Detection of Beta-Barrels from Medium Resolution Cryo-electron Microscopy Density Maps

Albert Ng

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Computer Science and Software Engineering

University of Washington

2018

Committee:

Dong Si

Kelvin Sung

Munehiro Fukuda

Program Authorized to Offer Degree:
Computing & Software Systems

University of Washington

Abstract

Detection of Beta-Barrels from Medium Resolution Cryo-electron Microscopy Density Maps

Albert Ng

Chair of the Supervisory Committee:
Dong Si
Computing & Software Systems

Cryo-electron microscopy is becoming one of the most popular techniques used to determine protein structures. At medium resolutions, secondary structures can be seen from within cryo-electron microscopy density maps. The automatic isolation and detection of these secondary structures directly from density maps would be helpful to many, such as drug researchers to enable faster and better drug design for targeted proteins. One such secondary structure is the β-barrel, a β-sheet based secondary structure that is commonly found as in cell membranes and transport proteins. This thesis proposes a novel method combining convolutional neural networks, genetic algorithms, and ray casting to perform automatic detection for β-barrels from within medium resolution cryo-electron density maps. This approach was tested using both experimentally produced and simulated cryo-electron microscopy density maps. This method achieved a sensitivity of 0.95 and specificity of 0.90 on simulated maps. However, this method was only able to obtain a sensitivity of 0.72 and specificity of 0.79 on experimentally produced density maps.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to first thank my thesis advisor Professor Dong Si of the Department of Computing and Software Systems at the University of Washington Bothell. I would not have been able to complete this thesis without his guidance, support, and patience along the way.

I would also like to thank my committee members, Professor Kelvin Sung and Professor Munehiro Fukuda, for their support on this thesis.

Lastly, I would like to thank my family and friends for their support during my time at the University of Washington Bothell.

Chapter 1

INTRODUCTION

1.1 Motivation

Proteins are the fundamental building block of all living things on Earth. Every living thing, from the smallest single-cell bacteria to humans, is primarily made up of proteins. Additionally, proteins are responsible for regulating all of the biological processes and functions that enable living things to move, think, etc.

Because of this, the understanding of proteins and how they work is vital for fields like drug design and drug development. For example, understanding how a key virus protein is able to attack and target cells would be invaluable to researchers. This would allow them to design drugs that can target specific mechanisms unique to the virus protein in order to effectively inhibit it and prevent disease.

The key to understanding how a protein works and the mechanisms that allow the protein to complete its function is through its structure. The specific shape and structure of a protein determines what other molecules are allowed to bond and interact with it, thus determining its function and purpose.

To obtain high quality and highly detailed images of the 3D structure of proteins, researchers have relied on methods such as X-ray crystallography and nuclear magnetic resonance (NMR). Most of the protein structures solved in the Protein Data Bank (http://www.rcsb.org/) [2] were solved using X-ray crystallography. It was one of the first methods capable of resolving protein structures at atomic resolutions, allowing researchers to pick out individual atoms from one another in the image.

However, there are limitations to X-ray crystallography. In order to perform X-ray crystallography, the molecule must first be crystallized. Larger proteins, such as ribosomes and

surface proteins, cannot be crystallized easily and only until recently has X-ray crystallography finally been able to work with these types of proteins [3]. Another issue with the crystallization is that it can modify the structure of the protein, such that it is no longer in its native, physiological structure when the 3D structure is obtained. Additionally, staining and chemical fixation, steps done to the molecular sample beforehand to improve image quality and detail, can change the protein and its structure.

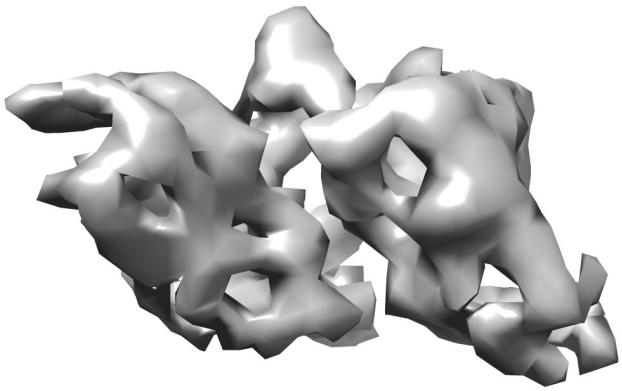

Figure 1.1: Example of Cryo-EM Density Map. Visualized using UCSF's Chimera [1]

Cryo-Electron Microscopy (cryo-EM) [4] is an increasingly popular technique that is also capable of taking high quality 3D images of proteins (see Figure 1.1), but it is capable of avoiding many of the limitations of other methods. Cryo-EM works by first freezing a sample of the molecule being studied to cryogenic (below $-180°$C or 93K) temperatures by placing the sample in an ethene bath cooled using liquid nitrogen. This has the advantage of preserving the sample in its native, physiological state. Afterwards, an electron microscope, which uses beams electrons instead of light for illumination, is then used to take two-dimensional (2D) images of the sample from as many different angles as possible. These images are then combined to reconstruct a full three-dimensional image, also known as a density map, of the

protein being studied.

Although cryo-EM was first invented and used in the 1970s, technological limitations prevented cryo-EM from being able to take high, atomic resolution density maps of proteins and it fell behind other methods like X-ray crystallography. However, with recent advances, cryo-EM has finally caught up and an increasing number of high, atomic resolution cryo-EM density maps have been produced.

Although cryo-EM can now produce high, atomic resolution cryo-EM density maps, there still are a large number of cryo-EM images produced in the previous decades that were resolved at lower, medium resolutions (˜5-10Å). At medium resolutions, the density surrounding each individual atom tend to merge and combine together. As a result, it is very difficult to differentiate one atom from another within the density maps at these resolutions.

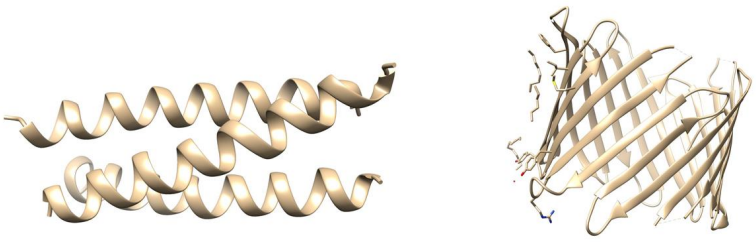

Figure 1.2: Two main types of secondary structures: α-helix (left) and β-sheets (right)

However, at medium resolutions, this merging of densities allow the secondary structures within proteins to become visible. Secondary structures are localized substructures within proteins. These secondary structures combine with one another to form the overall structure and shape of the protein. Additionally, these secondary structures themselves have common roles and functions within the proteins themselves. Because of this, the ability to detect secondary structures from within proteins can be very useful for understanding the overall structure of the protein.

The two main types of secondary structures are α-helices and β-sheets (See Figure 1.2).

α-helices are the most common form of secondary structure in proteins. Within medium resolution cryo-EM density maps, α-helices generally show up as thin cylinders of density. Additionally, α-helices are extremely regular and well defined. This has allowed it be fairly easy to predict and detect compared to β-sheets.

From within medium resolution cryo-EM density maps, β-sheets show up as thin sheets of density. However, unlike α-helices, β-sheets can fold, twist, and combine to form a myriad of different shapes and structures (See Figure 1.3), each with their own purpose and function. This has resulted in a far more difficult task to predict and detect β-sheets and all of its potential shapes and forms.

