10-16-2019

Protein Inter-Residue Distance Prediction Using Residual and Capsule Networks

Andrew Dillon
University of Missouri-St. Louis, ajdn72@mail.umsl.edu

Protein Inter-Residue Distance Prediction Using Residual and Capsule Networks

by

Andrew Jared Dillon

A Thesis

Submitted to The Graduate School of the

University of Missouri-St. Louis
in partial fulfillment of the requirements for the degree

Master of Science

In

Computer Science

December 2019

Advisory Committee

Badri Adhikari, Ph.D.
(Chairperson)

Sharlee Climer, Ph.D.

Mark Hauschild, Ph.D.

Abstract

The protein folding problem, also known as protein structure prediction, is the task of building three-dimensional protein models given their one-dimensional amino acid sequence. New methods that have been successfully used in the most recent CASP challenge have demonstrated that predicting a protein's inter-residue distances is key to solving this problem. Various deep learning algorithms including fully convolutional neural networks and residual networks have been developed to solve the distance prediction problem. In this work, we develop a hybrid method based on residual networks and capsule networks. We demonstrate that our method can predict distances more accurately than the algorithms used in the state-of-the-art methods. Using a standard dataset of 3420 training proteins and an independent dataset of 150 test proteins, we show that our method can predict distances 51.06% more accurately than a standard residual network method, when accuracy of all long-range distances are evaluated using mean absolute error. To further validate our results, we demonstrate that three-dimensional models built using the distances predicted by our method are more accurate than models built using the distances predicted by residual networks. Overall, our results, for the first time, highlight the potential of capsule-residual hybrid networks for solving the protein inter-residue distance prediction problem.

Acknowledgements

Writing a thesis is a significant undertaking. It took several months of focused research, discussion, and writing to produce this work. I would like to thank my family for their constant support, without which I would not have been able to manage all of this. They helped me to devote all the time I needed to completing this project. I would also like to thank my advisor, Professor Adhikari, for helping me decide on a thesis topic, offering ideas and guidance, and always being available to answer the many questions I had along the way - especially during weekends.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Proteins are the building blocks of life. They are responsible for many of our biological functions and structure. For example, the Myosin proteins play an important role in our muscle tissue, allowing it to contract (Lodish H, Berk A, Zipursky SL, et al., 2000). And the perhaps more widely known Hemoglobin protein carries oxygen through our bloodstream (Marengo-Rowe AJ, 2006). These are just two examples of the huge variety of proteins that we know about today. So what are these molecules made of and what makes them different from each other?

The Central Dogma of Molecular biology states that "DNA makes RNA makes protein" (Leavitt SA, 2010). What this means is that our DNA contains instructions for how to build particular proteins. This information is copied into RNA strands that, in turn, are used to construct protein molecules. Each RNA strand represents a par-

ticular protein. It does so by encoding the sequence of amino acids needed to construct that protein.

Thus, proteins are composed of amino acids. There are approximately 20 amino acids in human biology (National Research Council (US) Subcommittee on the Tenth Edition of the Recommended Dietary Allowances, 1989). The specific sequence of amino acids used to construct a protein makes it unique from all others. This amino acid sequence can, in effect, serve as a unique identifier of the protein. But a simple one dimensional strand of amino acids is not particularly useful. In order for a protein to serve any kind of interesting function, it must take on a three dimensional shape. This process of converting a protein from a one dimensional strand (primary structure) to a three dimensional shape (tertiary structure) is called protein folding (Alberts B, Johnson A, Lewis J, et al., 2002).

Protein folding is a very important and widely studied process. It is extremely challenging to predict what a protein's tertiary structure will be given only its primary structure. Predicting the tertiary structure of a protein given only its amino acid primary sequence is termed *de novo* protein structure prediction. According to Science, it is one of the top 100 unsolved problems in modern science (Science, 2005). Building a tool that can accurately perform *de novo* protein structure prediction is the holy grail of protein folding research.

Wet-lab experiments are the primary alternative to *de novo* protein structure prediction when determining a protein's tertiary structure. These experimental methods include techniques such as X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy. Because *de novo* protein structure prediction methods are still far from being perfected, wet-lab experiments have been used to discover the structure of most proteins for which we have true structures. However, wet-lab methods are expensive and can take months to years to complete. The potential to overcome these limitations is one of the key motivators driving protein folding research.

1.1 Protein Folding Methods & Deep Learning

The Critical Assessment of protein Structure Prediction (CASP) is a community wide and world wide experiment for advancing the state of the art in *de novo* protein structure prediction. CASP experiments have occurred every 2 years since 1994.

During each CASP experiment, models are submitted by competing teams from around the world. Each model predicts 3D structures for a set of proteins whose true structures were previously unknown (this eliminates any possibility of a model possessing foreknowledge of the true structures and thereby gaining an advantage). After all the submissions are complete, the new gold standard for *de novo*

protein structure prediction is revealed.

Most of the recent CASP models are built using the same underlying pipeline. This standard pipeline consists of the following steps (see **Figure 1.1**).

Given an amino acid sequence:

1. Generate multiple sequence alignments (MSA) and other features.
2. Enhance this data by performing feature engineering.
3. Predict a matrix of contact/distance values for all residue (amino acid) pairs in the protein.
4. Construct a 3D model using the contact map and other features.

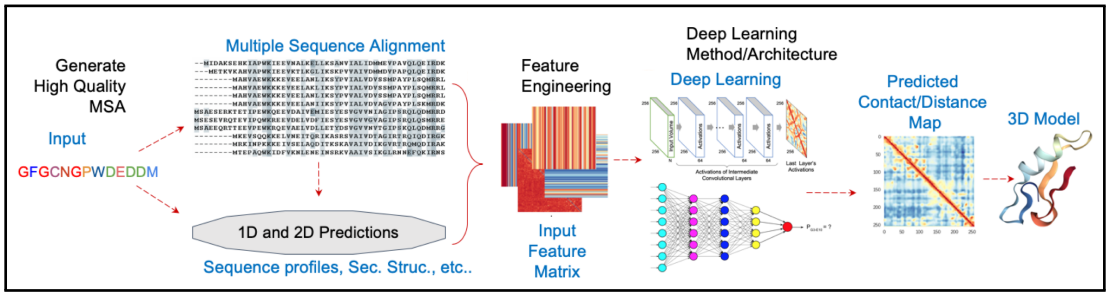

Figure 1.1: Standard 3D model pipeline

Pipeline that predicts a 3D protein model given a sequence of amino acids.

Predicting the residue pair contact map is one of the most important steps in this pipeline. Much of the recent progress that has

been made on the *de novo* protein folding problem stems from improvements to the contact map prediction step. The top performing models in the most recent CASP13 (2018) challenge all used deep learning architectures to produce these contact maps. In particular, all of these architectures were variants of convolutional neural networks (CNNs) and residual neural networks (ResNets).

