The End of Molecular Dynamics Era: Protein-Specific Conformation Generation Using Deep Learning

A Graduation Project Thesis Presented to

School of Information Technology and Computer Science

Nile University

In Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science

Submitted By:

Esraa Abdullah	19105371	Maram Zoughieb	19105793
Elzahraa Saeed	19106429	Rana Rizk	19105575
Omar Mohamed	19100397		

Under Supervision of:

Prof. Tamer Mohamed Ibrahim

Professor at Information Technology and Computer Science School, Nile University (NU), Egypt.

Spring 2023

ACKNOWLEDGEMENTS

Praise and thanks to Allah, the project was completed successfully within the given duration of time. We would like to thank all those who have given us the possibility to complete this report. First, we are heartily thankful to our supervisor, Tamer Mohamed Ibrahim / Professor at Information Technology and Computer Science School at Nile University (NU) for his guidance and advice, encouragement, and support throughout our project development from the beginning to the end of the project. We would also like to acknowledge with much appreciation to Eng. Mohamed El kerdawy and Eng. Wafaa Salah-eldin for willing to examine and give comments on our project. Besides, special thank you to Eng. Mohamed Anwar and Eng. Sara Badawy for following with us all the details, updates, and give us comments for developing our project. Finally, yet importantly, special thanks also go to our families who always support and encourage me throughout our final year project journey.

TABLE OF CONTENTS

Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Proteins are flexible molecules, and their dynamics are intimately connected to their function (Chu et al., 2013). Proteins always exist in a dynamic state and each state is called a conformation, this conformation is a 3d structure which consists of X, Y, Z that represents the position of the atom at a given time. Computational methods like molecular dynamics can be used to define their conformational space in order to clarify the relationship between their molecular structure and function in an organism. The paper that we replicate had tried a lot of techniques to get all the states of the protein but only the molecular dynamics technique maintained the high accuracy. However, the cost of running the MD simulation leads us to use deep learning to predict those conformations instead of running the simulation for every single protein which this project provides by replacing the MD simulation by a generative neural network which will be trained on a protein coming from the simulation. Deep learning such as autoencoder and variational autoencoder are types of neural network that attempts to first compress and then decompress a multidimensional input, so that the difference between input and output is minimized. As a whole, this study shows that neural networks may be used to investigate the molecular conformational space.

CHAPTER ONE: INTRODUCTION

1.1 Introduction

Proteins are flexible molecules, and their dynamics are intimately connected to their function (Chu et al., 2013). Flexibility is often a key determinant of protein function. Molecular function results from the configuration of individual atoms and the dynamics that go along with them. Simple molecules interact in specific ways to create phenomena of increasing complexity, which culminate in the precisely honed biological systems that finally enable life. The function may be altered by binding to particular ligands like ions, small molecules, lipids, or other proteins, as well as conformational rearrangements in response to local environmental changes as diverse as changes in pH, temperature, or electrostatic potential. Therefore, proteins should be viewed as a conformational ensemble with more or less accessible states rather than as a single static structure.

Several methods have been developed to examine the structure of proteins in order to learn more about their molecular functions. X-ray crystallography provides precious structural evidence at atomic resolution, but its drawback is that it locks proteins in a single well-defined conformation within a crystal lattice. Nuclear magnetic resonance is another way to learn about atoms. This technique also informs about dynamics and can sometimes identify multiple states, in case that the molecule of interest is not too large. Even with this wide range of methods, it is typically difficult to investigate the structure of a protein with numerous states, and even when just one conformation is present, its thermal changes may make data interpretation difficult.

Proteins always exist in a dynamic state and each state is called a conformation, this conformation is a 3d structure which consists of X, Y, Z that represents the position of the atom

at a given time. By iteratively generating new structures based on an initial, well-known atomic arrangement and a physical model of interatomic interactions, computational approaches like molecular dynamics (MD) simulations aim to characterize molecular conformational space.

MD describes how each atom moves in a unit of time. It's a cycle that can be demonstrated by identifying the initial position of the atoms and velocity, identifying the distance between the atoms, calculating the energy of interaction between the atoms, calculating the total sum of energy of interaction in the system, calculating the force applied by an atom on the other atoms, applying the newton law and calculate the acceleration "giving the force and the mass", and finally, predicting how the atoms will move giving the acceleration and the velocity by calculating the new velocity of each atom and where it will move according to time "the new position of the atoms". Overall, Molecular dynamics (MD) simulations can give valuable information about the behavior of molecules and materials at the atomic level, but the computational cost increases significantly, which limits the size and timescales of simulations. Deep learning models is a solution that can help to reduce the computational cost of MD simulations and used to predict molecular behavior beyond the era of MD simulations.

The use of deep learning, and more specifically generative neural networks such as autoencoder enrich the sampling of molecular conformational space. Although generative neural networks are frequently used in the processing of images, their use in the analysis of 3D point clouds is relatively new (Achlioptas et al., 2017). Proteins represent a particularly interesting application case in this area, since they do not feature a difficulty typical of raw 3D point clouds. Autoencoder is a type of neural network that attempts to first compress and then decompress a multidimensional input, so that the difference between input and output is minimized. Variational autoencoder functions similarly to a regular autoencoder. Nonetheless, we proceed to a little change of the encoding-

decoding procedure to incorporate some regularization of the latent space, instead of directly mapping the input data points into latent variables, the input data points get mapped to a multivariate normal distribution. The trained autoencoder and variational autoencoder using the conformations produced by proteins MD simulations can generate new, realistic protein conformations without the need of MD simulations.

1.2 Project Description

The deep learning models can end the era of molecular dynamic. The autoencoder is a type of neural network that consists of two main parts: encoder that takes an input data point and maps it to a lower-dimensional latent space representation, and a decoder that takes the latent space representation and maps it back to the initial input space. At each iteration, there is a comparison between the encoded-decoded output with the input data, then propagate the error backward through the architecture to update the weights of the networks. Thus, the overall autoencoder architecture (encoder, decoder) creates a bottleneck for data that ensures only the main structured part of the information can go through and be reconstructed. The encoded-decoded output can be compared with the initial input by getting the loss function or the reconstruction error that should be minimized. The variational autoencoder functions similarly to a regular autoencoder, but with a change in the encoding-decoding procedure to incorporate some regularization of the latent space, instead of directly mapping the input data points into latent variables, the input data points get mapped to a multivariate normal distribution.

The trained autoencoder and variational autoencoder using the conformations produced by proteins MD simulations can generate new, realistic protein conformations without the need of MD simulations.

1.3 Problem Statement

Molecular dynamics (MD) simulations can give valuable information about the behavior of molecules and materials at the atomic level, but the computational cost increases significantly, which limits the size and timescales of simulations. Deep learning models confirm that it can overcome these limitations. It can help to reduce the computational cost of MD simulations and used to predict molecular behavior beyond the MD simulations era. Deep learning models can generate new protein conformations without the need of MD simulations. Finally, they can end the molecular dynamic simulation era.

1.4 Thesis Structure

A general description of each chapter is given as below:

Chapter 1: consists of an overview of the project including the introduction of the project, problem statement, objectives, aim, significance, and the expected outcome.

Chapter 2: emphasizes on reviewing existing research papers using deep learning models. This chapter starts with the introduction and followed by the reviewing of existing research papers using deep learning models. It also included comparison of the reviewed research papers. Some comparison research papers using models on their methodology and results.

Chapter 3: concentrates on the methodology used to develop the deep learning models. This chapter will discuss on the methodology such as building and developing the models.

Chapter 4: gives emphasis to the results, and models testing and evaluation.

Chapter 5: is about the conclusion and further enhancement of the project. In this chapter, the achievement of the project is stated and the future enhancement of the project also included here. There is also an overall conclusion of the project.

CHAPTER TWO: LITERATURE REVIEW

2.1 Related Works

Going through the field of molecular dynamics which has significantly contributed to our understanding of protein conformation and dynamics. However, recent advancements in deep learning techniques have paved the way for alternative approaches to generate protein-specific conformations. In this comprehensive literature review, we will delve into the key aspects of generating protein-specific conformations using deep learning, with a specific focus on the Fibroblast Growth Factor (FGF) family. Additionally, we will discuss the important PDB preprocessing steps, periodic boundary condition correction, trajectory centering, RMSD calculations, and the utilization of autoencoders in this context. Each of these topics will be thoroughly explored and compared with relevant references, providing a comprehensive overview of the latest developments in the field.

The research also showed that the Fibroblast Growth Factor (FGF) family plays a crucial role in various biological processes, including cell growth, development, and tissue repair. These factors have been extensively studied due to their involvement in numerous physiological and pathological conditions. To gain deeper insights into their functional mechanisms and potential therapeutic applications, we will closely examine the structural dynamics of several FGF proteins. Specifically, we will analyze the conformational dynamics of FGF2_STAB (PDB ID: 1FQ9), 2P23, and 2QKE. By understanding the intricacies of these FGF proteins, we can elucidate their structural characteristics, explore their conformational changes, and provide valuable insights into their functional relevance.

We also found that to ensure the accuracy and reliability of protein structures, several preprocessing steps are typically employed [1]. These steps include modeling missing residues and heavy atoms, cleaning heteroatoms, adding missing hydrogens, and extracting protonation states of histidine residues. Each of these steps contributes to refining the protein structures and ensuring their suitability for subsequent analyses. For example, modeling missing residues and heavy atoms involves reconstructing regions of the protein structure that may be absent in the original data. Cleaning heteroatoms removes extraneous molecules that are not part of the protein of interest. Adding missing hydrogens ensures that the protein structure is complete and properly protonated, while extracting protonation states of histidine residues accounts for the varying charge states that this amino acid can adopt. By meticulously addressing these preprocessing steps, researchers can obtain high-quality protein structures that serve as a solid foundation for further investigations.

the research [2] highlighted that the periodic boundary conditions are frequently used in molecular dynamics simulations to mimic bulk solvent effects, they can introduce artifacts and distortions in protein conformational analysis. These artifacts arise due to the proximity of the protein to its periodic image, resulting in unphysical interactions and artificial structural perturbations. To overcome these challenges, various correction methods have been developed to mitigate artifacts and improve the accuracy of protein conformational analysis. One commonly used method is the "image removal" approach, where redundant images of the protein are

eliminated by considering only the central unit cell. Another approach involves applying distance-based cutoffs to discard interactions beyond a certain distance threshold. These correction methods ensure that the generated conformations accurately represent the true behavior of the protein, enabling more reliable interpretations and predictions.

The research [3] used the Trajectories Centering and Frame Selection algorithms. Centering trajectories and selecting representative frames are essential steps to reduce noise and extract meaningful information from molecular dynamics simulations. By centering trajectories, researchers can eliminate systematic biases introduced by the simulation setup, ensuring more accurate analyses. This process involves translating the trajectory frames to a common reference frame, often the center of mass of the protein or a specific atom within the protein. Additionally, selecting representative frames at regular intervals, such as taking frames every 100ps, enables capturing essential conformational changes and avoiding excessive computational costs. These practices are critical for obtaining reliable and informative results from molecular dynamics simulations. They enable researchers to focus on the most relevant structural transitions, ensuring that the generated conformations are representative of the protein's conformational ensemble.

Root Mean Square Deviation (RMSD) [4] is a widely used metric for quantifying structural differences between protein conformations. It provides a measure of how closely one conformation resembles another by calculating the average deviation between corresponding atoms in two structures. In the context of protein-specific conformation generation, RMSD plays a crucial role in assessing the conformational dynamics of the FGF family. By calculating

RMSD values, researchers can evaluate the accuracy of generated protein conformations and compare them with experimentally determined references. This allows for a comprehensive analysis of the structural fidelity of the generated conformations. Moreover, RMSD analysis can reveal conformational transitions, highlight significant structural changes, and aid in the identification of functional regions within the protein structure.

