# Probing Structure Disrupting Mutations in Helices through Reinforcement Learning

A thesis submitted to
Indian Institute of Science Education and Research, Pune
towards the partial fulfillment of the academic requirements
for the BS-MS Dual Degree Programme
by

**PRATHITH BHARGAV**


under the guidance of

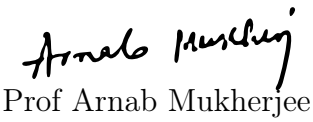
**PROF ARNAB MUKHERJEE**
**DEPARTMENT OF CHEMISTRY, IISER PUNE**
from May 2023 to Mar 2024

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,Pashan, Pune 411008, INDIA.

# Certificate

This is to certify that this dissertation entitled **Probing Structure Disrupting Mutations in Helices through Reinforcement Learning** submitted towards the partial fulfilment of the BS-MS degree at the Indian Institute of Science Education and Research, Pune represents original research carried out by Prathith Bhargav at Indian Institute of Science Education and Research Pune, under the supervision of Prof Arnab Mukherjee during academic year May 2023 to March 2024.

**Thesis Advisory Committee**
Prof Arnab Mukherjee
Dr Jagannath Mondal



Prof Arnab Mukherjee
Date: 10/05/2024

# Declaration

I, hereby, declare that the matter embodied in the report titled **Probing Structure Disrupting Mutations in Helices through Reinforcement Learning** is the result of the investigations carried out by me at the Department of Chemistry, Indian Institute of Science Education and Research Pune from the period 15-05-2023 to 15-03-2024 under the supervision of Prof Arnab Mukherjee and the same has not been submitted elsewhere for any other degree. Wherever others contribute, every effort is made to indicate this clearly, with due reference to the literature and acknowledgement of collaborative research and discussions.

Prathith Bhargav
Date: 09/05/2024

*Dedicated to my late ajji, H Vanajakshamma*

# Acknowledgements

First and foremost, I would like to thank my supervisor, Prof Arnab Mukherjee, for his mentoring, encouragement and support throughout the project. I am very thankful for the independence that I got in choosing and executing my project. I am very grateful to my expert, Dr Jagannath Mondal for his suggestions and comments. I express my thanks to all the present and past members of the Computational Chemistry and Biophysics group: Amal, Ashish, Bikirna, Mansi, Parth, Pratishruti, Rahul, Sarath, Sreyas, Sonali, Uddipan and Vishnu for providing a friendly and supportive academic environment. I express my gratitude to Ashish, Bikirna, Sonali and Uddipan for the discussions and help in implementing Metadynamics Simulations. I thank my collaborators, Ateek Shah from Whytamin Lab, IISER Pune, Pratishruti Panda and Vishnu Kadam from CCB Lab IISER Pune. For my collaborative project, I thank Dr Amrita Hazra (IISER Pune) for project co-supervision and support. I thank Dr Anirban Hazra (IISER Pune) and Dr Kausik Chakraborty (CSIR-IGIB) for their valuable suggestions. I want to acknowledge Dr Joy Monteiro (IISER Pune), who has been a supervisor of one of my other projects and has helped me navigate through my academic journey at IISER.

I acknowledge the Kishore Vaijnanik Protsahan Yojana, DST, Government of India, for the fellowship. I am grateful to the support and the resources provided by PARAM Brahma Facility under the National Supercomputing Mission, Government of India, at the Indian Institute of Science Education and Research, Pune.

Outside of academics, I am incredibly grateful to have had a fantastic set of friends. While this list is not exhaustive, I would like to thank Abhinav, Abhijith, Abhradeep, Arjun, Ashli, Atharva, Daksh, Gautham, Ronedy, Sanjay, Shreyas, Shridhar, Soham, Soumil, Sumanth, Subramanya, Vaishak, Varna, Vikram and Yash for making my student life at IISER a cherishable one. I also thank past and present members of Disha, the voluntary social organisation of IISER Pune. Experiences in Disha have made me a better and deeper thinker and I shall be forever grateful to it.

Amma and Dada have been supportive of all my decisions, and their constant support and encouragement to pursue science as a career has got me here. I want to thank my grandparents, who played a significant role in my upbringing. Ajji, with her consistent mentoring, ensured that I performed my best in whatever I did and Thatha, to this date, has ensured to engage me in all sorts of intellectual dialogues. Lastly, I would like to thank all the teachers who taught me.

# Abstract

Helices are important secondary structural motifs within proteins, formed via hydrogen bonds between amino acid main chain atoms, playing pivotal roles in numerous physiological processes. Variations in the properties of amino acid side chains confer distinct abilities to stabilise or destabilise these helical structures. While amino acids such as Alanine and Leucine are known to promote helix formation, others such as Proline and Glycine possess side chain attributes that can disrupt helical structures. The tendency of amino acids to form or break a helix can be measured through helical propensity. Despite extensive investigations into helical propensity using model peptides, a general model for predicting which amino acids can disrupt or break a helix has yet to be developed.

In tackling this challenge, we employ structural biochemistry techniques alongside reinforcement learning methodologies to develop a predictive model for helix-disrupting mutations. We start with a toy model consisting of helices with only 30 amino acids and train different models. Our results show that our model is successful in disrupting helical structures. Our results underscore the effectiveness of our approach in disrupting helical structures, highlighting the promising potential of reinforcement learning in addressing problems associated with protein structure disruption. Furthermore, this work acts as a proof of concept for similar models to be built and extended to more complex systems, such as protein mutants that cause disease.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

*This chapter is an introduction to the scientific concepts that underline the work that is presented later in this thesis. The following chapter will delve into the theoretical framework supporting the methodologies employed in this work. Useful references are given at the end of the chapter for the reader to gain a deeper understanding of concepts.*

Proteins, carbohydrates and nucleic acids, together known as biomacromolecules, are large polymers present in living organisms. Amongst these, proteins, formed as end products of transcription and translation, play crucial roles in organisms. Some proteins, like Cytochrome C Oxidase, act as enzymes that catalyse a variety of critical biochemical reactions, while others, like the Sodium Potassium Pump, hasten the transport of ions in and out of the cell. Proteins are also crucial in providing structure in living organisms—proteins like collagen and elastin form the extracellular matrix.

Proteins are formed when amino acids combine through a peptide bond, creating a polypeptide chain. Due to resonance(figure 1.1), the peptide bond acts as a partial double bond, thereby restricting rotation. Due to this restriction in rotation, a polypeptide chain can rotate only through two dihedral angles, as shown in figure1.2. While the sequence of amino acids in a protein is called its primary structure, polypeptide chains can form stretches of ordered structures called secondary structures by forming hydrogen bonds between the partially positive hydrogen of the N-H and the partially negatively charged carbonyl oxygen[1].

Broadly, two categories of secondary structures exist—the helix and the sheet. While helices form hydrogen bonds with the backbone atoms of the same regular secondary structures, strands form hydrogen bonds with other strands, forming the $\beta$ sheet. A category of helices, $\alpha$ helices, is formed due to hydrogen bonds between the $i^{th}$ and $i+4^{th}$ amino acids. Due to its structural constraints, $\alpha$ helices have an ideal set of dihedral angles, with $\phi = -57°$ and $\psi = -47°$. Consequently, alpha helices consist of 3.6 residues per turn and a pitch of 5.4 Å. A schematic $\alpha$ helix is shown in figure 1.3. While other helices, such as $3_{10}$ and the $\pi$ helices, exist in nature, they are less abundantly found than

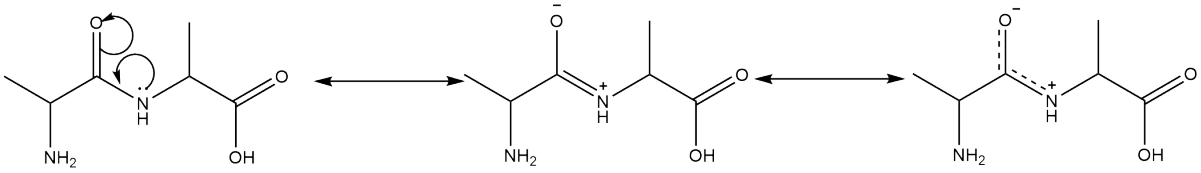


Figure 1.1: An Illustrative peptide bond between 2 alanines. Due to conjugation, the carbonyl carbon- nitrogen bond adapts a partial double bond character, thereby restricting rotation.

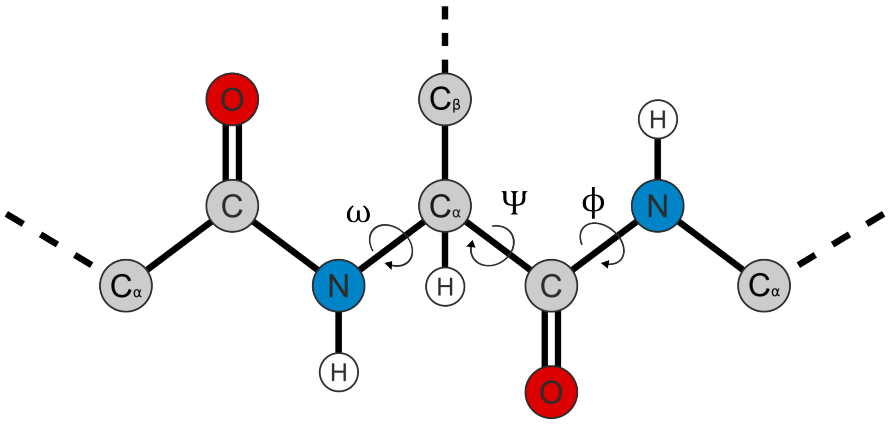



Figure 1.2: Three dihedral angles, $\omega$,$\phi$ and $\psi$. Out of these, $\omega$ does not change because of the partial bond character. $\psi$ is the dihedral angle of rotation about the C-$C_\alpha$ bond, while $\phi$ is the angle of rotation about the C-N bond

$\alpha$ helices and $\beta$ sheets.[1]

The process of protein folding leads to the creation of helices from the primary structure. Folding is driven by two factors: the negative enthalpic change accompanying the stabilisation of the structure due to hydrogen bonds and an increase in entropy resulting from the expulsion of water molecules from the peptide backbone[2]. Since polypeptides of alanine, which contains a methyl side chain, is a helix, the energetics of helix formation are favourable for the protein's main chain, and factors that hinder the formation of the helix must be introduced by side chains[3]. Since different amino acids possess different side chains, not all amino acid compositions favour the formation of a helix.

Helical propensity refers to the likelihood or tendency of specific amino acids to form an alpha-helical structure in proteins. Helical propensity has been extensively studied in peptides and proteins through theoretical and experimental methods. Pace and coworkers compiled a helical propensity of amino acids[4] based on data from 15 experiments. Their findings positioned Alanine as having the highest propensity for helix formation, with Glycine ranking the lowest, while Proline was excluded from the analysis. This result was concurrent with other studies that show a significant difference between Alanine and Glycine[5].

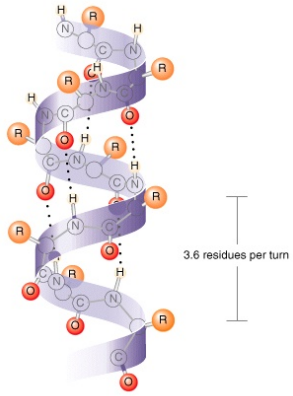



Figure 1.3: $\alpha$ helices have hydrogen bonds between the $i^{th}$ and $i+4^{th}$ amino acids, with 3.6 residues per turn

Research indicates that helical propensity is not uniform across amino acid positions within a helix. The substitution of Alanine with Glycine destabilises the helix more at the centre than at the edge of the helix[5]. Likewise, the propensity of polar amino acids is higher at the terminal region compared to the middle of the helix[6]. On the other hand, non-polar amino acids show different behaviours - Isoleucine and Valine are more favourably found in the terminal region, while Leucine and Methionine are more favourably found in the centre of the helix[7].

Helical propensity is identical between proteins and peptides[8], thus quantifying its robustness across chemical contexts. However, distinctions emerge between cytosolic and membrane helices — propensity values differ between cytosolic and membrane alpha helices. $\beta$ sheet prompting amino acids such as Isoleucine, Valine and Threonine have been found in transmembrane segments of membrane alpha helices[9]. Similarly, Proline, typically associated with the low helical propensity in cytosolic settings, exhibits different behaviour in membrane environments[10].

While some amino acids show high propensity, such as Alanine, others, such as Proline and Glycine, come at the bottom of the propensity scale and seldom are part of helices. Due to its cyclic side chain, Proline does not allow for favourable $\phi$ values for helix formation, causing kinks in the helical structure[11]. A single proline mutation is shown to cause a bend membrane-spanning helices, forming two transmembrane segments[12]. Meanwhile, due to its increased conformational flexibility, Glycine often acts as a helix terminator[13,14]. A study has also shown that an Aspartate mutation can destabilise a helical peptide at the C Terminal end while stabilising the helix on the N Terminal end[15].

Mutations and disruptions in helical structures are crucial in disease contexts as well. Most disease-causing mutations in proteins are known to disrupt secondary structures[16] A study published in 2007 by Muano Vihinen and others[16] showed that Arginine was the most mutated amino acid in $\alpha$ helices in 44 proteins associated with disease. In Membrane $\alpha$ helices, too, it has been shown that a polar amino acid substitution can create non-

native H-bonds that disrupt protein structure and function, potentially causing disease states[17].

Previous experimental studies on helical propensity and helix-disrupting mutations have primarily conducted investigations through the synthesis of various model peptides, alongside mutation introduction, to examine helical content using techniques such as X-ray crystallography, Nuclear Magnetic Resonance (NMR) spectroscopy, and Circular Dichroism spectroscopy[18]. In contrast, theoretical investigations into helical propensity have typically involved either bioinformatics analysis of existing data[16,17] or molecular dynamics simulations to assess dynamics following substitutions[19]. However, both experimental and theoretical approaches are time-intensive, and findings from bioinformatics methods may lack generalizability due to their reliance on limited data availability. Notably, there is a gap in the literature regarding a comprehensive examination of helix disruption in a generalized context. This is particularly important since alpha helices are structurally robust to mutations compared to other secondary structures[20].

Recent advances in deep learning have led to the development of accurate ultra-fast methods to predict the structure of a protein from a sequence. Through software such as AlphaFold[21] and ESMFold[22], it is now possible to computationally obtain the structure quickly. Once structures are obtained, obtaining the secondary structure content is straightforward using tools like DSSP[23], P-SEA[24] or other machine learning techniques[25,26]. These developments allow us to potentially conduct mutational studies on proteins followed by structural analysis in a fast and low-resource manner. Others have attempted such studies specifically to study the effects of variants on protein structure[27–29].

