_choice = ["V5", "V6", "V7", "V8", "V9", "V10"]

#Create a text file containing the wanted combination, this file is fed into R_Part1
file = open("combination.txt","w")
file.write("5,6,7,8,9,10")
file.close()

# Runs the R script, R_Part1.R through the terminal.
os.system("Rscript R_Part1.R")

# A function which aims to extract all the data produced from R_Part1 and stores them
in arrays to be used in the NN.
def run(set_num, combination, observation, learning_rate, training_epochs, n_hidden1,
n_hidden2, n_input):
    # Read files produced by R_Part1 and stores the data into a Data Frame.
    df_train = pd.read_csv("training_set%d_inputs_ran.csv" % set_num)
    df_GDT-HA = pd.read_csv("training_set%d_outputs_GDT-HA.csv" % set_num)
    df_test_inputs = pd.read_csv("testing_set%d_inputs.csv" % set_num)
    df_test_output = pd.read_csv("testing_set%d_outputs_GDT-HA.csv" % set_num)
    # Extracts the wanted data from the Data Frames above and converts the frame into a
Numpy-array.
    trainer = df_train.as_matrix(combination)
    label = df_GDT-HA.as_matrix(observation)
    test_inputs = df_test_inputs.as_matrix(combination)
    test_outputs = df_test_output.as_matrix(observation)
    h = my_mlp(set_num, trainer, label, learning_rate, training_epochs, n_hidden1,
n_hidden2, n_input, test_inputs, test_outputs)
    return h
```

```
def multilayer_perceptron(x, w1, w2, drop, out):
    # the first hidden layer
    layer_1 = tf.matmul(x, w1)
    layer_1 = tf.nn.dropout(layer_1, drop)
    # the second hidden layer
    layer_2 = tf.matmul(layer_1, w2)
    layer_2 = tf.nn.dropout(layer_2, drop)
    # Output layer with linear activation
    out_layer = tf.matmul(layer_2, out)
    return out_layer

def my_mlp (num, trainer, trainer_awn, learning_rate, training_epochs, n_hidden1,
n_hidden2, n_input, test_inputs, test_outputs):
    trX, trY= trainer, trainer_awn
    #create placeholders
    x = tf.placeholder(tf.float32, shape=[None, n_input])
    y_ = tf.placeholder(tf.float32, shape=[None, ])
    keep_prob = tf.placeholder("float")
    #create initial weights
    w1 = tf.Variable(tf.truncated_normal([n_input, n_hidden1], stddev=0.01))
    w2 = tf.Variable(tf.truncated_normal([n_hidden1, n_hidden2], stddev=0.01))
    out = tf.Variable(tf.truncated_normal([n_hidden2, 1], stddev=0.01))
    #predicted class and loss function
    y = multilayer_perceptron(x, w1, w2, keep_prob, out)
    # Reshapes the observational data.
    y_ = tf.reshape(y_, [-1, 1])
    # Cost function, aims to reduce the difference between the predictions and the
observational data.
    cross_entropy = tf.reduce_sum(tf.abs(y - y_))
    #training
    train_step =
tf.train.AdagradOptimizer(learning_rate=learning_rate).minimize(cross_entropy)
    correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
    init_op = tf.initialize_all_variables()
    saver = tf.train.Saver()
    # Start training.
    with tf.Session() as sess:
        # you need to initialize all variables
        sess.run(init_op)
      #training session, it is run multiple times equal to the set iterations/epochs.
        for i in range(training_epochs + 1):
```

```
          #feeds the training data, both combination data and observation data, into
the placeholders.
           sess.run([train_step, cross_entropy], feed_dict={x: trX, y_: trY,
keep_prob: 0.9})
        print("Accuracy:", accuracy.eval({x: test_inputs, y_: test_outputs, keep_prob:
1}))
       #Creates a Numpy array containing the final model predictions.
        best = sess.run(y, feed_dict={x: test_inputs, keep_prob: 1})
       #Saves the weights for each set seperatly.
        saver.save(sess,
'/home/filipe/Documents/Disseration/tensorflow/Data_searching/dropout/Rank/Model%d/mode
l' % num)
    return best

#train each data set to GDT-HA score (V13)
prediction1 = run(1, [combination_choice], ["V13"], 0.01, 550, 2, 3,
len(combination_choice))
numpy.savetxt('prediction_set1.out', prediction1)

prediction2 = run(2, [combination_choice], ["V13"], 0.01, 550, 2, 3,
len(combination_choice))
numpy.savetxt('prediction_set2.out', prediction2)

prediction3 = run(3, [combination_choice], ["V13"], 0.01, 550, 2, 3,
len(combination_choice))
numpy.savetxt('prediction_set3.out', prediction3)

# Runs the R script, R_Part2.R through the terminal.
os.system("Rscript R_Part2.R")
```

Appendix 4

```
# R_Part1
library(RSNNS)
library(data.table)

all1 <- fread("Global_QA_round1_all.out")
all2 <- fread("Global_QA_round2_all.out")
all <- rbind( all1, all2 )#combine data from both rounds
#remove data for where no native structures are available
DT <- subset( all, V1!="T0775" & V1!="T0779" & V1!="T0793" & V1!="T0795" & V1!="T0799"
& V1!="T0802" & V1!="T0804" & V1!="T0826" & V1!="T0828" & V1!="T0839" & V1!="T0842" &
V1!="T0844" & V1!="T0846" & V1!="T0850" )

training_set1 <- subset( DT, V1!="T0834" & V1!="T0798" & V1!="T0816" & V1!="T0845" &
V1!="T0822" & V1!="T0784" & V1!="T0833" & V1!="T0857" & V1!="T0763"
                         & V1!="T0782" & V1!="T0820" & V1!="T0854" & V1!="T0800" &
V1!="T0840" & V1!="T0832" & V1!="T0810" & V1!="T0827" & V1!="T0766"
                         & V1!="T0771" & V1!="T0858" & V1!="T0765" & V1!="T0855" &
V1!="T0847" & V1!="T0796" & V1!="T0778" & V1!="T0761" & V1!="T0764"
                         & V1!="T0821" )
#nrow(training_set1)
#9517 patterns/residues/rows in table
testing_set1 <-subset( DT, V1=="T0834" | V1=="T0798" | V1=="T0816" | V1=="T0845" |
V1=="T0822" | V1=="T0784" | V1=="T0833" | V1=="T0857" | V1=="T0763"
                       | V1=="T0782" | V1=="T0820" | V1=="T0854" | V1=="T0800" |
V1=="T0840" | V1=="T0832" | V1=="T0810" | V1=="T0827" | V1=="T0766"
                       | V1=="T0771" | V1=="T0858" | V1=="T0765" | V1=="T0855" |
V1=="T0847" | V1=="T0796" | V1=="T0778" | V1=="T0761" | V1=="T0764"
                       | V1=="T0821" )
#4586 patterns/residues/rows in table
#nrow(training_set1) + nrow(testing_set1) = 14103

training_set2 <- subset( DT, V1!="T0781" & V1!="T0829" & V1!="T0769" & V1!="T0836" &
V1!="T0759" & V1!="T0777" & V1!="T0852" & V1!="T0792" & V1!="T0818"
                         & V1!="T0772" & V1!="T0794" & V1!="T0811" & V1!="T0787" &
V1!="T0762" & V1!="T0825" & V1!="T0773" & V1!="T0801" & V1!="T0812"
                         & V1!="T0831" & V1!="T0760" & V1!="T0853" & V1!="T0815" &
V1!="T0856" & V1!="T0788" & V1!="T0805" & V1!="T0808" & V1!="T0835"
                         & V1!="T0843")
#9344 patterns/residues/rows in table
```

```
testing_set2 <- subset( DT, V1=="T0781" | V1=="T0829" | V1=="T0769" | V1=="T0836" |
V1=="T0759" | V1=="T0777" | V1=="T0852" | V1=="T0792" | V1=="T0818"
                        | V1=="T0772" | V1=="T0794" | V1=="T0811" | V1=="T0787" |
V1=="T0762" | V1=="T0825" | V1=="T0773" | V1=="T0801" | V1=="T0812"
                        | V1=="T0831" | V1=="T0760" | V1=="T0853" | V1=="T0815" |
V1=="T0856" | V1=="T0788" | V1=="T0805" | V1=="T0808" | V1=="T0835"
                        | V1=="T0843")
#4759 patterns/residues/rows in table
#nrow(training_set2) + nrow(testing_set2) = 14103

training_set3 <- subset( DT, V1!="T0819" & V1!="T0851" & V1!="T0790" & V1!="T0789" &
V1!="T0823" & V1!="T0813" & V1!="T0770" & V1!="T0803" & V1!="T0841"
                         & V1!="T0807" & V1!="T0848" & V1!="T0768" & V1!="T0785" &
V1!="T0817" & V1!="T0838" & V1!="T0797" & V1!="T0767" & V1!="T0780"
                         & V1!="T0837" & V1!="T0774" & V1!="T0786" & V1!="T0824" &
V1!="T0814" & V1!="T0830" & V1!="T0783" & V1!="T0849" & V1!="T0776"
                         & V1!="T0806")
#9345 patterns/residues/rows in table
testing_set3 <- subset( DT, V1=="T0819" | V1=="T0851" | V1=="T0790" | V1=="T0789" |
V1=="T0823" | V1=="T0813" | V1=="T0770" | V1=="T0803" | V1=="T0841"
                        | V1=="T0807" | V1=="T0848" | V1=="T0768" | V1=="T0785" |
V1=="T0817" | V1=="T0838" | V1=="T0797" | V1=="T0767" | V1=="T0780"
                        | V1=="T0837" | V1=="T0774" | V1=="T0786" | V1=="T0824" |
V1=="T0814" | V1=="T0830" | V1=="T0783" | V1=="T0849" | V1=="T0776"
                        | V1=="T0806")
#4758 patterns/residues/rows in table
#nrow(training_set3) + nrow(testing_set3) = 14103
#
# #randomise each training set
training_set1_ran <- training_set1[sample(1:nrow(training_set1), replace=FALSE),]
training_set1_inputs_ran <- training_set1_ran[, 3:12, with=FALSE]
training_set1_outputs_GDT-HA <- training_set1_ran[, 13, with=FALSE]
training_set1_outputs_GDT <- training_set1_ran[, 14, with=FALSE]
training_set1_outputs_MaxSub <- training_set1_ran[, 15, with=FALSE]
training_set1_outputs_TM-score <- training_set1_ran[, 16, with=FALSE]

testing_set1_inputs <- testing_set1[, 3:12, with=FALSE]
testing_set1_outputs_GDT-HA <- testing_set1[, 13, with=FALSE]
testing_set1_outputs_GDT <- testing_set1[, 14, with=FALSE]
testing_set1_outputs_MaxSub <- testing_set1[, 15, with=FALSE]
testing_set1_outputs_TM-score <- testing_set1[, 16, with=FALSE]

training_set2_ran <- training_set2[sample(1:nrow(training_set2), replace=FALSE),]
```

