UNIVERSIDADE D
COIMBRA

Gonçalo Coelho

DEEP GENERATIVE MODELS FOR PROTEIN REPOSITIONING

Dissertation in the context of the Master in Data Science and Engineering, advised by Prof. Joel P. Arrais and Nelson R. C. Monteiro presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September of 2023

This page is intentionally left blank.

Faculty of Sciences and Technology

Department of Informatics Engineering

Deep Generative Models for protein repositioning

Gonçalo dos Santos Coelho

Dissertation in the context of the Master in Data Science and Engineering, advised by Prof. Joel P. Arrais and Nelson R. C. Monteiro presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra

September 2023

UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

This work was developed in collaboration with:

Center for Informatics and Systems of the University of Coimbra

This page is intentionally left blank.

This page is intentionally left blank.

Acknowledgments

Gostaria de começar por agradecer ao Professor Joel P. Arrais pelo seu apoio e orientação ao longo deste percurso académico. As suas orientações e ensinamentos foram fundamentais para o sucesso deste trabalho, e estou verdadeiramente grato por ter tido a oportunidade de aprender e fazer parte em algo que ajudou a crescer não só intelectualmente, mas também pessoalmente.

Um agradecimento ao Nelson Monteiro, pelas revisões ao meu trabalho, fornecer ideias de como superar os problemas deste projeto, pela disponibilidade e a ajuda que sempre deu, ter apoiado e debatido as minhas ideias e a ajudar a perceber melhor como é o mundo da investigação. Em especial, por teres conseguido trazer em mim um lado mais rigoroso para o que faço no trabalho.

Quero agradecer aos meus pais por tudo o que fizeram por mim, o esforço, o sacrifício e terem feito os possíveis para conseguir estar onde estou hoje. Sei que sem vocês não estaria onde estou e o que sacrificaram por mim é insubstituível. Sei que nunca irei conseguir alcançar o que vocês fizeram por mim, mas espero que consiga retribuir de alguma forma.

Ao meu irmão por ter me aturado durante o tempo que esteve cá e que apesar de estar longe ainda é uma fonte de inspiração do que tenho a fazer. Quero também agradecer à minha sobrinha que acabou de nascer e me ter ajudado a manter acordado e focar na tese.

Aos meus amigos, especialmente aos que fiz na universidade, agradeço pelos eventos a que fomos, aos copos que bebemos, às quintas que foi possível eu ir, aos trabalhos, projetos e entregas de última e me ajudado a crescer numa melhor pessoa que sou hoje. Irei guardar todos estes momentos comigo pois foram os momentos mais alegres da universidade.

À minha cadela Kika que esteve comigo até ao último dia da entrega da tese e que

me acompanhou durante uns incríveis 14 anos, em que comeu metade dos meus trabalhos de casa e gostava de vaguear pela rua sozinha. Espero que esteja num lugar melhor.

Finalmente um grande obrigado à Quantunna pela experiência e a oportunidade de poder finalmente aprender o que é a música, como relaxar após épocas de exames chatas, reconhecer o valor nas tradições coimbrãs e ter viajado pelo país, e europa, com um grupo animado. Agradeço especialmente aos momentos aleatórias, às discussões e aos momentos de boémia que houve serão para sempre os momentos mais únicos e especiais que tive e que se voltasse ao inicio repetia.

Agradeço a todos outra vez porque sem vocês não seria o que seria hoje.

...

"Embrace the uncharted realms of knowledge, for in the fusion of human curiosity and AI's endless potential, lies the key to unlocking a future of boundless innovation and understanding."

ChatGPT

This page is intentionally left blank.

Resumo

Ao gerar uma proteína num laboratório, os cientistas isolam os genes que codificam a proteína de interesse e inserem-na num organismo hospedeiro, uma bactéria ou uma levedura. A cadeia de criação de novas proteínas é gerada, extraída, purificada e analisada. O processo de purificação é uma série de testes que exploram as propriedades físicas e químicas da proteína, como a hidrofobicidade e o tamanho. O processo de análise é mais para determinar a identidade e as funções da proteína, utilizando testes como o ELISA. Todo este processo é irregular e pode ser curto ou levar meses ou anos para completar o estudo da proteína.

A conceção de proteínas desempenha um papel importante e é uma forma inovadora de gerar proteínas de-novo. Este método permite criar proteínas "à medida" para tarefas específicas, como o reforço do sistema imunitário.

Por isso, algumas estratégias computacionais foram apresentadas ao longo do tempo para ajudar neste processo; o AlphaFold ajudou a prever a forma das proteínas, o modelo Rosetta ajudou a obter a energia mínima e a criação de proteínas 3-D específicas para uma função necessária, o Alinhamento de Sequências compara proteínas rapidamente para que se possa detetar se há alguma mutação, eliminação ou inserção, se pertencem à mesma família de proteínas para que possam ter a mesma função.

Neste trabalho, exploramos o universo da geração de proteínas através da implementação de diferentes modelos, nomeadamente em modelos de Processamento de Linguagem Natural (PLN). Para isso, analisamos a estrutura e a sequência das proteínas, encontramos uma forma de validar as sequências proteicas e, em seguida, implementamos um modelo capaz de capturar a complexidade das sequências proteicas e, posteriormente, gerar proteínas válidas.

Palavras-Chave

Conceção de Proteínas, Geração de Proteínas, Redes Neuronais baseadas em sequências, Aprendizagem Profunda.

This page is intentionally left blank.

Abstract

When generating a protein in a laboratory, scientists isolate the genes that code for the protein of interest and insert it into a host organism, a bacteria or yeast. The pipeline of creation of new proteins is generated, extracted, purified, and analyzed. The purification process is a series of tests that exploit the physical and chemical properties of the protein, such as hydrophobicity and size. The analysis process is more to determine the protein's identity and functions using tests like ELISA. This entire process is irregular and can be short or take months or years to complete the protein study.

Protein Design plays an important role and is a novel way to generate de-novo proteins. This method makes it possible to make "custom-made" proteins for specific tasks, such as boosting the immune system.

On that account, some computational strategies have been presented over time to aid in this process; AlphaFold helped in predicting proteins' shape, Rosetta's model helped in obtaining the minimum energy and the creation of 3-D proteins specific to a function needed, Sequence Alignment compares proteins quickly so it can be detected if there are any mutation, deletions or insertion if they belong to the same family of proteins so they can have the same function.

In this work, we explore the universe of protein generation by implementing different models, especially in Natural Language Processing (NLP) models. To do this, we analyze the structure and sequence of proteins, find a way to validate protein sequences, and then implement a model capable of capturing the complexity of protein sequences and later generating valid proteins.

Keywords

Protein Design, Protein Generation, Sequence-based Neural Networks, Deep Learning.

This page is intentionally left blank.

Contents

Abbreviations **xix**

List of Tables **xxi**

List of Figures **xxiii**

1 Introduction **1**

1.1 Context 1

1.2 Motivation 2

1.3 Objectives 3

1.4 Document Structure 4

2 Background **5**

2.1 Bioinformatics 5

2.1.1 Proteins Overview 5

2.2 Sequence Alignment 10

2.2.1 Smith-Waterman 11

2.2.2 Needleman-Wunsch 12

2.2.3 BLAST 13

2.2.3.1 Non-Redundant Database Integration 13

3 State of the Art **15**

3.1 Computational Protein Engineering 15

4 Methods **19**

4.1 Artificial Neural Networks 19

4.1.1 Recurrent Neural Network 22

4.1.2 Long Short-Term Memory 22

4.1.3 Sequence to Sequence 24

4.1.4 Transformer 25

4.1.5 Reinforcement Learning 28

4.1.5.1 Policy Gradient 30

4.1.5.2 Actor-Critic 31

4.1.5.3 Deep Q-Network 31

4.2 Protein Folding 32

4.2.1 AlphaFold 32

4.2.2 OmegaFold 33

4.3 Metrics 34

4.3.1 BLAST "E-value" 34

4.3.2 plDDT 35

5 Framework **37**

5.1 Generator 38

5.1.1 Parameterization for Different Models 38

5.2 Environment 40

5.3 Visualization 43

5.4 Dataset 44

6 Results and Discussion **47**

6.1 Shortcomings 59

7 Conclusion **61**

7.1 Future Work 62

A Appendix **63**

Bibliography **65**

This page is intentionally left blank.

Abbreviations

Adam Adaptive Moment Estimation

AdamW Adaptive Moment Estimation with Weight Decay

ANN Artificial Neural Network

BLAST Basic Local Alignment Search Tool

BPTT Backpropagation Through Time

CWGAN Conditional Wasserstein Generative Adversarial Network

DNN Deep Neural Network

DNA Deoxyribonucleic Acid

DQN Deep Q-Network

GAN Generative Adversarial Network

GRU Gated recurrent units

GPT Generative Pre-trained Transformer

IQR Interquartile Range

LSTM Long Short-Term Memory

ML Machine Learning

MSE Mean Squared Error

MSA Multiple Sequence Alignment

MLP Multilayer Perceptron

MLE Maximum Likelihood Estimation

NCBI National Center for Biotechnology Information

NMR Nuclear Magnetic Resonance

NLP Natural Language Process

NR Non-redundant

PLM Protein Language Model

PG Policy Gradient

plDDT Predicted Local Distance Difference Test

RL Reinforcement Learning

RNN Recurrent Neural Networks

RNA Ribonucleic Acid

Seq2Seq Sequence to Sequence

SMILES Simplified Molecular Input Line Entry System

SW Smith-Waterman

TBPTT Truncated Backpropagation Through Time

VAE Variational Autoencoders

List of Tables

5.1 Parameters settings for the stacked Long Short-Term Memory (LSTM) model. 40

5.2 Parameters settings for the transformer model. 40

5.3 Runtime AlphaFold vs OmegaFold based on the length of the protein sequence [77] 41

6.1 Validation results for 200 proteins using OmegaFold's plDDT values for four different approaches. 47

6.2 Secondary Structure Composition Analysis of 200 Protein Sequences Generated from Scratch and 368 Proteins with Fully Known and Valid Structures: A Per-Protein Classification Perspective 53

6.3 Comparison of Amino Acid Distribution in 200 Generated Proteins and Natural Occurring Proteins 56

A.1 Amino Acids 63

A.2 Ambiguous Amino Acids 64

This page is intentionally left blank.

List of Figures

2.1 Protein Structure Levels: Primary, Secondary, Tertiary, Quaternary [28] 9

4.1 LSTM Cell 23

4.2 Sequence to Sequence (Seq2Seq) Architecture 25

4.3 Transformer Architecture Overview 26

4.4 Multihead and Scale-Dot Product Attention 27

4.5 AlphaFold's Structure [6] 33

4.6 OmegaFold's Structure [73] 34

5.1 Framework Overview 37

5.2 An example of a protein structure visualization in the AlphaFold style. The visualization highlights the secondary structure elements and overall fold of the protein. 43

5.3 Protein vs AlphaFold Confidence Distribution 45

5.4 Preprocessing process in the Swiss-Prot dataset 45

6.1 Comparison of Protein Structure Prediction Models: Box Plot depicting the distribution of Predicted Local Distance Difference Test (plDDT) values for the Transformer with No Reinforcement Learning (RL), Transformer with RL, LSTM, and Random 48

6.2 Chimeric Protein Structure Generated by the Proposed Model 58

This page is intentionally left blank.

List of Algorithms

1 Policy Gradient: REINFORCE 30
2 Training:2nd phase 42

This page is intentionally left blank.

This page is intentionally left blank.

1

Introduction

1.1 Context

Designing novel and structurally functional proteins remains one of the most significant challenges in bioinformatics research. Protein generation aims to develop machine-learning models and deep-learning architectures to generate novel and functional proteins for various applications, such as drug discovery, enzyme design, and protein engineering.

Machine learning models for protein generation typically work by learning the statistical patterns of existing protein sequences in a given dataset and then using these patterns to generate new sequences similar to the input data. There is a plethora of Machine Learning architectures such as Recurrent Neural Networks (RNN), Generative Adversarial Network (GAN), Variational Autoencoders (VAE), or Transformers to generate new sequences.

Several challenges associated with protein generation using machine learning include ensuring the generated sequences are biologically feasible, stable and have desired functions. Additionally, generating novel sequences sufficiently different from existing sequences is challenging. Recent advances in machine learning and computational biology have shown promising results in generating novel and functional protein sequences [1].

Overall, protein generation using machine learning models has the potential to lead to significant breakthroughs in bioinformatics and decisive findings across drug discovery, protein engineering, and other applications in the life sciences.

1.2 Motivation

Proteins play a vital role in various biological processes and are the building blocks of life. They exhibit an incredible diversity of functions, structures, and interactions, crucial to understanding cellular processes, diseases, and drug design [2,3]. The traditional approach to studying proteins involves experimental techniques, which can be costly, time-consuming, and limited by the vast complexity of protein structures and sequences. In recent years, deep neural networks have emerged as promising tools for protein generation due to their ability to learn complex patterns and features from large datasets [4].

One of the most significant challenges in protein generation lies in predicting the 3D structure of a protein from its amino acid sequence. Proteins can fold into many unique 3D conformations, and understanding this 3D structure is crucial to deciphering their function. Experimentally determining protein structures through X-ray crystallography or Nuclear Magnetic Resonance (NMR) [5] is time-consuming and resource-intensive. A computational approach is template-based and does not consider multiple conformations. Deep neural networks offer a new paradigm, as they can capture the complex relationships between protein sequences and their corresponding 3D structures from existing structural data [6].

As protein generation using deep neural networks is a relatively nascent field, there is a need to have well-established methods and models. Developing robust and accurate models requires extensive research and validation to ensure their applicability across different protein families and functions. Moreover, creating large and high-quality datasets of protein sequences with annotated structures and functions is challenging due to the diversity and complexity of proteins [7].

Another significant hurdle in protein generation is that there is no universal formula to determine the structure or function of a protein solely from its sequence. The relationship between sequence and structure/function is highly intricate and context-dependent, making it challenging to develop a single standardized model. Deep neural networks can excel in recognizing patterns in data. With sufficient training on diverse datasets, they can capture the essence of these complex relationships, offering valuable insights into protein structure and function prediction.

However, protein sequences can vary significantly in length, ranging from a few dozen to several thousand amino acids [7]. The vast number of possible amino acid

arrangements makes it challenging to manually explore the vast sequence space. Additionally, proteins can suffer various post-translational modifications, increasing complexity. Deep neural networks can efficiently handle large-scale datasets and learn intricate patterns [8], allowing them to capture the diverse composition and length variations in protein sequences.

The sequence of a protein is not merely a linear string of amino acids; it also possesses complex spatial and structural arrangements. The arrangement of secondary structures like alpha-helices, beta-sheets, and loops significantly influences the protein's function [9]. Deep neural networks, especially those designed for sequence-to-sequence mapping or language modelling, can inherently account for context dependencies and hierarchical patterns, enabling them to generate sequences with desired structural features [10].

Achieving a capable model can benefit greatly. Besides possibly passing the need for X-ray crystallography, it can help expand the protein universe since the vast majority of proteins in nature remain unexplored due to the vastness of sequence space [11]. This expansion of the protein universe offers exciting prospects for discovering new functional proteins, enzymes, and potential therapeutic targets. By generating the 3D structures of proteins and their potential binding sites, researchers can virtually screen large compound libraries to identify drug candidates that may interact with specific proteins [12, 13]. This rational drug design approach can significantly speed up drug development, saving time and resources.

Moreover, several pathologies are connected to proteins with aberrant conformations or lack of function due to genetic mutations within the coding DNA region [14]. Thus, understanding how specific mutations affect protein structure and function is crucial for developing targeted therapies. Computational methods can help predict the impact of mutations on protein stability, folding, and interactions, offering valuable information for precision medicine. It can assist in identifying non-functional proteins, allowing for personalized drug design and treatment strategies based on the patient's specific protein profiles [15].

1.3 Objectives

The main goal of this master thesis is to explore and develop deep learning models capable of generating novel protein sequences with specific functions and properties. The leading objectives to fulfil are the following:

1. Build a dataset of curated proteins with 1D sequential data
2. Introduce a Deep Learning Model that generates proteins.
3. Exploit the capabilities of extracting the context of a sequence in the RNNs.
4. Build a baseline model to compare the results.
5. Build a deep learning architecture model based on a transformer.
6. Investigate novel training methodologies to enhance the model's performance.
7. Explore a validation process for the generated proteins.
8. Evaluation and validation of the performance of the models in identifying and validating.