Reinforcement Learning (RL) is a subset of machine learning where an agent learns to make decisions by interacting with an environment, receiving feedback, and refining its actions accordingly[30]. In the field of protein science, RL has been employed, particularly in protein design tasks. EvoPlay is a self-play RL framework developed to aid protein engineering endeavours that operates by iteratively mutating individual amino acid residues to optimize protein sequences. Utilizing Monte Carlo tree search techniques, EvoPlay efficiently explores the design space and has been shown to predict peptide binders with heightened affinity and proteins exhibiting improved fluorescence properties[31].

In light of this context, we develop a predictive model to identify mutations that disrupt helical structures. Our approach integrates methodologies from protein structure prediction, secondary structure analysis, and reinforcement learning. Inspired by existing frameworks[31,32], we conceptualize protein sequences as states, mutations as actions, and reward mutations that disrupt helices. Through comprehensive training on both cytosolic and membrane helix datasets, we demonstrate the efficacy of our model in accurately predicting disruptive mutations within helical regions. We show that our model effectively predicts mutations that disrupt all helices, those with high and low average propensity.

Utilizing well-tempered metadynamics simulations, we demonstrate the stability of the mutated structures.

## Useful Readings to understand topics in depth

1. Voet, Donald, and Judith G. Voet. *Biochemistry*. John Wiley & Sons, 2010. will provide context on basic biochemistry required to understand protein secondary structures.

2. Bertoline, Letícia MF, et al. "Before and after AlphaFold2: An overview of protein structure prediction." *Frontiers in Bioinformatics* 3 (2023): 1120370. will provide information on protein 3d structure prediction methods.

3. Hugging Face Deep RL Course. https://huggingface.co/learn/deep-rl-course/ is a useful resource to understand and implement reinforcement learning algorithms, which shall be briefly explained in the following chapter.

# Chapter 2

# Theory

*This chapter covers the theory behind the methods utilised in this thesis in chapters 3.*

## 2.1 Artificial Intelligence, Machine Learning and Reinforcement Learning

Artificial intelligence (AI) refers to technologies that perform tasks requiring human-like intelligence. For instance, software like ChemDraw can identify deviations from chemical valency rules, such as drawing a carbon with five bonds. ChemDraw operates on rule-based algorithms, adhering to predefined rules of chemistry. Thus, ChemDraw is an AI but not a machine learning algorithm (ML) since it does not "learn" from data but operates based on programmed rules.

Meanwhile, ML involves systems learning from data to make predictions or decisions. An example of an ML would be using curve fitting to validate Beer Lambert's law at low concentrations. Software like Origin or SciDavis learns the relationship between concentration and absorbance values to determine the slope and intercept. It is important to note, that all ML is AI, since all ML perform tasks that would otherwise require human intervention. On the other hand, all AI is not ML, since not all AI "learn" from data.

Reinforcement Learning (RL) represents a subset of ML that focuses on decision-making to maximize rewards in an environment akin to learning to play a game. Unlike Machine learning (where training data has labelled answers), RL agents learn by trial and error. They receive feedback after each action, helping them determine whether their choices were correct, neutral, or incorrect. RL agents have been previously used to train computers to play games and in the chemistry context, used to create models that can optimise reaction routes in organic synthesis[33].

## 2.2 Theory of Reinforcement Learning

To motivate the theory of reinforcement learning, we introduce a hypothetical game where the agent is a rat, and its goal is to navigate around a room and eat some cheese while being careful enough not to be eaten by the cat, which is its predator. So, in this game, the rat is the agent, the room where it lives is the environment, and the feedback it gets from the environment is whether it gets food or gets attacked.

To formalise this game mathematically, we can consider the room a discrete one-dimensional space. We can now consider only two possible actions: the rat can move left or right. For the sake of simplicity, we can assume that neither cat nor cheese moves. Mathematically, such a scenario is described using the Markov Decision Process framework. A Markov Decision Process $\mathcal{M}$ consists of a set of States $\mathbf{S}$, actions $\mathbf{A}$ and rewards associated with being in those states $\mathbf{R}$. In the scenario of the rat moving around, we can assume the following Markov Decision Process (MDP):

$$\mathbf{A} = \{\rightarrow, \leftarrow\} \tag{2.1}$$

$$\mathbf{S} = \{1, 2, 3, 4, 5, 6, 7\} \tag{2.2}$$

$$\mathbf{R} = \begin{cases} +100 & \text{if state has cheese} \\ -100 & \text{if state has cat} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Visually, this is shown in figure 2.1. If the rat successfully reaches the cheese, it receives a positive reward, the desired outcome. On the other hand, if it goes towards the cat, it receives a highly negative undesirable reward. Since our goal is to make the rat get the desirable outcome or the positive reward, we might think that the goal of Reinforcement Learning is to learn how to *maximise* the total reward $\mathbb{R}$.

$$\mathbb{R} = R_t + R_{t+1} + R_{t+2} + ... \tag{2.4}$$

where $t$ refers to the number of time steps. However, maximising the total reward $\mathbb{R}$ might not be sufficient in some scenarios, as demonstrated in the example below.

If the rat is in state 4 at time $t$ ($S_t = 4$), as shown in figure 2.1, it can take an action $\rightarrow$ to go to state 5. In this scenario, the rat receives an immediate reward ($R_t$) of 0 according to the reward function. It can either head straight towards the cheese and get a reward of 100 at $t+3$, or it can take a longer route, possibly with some back and forth, and still reach the cheese at $t+5$ to receive the same reward of 100. The total reward ($\mathbb{R}$) remains constant regardless of the number of steps taken. In this game, the rat might want not to waste much time to find the cheese. So now the goal is slightly modified to


|  | | | |  | | |  |
|---|---|---|---|---|---|---|---|
| Reward | -100 | 0 | 0 | 0 | 0 | 0 | 100 |
| State | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Figure 2.1: Illustration of Markov Decision Process described in equation 2.1

get the positive reward **fast**. To account for this modified goal, we use the sum of the discounted rewards or **return** $G$ as our quantity to be maximised. $G$ is defined as:

$$G(\tau) = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ... \tag{2.5}$$

where $(\gamma)$ is the discount factor used to weight future rewards and $\tau$ represents the trajectory of state-reward pairs, $\{(s_t, R_t), (s_{t+1}, R_{t+1})...\}$. By discounting rewards achieved later in the trajectory, the algorithm gives more weight to immediate rewards. For instance, if the mouse went from $s = 4$ to $s = 7$ in 3 steps, assuming a $\gamma = 0.95$ the $G$ would be

$$G(\tau_1) = 0 + 0 \times 0.95 + 100 \times 0.95^2 = 90.25 \tag{2.6}$$

Whereas if it took longer, for instance, if it reached the cheese by the 4th step, the $G$ would be

$$G(\tau_2) = 0 + 0 \times 0.95 + 0 \times 0.95^2 + 0 \times 0.95^3 + 100 \times 0.95^4 = 81.4 \tag{2.7}$$

thus $G(\tau_2)$ is preferred over $G(\tau_1)$.

To make the model learn, we wish to get to a function that maps state to action to maximise the return, $G$. Such a function is called a policy, represented as $\pi_\theta : s \rightarrow a$, denoted so, since $\pi$ is parameterised by $\theta$ where $\theta = \{\theta_1, \theta_2....., \theta_n\}$. In the work done in this thesis, we approximate this policy, $\pi$, using a deep learning method called a neural network.

To ensure that our policy function selects actions that maximize $G(\tau)$, we need to maximize the average value of the return or the expectation value of the return. In this scenario, the expectation of $G$ is called an optimization function, denoted as $J$, denoted as:

$$J_\theta = \mathbb{E}[G(\tau)] \tag{2.8}$$

$$J_\theta = \sum P(\tau)G(\tau)(\text{definition of expectation}) \tag{2.9}$$

Where $P(\tau)$ is the probability of taking a trajectory $\tau$. Note that the optimisation function is also parameterised by $\theta$. To maximise $J$, we need to change each of our parameters $\theta$ toward the steepest ascent. We achieve this through an iterative process called gradient ascent. Mathematically, this step can be written as :

$$\theta = \theta + \alpha \nabla_\theta J_\theta \tag{2.10}$$

$$\theta = \theta + \alpha \nabla_\theta \sum P(\tau)G(\tau) \tag{2.11}$$

Now, let us consider only the gradient of the equation 2.11. We can further simplify it to:

$$\nabla_\theta J = \nabla_\theta \sum P(\tau)G(\tau) \tag{2.12}$$

$$\nabla_\theta J = \sum \nabla_\theta P(\tau)G(\tau) \text{ (property of derivatives)} \tag{2.13}$$

We now multiply and divide by $P(\tau)$ , and simplify the expression

$$\nabla_\theta J = \sum P(\tau)\frac{\nabla_\theta P(\tau)}{P(\tau)}G(\tau) \tag{2.14}$$

$$\nabla_\theta J = \sum P(\tau)\nabla_\theta \log P(\tau)G(\tau) \tag{2.15}$$

$$\nabla_\theta J = \mathbb{E}[\log P(\tau)G(\tau)] \tag{2.16}$$

However, we take an empirical approximation of equation 2.16, and the update is done as follows:

$$\theta = \theta + \alpha \log P(\tau)G(\tau) \tag{2.17}$$

The learning rate, known as $\alpha$, dictates how much to update each parameter in the direction of the steepest ascent. This value becomes essential when we consider whether the model ends up learning or, in other words, converging.

## 2.3 Mode Collapse and Entropy Regularisation

At times, when a model achieves convergence, it may learn the easiest route to the answer instead of the best one. This means the optimization function has been updated to a local maximum instead of a global maximum. This situation is known as a mode collapse. We encounter this situation in the results described in the forthcoming chapter, and several methods are available to mitigate it. These methods are referred to as regularisation techniques. One such technique is Entropy Regularisation.

The first step to performing an Entropy Regularisation is calculating the information entropy from the predictions of the policy function. Information entropy $H$ can be described using the following expression:

$$H = -\sum P_i \log P_i \tag{2.18}$$

Where $P_i$ is the probability of taking action $i$, where $i \in A = \{0, 1, 2, ...\}$ This expression encourages exploration and prevents the system from converging into an overly deterministic policy.

We now modify 2.17 to include the entropy regularisation to get:

$$\theta = \theta + \alpha \log P(\tau)G(\tau) - \beta H \tag{2.19}$$

Where $\beta$ is called the entropic weight, by adding the negative entropy of the policy distribution to the objective function, we encourage the agent to explore different actions instead of consistently selecting the one it considers optimal. This technique can help the agent to develop a more robust and adaptive policy.

## 2.4 Understanding Dynamics of Molecules

Molecular Dynamics (MD) is a technique used to simulate the movement of particles within a system over time. Interactions in the system are determined using a Force Field created to fit known chemical properties[34]. At every time step, different potentials are computed, and by employing Newton's laws of motion($m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = \vec{\boldsymbol{F}}$), MD computes the positions and velocities of all particles at each time step. Among other quantities, Molecular Dynamics enables the calculation of various thermodynamic properties, including the Gibbs Free Energy ($G$), providing valuable insights into the system's behaviour.

One of the major hurdles in Molecular Dynamics simulations is the study of rare events that contain states separated by energy barriers. Sampling different states of the system in such cases comes at a high computational cost. To address this issue, enhanced sampling methods have been developed to accelerate the dynamics[35]. One such enhanced sampling technique is Well-Tempered Metadynamics, which is explained below.

Collective Variable based enhanced sampling methods, like the Well-Tempered Metadynamics, are used to enhance the sampling of slow processes. These methods employ collective variables, which are a low-dimensional mapping of the atomic coordinates of the system, to capture different states of the processes we are interested in. In Well-Tempered Metadynamics, a periodic Gaussian potential bias ($V(s,t)$) is added at every time step ($t$) to the system's potential energy along a selected collective variable $\mathbf{s}$. This bias helps the system overcome energy barriers. The following equation shows that the bias's height changes with time.

$$V(s,t) = \sum_{k_h\tau_{in}<1} W(k_h\tau_{in}) \exp\left[-\sum_{i=1}^{d} \frac{(s_i - s_i^0(k_h\tau_{in}))^2}{2\sigma_i^2}\right] \tag{2.20}$$

Here, $\tau_{in}$ is the time interval between bias depositions, $k_h$ is the number of Gaussians deposited, $W(k_h\tau_{in})$ is the height of the Gaussian deposited at time $t = k_h\tau_{in}$, $d$ is the dimensionality of the CV space $\mathbf{s}$, $s_i^0$ is the location where the last hill has been deposited, and $\sigma_i$ is the width of the Gaussian along the $i$-th component of the CV space. Gradually increasing the bias helps the system to explore the underlying configurational landscape and estimate the free energy surface[36].

# Chapter 3

# Methods

## 3.1 Reinforcement Learning Environment

As mentioned earlier, our goal is to identify the mutations required to disrupt a helical structure in a protein. To achieve this, we utilize a Markov Decision Process to model protein mutations. In this model, the agent is a 30 amino acid peptide, the peptide sequence is the states of the agent, mutations as actions, and rewards are assigned based on whether the sequence corresponds to an alpha helix or not. To create the reinforcement learning environment, we use Farama Gymnasium[37]. We describe each part of the Markov Decision Process and Reinforcement Learning environment in detail below:

### 3.1.1 States

We model the states of the peptide as sequence representations. We use a continuous distributed representation of the protein sequence as the state in our model. We use ProtVec[38] to generate these representations. ProtVec uses a Natural Language Processing 3-gram model trained on $>$ 500000 sequences of the SwissProt. ProtVec has been shown to be successful in various protein-related deep-learning tasks. Given any sequence, ProtVec supplies three $(100 \times 1)$ vectors for any sequence. We represent the state as the average of these three vectors.

### 3.1.2 Actions

We model single-point mutations to the peptide as actions. In our case, there are 600 actions (30 positions possible $\times$ 20 amino acids mutations). We also consider a scenario where Proline mutations are not allowed, wherein there are 570 actions (30 positions possible $\times$ 19 amino acids mutations).

### 3.1.3 Rewards

We obtain rewards through the following steps:

1. **Obtaining mutated structure:** We use ESMFold[22] to generate mutated structures. ESMFold is accurate and ultrafast compared to existing structure prediction software such as AlphaFold2 and uses just the protein sequence to generate the structure unlike AlphaFold, which needs a Multiple Sequence Alignment.