1.2 Contact Maps and Distance Maps

Residue pair contact maps and residue pair distance maps (which we will henceforth refer to as contact maps and distance maps, respectively) store information about the relative position of a protein's residues in its tertiary structure. Contact maps and distance maps are both $n \times n$ matrices of numbers, where n is the number of residues in the protein. However, these two structures differ in terms of the numbers that they store.

Contact maps store binary integer values, 1s and 0s, that indicate whether or not each pair of residues are in contact in the protein's tertiary structure. Distance maps, on the other hand, store continuous non-negative decimal values that indicate the shortest distance between each pair of residues in the protein's tertiary structure. As

such, distance maps store more information than contact maps.

Given a contact map M, residue i is said to be in contact with residue j if $M_{ij} = 1$. Note that contact maps and distance maps are always symmetric matrices. That is, they are mirrored over their diagonal and remain unchanged after being transposed. Note also that the unit used in distance maps is typically the Angstrom (Å), which is equal to 0.1 nanometers.

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0.0 & 8.2 & 7.9 \\ 8.2 & 0.0 & 6.1 \\ 7.9 & 6.1 & 0.0 \end{bmatrix}$$

Figure 1.2: Contact map and distance map

A contact map on the left and distance map on the right. Note that the diagonal of the contact map is composed entirely of 1s. This is because each diagonal element represents whether or not a particular residue is in contact with itself. And note that the diagonal of the distance map is composed of 0s. This is because the distance from any given residue to itself is 0.

Figure 1.2 contains an example contact map and distance map for the same imaginary protein. This protein contains 3 residues (note that real proteins are usually composed of hundreds of residues). In this example, we consider two residues to be in contact if their distance in the protein's tertiary structure is less than 8Å. Our example

protein has two pairs of residues in contact (pairs 1,3 and 2,3) and one pair that is not (pair 1,2).

Most deep learning models used for *de novo* protein structure prediction have been designed to predict contact maps. One reason for this is that deep learning models tend to perform better on classification tasks than regression tasks. However, recent work in the field (Xu, 2018) (Andrew S, John J, Demis H, 2018) has shown that the performance of protein folding pipelines can be improved by utilizing deep learning models that predict distance maps rather than contact maps.

1.3 Goals

The intent of this work is to investigate the potential of capsule networks (Sara S, Nicholas F, Geoffrey H, 2017) (Rodney L, Ulas B, 2018) to improve upon existing inter-residue distance prediction methods. In particular, we aim to discuss the characteristics of capsule networks that make them appealing candidates for contact map and distance map prediction. We also aim to identify the major roadblocks that have prevented their immediate adoption and propose methods to overcome them. Finally, after reviewing our results,

we will suggest opportunities for future research on this topic.

Chapter 2

Background

2.1 Residual Networks

Many deep learning models used in protein folding pipelines are based on the residual network architecture (Xu, 2018) (Adhikari, 2019a). Residual networks make use of skip connections to allow certain layers in the network to jump over other layers. This is in contrast to more traditional deep learning network architectures that simply stack each layer on top of the previous one, feeding each layer's output into the subsequent layer's input.

2.2 Capsule Networks

Capsule networks were introduced in 2011 by Hinton et al. (G. E. Hinton, A. Krizhevsky, S. D. Wang, 2011) and improved in 2017 by Sabour et al. (Sara S, Nicholas F, Geoffrey H, 2017). They have been successfully used in a wide variety of contexts, including classification of MNIST digits, estimating depth in UAV images (Sunil Prakash, Gaelan Gu, 2018), and segmenting images from CT lung scans (Rodney L, Ulas B, 2018). Capsule layers are novel in that they learn to model various properties of an entity (such as size, orientation, and skew) rather than the simple fact of its presence or absence, as in traditional convolutional and residual layers.

We believe that these characteristics of capsule layers make them highly applicable to the protein distance prediction problem. Proteins are composed of subunits called secondary structures. These structures assemble together to form the full 3D structure of a protein. The orientation and size of these secondary structures play a key role in determining the structure of the protein. Capsule layers are designed to learn and recognize these geometric properties. As such, they should be able to use this knowledge to produce better distance maps than more traditional deep learning models built without capsule layers (see **Figure 2.1**).

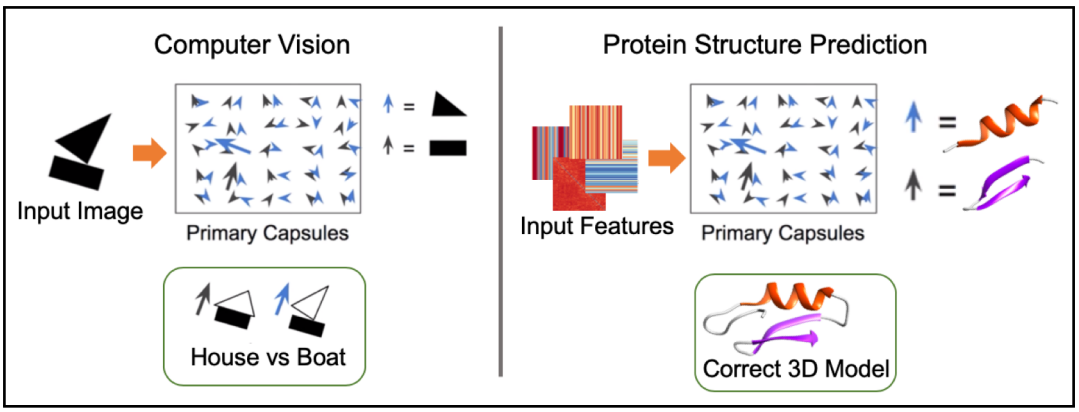

Figure 2.1: Capsules learn geometric properties

Capsules learn the relative orientation (angle) of individual objects to classify them. For example, classifying whether two objects form a house or a boat (left). Protein tertiary structures are composed of secondary structure units helices, coils, strands, and beta-sheets whose relative orientation is what makes the structures different (right). Capsules can learn this relative orientation of secondary structures.

Capsule networks are able to capture these geometric properties by storing neuron-level information in the form of vectors, rather than scalars as in traditional neural networks. A scalar value is only able to represent a single property. The magnitude of a neuron-level scalar typically represents the presence or absence of an entity. However, vectors are able to represent much more information. Each element of a vector can learn to represent specific properties of an entity. And the magnitude of the vector as a whole can itself represent an additional derivative property. In capsule networks, the magnitude of a capsule's vector output typically represents the presence or absence of an entity (as determined by its constituent properties).

2.3 Convolutional Capsules

The capsule layers described by Sabour et al. (Sara S, Nicholas F, Geoffrey H, 2017) have a couple of critical limitations that prevent them from being used for distance map prediction. First, distance map prediction is a regression task. But these capsule layers are constrained to classification tasks due to the squashing function. The squashing function is a nonlinear activation function that is used to push each capsule's length towards 0 or 1. Second, distance maps are relatively large, consisting of 256×256 matrices for our training samples. This necessitates equally large input features. However, these capsule layers can only operate on relatively small inputs on the order of 32×32 (the dimensions of the MNIST dataset used by Sabour et al.).