According to the paper [5], The tertiary structure of proteins, particularly the backbone conformation, is vital for understanding protein folding, stability, and function. The backbone, consisting of peptide bonds and the associated atoms, forms the structural framework of the protein, dictating its overall shape and stability. Extracting the backbone information from protein structures is a fundamental step in generating protein-specific conformations. This process involves retaining only the backbone atoms while discarding side chains and other non-backbone elements. By focusing on the backbone conformation, researchers can analyze critical features such as secondary structures (e.g., alpha helices, beta sheets), inter-residue interactions, and overall folding patterns. Understanding the backbone conformation is crucial for accurately capturing the essential characteristics of the protein and generating conformations that reflect its native structure.

One of the state-of-the-art research [6] used Autoencoders. Autoencoders is a class of deep neural networks, it has gained considerable attention in the field of protein conformation generation. These networks are capable of capturing intricate protein features and generating

protein-specific conformations. Autoencoders consist of an encoder network that compresses the input data into a low-dimensional latent space and a decoder network that reconstructs the input from the latent representation. In the context of protein conformation generation, the encoder-decoder architecture of autoencoders allows for the mapping of high-dimensional protein representations to lower-dimensional latent spaces and subsequent reconstruction of protein-specific conformations. By training the autoencoder on a large dataset of protein structures, the network learns the underlying structural patterns and can generate novel conformations that are consistent with the learned representations. This approach offers a promising alternative to traditional molecular dynamics simulations for generating protein-specific conformations, as it leverages the power of deep learning to capture complex structural features and generate diverse conformations. Furthermore, the use of autoencoders enables the exploration of the conformational landscape, identification of energetically favorable conformations, and potential applications in protein engineering and drug discovery.

2.2 Related works methodologies

Assessing the effectiveness and accuracy of various techniques is crucial. However, the current paper's methodology has several limitations that can be addressed in future studies. One such limitation is the subjective nature of correlation-based metrics used to evaluate the preservation of information in the latent space, as they may not provide accurate results. Additionally, relying solely on RMSD and DOPE to compare training and decoded structures is insufficient for comprehensively evaluating the model's quality. Similarly, using MMD and EMD to compare training and generated distributions lacks robustness in measuring distribution similarity. Moreover, the use of MSMs and implied timescales to estimate relaxation timescales is computationally expensive.

To overcome these limitations, future research can incorporate more objective measures for evaluating the model's quality, such as assessing the accuracy of predicted secondary structures. Supplementing RMSD and DOPE with additional measures, like the root-mean-square deviation of dihedral angles, would provide a more comprehensive evaluation. Moreover, replacing MMD and EMD with a more robust measure, such as the Wasserstein distance, would improve the accuracy of comparing distributions. Finally, utilizing less computationally expensive methods, such as the iterative Boltzmann sampling method, would be more feasible for estimating relaxation timescales.

2.3 Related works results

In the realm of enzymology, the conformational dynamics of adenosine kinase (ADK) play a vital role in its functionality. This literature review focuses on a study that explores these conformational transitions using molecular dynamics (MD) simulations and autoencoder-based techniques. Specifically, the authors employ four crystal structures of ADK to conduct their simulations.

To analyze the simulation data, the researchers train autoencoders and variational autoencoders with varying numbers of hidden layers. Among these models, the variational autoencoder with four hidden layers demonstrates superior performance, exhibiting high Spearman and Pearson coefficients, as well as low root-mean-square deviation (RMSD).

Moreover, the learned latent space derived from the variational autoencoder allows for the generation of similar or distinct protein conformations. By selecting points in the latent space that are nearby or distant, respectively, from the original structure, researchers can generate conformations with specific characteristics.

In terms of the MD simulations, the study compares four short trajectories to a single long trajectory. Surprisingly, the combined short simulations adequately cover a similar conformational space compared to the long trajectory. Interestingly, the short simulations sample regions near the LID-closed NMP-open structure (PDB id: 1DVR) more frequently than the long trajectory.

Additional insights can be gained from Figure 7B, which showcases a 2D angle map. It reveals a correlation between the NMP-CORE and LID-CORE angles, suggesting their significance in understanding ADK's conformational changes. Furthermore, Figure 9 exhibits implied timescales, demonstrating that the short combined simulations efficiently sample the ADK conformational space and can thus be employed for studying its dynamics.

Overall, this paper presents an innovative approach to investigate the conformational changes of ADK. The findings highlight the efficacy of the variational autoencoder with four hidden layers as a powerful tool for studying these transitions, providing valuable insights into ADK's behavior

In conclusion, the literature review has provided an in-depth exploration of the emerging trend of generating protein-specific conformations using deep learning techniques. By focusing on the Fibroblast Growth Factor (FGF) family, PDB preprocessing steps, periodic boundary condition correction, trajectory centering, RMSD calculations, and the utilization of autoencoders, we have highlighted the potential of deep learning in revolutionizing protein conformation generation. Through an extensive comparison with relevant references, we have shed light on the latest advancements and challenges in this rapidly evolving field. The utilization of deep learning techniques in protein conformation generation holds great promise for biomedical informatics, offering new avenues for understanding protein structure and function. Future research in this area will undoubtedly further expand our knowledge and facilitate the development of novel therapeutic interventions, such as structure-based drug design and protein engineering.

It also discussed deep learning approaches, such as the use of autoencoders, which is considered a powerful tool to generate protein-specific conformations with improved accuracy and efficiency. By training autoencoder models on large datasets of protein structures, researchers can capture the intricate structural features and generate diverse conformations that are consistent with the learned representations. This allows for the exploration of the conformational landscape, identification of energetically favorable conformations, and potential applications in protein engineering and drug discovery.

However, it is important to acknowledge the limitations and challenges associated with deep learning-based protein conformation generation. One major challenge is the availability of high-quality training datasets, as obtaining experimental structures for all possible protein conformations is impractical. Additionally, the generalization of deep learning models to unseen

protein structures and the interpretability of the generated conformations are areas that require further investigation.

Nevertheless, the integration of deep learning techniques in protein conformation generation represents a significant advancement in the field of molecular dynamics. It offers a complementary approach to traditional simulation methods, providing a rapid and efficient means to explore protein structure and function. The ability to generate protein-specific conformations using deep learning has the potential to revolutionize various aspects of biomedical informatics, including drug discovery, protein engineering, and understanding disease mechanisms.

Finally it highlighted that by focusing on the Fibroblast Growth Factor (FGF) family and exploring topics such as PDB preprocessing steps, periodic boundary condition correction, trajectory centering, RMSD calculations, and the utilization of autoencoders, we have highlighted the potential of deep learning in advancing protein conformation generation. While challenges and limitations remain, the integration of deep learning in this field offers exciting possibilities for further research and innovation. Continued efforts in this area will undoubtedly contribute to the development of new therapeutic strategies and deepen our understanding of protein structure and function in biomedical informatics.

CHAPTER THREE: METHODOLOGY

3.1 Protein Preparation

Protein preparation is a critical step in conducting MD simulations. The aim is to obtain a high-quality protein structure that is free from steric clashes and other artifacts that could affect the results of the simulation. In this study, we used PDB preprocessing to prepare the raw PDB files.

The first step, involved modeling missing residues and heavy atoms using Deep View.The software can automatically add missing atoms and residues based on the surrounding environment, ensuring that the protein structure is complete.

The second step, involved cleaning heteroatoms. Heteroatoms are non-protein molecules that are present in the crystal structure, such as water molecules or ligands. These molecules can interfere with the simulation and should be removed from the structure. Deep View was used to clean heteroatoms by selecting and deleting the unwanted atoms.

The third step, involved adding missing hydrogens. Hydrogen atoms are often missing from PDB structures, but they play a crucial role in MD simulations. They affect the electrostatic interactions between atoms, which can significantly impact the results of the simulation. In this study, Deep View was used to add missing hydrogens based on the protein's geometry.

The fourth step, involved extracting protonation states of histidine residues. Histidine residues can exist in different protonation states, which can affect the protein's behavior in the simulation. Chimera was used to extract the protonation states of histidine residues by calculating the pKa values of the residues and determining their protonation state based on the pH of the solution.

3.2 Protein Structures

After preparing the protein structures, we conducted MD simulations using GROMACS.

The first step, was to generate topology files, which specify the force field and water model used in the simulation. The force field is a set of mathematical equations that describe the interactions between atoms, while the water model defines the behavior of water molecules in the simulation. In this study, we used the CHARMM27 force field and the TIP3P water model.

The second step, was to solvate the protein in a box of water molecules. Solvation is necessary because it mimics the physiological environment in which proteins exist. It also allows the protein to move freely during the simulation. In this study, we used the Simple Point Charge (SPC) water model to solvate the protein.

The third step, was to minimize the energy of the system to remove any remaining steric clashes. Energy minimization involves minimizing the potential energy of the system by moving the atoms to their lowest energy state. This step is necessary because the protein structure may still contain steric clashes after the previous steps. In this study, we used the steepest descent algorithm to minimize the energy of the system.

The fourth step, was to equilibrate the system by gradually increasing the temperature and pressure of the system to reach the desired temperature and pressure of the simulation. This step involves two stages: NVT (constant number, volume, and temperature) and NPT (constant number, pressure, and temperature) equilibration. In the NVT equilibration stage, the number of particles, volume, and temperature of the system is held constant, while in the NPT equilibration stage, the pressure of the system is also held constant. During the equilibration stage, the system is allowed to adjust to the desired temperature and pressure gradually.

The final step, was to perform the production MD simulation, which involves running the simulation for a specified time. During the simulation, the positions and velocities of the atoms are recorded, and the resulting trajectory can be analyzed to determine the protein's structural and dynamic properties. In this study, we used a simulation time of 100 ns to study the behavior of the four different protein samples and their interaction with FGF.

3.3 GROMACS

To analyze the results of the simulation, we used various tools such as GROMACS analysis tools and VMD. We calculated different structural and dynamic properties of the proteins, including root-mean-square deviation (RMSD), root-mean-square fluctuation (RMSF), radius of gyration, and solvent accessible surface area. We also analyzed the protein-protein interaction by calculating the binding free energy between FGF and the four protein samples using the Molecular Mechanics/Generalized Born Surface Area (MM/GBSA) method. The results of the simulation showed that the flexibility of the protein plays a crucial role in its interaction with FGF. The rigid version of FGF2 and its close homolog, FGF2_STAB, had the strongest interaction with FGF, while the distant homolog, 2P23, showed a weaker interaction. The fold-switcher, 2QKE, showed a moderate interaction with FGF. The results also showed that FGF interacted primarily with the beta strands and loops of the protein.

3.4 Periodic Boundary Condition

After the production part has been finished, some processing needs to be done for the output files “XTC files“ that have the simulation trajectory. The first problem we encountered in the molecular dynamics “MD” section, there were periodic boundary conditions “PBC” in the parameter file without correction. However, the PBC part needed to be done as protein will periodically bounded all over the boxes, and atoms will be controlled. Though, PBC correction is indispensable for preventing protein fragmentation.

