CodOpt: Enhancing Drug and Vaccine Development by Using Deep Learning and Natural-Language Processing to Optimize Recombinant Codon Sequences via a Host-Independent Data Pipeline

Bhushan Mohanraj
The Lawrenceville School

With the mentorship of

Dr. Chao Lu
Assistant Professor in the Department of Genetics and Development
Columbia University Irving Medical Center

Dr. Ryan Urbanowicz
Assistant Professor in the Department of Computational Biomedicine
Cedars-Sinai Medical Center

March 19, 2023

Abstract

Recombinant expression underpins the synthesis of crucial recombinant proteins, such as vaccines and pharmaceuticals that combat diseases. In recombinant expression, host organisms such as *E. coli* express foreign genes to produce the corresponding proteins. Codon optimization is the technique of selecting specific codons to maximize protein production, which can amplify expression hundreds of times by utilizing differences in frequency between synonymous codons. Effective codon optimization can save millions of lives by enabling timely and low-cost development of vaccines and pharmaceuticals, especially during outbreaks such as the COVID-19 pandemic that require rapid therapeutic design to fight disease. Most optimization algorithms, however, replace almost every codon with its most frequent alternative, causing cell stress and protein misfolding by ignoring the significance of rare codons. Since evolutionary pressures have tuned high-expression genes for both efficiency and safety, neural networks can address the drawbacks of common optimization techniques by emulating highly expression natural genes from host organisms. In this research, specialized, sequence-to-sequence neural networks were developed to learn codon-usage patterns from genomic DNA. Over 10 million genes were collected for three heterologous hosts, clustered with distance-based greedy clustering, and filtered according to their global codon adaptation indices (gCAI, a predictor of protein expression based on codon bias) to compile five thousand high-expression, non-redundant genes for training. Models based on CNN, RNN, and transformer architectures were trained to predict the genes' codon sequences from their amino-acid sequences, and the models' hyperparameters were tuned over multiple training rounds. The most successful architecture, the stacked LSTM, significantly increased the average gCAI of the testing sequences, from 0.600 to 0.949, outperforming gold-standard approaches based on codon-frequency distributions. Additionally, the GC content of the optimized sequences was evaluated to ensure the sequences' safety and efficacy. Analyzing the features learned by the stacked LSTM revealed that the model could recognize multiple evolutionary phenomena related to codon usage, such as that rare codons correlate with hydrophobic protein regions. For confirming the efficacy of the optimized sequences, the CUT&Tag protein pA-Tn5 was expressed in *E. coli* to compare an unoptimized sequence with a sequence optimized by the stacked LSTM. As revealed by staining and gel electrophoresis, the gene optimized by CodOpt achieved significantly higher expression than the original. These results indicate that codon optimization with deep learning can outperform traditional solutions to accelerate the timely and low-cost synthesis of vaccines and pharmaceuticals. For deploying the models publicly, a web application was built where researchers can generate optimized sequences for their recombinant proteins, enabling the rapid design and production of crucial therapeutics.

Contents

1 Background

Recombinant proteins underlie contemporary biotechnology and biomedicine. In biology, proteins are essential macromolecules with numerous roles, such as maintaining cell structure, transporting various molecules, and catalyzing biochemical reactions. Although all living organisms produce proteins naturally, specific proteins can be produced synthetically in host organisms such as bacteria and fungi. As the primary mechanism for synthetic protein production, recombinant DNA technology underlies the discovery and development of medical treatments such as vaccines and pharmaceuticals [1]. Recombinant proteins can be produced quickly, affordably, and safely and have vital roles in combating diseases [2]. Numerous vaccines and pharmaceuticals produced with recombinant DNA technology have addressed global health challenges and enabled rapid improvements in human life.

- Recombinant COVID-19 vaccines have been instrumental in mitigating COVID-19 worldwide. The Oxford–AstraZeneca and Novavax COVID-19 vaccines, two of the most widely distributed vaccines of the pandemic, were developed and produced with recombinant DNA technology [3]. One year after the introduction of the Oxford–AstraZeneca vaccine, two billion doses were distributed to over one hundred and seventy countries, saving millions of lives amid the COVID-19 pandemic [4].

- Recombinant granulocyte colony-stimulating factor (GCSF) has improved treatment for patients experiencing a variety of cancers. Since chemotherapy can dramatically reduce levels of white blood cells, which are crucial to human immune systems, recombinant GCSF enables cancer patients to recover faster and with fewer infections [5].

- Recombinant human insulin, an early product of biotechnology, has instrumentally advanced the mitigation of diabetes [5]. By replacing insulin extracted from cows and pigs, recombinant human insulin has enabled reliable and flexible treatment for millions of diabetes patients [6].

- Recombinant flu vaccines have advanced the mitigation of influenza. Recombinant flu vaccines are far less expensive and faster to produce than egg-based and culture-based vaccines, as recombinant DNA technology bypasses the specialized requirements of manufacturing vaccines in eggs and cell cultures [7].

- Recombinant human growth hormone (HGH) has transformed the treatment of growth hormone deficiency. With recombinant HGH, children who experience this deficiency can achieve typical growth to avoid dwarfism and related conditions later in life [5].

Since recombinant proteins underlie the production of vaccines and pharmaceuticals, they are fundamental to mitigating diseases such as cancer, diabetes, and COVID-19. Therefore, efficiently producing recombinant proteins is essential to global health.

1.1 Genes and Proteins

Genes are the fundamental blueprints that define the development and operation of every living organism. Each gene contributes to a particular trait, such as human eye color or blood type. The transfer of genes from parents to children causes the inheritance of traits between generations.

1.1.1 DNA

In all living organisms, long molecules of *DNA* located in the centers of cells define genes. DNA molecules consist of smaller *nitrogenous bases* that determine the DNA molecule's genes. Each nitrogenous base is either adenine (A), cytosine (C), guanine (G), or thymine (T). The four bases form complementary bonding pairs: adenine and thymine can bond, and cytosine and guanine can bond.

Each DNA molecule consists of two strands of nucleotides: the coding strand and the template strand. The coding strand has a sequence of bases that defines multiple genes, while the template strand contains the bases complementary to the coding bases. For example, suppose that a coding strand includes the sequence "ACAG." Then, the corresponding template strand must have the complementary sequence "TGTC." Each gene corresponds to a sequence of bases within a coding strand of DNA. The sequence for each gene instructs the cells of an organism to produce a specific protein. Proteins are diverse macromolecules that perform the actions necessary to sustain life. They consist of smaller *amino acids*, which have twenty possibilities. The nitrogenous bases in a gene define the sequence of amino acids in the corresponding protein.

Through the cellular processes of transcription and translation, cells produce proteins according to the genes defined by their DNA.

1.1.2 Transcription and Translation

In transcription, cells copy a gene from a coding strand of DNA to produce *messenger RNA* (mRNA), a single isolated strand of nitrogenous bases. Cells replicate the coding strand by traversing the template strand and collecting a sequence of complementary bases, which will match the coding sequence. (Adenine, cytosine, and guanine occur in both DNA and RNA. Instead of thymine, mRNA contains uracil, a slightly different nitrogenous base.) The mRNA molecules produced by transcription enable the production of proteins in translation.

Ribosomes, structures found in the cytoplasmic fluid of every cell, facilitate protein production using the genetic information in mRNA. Each *codon* in an mRNA strand, a group of three consecutive nitrogenous bases, corresponds to one amino acid. As ribosomes traverse an mRNA strand, they facilitate the binding of *transfer RNA* (tRNA) molecules to each codon. Each tRNA molecule carries one amino acid and contains the codon *complementary* to the mRNA codon for that amino acid. When a ribosome encounters an mRNA codon, the corresponding tRNA molecule binds to the mRNA strand at its complementary codon.