The first issue, that of using capsule layers for regression tasks, will be addressed later in this work. However, the second issue can be resolved by using convolutional capsule layers. These layers were introduced by Rodney et al. for the SegCaps architecture (Rodney L, Ulas B, 2018). SegCaps is a state-of-the-art image segmentation network. It is able to support CT scan images of size 512×512 as input (much larger than our 256×256 inputs). In addition, it achieves top performance in the image segmentation field while reducing the

number of parameters in the network by up to 95.4%, compared to other top performing architectures (Rodney L, Ulas B, 2018).

Chapter 3

Methods

3.1 Dataset

We use the dataset provided by the IEEE ICMLA 2019 - Protein Inter-Residue Distance Prediction Challenge (Adhikari, 2019c) to train and evaluate our models. This dataset provides input features in the shape of $256 \times 256 \times 13$ input volumes. The output labels are 256×256 matrices of real numbers, where each matrix represents the distance map for a protein.

The $256 \times 256 \times 13$ input features are the result of substantial preprocessing of the original amino acid sequence. The PIDP Challenge's feature engineering pipeline starts by consuming a Multiple Sequence

Alignment (MSA) file as input. It then proceeds to enhance this input by running it through six different tools:

1. **PSIPRED** (McGuffin LJ, Bryson K, Jones DT, 2000) - generates three 1D features that represent secondary structure predictions. There are 3 state predictions for each residue of the input sequence. They indicate whether each amino acid will be part of a helix, beta-strand or coil in the final model.

2. **PSISOLV** (McGuffin LJ, Bryson K, Jones DT, 2000) - generates one 1D feature that represents solvent accessibility predictions. These are binary predictions of hydrophobicity for each residue. They indicate whether each amino acid will be 'exposed' to water or not.

3. **CCMpred** (Lab, 2018) and **FreeContact** (Kaján, László and Hopf, Thomas A. and Kalaš, Matúš and Marks, Debora S. and Rost, Burkhard, 2014) - each generates one 2D feature that represents coevolutionary signal predictions. These features capture the strength of covariance between all pairs of residue positions. These predictions can be considered as independent contact or distance predictions.

4. **Shannon Entropy Sum** (Jones DT, Singh T, Kosciolek T, Tetchner S, 2015) - generates one 1D feature that represents Shannon entropy of the alignment column.

5. **pstat_pots** (Jones DT, Singh T, Kosciolek T, Tetchner S, 2015) - generates one 2D feature that represents calculated contact potentials.

Thus, in total, these tools produce $3+1+1+1+1+1 = 8$ features for each protein in our dataset. Of these 8 features, 5 are 1D vectors and 3 are 2D matrices. However, our models are designed to accept only matrices as input. As such, before we can use these features to train our models we must convert the 5 vector features into matrices. We accomplish this by tiling each vector feature both horizontally and vertically. This operation maps each vector feature to two matrix features.

After mapping the 5 vector features to matrices, we possess $(3 \cdot 2) + (1 \cdot 2) + (1 \cdot 2) + 1 + 1 + 1 = 13$ channels of data for each protein in our dataset. Each 13 channel input matrix is conceptually comparable to an input image having 13 channels instead of the typical 3 (red, green, and blue).

Each of the 13 features is a square matrix of shape $L \times L$, where L is the length of the corresponding protein's amino acid primary structure. Before we use this data to training our model, we cap all features to size 256×256, as some proteins in our dataset have a

primary structure length greater than 256.

In accordance with deep learning best practice, we split our data into a training set, a validation set, and an independent test set. The training set consists of 3284 proteins, the validation set contains 136 proteins, and the test set consists of 150 proteins.

3.2 Capsule Activation Functions

Many capsule networks use the squashing function introduced by Sabour et al. (Sara S, Nicholas F, Geoffrey H, 2017) to force the length of each capsule's output vector into the range between 0 and 1. This works well for classification tasks where each capsule must learn to produce a vector whose length indicates the presence or absence of an entity.

$$v_j = \frac{||s_j||^2}{1 + ||s_j||^2} \frac{s_j}{||s_j||} \tag{3.1}$$

Squashing Function

Where:

- v_j is the output vector of the j^{th} capsule

- s_j is the j^{th} capsule's input vector

However, predicting distance maps necessitates an activation function capable of performing regression. Thus the activation function must output a continuous value and cannot simply trend towards binary activations. To this end, we first tried removing the squashing function entirely. However, the capsule activation function is an important source of nonlinearity for the network. Removing it altogether is not ideal.

Rather than simply removing the squashing function, we experimented with using elementwise ReLU (Xavier Glorot, Antoine Bordes, Yoshua Bengio, 2011) in its place. ReLU is commonly used in traditional neural network layers where it often works well for regression layer outputs. Our application of ReLU to capsule layer activations is novel and is a key contribution of this work.

$$v_j = max(0, s_{j|i}) \tag{3.2}$$

ReLU Function

Where:

- v_j is the output vector of the j^{th} capsule
- $s_{j|i}$ is each element of the j^{th} capsule's input vector

3.3 Cost Functions

Cost functions are used to evaluate the predictions of machine learning models. All deep learning networks require a cost function to be trained. The choice of cost function used to train distance prediction models is very important, as it is a key determinant of the model's performance. It is important to choose a cost function that capitalizes on the unique characteristics of the protein distance prediction problem.

We experimented with a few different cost functions to see what impact they would have on the performance of our models. Since our network's final output is a matrix of real numbers, our cost functions must be defined such that they accept a matrix of actual values and a matrix of predicted values. Each cost function must then return a single real number denoting the distance (or error) between the two input matrices.

The first cost function we tried was Mean Absolute Error (MAE). This is a very commonly used cost function that simply measures the average distance between two continuous variables. MAE is useful when the prediction error is equally important for all elements of the input.

$$MAE(y, \hat{y}) = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{3.3}$$

MAE Function

Where:

- y is a vector of actual distances
- $\hat{y}$ is a vector of predicted distances

Next, we tried the LogCosh cost function. LogCosh first computes the distance between the actual and predicted values. LogCosh is then defined as the logarithm of the hyperbolic cosine of this difference. It behaves similarly to mean squared error (MSE) but is not affected as strongly by occasional wildly incorrect predictions.

$$LogCosh(y, \hat{y}) = \frac{\sum_{i=1}^{n} log(cosh(y_i - \hat{y}_i))}{n} \tag{3.4}$$

LogCosh Function

Where:

- y is a vector of actual distances
- $\hat{y}$ is a vector of predicted distances

Finally, we tried a novel cost function that we designed specifically for the protein inter-residue distance prediction problem. We call this cost function LogCoshInv, as it is a modified version of the LogCosh function. We designed LogCoshInv to treat errors in small quantities as more important than errors in large quantities.

