Natural Language Processing for Language of Life (mRNA vaccine design)

by

Shashank Pathak

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

The COVID-19 pandemic accelerated the development of mRNA vaccines, yet identifying the optimal mRNA sequence for human use, particularly for the SARS-CoV-2 spike protein, remains challenging. This thesis focuses on optimizing the open reading frame (ORF), a crucial mRNA component composed of codons—triplets of nucleotides coding for amino acids. We introduce a novel 'valid-codon' masking strategy to streamline codon-to-amino acid mapping within the target protein sequence. This approach was competitive to the 'codon-box' method, which groups codons with identical nucleotide compositions. Our findings show that 'valid-codon' performs comparably to 'codon-box' in optimizing ORF sequences for gene expression. By integrating the masking strategy into a supervised fine-tuning (SFT) process using the pre-trained ProtBert model, we further optimize the ORF for humans for the SARS-CoV-2 spike protein. Results indicate that our fine-tuned models surpass the ORF sequences used in Moderna and Pfizer vaccines in terms of gene expression and stability.

Acknowledgements

I want to express my deepest gratitude to Professor Guohui Lin for his guidance, encouragement, and valuable advice during my Master's journey. His expertise has been critical in shaping this work and pushing my research potential.

Next, I would like to thank all the dissertation committee members (Prof. Guohui Lin, Prof. Nilanjan Ray, and Prof. Osmar Zaiane) for their invaluable time and efforts in reading my thesis and providing valuable feedback for improvement. I would like to thank Carson Barnstable for helping me with the initial data collection. I would also like to thank IST for providing me with technical support for running my experiments.

I am deeply grateful to my father (Chandra Kant Pathak), mother (Meera Pathak), and elder brother (Siddhant Pathak), who have always kept me uplifted during my lows and shown firm belief in me. I am grateful to all of my friends at UofA, especially Ranit, Supreeth, Kamal, Yogesh, Yash, Piyush, Mukul, Arghashree, Kushankur, Ganesh, and Mohan; their encouragement and support made this journey more exciting and fun in difficult times. Special thanks to Shashank Srivastava, Animesh Jaiswal and Nitin Singh for their unwavering support. Finally, I would like to thank Dr.Abhishek Srivastava for providing me with the initial opportunity to do research and for his constant guidance.

Table of Contents

List of Tables

List of Figures

Abbreviations

A: Adenine.

aa: Amino Acids.

API: Application Programming Interface.

BERT: Bidirectional Encoder Representation from Transformer.

bi-LSTM: Birdirectional Long Short Term Memory.

C: Cytosine.

CAI: Codon Adaptation Index.

CDF: Cumulative Distribution Function.

CH: Chinese-Hamster.

CHO: Chinese Hamster Ovary Cells.

CNN: Convolutional Neural Network.

DNA: Deoxy-Ribonucleic Acid.

FC Layer: Fully Connected Layer.

G: Guanine.

GC: Guanine-Cytosine.

GRU: Gated Recurrent Unit.

Hg19: Human Genome 19.

KS: Kolmogorov-Smirnov.

LeakyReLU: Leaky Rectified Linear Unit.

LLM: Large Language Model.

LM: Language Model.

MFE: Minimum Free Energy.

MLM: Masked Language Modelling.

mRNA: Messenger Ribonucleic Acid.

NCBI: National Center for Biotechnology Information.

NER: Named Entity Recognition.

NLP: Natural Language Processing.

nt: Nulceotides.

ORF: Open Reading Frame.

PEFT: Parameter Efficient Fine Tuning.

PLM: Pre-trained Language Model.

POS: Part-of-Speech.

RNA: Ribonucleic Acid.

RNN: Recurrent Neural Network.

RSCU: Relative Synonymous Codon Usage.

SARS-CoV-2: Severe Acute Respiratory Syndrome Coronavirus 2.

SFT: Supervised Fine Tuning.

T: Thymine.

tRNA: Transfer Ribonucleic acid.

U: Uracil.

UCSC: University of California, Santa Cruz.

UTR: Untranslated Region.

Chapter 1

Introduction

This thesis's research focus lies at the intersection between bioinformatics and deep learning, specifically in optimizing mRNA sequences for vaccine development. In order to lay down the motivation and objective of the thesis, we first introduce the necessary biological background in the following section.

1.1 Background

1.1.1 Central Dogma of Biology

Deoxyribonucleic acid (DNA) is the hereditary material in all living organisms, encoding the instructions necessary for cellular function and development. A gene is a segment of DNA that encodes the information required to synthesize a functional protein. DNA is composed of two strands forming a double helix, with each strand consisting of a sequence of four nucleotides: adenine (A), thymine (T), cytosine (C), and guanine (G). The sequence of these nucleotides encodes genetic information that is translated into proteins, which are the functional molecules responsible for biological processes. The process of translating genetic information from DNA to protein involves an intermediary step known as transcription, during which a messenger ribonucleic acid (mRNA) sequence is synthesized. Unlike DNA, mRNA is single-stranded and uses uracil (U) instead of T. A sample mRNA sequence such as **'AUGUUCGUGUUCCUGGUGCUGCUG...'**, is composed of several regions, including the 5' untranslated region (5' UTR), the open reading frame (ORF), and the 3' untranslated region (3' UTR). The ORF is a critical component of the mRNA sequence as it encodes the information necessary for protein synthesis [30]. The ORF is organized into codons, which are sequences of three adjacent nucleotides. For example, the aforementioned mRNA sequence can be expressed as a sequence of codons: **'{AUG}{UUC}{GUG}{UUC}{CUG}{GUG}{CUG}{CUG}..'** There are 64 possible codons, 61 of which code for specific amino acids, while three serve as stop signals to terminate the translation process. Translation, the process of protein synthesis, occurs in the ribosome, beginning at the start codon within the ORF and proceeding codon by codon until a stop codon is encountered. Each codon specifies an amino acid that is carried by transfer RNA (tRNA) to be added to the growing polypeptide chain. Once the entire ORF has been translated, the sequence of amino acids folds into a functional protein.

1.1.2 Codon Optimization

The existence of 64 possible codons but only 20 amino acids leads to codon degeneracy, where multiple codons can encode the same amino acid. For example, the amino acid arginine (Arg) can be encoded by six different codons. Table 1.1 lists all synonymous codons and the specific amino acids they encode. Codon degeneracy creates the need for codon optimization, which is the process of selecting the most efficient synonymous codons for a given amino acid in the target protein. Codon optimization is crucial because different organisms exhibit codon usage bias, a preference for certain codons over others when encoding the same amino acid. This bias is influenced by the availability of tRNAs, which carry specific amino acids to the ribosome during translation [17].

Table 1.1: Codon table describing grouping of synonymous codons for different functional amino acids.

Amino Acid	**Synonymous Codons**
Alanine (Ala)	GCT, GCC, GCA, GCG
Arginine (Arg)	CGT, CGC, CGA, CGG, AGA, AGG
Asparagine (Asn)	AAT, AAC
Aspartic Acid (Asp)	GAT, GAC
Cysteine (Cys)	TGT, TGC
Glutamic Acid (Glu)	GAA, GAG
Glutamine (Gln)	CAA, CAG
Glycine (Gly)	GGT, GGC, GGA, GGG
Histidine (His)	CAT, CAC
Isoleucine (Ile)	ATT, ATC, ATA
Leucine (Leu)	CTT, CTC, CTA, CTG, TTA, TTG
Lysine (Lys)	AAA, AAG
Methionine (Met)	ATG
Phenylalanine (Phe)	TTT, TTC
Proline (Pro)	CCT, CCC, CCA, CCG
Serine (Ser)	TCT, TCC, TCA, TCG, AGT, AGC
Threonine (Thr)	ACT, ACC, ACA, ACG
Tryptophan (Trp)	TGG
Tyrosine (Tyr)	TAT, TAC
Valine (Val)	GTT, GTC, GTA, GTG

1.1.3 ORF Expression and Stability

Gene expression, in the context of this thesis, refers to ORF expression, which is the process by which information from a gene is used to synthesize a functional protein [4]. The yield of a functional protein product is a key measure of ORF expression efficiency.

The stability of mRNA, which impacts how long the mRNA remains intact within a cell, is another critical factor affecting the duration and efficiency of protein synthesis. Two key indicators of mRNA stability are minimum free energy (MFE) and GC content. MFE measures the thermodynamic stability of the mRNA secondary structure, with lower MFE values indicating more stable structures that are less susceptible to degradation. GC content, the percentage of guanine and cytosine nucleotides in the mRNA sequence, also contributes to stability, as GC pairs form stronger triple bonds than AT pairs [31].

1.1.4 SARS-CoV-2 Virus and mRNA Vaccines

The Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) was responsible for the COVID-19 pandemic. It primarily infects human cells by binding to cell surface receptors via its spike protein [3]. The urgent need for effective vaccines led to the rapid development of mRNA vaccines, which represent a novel approach to immunization against SARS-CoV-2. These vaccines contain mRNA sequences that encode the SARS-CoV-2 spike protein. Once administered, the mRNA is taken up by human cells and translated into the spike protein, which is then recognized by the immune system. This recognition prompts the production of antibodies and activation of T-cells, thereby preparing the body to fight the actual virus upon exposure. Fig. 1.1 illustrates the mechanism of mRNA vaccines in humans. In mRNA vaccine design, optimizing codon usage can enhance translation efficiency, stability, and overall expression of the target protein in human cells. This optimization is vital for ensuring that the mRNA produces a high yield of the target protein (antigen) to elicit a strong immune response.

1.1.5 Wild-Type Sequences

Wild-type sequences refer to the naturally occurring genetic sequences in an organism, representing the standard or reference form of a gene as it occurs in nature. In mRNA

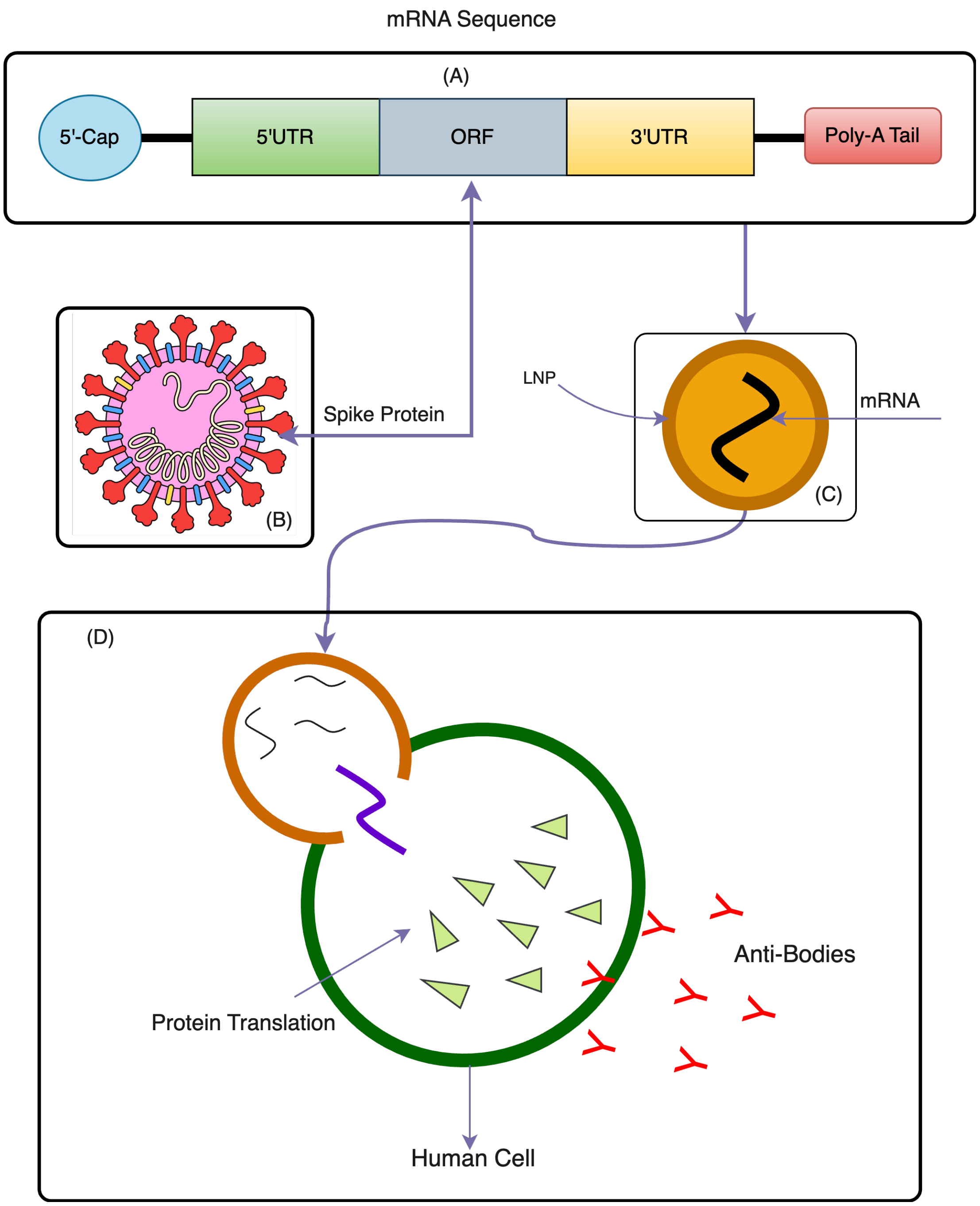

Figure 1.1: The design and working mechanism of mRNA-vaccine.

vaccine design, wild-type sequences of viral proteins, such as the SARS-CoV-2 spike protein, are often used as starting points for vaccine development. However, these wild-type sequences may not be optimized for expression in human cells, necessitating codon optimization.

1.1.6 Codon Adaptation Index (CAI)

The CAI is a quantitative measure used to predict the potential level of protein expression based on the codon usage bias within a given coding sequence. CAI as an empirical index has previously been used in various studies as a measure for protein expression [49]. Empirical measurement of protein expression potential is quantitatively assessed through the CAI, which hinges on the codon usage bias within a coding sequence. Initially, the calculation of Relative Synonymous Codon Usage (RSCU) values is required. RSCU evaluates the frequency of a specific codon relative to the average frequency of all synonymous codons encoding the same amino acid, thereby normalizing the number of synonymous codons. Mathematically, the RSCU for a codon j that encodes for amino acid a is defined as:

$$RSCU_{a,j} = \frac{X_{j,a}}{\frac{1}{n_a}\sum_{j=1}^{n_a} X_{a,j}} \tag{1.1}$$

Here $X_{j,a}$ represents the observed frequency of the j^{th} codon, n_a denotes the number of synonymous codons for the amino acid a, and the denominator corresponds to the average frequency of all synonymous codons for that amino acid. Using the $RSCU$, for each codon, its relative adaptiveness ($w_{a,j}$) is measured to determine the relative frequency of a certain codon over the optimal codon for a particular amino acid.

$$w_{a,j} = \frac{RSCU_{a,j}}{RSCU_{a,max}} \tag{1.2}$$

In Eq. 1.2, $RSCU_{a,max}$ determines the frequency usage of the optimal codon for amino acid a. Finally, the CAI is computed as the geometric mean of the relative adaptiveness values of each codon in the RNA/DNA sequences of length L. The CAI formula is expressed in Eq. 1.3.

$$CAI = \left(\prod_{k=1}^{L} w_k \right)^{\frac{1}{L}} \tag{1.3}$$

The CAI is instrumental as an index for expression, encapsulating the bias towards codons presumed to be translated with higher efficiency. Consequently, sequences with elevated CAI values are perceived to be optimized for enhanced expression within a specific host organism. This renders the CAI an invaluable quantitative metric for genetic engineering aimed at maximizing protein production or elucidating the determinants of natural protein expression levels.

1.1.7 Language Models (LMs)

Language models (LMs) are a fundamental component of natural language processing (NLP) that predict the probability distribution of a sequence of words. They form the basis for various applications, including machine translation, text generation, and speech recognition. The primary goal of a language model is to estimate the probability of a sequence of words. Given a sequence of words $w_1, w_2, \ldots, w_n$, the probability $P(w_1, w_2, \ldots, w_n)$ is decomposed using the chain rule of probability:

$$P(w_1, w_2, \ldots, w_n) = P(w_1) \times P(w_2 \mid w_1) \times \cdots \times P(w_n \mid w_1, w_2, \ldots, w_{n-1})$$

Language models aim to predict each word w_t in a sequence based on the preceding words $w_1, w_2, \ldots, w_{t-1}$.