```
training_set2_inputs_ran <- training_set2_ran[, 3:12, with=FALSE]
training_set2_outputs_GDT-HA <- training_set2_ran[, 13, with=FALSE]
training_set2_outputs_GDT <- training_set2_ran[, 14, with=FALSE]
training_set2_outputs_MaxSub <- training_set2_ran[, 15, with=FALSE]
training_set2_outputs_TM-score <- training_set2_ran[, 16, with=FALSE]

testing_set2_inputs <- testing_set2[, 3:12, with=FALSE]
testing_set2_outputs_GDT-HA <- testing_set2[, 13, with=FALSE]
testing_set2_outputs_GDT <- testing_set2[, 14, with=FALSE]
testing_set2_outputs_MaxSub <- testing_set2[, 15, with=FALSE]
testing_set2_outputs_TM-score <- testing_set2[, 16, with=FALSE]

training_set3_ran <- training_set3[sample(1:nrow(training_set3), replace=FALSE),]
training_set3_inputs_ran <- training_set3_ran[, 3:12, with=FALSE]
training_set3_outputs_GDT-HA <- training_set3_ran[, 13, with=FALSE]
training_set3_outputs_GDT <- training_set3_ran[, 14, with=FALSE]
training_set3_outputs_MaxSub <- training_set3_ran[, 15, with=FALSE]
training_set3_outputs_TM-score <- training_set3_ran[, 16, with=FALSE]

testing_set3_inputs <- testing_set3[, 3:12, with=FALSE]
testing_set3_outputs_GDT-HA <- testing_set3[, 13, with=FALSE]
testing_set3_outputs_GDT <- testing_set3[, 14, with=FALSE]
testing_set3_outputs_MaxSub <- testing_set3[, 15, with=FALSE]
testing_set3_outputs_TM-score <- testing_set3[, 16, with=FALSE]

GDT-HA <- rbind( testing_set1_outputs_GDT-HA, testing_set2_outputs_GDT-HA,
testing_set3_outputs_GDT-HA)
GDT <- rbind( testing_set1_outputs_GDT, testing_set2_outputs_GDT,
testing_set3_outputs_GDT)
MaxSub <- rbind( testing_set1_outputs_MaxSub, testing_set2_outputs_MaxSub,
testing_set3_outputs_MaxSub)
TM-score <- rbind( testing_set1_outputs_TM-score, testing_set2_outputs_TM-score,
testing_set3_outputs_TM-score)

#Creates data sets containing the data for the specified combination
combination <- fread("combination.txt")
if (1 <= length(combination)){
  q = data.matrix(combination[[1]])
  if (2 <= length(combination)){
    w = data.matrix(combination[[2]])
    if (3 <= length(combination)){
      e = data.matrix(combination[[3]])
      if (4 <= length(combination)){
```

```
        r = data.matrix(combination[[4]])
        if (5 <= length(combination)){
          t = data.matrix(combination[[5]])
          if (6 <= length(combination)){
            y = data.matrix(combination[[6]])
            if (7 <= length(combination)){
              u = data.matrix(combination[[7]])
              if (8 <= length(combination)){
                i = data.matrix(combination[[8]])
                if (9 <= length(combination)){
                  o = data.matrix(combination[[9]])
                  if (10 <= length(combination)){
                    p = data.matrix(combination[[10]])
                    training_set1_inputs_ran <- training_set1_ran[, c(q, w, e, r, t, y,
u, i, o, p), with=FALSE]
                    training_set2_inputs_ran <- training_set2_ran[, c(q, w, e, r, t, y,
u, i, o, p), with=FALSE]
                    training_set3_inputs_ran <- training_set3_ran[, c(q, w, e, r, t, y,
u, i, o, p), with=FALSE]
                    testing_set1_inputs <- testing_set1[, c(q, w, e, r, t, y, u, i, o,
p), with=FALSE]
                    testing_set2_inputs <- testing_set2[, c(q, w, e, r, t, y, u, i, o,
p), with=FALSE]
                    testing_set3_inputs <- testing_set3[, c(q, w, e, r, t, y, u, i, o,
p), with=FALSE]
                  } else {
                    training_set1_inputs_ran <- training_set1_ran[, c(q, w, e, r, t, y,
u, i, o), with=FALSE]
                    training_set2_inputs_ran <- training_set2_ran[, c(q, w, e, r, t, y,
u, i, o), with=FALSE]
                    training_set3_inputs_ran <- training_set3_ran[, c(q, w, e, r, t, y,
u, i, o), with=FALSE]
                    testing_set1_inputs <- testing_set1[, c(q, w, e, r, t, y, u, i, o),
with=FALSE]
                    testing_set2_inputs <- testing_set2[, c(q, w, e, r, t, y, u, i, o),
with=FALSE]
                    testing_set3_inputs <- testing_set3[, c(q, w, e, r, t, y, u, i, o),
with=FALSE]
                  }
                } else{
                  training_set1_inputs_ran <- training_set1_ran[, c(q, w, e, r, t, y,
u, i), with=FALSE]
```

```
                  training_set2_inputs_ran <- training_set2_ran[, c(q, w, e, r, t, y,
u, i), with=FALSE]
                  training_set3_inputs_ran <- training_set3_ran[, c(q, w, e, r, t, y,
u, i), with=FALSE]
                  testing_set1_inputs <- testing_set1[, c(q, w, e, r, t, y, u, i),
with=FALSE]
                  testing_set2_inputs <- testing_set2[, c(q, w, e, r, t, y, u, i),
with=FALSE]
                  testing_set3_inputs <- testing_set3[, c(q, w, e, r, t, y, u, i),
with=FALSE]
                }
              } else {
                training_set1_inputs_ran <- training_set1_ran[, c(q, w, e, r, t, y, u),
with=FALSE]
                training_set2_inputs_ran <- training_set2_ran[, c(q, w, e, r, t, y, u),
with=FALSE]
                training_set3_inputs_ran <- training_set3_ran[, c(q, w, e, r, t, y, u),
with=FALSE]
                testing_set1_inputs <- testing_set1[, c(q, w, e, r, t, y, u),
with=FALSE]
                testing_set2_inputs <- testing_set2[, c(q, w, e, r, t, y, u),
with=FALSE]
                testing_set3_inputs <- testing_set3[, c(q, w, e, r, t, y, u),
with=FALSE]
              }
            } else {
              training_set1_inputs_ran <- training_set1_ran[, c(q, w, e, r, t, y),
with=FALSE]
              training_set2_inputs_ran <- training_set2_ran[, c(q, w, e, r, t, y),
with=FALSE]
              training_set3_inputs_ran <- training_set3_ran[, c(q, w, e, r, t, y),
with=FALSE]
              testing_set1_inputs <- testing_set1[, c(q, w, e, r, t, y), with=FALSE]
              testing_set2_inputs <- testing_set2[, c(q, w, e, r, t, y), with=FALSE]
              testing_set3_inputs <- testing_set3[, c(q, w, e, r, t, y), with=FALSE]
            }
          } else {
            training_set1_inputs_ran <- training_set1_ran[, c(q, w, e, r, t),
with=FALSE]
            training_set2_inputs_ran <- training_set2_ran[, c(q, w, e, r, t),
with=FALSE]
            training_set3_inputs_ran <- training_set3_ran[, c(q, w, e, r, t),
with=FALSE]
```

```
            testing_set1_inputs <- testing_set1[, c(q, w, e, r, t), with=FALSE]
            testing_set2_inputs <- testing_set2[, c(q, w, e, r, t), with=FALSE]
            testing_set3_inputs <- testing_set3[, c(q, w, e, r, t), with=FALSE]
          }
        } else {
          training_set1_inputs_ran <- training_set1_ran[, c(q, w, e, r), with=FALSE]
          training_set2_inputs_ran <- training_set2_ran[, c(q, w, e, r), with=FALSE]
          training_set3_inputs_ran <- training_set3_ran[, c(q, w, e, r), with=FALSE]
          testing_set1_inputs <- testing_set1[, c(q, w, e, r), with=FALSE]
          testing_set2_inputs <- testing_set2[, c(q, w, e, r), with=FALSE]
          testing_set3_inputs <- testing_set3[, c(q, w, e, r), with=FALSE]
        }
      } else {
        training_set1_inputs_ran <- training_set1_ran[, c(q, w, e), with=FALSE]
        training_set2_inputs_ran <- training_set2_ran[, c(q, w, e), with=FALSE]
        training_set3_inputs_ran <- training_set3_ran[, c(q, w, e), with=FALSE]
        testing_set1_inputs <- testing_set1[, c(q, w, e), with=FALSE]
        testing_set2_inputs <- testing_set2[, c(q, w, e), with=FALSE]
        testing_set3_inputs <- testing_set3[, c(q, w, e), with=FALSE]
      }
    } else {
      training_set1_inputs_ran <- training_set1_ran[, c(q, w), with=FALSE]
      training_set2_inputs_ran <- training_set2_ran[, c(q, w), with=FALSE]
      training_set3_inputs_ran <- training_set3_ran[, c(q, w), with=FALSE]
      testing_set1_inputs <- testing_set1[, c(q, w), with=FALSE]
      testing_set2_inputs <- testing_set2[, c(q, w), with=FALSE]
      testing_set3_inputs <- testing_set3[, c(q, w), with=FALSE]
    }
  } else {
    training_set1_inputs_ran <- training_set1_ran[, c(q), with=FALSE]
    training_set2_inputs_ran <- training_set2_ran[, c(q), with=FALSE]
    training_set3_inputs_ran <- training_set3_ran[, c(q), with=FALSE]
    testing_set1_inputs <- testing_set1[, c(q), with=FALSE]
    testing_set2_inputs <- testing_set2[, c(q), with=FALSE]
    testing_set3_inputs <- testing_set3[, c(q), with=FALSE]
  }
}

