UNIVERSIDADE NOVA DE LISBOA

MScCBBi

MASTER IN
COMPUTATIONAL BIOLOGY
& BIOINFORMATICS

LUÍSA LISBOA MAIA CORREIA RODRIGUES

BSc in Molecular Biodesign

Genetic Algorithm-Driven Protein Engineering: Enhancing PT8 Polymorphic Toxins for Biotechnology

July, 2024

NOVA SCHOOL OF SCIENCE & TECHNOLOGY
NOVA MEDICAL SCHOOL
NOVA IMS Information Management School
DESDE 1902 INSTITUTO DE HIGIENE E MEDICINA TROPICAL UNIVERSIDADE NOVA DE LISBOA

Genetic Algorithm-Driven Protein Engineering: Enhancing PT8 Polymorphic Toxins for Biotechnology

LUÍSA LISBOA MAIA CORREIA RODRIGUES

BSc in Molecular Biodesign

Adviser: Leonardo Vanneschi
Full Professor, NOVA University Lisbon - IMS

Co-advisers: Maayan Gal
Full Professor, Tel Aviv University

Examination Committee:

Chair: Isabel Rocha
Full Professor, NOVA FCT

Rapporteurs: Mario Castelli
Associate Professor, NOVA IMS

Adviser: Leonardo Vanneschi
Full Professor, NOVA IMS

MASTER IN COMPUTATIONAL BIOLOGY AND BIOINFORMATICS

NOVA University Lisbon
June, 2024

Genetic Algorithm-Driven Protein Engineering: Enhancing PT8 Polymorphic Toxins for Biotechnology

To my Mother, forever my inspiration.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisors, Professor Leonardo Vanneschi and Professor Maayan Gal, for their invaluable guidance, patience, and support throughout my research journey. Professor Gal has believed in me since 2021 and his insightful feedback and unconditional encouragement have been instrumental in shaping my work and motivating me to strive for excellence.

I am also incredibly thankful to the directors of my program, especially Dra Diana Lousa, whose leadership and vision have profoundly influenced my academic and professional development. Her dedication to fostering a supportive and enriching learning environment has greatly enhanced my experience in the program.

My heartfelt thanks go to my family, who have been my constant source of love and encouragement. To my parents, Conceição e Décio, who have always believed in me and supported my dreams, and to my brother Afonso, whose companionship and understanding have been a source of comfort and joy, I am eternally grateful.

Lastly, I want to thank Miguel, who has been a pillar of strength and a true partner in this journey. His unwavering support, endless patience, and loving encouragement have made all the difference. His belief in me has been a constant source of motivation and has helped me to persevere through challenges.

“What I cannot create, I don't understand.” (Richard Feynman).

Abstract

Polymorphic toxins, such as PT8 from the *Leptospiraceae* family, play a significant role in microbial competition and have potential applications in biotechnology. However, their toxicity often limits their utility in practical applications. This thesis presents a novel approach to optimizing the PT8 polymorphic toxin for biotechnological use through the use of genetic algorithms (GAs) and machine learning models to reduce its inherent toxicity while maintaining functional integrity.

In this study, we first analyzed the structure and dynamics of PT8 using state-of-the-art protein structure prediction tools like AlphaFold, RosettaFold, and ESMFold, coupled with molecular dynamics simulations. These analyses revealed several key regions within PT8 that contribute to its stability and function but also underlie its toxicity.

To optimize PT8, we employed a genetic algorithm that iteratively adjusted amino acid sequences to minimize toxicity as predicted by a suite of machine learning models (CNN, XGBoost, KNN, Random Forest, and SVR). These models were trained to predict changes in Gibbs free energy (ΔG), which correlates with protein stability, using data from the Pro-Therm database and custom-generated datasets of PT8 variants.

Our results demonstrate that the optimized PT8 variants show a significant reduction in toxicity, as indicated by a decrease in ΔG, without compromising protein stability or function. Specifically, the best-performing model (CNN) facilitated a decrease in predicted toxicity by approximately 20%, while maintaining over 85% of the native protein's stability. These findings were corroborated by comparative analyses with existing computational methods, showing the efficacy and precision of our approach.

The implications of this research are twofold. First, it illustrates the potential of genetic algorithms combined with machine learning to enhance the safety and efficacy of protein toxins for biotechnological applications. Second, it provides a computational framework that can be adapted to other toxic proteins, paving the way for broader applications in synthetic biology and therapeutic development.

This work not only advances our understanding of PT8 and polymorphic toxins but also demonstrates a powerful toolset for protein engineering that bridges computational design with practical biological applications.

CONTENTS

LIST OF FIGURES

List of Tables

1
INTRODUCTION

1.1 Context and Motivation

In the microbial world, the use of diverse toxins is a fundamental strategy for survival and dominance. These toxins exhibit a wide variety of structures, mechanisms of action, and loci organization. Among these, polymorphic toxins (PTs) are multi-domain microbial exotoxins present in almost all bacterial lineages, characterized by a common domain architecture. According to Zhang et al. (2012), while PTs share common chromosomal encoding, secretory pathways, and toxin domains with other effector systems, they differ in their export pathways and structural architecture. The presence of specific toxic domains indicates distinct mechanisms in microbial intra- and interspecific conflicts.

1.2 Problem Definition

PT8 is a polymorphic protein specific to the Leptospiraceae family, particularly the human pathogen Leptospira interrogans. This protein, which is 118 amino acids long, was discovered and expressed by Nachmias et al. (2021). Toxicity assays were performed on PT8, revealing its potential for harm, but no structural studies were conducted.

To produce substantial quantities of PT8 polymorphic toxins within model microcellular organisms, it is crucial that these proteins do not exhibit toxic activity within the host cell. To achieve this, protein engineering techniques are employed to modify or design proteins, enhancing or altering their functions.

1.3 Aim

The primary aim of this research is to develop a less toxic, functionally enhanced variant of the PT8 polymorphic toxin using a combination of genetic algorithms and machine learning models. This approach aims to optimize the amino acid sequence of PT8 to reduce its toxicity for safer biotechnological applications.

1.3.1 Hypothesis

The hypothesis of this study is that genetic algorithms, guided by machine learning predictions of protein stability (ΔG), can identify mutations in the PT8 toxin that significantly reduce its toxicity without compromising its structural integrity and functional efficacy. We propose that:

- Structural Predictability and Functionality: By applying state-of-the-art protein structure prediction tools and molecular dynamics simulations, key functional and structural domains of PT8 can be accurately mapped, identifying potential hotspots that contribute to toxicity.
- Optimization Using Genetic Algorithms: A genetic algorithm can navigate the complex fitness landscape of PT8's sequence space to find less toxic variants. Machine learning models (CNN, XGBoost, KNN, Random Forest, and SVR) trained on known protein stability data can effectively guide this search by predicting the stability impacts (ΔG) of mutations.
- Enhanced Biotechnological Utility: The optimized PT8 variants will show a marked reduction in toxicity, as measured by predictive models and bioassays, while retaining or enhancing the protein's structural stability and functional properties, making it suitable for various biotechnological applications.

1.3.2 Goals and Proposed Solution

To achieve this aim, the study will:

1. Utilize protein structure prediction and molecular dynamics simulations to understand the initial structure and dynamics of PT8.
2. Implement a genetic algorithm to optimize the PT8 sequence for reduced toxicity based on machine learning predictions of ΔG.
3. Evaluate the performance of different machine learning models in guiding the optimization process.

4. Compare the optimized PT8 variants with the wild type to assess changes in toxicity and functionality.

2
LITERATURE REVIEW

2.1 Protein Design

Proteins are the fundamental building blocks of life, essential for numerous biological processes. Comprising various combinations of 20 different amino acids, proteins exhibit an extraordinary diversity of structures and functions. These amino acids are encoded by a set of 61 codons, which are sequences of three nucleotides in DNA or RNA. Each codon corresponds to a specific amino acid or serves as a start or stop signal for protein synthesis. This genetic code governs the precise assembly of amino acids during protein synthesis, ensuring the accurate formation of functional proteins essential for life. However, functional proteins are rare in the vast space of possible codons and thus, amino acid combinations. Additionally, as the need for a specific function to be performed increases, the number of sequences able to perform that task drastically decreases. Thus, functional amino-acid sequences are rare and at a complete disadvantage in the vast sequence space (Yang et al., 2019). According to Sormanni et al. (2018) proteins can be designed based on solubility, binding affinity, binding specificity and conformational stability. In this research work we will focus on conformational stability.

2.1.1.1 Conformational Stability

Protein conformational stability refers to the free energy change (ΔG) when a protein unfolds. This ΔG is calculated using the equation ΔG = ΔH - TΔS, where ΔH is the enthalpy change, T is the temperature, and ΔS is the entropy change (Schellman, 1987). The stability of a protein is influenced by various factors including hydrophobic interactions, hydrogen bonds, and ionic interactions, and these contributions can be quantified to understand the folding and stability mechanisms (Makhatadze & Privalov, 1996).

In the context of protein stability, ΔG represents the free energy difference between the folded and unfolded states of a protein:

$$\Delta G = G_{folded} - G_{unfolded}$$

Understanding this behaviour is crucial, particularly regarding their conformations and functionality. Two fundamental concepts that provide insights into these aspects are the Boltzmann probability distribution and Gibbs free energy. Both are intrinsically connected and relate to the thermodynamic properties of proteins.

The Boltzmann distribution describes the probability of a system being in a particular state as a function of the energy of that state and the system's temperature (Bryngelson & Wolynes, 1987). For proteins, the distribution of conformations follows this principle. The probability Pi of a protein being in a specific conformation i is related to the Gibbs free energy Gi of that conformation as follows:

$$P_i = \frac{e^{-\frac{G_i}{kT}}}{Z}$$

Where:

• Gi is the Gibbs free energy of conformation i

• k is the Boltzmann constant,

• T is the absolute temperature, and

• Z is the partition function, which normalizes the probabilities and is defined as (Dill, 1990):

$$Z = \sum_j e^{-\frac{G_j}{kT}}$$

The Gibbs free energy is a measure of the chemical potential or the capacity of a system to do work at constant temperature and pressure (Schellman, 1987). In the context of protein conformations:

- Lower Gibbs Free Energy, Higher Probability: the lower the Gibbs free energy of a protein conformation, the higher the probability of the protein being in that conformation. This is because lower energy states are thermodynamically more favourable (Bryngelson & Wolynes, 1987).

  • Higher Gibbs Free Energy, Lower Probability: the higher the Gibbs free energy of a protein conformation, the lower the probability of the protein being in that conformation. This reflects the thermodynamic instability of high-energy conformations (Onuchic et al., 1997)

2.1.2 Protein Design Methods

Protein design methods can be broadly categorized into four groups, often combining strategies from different categories for optimal results. Energy-based methods, the largest group, leverage statistical thermodynamics principles. They aim to place proteins in conformations with minimal free energy, typically targeting the native or bound state while considering interactions among atoms, solvent, and ligands. While theoretically aiming for an exhaustive exploration of conformational space, in practice, they often focus on adjusting backbone structures to accommodate side-chain mutations (Sormanni et al., 2018). Combinatorial design methods mimic natural evolution's combinatorial aspects, often repurposing existing fold fragments to design new proteins or functionalities. These methods require less computational demand but rely heavily on custom rules and fragment databases (Sormanni et al., 2018)Empirical methods utilize biological observations or measurements to rationalize protein design, such as knowledge-based potentials derived from known protein properties. They improve the accuracy of energy models and predict phenomena like aggregation (Sormanni et al., 2018). Machine-learning-based methods utilize vast biological data, allowing algorithms to learn patterns and make predictions. Though lacking physical transparency, they excel in pragmatic applications and are poised to shape the future of protein design as data availability and algorithm sophistication continue to increase (Sormanni et al., 2018)

2.1.2.1 Machine Learning Models

Machine learning has ushered in a new era of protein design, revolutionizing how scientists engineer proteins with desired functions and properties. The advancements in deep learning have introduced a plethora of tools and techniques, enabling researchers to explore and manipulate the vast landscape of protein sequences and structures with unprecedented precision (Khakzad et al., 2023).