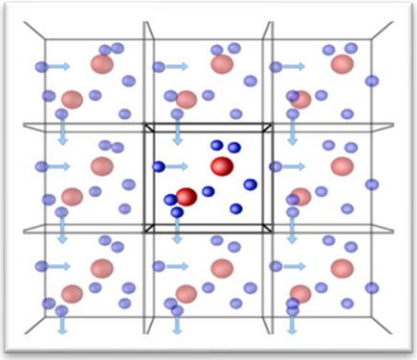


Figure1: Periodic Boundary Condition

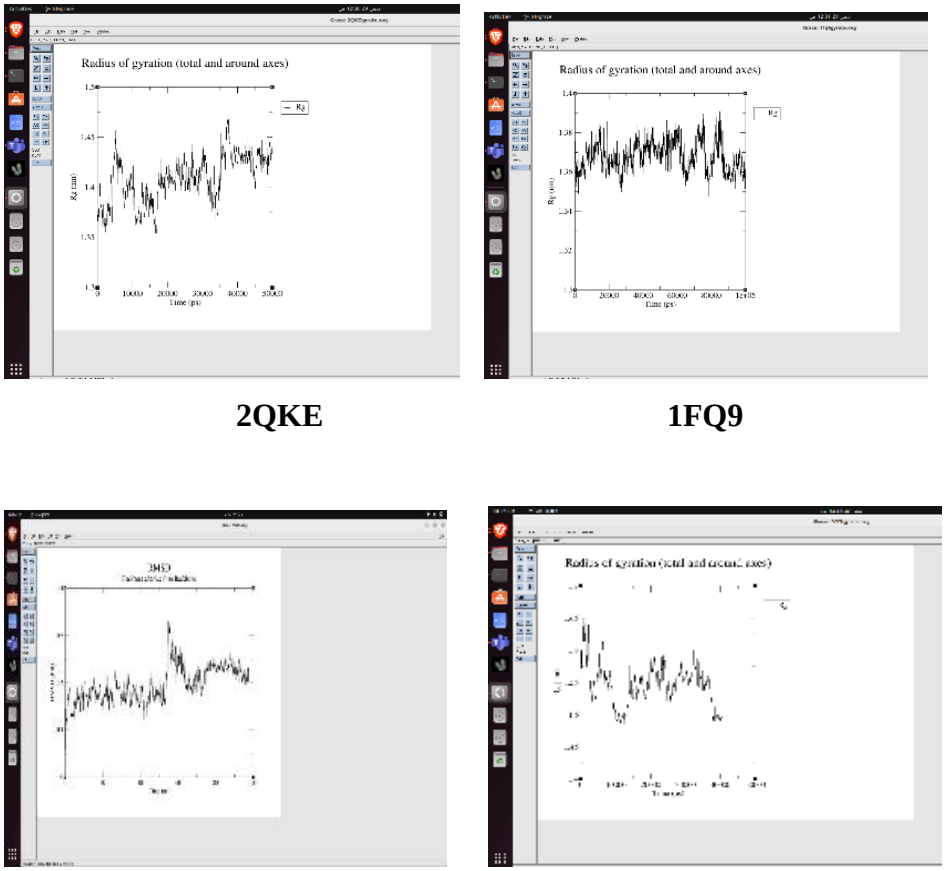
3.5 Trajectory Centering

Now, to make the previously mentioned centring for atoms, it will not let go free all over the boxes; conversely, it will be controlled so that frames could be taken correctly to measure Root Mean Square Deviation (RMSD) for each frame at the same coordinate. It is an essential step for deep learning training phase.

The output frames are pdb files containing every atom with all details which means that each frame is a movie tape it took snapshots each shot is a stable image for one pdb file every 100 ps.

3.6 Root Mean Square Error" RMSD"

RMSD used to check similarity between structures, the RMSD value gives the average deviation between the corresponding atoms of two proteins, however the smaller the RMSD, the more similar the two structures. According to proteins used in this project, As shown in this fig , RMSD value was smaller than 2 Å which is a reasonably acceptable RMSD real number for RMSD. It took the last pdb or origin pdb and compares it with output frames in the simulation.



2QKE **1FQ9**

2P23 **FGF2**

Figure2: RMSD for all_ poteins

3.7 Extracting Backbone

A step back will be taken and use the first XTC file ignoring the output pdb file because some additional steps need to be made on this file, the output will not be a PDB file it will be an XTC file" without PBC". Structures of trajectory file will be Aligned to the reference structure, progressively as it took the first frame and aligned it on the reference one and the second frame aligned it on the frame that has been fitted, So, all structures will be centred on the same origin.

To extract a new PDB file as in frames but whole pdb file will not be used , PDBs that will be extracted containing a new group with only backbone (tertiary structure) with atoms (CA, C, N) + CB group.

This study demonstrated the importance of protein preparation and MD simulations in studying protein-protein interactions. We used PDB preprocessing and GROMACS to prepare and simulate four different protein samples interacting with FGF. We analyzed the simulation results using various tools and found that the flexibility of the protein plays a crucial role in its interaction with FGF. The results provide valuable insights into the molecular mechanisms underlying the interaction between FGF and different proteins and could be useful in designing new drugs that target this interaction. Overall, the methodology used in this study provides a framework for conducting similar studies in the future.

4.1 Deep Learning Model

The methodology that has been used to carry out this project is based on building the deep learning models. Deep learning such as Autoencoder model, which is an unsupervised deep learning model, consisting of encoder and decoder. The encoder part encodes the dataset into a lower dimensional latent vector which then will be decoded using the decoder part to the original input. In this process, it is important to minimize the reconstruction error between the input data and the reconstructed one.

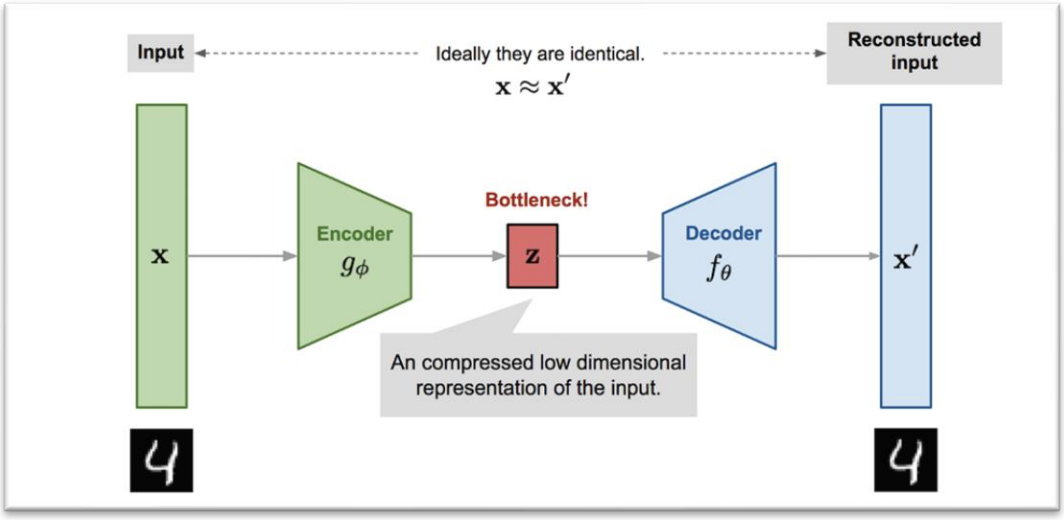


Figure3: Autoencoder Architecture

4.2 Language and libraries used

- Pytorch
- Python
- Biopython
- Bash Script
- Ubuntu

5.2 Dataset

The dataset we used to train the model was obtained from the MD Simulations for the four proteins:

- 2QKE
- FGF2
- 2P23
- 1FQ9

The trajectory file, which contains all the conformation of each protein, is considered the input of the model. This file is passed through a python script to fetch only the atoms' positions we are interested in. Those atoms are (N, C Alpha , C, and C Beta) where each atom has 3 coordinates X,Y, and Z.

For the 2QKE protein: after passing its trajectory file to the python script to fetch the important atoms' positions, the number of conformations are 500 where the number of atoms in each conformation are 392 atoms.

For the FGF2 protein: after passing its trajectory file to the python script to fetch the important atoms' positions, the number of conformations are 406 where the number of atoms in each conformation are 620 atoms.

For the 2p23 protein: after passing its trajectory file to the python script to fetch the important atoms' positions, the number of conformations are 975 where the number of atoms in each conformation are 544 atoms.

For the 1FQ9 protein: after passing its trajectory file to the python script to fetch the important atoms' positions, the number of conformations are 1001 where the number of atoms in each conformation are 516 atoms.

After obtaining the atoms' positions for all the conformations, the initial PDB structure is passed to another python script to obtain the protein backbone.

5.3 Data Loader

After the visualization of some conformations to make sure the dataset is right, a custom data loader is initiated to obtain the data in batches where the batch size is 4 conformations. We tried to apply normalization in different ways, but the results were wrong.

5.4 Alignment using kabsch torch and batch kabsch torch functions

The function tries to align two sets of points in 3D space which in our case is aligning the reconstructed conformation to the input conformation. First, calculating the center mass of each set. Second, calculate covariance matrix (for each prot in the batch). Third, making Optimal rotation matrix via SVD making sure that W is transposed. Forth, calculating the determinant sign for direction correction, then Creating the Rotation matrix U and calculating rotations. Finally, return centered and aligned. Using batch kabsch torch function to deal with batched data assuming that X, Y are both (B, 3, N).

5.5 Building the model

The autoencoder includes an encoder and decoder, both of which are feedforward neural networks. The encoder converts a tensor input to a lower-dimensional latent space representation. The decoder then applies this latent representation to the original input space. The used hidden dimensions were specified as a list of integers which are the number of atoms. There were four hidden dimensions which represented the number of neurons in each hidden layer of the fully connected network. The encoder and decoder have the same number of hidden dimensions, but decoder was in reverse order. Hidden dimensions were chosen specifically as they could affect the performance of autoencoder and made the reconstructed worse. Between every hidden layer batchnorm1d was used. We tried to use dropout layer with 0.2 and increased this value but the reconstructed output was getting worse.

(Degiacomi,2019) they tried to implement hidden dimensions with 3 and 5 layers, but 5 layers was better for them, we tried to use 5 layers but ended up with nan values, so the hidden dimensions ended up in our model to be 4 layers.

For the first step before measuring the loss function we have aligned the reconstructed output and the input together with batch kabsch torch function to get the mean square error between each correctly, then the used loss function was Mean Squared Error (MSE). The MSE loss measures the average squared difference between the reconstructed output and the original input across all dimensions of the tensor.

5.6 Model Training

The model was trained with 100 latent dimensions using Adam optimizer with a learning rate of 5e-4. The total number of iterations is set between 350 – 450, and the batch size was 4. During

training, the model iterated over the batches of data in the dataloader and computed the forward pass through the autoencoder. The reconstruction loss is then computed, and the gradients were backpropagated through the network. The optimizer then updated the model parameters and these operations kept being updated through the whole training.

For the latent dimension (Degiacomi,2019) they tried 2,3,5 and ended up using 2. We tried 2,3,10,15 and even 30, but it didn't improve the result. In addition, they used 50,100,200,300. We used only 16 so the model can learn, but also tried 50 and nothing has improved, then we updated the batch size to be only 4 for better results.

5.7 Bash script for conversion

We created a bash script to iterate over all confirmations which were npy files from the input and reconstructed output to align them on the backbone and converted them to pdb files.

5.9 Sampling

After training, we sampled from the latent space by randomly selecting. These random samples have been passed through the decoder to generate new outputs.