#Creates files containing all the data sets need to for the NN in python and also the
data needed for R_Part2
write.csv(training_set1_inputs_ran, file="training_set1_inputs_ran.csv")
write.csv(training_set2_inputs_ran, file="training_set2_inputs_ran.csv")
write.csv(training_set3_inputs_ran, file="training_set3_inputs_ran.csv")
```

```
write.csv(training_set1_outputs_GDT-HA, file="training_set1_outputs_GDT-HA.csv")
write.csv(training_set2_outputs_GDT-HA, file="training_set2_outputs_GDT-HA.csv")
write.csv(training_set3_outputs_GDT-HA, file="training_set3_outputs_GDT-HA.csv")
write.csv(testing_set1_inputs, file="testing_set1_inputs.csv")
write.csv(testing_set2_inputs, file="testing_set2_inputs.csv")
write.csv(testing_set3_inputs, file="testing_set3_inputs.csv")
write.csv(testing_set1_outputs_GDT-HA, file="testing_set1_outputs_GDT-HA.csv")
write.csv(testing_set2_outputs_GDT-HA, file="testing_set2_outputs_GDT-HA.csv")
write.csv(testing_set3_outputs_GDT-HA, file="testing_set3_outputs_GDT-HA.csv")
write.csv(GDT-HA, file="GDT-HA.csv")
write.csv(GDT, file="GDT.csv")
write.csv(MaxSub, file="MaxSub.csv")
write.csv(TM-score, file="TM-score.csv")
write.csv(testing_set1, file="testing_set1.csv")
write.csv(testing_set2, file="testing_set2.csv")
write.csv(testing_set3, file="testing_set3.csv")

#R_Part2
library(RSNNS)
library(data.table)

predictions_set1 <- fread("prediction_set1.out")
predictions_set1 <- data.matrix (predictions_set1)
predictions_set2 <- fread("prediction_set2.out")
predictions_set2 <- data.matrix (predictions_set2)
predictions_set3 <- fread("prediction_set3.out")
predictions_set3 <- data.matrix (predictions_set3)
GDT <- read.csv("GDT.csv")
GDT$X <- NULL
GDT-HA <- read.csv("GDT-HA.csv")
GDT-HA$X <- NULL
MaxSub <- read.csv("MaxSub.csv")
MaxSub$X <- NULL
TM-score <- read.csv("TM-score.csv")
TM-score$X <- NULL
testing_set1 <- read.csv("testing_set1.csv")
testing_set1$X <- NULL
testing_set2 <- read.csv("testing_set2.csv")
testing_set2$X <- NULL
testing_set3 <- read.csv("testing_set3.csv")
testing_set3$X <- NULL

#Combine all prediction data
```

```
predictions <- rbind(predictions_set1, predictions_set2, predictions_set3)

#test correlations pred v obs
cat(
     cor(predictions, GDT-HA, method="pearson"), cor(predictions, GDT-HA,
method="spearman"), cor(predictions, GDT-HA, method="kendall"),
     cor(predictions, GDT, method="pearson"), cor(predictions, GDT, method="spearman"),
cor(predictions, GDT, method="kendall"),
     cor(predictions, MaxSub, method="pearson"), cor(predictions, MaxSub,
method="spearman"), cor(predictions, MaxSub, method="kendall"),
     cor(predictions, TM-score, method="pearson"), cor(predictions, TM-score,
method="spearman"), cor(predictions, TM-score, method="kendall")
     , "\n", sep=" ", file = "Global_NN_both_rounds_correlations.dat",append = TRUE)

DT2 <- rbind( testing_set1, testing_set2, testing_set3)
DT2$V17 <- predictions #add predictions as last column (V17)
target_ids <- unique(DT2$V1)#get all IDs in data (unique variables in column $V1)

#setup empty arrays
NNtest <- c()

for(i in 1:length(target_ids) )
{
  #print(target_ids[i])
  set1 <-subset( DT2, V1==target_ids[i])

  #mean of ModFOLDclustQ_single_res_global_all, ProQ2_res_global_all,
CDA_res_global_all, DBA_res_global_all, SSA_res_global_all and
ModFOLD6_single_res_global_all #<--- 3rd BEST COMBO FOR RANKING
  NNtest <- rbind( NNtest, c( set1[which.max(set1$V17), ]$V1, set1[which.max(set1$V17),
]$V2, set1[which.max(set1$V17), ]$V13, set1[which.max(set1$V17), ]$V14,
set1[which.max(set1$V17), ]$V15, set1[which.max(set1$V17), ]$V16 ))
}

#standard error function for error bars
std_err <- function(x) sd(x)/sqrt(length(x))

#cumlative GDT-HA, GDT-TS, MaxSub & TM-scores of top models for each target ranked by
each global QA score
cumulativescores <- c()
cumulativescores <-  rbind( cumulativescores, c( sum(as.numeric(NNtest[,3])),
sum(as.numeric(NNtest[,4])), sum(as.numeric(NNtest[,5])), sum(as.numeric(NNtest[,6])),
```

```
std_err(as.numeric(NNtest[,3])), std_err(as.numeric(NNtest[,4])),
std_err(as.numeric(NNtest[,5])), std_err(as.numeric(NNtest[,6])) ))
#output table to a file
write.table( cumulativescores, file = "Global_NN_both_rounds_ranks.dat", sep = " ",
quote = FALSE, row.names = FALSE, col.names = FALSE, append = TRUE)
```

Appendix 5

```
# RSNNS_Para
library(RSNNS)
library(data.table)
all1 <- fread("Global_QA_round1_all.out")
all2 <- fread("Global_QA_round2_all.out")
all <- rbind( all1, all2 )#combine data from both rounds
#remove data for where no native structures are available
DT <- subset( all, V1!="T0775" & V1!="T0779" & V1!="T0793" & V1!="T0795" & V1!="T0799"
& V1!="T0802" & V1!="T0804" & V1!="T0826" & V1!="T0828" & V1!="T0839" & V1!="T0842" &
V1!="T0844" & V1!="T0846" & V1!="T0850" )

#seperate training and testing data into 3 subsets.
training_set1 <- subset( DT, V1!="T0834" & V1!="T0798" & V1!="T0816" & V1!="T0845" &
V1!="T0822" & V1!="T0784" & V1!="T0833" & V1!="T0857" & V1!="T0763"
                         & V1!="T0782" & V1!="T0820" & V1!="T0854" & V1!="T0800" &
V1!="T0840" & V1!="T0832" & V1!="T0810" & V1!="T0827" & V1!="T0766"
                         & V1!="T0771" & V1!="T0858" & V1!="T0765" & V1!="T0855" &
V1!="T0847" & V1!="T0796" & V1!="T0778" & V1!="T0761" & V1!="T0764"
                         & V1!="T0821" )
#nrow(training_set1)
#9517 patterns/residues/rows in table
testing_set1 <-subset( DT, V1=="T0834" | V1=="T0798" | V1=="T0816" | V1=="T0845" |
V1=="T0822" | V1=="T0784" | V1=="T0833" | V1=="T0857" | V1=="T0763"
                       | V1=="T0782" | V1=="T0820" | V1=="T0854" | V1=="T0800" |
V1=="T0840" | V1=="T0832" | V1=="T0810" | V1=="T0827" | V1=="T0766"
                       | V1=="T0771" | V1=="T0858" | V1=="T0765" | V1=="T0855" |
V1=="T0847" | V1=="T0796" | V1=="T0778" | V1=="T0761" | V1=="T0764"
                       | V1=="T0821" )
#4586 patterns/residues/rows in table
#nrow(training_set1) + nrow(testing_set1) = 14103

training_set2 <- subset( DT, V1!="T0781" & V1!="T0829" & V1!="T0769" & V1!="T0836" &
V1!="T0759" & V1!="T0777" & V1!="T0852" & V1!="T0792" & V1!="T0818"
                         & V1!="T0772" & V1!="T0794" & V1!="T0811" & V1!="T0787" &
V1!="T0762" & V1!="T0825" & V1!="T0773" & V1!="T0801" & V1!="T0812"
                         & V1!="T0831" & V1!="T0760" & V1!="T0853" & V1!="T0815" &
V1!="T0856" & V1!="T0788" & V1!="T0805" & V1!="T0808" & V1!="T0835"
                         & V1!="T0843")
#9344 patterns/residues/rows in table
```

```
testing_set2 <- subset( DT, V1=="T0781" | V1=="T0829" | V1=="T0769" | V1=="T0836" |
V1=="T0759" | V1=="T0777" | V1=="T0852" | V1=="T0792" | V1=="T0818"
                        | V1=="T0772" | V1=="T0794" | V1=="T0811" | V1=="T0787" |
V1=="T0762" | V1=="T0825" | V1=="T0773" | V1=="T0801" | V1=="T0812"
                        | V1=="T0831" | V1=="T0760" | V1=="T0853" | V1=="T0815" |
V1=="T0856" | V1=="T0788" | V1=="T0805" | V1=="T0808" | V1=="T0835"
                        | V1=="T0843")
#4759 patterns/residues/rows in table
#nrow(training_set2) + nrow(testing_set2) = 14103

training_set3 <- subset( DT, V1!="T0819" & V1!="T0851" & V1!="T0790" & V1!="T0789" &
V1!="T0823" & V1!="T0813" & V1!="T0770" & V1!="T0803" & V1!="T0841"
                         & V1!="T0807" & V1!="T0848" & V1!="T0768" & V1!="T0785" &
V1!="T0817" & V1!="T0838" & V1!="T0797" & V1!="T0767" & V1!="T0780"
                         & V1!="T0837" & V1!="T0774" & V1!="T0786" & V1!="T0824" &
V1!="T0814" & V1!="T0830" & V1!="T0783" & V1!="T0849" & V1!="T0776"
                         & V1!="T0806")
#9345 patterns/residues/rows in table
testing_set3 <- subset( DT, V1=="T0819" | V1=="T0851" | V1=="T0790" | V1=="T0789" |
V1=="T0823" | V1=="T0813" | V1=="T0770" | V1=="T0803" | V1=="T0841"
                        | V1=="T0807" | V1=="T0848" | V1=="T0768" | V1=="T0785" |
V1=="T0817" | V1=="T0838" | V1=="T0797" | V1=="T0767" | V1=="T0780"
                        | V1=="T0837" | V1=="T0774" | V1=="T0786" | V1=="T0824" |
V1=="T0814" | V1=="T0830" | V1=="T0783" | V1=="T0849" | V1=="T0776"
                        | V1=="T0806")
#4758 patterns/residues/rows in table
#nrow(training_set3) + nrow(testing_set3) = 14103
#
# #randomise each training set
training_set1_ran <- training_set1[sample(1:nrow(training_set1), replace=FALSE),]
training_set1_inputs_ran <- training_set1_ran[, 3:12, with=FALSE]
training_set1_outputs_GDT-HA <- training_set1_ran[, 13, with=FALSE]
training_set1_outputs_GDT <- training_set1_ran[, 14, with=FALSE]
training_set1_outputs_MaxSub <- training_set1_ran[, 15, with=FALSE]
training_set1_outputs_TM-score <- training_set1_ran[, 16, with=FALSE]

testing_set1_inputs <- testing_set1[, 3:12, with=FALSE]
testing_set1_outputs_GDT-HA <- testing_set1[, 13, with=FALSE]
testing_set1_outputs_GDT <- testing_set1[, 14, with=FALSE]
testing_set1_outputs_MaxSub <- testing_set1[, 15, with=FALSE]
testing_set1_outputs_TM-score <- testing_set1[, 16, with=FALSE]

training_set2_ran <- training_set2[sample(1:nrow(training_set2), replace=FALSE),]
```