1.4 Document Structure

The remainder of this document is organised into six different chapters. It begins with an introduction that presents the protein context, the work's impetus, and the objectives to be achieved to solve the proposed problem. Chapter 2, Background, introduces the main concepts needed to understand bioinformatics. Chapter 3, State of the Art, shows the main computational approaches used in protein discovery. Chapter 4, Methods, describes the methodology employed throughout the research work associated with this thesis, including the data and treatment used. Chapter 5, Framework, explains the architecture and the procedure to train and validate the model. Chapter 6, Results, presents the results and the discussion obtained from each model proposed. Finally, Chapter 7, Conclusion, concludes the master thesis and presents future approaches and possibilities for the proposed work.

2

Background

2.1 Bioinformatics

Bioinformatics focuses on exploring computational approaches to solve mostly challenges within the biology domain, to analyse and interpret biological data, for example, proteins, Deoxyribonucleic Acid (DNA), Ribonucleic Acid (RNA), bacteria, genome [16].

One of the main areas of research in bioinformatics involves protein classification, prediction and generation based on its primary attributes [17]. It can involve predicting the structure and function of the proteins. The tools and techniques include sequence alignment [18], protein structure prediction and molecular dynamics simulations.

Overall, bioinformatics is pivotal in driving many advances in biology and medicine. It provides new tools to interpret complex biological data, explore yet unknown data and discover new treatments that can improve human health and well-being.

2.1.1 Proteins Overview

Proteins play a pivotal role in the functioning of living organisms, as they are involved in a wide range of biological processes and interactions with themselves, other proteins or other molecules besides proteins. These include but are not limited to growth and maintenance, regulation of biochemical reactions, acting as messengers, providing structural support, maintaining proper pH levels, facilitating fluid balance, supporting the immune system, facilitating transportation and storage of nutrients, and supplying energy to the body. Proteins are present in our daily lives. Moreover, certain groups of proteins have been actively employed in several industries, with enzymes, for example, being widely utilised in the food, textile, and detergent

industries [19].

Proteins are complex biomolecules that perform various functions within living organisms. The functions of proteins depend on multiple parameters, including interactions with the environment, structural properties, and the specific amino acid sequences that make up the protein [20].

The primary structure of a protein refers to the linear sequence of amino acids that make up the protein. The folding of the protein into its characteristic three-dimensional shape, known as its tertiary structure, is determined by the interactions between the amino acid residues within the protein and is further influenced by the environment. A protein's secondary structure refers to the backbone's local arrangements, such as the alpha helix or beta sheet, that give rise to the tertiary structure. The quaternary structure of a protein refers to the interactions between multiple chains and the existence of multiple chains in a protein, which can affect its function [20].

The discovery of new proteins holds significant potential not only for the advancement of human health but also for the improvement of our daily lives. One example is the enzyme cellulase, which has been utilised in various applications such as detergents, paper processing, and other industries since the early 1990s. Cellulases are considered one of the most essential enzymes in the global market due to their versatility and effectiveness in breaking down cellulose, a significant component of plant-based materials. Therefore, identifying new proteins and their characteristics can benefit human health and lead to the development of new products and industries [21].

Overall, the structure of a protein is crucial to determine its function. Understanding its structural properties and interactions can provide valuable insight into its functions and potential applications.

Amino Acid

Amino acids are a group of organic molecules that are mainly composed of a primary amino group ($-NH_2$), an acidic carboxyl group ($-COOH$), and an organic R group (or side chain). Proteins comprise amino acids; each amino acid contains a central carbon atom (C) attached to the amino and carboxyl groups, an attachment to a hydrogen atom and the R group in the central carbon. The R group, or side chain, makes the main distinction between each amino acid, according to the chemical

properties of the R group [22].

Even though hundreds of amino acids have been found in nature, in proteins, there are usually 20 standard amino acids [23], and sometimes there are rare occurrences of more. However, these 20 are considered the default [24]. It can be observed in the following table A.1. Besides the specific amino acid codes, sometimes a placeholder is used when a peptide or protein analysis cannot conclusively identify the residue.

Protein Structure

Protein structures are composed of building blocks known as amino acids. These amino acids are linked by peptide bonds formed by the condensation reaction between the carboxyl group of one amino acid and the amino group of another, resulting in the removal of a water molecule (H_2O). The sequence of amino acids and the number of peptide bonds in a protein is determined by the genetic code in the DNA.

Proteins also have multiple levels of structure, each with a unique function and properties.

- **Primary Structure:** This is the linear sequence of amino acids that comprises the protein. It is the simplest level of protein structure. It determines the sequence of amino acids, of which there are 20 naturally occurring plus an indefinite number of amino acids made by chemical synthesis, genetic engineering, or a combination of both [25]. The primary structure, sequence order, side-chains and interactions also serve as a blueprint for the formation of the more complex levels of protein structure, such as the secondary, tertiary and quaternary structures [20].

- **Secondary Structure:** refers to the local arrangements of the backbone, such as the alpha helix or beta sheet, that give rise to the tertiary structure. The secondary structure is formed by interactions between the atoms of the backbone and the hydrogen bonds between the peptide bonds [20]. The alpha helix is a spiral structure that forms when the peptide bonds form hydrogen bonds between the carbonyl oxygen atoms of one amino acid and the nitrogen atoms of another amino acid four or five residues away. This structure creates a helix shape. The beta-pleated sheet is a flat structure formed when the peptide bonds form hydrogen bonds between the nitrogen atoms of one amino acid and the oxygen atoms of another amino acid five or more residues away.

This structure creates a flat sheet shape. The alpha helix and the beta-pleated sheet are essential in stabilising the tertiary structure of a protein. However, other less common secondary structures, such as beta turns, beta hairpins and loops, also exist.

- **Tertiary Structure:** The tertiary structure of a protein refers to the three-dimensional shape formed by the interactions between the amino acids' side chains (R groups). These interactions include bonding, repulsion and attraction of charges, hydrogen bonds, dipole-dipole interactions, and hydrophobic interactions. A unique bond in this structure is the disulfide bond, which links the sulfur-containing side chains of cysteines. The stability of the tertiary structure is crucial, as it needs to remain intact under different environmental conditions such as pH, temperature, and the presence of other molecules. The energy consumption from the interactions between the amino acid residues helps maintain the global energy minimum and the stability of the structure. [26]
- **Quaternary Structure:** Refers to organising multiple polypeptide chains into a single functional protein. This structure is relatively uncommon compared to single-unit proteins, which consist of a single polypeptide chain [20]—the presence of multiple polypeptide chains in protein results in the formation of a quaternary structure. Haemoglobin is a notable example of a protein with a quaternary structure, as it is composed of four polypeptide chains and plays a crucial role in transporting oxygen throughout the blood. The interaction between these multiple polypeptide chains allows for more complex functions and increased stability of the protein [27].

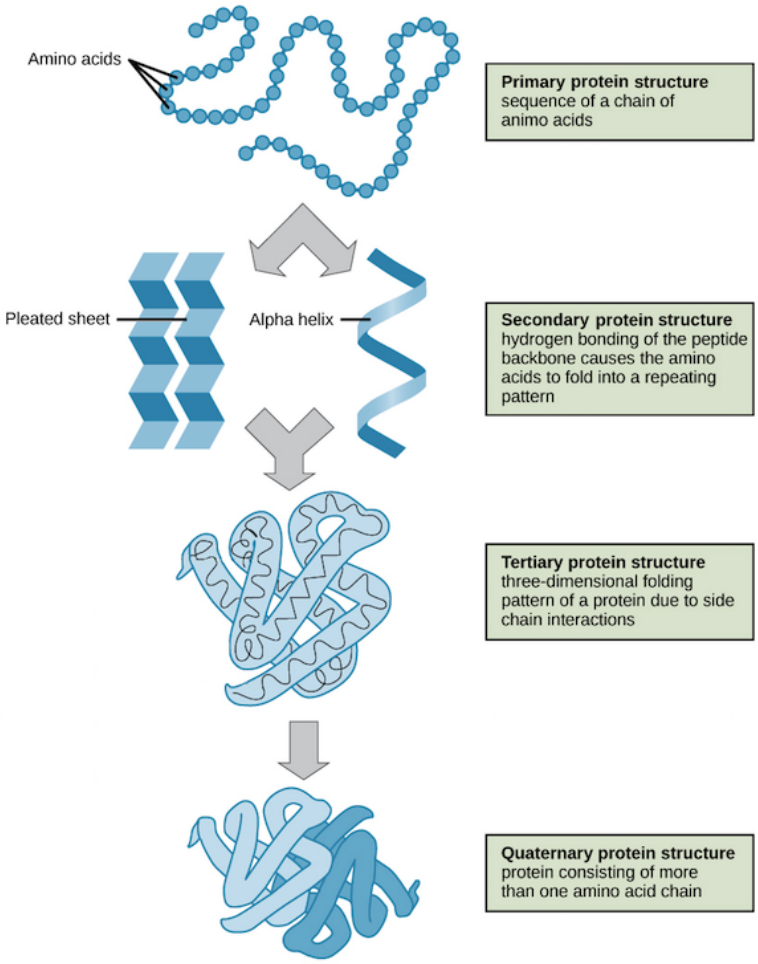


Figure 2.1: Protein Structure Levels: Primary, Secondary, Tertiary, Quaternary [28]

Proteins Design

Protein design uses computational methods and techniques to create new or modify existing proteins with specific properties or functions. It is done by manipulating the protein's amino acid sequence, tertiary structure, and quaternary structure and engineering new interactions between different parts of the protein [29]. Protein design is a powerful tool in biotechnology and medicine, as it can be used to create new enzymes, biosensors, and therapeutics [30–32].

One of the standard techniques used in protein design is rational design, where the protein's structure and function are understood and then manipulated to create a desired outcome. This method is mainly used when the desired protein already exists in nature, but with slight modifications, it can be improved [33].

Another technique is directed evolution, miming natural selection in the laboratory to create new proteins with desired properties. This method involves creating a library of random mutations in the protein of interest and selecting the variants with the desired properties through various screening methods [34].

Protein design is a highly interdisciplinary field involving researchers from various backgrounds, such as biochemistry, bioinformatics, and computer science. Advancements in protein design have led to the development of new therapeutics [30], autoimmune diseases, and genetic disorders. Protein design also has potential for applications in biotechnology, such as creating new enzymes for industrial processes [31], and in environmental science.

2.2 Sequence Alignment

Protein sequence alignment is comparing two or more amino acid sequences to identify regions of similarity or difference. It is an essential step in many bioinformatics analyses [35].

There are several algorithms and software tools available for performing protein sequence alignments, including global alignment methods such as Needleman-Wunsh [36] and local alignment methods such as Smith-Waterman [37]. These methods differ in how they score the alignments and the types of alignments they are best suited for.

Global alignment methods aim to align the entire length of two sequences, while

local alignment methods focus on finding the best matching region within the sequences. These methods can use different scoring matrices and gap penalties to evaluate the similarity between the sequences [38].

2.2.1 Smith-Waterman

The Smith-Waterman (SW) algorithm is a widely used method for local sequence alignment of proteins. The algorithm compares two protein sequences and identifies regions of similarity between them using a dynamic programming approach [37]. The method constructs a matrix H, indexed by (i,j), where i and j correspond to the length of the two aligned protein sequences. The matrix H is filled systematically based on a scoring function.

The scoring function used in the SW algorithm incorporates several parameters, such as the penalty for opening and extending gaps, the score for matches and mismatches, and a substitution matrix. The penalty for opening and extending gaps is used to penalise the insertion or deletion of amino acids in the alignment, and this penalty is typically subtracted from the overall alignment score. The score for matches and mismatches is based on a substitution matrix, which assigns a numerical value to the similarity between different amino acids. This score determines the similarity between the amino acids at a particular position in the aligned sequences.

The matrix H(i,j) is filled by considering the maximum possible score of a local alignment ending at a particular position (i,j) in the two sequences being aligned, taking into account the scores of the previous positions in the matrix, the gap penalties and the match/mismatch scores. The optimal local alignment corresponds to the highest score in the matrix H, representing the similarity between the aligned sequences.

Considering:

$A-$one sequence used in the method with length n,

$a_i-$the residue of sequence A in position i

$B-$the other sequence used in the method with length m

$b_i-$the residue of sequence B in position i

$s(a_i, b_j)-$The similarity score of a_i and b_i

$W_k-$the penalty of a gap that has length k

The algorithm will work as follows:

1. Determine the substitution matrix and the gap penalty.
2. Construct a scoring matrix H with the shape (n+1) by (m+1).
3. Fill H. $H_{k0} = H_{0l} = 0 \; for \; 0 \leq k \leq n \; \& \; 0 \leq l \leq m$

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k \leq 1}\{H_{i-k,j} - W_k\}, \\ \max_{l \leq 1}\{H_{i,j-l} - W_l\}, \\ 0 \end{cases}$$

4. Traceback. Starting at the highest score in the matrix H and ending at a cell where the score is 0, there is no similarity.

2.2.2 Needleman-Wunsch

The Needleman-Wunsch algorithm is a global sequence alignment algorithm used to align two sequences by maximising the overall similarity between them [36]. It is a dynamic programming algorithm that constructs a matrix, similar to the Smith-Waterman algorithm, where the rows and columns correspond to the positions in the two sequences being aligned.

The Needleman-Wunsch algorithm uses a scoring function that assigns a value for a match, a mismatch, and the insertion or deletion (gap) of an amino acid. The score for a match is typically positive, the score for a mismatch is typically negative, and the score for a gap is also typically negative. The algorithm begins by initialising the first row and column of the matrix with the cumulative gap penalties. Then, it systematically fills the rest of the matrix by considering the maximum possible alignment score for each position (i,j) in the two sequences being aligned, taking into account the scores of the previous positions in the matrix, the gap penalties, and the match/mismatch scores [36].

The optimal global alignment corresponds to the last element of the matrix; this score represents the degree of similarity between the two sequences being aligned. The final alignment can be reconstructed by tracing the matrix and identifying the path leading to the optimal alignment score [36].

2.2.3 BLAST

Basic Local Alignment Search Tool (BLAST) is a software tool used to search for similarities between a query sequence and a database of sequences. The most common use of BLAST is to search for similar sequences in public databases, such as the National Center for Biotechnology Information (NCBI) database, to identify homologs of a given query sequence and calculate its statistical significance. BLAST can be used to align sequences of various types, including DNA, RNA and proteins, and it can be run in different modes, depending on the type of sequences and the research question [39].

There are several versions of BLAST, each optimised for different sequences and research questions. The most commonly used versions are [39]:

- BLASTN: for aligning nucleotide sequences against a nucleotide database.
- BLASTP: for aligning a protein sequence against a protein database.
- BLASTX: for aligning a nucleotide sequence against a protein database, using a six-frame translation of the nucleotides.
- TBLASTN: for aligning a protein sequence against a nucleotide database, using a six-frame translation of the database sequences.
- TBLASTX: for aligning a nucleotide sequence against a nucleotide database, using a six-frame translation of both the query and the database sequences.

The BLAST algorithm starts by first looking for "seed" matches between the query sequence and the database sequences, which are short stretches of the query sequence used to initiate the alignment process. The algorithm calculates a score for each seed match based on the alignment of individual residues and the gap penalties. If the score is above a certain threshold, the match is considered a "hit". These hits are then extended in both directions using the dynamic programming approach to find the optimal local alignment. The algorithm then reports the best alignment(s) with the highest score and their location in the database [40].

2.2.3.1 Non-Redundant Database Integration

The Non-redundant (NR) database [41] constitutes a cornerstone of sequence similarity analysis within the BLAST tool provided by the National Center for Biotechnology Information. Its purpose lies in mitigating the challenge of sequence re-

dundancy, which arises from including multiple highly similar sequences originating from various sources.

To counteract redundancy, the NR database employs a systematic curation process that groups highly similar sequences into clusters, thereby retaining only one representative sequence per cluster. This strategy drastically reduces the size of the database while preserving the diversity of biological information. The resulting NR database is a curated collection of non-redundant sequences, offering a more accurate representation of the biological sequence space.

Integrating the NR database within BLAST searches revolutionises sequence similarity analysis. Its role in curbing redundancy and enhancing precision contributes to more meaningful results, ultimately empowering researchers to uncover intricate biological relationships more accurately and efficiently.

