EMNets: A Convolutional Autoencoder for Protein Surface Retrieval Based on Cryo-Electron Microscopy Imaging

Jingjing Yang

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Computer Science & Software Engineering

University of Washington

2018

Committee:

Dong Si, Chair

Clark Olson

Erika Parsons

Program Authorized to Offer Degree:

Computer Science & Software Engineering

University of Washington

Abstract

EMNets: A Convolutional Autoencoder for Protein Surface Retrieval Based on Cryo-Electron Microscopy Imaging

Jingjing Yang

Chair of the Supervisory Committee:
Dr. Dong Si
Computing and Software Systems

The function of a protein is mainly dependent on its exposed surfaces. The protein surface can provide rich information about a protein's function and evolution. Also, identifying protein surface structure is an essential step in the preliminary stages of drug design.

There are two worldwide databases for the three-dimensional (3D) structural data of biological molecules: Protein Data Bank (PDB) and Electron Microscopy Data Bank (EMDataBank). The 3D structures in PDB are determined by structural biologists using methods such as X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, and cryo-electron microscopy (cryo-EM), which shows the location of each atom relative to each other in the molecule. Unlike the atomic model in PDB, the 3D electron density maps in EMDataBank are determined by cryo-electron microscopy, which provides a general description of the protein surface structure and

placement of the helices. Because the 3D structures in PDB are solved atom by atom, PDB equips with real-time searches based on sequence, structure, and function. Comparing with PDB, searches based on structure similarity in the EMDataBank are far behind. High computational cost and rotation invariance are two main challenges for 3D protein surface similarity retrieval. Usually, a global descriptor is used to represent a 3D object in low dimensionality. However, traditional 3D object descriptors don't completely utilize the spatial information in 3D space.

In order to efficiently investigate protein function and evolutionary history, we introduce a global protein surface shape representation called EMNets descriptors. EMNets provides an effective and accurate way for protein surface representation and similarity search, and thus contributes to biomedical research. The method uses a Convolutional Autoencoder (CAE) neural network to learn the geometric information of 3D density maps in a data-driven manner. Our approach is the first research that applies neural networks to represent and compare global protein surfaces. Our method effectively represents a 3D cryo-electron microscopy density map by using a descriptor, which consists of only 256 numeric variables, called the EMNets descriptor. Based on the EMNets descriptor, we are able to retrieve similar protein surfaces using the K-nearest-neighbor (KNN) strategy in real-time. The search results of protein surface represented with the EMNets descriptor has shown high agreement with the existing Combinatorial Extension (CE) algorithm of sequence and structure similarity search. Overall, EMNets is a powerful tool for comparing 3D protein structures obtained by cryo-electron microscopy. In the future, we can build a real-time retrieval system based on the EMNets descriptors.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would first like to thank my thesis committee chair Dr. Dong Si at the University of Washington Bothell. I would like to express my grateful to him for the continuous support for my study and research, for his patience, enthusiasm, and immense knowledge.

I would also like to thank Albert Ng, Todor Avramov, and all the members in Data Analysis and Intelligent System (DAIS) group. Without their help and input, the research cannot be successfully conducted.

Besides my chair, I would like to thank the rest of my thesis committee: Dr. Clark Olson and Dr. Erika Parsons, for their encouragement, insightful comments. I am gratefully indebted to their valuable comments on this thesis.

Finally, I would like to express my profound gratitude to my parents and family for their unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them.

Chapter 1. INTRODUCTION

Identifying the surface structure of similar proteins provides rich information to researchers for investigating a protein's function and evolution. It is well known that the function of a protein is mainly dependent on its three-dimensional (3D) protein surface [1]. For example, the active site of an enzyme can carry out the catalytic reaction with certain substrate [2]. Also, protein-protein interactions are the physical contacts between two or more protein surfaces. Protein-Protein interactions can predict the unknown function of a protein by observing its interaction with one or more previously characterized proteins [3]. One of the most important applications of investigating protein surface is drug design. Once we know the surface structure of a protein, we are able to design small drug molecules to bind to it and block it functions. Identification of promising protein structures from a large database is an important step in the preliminary stages of drug design [4].

Based on the similarity property principle, similar protein surfaces always lead to similar property or functionality [5][6]. Also, the precise function of a protein depends on its exposed surfaces [7]. Moreover, identifying similar global surface structure can help biochemists and biophysicists trace the evolutionary history of a protein or a superfamily. If the structures of two proteins are similar indicating that the proteins diverged from a common ancestor, these proteins are grouped as a superfamily [8].

There are two worldwide databases for the three-dimensional (3D) structural data of biological molecules: Protein Data Bank[1] (PDB) and Electron Microscopy Data Bank[2] (EMDataBank). The Protein Data Bank (PDB) is a worldwide repository of atomic coordinate protein models, first announced in 1971 [9]. Structural biologists use methods such as X-ray crystallography, nuclear

[1] https://www.rcsb.org
[2] http://www.emdatabank.org

magnetic resonance (NMR) spectroscopy, and cryo-electron microscopy (cryo-EM) to determine the location of each atom relative to each other in the protein [10]. Among the three methods, most of the structures included in the PDB archive were determined using X-ray crystallography [10]. For this method, the protein is crystallized, then subjected to an intense beam of X-rays. X-ray crystallography can provide very detailed atomic information. However, the accuracy of the atomic structure is largely dependent on the quality of the crystallization. Some proteins can be damaged during the crystallization. Also, crystallization may take years for a single protein. Unlike X-ray crystallography, NMR spectroscopy places a protein in a strong magnetic field, and then probed with radio waves. The NMR spectroscopy is the primer method for studying flexible proteins and only limited to small or medium proteins.

Recent advances in rapid freezing technologies allow biochemists to freeze single particle and use cryo-electron microscopy (cryo-EM) to experimentally determine the protein surface structure [11]. Previously, cryo-electron microscopy provides the overall structural information of a protein at a slightly lower resolution. So cryo-EM can act as a complementary of X-ray crystallography. To study a protein structure, structural biologists can use a low-resolution microscopy to provide the overall shape of the molecule. And the smaller sub-components may be solved by X-ray crystallography [12].

With the PDB as the primary data resource, databases such as CATH [13] and Structural Classification of Proteins (SCOP) [14] soon started classifying the PDB data in different ways. CATH groups proteins into superfamily based on a common ancestor. SCOP classifies proteins based on their structural similarity.

Unlike PDB, the 3D structural data in EMDataBank (also referred as EMDB) are all determined by cryo-electron microscopy. Recently, cryo-EM has reached the atomic resolution,

which means now cryo-EM is able to do what X-ray crystallography do, at meanwhile, cryo-EM is faster without damaging the protein structures. More importantly, cryo-EM maintains the proteins in their soluble states. As a result, the 3D structures revealed by cryo-EM are closer to the native state than X-ray crystallography.

The robust growth of the cryo-EM structures in EMDB also reflects the increasing popularity of cryo-EM method. In 2017, over 5500 cryo-EM density maps and 1900 atomic models are released by structural biologist [11]. Comparing with PDB models, cryo-EM density maps released in EMDB are twice as much as atomic models released in PDB.

Further, high and medium-resolution cryo-EM data, which deposited into EMDataBank, has increased dramatically in recent years. A series of research is also conducted based on high and medium resolution cryo-EM density maps such as [15], [16], and [17], to detect the secondary structure based on cryo-EM. However, protein surface retrieval methods have not yet reached high and medium-resolutions.

As we mentioned, the atomic models in PDB have been solved atom by atom using X-ray crystallography and NMR spectroscopy. Unlike those two methods, cryo-EM basically provides the overall protein surface shape. It is easy for the atomic models in PDB to perform a structure-based search based on sequence or structural similarity. However, comparing with the PDB that allowed real-time database search based on sequence, structure, and function, structure similarity search in the EMDB is far behind, especially for medium and high-resolution data.

Currently, the online Electron Microscopy Databank (EMDataBank) contains more than 6,400 electron microscopy density maps. However, only 30% of these have corresponding PDB coordinate entries. In contrast, more than 3,000 protein functions still remain unknown. Nowadays, only with corresponding PDB entries, a cryo-EM density map can perform structure-based search

and similarity retrieval. As a result, it is necessary to perform a structure-based search directly using cryo-EM density maps in EMDB. Due to such urgent need of protein surface analysis, the importance and promise of surface-based protein characterization methods have been highlighted in recent decades [18].

To solve the problem of protein surface comparisons, this document proposes a 3D global protein surface representation, called EMNets, for similarity retrieval and shape reconstruction. Our goal is to develop a method for real-time searching of cryo-EM density maps from EMDB. Our model uses a real 3D Convolutional Autoencoder (CAE) neural network with unsupervised learning. Our approach is the first research that applies neural networks to represent and compare global protein surfaces.

Due to the unstructured nature of protein surfaces, the two main challenges for this research are (1) Computational complexity, (2) Rotation invariance. To adapt the deep learning model from 2D images to 3D objects, we need to first deal with a large number of computing jobs. A 3D volumetric object with low resolution would have the same dimensions as a high-resolution 2D image. For example, a 30×30×30 voxel volume sphere has similar computational complexity as a 165×165 image [19]. Such a large image also requires more parameters to train the model effectively. For the ModelNet10 dataset, pre-processing and training may take more than a week with one CPU [19]. We will discuss how we tackle these problems in the following sections.