```
training_set2_inputs_ran <- training_set2_ran[, 3:12, with=FALSE]
training_set2_outputs_GDT-HA <- training_set2_ran[, 13, with=FALSE]
training_set2_outputs_GDT <- training_set2_ran[, 14, with=FALSE]
training_set2_outputs_MaxSub <- training_set2_ran[, 15, with=FALSE]
training_set2_outputs_TM-score <- training_set2_ran[, 16, with=FALSE]

testing_set2_inputs <- testing_set2[, 3:12, with=FALSE]
testing_set2_outputs_GDT-HA <- testing_set2[, 13, with=FALSE]
testing_set2_outputs_GDT <- testing_set2[, 14, with=FALSE]
testing_set2_outputs_MaxSub <- testing_set2[, 15, with=FALSE]
testing_set2_outputs_TM-score <- testing_set2[, 16, with=FALSE]

training_set3_ran <- training_set3[sample(1:nrow(training_set3), replace=FALSE),]
training_set3_inputs_ran <- training_set3_ran[, 3:12, with=FALSE]
training_set3_outputs_GDT-HA <- training_set3_ran[, 13, with=FALSE]
training_set3_outputs_GDT <- training_set3_ran[, 14, with=FALSE]
training_set3_outputs_MaxSub <- training_set3_ran[, 15, with=FALSE]
training_set3_outputs_TM-score <- training_set3_ran[, 16, with=FALSE]

testing_set3_inputs <- testing_set3[, 3:12, with=FALSE]
testing_set3_outputs_GDT-HA <- testing_set3[, 13, with=FALSE]
testing_set3_outputs_GDT <- testing_set3[, 14, with=FALSE]
testing_set3_outputs_MaxSub <- testing_set3[, 15, with=FALSE]
testing_set3_outputs_TM-score <- testing_set3[, 16, with=FALSE]

GDT-HA <- rbind( testing_set1_outputs_GDT-HA, testing_set2_outputs_GDT-HA,
testing_set3_outputs_GDT-HA)
GDT <- rbind( testing_set1_outputs_GDT, testing_set2_outputs_GDT,
testing_set3_outputs_GDT)
MaxSub <- rbind( testing_set1_outputs_MaxSub, testing_set2_outputs_MaxSub,
testing_set3_outputs_MaxSub)
TM-score <- rbind( testing_set1_outputs_TM-score, testing_set2_outputs_TM-score,
testing_set3_outputs_TM-score)

#loops NN over all possible combinations of parameters, hidden units from 1-20 and
hidden units from 50-950
t = seq(1, 20, 1)
f = seq(50, 950, 50)
n = 1
while(n <= length(t)){
  r = t[n]
  h = 1
  n = n +1
```

```
  while(h <= length(f)){
    c = f[h]
    h = h + 1
    print(c(r,c))
    #try a NN with the ModFOLD6_rank combo of global score inputs
    #target_id, actualfilename, ModFOLDclustscore, ModFOLDclustQscore, ModFOLDclust2,
ModFOLDclustres, ModFOLDclustQres, ProQ2res, CDAres, DBAres, SSAres, ModFOLD6res
    #mean of ModFOLDclustQres+ProQ2res+CDAres+DBAres+SSAres+ModFOLD6res gives good top
model score (for each round and FM models) and reasonable correlations
    #cat( "3,5,9,10,12-0_5_100it_3_hidden\n", file =
"Global_NN_both_rounds_correlations.dat",append = TRUE)
    training_set1_inputs_ran <- training_set1_ran[, c(3,5,9,10,12), with=FALSE]
    training_set2_inputs_ran <- training_set2_ran[, c(3,5,9,10,12), with=FALSE]
    training_set3_inputs_ran <- training_set3_ran[, c(3,5,9,10,12), with=FALSE]
    testing_set1_inputs <- testing_set1[, c(3,5,9,10,12), with=FALSE]
    testing_set2_inputs <- testing_set2[, c(3,5,9,10,12), with=FALSE]
    testing_set3_inputs <- testing_set3[, c(3,5,9,10,12), with=FALSE]

    set1 = paste0("Global_-
0_5_3,5,9,10,12_",c,"it_",r,"_hidden.model.train_window_set1")
    #train to GDT-HA score
    model <- mlp(training_set1_inputs_ran, training_set1_outputs_GDT-HA, size = r,
learnFuncParams = c(0.5, 0.01), maxit = c, inputsTest = testing_set1_inputs,
targetsTest = testing_set1_outputs_GDT-HA)
    save(model, file = set1)
    predictions_set1 <- predict(model, testing_set1_inputs)

    set2 = paste0("Global_-
0_5_3,5,9,10,12_",c,"it_",r,"_hidden.model.train_window_set2")
    model <- mlp(training_set2_inputs_ran, training_set2_outputs_GDT-HA, size = r,
learnFuncParams = c(0.5, 0.01), maxit = c, inputsTest = testing_set2_inputs,
targetsTest = testing_set2_outputs_GDT-HA)
    save(model, file= set2)
    predictions_set2 <- predict(model, testing_set2_inputs)

    set3 = paste0("Global_-
0_5_3,5,9,10,12_",c,"it_",r,"_hidden.model.train_window_set3")
    model <- mlp(training_set3_inputs_ran, training_set3_outputs_GDT-HA, size = r,
learnFuncParams = c(0.5, 0.01), maxit = c, inputsTest = testing_set3_inputs,
targetsTest = testing_set3_outputs_GDT-HA)
    save(model, file= set3)
    predictions_set3 <- predict(model, testing_set3_inputs)
```

```
    predictions <- rbind(predictions_set1, predictions_set2, predictions_set3)

    #test correlations pred v obs
    cat( r, c, cor(predictions, GDT-HA, method="pearson"), cor(predictions, GDT-HA,
method="spearman"), cor(predictions, GDT-HA, method="kendall"),
         cor(predictions, GDT, method="pearson"), cor(predictions, GDT,
method="spearman"), cor(predictions, GDT, method="kendall"),
         cor(predictions, MaxSub, method="pearson"), cor(predictions, MaxSub,
method="spearman"), cor(predictions, MaxSub, method="kendall"),
         cor(predictions, TM-score, method="pearson"), cor(predictions, TM-score,
method="spearman"), cor(predictions, TM-score, method="kendall")
         , "\n", sep=" ", file = "Global_NN_both_rounds_correlations.dat",append =
TRUE)

    #test ranking - cumulative scores of top ranked models
    #doh = paste0("3,5,9,10,12-0_5_",c,"it_",r,"_hidden\n")
    #cat( doh, file = "Global_NN_both_rounds_ranks.dat",append = TRUE)

    DT2 <- rbind( testing_set1, testing_set2, testing_set3)
    DT2[,V17 := predictions ]#add predictions as last column (V17)
    target_ids <- unique(DT2$V1)#get all IDs in data (unique variables in column $V1)

    #setup empty arrays
    NNtest <- c()

    for(i in 1:length(target_ids) )
    {
      #print(target_ids[i])
      set1 <-subset( DT2, V1==target_ids[i])

      #mean of ModFOLDclustQ_single_res_global_all, ProQ2_res_global_all,
CDA_res_global_all, DBA_res_global_all, SSA_res_global_all and
ModFOLD6_single_res_global_all #<--- 3rd BEST COMBO FOR RANKING
      NNtest <- rbind( NNtest, c(set1[which.max(set1$V17), ]$V1,
set1[which.max(set1$V17), ]$V2, set1[which.max(set1$V17), ]$V13,
set1[which.max(set1$V17), ]$V14, set1[which.max(set1$V17), ]$V15,
set1[which.max(set1$V17), ]$V16 ))
    }

    #standard error function for error bars
    std_err <- function(x) sd(x)/sqrt(length(x))
```

```
    #cumlative GDT-HA, GDT-TS, MaxSub & TM-scores of top models for each target ranked
by each global QA score
    cumulativescores <- c()
    cumulativescores <-  rbind( cumulativescores, c( c, r, sum(as.numeric(NNtest[,3])),
sum(as.numeric(NNtest[,4])), sum(as.numeric(NNtest[,5])), sum(as.numeric(NNtest[,6])),
std_err(as.numeric(NNtest[,3])), std_err(as.numeric(NNtest[,4])),
std_err(as.numeric(NNtest[,5])), std_err(as.numeric(NNtest[,6])) ))
    #output table to a file
    #cat( "Round1+Round2\n", file = "Global_NN_both_rounds_ranks.dat", append = TRUE)
    write.table( cumulativescores, file = "Global_NN_both_rounds_ranks.dat", sep = " ",
quote = FALSE, row.names = FALSE, col.names = FALSE, append = TRUE)
  }
}

#Tensorflow_Para

import os
import tensorflow as tf
import numpy
import pandas as pd
sess = tf.InteractiveSession()