1.2 Thesis Motivation

The rapid spread of SARS-CoV-2 in 2019 underscored the urgent need for effective vaccines, driving a global effort to accelerate vaccine development. Among the various vaccine platforms, mRNA vaccines emerged as particularly promising due to their rapid design and production capabilities, as well as their demonstrated efficacy in eliciting strong immune responses [48, 53]. The successful deployment of mRNA vaccines, such as Pfizer-BioNTech's BNT162b2 and Moderna's mRNA-1273, has had a significant impact on public health, providing a critical tool against the COVID-19 pandemic [30].

1.2.1 Challenges in mRNA Vaccine Design

Designing mRNA vaccines presents several significant challenges. One primary obstacle is the combinatorially large ORF candidate space resulting from codon degeneracy, where multiple synonymous codons can encode the same amino acid. This degeneracy exponentially increases the number of possible ORF sequences that could encode the same target protein, making it infeasible to explore all potential sequences manually or through brute-force computational methods [30]. For instance, optimizing the ORF sequence for the spike protein of SARS-CoV-2 involves navigating a search space with a cardinality greater than 10^{632} [30].

In addition to codon selection, ORF stability is a critical factor in vaccine design. The ORF sequence must be stable enough to avoid degradation before it can be translated into the target protein within human cells. This requires careful optimization of the ORF's structural features, such as GC content and secondary structure, to balance stability with efficient translation [33].

1.2.2 Necessity of Codon Optimization

Given the challenges of mRNA vaccine design, codon optimization is an essential step. Codon optimization involves selecting the optimal sequence of codons to maximize translation efficiency, expression, and stability. This process is complicated by the need to account for codon usage bias, which varies between organisms and influences the efficiency with which different codons are translated into proteins [2, 45]. Codon bias is correlated with the availability of tRNA molecules in the cell, and optimizing codons to match the human tRNA pool can significantly enhance protein production. However, a simplistic approach to codon optimization—such as prioritizing only the most abundant tRNAs or the most preferred codon for each amino acid—can result in translation inefficiencies, premature translation termination, and protein misfolding [40].

1.2.3 Deep Learning in Genomics

Deep learning-based methods have consistently performed well in various domains, such as computer vision, natural language processing, healthcare, and genomics. In the context of biological sequences like ORFs and proteins, these models have shown great potential in capturing intricate sequence properties. Deep learning models, such as recurrent neural networks (RNNs) and transformer-based models, have advanced the understanding of complex sequential dependencies within biological sequences, making them well-suited for tasks like codon optimization [14, 57]. These models have outperformed traditional methods that rely on codon sampling from the codon distribution in the host, which focuses on maximizing the CAI index [46]. Additionally, mixed linear integer programming and graphical representations of primary protein sequences have been used [29, 47], but these methods struggle to capture long-range dependencies between amino acids in the protein sequence.

Beyond codon optimization, large language models (LLMs) have been developed by training on vast corpora of DNA, RNA, and protein sequences [7, 13, 15, 28, 34, 42]. These LLMs demonstrate the potential to revolutionize genomics by providing new insights into sequence functionality and optimization.

1.2.4 Thesis Objective

In this thesis, we experimented with different sequential deep learning models, such as long short-term memory (LSTM) and transformers, to learn the mapping of the ORF sequence optimized for humans for a given target protein sequence. The focus is on increasing the expression and stability of the optimized ORF sequence for humans against the SARS-Cov-2 spike protein, which is the target protein antigen.

1.2.5 Thesis Contribution

1. In our first contribution, we introduce a novel strategy of 'valid-codon' that uses a codon mask to enforce the correct mapping of codons to their specific amino acids, reducing the model's label space and simplifying the learning process. We investigate the 'valid-codon' method against 'codon-box' by performing statistical significance tests over its optimized ORF sequences across different organisms for expression.

2. In our second contribution, we explore the application of protein pre-trained language models (PLMs) for codon optimization, guided by the hypothesis that these models capture rich contextual information about amino acid interactions and general properties of protein sequences, which can be leveraged for effective codon optimization.

3. In our third contribution, we focused on increasing the stability of the optimized ORF along with the expression by curating a dataset of stable sequences for training the model. The stable sequences are curated by filtering sequences based on their MFE values.

1.3 Thesis Outline

The rest of the thesis is divided into the following chapters:

1. Chapter 2 reviews the relevant literature on codon optimization, mRNA vaccine design, and the application of deep learning in genomics.

2. Chapter 3 entails the material and methods used in the work. Sec. 3.1 and 3.2 explain the data collection and pre-processing steps involved. Further sections describe the model and metrics. In the last Sec. 3.4, the formulation of the codon optimization task as a natural language processing task is described.

3. Chapter 4 describes the first contribution of ‘valid-codon’ method and Sec. 4.1 gives insights of the same by performing a comparative analysis against other methods.

4. Chapter 5 discusses the utilization of pre-trained protein LLM for optimizing ORF sequences for humans against the SARS-CoV-2 spike protein. Independent testing is performed against the industry-approved vaccines optimized ORF on humans.

5. Chapter 6 explores new ideation and challenges for optimizing ORF for increasing the expression by modifying the objective function. It further looks into the avenues for generating mRNAs with user-tunable protein expression and stability.

6. Chapter 7 concludes the thesis with a summary of the results and plausible future improvements.

Chapter 2

Literature Review

The degeneracy of codons, wherein multiple codons encode the same amino acid, leads to the existence of multiple ORF sequences that can produce the same target protein. This degeneracy presents a challenge in identifying the optimal ORF sequence from a large pool of synonymous sequences, particularly for applications in gene therapy and synthetic biology. Optimizing these sequences is crucial as gene expression in a host organism is influenced by various factors, with codon usage bias being a critical one. Codon usage bias refers to the preference for certain codons over others in a given organism, which can significantly impact the efficiency of protein synthesis. Several tools and algorithms have been developed to optimize ORF sequences based on synonymous codon usage patterns, but recent advances in artificial intelligence, particularly deep learning, have brought new capabilities to the design of synthetic genes. In many cases, deep learning models have outperformed traditional algorithms in optimizing gene expression. In this chapter, we review the relevant literature on synthetic sequence design for ORFs, focusing on studies that utilize deep learning methods for codon optimization.

2.1 Deep Learning for Codon Optimization

In the study by Goulet et al. [16], a Recurrent Neural Network (RNN) was trained on 30,000 DNA sequences from Chinese Hamster Ovary (CHO) cells for the purpose of codon optimization. The model was tested on a dataset of 8,000 amino acid sequences, achieving an accuracy of 52.78%. This accuracy metric was defined by the model's ability to select codons that match those found in native DNA sequences. However, it is important to note that accuracy in this context does not directly account for codon usage bias or expression levels, which are crucial for optimizing protein production. Despite this, the model's performance significantly surpassed the "most abundant codon" approach. The RNN-optimized sequences were validated by comparing their properties with reference datasets, test set predictions, and the PD-L1 (CD274) DNA sequence. When transiently expressed, the PD-L1 sequence inferred by the RNN produced protein levels comparable to those achieved with conventionally optimized sequences. Moreover, the RNN-optimized sequence showed higher protein titer expression than both the ground truth sequence and the sequence optimized by the IDT tool. These results suggest that the trained RNN model can be generally useful for codon optimization in CHO cells, potentially outperforming traditional non-learning algorithms like GeneArt, GenScript, and Top Codon. The GC content of sequences generated by the RNN model (54%) was similar to those produced by GeneArt (56%) and GenScript (55%), highlighting the model's ability to implicitly learn important sequence features like GC content without explicit parameterization.

In another study, ICOR [26] introduced a codon optimization tool built on a bi-directional Long Short-Term Memory (bi-LSTM) network aimed at improving the heterologous expression of synthetic genes. Unlike traditional methods, ICOR focuses on learning the codon usage patterns and context-specific to the host genome. The model was trained on 7,406 *E.coli* genes extracted from NCBI, with CD-HIT-EST [23] used to filter out similar nucleotide sequences. Experiments were conducted using both one-hot encoding and Non-Linear Fisher Transform encoding of the amino acids in the input protein sequence. ICOR was compared against five different methods—ERC, GenScript, BFC, URC, and HFC on metrics such as CAI, GC content, cis-regulatory elements, codon-frequency distribution, and negative repeat elements.

ICOR demonstrated a 41.26% increase in CAI over native sequences, with a mean CAI of 0.9, representing a statistically significant improvement over other methods. The GC content of ICOR-optimized sequences fell within the ideal range of 30-70%, further validating the model's effectiveness. When evaluated on 40 benchmark sequences, ICOR performed comparably to or better than the other six methods in terms of codon frequency distribution, negative repeat elements, and cis-regulatory elements. This study highlights the potential of more sophisticated deep learning models, like bi-LSTM, for effective codon optimization in *E.coli*, contributing to cost-effective and productive recombinant protein production.

Fu et al. [14] introduced a novel approach to codon optimization using 'codon-box' encoding, which groups codons with identical nucleotide composition regardless of their order. This method was inspired by sequence annotation tasks in natural language processing (NLP) and involved training a BiLSTM-CRF model with and without codon box encoding on *E.coli* genes. The results showed negligible differences between the training and test accuracies of models with and without codon box encoding, with both achieving a training accuracy of 0.77 and a test accuracy of 0.52. However, the 25% difference between training and test accuracy suggests potential overfitting and a lack of robustness. The study used CAI as the main evaluation metric instead of accuracy as the primary evaluation metric. The native sequence's CAI was compared to those optimized by Genwiz, ThermoFisher, BiLSTM-CRF with codon box encoding (A), and BiLSTM-CRF without codon box encoding (B). The mean CAI achieved by model A was 0.98 across six independent test sequences, outperforming Genwiz, ThermoFisher, and the native sequence. However, it is important to note that an excessively high CAI could lead to an imbalance in tRNA availability, potentially reducing translation efficacy. The 'codon-box' encoding method resulted in optimized sequences that differed by 20-28% from those optimized by ThermoFisher, Genwiz, and model B. Experimental validation of protein expression was carried out by optimizing codon sequences for the FALVAC-1 and PTP4A3 proteins, showing improved protein expression and activity for both.

Gong et al. [15] introduced iDRO, a two-stage process for generating both ORF sequences and UTRs in mRNA design. This approach was unprecedented at the time and used highly expressive human genes from the Hg19 dataset for training, which forms the basis for the work presented in this thesis. In the first stage, ORF optimization was conducted using the same methodology as the 'codon-box' encoding with the BiLSTM-CRF model [14]. Although the study did not report CAI or other related metrics for ORF optimization, the second stage involved generating 5' and 3' UTRs using RNA-BART, an auto-regressive large language model. The UTR generation was treated as a machine translation task, with the input being the optimized ORF sequence from stage one. RNA-BART was pre-trained by masking 15% of the tokens in UTR and ORF sequences, learning dense vector representations to capture bidirectional context. The model was then fine-tuned for UTR generation. The 5' and 3' UTRs generated by iDRO were experimentally validated using the EGFP gene and compared with Pfizer and Moderna's approved vaccines, BNT162b2 and mRNA-1273, respectively. The iDRO-generated UTRs demonstrated higher protein expression and more micro-RNA binding sites. Additionally, the generated mRNAs exhibited lower minimum free energy (MFE), indicating structurally stable sequences. This work presented an integrated approach to ORF and UTR design, resulting in improved mRNA stability and expression.

Several other studies [1, 5, 6, 27] have employed deep generative models, such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and auto-regressive LLMs, for UTR generation as well. The encapsulation of ORF sequences for transfection (administering the sequence inside the human cell) remains a significant challenge in mRNA vaccine design [18, 37, 55]. Overall, machine learning and deep learning approaches are critical for mRNA vaccine design due to their ability to navigate the large combinatorial space of sequences while capturing subtle biological intricacies. These methods can be productively applied in gene therapy, enhancing both computational biological indices and in-vitro transfection outcomes.

Chapter 3

Material and Methods

3.1 Data Collection

In this work, the dataset comprises ORF sequences consisting of codons and their translated primary protein sequences consisting of amino acids. Datasets were collected for three organisms: Human Hg19 gene, *E.coli*, and Chinese-Hamster (CH). The *E.coli* data was collected from [14], containing a total of 5447 pairs of ORF and protein sequences. The CH data, collected from [16], included 38000 pairs of ORF and protein sequences. The collection of ORF and primary protein sequences for Hg19 genes was a comprehensive, multi-stage effort. The Hg19 dataset, also known as the GRCh37 reference genome, is a widely used reference in human genomics. It represents the human genome assembly that was released by the Genome Reference Consortium in 2009 and has been extensively used in research for its high-quality sequence data. Hg19 is particularly valuable for training because it provides a robust representation of gene sequences that are crucial for understanding genetic variations and their implications in human biology. The Hg19 genes were collected from UCSC [1] and consisted of a total of 18213 pairs. After collection, the UCSC ID of each gene was mapped to NCBI IDs using bioDBnet:db2db tool[2]. Using NCBI IDs for each Hg19 gene, ORF sequences were extracted from their complete nucleotide reference

[1]https://genome.ucsc.edu/cgi-bin/hgTables
[2]https://biodbnet-abcc.ncifcrf.gov/db/db2db.php

sequence. NCBI ORFfinder API [3] was used to extract ORF sequences for each gene. While using the ORFfinder API, the corner case of handling negative strands was addressed by complementing and reversing the reference nucleotide sequence.

Two types of datasets were curated for Hg19 genes: the first one containing random length ORFs and their corresponding amino acid translations and the second containing short ORFs for a given gene. The former dataset is referred to as 'Hg19_random' while the latter is 'Hg19_short'. Hg19_random contained a total of 18213 pairs of ORF and primary protein sequences, while the Hg19_short contained 419455 pairs of such sequences.

3.2 Data Pre-Processing

The sequences collected in each dataset underwent a series of filter tests and preprocessing steps. The filter tests were designed to eliminate biologically incorrect sequences. Subsequently, preprocessing steps like tokenization, numerical encoding, and padding were performed to prepare the sequences for deep learning models.

3.2.1 Filter Test

For each pair of sequences, the following seven tests were conducted:

- Test 1: This test verifies if the length of the ORF sequence is multiple of three. Since each amino acid maps to triplets of nucleotides known as codons, the ORF sequence length should be a multiple of three.

- Test 2: This test checks the biological correctness of the nucleotide sequence, ensuring that only valid nucleotide bases—A (Adenine), T (Thymine), G (Guanine), and C (Cytosine)—are present.

- Test 3: This test verifies that the primary protein sequence and ORF sequence start with the correct amino acid and codon, respectively. The start codon in

[3]https://www.ncbi.nlm.nih.gov/orffinder/

a valid ORF sequence is ATG, coding for Methionine (Met).

- Test 4: Similar to Test 3, this test checks that the ORF sequence ends with one of the valid stop codons (TAA, TAG, TGA).
- Test 5: This test ensures that each codon in the ORF sequence correctly codes for the corresponding amino acid in the primary protein sequence, maintaining the correct biological mapping between codons and amino acids as described in Table 1.1.
- Test 6: This test verifies that the protein sequence contains only valid amino acid characters, ensuring that all 20 standard amino acids are present.
- Test 7: This test checks that the length of the primary protein sequence is consistent with the ORF sequence length, ensuring a one-to-one correspondence between codons and amino acids.

Table 3.1 and Table 3.2 summarize the results of the filter tests for the Hg19_random and Hg19_short datasets, respectively. In the Hg19_random dataset, 45 out of 18,213 pairs of sequences failed one of the seven tests. In the Hg19_short dataset, 13,705 out of 419,455 pairs failed one of the seven tests. Notably, all sequences in the E. coli and Chinese hamster datasets passed the tests.

Table 3.1: Hg19_random filter test results

Test	1	2	3	4	5	6	7
Sequences Failed	0	0	29	0	0	17	0

Table 3.2: Hg19_short filter test results

Tests	1	2	3	4	5	6	7
No. Sequence Failed	0	0	126	13696	130	0	13581

3.2.2 Tokenization and Padding

After filtering, the sequences underwent tokenization and padding.

- **Tokenization:** Tokenization breaks down complex biological sequences into simpler, more manageable units called tokens. For protein sequence, each amino acid is treated as a word analogous to a natural language. The corresponding ORF sequence, composed of codons, codon is equivalent to a word in natural language. Hence, for an ORF sequence of length Nx3, the total number of words (codons) will be equal to N. Thus, the dimensionality of the ORF sequence reduces from Nx3 to N.

- **Numerical Encoding:**

  Deep learning models require numerical inputs. In this work, protein sequences and ORF sequences were converted into numerical formats using integer encoding. Each of the 20 standard amino acids was mapped to an integer from 1 to 20, while the 61 valid codons were assigned integers from 1 to 61. The integer 0 was reserved as a padding token for the next step.

- **Padding:** Sequences in the dataset vary in length. To process these sequences in deep learning models, they must be of uniform size. Padding involves adding a specific value (usually 0) to the end of shorter sequences, standardizing their lengths to match the longest sequence in each batch. In this work, sequences were padded on the right. The **pad_sequence** function from the **torch.nn.utils.rnn** module in PyTorch was used to dynamically pad sequences within each batch, optimizing memory usage by avoiding unnecessary padding.