2. **Obtaining secondary structural information:** We use the P-SEA algorithm implemented by the Biotite[39] package. P-SEA[24] assigns each amino acid in the peptide structure to a helix, strand or coil. Initially, we consider structure alpha helices in nature. We assume the structure is disrupted when the alpha-helical disruption (henceforth denoted as $\Delta\%$) goes above a cutoff. We calculate $\Delta\%$ as

$$\Delta\% = \frac{\text{total number of residues } - \text{number of residues allocated as a helix}}{\text{total number of residues}} \times 100 \tag{3.1}$$

3. **Reward Function**: The reward function is described in the equation below. The thresholds that we consider are 70,60 and 50. These scenarios correspond to 70%, 60% and 50 % disruption of the structure, respectively. When, after 15 mutations, the model does not achieve disruption, the episode ends.

$$R(s) = \begin{cases} 10 & \text{if number of mutations} < 15 \text{ and } \Delta\% \geq \text{threshold} \\ -0.01 & \text{if number of mutations} < 15 \text{ and } \Delta\% < \text{threshold} \\ -25 & \text{if number of mutations} = 15 \text{ and } \Delta\% < \text{threshold} \end{cases} \tag{3.2}$$

### 3.1.4 Policy Function Approximation

We approximate the policy function as a deep neural network. We use a Fully Connected Neural Network to predict mutations in a given sequence. We use an input layer with 100 neurons, a single layer containing 128 neurons and an output layer that contains the number of allowed mutations(600 in the Proline Case and 570 in the No-Proline Case). After the hidden layer, we use the *ReLu* action function defined as:

$$ReLu(x) = \max(0, x) \tag{3.3}$$

After the output layer, we use Softmax($\sigma$), defined as:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{3.4}$$

Where $K$ is the number of classes, $x_i$ is the raw score for class $i$.

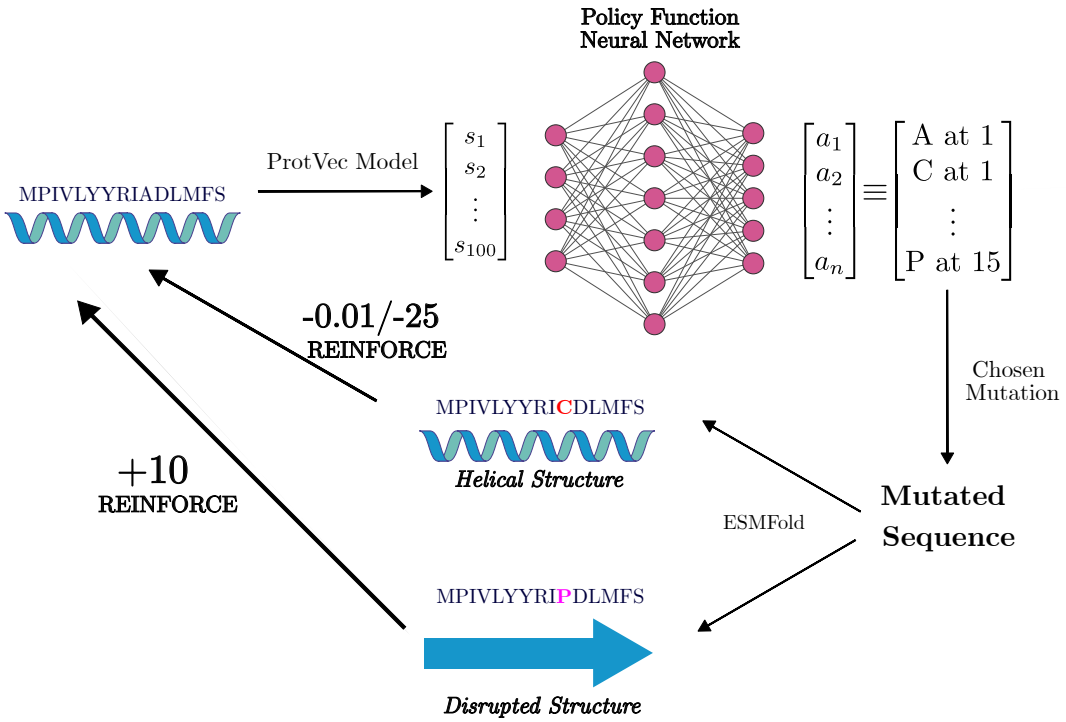



Figure 3.1: A Schematic of the Reinforcement Learning Environment for a sequence length of 15 residues

The output that we get is a set of probabilities, and the action is chosen by drawing from a multinomial distribution defined by the probabilities. We implement this algorithm using PyTorch[40].

A schematic of the RL Environment is given in figure 3.1

## 3.2 Reinforcement Learning Algorithm

We use the REINFORCE[41] algorithm with entropy regularisation, as described in Chapter 2. We update the parameters after every episode and perform gradient descent using the Adam[42] optimiser. We choose the learning rate $\alpha = 0.0007$ and the discount factor $\gamma = 0.95$. We vary the entropic factor $\beta$ and the percentage disruption. (More details in chapter 4).

### 3.2.1 Collecting Datasets

To train our model, we need sequences that are alpha helices in nature. Accounting for the possible differences in helical propensity due to the environment, we create 2 datasets. The details of the two datasets are given below:

#### TP Dataset - All Helices

We use the therapeutic peptide (TP) database[43] to construct this helix dataset. The therapeutic peptide database has been constructed by obtaining helical motifs from all proteins deposited in the Protein Data Bank. It has information related to the propensity of the helix and the number of contacts. We download the entire TP database and filter helices, which are 30 amino acids long. We further drop sequence duplicates to obtain a non-redundant dataset of 2066 helices. We then query the TP database (https://dyn.life.nthu.edu.tw/design) to obtain the initial structures. This dataset contains helices present in both cytosolic and membrane proteins.

#### Membranome Dataset - Single Spanning Transmembrane Helices

We use the Membranome[44], a dataset containing single-spanning transmembrane helices, meaning that the transmembrane helix is composed of a single helix. We download the Membranome Dataset (https://membranome.org/download ) and filter peptides that are 30 amino acids long. We further choose only those peptides modelled as a helix by ESMFold. We obtain 430 helical structures.

### 3.2.2 Train - Validation Split in Dataset

We split the **TP** dataset into train validation sets to obtain 1874 helices for training and 192 for validation. We do not split the **Membranome** dataset and use it only for training.

### 3.2.3 Training

The model was trained with different learning rates($\alpha$), discount factors($\gamma$) and different $\Delta$%(50, 60 and 70) and different $\beta$. While the maximum possible reward was 10, we define convergence as achieved when the rolling average of rewards exceeds 5. To obtain convergence, we train the model that allows Proline mutations for 8000 episodes, whereas the model that does not allow Proline mutations for 14000 episodes.

### 3.2.4 Validation and Validation Metrics

We validate the model by evaluating predictions made by the model on the validation split of the TP Dataset. We save structures produced post mutations and stop when the $\Delta$% breaches the threshold for the model. We further analyse the mutations that the model suggests for each protein in the validation set. We compute the following metrics that we define to choose the best model:

### Learning Efficiency ($\kappa$)

We consider the mean total episode reward $\hat{R}$ and obtain the learning efficiency, $\kappa$, using the following formula.

$$\kappa = \frac{\hat{R} + 25.14}{35.14} \tag{3.5}$$

$\kappa$ =1 when $\hat{R}$ is 10.0, the maximum possible total episode reward and $\kappa = 0$ when the $\hat{R}$ is -25.14, the least possible total episode reward. This quantity measures whether the model has learnt and not overfit for the training dataset.

### Mutation Entropy ($S$)

After 500 episodes of validation, we compute the frequencies of each amino acid that shows up in episodes that have been successful in structure disruption. We then calculate their frequencies and obtain the mutation entropy by

$$S = -\sum_{\text{all amino acids}} f(\text{amino acid}) \log f(\text{amino acid}) \tag{3.6}$$

where $f$ means the frequency. $S$ will give us an idea of whether there is a diversity in the amino acid mutations that the model is proposing. A higher $S$ indicates that the model suggests different amino acid mutations.

## 3.3 Free Energy Calculations Protocol

We conducted all-atom simulations of our systems in an explicit water solvent at pH 7.0, employing the AMBER99SB-ILDN[45] force field coupled with the TIP-3P water model[46]. Energy minimization was achieved through the Steepest Descent algorithm over 50000 steps. Subsequently, NVT and NPT equilibration runs were performed, followed by 5 ns NPT MD simulations at 300 K utilizing a Velocity Rescale[47](modified Berendsen) thermostat and Parrinello-Rahman barostat[48].

For Well-Tempered Metadynamics simulations using GROMACS[49] version 2021.4 patched with PLUMED 2.8.0[50], we biased our system along the collective variable **ALPHARMSD**. **ALPHARMSD** is a collective variable that probes the alpha helical content of a protein structure. It calculates the RMSD (Root Mean Square Deviation) distance between the configuration of six-residue segments within the protein and an idealized alpha helical structure. For a sequence of 30 amino acids, an **ALPHARMSD** value close to 0 implies that there is hardly any residue that is part of a helix, while a value closer to 30 implies that all residues in the peptide sequence are part of a helix.

To determine the appropriate $\sigma$ value, we conducted a brief 5 ns unbiased simulation. The values of $\sigma$ were selected based on one-third of the standard deviation of the col-

| | $\sigma$ |
|---|---|
| 1zik WT | 1.187 |
| 4k1c WT | 0.655 |
| 4k1c Mut | 0.031 |
| 1zik Mut | 0.210 |

Table 3.1: Values chosen for $\sigma$ for Well Tempered Metadynamics simulations

lective variable **ALPHARMSD** obtained from the unbiased simulations (refer to Table 3.1 for the chosen $\sigma$ values). We executed 200 ns long simulations for Well-Tempered Metadynamics with an initial bias factor of 10 and $\sigma$ values as determined previously. Convergence was observed across all simulations.

# Chapter 4

# Results

## 4.1 Training and Validating the Basic Model

To check for convergence, we train the reinforcement learning models setting $\Delta\% = 50$, 60, 70, on the TP training set(dataset with all helices, see chapter 3). These values correspond to 50 %, 60% and 70 % helix disruption. The convergence plots are given below in figure 4.1(a), demonstrating that the model has learned. Notably, it takes longer for the training corresponding to $\Delta\% = 70$ to converge.

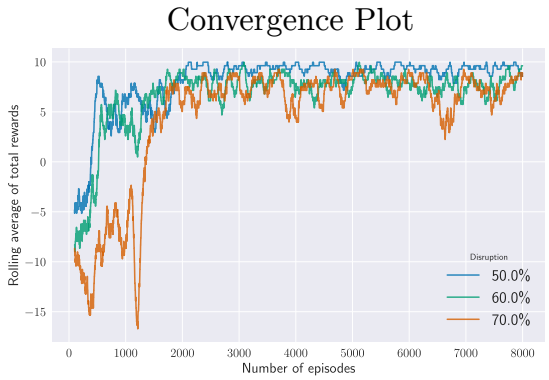
Upon observing convergence, we validate the model by running the predictions on the validation set for 500 episodes. We record metrics such as Learning Efficiency ($\kappa$), Mutation Entropy ($S$), average reward and the average number of mutations required for disruption(see section 3.2.4 for more details ). These metrics are summarised in the table 4.1.

| Percentage Disruption ($\Delta\%$) | Learning Efficiency ($\kappa$) | Mutation Entropy ($S$) | Average number of mutations required for disruption |
|---|---|---|---|
| 70 | 0.951 | 0.747 | 5.336 |
| 60 | 0.959 | 0.598 | 4.485 |
| 50 | 0.993 | 0.057 | 3.813 |

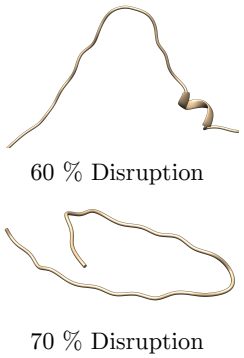
Table 4.1: Validation Metrics for Training on TP Dataset

We observe from table 4.1 that while the $\kappa$ (Learning Efficiency) values are high and the average number of mutations is small, the model shows small values of $S$, implying that the model is probably exercising a limited number of mutations. To investigate this further, we analyze the frequencies of each amino acid mutation suggested by the model and plot the landscape of the substitutions that have caused structure disruption.

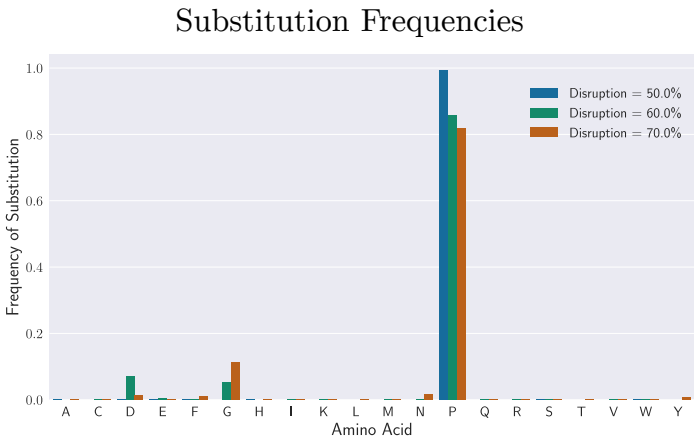
Figure 4.1 (c) shows that the model mode collapses or learns the most straightforward answer: mutating residues to proline. It is still successful in its disruption, except



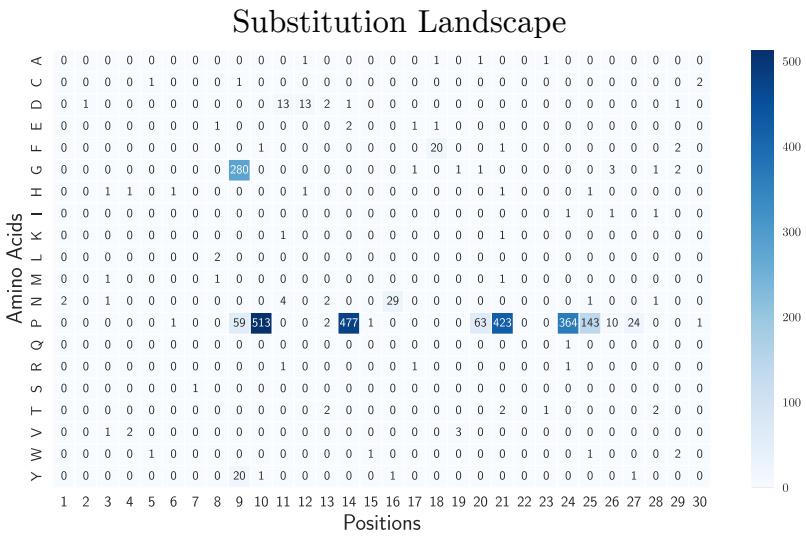

(a)




(b)




(c)




(d)

Figure 4.1: (a) Convergence Plot (b) Structures of Mutated Sequences (c) Substitution Frequencies and (d) Substitution Landscape as predicted by the Basic Model trained on the TP Dataset

that its repertoire of mutations is limited. Figure 4.1(b) shows the disrupted structures(corresponding to 60 and 70 % disruption) of the alpha helix of the protein human argininosuccinate (PDB ID: 1aos (323-352)), both having a characteristic bend associated with a Proline in the middle. We also observe Glycine mutations as we increase the $\Delta\%$, concurring with existing literature about disrupting mutations. The Substitution Landscape (4.1 (d)) shows where these mutations occur. While Proline prefers some positions to mutate, Glycine prefers position 9. While these are preliminary results, they corroborate with existing literature, thus confirming that our model has the potential to work. Hence, we move on to train a model that can predict diverse mutations by employing strategies discussed in the upcoming section.