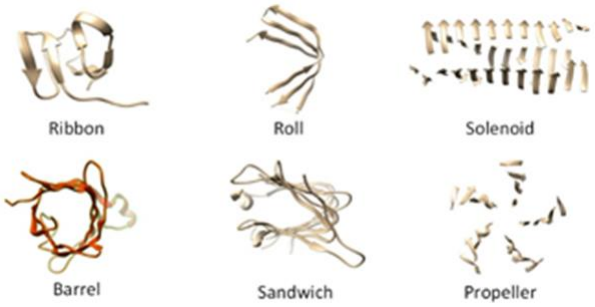

Figure 1.3: Examples of β-sheet based secondary structures

β-barrels are one such potential form that β-sheets can form [5]. β-barrels are one of the more commonly found β-sheet based secondary structures in proteins. They are found in cell membrane proteins, where they facilitate the movement of molecules entering and exiting the cell. Additionally, they are found in transport proteins, where they carry molecules to different cells across the body. β-barrels generally show up as a thin hollow cylinder in medium resolution cryo-EM density maps, although the shape of β-barrels can vary greatly.

1.2 Our Research

In this thesis, we present our novel method to perform automatic detection of β-barrels. Although the ultimate goal of our research is to perform automatic detection of all secondary

structures, because of the large number of β-sheet based secondary structures, we focused on one specific secondary structure. We chose to focus on the automatic detection of β-barrels, as they are one of the most commonly found secondary structures.

Our method is separated into multiple components. The first and main component of our method consists of a convolutional neural network that is responsible for labeling and detecting the β-barrel from medium resolution cryo-EM density maps. In order to increase the accuracy of our method, the output of the convolutional neural network undergoes post-processing. The first post-processing (and second overall) component consists of a genetic algorithm that attempts to fit an ideal cylinder to β-barrel region. The second post-processing component uses ray casting to accurately isolate the voxels associated with the β-barrel.

1.3 Outline of this Thesis

The rest of this thesis is organized as follows. In Chapter 2, a literature review is provided to explore previous work done in this area. In Chapter 3, a background on the major concepts used in this thesis is provided. In Chapter 4, the implementation details of our method is presented. In Chapter 5, the results from our method is presented along with discussion and analysis of these results. Lastly, in Chapter 6, we conclude the thesis by discussing any current limitations to our method and suggest potential future work that could be done to improve our method.

Chapter 2
LITERATURE REVIEW

In this chapter, an overview of the general history of secondary structure detection, secondary structure detection from cryo-electron density maps, and our work leading up to this thesis is presented.

2.1 Past Secondary Structure Detection

Research on methods and techniques to automatically identify the secondary structures located within proteins have been worked on for decades. The Chou-Fasman method [6], developed by Chou and Fasman in the 1970s, was one of the first techniques developed to be able to predict the location of secondary structures. An empirically-based method, the Chou-Fasman method would take the protein's amino acid sequence and output probabilities for each amino acid being part of a specific secondary structure. Additional methods such as the GOR method [7] and the one developed by Lim [8] would follow. These methods would achieve accuracies of approximately 55-60% [9]. Lately, machine learning based methods, like PSIPRED [10], would be developed in the 1990s and achieve up to around 80% accuracy [11].

All of these methods used the protein's amino acid sequence as input to detect the secondary structures located within the protein. The key limitation with using just the amino acid sequence as input, is that the amino acid sequence does not contain any information about how the protein amino acid sequence is oriented and folded in 3D space [12]. Secondary structures form when nearby amino acids interact in specific ways to bond and combine into specific shapes and these interactions may be done by amino acids not local to each other in the sequence [11]. As a result, there is no possible way that any method based on the amino

acid sequence alone can have this information.

2.2 Secondary Structure Detection For Cryo-EM

One way to avoid this limitation was to utilize three-dimensional images, such as the density maps produced by cryo-EM, as the input instead. Unlike the amino acid sequence, it is possible to take into account the localized interactions between amino acids within the sequence.

As α-helices consistently appear as cylinders at medium and lower resolutions, one of the first methods that used three-dimensional density maps for secondary structure detection was Jiang's Helixhunter [13] to detect α-helices. Further research would be done on automatic detection of α-helices on medium resolution cryo-EM density maps, achieving very high accuracies [14] [15] [16]. Because of this, the focus of our method was to perform detection on the more difficult β-sheet secondary structures.

Unlike α-helices, β-sheets can form and combine to form numerous different geometrical configurations and structures. As a result, performing any sort of automatic detection of β-sheets is far more difficult, as any such method would need to be able to handle a large number of potential shapes and geometries. Recent research [17] [18] done on β-sheet detection has been successful in detecting the β-sheet itself, but not the larger structure these sheets may form.

For β-barrels, some preliminary work has been started. In Si's *BarrelMiner*, a random sample consensus based method was used to attempt to fit an ideal cylinder to match the β-barrel using a cryo-EM density map [19]. However, a major issue *BarrelMiner* faced was that β-barrels themselves can vary greatly in shape and size. Simple ideal cylinders struggled to fit some of the more irregularly shaped β-barrels, resulting in reduced accuracy and detection rate.

2.3 Our Previous Work

We proposed a new method in [20] using a combination of genetic algorithms and ray casting to perform β-barrel detection. A genetic algorithm would be used to fit an ideal cylinder to the general area of the β-barrel. Using this ideal cylinder, the ray casting portion would be used to attempt to discover the actual β-barrel shape and account for the irregular shapes commonly found in β-barrels.

Following from this, we continued to refine our method in [21]. The two major targets for improvement were: the issue with the genetic algorithm fitted cylinder being either too long or too short compared to the actual β-barrel length and the amount of time that the ray casting portion was taking up. As a result of this, we developed a method to stretch and shrink the cylinder to better match the actual length of the β-barrel. Additionally, in order to reduce the computation time needed to run the ray casting portion, a kd-tree was implemented to help speed up the ray casting.

In [22], further optimization to improve the speed of the ray casting portion was done. To do this, we utilized graphical processing unit (GPU) based parallelization on the ray casting portion. However, we found that although the time needed to perform ray casting was drastically reduced and the maximum number of rays that we could feasibly cast without significant slow down increased, the accuracy did not improve. As a result, for this thesis, we focused our efforts on improving the processes upstream of the genetic algorithm and ray casting and attempted to explore convolutional neural networks as a solution to this problem.

Chapter 3
BACKGROUND

In this chapter, an overview of the three major concepts used in this thesis is presented: machine learnin and convolutional neural networks, genetic algorithms, and ray casting.

3.1 Machine Learning

Machine learning is the class of methods where computers are taught to perform a task without explicitly programming an algorithm to enable them to. This is different from the traditional approach in computer science towards solving a problem, where a specific algorithm and/or a set of rules is created to solve the problem. Instead, machine learning seeks to use prior examples to "teach" a machine to perform a task or solve a problem.

Machine learning is generally divided into two major categories: supervised learning and unsupervised learning. In supervised learning, the true labels of the training data is known beforehand. This allows us to train the machine learning model to recognize patterns and learn how to map the inputs to specific labels for output. Alternatively, in unsupervised learning, the true labels of the data being analyzed is not known beforehand. As a result, the goal of unsupervised learning is to discover any clusters of data in the input and extract features from the input itself.

One of the major tasks of machine learning and supervised learning is performing classification on a data set. In classification, the goal is to train a machine learning model that can take in input data and output a specific label using the attributes and features in the input data. Developing models capable of automatic and accurate classification for a wide variety of data is of great interest to many different fields, from computer security and image processing to biochemistry and medicine.

A variety of algorithms have been proposed and developed for performing classification. These include older statistical analysis based methods, such as Discriminant Analysis and Naive Bayes, to more modern classification algorithms, such as support vector machines, decision trees, and neural networks. This thesis will focus on neural networks used for classification.