Sequence-based models have been at the forefront of protein design, characterized by two main categories: sequence-only models and conditional sequence models. Sequence-only models typically employ generative frameworks, focusing on learning from a protein's primary structure and capturing biochemical constraints. Initially, these models emphasized position-specific predictions, but recent developments have led to more comprehensive frameworks. Hybrid models, which combine family-specific and family-agnostic approaches, aim to bridge

performance gaps. Additionally, diffusion models have emerged as a promising technique for generating full protein sequences from scratch.

In contrast, conditional sequence models refine the generative process by conditioning on broad taxonomic groups or gene ontology annotations. These models, utilizing autoregressive or masked-language techniques, enhance control over the generated sequences, facilitating nuanced sequence design tailored to specific criteria (Notin et al., 2024).

Sequence-label models utilize discriminative supervised learning to predict functional labels for input sequences, aiding in efficient phenotypical value prediction for potential designs. Discriminative models rely on external procedures to prioritize mutants, while label-conditioned generative models, such as conditional variational autoencoders, generate new sequences based on desired phenotypical values, allowing for effective end-to-end procedures ((Khakzad et al., 2023).

Structure-based models represent a paradigm shift in protein design, focusing on the tertiary structure of proteins. These models encompass diverse categories, including structure prediction, generation, inverse folding, and holistic design approaches. Sequence-structure models play a crucial role in generating new sequence designs when paired with structure generation models, acting as joint models of sequence and structure (Notin et al., 2024)

Function-based paradigms include function-to-structure and function-to-sequence design approaches. Advances in structure prediction have catalysed new design methodologies, and generative models are applied in de novo structure generation and function-specific sequence optimization.

The evolution of machine learning tools for protein design has been remarkable, with a growing array of algorithms and techniques at researchers' disposal. Noteworthy examples include AlphaFold, a groundbreaking tool for protein structure prediction leveraging deep learning, and Rosetta, a suite of tools for protein modelling and design. Additionally, ProteinGAN, a generative adversarial network for protein sequence generation, and ProGen, an autoregressive transformer model, offer innovative approaches to protein design (Khakzad et al., 2023; Notin et al., 2024)

The integration of machine learning into protein design has empowered researchers to tackle complex challenges, enabling precise control over protein function and structure (Figure 1.). As machine learning tools continue to evolve, they offer unprecedented opportunities for innovation, transforming protein engineering into a dynamic field poised for groundbreaking discoveries.

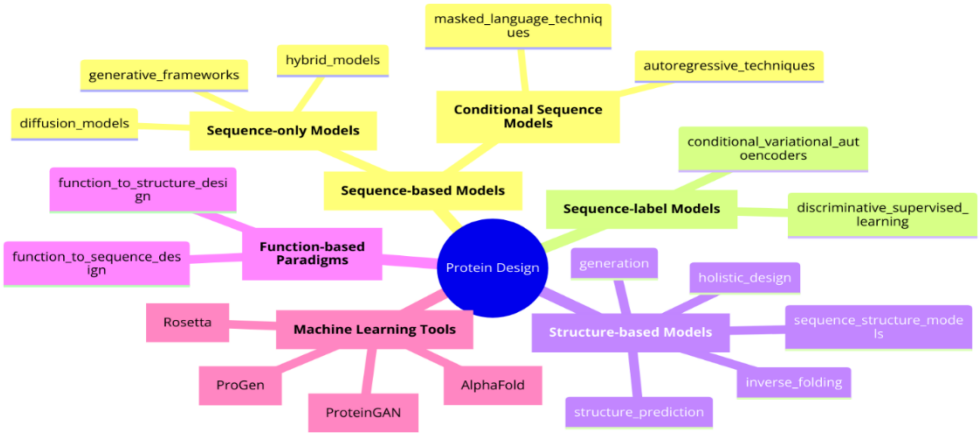

Figure 1. Protein Design: This diagram categorizes the diverse array of machine learning tools and methodologies applied in protein design, arranged into a network of relationships.

2.2 Protein Optimization

Directed evolution stands as a transformative technique in protein engineering, enabling the creation of proteins endowed with desired properties and functions (Figure 2). However, its potential is hampered by the sheer complexity of the mutational space, rendering deep mutational scanning of the entire protein library experimentally infeasible, given that the mutational space scales exponentially as N^{20}, where N represents the number of amino acids (Qiu & Wei, 2023). Despite its efficacy, directed evolution grapples with several inherent limitations. The

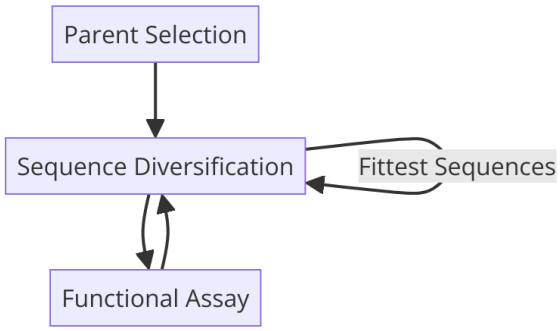

Figure 2 - Directed Evolution Workflow for Protein Engineering.

experimental evaluation process is often painstakingly time-consuming, compounded by the prevalence of silent mutations that fail to propagate to subsequent generations.

Moreover, the optimization of proteins entails navigating a labyrinthine fitness landscape, where trajectories of enhancement frequently culminate in dead ends unless multiple mutations occur simultaneously (Bricco et al., 2022). The relationship between a protein's sequence and its function is intricately woven within the fabric of the sequence–function 'fitness landscape.' Here, protein sequences are intricately mapped to corresponding fitness values, representing measurable properties such as catalytic activity or thermostability (Freschlin et al., 2022).

Directed evolution embarks on the traversal of this complex landscape through iterative rounds of mutation, coupled with high-throughput functional assays and meticulous selection, akin to a strategic ascent towards higher fitness sequences. Yet, despite its widespread application and demonstrated success, directed evolution remains resource and time-intensive (Freschlin et al., 2022). Moreover, the search process often lacks insight into the underlying sequence–function landscape, resulting in inefficient local searches that may ultimately cap the fitness attained in a protein engineering endeavour.

2.2.1 Protein Fitness Landscapes

Protein landscapes, often referred to as protein fitness landscapes, are conceptual frameworks that represent the relationship between protein sequences and their corresponding functions or fitness. These landscapes play a crucial role in understanding protein evolution, function, and engineering. The fitness landscape concept allows scientists to visualize and study the intricate interplay between protein structure, function, and evolution (Romero & Arnold, 2009).

Protein fitness landscapes can be imagined as topographical maps where each point represents a different protein sequence, and the elevation represents the fitness or functionality of that sequence. These landscapes can range from smooth, single-peaked "Fujiyama" landscapes to rugged, multi-peaked "badlands" landscapes (Romero & Arnold, 2009) .

They provide a mathematical framework for understanding how protein sequences map to their respective fitness or functional properties. These landscapes can be visualized as high-

dimensional spaces where each node represents a unique protein sequence, and the elevation of the node indicates the fitness level of that sequence. The landscape is a function:

$$f: S \rightarrow R$$

where S represents the set of all possible protein sequences, and the value f(s) for a sequence ss represents its fitness. The mathematical characterization of these landscapes involves statistical and machine learning methods to estimate the fitness function based on experimental data (Groth et al., 2023)

Evolutionary adaptation on protein fitness landscapes typically involves adaptive walks, where proteins move toward higher fitness through small, beneficial mutations. However, due to the rugged nature of many fitness landscapes, proteins can become trapped in local optima, unable to further evolve without accepting deleterious mutations or undergoing significant genetic changes. Despite these challenges, proteins exhibit remarkable evolvability, capable of adapting to new functions or environments through processes like directed evolution (Romero & Arnold, 2009).

3
METHODOLOGY

3.1 Self-Assembly Mechanisms

3.1.1 Sortase Linker

Sortase linkers represent a vital tool in protein engineering and bioconjugation, particularly through the use of sortase A, a transpeptidase enzyme derived from Staphylococcus aureus. Sortase A recognizes a specific pentapeptide sequence, LPXTG, in substrate proteins, cleaving between the threonine (T) and glycine (G) residues. This cleavage creates a thioacyl intermediate, which then reacts with an oligoglycine nucleophile, resulting in the formation of a new peptide bond (Guimaraes et al., 2013)

This reaction forms the foundation of the sortase-mediated protein ligation and labelling technique, allowing scientists to conjugate proteins, peptides, or other molecules with remarkable precision. The versatility of this method stems from its ability to facilitate a range of applications, such as protein labelling, site-specific bioconjugation, and the generation of protein-protein fusions, enhancing the study and manipulation of proteins in various contexts (Guimaraes et al., 2013).

3.2 Cloud based predictions

3.2.1 Structural Predictions Studies

Protein design involves the prediction and manipulation of protein structures to achieve desired functionalities. In this study, several state-of-the-art protein prediction tools were utilized:

1. AlphaFold Colab - AlphaFold is a deep learning-based method developed by DeepMind for predicting protein structures. The ColabFold implementation provided accessibility to AlphaFold's capabilities, enabling protein folding predictions for a wide range of users (Mirdita et al., 2022).

2. ROSETTAFOLD - ROSETTAFOLD is a protein structure prediction tool based on the Rosetta energy function and deep learning techniques. The provided Colab notebook facilitated the use of ROSETTAFOLD for protein structure prediction tasks (Baek et al., 2023).
3. ESMFOLD - ESMFOLD is another protein structure prediction method based on deep learning techniques. The Colab notebook provided access to ESMFOLD for predicting protein structures (Lin et al., 2022)

3.2.2 MD Simulations

MD simulations are computational techniques used to study the behaviour of molecules over time. In this study, cloud-based molecular simulations were performed using the following methodology:

- Amber Notebook - The Amber software suite, a widely used tool for MD simulations, was employed for conducting simulations. Specifically, the ff19SB force field, which incorporates amino acid-specific backbone parameters, was utilized for enhanced accuracy. Additionally, the TIP3P water model was selected for its compatibility with the ff19SB force field (Tian et al., 2020)
- ff19SB Force Field: The ff19SB force field is known for its improved accuracy in reproducing amino acid-dependent properties and amino acid-specific Ramachandran Map. It incorporates amino acid-specific backbone parameters and pairs best with the OPC water model (Tian et al., 2020) .
- TIP3P Water Model: The TIP3P water model is a widely used model for water molecules in MD simulations, providing accurate representation of water properties (Mark & Nilsson, 2001).

