

© Copyright 2019

Spencer Moritz

Deep Learning to Predict Protein Backbone Structure from High-Resolution Cryo-EM Density Maps

Spencer Moritz

A thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Computer Science & Software Engineering

University of Washington

2019

Committee:

Dong Si

Wooyoung Kim

Clark Olson

Program Authorized to Offer Degree:

Computing and Software Systems

University of Washington

Abstract

Deep Learning to Predict Protein Backbone Structure from High-Resolution Cryo-EM Density Maps

Spencer Alexander Moritz

Chair of the Supervisory Committee:
Assistant Professor Dong Si
Computing and Software Systems

Understanding a protein's structure can lead to the discovery of therapeutic protein drugs. However, imaging proteins remains a challenge due to their small size. Cryo-electron microscopy (cryo-EM) is a leading imaging technology that has recently been able to produce near atomic resolution images called electron density maps. However, predicting various protein structures remains a challenge on all but the most pristine density maps ($< 2.5\text{\AA}$ resolution). Here we introduce a deep learning model that uses a set of cascaded convolutional neural networks (CNNs) to predict the critical $C\alpha$ atoms along a protein's backbone structure. The cascaded-CNN (C-CNN) is a novel deep learning architecture comprised of multiple CNNs, each predicting a specific aspect of a protein's structure. This model predicts various levels of protein structures: secondary structure elements (SSEs), backbone structure, and $C\alpha$ atoms. It combines the results of each to

produce a complete prediction map. The cascaded-CNN is a semantic segmentation image classifier and was trained using thousands of simulated density maps. This method is largely automatic and only requires a recommended threshold value for each evaluated protein. A specialized tabu-search path walking algorithm was used to produce an initial backbone trace with C α placements. This method was tested on 50 experimental maps between 2.6Å and 4.4Å resolution. It outperformed several state-of-the-art prediction methods including RosettaES, MAINMAST, and a Phenix-based method by producing the most complete prediction models, as measured by percentage of found C α atoms. This method accurately predicted 88.5% (mean) of the C α atoms within 3Å of a protein's backbone structure surpassing the 66.8% mark achieved by the leading alternate method (Phenix based fully automatic method) on the same set of density maps. The C-CNN also achieved an average RMSD of 1.23Å for all 50 experimental density maps which is similar to the Phenix based fully automatic method. The source code and demo of this research has been published at <https://github.com/DrDongSi/Ca-Backbone-Prediction>.

TABLE OF CONTENTS

List of Figures	viii
List of Tables	xi
Chapter 1. Introduction	1
1.1 Motivation.....	1
1.2 Stakeholders	2
1.3 Cryogenic Electronic Microscopy (Cryo-EM)	3
1.4 Protein Backbone Structure	3
1.5 Protein Secondary Structure Detection	4
1.6 Problem Definition.....	5
Chapter 2. Background	6
2.1 Current Protein Prediction Models	6
2.2 Deep Learning Semantic Segmentation.....	8
Chapter 3. development methods.....	10
3.1 Data Collection/Generation	10
3.2 Initial Models	12
3.3 C α Atom Convolutional Neural Network.....	14
3.4 Cascaded Convolutional Neural Network	15
3.5 Convolutional Neural Network Architecture.....	16
3.6 End-To-End Model Pipeline.....	19

3.7	Pre-Processing.....	20
3.8	Network Input	21
3.9	Network Output	21
3.10	Path-Walking Algorithm.....	23
3.11	Graph Refinement.....	24
3.12	Computation.....	26
Chapter 4. results		26
4.1	Metrics	27
4.2	Simulated Density Maps.....	28
4.3	Experimental Density Maps.....	29
4.4	Comparison of Prediction Methods	33
4.5	Computation Time of Prediction Models	34
4.6	Comparison of C-CNN and the Fully Automatic Phenix Method.....	35
Chapter 5. discussion		37
5.1	False Positive $C\alpha$ Error Rate.....	37
Chapter 6. conclusions		39
6.1	Contributions of Research.....	39
6.2	Future Improvements	39
6.3	Conclusion	40
Chapter 7. Works Cited.....		41

LIST OF FIGURES

- Figure 1.1 Unmodified CCR5 receptor (yellow) protruding from a CD4 cell. The surrounding structure is the cell's membrane. The human immunodeficiency virus (HIV) uses a protein on its surface to bond to the CCR5 receptor to infect the CD4 cell..... 2
- Figure 1.2 Simulated Density Maps from protein 1a9h at different resolutions. (A) shows a high-resolution map with underlying backbone trace. (B) shows a medium-resolution map with underlying ribbon structure. α -helix structures are colored red, β -sheets are colored blue, and the turns/loops are colored yellow. 4
- Figure 1.3 (A) contains a snapshot of the unprocessed density map, EMD-6555. (B) is the goal output for EMD-6555 which shows the ground truth backbone trace with the $C\alpha$ atoms represented as small spheres along the backbone. The input density map is overlaid for reference..... 6
- Figure 3.1 Simulated Data Generation Pipeline including details about the data normalization process. The output of this pipeline was an HDF5 file containing all the data used to train the prediction model. 12
- Figure 3.2 Simple auto-encoder CNN architecture..... 13
- Figure 3.3 $C\alpha$ Atom CNN architecture. The input was a 64x64x64 3D matrix of electron density. The output was a $C\alpha$ cloud representing areas of high confidence in $C\alpha$ atoms. More details about this architecture are in the Convolutional Neural Network Architecture section. 14
- Figure 3.4 $C\alpha$ cloud prediction output from the $C\alpha$ Atom CNN shown in purple. The orange structure is the ground truth backbone trace. 15
- Figure 3.5 Cascaded Convolutional Neural Network. The input/output of each stage is shown as a gray cube with the given dimensions. Each CNN is represented by a tapered salmon rectangle. Results from each CNN are forwarded along with previous input data to the next CNN. 16
- Figure 3.6 Detailed architecture of each of the 3D convolutional neural networks (CNN). (A) contains the Secondary Structure Elements (SSE) CNN. (B) contains the Backbone CNN.

(C) contains the C α -Atom CNN. Each CNN, including all its layers, are shown within the salmon colored boxes. The input to each CNN is noted by the yellow cubes. The Backbone CNN (B) and the C α -Atom CNN (C) take input that is a contamination of various maps. The size of each input is noted by the dimensions listed at the base of each input cube. Each layer in each CNN is denoted by its name/function, kernel size (N \times N \times N), and finally the output number of filters (inside parenthesis) for that layer. Each leaky ReLU activation function used an alpha value of 0.1. The output of each CNN is noted by the purple cube. The dimensions for each output are listed at the base of the cube..... 18

Figure 3.7 Full Backbone Prediction Model. Model includes data preprocessing, the cascaded convolutional neural network (after training), and post-processing. 20

Figure 3.8 Confidence Map Output for EMDB-8410 (Chain-A). (A) is the combination of the α -helix and β -sheet prediction map after applying the max function, (loops/turns map omitted for readability). (B) is the backbone confidence map (>40% confidence) with the ground truth backbone structure shown for reference. (C) is the C α -Atom confidence map (>50% confidence) with the ground truth ball and stick representation of EMDB-8410 shown for reference..... 22

Figure 4.1 Final Backbone Prediction structures from the simulated 6eyw protein map. Ground Truth is colored teal while the predicted structures are colored tan. (A) contains the C α only backbone structures with the input density map overlaid on the image for reference. (B) displays the final ribbon prediction. The ribbon was enhanced using the secondary structure output maps from the Cascaded Convolutional Neural Network. 29

Figure 4.2 Final Backbone Prediction maps for various density maps. (A) EMDB-6272 (chain-A) at resolution 2.6Å. (B) EMDB-8410 (chain-A) at resolution 3.5Å (C) EMDB-5778 (chain-A), resolution 3.3Å. The left map in each subfigure contains the predicted vs. actual backbone structure. The right map in each subfigure contains the predicted vs. actual ribbon structure of the protein which specify the SSE classification. The pink trace is the predicted structures while the blue trace is the actual structure..... 32

Figure 4.3 Final Prediction Model for EMDB-5778 (chain-A) at 3.28Å resolution. (A) Our C-CNN method. (B) automatic Phenix method. (C) MAINMAST. (D) RosettaES. (E) The

ground truth model. Each figure overlays the input density map on top of the predicted structure. (A) Notes five areas where the C-CNN outperformed the Phoenix method.³³

Figure 4.4 Plot of RMSD as a function of resolution for the 50 experimental density maps. This compares our deep learning C-CNN method and the fully automatic Phenix Method.³⁵

Figure 4.5 Plot of Percentage of Found C α atoms as a function of resolution for the 50 experimental density maps. This compares our deep learning C-CNN method and the fully automatic Phenix Method. 36

Figure 5.1 Ball and Stick representation of the prediction map (pink) and the ground truth map (blue) of EMDB-8642, resolution 3.9Å. This map had a relatively low RMSD value of 1.38Å but also had a false positive C α rate of 8.3%. The expanded areas show examples of side chains that were incorrectly predicted as backbone structure resulting in false-positive C α placements. 37

Figure 5.2 Error-Rate vs. resolution for each experimental prediction map with an added exponential trendline..... 38

LIST OF TABLES

Table 4.1 Results of simulated data.	28
Table 4.2 Results from the evaluation of experimental density maps with the Phenix method's results shown as a benchmark.	30
Table 4.3 Comparison of EMDB-5778 (chain-A) among leading backbone prediction models. Each model had specific strengths, however the deep learning C-CNN produced the most complete model as measured by C α matching percentage.	34

ACKNOWLEDGEMENTS

I want to thank each of my collaborators from across the country: Renzhi Cao, Ligu Wang, Tianqi Wu, Jianlin Cheng, and Jie Hou. Your input throughout the research helped improve the overall quality and direction of this work. I also want to thank my friend and fellow graduate student Jonas Pfab for working on a separate α -helix optimization method and working tirelessly to organize the code into a public GitHub account. Finally, I want to thank the Chair and Advisor of this research, Dong Si for helping to develop my understanding of molecular biology and pushing me to achieve state-of-the-art results in this research field.

Chapter 1. INTRODUCTION

Proteins perform a vast array of functions within organisms. From molecule transportation, to mechanical cellular support, to immune protection, proteins are the central building blocks of life in the universe. Despite each protein being composed from a combination of the same 20 naturally occurring amino acids, a protein's functionality is mainly derived from its unique three-dimensional (3D) shape. Therefore, learning the details of a protein's 3D structure is a prerequisite to understanding its biological function.

1.1 MOTIVATION

There are many motivations for understanding the 3D structure of proteins. One such motivation is that knowing the structure of a protein makes it possible to develop synthetic protein molecules to aid those with specific medical conditions. Diabetes mellitus is one such condition that can be managed with synthetic insulin. Diabetes is caused when a person produces less of the insulin protein than is needed by their body. Insulin, a peptide hormone, is a protein that is shaped to perfectly bind with the insulin receptor on the surface of cells. When insulin bonds with a cell, the cell responds by opening its glucose channel to allow glucose molecules into the cell, thereby providing the cell with energy. By learning the shape of the insulin protein, it has become possible to develop synthetic insulin that works in the same way as naturally produced insulin.

Another motivation for this research is that it can produce breakthroughs in immunology. Viruses have proteins on their outer membrane that they use to infect living cells. These proteins bond to the membrane of a host cell in order to begin the infection process. By learning the 3D shape of these proteins, it is possible to develop therapies that prevent the infection process entirely. One such example of this involves the human immunodeficiency virus (HIV) which has

a specific protein on its surface that bonds to the CCR5 protein on the surface of CD4 immune cells. In 2018, the research scientist He Jiankui used a gene editing tool called CRISPR to mutate the CCR5 protein on the surface of the CD4 cells thereby preventing HIV from being able to enter the cell. This effectively made the cells immune to HIV. This procedure was particularly groundbreaking because it was performed on unborn embryos and is therefore considered gene-editing. While the ethically of this gene-editing is still up for debate, it would not have been possible to perform this procedure without first understanding the shape of HIV's membrane protein used to infect CD4 cells.

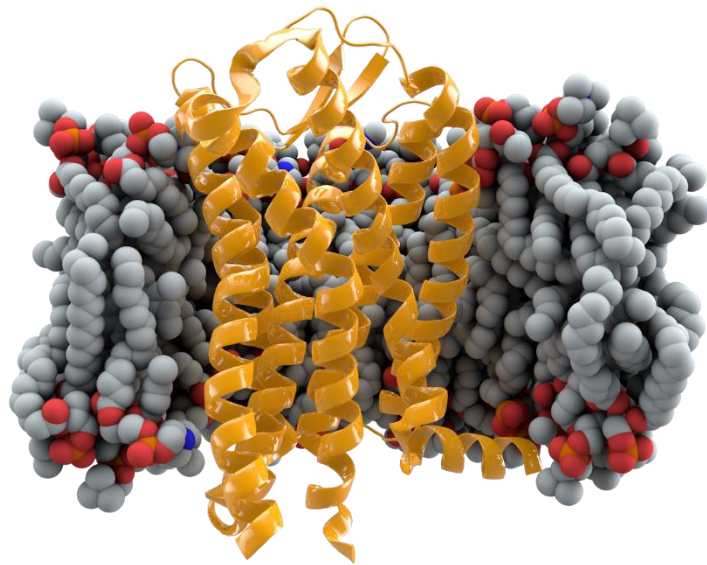


Figure 1.1 Unmodified CCR5 receptor (yellow) protruding from a CD4 cell. The surrounding structure is the cell's membrane. The human immunodeficiency virus (HIV) uses a protein on its surface to bond to the CCR5 receptor to infect the CD4 cell.