3.1.1 Neural Networks

Neural networks are an attempt to imitate how neurons work in living beings using a computer. The original implementation of the neural network was the perception [23]. In the perception, input would be weighted and summed together to produce a value. The value would be inputted into a simple activation function. If the value was above a specific threshold, the perception/neuron would fire and output a value of 1 as output. If the value was not above the threshold, the perception/neuron would output a value of 0. While training the perception with training data, the perception would learn how to classify input by adjusting the input weights. Further research would be done to optimize the performance and efficiency of these perceptions over the years. For example, alternatives to the original activation function were found to perform and/or train better, such as the very popular rectified linear unit (ReLU) activation function [24].

One of the major innovations to neural networks was to add multiple layers of perceptions to form a neural network. A limitation with the classic single layer perception was that it was not capable of handling non-linear separable functions [25]. Additional layers could be inserted between the input and output layers and these *hidden layers* were capable of learning to solve non-linearly separable problems [26]. These hidden layers are generally fully connected, which means that any single neuron in a given layer is connected to every neuron in the next layer. These multi-layer based neural networks would be referred to as Multi-Layer Perceptions (MLP).

3.1.2 Convolutional Neural Networks

However, in fields, such as image processing and computer vision, a major limitation of the application of multi-layer perceptions is the sheer potential size of the feature space that problems in these fields require. Because of large feature space necessary, the number of neurons, layers, and weights needed to fully handle such a feature space increases exponentially and requires an equally large enough training set to properly train. Additionally, the standard multi-layer perceptions do not take into account spatial locality between features. In image recognition, what features are nearby other features in the image is very important. For example, when attempting to recognize a person's face, the position of the eyes, nose, and mouth on the face and how they are oriented to each other is vital for recognition. This led to the development of convolutional neural networks (CNN).

LeCun [27] proposed the use of *convolution layers* within MLPs to tackle this limitation. In the convolution layer, the spatial relationship between local input features is extracted. This is done by using a smaller matrix of numbers, called the filter. This filter is shifted over the input by a specific distance every step, called the stride. At each step, the dot product of the filter and the input that the filter overlaps over is calculated and stored in an matrix for output. This is repeated until the entire input has been covered and the fully calculated output matrix is sent to the next layer in the CNN. By using the filter to extract features from a portion of the input, this greatly reduces the number of parameters and weights necessary compared to a traditional fully connected neural networks along with extracting localized features. The size of the filter, the values in the filter, and the stride size can be varied to optimize the performance of the CNN. For example, smaller filters is capable of extracting more information for the next layer at the cost of more steps required to cover the entire input. Conversely, larger filters are faster, but results in less information being passed to the next layer.

Additionally, convolutional neural networks include pooling layers after convolutional layers. This is done to further reduce the size of the input. This is also done to promote

translation invariance and to avoid overfitting, where the model is fit too well on training data and performs poorly on new test data as a result. A pooling layer is very similar to a convolutional layer. The only difference is that instead of calculating the dot product between the filter and input, a specific function is applied instead. The most popular function for the pooling layer is the max-pooling function. In max-pooling, the maximum value within the portion of the input that overlaps with the filter is selected and passed onto the next layer.

After a number of convolutional and pooling layers, the input is then fed to a traditional multi-layer perception. This is done to allow the convolutional neural network to learn how to perform classification on the input. As the input data has been drastically reduced in size and convolution has been performed to capture local features, the limitations that prevented use of a multi-layer perception is largely averted.

3.2 Genetic Algorithm

The genetic algorithm is an optimization algorithm based on the concept of natural selection [28]. The genetic algorithm attempts to find the solution to a problem by iterating through numerous generations of solutions that slowly evolve towards the optimal solution.

A genetic algorithm requires two primary components: a set of candidates that represent potential solutions and a fitness function that evaluates these candidates. Each candidate consists of a genome, or a set of properties that defines the candidate and is used in the fitness function to calculate fitness. The fitness function must be designed so that it will assign higher scores or "fitness" to candidates that represent more optimal solutions to the problem being solved.

The genetic algorithm is generally divided into three major steps: selection, crossover, and mutation. The selection step of the genetic algorithm involves selecting which candidates are allowed to pass down their "genes" to the next generation of candidates. There are numerous algorithms and methods to determine which candidates are selected [29], but generally candidates with higher fitness are prioritized for selection in order to pass on "successful" genes to successive generations. This is done so that successive generations

should have higher fitness than previous generations, therefore each generation is making progress towards reaching an optimal solution.

The next step in a genetic algorithm is the crossover step. In this step, the selected candidates and their genes are used to produce a new generation of candidates. Although this process may vary, multiple "parent" candidates will pass on a portion of their genes to create a child candidate. This process is repeated until a full set of child candidates is created. This step is done to create potential solutions that have not yet been explored.

The last step, mutation, is done to introduce randomness to the genetic algorithm. Without mutation, a genetic algorithm's performance would be completely dependent on the initial set of candidates and their genes. By introducing random mutations to child candidates after crossover, potentially new genome combinations of genes and solutions can be explored. Additionally, by introducing mutation, the randomness from mutation allows the algorithm to avoid becoming trapped in local minima/maxima.

Using the newly created child generation, these three steps can be repeated as necessary until either an optimal solution is found or a stop condition has been reached.

3.3 Ray Casting

Ray casting is a computer graphics technique that renders images by attempting to mimic how light rays and our eyes work to enable vision. First proposed in 1968 by Arthur Appel [30], ray casting represents the simplest algorithm in a class of graphics rendering techniques that attempt to mimic light called ray tracing [31].

Algorithm 1: Ray Casting Algorithm

Data: List of Objects, O; ray origin, P_0; ray direction, P

Result: Object that Ray Collided with, $O_(min)$

```
1  foreach Object O_i in O do
2      if O_i intersects with P then
3          Calculate distance, d_i, from P_0 to O_i
4          if d_i < d_(min) then
5              Set O_(min) to O_i
6              Set d_(min) to d_i
7          end
8      end
9  end
10 return O_(min);
```

In ray casting (see Algorithm 1), a ray is shot out from an origin point. From this origin point, the ray will iterate through all possible objects in the scene/universe (line 1). For each object, a check is made to determine if the ray intersects with the object (line 2). If the object intersects with the ray, the distance from the origin to the object is calculated (line 3). This is done in order to find the object that is closest from the origin in the path of the ray (line 4 to 6). The object that intersects with the ray and is closest to the object is returned as the result (line 10).

To render an image, a ray is casted for every pixel in the image. Based on the optical properties of the object the ray collided with (i.e color), the color and shading of the pixel is determined. This is repeated for every pixel in the image until the full image is rendered. This ray casting method can be extended to handle reflections, refractions, and other graphical details [31].

Chapter 4

IMPLEMENTATION

In this chapter, the overall implementation of our method (see Figure 4.1), its details, and how it was tested is presented.