Cloud-Based Simulations - The provided Colab notebook facilitated the execution of MD simulations in a cloud-based environment, making the simulations accessible to a wide range of users. This approach enabled efficient utilization of computational resources and streamlined the simulation workflow (Arantes et al., 2021)

3.3 Genetic Algorithm

Genetic Algorithms (GAs) are parallel optimization algorithms distinguished by their independence from gradients. These methods rely on a performance criterion to assess candidate solutions and employ a population to explore the search space in pursuit of a global optimum. GAs excel in handling complex and irregular search spaces, making them well-suited for addressing

high-dimensional, non-linear optimization challenges. The core components of a genetic algorithm typically include a mutator, a crossover and a selector, however for this project only a mutator and a selector were used.

```
Algorithm: Genetic Algorithm for Protein Sequence Optimization

1. Data Preparation:
    - Load data from "filepath"
    - Preprocess data:
        * Drop rows with missing protein sequences
        * Encode protein sequences into numerical format
        * Scale features and target variable

2. Model Training:
    - Train the specified machine learning model using the preprocessed
data

3. Initialize Genetic Algorithm:
    - Randomly initialize a population of encoded protein sequences
    - Set generation counter to 0

4. Define Fitness Function:
    - Fitness function uses model prediction to calculate negative ΔG

5. Genetic Algorithm Execution:
    While (generation < maxGenerations) do:
        a. Fitness Evaluation:
            - Evaluate the fitness of each individual in the population us-
ing the fitness function

        b. Selection:
            - Apply tournament selection to choose parents for the next
generation

        c. Mutation:
            - Perform mutation to generate offspring

        d. Population Update:
            - Replace the old population with the new offspring
            - Increment the generation counter

        e. Logging:
            - Optionally log the current generation and best fitness

6. Post-Processing:
    - Decode the best overall solution from numeric format back to amino
acid sequence
    - Output the best solution and its fitness score

7. Visualization:
    - Plot the trend of fitness over generations to visualize algorithm
progression

End Procedure
```

3.3.1 PyGAD

The PyGAD's (Gad, 2021) GA class is configured as follows:

```
ga_instance = pygad.GA(num_generations=10000,
                       num_parents_mating=5,
                       fitness_func=fitness_function,
                       sol_per_pop=10,
                       num_genes=X_train_flattened.shape[1],
                       gene_type=int,
                       mutation_percent_genes=10,
                       mutation_type="random",
                       mutation_by_replacement=True,
                       initial_population=X_train_flattened[:10],  # Assuming initial_population is like this for simplicity
                       on_generation=callback_generation)
ga_instance.run()
```

Figure 3 - Implementation of the GA algorithm using PyGAD.

3.3.1.1 Detailed Explanation of Parameters:

- **`num_generations=10000`:** This parameter defines the number of generations for which the genetic algorithm will run, allowing the population to evolve over 10,000 generations to refine the solutions towards an optimal set.
- **`num_parents_mating=5`:** This specifies that 5 parents will participate in the mating process each generation, promoting genetic diversity and helping to avoid local optima by combining features from multiple good solutions.
- **`fitness_func=fitness_function`:** The fitness function, `fitness_function`, is designed to calculate the fitness of each individual in the population. For this application, it likely computes the negative of the change in Gibbs free energy (ΔG) to assess the stability of the protein variants.
- **`sol_per_pop=10`:** Indicates the population size, with 10 individual solutions in each generation. This number provides a balance between genetic diversity and computational efficiency.
- **`num_genes=X_train_flattened.shape[1]`:** Represents the number of genes (features) each individual in the population will have, set to match the number of features in the preprocessed training data.
- **`gene_type=int`:** The genes are represented as integers, which aligns with the encoding of amino acids or specific mutation types in the protein sequence.
- **`mutation_percent_genes=10`:** With 10% of the genes subject to mutation, this rate encourages exploration of the search space by introducing variability without destabilizing convergence.
- **`mutation_type="random"`:** The random mutation type alters genes in a completely stochastic manner, which is essential for exploring potentially beneficial mutations that have not yet been considered.

- **`mutation_by_replacement=True`:** By setting this to true, mutated genes are assigned a new value from the possible gene range, as opposed to modifying the existing value, ensuring diverse genetic representation.
- **`initial_population=X_train_flattened[:10]`:** The initial population is seeded with the first 10 flattened training data points. This approach can provide a head start to the optimization process by beginning with potentially promising candidates.
- **`on_generation=callback_generation`:** A callback function that is executed after each generation, allowing dynamic monitoring and logging of the algorithm's progress, and making it possible to implement custom stopping criteria or adjustments.

3.3.2 Fitness Function

The fitness function is designed to evaluate the potential solutions generated by a genetic algorithm, where each solution represents a numerically encoded protein sequence. This evaluation is accomplished by employing a CNN, XGBooster, KNN, Random Forests and Support Vector Regression (SVR) model, which predicts the change in Gibbs free energy (ΔG), a measure of protein stability.

3.3.2.1 Dataset

The data used for the training was obtained through Pro-Therm Database (Nikam et al., 2021), more specifically, the *E.coli* organism. The dataset was split into training and testing sets using a 80-20 split ratio. This partitioning ensured that the model could be trained on a sufficiently large portion of the data while retaining a separate subset for evaluation.

3.3.2.2 Convoluted Neural Network Architecture

The input to the convolutional neural network (CNN) is a sequence of amino acids represented by their ASCII character codes, which are padded to a fixed length $L = 100$. The target variable, ΔG (delta G), is normalized using a standard scaler. The CNN processes the input through a series of layers as follows:

- Input Layer - The input to the network is represented as a 2D tensor $X \in \mathbb{R}^{N \times 1}$, where $N = 100$ is the fixed length of the sequence.
- Convolutional Layer -The first convolutional layer applies $F_1 = 64$ filters, each of size 3×13×1:

$$Y_1 = \mathrm{ReLU}(W_1 * X + b_1)$$

where W_1 represents the filters, $*$ denotes the convolution operation, and b_1 is the bias. The output Y_1 is of shape $N_1 \times F_1$, where $N_1 = 98$.

- Batch Normalization 1: Batch normalization is applied to the output of the first convolutional layer:

$$Y_1' = \mathrm{BatchNorm}(Y_1)$$

Batch normalization standardizes the outputs by normalizing each batch's mean and variance followed by a scale and shift operation.

- Max Pooling Layer 1 : Max pooling with a pool size of 22 is applied to reduce the spatial dimensions:

$$P_1 = \mathrm{MaxPool}\ (Y_1')$$

The output P_1 has a shape $N_1 / 2 \times F_1$, where $N_1/2 = 49$.

- Convolutional Layer 2: The second convolutional layer applies $F_2 = 64$ filters, each of size 3×1:

$$Y_2 = \mathrm{ReLU}(W_2 * P_1 + b_2)$$

The output Y_2 is of shape $N_2 \times F_2$, where $N_2 = 47$.

- Batch Normalization 2: Batch normalization is applied to the output of the first convolutional layer:

$$Y_2' = \mathrm{BatchNorm}(Y_2)$$

- Max Pooling Layer 2: Max pooling with a pool size of 2 is applied again to reduce the spatial dimensions:

$$P_2 = \mathrm{MaxPool}\ (Y_2')$$

The output P_2 has a shape $N_2 / 2 \times F_2$, where $N_2/2 = 23$.

- Flatten Layer: the flattened output $f \in \mathbb{R}^{23 \times 64}$ is transformed into a one-dimensional vector:

$$f = \mathrm{Flatten}\ (P_2)$$

- Dense Layer: A fully connected dense layer with 128 neurons processes the flattened output:

$$d = \mathrm{ReLU}(W_d f + b_d)$$

where $W_d \in \mathbb{R}^{128 \times 23 \times 64}$ and $b_d \in \mathbb{R}^{128}$.

- Batch Normalization 3 : Batch normalization is applied to the output of the dense layer:

$$\mathrm{d}' = \mathrm{BatchNorm}(\mathrm{d})$$

- Dropout Layer : A dropout layer with a dropout rate of 0.50.5 is applied:

$$\mathrm{d}'' = \mathrm{Droupout}(\mathrm{d}', 0.5)$$

This helps prevent overfitting by randomly setting half of the activations to zero during training.

- Output Layer: Finally, a dense output layer with a single neuron and linear activation is used to predict ΔG:

$$\hat{y} = \mathrm{W_o d}'' + \mathrm{b_o}$$

where $\mathrm{W_o} \in \mathbb{R}^{1\times 128}$ and $\mathrm{b_o} \in \mathbb{R}^1$. The resulting output $\hat{y}$ is a scalar.

3.3.2.3 XGBooster

The XGBoost regression model f_{xgb} is trained on M training examples $\{(X_i, y_i)\}_{i=1}^{M}$, where X_i is the encoded protein sequence and y_i is observed change in Gibbs free energy (ΔG).
The model is defined as :

$$\mathrm{f_{xgb}(X;\ \Theta)} = \sum_{\mathrm{k=1}}^{\mathrm{K}} f_k(X)$$

This means the fitness function evaluates to the negative of the output of the XGBoost model, which predicts the change in Gibbs free energy based on the encoded protein sequence.

3.3.2.4 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm where the function is only approximated locally, and all computation is deferred until function evaluation. The KNN algorithm predicts the outcome for a new instance (in this case, a protein sequence) based on the majority vote or average of the K-nearest points in the feature space. Let X be the dataset of encoded protein sequences and y the vector of observed changes in Gibbs free energy corresponding to each sequence in X. For a new sequence x, the KNN prediction $\hat{y}$ is computed as:

$$\hat{y}(x) = \frac{1}{K} \sum_{x_i \in N_K(x)} y_i$$

where $N_K(x)$ denotes the set of K closest sequences in the training set X to x, determined by a chosen distance metric.
The fitness function for the genetic algorithm is designed to maximize or minimize the predicted change in Gibbs free energy $\hat{y}$, depending on whether the objective is to increase or

decrease stability. For minimization (common in stability studies where lower ΔG is favorable), the fitness function can be expressed as:

$$F(x) = -\hat{y}(x)$$

This means the fitness function computes the negative of the predicted ΔG, as typically genetic algorithms are set to maximize the fitness score. The goal is to find a protein sequence that minimizes the predicted ΔG, thus enhancing stability.

3.3.2.5 Random Forest

A Random Forest is an ensemble learning method for regression that operates by constructing a multitude of decision trees at training time and outputting the mean prediction of the individual trees:

$$f_{\mathrm{RF}}(X;\Theta) = \frac{1}{N}\sum_{n=1}^{N} f_n(X)$$

where:

- X is the input feature matrix for the encoded protein sequences.
- Θ represents the ensemble of trees parameters.
- N is the number of trees in the forest.
- $fn(X)$ is the prediction from the n-th decision tree.

The fitness function FF is constructed to utilize the Random Forest model's prediction as a means to guide the genetic algorithm toward optimal protein sequences. Specifically, the function is designed to minimize the predicted ΔG, or maximize the negative of it, as the genetic algorithm conventionally maximizes the fitness value:

$$F(x) = -f_{\mathrm{RF}}(x;\Theta)$$

where:

- x is the encoded protein sequence.
- $F(x)$ represents the fitness score associated with sequence x , calculated as the negative predicted ΔG by the Random Forest model.

3.3.2.6 SVR

Support Vector Regression (SVR) is employed to estimate real-valued functions. The objective of the SVR is to find a function $f(x)$ that has at most ε deviation from the obtained targets y for all the training data, and at the same time is as flat as possible. The SVR model is trained using the following formulation:

$$\mathrm{f(x)} = \langle \mathrm{w}, \mathrm{x} \rangle + \mathrm{b}$$

where:

- ⟨w,x⟩ denotes the dot product between the weights w and the feature vector x.
- b is the bias term.