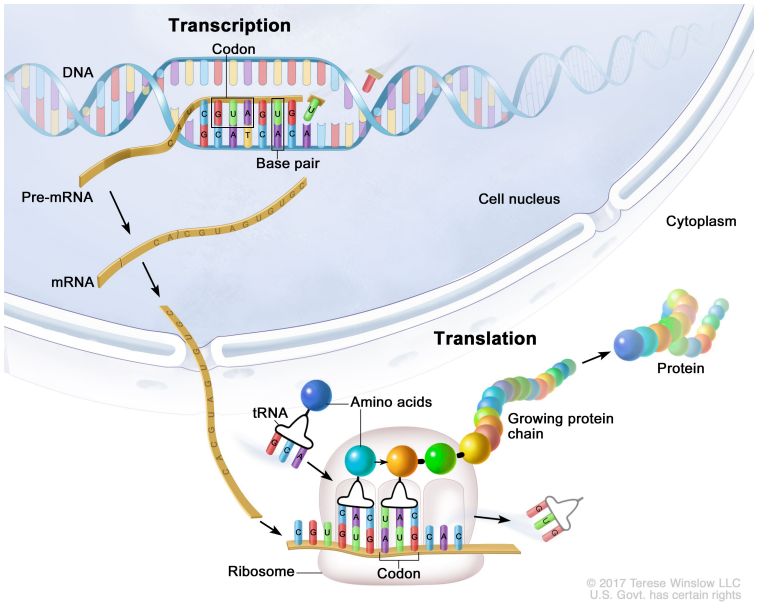


Figure 1: Transcription and translation in a eukaryotic cell [8].

As tRNA molecules bind at a ribosome, their amino acids merge into a developing *polypeptide chain*, a sequence of amino acids that will become a protein. After the amino acid detaches from the tRNA molecule, the tRNA molecule separates from the mRNA strand. Then, the ribosome moves to the next codon and continues growing the polypeptide chain. The polypeptide chain gradually folds into a fully functional protein during and after translation. Segments of the polypeptide chain fold into *secondary structures*, which fold into the final *tertiary structure* of the protein. These folded proteins may undergo further *post-translational modifications* before becoming fully functional.

The production of proteins from a gene—which requires transcription, translation, and protein folding—is *gene expression*.

Second letter

First letter	U	C	A	G	Third letter
U	UUU, UUC Phe; UUA, UUG Leu	UCU, UCC, UCA, UCG Ser	UAU, UAC Tyr; **UAA Stop**, **UAG Stop**	UGU, UGC Cys; **UGA Stop**; UGG Trp	U C A G
C	CUU, CUC, CUA, CUG Leu	CCU, CCC, CCA, CCG Pro	CAU, CAC His; CAA, CAG Gln	CGU, CGC, CGA, CGG Arg	U C A G
A	AUU, AUC, AUA Ile; **AUG Met**	ACU, ACC, ACA, ACG Thr	AAU, AAC Asn; AAA, AAG Lys	AGU, AGC Ser; AGA, AGG Arg	U C A G
G	GUU, GUC, GUA, GUG Val	GCU, GCC, GCA, GCG Ala	GAU, GAC Asp; GAA, GAG Glu	GGU, GGC, GGA, GGG Gly	U C A G

Figure 2: The standard genetic code, which defines the amino acid for each codon [9].

1.2 Heterologous Expression

Recombinant proteins, including essential vaccines and pharmaceuticals, are produced through *heterologous expression*, where a *host organism* expresses foreign genes introduced through recombinant DNA technology. Typical heterologous hosts include bacteria, such as *Escherichia coli*; yeast, such as *Saccharomyces cerevisiae*; and mammalian cells, such as Chinese hamster ovary (CHO) cells [5]. A key challenge in applying heterologous expression is achieving consistent, high expression, which reduces the time required to produce proteins and the resources needed to sustain host cells.

1.2.1 Codon Bias

A key consideration in heterologous expression is the host's *codon bias*. During translation, ribosomes in cells traverse mRNA strands and translate each codon via a complementary tRNA molecule. With four different nitrogenous bases, sixty-four possible codons occur in sequences. Three of these codons are "stop codons" that indicate the end of translation, with rare exceptions. The remaining sixty-one codons each correspond to one amino acid. Since more codons exist than the twenty possible amino acids, some amino acids are encoded by multiple codons (*synonymous codons*).

CCA(P) 0.83%	CGA(R) 0.37%	CAA(Q) 1.48%	CTA(L) 0.38%
CCG(P) 2.28%	CGG(R) 0.60%	CAG(Q) 2.94%	CTG(L) 5.22%
CCT(P) 0.72%	CGT(R) 2.03%	CAT(H) 1.27%	CTT(L) 1.13%
CCC(P) 0.57%	CGC(R) 2.12%	CAC(H) 0.93%	CTC(L) 1.08%
GCA(A) 2.04%	GGA(G) 0.86%	GAA(E) 3.91%	GTA(V) 1.08%
GCG(A) 3.27%	GGG(G) 1.15%	GAG(E) 1.81%	GTG(V) 2.62%
GCT(A) 1.53%	GGT(G) 2.43%	GAT(D) 3.24%	GTT(V) 1.82%
GCC(A) 2.57%	GGC(G) 2.87%	GAC(D) 1.92%	GTC(V) 1.53%
ACA(T) 0.78%	AGA(R) 0.25%	AAA(K) 3.35%	ATA(I) 0.50%
ACG(T) 1.49%	AGG(R) 0.16%	AAG(K) 1.08%	ATG(M) 2.74%
ACT(T) 0.90%	AGT(S) 0.93%	AAT(N) 1.87%	ATT(I) 2.97%
ACC(T) 2.33%	AGC(S) 1.62%	AAC(N) 2.16%	ATC(I) 2.43%
TCA(S) 0.77%	TGA(*) 0.13%	TAA(*) 0.21%	TTA(L) 1.38%
TCG(S) 0.90%	TGG(W) 1.52%	TAG(*) 0.03%	TTG(L) 1.31%
TCT(S) 0.86%	TGT(C) 0.52%	TAT(Y) 1.63%	TTT(F) 2.22%
TCC(S) 0.90%	TGC(C) 0.64%	TAC(Y) 1.20%	TTC(F) 1.60%

Figure 3: The codon bias for a collection of *Escherichia coli* genomes [10].

Synonymous codons appear equivalent because interchanging them does not affect the sequence of amino acids defined by a gene. However, these synonymous codons have different frequencies in genomic DNA. This phenomenon of *codon bias* has been observed in many organisms [11]. As shown in figure 3, for example, the codons "CTG" and "CTA" both encode the amino acid leucine. In the genome of *Escherichia coli*, however, "CTG" occurs almost fifteen times as often as "CTA" [12].

In unicellular organisms, the frequency of each codon correlates with the abundance of the corresponding tRNA molecule [11]. Therefore, the codons used by a gene can affect its expression level: the translation of a rare codon requires a rare tRNA molecule to bind to an mRNA strand, increasing translation time and reducing gene expression. In prokaryotic organisms, genes with high expression levels contain few rare codons [11]. This correlation demonstrates that replacing rare codons with frequent codons can increase expression levels.

1.2.2 Global Codon Adaptation Index

CAI The *codon adaptation index* (CAI) of a genetic sequence indicates how closely the sequence's codons match the frequent codons within a host genome [13]. Since frequent codons correlate with high natural expression, CAI is used extensively to predict the expression of recombinant genes [11]. CAI is the gold standard among algorithms that model expression levels according to codon usage.

As defined by Sharp and Li [13], CAI is calculated using a *reference set* of highly expressed genes from the host. For any codon c, let $f(c)$ be its number of occurrences in the reference set, and let $A(c)$ be the set containing its synonymous codons. Then, the *relative adaptiveness* or *weight* of the codon, $w(c)$, is the ratio between its frequency and the highest frequency among its synonymous codons,

$$w(c) = \frac{f(c)}{\max\limits_{c' \in A(c)} f(c')}.$$

For any sequence s that contains n codons, c_1 to c_n, the CAI of the sequence is the geometric mean of its codons' weights,

$$\mathrm{CAI}(s) = \sqrt[n]{\prod_{i=1}^{n} w(c_i)}.$$

gCAI The *global codon adaptation index* algorithm (gCAI) addresses drawbacks of the CAI algorithm. While calculating CAI requires a predetermined reference set of highly expressed genes, the gCAI algorithm recursively determines a reference set using only a single sequenced genome from the host organism. Determining a reference set without expression data is crucial because some recombinant hosts may have no publicly available expression data for reserachers to determine a CAI reference set explicitly.

The recursive algorithm begins by considering the entire genome as a reference set S_1. Then, for each set S_k, set S_{k+1} is determined by calculating the gCAI for all genes and selecting either the top $\frac{|S_k|}{2}$ genes or the top $\frac{|S_1|}{100}$ genes, whichever set is larger. The algorithm converges when $S_{k+1} = S_k$, eventually creating a reference set with $\frac{|S_1|}{100}$ genes, or 1% of the original genome. (This convergence occurred for all fifteen prokaryotic and eukaryotic genomes originally tested.)