3

State of the Art

3.1 Computational Protein Engineering

In recent years, there has been a significant increase of computational methodologies capable of challenging the task that is Protein Design, from predicting the structure [42, 43] to the function of proteins [44]. These models learn the distribution of natural proteins and can generate new, unseen proteins with the desired properties.

In 2000, the Rosetta software was released. This software is suited for computational protein structure prediction, including protein structure prediction, protein-protein docking, and ligand-binding prediction [45]. Rosetta primarily focuses on predicting the structure of proteins based on their amino acid sequence. However, it can also be used for protein design. It uses computational methods to search for sequences that fold into a specific structure or conformation, such as energy-based scoring, structural constraints, and evolutionary information [46]. The disadvantages of Rosetta's are the complexity, due to the number of options and parameters [45], the dependence on experimental data, Rosetta's predictions are often improved when using experimental data, such as X-ray crystal structures [46], making Rosetta time-consuming and less useful for predictions of completely novel or unknown proteins.

In 2001, Baker et al. [17] proposed a model that classifies enzymes based on their individual properties. A part of the model is pre-trained by replicating unlabelled proteins using a stacked LSTM. It successfully learned how the sequence follows after it is fine-tuned to classify. It also achieved good results in classifying the type of enzyme it is.

Recurrent Neural Networks can achieve good results in classifying and generating protein sequences. Liu et al. 2017 [47] used an LSTM model that learns the sequence

of the protein and can classify its function, later on using BLAST to validate the results, showing the possibility that an LSTM is capable of grasping the sequence and understanding how it functions without the usage of more advanced protein structures. In 2017, Cao et al. 2017 [48] applied fine-tuning and transfer learning so the drug generator could generate new drugs to target specific targets and proteins. Despite focusing on drug development, this proposal still considered how the protein functions and the behaviour of proteins.

The work of Karimi et al., 2019 [42] proposes the use of a semi-supervised framework GAN, named Conditional Wasserstein Generative Adversarial Network (CWGAN). Here, it has a generator, a discriminator network, and an oracle that feeds the generator on how the protein would fold. The CWGAN uses the Wasserstein distance metric instead of the Jensen-Shannon [49]. It will generate protein sequences with a secondary structure similar to an alpha helix. Despite achieving good results, the model is limited to one type of protein folding and a limited dataset of just alpha-helix protein structures. Sabban et al. 2020 [43] also propose a stacked LSTM capable of generating de-novo helical proteins for the generator. However, it suffers from the limitation of just a type of protein structure. Also, the lack of comparisons to the other state-of-the-art works does not corroborate the results.

As evidenced in the state of the art, most works are around the 3D element of the proteins and not the more novel primary structure. In another area of bioinformatics, such as drug development, it is possible to work with a 1D sequence. Grechishnikova proposes a Transformer, where given a protein amino acid sequence, it outputs a drug string sequence, known as Simplified Molecular Input Line Entry System (SMILES) [50]. The encoder of the transformer will transform the amino acid sequence into a continuous representation that is then passed to the decoder. The decoder may consult any time the output is given by the encoder due to the attention mechanism in the transformer.

Advancements in text generation have paved the way for knowledge transfer to the realm of biological string sequence despite the distinct domains involved. Notably, Goodfellow's introduction of GAN has found application in numerous papers, facilitating the generation and validation of proteins [51]. Another milestone was the introduction of Seq2Seq models [52], which introduced the encoder-decoder architecture. Subsequently, Transformers [53] adapted this structure, incorporating an attention mechanism to weigh the significance of various input elements. These developments culminated in creating the latest iteration of text generation models,

known as Generative Pre-trained Transformer (GPT) [54]. GPT-3, an unsupervised pre-training language model, can respond accurately to prompts, representing a significant leap in text generation.

This page is intentionally left blank.

4

Methods

4.1 Artificial Neural Networks

Machine Learning (ML) is a subset of artificial intelligence that focuses on developing algorithms and statistical models. These models can learn and make predictions or decisions without explicit instructions. There are several types of ML, such as supervised learning, unsupervised learning, and semi-supervised learning [55].

Supervised learning is when the model is trained on labelled data, meaning that the input and the output are provided during training. The model learns how to map the inputs to results, and later, it can be applied to new and unseen inputs.

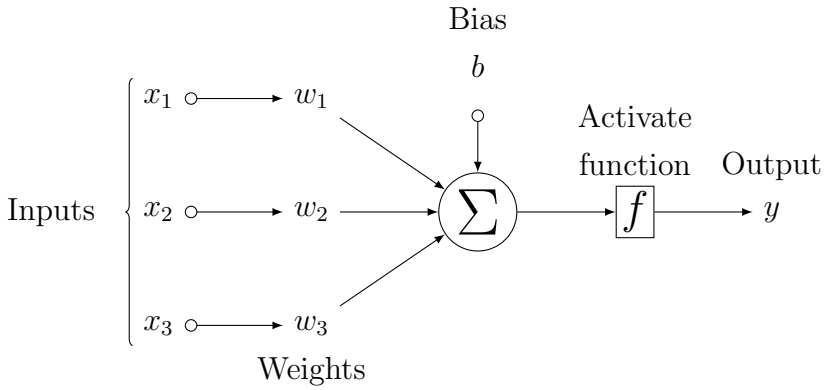
The opposite of supervised learning is unsupervised learning; the model is trained on unlabeled data and tries to discover patterns. The main goal of unsupervised learning is to understand the underlying structure of the data and find hidden patterns or unknown features.

Semi-supervised learning is a combination of supervised and unsupervised learning. The algorithm is provided with a dataset that contains labelled and unlabeled data. This way, the labelled examples are used to train the model and the unlabeled examples are used to help the model learn more about the data structure.

Artificial Neural Network (ANN) is a ML model inspired by how the human brain works. For example, the brains have neurons, and an ANN consists of connected processing units known as nodes, perceptrons or neurons [56,57]. Each neuron takes inputs, performs calculations, and produces an output. The inputs to a neuron are typically the outputs from other neurons or its output. Some neuron's calculation is the dot product, summation and activation function.

A typical structure of a neuron in an ANN includes the following components:

- Inputs, x_i: These are the values passed into the neuron, typically the output values of other neurons in the network.
- Weights, w_i: These are the values used to multiply the input values, which are used to adjust the importance of the input values.
- Bias: This value is added to the sum of the inputs and weights, which helps to adjust the neuron output.
- Summation function, $\sum$: This mathematical function adds up all the inputs multiplied by the weights and adds the bias value.
- Activation function, f: This mathematical function is applied to the summation function's output. It is used to introduce non-linearity in the output, which helps the network to learn complex relationships in the data.



The Multilayer Perceptron (MLP) consists of an input layer, one or more hidden layers, and an output layer. The input layer receives and passes the input data to the hidden layers. Each hidden layer processes the data using a set of weights and biases and applies an activation function to introduce non-linearity. The output layer produces the final output of the network.

Deep Neural Network (DNN) are a type of ANN with a deep architecture, meaning they have multiple layers, typically many hidden layers. These layers comprise interconnected nodes or neurons, which process and transmit information. The structure of a DNN can be composed of an input layer, one or more hidden layers, and an output layer. The input layer receives the input data, the hidden layers process the data, and the output layer provides the predictions or output. Thanks to their deep architecture and non-linear activation functions, DNNs can learn complex patterns and relationships in the data.

The main problem in DNN is that it needs a large dataset and algorithms to adjust the connections between the neurons to learn appropriately. This is because DNNs typically have many parameters or connections between nodes, which require a lot of data and computational resources to optimise. Additionally, DNNs are prone to overfitting, where the model becomes too specialised to the training data and cannot generalise well to new data. To solve these challenges, regularisation and early stopping can be used to prevent overfitting and improve the model's generalisation. Transfer learning techniques can also leverage pre-trained models and fine-tune them on specific datasets, reducing the data and computational resources required to train a DNN [58].

The effectiveness of deep neural network (DNN) models hinges significantly on the choice of the gradient descent optimizer and the loss function.

There is also a need for updating in a neural network, which typically involves using a loss function, also known as a cost function, to measure the error or difference between the predicted and actual output. The goal is to minimise this error. This process is known as backpropagation, where the error is propagated back through the network, and the neurons' weights and biases are adjusted to reduce the error. The adjustments to the weights and biases are determined by optimisation algorithms such as gradient descent, which calculates the gradient of the loss function concerning the network's parameters and updates them in the opposite direction to minimise the loss. This process is repeated multiple times during the training until the error reaches an acceptable level or a stopping criterion is met [59].

Gradient descent serves as the cornerstone for training deep neural networks. It's an iterative optimization algorithm to discover a given function's minimum (or maximum). In artificial neural networks, gradient descent is crucial in adjusting the model's parameters to minimize a specified loss function. A key element in this process is the learning rate, which governs the step size required to reach a local minimum. In essence, the loss function quantifies prediction errors, facilitating the calculation of gradients that are subsequently backpropagated through the network—from the output layer to the input layer—to update the corresponding weights. Two challenges often encountered during this process are the vanishing and exploding gradient issues. The former occurs when the error becomes too small, leading to minimal updates when it reaches the input layer. Conversely, the latter occurs when the gradient grows exponentially as it is propagated backwards.

4.1.1 Recurrent Neural Network

A RNN is a neural network that processes sequential data. It comprises a series of interconnected neurons capable of retaining information from prior inputs, allowing it to process data with temporal dependencies.

RNNs are particularly useful for tasks such as natural language processing [60–62], speech recognition, and time series prediction. They comprise a series of layers, typically including an input layer, one or more hidden layers, and an output layer.

The main characteristic of RNNs is the presence of recurrent units, which are similar to traditional feedforward neurons, but with the addition of where the output of a neuron is fed back as input to the same neuron, allowing information to be passed from one-time step to the next [63].

Their capacity to transmit information to neighbouring neurons within the same layer facilitates the temporal persistence of information, having a *memory*. The neuron's output depends on the input and its predecessors with this procedure.

There are several types of RNNs, including the traditional Vanilla RNN, LSTM networks [64], and Gated recurrent units (GRU) networks. LSTM and GRU networks are variations of the Vanilla RNN designed to address the problem of vanishing gradients, which can occur when training RNNs with long-term dependencies [65].

RNNs can be trained using various optimisation algorithms, such as Backpropagation Through Time (BPTT) and the more recent Truncated Backpropagation Through Time (TBPTT) algorithm [66]. The critical difference between BPTT and the standard backpropagation algorithm is that in BPTT, the error is propagated through all the time steps in the sequence, not just the final time step. It allows the network to learn dependencies between the input and output at different time steps and better suits sequential data tasks. However, BPTT has the drawback of requiring a large amount of memory to store the gradients, which can be an issue when training on long sequences. To alleviate this, TBPTT is used, where only a limited number of time steps are used to backpropagate the error.

4.1.2 Long Short-Term Memory

An LSTM cell is a RNN capable of learning long-term dependencies in sequential data. The LSTM cell, Figure 4.1, is composed of several gates that control the

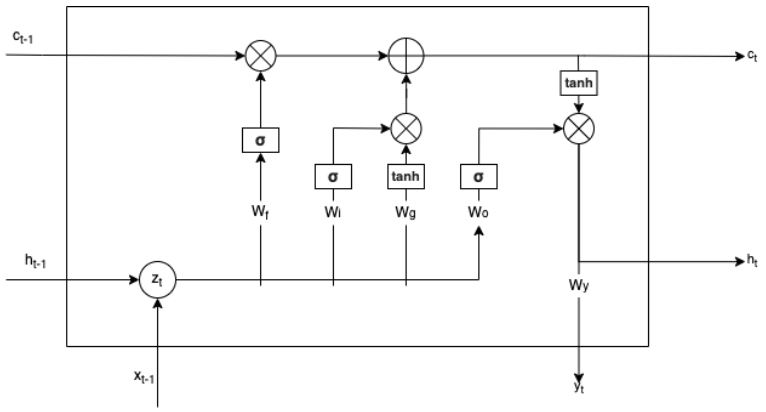


Figure 4.1: LSTM Cell

flow of information through the cell, which allows the model to remember or forget information [64] selectively. The basic structure of an LSTM cell includes the following:

Forget Gate f: This gate defines how much one must forget.

Input Gate i: This gate decides how the present input needs to flow.

Gate gate g: This gate treats the hidden state in the LSTM.

Output Gate o: This gate is responsible for the final output and hidden state.

Each gate is composed of a sigmoid layer and a point-wise multiplication operation. The sigmoid layer outputs a value between 0 and 1, which acts as a "gate" that controls the flow of information. The point-wise multiplication operation applies to the input, forget, and output states.

The LSTM cell updates its state at each time step using the current input, the previous state, and the learned parameters. The following formulas are used for this update:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$g_t = \tanh(W_g \cdot [h_{t-1}, x_t] + b_g)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$

Where i_t, f_t, o_t and g_t are the input, forget, output and cell input gate activations, respectively, at time step t, h_{t-1} and x_t are the previous hidden state and current input, W denotes the weight matrices, b denotes the bias vectors, σ is the sigmoid function, and $\odot$ denotes element-wise multiplication.

4.1.3 Sequence to Sequence

Mikolov introduced Seq2Seq in his Ph.D. Thesis [67]. It receives an input sequence and generates another sequence. Due to the vanishing gradients problem, most Seq2Seq based architectures rely on using LSTM cells since they do not suffer from the vanishing gradient as much.

In Figure 4.2, the Seq2Seq model is an architecture that consists of two RNNs, an encoder and a decoder.

Encoder: It transforms the sequence to a fixed-length hidden vector for each element in the sequence. This hidden vector represents the context for that element.

The encoder RNN processes the input sequence one step at a time, updating its hidden state h_t at each time step t according to the following formulas:

$$h_t = \text{RNN}(x_t, h_{t-1})$$

$$c_t = \text{RNN}(x_t, c_{t-1})$$

where x_t is the input at time step t, h_t is the hidden state at time step t, and c_t is the memory cell state at time step t.

Decoder: It takes in the hidden vectors from the encoder, its hidden states and the current element to produce the next hidden state and the next element.

The decoder RNN then takes in the context vector c_T and generates the output sequence one step at a time, updating its hidden state s_t and output y_t at each time step t according to the following formulas:

$$s_t = \text{RNN}(y_{t-1}, s_{t-1}, c_T)$$

$$y_t = \text{softmax}(W_y \cdot s_t)$$

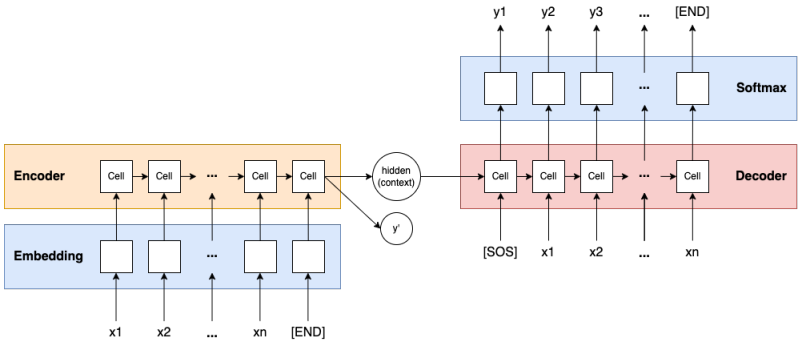


Figure 4.2: Seq2Seq Architecture

where y_{t-1} is the output at time step t-1, s_t is the hidden state at time step t, c_T is the final memory cell state from the encoder, and W_y is a weight matrix, sometimes it can be a different activation function besides softmax.

The traditional Seq2Seq model has some disadvantages, such as the inability to handle long sequences efficiently and the tendency to generate repetitive or generic responses. To address these issues, extensions to the Seq2Seq model, such as the attention-based Seq2Seq and the Transformer architecture, have been proposed [68].

4.1.4 Transformer

An architecture used for Natural Language Process (NLP) is the Transformers, Figure 4.3. This architecture uses a mechanism known as **Attention**, capturing the features between two input sequences and effectively modelling the relationships and dependencies within the sequence [53].

The structure of this model is an encoder-decoder structure. The encoder is composed of a stack of N identical layers. Each layer possesses sub-layers. The first is a multi-head self-attention mechanism, and the second is a fully connected neural network. The output of each layer is summed and then normalised. Here the encoder maps an input sequence of symbols $(x_1, ..., x_n)$ to a sequence of continuous representation $\mathbf{z} = (z_1,..., z_n)$.