3.3 Model

3.3.1 Embedding Layer

Embedding layers are a fundamental component in neural networks, especially in NLP. They serve as a bridge between the raw input data and their numerical representations required for processing by neural networks. In the context of language processing, each word in the vocabulary is represented as a unique integer, and this integer is then mapped to a dense vector within the embedding space. The dense vector captures more information about the word, including its relationship to other words, in a more compact form than the sparse one-hot encoding.

In this work, the embedding dimension was set to 61, representing the 61 unique codons used, while the vocabulary size was set to 21, representing the 20 standard amino acids plus a padding token.

3.3.2 Long Short Term Memory (LSTM)

For sequential data like sentences (here protein and ORF sequence) and time series, LSTMs tend to perform better as they are capable of capturing sequential context. LSTMs are an improvised version of RNNs as they tend to solve the bottleneck of capturing long-term dependencies, also known as the vanishing gradient problem in RNNs, to a significant extent [21]. The concept of gates was introduced in LSTMs to regulate the information flowing through each step. Inputs to a single unit of an LSTM network are the previous time step cell state (C_{t-1}), hidden state (H_{t-1}), and current time step input (X_t). The outputs from a single unit are the current time step cell state (C_t), and hidden state (H_t). At each time step, depending upon the task like classification or regression, H_t is wrapped up through suitable dense layers for the prediction task. A single unit of the LSTM network involves the following four gates:

1. **Forget gate:** The forget gate decides what information should be discarded

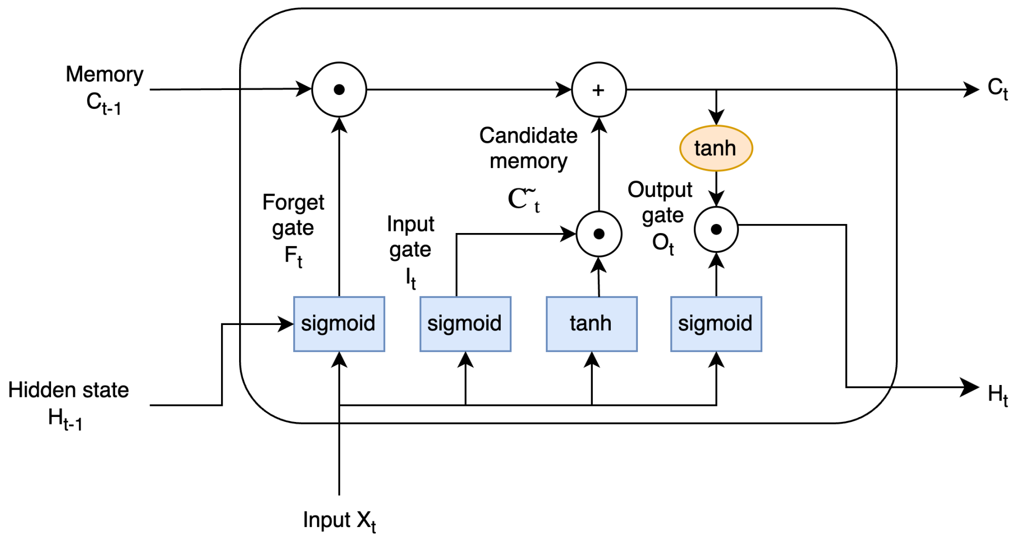

Figure 3.1: A single unit of LSTM network with the description of different gates.

from the cell state. It takes the previous hidden state H_{t-1} and the current time step input X_t.

$$F_t = \sigma(W_f \cdot [H_{t-1}, X_t] + b_f) \tag{3.1}$$

Here σ represents the sigmoid function. W_f is the weight matrix and b_f is the bias. The sigmoid function outputs values between 0 and 1, indicating how much of each component of the cell state should be retained.

2. **Input gate:** The input gate decides what new information will be stored in the cell state. It involves two parts: a sigmoid layer and a tanh layer. The sigmoid layer decides which values to update, and the tanh layer creates a vector of new candidate value $\tilde{C}_t$, that could be added to the state.

$$I_t = \sigma(W_i \cdot [H_{t-1}, X_t] + b_i) \tag{3.2}$$

$$\tilde{C}_t = \tanh(W_C \cdot [H_{t-1}, X_t] + b_C) \tag{3.3}$$

Here, I_t, is the output of the sigmoid gate, and W_i, b_i, W_c, b_c are the corresponding weights and biases for input gate and new cell state operations, respectively.

3. **Cell state update gate:** The new cell state is updated by adding what is to be forgotten from the previous cell state (F_t) and what new information is to be added from the I_t.

$$C_t = F_t * C_{t-1} + I_t * \tilde{C}_t \tag{3.4}$$

   Here, '*' and '+' are element-wise multiplication and additive operations.

4. **Output gate:** Finally, the output gate decides what the next hidden state should be. The hidden state contains information about the previous inputs and is used for predictions. The output gate first decides which parts of the cell state to output, and then a tanh function is applied to the cell state (to normalize the values between -1 and 1), and it's multiplied by the output of the sigmoid gate.

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.5}$$

$$h_t = O_t * \tanh(C_t) \tag{3.6}$$

   Here, o_t is the output of the sigmoid gate applied to the current input and the previous hidden state, and W_o and b_o are the corresponding weights and biases learned.

3.3.3 Dense Layer

Dense layers, also known as fully connected layers, play a critical role in processing and learning from data. Each neuron in a dense layer is connected to every neuron in the previous layer, with weights and biases determining the output. The output is then passed through an activation function, introducing non-linearity to capture complex patterns.

$$Y = f(W.X + b) \tag{3.7}$$

Here, f is an activation function, and X is the input to the layer.

3.3.4 Regularization Methods

Regularization techniques are employed to prevent overfitting in a neural network. Two popular methods used in this work are Dropout and Batch Normalization.

- **Dropout:** In this regularization technique, randomly selected neurons are ignored during training, i.e., they do not contribute to downstream neurons temporarily during the forward pass. Dropout helps prevent co-adaptation [50] between upstream and current layer neurons.

- **Batch Normalization:** Batch normalization standardizes the output of a layer for each mini-batch, alleviating internal covariate shift and improving model training [25].

3.3.5 Dense bi-LSTM

The baseline model used to train sequences for codon optimization was the 'Dense bi-LSTM' network. This model includes an embedding layer (Sec. 3.3.1) to transform the input into fixed-size dense vectors that allow the model to capture the semantics of the input sequences in a high-dimensional space. A bi-directional LSTM network was employed to capture dependencies in both forward and backward directions. For the bi-directional nature, two LSTM (Sec. 3.3.2) networks are stacked on top of each other. One of them processes the input sequence from left to right, while the other one processes the input sequence from right to left. The hidden layers' outputs are stacked, giving the final output from the bi-LSTM network. The dense component of the 'Dense bi-LSTM' network comprises of fully connected layers (FC layers) discussed in Sec. 3.3.3. Four FC layers are used after the bi-LSTM networks, which map the bi-LSTM network representations to a desired output space. Dropout and batch normalization (Sec. 3.3.4) were applied after each fully connected layer, except the final one, to prevent overfitting.

Activation Function:

Leaky Rectified Linear Unit (LeakyReLU) [36] was used as an activation function for introducing non-linearity after each FC layer, except the last one. LeakyReLU addresses the ”dying ReLU” problem, where neurons stop learning due to zero gradients for negative inputs [11]. The LeakyReLU function is defined in Eq. 3.8, where x is the input to the neuron, f(x) is the output, and α is a small constant close to 0, which does not allow the gradients to become 0. In this work, 0.1 was taken as the default value of α.

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ \alpha x & \text{otherwise.} \end{cases} \tag{3.8}$$

The gradient of the LeakyReLU function is defined in Eq. 3.9

$$f'(x) = \begin{cases} 1 & \text{if } x > 0, \\ \alpha & \text{otherwise.} \end{cases} \tag{3.9}$$

3.4 Proposed Methodology

The core problem addressed in this work is finding the optimal ORF sequence composed of codons, given a target protein sequence. Due to the degeneracy of codons, multiple codon sequences can encode the same protein, making it essential to incorporate codon bias rules found in the host cellular environment. In this work, the problem is approached as an NLP task due to the textual nature of ORF and protein sequences. Specifically, codon optimization of ORF is treated as a sequence tagging task, where the goal is to classify or tag the optimal codon against each amino acid in the target protein sequence. Sequence tagging task in the field of NLP is to label components of a sequence, such as words in a sentence. For example, in a natural language, each word can be classified into different categories like person, place, etc. Here, amino acids are treated as words in the protein sequence, and the task is to label each amino acid with the optimal codon. Fig. 3.2 illustrates the idea. The

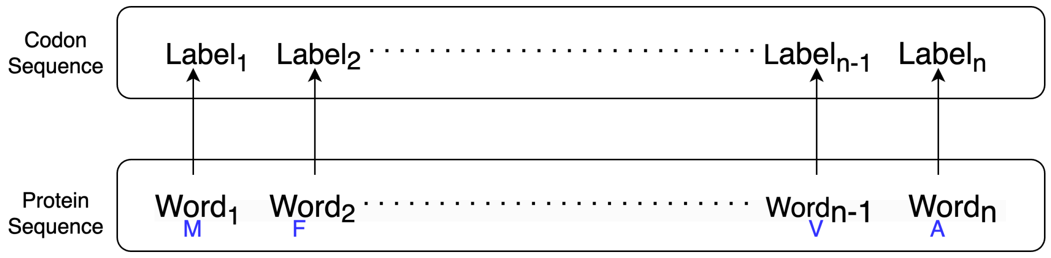

Figure 3.2: Sequence Labelling methodology

sequence tagging approach is further discussed in Chapters 4 and 5, along with the experimental results.

Chapter 4

Comparative Analysis between Codon-Box, Valid-Codon, All-Codon Method of Training

In this chapter, we investigate three different methods of training for codon optimization [Sec. 1.1.2]. Before going to the experiment the next section provides a background of the three methods.

4.1 Background of Methods

4.1.1 All-Codon

The 'all-codon'- method treats codon optimization as a multi-label classification problem where the model selects one out of 61 possible codons (excluding stop codons) for each amino acid in the input protein sequence. This method does not constrain the label space, as it must distinguish between all possible codons at each step.

4.1.2 Codon-Box

The 'codon-box' method, introduced by Fu et al. [14], involves grouping codons based on their nucleotide composition rather irrespective of their order or the specific amino acid they encode. This method reduces the label space from 61 to 20 classes by grouping codons that share the same nucleotide composition into "boxes". Table 4.1 describes all the 20 codon boxes.

Table 4.1: Codon Box with their corresponding codons and amino acids

Key	**Codon Box**	**Codons**	**Amino Acid**
j	AGT	AGT, ATG, GAT, GTA	Ser, Met, Asp, Val
g	CGT	CTG, TCG, TGC, CGT, GTC, GCT	Arg, Ser, Cys, Ala
k	GGT	GGT, TGG, GTG	Gly, Trp, Val
b	CTT	CTT, TCT, TTC	Leu, Ser, Phe
d	GTT	GTT, TTG, TGT	Val, Leu, Cys
s	GCG	GCG, GGC, CGG	Ala, Gly, Arg
q	AGG	GAG, GGA, AGG	Glu, Gly, Arg
r	ACG	GCA, CGA, AGC, GAC, ACG, CAG	Ala, Arg, Ser, Asp
n	CCG	GCC, CGC, CCG	Ala, Arg, Pro
o	AAC	CAA, ACA, AAC	Gln, Thr, Asn
m	CAC	CAC, ACC, CCA	His, Thr, Pro
f	CCT	TCC, CTC, CCT	Ser, Leu, Pro
u	AAG	GAA, AAG, AGA	Glu, Lys, Arg
h	ACT	CAT, ACT, TCA, TAC, ATC, CTA	His, Thr, Ser, Tyr, Ile, Leu
p	GGG	GGG	Gly
l	CCC	CCC	Pro
t	AAA	AAA	Lys
c	ATT	TAT, ATT, TTA	Tyr, Ile, Leu
a	TTT	TTT	Phe
i	ATA	ATA, AAT	Ile, Asn

4.1.3 Valid-Codon

The 'valid-codon' method proposed in this thesis is built on the similar premise as 'codon-box' for reducing the label space. It reduces the label space by restricting the model to focus exclusively on synonymous codons at each step of the protein sequence. This method applies a codon mask in the final classification layer (Eq. 4.1), which restricts the model to consider only the valid synonymous codons for each amino acid. The mask is a vector $CV \in \mathbb{R}^{61\times 1}$, where non-synonymous codons for a given amino acid are assigned a high negative value (-10^9).

$$masked\_logits = \sum_{t=0}^{L} output\_logits_t + CV_{t^{th}amino\_acid} \tag{4.1}$$

For e.g., in the case of Alanine (Ala), out of 61 indices for output codon classes, the ones that do not correspond to the indices of codons, namely, 'GCT', 'GCC', 'GCA' and 'GCG,' will have negative values assigned in CV.

4.2 Objective and Motivation

In this chapter, we investigate two key hypotheses:

1. The 'valid-codon' method, which reduces the label space by enforcing the model to focus only on synonymous codons, simplifies model learning when compared to the 'all-codon' method.

2. Grouping codons in a certain way, such as in the 'codon-box' method, does not provide a significant advantage in improving model performance on expression efficiency.

4.3 Experimental Setup

4.3.1 Data

The datasets used in this study include human Hg19 genes, *E.coli*, and Chinese-Hamster protein:ORF pair sequences. After pre-processing (Sec. 3.2), the datasets were split into training, validation, and test sets in a 0.6/0.2/0.2 ratio. The number of samples in each split is shown in Table 4.2. Datasets were converted to pytorch

Table 4.2: Train test split of sequences in Hg19_random, *E.coli* and Chinese-Hamster.

Dataset	**Train**	**Validation**	**Test**
Hg19_random	11584	2896	3620
E.coli	3485	872	1090
Chinese-Hamster	24320	6080	7600

dataloaders and saved in pickle format with a batch_size = 32.

4.3.2 Model and Training Setup

The experiments were conducted using the 'Dense bi-LSTM' network described in Sec. 3.3.5. The hyper-parameters of the model were tuned on a set of values. The experiment settings, i.e., hyper-parameters, seed, and model, across all the three different methods of training (Sec. 4.1) were kept identical. The embedding layer configurations had *num_embeddings* = 21 and *embedding_dim* = 64. The bi-LSTM module in network had an *input_size* = 64, *hidden_size* = 128, *num_layers* = 4 and *dropout* = 0.5. The 'Dense bi-LSTM' network was followed by four fully connected layers (FC layers). For the subsequent three FC layers, the input dimension was the output dimension of the previous layer. The second, third, and fourth FC layer's output dimensions were 256,128, and 61, respectively, and the input dimension was equal to the previous layer's output dimension.