Additionally, the gCAI algorithm adjusts codon weights to account for codon usage in eukaryotic organisms. In eukaryotes, there exists a negative correlation between gene length and codon bias, so rare codons can occur at relatively high frequencies in longer genes. If a reference set contains multiple long genes, this negative correlation causes unusually high

CAI weights for rare codons. To address this complication, gCAI codon weights are lowered for codons that occur only in some reference genes.

Consider some reference set S. For any codon c, let $f(c)$ be its number of occurrences in the reference set, and let $A(c)$ be the set containing its synonymous codons. Furthermore, let $S_c \subseteq S$ be the set containing all reference sequences in which c occurs. Then, the *weight* of the codon, $w_g(c)$, is

$$w_g(c) = \frac{|S_c|}{|S|} \cdot \frac{f(c)}{\max\limits_{c' \in A(c)} f(c')}.$$

For any sequence s that contains n codons, c_1 to c_n, the gCAI of the sequence is the geometric mean of its codons' weights,

$$\text{gCAI}(s) = \sqrt[n]{\prod_{i=1}^{n} w_g(c_i)}.$$

1.3 Codon Optimization

When using genetic sequences for heterologous expression, *codon optimization* enhances protein production by increasing translational efficiency. Since rare codons act as bottlenecks during translation, replacing them with frequent codons can alleviate slowdowns in translation, increasing protein production. (Since frequent codons have greater CAI weights than rare codons, this replacement increases CAI and thus enhances gene expression.) Among the sequence-dependent properties that affect heterologous expression, codon usage correlates most closely with protein expression [14]. Therefore, optimizing the codons of genetic sequences is essential to achieving efficient heterologous expression.

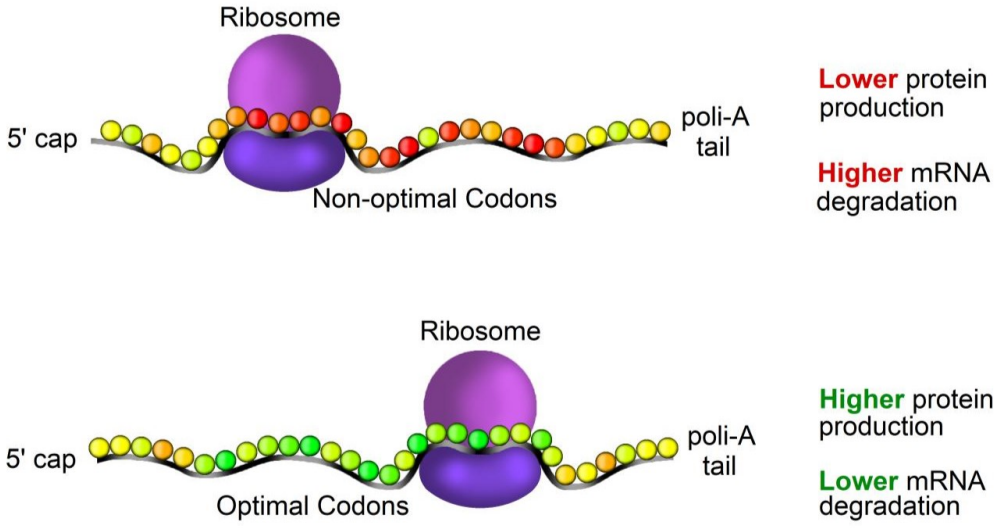


Figure 4: Codon optimization enables improved protein expression [15].

1.3.1 Standard Techniques

Researchers have developed many algorithms and commercial services for optimizing codon sequences to improve expression [16]. The most common algorithms for codon optimization replace each codon with its most frequent synonymous alternative [16]. Despite the enhancements offered by codon optimization, these standard techniques ignore the unanticipated consequences of eliminating rare codons.

Standard techniques assume that using frequent codons alone will maximize expression levels. However, this strategy depletes the corresponding tRNA molecules, generating an unbalanced tRNA pool [17]. This imbalance induces metabolic stress in the host cells and inhibits their growth [17]. Therefore, the exclusive use of frequent codons can impede heterologous protein production by causing tRNA imbalance and hindering cell growth and proliferation.

Furthermore, standard techniques assume that interchanging synonymous codons does not affect the resulting proteins. Although rare codons can slow translation, these slowdowns are often essential to protein folding [16]. Therefore, interchanging synonymous codons can induce protein misfolding, impairing protein stability and function [16]. Thus, using only frequent codons disrupts the creation of functional proteins by ignoring the role of rare codons in translation.

1.3.2 Impacts of Protein Misfolding on Vaccines and Pharmaceuticals

Standard codon optimization techniques impair the efficient production of functional proteins by inducing metabolic stress and protein misfolding in hosts. These consequences can damage therapeutic proteins such as vaccines and pharmaceuticals [18]. When patients receive medication, misfolded recombinant proteins can trigger the human immune system to produce anti-drug antibodies (ADAs) that impair the efficacy of functional proteins [19].

For example, *erythropoietin* (EPO) is a natural hormone that stimulates the production of red blood cells (RBCs). Recombinant human EPO can address anemia, a disease where the blood cannot deliver enough oxygen to cells. However, misfolded EPO can trigger the production of ADAs that neutralize functional EPO, natural or recombinant, hindering RBC production [19]. Similarly, misfolded proteins for the following pharmaceuticals have triggered anti-drug antibodies that impair patient treatment [19].

- Recombinant *interferon beta* for treating the autoimmune disease multiple sclerosis.
- Recombinant *megakaryocyte growth and differentiation factor* (MGDF) for treating the blood disease thrombocytopenia.

When using codon optimization, however, researchers may only recognize the consequences of protein misfolding during late-stage clinical trials or after launching a medication [18]. Therefore, algorithms for codon optimization must increase expression levels while maintaining the structural and functional integrity of the recombinant proteins produced by heterologous expression.

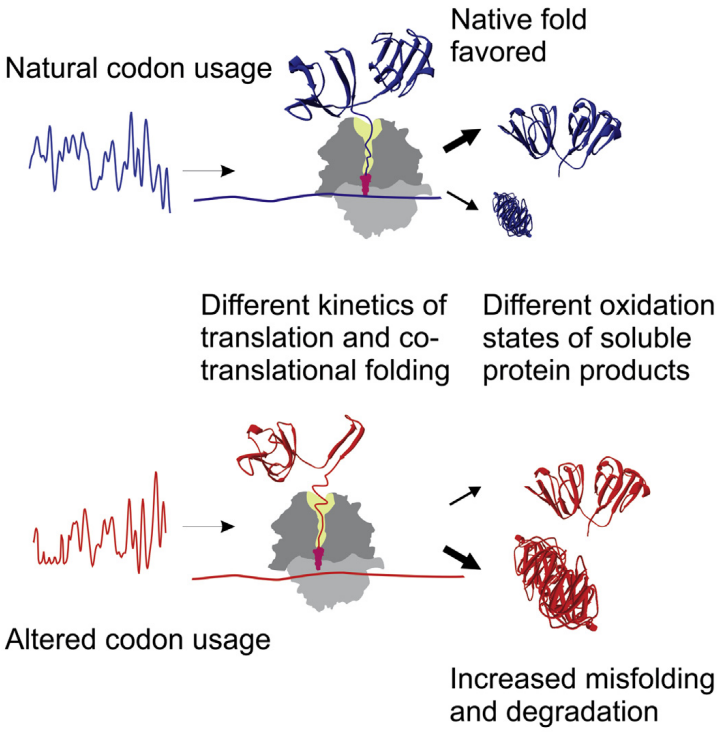


Figure 5: Altering codon sequences can induce misfolding [20].

2 Research Questions and Hypotheses: A Deep Learning Approach

The field of *machine learning* (ML) enables computers to create predictions and decisions after learning from data. *Deep learning* (DL) extends machine learning with *neural networks* (NNs), computational models inspired by the human brain. Deep learning can mitigate the challenges of codon optimization by understanding the contextual codon usage in genomic sequences that contribute to both protein expression and protein stability.