This is a useful characteristic for distance map prediction models, because when evaluating a predicted distance map against the ground truth, we care more about the accuracy of distances predicted for residue pairs that are close to one another in the true 3D structure than for residue pairs that are relatively far apart. This is because the accuracy of predicted distances for residue pairs that are relatively close to one another has a larger impact on the accuracy of 3D models generated using these predictions than residue pairs that are relatively far apart.

LogCoshInv is defined as the LogCosh of the inverse of the predicted and actual values, each multiplied by 100. We chose to multiply by 100 because protein inter-residue distances usually range between 3.9 and 100 Angstroms. We also add a small value, ϵ, to the denominator of each inverse to prevent division by zero.

$$LogCoshInv(y, \hat{y}) = \frac{\sum_{i=1}^{n} log(cosh(\frac{100}{y_i+\epsilon} - \frac{100}{\hat{y}_i+\epsilon}))}{n} \tag{3.5}$$

LogCoshInv Function

Where:

- y is a vector of actual distances
- $\hat{y}$ is a vector of predicted distances

For clarity of exposition, in equations 3-5, we define MAE, $LogCosh$, and $LogCoshInv$ such that they operate on two input vectors. However, for our purposes, we must redefine each of these cost functions to operate on the matrices that constitute our distance maps. In equations 6-8, Y denotes a matrix of actual distances and $\hat{Y}$ denotes a matrix of predicted distances.

$$MAE(Y, \hat{Y}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} |Y_{ij} - \hat{Y}_{ij}|}{n \cdot m} \tag{3.6}$$

MAE for Matrices

$$LogCosh(Y, \hat{Y}) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} log(cosh(Y_{ij} - \hat{Y}_{ij}))}{n \cdot m} \tag{3.7}$$

LogCosh for Matrices

$$LogCoshInv(Y, \hat{Y}) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} log(cosh(\frac{100}{Y_{ij}+\epsilon} - \frac{100}{\hat{Y}_{ij}+\epsilon}))}{n \cdot m} \tag{3.8}$$

LogCoshInv for Matrices

3.4 Baseline Model

In order to experimentally determine the effect of using capsule layers in our distance map prediction model, it is necessary to have a baseline to compare against. For our baseline model, we create a deep residual network consisting of 32 residual blocks. The output of the final layer is the predicted distance map. This baseline model architecture is depicted in **Figure 3.1**.

We chose to use 32 residual blocks in our baseline model because we found it to be a reasonable trade-off between maximizing the accuracy of our predicted distance maps and minimizing model training

time and consumption of our limited computational resources.

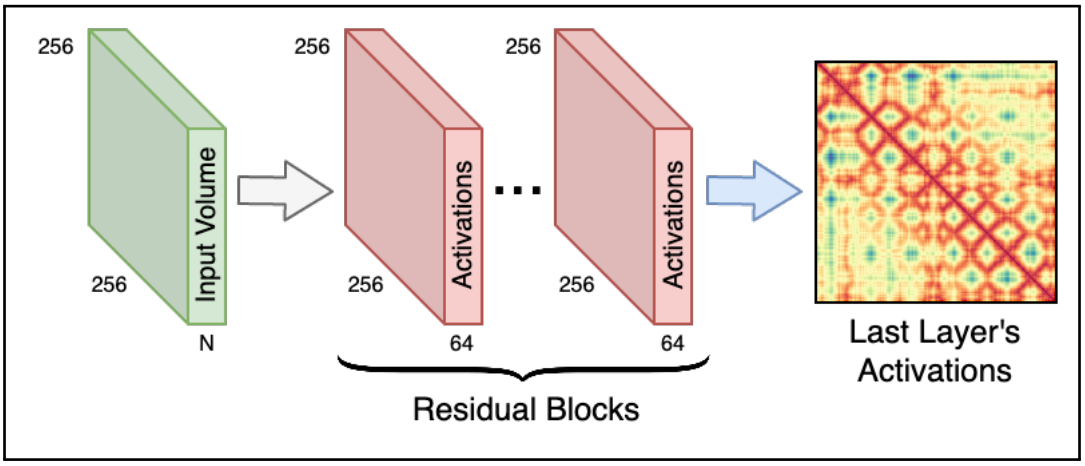

Figure 3.1: Baseline residual network architecture

All successful methods in the most recent CASP challenge (CASP13) use residual networks, including DeepMind's AlphaFold (Andrew S, John J, Demis H, 2018), ResPRE (Li Yang, Hu Jun, Zhang Chengxin, Yu Dong-Jun, Zhang Yang, 2019) by Yang Zhang's group at the University of Michigan, and DeepMetaPSICOV (Kandathil Shaun, Greener Joe, Jones David, 2019) from David Jones' group at University College London. This is why we chose a residual network as our baseline.

3.5 CapsDist Model

All capsule network architectures begin with at least one convolutional layer after the input volume and before the capsule layers.

However, we wish to test the effect of using multiple convolutional layers in front of the capsule layers. As such, our strategy was to iteratively replace the residual blocks of our baseline architecture with capsule layers. This CapsDist architecture is depicted in **Figure 3.2**.

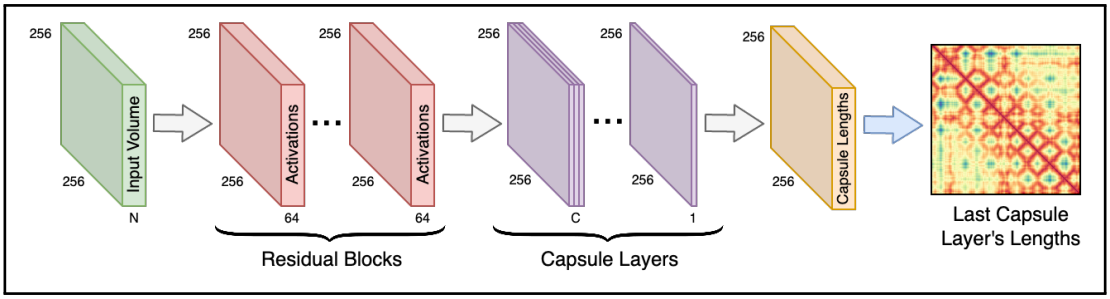

Figure 3.2: The proposed CapsDist architecture

Because capsules represent information at the neuron level as vectors rather than scalars, the final capsule layer's output is a volume of the shape $256 \times 256 \times 16$ (we use 16 element vectors in the CapsDist model). However, the distance map must be of the shape $256 \times 256 \times 1$. This requires the final capsule layer's output to be transformed by the Capsule Lengths layer to convert the 16 dimensional vectors into scalars. The Capsule Lengths layer does this by simply taking the length of each vector, which is a standard mathematical vector operation.

3.6 Implementation

We built our models with Python using the Keras (`https://keras.io/`) module with Tensorflow (`https://www.tensorflow.org/`) as our backend. We trained and evaluated our models on Google's Cloud AI Platform (`https://cloud.google.com/ai-platform/`). Our AI Platform project utilized 16 CPUs and 2 Nvidia V100 GPUs. We also made use of the University of Missouri System's HPC Lewis (`http://docs.rnet.missouri.edu/Services/hpc_compute`) cluster to build and evaluate 3D protein models.