The models were trained using the cross-entropy loss function (Eq. 4.2) for 30 epochs, with early stopping implemented to prevent overfitting.

$$L_{CEL} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\log(p_{i,t,y_{i,t}}) \tag{4.2}$$

$$p_{i,t,c} = \frac{\exp(z_{i,t,c})}{\sum_{k=1}^{C}\exp(z_{i,t,k})}$$

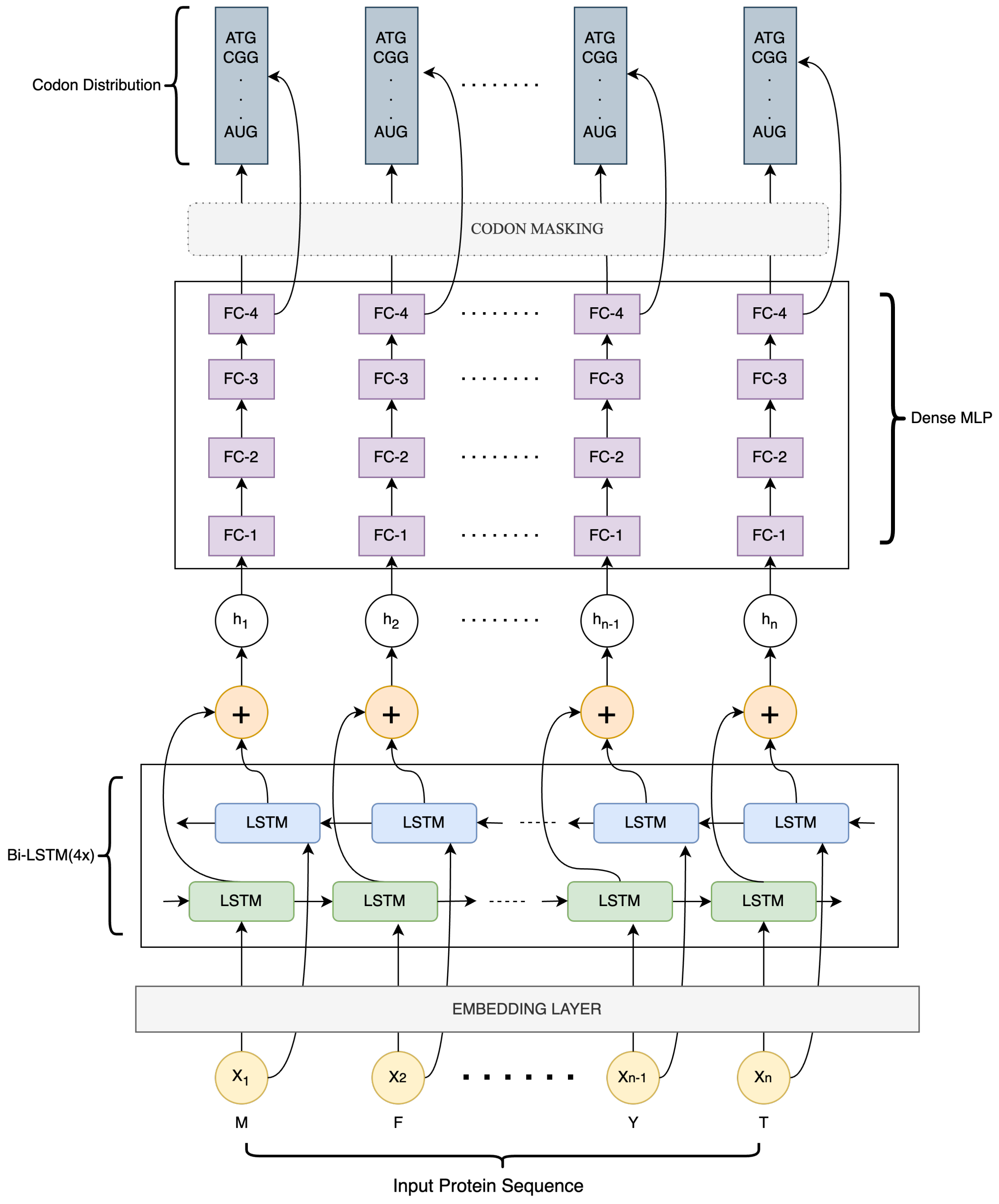

Figure 4.1: Dense bi-LSTM encoder architecture with masking

In Eq. 4.2, $z_{i,t,c}$ denotes the output logit for class c (one of the 61 codons),$p_{i,t,y_{i,t}}$ is the predicted probability of the true class $y_{i,t}$, for the i^{ith} sequence at timestep t. Mean reduction was used over the batch of $N = 32$ sequences.

4.3.3 Evaluation Criterion

The predicted ORF sequences were evaluated using the Codon Adaptation Index (CAI), as detailed in Sec. 1.1.6. The CAI measures the potential expression level of a protein based on the codon usage in the predicted sequences compared to native sequences.

4.4 Results and Discussion

The performance of the three methods—'all-codon', codon-box', and 'valid-codon'—was assessed based on the CAI values obtained from the test sets across all three datasets.

Table 4.3: Each of the three methods results trained on Hg19, *E.coli* and Chinese-Hamster datasets.

Organism	Method	Train Codon Match Accuracy	Validation Codon Match Accuracy	Test Codon Match Accuracy	CAI Test (mean)	CAI Test (mean) Basis ORF Sequences
Hg19_random	all-codon	0.538	0.539	0.548	0.87	0.77
	valid-codon	0.568	0.542	0.548	0.89	
	codon-box	0.558	0.544	0.539	0.90	
E.coli	all-codon	0.581	0.556	0.552	0.85	0.70
	valid-codon	0.591	0.558	0.553	0.85	
	codon-box	0.581	0.556	0.552	0.87	
Chinese-Hamster	all-codon	0.560	0.552	0.551	0.89	0.79
	valid-codon	0.569	0.552	0.558	0.90	
	codon-box	0.550	0.540	0.548	0.90	

On the Hg19_random test set, the 'all-codon' method improved the mean CAI from 0.77 to 0.87 ± 0.05, representing a 13.29 % improvement. The 'valid-codon' method further increased the CAI index to 0.89 ± 0.06, a 15.39 % increment. The 'codon-box' method achieved a negligibly higher CAI of 0.90 ± 0.04, a 17.11 % increase over the wild-type sequences, also known as basis ORF sequences in the dataset. The quantitative analysis of 'valid-codon' versus codon-box is described in Fig. 4.2.

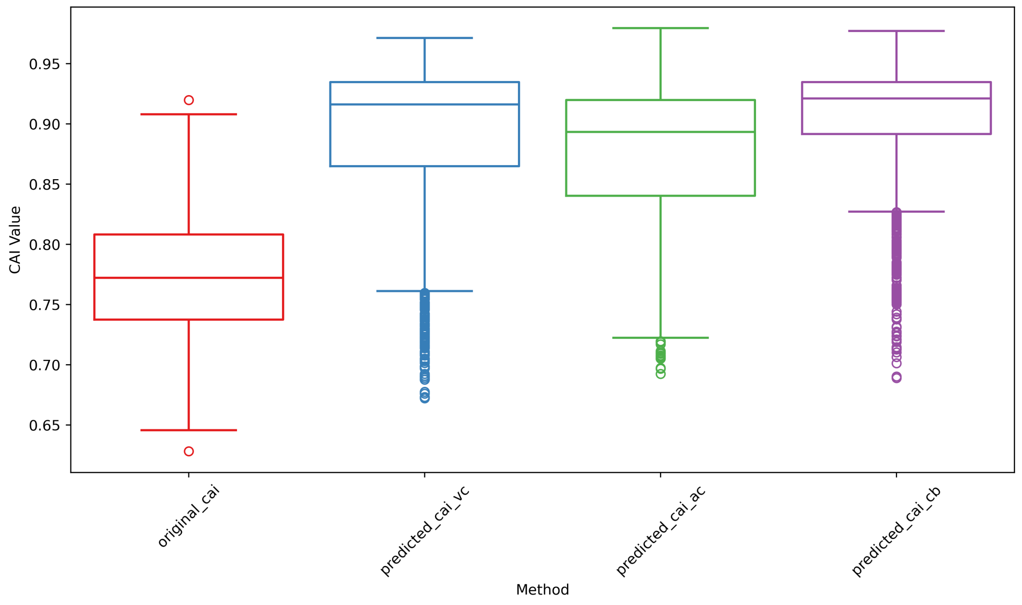

Figure 4.2: CAI values comparison across three different methods on Hg19_random test sequences.

On the *E.coli* test set, the 'all-codon' method improved the mean CAI by 23.3% to 0.85 ± 0.06. The 'valid-codon' method matched this performance with a CAI of 0.85 ± 0.05, while the 'codon-box' method achieved a slightly higher CAI of 0.87 ± 0.04, representing a 25.9% increase (Fig. 4.3).

On the Chinese-Hamster test set, all three methods performed similarly, with each achieving a mean CAI of 0.89 ± 0.003, representing a 13.9% improvement over the basis ORF sequences. The slight difference in the mean CAI value across methods was observed at the third decimal place (Fig. 4.4).

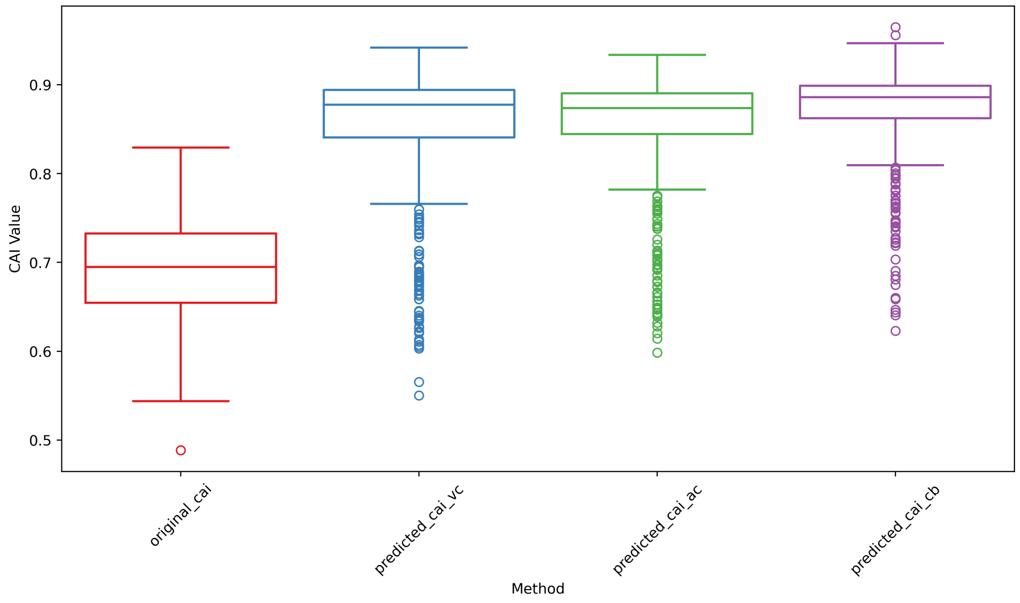

Figure 4.3: CAI values comparison across three different methods on *E.coli* test sequences.

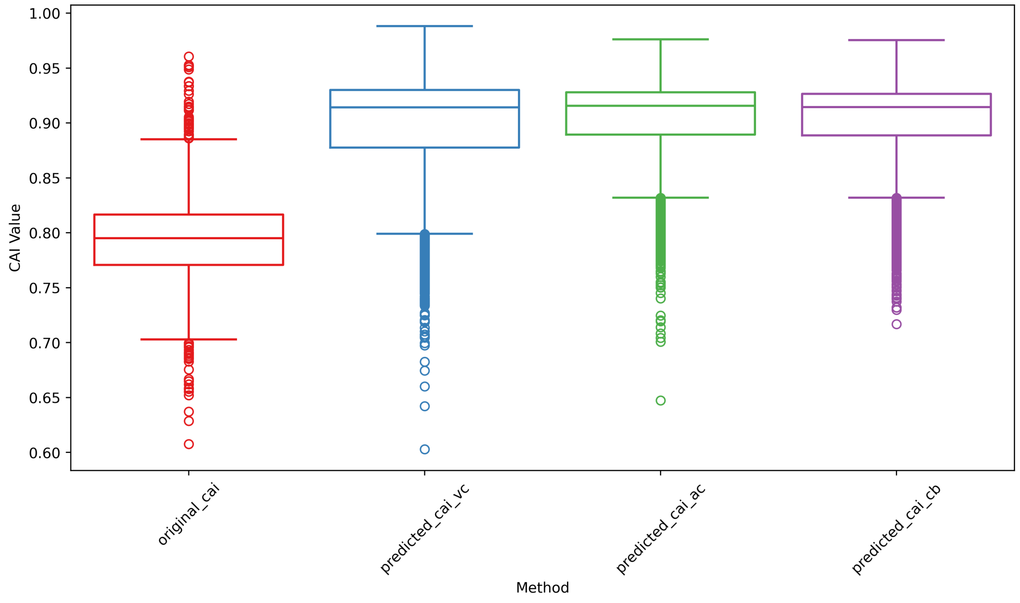

Figure 4.4: CAI values comparison across three different methods on Chinese-Hamster test sequences.

4.4.1 Statistical Test

To statistically evaluate the difference between methods, a two-sample Kolmogorov-Smirnov (KS-test) [32] was conducted to compare the empirical cumulative distribution function (CDF) of CAI values. The KS statistic measures the maximum absolute difference between the CDFs of the two methods (Eq. 4.3). The D-statistic value given by the KS test ranges from 0 to 1, where 0 indicates that the two CDFs are perfectly equal, whereas 1 represents completely different distributions.

$$D = \sup_{x} |CDF_{valid-codon}(x) - CDF_{codon-box}(x)| \tag{4.3}$$

For the Hg19_random dataset, the D-statistic between the 'all-codon' and 'codon-box' methods was greater than 0.20, indicating significant differences. The D-statistic between 'valid-codon' and 'codon-box' methods was less than 0.10, indicating 90% similarity in CAI values. On the other hand, for 'valid-codon' vs 'codon-box', the D-statistic was found to be < 0.10, showing 90% similarity between their predicted CAI values on test sequences (Fig. 4.5).

On the E. coli dataset, the D-statistic between the 'codon-box' and 'all-codon' methods was 0.216, indicating significant differences. The D-statistic between 'valid-codon' and codon-box' was 0.16, indicating 84% similarity (Fig. 4.6).

In the Chinese-Hamster dataset, the D-statistic values between all three methods were very close, with D-statistics around 0.08, indicating minimal differences between the CDFs of the CAI values for the 'valid-codon', 'codon-box', and 'all-codon' methods.

The p-value was found to be >0.01 for every KS test between the 'valid-codon' and 'codon-box' methods on each of the datasets except E. coli. It statistically proves no significant difference between the 'codon-box' and 'valid-codon' methods.

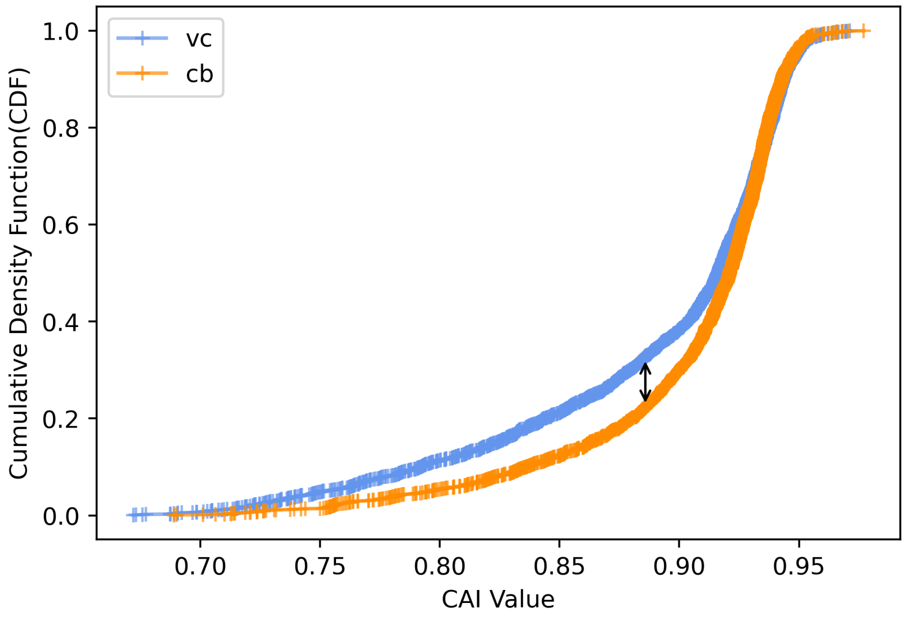

Figure 4.5: CDF difference between test set CAIs of vc (‘valid-codon’) and cb (‘codon-box’) for Hg19

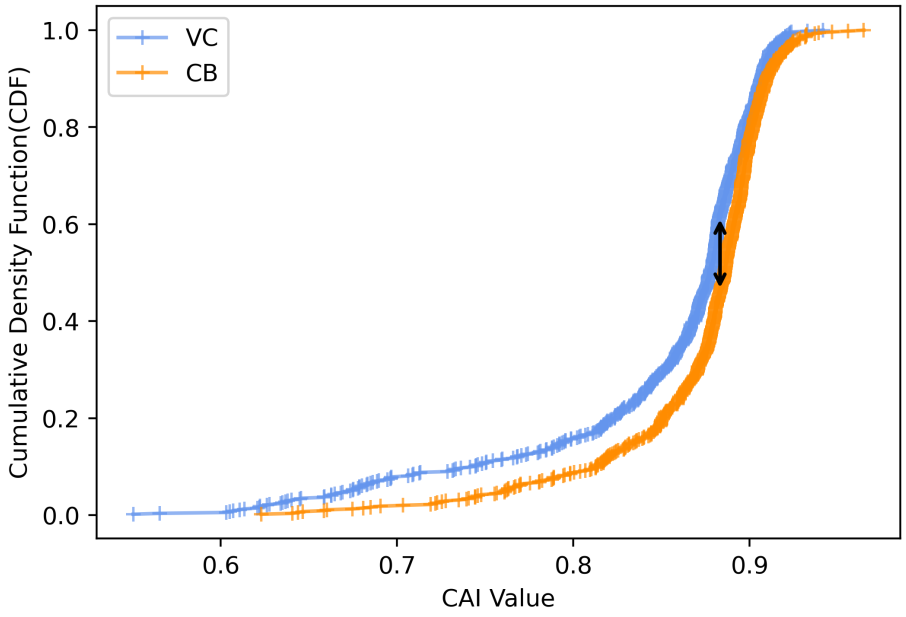

Figure 4.6: CDF difference between test set CAIs of vc (‘valid-codon’) and cb (‘codon-box’)for *E.coli*