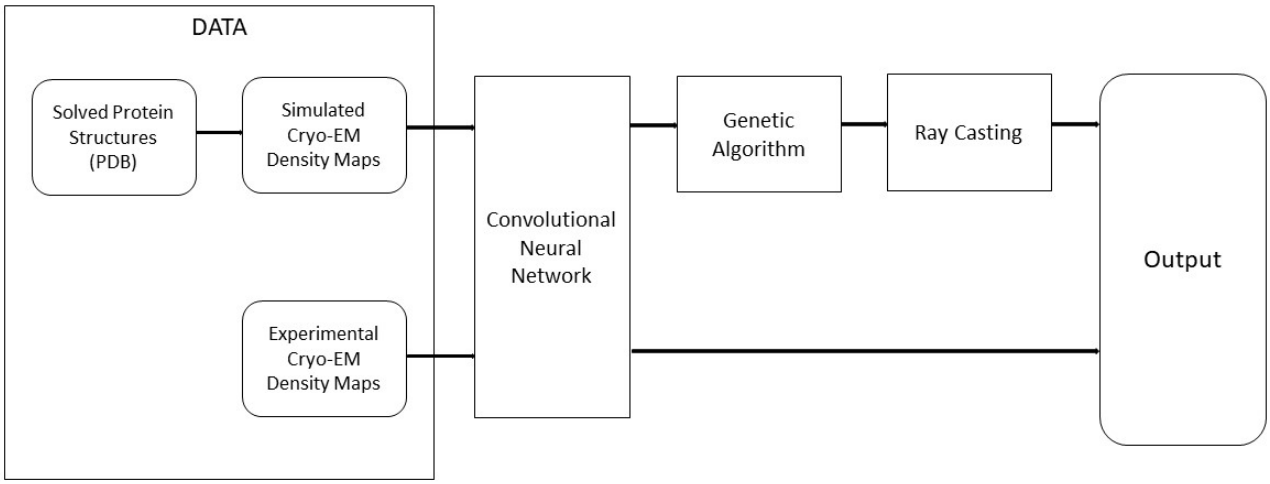

Figure 4.1: Overview of our method

4.1 Cryo-EM Density Maps (MRC)

In order to test our method's viability and its accuracy, cryo-EM density maps were used to train the convolutional neural network and to test the overall ability of our method to detect β-barrels. These cryo-EM density maps would be stored in the MRC file format, the standard file format for cryo-EM density maps [32]. Each MRC file consists of a three-dimensional grid of voxels (a 3D pixel) and a header that contains the metadata for the density map. Each voxel in the grid contains a value corresponding to the electron density at that voxel position.

4.1.1 Experimentally Produced Density Maps

Experimentally produced cryo-EM density maps between 5 and 10 Å (medium resolution) were obtained from the EMDataBank (http://www.emdatabank.org/) [33]. Only density maps for proteins that had a correspondingly solved protein structure were selected, as we needed to be able to determine the accuracy of our detection method. Solved protein structures were obtained from the CATH database (http://www.cathdb.info/) [34] and the Protein Data Bank (http://www.rcsb.org/) [2]. These were stored in the standard Protein Data Bank (PDB) file format. In the end, a total of 8 experimental cryo-EM density maps at medium resolution containing β-barrels were found. Due to the limited quantity of experimental cryo-EM density maps, none of these maps were used for training and validation and all were used for testing.

Additionally, because experimental cryo-EM density maps can use any maximum or minimum for their electron density values, all experimental density maps had their electron density values normalized to between 0 and 1. This was done to both allow for fair comparison of electron density between different experimental maps, but also to match simulated cryo-EM density maps which are automatically scaled to between 0 and 1.

4.1.2 Simulated Density Maps

Due to the limited quantity of experimentally produced cryo-EM density maps, simulated cryo-EM density maps would be used for both training and testing to augment the data set. These simulated density maps are created using the actual protein structure (stored as a PDB file) and attempts mimic how an ideal cryo-EM density map would look if cryo-EM was actually performed on that protein. These simulated cyro-EM density maps were created using the *pdb2mrc* command in EMAN2 [35] with the PDB files. Each simulated map was simulated at a resolution of 9 Å, using a box size of *64* x *64* x *64*, and at 1Å per voxel. The PDB files were obtained from the CATH database (http://www.cathdb.info/) [34] and the Protein Data Bank (http://www.rcsb.org/) [2]. In the end, a total of 53 simulated cryo-EM density maps were created. These were divided into a training set of 31 density maps, a validation set of 11 density maps, and a testing set of 11 density maps.

4.1.3 Augmentation

Data augmentation has been proven to increase the effectiveness of neural networks by both increasing the amount of data available to train the neural network and to help avoid over-fitting of the model during training [36] [37]. Data augmentation was performed on the simulated cryo-EM density maps in order to maximize our limited data set. Each original simulated cryo-EM density map would be rotated 90° in the x, y, and z directions to produce six additional cryo-EM density maps each.

4.1.4 Labeling

In order to train the convolutional neural network to perform classification and for us to be able to verify the accuracy of the detection later, the training data must be labeled with the correct classification beforehand. For this method, only two labels/classes were used for each voxel in the density maps: part of a β-barrel and not part of a β-barrel.

Using the solved protein structure in the PDB file, a voxel would be labeled as part of a

β-barrel if it was within 2 Å of any α-carbon of an amino acid that is part of a β-barrel. An α-carbon represents the central carbon atom within an amino acid. In general, it is common to approximate the location of the amino acid with the location of its α-carbon. If the voxel is not within 2 Å, our labeling process would label it is not part of a β-barrel.

4.2 Convolutional Neural Network

The purpose of the convolutional neural network is to scan through the cryo-EM density map and isolate the voxels associated with the β-barrel. In order to label each voxel individually, our method feeds specific subsection of the entire density map, called patches, to the convolutional neural network instead of the whole density map [38] [39]. This is done because feeding the entire density map into the convolutional neural network would only return a single label (β-barrel or not) for the entire density map and our method needs labels for every voxel. To get around this limitation, we break up the entire density map into patches.

For each voxel in the density map, the electron density values surrounding that voxel is obtained in order to form the patch. For this method, we used an *9* x *9* x *9* patch size, so that the voxel of interest is at (*5*,*5*,*5*). For voxels outside of the original dimensions of the density map, like for the *64* x *64* x *64* box for simulated density maps, the electron density value is assumed to be 0. Google's Tensorflow [40] library was used to build and run the convolutional neural network.

The convolutional neural network, as seen in Figure 4.2, consists of two convolutional layers, two max-pooling layers, and a single dense layer. The first convolutional layer (CL1) takes in the *9* x *9* x *9* patch and applies 32 *3* x *3* x *3* filters to the patch using a stride size of 2. Additionally, this convolutional layer used an eLU activation function [41] and a zero padding of 1. After CL1, a max-pooling layer (MP1) using *2* x *2* x *2* filters and a stride of 2 was applied to downsample the input. The second convolution layer (CL2) takes in the resulting *4* x *4* x *4* input and applies 64 *3* x *3* x *3* filters to the input using a stride size of 1. Like the first convolutional layer, a zero padding of 1 was used to maintain the size and an eLU activation function was used. Lastly, a max-pooling layer (MP2) is applied to the

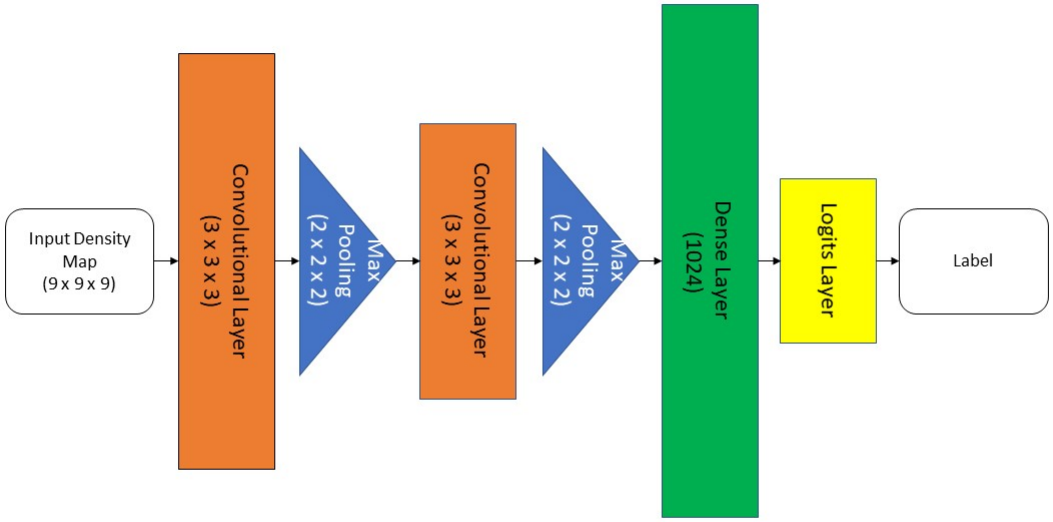

Figure 4.2: Architecture of the Convolutional Neural Network

output of the second convolution layer using a *2* x *2* x *2* filter and a stride size of 2.