The decoder comprises N layers and possesses the same layers as the encoder and an additional multi-head attention focusing on the encoder's output. The decoder also receives the pretending target as output. However, it shifts to the right, but the multi-head attention responsible for the input, which is the target shifted to the right, the self-attention is modified so that the sub-layer prevents from attending to

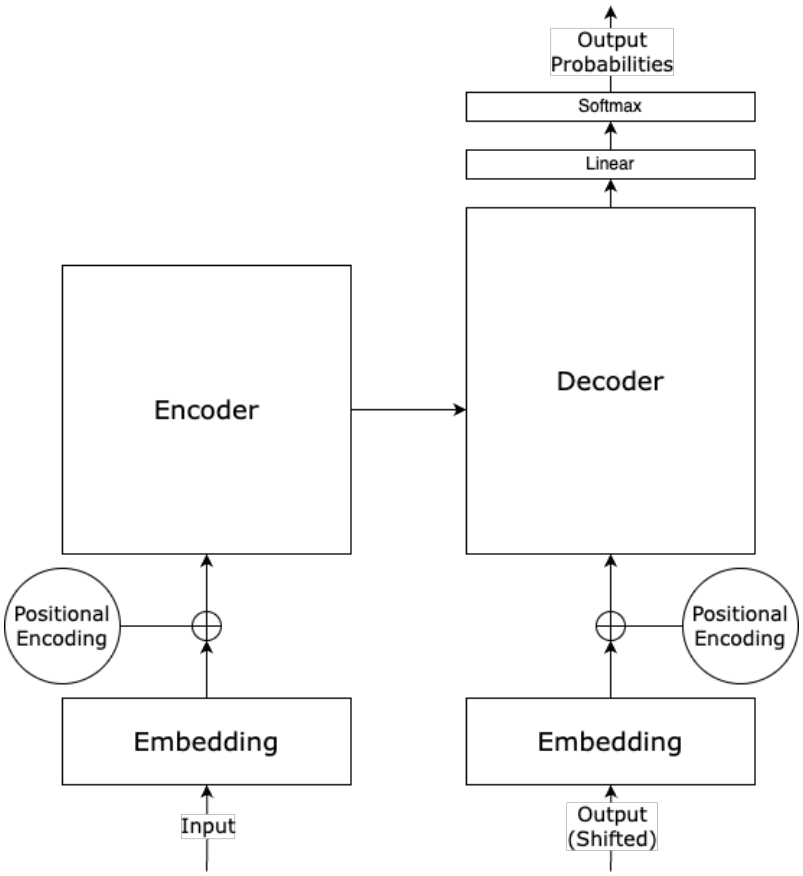


Figure 4.3: Transformer Architecture Overview

subsequent positions. By utilizing **z**, the decoder generates an output of sequence ($y1$,...,y_m) of symbols one element at a time. The model is also regressive, consuming the previously generated symbols as additional input when generating the next symbol.

The Transformer architecture employs a consistent structure in which stacked self-attention and point-wise, fully connected layers are utilized for both the encoder and decoder components.

The attention in the Transformer consists of a **Scaled Dot-Product Attention**. The input consists of queries, keys and values. Here, the output consists of a dot product of the query with all keys divided by the square root of the dimension of keys, $\sqrt{d_k}$, and apply a softmax to obtain the weights on the values, Equation 4.1.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{4.1}$$

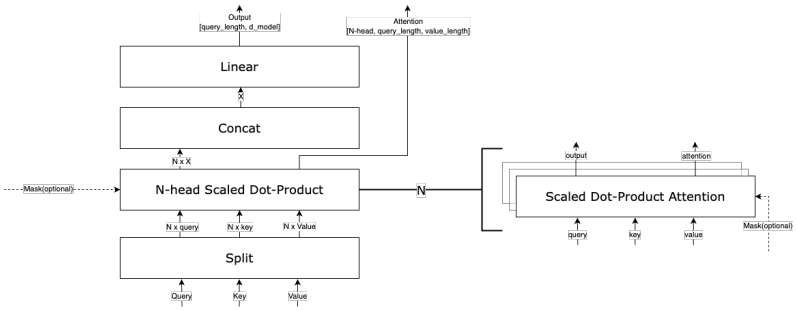


Figure 4.4: Multihead and Scale-Dot Product Attention

However, the model does not just perform a single attention function. The model projects the different inputs a number of times, h, with different linear projections. An attention function is performed in parallel on each projection version of queries, keys and values. All values are concatenated and projected, resulting in the final values, as depicted in the equation 4.2.

$$\begin{aligned} &MultiHead(Q,K,V) = Concat(head_1, ..., head_h)W^O \\ &,where head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{4.2}$$

Besides the attention sub-layers, as mentioned before, both the decoder and the encoder contain a fully connected feedforward network. It consists of two linear transformations with a ReLU activation in between, equation 4.3.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{4.3}$$

Since there is no recurrence nor convolution, to use the order of the sequence, there is a need to inject the relative and absolute position of the tokens in the sequence. To achieve this, a "positional encoding" is added to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the exact dimensions as the embeddings so that they can be summed. There are different types of positional encodings. The one used in the original work is the sine and cosine functions of different frequencies.

$$PE_{pos,2i} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{pos,2i+1} = cos(pos/10000^{2i/d_{model}})$$

Interpretability is a vital aspect of deep learning techniques, and Transformers offer a valuable tool for achieving it: attention mechanisms. These mechanisms allow us to visualize the model's decision-making process, and one popular method for doing so is through attention maps. Attention maps illustrate how different input tokens influence each other's importance when generating output. They come in various forms, including heatmap-based representations and Token-to-Token Attention, which reveals how each token in the input sequence contributes to other tokens in the same sequence.

Transformers often use multiple attention heads and have multiple layers, allowing the model to capture different types of relationships between tokens. It can visualize the attention patterns learned by each attention head or how the model's focus changes as it processes information through deeper layers.

4.1.5 Reinforcement Learning

RL is a distinct approach to machine learning that addresses the challenge of training intelligent agents to make sequential decisions in an environment to maximise a reward signal. It differs from supervised and unsupervised learning, which are two other widely used learning paradigms, in terms of the learning process and the types of problems they tackle [69–71].

RL involves studying a Markov decision process, where an agent interacts with the environment ϵ over discrete time steps t. At each time step, the agent receives a state s_t from a state space S and selects an action a_t from an action space, guided by a policy $\pi(a_t|s_t)$. In this framework, maximising the expected cumulative reward becomes a central objective.

To address this objective, the concept of the Q-function emerges as a fundamental component of the Markov decision process. The Q-function represents an agent's expected cumulative reward by taking a particular action a in a given state s and following a policy π. In mathematical terms, the Q-function is defined as:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi\right] \tag{4.4}$$

where:

- $Q^{\pi}(s,a)$ represents the Q-value or expected cumulative reward for taking ac-

tion a in state s and following policy π thereafter.

- r_t is the reward obtained at time step t.
- γ is the discount factor determining the importance of future rewards compared to immediate rewards.

The Q-function (4.4) plays a vital role in making optimal decisions by guiding the agent to choose actions that maximise the expected cumulative reward. Different algorithms, such as Q-learning and Deep Q-Network (DQN), aim to learn and estimate the optimal Q-function to enable optimal decision-making in Reinforcement Learning tasks [70, 72].

The provided Q-function definition assumes an infinite horizon setting, where the summation extends to infinity. However, in practice, there are often finite-horizon or episodic settings where the summation is limited to a fixed number of time steps or episodes.

As mentioned in a previous statement, the objective is to maximize the expected reward from the actions taken. So, it was determined that the focus should be on the following objectives:

$$\max \mathbb{E}_{\hat{y}_1,\ldots,\hat{y}_T \sim \pi_\theta(\hat{y}_1,\ldots,\hat{y}_T)} \left[r\left(\hat{y}_1, \ldots, \hat{y}_T\right) \right] \tag{4.5}$$

$$\max_y A_\pi\left(s_t, y_t\right) \tag{4.6}$$

$$\max_y A_\pi\left(s_t, y_t\right) \rightarrow \mathrm{Max}_y\, Q_\pi\left(s_t, y_t\right) \tag{4.7}$$

There are multiple approaches to addressing this problem. One method involves utilizing Equation 4.5, while another leverages the expected discounted reward $\mathbb{E}[R_t = \sum^T \theta = t\phi^{\theta - t} r\theta]$. Alternatively, we can aim to maximize the advantage function (Equation 4.6) or solve it by optimizing the Q-function using Equation 4.7.

The most straightforward algorithm designed to address Equation 4.5 is known as the *Policy Gradient (PG)*. In contrast, the actor-critic method expands upon Equation 4.6 by incorporating the Q-function. Enhanced strategies can be developed by optimizing Q-functions, as shown in Equation 4.7, which can further improve policy gradient and actor-critic methods.

4.1.5.1 Policy Gradient

PG is based on the principle *We observe and act*. It means the policy varies according to the application used, so it observes and then chooses how to behave. For in-text applications that receive text as input, X tries to generate the output y, being the policy a language model, $p(y|X)$. For a sequence output, when it reaches the end of the sequence, it compares the current policy ($\hat{y}_t$) against the ground-truth action sequence (y_t) and calculates a reward based on the metric used. The objective is to optimize the agent's parameters to maximize its reward. These algorithms do not estimate Q-values; instead, they prioritise the policy's parameters directly to maximize the expected cumulative reward. An example of such an algorithm that employs policy gradients is REINFORCE, algorithm 1, [72].

Algorithm 1 Policy Gradient: REINFORCE

Input: Input sequences (X), ground-truth output sequences(Y), and if possible an already pre-trained policy model(π_0)
Output: Trained model with REINFORCE
while not converged **do**
  Select a batch of size N from X and Y
  Observe the sequence reward and calculate the baseline r_b
  Calculate the loss
  Update the parameters of the network

The main challenge of REINFORCE is the high variance since only one sample is used at a given time step. Consequently, one solution is to sample N sequences of actions and update the gradient by averaging all those values. The result is a loss similar to the Equation 4.8.

$$\mathcal{L}_0 = \frac{1}{N}\sum_{i=1}^{N}\sum_{t}\log \pi_\theta(\hat{y}_i|\hat{y}_{i,t-1,s_{i,t},c_{i,t-1}}) \times (r(\hat{y}_{i,1},...,\hat{y}_{i,T}) - r_b) \tag{4.8}$$

A challenge stems from the fact that rewards are observed only after the entire sequence of actions is completed. This delay can be problematic when the model needs to make critical decisions at specific points within the sequence, and the delayed reward may not accurately reflect the quality of those decisions. Consequently, the REINFORCE algorithm requires waiting until the end to assess performance, resulting in slower convergence and potentially suboptimal results. One effective solution is to pre-train the model for several epochs before transitioning to the REINFORCE algorithm.

4.1.5.2 Actor-Critic

The actor-critic approach is a temporal difference version of policy gradients (PG). It involves two networks: the **Actor** and the **Critic**. The Actor is responsible for selecting actions, while the Critic assesses the quality of those actions and provides feedback.

To mitigate variance, PG incorporates a baseline reward. A model is employed to compute this baseline reward in the actor-critic framework. Here's how it works: The Actor generates samples and policy states at times t and $t+1$. The Critic estimates the value function $V_\pi(s;\Psi)$ and provides the Actor with estimations. The Actor then leverages these estimations to calculate reward approximations and update the loss, as depicted in Equation 4.9.

$$\mathcal{L}_0 = \frac{1}{N}\sum_{i=1}^{N}\sum_{t}\log \pi_\theta(\hat{y}_i|\hat{y}_{i,t-1,s_{i,t},c_{i,t-1}}) \times A_\Psi(s_{i,t},y_{i,t}) \tag{4.9}$$

4.1.5.3 Deep Q-Network

The 'Q' in Q-Learning represents a specific reinforcement learning technique called off-policy temporal difference learning. Unlike REINFORCE, which requires waiting until the end of an episode to calculate the final reward and discounted rewards, Q-Learning uses the Bellman equation to update the value function for a given State-Action pair. This enables Q-Learning to consider future rewards and efficiently update the value functions for all actions without waiting for episode completion [70].

The Bellman Equation consists of the agent updating the current perceived value with the estimated optimal future reward, which assumes that the agent will search through all existing actions and choose the state action pair with the highest Q-value.

$$Q(S_t, A_t) = (1-\alpha)Q(S_t,A_t) + \alpha * (R_t + \lambda * max_\alpha Q(S_{t+1}, a)) \tag{4.10}$$

The process in the DQN is first to initialise Main and Target neural networks and, secondly, choose an action using the Epsilon-Greedy Exploration Strategy. Finally, update the network weights using the Bellman Equation.

Compared to the other methods, this model is forced to use two neural networks.

Both networks must have the same architecture but different weights. In every N step, the weights from the leading network are copied to the target network.

The Epsilon-Greedy exploration strategy chooses a random action with an epsilon probability and exploits the best-known action with 1-epsilon. The best action obtained from the neural networks is the one with the most significant value predicted, which also represents the Q-value [72].

4.2 Protein Folding

Protein folding refers to the process by which a newly synthesised or denatured protein assumes its functional three-dimensional structure. Proteins are linear chains of amino acids, and their function is intricately tied to their three-dimensional shape.

During protein folding, the linear chain of amino acids folds and twists into a specific three-dimensional conformation. This process is guided by different interactions of the protein chain with the surroundings, such as hydrogen bonding, electrostatic interactions, hydrophobic interactions, and minimal energy consumption.

The folding of a protein is crucial for the proper function. If the proteins fail to fold correctly, they may become unstable, non-functional and even harmful to cells.

In informatics, protein folding refers to the computational modelling and simulation of protein structures and their dynamics. It usually requires the use of machine learning and the use of large protein databases. The three state-of-the-art algorithms in protein folding are OmegaFold, ESMFold and AlphaFold. AlphaFold uses a network-based model, ESMFold leverages a large-scale language model for protein prediction, and OmegaFold is a deep transformer-based protein language model.

4.2.1 AlphaFold

In AlphaFold, Figure 4.5 heavily relies on *Multiple Sequence Alignment (MSA)* for the prediction of 3D structure, considering that it is also used as inputs of the model, which maps the evolutionary relationship between corresponding residues of genetically related sequences, to achieve highly accurate predictions. It is widely accepted that MSA-dependent structural prediction tools gain 3D positional context clues from pairs of residues that co-evolve over time. However, it makes them reliant on naturally occurring protein sequences. It also identifies similar protein sequences that may have a structure similar to the input [6].

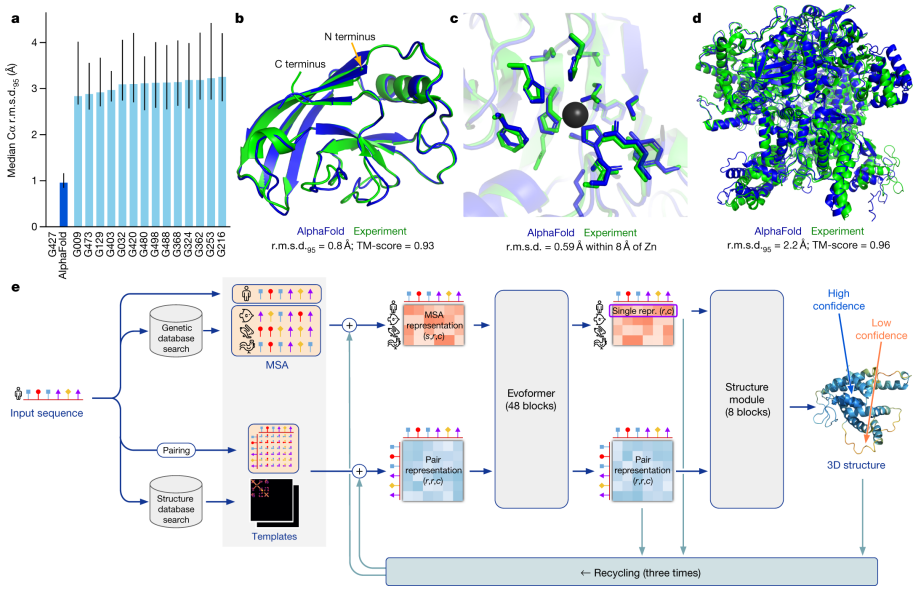


Figure 4.5: AlphaFold's Structure [6]

In the second step, AlphaFold uses an attention-based neural network architecture known as a Transformer-based to predict the 3D structure of the target protein. The network considers the predicted distances between pairs of amino acids in the protein sequence and the "similar templates". Later, it will refine the representation for the MSA and pair interactions and exchange information between both.