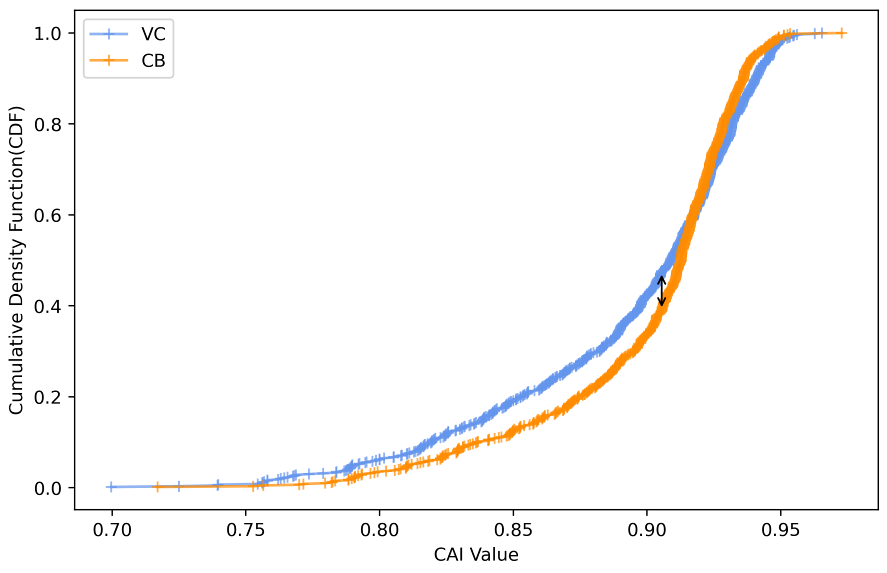

Figure 4.7: CDF difference test set CAIs of vc (‘valid-codon’) and cb (‘codon-box’)for Chinese-Hamster

4.4.2 Conclusion

The comparative analysis in this chapter provides valuable insights into the effectiveness of label space reduction strategies in codon optimization. The results showed that the ‘valid-codon’ method performed competitively with the ‘codon-box’ method across all datasets. This finding supports hypothesis 2, suggesting that the specific method of grouping codons, such as in codon-boxes, does not offer a significant advantage over other reduction strategies like ‘valid-codon’, which focuses solely on synonymous codons. Moreover, both the ‘valid-codon’ and ‘codon-box’ methods demonstrated marginally better performance in terms of the CAI compared to the ‘all-codon’ method. This outcome supports hypothesis 1, indicating that reducing the label space—whether through ‘codon-box’ grouping or ‘valid-codon’ enforcement—can indeed simplify model learning. These results suggest that while label space reduction is beneficial, the choice between methods like ‘valid-codon’ and codon-box may not be critical, as both offer comparable advantages.

Chapter 5

Codon Optimization for Enhancing ORF Expression and Stability in Humans

In this chapter, we investigate the application of the pre-trained protein language model, ProtBert, for optimizing ORF sequences to enhance expression and stability in humans. The potential research gap explored in this chapter's experiment is that the pre-trained protein language model, to the best of our knowledge, has not been utilized for the downstream task of codon optimization.

5.1 Background

In the field of NLP, transformers [54] have shown great potential over traditional RNNs. Every LM that has reached state-of-the-art results uses transformer architecture at its foundation. The following subsections contain the necessary background to understand the objective and methodology of utilizing the ProtBert for our experiment.

5.1.1 Transformer Model

The transformer model introduced by Vaswani et al. [54] revolutionized the field of NLP by eliminating the need for RNNs and introducing a self-attention mechanism. The transformer architecture consists of an encoder-decoder structure, with both the encoder and decoder being composed of multiple identical layers.

Self-Attention Mechanism

The core of the transformer model is the self-attention mechanism, which allows the model to weigh the importance of different words in a sequence when encoding a particular word. Given an input sequence, the self-attention mechanism computes three vectors for each word: Query (Q), Key (K), and Value (V).

The attention score for each word is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5.1}$$

In Eq. 5.1, Q, K, and V are the query, key, and value matrices. d_k is the dimension of the keys. The softmax function is used to ensure that the attention scores sum to 1, providing a probability distribution over the input words.

The self-attention mechanism allows the transformer to focus on different parts of the input sequence, capturing long-range dependencies that are difficult to model with traditional RNNs.

Multi-Head Attention

To allow the model to attend to information from different representation subspaces, the transformer uses multi-head attention (Eq. 5.2). Each attention head processes the information in parallel, capturing different aspects of the input sequence.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_{16}).W_o \tag{5.2}$$

In Eq. 5.2, W_o is the output weight matrix used to project the concatenated multi-head attention outputs.

Feed-Forward Networks (FFN)

Each encoder and decoder layer contains a feed-forward neural network that processes the output of the self-attention layer. This layer is applied to each position in the sequence independently and identically, allowing the model to apply non-linear transformations to the input representations.

5.1.2 Bidirectional Encoder Representation from Transformers (BERT)

BERT (Bidirectional Encoder Representations from Transformers) [9] is a transformer-based model that achieved state-of-the-art results in many NLP tasks. BERT is an encoder-only model that leverages the bidirectional nature of the transformer to capture the context of each word in both directions (left-to-right and right-to-left). BERT is pre-trained using a self-supervised learning method known as masked language modeling (MLM). In MLM, 15% of the input tokens are randomly masked, and the model is trained to predict these masked tokens. This approach enables BERT to understand the meaning of words based on their surrounding context, making it highly effective for downstream NLP tasks. Through MLM, BERT learns rich, context-sensitive representations of words.

5.1.3 Transfer Learning and Fine-Tuning

Transfer learning is a deep learning technique in which a model trained on one task is adapted to perform a different but related task. In the context of BERT, transfer learning involves fine-tuning the pre-trained model on a specific downstream task, such as text classification or question answering [43].

Different strategies have been developed for optimally carrying out fine-tuning processes [39]. Parameter Efficient Fine Tuning (PEFT), one such category of fine-tuning, has gained popularity due to its ability to reduce training cost and time [10]. PEFT can be divided into three categories: selective fine-tuning, adaptive fine-tuning, and reparameterization.

Selective Fine-Tuning Only a subset of layers is fine-tuned, often the last few layers, while the rest of the model remains frozen. This approach reduces computational costs and mitigates the risk of overfitting.

Adaptive Fine-Tuning Adaptive fine-tuning involves adding adapter neural layers in PLM, keeping all the pre-trained parameters frozen during fine-tuning [19]. It helps in significantly decreasing the number of trainable parameters, as the adapter parameters only need to be learned.

Reparameterization Reparameterization involves training all PLMs using computationally efficient techniques like low-rank adaptation (LoRA) [22].

Different LMs have been utilized in the past for different genomics tasks due to their ability to learn generalized knowledge during the pre-training phase [34, 38, 41, 42].

5.1.4 ProtBert: A Protein Language Model

The ProtTrans, introduced by Elnaggar et al. [12], leveraged the architecture of transformers to develop models specifically tailored to protein sequences.

These models, which include both auto-encoder and auto-regressive variants, were extensively trained on massive protein datasets, such as Uniref50 [52], Uniref100 [52], and BFD [51].

The principle behind ProtTrans is that protein sequences have an inherent structure and physio-chemical properties that can be learned by these models. ProtTrans models treat individual amino acids as tokens (analogous to words in a sentence). The primary objective is to generate rich embeddings that can be used for various downstream tasks, ranging from per-residue predictions (like secondary structure prediction in Q3 and Q8 formats) to sequence-level predictions (such as determining whether a protein is membrane-bound or water-soluble). These embeddings are derived from the final hidden layers of the model and are typically pooled using invariant functions (like maximum, sum, or mean) for classification tasks.

Among the models developed under ProtTrans, the ProtBert model stands out as a particularly powerful auto-encoder model due to its bidirectional nature, which allows it to capture context from both directions of a sequence, making it well-suited for tasks like codon optimization. Unlike uni-directional auto-regressive models, auto-encoder models like ProtBert can learn richer, context-aware representations, which are crucial for understanding complex biological sequences.

ProtBert Architecture

The architecture of ProtBert is rooted in the BERT model, which introduced the concept of bidirectional context gathering in transformer-based models. ProtBert was pre-trained on protein sequences in the UniRef100 dataset. It diverges slightly from traditional BERT as it focuses solely on masked language modeling (MLM). Protein sequences do not exhibit inter-sequence relationships analogous to sentences in a paragraph, making tasks like Next Sentence Prediction (NSP) irrelevant. ProtBert has a deep architecture and utilizes 30 transformer layers, compared to the 12 layers in the original BERT model. Each layer comprises multi-head self-attention mechanisms and feed-forward networks. The embeddings generated by ProtBert are 1024-dimensional, capturing the rich context of each amino acid in the protein sequence.

Table 5.1 describes the different hyperparameters of the model. Fig. 5.1 illustrates how the last layer hidden state (Fig. 5.1) representation for each token was utilized as per residue embeddings for amino acids in the input protein sequence.

5.2 Objective and Motivation

The central hypothesis of this chapter is that the ProtBert model contains rich, general knowledge of amino acid properties that can be effectively utilized for optimizing the ORF sequences in mRNA vaccines. Specifically, this knowledge is expected to enhance the process of codon optimization, when fine-tuned for human-specific ex-

Table 5.1: Bert-attention layer hyper-parameters values of the self-attention layer used in ProTrans for pre-training.

Hyperparameters	**Value**
Encoder Layers	30
Embedding dimension (E_d)	1024
q, k and v dimension ($d_k = d_v$)	64
Dimension of Model (d_{model})	1024
Number of Heads (n_{head})	16

pression and stability. The objective is to utilize the embeddings learned by ProtBert, which capture complex interactions and physiochemical characteristics of amino acids. Building upon the objective, the motivation for this research is to evaluate the effectiveness of the fine-tuned ProtBert model in optimizing ORF sequences specifically for human use. This evaluation is carried out by benchmarking the optimized ORF sequences against the industry-approved mRNA vaccines, such as Pfizer (BNT162b2) and Moderna (mRNA-1273), which target the SARS-CoV-2 spike protein. We use key metrics such as CAI, MFE and GC-Content to assess optimized ORF sequences for expression and stability.

5.3 Experimental Setup

5.3.1 Dataset Curation

The dataset used is derived from the Hg19_short, discussed in Sec. 3.1. The stability of ORF sequences is a crucial factor in mRNA vaccine design, as it directly impacts the longevity and effectiveness of the vaccine. A stable mRNA sequence is less likely to degrade prematurely within human cells, thus ensuring that the target protein can be synthesized effectively and for a longer duration. We filter the sequences on the basis of their length and stability. Sequences with length greater than 150 amino acids (aa) were removed. After this initial filtration, 24530 sequences out of the original 40,000 were retained. The second round of filtering focused on the stability of the remaining sequences, as measured by their MFE values. The MFE of an RNA sequence was calculated using the RNAFold tool [35] at the normal human body temperature, i.e. 37°C. Sequences were selected such that they have a mean MFE value of -40.5 $\pm$ 9.8 kcal/mol, ensuring that the dataset included sequences with a diverse range of stability levels. This diversity in stability is hypothesized to be crucial for the model to learn the nuanced relationship between sequence composition and stability. This would ultimately allow the model to generate ORF sequences that are both stable and optimally expressible in human cells. The distribution statistics of the dataset are described in Table 5.2. This filtering process not only enhances the quality of the dataset but also impacts the model's learning by enabling it to recognize and prioritize sequences that are more likely to yield stable and effective mRNA vaccines. The final dataset, after filtering, contained 4,877 sequences, providing a robust and stable foundation for the fine-tuning process.

Table 5.2: Distribution of sequence length and MFE of the filtered dataset

Sequence Features	**count**	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
sequence_length	4877.00	112.12	21.40	78.00	93.00	111.00	129.00	150.00
mfe_original	4877.00	-40.50	9.82	-96.50	-46.59	-39.40	-33.00	-23.50

For independent testing (Table 5.3), we increased the protein sequence length's upper limit to 500 aa. Additionally, the ORF sequences from the Pfizer (BNT162b2) and Moderna (mRNA-1273) vaccines, both targeting the SARS-CoV-2 spike protein, were used as benchmarks. These sequences were obtained from UCSC [1], and their ORF sequences were extracted using the NCBI ORF Finder API [2].

[1]https://genome.ucsc.edu/cgi-bin/hgSearch?search=BNT162b2&db=wuhCor1

[2]https://www.ncbi.nlm.nih.gov/orffinder/

Table 5.3: All different kinds of sequences from cross organism used for testing the fine-tuned ProtBert and baseline model. The max length column reflects the dataset's maximum length of the protein sequence in terms of amino acids (aa).

Test-Dataset	**No. of Sequence**	**Max Length**
Hg19_short	976	500aa
E.coli	361	500aa
Chinese-Hamster	667	500aa
SARS-Cov-2 (Wuhan S1-Spike Protein)	1	1271aa

5.3.2 Model and Training Setup

The training setup for the ProtBert model involves extracting 1024-dimensional embeddings for each amino acid in the protein sequence. These embeddings are passed through an adapter module that predicts the optimal codon sequence. The adapter module includes a softmax classifier [44] applied in a time-distributed manner, ensuring that each input token (amino acid) is mapped to one of the 61 possible codons. It was then passed through a codon-masking layer introduced as 'valid-codon' in Sec. 4.1.

Fine-tuning of ProtBert was conducted using two distinct approaches. In the first approach, 'Adaptive-ProtBert', we kept all ProtBert parameters frozen and added a new classifier module. This classifier, trained specifically for codon optimization, received 1024-dimensional vectors representing each amino acid from the pre-trained ProtBert layers. The classifier then predicted the optimal codon as a sequence tagging task discussed in Sec. 3.4. The second approach,' Adasel-ProtBert', combined adaptive and selective fine-tuning. In this method, the weights of the BertOutput layer and the classifier module were unfrozen. This allowed the model to fine-tune not only the classifier but also the final layers of ProtBert, enabling it to adapt better to the specific task of codon optimization while retaining its pre-trained knowledge. Figure 5.1 illustrates the architecture used for these approaches. Both methods utilized

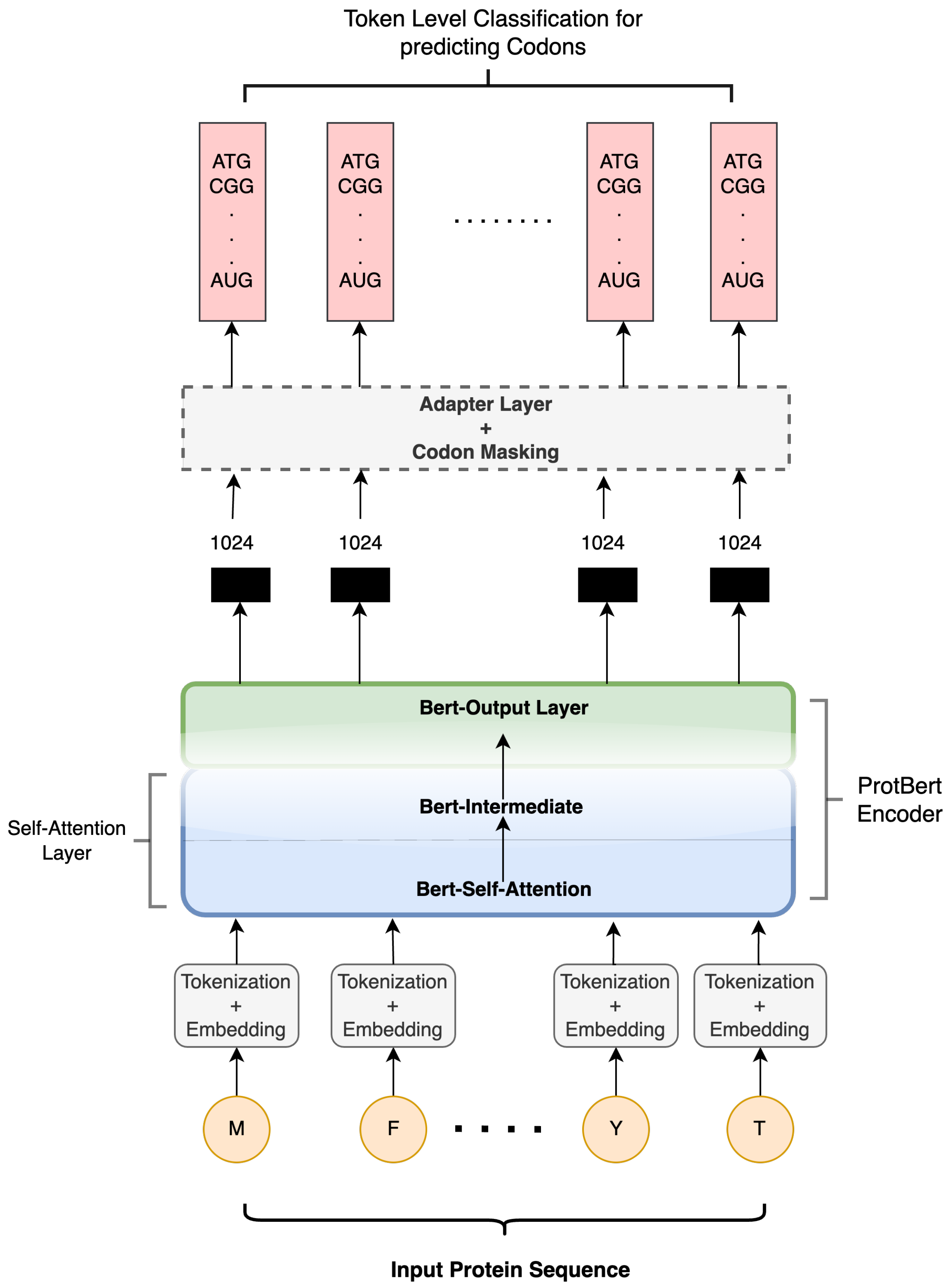

Figure 5.1: ProtBert Architecture with adapter layer and codon masking. The dense vector of dimension 1024 for each amino acid was utilized for codon optimization.

codon masking to ensure biologically valid predictions. Table 5.4 contains the best value of the hyper-parameters chosen for training. Another model used in this experiment is the 'Dense bi-LSTM' (Sec. 3.3.5) network with the same hyper-parameters mentioned in Sec. 4.3.2. Fig. 4.1 illustrates the architecture of 'Dense bi-LSTM'. This setup provides a robust framework for evaluating how well-fine-tuned transformer models like ProtBert can optimize ORF sequences compared to traditional deep-learning approaches. Fig. 5.2 illustrates the training pipeline used in this experiment.