## 4.2 Combating Mode Collapse

We use a Regularisation technique called Entropy Regularisation, which encourages the model to learn diverse actions to prevent mode collapse (see chapter 2 for details). By adjusting the entropic factor, $\beta$, which determines the significance of information entropy during the learning process, we can train our model to recognize a variety of mutations. To achieve this, we set $\beta$=0.001 and train the models accordingly. Once the models are trained, we assess their accuracy by testing them on the validation dataset and evaluating their ability to predict mutations that disrupt the structure. Upon examining the validation metrics (see table 4.2 ), while the models have learned, we observe that the Mutation Entropy ($S$) has not considerably altered. To investigate this, we plot the mutation landscape (see figure 4.2 ). While different amino acids might not have been chosen, Proline substitutions are distributed across positions due to the entropy effects, reflected in the $S$ value.

## 4.3 Effect of Helix Environment on Training

As discussed in chapter 1, helical propensities are shown to differ between cytosolic and membrane helices. To investigate this better through our model, we choose the single-spanning membrane helices, henceforth called the **Membranome** dataset(see chapter 3 for more details). We train on the Membranome dataset with and without Entropy Regularisation ($\beta = 0$ and $\beta = 0.001$). We further validate these models by running the predictions and computing the metrics. These metrics are shown in table 4.3.

Table 4.3 shows that the learning is still efficient, with comparable $\kappa$ values with and without entropy regularisation. However, to our surprise, the Mutation Entropy $S$ does not seem to be affected. This is possible because while the options for mutations have increased during training, the model has learnt to mutate the same amino acid at different positions. Aspartate is mutated at different positions, as observed in supplementary figure

| $\beta$ | Percentage Disruption ($\Delta\%$) | Learning Efficiency ($\kappa$) | Mutation Entropy ($S$) | Average number of mutations required for disruption |
|---|---|---|---|---|
| 0.0 | 70 | 0.951 | 0.747 | 5.336 |
| 0.0 | 60 | 0.959 | 0.598 | 4.485 |
| 0.0 | 50 | 0.993 | 0.057 | 3.813 |
| 0.001 | 70 | 0.957 | 0.719 | 5.436 |
| 0.001 | 60 | 0.981 | 0.056 | 4.466 |
| 0.001 | 50 | 0.993 | 0.015 | 3.491 |

Table 4.2: Validation Metrics for Training on TP Dataset with and without Entropy Regularisation

C.1. Upon inspection, we observe, through plotting the Substitution Frequencies, that new amino acids are being substituted. We see an increase in the substitution frequencies of polar amino acids such as Aspartate, Glutamate, Asparagine and Arginine. This is further observed upon entropy regularisation as seen in figure 4.3. This is noteworthy since membrane proteins are known not to contain polar amino acids[17]. However, while the model has been trained differently, we still observe a mode collapse to Proline, and in one case (4.3(a), Disruption = 50 %), Glycine. However, the model that has mode collapsed to Glycine has very low $\kappa$ value, indicating that it has not learned efficiently. Hence, while we have employed entropy regularisation to encourage diverse actions by the model, we have not fully mitigated mode collapse.

| $\beta$ | Percentage Disruption ($\Delta\%$) | Learning Efficiency ($\kappa$) | Mutation Entropy ($S$) | Average number of mutations required for disruption |
|---|---|---|---|---|
| 0.0 | 70 | 0.881 | 0.951 | 5.594 |
| 0.0 | 60 | 0.959 | 1.283 | 5.354 |
| 0.0 | 50 | 0.066 | 0 | 1 |
| 0.001 | 70 | 0.941 | 0.822 | 5.34 |
| 0.001 | 60 | 0.939 | 1.038 | 4.751 |
| 0.001 | 50 | 0.993 | 0.852 | 3.948 |

Table 4.3: Validation metrics for training on the Membranome dataset without and with entropy regularisation

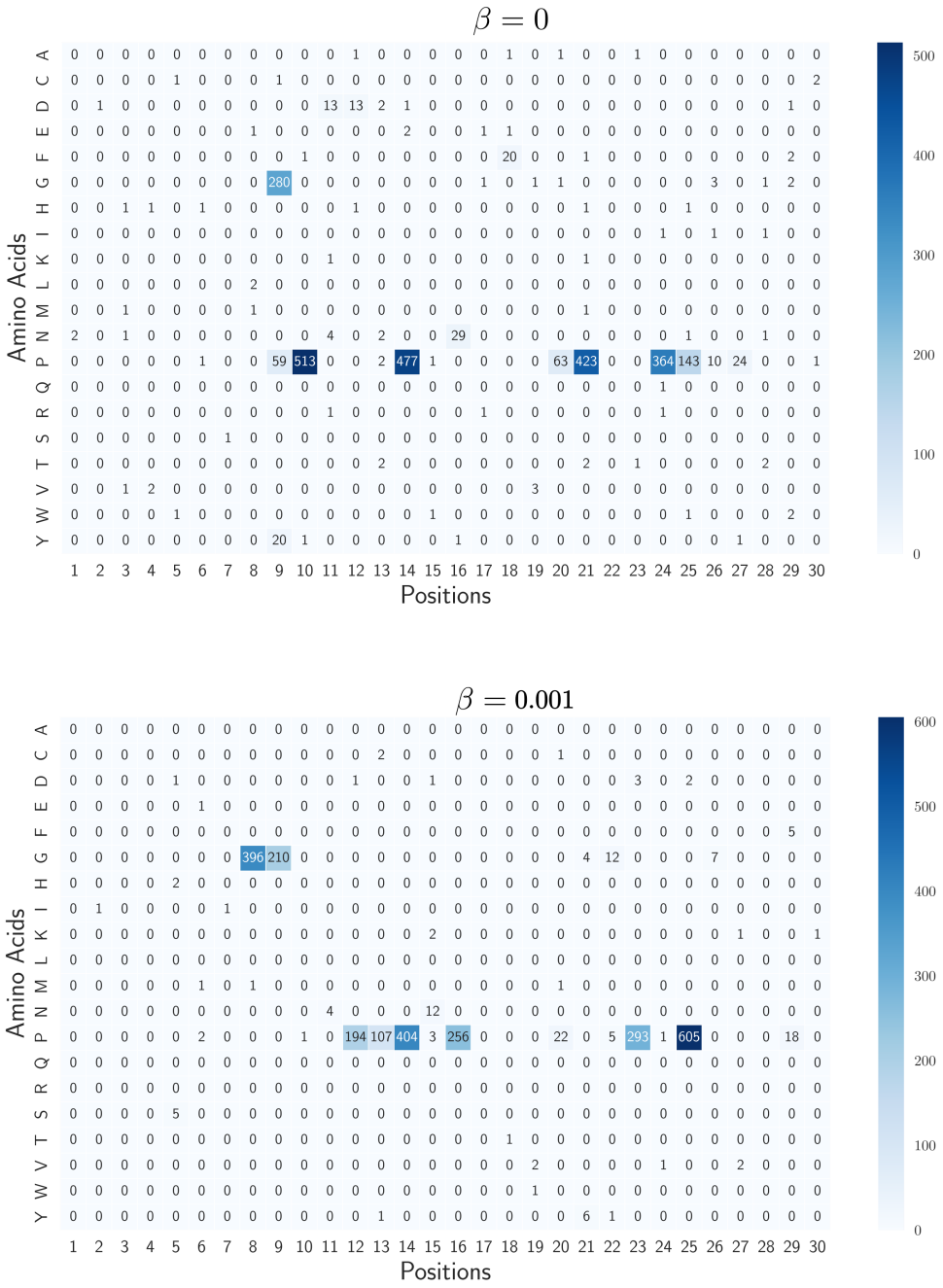



Figure 4.2: Comparison between Substitution Landscape before and after Entropy Regularisation, Training on the TP dataset

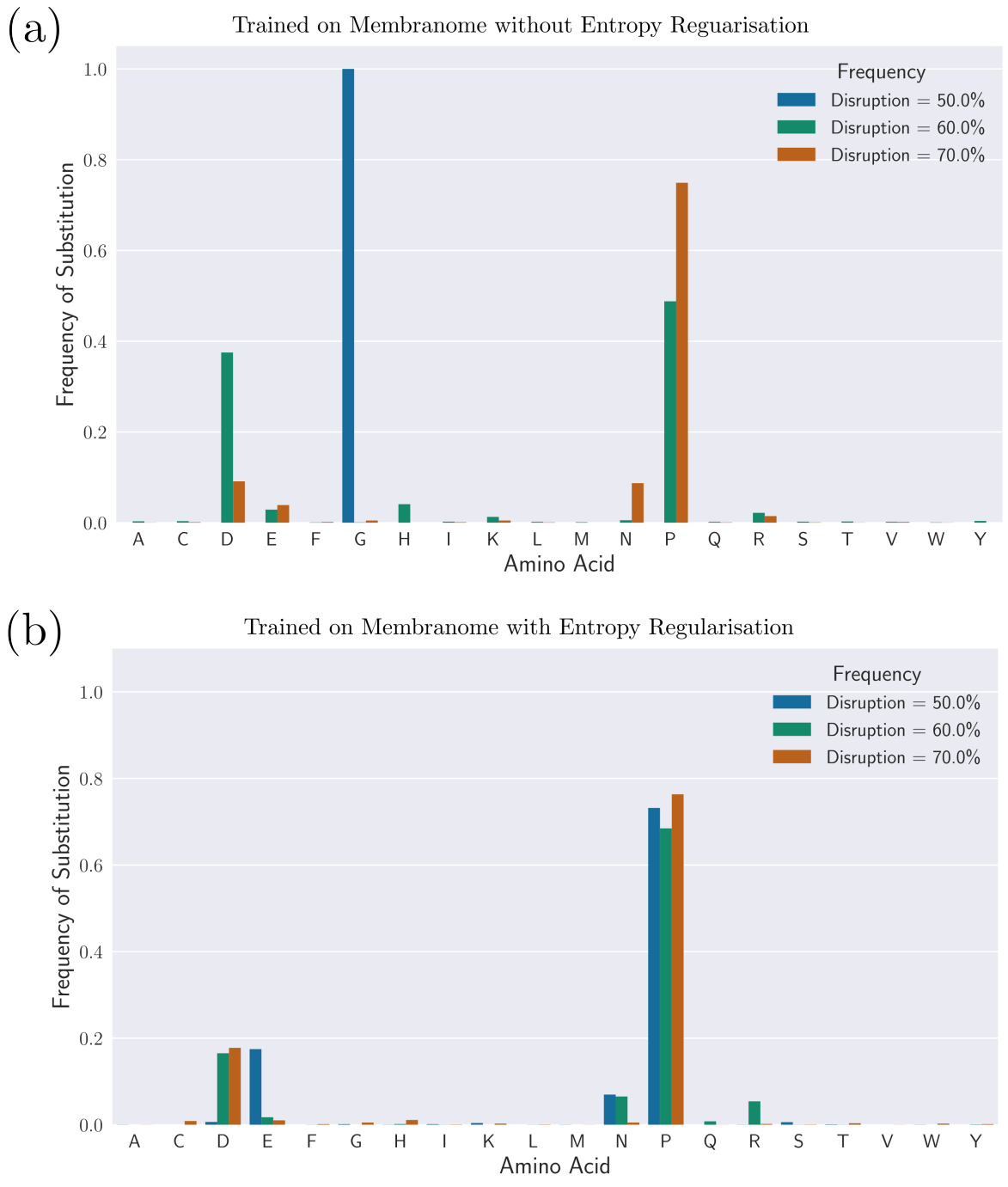



Figure 4.3: Substitution Frequencies of Different amino acids trained on the Membranome Dataset trained (a) without Entropy Regularisation ($\beta = 0$) and (b) with Entropy Regularisation ($\beta = 0.001$)

| model index | $\beta$ | Percentage Disruption ($\Delta\%$) | Training Dataset | Learning Efficiency ($\kappa$) | Mutation Entropy ($S$) | Average number of mutations required for disruption |
|---|---|---|---|---|---|---|
| 1 | 0.0 | 70 | TP | 0.743 | 1.524 | 7.734 |
| 2 | 0.0 | 60 | TP | 0.815 | 0.888 | 6.392 |
| 3 | 0.0 | 50 | TP | 0.921 | 0.946 | 5.924 |
| 4 | 0.001 | 70 | TP | 0.854 | 1.624 | 7.166 |
| 5 | 0.001 | 60 | TP | 0.887 | 1.401 | 6.588 |
| 6 | 0.001 | 50 | TP | 0.939 | 1.251 | 5.726 |
| 7 | 0.0 | 70 | Mem | 0.685 | 2.167 | 7.714 |
| 8 | 0.0 | 60 | Mem | 0.647 | 0.96 | 6.256 |
| 9 | 0.0 | 50 | Mem | 0.775 | 1.168 | 5.51 |
| 10 | 0.001 | 70 | Mem | 0.543 | 1.365 | 7.331 |
| 11 | 0.001 | 60 | Mem | 0.643 | 1.303 | 6.394 |
| 12 | 0.001 | 50 | Mem | 0.759 | 1.384 | 5.853 |

Table 4.4: Validation metrics for Training without Proline as an available mutation

## 4.4 Examining Helix breaking without Proline

Since we observe mode collapse even after introducing Entropy Regularisation and changing the training dataset, we remove proline from the possible amino acids to mutate. We train on both TP and Membranome Datasets. Since with 8000 episodes, we do not observe convergence; we train these models for 14000 episodes. The validation metrics of these trained models are given in table 4.4.

We observe considerably improved values of Mutational Entropy ($S$). However, this is accompanied by a noticeable decrease in learning efficiency ($\kappa$ ) and an increase in the average number of mutations required to disrupt the structure. Through the metrics shown in table 4.4, we wish to pick the "best" model. Since we are also interested in maximum disruption, we only consider those models that disrupt the structure by 70 %. Amongst these, we choose model 4 (highlighted in colour yellow in table 4.4 ) because it has both acceptable values of $S$ and $\kappa$. We perform all further analysis using this model, and all further references in this section are about this model.

## 4.5 Our model can break all kinds of helices

The TP Database provides average helical propensity and contact information for all helices in its database. This information is helpful since it provides information on whether the given sequence remains a helix independently of its protein. Helices with higher propensity and lower contact remain a helix even in solution, while those with lower propensity and higher contacts are unlikely to remain stable independent of the protein[43]. We wish to examine if our model successfully breaks both helices. In our validation set, we arranged the helices based on their propensity and contact values and checked if the model failed to break any. We find that the model is agnostic to values of helical propensity and contacts.