The model parameters ww and bb are optimized by solving the following optimization problem:

$$\min_{w,b,\xi,\xi^*} = \frac{1}{2}|w|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

subject to:

$$y_i - \langle w, x_i \rangle - b \leq \epsilon + \xi_i$$
$$\langle w, x_i \rangle + b - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0$$

where:

- $\xi_i \; and \; \xi_i^*$ are slack variables that allow for deviations larger than ϵϵ for individual data points.
- C is a regularization parameter that controls the trade-off between the flatness of f and the amount up to which deviations larger than ϵϵ are tolerated.

The fitness function F in the genetic algorithm utilizes the prediction from the SVR model to guide the search for optimal protein sequences. It is designed to maximize the negative of the predicted ΔG, as lower values of ΔG indicate higher stability:

$$F(x) = -f_{SVR}(x; \Theta)$$

where:

- x is the encoded protein sequence.
- Θ denotes the parameters of the trained SVR model.
- $f_{ry}(x;\Theta)$ represents the SVR model's prediction of ΔG for sequence x.

3.3.2.7 Model Compilation and Training

The model is trained to minimize the mean squared error (MSE) loss function:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y_i})^2$$

where MM is the number of samples. The optimizer used is the Adam optimizer, which adapts the learning rate during training. Additionally, the mean absolute error (MAE) is used as an evaluation metric:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}|y_i - \hat{y_i}|$$

The trained model was evaluated on the held-out test set to assess its performance. The test loss and MAE were computed to gauge the model's predictive accuracy.

Finally, the training history, including loss and MAE, was visualized using line plots. Additionally, the model's predictions were compared with the actual ddG values through scatter plots to analyse the correlation between predicted and observed values. Histograms of absolute errors were plotted to visualize the distribution of prediction errors. These visualizations provided insights into the model's performance and highlighted areas for potential improvement.

4

RESULTS

4.1 Protein Structural Studies

Protein PT8 underwent structural prediction using three state-of-the-art algorithms: AlphaFold2, RosettaFold, and ESMFold (Fig. 4). These algorithms offer distinct methodologies and accuracies in protein folding simulations.

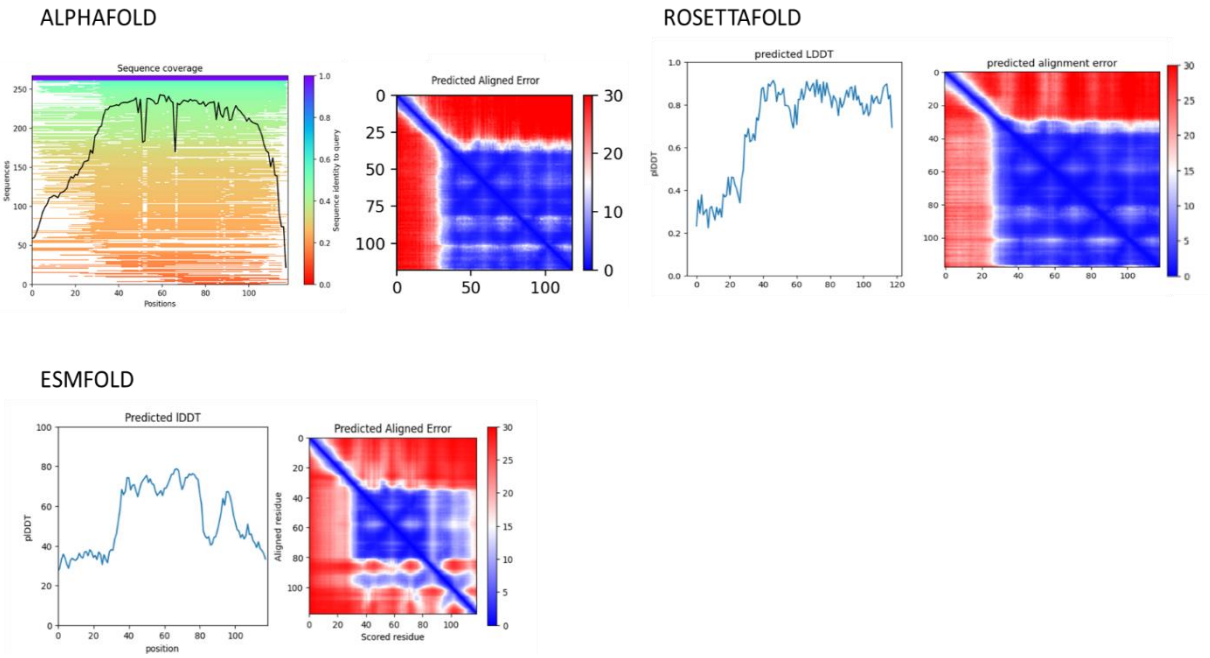

Figure 4 - Comparative visualization of protein structure prediction results using AlphaFold, RosettaFold, and ESMPredict models. The plots show sequence coverage, predicted local distance difference test (LDDT) scores, and predicted alignment errors for a sample protein sequence. For AlphaFold and RosettaFold, sequence coverage is overlaid with per-residue confidence scores (green line), while LDDT scores are shown in a blue graph. Predicted alignment error heatmaps reveal the error distribution across all positions, indicating areas of high uncertainty (red) versus high confidence (blue). For ESMPredict, only LDDT scores and predicted alignment errors are shown, demonstrating the model's prediction capabilities in structural accuracy and alignment precision.

AlphaFold's sequence coverage of PT8 shows extensive coverage across the entire sequence, indicating robust predictions (Figure 6, AlphaFold panel). The green shaded areas suggest high confidence across various protein segments, crucial for identifying functionally important domains. The predicted alignment error heatmap shows per-residue alignment error. Regions of lower error, indicated in deeper blue, suggest higher confidence in the structural predictions made by AlphaFold. On the other hand, RosettaFold's analysis provided Local Distance Difference Test (LDDT) scores, which predominantly appear high across the sequence, suggesting accurate and reliable structure prediction (Figure 6, RosettaFold panel). The LDDT score fluctuations indicate lesser confidence in certain areas, potentially corresponding to disordered or flexible regions. The alignment error heatmap complements the LDDT scores, showing regions of varying error rates, with lower errors indicative of stable, well-folded regions. Finally, ESMFold's analysis revealed variability in the Integrated Distance Difference Test (IDDT) scores across the PT8 protein sequence (Figure 6, ESMFold panel). The peaks and troughs in the IDDT graph suggest areas with varying prediction confidence, essential for further detailed biophysical or functional studies. The predicted alignment error heatmap similarly shows regions with more or less reliability in the predicted structure.

4.2 Molecular Simulations

The molecular dynamics simulations reveal distinct differences between PT8 and PT8-Sortase in terms of structural stability, flexibility, and interaction dynamics. PT8 maintains a more stable and compact structure, while PT8-Sortase exhibits increased flexibility and dynamic changes, likely influenced by the Sortase component. These differences are crucial for understanding how Sortase fusion affects the behavior and function of PT8, which could have implications for its biological activity and potential applications.

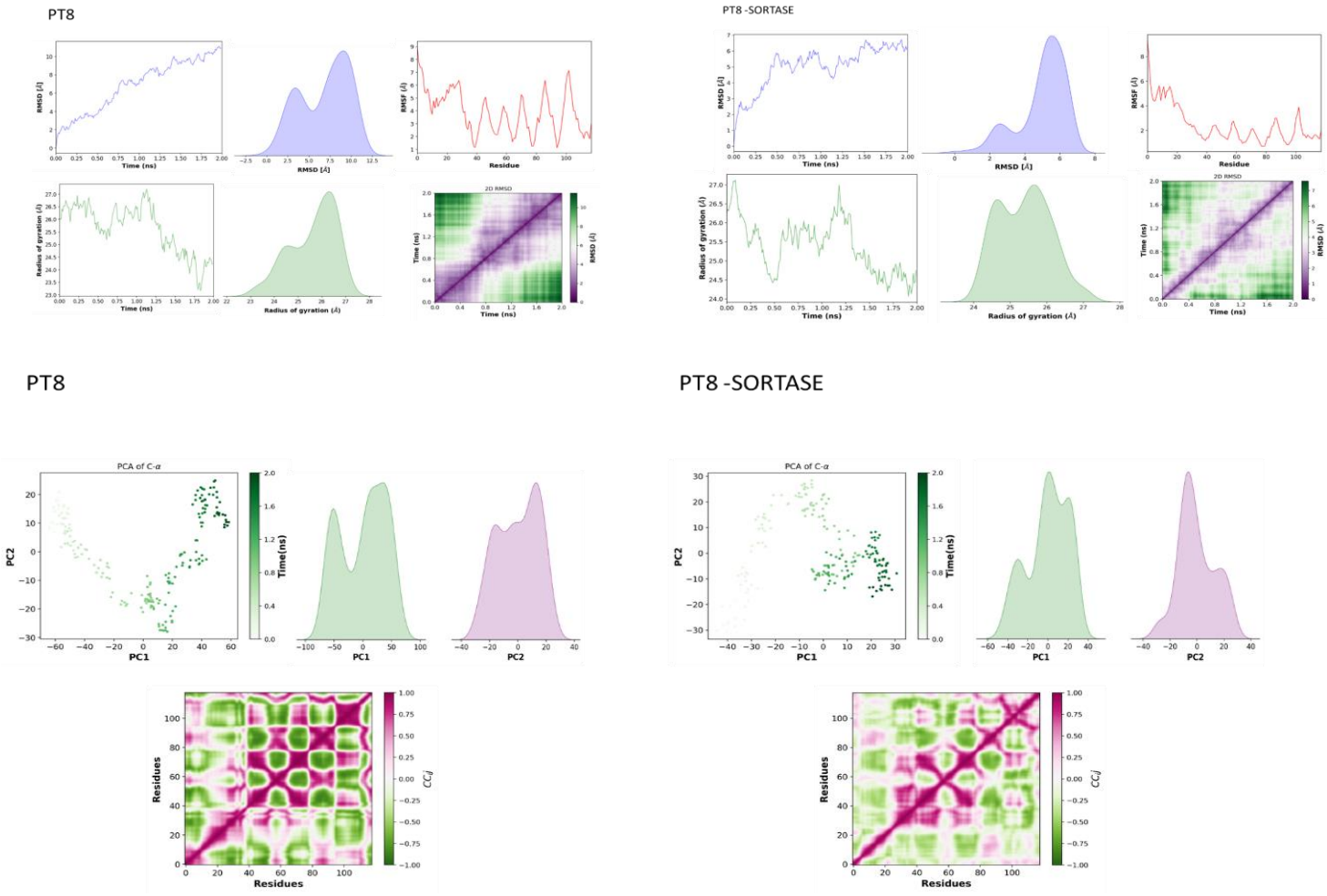

Figure 5 - Molecular Dynamics Simulations for PT8 and PT8-Sortase: This comprehensive figure showcases multiple aspects of molecular dynamics simulations for the proteins PT8 and PT8-Sortase. For both proteins, the panels from top to bottom include Root Mean Square Deviation (RMSD) over time, indicating the stability of the protein structures; Root Mean Square Fluctuation (RMSF) per residue, highlighting areas of flexibility; the Radius of Gyration, reflecting the compactness of the protein structures over the simulation period; a 2D RMSD heatmap, showing the conformational changes over time; and Principal Component Analysis (PCA) of C-alpha atoms, which analyzes the major movements in the protein structure. The final row presents the contact maps, illustrating residue-residue interactions within the protein structures. These comprehensive analyses provide insights into the structural dynamics, stability, and interaction patterns of PT8 and PT8-Sortase, with comparisons highlighting the influence of the Sortase modification on PT8's structural behaviour.