5.10 Validation

5.10.1 Cross Validation & Concatenation

Three bash scripts were created to iterate over:

- First bash script, we aligned two inputs in pairs to calculate the RMSD “root mean square error” between them.
- Second bash script, we aligned two outputs in pairs to calculate the RMSD “root mean square error” between them.
- Third bash script, we aligned one input and one output in pairs to calculate the RMSD “root mean square error” between them.

The generated RMSDs from scripts were TSV files. We concatenated these TSV files to one file for each script.

5.10.2 DSSP “Dictionary of Secondary Structure of Proteins”

Before we checked the secondary structure, we added missing atoms for input and reconstructed output using DeepView software.

We implemented a code with python that read the secondary structure assignments from two DSSP files, one for input and one for output, converted them to NumPy array, and finally calculated the Q3 score, which is a measure of the accuracy of the predicted secondary structure assignments.

CHAPTER FOUR: RESULTS AND DISCUSSION

6.1 Introduction

Protein structure analysis is a critical component of many fields, including biochemistry, molecular biology, and drug development. Understanding the structure of proteins can provide insights into their function and interactions and can inform the design of new therapeutic agents. However, accurate analysis of protein structure requires careful preprocessing of the data to ensure that only the relevant information is analyzed.

6.1.1 MD simulation

In this study, we performed preprocessing of four proteins, to prepare them for further analysis.

- FGF2 wildtype
- 1FQ9
- FGF19 “2P23”
- KaiB “2QKE”

The first step in preprocessing the proteins was to remove extraneous molecules, such as heteroatoms and solvent molecules. Heteroatoms are atoms other than carbon or hydrogen and can be found in molecules such as ligands or cofactors. Removing heteroatoms is important to ensure that only the protein of interest is analyzed, and any extraneous molecules are excluded. Solvent molecules, on the other hand, are molecules used to dissolve the protein in solution, and can interfere with accurate analysis of the protein structure. Removing solvent molecules is important to obtain an accurate representation of the protein structure.

The software program Chimera was used for removing heteroatoms and solvent molecules from the proteins. Chimera is a powerful software tool for molecular visualization and modeling and is widely used in the field of protein structure analysis. To remove heteroatoms and solvent molecules, we loaded the protein structures into Chimera, and used the “delete” function to remove the extraneous molecules. This step was completed for all four proteins.

After removing heteroatoms and solvent molecules, the next step in preprocessing the proteins was to isolate single chains. Proteins are composed of chains of amino acids and analyzing individual chains can provide insights into their specific functions and interactions. Isolating single chains is important to ensure that only the chain of interest is analyzed, and any extraneous chains are excluded.

To isolate single chains, we used Chimera again. We loaded the protein structures into Chimera and the “select chain” function has been used to isolate the chain of interest. For FGF2 wildtype the single chain of interest was identified by chain ID A, 1FQ9, the single chain of interest was identified by chain ID A. For FGF19 2P23, the single chain of interest was also identified by chain ID A. For KaiB 2QKE, the single chain of interest was identified by chain ID A as well.

After isolating the single chains, the next step in preprocessing the proteins was to add missing hydrogens. Hydrogens are the most abundant element in proteins and are critical for accurate analysis of the protein structure. Hydrogens are typically not included in protein structure files obtained from X-ray crystallography or NMR spectroscopy and must be added manually to obtain an accurate representation of the protein structure.

To add missing hydrogens, we used Chimera again. We loaded the protein structures into Chimera and used the “add hydrogens” function to add missing hydrogens. This step was completed for all proteins.

The final step in preprocessing the proteins was to add missing heavy atoms. Heavy atoms are atoms other than hydrogen and are critical for accurate analysis of the protein structure. Heavy atoms are typically missing from protein structure files obtained from X-ray crystallography or NMR spectroscopy and must be added manually to obtain an accurate representation of the protein structure.

To add missing heavy atoms, we used the software program Deep View. Deep View is a powerful software tool for molecular visualization and modeling and is widely used in the field of protein structure analysis. We loaded the protein structures into Deep View and the “build” function has been used to add missing heavy atoms. This step was completed for all proteins.

After completing all the preprocessing steps, we were able to obtain accurate representations of the protein structures for further analysis. By removing heteroatoms and solvent molecules, isolating single chains, and adding missing hydrogens and heavy atoms, we were able to obtain a clear and accurate representation of the proteins.

The preprocessing steps we performed were critical for accurate analysis of the protein structures and are commonly used in the field of protein structure analysis. Preprocessing of protein structures is an important step in many fields, including drug development, where accurate analysis of protein structures is critical for the design of new therapeutics.

Preprocessing steps have been performed for four proteins, FGF2 wildtype, 1FQ9, FGF19 “2P23”, and KaiB “2QKE”, to prepare them for further analysis. We removed heteroatoms and solvent molecules using Chimera, isolated single chains using Chimera, added missing hydrogens using Chimera, and added missing heavy atoms using Deep View. By performing these preprocessing steps, we were able to obtain accurate representations of the protein structures for further analysis. The preprocessing steps we performed are critical for accurate analysis of protein structures and are commonly used in the field of protein structure analysis.

After preparing proteins for deep learning model, as results of extracting backbone processes, frames will be skipped every 100 ps and make those frames as pdb files with only selected new atoms. It is essential as representative could not replace it.

- Firstly, skipping is fairly good and without it, it will extract billions of PDBs and total simulation will be thousands of ps .
- Representatives will affect data, it will be so small and not valid for training model.

After handling MD simulation part, timeframes coordinates will be converted into array format to make it easy for training model to understand the - MD trajectories - output.

6.1.2 Deep Learning Model

We needed to Convert timeframes coordinates to numpy array format so that, model could deal with it - after skipping frames - selecting only Ca, C, N, and Cb atoms to represent the proteins' backbone and sidechains directions. For the first step on this model, we extracted number of atoms and its conformations for each protein, the conformation of a protein refers to its 3D shape and arrangement of atoms.

- **2QKE:** number of atoms: **392** , number of conformations : **500.**
- **FGF2**: number of atoms: **620** , number of conformations : **406.**
- **1FQ9:** number of atoms: **516 ,** number of conformations: **1001.**
- **2P23:** number of atoms**:** **544 ,** number of conformations: **975.**

The next steps have been done to all proteins using their trajectories.

6.1.2.1 Visualize all array

Visualization of 400 conformations for each protein to make sure all arrays have been extracted from trajectory file.

2QKE Protein:

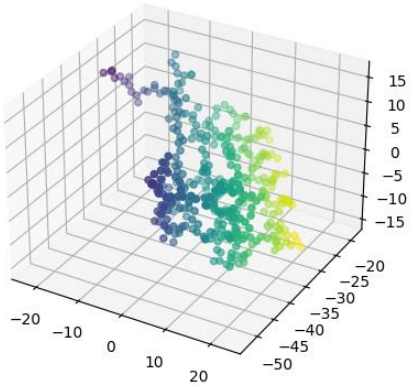


Figure 4.1: 2QKE_all_array ['400']

FGF2 Protein:

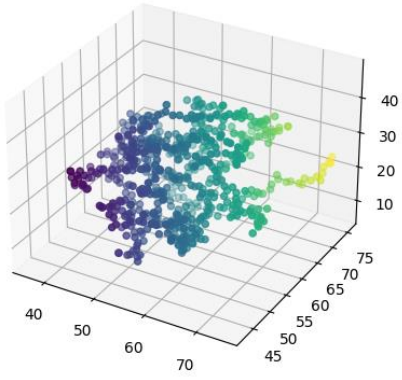


Figure 4.2: FGF2_all_array ['400']

1Fq9 Protein:

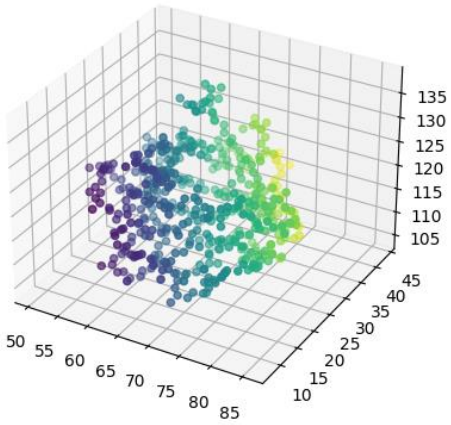


Figure 4.3:1Fq9_all_array [‘400]

2P23 Protein:

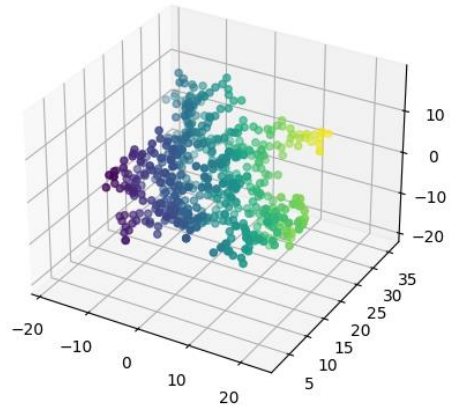


Figure 4.4: 2P23_all_array [‘400’]

7.2 Autoencoder

The autoencoder tries to learn a lower-dimensional representation of the input data that captures the most important features or patterns of the data.

7.2.1 Dataset & Data loader

A data loader in PyTorch is a utility that helps to load data in batches for efficient processing during training and evaluation of our model, the results of a data loader were batches of data that can be fed into a machine learning model. Each batch typically includes a set of input data and their corresponding labels with generated samples -gt_samples- shape of (number of atoms,3). Then we visualized gt_samples after turning data into batches.

2QKE Protein:

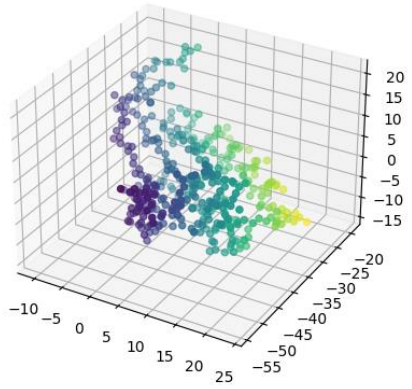


Figure 5.1 batched 2QKE_gt_samples

FGF2 Protein:

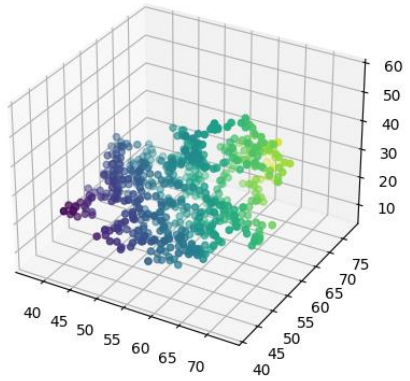


Figure 5.2 batched FGF2_gt_samples

1Fq9 Protein:

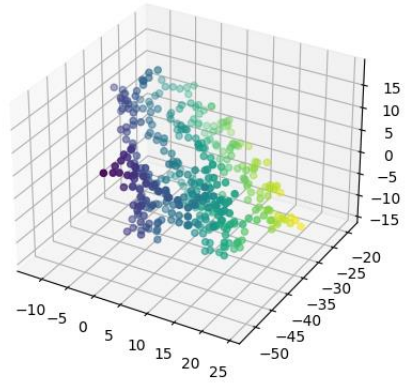


Figure 5.3 batched 1Fq9_gt_samples

2P23 Protein:

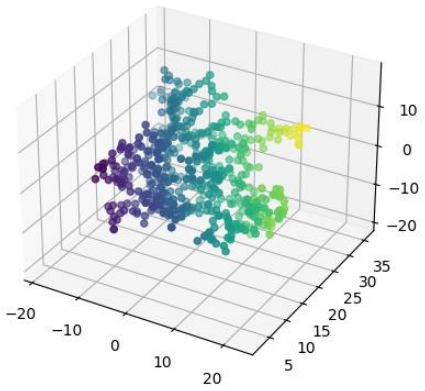


Figure 5.4 batched 2P23_gt_samples

7.2.2 Training Data

7.2.2.1 2QKE Protein

7.2.2.1.1 Reconstructed after training

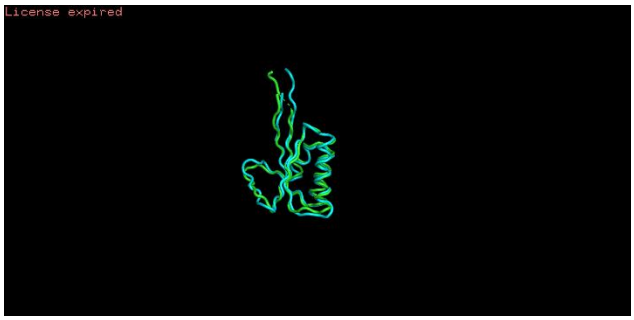


Figure 6 2QKE_Reconstructed input and output

7.2.2.1.2 Loss Function

Loss function measures the difference between the predicted output of model and the correct output. The goal is to minimize the loss function, which means that it tries to find the best possible model that fits the given data. For 2QKE Protein loss function was 0.6 which is very good after training the whole data with 450 iterations.