## 4.6 Analysis of Mutations predicted by the chosen model

To understand the mutations predicted, we plot substitution frequencies (see figure 4.4 (a)). We observe that amino acids with known high propensity(such as Alanine and Leucine), show lower frequencies, as expected, while those with lower propensity (such as Glycine, Serine and Cysteine) show a higher frequency. We also observe that amino acids with positively charged polar side chains - Arginine (R), Histidine (H) and Lysine (K) show lower frequencies of substitution. The opposite, however, is not reflected in amino acids with negatively charged side chains. While Aspartate(D) is often predicted as a mutation, the same is not seen with Glutamate (G). This is surprising since both these amino acids are different by a single methylene group. This pattern is repeated again between Glutamine and Aspargine - while Glutamine is often predicted as a mutation, Aspargine is seldom predicted. We further analyze the positions at which mutations are predicted. Glycine prefers non-central positions, specifically 10-12 and 24. Aspartate(D), on the other hand, prefers position 22.

Lastly, we analyse what residues are modified. In this context, we call the residue previously present as "previous amino acid" and the residue predicted by the model as the "mutation". This is plotted as a heatmap in figure 4.5. We see that for the Glycine mutation, Alanine is the most frequent previous amino acid residue, followed by Leucine. Meanwhile, Proline is hardly mutated. This is concurrent with previous calculations of propensity. This analysis shows that the model has learnt to make mutations corroborating existing literature.

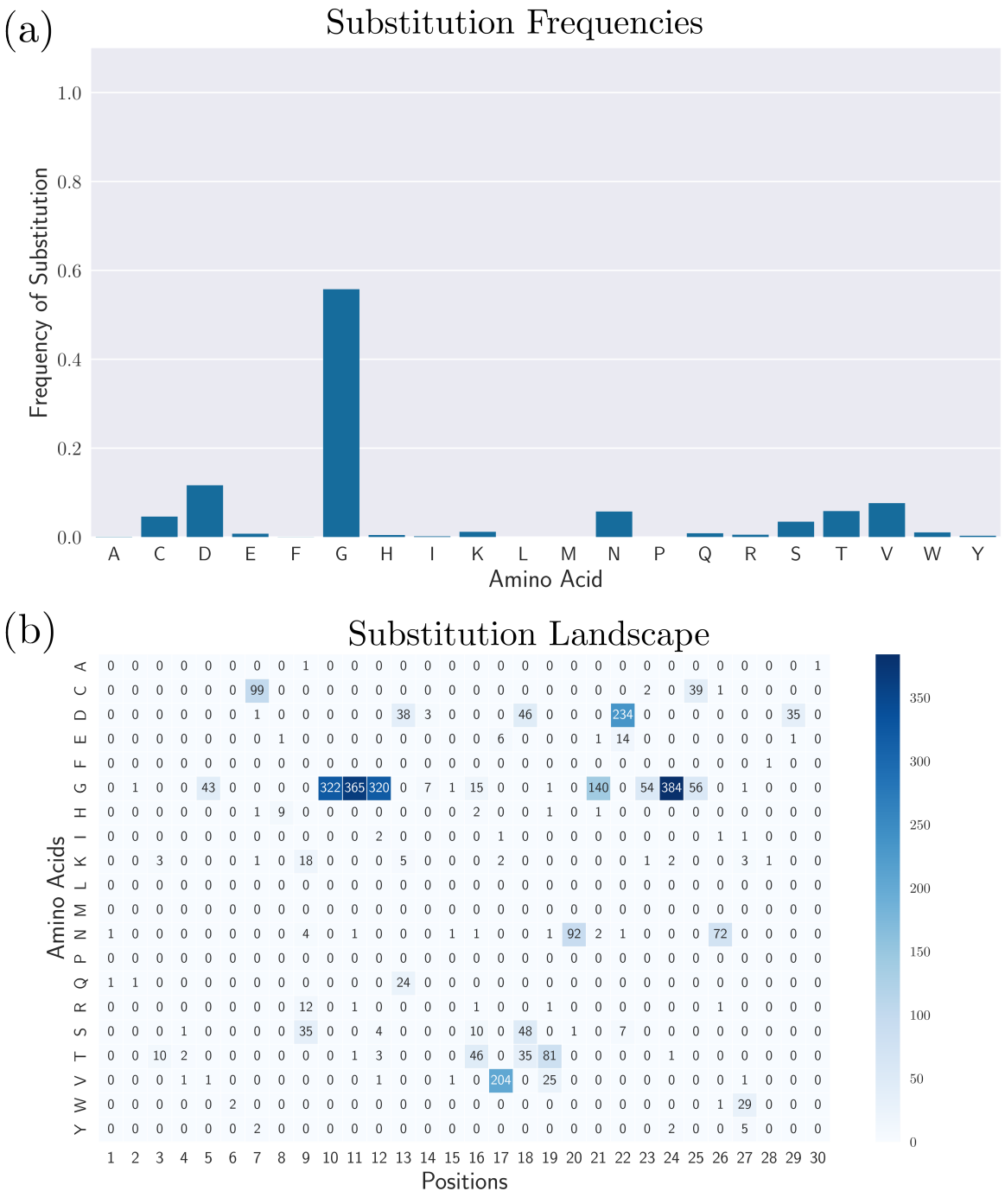



Figure 4.4: Frequencies and Landscapes of mutations as predicted in the chosen model.

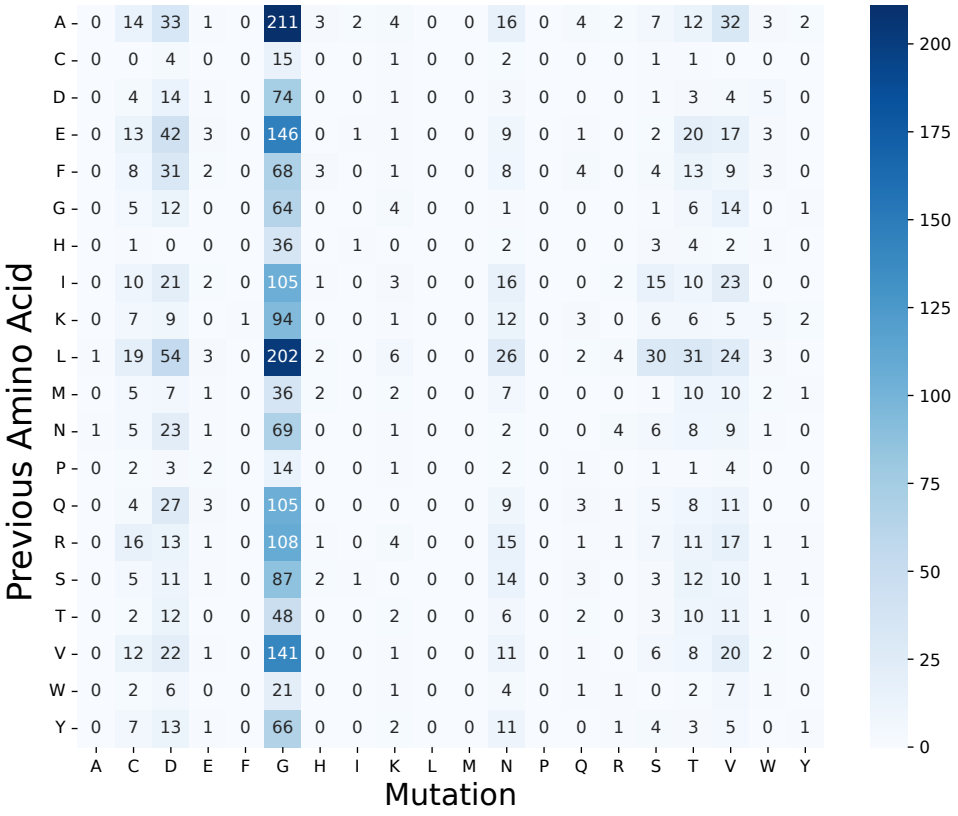



Figure 4.5: Heatmap of mutated and mutations predicted by the chosen model

| | Sequence |
|---|---|
| 1zik WT | RMKQLEDKVEELLSKKYHLENEVARLKKLV |
| 1zik Mutant | RMKQGEDKVGGGLSKKVSTEGDVAGLKKLV |
| 4k1c WT | QVRIVQASMLGSLLSNLLLVLGLCFIFGGY |
| 4k1c Mutant | QVRIGQAHMGGGLGSTVLTVLDLCCNFGGY |

Table 4.5: IDs and sequences whose free energy surface has been computed

## 4.7 Muated structures predicted by our Model are energetically stable.

We use ESMFold as our structure predictor to predict mutations. However, since the mutations we predict might not be available on the corpus that it is trained on, it is possible that ESMFold is predicting the wrong structures. Therefore, we perform well-tempered metadynamics simulations to check if the predicted mutations are stable. We examine the free energy landscapes of the predicted structures of the original sequence (WT) and the sequence after performing predicted mutations (Mut) by biasing our simulations along a collective variable called **ALPHARMSD**(details given in chapter 3). For a helix of 30 residues, while an **ALPHARMSD** value close to 0 corresponds to a complete coil or beta-sheet, a value close to 30 corresponds to a structure with all residues part of a helix. Since these calculations are expensive, we restrict our analysis to 2 cases. The sequences are given in table 4.5.:

1. helix with high propensity and low contact - the helix 1-30 of Leucine Zipper Transcription Factor (PDB: 1zik)

2. helix with low propensity and high contact - the helix 118-147 of the VCX1 Calcium/Proton r in *S cerevesiae* (PDB: 4k1c)

Upon running well-tempered metadynamics simulations for 200 ns, we observe that both mutants are stable in a non-helix form, as seen in figures 4.6 and 4.7. In both 1zik and 4k1c cases, the WT free energy surface samples more of the CV space than the Mutant. However, these results are preliminary since both 1zik and 4tpj WT simulations do not show the global minimum as expected close to 30. Hence, running more extensive simulations to study the stability and energetics is necessary.

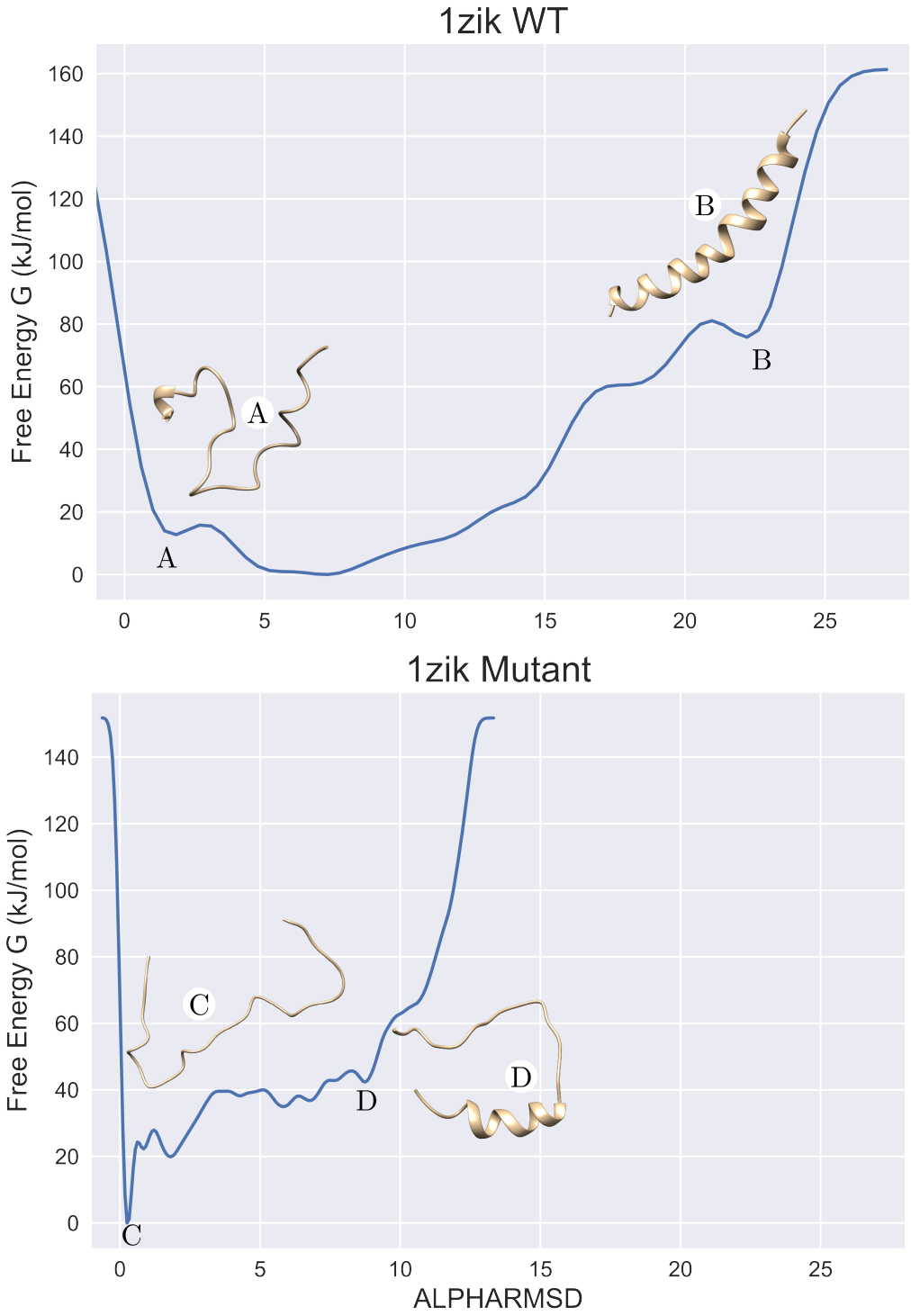



Figure 4.6: Free Energy Plots for Wild Type and Mutant Structures of helix 1zik (helix with high propensity and low contact)