# A function which aims to extract all the data produced from R_Part1 and stores them
in arrays to be used in the NN.
def run(set_num, combination, observation, learning_rate, training_epochs, n_hidden1,
n_hidden2, n_input):
       # Read files produced by R_Part1 and stores the data into a Data Frame.
        df_train = pd.read_csv("training_set%d_inputs_ran.csv" % set_num)
        df_GDT-HA = pd.read_csv("training_set%d_outputs_GDT-HA.csv" % set_num)
        df_test_inputs = pd.read_csv("testing_set%d_inputs.csv" % set_num)
        df_test_output = pd.read_csv("testing_set%d_outputs_GDT-HA.csv" % set_num)
       # Extracts the wanted data from the Data Frames above and converts the frame
into a Numpy-array.
        trainer = df_train.as_matrix(combination)
        label = df_GDT-HA.as_matrix(observation)
        test_inputs = df_test_inputs.as_matrix(combination)
        test_outputs = df_test_output.as_matrix(observation)
        h = my_mlp(set_num, trainer, label, learning_rate, training_epochs, n_hidden1,
n_hidden2, n_input, test_inputs, test_outputs)
        return h

def multilayer_perceptron(x, w1, w2, drop, out):
    # the first hidden layer
```

```
    layer_1 = tf.matmul(x, w1)
    layer_1 = tf.nn.dropout(layer_1, drop)
    # the second hidden layer
    layer_2 = tf.matmul(layer_1, w2)
    layer_2 = tf.nn.dropout(layer_2, drop)
    # Output layer with linear activation
    out_layer = tf.matmul(layer_2, out)
    return out_layer

def my_mlp (num, trainer, trainer_awn, learning_rate, training_epochs, n_hidden1,
n_hidden2, n_input, test_inputs, test_outputs):
    trX, trY= trainer, trainer_awn
    #create placeholders
    x = tf.placeholder(tf.float32, shape=[None, n_input])
    y_ = tf.placeholder(tf.float32, shape=[None, ])
    keep_prob = tf.placeholder("float")
    #create initial weights
    w1 = tf.Variable(tf.truncated_normal([n_input, n_hidden1], stddev=0.01))
    w2 = tf.Variable(tf.truncated_normal([n_hidden1, n_hidden2], stddev=0.01))
    out = tf.Variable(tf.truncated_normal([n_hidden2, 1], stddev=0.01))
    #predicted class and loss function
    y = multilayer_perceptron(x, w1, w2, keep_prob, out)
    # Reshapes the observational data.
    y_ = tf.reshape(y_, [-1, 1])
    # Cost function, aims to reduce the difference between the predictions and the
observational data.
    cross_entropy = tf.reduce_sum(tf.abs(y - y_))
    #training
    train_step =
tf.train.AdagradOptimizer(learning_rate=learning_rate).minimize(cross_entropy)
    correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
    init_op = tf.initialize_all_variables()
    saver = tf.train.Saver()
    # Start training.
    with tf.Session() as sess:
        # you need to initialize all variables
        sess.run(init_op)
       #training session, it is run multiple times equal to the set iterations/epochs.
        for i in range(training_epochs + 1):
           #feeds the training data, both combination data and observation data, into
the placeholders.
```

```
            sess.run([train_step, cross_entropy], feed_dict={x: trX, y_: trY,
keep_prob: 0.9})
        print("Accuracy:", accuracy.eval({x: test_inputs, y_: test_outputs, keep_prob:
1}))
       #Creates a Numpy array containing the final model predictions.
        best = sess.run(y, feed_dict={x: test_inputs, keep_prob: 1})
       #Saves the weights for each set seperatly.
        saver.save(sess,
'/home/filipe/Documents/Disseration/tensorflow/Data_searching/dropout/Rank/Model%d/mode
l' % num)
    return best

# Create a list containing the methods which are too be combined.
# Key: ModFOLD5_single_orig_global (3), ModFOLDclustQ_single_orig_global (4),
ModFOLDclust2_single_orig_global (5), ModFOLD5_single_res_global (6),
ModFOLDclustQ_single_res_global (7), ProQ2_res_global (8), CDA_res_global (9),
DBA_res_global (10), SSA_res_global (11), ModFOLD6_single_res_global (12).
combination_choice = ["V9", "V11", "V12"]

#Create a text file containing the wanted combination, this file is fed into R_Part1
file = open("combination.txt","w")
file.write("9, 11, 12")
file.close()

# Runs the R script, R_Part1.R through the terminal.
os.system("Rscript Data_searching_Part1.R")

#loops NN over all possible combinations of parameters, hidden units from 1-20 and
hidden units from 50-950
for inter in range(50, 1000, 50):
    for hidden1 in range(1, 6, 1):
        for hidden2 in range(1, 6, 1):
           #train each data set to GDT-HA score (V13)
            prediction1 = run(1, [combination_choice], ["V13"], 0.01, inter, hidden1,
hidden2, len(combination_choice))
            numpy.savetxt('prediction_set1.out', prediction1)

            prediction2 = run(2, [combination_choice], ["V13"], 0.01, inter, hidden1,
hidden2, len(combination_choice))
            numpy.savetxt('prediction_set2.out', prediction2)

            prediction3 = run(3, [combination_choice], ["V13"], 0.01, inter, hidden1,
hidden2, len(combination_choice))
```

```
 numpy.savetxt('prediction_set3.out', prediction3)

# Runs the R script, R_Part2.R through the terminal.
 os.system("Rscript Para_part2.R")
```

Appendix 6

```
# RSNNS_Data_search

library(RSNNS)
library(data.table)
all1 <- fread("Global_QA_round1_all.out")
all2 <- fread("Global_QA_round2_all.out")
all <- rbind( all1, all2 )#combine data from both rounds
#remove data for where no native structures are available
DT <- subset( all, V1!="T0775" & V1!="T0779" & V1!="T0793" & V1!="T0795" & V1!="T0799"
& V1!="T0802" & V1!="T0804" & V1!="T0826" & V1!="T0828" & V1!="T0839" & V1!="T0842" &
V1!="T0844" & V1!="T0846" & V1!="T0850" )

#seperate training and testing data into 3 subsets.
training_set1 <- subset( DT, V1!="T0834" & V1!="T0798" & V1!="T0816" & V1!="T0845" &
V1!="T0822" & V1!="T0784" & V1!="T0833" & V1!="T0857" & V1!="T0763"
                         & V1!="T0782" & V1!="T0820" & V1!="T0854" & V1!="T0800" &
V1!="T0840" & V1!="T0832" & V1!="T0810" & V1!="T0827" & V1!="T0766"
                         & V1!="T0771" & V1!="T0858" & V1!="T0765" & V1!="T0855" &
V1!="T0847" & V1!="T0796" & V1!="T0778" & V1!="T0761" & V1!="T0764"
                         & V1!="T0821" )
#nrow(training_set1)
#9517 patterns/residues/rows in table
testing_set1 <-subset( DT, V1=="T0834" | V1=="T0798" | V1=="T0816" | V1=="T0845" |
V1=="T0822" | V1=="T0784" | V1=="T0833" | V1=="T0857" | V1=="T0763"
                       | V1=="T0782" | V1=="T0820" | V1=="T0854" | V1=="T0800" |
V1=="T0840" | V1=="T0832" | V1=="T0810" | V1=="T0827" | V1=="T0766"
                       | V1=="T0771" | V1=="T0858" | V1=="T0765" | V1=="T0855" |
V1=="T0847" | V1=="T0796" | V1=="T0778" | V1=="T0761" | V1=="T0764"
                       | V1=="T0821" )
#4586 patterns/residues/rows in table
#nrow(training_set1) + nrow(testing_set1) = 14103

training_set2 <- subset( DT, V1!="T0781" & V1!="T0829" & V1!="T0769" & V1!="T0836" &
V1!="T0759" & V1!="T0777" & V1!="T0852" & V1!="T0792" & V1!="T0818"
                         & V1!="T0772" & V1!="T0794" & V1!="T0811" & V1!="T0787" &
V1!="T0762" & V1!="T0825" & V1!="T0773" & V1!="T0801" & V1!="T0812"
                         & V1!="T0831" & V1!="T0760" & V1!="T0853" & V1!="T0815" &
V1!="T0856" & V1!="T0788" & V1!="T0805" & V1!="T0808" & V1!="T0835"
                         & V1!="T0843")
#9344 patterns/residues/rows in table
```

```
testing_set2 <- subset( DT, V1=="T0781" | V1=="T0829" | V1=="T0769" | V1=="T0836" |
V1=="T0759" | V1=="T0777" | V1=="T0852" | V1=="T0792" | V1=="T0818"
                        | V1=="T0772" | V1=="T0794" | V1=="T0811" | V1=="T0787" |
V1=="T0762" | V1=="T0825" | V1=="T0773" | V1=="T0801" | V1=="T0812"
                        | V1=="T0831" | V1=="T0760" | V1=="T0853" | V1=="T0815" |
V1=="T0856" | V1=="T0788" | V1=="T0805" | V1=="T0808" | V1=="T0835"
                        | V1=="T0843")
#4759 patterns/residues/rows in table
#nrow(training_set2) + nrow(testing_set2) = 14103

training_set3 <- subset( DT, V1!="T0819" & V1!="T0851" & V1!="T0790" & V1!="T0789" &
V1!="T0823" & V1!="T0813" & V1!="T0770" & V1!="T0803" & V1!="T0841"
                         & V1!="T0807" & V1!="T0848" & V1!="T0768" & V1!="T0785" &
V1!="T0817" & V1!="T0838" & V1!="T0797" & V1!="T0767" & V1!="T0780"
                         & V1!="T0837" & V1!="T0774" & V1!="T0786" & V1!="T0824" &
V1!="T0814" & V1!="T0830" & V1!="T0783" & V1!="T0849" & V1!="T0776"
                         & V1!="T0806")
#9345 patterns/residues/rows in table
testing_set3 <- subset( DT, V1=="T0819" | V1=="T0851" | V1=="T0790" | V1=="T0789" |
V1=="T0823" | V1=="T0813" | V1=="T0770" | V1=="T0803" | V1=="T0841"
                        | V1=="T0807" | V1=="T0848" | V1=="T0768" | V1=="T0785" |
V1=="T0817" | V1=="T0838" | V1=="T0797" | V1=="T0767" | V1=="T0780"
                        | V1=="T0837" | V1=="T0774" | V1=="T0786" | V1=="T0824" |
V1=="T0814" | V1=="T0830" | V1=="T0783" | V1=="T0849" | V1=="T0776"
                        | V1=="T0806")
#4758 patterns/residues/rows in table
#nrow(training_set3) + nrow(testing_set3) = 14103
#
# #randomise each training set
training_set1_ran <- training_set1[sample(1:nrow(training_set1), replace=FALSE),]
training_set1_inputs_ran <- training_set1_ran[, 3:12, with=FALSE]
training_set1_outputs_GDT-HA <- training_set1_ran[, 13, with=FALSE]
training_set1_outputs_GDT <- training_set1_ran[, 14, with=FALSE]
training_set1_outputs_MaxSub <- training_set1_ran[, 15, with=FALSE]
training_set1_outputs_TM-score <- training_set1_ran[, 16, with=FALSE]

testing_set1_inputs <- testing_set1[, 3:12, with=FALSE]
testing_set1_outputs_GDT-HA <- testing_set1[, 13, with=FALSE]
testing_set1_outputs_GDT <- testing_set1[, 14, with=FALSE]
testing_set1_outputs_MaxSub <- testing_set1[, 15, with=FALSE]
testing_set1_outputs_TM-score <- testing_set1[, 16, with=FALSE]

training_set2_ran <- training_set2[sample(1:nrow(training_set2), replace=FALSE),]
```