1.2 STAKEHOLDERS

The primary beneficiaries of this research include the many different areas of Molecular Biology. The field of molecular biology pertains to the structure and function of macromolecules, such as proteins and nucleic acids, which are essential to life. Learning the 3D structure of proteins

will aid in the development of therapeutic protein drugs such as synthetic insulin. Additionally, this research will help unlock the mechanisms by which newly discovered pathogens infect their host cells. This could lead to the development of vaccines and other drug therapies.

1.3 CRYOGENIC ELECTRONIC MICROSCOPY (CRYO-EM)

Currently, one of the leading techniques for determining the atomic structure of proteins is cryogenic electron microscopy (cryo-EM). Cryo-EM uses a high-energy electron beam to image biological specimens. The process begins with liquid nitrogen vitrification. This rapid cooling of the specimen prevents water molecules from crystalizing, thereby preventing damage to the structural components of the specimen. Once cooled, the sample is then imaged with an electron beam from multiple angles. Each angle produces a 2D image of the specimen. The final step of the cryo-EM process combines the 2D images from each angle to reconstruct a 3D representation of the specimen.

In the past five years, more than 1,000 protein structures have been imaged using cryo-EM at 4Å (1Å=10⁻¹⁰m) resolution or better. Among them, many are of detergent-solubilized membrane proteins [1] [2] [3] [4] [5] [6]. These high-resolution images make it possible to produce atomic level 3D models from the density maps. Proteins imaged with cryo-EM are represented as 3D matrices where each voxel (3D pixel) contains an electron density scalar value. These density maps are publicly available and can be downloaded from the EM databank¹.

1.4 PROTEIN BACKBONE STRUCTURE

From clean, high-resolution EM density maps (< 5Å) it is possible to distinguish the backbone structure of a protein [7] [8] [9]. A protein's backbone is a continuous chain of atoms that runs

¹ <https://www.emdataresource.org/search.html>

throughout the length of a protein, see Figure 1.2A The backbone structure consists of a repeated sequence of three atoms (carbon, nitrogen, alpha-carbon). Of these three atoms, the alpha-carbon ($C\alpha$) is particularly important as it is the central point for each amino acid residue within the protein. Therefore, predicting not only a protein's backbone but also the locations of each $C\alpha$ along that backbone can help determine where specific amino acids are located throughout the protein structure.

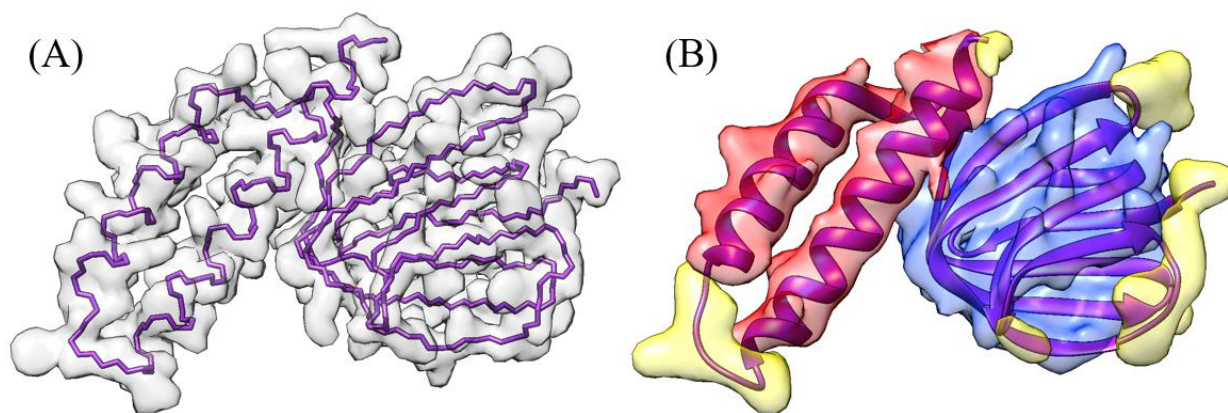


Figure 1.2 Simulated Density Maps from protein 1a9h at different resolutions. (A) shows a high-resolution map with underlying backbone trace. (B) shows a medium-resolution map with underlying ribbon structure. α -helix structures are colored red, β -sheets are colored blue, and the turns/loops are colored yellow.

1.5 PROTEIN SECONDARY STRUCTURE DETECTION

In addition to the backbone features of a protein, some of the most visually dominant features of cryo-EM density maps are the secondary structure elements (SSEs), see Figure 1.2B. The three SSEs are α -helices, β -sheets, and turns/loops. At medium resolution, α -helices appear as long cylinders with a radius of approximately 2.3\AA . β -sheets consist of multiple parallel beta strands that connect laterally by hydrogen bonds. While only distinguishable at 6\AA resolution or better, β -sheets appear as flat or slightly wavy sheets. Turns/Loops are the final SSE. They occur in locations

where the polypeptide chain of the protein reverses its overall direction. When imaged with cryo-EM, turns/loops often appear faint due to their relatively low electron density. This makes them one of the most challenging SSE to classify.

There are many methods for identifying SSEs at medium resolutions [10] [11] [12] [13] [14] [15] [16]. However, at higher resolution ($< 2.5\text{\AA}$), the classic α -helix and β -sheet structures are not easily recognizable to the human eye. This is due to the large number of side chains that protrude off the backbone chain in high-resolution data. This makes predicting the SSEs at high-resolution potentially more difficult than at medium resolution.

1.6 PROBLEM DEFINITION

The goal of this research is to create a full end-to-end model that can predict the backbone structure and $C\alpha$ atoms of proteins from cryo-EM density maps, see Figure 1.3. Additionally, the model should be as automatic as possible, meaning that it will require few or no parameters to evaluate a density map. The core of the model will use a semantic segmentation image classifier composed of convolutional neural network(s) (CNN). In addition to the deep learning CNN, this model will employ the necessary preprocessing and postprocessing techniques to achieve state-of-the-art results similar or better than existing prediction models.

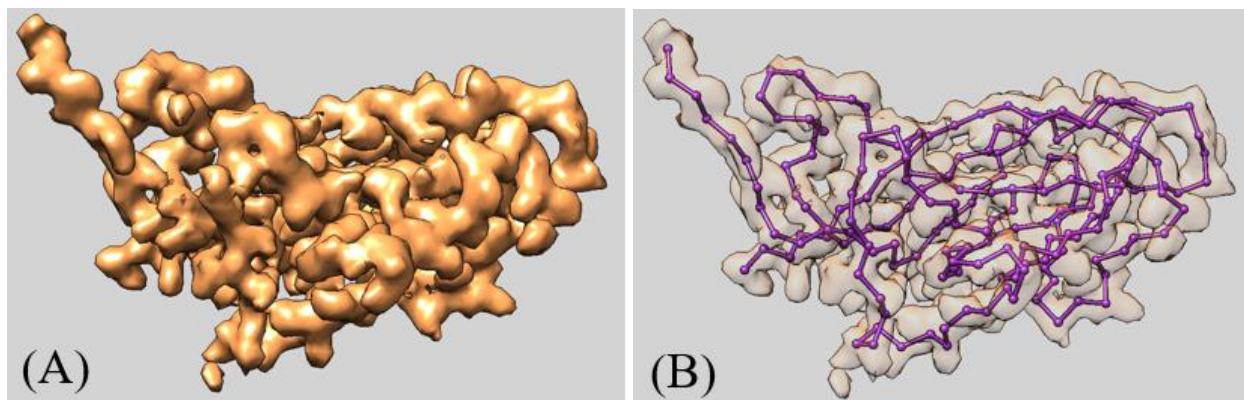


Figure 1.3 (A) contains a snapshot of the unprocessed density map, EMD-6555. (B) is the goal output for EMD-6555 which shows the ground truth backbone trace with the C α atoms represented as small spheres along the backbone. The input density map is overlaid for reference.

Chapter 2. BACKGROUND

2.1 CURRENT PROTEIN PREDICTION MODELS

Ever since the first experimental density maps were released for protein structures, researchers have been developing software models to predict the various structural elements from each map. Some of the leading software models are now able to predict the atomic structure of a protein from its electron density map.

Phenix is a widely used molecular prediction software suite that has often been used in research since its initial release in 2010 [17]. A recent 2018 paper introduced a new molecular prediction method that combined the Phenix prediction software along with advanced post-processing techniques [18]. This method, henceforth referred to as the Phenix method, produced some of the most-complete prediction models. As a result, we used this method as a metric benchmark for this research.

The Phenix method is a fully-autonomous prediction method that only requires a density map and a nominal resolution value as input. This method first sharpens the input density map using an

automated map sharpening algorithm which aims to maximize the connectivity of high-density regions [19]. Then, for each part of the structure, various atomic models are generated using several independent prediction models, including one for SSEs and one for backbone tracing, among others [20] [21] [22]. The results from these predictions are combined and used to produce an initial predicted structure. This structure is then refined using any symmetry that is present in the protein. The Phenix method was tested on 476 experimental density maps and has, to date, produced the most complete prediction maps. This method also uses a unique set of metrics to measure the effectiveness of the prediction method. The RMSD method uses a one-to-one mapping of predicted to ground-truth C α atom but only includes atoms that are within 3Å of the ground truth model. To measure prediction model completeness, the Phenix method calculates the percentage of matching C α atoms between the predicted and ground-truth model within the same 3Å space. We use the same metrics when evaluating our deep-learning prediction technique.

RosettaES is a protein modeling software tool first developed at the University of Washington. RosettaES employs a modeling technique which consists of two general components: conformation sampling and energy evaluation [23]. Conformational sampling uses well-established physical characteristics of molecular structure as guides for model prediction. Examples of such characteristics include: the common torsion angles of atoms in the backbone structure, or the radius of α -helix secondary structures. Each of these structures has a very narrow band of potential values making them excellent constants to use when modeling protein structure. The energy evaluation process calculates the total energy of a predicted protein based on each predicted atom position along with each bonding angle between them. This value is compared to the expected lowest-energy state, which can be calculated from sequence information. Given that the lowest-energy state is likely closest to the native state of the protein, slight adjustments are

made to the predicted protein structure to minimize the energy within its atomic structure thereby optimizing the prediction map for the protein.

Another leading backbone prediction model is the MAINMAST algorithm developed by researchers at Purdue University [24]. MAINMAST produces a backbone trace, consisting of a set of $C\alpha$ atoms, from high density regions of an electron density map. This algorithm first identifies regions of high-density (high-density points are likely to be backbone structure) using mean shifting and then transforms them into a minimum spanning tree (MST). A Tabu search algorithm is applied to find a few thousand possible MSTs. For each MST, the amino acid sequence is mapped on the longest path in the tree using areas of high density as likely $C\alpha$ atom locations. Each MST is rated based on the best fit. The highest scoring tree is chosen as the final prediction of the model.

2.2 DEEP LEARNING SEMANTIC SEGMENTATION

This research aimed to use deep learning to create a predictive model capable of detecting the backbone structure and $C\alpha$ atoms from electron density maps. The field of deep learning has proven to be very successful in the fields of image recognition and image classification [25] [26] [27]. The goal of regular image classification is to assign a single label to an input image. However, our problem is different because we want to assign a label to each pixel of the input image. This type of image classification is called semantic segmentation. With semantic segmentation, each 2D-pixel or 3D-voxel of an image is classified independently.

Until recently, semantic segmentation was accomplished through patch classification. Patch classification takes a slice of the input data and runs it through a convolutional neural network (CNN). However, patch classification only classifies the center pixel of each patch meaning that the CNN would have to process a new patch for each pixel in the image. This technique is preferred

when computing resources are limited because processing a small patch is much less computationally expensive than processing a full image.

However, with the recent advances in GPU technology, fully connected end-to-end networks are now able to perform semantic segmentation on full images in one pass. In 2014, research at UC Berkeley used a Fully Convolutional Network (FCN) to perform semantic segmentation on the PASCAL-Context dataset [28]. Their method used an encoder-decoder architecture that removed the need for patch classification by essentially combining the calculations of the overlapping patch regions into a single end-to-end network. In 2015, the network Segnet aimed to improve the encoder-decoder architecture by forwarding the max-pooling indices from the encoder layer to the decoder layer to prevent the loss of global information in the image [29]. Later that year, researchers at Princeton University used a technique called dilated convolution which made it possible to perform semantic segmentation without the encoder-decoder architecture [30]. Dilated convolutions are preferred when a convolution needs to increase the field of view without reducing the resolution of the image.

This research leveraged the architecture of each of these semantic segmentation classifiers. Previous deep learning methods that were used for SSE prediction used patch classification [11]. Our model levered the Fully Connected Network design to eliminate the need for patch classification and instead use semantic segmentation to classify a full 3D image in a single pass. It also used data forwarding, inspired by Segnet, to allow for segregated learning. Finally, this model used dilated convolutions to increase the field of view while maintaining the input image dimensionality.

Chapter 3. DEVELOPMENT METHODS

3.1 DATA COLLECTION/GENERATION

Predictive models are only successful if they are trained with representative data. Since, experimental high-resolution cryo-EM maps are still somewhat scarce, we trained our model using simulated cryo-EM maps instead of experimental maps. This design decision allowed us to save all the experimental maps for verification.