The interactions and MSA information output is taken to the structure module. It takes the "MSA representation" and the "pair representation" and leverages them to construct a three-dimensional model of the structure. The result is a list of cartesian coordinates representing each protein atom's position.

After generating a final structure, it will take all the information, MSA, pair and predicted structure, and repeat the process several times to the start of the Transformer-based model.

4.2.2 OmegaFold

To overcome the need for MSA in the prediction process of languaged-based models associated with protein sequences, OmegaFold has a language modelling component called OmegaPLM that uses Transformers and attention mechanisms to learn individual and pairwise residue representations for each protein sequence. Therefore, they achieve better accuracy on structure prediction in orphan proteins and antibody design as they do not require MSA as their input. However, it achieves lower

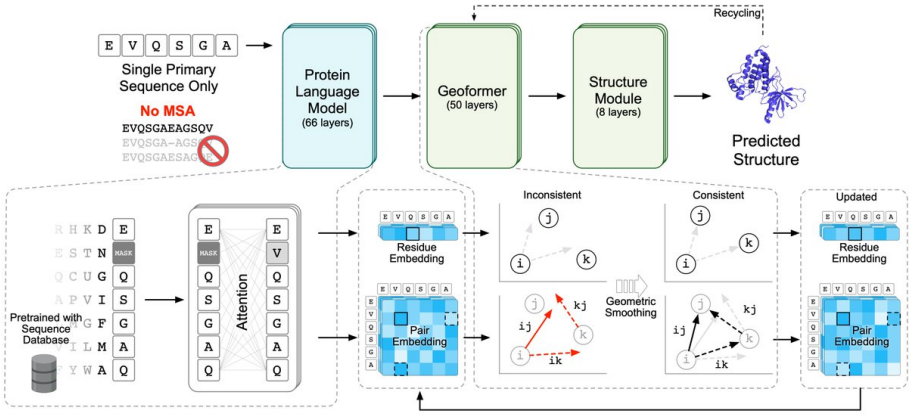


Figure 4.6: OmegaFold's Structure [73]

scores for proteins already possessing MSA. It makes the non-MSA-reliant better for protein engineering tasks.

The OmegaFold introduces the OmegaPLM, Figure 4.6, a deep Transformer-basedProtein Language Model (PLM), which is initially trained on an extensive collection of unaligned and unlabelled model protein sequences, learning single and pairwise residues embeddings. OmegaPLM captures structural and functional information encoded in the amino acid sequences. After this is fed to the GeoFormer, a new geometry-inspired transformer neural network further distils the structure and physical relationship between amino acids. Lastly, a structural module predicts the 3D structure of all the atoms [73]. The Geoformer's primary purpose is to make the embeddings from the OmegaPLM more geometrically consistent. It captures the geometry of a protein structure with representation, which is then projected to a structure module. Depending on the OmegaFold outputs, either a PDB file or recycle the process again, refining the structure until a certain number of iterations.

4.3 Metrics

4.3.1 BLAST "E-value"

The expectation value, E-value, returned by BLAST corresponds to the measure of the reliability of the similarity of the query to the sequence.

$$E = m * n/2^S \tag{4.11}$$

The equation 4.11 represents how the E-value is calculated. Where parameter m represents the size of the query, n represents the size of the database where BLAST was performed, and S is the measure of the similarity of the query to the sequence given. Furthermore, there are no database size limitations since this encompasses the entire UniProt database, which consists of at least 24,815,519 entries as of 2022, exclusively within UniProtKB. [74].

The main problem with the E-value is that it tends to be conservative when the query sequence is short, making it difficult to achieve a high S value. However, BLAST employs strategies to mitigate this problem. BLAST employs various statistical methods and algorithms to adjust the E-value calculation, considering factors like query length and database size, thus helping to provide more accurate significance estimates for shorter query sequences.

The ideal value for sequence similarity is closer to 0, indicating a high degree of similarity. Values approaching 0.01 are considered suitable matches for homology. When the similarity score exceeds 10, it suggests few significant matches, potentially indicating relations between the sequences.

In the range between 0.01 and 10, the match is considered less significant, but it may hint at a tentative remote homology relationship. In this case, some smaller sections of the sequences exhibit a degree of similarity.

4.3.2 plDDT

To correctly predict the 3D structure of a specific protein, it is critical to compute a well-calibrated and sequence-resolved confidance score. Additionally, when predicting complete chains, it is necessary to have high confidence. AlphaFold and OmegaFold produce a per-residue confidence metric called the plDDT [6].

The plDDT score is assigned to individual amino acid residues within the protein sequence. This assignment aids in the characterization of the prediction quality at a local level. The score falls into distinct value intervals, each indicative of a different degree of confidence in the prediction accuracy.

Residues with plDDT scores falling within the $[90, 100]$ range represent regions of high-confidence predictions. According to the model's robust estimation, these residues are anticipated to closely align with the true protein structure.

In the $[70, 90[$ range, plDDT scores mean moderate confidence in the prediction.

Even though these residues will likely adopt a structure resembling the true protein configuration, minor uncertainties may exist regarding their precise spatial arrangement.

Predictions characterized by plDDT scores ranging $[50, 70[$ belong to the moderate confidence category. The model's confidence in these regions is reduced, implying potential local deviations from the protein structure.

Residues scoring within the interval $[30, 50[$ indicate low-confidence predictions. The model's certainty in the accuracy of the predicted structure for these residues is diminished, possibly suggesting significant deviations from the true structure.

Lastly, plDDT scores within the $[0, 30[$ range correspond to regions with the lowest confidence predictions. These predictions are perceived as least reliable, meaning a higher likelihood of inaccuracies in the predicted protein structure.

Considering that the plDDT score is assigned individually to each residue, it is necessary to compute the mean score to obtain a global estimation of the predicted structures. Equation 4.12 computes perspective on the confidence of the structure's existence in its entirety.

$$GlobalplDDT(p) = \frac{\sum plDDT(p_i)}{length(p)} \tag{4.12}$$

5

Framework

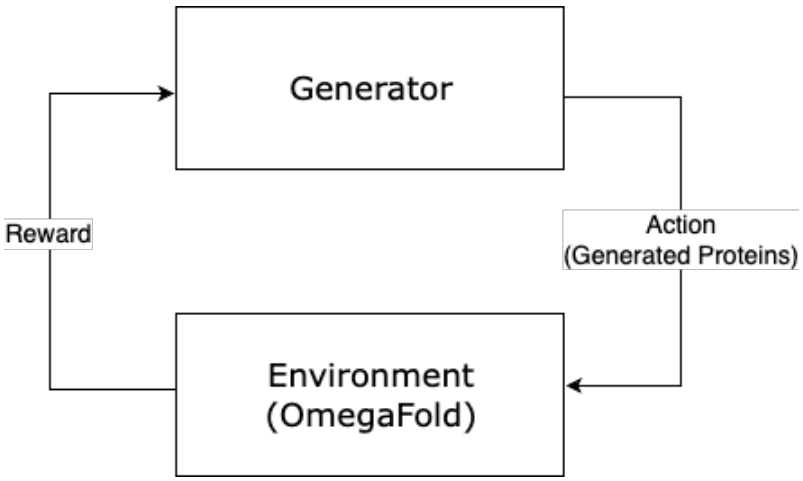


Figure 5.1: Framework Overview

The framework comprises two central components, illustrated in Figure 5.1. The 'Generator' is responsible for creating protein sequences, while the 'Environment' assesses the quality of these generated proteins.

Within the 'Generator,' the model follows the structure depicted in Figure 5.1 to produce valid, real-world protein sequences.

The training involves a two-phase generative model for protein sequences, combining a sequence-to-sequence approach with reinforcement learning. In the initial phase, the model learns the grammar and patterns of protein sequences from a provided dataset. It predicts the following amino acids at each step using a recurrent neural network, generating a sequence of probabilities for these amino acids. Training employs Maximum Likelihood Estimation (MLE) to minimize the cross-entropy loss between the predicted and actual sequences. Additionally, an extra 100 epochs of training are conducted for the Transformer model with proteins containing a noise level of 0.6.

In the second phase, the model undergoes refinement through RL. Another model, called the 'evaluator,' assesses a specific number of proteins generated by the 'generative model' and assigns quality scores based on predefined criteria, indicating confidence in the 3D protein structure. The 'generative model' is then trained to maximize the quality scores assigned by the evaluator using policy gradient methods.

Ultimately, the goal is to achieve a substantial percentage of proficient proteins that conform to real-world standards and exhibit diversity.

5.1 Generator

This work employed two Generator models: a stacked LSTM and a Transformer. The LSTM served as a baseline for result comparison, while the Transformer demonstrated the capabilities of deep neural networks in protein generation. Both models received tokenised protein sequences as input, subsequently padded to ensure consistent outputs. This approach allowed the models to learn the structural aspects of protein sequences. The loss function applied during training was cross-entropy.

The protein prediction process within these models follows a random search methodology. It involves taking the model's output, dividing it by a user-defined temperature, and passing it through a Softmax function, as depicted in Equation 5.1. This operation yields probabilities for each token in the sequence. Subsequently, a token is randomly selected based on these probabilities.

$$p(token) = Softmax(model(X)/temperature) \tag{5.1}$$

The introduction of the temperature parameter influences the distribution of these probabilities. Lower temperatures increase the likelihood of obtaining high-quality sequences, while higher temperatures introduce more randomness. For this work, a default temperature of 0.8 was chosen, striking a balance between introducing randomness and favouring high-quality sequence generation.

5.1.1 Parameterization for Different Models

Several parameters are available for adjustment in the context of training these models. However, the primary objective is to explore the feasibility of generating 1D protein sequences. Given the absence of a definitive set of parameters, we base

our choices on related works within the field of protein generation.

The effectiveness of deep neural network (DNN) models hinges significantly on the choice of the gradient descent optimizer and the loss function. Different variants of gradient descent exist, and for this context, we utilized the Adaptive Moment Estimation with Weight Decay (AdamW) optimizer in both the baseline LSTM and the Transformer models.

AdamW [75] represents an enhanced iteration of the Adaptive Moment Estimation (Adam) optimization algorithm, explicitly addressing the weight decay regularization challenge present in the original Adam optimizer. Weight decay is a valuable technique for preventing overfitting. However, Adam applies weight decay indiscriminately to both gradients and model parameters, leading to undesirable behaviour. AdamW resolves this issue by decoupling weight decay from the optimization process and directly applying it to the model parameters during weight updates. This approach enhances training stability, particularly in deep learning models where weight decay is critical in overfitting prevention. AdamW has gained popularity across various machine learning tasks and architectures thanks to its ability to separate weight decay from optimization cleanly.

The choice of a loss function depends on the type of machine learning problem being addressed, such as regression, classification, or other specialized tasks. This kind of task was used in the 1st phase, a Cross-Entropy Loss, and in the 2nd phase, where it was the reinforcement learning, Mean Squared Error (MSE).

The structure of the stacked LSTM is six layers of long short-term memory layers that map the output into the number of tokens with a linear function. The last layers are bidirectional and also have a dropout layer between one another. At the beginning of the layer, it has an embedding to better represent the proteins in a structure that the computer can easily understand. This model's prediction is more exhaustive because it predicts one token will repeat the whole process until it reaches the max length or an ending token. It had parameters similar to Table 5.1.

The structure of the Transformer follows the default values in the paper 'Attention is all you need', [53], with some minor tweaks. The embedding has a size of 512, and the number of heads is 16 due to the complexity of the proteins and their relationships between the amino acids. The dropout is still 0.2; the inputs are the protein sequence, and the target shifts to the right. Ultimately, it will be like Table 5.2.

Table 5.1: Parameters settings for the stacked LSTM model.

Parameters	Value
Epochs	200
Batch size	64
LSTM layers	5
Dropout Rate	0.2
Embedding Dimension	200
Bidirectional	True
Vocabulary Size	27
Learning Rate	0.0001
Hidden LSTM dimension	256

Table 5.2: Parameters settings for the transformer model.

Parameters	Value
Epochs without noise	200
Epochs with noise	100
Noise percentage	60%
Batch size	16
Transformer layers	6
Dropout Rate	0.2
Embedding Dimension	512
Number of Heads	16
Vocabulary Size	27
Learning Rate	0.0001
Hidden Transformer dimension	512
Step Function	Step=30;gamma=0.1

5.2 Environment

Accurately validating proteins generated through computational methods is critical to ensure their functional relevance and reliability. Predicting the tertiary structure of protein sequences using reliable models that excel in protein folding, such as AlphaFold and OmegaFold, presents a promising approach for this task. Adequately folded proteins assume their functional tertiary structure, enabling them to effectively carry out specific biological roles. In contrast, misfolding or improper folding can lead to protein dysfunction and contribute to diseases like Alzheimer's, Parkinson's, and prion-related disorders. In this subsection, we delve into the process of protein validation, the choice of the OmegaFold model over AlphaFold for specific scenarios, and the development of a reward function to incentivise the generation of valid proteins.

AlphaFold and OmegaFold's performance varies depending on the size and the sequence type. Suppose it is a single-sequence input. OmegaFold performs better with orphan proteins and antibodies, achieving higher statistical prediction accuracy than AlphaFold. In that case, orphan proteins do not have similar proteins to make a proper multiple sequence alignment. However, if the sequence is long, above the 2000 amino acids, the results are better for the AlphaFold due to numerous sequence alignments [73, 76].

In terms of runtime, the OmegaFold is faster than the AlphaFold for short protein sequences. However, the difference is increasingly slower the lengthier the sequence is Alphafold surpasses OmegaFold; this can be checked in Table 5.3, [77].

Table 5.3: Runtime AlphaFold vs OmegaFold based on the length of the protein sequence [77]

Sequence\Architecture	AlphaFold	OmegaFold
50	45	3.66
100	55	7.42
200	91	34.07
400	210	110
800	810	1425
1000	2800	Failed

So, considering protein folding prediction for the proteins generated, factors like protein family similarity, sequence characteristics, and length play a crucial role in selecting an appropriate model. OmegaFold, renowned for its efficiency in predicting protein folding for shorter sequences and orphan proteins, emerges as the better option in cases where the proteins may lack similarity to known protein families and have unique characteristics. Its ability to handle sequences with fewer than 500 amino acids while being faster and designed for individual sequences further solidifies its suitability for handling unknown proteins.

We employ the plDDT score to validate the generated proteins, a well-established confidence measure in protein folding models. The plant score quantifies the accuracy of the predicted protein folding by measuring the difference between predicted and experimental local distances in the protein structure. Based on the findings in AlphaFold's paper [6], proteins with plDDT scores above 90% are deemed scientifically correct. In contrast, those with scores above 70% have a generally accurate backbone, with neighbouring values also considered exemplary.

The validation of proteins through the plant score provides a basis for designing a reward function that promotes the generator to produce valid proteins while penalising misfolded sequences. We adopt the plant score as a threshold for a good protein, accepting values near 0.7 and offering substantially low rewards for other cases. The reward function (Equation 5.2) provides a reward of 1 for sequences with plant scores greater than or equal to 0.7, indicating the generation of valid proteins. Conversely, sequences with lower plant scores receive a reward of 0.1, discouraging the production of misfolded or improper protein structures.

$$Reward(p) = \begin{cases} 1 & if\ plDDT(p) >= 0.75 \\ plDDT(p) & if\ 0.75 > plDDT(p) >= 0.6 \\ 0.1 & if\ 0.6 > plDDT(p) >= 0.4 \\ 0 & else \end{cases} \tag{5.2}$$

The reward function forms the core of the RL approach employed to train the generator. The generator is pre-trained on a large dataset of protein sequences to comprehend the underlying grammar in protein structures. Subsequently, the generator generates 200 protein sequences and recycles them again, giving them to the model as input so it can improve the generation step; with this, each sequence is treated as an action of the model. The Environment then rewards these actions based on the validation results using the plant score. The RL approach allows us to handle the challenge of determining rewards only upon completing each protein sequence. It is similar to the algorithm in Reinforcement Learning in Policy Gradient, Chapter 4.1.5.1, of REINFORCE, 1. It was the preferred choice since we can only determine if it is good when it reaches the end of a protein sequence, and only then can we determine the reward. In the end, it follows the algorithm 2, where the starting token is just the start of the sequence token.