```
training_set2_inputs_ran <- training_set2_ran[, 3:12, with=FALSE]
training_set2_outputs_GDT-HA <- training_set2_ran[, 13, with=FALSE]
training_set2_outputs_GDT <- training_set2_ran[, 14, with=FALSE]
training_set2_outputs_MaxSub <- training_set2_ran[, 15, with=FALSE]
training_set2_outputs_TM-score <- training_set2_ran[, 16, with=FALSE]

testing_set2_inputs <- testing_set2[, 3:12, with=FALSE]
testing_set2_outputs_GDT-HA <- testing_set2[, 13, with=FALSE]
testing_set2_outputs_GDT <- testing_set2[, 14, with=FALSE]
testing_set2_outputs_MaxSub <- testing_set2[, 15, with=FALSE]
testing_set2_outputs_TM-score <- testing_set2[, 16, with=FALSE]

training_set3_ran <- training_set3[sample(1:nrow(training_set3), replace=FALSE),]
training_set3_inputs_ran <- training_set3_ran[, 3:12, with=FALSE]
training_set3_outputs_GDT-HA <- training_set3_ran[, 13, with=FALSE]
training_set3_outputs_GDT <- training_set3_ran[, 14, with=FALSE]
training_set3_outputs_MaxSub <- training_set3_ran[, 15, with=FALSE]
training_set3_outputs_TM-score <- training_set3_ran[, 16, with=FALSE]

testing_set3_inputs <- testing_set3[, 3:12, with=FALSE]
testing_set3_outputs_GDT-HA <- testing_set3[, 13, with=FALSE]
testing_set3_outputs_GDT <- testing_set3[, 14, with=FALSE]
testing_set3_outputs_MaxSub <- testing_set3[, 15, with=FALSE]
testing_set3_outputs_TM-score <- testing_set3[, 16, with=FALSE]

GDT-HA <- rbind( testing_set1_outputs_GDT-HA, testing_set2_outputs_GDT-HA,
testing_set3_outputs_GDT-HA)
GDT <- rbind( testing_set1_outputs_GDT, testing_set2_outputs_GDT,
testing_set3_outputs_GDT)
MaxSub <- rbind( testing_set1_outputs_MaxSub, testing_set2_outputs_MaxSub,
testing_set3_outputs_MaxSub)
TM-score <- rbind( testing_set1_outputs_TM-score, testing_set2_outputs_TM-score,
testing_set3_outputs_TM-score)

#loops over the NN as until score is higher than the score specified
while(sum(as.numeric(NNtest[,3])) < 32.2){
  #try a NN with the ModFOLD6_rank combo of global score inputs
  #target_id, actualfilename, ModFOLDclustscore, ModFOLDclustQscore, ModFOLDclust2,
ModFOLDclustres, ModFOLDclustQres, ProQ2res, CDAres, DBAres, SSAres, ModFOLD6res
  #mean of ModFOLDclustQres+ProQ2res+CDAres+DBAres+SSAres+ModFOLD6res gives good top
model score (for each round and FM models) and reasonable correlations
  cat( "8,9,11-0_5_100it_3_hidden\n", file =
"Global_NN_both_rounds_correlations.dat",append = TRUE)
```

```
  #create variables containing a matrix of only the methods included in the combination
  training_set1_inputs_ran <- training_set1_ran[, c(8,9,11), with=FALSE]
  training_set2_inputs_ran <- training_set2_ran[, c(8,9,11), with=FALSE]
  training_set3_inputs_ran <- training_set3_ran[, c(8,9,11), with=FALSE]
  testing_set1_inputs <- testing_set1[, c(8,9,11), with=FALSE]
  testing_set2_inputs <- testing_set2[, c(8,9,11), with=FALSE]
  testing_set3_inputs <- testing_set3[, c(8,9,11), with=FALSE]

  #train to GDT-HA score
  model <- mlp(training_set1_inputs_ran, training_set1_outputs_GDT-HA, size = 2,
learnFuncParams = c(0.5, 0.001), maxit = 550, inputsTest = testing_set1_inputs,
targetsTest = testing_set1_outputs_GDT-HA)
  save(model, file="Global_8,9,11-0_5_550it_2_hidden.model.train_window_set1")
  predictions_set1 <- predict(model, testing_set1_inputs)

  model <- mlp(training_set2_inputs_ran, training_set2_outputs_GDT-HA, size = 2,
learnFuncParams = c(0.5, 0.001), maxit = 550, inputsTest = testing_set2_inputs,
targetsTest = testing_set2_outputs_GDT-HA)
  save(model, file="Global_8,9,11-0_5_550it_2_hidden.model.train_window_set2")
  predictions_set2 <- predict(model, testing_set2_inputs)

  model <- mlp(training_set3_inputs_ran, training_set3_outputs_GDT-HA, size = 2,
learnFuncParams = c(0.5, 0.001), maxit = 550, inputsTest = testing_set3_inputs,
targetsTest = testing_set3_outputs_GDT-HA)
  save(model, file="Global_8,9,11-0_5_550it_2_hidden.model.train_window_set3")
  predictions_set3 <- predict(model, testing_set3_inputs)

  predictions <- rbind(predictions_set1, predictions_set2, predictions_set3)

  #test correlations pred v obs
  cat( "ModFOLD7_NN_test_GDT-HA",
       cor(predictions, GDT-HA, method="pearson"), cor(predictions, GDT-HA,
method="spearman"), cor(predictions, GDT-HA, method="kendall"),
       cor(predictions, GDT, method="pearson"), cor(predictions, GDT,
method="spearman"), cor(predictions, GDT, method="kendall"),
       cor(predictions, MaxSub, method="pearson"), cor(predictions, MaxSub,
method="spearman"), cor(predictions, MaxSub, method="kendall"),
       cor(predictions, TM-score, method="pearson"), cor(predictions, TM-score,
method="spearman"), cor(predictions, TM-score, method="kendall")
       , "\n", sep=" ", file = "Global_NN_both_rounds_correlations.dat",append = TRUE)

  #test ranking - cumulative scores of top ranked models
```

```
  cat( "8,9,11-0_5_100it_3_hidden\n", file = "Global_NN_both_rounds_ranks.dat",append =
TRUE)

  DT2 <- rbind( testing_set1, testing_set2, testing_set3)
  DT2[,V17 := predictions ]#add predictions as last column (V17)
  target_ids <- unique(DT2$V1)#get all IDs in data (unique variables in column $V1)

  #setup empty arrays
  NNtest <- c()

  for(i in 1:length(target_ids) )
  {
    #print(target_ids[i])
    set1 <-subset( DT2, V1==target_ids[i])

    #mean of ModFOLDclustQ_single_res_global_all, ProQ2_res_global_all,
CDA_res_global_all, DBA_res_global_all, SSA_res_global_all and
ModFOLD6_single_res_global_all #<--- 3rd BEST COMBO FOR RANKING
    NNtest <- rbind( NNtest, c( set1[which.max(set1$V17), ]$V1,
set1[which.max(set1$V17), ]$V2, set1[which.max(set1$V17), ]$V13,
set1[which.max(set1$V17), ]$V14, set1[which.max(set1$V17), ]$V15,
set1[which.max(set1$V17), ]$V16 ))
  }

  #standard error function for error bars
  std_err <- function(x) sd(x)/sqrt(length(x))

  #cumlative GDT-HA, GDT-TS, MaxSub & TM-scores of top models for each target ranked by
each global QA score
  cumulativescores <- c()
  cumulativescores <-  rbind( cumulativescores, c( "Method", "GDT-HA", "GDT-TS",
"MaxSub", "TM-score", "Std_err_GDT-HA", "Std_err_GDT-TS", "Std_err_MaxSub",
"Std_err_TM-score" ))
  cumulativescores <-  rbind( cumulativescores, c( "NNtest",
sum(as.numeric(NNtest[,3])), sum(as.numeric(NNtest[,4])), sum(as.numeric(NNtest[,5])),
sum(as.numeric(NNtest[,6])), std_err(as.numeric(NNtest[,3])),
std_err(as.numeric(NNtest[,4])), std_err(as.numeric(NNtest[,5])),
std_err(as.numeric(NNtest[,6])) ))
  #output table to a file
  cat( "Round1+Round2\n", file = "Global_NN_both_rounds_ranks.dat", append = TRUE)
  write.table( cumulativescores, file = "Global_NN_both_rounds_ranks.dat", sep = " ",
quote = FALSE, row.names = FALSE, col.names = FALSE, append = TRUE)
}
```

```
# TensorFlow_Data_search

import itertools
import os
import os.path
import tensorflow as tf
import numpy
import pandas as pd
sess = tf.InteractiveSession()