Simulated cryo-EM maps can be generated from solved proteins using a script from the EMAN2 package called `pdb2mrc` [31]. The Protein Data Bank² (PDB) contains thousands of solved proteins stored as PDB files. The `pdb2mrc` script works by taking each atom in a PDB file and simulating a 3D Gaussian electron density. It then sums the Gaussian density of all the simulated atoms on a 3D grid to produce a complete electron density map for the entire protein. This simulation method produces electron density maps that are very representative of their experimental counterparts with the primary difference being that a simulated map has no experimental inaccuracies.

To produce a large enough dataset for training, we used over 7,000 PDB files to generate simulated density maps using the `pdb2mrc` script. Each map was simulated at a different resolution to produce a higher amount of variance in the training set and prevent overfitting in the model. In addition to simulating the full PDB structure, we also had to simulate the labeled data for each map. This step involved editing the PDB file to retain only atoms that were part of the predicted structure and running `pdb2mrc` to produce a labeled map for each structure being predicted. When selecting PDB files, we selected maps with a sequence structure that was at least 50% unique

² <https://www.rcsb.org/>

relative to all other maps in the training set. This ensured that our training data was diverse and well-representative of the large range of protein structure found in nature.

The data generation pipeline is outlined in Figure 3.1. The output of this pipeline was two hierarchical data files (HDF5) files: one training file and one testing file. The training file consisted of approximately 24,000 simulated density maps³ along with the corresponding SSE, backbone, and C α labeled maps for each protein. The testing HDF5 file, which was generated to measure the accuracy of the neural network as it trained, consisted of 1,024 maps along with the same configuration of labeled maps.

To ensure uniformity among each electron density map, extensive data normalization was used to produce a common input data format. There were five data normalization steps, see Figure 3.1. First, all voxels with an electron density less than a resolution dependent threshold were zeroed out⁴. This removed low intensity areas of the simulated maps which differed significantly from their experimental counterparts. It also allowed the neural network to exclusively train with voxels of high-intensity, which removed much of the noise in the image. After this step, all voxel intensity values were reduced by a threshold value so that the spectrum of electron densities within each map started at zero. The third step of data normalization involved dividing all voxels by the median voxel value in the electron density map. This step normalized the voxel values and ensured that each map had a similar data range. After this, data outliers were removed by capping all voxels at the 98th-percentile voxel intensity. Finally, each training map was copied and rotated by a varied angle to increase the total training data size. This entire normalization process helped transform the large range of data among the variety of density maps into a more uniform distribution.

³ Approximately 6000 unique maps were used for training. Each was rotated by 0°, 90°, 180° and 270° to increase the training data size by 4x.

⁴ A cutoff threshold was selected for each simulated resolution. At each resolution all voxels less than the threshold were set to zero in order to remove low intensity noise from the density map.

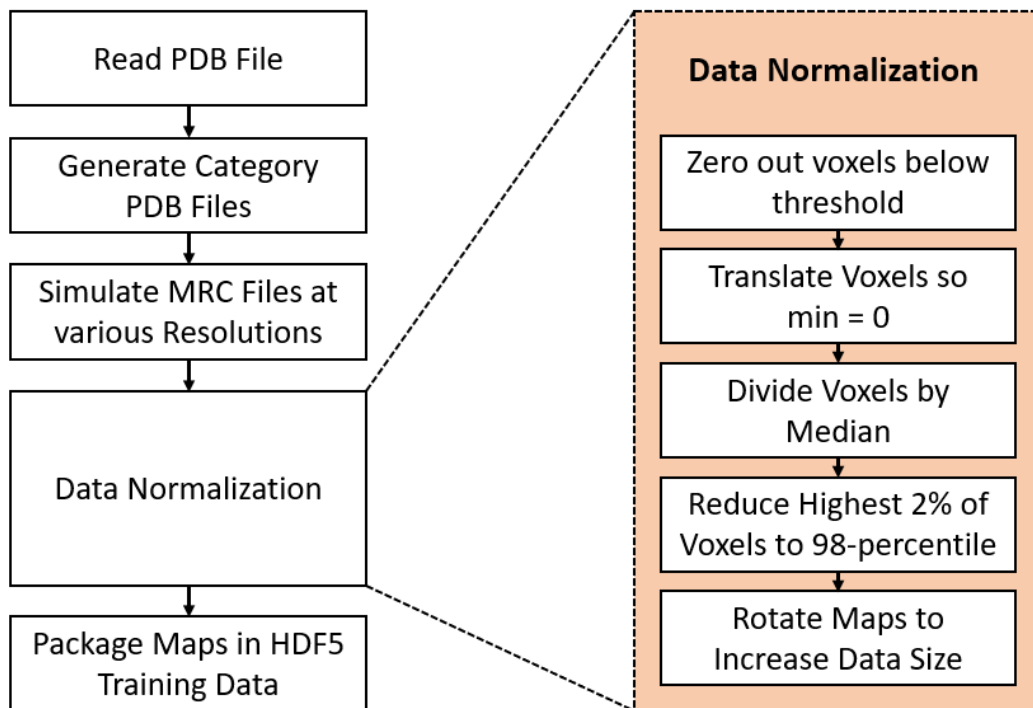


Figure 3.1 Simulated Data Generation Pipeline including details about the data normalization process. The output of this pipeline was an HDF5 file containing all the data used to train the prediction model.

3.2 INITIAL MODELS

Since previous students had used the Python based TensorFlow API in their machine learning projects, I also decided to use the Python version of TensorFlow to develop this deep learning model.

After getting acquainted with TensorFlow, the first attempted semantic segmentation classifier was an auto-encoder CNN, see Figure 3.2. This architecture used a series of convolution and pooling layers to compress the input into an encoded representation. Then it would expand the encoded representation using convolution and up-sampling layers. By compressing the data in the encoder section, the network should learn how to predict only relevant information from the input

image. Then in the decoder section, the network would distribute the relevant information to produce the desired output image.

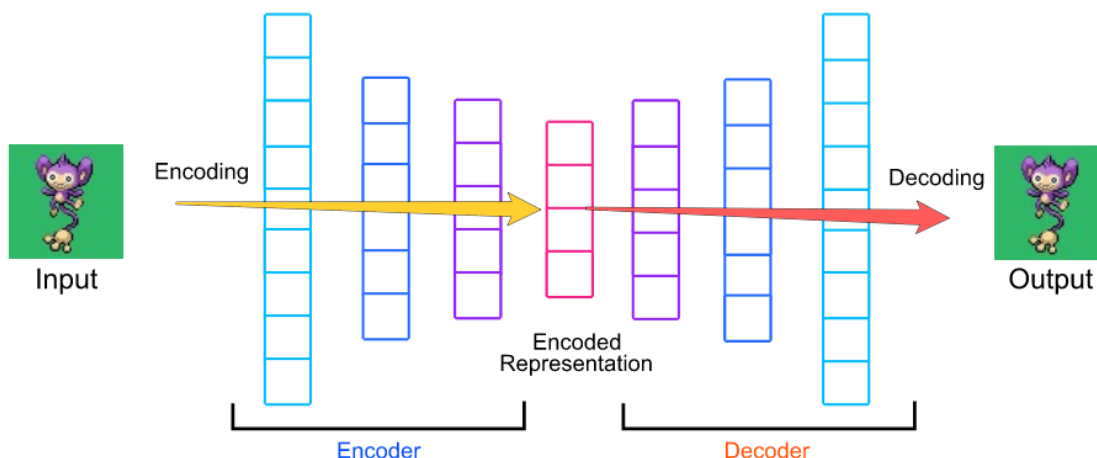


Figure 3.2 Simple auto-encoder CNN architecture.

This initial model was tested on a simple simulated protein data set which only included binary labels on medium resolution α -helices. Unfortunately, the model only achieved 60% voxel-wise accuracy. Considering that a random chance would achieve 50% accuracy with a binary data set, this initial model was not successful. In hindsight, this was likely an issue of data locality. Local data is data near the pixel currently being classified while global data is data from regions throughout the input image. Both are needed in semantic segmentation problems. However, protein structures are usually very dependent on local and not global data because the protein structures are usually only a few voxels wide. Therefore, this architecture was not appropriate for this problem space because it would compress the density maps in the encoder section and lose most of the local data (and retain the global data instead).

3.3 C α ATOM CONVOLUTIONAL NEURAL NETWORK

After the auto-encoder classifier failed to produce a successful result, we used the learnings to create a new architecture. This new architecture was designed to maintain data locality rather than focus on global data. For this network, we aimed to predict regions in the electron density maps with C α Atoms. This architecture was a more traditional architecture that aimed to use a combination of regular convolutions and dilated convolutions to increase the receptive field. This architecture is shown in Figure 3.3. More details about this architecture, including design decisions, are listed in the Convolutional Neural Network Architecture section.

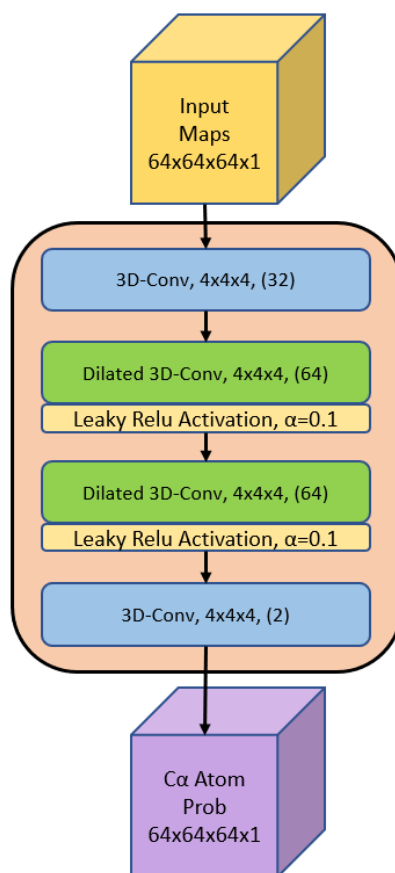


Figure 3.3 C α Atom CNN architecture. The input was a 64x64x64 3D matrix of electron density. The output was a C α cloud representing areas of high confidence in C α atoms. More details about this architecture are in the Convolutional Neural Network Architecture section.

This initial model only predicted what are called $C\alpha$ Clouds, which consist of voxels that have a high-confidence of being near a $C\alpha$ Atom. An example of one of the initial $C\alpha$ clouds is shown in Figure 3.4. As evident by the predicted purple $C\alpha$ clouds around the orange ground truth $C\alpha$ Atoms, this method was clearly predicting the backbone structure with some success. However, there were still issues with the $C\alpha$ cloud predictions. There were many false positive $C\alpha$ clouds along amino acid side chains. Additionally, some of the SSEs were also a challenge. The loops/turns had very weak prediction clouds while many of the α -helices had misplaced $C\alpha$ clouds.

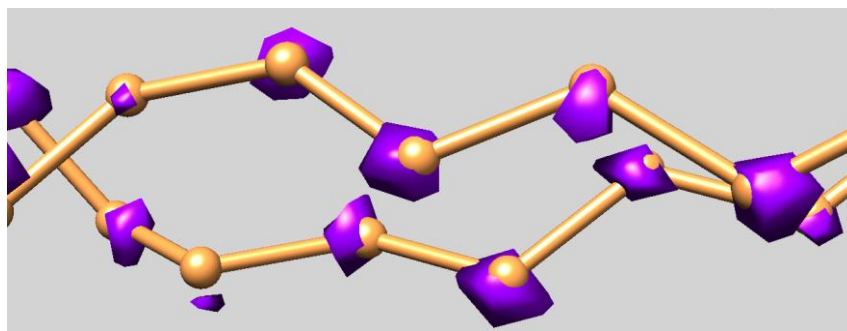


Figure 3.4 $C\alpha$ cloud prediction output from the $C\alpha$ Atom CNN shown in purple. The orange structure is the ground truth backbone trace.

3.4 CASCADED CONVOLUTIONAL NEURAL NETWORK

To address the issues from the $C\alpha$ Atom CNN, the next iteration of the prediction model added two new CNNs before the $C\alpha$ Atom CNN. The idea was to aid the $C\alpha$ Atom CNN by feeding it additional input pertaining to the protein's structure. This new model was called a Cascaded-CNN (C-CNN) because the input from each CNN was forwarded to the next CNN along with the original input density map. It is shown in Figure 3.5.

In the C-CNN each CNN was designed to predict a different protein structure. The first network was the SSE CNN. This CNN predicted voxels as α -helices, β -sheets, or loops/turns and output a confidence map for each SSE. The three SSE maps along with the input electron density

map were forwarded to the backbone CNN which produced two confidence maps representing whether each voxel was part of the backbone structure of the protein or not. The final CNN in the C-CNN was the C α -Atom CNN. This network took all the previous maps and produced two output maps representing the confidence of a voxel being part of a C α atom or not.