Algorithm 2 Training:2nd phase

Input: Trained generator model(G), if needed, starting tokens(X), OmegaFold(π_0)

Output: Trained model with Reinforce Learning

while not converged or episodes < 200 **do**

Generate from G 200 proteins sequences using X as starting tokens

Observe the sequence reward using π_0 and calculate the baseline r_b

Calculate the loss

Update the parameters of the network

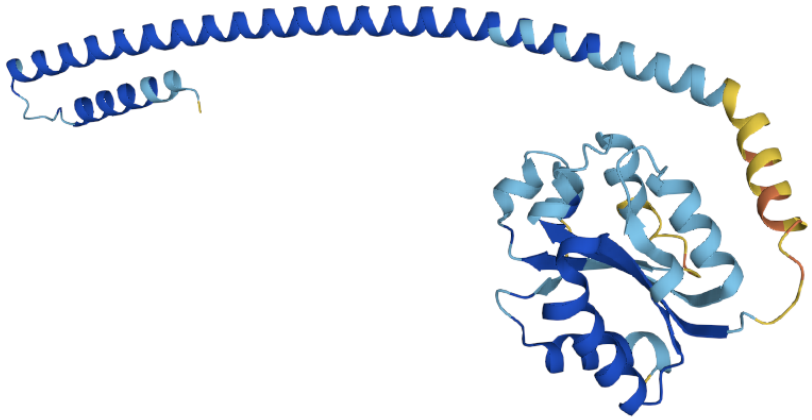


Figure 5.2: An example of a protein structure visualization in the AlphaFold style. The visualization highlights the secondary structure elements and overall fold of the protein.

5.3 Visualization

The visual representation of protein structures using the AlphaFold style, leveraging three-dimensional (3D) Protein Data Bank (PDB) visualizations.

The AlphaFold approach provides accurate protein structure predictions. It offers a unique visualization style that captures the essential features of protein architecture, as shown in Figure 5.2.

The AlphaFold-style visualization provides several key insights into the predicted protein structure:

- **Secondary Structure Elements**: The visualization highlights alpha-helices, beta-strands, and other secondary structure elements, enabling a quick assessment of the protein's fold.
- **Visual Confirmation**: Presenting the predicted protein structure in an AlphaFold-style visualization allows us to assess the accuracy and validity of the generated structures visually. This verification strengthens the credibility of the results.
- **Exemplifying Complex Relationships**: In complex scenarios where se-

quence motifs influence intricate structural elements, visualizations act as a visual narrative that simplifies the depiction of complex relationships. These visualizations spotlight how specific sequence characteristics influence the final three-dimensional fold and how complex it is; one of these is beta-sheets

5.4 Dataset

The database was extracted from UniProt. The database is also known as Swiss-Prot [78]. This database is a comprehensive, high-quality, manually annotated protein sequences and functional information resource. The Swiss Institute of Bioinformatics maintains it in collaboration with the European Molecular Biology Laboratory.

One prominent feature of Swiss-Prot is its high-quality annotation of protein sequences. Each sequence is curated by experts who review the available experimental evidence and literature to identify the protein's function, sub-cellular localisation, post-translational modifications, and other relevant information. This feature makes the model biased, generating knowledge of already proven valid proteins and can have a structure associated with it [79].

AlphaFold has introduced an additional feature, the plant value, representing the confidence level of the predicted protein sequence's 3D structure. This attribute provides insights into the protein's physical properties and increases the reliability of the predictions in real-world applications [6].

The data acquisition phase yielded 437 272 proteins, each characterised by unique size, function, and species features. Furthermore, each protein sequence was assigned a corresponding global plant value, which provides an estimate of the accuracy of the predicted 3D structure.

As illustrated in Figure 5.3, the dataset distribution highlights a higher prevalence of protein sequences at the edge of the plant versus sequence length graph. It suggests that the most reliable and least noisy data points are located in this area. Consequently, we have excluded all data points with scores below 70 and above 600 in sequence length to ensure high-quality data for further analysis.

As a result, the total number of proteins in the dataset decreased from 437,272 to 333,533. By filtering out proteins with plant scores below 70 and above 600 and limiting the sequence length, we introduced a more substantial bias towards shorter and 3D-oriented proteins more likely to represent valid structures in the real world.

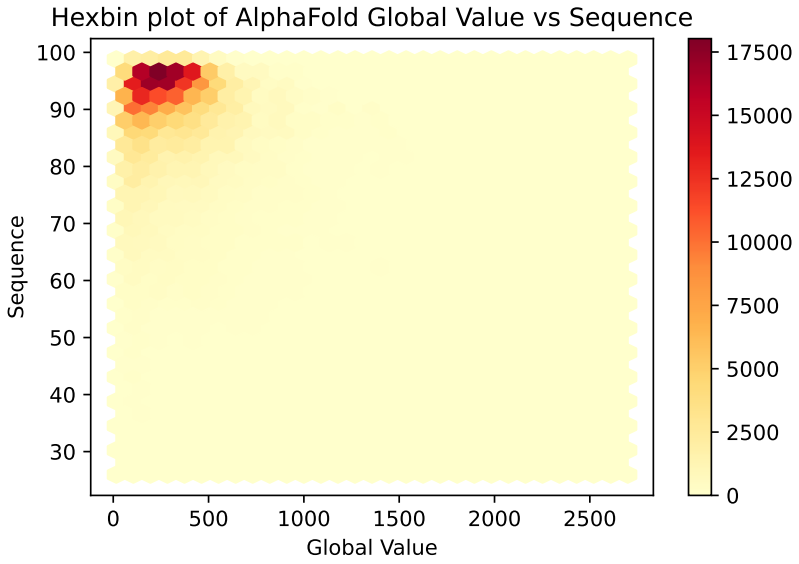
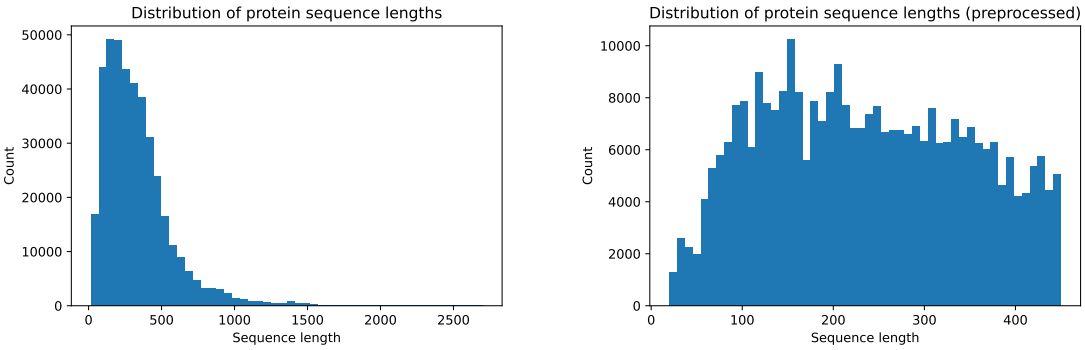


Figure 5.3: Protein vs AlphaFold Confidence Distribution

The difference in length distribution between the original and filtered datasets is shown in Figure 5.4.

After the dataset is defined, we apply tokenisation; tokenisation breaks down a protein sequence into smaller units, which can be used as inputs for downstream NLP tasks. In protein sequences, tokenisation can be achieved by representing each amino acid letter as a unique numerical value. To tokenise a protein sequence to individual characters with respective numbers, we can map each amino acid letter to its corresponding integer value.



(a) Original **(b)** Pre-Processed

Figure 5.4: Preprocessing process in the Swiss-Prot dataset

In addition to tokenising each amino acid letter, we can add special characters to indicate the start and end of the sequence and pad characters to ensure that all sequences have the same length. Garanting in the future that the model receives proteins of consistent size.

6

Results and Discussion

In protein generation exploration, it is crucial to draw comparisons between generated and random sequences to discern the distinctions and understand how an original sequence fits into the protein universe. This comparative analysis allows us to gauge the similarity between generated and genuine proteins while highlighting the differences from random strings of letters. Acknowledging that each generated protein may exhibit distinct characteristics, potentially deviating from naturally occurring proteins, is essential. The diversity in protein characteristics underscores the complexity of this field. To determine the validity of proteins, the possibility to do this was to check if there is a possibility of 3D proteins and compare them to different approaches. We use OmegaFold and the formula in 4.3.2 to obtain the global value and use the median to have a better resolution on the distribution. To add, this also follows the procedure of generating a sequence and then feeding it again to the model to add some corrections if needed or make it to the appropriate length. However, we must know if the model is learning, if it is better than some baseline models, if it can capture protein intricacies and if there is some novelty in generated proteins. Thus, there are five questions it must answer.

Table 6.1: Validation results for 200 proteins using OmegaFold's plDDT values for four different approaches.

	Mean	St. Dev.	1st Quartile	Median	3rd Quartile
Transformer with No RL	39.20	15.75	27.08	34.61	46.21
Transformer with RL	43.32	18.05	28.88	38.41	54.29
LSTM	32.55	14.97	25.66	28.51	33.78
Random	30.38	11.83	23.12	25.67	33.19

1. **Does Transformer Performance Improve with Reinforcement Learning for Protein Structure Prediction?**

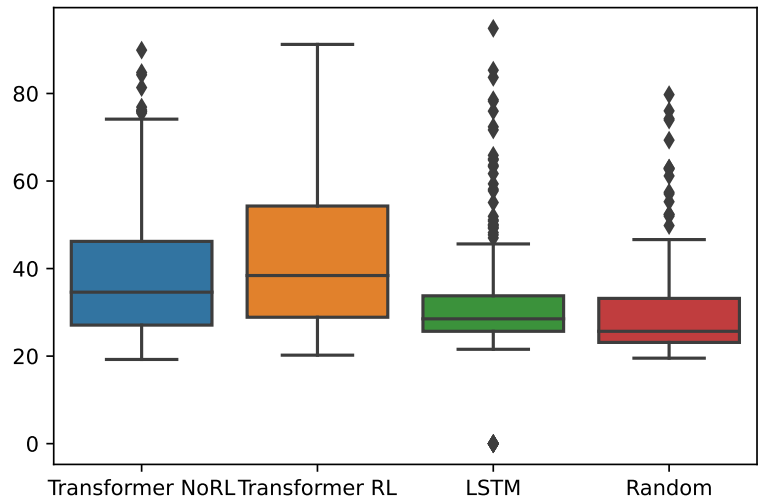


Figure 6.1: Comparison of Protein Structure Prediction Models: Box Plot depicting the distribution of plDDT values for the Transformer with No RL, Transformer with RL, LSTM, and Random

The proposed approach could also receive an extra training procedure using Reinforcement Learning using the OmegaFold as the Environment and check if it can be used as one. The plDDT values were also used to check if they were close to an accurate representation.

According to Table 6.1, we can conclude that the median plDDT score increases from 34.61 to 38.41 when RL is used. This median score suggests that the model's predicted structures are closer to the ground truth with RL. The mean plDDT score also increases from 39.20 to 43.32 with RL. This means that the predictions' overall quality tends to be higher when RL is used. The standard deviation increases from 15.75 to 18.05 when RL is used. This standard deviation score implies that the plDDT scores are more spread out from the mean when RL is employed. It could suggest that RL introduces more variability in prediction quality.

The Interquartile Range (IQR), represented by the difference between the third and the first quartiles, measures the spread of the middle 50% of the data. In both cases, the IQR is quite large, indicating substantial variation in the quality of predictions. The larger IQR in the RL condition ($Q3 - Q1 =$

$54.29 - 28.88 = 26.21$) compared to the non-RL condition ($Q3 - Q1 = 46.21 - 27.08 = 19.13$) suggests that while RL might lead to higher median and mean scores, it also introduces more variability in the quality of predictions.

Based on this data, using RL results in slightly better median and mean plDDT scores, which suggests that the model is better at generating protein sequences. However, the increased variability also implies that there might be cases where the model performs worse with RL.

It was also applied the t-test with a significance level of 0.05; conducting hypothesis tests can help determine if the observed differences are statistically significant and not by randomness. The premise is: Is there a significant difference between the two conditions ("No RL" and "Yes RL")?

- H_0: The two models have no significant difference.
- H_a: The two models have a significant difference.

The results were a t-statistic of -2.42 and a p-value of 0.0158. The t-statistic value -2.42 suggests that the means of the two conditions are significantly different. The negative sign indicates that the mean plDDT values for the "Yes RL" condition tend to be higher than for the "No RL" condition. The p-value of 0.0158 is less than the significance level of 0.05. This p-value is more significant since it will determine whether the null hypothesis is rejected. Since it is less than 0.05, it indicates that the difference is statistically significant at the 0.05 significance level. In other words, the likelihood of observing such a difference due to random chance is less than 0.05. However, it is also essential to consider other parameters, such as the plDDT distribution.

Given these results and observing the plDDT distribution in Figure 6.1, we can confidently conclude that using reinforcement learning (RL) in the model's predictions statistically impacts the plDDT values and produces fewer outliers. The "Yes RL" condition produces higher plDDT values than the "No RL" condition. However, it must also be added that the results can be biased due to the Environment being the OmegaFold.

2. **How does the proposed model behave against a simple LSTM and random sequences?**

By referring to Table 6.1 between the LSTM, the Random, and the Trans-

former with RL, the validation results for the LSTM approach exhibit a mean plDDT value of 32.55, indicating an unsatisfactory performance in generating protein sequences because it is distant from the threshold considered satisfactory in OmegaFold. The relatively high standard deviation (14.97) suggests considerable variability in performance across the 200 proteins evaluated. The IQR of 8.11 suggests that most predictions fall within a relatively narrower performance range, highlighting varying degrees of success across different proteins.

Contrasting the LSTM approach, the Random approach yields notably lower mean (30.38) and median (25.67) plDDT values, signifying a lack of meaningful predictive power for protein sequences. The minor standard deviation (11.83) and wider IQR (10.06) indicate a more consistent yet overall unsatisfactory performance, emphasizing the inability of the Random approach to provide accurate protein structure predictions as it was expected since the OmegaFold would give the sequences a deficient score since it does not follow a paradigm.

The Transformer with RL approach emerges as the most promising among the three methodologies, boasting the highest mean (43.32) and median (38.41) plDDT values, closer to the threshold in OmegaFold's confidence of 50. These higher results suggest a superior capability in predicting accurate protein structures compared to LSTM and Random approaches. However, the larger standard deviation (18.05) and substantial IQR (25.41) reveal notable variability in performance across different proteins. This variability could stem from the inherent complexity of protein structures and their diverse challenges for predicting the OmegaFold and its biases.

We further bolster our analysis by applying box plots to each case, as depicted in Figure 6.1. The utilization of box plots offers a visual representation that complements our earlier findings and augments the understanding of our results. Despite having already presented the quartiles and IQR in our initial analysis, including box plots introduces an additional layer of insights, particularly the detection of potential outliers that bear significance in the context of the examined sequence generation approaches.

Multiple 0 values in the LSTM case emerge as a distinct outlier. This peculiarity stems from instances where sequence generation failed to produce valid results; no sequence was generated or too small of a sequence to be considered

a protein sequence. While this occurrence introduces a challenge inherent to the LSTM approach, it simultaneously underscores the necessity for addressing such limitations in future iterations.

The presence of outliers in the Random case hints at instances where the unpredictable nature of the random sequence generation led to unexpected outcomes. This phenomenon underscores the inherent variability and randomness inherent to the approach, highlighting the essential role of controlled methodologies for sequence generation.

The most intriguing outliers are observed within the plDDT. Extremely high plDDT values beckon attention due to their potential implications. These instances may signify predictions that remarkably mirror experimental data, hinting at high-confidence regions or structurally accurate motifs within the generated sequences. However, it is essential to tread with caution, as the possibility exists that such extreme plDDT values could be the result of coincidental alignment with experimental data—a fortunate occurrence rather than a definitive, accurate prediction.