- How can deep learning amplify recombinant protein expression many times by emulating highly expressed genes to optimize codon sequences? Which sequence-to-sequence model architecture will best improve predicted protein expression?
- By learning evolutionary patterns embedded in high-expression genes and emulating them, how can deep learning address the drawbacks of common optimization techniques, such as metabolic stress and protein misfolding?
- How can a web application give researchers around the globe access to efficient codon optimization tools to accelerate vaccine and pharmaceutical development and save lives?

Deep-learning models extract features from data at multiple levels of abstraction [21]. Although other artificial intelligence techniques require manual feature engineering, neural networks extract features directly from data and discover patterns that human analysis cannot detect [22]. Therefore, neural networks could analyze large datasets of genomic sequences to learn the contextual usage of different codons. In particular, evolutionary pressure has tuned natural genomic sequences for the efficient production of stable proteins. Standard optimization algorithms, however, disregard evolutionary patterns because they cannot comprehend high-level abstractions [16]. By understanding these abstractions, neural networks could learn to recognize essential sequence properties that contribute to protein stability. With this understanding, deep learning can address the drawbacks of standard optimization techniques, such as metabolic stress and dangerous protein misfolding.

Neural networks are computational models that underpin deep learning and are inspired by the brain. Each neural network contains *neurons*, computational units with numerical values determined by their connections. Neurons occur in successive *layers* that transform data from previous layers and send them to subsequent layers. During *forward propagation*, a neural network generates predictions for input data by calculating the numerical values for each layer of neurons. The network calculates the values for each layer according to the *weights* of the connections between the layers. (Connections with more weight affect the calculations for a neuron more than those with less weight.) Each layer of the network generates a more abstract representation of the data until the output layer produces a prediction.

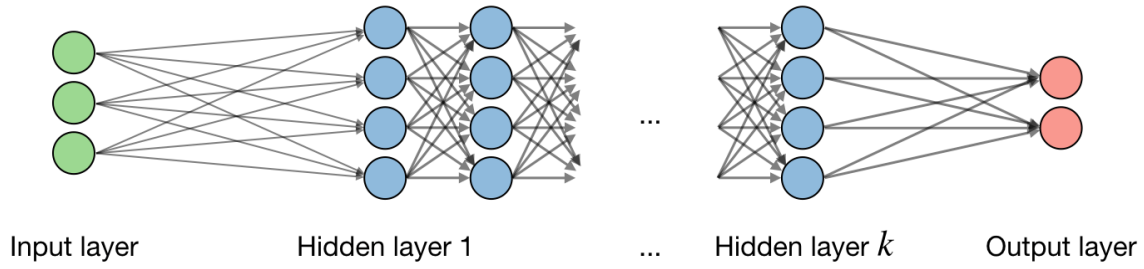


Figure 6: A standard deep neural network, with multiple hidden layers of neurons that are connected to previous and subsequent layers [23].

Neural networks are trained according to their performance on example data. Training a neural network requires a dataset containing input and expected output examples. For example, the widely used *MNIST* dataset includes images of numbers and a label from zero to nine for each image [24].

During training, the weights of a neural network receive random initial values. Then, the network's predictions for the input examples are compared to the expected outputs. The difference between the predicted and expected results are measured with a *loss function*. The network's weights are adjusted through a *backpropagation* algorithm to reduce the loss function gradually, improving the network's performance. This adjustment is performed for every input and expected output within the dataset. After training ends, the network's weights, which capture the abstractions learned during training, are saved. With these weights, the neural network can generate predictions for new data.

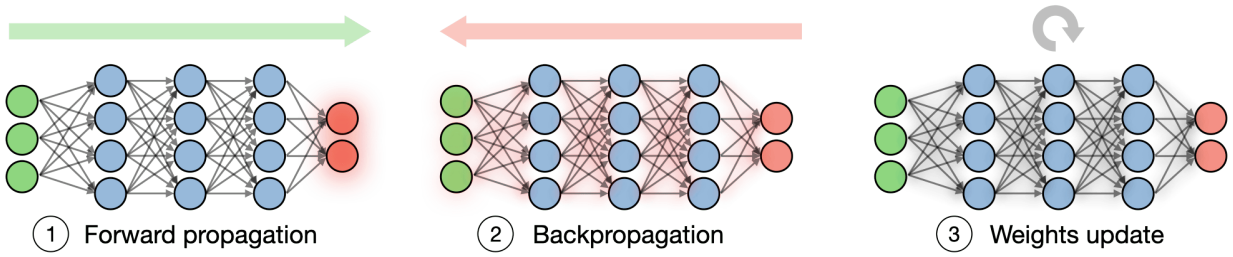


Figure 7: The training of a neural network, where the network generates predictions in forward propogation, its output is analyzed in backpropogation, and its weights are updated [25].

Neural network *architectures* are specific arrangements of the neurons and connections within a network. Various architectures have been designed to analyze particular data types and underpin different deep learning subfields, such as computer vision (CV) and natural language processing (NLP). The techniques used for CV, NLP, and transformer models can apply to codon optimization.

Computer Vision Convolutional neural networks (CNNs) are applied in computer vision to analyze images and detect their features.

CNNs analyze images with *spatial invariance*, as they can discern shapes and features throughout a picture rather than only one section. This spatial invariance can help optimize genetic sequences because each amino acid can occur at any location. Therefore, models that analyze sequences should learn the contextual occurrence of amino acids from training sequences and apply this understanding to new sequences where those amino acids occur.

Furthermore, successive convolutional layers extract progressively more abstract features from more expansive areas of an image. This progressive understanding applies to genetic sequences because amino-acid chains fold into secondary and tertiary structures, and the codons throughout a chain affect its folding. Therefore, models that analyze sequences should understand their features at progressively larger scales.

Natural Language Processing Recurrent neural networks (RNNs) apply to natural language processing and other sequence analyses, such as predicting stock prices over time.

RNNs use contextual understanding to generate predictions. For example, consider an RNN that translates English to Spanish. Although an English word may have different Spanish translations, the model can recognize which translations are incorrect given the surrounding phrases and sentences. This contextual knowledge applies to the optimization codon sequences because rare codons occur according to specific contextual requirements.

Unidirectional RNNs use previous sequence data to generate a prediction for each value. *Bidirectional RNNs*, however, use both previous and subsequent data. This research compares both unidirectional and bidirectional RNNs for codon optimization. Unlike unidirectional networks, bidirectional networks can analyze the features before and after each amino acid. However, unidirectional models reflect the biological fact that codon sequences are translated in one direction. Many sequence properties relate to this unidirectional translation. For example, rare codons enable co-translational protein folding, which is crucial

for protein production and begins before an entire polypeptide chain is constructed. Thus, unidirectional RNNs can identify significant features of a sequence without knowing its later codons. Therefore, building both unidirectional and bidirectional RNNs offers a valuable conceptual comparison.

Transformers Transformer models, devised in 2017 by AI researchers at Google, have revolutionized artificial intelligence and sparked exponential growth in the size and capabilities of preeminent neural networks. Originally developed as an alternative to RNNs, transformers have gained prominence in all subfields of artificial intelligence, including natural-language processing and computer vision. Furthermore, transformers underlie groundbreaking systems such as the famed ChatGPT chatbot developed by OpenAI. Transformers use *attention mechanisms* to understand the significance of each element within a sequence. Attention mechanisms were originally developed to improve the performance of RNN models for long sequences. However, researchers have found that using attention mechanisms alone, without recurrent layers, can surpass the performance of RNNs with attention mechanisms and achieve significantly lower training times. Thus, the transformer architecture uses only attention mechanisms and feedforward layers for sequence processing.

Hypotheses With neural networks such as CNNs, RNNs, and transformers, deep learning can address the drawbacks of standard techniques for codon optimization. Codon optimization enables efficient heterologous expression for producing recombinant proteins. However, traditional optimization techniques cause disruptive protein misfolding and metabolic stress. In medications, misfolded proteins harm patients by fostering anti-drug antibodies that counteract natural and recombinant proteins. Therefore, novel techniques for codon optimization must enhance expression without these detrimental consequences.

- By learning to predict the genes for highly expressed natural proteins and applying this capability to recombinant proteins, deep learning can achieve the same expression levels as natural highly expressed genes. Furthermore, by avoiding tRNA imbalance and metabolic stress that damage host cells, neural networks can surpass the expression levels achieved by common optimization techniques.