This document is organized as follows: Section 2 reviews relevant previous work on 3D object similar retrieval using traditional mathematical methods as well as existing machine-learning based approaches. Section 3 presents a detailed model design and retrieval strategy we took on this research. Section 4 presents the training and testing results under different experiment scenarios. Finally, Section 5 presents the conclusion of our project and discusses the future.

Chapter 2. RELATED WORK

The protein surfaces provide important clues for understanding function, evolution, and interaction of the proteins. Computational methods can effectively use such structure information to identify similarity and dissimilarity in protein's global and local structures. In this section, we will discuss how previous research represent protein surfaces. Before moving to the related work, we will briefly introduce some biological knowledge about protein surface structures that can help to understand the following sections.

2.1 FUNDAMENTALS OF PROTEIN STRUCTURES

A protein is a polypeptide chain made up of amino acid residues linked together in a definite sequence [20]. It is well known that a protein basically contains four elements: carbon, hydrogen, nitrogen, and oxygen. Amino acids consist of a central carbon atom, also known as the alpha carbon (Cα), bonded to an amino group (NH_2), a carboxyl group ($COOH$), and a hydrogen atom [21]. There are 20 types of amino acids commonly found in proteins. With different types of amino acids, the linear protein sequences can vary from about 50 to more than 28,000 amino acid residues [22].

Figure 2.1 Four levels of 3D protein structures. Use PDB 1axc as an example [23].

There are four distinct levels of protein structure: primary, secondary, tertiary and quaternary structure (see Figure 2.1). The primary structure refers to the sequence of amino acids in the polypeptide chain. Secondary structure refers to highly regular local sub-structures on backbone chain. The most common secondary structures are α-helix and ß-sheet. As we can see from Figure 2.1, in tertiary structures, the α-helix and the ß -sheet are folded into a compact global structure. Finally, the quaternary structure consists of two or more tertiary structure. The structures in PDB are basically tertiary structures and quaternary structures. Usually, to find similar proteins, we need to investigate a protein from quaternary to the primary structure. If all four levels are matching, we can tell that the two proteins are similar. Such matching is difficult because a protein sequence contains a large number of amino acids. The computational complexity is high when searching for similar structures in a large database.

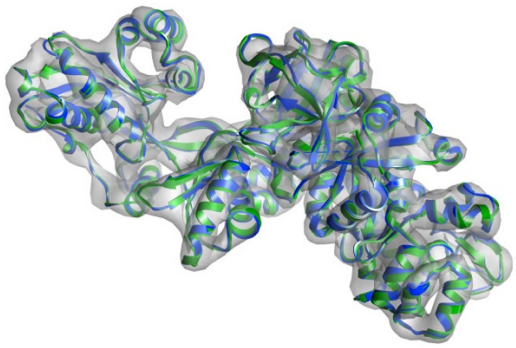


Figure 2.2 Backbone structures vs. molecular surfaces of Elongation factor 2 (EF-2) [24].

The blue and green ribbons in the figure are the backbone structures. The molecular surfaces are shown in gray.

The protein structures represented in PDB are called the backbone structures or backbone chain. With such representation, the proteins can be drawn as a ribbon. Besides, there is another protein

structure representation called molecular surfaces. The Connolly surface is one of the most common ways to represent the molecular surface, which rolls a probe sphere on protein backbone atoms and trace the center of the probes to construct a surface [25]. Apart from the Connolly surface, a molecular surface can also be generated by overlapping the 3D Gaussian function at each atom of proteins [26]. The protein surfaces captured by cryo-EM are more like the molecular surfaces. This explains why the protein structures in PDB and EMDB look quite different as shown in Figure 2.2. The protein surfaces in our research are basically the molecular surfaces simulated by the backbone structures in PDB using a 3D Gaussian function. The details will be explained in the Method section.

2.2 3D OBJECT REPRESENTATION

Currently, more and more researchers focus on the 3D object representation and searching algorithms. Some of the algorithms can be readily applied to protein surface analysis. In this section, we will first discuss classic 3D object representation methods. Also, we will introduce a previous research called EMSurfer for protein surface analysis, which uses one of the classic 3D object representation method. Finally, we will discuss how deep learning methods can help with our problems.

2.2.1 *General Object Comparison*

Usually, a 3D object representation is based on either global or local descriptors [27]. Global descriptors encode the 3D object surface and contain enough information to recognize similar objects from different viewing angles, while local descriptors represent a specific small area of an object. In this document, we will focus on global descriptors of protein surfaces.

Typically, there are two types of methods for general 3D object surface representation: view-based methods and 3D function-based methods. The light-field descriptor (LFD) is the most well-known view-based method. LFD projects a 3D object into 2D images from different viewing angles, capturing 20 distributed silhouettes of the 3D object from 10 rotational positions. A similarity value can be compared from the same viewing angles [28]. LFD first introduces the concept of view-based method and draw the 2D image processing into 3D. LDF also becomes the benchmark model for a lot of later work. However, spatial information between images is not fully captured in this model. Rotation invariance is a big problem to all view-based methods. In order to improve the robustness against rotations, LFD rotates two comparing objects until the highest overall similarity is reached. This solution eliminates the rotation problem but is still a brute force solution.

Unlike view-based methods, 3D function-based methods involve mathematical transformations to represent a 3D shape directly using a 3D function. These mathematical transformations include the 3D version of 2D transformations, such as Fourier, Hough, Radon, and wavelet transformations [18]. Among them, a widely used 3D function is called Spherical Harmonics. The mathematical properties of spherical harmonics make it invariant to Euclidean transformations [29]. Since the Spherical Harmonic Descriptor (SHD) works better on star-like shape or sphere objects, it didn't widely use in general comparison.

Similar to spherical harmonics, the 3D Zernike descriptor (3DZD) is introduced as another 3D function-based method, especially for protein surface retrieval [1]. 3DZD computes the surface distance from the center of an object and incorporates it into the Zernike-Canterakis polynomials. Rotation invariance is achieved by introducing spherical harmonics into Zernike moments. Although the 3DZD of rotated proteins are not exactly the same, the distances between them are

within a certain threshold. Moreover, the resolution also affects the comparison. To solve this problem, 3DZD applies different orders for different resolutions. For inputs with different resolutions, 3DZD normalizes them into the same order. As a result, the relative distance of two inputs will be the same. However, 3DZD only gains better performance in low-resolution data since low-resolution data has lower computational costs.

EMSurfer is a real-time global protein surface retrieval tool based on 3DZD [30]. Currently, EMSurfer contains around 2,000 cryo-EM density maps in their database. With the Zernike-Canterakis polynomial, a 3D protein structure can be represented within 121 numbers, and thus facilitate comparison between two proteins [31]. 3DZD shows good rotation invariance and resolution adjustment. However, EMSurfer can only retrieve low-resolution protein structures.

2.2.2 *Deep Representation*

With the development of machine learning and high-performance computing, the great generative power of deep learning models has allowed researchers to outperform the conventional approaches. There has been a large body of insightful research conducted on 3D objects deep representation. Most of the work utilizes a Convolutional Neural Network (CNN) to extract the features and reduces the dimensionality. One representative work is 3D ShapeNets [19]. 3D ShapeNets use a Convolutional Deep Belief Network (CDBN) to learn a geometric 3D shape and represent a 3D voxel grid by binary variables. For a similar search, 3D ShapeNets use k-nearest-neighbors for matching two objects in a low-resolution voxel space. The experimental results outperform among Light Field Descriptor (LFD) and Spherical Harmonic Descriptor (SHD). However, for the data collected by RGB-D camera, this approach didn’t integrate or make full use of geometric information. 3DShapeNets simply treat the depth channel as an additional channel and use the view-based method for comparison.

A recent project called VoxNet has shown the capability of deep learning on 3D objects classification [32]. VoxNet predicts an object class label using raw volumetric data rather than point clouds from RGB-D. Because volumetric data contains richer information than points clouds, VoxNet results in a more discriminative representation, so-called the real 3D CNN. However, computational complexity increases at the same time. To achieve real-time object recognition performance, this approach only uses a basic CNN architecture to reduce the number of parameters and increase computational efficiency. The model first estimates a spatial occupancy by a volumetric grid. Then a 3D CNN predicts a class label directly from the occupancy grid. This CNN is constructed by three types of layers: two convolution layers, one polling layer, and two fully connected layers. Experimental results show that VoxNet outperforms 3DShapeNets in the most tasks because VoxNet performs real 3D CNN instead of view-based methods. To solve the rotation variance problem, VoxNet introduces a method called rotation augmentation, which uses multiple rotated input and shares the weights across rotations. However, this solution only shows an approximate rotational invariance.

Table 2.1 shows the comparison between all the previous methods we mentioned.

Table 2.1 Comparison between related works

	Type	Key techniques	Test dataset	Shape preference	Invariance	
					Rotation	Resolution
General 3D Object Representation						
LFD	View-based	Projection into 2D	Public 3D models	Low-resolution basic shape	View-based	N/A
SHD	3D function-based	Spherical function	Viewpoint dataset	Basic shape	Concentric spheres	N/A
ShapeNets	Deep learning	CDBN	ModelNet10 ModelNet40	Low-resolution point cloud data	View-based	Low
VoxNet	Deep learning	Basic CNN	ModelNet10	Volumetric data	View-based	N/A
Protein Surface Representation						
EMSurfer	3D function-based	3DZD	Simulated data from PDB	Low-resolution EM density map	Zernike Moments	Low

From Table 2.1 we can see that the rotation invariance of most previous research is based on view-based methods. Also, comparing with the low-resolution EMSurfer, our goal is to represent a medium-resolution cryo-EM density map. The rotation invariance of EMNets is achieved by using deep learning methods.