The distinction between true accuracy and fortuitous alignment becomes evident when contrasting these extreme plDDT values with those emerging from the Random case. Here, the fortuity of achieving high plDDT values highlights the chance nature of the phenomenon. This revelation reiterates the necessity for robust methodologies that transcend chance and can consistently produce high-quality predictions.

It was also applied a t-test comparing the LSTM model and the Transformer to confirm that the model performs better, helping determine whether the observed differences in mean performance between the two models are likely due to chance or are genuinely reflective of a difference in model effectiveness.

- H_0: The two models have no significant difference.
- H_a: The two models have a significant difference.

The results were a t-statistic of approximately -6.48 and a p-value extremely low of $2.83e-10$. With these results, it can be determined that the calculated t-statistic is approximately -6.48. This t-statistic measures the magnitude of the difference between the means of the two approaches' plDDT values, indicating that the Transformer approach's mean is significantly higher than that

of LSTM. The p-value obtained from the t-test is remarkably low, approximately $2.83e-10$. This p-value signifies the probability of observing such extreme differences in means by chance. A p-value of this magnitude is well below conventional significance thresholds, providing strong evidence to reject the null hypothesis that there is no significant difference between the two approaches. Instead, it supports the conclusion that the difference in means is statistically significant.

In our exploration of protein sequence generation methodologies, we systematically investigated the capabilities of deep learning models. Our analysis, conducted through both visual and quantitative approaches, revealed valuable insights.

Box plots were employed to illustrate data distribution and characteristics, shedding light on distinct patterns that highlight the inherent variability in each approach. Additionally, statistical validation using t-tests highlighted significant performance disparities between the Transformer and LSTM models, solidifying the credibility of our findings.

Table 6.1 succinctly summarizes key statistics, providing a concise overview of each model's performance.

In summary, our exploration unveiled a noteworthy contrast: advanced models offer precision in protein sequence generation compared to the stochastic nature of random approaches. This emphasizes the transformative potential of deep learning methods, particularly the Transformer architecture, in generating protein sequences with predictive capabilities.

3. **Does the proposed model capture common and complex secondary structures?**

In protein structure prediction, the accurate characterization of secondary structures, including alpha helices and beta sheets, is fundamental. This aims to provide insight into the model's effectiveness in protein sequence generation.

The prevalence of Alpha-Helices observed in 98.5% proteins highlights the tendency for this well-defined secondary structure element to emerge. Alpha-helices are shared and stable motifs formed by hydrogen bonds between neighbouring residues along the polypeptide chain. The recurring pattern of Alpha-

Table 6.2: Secondary Structure Composition Analysis of 200 Protein Sequences Generated from Scratch and 368 Proteins with Fully Known and Valid Structures: A Per-Protein Classification Perspective

	Alpha-Helix	Beta Sheets	Coil	3_{10} Helix	π Helix
Generated Proteins from Transformer RL	98.5%	51.5%	100%	71.0%	10.5%
Generated Proteins from LSTM	94.0%	64.8%	95.9%	78.9%	8.5%
Natural Proteins	97.0%	96.2%	100%	95.9%	24.7%

Helices contributes to their predictability.

Coils are dynamic regions without a fixed secondary structure, allowing proteins to adopt diverse conformations. Their ubiquity across all 200 generated proteins underscores the intrinsic flexibility and variability of protein sequences. Coil regions challenge precise sequence-to-structure predictions, reflecting the intricate interplay of factors governing protein folding.

Although Beta Sheets are relatively common, their formation introduces intricacies due to precise hydrogen bonding alignment and potential steric hindrances. Beta Sheets are present in 51.5 proteins, essential for structural diversity in the generated sequences. Their coexistence with other structural elements emphasizes the complexity inherent in protein sequence design.

The occurrence of 3_{10} Helices in 71% proteins and π Helices in 10.5% proteins introduces additional layers of complexity. These less common secondary structure elements are notable for their distinctive conformations, often arising in specific structural contexts. The presence of these helices in the generated sequences demonstrates their rich structural diversity.

When comparing the secondary structure compositions of generated proteins to natural proteins of similar size and scientifically validated structures in Table 6.2, notable differences become apparent. Firstly, generated proteins tend to have a higher percentage of Alpha-Helices, indicating that the model predictor is proficient in generating this structural element. However, there are notable shortcomings in replicating Beta Sheets in generated proteins, and we observe an overrepresentation of 3_{10} Helices and π Helices. These deviations suggest that the predictor may have biases, limitations, or a lack of representation in the dataset.

Another intriguing observation is the 100% representation of Coil regions in

generated proteins. This leads us to question how unassigned or ambiguous structural regions are classified in this context. Importantly, it's crucial to remember that the source of these generated structures is a protein folding model predictor. This predictor introduces inherent biases and does not consider environmental factors, such as pH, temperature, or ionic strength, which can significantly impact protein stability and structure.

Coil regions may become more prevalent in specific environmental conditions due to the disruption of hydrogen bonds or other stabilizing interactions that typically favour the formation of Alpha-Helices or Beta Sheets. The inability of predictive models to account for such environmental influences contributes to the observed differences between generated and natural proteins.

In comparing the secondary structure composition of proteins generated by two distinct models, Transformer RL and LSTM, notable differences emerge. The Transformer RL model excels in generating Alpha-Helices, making it a strong choice for applications where this secondary structure element is crucial. On the other hand, the LSTM model demonstrates superior performance in generating Beta Sheets, offering an advantage for those seeking proteins rich in this structure. Both models exhibit a high representation of Coil regions. The LSTM model has a slight edge in generating 3_{10} Helices, while both models exhibit a low representation of π Helices. However, the distance of both models in these structures is too small to have a preference just for the structure.

In summary, the proposed model excels in generating Alpha-Helices but faces limitations in replicating other structural elements to the same level as in nature. The high occurrence of Coil regions in generated proteins may be due to unresolved structural regions and the predictor's omission of environmental factors. These findings underscore the need to understand the predictor's biases and the impact of environmental conditions on protein structure. Our analysis reveals the diverse landscape of structural motifs, from predictable Alpha-Helices to adaptable Coils and precision-demanding Beta Sheets.

4. **Do the Amino Acids distribute similarly to the naturally occurring proteins?**

   The comparison of amino acid distribution between the generated and naturally occurring proteins provides valuable insights into the characteristics of the generated sequences. As can be seen in Table 6.3, Amino acids such as

Methionine (M), Glutamine (Q), and Tryptophan (W) show differences in distribution between the two datasets. These variations may indicate unique preferences in the sequence generation process, or their representation in the dataset could be more predominant. A notable contrast is seen in Lysine (K), which is significantly more prevalent in the generated proteins than the natural ones. This contrast suggests a potential bias in the sequence generation model towards incorporating lysine-rich sequences.

The overrepresentation of Lysine (K) and Valine (V) in the generated proteins indicates distinct sequence preferences. These patterns could be driven by the architecture of the sequence generation model or other factors that influence amino acid selection. However, Aspartic Acid (D) and Serine (S) exhibit similar distributions in both datasets. This similarity suggests that the sequence generation process captured some common patterns in natural sequences.

The varying amino acid distributions could have implications for the structural and functional properties of the generated proteins. For instance, the elevated presence of Lysine (K) might impact the sequences' electrostatic interactions and folding propensity.

While the differences in amino acid distribution are evident, their statistical significance should be assessed. Conducting hypothesis tests can determine whether the observed deviations are statistically meaningful or arise due to chance. A used test for comparing two groups of categorical data, such as amino acid counts, is the Chi-squared test of independence. So the hypothesis is:

- H_0: No significant difference in the amino acid distribution between the generated and naturally occurring proteins exists.
- H_a: There is a significant difference in the distribution of amino acids between the two datasets.

After applying the Chi-squared test, it gave an insight into the correlation between both distributions.

- **Chi-squared statistic:** The calculated Chi-squared statistic is approximately 6.83. This statistic represents the overall divergence between the observed and expected counts of amino acids in the two datasets. The statistical value serves as an indicator of the magnitude of disparity

Table 6.3: Comparison of Amino Acid Distribution in 200 Generated Proteins and Natural Occurring Proteins

Amino Acid	Generated (%)	Natural Occurring (%)
M	1.39	2.32
E	5.22	6.32
T	4.00	5.53
P	4.14	5.02
S	5.59	7.14
K	12.92	5.19
V	5.01	6.73
L	11.69	9.68
N	3.81	3.93
Q	3.07	3.90
F	3.18	3.87
Y	2.34	2.91
I	8.76	5.49
D	3.82	5.49
R	4.25	5.78
A	9.56	8.76
H	1.50	2.26
W	0.39	1.25
G	8.07	7.03
C	1.28	1.38

between the distributions.

- **Degrees of freedom:** The degree of freedom is 19, the number of amino acids being compared minus 1. This value helps determine the critical value for assessing the statistical significance.
- **p-value:** The p-value is approximately 0.995. This p-value reflects the probability of observing the observed differences (or more extreme differences) in amino acid distribution between the generated and naturally occurring proteins if there were no actual differences in the distributions.

The result with a p-value of approximately 0.995 suggests no statistically significant evidence to reject the null hypothesis. In other words, the observed differences in amino acid distribution between the two datasets could plausibly occur due to random variation rather than reflecting an actual difference in distribution. It is essential to consider the high p-value in context. The high p-value suggests that the observed differences are not substantial enough

to be confidently attributed to differences in amino acid distribution between the generated and naturally occurring proteins. When discussing these results, addressing the limitations and implications is valuable. While the Chi-squared test did not identify a statistically significant difference, other factors, such as sample size or model biases, could contribute to the observed variations.

In conclusion, our analysis provides valuable insights into the amino acid distribution in generated protein sequences. While statistical tests suggest non-significant differences, the implications extend beyond statistical significance. These findings underscore the nuanced interplay between computational methods and biological complexity, motivating further exploration and refinement of protein sequence generation approaches.

5. **To what extent do generated protein sequences deviate from natural proteins, and how do these deviations contribute to unique protein architectures?**

   To answer this question, a chimeric approach was used. Chimeric proteins result from the fusion of segments from two or more distinct genes. These segments can come from different genes within the same genome or from different organisms. Chimeric sequences can affect protein function, structure, and biological roles.

   Additionally, the very nature of chimeric proteins, formed by combining segments from disparate proteins, inherently leads to combined functional domains, motifs, chemical values, and structural features that might not be commonly observed in natural proteins. This fusion of diverse elements could engender novel functionalities or unique binding properties.

   To find if a protein is chimeric, we used BLAST. We compared the 200 generated proteins to the NR database (subsection 2.2.3) and determined if it has more than one alignment with different protein families, having an e-value less than 10.

   In the end, 42 chimeric sequences among the generated set were present, underscoring the deliberate integration of distinct protein segments. This prevalence of chimerism contributes to a departure from conventional protein composition, manifesting as a unique subset defined by fused segments from disparate origins.

The analysis highlights discernible deviations in structural attributes within the generated protein sequences. The subset of chimeric sequences, constituting almost one-quarter of the total, serves as a focal point of distinction. Chimeric assembly introduces unique structural motifs that deviate from the norms observed in naturally occurring proteins.

Although the term "chimerism" remains implicit, its influence is evident through the alignment of chimeric sequences against the NR database. This alignment underlines the biological relevance of these sequences and their resonance with known entities. This unspoken interaction underscores chimeric influence in shaping unique protein architectures.

Chimerism, the fusion of distinct protein segments, can impact the plDDT scores of the generated protein sequences. As chimeric sequences integrate segments from different sources, they might introduce regions of more significant structural uncertainty due to the blending of diverse structural motifs. Consequently, specific regions within chimeric sequences might exhibit lower plDDT scores, reflecting higher structural uncertainty or variability. These scores can explain the values having a mean plDDT value of 43 and also explain when some areas on some proteins have high scores and then low scores on others zones, and this can be observed in Figure 6.2.

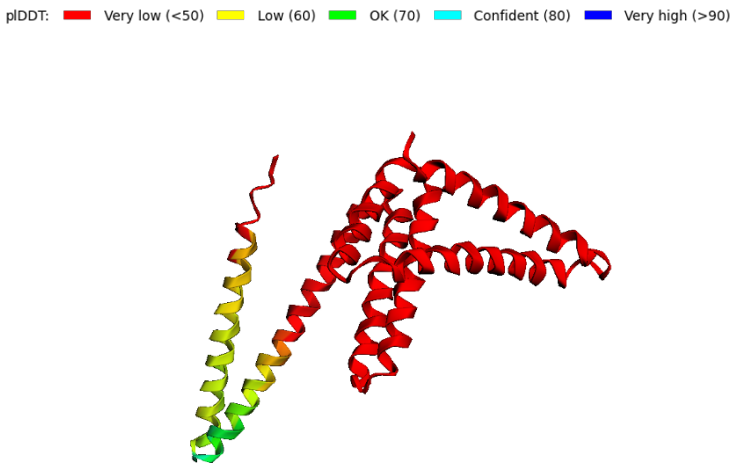


Figure 6.2: Chimeric Protein Structure Generated by the Proposed Model

In conclusion, the investigation into the characteristics and properties of the

generated protein sequences unveils discernible deviations from their natural counterparts. The chimeric subset, supported by NR database alignment, constitutes a distinct dimension of this analysis. These deviations, marked by structural and functional variations, provide insights into the potential emergence of unique protein architectures—an arena ripe for further exploration within protein science.

6.1 Shortcomings

While our study has yielded valuable insights into protein sequence generation using advanced machine learning techniques, several limitations inherent to the methodology and constraints of the investigation merit careful consideration. While not diminishing the significance of our findings, these limitations underscore the complexities and nuances inherent in the pursuit of accurate protein structure prediction.

Model Complexity and Computational Resources: The utilization of an intricate Transformer model—acknowledged for its prowess in sequence-to-sequence tasks—introduced computational challenges. Specifically, the model's architectural depth and parameter richness demanded substantial computational resources. However, the formidable computational demands were accompanied by a trade-off in execution time. In future work, generative transformer models such as GPT could be used, a State-of-the-Art model in generating sequences. Despite our use of AlphaFold, driven by the unavailability of OmegaFold at the onset of the study, the substantial time investment required for generating proteins, even with the former's efficiency, remained a factor.

Protein Length and Training Constraints: The inherent complexity of proteins, particularly those surpassing a length of 300 residues, introduces challenges in training deep learning models efficiently. Our study encountered limitations stemming from the impracticality of training on proteins of realistic length within reasonable time frames. While our investigation focused on sequences of lengths manageable within our computational resources, we acknowledge that this limitation might restrict the generalizability of our findings to longer protein sequences.

Absence of Comprehensive Evaluation Metrics: A notable limitation pertains to the evaluation metrics employed in our study. The absence of comprehensive metrics tailored to sequence generation and structural prediction necessitated creative adaptations of existing metrics. While our evaluation criteria included fundamental

sequence validity, the absence of specialized metrics for assessing the structural fidelity of generated sequences merits attention. This limitation emphasizes the need to develop rigorous and contextually relevant evaluation methodologies.

Complexity of Reinforcement Learning: Our foray into employing reinforcement learning (RL) introduced complexities inherent to this paradigm. The lack of a universally established reward function framework demanded iterative trial-and-error exploration. This process may inadvertently introduce biases in the policy function's design, underscoring the intricate interplay between algorithmic architecture and the problem domain.

Data Accessibility and Bias: Our study was conducted on accessible datasets, eliminating potential data-related challenges. Nonetheless, the convenience of data accessibility gives rise to apprehensions regarding the adequacy of representation and the potential introduction of bias within the training data. Ensuring a comprehensive and diverse dataset that accurately mirrors the complexity and diversity of protein structures remains an ongoing challenge in protein structure prediction.

7

Conclusion

This investigation's core is a comprehensive analysis of protein generation, viewed through the lens of sequence data. As we conclude this research, we thoroughly assess the implications of our findings within the context of protein science.

Our primary objective was to systematically assess the capabilities of machine learning models in generating intricate protein sequences. To achieve this goal, we conducted an in-depth analysis of sequence data to uncover the underlying factors influencing the generation of these vital biological components. Our investigation involved three fundamental models: LSTM, Transformer, and a framework combining the Transformer with OmegaFold, allowing us to assess their protein generation potential.