- Evolution has embedded information in natural genes that ensures protein stability and prevents protein misfolding. Standard optimization algorithms, however, disregard this information because they cannot comprehend high-level abstractions. By emulating patterns embedded within these genes by evolutionary pressure, deep learning can address the metabolic stress and protein misfolding that current solutions cause.

- An accessible and simple web application can provide global access to codon optimization tools based on deep learning. Since deep learning models are speedy, such a web application could quickly provide researchers with optimized sequences for different heterologous hosts. With these sequences, researchers could accelerate their work in developing and testing new medicines; labs and companies could accelerate the production of crucial recombinant proteins that save lives.

3 Methods

Neural networks for codon optimization were built for three standard heterologous hosts: *Escherichia coli*, baker's yeast (*Saccharomyces cerevisiae*), and Chinese hamster ovary (CHO) cells (*Cricetulus griseus*). The models were constructed through a *host-independent* pipeline that enables researchers to build similar models for codon optimization with new heterologous hosts. The pipeline begins with DNA sequences from publicly available genomes of the host species and performs the following steps to build a neural network.

- Clustering the sequences to remove redundancy between the different genomes for each host organism and choosing centroid sequences from the clusters.
- Validating the sequences to ensure they have no ambiguous bases, have valid start codons and stop codons, and are neither predicted nor hypothetical genes.
- Filtering the sequences according to their predicted expression levels (gCAIs) and constructing a training dataset of those sequences with gCAIs above the 80th percentile.
- Building and training neural networks to predict the codon sequences in the training dataset from their amino-acid sequences.

3.1 Datasets

Sequence data from the NCBI Assembly database were used to train and test the deep-learning models. The NCBI Assembly database includes genetic sequencing data for hundreds of thousands of organisms [26]. The assemblies contain the sequencing data from experiments performed by researchers and have varying levels of detail and annotation.

For each host species, available assemblies were downloaded from the Assembly database. Although the database provides complete data from each sequencing experiment, only the genomic coding sequences were used for training the models, as the models optimize codon sequences for translation into proteins. Therefore, the coding sequences for each assembly were downloaded in the FASTA file format.

Escherichia coli, the most extensively studied model organism, had over 190,000 distinct Assembly entries. Due to the size of the sequence data for these entries, the download was restricted to the complete genomes available in the NCBI RefSeq database, which the NCBI curates to have less redundancy than the general Assembly database. This download consisted of over two thousand genomes with over ten million sequences, still more than all available data for the other host species.

3.2 Technology

3.2.1 Models

All data analyses were performed with the Python programming language, which has readable syntax and extensive libraries for data science and deep learning. Data analysis and modeling were performed with the Python libraries Biopython (version 1.79), Matplotlib (version 3.6), and TensorFlow (version 2.10).

Biopython is a comprehensive library for processing and manipulating various biological data, such as sequences, sequence annotations, and expression data [27]. Biopython was used to parse FASTA files, which store genetic sequences and their metadata.

Matplotlib is a library used extensively in data science to visualize and plot data [28]. Using an array of data, Matplotlib can produce various plots, such as violin plots for one-dimensional distributions and line graphs for two-dimensional relationships. Matplotlib was used to create violin plots of gCAI distributions and other plots of network performance.

TensorFlow is a comprehensive library for accelerated numerical computations and for building neural networks [29]. TensorFlow uses the CUDA toolkit to accelerate array computations with graphics processing units (GPUs) that specialize in parallel computation (performing multiple numerical operations simultaneously). TensorFlow functions and the Keras API were used to manipulate arrays of sequence data and build neural networks.

3.2.2 Web Application

The web application was developed with the Python programming language and the Jinja templating engine. The web application was built with the Python libraries Flask (version 2.1.1), SQLAlchemy (version 1.4.46), and WTForms (version 3.0.1).

3.3 Preprocessing Genomic Data

3.3.1 Clustering Sequences

The models were trained using publicly available genomes from multiple organisms for each host species. Since these genomes contain similar sequences, their data were clustered to remove redundancy in the dataset. Various tools exist for clustering genetic sequences, such as VSEARCH, USEARCH, and CD-HIT-EST [30, 31, 32]. VSEARCH was used because of its efficient implementation and ability to integrate with self-managed computing environments.

VSEARCH was utilized to cluster the Assembly sequences for each host: sequences with pairwise similarities above 90% were clustered together. The centroid sequences, one from each cluster, were then stored in a FASTA format with their original metadata. These centroid sequences formed the basis of the training datasets for the neural networks.

3.3.2 Validating Sequences

After clustering the genomic sequences and retrieving the corresponding centroid sequences, the centroid sequences were validated before being used to train the deep-learning models. Any centroid sequence that did not satisfy the following criteria was deemed invalid.

- *No ambiguous bases.* After genomic sequencing, some nitrogenous bases remain ambiguous. The Assembly data indicates these ambiguities with specific IUPAC (the International Union of Pure and Applied Chemistry) codes. For example, the IUPAC code "R" indicates an ambiguous base that is either adenine (A) or guanine (G). Sequences with ambiguities cannot be represented as arrays, so they were deemed invalid.

- *No invalid codons.* The length of every genetic sequence must be a multiple of three, as codons contain three bases. Furthermore, every sequence must have a valid start

codon and stop codon for the host species. Any sequence without an acceptable length or valid start and stop codons, possibly due to sequencing errors, was deemed invalid.

- *Neither a pseudogene nor encoding for a hypothetical protein.* Some of the Assembly sequences had annotations provided by NCBI. These annotations indicated whether each sequence was a *pseudogene* (a nonfunctional segment of DNA) or encoded a *hypothetical protein* (a protein predicted to exist from sequencing alone). Any pseudogene or hypothetical protein sequence was deemed invalid.

3.3.3 Filtering Sequences

After validation, the sequences for each host were filtered according to their predicted expression levels to collect a dataset for training the neural networks. The global codon adaptation index (gCAI) was calculated for each validated sequence to predict its expression level. These gCAI values were collected and sorted, and the 80th percentile for gCAI for the validated sequences was calculated. The modeling dataset was constructed with every sequence whose gCAI was above the 80th percentile. This collection produced a dataset of highly expressed genes from which the neural networks could learn.

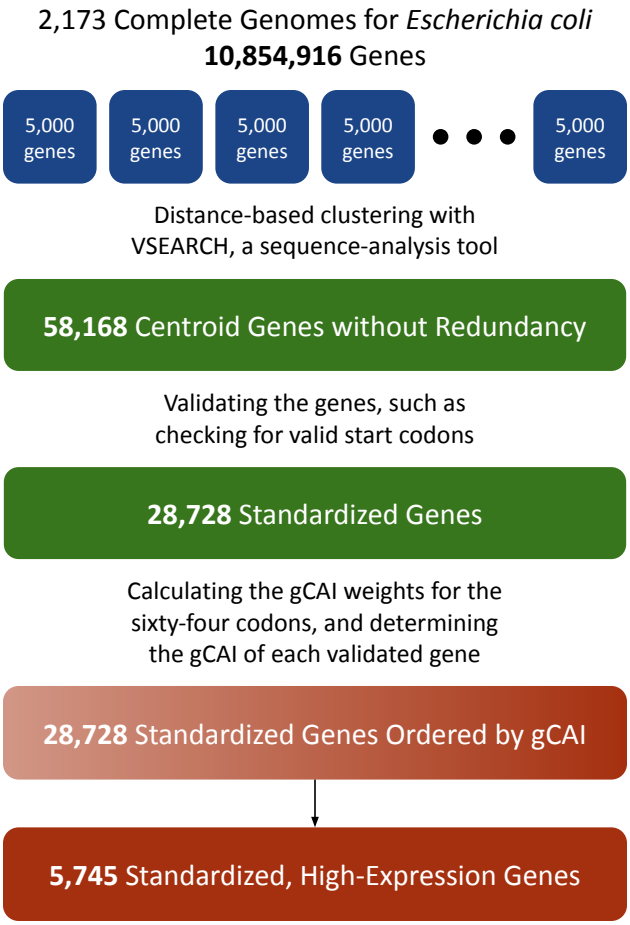


Figure 8: The pipeline for distilling millions of genes into a standardized training dataset.

3.4 Building and Training Models

With this dataset of highly expressed genes, neural networks of various architectures were built for codon optimization. Each neural network accepts a sequence of amino acids as input and outputs the predicted, optimal codon sequence.