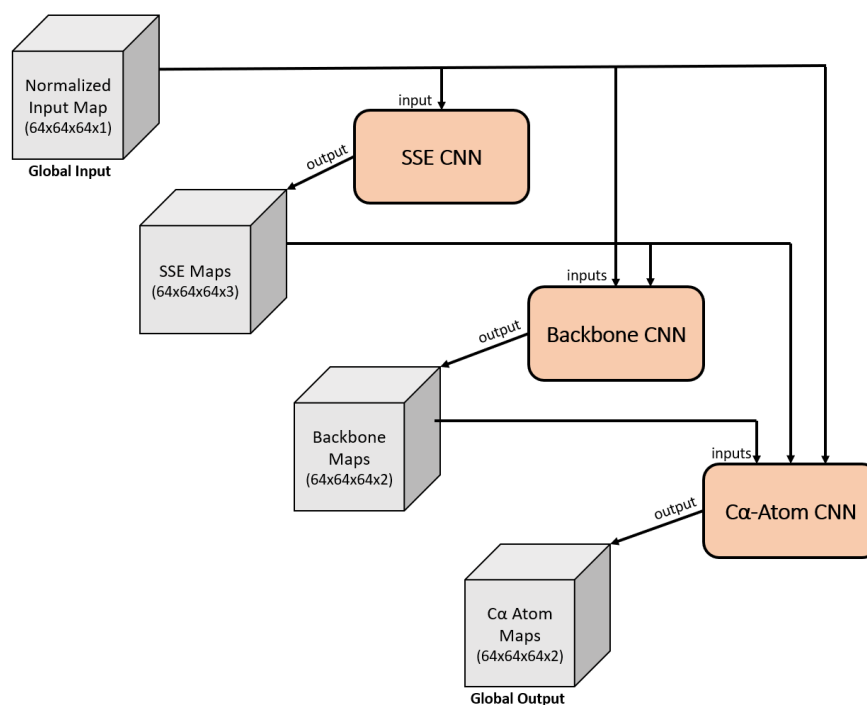


Figure 3.5 Cascaded Convolutional Neural Network. The input/output of each stage is shown as a gray cube with the given dimensions. Each CNN is represented by a tapered salmon rectangle. Results from each CNN are forwarded along with previous input data to the next CNN.

3.5 CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE

The three neural networks of the C-CNN were very similar, each having the same number of layers and same type of layers. Their detailed structure is illustrated in Figure 3.6. Each neural network had four layers. The first and fourth layers were regular 3-D convolutional layers with a stride of one. The second and third layers were dilated convolutions, also with a stride of one. Each dilated convolutional layer used a dilation rate of two. Following each dilated convolution was a

leaky ReLU activation function. A leaky ReLU was preferred over a standard ReLU activation function to improve back-propagation and reduce the problem of vanishing gradient descent. Dilated convolutional layers were used to increase the receptive field while maintaining the image size. This is crucial for semantic segmentation because it maintains a one-to-one ratio of input voxel to output voxel.

Each of the three neural networks used the same number of filters per layer for the first three layers: 1st layer: 32 filters, 2nd layer: 64 filters, and 3rd layer: 64 filters. The 4th and final layer differed for each network, but it was always equal to the number of output classes. Using more filters usually leads to higher accuracy. However, even a small increase from these numbers greatly slowed the network training. Therefore, we settled with these values as it was an optimal compromise between accuracy and speed. Other than the slight difference in filters per layer, the only other difference among the networks was a small difference in kernel size in the standard convolutional layers. The kernel size was larger (5x5x5 vs. 4x4x4) in the backbone and SSE CNNs to account for the need for a larger receptive field to better predict those structural elements.

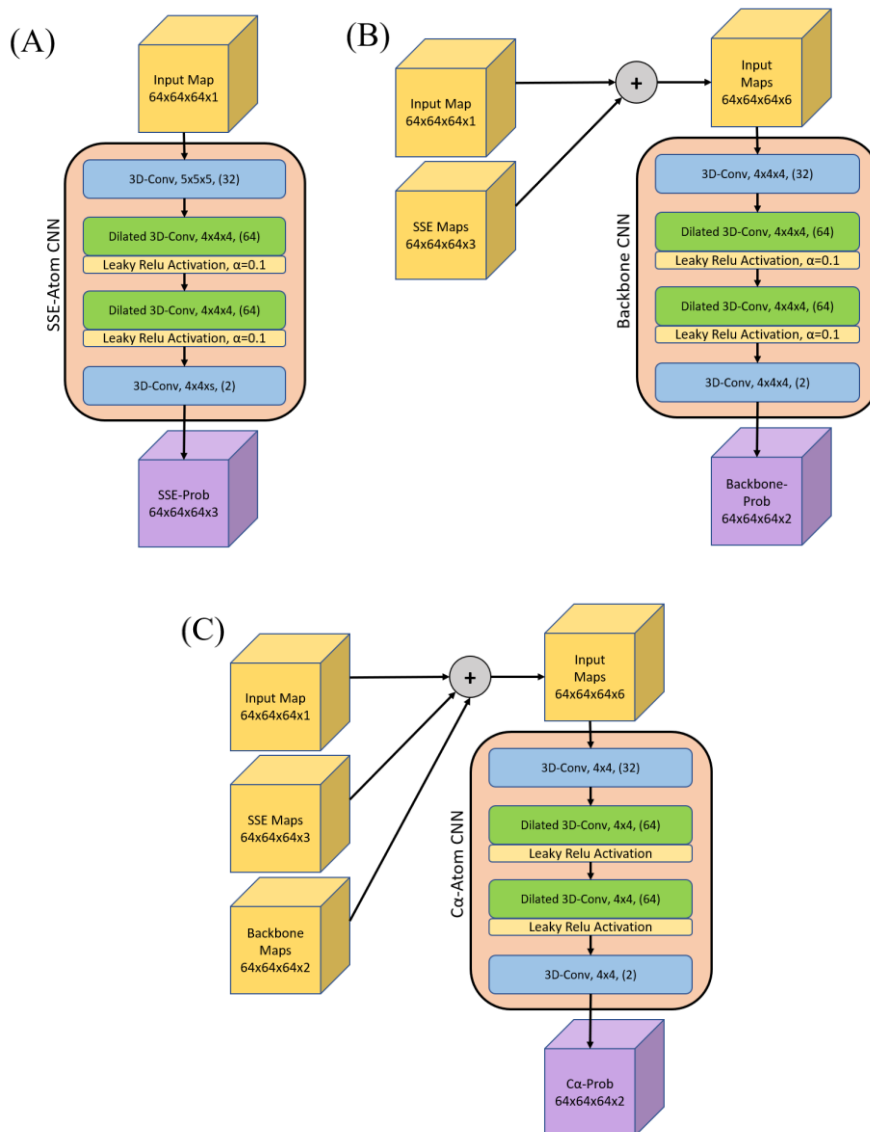


Figure 3.6 Detailed architecture of each of the 3D convolutional neural networks (CNN). (A) contains the Secondary Structure Elements (SSE) CNN. (B) contains the Backbone CNN. (C) contains the $C\alpha$ -Atom CNN. Each CNN, including all its layers, are shown within the salmon colored boxes. The input to each CNN is noted by the yellow cubes. The Backbone CNN (B) and the $C\alpha$ -Atom CNN (C) take input that is a contamination of various maps. The size of each input is noted by the dimensions listed at the base of each input cube. Each layer in each CNN is denoted by its name/function, kernel size ($N \times N \times N$), and finally the output number of filters (inside parenthesis) for that layer. Each leaky ReLU activation function used an alpha value of 0.1. The output of each CNN is noted by the purple cube. The dimensions for each output are listed at the base of the cube.

3.6 END-TO-END MODEL PIPELINE

The cascaded convolutional neural network is only a piece of the full backbone prediction model. The full model is shown in Figure 3.7. The primary input to the full model was an MRC file or MAP file which contained a 3D tensor of the electron density of the protein. The only other input was a manually selected threshold value which was used to zero out low intensity voxel values⁵. Each density map requires a different threshold value. The recommended threshold value varies by density map, however, the recommended contour level on the EM Databank website⁶ is a good value to start with. The final output of the entire prediction model was a PDB file containing a set of traces where each trace is a set of connected C α atoms. This PDB file also contains SSE labels for each C α atom. These labels are determined from both the SSE output maps and the helix-refinement. Using the final output map the RMSD and the percentage of matched C α atoms metrics were calculated.

⁵ The manually selected threshold value was determined by viewing the density map in Chimera and selecting a cutoff value that made the SSEs appear similar in size and structure to the SSEs of the simulated density maps that were used to train the neural networks.

⁶ <https://www.emdataresource.org/index.html>

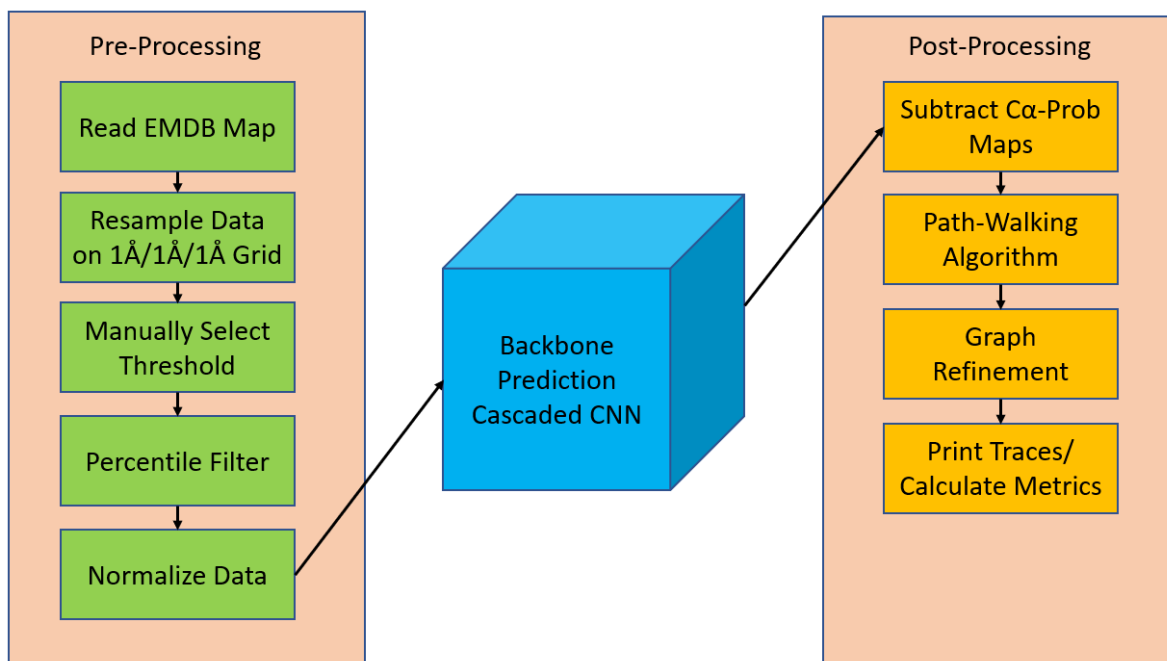


Figure 3.7 Full Backbone Prediction Model. Model includes data preprocessing, the cascaded convolutional neural network (after training), and post-processing.

3.7 PRE-PROCESSING

The goal of pre-processing experimental density maps before sending them into the cascaded convolutional neural network is to make them as similar as possible to the simulated maps that the C-CNN was trained with. Unlike simulated maps, experimental maps have large variance in local classified resolutions, electron density values, and molecular shapes. This can be attributed to the wide range in flexibility of biological molecules, cryo-EM imaging devices, different experimental procedures, and small natural artifacts that appear as part of the cryo-EM imaging process. Combined, these issues make experimental maps difficult to normalize.

The first step of preprocessing was to remove any noise or irrelevant electron density data. This step was accomplished by zeroing out all voxels that were greater than 6\AA away from the ground truth structure of the protein map. Once cleaned, the density map was resampled so that each dimension (x, y, z) had a voxel-size of exactly 1\AA . There is a wide range of voxel-sizes for

each experimental map and many often have a different value for each axis. Therefore, resampling was crucial because the C-CNN was trained with simulated maps that had a voxel-size of exactly $1\text{\AA}\times 1\text{\AA}\times 1\text{\AA}$. This step was easily accomplished by using the UCSF Chimera tool along with the internal Chimera command *vop* resample.

After resampling, the new map was preprocessed using the same method as outlined in Figure 3.1 with the only difference being that the threshold was manually selected for each density map.

3.8 NETWORK INPUT

Since CNNs are fixed by their architecture, they can only accept input with exact dimensions. The C-CNN was designed to take a $64\times 64\times 64$ tensor representing the 3D electron density of a protein. However, because density maps vary greatly in size across each dimension an extra step was required before the model could process maps of a different size. Each map was split into $64\times 64\times 64$ cubes that overlapped by 7 voxels⁷ on each face. Each cube was evaluated by the C-CNN independently and then the resulting output cubes were stitched back together to reconstruct the full image. However, only the center $50\times 50\times 50$ voxels were used to reconstruct the image. At each face the 7-voxel overlap region was disregarded. This method allowed us to process density maps of any size without losing spatial information at each cube's boundary.

3.9 NETWORK OUTPUT

After the C-CNN processed the input map it produced confidence maps for the protein's SSEs, backbone, and $C\alpha$ atom locations. The output maps for EMDB-8410 (chain-A) are shown in Figure 3.8. Each voxel in the map was assigned a confidence value by the network. The final classification

⁷ The 7 voxels value was derived as the overlap value because it was half of the difference between the full input map size ($64\times 64\times 64$) and the retained map size ($50\times 50\times 50$). This allowed the map splitter class to move 50 voxel chunks each time a new section was fed into the network.

of a voxel was determined by the max voxel value of each of the output maps for a given neural network.