2.3 Convolutional Autoencoder

Autoencoders are a type of neural network used unsupervised learning. An autoencoder is usually composed of two main parts: encoder and decoder. The encoder maps the input into a hidden layer space called *code*, for the purpose of dimensionality reduction. The decoder then reconstructs the input from the *code* (see Figure 2.3). This *code* is actually a single layer of the neural network with the dimensionality of our choice, which can represent the input. Typically, the decoder architecture is the mirror image of the encoder. The goal of an autoencoder/decoder is to minimize the reconstruction error rate, making the output as close as possible to the input.

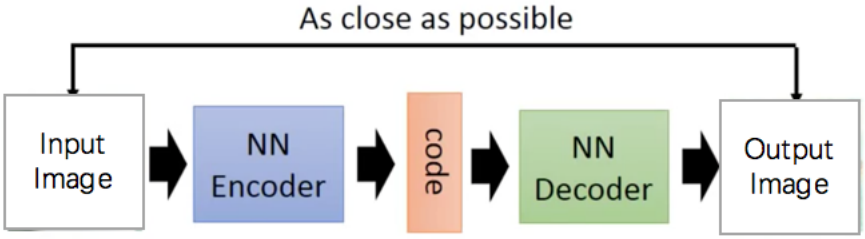


Figure 2.3 Schematic structure of an autoencoder

An autoencoder is usually considered as a lossy data compression algorithm. It can learn automatically from specific types of data. An autoencoder architecture consists of 2 basic steps: encoding step and decoding step (see Figure 2.3). Each step consists of a neural network whose architecture is usually symmetric. Among different types of neural networks, CNNs are widely

used in image-related tasks [33][34][35]. As a result, autoencoder with a convolutional neural network can be a great choice to learn 3D protein surface descriptor.

An autoencoder with the convolutional neural network is usually called convolutional autoencoder (CAE). Usually, a CAE consists of convolutional layers, maxpooling layers, deconvolutional layers and upsampling layers. Unlike previous work, a CAE achieves the rotation and resolution invariance by adding maxpooling layers when encoding inputs. The maxpooling operation helps the higher abstract layer become invariant to small translation and reduces computational cost as well.

Experiments have shown the possibility that CAE can solve the rotation invariance task successfully without using traditional view-based methods. For CAE, it gains invariant features by aggregating multiple low-level features over close neighbors and forms a higher-level normalized abstraction. [36] has demonstrated by comparing to a subsampling operation with a maxpooling operation, shows that the latter is more useful when capturing invariance in image-like data. This is because the small variations of the input image will become smoother by overlapping the pooling windows. Also, [37] has already built a convolutional deep belief network and used a probabilistic maxpooling layer to achieve rotation invariant. As the paper mentioned, encoding the results of a convolutional layer with a maxpooling layer can effectively shrink the representations, reducing computational burden, and thus making the higher layer representation invariant into small rotations [37]. Both experiments help us determine the aggregation strategy, which is: using the maxpooling layer to recognize the small rotation invariant. This data-driven method is able to significantly outperform the traditional view-based methods.

One representative work that involves a CAE is residual deconvolutional networks (RDN) for brain electron microscopy image segmentation [38]. RDN can enlarge the contextual information

in brain electron microscopy image, as well as preserve the pixel-level resolution. They proposed a deconvolution scheme which is similar with decoder and achieved excellent reconstruction results for segmentation.

Chapter 3. METHOD

In this section, we cover three design aspects of our convolutional autoencoder: data pre-processing, convolutional autoencoder model, and a retrieval system (see Figure 3.1).

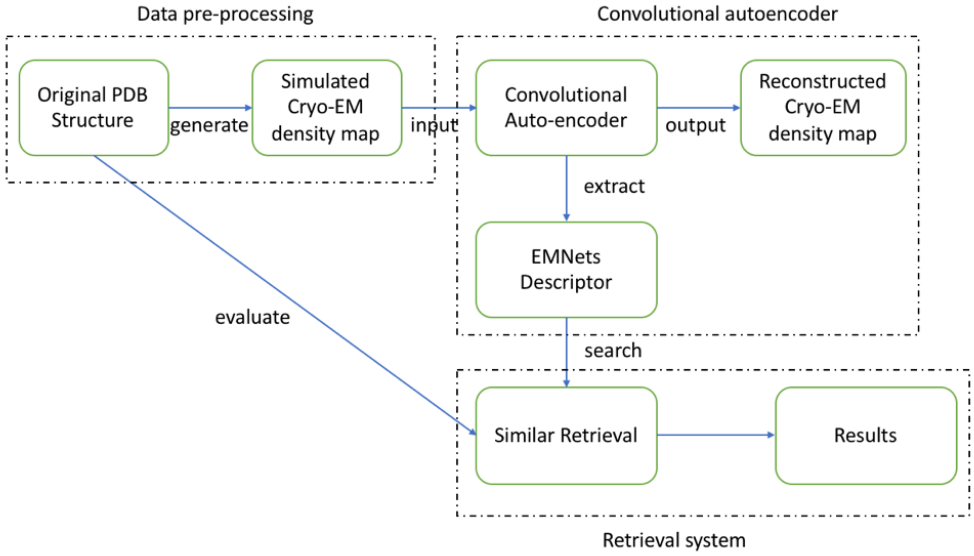


Figure 3.1 Overall design aspects

In the data pre-processing stage, we select atomic models from the PDB and generate the cryo-EM molecular surfaces from the original backbone structures. The simulated cryo-EM density maps then can be sent into the convolutional autoencoder (CAE). The EMNets descriptor is derived from a hidden layer of the CAE, which has low dimensionality comparing with original density maps. An EMNets descriptor can be viewed as a barcode, which contains global shape information of a cryo-EM density map. The main operation performed by retrieval system involves comparing the query cryo-EM density maps with other density maps in the database using EMNets descriptors.

3.1 DATA PRE-PROCESSING

Protein Data Bank (PDB) is a primary database consists of data derived experimentally, for example, X-ray crystallography, NMR spectroscopy and cryo-electron microscopy (cryo-EM).

The PDB is viewed as a key resource for structural biotechnologists and scientists from other areas [39]. Typically, an entry in PDB is represented as .pdb file format. The .pdb file format provides a standard representation for 3D macromolecular structure, where contains information such as xyz atomic coordinate records, atomic connectivity records, identification of residues, and so on [40]. With such format, each data entry is completely defined. Besides, all .pdb files submitted to Protein Data Bank are checked by annotators for accuracy [41].

Apart from completeness and correctness of data, Protein Data Bank can also perform a simple and advanced search based on annotations related to sequence, structure, and function. The advanced search enables query by specific categories such as structure description, number of chains, molecular weight, and secondary structure content. There are two considerations for us to select proteins from PDB: 1) The data entry has to be proteins, not DNA or RNA. 2) The molecular weight of selected data entry is between 10k Dalton (Da) and 20kDa. Molecular weight is defined as the sum of all atomic weights in a molecule. The proteins whose molecular weight range from 14kDa and 24kDa are classified as small molecules [42]. A small molecule has relatively fewer atoms, chains, and ligands comparing with larger proteins. In order to simplify the experiments and reduce the computational complexity, we decide to use small molecules. The PDB currently contains 140, 109 entries including proteins, DNA, RNA, and DNA/RNA hybrid. Among them, 130,068 PDB entries are proteins. According to the search results in May 2018, the count of small proteins is around 15,000 [40].

PDB file format reveals rich information about protein secondary structure such as α-helices and ß -sheets. However, data entries from PDB only contains atoms coordinate. To study protein surface, we need to generate a cryo-electron density map from a .pdb file, which indicates the molecular surface. Such conversion is done by pdb2mrc.py function in EMAN2 library. EMAN2

is a powerful greyscale image processing library primarily focused on cryo-EM density maps [43]. One useful function in EMAN2 is pdb2mrc, which generates a 3D cryo-EM density map from PDB files. The pdb2mrc function uses 1/2 width of Gaussian distribution in Fourier space for each atom [44].

When converting from PDB file to the cryo-EM density map, we need to assign two parameters: 1) The resolution of the density map. 2) The bounding box sizes. The resolution measures the ability of the microscope to resolve two points separated by a given distance. In cryo-electron microscopy, the resolution is defined as the orders of Fourier components available for the Fourier synthesis of the image [45]. Angstrom (Å) is the common unit for resolution. The higher the resolution is, the more details a density map can hold. Recently, cryo-electron microscopy resolution continues to improve, achieving atomic-level resolution. In order to discover the protein similarity in medium resolutions (5Å to 10Å), we convert the .pdb file into 8Å resolution density maps. The format of cryo-EM density maps is MRC file. MRC file format is now an industry standard in cryo-electron density maps, which describes the structure information of a protein as a 3D grid of voxels.