We have demonstrated some ability to generate protein sequences with complex secondary structures using machine learning models, particularly Transformers. This discovery underscores the presence of predictability and latent patterns within protein sequences. It becomes evident that the outcomes are not random but reflect the model's capacity to decipher and construct structured sequences. Notably, these structured elements encompass alpha helices, beta sheets, and coils, which are integral to protein architecture, serving both functional and structural roles. Our findings shed light on the capability of machine learning models to simulate intricate structural motifs, thereby advancing our understanding of protein sequence generation within a computational framework.

We applied a robust statistical approach further to bolster the credibility and significance of our exploration. OmegaFold's mean value of 43.32 and its positively skewed distribution underscore the model's proficiency in protein generation. Additionally, by employing t-tests, we quantified the significant performance gap between the Transformer and LSTM models, providing statistical validation for our observed

trends. Furthermore, the t-test comparing the Transformer's performance with that of random sequences reaffirmed that the model had learned and could handle the complexity inherent in protein sequences.

In conclusion, our research has illuminated the potential of machine learning models, particularly the Transformer, in generating intricate protein sequences with complex secondary structures. These findings underscore the predictive power inherent in protein sequences and their generation.

7.1 Future Work

In protein sequence generation, numerous promising avenues beckon for exploration. Among these, the utilization of transfer learning stands out as a potent tool. Leveraging transfer learning to engineer protein sequences with specific attributes is promising for real-world applications.

Our quest to design proteins with tailored traits—ranging from stability in extreme conditions to hydrophobicity and more—confronts a challenge: data scarcity. To overcome this problem, Transfer Learning emerges as a potential solution, drawing insights from diverse datasets to inform the creation of sequences optimized for distinct functionalities.

In addition to Transfer Learning, we explore the potential of deploying advanced models suited for sequence generation tasks. Models like GPT, initially designed for language tasks, serve as a strong foundation. We explore how integrating such advanced models can enhance our ability to design sequences with specific characteristics, ushering in new possibilities.

Furthermore, we emphasize the importance of evaluation metrics. Generating sequences with known traits enables the establishment of benchmarks for comparison. This approach allows for a more nuanced assessment of the model's predictive capabilities. The interplay between generated sequences and naturally occurring proteins possessing similar attributes has the potential to advance our understanding of sequence-structure relationships and the reliability of predictions.

A

Appendix

Appendix A

Table A.1: Amino Acids

Amino Acid	3-letter	1-letter	Ambudance in Proteins (%) [80]
Alanine	Ala	A	8.76
Arginine	Arg	R	5.78
Asparagine	Asn	N	3.93
Aspartate	Asp	D	5.49
Cysteine	Cys	C	1.38
Glutamine	Gln	Q	3.9
Glutamate	Glu	E	6.32
Glycine	Gly	G	7.03
Histidine	His	H	2.26
Isoleucine	Ile	I	5.49
Leucine	Leu	L	9.68
Lysine	Lys	K	5.19
Methioninr	Met	M	2.32
Phenylalanine	Phe	F	3.87
Proline	Pro	P	5.02
Serine	Ser	S	7.14
Threonine	Thr	T	5.53
Tryptophan	Trp	W	1.25
Tyrosine	Tyr	Y	2.91
Valine	Val	V	6.73

Table A.2: Ambiguous Amino Acids

Ambiguous Amino Acids	3-letter	1-letter	Amino Acids included
Any / Unknown	Xaa	X	All
Asparagine or Aspartate	Asx	B	D, N
Glutaine or Glutamate	Glx	Z	E, Q
Leucine or Isoleucine	Xle	J	I, L

Bibliography

[1] R. Pearce and Y. Zhang, "Deep learning techniques have significantly impacted protein structure prediction and protein design," *Current opinion in structural biology*, vol. 68, pp. 194–207, 2021.

[2] Y. Zeng, T. Zhao, and A. R. Kermode, "A conifer abi3-interacting protein plays important roles during key transitions of the plant life cycle," *Plant Physiology*, vol. 161, no. 1, pp. 179–195, 2013.

[3] F. Gebauer, T. Schwarzl, J. Valcárcel, and M. W. Hentze, "Rna-binding proteins in human genetic disease," *Nature Reviews Genetics*, vol. 22, no. 3, pp. 185–198, 2021.

[4] S. Aydin, "A short history, principles, and types of elisa, and our laboratory experience with peptide/protein analyses using elisa," *Peptides*, vol. 72, pp. 4–15, 2015.

[5] A. Ballone, "A new soaking procedure for x-ray crystallographic structural determination of protein-peptide complexes," *Acta Crystallographica Section F: Structural Biology Communications*, 10 2020.

[6] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, and et al., "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, p. 583–589, 2021.

[7] T. U. Consortium, "UniProt: the Universal Protein Knowledgebase in 2023," *Nucleic Acids Research*, vol. 51, no. D1, pp. D523–D531, 11 2022. [Online]. Available: https://doi.org/10.1093/nar/gkac1052

[8] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[9] D. Komander, "The emerging complexity of protein ubiquitination," *Biochemical society transactions*, vol. 37, no. 5, pp. 937–953, 2009.

[10] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "Disan: Directional self-attention network for rnn/cnn-free language understanding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[11] A. D. Hanson, A. Pribat, J. C. Waller, and V. d. Crécy-Lagard, "'unknown'proteins and 'orphan'enzymes: the missing half of the engineering parts list–and how to find it," *Biochemical Journal*, vol. 425, no. 1, pp. 1–11, 2010.

[12] K. Yamasaki, V. T. G. Chuang, T. Maruyama, and M. Otagiri, "Albumin–drug interaction and its clinical implication," *Biochimica et Biophysica Acta (BBA)-General Subjects*, vol. 1830, no. 12, pp. 5435–5443, 2013.

[13] G. M. West, C. L. Tucker, T. Xu, S. K. Park, X. Han, J. R. Yates III, and M. C. Fitzgerald, "Quantitative proteomics approach for identifying protein–drug interactions in complex mixtures using protein stability measurements," *Proceedings of the National Academy of Sciences*, vol. 107, no. 20, pp. 9078–9082, 2010.

[14] E. A. Nigg, "Origins and consequences of centrosome aberrations in human cancers," *International journal of cancer*, vol. 119, no. 12, pp. 2717–2723, 2006.

[15] J. Li, L. Chen, Y.-H. Zhang, X. Kong, T. Huang, and Y.-D. Cai, "A computational method for classifying different human tissues with quantitatively tissue-specific expressed genes," *Genes*, vol. 9, no. 9, p. 449, Sep 2018. [Online]. Available: http://dx.doi.org/10.3390/genes9090449

[16] M. E. Steiper, "Bioinformatics: A practical guide to the analysis of genes and proteins. edited by andreas d. baxevanis and b. f. francis ouellette. new york: John wiley and sons. 2001. 649 pp. isbn 0-471-38390-2." *American Journal of Physical Anthropology*, vol. 127, no. 1, p. 125–126, 2005.

[17] D. Baker and A. Sali, "Protein structure prediction and structural genomics," *Science*, vol. 294, no. 5540, p. 93–96, 2001.

[18] D. Okada, F. Ino, and K. Hagihara, "Accelerating the smith-waterman algorithm with interpair pruning and band optimization for the all-pairs comparison of base sequences," *BMC Bioinformatics*, vol. 16, no. 1, 2015.

[19] R. Singh, M. Kumar, A. Mittal, and P. K. Mehta, "Microbial enzymes: Industrial progress in 21st century," *3 Biotech*, vol. 6, no. 2, 2016.

[20] A. Gruber, *A06*. National Center for Biotechnology Information (US).

[21] U. Ejaz, M. Sohail, and A. Ghanemi, "Cellulases: From bioactivity to a variety of industrial applications," *Biomimetics*, vol. 6, no. 3, p. 44, 2021.

[22] "Amino acid," accessed in 31-Nov-2023. [Online]. Available: https://www.britannica.com/science/amino-acid

[23] S. Hormoz, "Amino acid composition of proteins reduces deleterious impact of mutations," *Scientific Reports*, vol. 3, no. 1, 2013.

[24] R. P. Bywater, "Why twenty amino acid residue types suffice(d) to support all living systems," *PLOS ONE*, vol. 13, no. 10, 2018.

[25] Y. Fan, C. R. Evans, and J. Ling, "Rewiring protein synthesis: From natural to synthetic amino acids," *Biochimica et Biophysica Acta (BBA) - General Subjects*, vol. 1861, no. 11, p. 3024–3029, 2017.

[26] R. K. Murray, D. K. Granner, V. W. Rodwell, and H. A. Harper, *Harper's Illustrated Biochemistry*. Lange Medical Books/McGraw-Hill, 2006.

[27] M. H. Ahmed, M. S. Ghatge, and M. K. Safo, "Hemoglobin: Structure, function and allostery," *Subcellular Biochemistry*, p. 345–382, 2020.

[28] Image, accessed in 30-Nov-2023. [Online]. Available: https://www.cdn.kastatic.org/ka-perseus-images

[29] "Biol 152: Minimalist design of peptide and protein catalysts." [Online]. Available: https://plan.core-apps.com/acsnola2018/abstract/3b00ff75f22454219cca274e14edadf8

[30] S. A. Marshall, G. A. Lazar, A. J. Chirino, and J. R. Desjarlais, "Rational design and engineering of therapeutic proteins," *Drug Discovery Today*, vol. 8, no. 5, p. 212–221, 2003.

[31] Y. Gu, X. Xu, Y. Wu, T. Niu, Y. Liu, J. Li, G. Du, and L. Liu, "Advances and prospects of bacillus subtilis cellular factories: From rational design to industrial applications," *Metabolic Engineering*, vol. 50, p. 109–121, 2018.

[32] F. Ding and N. V. Dokholyan, "Emergence of protein fold families through rational design," *PLoS Computational Biology*, vol. 2, no. 7, 2006.

[33] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church, "Unified rational protein engineering with sequence-based deep representation learning," *Nature Methods*, vol. 16, no. 12, p. 1315–1322, 2019.

[34] S. S. Linse, "Directed evolution of enzymes and binding proteins," accessed in 22-Nov-2022. [Online]. Available: https://www.nobelprize.org/uploads/2018/10/advanced-chemistryprize-2018.pdf

[35] S. M. GENDEL, "Sequence analysis for assessing potential allergenicity," *Annals of the New York Academy of Sciences*, vol. 964, no. 1, p. 87–98, 2006.

[36] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, p. 443–453, 1970.

[37] P. Compeau and P. Pevzner, *5*, 2nd ed. Active Learning Publishers, 2015, vol. 1.

[38] C. Barton, T. Flouri, C. S. Iliopoulos, and S. P. Pissis, "Global and local sequence alignment with a bounded number of gaps," *Theoretical Computer Science*, vol. 582, pp. 1–16, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397515002200

[39] "Blast: Basic local alignment search tool," accessed in 10-Nov-2023. [Online]. Available: https://blast.ncbi.nlm.nih.gov/Blast.cgi

[40] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, p. 403–410, 1990.

[41] S. McGinnis and T. L. Madden, "Blast: at the core of a powerful and diverse set of sequence analysis tools," *Nucleic acids research*, vol. 32, no. suppl_2, pp. W20–W25, 2004.

[42] M. Karimi, S. Zhu, Y. Cao, and Y. Shen, "De novo protein design for novel folds using guided conditional wasserstein generative adversarial networks (gcwgan)," 2019.

[43] S. Sabban and M. Markovsky, "Ramanet: Computational de novo helical protein backbone design using a long short-term memory generative adversarial neural network," *F1000Research*, vol. 9, p. 298, 2020.

[44] R. Sharan, I. Ulitsky, and R. Shamir, "Network-based prediction of protein function," *Molecular Systems Biology*, vol. 3, no. 1, p. 88, 2007.

[45] R. Bonneau, J. Tsai, I. Ruczinski, D. Chivian, C. Rohl, C. E. Strauss, and D. Baker, "Rosetta in casp4: Progress in ab initio protein structure prediction," *Proteins: Structure, Function, and Genetics*, vol. 45, no. S5, p. 119–126, 2001.

[46] G. A. Khoury, J. Smadbeck, C. A. Kieslich, and C. A. Floudas, "Protein folding and de novo protein design for biotechnological applications," *Trends in Biotechnology*, vol. 32, no. 2, pp. 99–109, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167779913002266

[47] X. L. Liu, "Deep recurrent neural network for protein function prediction from sequence," 2017.

[48] R. Cao, C. Freitas, L. Chan, M. Sun, H. Jiang, and Z. Chen, "Prolango: Protein function prediction using neural machine translation based on a recurrent neural network," *Molecules*, vol. 22, no. 10, p. 1732, 2017.

[49] H. Ni, L. Szpruch, M. Wiese, S. Liao, and B. Xiao, "Conditional sig-wasserstein gans for time series generation," *SSRN Electronic Journal*, 2020.

[50] D. Grechishnikova, "Transformer neural network for protein-specific de novo drug generation as a machine translation problem," *Scientific Reports*, vol. 11, no. 1, 2021.

[51] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[52] T. Mikolov, "Statistical language models based on neural networks," Ph.D. thesis, Brno University of Technology, Faculty of Information Technology, 2012. [Online]. Available: https://www.fit.vut.cz/study/phd-thesis/283/

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[54] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[55] J. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed, "Supervised, unsupervised, and semi-supervised feature selection: A review on gene selection," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 13, pp. 1–1, 09 2015.

[56] E. Grossi and M. Buscema, "Introduction to artificial neural networks," *European Journal of Gastroenterology amp; Hepatology*, vol. 19, no. 12, p. 1046–1054, 2007.

[57] "Artificial neural network for drug design, delivery and disposition," 2016.

[58] S. Salman and X. Liu, "Overfitting mechanism and avoidance in deep neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1901.06566

[59] S. Kostadinov, "Understanding backpropagation algorithm," Aug 2019. [Online]. Available: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

[60] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," 2017. [Online]. Available: https://arxiv.org/abs/1702.01923

[61] D. Datta, P. Evangeline, D. Mittal, and A. Jain, "Neural machine translation using recurrent neural network," *International Journal of Engineering and Advanced Technology*, vol. 9, pp. 1395–1400, 04 2020.

[62] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," 01 2011, pp. 1017–1024.

[63] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018. [Online]. Available: http://arxiv.org/abs/1808.03314

[64] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, no. 10, p. 2451–2471, 2000.

[65] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, pp. 107–116, 04 1998.

[66] M. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," *Complex Systems*, vol. 3, 01 1995.

[67] "Statistical language models based on neural networks." [Online]. Available: https://www.fit.vut.cz/study/phd-thesis/283/.en

[68] S. Hao, D.-H. Lee, and D. Zhao, "Sequence to sequence learning with attention mechanism for short-term passenger flow prediction in large-scale metro system," *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 287–300, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X19300245

[69] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[70] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[71] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[72] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel, "A survey of reinforcement learning informed by natural language," *arXiv preprint arXiv:1906.03926*, 2019.

[73] R. Wu, F. Ding, R. Wang, R. Shen, X. Zhang, S. Luo, C. Su, Z. Wu, Q. Xie, B. Berger, J. Ma, and J. Peng, "High-resolution de novo structure prediction from primary sequence," *bioRxiv*, 2022. [Online]. Available: https://www.biorxiv.org/content/early/2022/07/22/2022.07.21.500999

[74] Embl-Ebi, "Current release statistics." [Online]. Available: https://www.ebi.ac.uk/uniprot/TrEMBLstats

[75] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[76] A. R. Naik, "Protein wars part 2: It's omegafold vs alphafold," Aug 2022, accessed in 20-July-2023. [Online]. Available: https://analyticsindiamag.com/protein-wars-part-2-its-omegafold-vs-alphafold/

[77] Benchmark in Machine Learning Methods for Protein Folding, accessed in 20-July-2023. [Online]. Available: https://310.ai/2023/05/17/benchmarking-machine-learning-methods-for-protein-folding-a-comparative-study-of-esmfold-omegafold-and-alphafold/

[78] 2023, accessed in 23-July-2023. [Online]. Available: https://www.uniprot.org/help/downloads

[79] S. Bienert, A. Waterhouse, T. A. De Beer, G. Tauriello, G. Studer, L. Bordoli, and T. Schwede, "The swiss-model repository—new features and functionality," *Nucleic acids research*, vol. 45, no. D1, pp. D313–D319, 2017.

[80] L. P. Kozlowski, "Proteome-pi: Proteome isoelectric point database," *Nucleic Acids Research*, vol. 45, no. D1, 2016.

This page is intentionally left blank.