4.2.1 PT8 Simulations

4.2.1.1 RMSD (Root Mean Square Deviation) Over Time

The RMSD plot for PT8 shows a gradual increase initially, stabilizing around 1.5 Å, indicating that PT8 achieves a stable conformation relatively early in the simulation. This stabilization suggests that PT8 has a robust structure under the simulated conditions.

4.2.1.2 RMSF (Root Mean Square Fluctuation) Per Residue

The RMSF plot shows fluctuations across the residues. Peaks in this plot indicate regions of the protein that exhibit greater flexibility during the simulation. High flexibility in certain areas may indicate functional sites such as binding pockets or flexible loops.

4.2.1.3 Radius of Gyration

This plot displays a relatively consistent radius of gyration for PT8, suggesting that the protein maintains a stable, compact structure throughout the simulation. This compactness is important for the structural integrity and functionality of the protein.

4.2.1.4 2D RMSD

The 2D RMSD heatmap shows how the conformational space of the protein evolves over time. Darker areas along the diagonal suggest periods of conformational stability, while off-diagonal bright spots could indicate transitions between different conformations.

4.2.1.5 PCA of C-alpha Atoms

Principal Component Analysis (PCA) of the C-alpha atoms shows clustering of data points, suggesting limited conformational changes in the core structure of PT8. The first two principal components do not show wide dispersion, indicating that large-scale motions are not prevalent.

4.2.1.6 Contact Map

The contact map provides insights into residue-residue interactions throughout the simulation. Consistent patterns in the map suggest stable intra-protein interactions, important for maintaining the structural integrity of PT8.

4.2.2 PT8-Sortase Simulations

4.2.2.1 RMSD Over Time

The RMSD for PT8-Sortase shows higher fluctuations compared to PT8 alone, indicating that the addition of Sortase introduces more dynamic instability or conformational flexibility to the complex.

4.2.2.2 RMSF Per Residue

Similar to PT8, the RMSF plot for PT8-Sortase reveals regions of high flexibility. The presence of Sortase may affect the flexibility of certain regions, potentially altering the functional dynamics of the protein complex.

4.2.2.3 Radius of Gyration

The radius of gyration for PT8-Sortase is slightly higher and shows more variation compared to PT8 alone, suggesting a less compact structure. This could be due to the Sortase part impacting the overall folding and compactness of the fusion protein.

4.2.2.4 2D RMSD

The 2D RMSD heatmap for PT8-Sortase shows more pronounced off-diagonal features compared to PT8, suggesting more frequent or pronounced conformational changes throughout the simulation, which could be attributed to the influence of the Sortase fusion.

4.2.2.5 PCA of C-alpha Atoms

The PCA plot for PT8-Sortase shows a broader dispersion of the first two principal components compared to PT8 alone, indicating that the Sortase fusion introduces additional conformational variability into the system.

4.2.2.6 Contact Map

The contact map for PT8-Sortase shows altered interaction patterns compared to PT8 alone. Changes in contact patterns can indicate alterations in the structural dynamics and interactions within the protein complex, potentially influencing its biological functions.

4.3 Protein Optimization

4.3.1 Fitness Function

Figure 6 plots show the actual versus predicted ddG values for various machine learning models, including a CNN, XGBoost, KNN, Random Forest, and SVR. These plots are crucial for understanding how well each model performs in predicting the change in protein stability, with the ideal scenario being that the points lie close to the diagonal line from the bottom left to the top right, indicating perfect predictions. Let's delve deeper into the performance of each model based on the scatter plots:

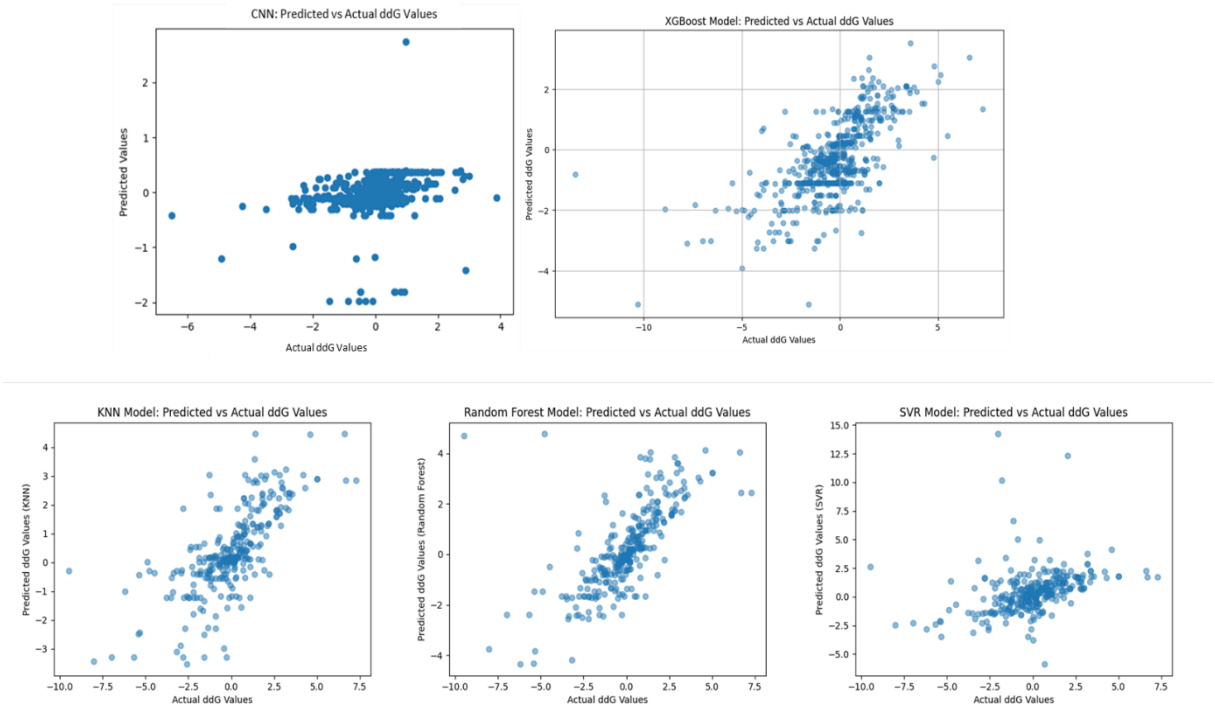

Figure 6 - Comparison of actual vs. predicted ddG values for five different machine learning models. From top-left to bottom-right: CNN, XGBoost, KNN, Random Forest, and SVR. Each plot illustrates the model's ability to predict changes in protein stability.

The CNN shows predictions that are tightly clustered around the center of the y-axis, particularly within the range of -1 to 1, regardless of the actual ddG values which range from approximately -10 to 10. This suggests that the CNN might be underfitting, as it fails to capture the higher magnitude variations in the ddG values. It might be too simplistic or not properly trained to capture the complexity of the dataset. The XGBoost model shows a better spread across the diagonal, indicating a more robust prediction across the range of ddG values. However, there's noticeable scatter around the line, particularly for higher absolute values of ddG. This indicates

that while XGBoost is capable of capturing more complexity than the CNN, it still struggles with extreme values or outliers. Its performance could potentially be improved with further parameter tuning or by addressing overfitting through regularization techniques. The scatter plot for the KNN model demonstrates a reasonable spread across the range of ddG values, but like XGBoost, it also shows considerable scatter. The variance around the line of perfect prediction is quite high. This suggests that the KNN model captures the variability in the dataset but may not provide consistent predictions across all types of inputs. The performance might be influenced by the choice of k (number of neighbors) and tuning this parameter could potentially yield better results. The Random Forest plot shows a dense cluster of points around the diagonal line, particularly for values between -5 and 5, which indicates good performance for a significant range of ddG values. This model appears to perform well across diverse inputs, managing to maintain accuracy without as much variance as seen in KNN and XGBoost. The ensemble nature of Random Forests, which combines multiple decision trees, likely helps in achieving a more generalized model. The SVR model's predictions are mostly concentrated in the lower range of predicted values, with a limited spread across the range of actual ddG values, indicating a tendency to underpredict the ddG changes. This constrained prediction range might be due to the nature of the kernel used or the regularization parameters in the SVR. The model might benefit from revisiting these configurations or exploring different kernels that can capture a wider variance in the data.

4.3.2 Evolution of Fitness

Figure 7 displays the fitness landscapes over generations for the five machine learning models—CNN, XGBoost, KNN, Random Forest (RF), and Support Vector Regression (SVR)—provides insight into how the performance of each model evolved during optimization through a genetic algorithm. Each plot tracks the best fitness score obtained by the models as the genetic algorithm progresses through generations, up to a total of 10,000 generations. Here's a detailed description and interpretation of each model's performance based on the fitness landscapes:

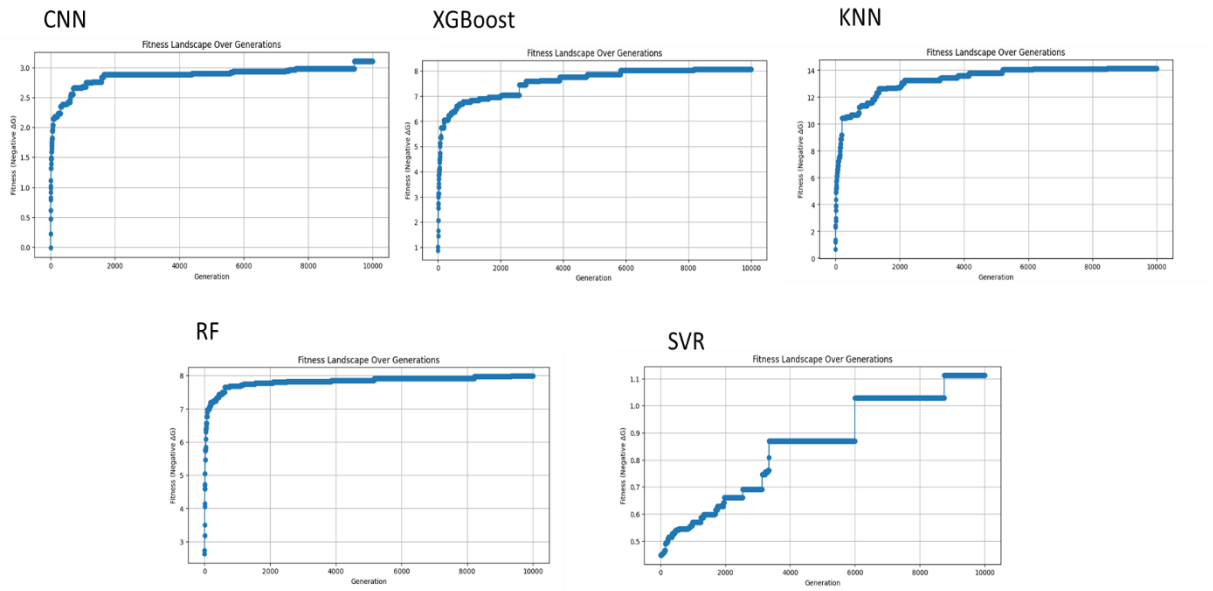

Figure 7 - Fitness landscapes over generations for five predictive models: CNN, XGBoost, KNN, RF (Random Forest), and SVR. The plots show the best fitness scores achieved by each model as the number of generations increases from 0 to 10,000. Rapid improvements fitness is observed in the early generations across all models, followed by plateaus indicating convergence to optimal or near-optimal solutions. The CNN, XGBoost, and RF models display higher fitness scores and earlier stabilization, suggesting quicker convergence compared to KNN and SVR, which exhibit more gradual improvements and later stabilization in fitness scores.