Table 5.4: Hyper-Parameters for fine-tuning ProtBert model and baseline 'Dense bi-LSTM' model. The max length parameter determines the context size for the LM. The acronym aa stands for amino acid.

Hyperparameter	**Fine-Tuning ProtBert**	**Dense-bilstm**
Dataset	Hg19_short	Hg19_short
batch_size	32	32
masking-codon	✓	✓
Epochs	15	15
Optimizer	Adam	Adam
Learning rate (lr)	0.01	0.001
Early Stopping	✓	✓
Patience	5	5
max length	150 aa	150 aa

Fig. 5.2 illustrates the different components and the flow of training pipeline used in this experiment.

5.4 Results and Discussion

Following the training phase, the performance of the three models, Adasel-ProtBert, Adaptive-ProtBert, and Dense bi-LSTM, was evaluated on held-out test data from the Hg19_short dataset and independent test sets from *E.coli*, Chinese-Hamster(CH)

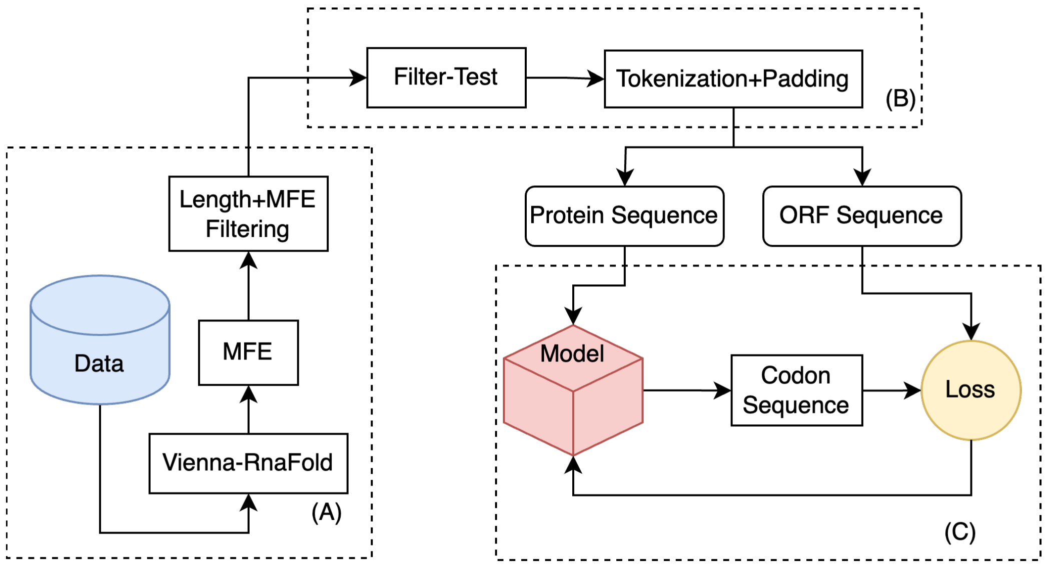

Figure 5.2: Codon-Optimization methodology on sequences filtered on length and stability.

and SARS-CoV-2 (Wuhan) spike protein sequences. We compare these models with four other design types. The Wild-Type sequence represents the naturally occurring genetic sequence of the SARS-CoV-2 (Wuhan) spike protein, serving as a baseline for assessing improvements made through optimization. The Pfizer (BNT162b2) and Moderna (mRNA1273) vaccines are industry-standard examples of codon optimization, where ORF sequences have been carefully engineered for enhanced expression and stability in human cells. These vaccines set a benchmark for evaluating new models' effectiveness. Lastly, the Linear-Design approach is a non-deep learning-based state-of-the-art method that prioritizes the thermodynamic stability of the mRNA sequence, often at the expense of expression levels. The primary metrics we used for measuring the optimized ORF fidelity from our trained models were CAI, MFE, and GC-Content. The following subsections provide a detailed analysis of the results across different test sets.

5.4.1 SARS-CoV-2 Spike Protein Test

The fine-tuned models, alongside the Dense bi-LSTM were evaluated on the ORF sequences of the SARS-CoV-2 (Wuhan) spike protein, with the Pfizer (BNT162b2) and Moderna (mRNA1273) vaccines ORFs serving as benchmarks.

As shown in Table 5.5, Adasel-ProtBert achieved the highest CAI of 0.97, surpassing both industry-standard vaccines and other design types. This high CAI value suggests a strong alignment of codon usage with human cellular machinery, which is expected to enhance expression levels. The MFE value of -1520.1 kcal/mol indicated robust thermodynamic stability, competitive to Dense bi-LSTM, which had an MFE of -1563.7 kcal/mol. The GC-Content for Adasel-ProtBert was also notably high at 0.602, indicating a strong potential for mRNA stability. Adaptive-ProtBert showed a

Table 5.5: Comparison of ORF sequences from the model against different other design algorithms.

Design-Type	**CAI**	**MFE (kcal/mol)**	**GC-Content**
Wild-Type	0.67	-1111.5	0.37
Adasel-ProtBert	$\boxed{\mathbf{0.97}}$	$\boxed{-1520.1}$	$\boxed{\mathbf{0.602}}$
Adaptive-ProtBert	0.94	$\boxed{\mathbf{-1544.0}}$	$\boxed{0.582}$
Dense bi-LSTM	**0.95**	$\boxed{\mathbf{-1563.7}}$	0.580
Pfizer (BNT162b2)	0.94	-1314.12	0.570
Moderna (mRNA1273)	$\boxed{\mathbf{0.97}}$	-1481.80	$\boxed{\mathbf{0.62}}$
Linear-Design	0.724	$\boxed{\mathbf{-2477.7}}$	0.536

slightly lower CAI of 0.94 but had an MFE of -1544.0 kcal/mol, suggesting a slightly more stable RNA structure. Its GC-Content was competitive to Adasel-ProtBert at 0.582, reinforcing its potential for enhanced stability.

The Dense bi-LSTM performed comparably, with a CAI of 0.95, an MFE of -1563.7 kcal/mol, and a GC-Content of 0.580, making it competitive with the fine-tuned models.

The Wild-Type sequence (Table 5.5) from the SARS-CoV-2 (Wuhan) virus was significantly outperformed by all models across all metrics, demonstrating the effectiveness of codon optimization. Notably, both Adasel-ProtBert and Adaptive-ProtBert exceeded the performance of Pfizer's BNT162b2 and were on par with or slightly better than Moderna's mRNA1273 in terms of CAI and GC-Content (Fig. 5.3).

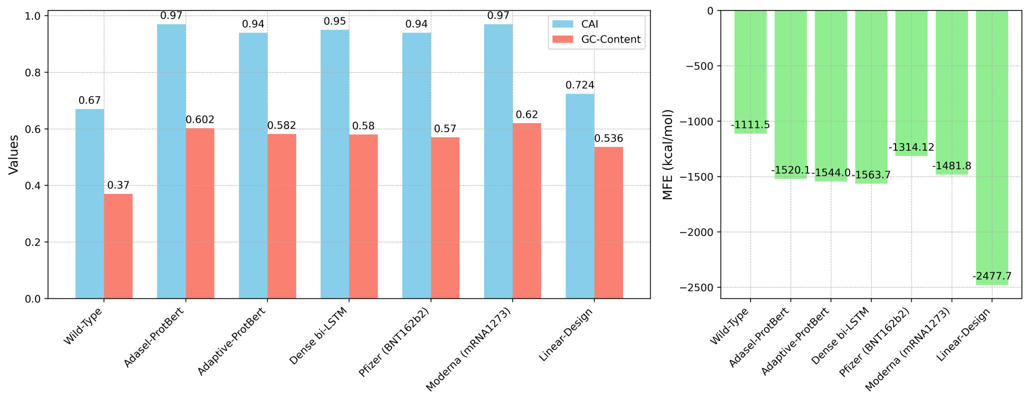

Figure 5.3: CAI, GC and MFE for different design methodologies.

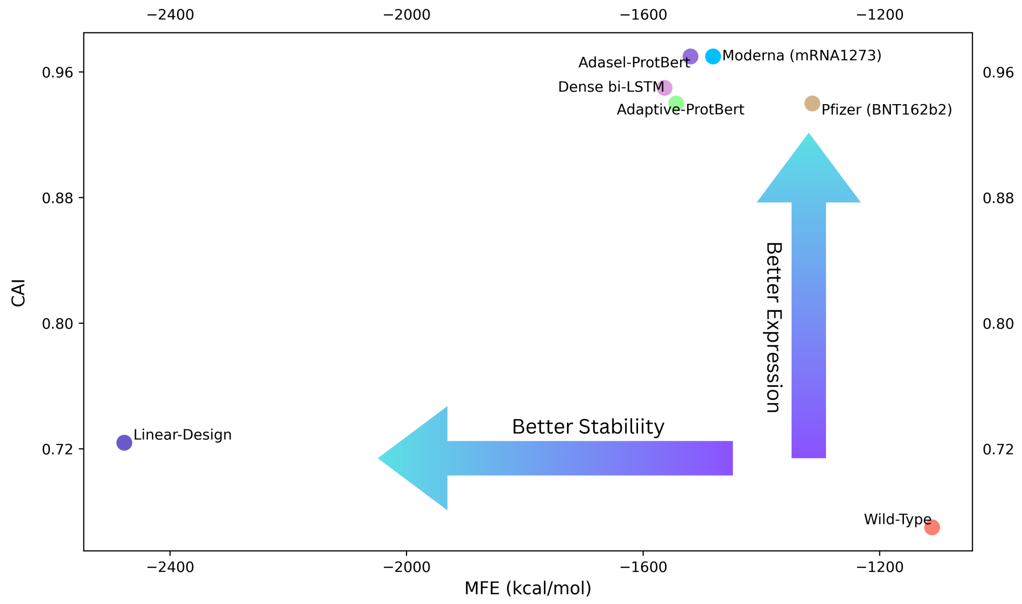

Figure 5.4: Comparison on CAI and stability of different design types

Summary Overall, the results demonstrate the potential of using deep learning models to enhance the ORF design for the SARS-CoV-2 spike protein. Each model presents a unique set of strengths: Adasel-ProtBert offers a well-rounded performance across all metrics, Adaptive-ProtBert excels in thermodynamic stability, while the Dense bi-LSTM remains competitive.

5.4.2 Cross Organism Test

In this test, we assessed the performance of models trained on Hg_19 for codon optimization on *E.coli* and Chinese-Hamster genes. The motivation behind this test is to validate that the models are optimized for human codon usage patterns. The models were assessed using the same three metrics: CAI, MFE, and GC-Content, as in the previous section.

Table 5.6: Results for each of the models trained on Hg19_short across three organisms: Human(Hg19), *E.coli* and Chinese-Hamster on metrics CAI and MFE

Model	**Metric**	**Hg19**		***E.coli***		**Chinese-Hamster**	
		Predicted ORF	Wild-Type	Predicted ORF	Wild-Type	Predicted ORF	Wild-Type
Adasel-ProtBert	CAI	0.94	0.74	0.61	0.67	0.97	0.80
	MFE (kcal/mol)	-47.23	-39.97	-105.51	-84.55	-146.30	-117.43
	GC-Content	0.68	0.60	0.61	0.47	0.62	0.52
Adaptive-ProtBert	CAI	0.94	0.74	0.59	0.67	0.96	0.80
	MFE (kcal/mol)	-46.80	-39.97	-104.64	-84.55	-149.96	-117.43
	GC-Content	0.68	0.60	0.61	0.47	0.62	0.52
Dense bi-LSTM	CAI	0.94	0.74	0.60	0.67	0.96	0.80
	MFE (kcal/mol)	-45.90	-39.97	-103.90	-84.55	-147.8	-117.43
	GC-Content	0.67	0.60	0.60	0.47	0.61	0.52

Human Genome Analysis All models demonstrated a 27.03% increase in CAI, from 0.74 to 0.94, indicating a significant improvement in the predicted sequences alignment with human codon usage preferences. This optimization suggests the potential for enhanced protein expression in human cells. MFE improvements ranged from 15.91% to 19.59%, with values improving from -39.97 kcal/mol in the original sequences to between -45.90 and -47.23 kcal/mol in the predictions, pointing to increased mRNA structural stability. GC-Content increment of 13.33% across all models was observed, enhancing from 0.60 to between 0.67 and 0.68, which correlates with the improved stability metrics. Table 5.6 contains the results for each of the models for human genes.

Escherichia coli (*E.coli*) Dataset On the E. coli test set, the models showed a decrease in CAI by 8.96% for 'Adasel-ProtBert' (0.67 to 0.61), 11.94% for 'Adaptive-ProtBert' (0.67 to 0.59) and 10.44% for 'Dense bi-LSTM' (0.67 to 0.60), potentially indicating less biased codon usage for bacterial expression systems. However, MFE showed consistent improvement across all models, with increments ranging from 22.74% to 23.93%. This enhancement suggested that despite the decrease in CAI, the structural stability of mRNA remained intact in bacterial cells. GC-Content improvements were substantial, with a 29.79% increase, which contributed to the increased mRNA stability observed. Table 5.6 contains consolidated results of each model.

Chinese-Hamster Dataset For the Chinese-Hamster test set, CAI improvements were large, with an increase of 21.25% for 'Adasel-ProtBert' and a 20% increase for both 'Adaptive-ProtBert' and 'Dense bi-LSTM'. MFE enhancements were also notable, with increases ranging from 25.83% to 27.72%, indicating the generation of highly stable mRNA configurations. GC-Content increased by 19.23% across most models, aligning with the trends seen in human genomic data.

Observation

The results from the cross-organism test provided interesting insights, particularly regarding the Chinese-Hamster and human Hg19 genes. Models trained on Hg19 genes showed efficient performance on Chinese-Hamster genes but not on E. coli. This discrepancy led to further investigation into the reason behind these results, specifically analyzing codon distribution patterns across the three organisms.

Codon Distribution Analysis

We came across an interesting empirical emerged from the cross-organism test results. While all models showed significant improvement over the Wild-Type sequences for Hg19 and Chinese-Hamster genes, they did not exhibit the same level of improvement for *E.coli* sequences. The goal of the encoder language model was to learn the probability distribution over codons (tokens) and find the most probable one. This led us to investigate the similarity among their codon distribution for each amino acid using weight vectors from the CAI calculation.

During CAI calculation, the weights of each codon are extracted from the Relative Synonymous Codon Usage (RSCU). For example, the weight vector for the amino acid Alanine (A) is defined as $W_A = [w_{A,gct}, w_{A,gcc}, w_{A,gca}, w_{A,gcg}]$, where 'GCT', 'GCC', 'GCA' and 'GCG' are synonymous codons. Each element in W_A is the relative adaptiveness (Eq. 1.2) of the codon for Alanine. Similar to previous experiments, reference sequences from each organism were used to calculate the relative adaptiveness of each codon, resulting in weight vectors for each amino acid across the three organisms: humans (Hg19), *E.coli*, and Chinese-Hamster.