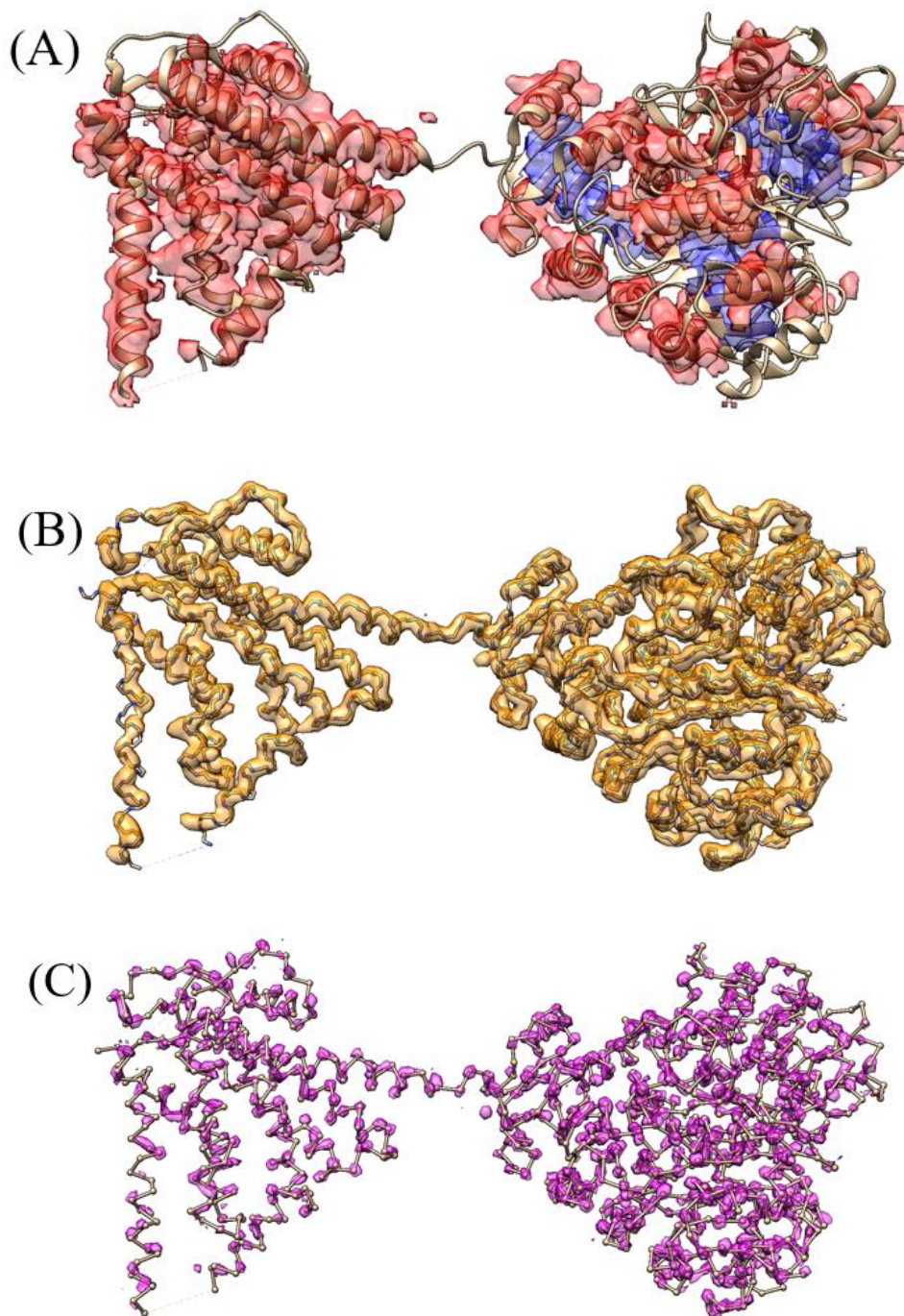


Figure 3.8 Confidence Map Output for EMD-8410 (Chain-A). (A) is the combination of the α -helix and β -sheet prediction map after applying the max function, (loops/turns map omitted for

readability). (B) is the backbone confidence map (>40% confidence) with the ground truth backbone structure shown for reference. (C) is the C α -Atom confidence map (>50% confidence) with the ground truth ball and stick representation of EMDB-8410 shown for reference.

3.10 PATH-WALKING ALGORITHM

Although the C-CNN assigns confidence values to specific features of the protein, post-processing algorithms were required to piece together that information and generate a final prediction trace. This was accomplished with a path-walking technique that processed the confidence maps to produce a final PDB file that contained exact C α atom locations along the protein's backbone.

The path-walking technique walked through high-confidence areas of the backbone map and connected areas of high C α atom confidence using a novel tabu-search algorithm designed specifically for this research. The tabu-search algorithm scored each potential future movement based on a location's local density prediction confidence and distance. Additionally, it also incorporated the backbone atom torsion angles and common radius of α -helix secondary structures as weights when finding the optimal next C α atom.

The path-walking algorithm walked until it either reached an area of the protein that had already been processed or until it reached an area of the protein where no more suitable C α atoms could be found. Upon reaching the end of a single trace, the path-walking algorithm would search the C α confidence map for any other areas of the protein that might contain additional untraced C α atoms and, if found, would walk each additional trace. This process was repeated until all high-confidence areas of the C α prediction map had been explored. The output of the path-walking algorithm was a PDB file consisting of a series of disconnected traces where each trace contained a chain of C α atoms.

3.11 GRAPH REFINEMENT

The disconnected traces from the path-walking algorithm represented partial backbone traces in the protein. However, the overall accuracy of these traces was low because many side chains and shortcuts between backbone structures were incorrectly classified as backbone traces as a result of the path-walking algorithm. To remedy this issue, two refinement steps were required to improve the predicted traces: path combination and backbone refinement. In order to complete these two steps, the backbone traces were converted from a list of $C\alpha$ locations and connections into a graph where each $C\alpha$ atom was represented by a vertex and each connection to another $C\alpha$ atom was represented by an edge.

The goal of the path combination step was to combine a set of disjoint graphs (formerly traces) into a fully connected graph that is more representative of the protein's backbone structure. We used a depth first search to walk from any given $C\alpha$ atom within a disjoint graph to both end points of that trace. Using these endpoints, this algorithm would then examine all other $C\alpha$ nodes in the protein graph to determine if another $C\alpha$ atom was within 3Å of the end point $C\alpha$ atom. If it was, then the end point $C\alpha$ atom's location was reassigned to be equal to the other $C\alpha$ atom. This process helped combine neighboring disjoint graphs (traces) into a fully connected graph.

After path combination, the fully connected graph resembled a protein's backbone structure, but it still had many side chain and backbone trace shortcut connections. These false-positive connections meant that many $C\alpha$ nodes in the graph had three or even four edge connections to other $C\alpha$ atoms. The next step in the graph refinement process was to remove the false-positive connections so that the remaining graph only contained true-positive $C\alpha$ node and $C\alpha$ edge connections. This refinement process was broken down into three steps: side chain removal, loop removal, dead-end point removal.

Side chain removal involved examining every $C\alpha$ node in the graph to determine if it might be part of a false-positive side chain connection. If a node had three or more edges (trinary node), it was likely that one of the three edges was a side-chain connection. This algorithm would use a depth first search (DFS) to walk along each of the three paths leading from a trinary node and stop once it found either an ending node (only one edge), or another node with three or more edges. It would compare the total depth reached for each of the three DFS edge walks. If one path had a depth of three or less while the other two paths both had depths greater than the shortest path, then the shortest path was considered a side chain connection and removed from the graph. This algorithm proved to be very effective at removing side chain and false-positive shortcut connections between true parallel backbone traces.

After removing the side chains from the fully connected graph, it was necessary to remove small loops within the graph. These loops were the result of false-positive shortcut traces within α -helix elements of the protein. The goal of this method was to remove the false-positive half of each loop leaving the true backbone structure in place. The approach was similar to side chain removal. It would find any $C\alpha$ nodes that contained three or more edges and then path walk each trace until it reached an end-node or trinary connection. However, in this case, if two paths terminated in the same trinary node then the combined two paths were considered a loop. To remove the false-positive side of the loop, this method would calculate the density along each path using a 1\AA radius cylinder and remove the path with the lower average density value. This approach made the assumption that the backbone structure of the protein had a higher density than another false-positive path.

The final step of the graph refinement process was to remove dead-end nodes. These resulted from side-chains that did not connect to another backbone trace of the protein but did nonetheless

protrude off the true backbone structure. Removing these was accomplished by finding all trinary $C\alpha$ nodes in the graph and then walking down each trace extending from that node. If any path had a depth of two or less and ended in a dead-end node then it was considered a side-chain and removed from the graph.

3.12 COMPUTATION

The C-CNN was trained with 25,000 simulated protein maps, each with a size of 64x64x64 voxels. Training was accomplished with the Python TensorFlow Library on a Nvidia GTX 1070 GPU. Training was stopped after 15 epochs to prevent overfitting and took about 24 hours. Density Map prediction, which involved running a preprocessed density map through the saved C-CNN only, was completed using the same GPU and took about 15 seconds to produce the five output prediction maps (three SSEs, backbone, and $C\alpha$ atoms) for a map size of approximately 100x100x100 voxels. The path-walking algorithm was the most time-consuming aspect of the prediction process. A map of approximately 1000 $C\alpha$ atoms took about 20 minutes to compute. All computation used a machine with an Intel 6 core i7-8700K CPU clocked at 3.7GHz with 16GB of RAM. To increase throughput, a new process was spawned for each path-walking task (one per protein, with a maximum of 12 at a time).

Chapter 4. RESULTS

This method was tested with both simulated and experimental maps. Simulated density maps were generated using the pdb2mrc script from the EMAN2 package while experimental maps were downloaded from the EM Databank. Both experimental and simulated maps underwent the same pre-processing steps as outlined in the Data Collection/Generation section before being evaluated.

4.1 METRICS

A variety of metrics were used to measure the effectiveness of our method. One primary metric was the root-mean-squared-deviation (RMSD) which measures the standard deviation of the distance between atoms in two models. In this case, the two models were the predicted C α atoms and the ground truth C α atoms as defined in the PDB file. The output from our model often consisted of partial backbone traces when the confidence was not high enough to form a complete backbone trace. With partial backbone traces it is difficult to use traditional RMSD algorithms to measure the effectiveness of a prediction method. As a result, we decided to follow the same method used by the fully autonomous Phenix method [18] which compares each C α atom in the ground truth model to the closest C α atom in the predicted method using a one-to-one mapping. This RMSD method walks each predicted backbone trace and pairs it with the closest C α atom in the ground truth structure. This produces lower/better RMSD values than other methods [11] [24] because it allows for C α skips in the ground truth backbone trace.

Another primary metric that we focus on in this research is the percentage of predicted C α atoms within 3Å (% C α in 3Å) of the ground truth structure. This metric is a good measure of prediction map completeness because higher values mean that a higher percentage of the ground truth atoms were found. This metric is calculated in a similar way to the RMSD metric: by walking down each predicted trace and pairing each predicted C α atom with its closest ground truth C α atom. This metric requires a one-to-one mapping. We compare our results for this metric to the Phenix method which used the exact same metric and metric calculation method.

In addition to these metrics we also include for each tested density map: the number of predicted C α atoms, the number of actual C α atoms, and the number of false positive C α atoms.

4.2 SIMULATED DENSITY MAPS

Table 4.1 shows the results for seven simulated density maps. Each density map was generated from the pdb2mrc script at 6Å. Although 6Å is considered medium resolution, pdb2mrc used inflated resolution values. Therefore, a 6Å resolution simulated map was most similar to an experimental map at 3.5Å resolution.

Table 4.1 Results of simulated data.

PDB ID	Model Cα	Native Cα	RMSD (Å)	% Cα in 3Å	FP%
3i2n	351	345	0.91	99.1	0.6
3n2t	328	327	0.90	99.4	0.3
3qc7	165	164	0.97	94.8	0.0
5i68	661	662	0.92	99.4	0.2
6ahv	348	345	0.89	99.7	0.3
6eyw	384	381	0.87	100.0	0.3
6g61	112	111	0.84	98.2	0.9
Avg.			0.90	98.7	

The simulated results proved to be very accurate relative to the true protein structure. With an average RMSD of 0.90Å per C α atom, the predicted backbone structure was almost a perfect match to the true structure. Additionally, the predicted results produced nearly complete backbone traces as evident by an average of 98.7% of ground truth C α atoms being within 3Å of a predicted C α atom. Finally, the false positive rate was very low for each prediction (no more than 2 false positive atoms per protein). This translates to a very high sensitivity for each prediction. Figure 4.1 shows the final prediction maps of both the C α -only backbone structure along with the ribbon representation of protein 6eyw.

The low RMSD values, minimal false positive C α atoms, and high matching percentage of C α atoms within 3Å of the real structure demonstrates the effectiveness of this method with simulated density maps. However, these maps have the advantage over experimental maps in that

they do not contain any experimental inaccuracies or real-world noise to distort the image. Additionally, these simulated maps were generated with the same pdb2mrc script that the training data was generated with. This means that the C-CNN likely learned features of simulated maps very accurately and possibly overfit the data thereby leading to the high accuracy metrics.