3.7 Evaluation Metrics

We will evaluate the performance of our proposed CapsDist architecture against that of our baseline by comparing (a) the distance maps they produce and (b) 3D protein models constructed using these distance maps as input.

In order to compare a distance map produced by our baseline model for a given protein against one produced by our CapsDist model for the same protein, we must first score each distance map. We will produce these scores by computing the MAE, which is defined in

terms of a predicted distance map and a true distance map. The MAE is defined as the mean absolute error of the predicted distance map against the true distance map. Lower MAE scores are better.

Just as we must score distance maps to compare our CapsDist architecture against our baseline, we must also score 3D protein models. We will do so using the following three metrics, each of which measures the similarity between two protein tertiary structures:

- Template modeling score (TM-score). All TM-scores are between 0 and 1. A score of 1 indicates a perfect match.
- Root-mean-square deviation (RMSD). RMSD scores are continuous values with a minimum of 0 and unbounded maximum. Lower scores indicate greater similarity.
- Global distance test total score (GDTTS). GDTTS scores are continuous values with a minimum of 0 and maximum of 100. A score of 100 indicates a perfect match.

Chapter 4

Results

4.1 Capsule Layer to Residual Block Ratio

We tested 32 different variations of our CapsDist architecture, each with a different ratio of capsule layers to residual blocks. At each iteration, we trained and evaluated the network to find its best MAE score on the validation dataset. This allowed us to find the optimal number of residual blocks and capsule layers. The scores at each iteration are depicted in **Figure 4.1**.

From these results it is evident that networks built using a combination of residual blocks and capsule layers perform better than

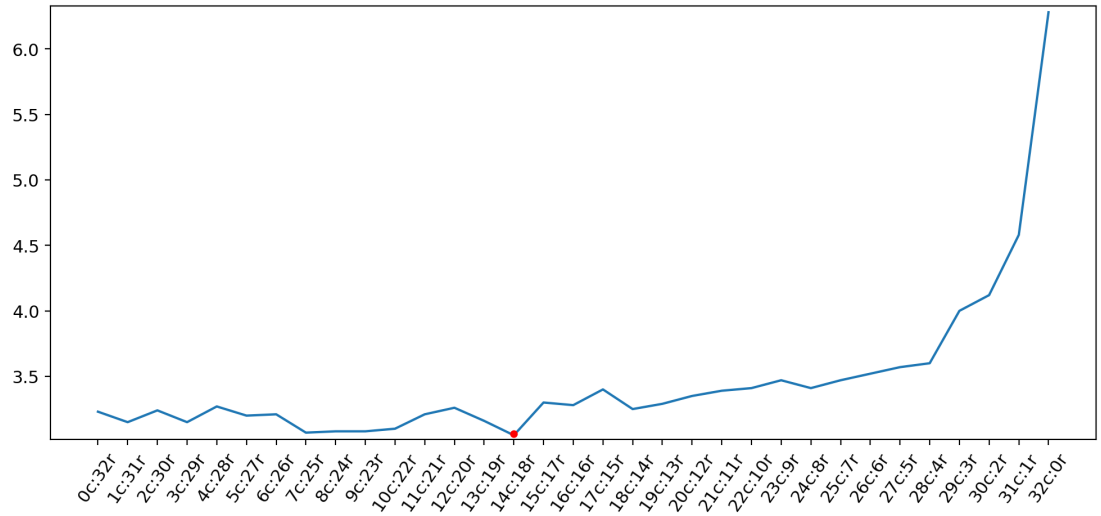

Figure 4.1: Capsule layer to residual block ratio

The lowest MAE achieved on the validation set for each variation of the CapsDist architecture. Note that the left-hand side of the colon denotes the number of capsule layers, whereas the right-hand side denotes the number of residual blocks. E.g. variation 14c:18r contains 14 capsule layers and 18 residual blocks.

networks that use only residual blocks or only capsule layers. Most networks with between 7 to 10 and 13 or 14 capsule layers perform quite well. The best performing network has 14 capsule layers.

4.2 Activation Function Performance

We trained and tested two variants of the CapsDist architecture with different capsule activation functions. One variant used the ReLU activation function (Xavier Glorot, Antoine Bordes, Yoshua Bengio, 2011) and the other used the identity function (equivalent

to removing the activation function entirely).

We find that the CapsDist architecture performs best with the ReLU activation function. **Table 4.1** lists the scores achieved by CapsDist with each of the activation functions.

Activation Function	**Best Validation MAE**	**Test MAE**
ReLU	3.21	1.61
Identity	3.37	1.68

Table 4.1: CapsDist performance by activation function

Performance of the CapsDist architecture with ReLU and Identity capsule activation functions.

4.3 Cost Function Impact

We trained our CapsNet architecture using 3 different cost functions: MAE, $LogCosh$, and $LogCoshInv$. We find that models trained using $LogCosh$ perform better than those trained with MAE. However, models trained with $LogCoshInv$ perform better than those trained with either MAE or $LogCosh$. We hypothesize that this is because $LogCoshInv$ gives more weight to errors for long-range residue pairs than for nearby ones. This is useful because the accuracy of predicted distances for long-range residue pairs is especially

important when predicting how a protein will fold. **Table 4.2** lists the scores achieved by CapsDist with each of the cost functions.

Cost Function	**Best Validation MAE**	**Test MAE**
MAE	3.43	1.90
$LogCosh$	3.38	1.83
$LogCoshInv$	3.21	1.61

Table 4.2: CapsDist performance by cost function

Performance of the CapsDist architecture trained with MAE, $LogCosh$, and $LogCoshInv$ cost functions.

4.4 CapsDist vs. Baseline Comparison

We trained our CapsDist and baseline models on our training dataset of 3284 proteins for 150 epochs. Our CapsDist model training job took 11.6 hours to complete. Our baseline model, however, took 13.2 hours. This disparity in training time is due to the number of trainable parameters in each model. Our CapsDist architecture has 672,634 trainable parameters, whereas our baseline architecture has 1,189,979. This helps to underscore the value of building models with fewer parameters.

It is important to note that the parameter reduction we achieved with our CapsDist architecture does not come at the price of accu-

racy. In fact, our CapsDist model scores better than the baseline model on both the validation and test datasets. See **Table 4.3** for the MAEs achieved by each model on the two datasets.

Architecture	**Best Validation MAE**	**Test MAE**
CapsDist	3.21	1.61
Baseline	4.16	3.29

Table 4.3: Performance of CapsDist and Baseline architectures

4.5 3D Model Comparison

We used DISTFOLD (Adhikari, 2019b) to build 3D protein models from distance maps produced by our CapsDist and baseline architectures. For each of the 150 proteins in our test dataset, 21 candidate models were produced, each with a TM-score, RMSD, and GDTTS. For each protein, we selected the candidate with the highest TM-score. We performed these steps once for each architecture. The results are summarized in **Table 4.4**. See **Table 5.1** in Appendix B for a listing of the full results. See Appendix A for visualizations of predicted 3D models for two of the proteins in the test dataset.