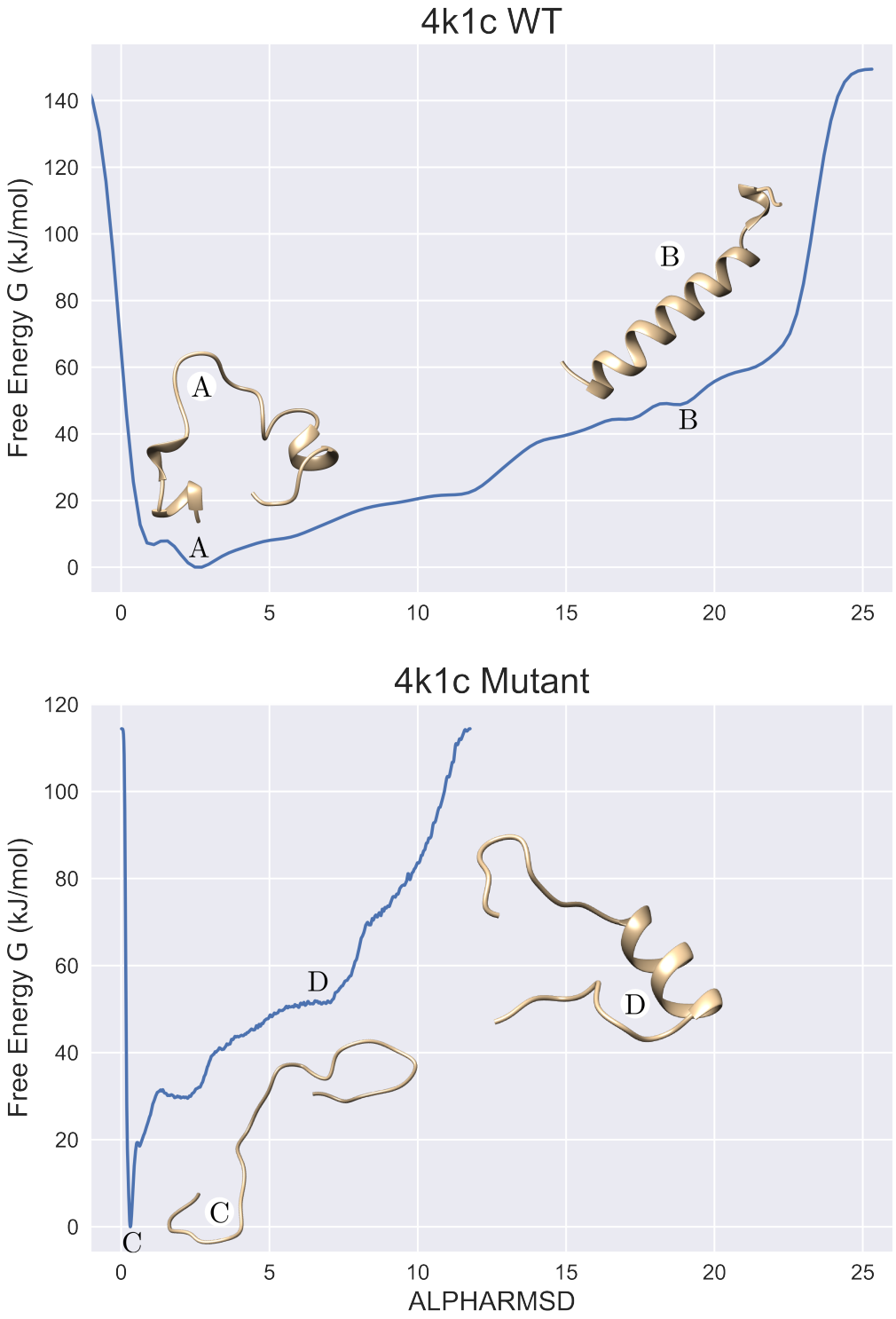



Figure 4.7: Free Energy Plots for Wild Type and Mutant Structures of helix 4k1c (helix with low propensity and high contact)

## 4.8 Our model has the potential to predict mutations that break whole protein structures

While we have been able to successfully disrupt helical structures, we wish to explore whether mutations that disrupt the structure of a helix can lead to a disruption in the structure of whole proteins. To this end, we perform mutations on the mouse prion protein (PDB ID: 1xyx) and the human prion protein (PDB ID: 1hjm), as mutations in these proteins are known to induce misfolding, leading to diseases such as Kuru and Creutzfeldt-Jakob Disease (CJD). While the physiological form of prion protein ($PrP^{C}$) predominantly adopts an alpha-helical conformation, the misfolded form or scrapie ($PrP^{SC}$)is characterized by beta-sheet enrichment[51].

To test this, we extract helix 3 from each protein and run our model 200 times. Based on the mutations predicted, we predict the structure of the entire protein to see if the disruption in the helical structure leads to a disruption in protein structure. To measure the disruption, we consider a difference in RMSD $>$ 15 Åbetween the wild-type and mutant structures.

Our results show that five mutant human prion proteins and one mouse prion protein satisfy this criterion. An example of disruption in mouse prion protein and human prion protein has been shown in figures 4.8 and 4.9 respectively. While in the mouse prion protein, nine mutations are required (4.8(b)), in the human prion protein case, six mutations are required for whole structure disruption(4.9(b)). Notably, in the mutated mouse prion protein, we observe more beta sheets than alpha helices, possibly hinting at a $PrP^{SC}$ form of the protein. However, further analysis is required to confirm these findings as they are based on static structures predicted by ESMFold.

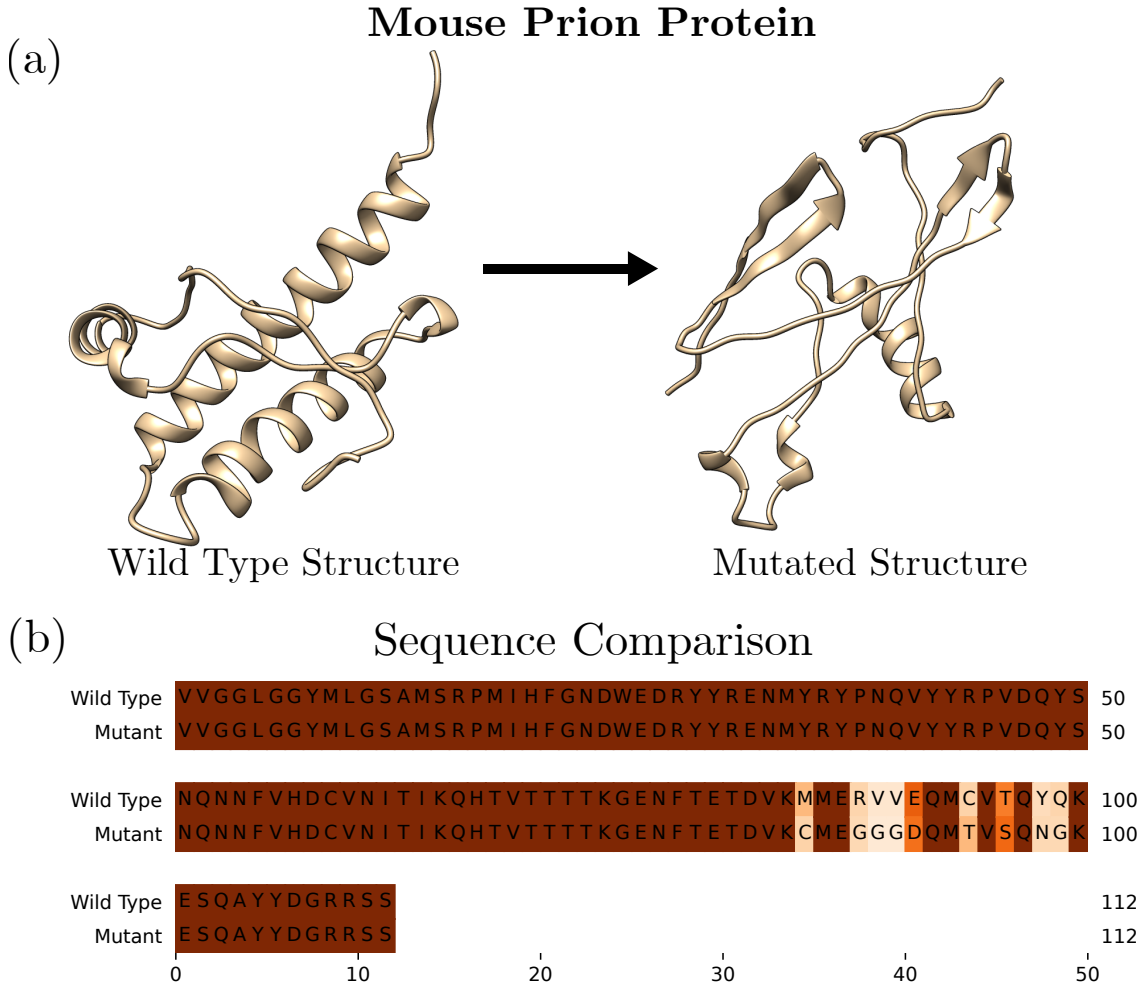



Figure 4.8: (a) Structures and (b) Sequences before and after predicted mutation in helix three of Mouse Prion Protein (PDB:1xyx)

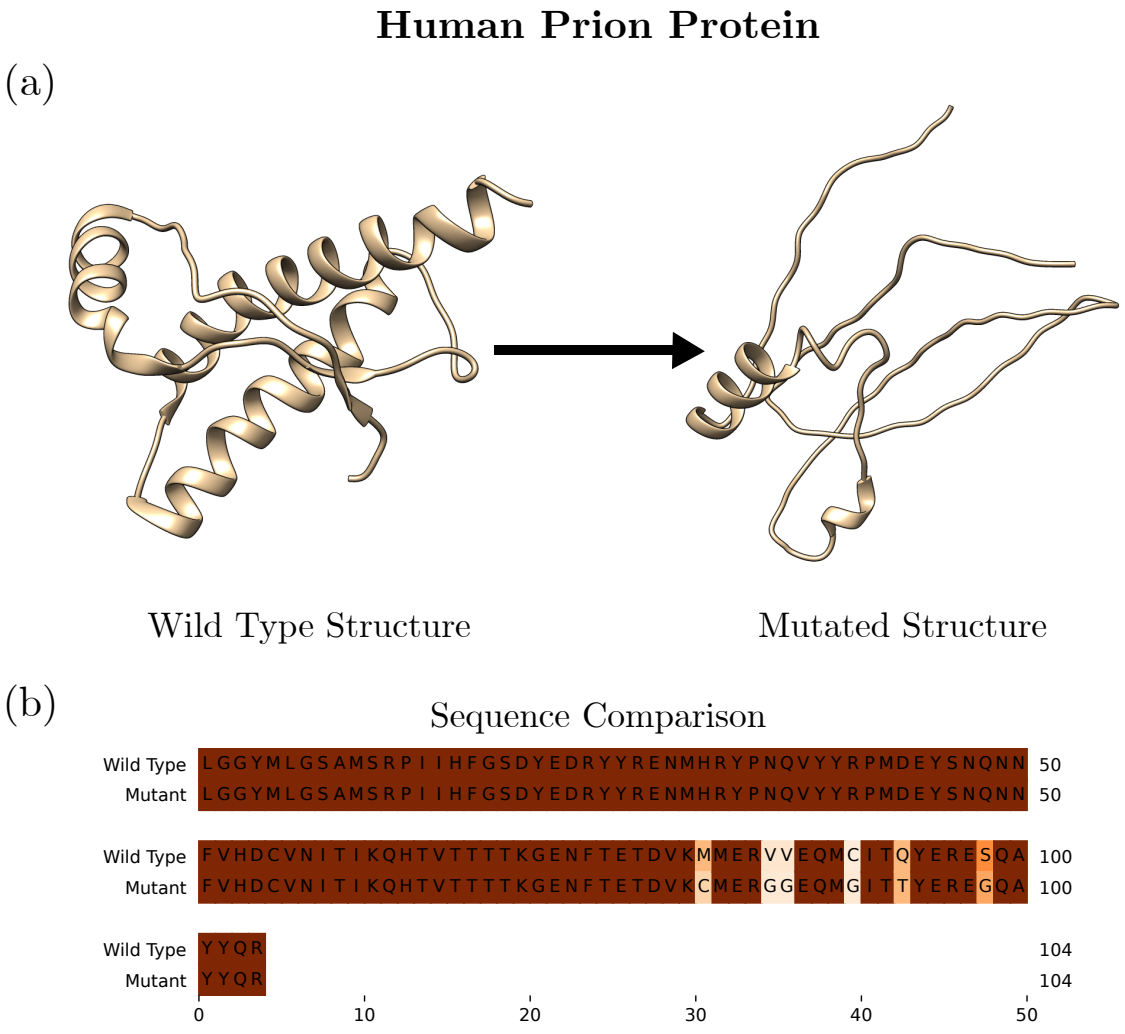



Figure 4.9: (a) Structures and (b) sequences before and after predicted mutation in helix three of Human Prion Protein (PDB:1hjm)

# Chapter 5

# Conclusion

We set out to build a general model capable of inducing mutations that disrupt helical secondary structures. Our strategy involved constructing a reinforcement learning (RL) model where sequences served as states, mutations as actions, and disruption as a reward. We curated diverse datasets of helices and employed them to train our RL model, utilising sequences from these datasets as initial states.

While our reinforcement learning model was successful at training, it learnt the most straightforward mutation: Proline. We attempted to mitigate this using entropy regularisation and could achieve slightly better results. Seeking deeper insights into structure-disrupting mutations, we systematically excluded Proline as a potential mutation and retrained our model. Subsequently, we identified a well-performing model based on validation metrics and utilised it to predict structure-disrupting mutations.

Our analyses demonstrated that the model effectively perturbs helical structures, with the predicted mutations aligning well with existing literature. Furthermore, employing well-tempered metadynamics simulations, we established the stability of the disrupted helix structure. We also showed that some mutations predicted using our model can disrupt whole protein structures. However, since some of these results are preliminary, further analysis is required to validate these findings.

Through our novel approach, we present a proof of concept illustrating the application of reinforcement learning in addressing problems related to protein structure disruption. The capability of such reinforcement learning models remains to be explored in more complex protein systems.

# Bibliography


(1) Voet, D.; Voet, J. G., *Biochemistry*, Google-Books-ID: 8mV19Jz_v4AC; John Wiley & Sons: 2010.

(2) Dragan, A.; Crane-Robinson, C.; Privalov, P. L. Thermodynamic basis of the $\alpha$-helix and DNA duplex. *European Biophysics Journal* **2021**, *50*, 787–792.

(3) Aurora, R.; Creamer, T. P.; Srinivasan, R.; Rose, G. D. Local interactions in protein folding: lessons from the $\alpha$-helix. *Journal of Biological Chemistry* **1997**, *272*, 1413–1416.

(4) Nick Pace, C.; Martin Scholtz, J. A Helix Propensity Scale Based on Experimental Studies of Peptides and Proteins. *Biophysical Journal* **1998**, *75*, 422–427.

(5) Chakrabartty, A.; Schellman, J. A.; Baldwin, R. L. Large differences in the helix propensities of alanine and glycine. *Nature* **1991**, *351*, 586–588.

(6) Petukhov, M.; Uegaki, K.; Yumoto, N.; Serrano, L. Amino acid intrinsic -helical propensities III: Positional dependence at several positions of C terminus. *Protein Science : A Publication of the Protein Society* **2002**, *11*, 766–777.

(7) Petukhov, M.; Muñoz, V.; Yumoto, N.; Yoshikawa, S.; Serrano, L. Position dependence of non-polar amino acid intrinsic helical propensities. *Journal of molecular biology* **1998**, *278*, 279–289.

(8) Myers, J. K.; Pace, C. N.; Scholtz, J. M. Helix Propensities Are Identical in Proteins and Peptides. *Biochemistry* **1997**, *36*, Publisher: American Chemical Society, 10923–10929.

(9) Li, S.-C.; Deber, C. M. A measure of helical propensity for amino acids in membrane environments. *Nature structural biology* **1994**, *1*, 368–373.

(10) Li, S.-C.; Goto, N. K.; Williams, K. A.; Deber, C. M. Alpha-helical, but not beta-sheet, propensity of proline is determined by peptide environment. *Proceedings of the National Academy of Sciences* **1996**, *93*, 6676–6681.