Principal component analysis (PCA) [56] was performed on 18 amino acid weight vectors, eliminating two singular vectors for Methionine (M) and Tryptophan (T). The analysis revealed that except for Cysteine (C), the weight vectors for Hg19 and Chinese-Hamster showed high similarity, unlike those for E. coli (Fig. 5.5). This finding empirically suggests that human and Chinese-Hamster genes share a similar codon distribution, which likely contributed to the superior performance of the models on these genomes compared to E. coli.

5.5 Conclusion

The results of this chapter reinforce the hypothesis that ProtBert, with its rich, general knowledge of amino acid properties, can be effectively fine-tuned for codon opti-

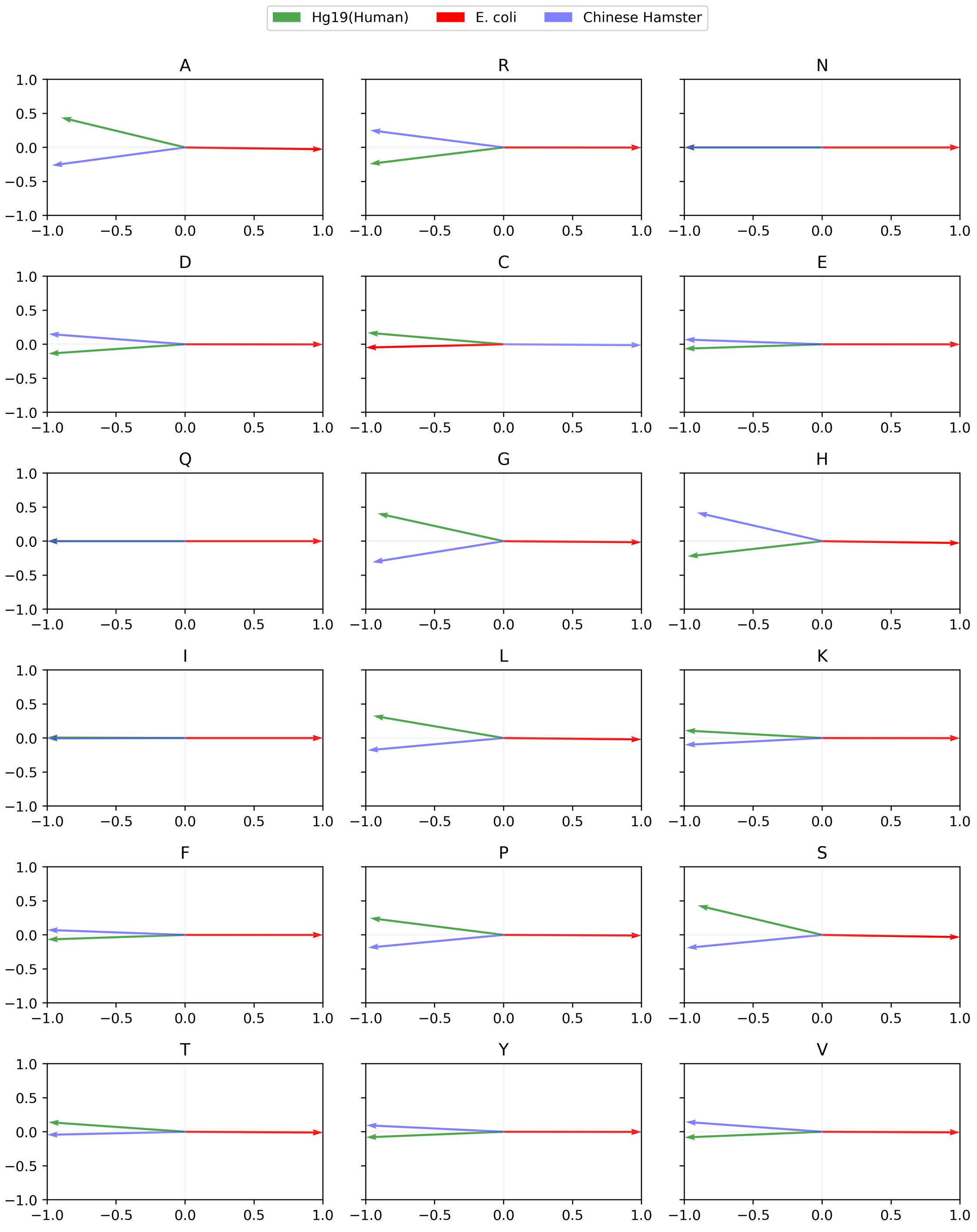

Figure 5.5: Weight vectors projection of different amino acids for each of Hg19,*E.coli* and Chinese-Hamster gene sequences. Each of the letters for a subplot denotes a specific amino acid.

mization in ORF sequences, specifically for human mRNA vaccines. Adasel-ProtBert demonstrated superior performance in balancing CAI, MFE, and GC-Content, suggesting its potential to optimize mRNA sequences better than current industry standards, such as Pfizer's and Moderna's vaccines.

The cross-organism tests further validated that models trained on Hg19 were optimized for human codon usage patterns but were less effective on E. coli, highlighting the importance of codon distribution similarity. These findings confirm that fine-tuned models can potentially enhance mRNA vaccine design by optimizing for both expression and stability, aligning with the chapter's objectives.

Chapter 6

New Explorations and Challenges

6.1 Motivation: Smooth CAI as a Regularizer

Protein expression is an important factor, as focused previously in all the experiments. CAI [49] has been used as an evaluation metric in previous experiments to measure the expression level of the synthetic genes predicted from the model. In prior experiments (Sec. 4.1, and Chapter 5), models showed inherent learning of codon distribution and usage bias of a host genome during training. In this experiment, the main ideology was to modify the loss function by utilizing CAI as an additional penalizing term alongside cross-entropy loss. Including CAI in objective function was not trivial; modifications performed to overcome the challenges are discussed in the next section.

6.1.1 CAI Formulation for Smoothness

In order to train any machine learning model, the smoothness of the loss function is a fundamental requirement to perform gradient descent in back-propagation. Unfortunately, the smoothness of CAI gets rendered because of the 'max' operator used in relative-adaptiveness (Eq. 1.2) calculation of the i^{th} codon encoding a^{th} amino acid.

Two primary adaptations were performed in CAI formulation to overcome the challenge of non-differentiability. At first, the max operation was replaced with softmax to calculate relative adaptiveness. The modified formulation for RSCU is described in Eq. 6.1 below.

$$w_{a,j} = \frac{e^{RSCU_{a,j}}}{\sum_{k=1}^{n_a} e^{RSCU_{a,k}}} \tag{6.1}$$

The second modification was performed in CAI calculation, adapting the geometric mean over relative adaptiveness (Eq. 1.3). As logarithm is a monotonic function, for mathematical convenience in gradient calculation, the log operation was induced over it.

$$\begin{gathered}\text{CAI} = \left(\prod_{k=1}^{L} w_k\right)^{\frac{1}{L}} \\ \log(\text{CAI}) = \log\left\{\left(\prod_{k=1}^{L} w_k\right)^{\frac{1}{L}}\right\} = \frac{1}{L}\cdot\left(\sum_{k=1}^{L} log(w_k)\right)\end{gathered} \tag{6.2}$$

Eq. 6.3 describes the formulation of log-softmax-cai referred to as $Smooth_{CAI}$ in this work.

$$\therefore Smooth_{CAI} = \frac{1}{L}\cdot\left(\sum_{k=1}^{L} log(w_k)\right) \tag{6.3}$$

6.1.2 Experimental Setup

Hg19_short was used with a similar train, validation, and test split done for the experiment in Sec. 5.1. After the filter test, random 10000 sequences with lengths less than equal to 150 nt were retained in order to have faster training. The hyper-parameters, along with the model used for training, were set identically to 'valid-codon' in Sec. 4.3.2.

Loss Function: The modified loss function (L_{CAI}) used in this experiment was done as additive penalization of $Smooth_{CAI}$ term. Since the goal is to maximize CAI, hence it was transformed into maximization by subtracting the $Smooth_{CAI}$. Eq. 6.4 describes the L_{CAI} loss.

$$L_{CAI} = L_{CEL} - Smooth_{CAI} \tag{6.4}$$

Here, L_{CEL} is taken from the Eq. 4.2.

6.1.3 Results and Discussion

The model was trained using 'all-codon' and 'valid-codon' methods. Both the methods showed better CAI than their wild-type sequences. However, there was no significant improvement observed in the CAI of test sequences over the previous methods (Chapters 4 and 5) where L_{CAI} was not utilized as an additional penalty term.

The 'valid-codon' method gave the highest expression value of 0.90 when compared with the 'all-codon' method (CAI=0.81). It demonstrated one more instance where the 'valid-codon' method performed better than the 'all-codon.' Overall, in this work, a new loss function was formulated with a hypothesis to increase the gene expression level by penalizing the model for preferring non-biased codons. Although the results did not come out to be positive, they can be an ideation step for future work.

6.2 User Defined RNA design

6.2.1 Motivation

In the above experiment, the focus was on codon optimization for increasing protein expression. Stability is an important aspect of designing RNA sequences because of its impact on efficient translation. For an ORF sequence the stability is not only affected by codons but also gets affected by the individual nucleotide bases. The percentage of GC-Content plays an important role in maintaining the stability of mRNAs within the cell. A multi-objective optimization approach was conducted in this experiment to perform codon optimization for user-specified CAI and stability value. There were two different methodologies for calculating the stability of the codon sequence. Both of them are discussed in the next section, describing the experimental setup for training along with the trade-offs between the two methodologies of calculating the stability.

6.2.2 Experimental Setup

Data

The data used in this experiment was the same as used in the previous experiment. The motivation behind taking shorter sequences is to account for the cubic polynomial complexity of prediction tools.

Stability Prediction Setup

User-defined RNA design involves specifying the required stability for the predicted ORF sequence. Stability is inversely proportional to degradation, keeping it as a base ground; the main focus was to quantify the degradation of the ORF sequence.

OpenVaccine Challenge published on kaggle [8] sought to predict the degradation of mRNAs under different environmental conditions. The participants were required to build a machine-learning model to predict the degradation values at each base of the input mRNA sequence under five different conditions. It was a supervised learning problem where the input features were the nucleotide sequence, its secondary structure, the loop type at each base, and the base-pairing probability matrix (bpps). In this multi-regression setting, the loss function to be optimized was the mean column-wise root mean squared error (MCRMSE).

The top performers in the contest published their results and achieved the best MCRMSE score of 0.23 [20]. RNAdegformer a transformer based model proposed in this work was utilized as an auxiallary to quantify the degradation value of the predicted ORF sequences in the codon optimization task with user defined stability and CAI. The pre-trained RNAdegformer predicted the degradation rate at each base of the predicted ORF sequence. In order to condense the degradation vector ($\mathbb{R}^{1\times5}$)in a singular real value, the mean-of-mean of degradation values was taken. The mean-of-mean is defined initially as taking an average of 5 degradation target values at each base (locally) and then taking the average for the complete sequence length (globally). This singular degradation value is referred to as avg_deg in this work (Eq. 6.5).

Secondary Structure Tools Setup The RNAdegformer model requires a secondary structure, loop type, and bpps matrix of the mRNA sequence (ORF sequence) alongside the nucleotide sequence. Features were generated at 37°, using RNAFold [35]. There were two ways to set up the RNAFold, one is through arnie[1] as wrapper over the source code of Vienna-RNAfold and the other one using ViennaRNA python package[2]. The training time, however, was found to be less for the latter. The use of an Arnie wrapper costs 72.30 seconds per iteration during the training, whereas, on the other hand, Vienna-RNAfold took only 54.36 seconds per iteration, reducing the computation cost by 24.81% per iteration in both training as well as inference.

Stability with RNAdegformer and Vienna-RNAfold

The five degradation rates predicted by RNAdegformer were reactivity (reactivity values), deg_pH10 (likelihood of degradation at base with pH=10), deg_Mg_pH10 (likelihood of degradation at the base with pH=10 and gestating with magnesium), deg_50C (likelihood of degradation at base without incubating with magnesium at 50°C) and deg_Mg_50C (likelihood of degradation after gestation with magnesium at 50°C), each of them being real values. To condense the degradation rate predicted by RNAdegformer at each base of the mRNA sequence to a single real value, the mean-of-mean was taken as described in Eq. 6.5 below.

$$avg\_deg = \frac{1}{N} \times \sum_{i=1}^{N} \left(\frac{1}{5} \times \sum_{j=1}^{5} base\_degradation_{i,j} \right) \tag{6.5}$$

Here (Eq. 6.5), N refers to the number of nucleotides in the sequence, and $base\_degradation_{i,j}$ determines the predicted j^{th} degradation rate at the i^{th} base of the mRNA sequence. It condensed the avg_deg of the sequences between 0 and 1.

In order to reduce the overhead computations at each instance of training, the

[1]https://github.com/DasLab/arnie
[2]https://pypi.org/project/ViennaRNA/

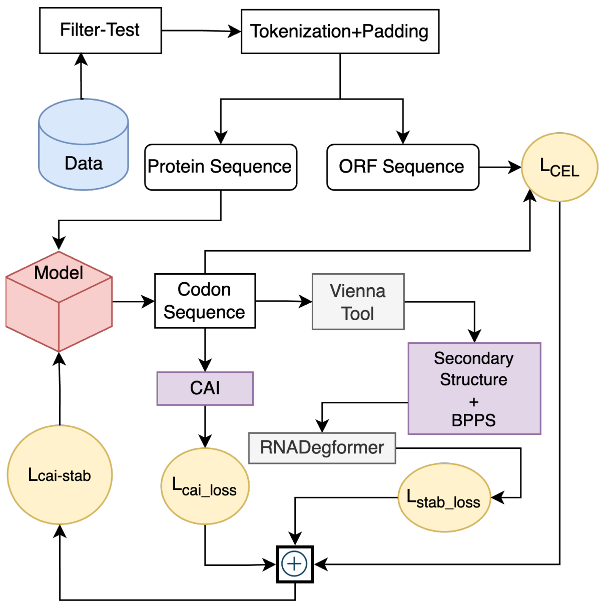

Figure 6.1: Codon Optimization with user-defined CAI and stability, where stability is predicted by pre-trained RNAdegformer

degradation rate prediction from the RNAdegformer was replaced with MFE from RNAFold. It cut off the cost of calculating the bpps matrix, secondary structure, and degradation rates at each base used for RNAdegFormer. It helped to reduce per iteration training time from 54.36 sec to 3.61 sec, i.e., a decrease of 93.3%. In order to condense MFE values between 0 and 1, it was length normalized.

Fig. 6.1 shows the training pipeline using RNAdegformer, and Fig. 6.2 illustrates the training pipeline for user-defined mRNA sequence using minimum free energy. Similar to previous experiments, 'Dense bi-LSTM' (Fig. 4.1) and 'Adaptive-ProtBert' (Fig. 5.1) were used for training purpose.

Loss and Evaluation Criterion

As mentioned in the earlier experiments, the Sparse Cross-Entropy Loss was used as an objective function, directing the model to learn the inherently codon distribution of ground truth. Apart from it, two other loss components were added in this experiment: cai loss (L_{cai_loss}) and stability loss (L_{stab_loss}).

The user-defined parameters req_cai and req_stab determined the required CAI

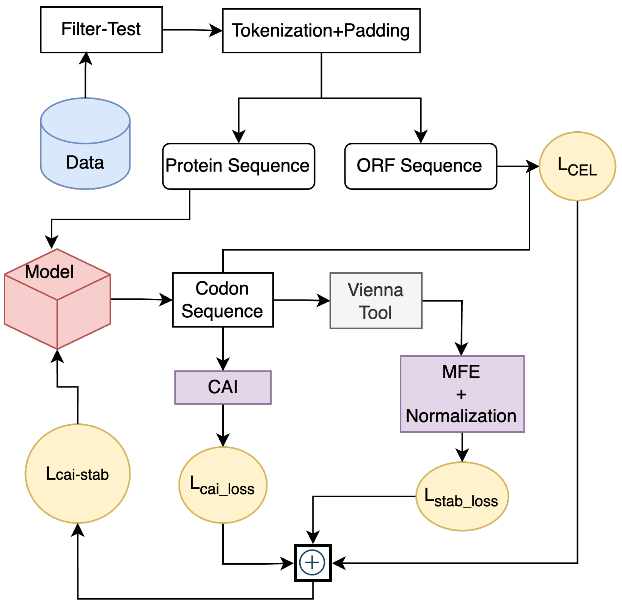

Figure 6.2: Codon Optimization with user-defined CAI and stability, where stability is calculated as minimum free energy by Vienna-RNAfold at 37°C.

index and stability for the predicted codon sequence from the trained model. In order to direct the model towards a specific CAI index and stability, mean squared error losses corresponding to each of the metrics were included. The final multi-objective loss function to be optimized was $L_{cai-stab}$, described in the Eq. 6.6 below.

$$L_{cai-stab} = L_{CEL} + \alpha * L_{cai\_loss} + (1-\alpha) * L_{stab\_loss} \tag{6.6}$$

$$L_{cai\_loss} = (\frac{1}{N}\sum_{i=1}^{N}(CAI_{pred} - req\_cai)^2)) \tag{6.7}$$

$$L_{stab\_loss} = (\frac{1}{N}\sum_{i=1}^{N}(Stability_{pred} - req\_stab)^2)) \tag{6.8}$$

In Eq. 6.6, α was used as a hyper-parameter to have a weighted multi-objective loss function. Since CAI index and stability of mRNA sequences are conflicting parameters [30], therefore it was formulated as α and $1-\alpha$ as the coefficients L_{cai_loss} and L_{stab_loss} respectively. The L_{cai_loss} and L_{stab_loss} of predicted sequences were averaged over the entire batch of size N.