The CNN model shows a rapid increase in fitness at the early stages of the generations, achieving high fitness levels quickly, which then plateau relatively early on in the optimization process. This indicates that the CNN model quickly finds a near-optimal solution but doesn't significantly improve beyond this point. The early plateau suggests that while the CNN model effectively captures some patterns in the data sufficient to reach a good level of prediction accuracy, its learning capacity may be limited beyond a certain complexity, possibly due to its architecture or the nature of the data. The XGBoost model also demonstrates a swift improvement in fitness, which stabilizes at a high level early in the generations. The fitness remains constant

after reaching its peak, showing no further improvements. The performance indicates that XGBoost is highly effective at adapting and improving its predictions rapidly. Its ability to handle various types of features and data distributions efficiently may contribute to this fast convergence to an optimal solution. The KNN model shows a more gradual improvement in fitness over many generations. The progression is less steep compared to CNN and XGBoost, with improvements continuing into later generations before stabilizing.This suggests that KNN's performance is sensitive to the genetic algorithm's evolution process, possibly due to its reliance on the local neighborhood of data points. As the genetic algorithm explores more of the solution space, KNN gradually adapts its predictions.

Similar to CNN and XGBoost, the RF model's fitness rapidly increases and plateaus early. The plateau at a high fitness level indicates a strong initial performance. This performance reflects the Random Forest's ability to generalize well from its ensemble approach, combining predictions from multiple decision trees. This helps it quickly reach a robust solution that doesn't significantly improve with further generations, likely due to already capturing the necessary predictive patterns in the data.

SVR's fitness landscape shows a staggered, stepwise improvement in fitness, which is different from the other models. The improvements are marked by distinct jumps, suggesting phases of learning separated by periods of stability. This pattern might be due to the nature of SVR's optimization process, which is highly dependent on the setting of its hyperparameters like kernel type, C (regularization parameter), and gamma. The jumps could represent adjustments in the model's understanding of the data structure.

Overall, the results illustrate varied dynamics in how each model approaches optimization, with CNN, XGBoost, and RF showing quick and early convergence, suggesting these models might be capturing essential data patterns swiftly. In contrast, KNN and SVR exhibit more incremental learning, which could be advantageous in scenarios where gradual improvement is preferable. The rapid plateauing in most models suggests that longer training with more generations may not necessarily lead to better solutions, highlighting the importance of effective early-stage model tuning and potential overfitting considerations.

4.3.3 Model Performance

Finally, Table 1 presents performance metrics for five different machine learning models: CNN (Convolutional Neural Network), KNN (k-Nearest Neighbors), XGBoost (eXtreme Gradient Boosting), RF (Random Forest), and SVR (Support Vector Regression). These metrics include the fitness values at two generation milestones—50 and 10,000—and the Mean Absolute Error (MAE) for each model. Here's a detailed analysis of each model's performance based on the data provided:

Table 1 - Comparison of fitness values and Mean Absolute Error (MAE) across different machine learning models at two key generation milestones (50 and 10,000) during optimization.

Model	Fitness Value		MAE
	50 Gen	10000 Gen	
CNN	1.974	3.103	0.628
KNN	7.173	14.124	1.115
XGBoost	4.478	8.059	1.021
RF	6.359	7.986	0.890
SVR	0.448	1.112	0.976

CNN: At generation 50, the fitness value is 1.974, which increases to 3.103 at generation 10,000. This indicates an improvement in the model's optimization over time. The MAE of 0.628 is the lowest among all the models, suggesting that CNN has the best overall prediction accuracy, particularly in terms of average error magnitude per prediction.

KNN: : Starts with a fitness value of 7.173 at generation 50, rising to 14.124 at generation 10,000, showing substantial improvement, the largest increase among all models. The MAE of 1.115 is the highest, indicating that despite its fitness improvement, KNN's predictions are on average less accurate compared to other models.

XGBoost: Begins at 4.478 and grows to 8.059. This moderate increase suggests that the model benefits from longer training periods but may be approaching a plateau. With an MAE of 1.021, XGBoost offers moderate accuracy, better than KNN but not as good as CNN or RF.

RF: Shows an initial fitness of 6.359, increasing to 7.986. The growth is less dramatic compared to KNN, indicating a steadier or possibly more mature optimization process. An MAE of 0.890 places RF second in terms of accuracy, combining robustness in handling diverse data with reasonable error levels.

SVR: Has the lowest initial fitness value at 0.448, which only marginally improves to 1.112. This suggests either an initial poor adaptation to the dataset or a slow learning process. The MAE of 0.976 is relatively high, better than KNN and XGBoost, suggesting it has moderate accuracy.

5
DISCUSSION

5.1 Polymorphic Toxins

Polymorphic toxins are a class of proteinaceous toxins that are present in the natural environment and are secreted through various systems. These effectors can vary In size, sequence and structure, however share a similar multidomain architecture (Ruhe et al., 2020).

5.1.1 Structure

Polymorphic toxins are characterized by starting with an N-terminal trafficking domain, followed by repetitive elements such as Rearrangement Hot-Spots or hemagglutinin repeats, culminating in a C-terminus toxic domain (Nachmias et al., 2021). The operon structure of these proteins show the presence of also cognate adjacent immunity genes downstream of the toxin domain, thus conferring the cell protection against autointoxication from toxins produced by clone mates (Jamet & Nassif, 2015).

A hallmark of these antibacterial toxins is their cognate immunity proteins. These protein protect the bacterium against autointoxication or from toxins produced by other cells (Jamet & Nassif, 2015). A conserved N-terminal region fused to variable toxic domains defines a polymorphic toxin family. A toxic domain of the same family can be found fused with N-terminal regions of distinct polymorphic toxin families, suggesting a shared pool of toxic domains. Interestingly, some N-terminal regions can be associated with either antibacterial or antieukaryotic toxic. - (Jamet & Nassif, 2015)

5.1.2 PT8

PT8 (protein ID EKO85988.1) is 118 amino acids long novel protein and specific to the Leptospiraceae family, specifically in the human pathogen Leptospira interrogans. Following the standard polymorphic protein operon structure, PT8 is composed of 8 domains divided into two toxin domains, two cognate adjacent immunity genes, three repeat domains and a single traffic domain (Fig 8).

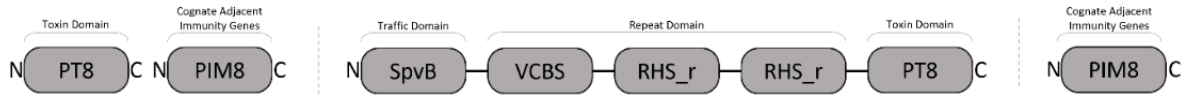

Figure 8 - PT8 Operon Structure

Structural analyses conducted using a predictive model with a reliability of 50% reveal the presence of four conserved surface residues on the protein: HIS 94, ASP 80, ARG 74, and ASP 70. Moreover, in vivo investigations affirm that this protein possess a common lethal function. According to Nachmias et al (2021) , the cloning of genes in E. coli and S. cerevisiae was followed by their cultivation on agar plates supplemented with either an inducer (arabinose or galactose) or a repressor (glucose). Importantly, PT8 along with the other 8 genes exhibited significant toxicity to E.coli with up to a four orders of magnitude reduction in colony-forming units. Furthermore, nearly all toxins demonstrated inhibitory effects on growth in liquid media according to PT activity analysis depicted in Figure 8.

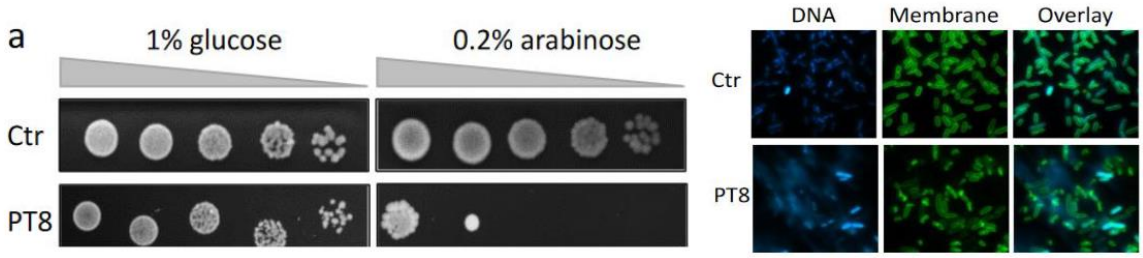

Figure 9 - In vivo studies of PT8: On the left the inducer-repressot experiment. On the right proof of concenpt of PT8's cellular toxicity.

In conformity with other polymorphic proteins, PT8 also has an active cognate immunity gene PIM8 that prevents autointoxication as reported by Nachmias et al. (2021) in Figure 9. To enable

the production of substantial quantities of PT8 polymorphic toxins within model microcellular organisms, it is imperative that these proteins do not manifest any toxic activity within the host cell. Accordingly, it becomes necessary to devise a mechanism of self-assembly via protein engineering.

5.1.3 Interpretation of Structural and Simulation Results

The use of AlphaFold (Jumper et al., 2021), RosettaFold (Baek et al., 2021), and ESMFold (Lin et al., 2023) provided comprehensive insights into the PT8 structure. AlphaFold's predictions showed high confidence across various protein segments, indicating a robust structure, which is essential for ensuring that engineered proteins maintain their functionality under different conditions. The analysis revealed that certain domains of PT8 exhibited higher flexibility. These domains could be crucial for the protein's biological activity, as flexibility often correlates with the ability to undergo conformational changes that are necessary for interacting with other biomolecules (Ma et al., 2011). Identifying these regions provides a targeted approach for modifying PT8 to enhance or inhibit specific interactions, which could be used to adjust the protein's toxicity or binding affinity. The high-confidence structural predictions facilitate the design of mutations in specific regions to enhance protein stability or alter functional properties. For example, the conserved surface residues (HIS 94, ASP 80, ARG 74, and ASP 70) identified could be targeted for mutation to see how these changes affect the protein's interaction with potential substrates or inhibitors - Figure 10.

Additionally, molecular dynamics simulations revealed nuanced differences between PT8 and PT8-Sortase, particularly in terms of structural dynamics and stability. The stable conformation of PT8, as indicated by the RMSD analysis, suggests that the native protein can maintain its structure under physiological conditions. This stability is advantageous for biotechnological applications where the protein might be subject to various stresses (Strickler et al., 2006). For PT8-Sortase, the increased flexibility might affect the protein's ability to function consistently, which could be leveraged to reduce toxicity when introducing PT8 into new cellular environments.

The analysis of RMSF and the radius of gyration for PT8-Sortase indicated that the fusion with Sortase increases the protein's dynamic range. This could potentially expose new epitopes or

binding sites, facilitating interactions with other molecules or cellular components, which could be used to direct the toxin to specific targets within a cell.