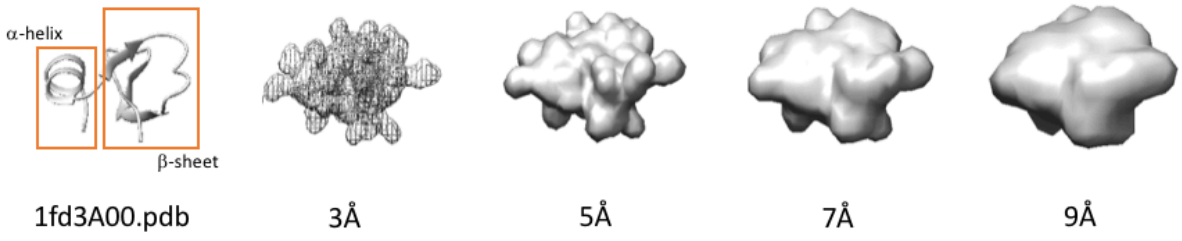


Figure 3.2 PDB structure and density maps of protein 1fd4_ChainA_Domain00. Generated by pdb2mrc function.

Figure 3.2 shows the original PDB structure and simulated density maps generated from 1fd3_A00.pdb. From left to right is 1fd3_A00.pdb, cryo-EM density maps of 1fd3_A00.pdb in 3Å, 5Å, 7Å, 9Å. 1fd3_A00 is the PDB ID in Protein Data Bank. 1fd3 is the unique 4-character ID of the protein structure. It is automatically assigned and does not have meaning. Usually, a protein structure can split into consecutive polypeptide chains. For example, 1fd3 has 2 chains: A and B. 1fd3A00.pdb in Figure 3.2 only shows chain A of 1fd3.pdb. A protein domain is a part of a protein sequence and structure that can evolve. 1fd3.pdb chain A only has 1 domain which numbered as 00. As mentioned before, the density maps of 1fd3_A00.pdb in 3Å, 5Å, 7Å, 9Å is generated by pdb2mrc function in EMAN2 and shown in UCSF Chimera. UCSF Chimera is a software for interactive visualization and analysis of molecular structures including density maps and secondary structure [46]. The visualization of density maps in our paper is mostly done by UCSF Chimera manually.

From Figure 3.2 we can see that 1fd3_A00.pdb contains one α-helix on the left and one ß-sheet on the right. α-helix and ß -sheet are the most common secondary structure of proteins. Protein secondary structure plays an important role in protein classification and folding [47]. α-helix is a right-hand spiral helix, while ß-sheet consists of ß strand connected by hydrogen bonds. We simulate the surface of 1fd3_A00.pdb with different resolutions based on a Gaussian distribution. From the density maps in Figure 3.2, we can see that low-resolution such as 9Å looks more smooth than high resolution such as 3Å, 5Å, which means that low resolution data contains less information than higher resolutions.

Another parameter for the pdb2mrc function, the bounding box size, is set to unify the input size of the CAE. Usually, most deep learning frameworks use the same image size for training and testing. We also choose 64×64×64 as our bounding box size for all density maps. The bounding

box size is defined as the number of voxels in the volume grid. According to our experience, most small proteins can fit in 64×64×64. However, it is possible for some proteins to exceed that size. As a result, we narrow our selection of molecular weight from 14kDa – 24kDa to 10kDa – 20kDa. With lower molecular weight, we can make sure that all the density maps can fit in our bounding box size. It is still possible that a small number of proteins whose molecular weight is between 10kDa - 20kDa cannot fit in 64×64×64 box sizes. However, based on our observation, since out-of-box proteins are fewer than the others, our model is not influenced by those proteins. For larger proteins which cannot fit in 64×64×64 bounding box sizes, we can patch them into 64×64×64 box sizes and train our model in patches. After selection, the total number of small proteins is 15,747. We then split 15,747 density maps into 12,598 training datasets and 3,149 testing datasets.

Instead of real-world cryo-EM density maps such as entries in EMDataBank, we choose simulated cryo-EM density maps. There are two reasons: (1) Real word cryo-EM density map has considerable noise, which can affect the training model. (2) A real-world cryo-EM density map is usually in fixed resolution. However, pdb2mrc is able to convert PDB files into different resolutions. This will help our model adapt different resolutions of input. However, our ultimate goal is to apply our CAE to real-world density maps.

MRC is the standard file format for cryo-EM density maps. A 3D cryo-EM density map contains information such as metadata about 3D volumes, actual volume data, and extended header for symmetry operators for crystallographic applications [48]. However, headers and extended header are not necessary for our CAE. Instead of metadata in MRC files, we only need volume data which contains 3D volumetric information. As a result, we use Mrcfile library to extract and store the volume data from the original MRC files. Mrcfile is a Python library which provides basic MRC I/O functions that can read and write standard MRC files easily [49]. It also supports

inspecting and correcting MRC header files. With Mrcfile library, we are able to extract volumetric shapes from the original MRC files and convert it into 3D arrays. Later on, we can send these 3D arrays as the input to CAE.

Although Mrcfile library can help us read volumetric information, we still need to normalize the 3D arrays. The range of the volume can be varied on different MRC files. It is common to normalize the input data if they are in different ranges. In our data preprocessing, we normalize the data based on the maximum value. For volumes that lower than zero (can be noises), we replace them with zero. For volume higher than 0, we divide them by the maximum value in the 3D array. After normalization, we write all the 3D arrays into an HDF5 file for further training and testing process. An HDF5 file is a file format for storing and managing data. It is flexible and commonly used in machine learning projects. The final input data contains 15,747 density maps. The HDF5 file is around 2GB.

3.2 CONVOLUTIONAL AUTOENCODER MODEL

Autoencoders are a type of artificial neural networks that efficiently represent the input data, which is called *code*, in an unsupervised manner. This *code* usually has lower dimensionality comparing with the input data. As a result, autoencoders act as a powerful data compression and dimensionality reduction algorithm. To better represent the input data, autoencoders use a decoder to reconstruct the encoded inputs with minimal error. In this section, we will explain the architecture of our autoencoder, what types of constraints can be imposed, and how we implement them.

In the design of our model, we intend to achieve two goals: 1) Minimize the reconstruction error between the input cryo-EM density maps and reconstruction density maps. 2) Generate *code* to significantly reduces the input dimensionality. The CAE architecture is mainly inspired by the

ideas in [38]. Similar with the architecture of the residual deconvolutional network in [38], we basically mirror the decoder of the network with the encoder.

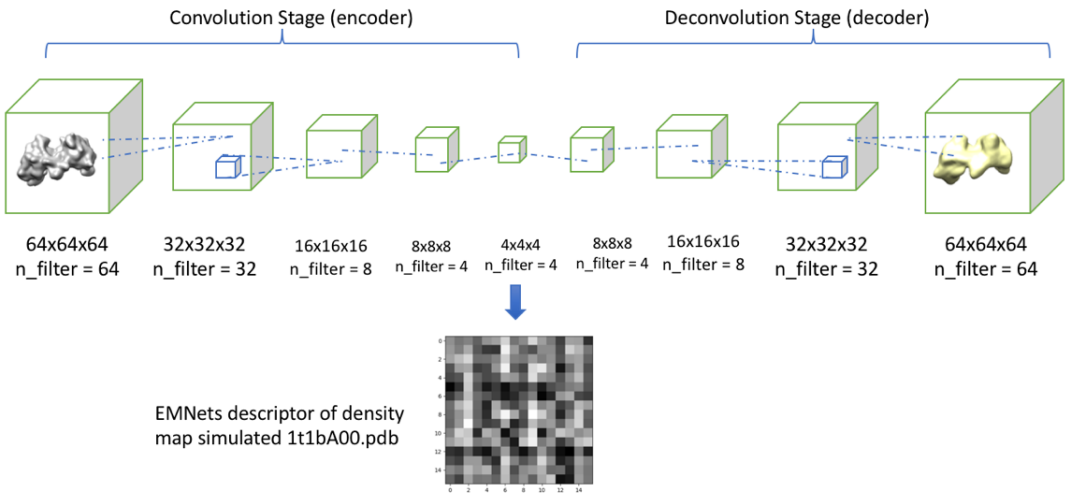


Figure 3.3 Architecture and training process of EMNets CAE. After we extract EMNets descriptor from the encoder, we resize them into 16×16 grayscale images. Here shows an example EMNets descriptor of simulated cryo-EM density map of 1t1b_A00.pdb.

Figure 3.3 shows the architecture and training process of our CAE. As we mentioned previously, our input size is 64×64×64. We define our descriptor size as 4×4×4×4 = 256, which significantly reduces the input dimensionality. Based on our input and descriptor size, we propose the architecture as Figure 3.3 shows. The network contains 4 convolutional layers, 4 maxpooling layers, 4 deconvolutional layers, and 4 upsampling layers. Since our inputs are in 3D, we use 3D convolutional layers to implement spatial convolution over volumes.