6.2.3 Results and Discussion

The experiments were carried out on different values of $alpha = 0.5$, 0.3, and 0.1. The stable value of α was taken as 0.5, as it helped the learning to be stable for both CAI and stability losses in Eq. 6.6. Table 6.1 shows the comparison of different models with each of the stability tools on CAI and stability values.

Table 6.1: User-defined ORF sequence results.

Model	α	CAI			Stability			Stability-Tool
		Predicted CAI	Required CAI	L_{cai_loss}	Predicted Stability	Required Stability)	L_{stab_loss}	
Dense bi-LSTM	0.5	0.828	0.80	0.005	0.42	0.20	0.063	RNADegformer
Dense bi-LSTM	0.5	0.826	0.80	0.004	-0.20 kcal/mol	-0.50 kcal/mol	0.096	Vienna-RnaFold
Adaptive-ProtBert	0.5	0.864	0.80	0.004	0.45	0.25	0.042	RNADegformer
Adaptive-ProtBert	0.5	0.871	0.80	0.004	-0.21 kcal/mol	-0.50 kcal/mol	0.094	Vienna-RnaFold

Both the ‘Dense bi-LSTM’ and ‘Adaptive-ProtBert’ showed less accurate CAI to the req_cai = 0.80. The ‘Dense bi-LSTM’ predicted ORF sequences CAI were 0.828 and 0.826. They are comparatively better than the ‘Adapative-ProtBert’, which had CAI values of 0.864 and 0.871. These results suggested that both models were not effective enough in choosing codons to direct the ORFs toward the required CAI.

For the second metric (stability), high variability was observed in the results between two different tools for stability quantification. With RNADegformer, neither model was able to predict ORFs near the required stability. ‘Dense bi-LSTM’ had a degradation value of 0.42 compared to req_stab = 0.20. On the other hand, ‘Adaptive-ProtBert’ had a degradation of 0.45 compared to req_stab = 0.25. RNAFold with both the models had stability values of -0.20 kcal/mol (Dene bi-LSTM and -0.21 kcal/mol (‘Adaptive-ProtBert’)), quite far away from the req_stab = -0.50 kcal/mol, resulting in higher stability losses of 0.096 and 0.094.

This showed that the trained models were not able to come even considerably close to the required stability. In conclusion, the ‘Dense bi-LSTM’ and ‘Adaptive-ProtBert’ models showed slightly better results in CAI than their performance on stability.

Chapter 7

Conclusion and Future Work

The COVID-19 pandemic of 2019 highlighted the urgent need for rapid and effective vaccine development. We leveraged deep learning techniques to optimize codon sequences within the open reading frames (ORFs) of mRNA, focusing on enhancing their efficiency for therapeutic applications. This research demonstrated the potential of using advanced machine-learning models to push the boundaries of mRNA vaccine design.

We first explored the efficacy of the 'codon-box' method compared to the novel 'valid-codon' method and the baseline 'all-codon' method. Our findings showed that all three methods effectively avoided mutation errors, confirming the strength of the Dense bi-LSTM model in mapping amino acids to their correct synonymous codons. The 'valid-codon' method, which involved masking nonsynonymous codons, performed comparably to the 'codon-box' method. This result indicates that special grouping of codons, as done in the 'codon-box' method, is not significant and that simplifying the learning process allows the model to better capture codon usage bias, leading to enhanced expression levels.

Next, we applied a pre-trained protein language model, specifically ProtBert, for the downstream task of codon optimization. This work marked the first instance of using protein language models for this purpose. Despite using relatively less training data and fewer computational resources, the Adasel-ProtBert model outperformed both the baseline Dense bi-LSTM' model and clinically approved vaccines in terms of expression and stability in it's optimized ORF sequence for SARS-CoV-2 (Wuhan) spike protein. Additionally, all models produced optimized ORFs with higher CAI and MFE values compared to Wild-Type sequences across human, E. coli, and Chinese-Hamster genomes.

While we utilized NLP techniques for the codon optimization task in ORFs, we also encountered challenges that suggest avenues for future exploration. For example, our efforts to generate user-defined ORF with tunable CAI and stability using a multi-objective loss function revealed the complexities of achieving a balance between expression and stability. Similarly, our exploration of using CAI as a smooth loss function for regularization, although not yielding the desired results, provides a basis for future research to refine and enhance this approach.

7.1 Future Work

This research lays the foundation for several future directions. First, we could explore more advanced models, such as graph convolutional neural networks (GNNs) [24], which may better capture the structural intricacies of protein sequences. While our work focused on the computational design of optimal ORFs, it is essential to validate the biological efficacy of these sequences in real-world therapeutic contexts. Moreover, future research could expand beyond ORF optimization to address other components of mRNA vaccines, such as untranslated regions (UTRs), the 5' cap, and the length of the poly-A tail. Integrating these elements into a comprehensive optimization framework could significantly enhance the overall effectiveness of mRNA vaccines. In conclusion, this thesis demonstrated that language models, particularly those pre-trained on protein sequences, hold significance for therapeutic applications such as mRNA vaccine design. By optimizing codon sequences for human genes, this thesis provides valuable insight for future work in the field of mRNA therapeutics, with the potential to improve vaccine design and efficacy.

Bibliography

[1] S. Barazandeh, F. Ozden, A. Hincer, U. O. S. Seker, and A. E. Cicek, "Utrgan: Learning to generate 5'utr sequences for optimized translation efficiency and gene expression," *bioRxiv*, pp. 2023–01, 2023.

[2] S. K. Behura and D. W. Severson, "Codon usage bias: Causative factors, quantification methods and genome-wide patterns: With emphasis on insect genomes," *Biological Reviews*, vol. 88, no. 1, pp. 49–61, 2013.

[3] S. Belouzard, J. K. Millet, B. N. Licitra, and G. R. Whittaker, "Mechanisms of coronavirus cell entry mediated by the viral spike protein," *Viruses*, vol. 4, no. 6, pp. 1011–1033, 2012.

[4] S. E. Calvo, D. J. Pagliarini, and V. K. Mootha, "Upstream open reading frames cause widespread reduction of protein expression and are polymorphic among humans," *Proceedings of the National Academy of Sciences*, vol. 106, no. 18, pp. 7507–7512, 2009.

[5] S. Castillo-Hair *et al.*, "Optimizing 5'utrs for mrna-delivered gene editing using deep learning," *bioRxiv*, pp. 2023–06, 2023.

[6] S. M. Castillo-Hair and G. Seelig, "Machine learning for designing next-generation mrna therapeutics," *Accounts of Chemical Research*, vol. 55, no. 1, pp. 24–34, 2021.

[7] J. Chen *et al.*, "Interpretable rna foundation model from unannotated data for highly accurate rna structure and function predictions," *arXiv preprint arXiv:2204.00300*, 2022.

[8] R. Das, *OpenVaccine: COVID-19 mRNA vaccine degradation prediction*, 2020. [Online]. Available: https://kaggle.com/competitions/stanford-covid-vaccine.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[10] N. Ding *et al.*, "Parameter-efficient fine-tuning of large-scale pre-trained language models," *Nature Machine Intelligence*, vol. 5, no. 3, pp. 220–235, 2023.

[11] A. K. Dubey and V. Jain, "Comparative study of convolution neural network's relu and leaky-relu activation functions," in *Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018*, Springer, 2019, pp. 873–880.

[12] A. Elnaggar *et al.*, “Prottrans: Towards cracking the language of life’s code through self-supervised deep learning and high performance computing,” *bioRxiv*, 2020. DOI: 10.1101/2020.07.12.199554. [Online]. Available: https://www.biorxiv.org/content/early/2020/07/21/2020.07.12.199554.

[13] A. Elnaggar *et al.*, “Prottrans: Toward understanding the language of life through self-supervised learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 7112–7127, 2021.

[14] H. Fu *et al.*, “Codon optimization with deep learning to enhance protein expression,” *Scientific Reports*, vol. 10, no. 1, p. 17 617, 2020.

[15] H. Gong *et al.*, “Integrated mrna sequence optimization using deep learning,” *Briefings in Bioinformatics*, vol. 24, no. 1, bbad001, 2023.

[16] D. R. Goulet, Y. Yan, P. Agrawal, A. B. Waight, A. N.-s. Mak, and Y. Zhu, “Codon optimization using a recurrent neural network,” *Journal of Computational Biology*, vol. 30, no. 1, pp. 70–81, 2023.

[17] C. Gustafsson, S. Govindarajan, and J. Minshull, “Codon bias and heterologous protein expression,” *Trends in Biotechnology*, vol. 22, no. 7, pp. 346–353, 2004.

[18] P. J. Harrison *et al.*, “Deep-learning models for lipid nanoparticle-based drug delivery,” *Nanomedicine*, vol. 16, no. 13, pp. 1097–1110, 2021.

[19] R. He *et al.*, “On the effectiveness of adapter-based tuning for pretrained language model adaptation,” *arXiv preprint arXiv:2106.03164*, 2021.

[20] S. He, B. Gao, R. Sabnis, and Q. Sun, “Rnadegformer: Accurate prediction of mrna degradation at nucleotide resolution with deep learning,” *Briefings in Bioinformatics*, vol. 24, no. 1, bbac581, 2023.

[21] S. Hochreiter and J. Schmidhuber, “Long short-term memory. neural computation, 9 (8), 1735-1780,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] E. J. Hu *et al.*, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.

[23] Y. Huang, B. Niu, Y. Gao, L. Fu, and W. Li, “Cd-hit suite: A web server for clustering and comparing biological sequences,” *Bioinformatics*, vol. 26, no. 5, pp. 680–682, 2010.

[24] V. N. Ioannidis, A. G. Marques, and G. B. Giannakis, “Graph neural networks for predicting protein functions,” in *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, IEEE, 2019, pp. 221–225.

[25] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[26] R. Jain, A. Jain, E. Mauro, K. LeShane, and D. Densmore, “Icor: Improving codon optimization with recurrent neural networks,” *BMC Bioinformatics*, vol. 24, no. 1, p. 132, 2023.

[27] R. P. de Jongh, A. D. van Dijk, M. K. Julsing, P. J. Schaap, and D. de Ridder, "Designing eukaryotic gene expression regulation using machine learning," *Trends in Biotechnology*, vol. 38, no. 2, pp. 191–201, 2020.

[28] J. Jumper *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[29] O. Karaşan, A. Şen, B. Tiryaki, and A. E. Cicek, "A unifying network modeling approach for codon optimization," *Bioinformatics*, vol. 38, no. 16, pp. 3935–3941, 2022.

[30] Y.-A. Kim *et al.*, "Computational design of mrna vaccines," *Vaccine*, 2023.

[31] G. Kudla, L. Lipinski, F. Caffin, A. Helwak, and M. Zylicz, "High guanine and cytosine content increases mrna levels in mammalian cells," *PLoS Biology*, vol. 4, no. 6, e180, 2006.

[32] J. R. Lanzante, "Testing for differences between two distributions in the presence of serial correlation using the kolmogorov–smirnov and kuiper's tests," *International Journal of Climatology*, vol. 41, no. 14, pp. 6314–6323, 2021.

[33] K. Leppek *et al.*, "Combinatorial optimization of mrna structure, stability, and translation for rna-based therapeutics," *Nature Communications*, vol. 13, no. 1, p. 1536, 2022.

[34] S. Li *et al.*, "Codonbert: Large language models for mrna design and optimization," *bioRxiv*, pp. 2023–09, 2023.

[35] R. Lorenz *et al.*, "Viennarna package 2.0," *Algorithms for Molecular Biology*, vol. 6, pp. 1–14, 2011.

[36] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, Atlanta, GA, vol. 30, 2013, p. 3.

[37] R. Maharjan *et al.*, "Comparative study of lipid nanoparticle-based mrna vaccine bioprocess with machine learning and combinatorial artificial neural network-design of experiment approach," *International Journal of Pharmaceutics*, vol. 640, p. 123 012, 2023.

[38] J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, and A. Rives, "Language models enable zero-shot prediction of the effects of mutations on protein function," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 287–29 303, 2021.

[39] B. Min *et al.*, "Recent advances in natural language processing via large pre-trained language models: A survey," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–40, 2023.

[40] D. D. Nedialkova and S. A. Leidel, "Optimization of codon translation rates via trna modifications maintains proteome integrity," *Cell*, vol. 161, no. 7, pp. 1606–1618, 2015.

[41] E. Nijkamp, J. A. Ruffolo, E. N. Weinstein, N. Naik, and A. Madani, "Progen2: Exploring the boundaries of protein language models," *Cell Systems*, vol. 14, no. 11, pp. 968–978, 2023.

[42] R. J. Penić, T. Vlašić, R. G. Huber, Y. Wan, and M. Šikić, "Rinalmo: General-purpose rna language models can generalize well on structure prediction tasks," *arXiv preprint arXiv:2403.00043*, 2024.

[43] G. Puccetti, A. Miaschi, and F. Dell'Orletta, "How do BERT embeddings organize linguistic knowledge?" In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, E. Agirre, M. Apidianaki, and I. Vulić, Eds., Online: Association for Computational Linguistics, Jun. 2021, pp. 48–57. DOI: 10.18653/v1/2021.deelio-1.6. [Online]. Available: https://aclanthology.org/2021.deelio-1.6.

[44] X. Qi, T. Wang, and J. Liu, "Comparison of support vector machine and softmax classifiers in computer vision," in *2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, IEEE, 2017, pp. 151–155.

[45] T. E. Quax, N. J. Claassens, D. Söll, and J. van der Oost, "Codon bias as a means to fine-tune gene expression," *Molecular Cell*, vol. 59, no. 2, pp. 149–161, 2015.

[46] M. J. Ranaghan, J. J. Li, D. M. Laprise, and C. W. Garvie, "Assessing optimal: Inequalities in codon optimization algorithms," *BMC Biology*, vol. 19, pp. 1–13, 2021.

[47] A. Şen, K. Kargar, E. Akgün, and M. Ç. Pınar, "Codon optimization: A mathematical programing approach," *Bioinformatics*, vol. 36, no. 13, pp. 4012–4020, 2020.

[48] O. Sharma, A. A. Sultan, H. Ding, and C. R. Triggle, "A review of the progress and challenges of developing a vaccine for covid-19," *Frontiers in Immunology*, vol. 11, p. 585 354, 2020.

[49] P. M. Sharp and W.-H. Li, "The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications," *Nucleic acids research*, vol. 15, no. 3, pp. 1281–1295, 1987.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[51] M. Steinegger, M. Mirdita, and J. Söding, "Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold," *Nature Methods*, vol. 16, no. 7, pp. 603–606, 2019.

[52] B. E. Suzek, Y. Wang, H. Huang, P. B. McGarvey, C. H. Wu, and U. Consortium, "Uniref clusters: A comprehensive and scalable alternative for improving sequence similarity searches," *Bioinformatics*, vol. 31, no. 6, pp. 926–932, 2015.

[53] S. P. Teo, "Review of covid-19 mrna vaccines: Bnt162b2 and mrna-1273," *Journal of Pharmacy Practice*, vol. 35, no. 6, pp. 947–951, 2022.

[54] A. Vaswani *et al.*, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[55] W. Wang, S. Feng, Z. Ye, H. Gao, J. Lin, and D. Ouyang, "Prediction of lipid nanoparticles for mrna vaccines by the machine learning algorithm," *Acta Pharmaceutica Sinica B*, vol. 12, no. 6, pp. 2950–2962, 2022.

[56] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[57] D. K. Yang, S. L. Goldman, E. Weinstein, and D. Marks, "Generative models for codon prediction and optimization," *Machine Learning in Computational Biology*, 2019.