7.2.2.1.3 *Cross Validation*

To get RMSD for all inputs and outputs after training - smaller RMSD than 2 the better it gets- for example, RMSD for input_235 and output _235 for 2QKE was **0.857 A**^{0}.

7.2.2.1.4 *Concatenation of TSV files for inputs and outputs*

After, we need to collect whole RMSD in one file for inputs and outputs data. The following figures show the RMSDs and its histograms.

For (fig 3.2) it shows the best (0.75 A˚ RMSD), average (1.50 A˚ RMSD), and worst (2.50 RMSD A˚) models produced.

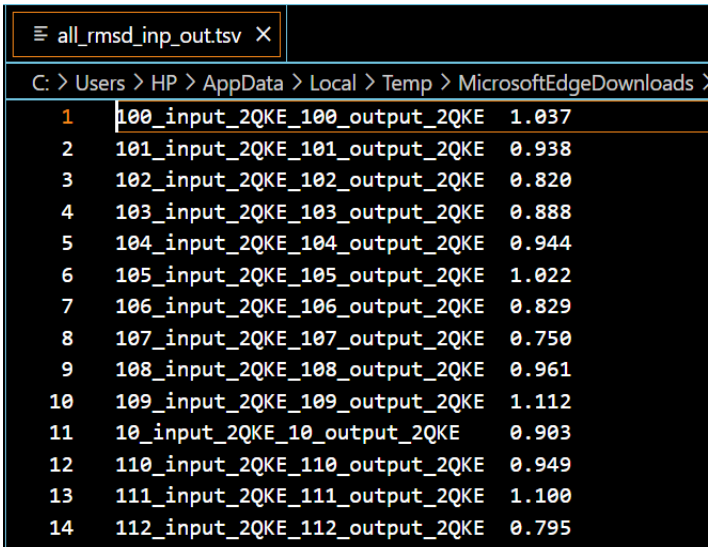


Figure 7.1 RMSD for 2QKE input_output

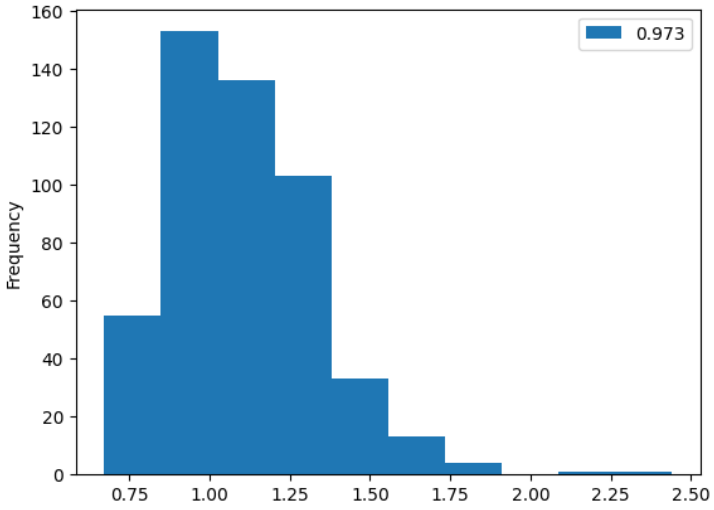


Figure 7.2 RMSD histogram for 2QKE input_output

7.2.2.1.5 Diversity

For diversity step, it is important for developing accurate and robust model for protein structure prediction, we also generated RMSDs between inputs_inputs and outputs_outputs. The following figures show that both have the best (0.75 A˚ RMSD), average (1.50 A˚ RMSD), and worst (2.50 A˚ RMSD) models produced.

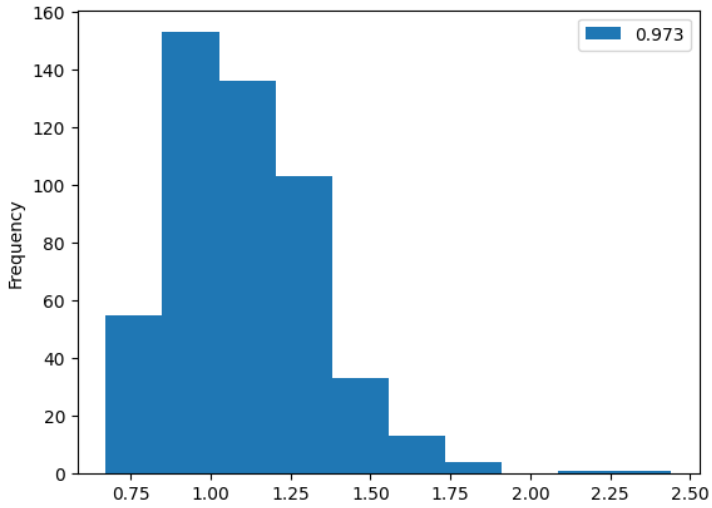


Figure 8.1 RMSD_diversity histogram for input_intput

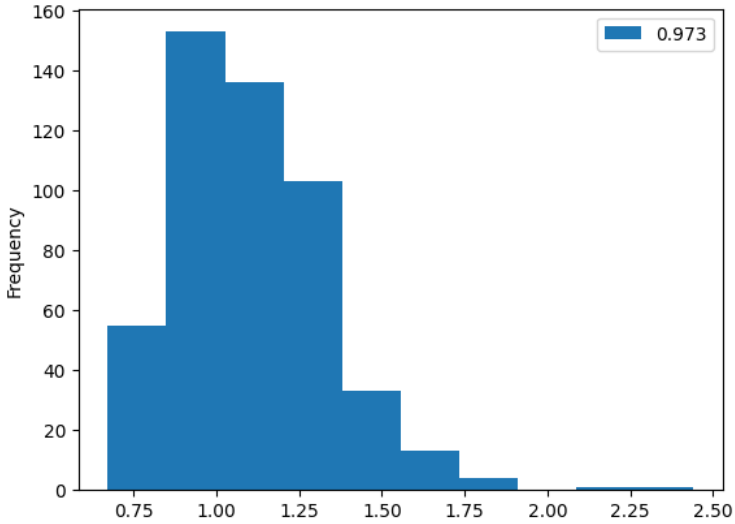


Figure 8.2 RMSD_diversity histogram output_output

7.2.2.1.6 Ramachandran Plot

Ramachandran plot is used in the development and validation of computational methods for protein structure prediction. The following figures show that input and output in alpha-helix and beta-sheets (red region) are almost the same. Points in the same area means it is good quality. data.

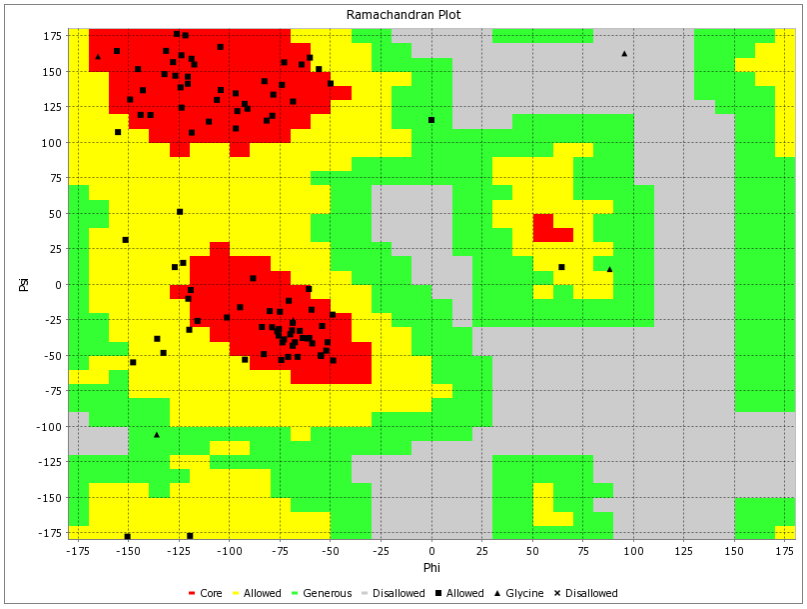


Figure 9.1 Ramachandran for inputs

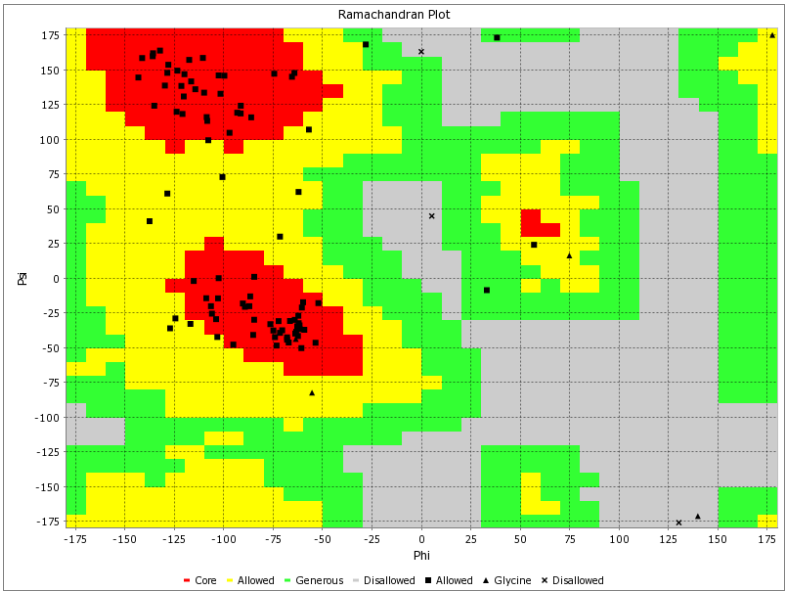


Figure 9.2 Ramachandran for outputs

7.2.2.2 FGF2 Protein

7.2.2.2.1 Reconstructed after training

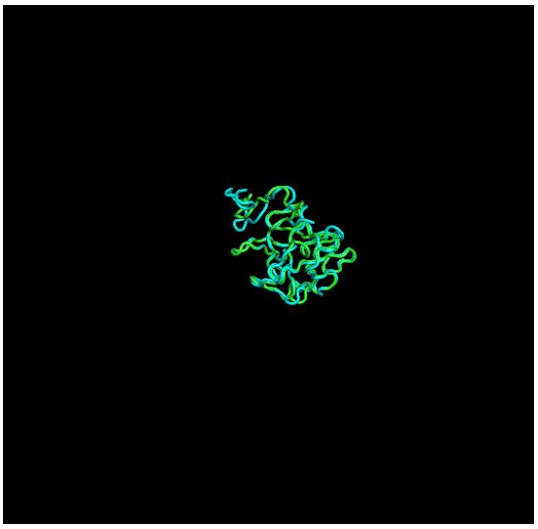


Figure 10 FGF2_Reconstracted input and output

7.2.2.2.2 Loss Function

Loss function measures the difference between the predicted output of model and the correct output. The goal is to minimize the loss function, which means that it tries to find the best possible model that fits the given data. For FGF2 Protein loss function was 2.8 which is very good after training the whole data with 450 iterations.