The output coming out of the max-pooling layer is then fed into a single dense layer consisting of 1024 neurons activated using the ReLU activation function. This layer is responsible for performing classification on the features extracted by the previous layers. The output of this dense layer is then fed into a Logits layer. Using softmax normalization, the Logits layer will then attempt to map class probabilities from the dense layer output. These probabilities will then be used to determine which of two labels (part of a β-barrel and not part of a β-barrel) is output as the final result.

The convolutional neural network was trained using the Gradient Descent Optimizer using a learning rate of 0.01. Tensorflow's sparse softmax cross entropy with logits was used as the training loss function. The training batch size was 10,000 patches for 1000 epochs.

Once the patch has gone through the convolutional neural network, the label output by the CNN is now associated with the voxel at the center of the patch (*5,5,5*).

4.3 Genetic Algorithm

The purpose of the genetic algorithm is to fit an ideal cylinder to the center of the β-barrel region within the density map. Once this ideal cylinder is fitted, the ray casting can then be performed to isolate the β-barrel voxels from within the density map.

4.3.1 Candidate

The candidate used in our genetic algorithm is an ideal cylinder, defined by seven parameters [x_1, y_1, z_1, x_2, y_2, z_2, r]. These parameters represent the two points at the center of the two circles/ends of the cylinder along with the radius of the cylinder. The cylinder is considered to be hollow and open ended to match β-barrels. The initial set of cylinder candidates is produced by randomly selecting values for each parameter. For the six coordinate parameters, a value is randomly selected but bound by the dimensions of the density map. The radius parameter is randomly selected between 2 Å and 10Å.

4.3.2 Selection, Crossover, Mutation

The selection step is performed by first sorting the cylinder candidates by their fitness score. Afterwards, the top fifty percent of candidates are then selected to be parents for the next generation. The lowest fifty percent of candidates is discarded. Using these selected cylinder candidates, crossover is performed by randomly pairing up two selected cylinders together. Each parent pairing will then produce two children cylinders. For each parameter, one of the child cylinders will randomly select one of the parent's parameters and the other child will select the parameter that was not chosen. This is repeated until the both child cylinders have a full complement of parameters. For the mutation step, during the creation of each child cylinder, for each parameter there is a small chance that it will mutate and vary by

a percentage. The percentage that each parameter will vary by is determined using a non-uniform mutation strategy.

4.3.3 Fitness Score

The fitness score attempts to define how well the ideal cylinder fits to the β-barrel region. The fitness score, F, is given by Equation 4.1.

$$F = \sum_{i}^{N} \frac{(d_i)^2}{N} + \sum_{j}^{M} (d_j)^3 \tag{4.1}$$

The first half of Equation 4.1 attempts to calculate the mean squared distance. N represents the total number of voxels in the density map and d_i is the distance between the voxel and the cylinder. The second half of Equation 4.1 attempts to calculate a penalty for voxels being located inside of the β-barrel wall. M represents total number of voxels inside of or on the β-barrel surface and d_j is the distance between any such voxel and the surface of the cylinder.

4.4 Ray Casting

Once the genetic algorithm has fitted an ideal cylinder to the β-barrel region, ray casting is done to isolate the voxels associated with the β-barrel. This step is performed because β-barrels can vary greatly in shape and no single shape can represent them all. By fitting a simple ideal cylinder to the β-barrel region and using ray casting to identify the β-barrel voxels, our method is capable of handling β-barrels of any shape.

The cylinder is split up into evenly divided slices along the long axis of the cylinder. Within each slice, rays are shot every 1° from the cylinder outwards (See Figure 4.3). The origin point of each ray is the midpoint between the top and bottom of the slice on the central axis of the cylinder. The objects in this case is every voxel remaining in the density map. Because voxels are infintesimally small, the casted rays are given a "thickness" of 0.5 Å. The ray will then find the closest voxel to the cylinder slice. This returns a set of voxels

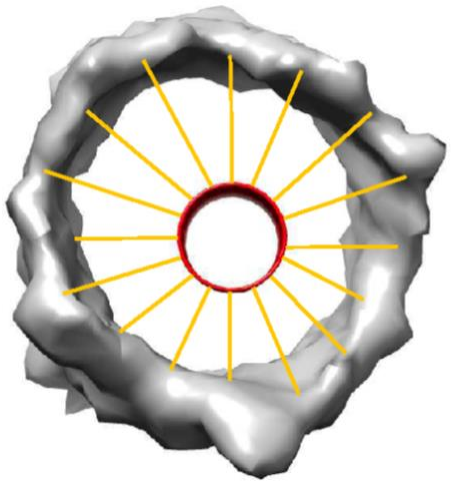

Figure 4.3: Ray Casting. Red circle represents the fitted cylinder and the yellow lines represents the rays being casted to find the β-barrel walls.

per slice of the rays that collided with it.

4.4.1 Parallelization

One of the key limitations to ray casting is its relatively high computational cost. The computational cost of ray casting scales with both number of rays casted along with the number of objects in the scene. Given that a *64* x *64* x *64* density map contains 262,144 voxels, repeating the ray casting algorithm for thousands of rays will increase the time needed to execute this method. In our previous work, this was a major bottleneck as the ray casting portion would dominate the time needed for completion along with limiting the number of rays that could be casted [20] [21]

Fortunately, the ray casting algorithm is embarrassingly parallel, as the algorithm does not contain many of the traditional limitations that prevent easy parallelization [42]. Each ray does not depend on data from any other ray, only the set of objects/voxels and the direction of the ray. As a result, we decided to parallelize the ray casting such that each ray would be run on its own thread.

However, the average CPU contains two to four cores, limiting the number of threads that can be run in parallel to two or four. Given that we are potentially casting thousands of rays per density map, this does not seem. As a result of this, we chose to take advantage of accelerators, such as graphical processing units (GPU), to speed this up even further. GPUs are designed to run far more, potentially thousands of, threads simultaneously compared to the average CPU and, as a result, achieve massive speed up for algorithms that can be parallelized [43].

For this thesis, the C++ Accelerated Massive Parallelism (AMP) library was used to offload the ray casting computations onto the GPU [44]. The results of this parallelization can be seen in Table B.1 and Table B.2 in Appendix B.

4.5 Postprocessing

One limitation that the genetic algorithm has when attempting to fit the cylinder into the center of the β-barrel region is that the resulting cylinder tends to be of the wrong length. To account for this, the cylinder can be stretched or shrunk. Starting from the two outer slices, the ratio of rays colliding with a voxel over the number of rays shot is calculated. If this ratio is above 0.5, an additional slice is added to that end of the cylinder. Rays are then casted from this slice to determine the ratio for that slice and this is repeated until a slice that does not meet the 0.5 ratio is reached.

However, if the ratio is below 0.5, then the slice is removed and the next slice in the cylinder is checked to see if it meets or exceeds the ratio. This continues until a slice that has at least a ratio of 0.5 is reached. If no slice reaches a ratio of 0.5, then all of the slices will have been removed and our method would determine that there is no β-barrel in this density map.

Additionally, because the β-barrel may not perfectly enclose the cylinder and/or gaps may form in the β-barrel wall. To take into account rays that may be cast through these gaps, voxels that are more than two standard deviations (σ) away from the mean distance between voxels and the cylinder surface are removed from the solution.

The sets of voxels for each slice are then combined together, giving us the set of voxels that represent the β-barrel. Unlike attempting to fit an single shape to match β-barrels, which can vary greatly in shape and size, this allows our method the ability to handle many different types of β-barrels.