(11) Piela, L.; Némethy, G.; Scheraga, H. A. Proline-induced constraints in $\alpha$-helices. *Biopolymers: Original Research on Biomolecules* **1987**, *26*, 1587–1600.

(12) Nilsson, I.; von Heijne, G. Breaking the camel's back: proline-induced turns in a model transmembrane helix. *Journal of Molecular Biology* **1998**, *284*, 1185–1189.

(13) O'Neil, K. T.; DeGrado, W. F. A thermodynamic scale for the helix-forming tendencies of the commonly occurring amino acids. *Science* **1990**, *250*, 646–651.

(14) Bajaj, K.; Madhusudhan, M. S.; Adkar, B. V.; Chakrabarti, P.; Ramakrishnan, C.; Sali, A.; Varadarajan, R. Stereochemical Criteria for Prediction of the Effects of Proline Mutations on Protein Stability. *PLOS Computational Biology* **2007**, *3*, Publisher: Public Library of Science, e241.

(15) Huyghues-Despointes, B. M.; Scholtz, J. M.; Baldwin, R. L. Effect of a single aspartate on helix stability at different positions in a neutral alanine-based peptide. *Protein Science : A Publication of the Protein Society* **1993**, *2*, 1604–1611.

(16) Khan, S.; Vihinen, M. Spectrum of disease-causing mutations in protein secondary structures. *BMC Structural Biology* **2007**, *7*, Publisher: BMC, 56.

(17) Partridge, A. W.; Therien, A. G.; Deber, C. M. Polar mutations in membrane proteins as a biophysical basis for disease. *Peptide Science: Original Research on Biomolecules* **2002**, *66*, 350–358.

(18) Greenfield, N. J. Using circular dichroism spectra to estimate protein secondary structure. *Nature protocols* **2006**, *1*, 2876–2890.

(19) Hermans, J. Molecular dynamics simulations of helix and turn propensities in model peptides. *Current Opinion in Structural Biology* **1993**, *3*, 270–276.

(20) Abrusán, G.; Marsh, J. A. Alpha Helices Are More Robust to Mutations than Beta Strands. *PLoS Computational Biology* **2016**, *12*, e1005242.

(21) Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A., et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **2021**, *596*, 583–589.

(22) Lin, Z.; Akin, H.; Rao, R.; Hie, B.; Zhu, Z.; Lu, W.; Smetanin, N.; Verkuil, R.; Kabeli, O.; Shmueli, Y., et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **2023**, *379*, 1123–1130.

(23) Kabsch, W.; Sander, C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers: Original Research on Biomolecules* **1983**, *22*, 2577–2637.

(24) Labesse, G.; Colloc'h, N.; Pothier, J.; Mornon, J.-P. P-SEA: a new efficient assignment of secondary structure from C$\alpha$ trace of proteins. *Bioinformatics* **1997**, *13*, 291–295.

(25) Lyu, Z.; Wang, Z.; Luo, F.; Shuai, J.; Huang, Y. Protein secondary structure prediction with a reductive deep learning method. *Frontiers in Bioengineering and Biotechnology* **2021**, *9*, 687426.

(26) Yu, C.-H.; Chen, W.; Chiang, Y.-H.; Guo, K.; Martin Moldes, Z.; Kaplan, D. L.; Buehler, M. J. End-to-end deep learning model to predict and design secondary structure content of structural proteins. *ACS biomaterials science & engineering* **2022**, *8*, 1156–1165.

(27) Cheng, J.; Novati, G.; Pan, J.; Bycroft, C.; Žemgulytė, A.; Applebaum, T.; Pritzel, A.; Wong, L. H.; Zielinski, M.; Sargeant, T., et al. Accurate proteome-wide missense variant effect prediction with AlphaMissense. *Science* **2023**, *381*, eadg7492.

(28) Schmidt, A.; Röner, S.; Mai, K.; Klinkhammer, H.; Kircher, M.; Ludwig, K. U. Predicting the pathogenicity of missense variants using features derived from AlphaFold2. *Bioinformatics* **2023**, *39*, btad280.

(29) Ittisoponpisan, S.; Islam, S. A.; Khanna, T.; Alhuzimi, E.; David, A.; Sternberg, M. J. Can predicted protein 3D structures provide reliable insights into whether missense variants are disease associated? *Journal of molecular biology* **2019**, *431*, 2197–2212.

(30) Sutton, R.; Barto, A., *Reinforcement Learning: An Introduction*; A Bradford book; MIT Press: 1998.

(31) Wang, Y.; Tang, H.; Huang, L.; Pan, L.; Yang, L.; Yang, H.; Mu, F.; Yang, M. Self-play reinforcement learning guides protein engineering. *Nature Machine Intelligence* **2023**, *5*, 845–860.

(32) Angermueller, C.; Dohan, D.; Belanger, D.; Deshpande, R.; Murphy, K.; Colwell, L. In *International conference on learning representations*, 2019.

(33) Zhang, C.; Lapkin, A. A. Reinforcement learning optimization of reaction routes on the basis of large, hybrid organic chemistry–synthetic biological, reaction network data. *Reaction Chemistry & Engineering* **2023**, *8*, 2491–2504.

(34) Meller, J. et al. Molecular dynamics. *Encyclopedia of life sciences* **2001**, *18*.

(35) Yang, Y. I.; Shao, Q.; Zhang, J.; Yang, L.; Gao, Y. Q. Enhanced sampling in molecular dynamics. *The Journal of chemical physics* **2019**, *151*.

(36) Ray, D.; Parrinello, M. Kinetics from metadynamics: Principles, applications, and outlook. *Journal of Chemical Theory and Computation* **2023**, *19*, 5649–5670.

(37) Towers, M. et al. Gymnasium, 2023.

(38) Asgari, E.; Mofrad, M. R. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one* **2015**, *10*, e0141287.

(39) Kunzmann, P.; Müller, T. D.; Greil, M.; Krumbach, J. H.; Anter, J. M.; Bauer, D.; Islam, F.; Hamacher, K. Biotite: new tools for a versatile Python bioinformatics library. *BMC bioinformatics* **2023**, *24*, 236.

(40) Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in PyTorch. **2017**.

(41) Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* **1992**, *8*, 229–256.

(42) Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**.

(43) Tsai, C.-Y. et al. Helical structure motifs made searchable for functional peptide design. *Nature Communications* **2022**, *13*, Number: 1 Publisher: Nature Publishing Group, 102.

(44) Lomize, A. L.; Lomize, M. A.; Krolicki, S. R.; Pogozheva, I. D. Membranome: a database for proteome-wide analysis of single-pass membrane proteins. *Nucleic Acids Research* **2017**, *45*, D250–D255.

(45) Lindorff-Larsen, K.; Piana, S.; Palmo, K.; Maragakis, P.; Klepeis, J.; Dror, R. DE Shaw Improved side-chain torsion potentials for the Amber ff99SB protein force field., 2010, 78. *DOI: https://doi. org/10.1002/prot*, *22711*, 1950–1958.

(46) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. Comparison of simple potential functions for simulating liquid water. *The Journal of chemical physics* **1983**, *79*, 926–935.

(47) Bussi, G.; Donadio, D.; Parrinello, M. Canonical sampling through velocity rescaling. *The Journal of chemical physics* **2007**, *126*.

(48) Parrinello, M.; Rahman, A. Polymorphic transitions in single crystals: A new molecular dynamics method. *Journal of Applied physics* **1981**, *52*, 7182–7190.

(49) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **2015**, *1*, 19–25.

(50) Tribello, G. A.; Bonomi, M.; Branduardi, D.; Camilloni, C.; Bussi, G. PLUMED 2: New feathers for an old bird. *Computer physics communications* **2014**, *185*, 604–613.

(51) Nagy, J. K.; Sanders, C. R. Destabilizing mutations promote membrane protein misfolding. *Biochemistry* **2004**, *43*, 19–25.

(52) Lu, H.; Zhou, Q.; He, J.; Jiang, Z.; Peng, C.; Tong, R.; Shi, J. Recent advances in the development of protein–protein interactions modulators: mechanisms and clinical trials. *Sig Transduct Target Ther* **2020**, *5*, Number: 1 Publisher: Nature Publishing Group, 1–23.

(53) Sedov, I. A.; Zuev, Y. F. Recent Advances in Protein–Protein Interactions. *Int J Mol Sci* **2023**, *24*, 1282.

(54) Osman, A. Yeast two-hybrid assay for studying protein-protein interactions. *Methods Mol Biol* **2004**, *270*, 403–422.

(55) Lin, J.-S.; Lai, E.-M. Protein-Protein Interactions: Co-Immunoprecipitation. *Methods Mol Biol* **2017**, *1615*, 211–219.

(56) Louche, A.; Salcedo, S. P.; Bigot, S. Protein-Protein Interactions: Pull-Down Assays. *Methods Mol Biol* **2017**, *1615*, 247–255.

(57) Vakser, I. A. Protein-Protein Docking: From Interaction to Interactome. *Biophys J* **2014**, *107*, 1785–1793.

(58) Li, Y.; Zhang, X.; Cao, D. The Role of Shape Complementarity in the Protein-Protein Interactions. *Sci Rep* **2013**, *3*, Number: 1 Publisher: Nature Publishing Group, 3271.

(59) McCoy, A. J.; Chandana Epa, V.; Colman, P. M. Electrostatic complementarity at protein/protein interfaces11Edited by B. Honig. *Journal of Molecular Biology* **1997**, *268*, 570–584.

(60) Gupta, A.; Mukherjee, A. Capturing surface complementarity in proteins using unsupervised learning and robust curvature measure. *Proteins: Structure, Function, and Bioinformatics* **2022**, *90*, 1669–1683.

(61) Dunham, B.; Ganapathiraju, M. K. Benchmark Evaluation of Protein–Protein Interaction Prediction Algorithms. *Molecules* **2021**, *27*, 41.

(62) Dubourg-Felonneau, G.; Wesego, D. M.; Akiva, E.; Varadan, R. PiNUI: A Dataset of Protein-Protein Interactions for Machine Learning. *bioRxiv* **2023**, 2023–12.

(63) Del Toro, N. et al. The IntAct database: efficient access to fine-grained molecular interaction data. *Nucleic Acids Research* **2022**, *50*, D648–D653.

(64) Varadi, M. et al. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research* **2022**, *50*, D439–D444.

(65) Trabuco, L. G.; Betts, M. J.; Russell, R. B. Negative protein–protein interaction datasets derived from large-scale two-hybrid experiments. *Methods* **2012**, *58*, 343–348.

(66) Sanner, M. F.; Olson, A. J.; Spehner, J.-C. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers* **1996**, *38*, 305–320.

(67) Li, L.; Li, C.; Sarkar, S.; Zhang, J.; Witham, S.; Zhang, Z.; Wang, L.; Smith, N.; Petukh, M.; Alexov, E. DelPhi: a comprehensive suite for DelPhi software and associated resources. *BMC Biophysics* **2012**, *5*, 9.

(68) Panday, S. K.; Shashikala, M. H.; Koirala, M.; Pahari, S.; Chakvorty, A.; Peng, Y.; Li, L.; Jia, Z.; Li, C.; Alexov, E. Modeling electrostatics in molecular biology: A tutorial of DelPhi and associated resources [Article v1. 0]. *Living Journal of Computational Molecular Science* **2019**, *1*, 10841–10841.

(69) The UniProt Consortium UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research* **2019**, *47*, D506–D515.

(70) Fowler, N. J.; Williamson, M. P. The accuracy of protein structures in solution determined by AlphaFold and NMR. *Structure* **2022**, *30*, 925–933.e2.

(71) A practical guide to large-scale docking | Nature Protocols.

(72) Che, X.; Liu, Q.; Zhang, L. An accurate and universal protein-small molecule batch docking solution using Autodock Vina. *Results in Engineering* **2023**, *19*, 101335.

(73) Webb, B.; Sali, A. Comparative Protein Structure Modeling Using $_e$*print*. *Current Protocols in Bioinformatics* **2016**, *54*, 5.6.1–5.6.37.

(74) Mirdita, M.; Schütze, K.; Moriwaki, Y.; Heo, L.; Ovchinnikov, S.; Steinegger, M. ColabFold: making protein folding accessible to all. *Nature Methods* **2022**, *19*, Number: 6 Publisher: Nature Publishing Group, 679–682.

(75) McGovern, S. L.; Shoichet, B. K. Information Decay in Molecular Docking Screens against Holo, Apo, and Modeled Conformations of Enzymes. *Journal of Medicinal Chemistry* **2003**, *46*, Publisher: American Chemical Society, 2895–2907.

(76) Chapman, B.; Chang, J. Biopython: Python tools for computational biology. *ACM SIGBIO Newsletter* **2000**, *20*, 15–19.

(77) Corso, G.; Deng, A.; Fry, B.; Polizzi, N.; Barzilay, R.; Jaakkola, T. Deep Confident Steps to New Pockets: Strategies for Docking Generalization. *arXiv preprint arXiv:2402.18396* **2024**.

(78) Meli, R.; Biggin, P. C. spyrmsd: symmetry-corrected RMSD calculations in Python. *Journal of Cheminformatics* **2020**, *12*, 1–7.

# Appendix A

# Collaborative Project - 1

## Predicting Protein-Protein Interactions Using Curvature and Potentials as Graph Node Features

*This chapter describes the work done during the course of my MS-Project along with Vishnu Kadam, a BS-MS Student working in CCB Lab, IISER Pune. This appendix describes my contribution to the work.*

## A.1 Introduction

Protein-protein interactions are paramount to biology, driving a large number of biological processes[52], including cell-signalling, antibody-antibody complexes, cell proliferation, and apoptosis amongst others[53]. Protein-protein interactions form an interaction network termed 'Interactome'. The study of the interactome is of profound importance since aberrant protein-protein interactions can lead to various diseases such as cancer, infectious and neurodegenerative diseases. Protein-protein interactions can be inferred through different kinds of experiments, including Yeast 2 hybrid experiments[54],co-immunoprecipitation[55] and pull-down experiments[56].

Advances in InSilico methods, including docking[57] have allowed the prediction of protein interactions. However, docking suffers from several inaccuracies, thus prompting a cautious approach to its usage. At the heart of protein-protein interactions lie two kinds of interactions - physiochemical and stereochemical. Shape complementarity has been implicated at the centre of protein-protein interactions before[58]. Meanwhile, studies have shown that interfaces are complementary in potential but not in charge[59]. Previously, our group reported an unsupervised learning approach to predict the shape complementarity of protein-protein interactions. They divide the reduced surface of a protein into several clusters using hierarchical clustering, followed by a sphere fitting to find the curvature of each cluster. They later used a complementarity measure to obtain the complementarity

score for various complexes[60].