The networks train by learning to emulate the highly expressed genes. Evolution has optimized these genes to both ensure efficient protein production and prevent dangerous protein misfolding. Therefore, emulating the contextual codon usage of these genes can enhance heterologous expression without the drawbacks of standard optimization techniques, such as protein misfolding and metabolic stress. By training to predict codon sequences from amino-acid sequences, the networks must learn the contextual significance of rare codons to determine where those rare codons should occur.

Using the amino-acid sequences of recombinant proteins, the networks can then generate optimized codon sequences that resemble natural, highly expressed genes. Multiple networks with different architectures were built for systematically comparing their abilities to predict the codon sequences of highly expressed genes.

3.4.1 Architectures

Each neural network for codon optimization begins with an input layer that accepts an array of one-hot encoded amino acids. A one-hot encoded array of amino acids is an array of vectors that contain 20 numbers each. The vector for a given amino acid contains a 1 at the index (from 1 to 20) for that amino acid and a 0 at every other index.

Each neural network generates *logits* for the sixty-four possible codons. However, only some of these codons encode each amino acid, while the others would incorrectly produce a different protein sequence. Therefore, each neural network includes a restriction layer that excludes the invalid codons for each amino acid. Although the neural networks learn the genetic code during training, the output space should be restricted to prevent rare incorrect substitutions. The restriction layer sets the logit for each invalid codon to the minimum possible value. (For 32-bit floating-point numbers, this minimum is approximately $-2^{128} \approx -3.4 \cdot 10^{38}$.) Under the softmax function, which converts logits to probabilities, these minimized logits become zero, eliminating invalid codons from the output space.

Convolutional Neural Networks A sequence-to-sequence CNN was built for codon optimization using convolutions with strides of one. Pooling layers were avoided because the input sequences have variable lengths, so transposed convolutions would generate an incorrectly sized output due to rounding. With multiple convolutions, the neurons in subsequent layers have larger receptive fields and can capture more sequence features. Inspired by the UNet architecture [33], the network includes skip-layer connections from earlier convolutional layers to later ones. These connections reveal the ordering of the initial sequence to the last layers that build the output. After the hidden layers, the network generates logits for the sixty-four codons with a final convolution.

Recurrent Neural Networks Multiple RNNs were built for codon optimization, including a recurrent neural network (RNN), a gated recurrent unit (GRU) network, and a long

short-term memory (LSTM) network. Each RNN contains a computational *cell* that processes sequence elements according to its internal weights and has a dynamic *hidden state* that stores the memory of the cell. When an RNN cell analyzes a sequence, it merges the new elements into its previous hidden state. This recurrent cycle allows the cell to remember significant features over time, a crucial capability for analyzing sequence relationships. For codon optimization, remembering the surrounding amino acids is essential for determining the optimal codons, as different codons occur in particular contexts.

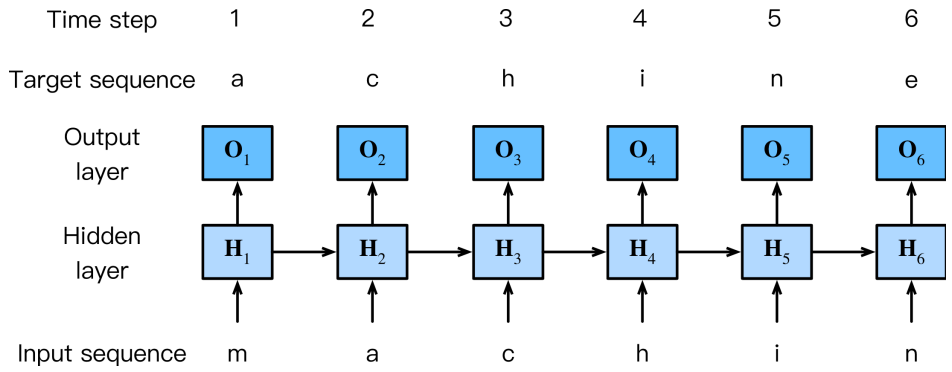


Figure 9: The operation of a standard RNN, where the cell processes each element of an input sequence and maintains an internal state for storing previous information [34].

The long short-term memory (LSTM) and gated recurrent unit (GRU) architectures extend the standard RNN cell to achieve better memory. Each LSTM cell includes a *forget gate*, an *input gate*, and an *output gate*. These computational units regulate how data is incorporated into the state of the cell. When the cell encounters new data, the forget gate determines whether the previous state is unnecessary, the input gate determines how the cell state should change, and the output gate determines the output for the new data. Each gate has internal weights that are trained over time. These gates enable LSTM networks to retain information when processing long sequences. GRU cells resemble LSTM cells but have fewer operations, making them less computationally intensive. Each GRU cell has two gates that update the cell state and choose which information to ignore. Like LSTM networks, GRU networks can recall information in long sequences better than standard RNNs. This ability is essential to codon optimization because amino acids far apart can interact heavily during protein folding.

Both unstacked models (with a single RNN, GRU, or LSTM layer) and stacked models (with multiple successive layers) were built to compare the three architectures. Stacked networks involve multiple recurrent cells in sequence, where the outputs of one recurrent cell are passed to later cells. Although the stacked models require more computation and time to train, their stacked cells enable them to learn from training data at greater levels of abstraction. After their recurrent cells, the networks generate logits for the sixty-four codons with a fully connected layer.

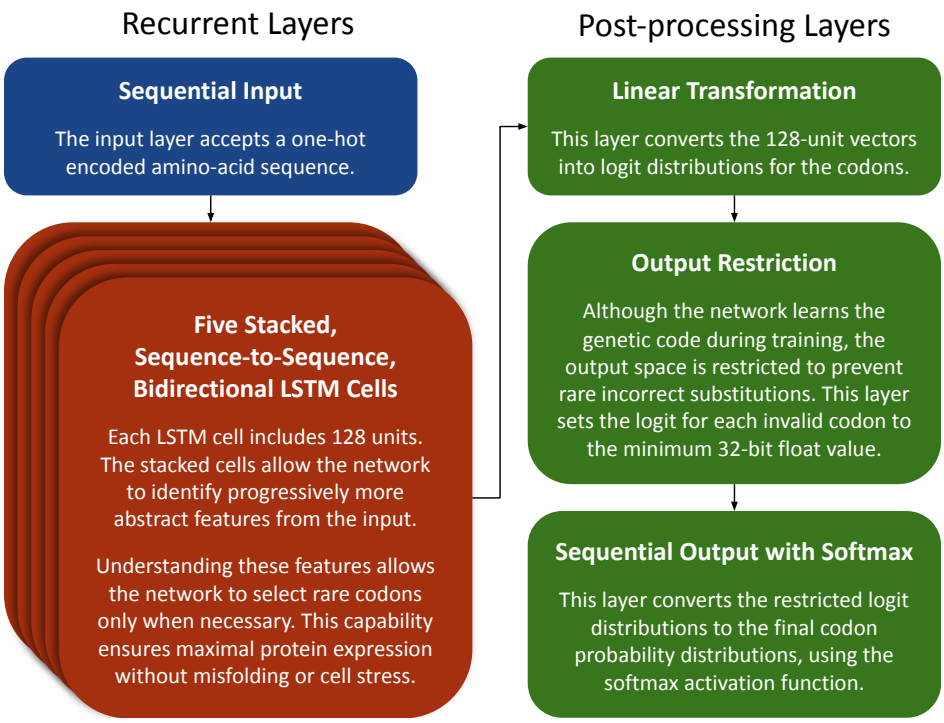


Figure 10: The architecture of the stacked LSTM model, including five successive LSTM cells that abstract progressively more abstract features from the input sequence.

Transformers A sequence-to-sequence transformer was built using a multihead attention mechanism. The model's attention block included four concurrent attention heads with sizes of 1024. Following the attention heads were two convolutional layers, the latter of which generates the logit distributions for the codons.

3.4.2 Model Training

The training datasets for the host species were randomly divided, with 70% used for training the neural networks, 15% used for validation, and 15% used for testing. The neural networks were trained for multiple epochs, and training was stopped manually after the increase in categorical accuracy per epoch became negligible. The training sequences had varying lengths, ranging from fifty amino acids to thousands, so the training batches contained one sequence each. The Adam optimizer was applied with a learning rate between 10^{-4} and 10^{-5}, depending on the model architecture.