7.2.2.2.3 *Cross Validation*

To get RMSD for all inputs and outputs after training - smaller RMSD than 2 the better it gets- for example, RMSD for input_280 and output _280 for FGF2 was **0.899 A**^{0}.

7.2.2.2.4 *Concatenation of TSV files for inputs and outputs*

After, we need to collect whole RMSD in one file for inputs and outputs data. The following figures show the RMSDs and its histograms.

For (fig 6.2) it shows the best (> 1.0 A˚ RMSD), average (2.0 A˚ RMSD), and worst (3.0 A˚ RMSD) models produced.

all_rmsd_inp_out.tsv ×

C: > Users > HP > Downloads > all_rmsd_inp_out.tsv

1	100_input_fgf2_100_output_fgf2	0.964
2	101_input_fgf2_101_output_fgf2	1.100
3	102_input_fgf2_102_output_fgf2	1.476
4	103_input_fgf2_103_output_fgf2	0.681
5	104_input_fgf2_104_output_fgf2	0.852
6	105_input_fgf2_105_output_fgf2	0.720
7	106_input_fgf2_106_output_fgf2	0.805
8	107_input_fgf2_107_output_fgf2	0.860
9	108_input_fgf2_108_output_fgf2	1.239
10	109_input_fgf2_109_output_fgf2	1.351
11	10_input_fgf2_10_output_fgf2	1.887
12	110_input_fgf2_110_output_fgf2	1.136
13	111_input_fgf2_111_output_fgf2	1.066

Figure 11.1 RMSD for FGF2 input_output

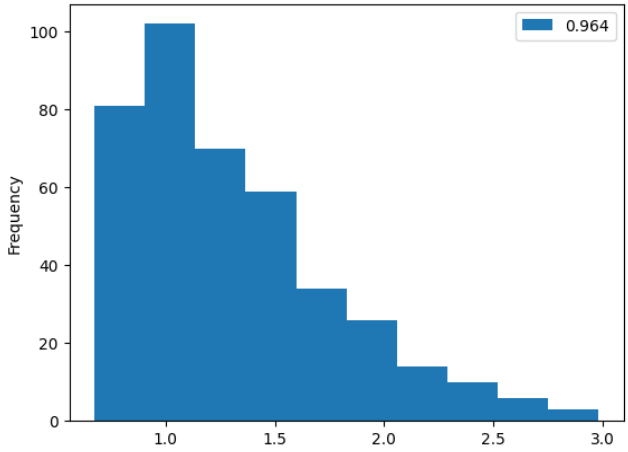


Figure11.2 RMSD histogram for FGF2 input_output

7.2.2.2.5 Diversity

For diversity step, it is important for developing accurate and robust model for protein structure prediction, we also generated RMSDs between inputs_inputs and outputs_outputs. The following figures show that both have the best (<1.0 A˚ RMSD), average (2.0 A˚ RMSD), and worst (3.0 A˚ RMSD) models produced.

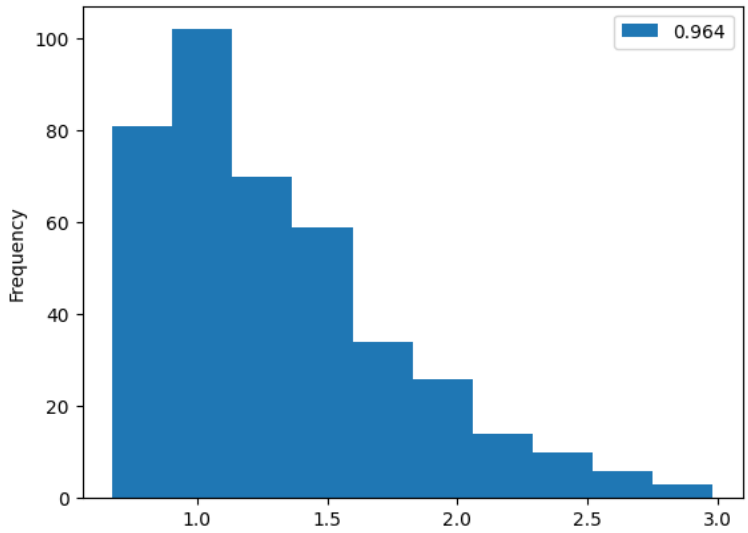


Figure 12.1 RMSD_diversity histogram for input_intput

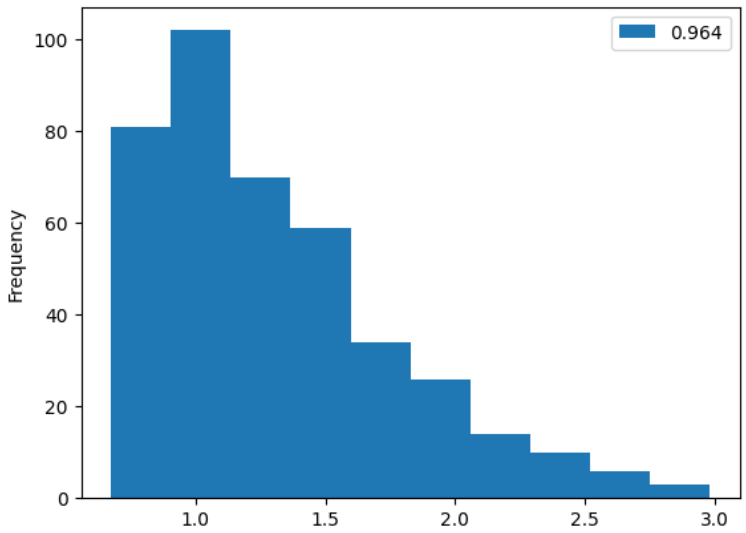


Figure 12.2 RMSD_diversity histogram output_output

7.2.2.2.6 Ramachandran Plot

Ramachandran plot is used in the development and validation of computational methods for protein structure prediction. The following figures show that input and output in alpha-helix and beta-sheets (red region) are almost the same. Points in the same area means it is good quality data.

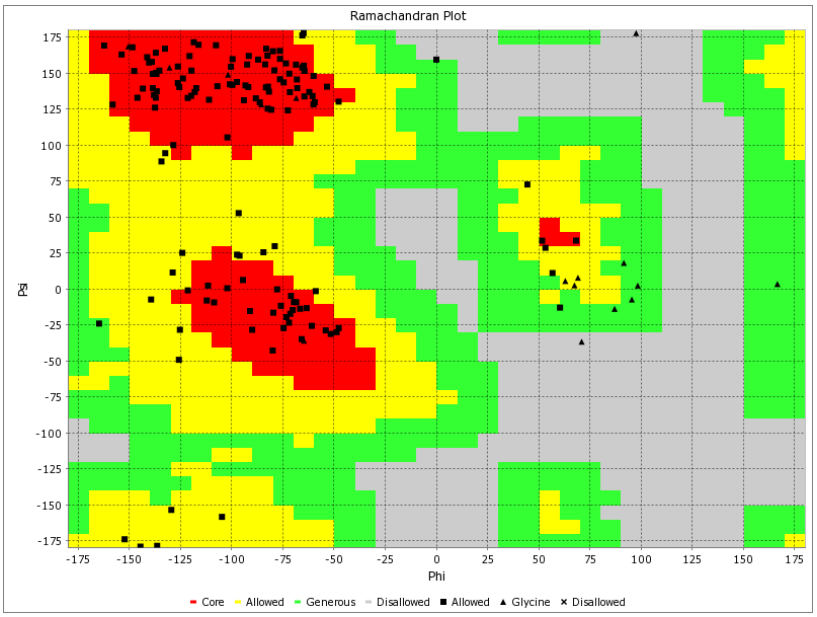


Figure 13.1 Ramachandran for inputs

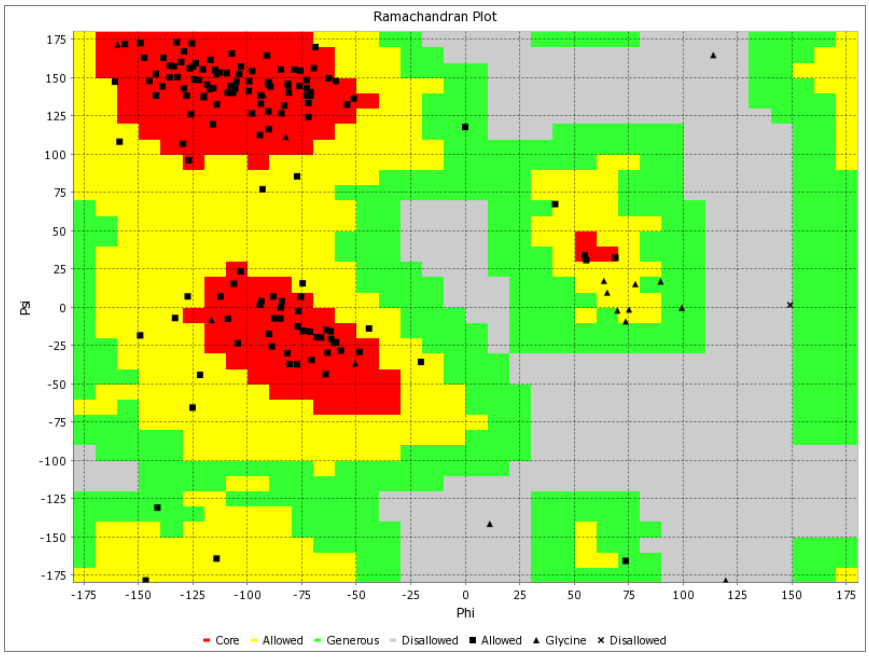


Figure 13.2 Ramachandran for outputs

7.2.2.3 1Fq9 Protein

7.2.2.3.1 Reconstructed after training

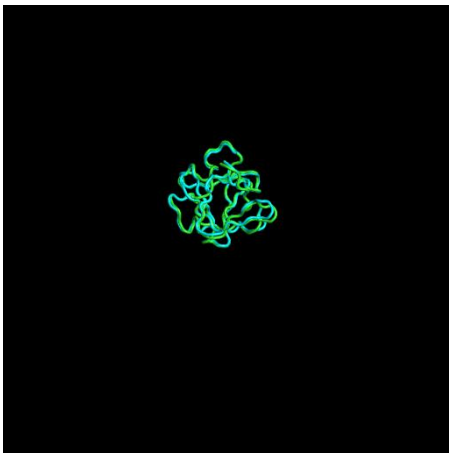


Figure 14 1Fq9_Reconstruction input and output

7.2.2.3.2 Loss Function

Loss function measures the difference between the predicted output of model and the correct output. The goal is to minimize the loss function, which means that it tries to find the best possible model that fits the given data. For 1Fq9 Protein loss function was 0.5 which is very good after training the whole data with 350 iterations.