4.6 Metrics

To determine and quantify the detection accuracy of our method, two metrics were used: the sensitivity and the specificity.

4.6.1 Sensitivity

Sensitivity (*Sens*) represents the true positive rate, or the ratio of β-barrel α-carbons that were correctly detected. A sensitivity of 1 would mean that our method was capable of detecting every single α-carbon that is a part of a β-barrel. The sensitivity for this method is calculated using Equation 4.2.

$$Sensitivity = \frac{\#\ of\ \beta\ barrel\ \alpha\ carbons\ detected}{total\ \#\ of\ \beta\ barrel\ \alpha\ carbons} \tag{4.2}$$

4.6.2 Specificity

Specificity (*Spec*) represents the true negative rate, or the ratio of non β-barrel α-carbons that were correctly identified. A specificity of 1 would mean that zero non β-barrel α-carbons were identified as part of a β-barrel. The specificity for this method is calculated using Equation 4.3.

$$Specificity = 1 - \frac{\#\ of\ non\text{-}\beta\ barrel\ \alpha\ carbons\ detected}{total\ \#\ of\ non\text{-}\beta\ barrel\ \alpha\ carbons} \tag{4.3}$$

Chapter 5

RESULTS

Using the previously described methods, the results of testing those methods are presented and discussed in this chapter.

5.1 Experimental Setup

The following results were evaluated using a desktop computer dual-booted Windows 10 and Ubuntu 18.04. The computer contained an Intel i7-4790k CPU, 16 GPU of RAM, and a nVidia GeForce 980Ti. The convolutional neural network was run using the Python 3.5 branch of the Tensorflow library 1.7 and was performed on the Ubuntu 18.04 partition. The genetic algorithm and ray casting portions of the method were run on the Windows 10 partition of the desktop computer.

For each density map that was tested, two different versions were tested for. The first test involved using the convolutional neural network only with no postprocessing. The resulting labeled output would be checked for detection accuracy directly. The second test was to utilize both the convolutional neural network and the genetic algorithm/ray casting.

These two tests were done in order to determine the performance of the convolutional neural network alone and to determine if performing the genetic algorithm and ray casting on the output improves the detection accuracy.

5.2 Results

In Table 5.1, the average results for both simulated and experimental cryo-EM density maps are shown for both tests. *CNN only* presents the results for just using the convolutional neural network for detection. *CNN + GA* presents the results using the convolutional neural

network, genetic algorithm, and ray casting.

Category	**CNN only**		**CNN + GA**	
Simulated	*Sensitvitiy*	*Specificity*	*Sensitvitiy*	*Specificity*
Average	0.95 ± *0.02*	0.82 ± *0.04*	0.93 ± *0.03*	0.90 ± *0.02*
Experimental	*Sensitvitiy*	*Specificity*	*Sensitvitiy*	*Specificity*
Average	0.72 ± *0.09*	0.66 ± *0.04*	0.72 ± *0.09*	0.79 ± *0.05*

Table 5.1: Detection Accuracy on Cryo-EM Density Maps (the 95% confidence interval is shown to the right of each result)

For the complete results for individual density maps, see Table A.1 and Table A.2 in Appendix A.

5.3 Discussion

The below images of the protein structures and the density maps were created using UCSF's Chimera [1].

5.3.1 Simulated Density Maps

From the results shown in Table 5.1, the accuracy for detecting simulated density maps is generally high. This suggests that for simulated density maps, our method is capable of accuracy detection of the β-barrel. However, the specificity for the CNN only version of our method was relatively low. Fortunately, once the genetic algorithm and ray casting was applied to the output of the convolutional neural network, this specificity increased from 0.82 to 0.90. Performing a t-Test on the specificity results in a p-value of 0.000252, which suggests that this increase in specificity was statistically significant. On the other hand, this increase in specificity did result in a small decrease in sensitivity from 0.95 to 0.93. Performing a

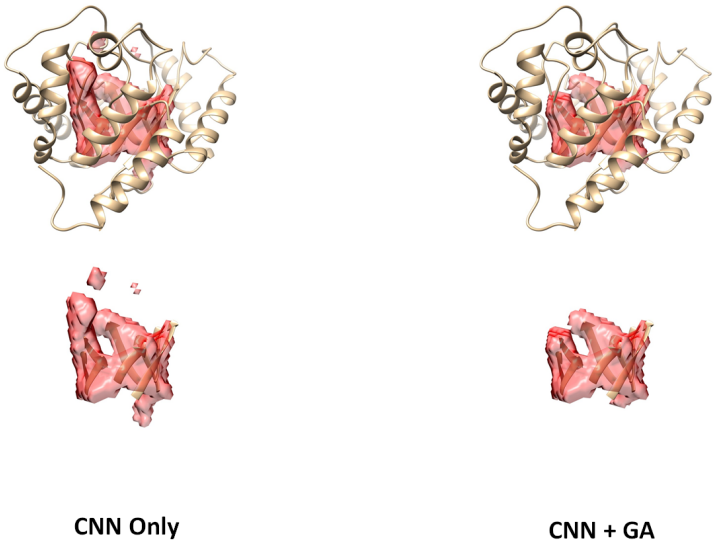

Figure 5.1: Results for Simulated Density Map for protein 1AJZ

t-Test on the sensitivity results in a p-value of 0.00534. This suggests that the decrease is statistically significant and that the increase in specificity did result in a significant decrease in sensitivity for simulated density maps.

This suggests that applying the genetic algorithm and ray casting to the output of the convolutional neural network seems to clean up signficant amounts of misclassified β-barrel density. This can be seen in Figure 5.1, where the detected β-barrel density (shown in red) is shown over the full true protein structure (top row) and the actual location of the β-barrel (bottom row) for both the CNN only run and the CNN + GA run.

In the CNN only run, the upper left and lower right portions of the density map was misclassified by the convolutional neural network. This is understandable as the local structure of those areas resemble a flat anti-parallel plane between two strands of the protein, which is very similar to the β-sheets that make up the walls of β-barrels. Once this output is fed into the genetic algorithm and ray casting, however, this misclassified density has been properly removed from the final output. As this misclassified density protrudes from the actual β-barrel, our ray casting method is capable of identifying this density as not part of

the β-barrel as it does not form a complete ring and removing it from the output. This results in the more accurately detected β-barrel seen in the CNN + GA portion of Figure 5.1.

5.3.2 Experimental Density Maps

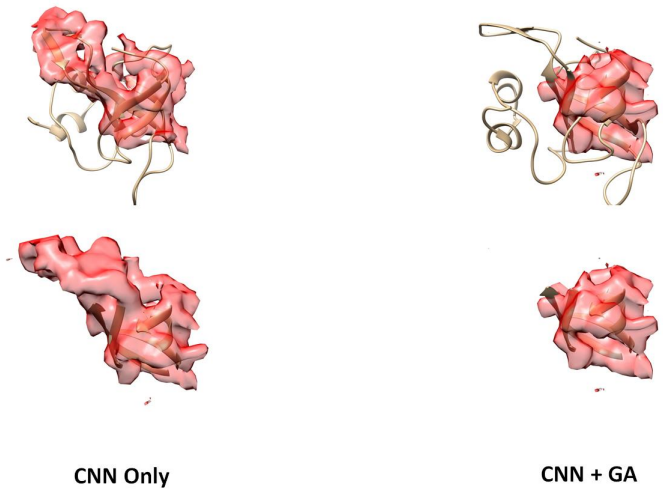

Figure 5.2: Results for Experimental Density Map 6396 for protein 5A9Z

However, the results also suggest that our method is not as accurate at predicting the β-barrel for experimental density maps. The sensitivity for both the CNN only version and the CNN + GA version was 0.72. Performing a t-Test on the sensitivity results in a p-value of 0.052. This suggests there is no significant change in the sensitivity between the two methods. However, as the p-value is very close to the 0.05 threshold, further testing should be done to confirm this. However, like in the simulated maps, using the CNN + GA version increased the specificity from 0.66 to 0.79. Performing a t-Test on the specificity results in a p-value of 0.000032. This suggests that the increase in specificity is statistically significant.