Model	**Avg TM-score**	**Avg RMSD**	**Avg GDTTS**
CapsDist	0.49	10.35	41.86
Baseline	0.48	10.52	40.28

Table 4.4: Average 3D model scores

Average scores for the 3D protein models created using distance maps produced from the CapsDist and baseline models.

Chapter 5

Conclusion

We have shown that convolutional capsule layers using the ReLU activation function improve the performance of deep learning networks designed to predict protein residue pairwise distance maps. These improved distance maps in turn enhance the overall performance of protein folding pipelines. In addition, we developed a new cost function tailored specifically for the PIDP problem: $LogCoshInv$. We have demonstrated that training networks with $LogCoshInv$ produces better performing networks than those trained with the more traditional MAE and $LogCosh$ cost functions.

Capsule networks hold significant potential to improve upon state-of-the-art deep learning networks in the field of protein modeling. We believe that our work has only begun to explore this potential.

Opportunities for future research include attempting to use multiple capsule types per layer, adding a reconstruction output to the network, and finding a way to increase the ratio of capsule layers to residual blocks without sacrificing model performance.

Appendix A

Figure 5.1 and **Figure 5.2** depict 3D visualizations of protein models predicted by the Baseline and CapsDist networks. Each figure contains two images (left and right), each of which contains two protein models (green and brown). In each image, the green model represents the protein's true structure. The brown models in the left images represent the best prediction of the Baseline network for each protein. The brown models in the right images represent the CapsDist network's best prediction for each protein.

The 3D visualizations in **Figure 5.1** and **Figure 5.2** were created using UCSF Chimera (Pettersen EF, Goddard TD, Huang CC, Couch GS, Greenblatt DM, Meng EC, Ferrin TE, 2004), developed by the Resource for Biocomputing, Visualization, and Informatics at the University of California, San Francisco, with support from NIH P41-GM103311.