7.2.2.3.3 *Cross Validation*

To get RMSD for all inputs and outputs after training - smaller RMSD than 2 the better it gets- for example, RMSD for input_135 and output _135 for 1Fq9 was **0.507 A**^{0}.

7.2.2.3.4 *Concatenation of TSV files for inputs and outputs*

After, we need to collect whole RMSD in one file for inputs and outputs data. The following figures show the RMSDs and its histograms.

For (fig 9.2) it shows the best (0.4 A˚ RMSD), average (0.6 A˚ RMSD), and worst (< 0.8 A˚ RMSD) models produced.

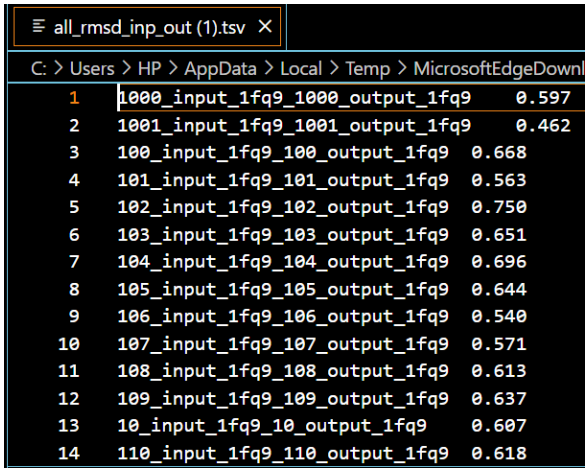


Figure 15.1 RMSD for 1Fq9 input_output

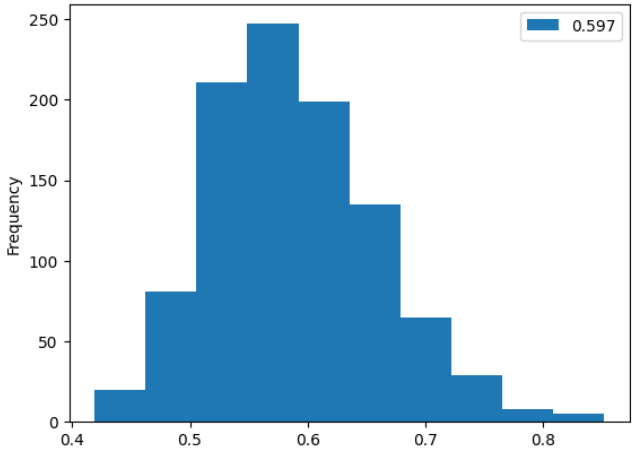


Figure 15.2 RMSD histogram for 1Fq9 input_output

7.2.2.3.5 Diversity

For diversity step, it is important for developing accurate and robust model for protein structure prediction, we also generated RMSDs between inputs_inputs and outputs_outputs. The following figures show that both have the best (0.4 RMSD), average (0.6 A˚ RMSD), and worst (< 0.8 A˚ RMSD) models produced.

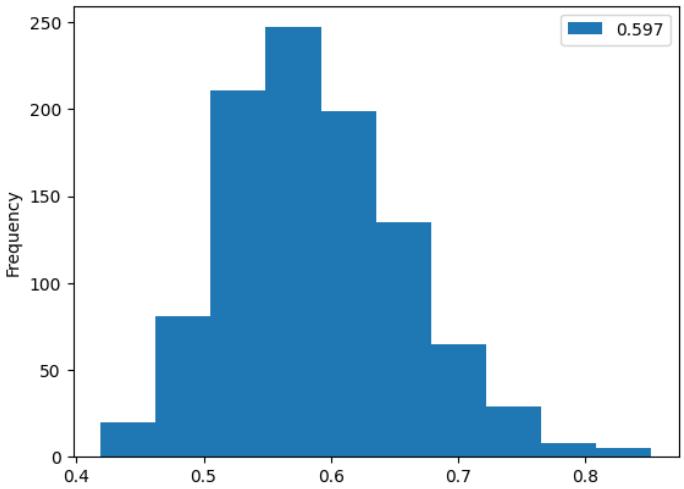


Figure 16.1 RMSD_diversity histogram for input_intput

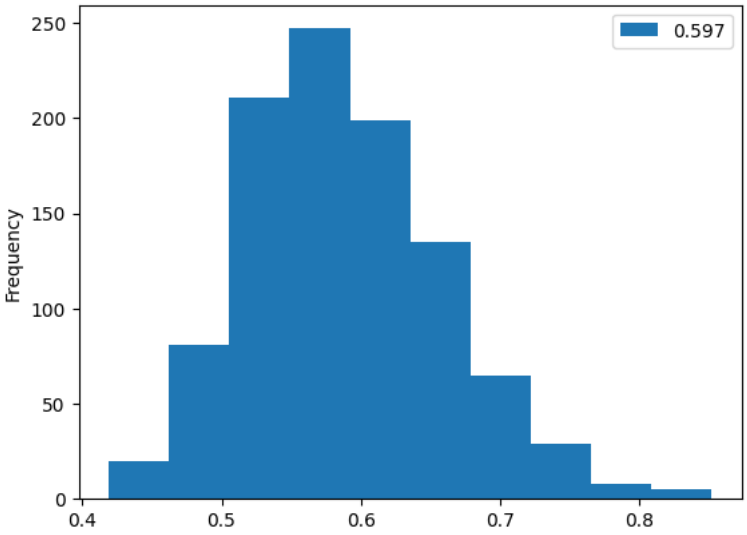


Figure 16.2 RMSD_diversity histogram output_output

7.2.2.3.6 Ramachandran Plot

Ramachandran plot is used in the development and validation of computational methods for protein structure prediction. The following figures show that input and output in alpha-helix and beta-sheets (red region) are almost the same. Points in the same area means it is good quality data.

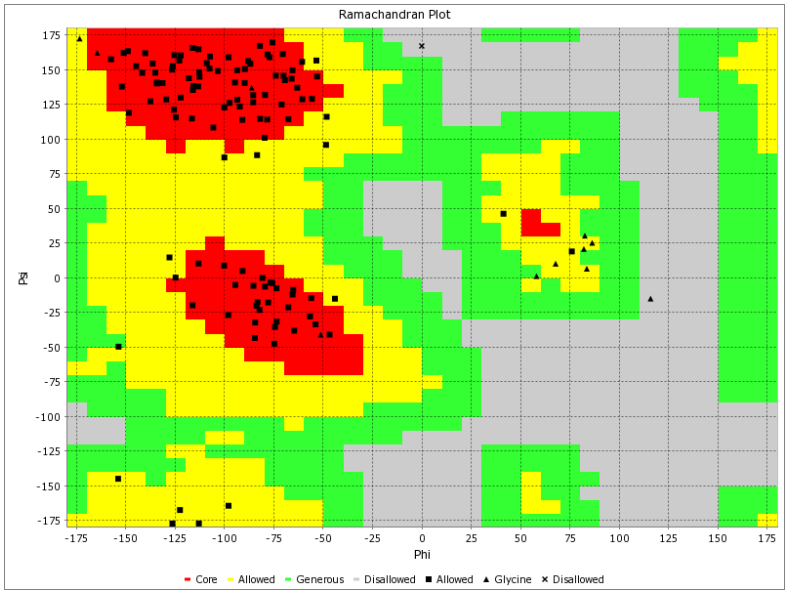


Figure 17.1 Ramachandran for inputs

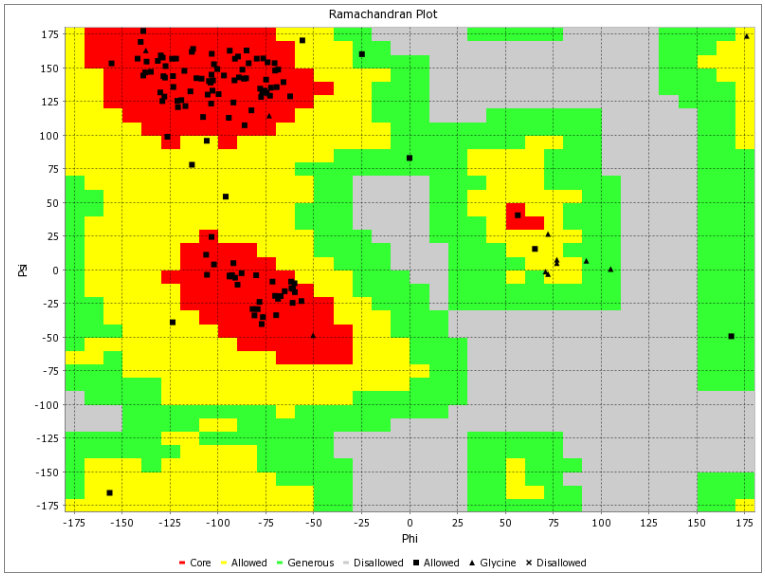


Figure 17.2 Ramachandran for outputs

7.2.2.4 2P23 Protein

7.2.2.4.1 Reconstructed after training

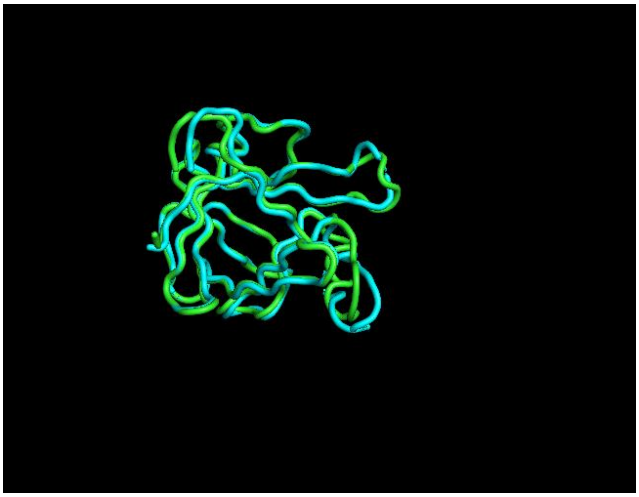


Figure 18 2P23_Reconstructed for inputs and outputs

7.2.2.4.2 Loss Function

Loss function measures the difference between the predicted output of model and the correct output. The goal is to minimize the loss function, which means that it tries to find the best possible model that fits the given data. For 2P23 Protein loss function was 0.35 which is very good after training the whole data with 350 iterations.

7.2.2.4.3 *Cross Validation*

To get RMSD for all inputs and outputs after training - smaller RMSD than 2 the better it gets- for example, RMSD for input_898 and output _898 for 2P23 was **0.87A**^{0}.

7.2.2.4.4 *Concatenation of TSV files for inputs and outputs*

After, we need to collect whole RMSD in one file for inputs and outputs data. The following figures show the RMSDs and its histograms.