# A function which aims to extract all the data produced from R_Part1 and stores them
in arrays to be used in the NN.
def run(set_num, combination, observation, learning_rate, training_epochs, n_hidden1,
n_hidden2, n_input):
       # Read files produced by R_Part1 and stores the data into a Data Frame.
        df_train = pd.read_csv("training_set%d_inputs_ran.csv" % set_num)
        df_GDT-HA = pd.read_csv("training_set%d_outputs_GDT-HA.csv" % set_num)
        df_test_inputs = pd.read_csv("testing_set%d_inputs.csv" % set_num)
        df_test_output = pd.read_csv("testing_set%d_outputs_GDT-HA.csv" % set_num)
       # Extracts the wanted data from the Data Frames above and converts the frame
into a Numpy-array.
        trainer = df_train.as_matrix(combination)
        label = df_GDT-HA.as_matrix(observation)
        test_inputs = df_test_inputs.as_matrix(combination)
        test_outputs = df_test_output.as_matrix(observation)
        h = my_mlp(set_num, trainer, label, learning_rate, training_epochs, n_hidden1,
n_hidden2, n_input, test_inputs, test_outputs)
        return h

def multilayer_perceptron(x, w1, w2, drop, out):
    # the first hidden layer
    layer_1 = tf.matmul(x, w1)
    layer_1 = tf.nn.dropout(layer_1, drop)
    # the second hidden layer
    layer_2 = tf.matmul(layer_1, w2)
    layer_2 = tf.nn.dropout(layer_2, drop)
    # Output layer with linear activation
    out_layer = tf.matmul(layer_2, out)
    return out_layer

def my_mlp (num, trainer, trainer_awn, learning_rate, training_epochs, n_hidden1,
n_hidden2, n_input, test_inputs, test_outputs):
```

```
    trX, trY= trainer, trainer_awn
    #create placeholders
    x = tf.placeholder(tf.float32, shape=[None, n_input])
    y_ = tf.placeholder(tf.float32, shape=[None, ])
    keep_prob = tf.placeholder("float")
    #create initial weights
    w1 = tf.Variable(tf.truncated_normal([n_input, n_hidden1], stddev=0.01))
    w2 = tf.Variable(tf.truncated_normal([n_hidden1, n_hidden2], stddev=0.01))
    out = tf.Variable(tf.truncated_normal([n_hidden2, 1], stddev=0.01))
    #predicted class and loss function
    y = multilayer_perceptron(x, w1, w2, keep_prob, out)
    # Reshapes the observational data.
    y_ = tf.reshape(y_, [-1, 1])
    # Cost function, aims to reduce the difference between the predictions and the
observational data.
    cross_entropy = tf.reduce_sum(tf.abs(y - y_))
    #training
    train_step =
tf.train.AdagradOptimizer(learning_rate=learning_rate).minimize(cross_entropy)
    correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
    init_op = tf.initialize_all_variables()
    saver = tf.train.Saver()
    # Start training.
    with tf.Session() as sess:
        # you need to initialize all variables
        sess.run(init_op)
       #training session, it is run multiple times equal to the set iterations/epochs.
        for i in range(training_epochs + 1):
           #feeds the training data, both combination data and observation data, into
the placeholders.
            sess.run([train_step, cross_entropy], feed_dict={x: trX, y_: trY,
keep_prob: 0.9})
        print("Accuracy:", accuracy.eval({x: test_inputs, y_: test_outputs, keep_prob:
1}))
       #Creates a Numpy array containing the final model predictions.
        best = sess.run(y, feed_dict={x: test_inputs, keep_prob: 1})
       #Saves the weights for each set seperatly.
        saver.save(sess,
'/home/filipe/Documents/Disseration/tensorflow/Data_searching/dropout/Rank/Model%d/mode
l' % num)
    return best
```

```
#End file created by TensorFlow_Data_searching_Part2 when the NN is score higher than
the score specified
while os.path.exists("end.csv") == False:
    # Create a list containing the methods which are too be combined.
    # Key: ModFOLD5_single_orig_global (3), ModFOLDclustQ_single_orig_global (4),
ModFOLDclust2_single_orig_global (5), ModFOLD5_single_res_global (6),
ModFOLDclustQ_single_res_global (7), ProQ2_res_global (8), CDA_res_global (9),
DBA_res_global (10), SSA_res_global (11), ModFOLD6_single_res_global (12).
    combination_choice = ["V9", "V11", "V12"]

    #Create a text file containing the wanted combination, this file is fed into
R_Part1
    file = open("combination.txt","w")
    file.write("9, 11, 12")
    file.close()

    # Runs the R script, R_Part1.R through the terminal.
    os.system("Rscript Data_searching_Part1.R")

    #train each data set to GDT-HA score (V13)
    prediction1 = run(1, [combination_choice], ["V13"], 0.001, 100, 5, 4,
len(combination_choice))
    numpy.savetxt('prediction_set1.out', prediction1)

    prediction2 = run(2, [combination_choice], ["V13"], 0.001, 100, 5, 4,
len(combination_choice))
    numpy.savetxt('prediction_set2.out', prediction2)

    prediction3 = run(3, [combination_choice], ["V13"], 0.001, 100, 5, 4,
len(combination_choice))
    numpy.savetxt('prediction_set3.out', prediction3)