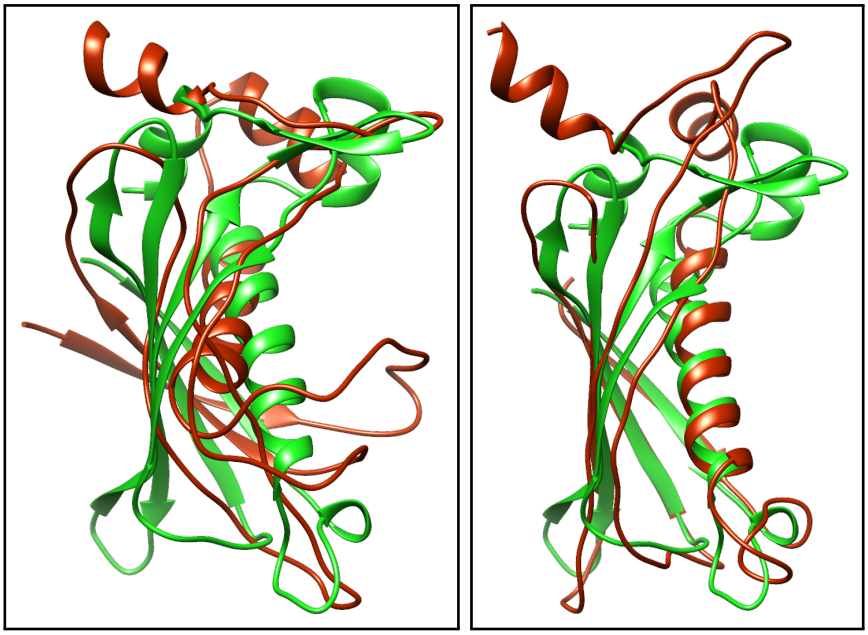

Figure 5.1: Protein 1lo7A - 3D Models

0.44 Baseline TM-score (left). 0.63 CapsDist TM-score (right). 19% improvement.

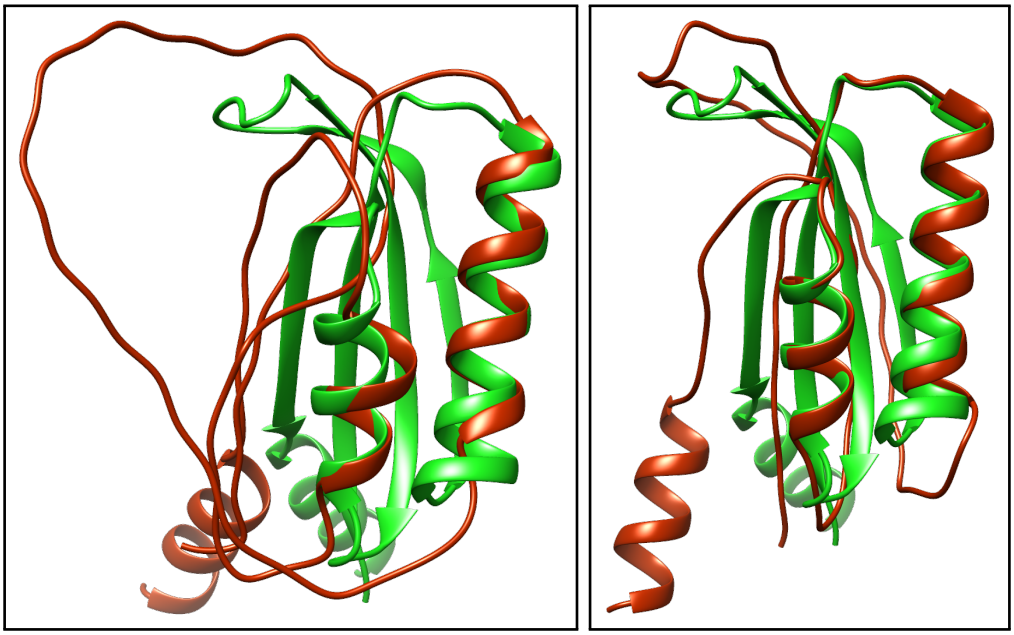

Figure 5.2: Protein 1vmbA - 3D Models

0.40 Baseline TM-score (left). 0.65 CapsDist TM-score (right). 24% improvement.

Appendix B

Protein	Baseline RMSD	CapsDist RMSD	Baseline TM-score	CapsDist TM-score	TM-score % Impr.
1abaA	10.14	5.30	0.4151	0.4751	+6%
1kidA	21.78	22.15	0.3507	0.3925	+4%
1fcyA	21.51	25.13	0.4890	0.4927	+0%
1dsxA	8.33	10.22	0.3422	0.3310	-1%
1jbeA	5.62	4.71	0.5943	0.7274	+13%
1o1zA	9.90	6.53	0.4978	0.5976	+10%
1ctfA	3.02	3.10	0.6597	0.7071	+5%
1avsA	7.15	9.87	0.3994	0.4158	+2%
1jvwA	14.35	8.63	0.4641	0.4889	+2%
1bdoA	6.12	6.19	0.4045	0.4092	+0%
1hxnA	26.64	31.20	0.2064	0.2003	-1%
1ku3A	5.35	3.93	0.5082	0.5841	+8%
2phyA	17.40	14.64	0.2812	0.2372	-4%
1d1qA	4.78	4.11	0.6489	0.6853	+4%
1jyhA	9.74	9.21	0.3683	0.3770	+1%
1xdzA	10.52	12.82	0.4802	0.5841	+10%
1gzcA	20.45	16.32	0.3409	0.3684	+3%
1j3aA	9.63	9.29	0.4425	0.4975	+5%
2hs1A	17.08	18.00	0.2333	0.1921	-4%
1jo0A	10.29	7.86	0.4497	0.5069	+6%
1rybA	5.13	5.51	0.6950	0.6847	-1%
5ptpA	4.51	4.90	0.7493	0.7007	-5%
1t8kA	3.73	4.70	0.6312	0.6170	-1%
1atzA	4.99	4.38	0.5591	0.5476	-1%
1fk5A	6.93	6.85	0.5681	0.4718	-10%
1wjxA	9.67	6.84	0.4275	0.4968	+7%
1kq6A	15.95	16.13	0.4406	0.4187	-2%
1bsgA	12.42	12.61	0.5603	0.5224	-4%
1cjwA	14.06	13.15	0.4770	0.4645	-1%
1i1jA	19.76	18.65	0.2765	0.2538	-2%
1pkoA	10.35	12.13	0.3890	0.4722	+8%
1guuA	4.34	5.65	0.4940	0.5009	+1%

1tifA	9.43	6.71	0.3773	0.3469	-3%
1fx2A	10.98	8.81	0.5034	0.5527	+5%
1josA	7.63	8.27	0.5679	0.4996	-7%
1h98A	13.05	14.91	0.2627	0.2740	+1%
1bebA	12.45	11.77	0.3166	0.3354	+2%
1brfA	11.24	10.83	0.2432	0.3043	+6%
1vmbA	9.69	4.55	0.4085	0.6501	+24%
1dixA	20.41	23.20	0.3063	0.2791	-3%
1c44A	19.46	8.59	0.2973	0.3890	+9%
1aoeA	7.38	5.73	0.5510	0.6557	+10%
1gz2A	6.23	5.50	0.5839	0.5983	+1%
1jfxA	7.43	11.37	0.5506	0.4872	-6%
1roaA	11.67	12.21	0.3867	0.3513	-4%
1i71A	10.93	7.71	0.3003	0.3164	+2%
1tzvA	3.85	3.88	0.7216	0.7882	+7%
1lo7A	7.50	5.50	0.4438	0.6318	+19%
1behA	15.36	16.35	0.3329	0.3257	-1%
1lpyA	11.03	8.34	0.3983	0.4586	+6%
1ql0A	17.44	19.54	0.3683	0.3417	-3%
1g2rA	8.75	7.49	0.4631	0.4877	+2%
1ihzA	5.47	5.29	0.5464	0.5896	+4%
1jl1A	7.96	7.34	0.6713	0.5929	-8%
2cuaA	16.98	17.95	0.3784	0.3187	-6%
1lm4A	11.16	11.03	0.5629	0.5454	-2%
1ej0A	4.14	5.53	0.7053	0.6795	-3%
1whiA	10.61	14.36	0.3073	0.2689	-4%
1qf9A	8.29	6.76	0.5977	0.6553	+6%
1a6mA	3.77	6.20	0.7298	0.6807	-5%
1h0pA	8.15	11.05	0.6417	0.5886	-5%
1qjpA	10.43	11.42	0.3414	0.2873	-5%
1m8aA	9.37	9.52	0.2866	0.3002	+1%
2tpsA	8.26	7.82	0.5931	0.6751	+8%
1i58A	7.48	6.13	0.4413	0.5960	+15%
1xkrA	14.33	24.08	0.2958	0.2567	-4%
1svyA	13.09	8.49	0.3715	0.4166	+5%
1g9oA	5.37	4.94	0.4765	0.4896	+1%
1jfuA	12.23	11.62	0.3945	0.5871	+19%

1vfyA	11.35	12.59	0.2772	0.2377	-4%
1f6bA	8.34	9.30	0.5958	0.5297	-7%
1fqtA	7.56	7.82	0.4731	0.4415	-3%
1tqgA	2.50	2.37	0.7735	0.8189	+5%
1fl0A	21.37	16.64	0.3131	0.3833	+7%
1xffA	7.99	5.75	0.5550	0.6616	+11%
1vhuA	7.05	7.43	0.6575	0.6629	+1%
1iibA	2.98	3.33	0.6875	0.6315	-6%
1cxyA	7.43	5.92	0.3364	0.3552	+2%
1i4jA	6.26	6.20	0.6101	0.5372	-7%
1fvkA	17.41	17.76	0.4656	0.4227	-4%
1k7cA	15.12	14.94	0.4754	0.4186	-6%
3dqgA	14.28	13.78	0.3166	0.3302	+1%
1tqhA	12.49	15.61	0.6081	0.6325	+2%
1rw7A	16.28	13.24	0.3967	0.5030	+11%
1h4xA	9.50	6.62	0.5338	0.5512	+2%
1a70A	16.95	11.10	0.3315	0.3165	-2%
1hdoA	8.53	6.95	0.6153	0.7445	+13%
1d0qA	5.51	9.08	0.5573	0.5266	-3%
2vxnA	4.31	4.34	0.7468	0.7572	+1%
1nb9A	8.42	9.21	0.5271	0.6070	+8%
1c9oA	9.13	9.17	0.3001	0.3369	+4%
1chdA	5.18	4.95	0.7285	0.7261	0%
1smxA	9.40	14.54	0.3254	0.2262	-10%
1m4jA	15.43	16.15	0.4183	0.3431	-8%
1cc8A	5.80	3.18	0.4752	0.6039	+13%
2arcA	19.51	28.95	0.3204	0.3110	-1%
1jwqA	3.51	3.13	0.7853	0.7783	-1%
2mhrA	11.22	8.36	0.6550	0.6104	-4%
1dqgA	34.04	20.31	0.1974	0.1704	-3%
1htwA	6.18	7.77	0.5865	0.5856	0%
1fvgA	20.25	19.75	0.5218	0.5639	+4%
1rw1A	4.69	4.84	0.6849	0.6132	-7%
1k7jA	7.78	10.47	0.6519	0.5665	-9%
1hfcA	8.11	11.35	0.4906	0.4019	-9%
1gmxA	4.84	5.93	0.6095	0.6917	+8%
1ny1A	23.41	23.73	0.4832	0.4916	+1%