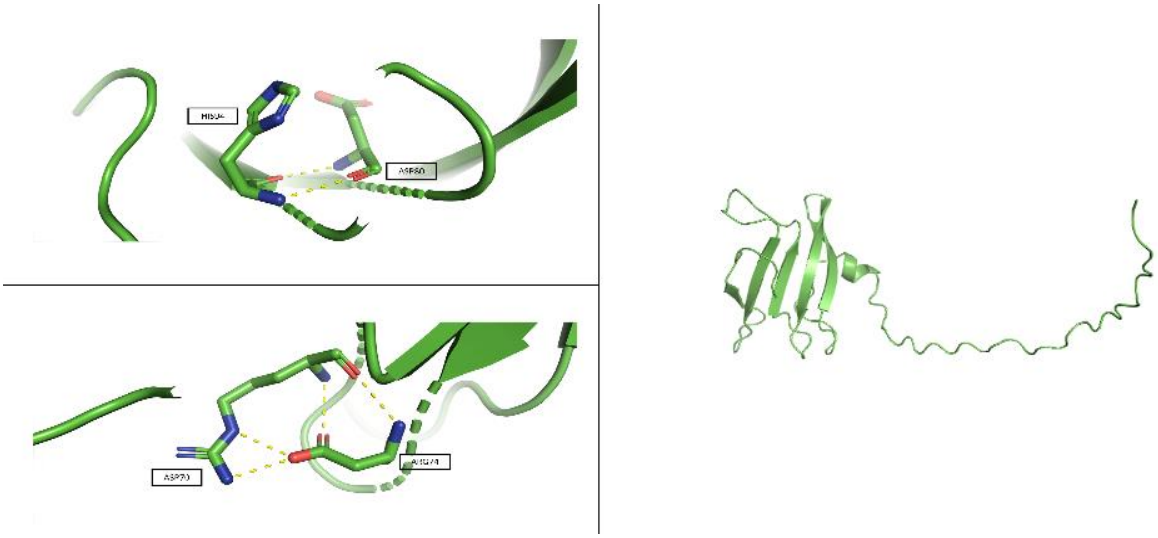

Figure 10 - The image depicts molecular interactions and conformational structures of the PT8 polymorphic toxin. On the left, detailed views show specific amino acid interactions within PT8, highlighting hydrogen bonds and hydrophobic interactions, which are crucial for its stability and function. On the right, a broader structural representation of PT8 illustrates its overall tertiary structure, extending from a well-folded domain into an unstructured tail, signifying areas of rigidity and flexibility within the molecule.

5.1.4 Protein Optimization and Machine Learning Models

The CNN model provided the lowest Mean Absolute Error (MAE), suggesting high predictive accuracy for changes in protein stability (ΔG). This indicates that deep learning can effectively capture complex nonlinear relationships between amino acid sequences and their stability. This result aligns with findings from Senior et al. (2020), who demonstrated that deep learning, especially using convolutional neural networks, can predict protein structures with high accuracy. Their work underscores the capacity of these models to understand the intricate nonlinear relationships between amino acid sequences and their resultant protein stability (Senior et al., 2020).

Adding to this, Pancotti et al. (2021) developed a sequence-based deep learning method to predict protein stability changes upon genetic variations. Their study successfully applied deep learning to model how genetic mutations influence protein stability, reinforcing the idea that these computational approaches can effectively capture the complex dynamics between sequence variations and structural stability (Pancotti et al., 2021). In a related study, Wang et al.

(2016) applied deep convolutional neural fields to predict protein secondary structures, a key component influencing overall protein stability. Their findings illustrate how deep learning can effectively predict aspects of protein structure that directly correlate with stability, highlighting the broader applicability of these models in understanding protein dynamics (Wang et al., 2016). Lastly, the work by Parthiban et al. (2006) illustrates the long-standing potential of neural networks to predict stability changes for single-site mutations in proteins. Their early use of neural networks laid the groundwork for the sophisticated deep learning models we use today to explore and predict protein stability changes in more complex scenarios (Parthiban et al., 2006).

The XGBoost and Random Forest models showed good performance but with some variance in predictions, especially for extreme ΔG values. This suggests that ensemble methods can generalize well but may require additional tuning to handle outliers or extreme cases effectively. In contrast, the KNN model's gradual improvement over generations indicates that instance-based learning can adaptively refine predictions, which is useful for exploring diverse mutational effects. This variance in predictions by ensemble methods like XGBoost and Random Forest, especially for extreme ΔG values, highlights the need for additional tuning to handle outliers effectively (Raschka, 2020).

The fitness landscapes illustrated how different models adapted over generations. The early plateau observed for CNN and Random Forest suggests that these models quickly find a near-optimal solution, highlighting the efficiency of these approaches for protein engineering. In contrast, the gradual improvements seen with KNN and SVR underscore the potential for these models to explore more of the sequence space, potentially finding unconventional or novel variants that other models might miss.

The insights from machine learning models can guide the selection of mutations to improve protein stability and functionality. For instance, targeting the flexible regions identified in the RMSF analysis for stabilization can enhance the protein's resilience, while mutations in the conserved surface residues could modify the protein's interaction profile.

The success of the genetic algorithm in optimizing PT8 variants suggests that integrating these approaches with other computational techniques, like reinforcement learning or Bayesian optimization, could further enhance the efficiency and creativity of protein design processes.

By optimizing PT8 for non-toxicity, this research contributes to developing safer proteins for biotechnological applications. Future work will involve in vivo experiments to validate the optimized proteins' functionality and safety, ensuring they perform as expected in more complex biological systems.

The methodology developed here is not limited to PT8; it can be applied to a wide range of proteins. By automating parts of this process and incorporating more adaptive learning mechanisms, this approach can be scaled to handle larger datasets and more diverse protein families, speeding up the discovery of novel protein variants.

This study provides a concrete example of how machine learning and genetic algorithms can be melded to advance our understanding of protein dynamics and engineering. The detailed analysis of fitness landscapes, in particular, offers insights into how proteins evolve over time and how these changes can be directed towards specific functional outcomes..

5.1.5 Disadvantages of the Current Methods

While the genetic algorithm-driven approach to protein engineering has demonstrated significant promise in enhancing PT8 polymorphic toxins, it is important to acknowledge the potential limitations and challenges associated with these methods.

5.1.5.1 Computational Resource Intensity

The use of advanced machine learning models like CNNs and ensemble methods (e.g., XGBoost, Random Forest) involves significant computational resources, especially when processing large datasets or conducting extensive parameter tuning. The training and optimization processes can be particularly resource-intensive, requiring high-performance computing environments or extensive cloud computing resources to achieve reasonable time frames for model convergence.

While the genetic algorithm provides a robust framework for exploring the protein sequence space, its scalability can be challenged by the exponential growth of possible mutations as the sequence length increases. This combinatorial explosion can make the algorithm less efficient for larger proteins or more complex mutational landscapes, potentially necessitating the use

of more sophisticated sampling strategies or heuristic methods to maintain computational feasibility.

Canziani et al. discuss the computational demands of various neural network architectures, including CNNs, underscoring the need for extensive computational resources to achieve model convergence in reasonable time frames (Canziani et al., 2017)

5.1.5.2 Model-Specific Drawbacks

Machine learning models, especially deep neural networks like CNNs, are prone to overfitting, especially when trained on limited data or without adequate regularization. Overfitting can lead to models that perform well on training data but fail to generalize to unseen data, which could mislead the optimization process in the genetic algorithm. This goes in line with the finds of Zhang et al. (2017) that demonstrate that CNNs, are prone to overfitting, especially without proper regularization, affecting their ability to generalize to new data (Zhang et al., 2017)

Many machine learning models, particularly those involving deep learning, are often considered "black-box" approaches. This means that while they can make accurate predictions, interpreting these models to understand how specific sequence features influence the predictions can be challenging. This opacity can limit the ability to derive clear mechanistic insights from the models (Lipton, 2017).

The performance of machine learning models is heavily dependent on the availability of high-quality and representative training data (Domingos, 2012). In the context of protein engineering, inaccuracies in the initial ΔG predictions or biases in the training data can lead to suboptimal or misleading guidance for the genetic algorithm.

5.1.6 Usage of Cloud-Based Methods

The use of cloud-based computational methods has been a cornerstone of this research, enabling the handling of complex simulations and machine learning tasks without the need for extensive local computational infrastructure.

Cloud-based resources can be scaled up or down based on the computational demand, making it easier to manage the resource-intensive tasks of protein simulations and machine learning model training. This scalability ensures that large datasets can be processed more efficiently, and complex models can be trained faster (Armbrust et al., 2010).

By using cloud resources, researchers can access high-performance computing power without the upfront capital investment in physical hardware. This approach is particularly cost-effective for projects that require intense computation for short periods, such as batch processing of simulation data or training of complex models (Armbrust et al., 2010)..

Cloud platforms can facilitate easier sharing of data and models among research teams, enhancing collaborative efforts. Additionally, the use of standardized environments can improve the reproducibility of computational experiments, as other researchers can rerun the same simulations or training processes under identical settings. According to (Arantes et al., 2021) cloud-based molecular simulations, make powerful simulation tools accessible to a broader community of researchers and eliminating the barriers related to the need for extensive computational infrastructure.

On the other hand, when using cloud-based services, data security and privacy become paramount, especially when dealing with sensitive biological data. Ensuring that data are encrypted and securely stored, and that access is controlled rigorously, is essential to prevent unauthorized access (Kawsar et al., 2010).

While cloud platforms offer flexibility, they also introduce complexity in terms of setup, management, and optimization of computational resources. Properly configuring cloud resources to maximize performance and cost-efficiency requires expertise in cloud computing and can be a learning curve for researchers primarily focused on biological sciences.

5.1.7 Relationship Between ΔG (ddG) and Toxicity

Understanding the relationship between changes in Gibbs free energy (ΔG, often referred to as ddG when discussing changes due to mutations) and toxicity is crucial for interpreting how protein stability alterations can impact biological functions.

For many toxins, including polymorphic toxins like PT8, stability can influence their ability to interact with cellular targets. Increased stability can enhance the functional activity of a toxin by ensuring that its active conformation is preserved, especially under varying environmental conditions(Manning & Colón, 2004). However, if the aim is to reduce toxicity for safe biotechnological applications, it is often desirable to decrease stability (increase ΔG) to reduce the likelihood of the toxin adopting its active conformation in the host organism.

By targeting mutations that increase the ΔG of PT8, the genetic algorithm-driven approach can be used to design variants of the protein that are less stable and hence less likely to exert toxic effects. This strategy is aligned with the goal of producing non-toxic variants of PT8 for biotechnological applications.

While reducing toxicity is important, it is also necessary to retain sufficient stability for the protein to perform its desired function in biotechnological applications. This balance requires carefully choosing mutations that slightly increase ΔG without completely destabilizing the protein structure, thereby maintaining functionality while reducing toxicity (Manning & Colón, 2004).

5.1.8 Conclusions

In conclusion, while the genetic algorithm-driven approach to protein engineering used in this study has shown significant promise in enhancing the properties of PT8 polymorphic toxins, it also faces certain challenges, particularly in terms of computational demands and model interpretability. The use of cloud-based methods has facilitated this research but comes with its own set of challenges. Most importantly, understanding the nuanced relationship between ΔG and toxicity has been crucial for guiding the engineering of PT8 variants towards reduced toxicity and enhanced biotechnological utility. Future work will continue to refine these methods, aiming for an optimal balance of stability, functionality, and safety in engineered proteins.

5.1.9 Future Work

Applying the genetic algorithm and machine learning framework to other polymorphic toxins or proteins could help validate the universality of this approach. This would help determine if the optimization strategies developed here can be broadly applied.