Interactomes can be presented as a network where each node represents a protein while each edge represents an interaction between those proteins. Several interactome databases exist, compiled through several biological methods, as mentioned above. Similarly, there also exist non-interactome databases such as the Negatome and Negative Databases. Methods exist for identifying protein-protein interactions but suffer from incorrect database preparation[61]. Recently, efforts have also been made to account for inherent biases.[62]

Through this work, we create a model that predicts protein-protein interactions given the structure. We build novel datasets to prevent inaccuracies arising due to the dataset. We represent both proteins as graphs, with each node as curvature and electrostatic potential. We further pass these through a graph convolutional network (GCN) and predict interactions. We obtain results with high accuracy when this model is run to predict human and yeast protein-protein interactions.

## A.2 Materials and Methods

### A.2.1 Positive Dataset Preparation

We use the IntAct Database[63] to prepare positive examples for our model. We select proteins. We further select only "direct interactions", with direct contact between molecules. We also only select intra-organism molecular interactions. To get the structures of these proteins, we query the AlphaFold DB[64] and only those present in the database are used for further experiments.

### A.2.2 Negative Dataset Preparation

We use the NEGATIVE[65] dataset of proteins. Here we select all entries from *Homo Sapiens* and *Saccharomyces cerevisiae*. We select the rows with `shortestPath:9`, and extract the relevant protein structures from the AlphaFold DB[64]

### A.2.3 Preparing the Surface Files with Potential

We calculate the Molecular Surface of the protein using the MSMS[66] Software, with density = 3.0 and radius = 1.5 Å. Further, we use Delphi[67] to assign the potential on the surface. We set percent fill = 70 %, with a scale = 2[68]. We use the Amber98 force field with a salt concentration of 0.15 to set the charges and calculate the electrostatic potential on the protein surface.

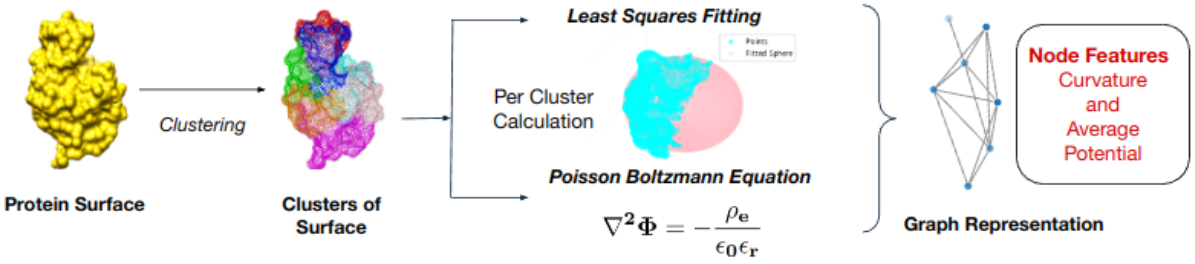



Figure A.1: Schematic representation of preparation of datasets

### A.2.4 Finding Curvature of Obtained Surfaces

We use our in-house algorithm[60] to cluster the protein surfaces into clusters while using a threshold for clustering=15. We further calculate the geometric centre and the average electrostatic potential for each cluster. This data is used later to construct a graph out of a protein. A schematic of the dataset preparation process is shown in figure A.1

The graphs are further passed on to a Graph Convolutional Network to predict interactions. Since Vishnu Kadam implemented graph creation and neural networks, I will not describe them here.

### A.2.5 Dataset Details

We show the compiled datasets for Humans (*Homo sapiens*) and Yeast (*Saccharomyces cerevisiae*) in the table below:

| Organism | Positive Interaction Pairs | Negative Interaction Pairs |
|---|---|---|
| Humans | 3801 | 1480 |
| Yeast | 3987 | 42090 |

Table A.1: Curated dataset of protein-protein interactions

# Appendix B

# Collaborative Project - 2

## Hypotheisising Mechanisms of Antibiotic Tolerance in LoopSwap Mutant of *E coli* MG-1655 through Proteome Wide Docking

*This chapter describes my contribution to the collaborative work done during the course of my MS-Project done along with Pratishruti Panda(CCB Lab,IISER Pune) and Ateek Shah (Whytamin Lab IISER Pune).*

## B.1 Introduction

Flavin adenine dinucleotide (FAD) is a redox-active coenzyme involved in several enzymatic reactions in metabolism. In *E. coli* MG-1655, the FAD synthase (FADS) enzyme produces FAD from Flavin Mononucleotide using phosphorylation using the kinase domain and adenylation using the adenylyl transferase domain.

Our collaborator, Ateek Shah from Whytamin Lab, IISER Pune, modified regions of the Adenyl Transferase domain of the wt-FADS to obtain a LoopSwap variant of the enzyme. The LoopSwap variant produces analogues of FAD, namely FUD, FCD, and FGD (FAD and its analogous are hereby referred to as FNDs).

Upon obtaining the LoopSwap mutant, he conducted antibiotic resistance assays in collaboration with Nishad Matange's Lab at IISER Pune. After screening against various classes of antibiotics, they found that the LoopSwap mutant displays resistance against aminoglycoside antibiotics, such as Kanamycin, Gentamicin, etc.

FAD regulates proton gradients in a cell's inner membrane, which can affect aminoglycoside uptake. Ateek's experiments suggest that the concentration of FAD in the mutant cells could influence resistance. We hypothesise that resistance occurs due to differential binding of proteins involved directly or in upstream processes leading to resistance mechanisms. To test this hypothesis, we dock the entire *E. coli* MG 1655 proteome with the

FNDs and check for differential binding.

## B.2 Methods

### B.2.1 Preparation of the structural database for docking

To dock the entire proteome against FNDs, we require the 3-dimensional structures of all the *E. coli* MG-1655 strain proteins. In literature, as far as our understanding goes, no unambiguous one-protein, one-structure database for *E. coli* exists. While AlphaFold DB[64] provides structures for almost all proteins, it does not consider experimentally determined structures. Hence, we construct a database that maps each protein of the *E. coli* proteome to a structure while considering the best structure from experimental and computational approaches.

We collect the UniProt accession IDs of the proteins in the *E. coli* MG-1655 proteome from the UniProt Database[69] for each UniProt Accession ID. The queried data obtained for a UniProt ID is shown for representative purposes in B.1. We use the following algorithm to select the best structure for each UniProt ID.

1. We check if an X-ray or cryo-EM structure is present for the UniProt ID. If we do not find an experimentally determined structure, we choose the AlphaFold DB structure. We exclude NMR structures since AlphaFold has been shown to outperform NMR in structural accuracy[70], specifically in tasks where a static structure is required, such as docking.

2. If multiple X-ray or cryo-EM structures are present, we choose the structure with the highest resolution, provided the resolution is better than a chosen cutoff resolution of 3 Å[71,72]. We further select only those structures that have fewer than 11 missing residues. We use the structures with less than six missing residues directly for docking, while structures with missing residues between 6 and 10 are modelled using MODELLER[73].

3. Some UniProt IDs do not have an AlphaFold predicted structure in the dataset because they have multiple isoforms. We model the structures of some of these proteins using ColabFold[74]. We exclude proteins with $<$16 amino acids since ColabFold cannot model them.

Representations of the protein, including whether the protein is holo or apo, are known to affect docking scores[75]. Considering this, we create datasets of experimental structures having bound ligands (holo) and no bound ligands (apo). We create these datasets by querying the RCSB PDB API for "non-polymer bound ligands" in PDB IDs

| uniprot_id | database | id | method | resolution | chains | length |
|---|---|---|---|---|---|---|
| P00350 | PDB | 2ZYA | X-ray | 1.60 Å | A/B=1-468 | 468 |
| P00350 | PDB | 2ZYD | X-ray | 1.50 Å | A/B=1-468 | 468 |
| P00350 | PDB | 3FWN | X-ray | 1.50 Å | A/B=1-468 | 468 |

Table B.1: Representative example of queried data for UniProt ID - P00350 (6-phosphogluconate dehydrogenase). Such data is present for all proteins whose UniProt IDs have experimentally determined structure(s)

corresponding to the UniProt IDs of dataset 2. If "non-polymer bound ligands" return a value greater than 0, we classify the structure as holo. If not, the structure is classified as apo. We provide a breakdown of all the structures and their respective datasets in B.2

Datasets 1, 3, and 5 need no further preprocessing since they are obtained for the particular UniProt ID. However, since structures in datasets 2, 9, and 10 can contain multiple proteins, we use BioPython[76] to obtain the structure corresponding to the UniProt ID using chain information from the PDB structure.

### B.2.2 Choosing Docking Algorithm For Docking

While AutoDock Vina is considered as the gold standard in Molecular Docking, it poses some challenges in our use case. We wish to obtain binding sites, along with their binding energies to find differential binding between the different FNDs, in a fast and accurate model. DiffDock-L[77], which is a diffusion based generative model, has shown promising results in binding pose prediction and has shown to outperform traditional blind docking programs. In order to validate this method, we choose protein structures from dataset 10 (Holo Structures) that contain FAD bound to them. We obtain 11 such structures. We dock FAD to these proteins through DiffDock-L and calculate RMSD using sPyRMSD[78] and find that 9 out of 11 docking poses are predicted within 2 accuracy (results shown in table B.3). Since the accepted accuracy cutoff is 2 Å, we choose DiffDock-L as our algorithm. Docking studies are yet to be conducted on the 4705 proteins to investigate proteins that have differential binding.

| **Dataset ID** | **Number of Proteins** | **Source of the Structures and their Description** |
|---|---|---|
| 1 | 2715 | AlphaFold DB (structures that have only AlphaFold DB structures) |
| 2 | 1116 | RCSB PDB (experimental structures satisfying resolution and missing residue cutoffs) |
| 3 | 17 | Inhouse Generated using ColabFold (structures neither present in RCSB PDB nor in AlphaFold DB) |
| 4 | 9 | RCSB PDB, modelled using MODELLER (experimental structures satisfying resolution cutoff but having between 6 to 10 missing residues.) |
| 5 | 21 | NONE (Proteins with less than 16 amino acids without an experimental structure) |
| 6 | 525 | AlphaFold DB (proteins that have experimental structures that do not meet resolution or missing residue cutoff) |
| | 4403 | Total Number of Proteins in *E coli* MG-1655 |
| **Extended Datasets** | | |
| 9 | 790 | RCSB PDB (Experimental structures that satisfy resolution and missing residues cutoff and have bound ligands) |
| 10 | 650 | RCSB PDB (Experimental structures that satisfy resolution and missing residues cutoff and do not have bound ligands) |

Table B.2: Dataset ID and corresponding sources of the protein structures in E-coli in MG-1655

| UniProt ID | RMSD |
|---|---|
| P00363 | 1.053624 |
| P07003 | 0.761699 |
| P77324 | 18.35138 |
| P28861 | 1.540383 |
| P77212 | 1.316039 |
| P08142 | 1.22847 |
| P0AC41 | 0.987394 |
| P24232 | 5.13723 |
| P10902 | 0.838798 |
| P42593 | 1.394449 |
| P35340 | 0.821373 |

Table B.3: RMSD between ligands present in the holo structures and the first ranked docked predicted by DiffDock-L

# Appendix C

# Supporting Information for MS Project

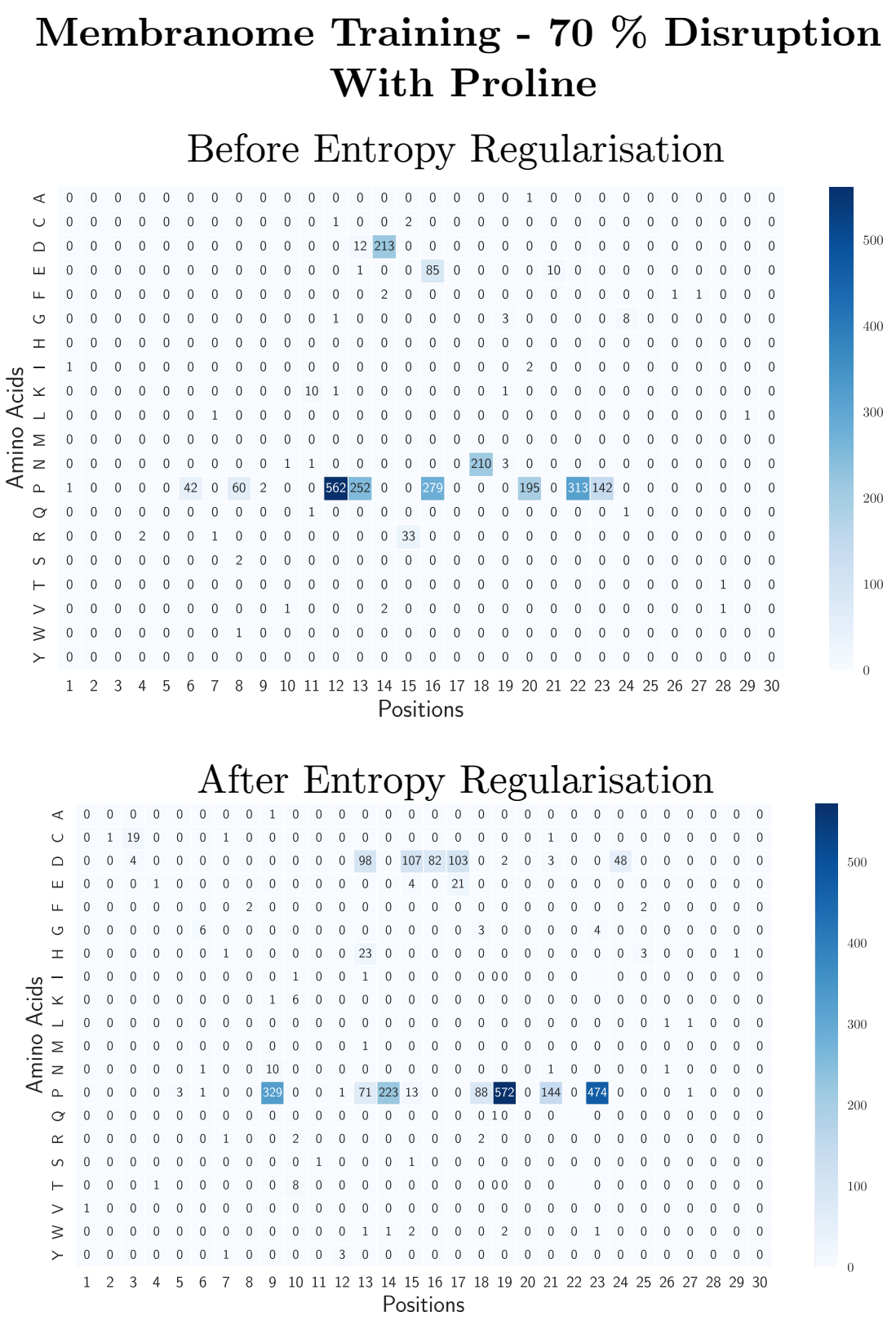



Figure C.1: Substitution Landscape with and without entropy regularisation. Training on Membranome Dataset with Proline