    # Runs the R script, R_Part2.R through the terminal.
    os.system("Rscript Data_searching_Part2.R")
```

Appendix 7

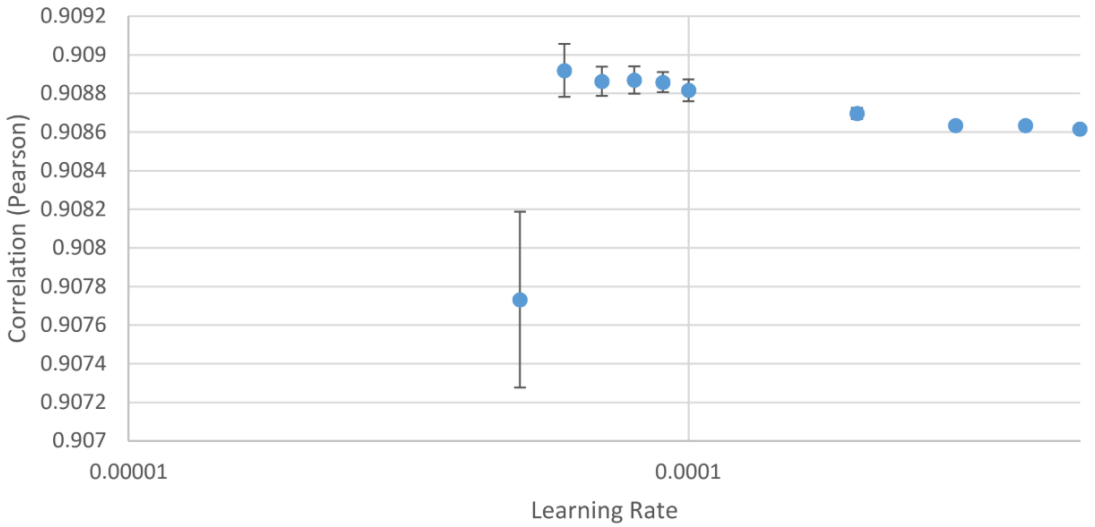

Figure S7. **Scatter chart showing how the correlation changes with learning rate where the rate is between 0.00005 and 0.0005.** Error bars are calculated by taking the standard deviation of 10 runs of the network using the same hyperparameters.

Appendix 8

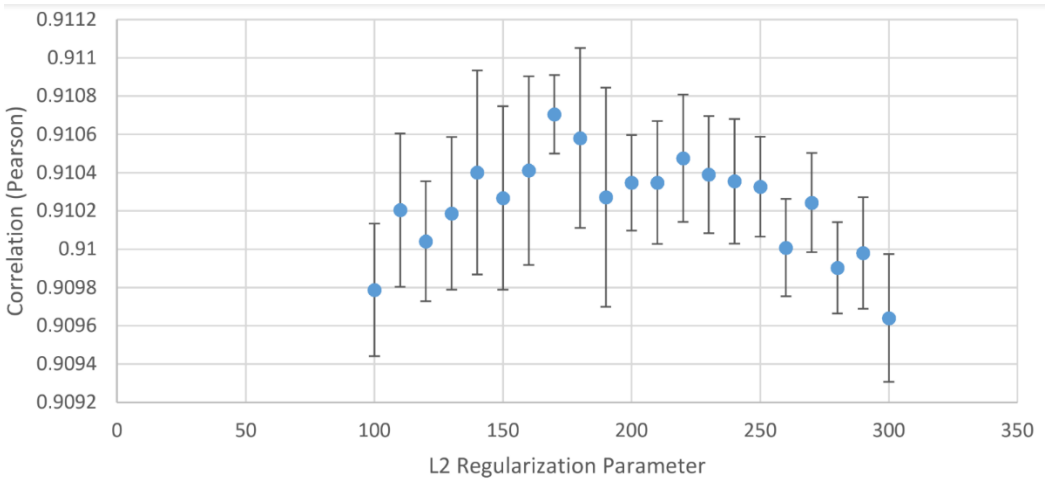

Figure S8. **Scatter chart showing how the correlation changes with L2 regularisation, where the L2 parameter is between 100 and 300.** Error bars are calculated by taking the standard deviation of 10 runs of the network using the same hyperparameters.

Appendix 9

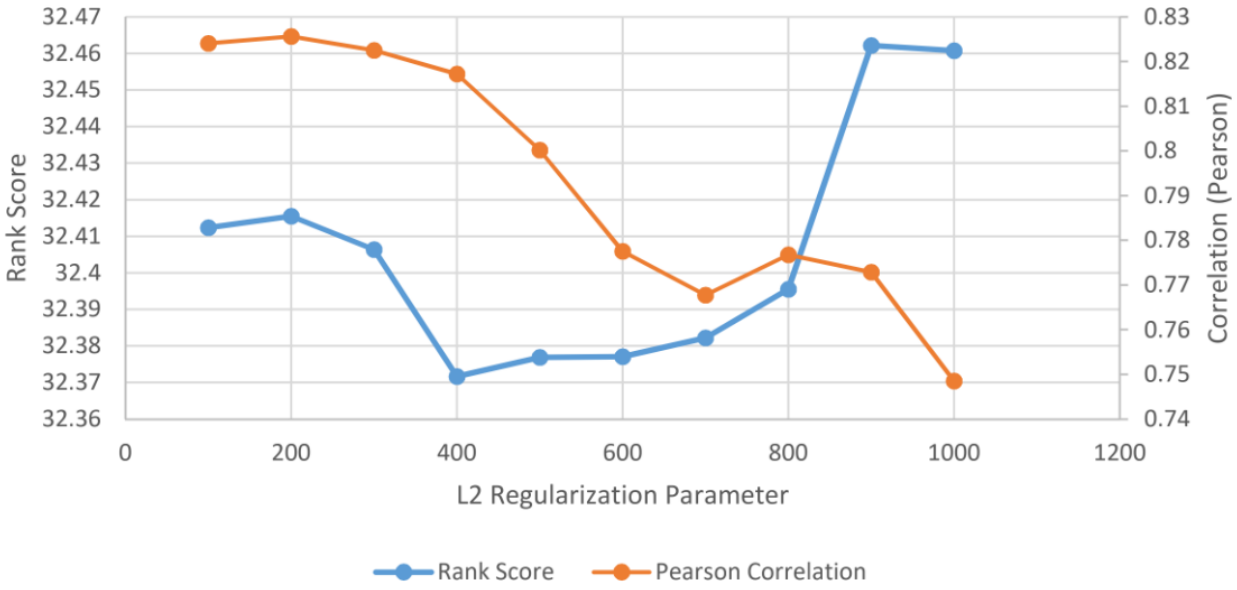

Figure S9. Line chart showing how rank score and correlation change with the L2 parameter between values of 100 and 1000. Error bars are excluded for clarity.

Appendix 10

	Average lDDT		Average CAD score		Average lDDT-BS	
Server Name	**Dif.**	**Ref.**	**Dif.**	**Ref.**	**Dif.**	**Ref.**
Robetta	-2.15	69.04	-0.03	0.69	2.66	67.81
IntFOLD4-TSb	0	66.89	0	0.66	0	70.47

Table S10. Performance of IntFOLD4-TS versus Robetta. CAMEO-3D: Common Subset Comparison, 1-year Performance (2016-07-01 - 2017-06-24) (891 targets - 2 methods). IntFOLD4_TS is the reference server (listed as server58, or IntFOLD4-TSb on CAMEO). Data are from http://www.cameo3d.org/. The table is sorted by difference in Average lDDT score.

Appendix 11

	Average lDDT		Average CAD score		Average lDDT-BS	
Server Name	**Dif.**	**Ref.**	**Dif.**	**Ref.**	**Dif.**	**Ref.**
Robetta	-2.43	71.88	-0.03	0.7	3.53	70.45
IntFOLD4-TSb	0	69.45	0	0.68	0	73.98
RaptorX	0.11	69.34	0	0.68	5.54	68.43
IntFOLD3-TS	1.55	67.9	0.02	0.66	3.52	70.45
IntFOLD2-TS	1.74	67.71	0.02	0.66	4.33	69.64
HHpredB	2.37	67.08	0	0.67	4.07	69.91
SWISS-MODEL	3.45	66	0.03	0.64	2.86	71.12
SPARKS-X	6.19	63.26	0.04	0.64	7.19	66.78
Princeton_TEMPLATE	9.84	59.61	0.09	0.59	17.92	56.06
NaiveBLAST	11.74	57.71	0.12	0.56	10.69	63.29

Table S11. Performance of IntFOLD4-TS versus other servers. CAMEO-3D: Common Subset Comparison, 6-months Performance (2016-12-30 - 2017-06-24) (304 targets - 10 methods). IntFOLD4_TS is the reference server (listed as server58, or IntFOLD4-TSb on CAMEO). Data are from http://www.cameo3d.org/. The table is sorted by difference in Average lDDT score.

Appendix 12

	Avg. lDDT		Avg. CAD-score		Avg. lDDT-BS	
Server Name	**Dif.**	**Ref.**	**Dif.**	**Ref.**	**Dif.**	**Ref.**
Robetta	-1.67	72.45	-0.02	0.7	5.54	71.32
IntFOLD5-TS	0	70.78	0	0.69	0	76.85
RaptorX	0.18	70.6	0	0.68	4.84	72.01
IntFOLD4-TS	0.45	70.33	0	0.68	0.27	76.58
IntFOLD3-TS	1.82	68.96	0.02	0.67	1.43	75.42
SWISS-MODEL	2.64	68.14	0.03	0.66	0.47	76.38
HHpredB	3.64	67.14	0.01	0.67	6.26	70.6
M4T-SMOTIF-TF	5.46	65.32	0.05	0.64	1.79	75.07
SPARKS-X	6.39	64.38	0.04	0.64	5.94	70.92
PRIMO	7.17	63.61	0.06	0.63	5.02	71.83
PRIMO_BST_CL	7.17	63.61	0.06	0.63	5.02	71.83
PRIMO_BST_3D	8.47	62.3	0.07	0.61	6.92	69.93
PRIMO_HHS_3D	8.86	61.92	0.08	0.61	6.95	69.91
PRIMO_HHS_CL	9.31	61.47	0.08	0.61	9.26	67.59
NaiveBLAST	9.74	61.04	0.1	0.59	4.37	72.49
Princeton_TEMPLATE	10.26	60.52	0.09	0.59	18.63	58.22
Phyre2	13.15	57.62	0.06	0.63	4.74	72.12

Table S12. Independent benchmarking of tertiary structure predictions with CAMEO 3D data. Performance results for 1 year of data (2018-01-26 to 2019-01-19) are shown for a common subset of 199 targets for all the 17 public methods. The reference method is IntFOLD5-TS and the table is sorted by average lDDT. Data are downloaded from http://www.cameo3d.org/.

Appendix 13

	Avg. lDDT		Avg. CAD-score		Avg. lDDT-BS	
Server Name	**Dif.**	**Ref.**	**Dif.**	**Ref.**	**Dif.**	**Ref.**
Robetta	-1.32	69.07	-0.02	0.68	5.8	66
IntFOLD5-TS	0	67.75	0	0.67	0	71.81
IntFOLD4-TS	0.52	67.23	0	0.66	0.24	71.57
RaptorX	0.58	67.17	0	0.66	4.85	66.95
IntFOLD3-TS	2.1	65.65	0.02	0.65	1.9	69.9

Table S13. Intensive independent benchmarking of tertiary structure predictions with CAMEO 3D data. Performance results for 1 year of data (2018-01-26 to 2019-01-19) are shown for a common subset of 575 targets for the top 3 public methods plus the older versions of IntFOLD. The reference method is IntFOLD5-TS and the table is sorted by average lDDT. Data are downloaded from http://www.cameo3d.org/.

Appendix 14

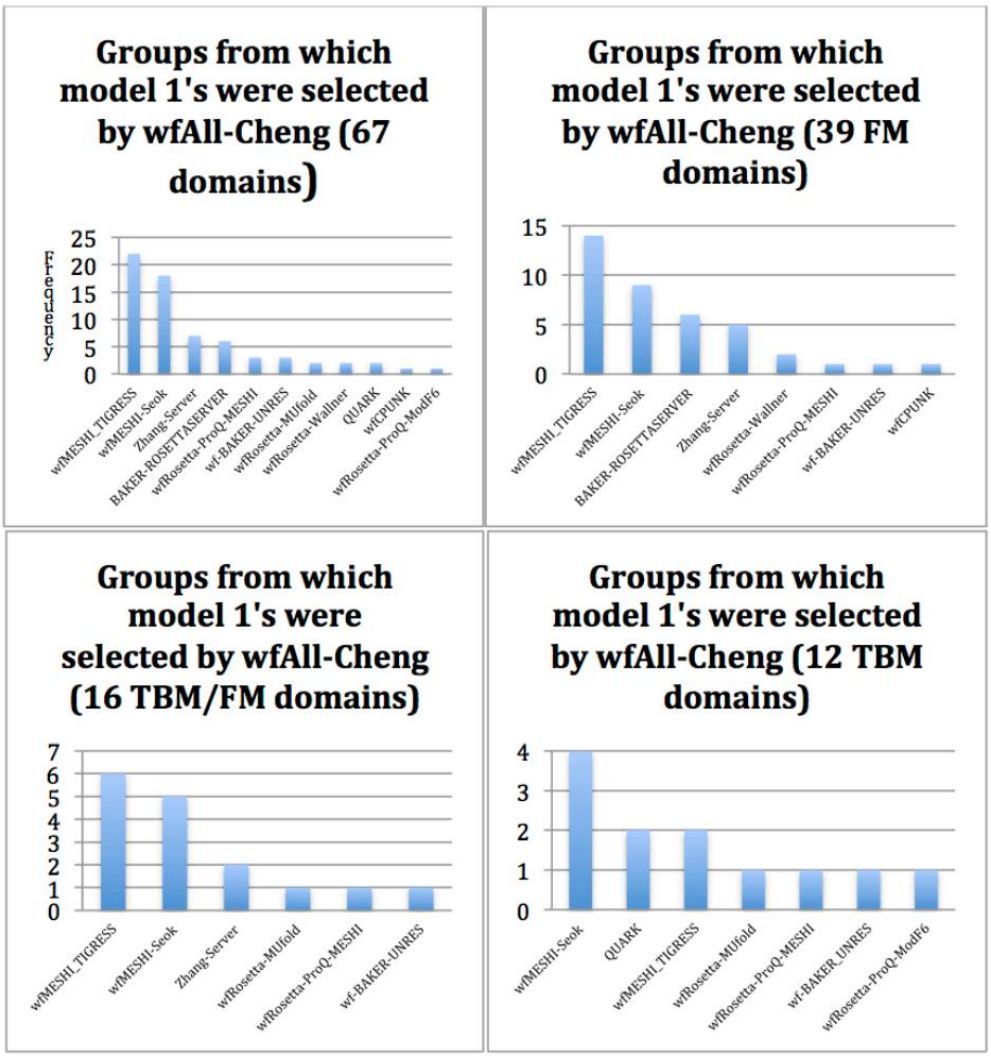

Figure S13. **The wfAll-Cheng pipeline selected its model 1 among all models contributed by the WeFold pipelines as well as servers models.** (Top left) Of the 67 CASP12 domains released so far, the wfAll-Cheng pipeline selected 22 models submitted by pipeline wfMESHI_TIGRESS and 18 models submitted by pipeline wfMESHI-Seok as model 1. (Top right) Of the 39 FM CASP12 domains released so far, the wfAll-Cheng pipeline selected 14 models submitted by pipeline wfMESHI_TIGRESS and 9 models submitted by pipeline wfMESHI-Seok as model 1. (Bottom left) Of the 16 TBM/FM CASP12 domains released so far, the wfAll-Cheng pipeline selected 6 models submitted by pipeline wfMESHI_TIGRESS and 5 models submitted by pipeline wfMESHI-Seok as model 1. (Bottom right) Of the 12 TBM CASP12 domains released so far, the wfAll-Cheng pipeline selected 4 models submitted by pipeline wfMESHI_Seok and 2 models submitted by pipeline wfMESHI-TIGRESS as model 1.