The most important layers in our model are the convolutional layers and the maxpooling layers. Typically, a CNN stacks one or more convolutional layers with a pooling layer, then another few convolutional layers, and another pooling layer. A convolutional layer is capable of assembling low-level features in the first hidden layer to the next hidden layer. There are 4 hyperparameters we need to consider when fine-tuning a convolutional layer: strides, padding, filter size, and

activation function. Strides define how the filter convolves along the inputs. We will use the default stride value, which is equal to 1. The padding value, which can be VALID or SAME, decides whether the output will pad zeros around the border. For all the convolutional layers in our model, we use the SAME padding, which means that the output neurons of a hidden layer are equal to the input neurons. Most convolutional neural networks use small filter size to preserve the spatial information such as [34] and [35]. In our experiments, we compare the filter size of 3×3×3 and 5×5×5 using cross-validation. The ratio between the training and validation dataset is 3:1. After cross-validation, we found that 5×5×5 is the best filter size. Finally, we use ReLU (Rectified Linear Unit) as the activation function. The function returns 0 if receives a negative value and returns the value back if the input is a positive value. The constant gradient of ReLU makes it a faster learner. Nowadays, ReLU function is the most widely used activation function in the convolutional neural networks because it can reduce the likelihood of vanishing gradient.

The goal of the maxpooling layer is to subsample the input image and reduce the computational complexity and memory usage. The maxpooling layer can also enhance the robustness of a model to the rotation. Similarly, a upsampling layer is to upsample the input image to restore the original input size. In our model, we will use 2×2×2 kernel, which is the most common kernel size of the maxpooling and upsampling layers.

Convolutional neural networks require a huge amount of memory and RAM, especially during training. For forward propagation, typically you need all the intermediate values of the last layer of activations, and backpropagation takes almost twice the memory than forward propagation. For example, assuming that the input size is 64×64×64, and the first hidden layer outputs 128 feature maps of size 64×64×64, and if the feature map is represented using 32-bit floats, then the convolutional layer's output occupies $128 \times 64 \times 64 \times 64 \times 32 = 134$MB of RAM. If a training

batch contains 100 instances, then this layer will use over 12GB of RAM. Also, every parameter computed during the forward propagation needs to be preserved for the backpropagation, so the amount of RAM needs to be the total amount of RAM required for all layers. To implement our model, we use NVIDIA TITAN X (Pascal) GPU with 12189MB memory. In order to avoid out-of-memory errors, we use 25 as the mini-batch size based on the GPU memory.

One of the most important hyperparameters for the autoencoder is the loss function. The loss function measures how close the reconstructed output is comparing with the original input. There are two loss functions typically used in autoencoder: mean square error and binary cross-entropy. Since we normalize our input cryo-EM density maps, we use binary cross-entropy as our loss function. For a model $f(x) \equiv \hat{x}$, where x is the input, $\hat{x}$ is the predicted value, the binary cross-entropy defined on the instance of input x_k and prediction $\widehat{x_k}$ is given by:

$$l\big(f(x)\big) = -\sum_k (x_k \log(\widehat{x_k}) + (1 - x_k)\log\,(1 - \widehat{x_k})) \tag{3.1}$$

where l is the loss function of model $f(x)$, defined as the sum of Bernoulli cross-entropy.

Our model implementation is based on the publicly available python-based Tensorflow [50] and Keras [51]. We train our CAE using backpropagation with adaptive learning rate method. Unlike classic stochastic gradient descent, adaptive learning rate method dynamically adapts a per-parameter learning rate. The mini-batch size is 25 as the 3D inputs and filters require a lot of memory. To find the best epochs, we train our model on 100, 200, 500 epochs and visualize the reconstruction density maps. Random initialization is used for all layers. The experiments were carried out on an NVIDIA TITAN X GPU machine. Each epoch takes roughly 12 minutes of training and validation[3].

[3] source code available at https://github.com/kdj842969/EMNets

After training and validating, we save the pre-trained model and test on testing datasets. Finally, we compute the EMNets descriptor by sending full datasets back to the pre-trained model and extract the last layer of encoder whose size is 256. Apart from calculating the EMNets descriptor, we also save the reconstructed cryo-EM density maps to analyze the results visually. No post-processing is involved in the reconstruction results.

Inspired by 3D ShapeNets, we use the k-nearest-neighbors (KNN) to match our EMNets descriptor. We retrieve the top 8 similar EMNets descriptors based on Euclidean distance with the encoded size of 256. To implement KNN, we calculate the pair-wise Euclidean distance of their EMNets descriptors. We first send all 15,747 cryo-EM density maps into the pre-trained CAE. Then we extract the hidden layer whose size is $4 \times 4 \times 4 \times 4 = 256$ from the CAE, as the EMNets descriptors. We write all the 15,747 EMNets descriptors into the HDF5 file for reuse. For each query, we calculate the Euclidean distance between its EMNets descriptor and all the other 15,746 EMNets descriptors in the HDF5 file. The time complexity is $O(N^2)$ where N is the full size of our dataset. Next time if we search for a density map which is not in our dataset, the time complexity will be $O(N)$.

After calculating the pair-wise distance, we keep the top 8 retrieval results with minimum Euclidean distance. All the retrieval results are sorted by the Euclidean distance between the query and retrieval EMNets descriptors. To evaluate our database retrieval performance, we also measure the speed of search. This evaluation is performed on a computer with Intel Core i5 (2.7 GHz) processor with 8GB memory. One single query using KNN with EMNets descriptors takes only 360ms, which means EMNets is capable of real-time search.

Chapter 4. RESULT

To evaluate our model, our experiments focus on the following results: (1) Reconstruction results, training and validation loss of CAE; (2) Top K retrieval results based on KNN. The following section is organized based on these two focuses.

4.1 RECONSTRUCTION RESULTS

In order to find proper iterations for training, we train our model with 100, 200, and 500 epochs. We compare the original density maps with reconstructed density maps. Apart from the visual comparison, we also align two density maps and calculate the correlation between them. This alignment is finished by using EMAN2 *fit in map* function [43]. This function basically aligns two density maps to the position where they have maximum overlap. The overlap is defined as the inner product of two grid points:

$$overlap = \langle u, v \rangle \tag{4.1}$$

The correlation between two density maps reflects the cosine similarity between the two density maps, which is defined as:

$$correlation = \frac{\langle u,v \rangle}{|u||v|} \tag{4.2}$$

where u and v are the original and reconstructed vector of density maps. Based on the definition of cosine similarity, the correlation value of a map with itself should be equal to 1. As a result, a higher correlation between two density maps indicates higher similarity.

Table 4.1 Reconstruction results for 100, 200, and 500 epochs.

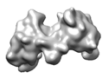
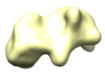
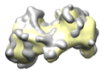
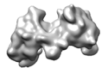
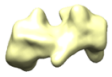
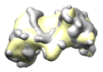
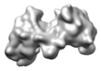
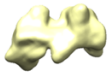
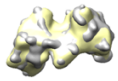
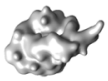
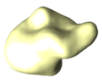
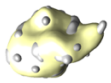
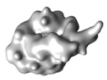
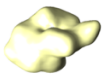
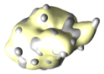
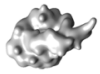
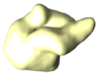
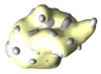
PDB ID	Epochs	Simulated	Reconstructed	Overlapping	correlation
3wkl_A00.pdb	100				0.967
	200				0.9672
	500				0.9817
2eef_A00.pdb	100				0.9603
	200				0.9723
	500				0.9811

Table 4.1 shows two examples of reconstruction results for 100, 200, and 500 epochs. We randomly choose two density maps simulated from 3wkl_A00.pdb and 2eef_A00.pdb. Also, we overlap the original density maps with reconstructed density maps in order to compare their shapes and calculate the correlation. Although the difference between 200 epochs and 500 epochs reconstructed density maps cannot be viewed clearly from Table 4.1, we can tell that 500 epochs are better than the other two epochs, because 500 epochs have the highest correlation score among all the three epochs. However, as we can see from the 500 epochs overlapping density map, although the reconstructed density maps capture the rough shape of the original density maps, some details cannot be restored completely.

All cryo-EM density maps in this section are displayed by UCSF Chimera [46]. One important display option in UCSF Chimera is the threshold. For protein surface visualization, a threshold is

a single number that limits the density displays in a zone. For density values lower than the threshold, UCSF Chimera won't display them on the screen. Thresholding acts like a filter that filter out the small noises outside the protein surfaces. Initial thresholds are set automatically by UCSF Chimera when displaying the cryo-EM density maps. Neither training nor testing of our model has used thresholding, for two reasons: (1) Automatic thresholding is a big issue in cryo-EM density maps. Different cryo-EM density map has a different optimal threshold. Currently, there is no any approach to automatically select the best threshold for a cryo-EM density map. (2) Since we have done normalization to our inputs, the small noises outside the protein surfaces won't affect our model. The experiment results also prove that threshold is unnecessary in the data pre-processing. In Table 4.2, the original simulated cryo-EM density maps are displayed using their initial thresholding. For the reconstructed cryo-EM density maps, we adjust their thresholds the same as their simulated cryo-EM density map in order to compare them in the same conditions.