For (fig 12.2) it shows the best (0.6 A˚ RMSD), average (1.0 A˚ RMSD), and worst (1.6 A˚ RMSD) models produced.

all_rmsd_inp_out (1).tsv

C: > Users > HP > Downloads > all_rmsd_inp_out (1).tsv

1	100_input_fgf19_h_2p23_100_output_fgf19_h_2p23	0.906
2	101_input_fgf19_h_2p23_101_output_fgf19_h_2p23	0.936
3	102_input_fgf19_h_2p23_102_output_fgf19_h_2p23	0.984
4	103_input_fgf19_h_2p23_103_output_fgf19_h_2p23	1.180
5	104_input_fgf19_h_2p23_104_output_fgf19_h_2p23	0.807
6	105_input_fgf19_h_2p23_105_output_fgf19_h_2p23	1.083
7	106_input_fgf19_h_2p23_106_output_fgf19_h_2p23	0.913
8	107_input_fgf19_h_2p23_107_output_fgf19_h_2p23	1.097
9	108_input_fgf19_h_2p23_108_output_fgf19_h_2p23	1.118
10	109_input_fgf19_h_2p23_109_output_fgf19_h_2p23	1.021
11	10_input_fgf19_h_2p23_10_output_fgf19_h_2p23	0.998
12	110_input_fgf19_h_2p23_110_output_fgf19_h_2p23	0.964
13	111_input_fgf19_h_2p23_111_output_fgf19_h_2p23	1.045
14	112_input_fgf19_h_2p23_112_output_fgf19_h_2p23	1.001

Figure 19.1 RMSD for 2P23 input_output

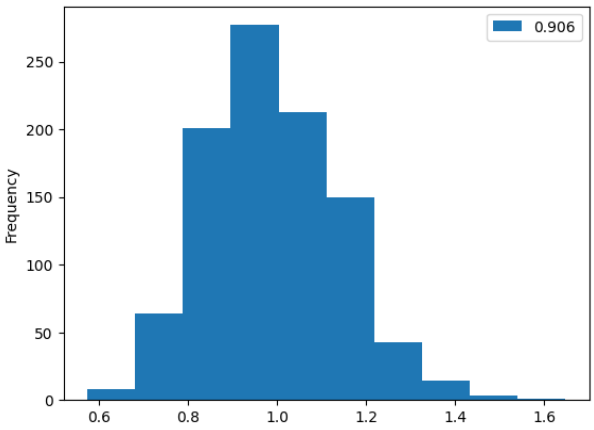


Figure 19.2 RMSD histogram for 2P23 input_output

7.2.2.4.5 Diversity

For diversity step, it is important for developing accurate and robust model for protein structure prediction, we also generated RMSDs between inputs_inputs and outputs_outputs. The following figures show that both have the best (0.6 RMSD), average (1.0 A˚ RMSD), and worst (1.6 A˚ RMSD) models produced.

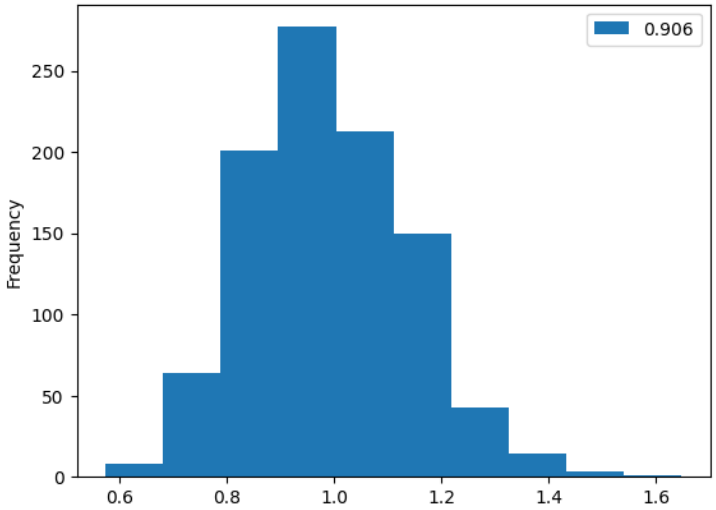


Figure 20.1 RMSD_diversity histogram for input_intput

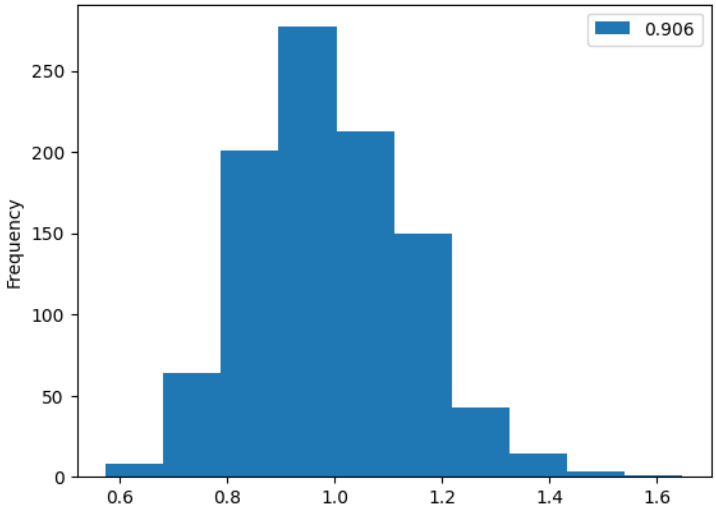


Figure 20.2 RMSD_diversity histogram output_output

7.2.2.4.6 Ramachandran Plot

Ramachandran plot is used in the development and validation of computational methods for protein structure prediction. The following figures show that input and output in alpha-helix and beta-sheets (red region) are almost the same. Points in the same area means it is good quality data.

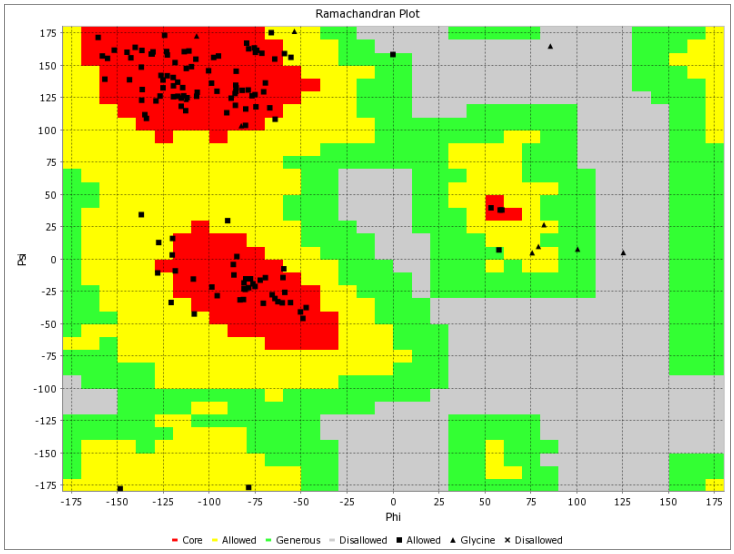


Figure 21.1 Ramachandran for inputs

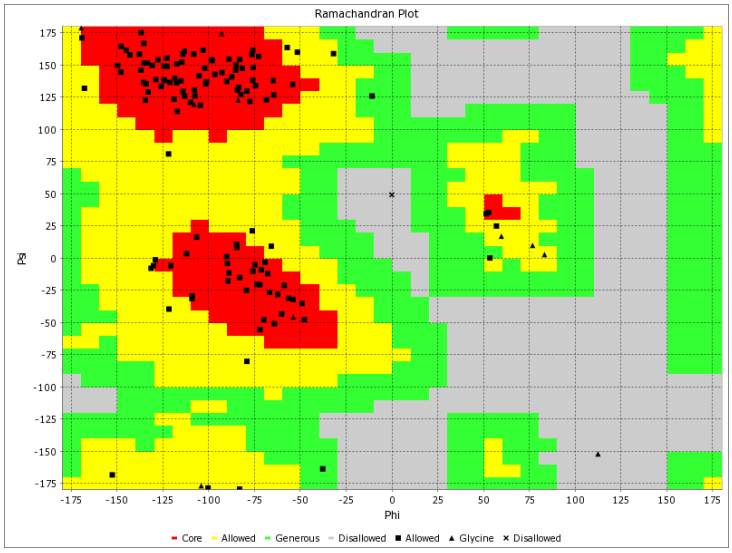


Figure 21.2 Ramachandran for outputs

7.2.2.4 DSSP "Dictionary of Secondary Structure of Proteins"

The secondary structure assignments in the DSSP output file can be used to evaluate the accuracy of computational methods that predict protein secondary structure from sequence information. Based on that, we have checked the accuracy for some proteins. Accuracy has been resulting **0.94.**

7.2.3 Model Testing

No.	Testing Method	Action Done	Expected Result	Result
1	Trajectory files extraction.	Extraction has been done correctly.	Get conformations and atoms	Pass
2	Loading data	Using dataloader and get turn it into batches	Data turned into batches and visualize gt_samples.	Pass
3	Alignment,kabsch and batch kabsch.	Each protin has been aligned and loss function has dealt with batches	Data has been aligned	Pass
4	Autoencoder	Autoencoder has been built to deal with the data.	Train over the whole conformations and get the best loss function.	Pass

Table1: Model Testing

No.	Testing Method	Action Done	Expected Result	Result
1	Cross Validation	Iterate over the input and output pairs to calculate RMSD	Get .TSV files including RMSDs for each input and output for each protein	Pass
2	Concatenation of TSV files for inputs and outputs	All .TSV files have been concatenated	Get one file including all RMSD results	Pass
3	Diversity	Applying diversity to get RMSD for inputs together and outputs together then concatenate.	Get one file after including all RMSD for input_input and output_output.	Pass
4	DSSP	Get the full atomic structure to get the 2nd structure.	DSSP files to get accuracy of some proteins	Pass

Table 2: Model Testing validation

CHAPTER FIVE: FUTURE WORKS AND CONCLUSION

8.1 Future Works

The current model has the potential for further improvement by incorporating additional modules such as:

- There is a need to develop user-friendly tools that enable researchers to easily access and leverage from this model. As a future direction, we plan to develop a web-based tool. This tool will allow researchers to easily input their proteins and receive accurate and efficient predictions of new conformations.
- Generate candidate subunits for protein docking.

8.2 Conclusion

To sum up, this project aims to replace molecular dynamics simulation with generative neural network. We look at how deep learning, specifically generative neural networks like autoencoder, can be used to improve the sampling of molecular conformational space. Recently, generative neural networks have been presented as a tool for discovering collective variables, which can be used to extract kinetic information from molecular simulations or to direct the selection of underexplored regions.

We used the conformations generated by protein MD simulations as examples to train an autoencoder. Substantial sampling is required, however, to generate accurate results that may be used to justify experimental data or forecast outcomes before trials are conducted. We have showed that how a generative neural network trained on protein structures generated by molecular simulation may be utilized to generate new, realistic conformations that complement existing ones.

REFERENCES

[1] Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. (2017). Learning representations and generative models for 3D point clouds. ArXiv, 1707.02392.

[2] Bahar, I., et al. "What is the root mean square deviation (RMSD) and how is it calculated?" Proteins: Structure, Function, and Bioinformatics 78.2 (2010): 203-214

[3] Chen, H., et al. "A review of protein structure preprocessing methods." Briefings in bioinformatics 19.2 (2018): 226-241.

[4] Chu, X., Gan, L., Wang, E., and Wang, J. (2013). Quantifying the topography of the intrinsic energy landscape of flexible biomolecular recognition. Proc. Natl. Acad. Sci. U S A 110, E2342–E2351.

[5] Li, Y., et al. "Periodic boundary conditions in molecular dynamics simulations of proteins: Artifacts, corrections, and applications." Journal of chemical theory and computation 15.12 (2019): 6327-6346.

[6] Matteo T. Degiacomi ,Department of Chemistry, Durham University, South Road, Durham DH1 3LE, UK 2Lead ,https://doi.org/10.1016/j.str.2019.03.018

[7] Wang, Y., et al. "Trajectories centering and frame selection algorithms for extracting meaningful information from molecular dynamics simulations." Journal of computational chemistry 41.24 (2020): 2378-2388.

[8] Xu, D., et al. "Backbone conformation analysis of proteins: A review." Journal of chemical information and modeling 61.1 (2021): 36-53.