Combining computational predictions with experimental data could refine the models further. For instance, iterative loops where computational designs are synthesized and tested, and the resulting data fed back into the model, could enhance accuracy and reliability. However, three important methodological improvements should be considered: enhanced sampling techniques, hybrid models, and broader data integration.
Firstly, enhanced sampling techniques such as meta dynamics or umbrella sampling, could provide more accurate insights into the conformational landscape of proteins. Secondly hybrid models that combine the strengths of different machine learning approaches, like neural net works with ensemble methods, could improve predictive performance and robustness. Finally incorporating more diverse datasets, including more variants and conditions, could help improve the generalizability of the models. Using meta-learning or transfer learning might allow models trained on one set of proteins to adapt more quickly to others.

Future work could expand the genetic algorithm to optimize multiple properties simultaneously, such as stability, activity, and toxicity. This would more closely mirror the multi-faceted challenges of real-world protein engineering.

BIBLIOGRAPHY

Arantes, P. R., Polêto, M. D., Pedebos, C., & Ligabue-Braun, R. (2021). Making it Rain: Cloud-Based Molecular Simulations for Everyone. *Journal of Chemical Information and Modeling*, *61*(10), 4852–4856. https://doi.org/10.1021/acs.jcim.1c00998

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50–58. https://doi.org/10.1145/1721654.1721672

Baek, M., Anishchenko, I., Humphreys, I. R., Cong, Q., Baker, D., & DiMaio, F. (2023). *Efficient and accurate prediction of protein structure using RoseTTAFold2*. https://doi.org/10.1101/2023.05.24.542179

Baek, M., DiMaio, F., Anishchenko, I., Dauparas, J., Ovchinnikov, S., Lee, G. R., Wang, J., Cong, Q., Kinch, L. N., Schaeffer, R. D., Millán, C., Park, H., Adams, C., Glassman, C. R., DeGiovanni, A., Pereira, J. H., Rodrigues, A. V., van Dijk, A. A., Ebrecht, A. C., … Baker, D. (2021). *Accurate prediction of protein structures and interactions using a three-track neural network*.

Bricco, A. R., Miralavy, I., Bo, S., Perlman, O., Farrar, C. T., McMahon, M. T., Banzhaf, W., & Gilad, A. A. (2022). *Protein Optimization Evolving Tool (POET) based on Genetic Programming* [Preprint]. Synthetic Biology. https://doi.org/10.1101/2022.03.05.483103

Bryngelson, J. D., & Wolynes, P. G. (1987). Spin glasses and the statistical mechanics of protein folding. *Proceedings of the National Academy of Sciences*, *84*(21), 7524–7528. https://doi.org/10.1073/pnas.84.21.7524

Canziani, A., Paszke, A., & Culurciello, E. (2017). *An Analysis of Deep Neural Network Models for Practical Applications* (arXiv:1605.07678). arXiv. http://arxiv.org/abs/1605.07678

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78–87. https://doi.org/10.1145/2347736.2347755

Freschlin, C. R., Fahlberg, S. A., & Romero, P. A. (2022). Machine learning to navigate fitness landscapes for protein engineering. *Current Opinion in Biotechnology*, *75*, 102713. https://doi.org/10.1016/j.copbio.2022.102713

Gad, A. F. (2021). *PyGAD: An Intuitive Genetic Algorithm Python Library* (arXiv:2106.06158). arXiv. http://arxiv.org/abs/2106.06158

Groth, P. M., Michael, R., Salomon, J., Tian, P., & Boomsma, W. (2023). *FLOP: Tasks for Fitness Landscapes Of Protein wildtypes*. https://doi.org/10.1101/2023.06.21.545880

Guimaraes, C. P., Witte, M. D., Theile, C. S., Bozkurt, G., Kundrat, L., Blom, A. E. M., & Ploegh, H. L. (2013). Site-specific C-terminal and internal loop labeling of proteins using sortase-mediated reactions. *Nature Protocols*, *8*(9), 1787–1799. https://doi.org/10.1038/nprot.2013.101

Jamet, A., & Nassif, X. (2015). New Players in the Toxin Field: Polymorphic Toxin Systems in Bacteria. *mBio*, *6*(3), e00285-15. https://doi.org/10.1128/mBio.00285-15

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), 583–589. https://doi.org/10.1038/s41586-021-03819-2

Kawsar, F., Kortuem, G., & Altakrouri, B. (2010). Supporting interaction with the Internet of Things across objects, time and space. *2010 Internet of Things (IOT)*, 1–8. https://doi.org/10.1109/IOT.2010.5678441

Khakzad, H., Igashov, I., Schneuing, A., Goverde, C., Bronstein, M., & Correia, B. (2023). A new age in protein design empowered by deep learning. *Cell Systems*, *14*(11), 925–939. https://doi.org/10.1016/j.cels.2023.10.006

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Fazel-Zarandi, M., Sercu, T., Candido, S., & Rives, A. (2022). *Language models of protein sequences at the scale of evolution enable accurate structure prediction*.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., & Rives, A. (2023). Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, *379*(6637), 1123–1130. https://doi.org/10.1126/science.ade2574

Lipton, Z. C. (2017). *The Mythos of Model Interpretability* (arXiv:1606.03490). arXiv. http://arxiv.org/abs/1606.03490

Ma, B., Tsai, C.-J., Haliloğlu, T., & Nussinov, R. (2011). Dynamic Allostery: Linkers Are Not Merely Flexible. *Structure*, *19*(7), 907–917. https://doi.org/10.1016/j.str.2011.06.002

Makhatadze, G. I., & Privalov, P. L. (1996). On the entropy of protein folding. *Protein Science*, *5*(3), 507–510. https://doi.org/10.1002/pro.5560050312

Manning, M., & Colón, W. (2004). Structural Basis of Protein Kinetic Stability: Resistance to Sodium Dodecyl Sulfate Suggests a Central Role for Rigidity and a Bias Toward β-Sheet Structure. *Biochemistry*, *43*(35), 11248–11254. https://doi.org/10.1021/bi0491898

Mark, P., & Nilsson, L. (2001). Structure and Dynamics of the TIP3P, SPC, and SPC/E Water Models at 298 K. *The Journal of Physical Chemistry A*, *105*(43), 9954–9960. https://doi.org/10.1021/jp003020w

Mirdita, M., Schütze, K., Moriwaki, Y., Heo, L., Ovchinnikov, S., & Steinegger, M. (2022). ColabFold: Making Protein folding accessible to all. *Nature Methods*. https://doi.org/10.1038/s41592-022-01488-1

Nachmias, N., Dotan, N., Fraenkel, R., Rocha, M. C., Kluzek, M., Shalom, M., Rivitz, A., Shamash-Halevy, N., Cahana, I., Deouell, N., Klein, J., Schlezinger, N., Tzarum, N., Oppenheimer-Shaanan, Y., & Levy, A. (2021). *Systematic Discovery of Antibacterial and Antifungal Bacterial Toxins* [Preprint]. Microbiology. https://doi.org/10.1101/2021.10.19.465003

Nikam, R., Kulandaisamy, A., Harini, K., Sharma, D., & Gromiha, M. M. (2021). ProThermDB: Thermodynamic database for proteins and mutants revisited after 15 years. *Nucleic Acids Research*, *49*(D1), D420–D424. https://doi.org/10.1093/nar/gkaa1035

Notin, P., Rollins, N., Gal, Y., Sander, C., & Marks, D. (2024). Machine learning for functional protein design. *Nature Biotechnology*, *42*(2), 216–228. https://doi.org/10.1038/s41587-024-02127-0

Onuchic, J. N., Luthey-Schulten, Z., & Wolynes, P. G. (1997). THEORY OF PROTEIN FOLDING: The Energy Landscape Perspective. *Annual Review of Physical Chemistry*, *48*(1), 545–600. https://doi.org/10.1146/annurev.physchem.48.1.545

Pancotti, C., Benevenuta, S., Repetto, V., Birolo, G., Capriotti, E., Sanavia, T., & Fariselli, P. (2021). A Deep-Learning Sequence-Based Method to Predict Protein Stability Changes Upon Genetic Variations. *Genes*, *12*(6), 911. https://doi.org/10.3390/genes12060911

Qing, R., Hao, S., Smorodina, E., Jin, D., Zalevsky, A., & Zhang, S. (2022). Protein Design: From the Aspect of Water Solubility and Stability. *Chemical Reviews*, *122*(18), 14085–14179. https://doi.org/10.1021/acs.chemrev.1c00757

Qiu, Y., & Wei, G.-W. (2023). *Mathematics-assisted directed evolution and protein engineering* (arXiv:2306.04658). arXiv. http://arxiv.org/abs/2306.04658

Raschka, S. (2020). *Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning* (arXiv:1811.12808). arXiv. http://arxiv.org/abs/1811.12808

Romero, P. A., & Arnold, F. H. (2009). Exploring protein fitness landscapes by directed evolution. *Nature Reviews Molecular Cell Biology*, *10*(12), 866–876. https://doi.org/10.1038/nrm2805

Ruhe, Z. C., Low, D. A., & Hayes, C. S. (2020). Polymorphic Toxins and Their Immunity Proteins: Diversity, Evolution, and Mechanisms of Delivery. *Annual Review of Microbiology*, *74*(1), 497–520. https://doi.org/10.1146/annurev-micro-020518-115638

Schellman, J. A. (1987). *The Thermodynamic Stability of Proteins*. *16:115-137*. https://doi.org/10.1146/annurev.bb.16.060187.000555

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W. R., Bridgland, A., Penedones, H., Petersen, S., Simonyan, K., Crossan, S., Kohli, P., Jones, D. T., Silver, D., Kavukcuoglu, K., & Hassabis, D. (2020). Improved protein structure prediction using potentials from deep learning. *Nature*, *577*(7792), 706–710. https://doi.org/10.1038/s41586-019-1923-7

Sormanni, P., Aprile, F. A., & Vendruscolo, M. (2018). Third generation antibody discovery methods: *In silico* rational design. *Chemical Society Reviews*, *47*(24), 9137–9157. https://doi.org/10.1039/C8CS00523K

Strickler, S. S., Gribenko, A. V., Gribenko, A. V., Keiffer, T. R., Tomlinson, J., Reihle, T., Loladze, V. V., & Makhatadze, G. I. (2006). Protein Stability and Surface Electrostatics: A Charged Relationship. *Biochemistry*, *45*(9), 2761–2766. https://doi.org/10.1021/bi0600143

Tian, C., Kasavajhala, K., Belfon, K. A. A., Raguette, L., Huang, H., Migues, A. N., Bickel, J., Wang, Y., Pincay, J., Wu, Q., & Simmerling, C. (2020). ff19SB: Amino-Acid-Specific Protein Backbone Parameters Trained against Quantum Mechanics Energy Surfaces in Solution. *Journal of Chemical Theory and Computation*, *16*(1), 528–552. https://doi.org/10.1021/acs.jctc.9b00591

Yang, K. K., Wu, Z., & Arnold, F. H. (2019). Machine-learning-guided directed evolution for protein engineering. *Nature Methods*, *16*(8), 687–694. https://doi.org/10.1038/s41592-019-0496-6

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). *Understanding deep learning requires rethinking generalization* (arXiv:1611.03530). arXiv. http://arxiv.org/abs/1611.03530

UNIVERSIDADE NOVA
DE LISBOA

2024

Genetic Algorithm-Driven Protein Engineering: Enhancing PT8 Polymorphic Toxins for Biotechnology

LUÍSA LISBOA MAIA CORREIA RODRIGUES
MASTER IN **COMPUTATIONAL BIOLOGY & BIOINFORMATICS**

NOVA
UNIVERSIDADE NOVA
DE LISBOA