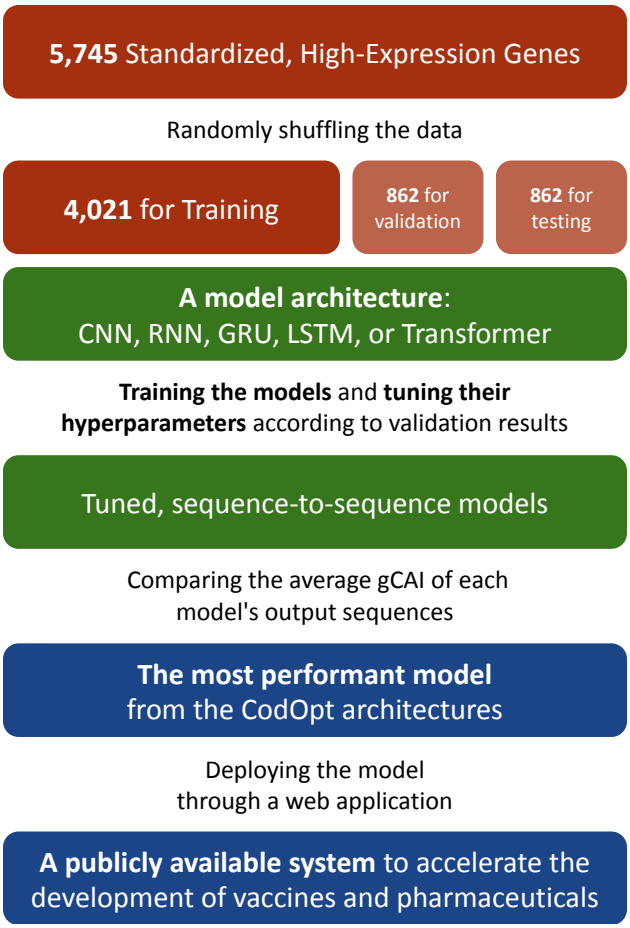


Figure 11: The pipeline for utilizing the highly expressed genes collected for each host organism to train neural networks for codon optimization.

The categorical cross-entropy (CCE) loss function, which measures the difference between probability distributions, was used to train the neural networks. CCE quantifies the difference between two sets of probability distributions. The CodOpt models output one codon probability distribution for each input amino acid. Using CCE, each output probability distribution was compared to the target probability distribution for each amino acid, which contains a 1 for the correct codon and a 0 for the sixty-three incorrect codons.

Let T and O be the target and output tensors for one batch. Each tensor has shape $(I, L, 64)$, with I input sequences in the batch, L amino acids per input sequence (using padding if needed), and 64 codon probabilities per amino acid. Then,

$$\mathrm{CCE}(T, O) = -\frac{1}{IL} \sum_{i=1}^{I} \sum_{l=1}^{L} \sum_{n=1}^{64} T_{i,l,n} \log O_{i,l,n}.$$

By reducing the cross entropy, the neural networks were guided to learn the contextual occurrence of rare codons and achieve confidence in placing frequent codons. Then, the networks were compared according to the average global codon adaptation index (gCAI) that they reached. Therefore, the most performant models understood the contextual placement

of rare codons (reducing cross-entropy) to use frequent codons in other situations (increasing gCAI).

Configuration The neural-network hyperparameters were tuned during multiple training experiments to improve performance without increasing network size dramatically. Model performance was judged using the average cross-entropy loss and global codon adaptation index (gCAI).

Model	Hidden Output Sizes	Optimizer and LR	Encoding
CNN	16, 32, 64, 32, 16	Adam (10^{-5})	One-Hot
RNN	512	Adam (10^{-4})	One-Hot
Stacked RNN	256, 256, 256, 256, 256		One-Hot
GRU	512	Adam (10^{-4})	One-Hot
Stacked GRU	256, 256, 256, 256, 256		One-Hot
LSTM	256	Adam (10^{-4})	One-Hot
Stacked LSTM	128, 128, 128, 128, 128		One-Hot
Transformer	1024 (Four Attention Heads) 64 (Convolutional Layer)	Adam (10^{-5})	One-Hot

Table 1: The final hyperparameter configurations, chosen after multiple training runs.

3.5 Evolutionary Feature Analysis

Feature analysis visualizes the patterns that neural networks have learned during training, enabling researchers to understand a model's predictions. These visualizations can reveal new insights, especially for developing domains such as codon optimization.

After the CodOpt models were tuned and compared, feature analysis was used to confirm that the models learned to recognize evolutionary phenomena that underlie codon usage in highly expressed genes. As determined by testing performance, the most successful CodOpt model was a recurrent neural network. During training, every unit of every recurrent cell learns to identify specific features and sequence properties from input data.

For understanding the features learned by the most performant model, the output of every recurrent unit was captured as the model performed prediction on the testing dataset. The output values were transformed into a sequence of colors for visualization, using a linear color gradient with red for the lowest negative numbers, gray for zero, and blue for the highest positive numbers. A single heatmap of these output values was created for each unit, using a series of HTML `span` elements to visualize the activations. Each unit plot depicts the amino acids in the input sequence, with background colors determined by the color sequence calculated earlier. The output heatmaps were inspected for features learned by the model.

3.6 Optimized Recombinant Expression

For validating the CodOpt models experimentally, a recombinant protein was expressed in Escherichia coli cells, using both an original DNA sequence and a sequence optimized with CodOpt. Researchers at the Lu Lab in the Columbia University Irving Medical Center conducted the procedure for this lab trial.

3.6.1 Expression Plasmid Cloning

The DNA sequence encoding the protein A and Tn5 transposase fusion protein (pA-Tn5) was obtained from a publicly available plasmid map (Addgene #124601). After codon optimization with the most performant CodOpt model, the optimized DNA sequence was synthesized and cloned into the same expression vector using restriction digestion enzymes NcoI and BamHI. The expression plasmid was verified using gel electrophoresis and Sanger sequencing.

3.6.2 Recombinant Protein Expression

Recombinant protein expression was performed as previously described [35]. The original and optimized plasmids expressing pA-Tn5 was transformed into C3013 cells (NEB) according to manufacturer instructions. Two colonies from each plasmid transformation were inoculated into 15 mL LB medium overnight in a shaking incubator. 0.5 mM of fresh Isopropyl ß-D-1-thiogalactopyranoside (IPTG) was added to induce recombinant protein expression, and the culture was incubated at 37 °C on a shaker for 4 hours. After centrifugation at 10,000 rpm and 4 °C for 30 minutes, uninduced and IPTG-induced pellets derived from the bacterial cultures were resuspended in chilled HEGX Buffer (20 mM HEPES-KOH at pH 7.2, 0.8 M NaCl, 1 mM EDTA, 10% glycerol, 0.2% Triton X-100) including 1× Roche Complete EDTA-free protease inhibitor tablets. The lysate was sonicated while chilled. The sonicated lysate was run on SDS-PAGE gel at 150 V for 1 hour. After running, the gel was rinsed in deionized water and incubated with 20 ml SimplyBlue SafeStain (Thermo Fisher) for 1 hour with gentle agitation. After staining, the gel was rinsed twice with 100 ml of deionized water each time for 1 hour. A brightfield image was saved using a scanner.

4 Results

4.1 Model Performance and Statistical Analysis

The average gCAI for the testing sequences increased significantly after optimization, demonstrating that the CodOpt models can substantially enhance protein expression. Before optimization, the average gCAI over the testing dataset for *Escherichia coli* was 0.600. After many iterations of training and tuning, the average gCAIs achieved by all eight model architectures on the testing data were as follows.

Model	Global Codon Adaptation Index
CNN	0.923
RNN	0.928
Stacked RNN	0.915
GRU	0.926
Stacked GRU	0.933
LSTM	0.945
Stacked LSTM	**0.949**
Transformer	0.941

Table 2: The testing gCAI average achieved by each model architecture trained for *Escherichia coli*.

Furthermore, the gCAI distributions after optimization, which represent the frequencies of different gCAI values among the optimized sequences, were as follows.