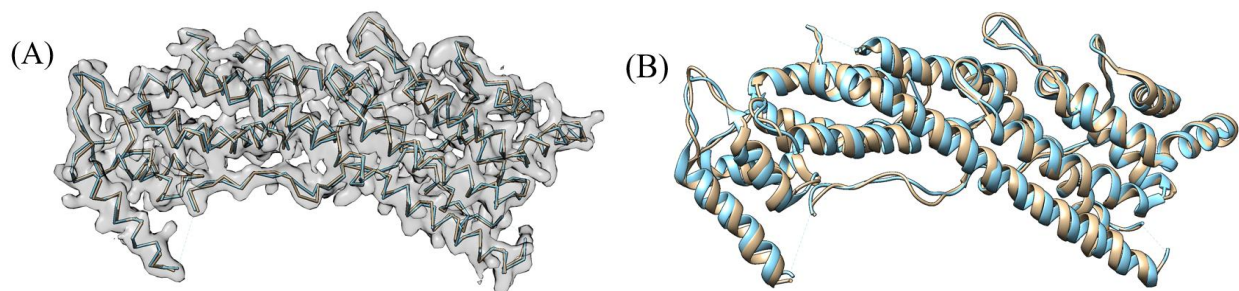


Figure 4.1 Final Backbone Prediction structures from the simulated 6eyw protein map. Ground Truth is colored teal while the predicted structures are colored tan. (A) contains the $C\alpha$ only backbone structures with the input density map overlaid on the image for reference. (B) displays the final ribbon prediction. The ribbon was enhanced using the secondary structure output maps from the Cascaded Convolutional Neural Network.

4.3 EXPERIMENTAL DENSITY MAPS

The real test of our backbone prediction model involved experimental density maps. Experimental maps ranging in resolution from 2.6Å to 4.3Å were evaluated using the backbone prediction model. Table 4.2 tabulates the results for each experimental density map. Each density map was downloaded from the EM databank. In addition to the normal preprocessing steps outlined in the Pre-Processing section, each density map underwent an additional cleaning step before being evaluated. To remove artifacts from the cryo-EM imaging process unrelated to the protein structure, all voxels that were further than approximately 5Å from the ground truth structure were zeroed out using built in Chimera functions. This cleaning step essentially cut the protein out from the raw density map. However, this step maintained the experimental protein's

shape and density and did not change or manipulate any voxel data within the protein’s experimental structure.

Our results were compared to the fully automatic Phenix method [18] in two categories: RMSD and % C α Matching within 3Å. The Phenix method results are listed alongside our results in Table 4.2.

Table 4.2 Results from the evaluation of experimental density maps with the Phenix method’s results shown as a benchmark.

EMDB ID	PDB ID	Res. (Å)	Model C α ⁸	Native C α ⁹	FP% ¹⁰	% C α in 3Å	RMSD (Å)	Phenix % C α in 3Å	Phenix RMSD (Å)
2513	4ci0	3.36	9634	10716	2.5	83.3	1.32	62.5	1.33
3061	5a63	3.4	1293	1223	4.0	95.6	1.09	73.9	1.15
3121	5aco	4.36	2639	2415	10.7	83.8	1.42	57.0	1.63
3222	5flu	3.8	1895	2119	0.9	82.6	1.59	62.0	1.76
3238	5fn3	4.1	1313	1318	2.3	83.6	1.51	56.1	1.50
3601	5n8o	3.9	1431	1433	6.2	86.0	1.38	50.0	1.29
3785	5odv	4.0	3818	4968	10.9	62.4	1.46	40.1	1.73
4054	5lij	4.2	169	152	0.6	97.4	1.43	84.2	1.44
4062	5ljv	3.64	2006	1950	3.4	93.7	1.09	48.3	1.21
4128	5lzp	3.5	6059	6182	0.5	92.7	1.16	77.1	1.13
5623	3j9i	3.3	6002	5978	0.6	97.0	1.03	72.4	0.87
5778 ¹¹	3j5p	3.28	512	592	1.0	78.4	1.32	55.6	1.15
5995	3j7h	3.2	4028	4088	1.7	93.2	1.09	72.9	1.12
6224	3j9c	2.9	2959	2961	1.0	94.1	1.02	71.9	0.99
6239	3j9d	3.3	762	746	2.1	98.3	0.99	74.8	0.99
6240	3j9e	3.3	445	520	0.7	83.5	0.97	69.4	0.82
6272	3j9s	2.6	1227	1191	0.3	99.2	0.88	88.9	0.69
6324	3ja7	3.6	4297	5136	2.0	77.8	1.2	69.4	1.13
6346	3jaf	3.8	1706	1710	1.7	94.1	1.21	72.8	1.10
6408	3jb6	3.3	1735	1731	0.8	96.9	0.96	79.3	0.83
6488	3jby	3.7	1923	1802	13.7	88.8	1.14	57.3	1.18
6551	3jcf	3.8	1853	1745	1.8	96.0	1.24	74.2	1.15
6630	3jcz	3.26	2932	2976	1.1	91.7	1.07	66.1	1.05
6631	3dj0	3.47	2904	2976	2.0	90.3	1.16	59.5	1.07

⁸ The number of C α atoms modeled by our C-CNN method.

⁹ The number of C α atoms in the ground-truth PDB structure.

¹⁰ Percentage of C α atoms in the predicted model that are more than 3Å away from any ground-truth C α atom.

¹¹ EMDB-5778 Chain-A only

EMDB ID	PDB ID	Res. (Å)	Model C α ⁸	Native C α ⁹	FP% ¹⁰	% C α in 3Å	RMSD (Å)	Phenix % C α in 3Å	Phenix RMSD (Å)
6676	5wq8	3.26	5680	7440	1.0	73.4	1.05	54.2	0.93
6703	5x58	3.2	3091	3159	5.7	85.5	1.28	53.1	1.27
6744	5xno	3.5	1074	854	24.6	87.7	1.18	55.4	1.22
6770	5xsy	4.0	1407	1312	7.9	90.3	1.35	68.5	1.21
7063	6b7n	3.3	3062	2898	4.2	95.4	1.12	71.3	1.21
8015	5gaq	3.1	2607	2592	0.5	98.8	0.92	79.2	1.1
8069	5i08	4.04	2843	2874	2.9	88.5	1.32	66.7	1.51
8072	5i68	3.37	5379	5296	2.8	93.8	1.07	84.6	1.07
8200	5k47	4.22	2000	1936	1.9	91.9	1.26	71.7	1.29
8331	5szs	3.4	3716	3531	4.9	94.6	1.08	68.6	1.04
8405	5tfy	3.4	4041	4264	0.9	92.7	1.09	70.7	1.25
8410	5tj6	3.5	3706	3560	1.9	96.2	1.13	67.6	1.07
8515	5u70	3.76	3723	3608	0.7	96.6	1.13	76.5	1.19
8637	5v6p	4.1	508	540	2.9	83.3	1.6	80	1.37
8642	5v7v	3.9	807	613	8.3	89.6	1.38	76.3	1.28
8651	5va2	3.8	2172	2232	2.8	82.7	1.54	69.7	1.43
8658	5vc7	3.9	2956	3264	3.8	80.9	1.5	68.2	1.49
8697	5vjh	4.0	2943	3485	3.8	76.5	1.41	72.5	1.39
8712	5vms	3.7	1869	1936	1.9	90.8	1.17	64.9	1.22
8764	5w3s	2.94	1694	1940	4.1	79.2	1.07	66.6	0.97
8767	5w5f	3.4	2953	2934	1.1	97.2	1.36	52.8	1.43
8782	5w81	3.37	1172	1173	4.7	89.1	1.44	66.7	1.38
8784	5w9i	3.6	3763	3618	5.1	89.9	1.3	53.8	1.25
8794	5wc0	4.4	1342	1660	3.1	71.9	1.61	64.3	1.53
8882	5wpt	3.75	1718	1816	1.9	87.4	1.13	68.5	1.02
9515	5gjw	3.9	2684	2678	6.5	82.7	1.48	54.6	1.40
Avg.						88.5	1.23	66.8	1.22

The results on experimental data show that our method was very similar to the automatic Phenix method with respect to RMSD. The C-CNN was able to achieve a mean RMSD of 1.23Å while the Phenix method achieved a mean RMSD of 1.22Å. Both methods were tested on the same set of 50 experimental density maps. Our method was able to produce a much higher average C α percentage matching within 3Å than the Phenix method (88.5% vs. 66.8%). This significant improvement is primarily a result of our method’s ability to predict backbone structure in relatively low-confident regions while the automatic Phenix method made no prediction in these areas. By achieving similar RMSD metrics but improved C α matching percentages our method has clearly

demonstrated an improvement over the automatic Phenix method in terms of $C\alpha$ /backbone prediction. Figure 4.2 shows the final predicted map for three experimental density maps using our deep learning method.

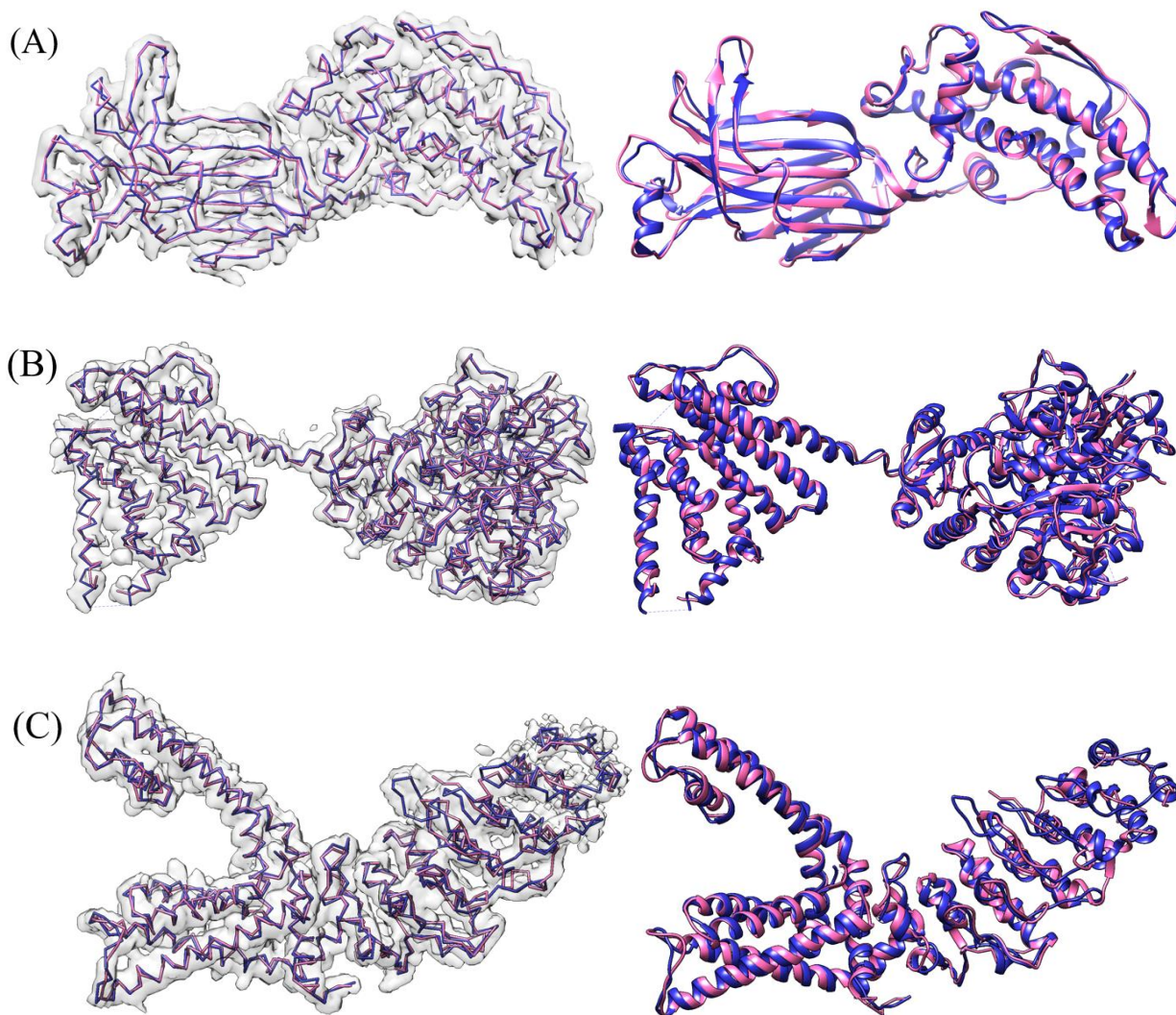


Figure 4.2 Final Backbone Prediction maps for various density maps. (A) EMDB-6272 (chain-A) at resolution 2.6Å. (B) EMDB-8410 (chain-A) at resolution 3.5Å (C) EMDB-5778 (chain-A), resolution 3.3Å. The left map in each subfigure contains the predicted vs. actual backbone structure. The right map in each subfigure contains the predicted vs. actual ribbon structure of the protein which specify the SSE classification. The pink trace is the predicted structures while the blue trace is the actual structure.

4.4 COMPARISON OF PREDICTION METHODS

As introduced in the Current Protein Prediction Models section, there are three other leading backbone prediction models: Phenix, MAINMAST, and RosettaES. We compare our deep learning method to the other leading prediction models by evaluating EMDB-5778 (chain-A) with each method. The prediction maps for each method are shown in Figure 4.3.