We assumed that due to the similarities between experimental and simulated cryo-EM density maps, we would only train on the far more numerous and easily obtained simulated density maps and save experimental density maps for testing. However, the large drop in

accuracy between experimental and simulated density maps using our method suggest that this assumption is incorrect.

In Figure 5.2, the results are shown for an experimental density map. The detected β-barrel density is shown in red and overlain on top of the full true protein structure (top row) and the actual β-barrel location (bottom row).

The additional misclassified density in the upper left portion of the red density map surrounds a small isolated β-sheet that is not part of the β-barrel. This results in the convolutional neural network labeling that section of the density map and density surrounding it as part of the β-barrel. When this output is fed into the genetic algorithm and ray casting, we can see that it removes this misclassified density from the output, thus returning output that more accurately identifies the β-barrel from within the density map.

5.3.3 Additional Observations

As noted earlier, these results suggests that our convolutional neural network is correctly searching for density that resembles the β-sheets that make up β-barrels. However, because the convolutional neural network is designed to take in only a small portion (a patch) of the entire density map, it classifies anything that resembles the β-sheet without regard for whether or not it is actually part of a β-barrel wall or not. Without being any knowledge of the density map structure outside of the patch, it cannot make the distinction between isolated β-sheets, like Figure 5.2, and the actual β-barrel.

Chapter 6

CONCLUSION

This thesis has described a method that attempts to perform automatic detection of β-barrel secondary structures from within medium resolution cryo-electron microscopy density maps. The ability to automatically detect β-barrels from within cryo-EM density maps would greatly help researchers study proteins and aid drug researchers develop drugs that can better target specific proteins.

6.1 Conclusion

The method discussed in this thesis utilizes a combination of a convolutional neural network, genetic algorithm, and ray casting to attempt to detect β-barrels from any medium resolution cryo-EM density map. A convolutional neural network was trained to be able to label any voxel in the density map as either part of a β-barrel or not when given a small subsection of the entire density map surround the voxel. Once the density map is labeled, a genetic algorithm is used to attempt to place an ideal cylinder into the center of the β-barrel region. Afterwards, ray casting is performed using the cylinder as the reference point to search for the voxels surrounding the cylinder. This set of voxels is then returned as the detected β-barrel voxels from within the density map.

The results suggest that our method is capable of accurately detecting β-barrels from simulated cryo-EM density maps, as these averaged a sensitivity of 0.95 and a specificity of 0.90. However, our method struggles with detecting β-barrels from experimental density maps, where it averaged a sensitivity of 0.72 and a specificity of 0.79. Additionally, our results suggest that using the convolutional neural network alone can result in poor specificity. Fortunately, the combination of the convolutional neural network with the genetic algorithm

and ray casting can significantly improve the specificity.

6.2 Future Work

As discussed earlier in Chapter 5, expanding the convolutional neural network to be able to take into account spatial information that isn't just local to the voxel being labels could reduce the number of false positives. This could be done by attempting to increase the patch size to allow the neural network to "see" farther out. Alternatively, the patch based convolutional neural network could be replaced by a different architecture/system that can perform voxel-wise classification of the density map in one go.

Additionally, better and/or improved training of the convolutional neural network along with optimizations to the convolutional neural network's architecture could be done to increase accuracy. All of the above improvements could increase the accuracy to the point where the genetic algorithm and ray casting is no longer necessary and true end-to-end detection can be performed.

Additionally, the training of the convolutional neural network could be repeated except including experimental density maps into the training set. This could be explored to see if it is possible to obtain similar detection accuracy between the simulated and experimental density maps. Lastly, though the scope of this thesis was focused on the one specific type of secondary structure, adapting this method to other secondary structures is another potential direction of research that could be pursued.

Lastly, work should be done to explore how to modify and expand the method discussed in this thesis to perform automatic detection of other secondary structures, specifically secondary structures that are geometrically similar to β-barrels.

BIBLIOGRAPHY

[1] Eric F. Pettersen, Thomas D. Goddard, Conrad C. Huang, Gregory S. Couch, Daniel M. Greenblatt, Elaine C. Meng, and Thomas E. Ferrin. UCSF Chimera–a visualization system for exploratory research and analysis. *Journal of Computational Chemistry*, 25(13):1605–1612, October 2004.

[2] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, January 2000.

[3] Elisabeth P Carpenter, Konstantinos Beis, Alexander D Cameron, and So Iwata. Overcoming the challenges of membrane protein crystallography. *Current Opinion in Structural Biology*, 18(5):581–586, October 2008.

[4] M. Adrian, J. Dubochet, J. Lepault, and A. W. McDowall. Cryo-electron microscopy of viruses. *Nature*, 308(5954):32–36, March 1984.

[5] William C Wimley. The versatile -barrel membrane protein. *Current Opinion in Structural Biology*, 13(4):404–411, August 2003.

[6] Peter Y. Chou and Gerald D. Fasman. Prediction of protein conformation. *Biochemistry*, 13(2):222–245, January 1974.

[7] J. Garnier, D. J. Osguthorpe, and B. Robson. Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *Journal of Molecular Biology*, 120(1):97–120, March 1978.

[8] V. I. Lim. Structural principles of the globular organization of protein chains. A stereochemical theory of globular protein secondary structure. *Journal of Molecular Biology*, 88(4):857–872, October 1974.

[9] Kabsch Wolfgang and Sander Christian. How good are predictions of protein secondary structure? *FEBS Letters*, 155(2):179–182, May 1983.

[10] D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2):195–202, September 1999.

[11] Ofer Dor and Yaoqi Zhou. Achieving 80% ten-fold cross-validated accuracy for secondary structure prediction by large-scale training. *Proteins: Structure, Function, and Bioinformatics*, 66(4):838–845, March 2007.

[12] B. Rost. Review: protein secondary structure prediction continues to rise. *Journal of Structural Biology*, 134(2-3):204–218, June 2001.

[13] W. Jiang, M. L. Baker, S. J. Ludtke, and W. Chiu. Bridging the information gap: computational tools for intermediate resolution structure interpretation. *Journal of Molecular Biology*, 308(5):1033–1044, May 2001.

[14] A. Dal Palù, J. He, E. Pontelli, and Y. Lu. Identification of alpha-helices from low resolution protein density maps. *Computational Systems Bioinformatics. Computational Systems Bioinformatics Conference*, pages 89–98, 2006.

[15] Mirabela Rusu and Willy Wriggers. Evolutionary bidirectional expansion for the tracing of alpha helices in cryo-electron microscopy reconstructions. *Journal of Structural Biology*, 177(2):410–419, February 2012.

[16] Dong Si, Shuiwang Ji, Kamal Al Nasr, and Jing He. A machine learning approach for the identification of protein secondary structure elements from electron cryo-microscopy density maps. *Biopolymers*, 97(9):698–708, September 2012.

[17] Dong Si and Jing He. Beta-sheet Detection and Representation from Medium Resolution Cryo-EM Density Maps. In *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*, BCB'13, pages 764:764–764:770, New York, NY, USA, 2013. ACM.

[18] Dong Si and Jing He. Combining image processing and modeling to generate traces of beta-strands from cryo-EM density images of beta-barrels. *Conference proceedings: ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*, 2014:3941–3944, 2014.

[19] Dong Si. Automatic Detection of Beta-barrel from Medium Resolution Cryo-EM Density Maps. In *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '16, pages 156–164, New York, NY, USA, 2016. ACM.