1ej8A	11.15	8.42	0.2722	0.3638	+9%
1atlA	14.15	16.69	0.4345	0.3437	-9%
1kqrA	17.26	22.54	0.2939	0.2273	-7%
1ktgA	10.34	5.42	0.4876	0.5409	+5%
1aapA	6.74	7.92	0.3573	0.3818	+2%
1dbxA	5.02	7.18	0.6125	0.5749	-4%
1im5A	10.37	8.36	0.4295	0.6103	+18%
1fnaA	13.05	9.66	0.3835	0.3978	+1%
1ag6A	11.12	11.67	0.3830	0.3762	-1%
3borA	5.51	5.31	0.7009	0.7195	+2%
1mugA	9.08	9.82	0.4860	0.5003	+1%
1kw4A	5.72	5.32	0.4229	0.5077	+8%
1mk0A	10.88	13.72	0.4073	0.3450	-6%
1bkrA	3.82	4.21	0.6210	0.6095	-1%
1hh8A	28.46	25.99	0.2688	0.4226	+15%
1jo8A	7.54	13.43	0.2734	0.2882	+1%
1dmgA	18.48	18.09	0.3331	0.4020	+7%
1vjkA	7.34	7.18	0.3679	0.4648	+10%
1k6kA	2.94	2.87	0.7774	0.7568	-2%
1h2eA	9.70	7.97	0.5913	0.6951	+10%
1i1nA	11.61	9.40	0.5420	0.5858	+4%
1ckeA	8.29	13.62	0.5560	0.4082	-15%
1i5gA	12.39	12.38	0.3620	0.3812	+2%
1iwdA	7.69	6.97	0.5898	0.5916	+0%
1wkcA	8.28	5.77	0.6404	0.6329	-1%
1pchA	3.54	4.43	0.7098	0.6978	-1%
1cznA	11.56	6.94	0.5545	0.6091	+5%
1npsA	9.87	11.78	0.3583	0.2997	-6%
1gmiA	11.29	8.54	0.4262	0.4708	+4%
1ek0A	4.96	4.01	0.7139	0.7247	+1%
1dlwA	5.59	3.68	0.5967	0.6703	+7%
1w0hA	9.84	7.14	0.5196	0.6494	+13%
1c52A	14.84	16.63	0.3855	0.3639	-2%
1d4oA	9.93	8.48	0.4435	0.4836	+4%
1vp6A	12.64	12.40	0.4440	0.4933	+5%
1a3aA	4.13	4.89	0.6680	0.6312	-4%
1r26A	5.71	7.24	0.6481	0.5690	-8%

1jkxA	8.48	8.33	0.6121	0.6837	+7%
1jbkA	8.06	11.08	0.4317	0.4040	-3%
1p90A	7.51	10.16	0.4786	0.3840	-9%
1eazA	5.88	6.23	0.4312	0.4833	+5%
1nrvA	10.63	10.74	0.2887	0.3635	+7%
1gbsA	23.94	27.27	0.4734	0.3496	-12%
1ne2A	21.33	18.18	0.3127	0.4403	+13%
Average	**10.52**	**10.35**	**0.48**	**0.49**	**+1.45%**

Table 5.1: Exhaustive 3D model scores

Complete list of TM and RMSD scores for the 3D models built with CapsDist and Baseline models, for each of the 150 proteins in our test dataset.

Bibliography

Adhikari, B. (2019a). Deepcon: Protein contact prediction using dilated convolutional neural networks with dropout. *bioRxiv*.

Adhikari, B. (2019b). Distfold. `https://github.com/badriadhikari/DISTFOLD`.

Adhikari, B. (2019c). Ieee icmla 2019 pidp challenge. `https://github.com/badriadhikari/IEEE-ICMLA-2019-PIDP-Challenge`.

Alberts B, Johnson A, Lewis J, et al. (2002). The shape and structure of proteins. `https://www.ncbi.nlm.nih.gov/books/NBK26830/`.

Andrew S, John J, Demis H (2018). Alphafold: Using ai for scientific discovery. `https://deepmind.com/blog/article/alphafold`.

G. E. Hinton, A. Krizhevsky, S. D. Wang (2011). Transforming auto-encoders. `https://www.cs.toronto.edu/~hinton/absps/transauto6.pdf`.

Jones DT, Singh T, Kosciolek T, Tetchner S (2015). Metapsicov: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. `https://www.ncbi.nlm.nih.gov/pubmed/25431331`.

Kaján, László and Hopf, Thomas A. and Kalaš, Matúš and Marks, Debora S. and Rost, Burkhard (2014). Freecontact: fast and free software for protein contact prediction from residue co-evolution. *BMC Bioinformatics*.

Kandathil Shaun, Greener Joe, Jones David (2019). Prediction of interresidue contacts with deepmetapsicov in casp13. *Proteins: Structure, Function, and Bioinformatics*.

Lab, S. (2018). Ccmpred. https://github.com/soedinglab/CCMpred.

Leavitt SA (2010). Deciphering the genetic code: Marshall nirenberg. https://history.nih.gov/exhibits/nirenberg/glossary.htm. [Online; accessed 28-September-2019].

Li Yang, Hu Jun, Zhang Chengxin, Yu Dong-Jun, Zhang Yang (2019). Respre: high-accuracy protein contact prediction by coupling precision matrix with deep residual neural networks. https://www.ncbi.nlm.nih.gov/pubmed/31070716.

Lodish H, Berk A, Zipursky SL, et al. (2000). Molecular cell biology. 4th edition - section 18.3, myosin: The actin motor protein. https://www.ncbi.nlm.nih.gov/books/NBK21724/. [Online; accessed 19-October-2019].

Marengo-Rowe AJ (2006). Structure-function relations of human hemoglobins. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1484532/. [Online; accessed 19-October-2019].

McGuffin LJ, Bryson K, Jones DT (2000). The psipred protein structure prediction server. https://www.ncbi.nlm.nih.gov/pubmed/10869041.

National Research Council (US) Subcommittee on the Tenth Edition of the Recommended Dietary Allowances (1989). Protein and amino acids. https://www.ncbi.nlm.nih.gov/books/NBK234922/. [Online; accessed 19-October-2019].

Pettersen EF, Goddard TD, Huang CC, Couch GS, Greenblatt DM, Meng EC, Ferrin TE (2004). Ucsf chimera - a visualization system for exploratory research and analysis. https://www.ncbi.nlm.nih.gov/pubmed/15264254.

Rodney L, Ulas B (2018). Capsules for object segmentation. https://arxiv.org/abs/1804.04241.

Sara S, Nicholas F, Geoffrey H (2017). Dynamic routing between capsules. https://arxiv.org/abs/1710.09829.

Science (2005). So much more to know. https://science.sciencemag.org/content/309/5731/78.2.

Sunil Prakash, Gaelan Gu (2018). Simultaneous localization and mapping with depth prediction using capsule networks for uavs. https://arxiv.org/abs/1808.05336.

Xavier Glorot, Antoine Bordes, Yoshua Bengio (2011). Deep sparse rectifier neural networks. http://proceedings.mlr.press/v15/glorot11a.html.

Xu, J. (2018). Distance-based protein folding powered by deep learning. https://arxiv.org/abs/1811.03481.