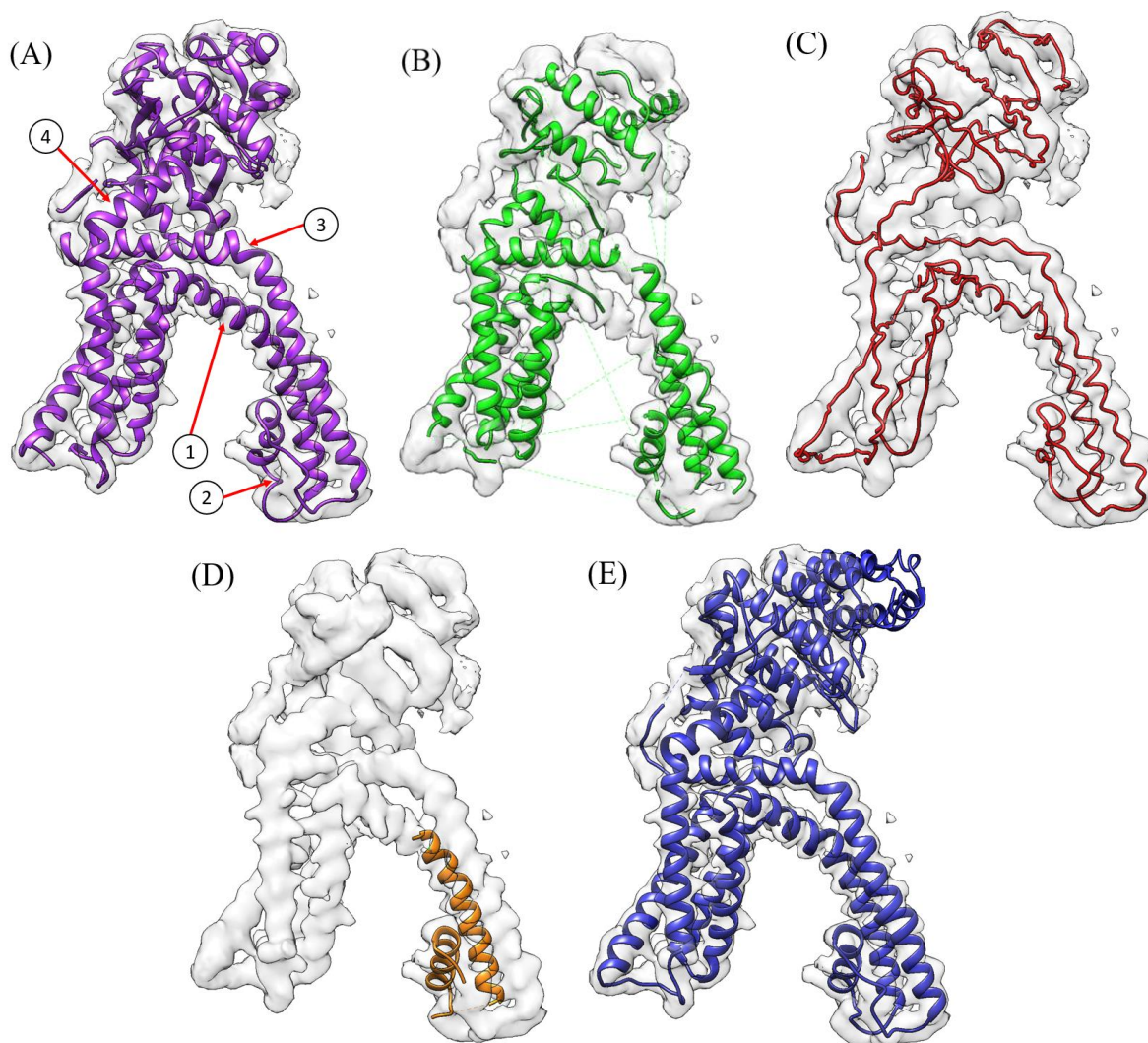


Figure 4.3 Final Prediction Model for EMDB-5778 (chain-A) at 3.28Å resolution. (A) Our C-CNN method. (B) automatic Phenix method. (C) MAINMAST. (D) RosettaES. (E) The ground truth model. Each figure overlays the input density map on top of the predicted structure.

(A) Notes five areas where the C-CNN outperformed the Phoenix method.

Table 4.3 compares the prediction metrics of all four prediction methods. The RosettaES model produced the most accurate prediction map but it was only able to predict 7.9% of the C α Atoms from the ground-truth model. The MAINMAST method, which only predicts backbone structure and not SSEs, found the highest percentage of C α Atoms among the three previous prediction methods. However, MAINMAST has a relatively high RMSD of 2.20Å and also produced the highest false positive C α Atom percentage. Among the three other prediction methods the Phenix method was arguably the best method for EMDB-5778, producing decent metrics in all three categories (Percentage of Matching C α Atom, RMSD, and false positive C α Atom percentage). However, our deep-learning method was able to achieve a much higher Percent C α Atom Matching and a lower false positive C α Atom percentage with only a small increase in the RMSD. Figure 4.3A points out four areas where our prediction model outperformed the Phenix method by producing a more-complete prediction map.

Table 4.3 Comparison of EMDB-5778 (chain-A) among leading backbone prediction models. Each model had specific strengths, however the deep learning C-CNN produced the most complete model as measured by C α matching percentage.

Method	% C α in 3Å	RMSD (Å)	FP%
C-CNN	78.4	1.32	1.0
Phenix	55.6	1.15	3.0
MAINMAST	59.6	2.20	6.5
RosettaES	7.9	0.91	0.0

4.5 COMPUTATION TIME OF PREDICTION MODELS

While our C-CNN model took about 24 hours to train on a single machine with a GTX 1070 GPU, after the training had completed, the full end-to-end prediction for a density map with approximately 1000 C α Atoms took about 20 minutes to complete. The C-CNN prediction software was running on the same GTX 1070 GPU with a single CPU core. In contrast, the existing

methods usually required a lot of computing resources and were also very time-consuming. In our experiments, RosettaES took 5 days with 20 CPUs to complete, MAINMAST finished in about 18 hours with 1 CPU, and Phenix method took several hours with 6 CPUs.

4.6 COMPARISON OF C-CNN AND THE FULLY AUTOMATIC PHENIX METHOD.

The results for each prediction map in Table 4.2 are plotted and shown in Figure 4.4 and Figure 4.5. Figure 4.4 plots the RMSD vs. the labeled resolution for each density map. The mean RMSD of our deep-learning method was 1.23Å which is just slightly worse than the Phenix method's RMSD of 1.22Å. Figure 4.4 shows that the performance of our deep-learning method was very similar to the Phenix method across all resolution values. Additionally, as one might expect, the RMSD for both prediction methods increased as a function of the labeled resolution.

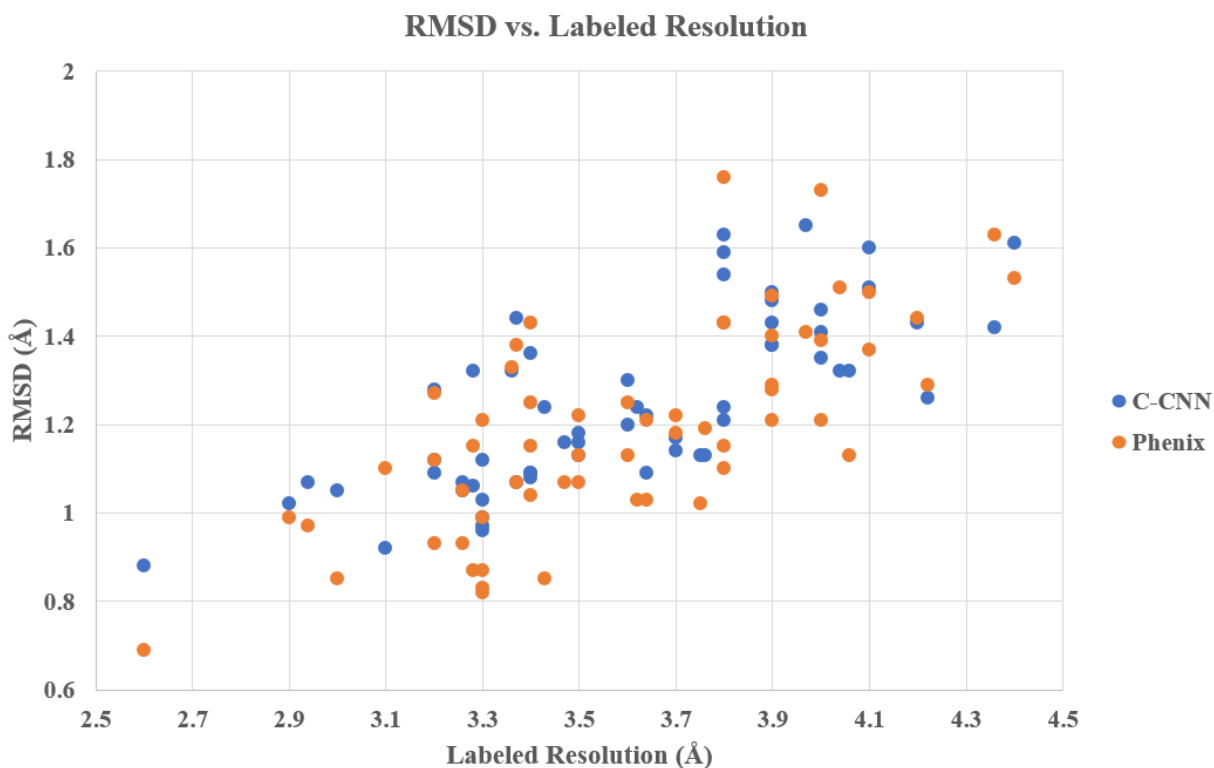


Figure 4.4 Plot of RMSD as a function of resolution for the 50 experimental density maps. This compares our deep learning C-CNN method and the fully automatic Phenix Method.

Figure 4.5 plots the percentage of matching $C\alpha$ atoms vs. labeled resolution for each density map. This figure shows how our deep-learning method found a higher percentage of $C\alpha$ atoms than the Phenix method across all resolutions. Our method found a mean of 88.5% of the $C\alpha$ atoms for the 50 experimental density maps. This is significantly better than the Phenix method which only found a mean of 66.8% $C\alpha$ atoms from the same 50 experimental maps.

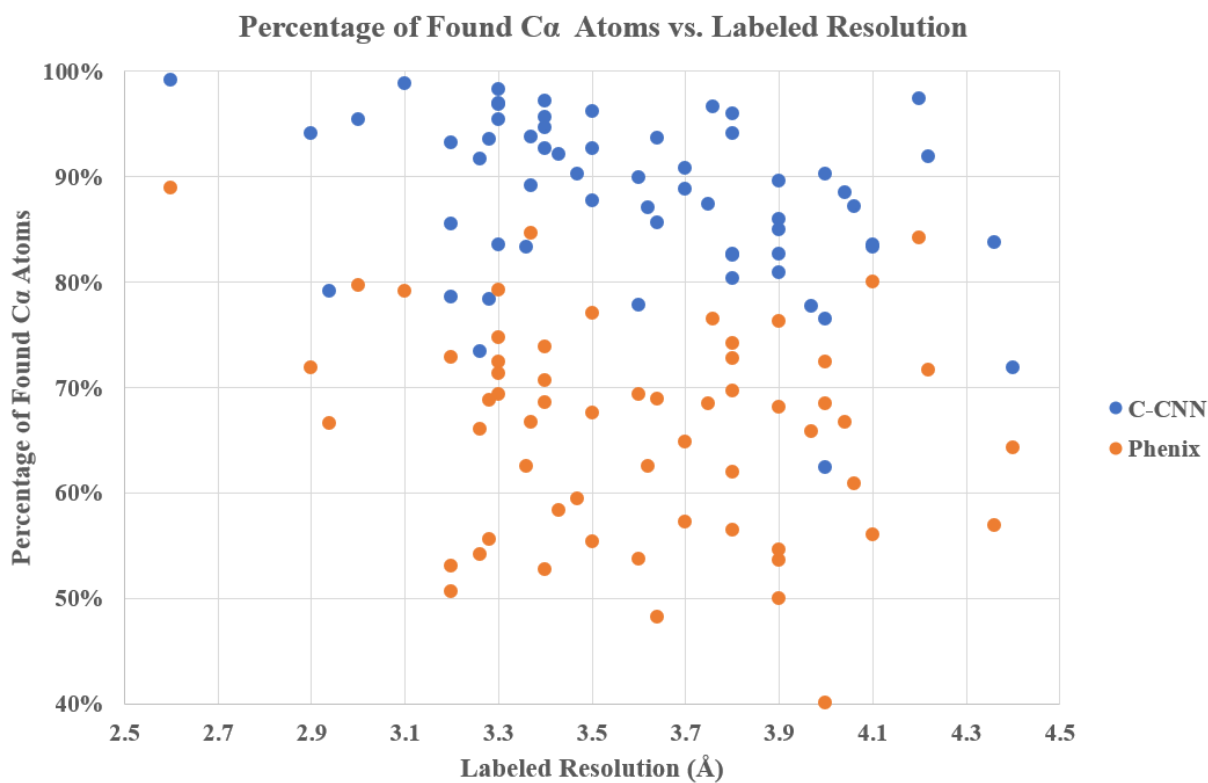


Figure 4.5 Plot of Percentage of Found $C\alpha$ atoms as a function of resolution for the 50 experimental density maps. This compares our deep learning C-CNN method and the fully automatic Phenix Method.

Chapter 5. DISCUSSION

5.1 FALSE POSITIVE C α ERROR RATE

The RMSD metric in conjunction with the percentage of matching C α metric is useful to determine the accuracy of C α atom placements that are in-line with C α atoms within the ground truth model. However, as is often the case with backbone prediction, side-chains or other noise in a density map will appear as backbone structure to a prediction model. This can cause the model to produce false positive C α atoms that are clearly not in alignment with the true backbone structure of the protein, see Figure 5.1. An effective prediction model will limit the number of false positive C α atoms by correctly distinguishing side-chains or other noise from the true backbone structure and not place C α atoms in these areas. In this research, we measure the number of false positive C α atom for each density map, see Table 4.1 and Table 4.2. A false positive is defined as any C α atom that is placed more than 3Å from any C α atom within the ground truth structure.