[20] Albert Ng and Dong Si. Genetic Algorithm Based Beta-Barrel Detection for Medium Resolution Cryo-EM Density Maps. In *Bioinformatics Research and Applications*, Lecture Notes in Computer Science, pages 174–185. Springer, Cham, May 2017.

[21] Albert Ng and Dong Si. Beta-Barrel Detection for Medium Resolution Cryo-Electron Microscopy Density Maps Using Genetic Algorithms and Ray Tracing. *Journal of Computational Biology*, 25(3):326–336, October 2017.

[22] Albert Ng, Adedayo Odesile, and Dong Si. GPU Accelerated Ray Tracing for the Beta-Barrel Detection from Three-dimensional Cryo-EM Maps. March 2018.

[23] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[24] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, June 2011.

[25] Marvin L. Minsky and Seymour A. Papert. *Perceptrons: Expanded Edition*. MIT Press, Cambridge, MA, USA, 1988.

[26] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989.

[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.*, 1(4):541–551, December 1989.

[28] John H. Holland. Genetic Algorithms. *Scientific American*, 267(1):66–73, 1992.

[29] Khalid Jebari. Selection Methods for Genetic Algorithms. *International Journal of Emerging Sciences*, 3:333–344, December 2013.

[30] Arthur Appel. Some Techniques for Shading Machine Renderings of Solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), pages 37–45, New York, NY, USA, 1968. ACM.

[31] Turner Whitted. An Improved Illumination Model for Shaded Display. *Commun. ACM*, 23(6):343–349, June 1980.

[32] Anchi Cheng, Richard Henderson, David Mastronarde, Steven J. Ludtke, Remco H. M. Schoenmakers, Judith Short, Roberto Marabini, Sargis Dallakyan, David Agard, and Martyn Winn. MRC2014: Extensions to the MRC format header for electron cryo-microscopy and tomography. *Journal of Structural Biology*, 192(2):146–150, November 2015.

[33] Catherine L. Lawson, Ardan Patwardhan, Matthew L. Baker, Corey Hryc, Eduardo Sanz Garcia, Brian P. Hudson, Ingvar Lagerstedt, Steven J. Ludtke, Grigore Pintilie, Raul Sala, John D. Westbrook, Helen M. Berman, Gerard J. Kleywegt, and Wah Chiu. EMDataBank unified data resource for 3dem. *Nucleic Acids Research*, 44(D1):D396–D403, January 2016.

[34] Natalie L. Dawson, Tony E. Lewis, Sayoni Das, Jonathan G. Lees, David Lee, Paul Ashford, Christine A. Orengo, and Ian Sillitoe. CATH: an expanded resource to predict protein function through structure and sequence. *Nucleic Acids Research*, 45(D1):D289–D295, January 2017.

[35] Guang Tang, Liwei Peng, Philip R. Baldwin, Deepinder S. Mann, Wen Jiang, Ian Rees, and Steven J. Ludtke. EMAN2: An extensible image processing suite for electron microscopy. *Journal of Structural Biology*, 157(1):38–46, January 2007.

[36] Luis Perez and Jason Wang. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *arXiv:1712.04621 [cs]*, December 2017.

[37] Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. Improved Relation Classification by Deep Recurrent Neural Networks with Data Augmentation. *arXiv:1601.03651 [cs]*, January 2016.

[38] 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)*, pages 41–46, December 2016.

[39] Dan C. Cireşan, Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2843–2851, USA, 2012. Curran Associates Inc.

[40] *TensorFlow: A system for large-scale machine learning.*

[41] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv:1511.07289 [cs]*, November 2015.

[42] Erik Reinhard and Frederik W. Jansen. Rendering large scenes using parallel ray tracing. *Parallel Computing*, 23(7):873–885, July 1997.

[43] D. De Donno, A. Esposito, L. Tarricone, and L. Catarinucci. Introduction to GPU Computing and CUDA Programming: A Case Study on FDTD [EM Programmer's Notebook]. *IEEE Antennas and Propagation Magazine*, 52(3):116–122, June 2010.

[44] Kate Gregory and Ade Miller. *C++ AMP: Accelerated Massive Parallelism with Microsoft Visual C++*. " O'Reilly Media, Inc.", 2012.

Appendix A

COMPLETE RESULTS

Table A.1 presents the results of this thesis for simulated cryo-EM density maps by PDB ID. Due to the fact that augmentation was performed on the simulated density maps, the below numbers are averaged between the different augmented density maps as they originated from the same protein/density map.

Table A.1: Detection Accuracy for Simulated Cryo-EM Density Maps

PDB ID	CNN only		CNN + GA	
	Sens	*Spec*	*Sens*	*Spec*
1AJZ_A	1.00	0.83	1.00	0.92
1AL7_A	0.94	0.85	0.90	0.88
1JB3_A	0.97	0.82	0.96	0.87
1NNX_A	1.00	0.89	0.98	0.93
1TIM_A	0.99	0.76	0.94	0.85
1Y0Y_A	0.93	0.69	0.93	0.87
2DYI_A	0.92	0.84	0.87	0.91
2F01_A	0.97	0.85	0.94	0.89
2VDF_A	0.89	0.89	0.89	0.92
3GP6_A	0.92	0.83	0.92	0.90
3ULJ_A	0.93	0.75	0.92	0.91
Average	0.95	0.82	0.93	0.90

Table A.2 presents the results for experimental cryo-EM density maps by EMDB ID.

Table A.2: Detection Accuracy for Experimental Cryo-EM Density Maps

EMDB ID/Resolution	CNN only		CNN + GA	
	Sens	*Spec*	*Sens*	*Spec*
1657 (5.8Å)	0.63	0.62	0.63	0.80
1780 (5.5Å)	0.71	0.63	0.71	0.73
1849_L (8.25Å)	0.62	0.72	0.60	0.82
1849_W (8.25Å)	0.63	0.65	0.62	0.73
2169 (8.1Å)	0.79	0.61	0.79	0.71
2605 (5.5Å)	0.68	0.64	0.68	0.83
5036 (6.7Å)	0.80	0.71	0.79	0.84
6396 (6.4Å)	0.93	0.69	0.93	0.87
Average	0.72	0.66	0.72	0.79

Appendix B

GPU PARALLELIZATION PERFORMANCE

Table B.1 and B.2 presents the results of the parallelization using the C++ Accelerated Massive Parallelism library compared to the original CPU based implementation.

Original represents the original CPU based set up using OpenMP and eight CPU threads. *GPU* represents the results using the C++ AMP library on a nVidia GeForce 980 TI. The results shown are displayed in seconds.

Table B.1: Parallelization Results for Simulated Cryo-EM Density Maps

PDB ID	Voxels	*Original*	*GPU*
1AJZ_A	20242	137	36
1AL7_A	26189	184	37
1JB3_A	11808	68	20
1NNX_A	6938	49	20
1TIM_A	16999	127	29
1Y0Y_A	25434	196	56
2DYI_A	12879	83	35
2F01_A	8963	65	26
2VDF_A	16051	104	24
3GP6_A	12820	75	37
3ULJ_A	6849	41	18
Average		102.64	30.73

Table B.2: Parallelization Results for Experimental Cryo-EM Density Maps

EMDB ID(Res)	Voxels	*Original*	*GPU*
1657 (5.8Å)	3720	7	2
1780 (5.5Å)	4570	11	3
1849_L (8.25Å)	2592	4	1.07
1849_W (8.25Å)	2509	4	1.03
2169 (8.1Å)	1800	4	1.10
2605 (5.5Å)	867	2	1.08
5036 (6.7Å)	3524	8	3
6396 (6.4Å)	3051	6	2
Average		5.75	1.77