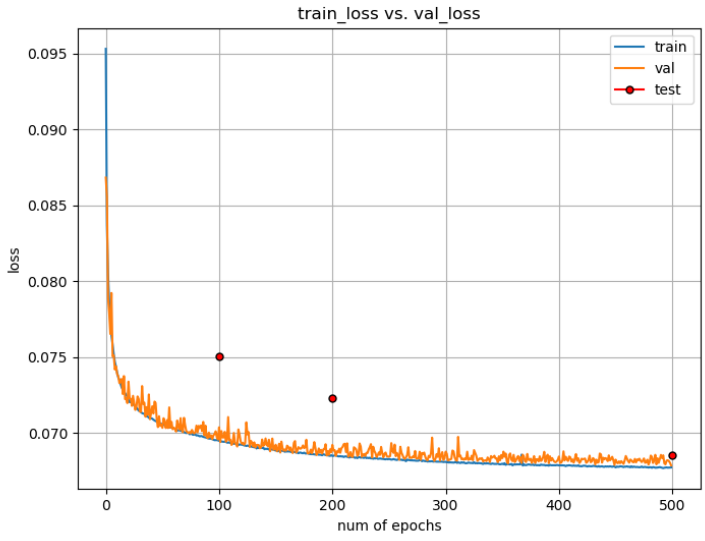


Figure 4.1 Training loss versus validation loss for 500 epochs.

Figure 4.1 shows the training loss on 9,448 training datasets versus validation loss on 3,150 validation datasets. As we can see, the training loss and validation loss basically follow the same

tendency. Both loss curves reduce rapidly at the beginning and become steady after 200 epochs. The orange plot of validation loss curve is slightly higher and unstable compared with the blue training loss curve, which means our model is in a good fit. The training loss at 500 epochs is 0.0677 while the validation loss is 0.0679. We also calculate the testing loss for 3,149 testing datasets which is 0.0685. We can see that our model performs well, since the difference between training and testing loss is only 0.0008.

As we mentioned in the Method section, each epoch usually takes 12 minutes. For 500 epochs, the training process will take 100 hours. We also mark the testing loss for 100, 200, and 500 epochs on Figure 4.1 using red dots. Based on our experiments, only after 500 epochs, the testing loss can be under 0.070. Also, comparing the difference between the training and testing loss, only 500 epochs have the difference under 0.001. Here we use the testing loss of 0.070 as a threshold to select the best epochs. Obviously more epochs will give better results. But more epochs also mean a longer trained time. As a result, we suggest using fewer epochs such as 300 epochs in practice.

Table 4.2 shows the reconstruction results of 15 cryo-EM density maps. We randomly select 15 reconstruction results from the test datasets and calculate their correlation with the original density maps. Apart from the correlation between two density maps, we also use the Hausdorff distance to determine the structural similarity. The Hausdorff distance is commonly used in computer vision and protein structure comparison. Instead of forming a one-to-one mapping like cosine similarity, the Hausdorff distance builds a many-to-many correspondence between two density maps by taking the max-min Euclidean distance of all relative position [52]. Given two point sets $A = \{a_1,\ a_2, \cdots, a_n\}$ and $B = \{b_1,\ b_2, \cdots, b_m\}$, the one-sided Hausdorff distance from A to B is defined as:

$$\tilde{\delta}_H(A,B) = max_{a\in A} min_{b\in B} \|a-b\| \quad (4.3)$$

From equation (4.3) we can tell that if two sets are in the small Hausdorff distance, they are supposed to be more similar.

Table 4.2 Reconstruction results of 15 cryo-EM density maps.

PDB ID	Correlation	Hausdorff distance (Å)
2eef_A00	0.9495	7.1177206
1uph_A00	0.9635	4.9343534
3lpy_B00	0.9674	5.4190745
1wy9_A00	0.9688	4.226468
1fu5_A00	0.9719	4.1141
1coo_A00	0.9729	3.380761
4nr6_A00	0.9751	3.694622
2frk_X00	0.9779	3.937251
4rsd_A00	0.978	3.1533382
4nhs_A00	0.9829	3.138124
1mlf_A00	0.9919	2.3076963
1laa_A00	0.9928	2.5207515
Average	**0.9745**	**3.9450732**
Std	**0.01069**	**1.22487**

Comparing with correlation, Hausdorff distance takes translation and rotation into consideration. Also, Hausdorff distance is more sensitive to small perturbations on 3D surfaces. As we can see from Table 4.2, the standard deviation of the Hausdorff distance is 1.22487. While correlation has a smaller standard deviation value which is 0.0106. We also sort Table 4.2 based on the correlation. The experiments show that Hausdorff distance is basically inversely in proportional to the correlation. The lowest Hausdorff distance is 2.5207515 for 1laa_A00.pdb. This density map also has the largest correlation 0.9928, which means that the reconstructed density map of 1laa_A00.pdb is more similar to its original density map comparing with all the other 14 density maps.

4.2 RETRIEVAL RESULTS

In order to evaluate our retrieval results, we randomly select 1 density maps from the full datasets and display its retrieval results. Table 4.3 shows the retrieval results of density map simulated by 1stp_A00.pdb:

Table 4.3 EMNets retrieval results of 1stp_A00.pdb

Query PDB ID: PDB 1stp_A00									
Query simulated density map					**Corresponding EMNets Descriptor**				
Top 8 Retrieval Results									
Simulated density map	1 	2 	3 	4 	5 	6 	7 	8 	
EMNets Descriptor									
PDB ID	3pk2_A00	4oka_A00	2iza_A00	1n4j_A00	2izj_A00	1vwg_B00	1vwb_B00	3r5p_A00	**Average**
Euc distance	0.1899	0.1899	0.4355	0.4578	0.5338	0.5347	0.5347	0.5981	**0.4343**
Correlation	0.9954	0.9955	0.9905	0.9893	0.9906	0.9695	0.9689	0.8719	**0.97145**
Z-score	6.7	6.7	6.7	6.4	6.7	6.6	6.6	2.6	**6.125**
Sequence identity	98.3%	98.3%	100%	94.9%	100%	92.8%	91.3%	2.7%	**84.8%**

From Table 4.3 we can see that the top 8 retrieval results of 1stp_A00.pdb are visually similar with the query density maps. The query and retrieval density maps we show here are the original cryo-EM density maps directly convert from PDB using the pdb2mrc function. We sort the retrieval results based on the Euclidean distance between the query and retrieval EMNets descriptor. Besides, we calculate the correlation between the top 8 retrieval results and query density maps using the UCSF Chimera *fit in map* function. Although some retrieval results seem to be different from the query density maps such as 3r5p_A00.pdb, the overall performance of our retrieval algorithm shows that EMNets descriptor is capable of representing the global surface of a cryo-EM density map and contains enough information for similar retrieval. For retrieval density map

such as 3r5p_A00.pdb, although it looks less similar comparing with other retrieval results, its EMNets descriptor has similar distribution with the query descriptor.

It is often difficult to quantitatively evaluate the cryo-EM density map retrieval results. In our experiments, we basically consider two evaluation schemes: Scheme 1 is the correlation, which indicates the cosine similarity between two 3D objects statistically; Scheme 2 is the z-score and sequence identity calculated by combinatorial extension (CE) algorithm. CE is a method for calculating PDB pairwise structure alignments. It typically aligns two PDB polypeptide chains based on Cα positions [53]. The z-score of CE is calculated by the root mean square deviation (RMSD) and the number of gaps in the final alignment, which reflects how good the alignment is. Typically, proteins within a similar folder will have a z-score of 3.5 or above. Also, sequence identity directly calculates the similarity between two polypeptide sequences after alignment. Since we use simulated density maps, CE is a good quantitative evaluation based on protein sequence.

From Table 4.3 we can see that the top 7 retrieval results of 1stp_A00 whose z-score is around 6.6. Their sequence identity also shows that the top 7 retrieval results have over 90% similarity with the query density map 1stp_A00. The average z-score of the top 8 retrieval results is 6.125 and the average sequence identity is 84.8%. This quantitative evaluation also shows agreement with the correlation results in Table 4.3.

Table 4.4 shows five more queries randomly selected from 15,747 density maps. We search for the top 5 similar proteins for each query.

Table 4.4 Retrieval results of 5 queries based on EMNets descriptor

PDB		Retrieved PDB	Euc distance	Correlation	Z-score	Sequence identity
	1	1b4l_A00	0.09629816	0.9983	7	99.30%
	2	1f1a_A00	0.11389227	0.9986	7	99.30%
	3	2jcw_A00	0.1192864	0.9982	7	99.30%
1b4t_A00	4	1jcv_A00	0.14460532	0.9977	7	99.30%
	5	1f18_A00	0.18348444	0.9964	6.9	98.70%
	Average		**0.131513318**	**0.99784**	**6.98**	**99.18%**
	Std		**0.033810147**	**0.000867756**	**0.04472136**	**0.002683282**
	1	4dp5_X00	0.07909537	0.9975	6.4	100.00%
	2	4dp2_X00	0.08746006	0.9969	6.4	100.00%
	3	4dp4_X00	0.13216056	0.994	6.4	100.00%
4dp6_X00	4	4dp0_X00	0.3181451	0.9927	6.4	100.00%
	5	2dib_A00	0.56182781	0.8367	4.1	11.90%
	Average		**0.23573778**	**0.96356**	**5.94**	**82.38%**
	Std		**0.20638377**	**0.070944824**	**1.02859127**	**0.393995178**
	1	5cxa_A00	0.18794472	0.9903	7	100.00%
	2	3lir_A00	0.20026855	0.9845	7	99.40%
	3	3lik_A00	0.20197138	0.9874	7	99.40%
5d3c_A00	4	3tsk_A00	0.26549518	0.9837	6.9	99.40%
	5	3lil_A00	0.27763169	0.9878	7	99.40%
	Average		**0.226662304**	**0.98674**	**6.98**	**99.52%**
	Std		**0.041566795**	**0.002668895**	**0.04472136**	**0.002683282**
	1	1dxc_A00	0.11607486	0.9984	7	100.00%
	2	1naz_A00	0.18647471	0.9945	7	100.00%
	3	2w6x_A00	0.22123794	0.9947	7	98.70%
1dxd_A00	4	2w6w_A00	0.35647606	0.9935	7	97.40%
	5	1do1_A00	0.42092853	0.9961	7	98.00%
	Average		**0.26023842**	**0.99544**	**7**	**98.82%**
	Std		**0.125327819**	**0.001896839**	**0**	**0.01171324**
	1	3pk2_A00	0.18988845	0.9954	6.7	98.30%
	2	4oka_A00	0.18988845	0.9955	6.7	98.30%
	3	2iza_A00	0.43547087	0.9905	6.7	100.00%
1stp_A00	4	1n4j_A00	0.45782284	0.9893	6.4	94.90%
	5	2izj_A00	0.53377693	0.9906	6.7	100.00%
	Average		**0.36136951**	**0.99226**	**6.64**	**98.30%**
	Std		**0.16072498**	**0.00295686**	**0.13416408**	**0.02082066**
Summary	**Average**		**0.24310427**	**0.987168**	**6.708**	**95.64%**