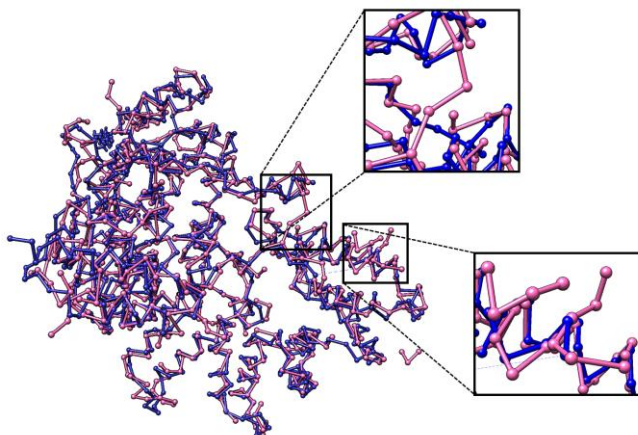


Figure 5.1 Ball and Stick representation of the prediction map (pink) and the ground truth map (blue) of EMDB-8642, resolution 3.9Å. This map had a relatively low RMSD value of 1.38Å but also had a false positive C α rate of 8.3%. The expanded areas show examples of side chains that were incorrectly predicted as backbone structure resulting in false-positive C α placements.

Normalizing the number of false positive C α atoms by the size of a protein produces a false positive rate for C α atom placement. In short this is called the C α error rate. This metric can be derived from the number of false-positive C α atoms and the total number of predicted C α atoms, see Equation 1.

$$\text{Error Rate} = \frac{\# \text{ false positive C}\alpha}{\text{total \# predicted C}\alpha} \quad (1)$$

Figure 5.2 plots the C α atom error rate as a function of resolution for the 50 experimental density maps evaluated by our model. The error rate of C α atom prediction is inversely related to the labeled resolution of a density map. However, our deep learning model produced a relatively low error rate (< ~5%) for all density maps with a labeled resolution value of 3.7Å or better. The error rate of density maps with a lower resolution than 3.7Å had a high variance and thus had lower prediction confidence.

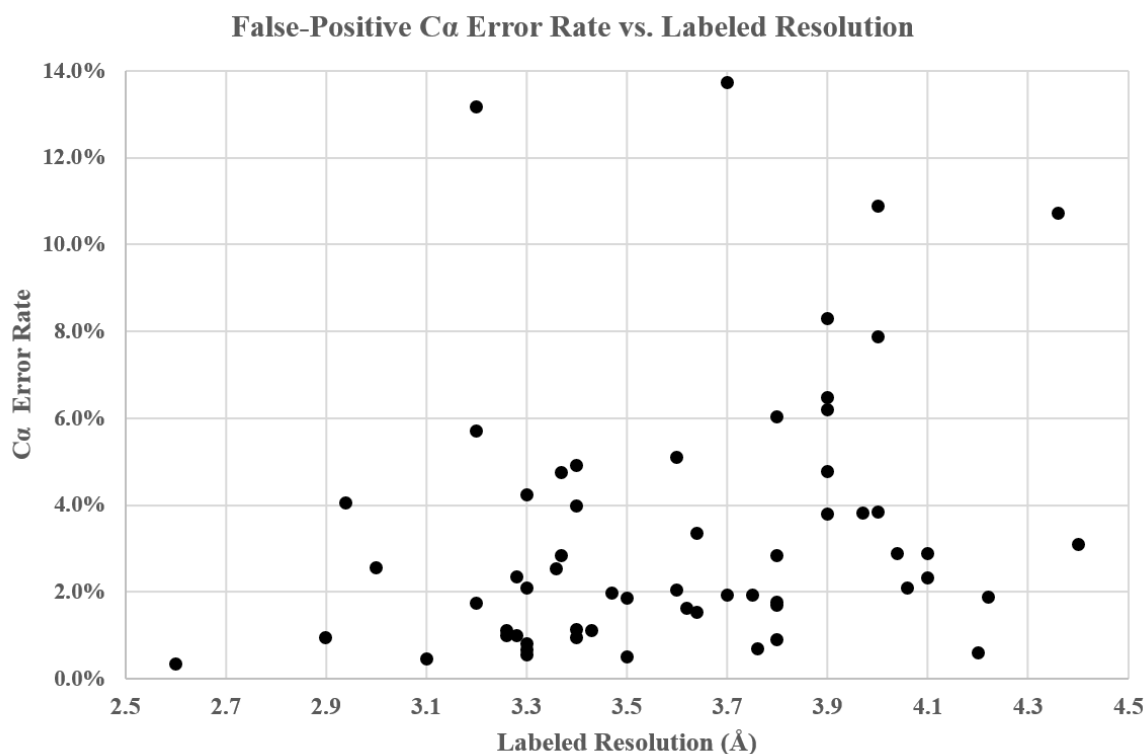


Figure 5.2 Error-Rate vs. resolution for each experimental prediction map with an added exponential trendline.

Chapter 6. CONCLUSIONS

6.1 CONTRIBUTIONS OF RESEARCH

This research has demonstrated the effectiveness of semantic segmentation machine learning to predict protein structures. This is significant because each of the existing backbone prediction models does not use machine learning to produce their prediction results. Rather they rely on advanced computational models to produce their output. This research also demonstrated the effectiveness of the novel cascaded-CNN model that predicts different levels of a protein's structure using different CNNs. Finally, this research was able to outperform all the leading prediction models by producing the most complete prediction maps as measured by the percentage of matching $C\alpha$ atoms.

6.2 FUTURE IMPROVEMENTS

We found during development that the biggest improvements in accuracy came as the result of adding more convolutional neural networks to the C-CNN. Originally, this method used only one network, the $C\alpha$ -Atom prediction network. It was only after adding the SSE and Backbone CNNs that we were able to achieve the results outlined in this paper. Future work might be able to incorporate other CNNs into the C-CNN such as an Amino Acid network or an individual atom network. Adding these networks might also help match the sequenced DNA of the protein to the density map.

As the number of publicly available experimental density maps continues to grow, it may become possible to train neural networks using experimental data instead of simulated data. Our research used simulated data to train our network, but experimental training data may improve results further. This change would prevent the model from over-fitting the simulated data and give

it a wider, more representative data set to train with. Also, by adding an additional 1x1 scalar input to the network to denote the resolution of an experimental map, the C-CNN could train to differentiate between map resolution. This change would allow the network to learn each resolution with more independence.

This research relied on a manually chosen threshold value to normalize each experimental density map before evaluating it with the C-CNN. This manual step is not ideal as it requires subjective input by the user. Further improvements could remove this manual step by adding an automatic method which use traits such as map surface area or resolution to automatically calculate a threshold value for each density map. Another effective method could train a CNN, also using semantic segmentation, to automatically modify density maps into a preprocessed state.

6.3 CONCLUSION

In summary, we presented an effective method for protein backbone prediction from high resolution cryo-EM density maps that used deep learning. This approach used three cascaded convolutional neural networks to produce confidence maps for some of the major structural components of proteins. These confidence maps were used in union with a tabu-search path-walking algorithm to construct backbone traces along with the corresponding C α atom positions. Our method out-performed the Phenix based fully automatic model building method by producing backbone traces that were more complete (88.5% vs. 66.8%) as measured by percentage of matching C α atoms. Further research may improve this method further by incorporating other structural aspects of protein molecules within the cascaded convolutional neural network or training the networks with experimental data.

Chapter 7. WORKS CITED

- [1] M. Liao, E. Cao, D. Julius and Y. Cheng, "Structure of the TRPV1 ion channel determined by electron cryo-microscopy," *Nature*, vol. 504, pp. 107-112, 2013.
- [2] G. Fan, M. L. Baker, Z. Wang, M. R. Baker, P. A. Sinyagovskiy, W. Chiu, S. J. Ludtke and I. I. Serysheva, "Gating machinery of InsP 3 R channels revealed by electron cryomicroscopy," *Nature*, vol. 527, pp. 336-341, 2015.
- [3] Z. Yan, X.-c. Bai, C. Yan, J. Wu, Z. Li, T. Xie, W. Peng, C.-c. Yin, X. Li, S. H. W. Scheres, Y. Shi and N. Yan, "Structure of the rabbit ryanodine receptor RyR1 at near-atomic resolution," *Nature*, vol. 517, pp. 50-55, 2015.
- [4] R. K. Hite, P. Yuan, Z. Li, Y. Hsu, T. Walz and R. MacKinnon, "Cryo-electron microscopy structure of the Slo2.2 Na⁺-activated K⁺ channel," *Nature*, vol. 527, pp. 198-203, 2015.
- [5] R. K. Hite, X. Tao and R. MacKinnon, "Structural basis for gating the high-conductance Ca²⁺-activated K⁺ channel," *Nature*, vol. 541, pp. 52-57, 2017.
- [6] X. Tao, R. K. Hite and R. MacKinnon, "Cryo-EM structure of the open high-conductance Ca²⁺-activated K⁺ channel," *Nature*, vol. 541, pp. 46-51, 2017.
- [7] A. e. a. Domanska, "A 2.8-angstrom-resolution cryo-electron microscopy structure of human parechovirus 3 in complex with Fab from a neutralizing antibody," *Journal of virology*, vol. 93, no. 4, pp. e01597-18, 2019.
- [8] M. W. e. a. Martynowycz, "Collection of Continuous Rotation MicroED Data from Ion Beam-Milled Crystals of Any Size," *Structure*, 2019.
- [9] L. e. a. Zhu, "Structures of Coxsackievirus A10 unveil the molecular mechanisms of receptor binding and viral uncoating," *Nature communications*, vol. 9, no. 1, p. 4985, 2018.
- [10] D. Si, S. Ji, K. A. Nasr and J. He, "A Machine Learning Approach for the Identification of Protein Secondary Structure Elements from Electron Cryo-Microscopy Density Maps," *Biopolymers*, vol. 97, no. 9, pp. 698-708, 2012.
- [11] R. Li, D. Si, T. Zeng, S. Ji and J. He, "Deep convolutional neural networks for detecting secondary structures in protein density maps from cryo-electron microscopy," in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2016.
- [12] D. Si and J. He, "Combining image processing and modeling to generate traces of beta-strands from cryo-EM density images of beta-barrels," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Chicago, 2014.
- [13] A. D. Palu, J. He and E. Pontelli, "Identification of α -helices from low resolution protein density maps," in *Computational Systems Bioinformatics*, 2006.
- [14] D. Si and J. He, "Beta-sheet Detection and Representation from Medium Resolution Cryo-EM Density Maps," in *In Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*, 2013.
- [15] D. Si and J. He, "Si, Dong, and Jing He. "Tracing beta strands using StrandTwister from cryo-EM density maps at medium resolutions," *Structure*, vol. 22, no. 11, pp. 1665-1676, 2014.
- [16] A. Ng and D. Si, "Genetic Algorithm Based Beta-Barrel Detection for Medium Resolution Cryo-EM Density Maps," *Springer*, pp. 174-185, 2017.
- [17] P. D. e. a. Adams, "PHENIX: a comprehensive Python-based system for macromolecular structure solution," *Acta Crystallographica Section D: Biological Crystallography*, vol. 66, no. 2, pp. 213-221, 2010.
- [18] T. C. P. D. A. P. V. A. a. O. V. S. Terwilliger, "A fully automatic method yielding initial models from high-resolution electron cryo-microscopy maps," *Nature methods*, vol. 1, 2018.
- [19] T. C. e. a. Terwilliger, "Automated map sharpening by maximization of detail and connectivity," *Acta Crystallographica Section D: Structural Biology*, vol. 74, no. 6, 2018.
- [20] T. C. Terwilliger, "Rapid model building of α -helices in electron-density maps," *Acta Crystallographica Section D: Biological Crystallography*, vol. 66, no. 3, pp. 268-275, 2010.

- [21] T. C. Terwilliger, "Automated main-chain model building by template matching and iterative fragment extension," *Acta Crystallographica Section D*, vol. 59, no. 1, pp. 38-44, 2003.
- [22] T. C. Terwilliger, "Rapid chain tracing of polypeptide backbones in electron-density maps," *Acta Crystallographica Section D: Biological Crystallography*, vol. 66, no. 3, pp. 285-294, 2010.
- [23] B. e. a. Frenz, "RosettaES: a sampling strategy enabling automated interpretation of difficult cryo-EM maps," *Nature methods*, vol. 14, no. 8, p. 797, 2017.
- [24] G. Terashi and D. Kihara, "De novo main-chain modeling for EM maps using MAINMAST," *Nature*, 2018.
- [25] X. Z. S. R. J. S. Kaiming He, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [26] B. H. B. B. C. D. M. A. Y. N. Richard Socher, "Convolutional-Recursive Deep Learning for 3D Object Classification," *Advances in neural information processing systems*, 2012.
- [27] Y. B. G. H. Yann LeCun, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
- [28] E. S. T. D. Jonathan Long, "Fully Convolutional Networks for Semantic Segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, pp. 3431-3440, 2015.
- [29] V. A. K. a. R. C. Badrinarayanan, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv*, vol. 1, 2015.
- [30] F. a. V. K. Yu, "Multi-scale context aggregation by dilated convolutions," *arXiv*, vol. 1, 2015.
- [31] G. Tang, L. Peng, P. R. Baldwin, D. S. Mann, W. Jiang, I. Rees and S. J. Ludtke, "EMAN2: An extensible image processing suite for electron microscopy," *Journal of Structural Biology*, vol. 157, pp. 38-46, 2007.