The retrieved PDB ID in Table 4.4 is sorted by Euclidean distance between query and retrieval EMNets descriptor. As we can see from each query in Table 4.4, the top 3 retrieved density maps in each query always have a high sequence identity over 98%, which means that their structures are almost the same with every query density map. Smaller Euclidean distances imply higher correlation, higher z-score, and higher sequence identity. However, some retrieval results are not good enough, such as the 5th retrieval result of 4dp6_X00.pdb. Although the z-score of 2dib_A00.pdb is 4.1 > 3.5, which means it is in the same folder with 4dp6_X00.pdb, the sequence

identity between 2dib_A00.pdb and 4dp6_X00.pdb is only 11.9%. The reason for this deviation is that autoencoder is a lossy compression algorithm which uses inexact approximations and partial data discarding to represent the original input. Potential data loss cannot be fully avoided when constructing a 256 EMNets descriptor from a 64×64×64 density map.

Chapter 5. CONCLUSION

In this document, we utilize EMNets descriptors for the purpose of shape-based retrieval of 3D protein surfaces. We discuss previous general 3D object shape descriptors such as LFD and SHD. Also, 2D image retrieval using autoencoder is introduced in related work. Furthermore, we consider the implementation issues: the high computational complexity and rotation invariance. As a cure to these two problems, we apply CAE to compress the protein structure and keep the 3D shape information. Our experiment results show that EMNets descriptors are capable of retrieve similar protein surfaces.

There are three contributions of our work: 1) In our model, we only use the 3D surface shape. There is no protein sequence information involved in EMNets descriptors. 2) We fully utilize the 3D shape information using the 3D convolutional layer. Previous 3D shape comparison mostly used view-based methods or mathematical models. Though their input data is 3D, they are actually 2D or 2.5D. Our work learns the 3D structure using an unsupervised learning manner in a data-driven way and keeps the space information as much as possible. 3) Finally, we propose a novel approach to retrieval 3D cryo-EM density maps that significantly reduced the computational complexity of protein retrieval.

Comparing with the EMSurfer, our EMNets descriptor has the following advantages: First, since the EMNets descriptor works on medium-resolution volumetric data, its input contains more valuable details than low-resolution data in the EMSurfer database. Second, the use of a CAE is a state-of-art approach that effectively solves the rotation invariance problem than the conventional view-based methods. Moreover, the deep learning algorithm makes it possible to input real cryo-EM density maps and produce reasonable results without any prior knowledge of the data.

There are two reasons we could not perform any quantitative comparison between our model and the EMSurfer. First, the experiments conducted in the EMSurfer paper only contains 2,337 cryo-EM density maps [30]. However, we have 15,747 density maps, which is larger than EMSurfer. Second, the retrieval performance of EMSurfer is based on four different backbone surface representations, which was generated using different kinds of heavy atoms such as Cα, C, N, and O atoms. For example, they assume that the isosurface of low resolution cryo-EM density maps resembles the CACNO representation. Comparing with real-world cryo-EM density maps, those backbone surface representations have better agreement with CE algorithm. However, our model learns directly from cryo-EM density maps instead of backbone surface representation. As a result, EMNets descriptor is incomparable with the EMSurfer.

Although our method achieves good retrieval results visually and quantitatively, it still needs to be improved. First of all, we didn't solve the resolution variation problem completely. Currently, we build separate models for each resolution. However, real-world cryo-EM density maps are in different resolutions. As a result, it is necessary to combine different models and achieve resolution invariance in the next step. Second, when searching for similar proteins in our datasets, we need to pad the input density maps to the same size. However, such padding is not applicable for the larger proteins whose size exceed the bounding box size. Larger proteins have more complex shapes and structures. As a result, it is harder for our model to learn larger proteins with more chains or ligands. For the next step, we can patch the large proteins into smaller box size and thus train the model with patches.

Our method also has two limitations. First, although our method uses deep learning and high-performance computing to accelerate the training process, our method is more computationally expensive than the traditional algorithms such as the LFD and SHD. As we mentioned, to better

reconstruct the original inputs, we should use at least 500 epochs. And 500 epochs may take 100 hours (almost 5 days) to train. However, once we save our pre-trained model, the query can be in real-time. Second, our method only utilizes the shape information of the cryo-EM density maps. Apart from the shape information, a cryo-EM density map also contains metadata such as the evolutionary relationship and resolution. That metadata can also indicate the protein similarity. However, the metadata cannot be preserved during the data pre-processing stage.

Nevertheless, EMNets can still be utilized in many areas. First of all, EMNets can assist biologists to discover the evolution of a protein and help biochemists with drug design. Second, EMNets can be a powerful tool for protein comparison in just a few seconds. Our next step is to use real-world density maps to train and test the model. Since an autoencoder is good at image denoising, it is promising to use our model architecture on noisy real-world inputs. Finally, we can apply a very deep autoencoder on our data to solve the resolution variation problem. In the further future, we can build a database and retrieval system based on the EMNets descriptors.

Apart from the protein surface similar retrieval, the idea of using a CAE as data compression algorithm can also be generalized in general 3D objects comparison. Comparing with cryo-EM density maps which have barrels and helices on their surfaces, a general 3D object contains fewer details. The architecture of our CAE is ready to be utilized on general 3D objects with proper input sizes. A preliminary demo[4] has been presented using 3D ShapeNets dataset [19]. Our experiment shows that the 3D CAE successfully reconstructs the original input with fine details. In the future, the CAE can definitely be used in general 3D objects classification and similar retrieval.

[4] The source code of this demo is available at https://github.com/kdj842969/3D-Autoencoder

BIBLIOGRAPHY

[1] L. Sael *et al.*, "Fast protein tertiary structure retrieval based on global surface shape similarity," *Proteins Struct. Funct. Bioinforma.*, vol. 72, no. 4, pp. 1259–1273, 2008.

[2] G. M. Cooper, "The central role of enzymes as biological catalysts," Sinauer Associates, 2000.

[3] E. C. Reeve, *Encyclopedia of genetics*. Routledge, 2014.

[4] V. Venkatraman, P. R. Chakravarthy, and D. Kihara, "Application of 3D Zernike descriptors to shape-based ligand similarity searching," *J. Cheminformatics*, vol. 1, no. 1, p. 19, 2009.

[5] A. Bender and R. C. Glen, "Molecular similarity: a key technique in molecular informatics," *Org. Biomol. Chem.*, vol. 2, no. 22, pp. 3204–3218, 2004.

[6] J. B. Hendrickson, "Concepts and Applications of Molecular Similarity," *Science*, vol. 252, no. 5009, pp. 1189–1190, 1991.

[7] C. M. O'Connor, J. U. Adams, and J. Fairman, "Essentials of cell biology," *Camb. NPG Educ.*, 2010.

[8] C. Pál, B. Papp, and M. J. Lercher, "An integrated view of protein evolution," *Nat. Rev. Genet.*, vol. 7, no. 5, p. 337, 2006.

[9] E. E. Abola, F. C. Bernstein, and T. F. Koetzle, "The protein data bank," in *Neutrons in Biology*, Springer, 1984, pp. 441–441.

[10] "PDB101: Learn: Guide to Understanding PDB Data: Methods for Determining Structure," *RCSB: PDB-101*. [Online]. Available: https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/methods-for-determining-structure. [Accessed: 30-Jul-2018].

[11] M. Tagari, R. Newman, M. Chagoyen, J.-M. Carazo, and K. Henrick, "New electron microscopy database and deposition system," *Trends Biochem. Sci.*, vol. 27, no. 11, p. 589, 2002.

[12] H.-W. Wang and J.-W. Wang, "How cryo-electron microscopy and X-ray crystallography complement each other," *Protein Sci.*, vol. 26, no. 1, pp. 32–39, 2017.

[13] F. M. G. Pearl *et al.*, "The CATH database: an extended protein family resource for structural and functional genomics," *Nucleic Acids Res.*, vol. 31, no. 1, pp. 452–455, 2003.

[14] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "SCOP: a structural classification of proteins database for the investigation of sequences and structures," *J. Mol. Biol.*, vol. 247, no. 4, pp. 536–540, 1995.

[15] A. Ng and D. Si, "Beta-Barrel Detection for Medium Resolution Cryo-Electron Microscopy Density Maps Using Genetic Algorithms and Ray Tracing," *J. Comput. Biol.*, vol. 25, no. 3, pp. 326–336, 2018.

[16] P. Collins and D. Si, "A Graph Based Method for the Prediction of Backbone Trace from Cryo-EM Density Maps," in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, 2017, pp. 691–697.

[17] D. Si and J. He, "Modeling beta-traces for beta-barrels from cryo-EM density maps," *BioMed Res. Int.*, vol. 2017, 2017.

[18] L. Sael and D. Kihara, "Protein surface representation and comparison: New approaches in structural proteomics," *Biol. Data Min.*, p. 2009, 2009.

[19] Z. Wu *et al.*, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[20] J. S. Richardson, “The anatomy and taxonomy of protein structure,” in *Advances in protein chemistry*, vol. 34, Elsevier, 1981, pp. 167–339.
[21] “Introduction to proteins and amino acids,” *Khan Academy*. [Online]. Available: https://www.khanacademy.org/science/biology/macromolecules/proteins-and-amino-acids/a/introduction-to-proteins-and-amino-acids. [Accessed: 03-Aug-2018].
[22] Y. Pan, J. Wang, and M. Li, *Algorithmic and artificial intelligence methods for protein bioinformatics*, vol. 22. John Wiley & Sons, 2013.
[23] “Protein structure,” *Wikipedia*. 07-Jul-2018.
[24] G. F. Schröder, A. T. Brunger, and M. Levitt, “Combining efficient conformational sampling with a deformable elastic network model facilitates structure refinement at low resolution,” *Structure*, vol. 15, no. 12, pp. 1630–1641, 2007.
[25] D. Kihara, L. Sael, R. Chikhi, and J. Esquivel-Rodriguez, “Molecular surface representation using 3D Zernike descriptors for protein shape comparison and docking,” *Curr. Protein Pept. Sci.*, vol. 12, no. 6, pp. 520–530, 2011.
[26] J. C. Mitchell, R. Kerr, and L. F. Ten Eyck, “Rapid atomic density methods for molecular shape characterization1,” *J. Mol. Graph. Model.*, vol. 19, no. 3–4, pp. 325–330, 2001.
[27] S. H. Kasaei, A. M. Tomé, L. S. Lopes, and M. Oliveira, “GOOD: A global orthographic object descriptor for 3D object recognition and manipulation,” *Pattern Recognit. Lett.*, vol. 83, pp. 312–320, 2016.
[28] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, “On visual similarity based 3D model retrieval,” in *Computer graphics forum*, 2003, vol. 22, pp. 223–232.
[29] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, “Rotation invariant spherical harmonic representation of 3 d shape descriptors,” in *Symposium on geometry processing*, 2003, vol. 6, pp. 156–164.
[30] J. Esquivel-Rodríguez, Y. Xiong, X. Han, S. Guang, C. Christoffer, and D. Kihara, “Navigating 3D electron microscopy maps with EM-SURFER,” *BMC Bioinformatics*, vol. 16, no. 1, p. 181, 2015.
[31] X. Han, Q. Wei, and D. Kihara, “Protein 3D Structure and Electron Microscopy Map Retrieval Using 3D-SURFER2. 0 and EM-SURFER,” *Curr. Protoc. Bioinforma.*, pp. 3–14, 2017.
[32] D. Maturana and S. Scherer, “Voxnet: A 3d convolutional neural network for real-time object recognition,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 2015, pp. 922–928.
[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[34] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *ArXiv Prepr. ArXiv14091556*, 2014.
[35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[36] D. Scherer, A. Müller, and S. Behnke, “Evaluation of pooling operations in convolutional architectures for object recognition,” in *Artificial Neural Networks–ICANN 2010*, Springer, 2010, pp. 92–101.

[37] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 609–616.
[38] A. Fakhry, T. Zeng, and S. Ji, "Residual deconvolutional networks for brain electron microscopy image segmentation," *IEEE Trans. Med. Imaging*, vol. 36, no. 2, pp. 447–456, 2017.
[39] A. Bagchi, "A Brief Overview of a Few Popular and Important Protein Databases," *Comput. Mol. Biosci.*, vol. 2, no. 04, p. 115, 2012.
[40] P. D. Bank, "Protein Data Bank," *Nat. New Biol*, vol. 233, p. 223, 1971.
[41] H. M. Berman, "The protein data bank: a historical perspective," *Acta Crystallogr. A*, vol. 64, no. 1, pp. 88–95, 2008.
[42] M. Barber, B. N. Green, and K. R. Jennings, "The analysis of small proteins in the molecular weight range 10–24 kDa by magnetic sector mass spectrometry," *Rapid Commun. Mass Spectrom.*, vol. 1, no. 5, pp. 80–83, 1987.
[43] G. Tang *et al.*, "EMAN2: an extensible image processing suite for electron microscopy," *J. Struct. Biol.*, vol. 157, no. 1, pp. 38–46, 2007.
[44] S. J. Ludtke, P. R. Baldwin, and W. Chiu, "EMAN: semiautomated software for high-resolution single-particle reconstructions," *J. Struct. Biol.*, vol. 128, no. 1, pp. 82–97, 1999.
[45] J. Frank, *Three-dimensional electron microscopy of macromolecular assemblies: visualization of biological molecules in their native state*. Oxford University Press, 2006.
[46] E. F. Pettersen *et al.*, "UCSF Chimera—a visualization system for exploratory research and analysis," *J. Comput. Chem.*, vol. 25, no. 13, pp. 1605–1612, 2004.
[47] Y.-Y. Ji and Y.-Q. Li, "The role of secondary structure in protein structure selection," *Eur. Phys. J. E*, vol. 32, no. 1, pp. 103–107, 2010.
[48] A. Cheng *et al.*, "MRC2014: Extensions to the MRC format header for electron cryo-microscopy and tomography," *J. Struct. Biol.*, vol. 192, no. 2, pp. 146–150, 2015.
[49] T. Burnley, C. M. Palmer, and M. Winn, "Recent developments in the CCP-EM software suite," *Acta Crystallogr. Sect. Struct. Biol.*, vol. 73, no. 6, pp. 469–477, 2017.
[50] M. Abadi *et al.*, "TensorFlow: A System for Large-Scale Machine Learning.," in *OSDI*, 2016, vol. 16, pp. 265–283.
[51] F. Chollet, *Keras*. 2015.
[52] W.-L. Hung and M.-S. Yang, "Similarity measures of intuitionistic fuzzy sets based on Hausdorff distance," *Pattern Recognit. Lett.*, vol. 25, no. 14, pp. 1603–1611, 2004.
[53] I. N. Shindyalov and P. E. Bourne, "Protein structure alignment by incremental combinatorial extension (CE) of the optimal path.," *Protein Eng.*, vol. 11, no. 9, pp. 739–747, 1998.

APPENDIX A

Table A-1 Top 5 retrieval results of 4 more queries based on EMNets Descriptor

PDB		Retrieved PDB	Euc distance	Correlation	Z-score	Sequence identity
1t1b_A00	1	1t1a_A00	0.06065013	0.9993	6.8	100.00%
	2	1ota_A00	0.08644435	0.9985	6.8	100.00%
	3	1t19_A00	0.09598971	0.9983	6.8	100.00%
	4	3ve4_A00	0.13149369	0.9964	6.7	99.20%
	5	4bbv_A00	0.13231469	0.9971	6.8	99.20%
	Average		**0.10137851**	**0.99792**	**6.78**	**99.68%**
	Std		**0.03071987**	**0.00115845**	**0.04472136**	**0.00438178**
2yl3_A00	1	2yl1_A00	0.09643631	0.9971	6.8	99.20%
	2	3zqy_A00	0.13024815	0.9954	6.8	99.20%
	3	3ztz_A00	0.14083513	0.9983	6.9	100.00%
	4	2yl7_A00	0.16083739	0.9893	6.8	100.00%
	5	4d4x_A00	0.2709025	0.971	6.7	98.40%
	Average		**0.1598519**	**0.99022**	**6.8**	**99.36%**
	Std		**0.06631886**	**0.01128836**	**0.07071068**	**0.00669328**
1m8s_A00	1	1m8r_A00	0.03417638	0.9999	6.8	100.00%
	2	1bk9_A00	0.19792862	0.9967	6.8	96.80%
	3	1psj_A00	0.20592989	0.9961	6.8	96.80%
	4	1lds_A00	0.83153634	0.9017	6.8	6.10%
	5	1ae7_A00	0.87322339	0.8877	6.8	36.90%
	Average		**0.42855892**	**0.95642**	**6.8**	**67.32%**
	Std		**0.39319489**	**0.05657784**	**0**	**0.43241728**
5hdc_A00	1	5hd3_A00	0.0662727	0.9986	6.8	100.00%
	2	5hds_A00	0.07463924	0.9987	6.8	100.00%
	3	5hdd_A00	0.10061058	0.9983	6.8	100.00%
	4	4wl9_A00	0.11479353	0.997	6.8	99.20%
	5	2phy_A00	0.1428018	0.997	7	99.20%
	Average		**0.09982357**	**0.99792**	**6.8**	**99.68%**
	Std		**0.0309506**	**0.00085264**	**0**	**0.00438178**