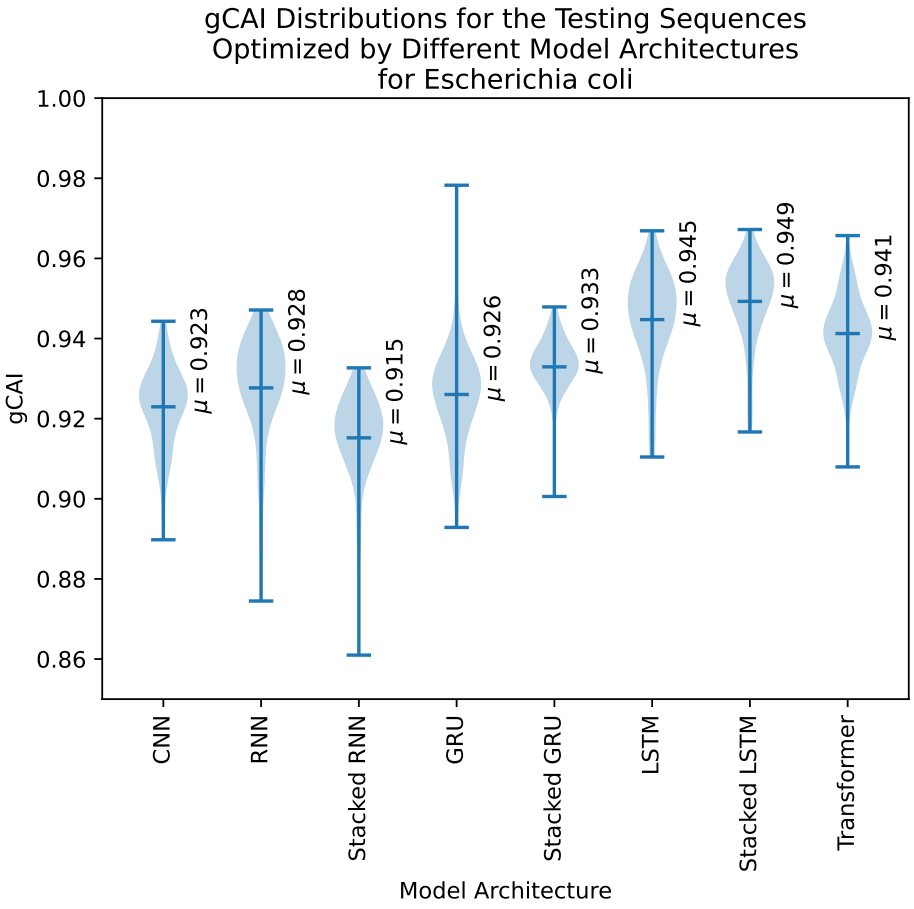


Figure 12: The gCAI distributions for the sequences optimized by all eight model architectures trained for *Escherichia coli*.

The most performant model for *Escherichia coli*, the stacked LSTM network, achieved an average optimized gCAI of 0.949 over the testing sequences, an improvement of 58% compared to the original 0.600. By a one-sided Wilcoxon signed-rank test, the gCAIs of the optimized sequences ($\mu = 0.949$) were significantly greater than the gCAIs of the original testing sequences ($\mu = 0.600$), with a p-value of 4.14×10^{-16}.

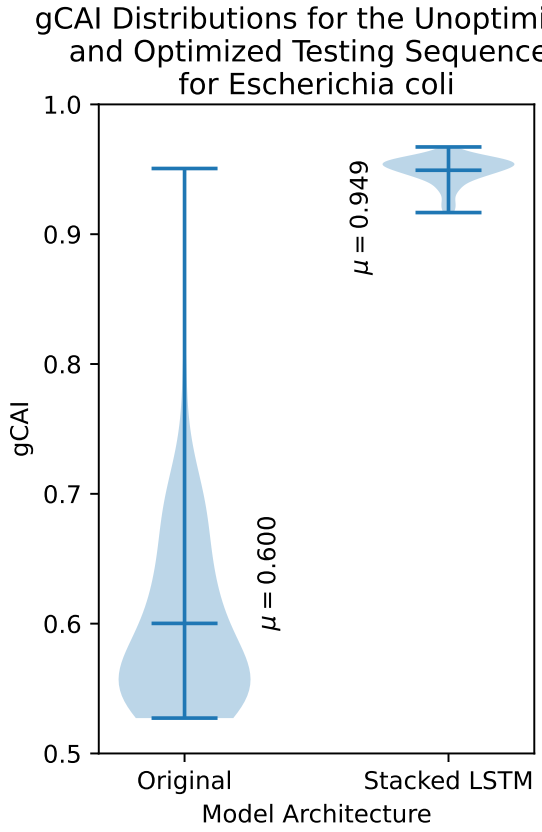


Figure 13: The gCAI distributions for the original testing sequences from *Escherichia coli* and the sequences optimized by the stacked LSTM.

Furthermore, the stacked LSTM achieved the highest average gCAI for baker's yeast and Chinese hamster ovary cells.

Species	Original gCAI	Optimized gCAI
Escherichia coli	0.600	0.949
Baker's Yeast	0.723	0.970
CHO Cells	0.524	0.948

Table 3: The original gCAI average over the highly expressed genes for each host species and the optimized gCAI average after appyling the stacked LSTM model to those genes for each host species.

The *GC content* of a genetic sequence equals the percentage of its bases that are either guanine or cytosine. For some gene g, let A, C, G, and T be the number of occurrences of adenine, cytosine, guanine, and thymine within g, respectively. Then,

$$\mathrm{GC}(g) = \frac{G + C}{A + T + G + C} \times 100\%.$$

Values for GC content below 30% or above 70% can cause the formation of mRNA secondary structure that inhibits translation, hurting protein production. Therefore, the GC content was measured for all optimized sequences to ensure that the models did not change drastically codon frequencies or affect GC content. The models all produced sequences with GC contents between 30% and 70%, so they do not affect GC content adversely.

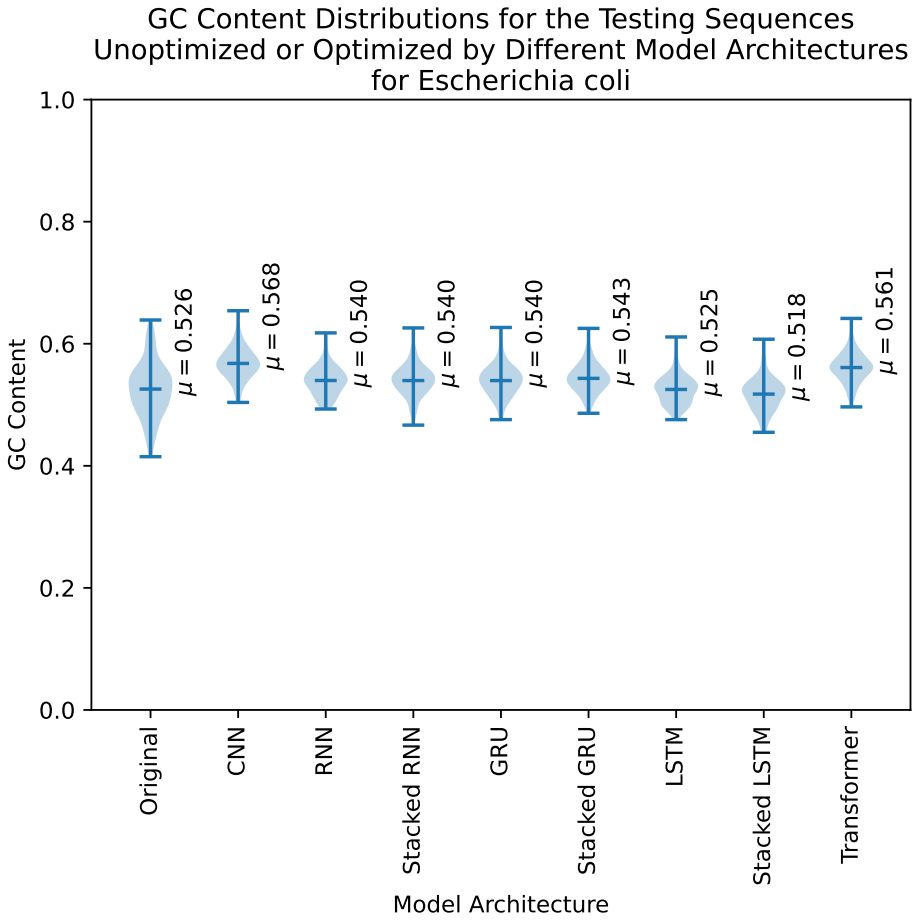


Figure 14: The distributions of GC content for the sequences optimized by all eight model architectures trained for *Escherichia coli*.

4.2 Evolutionary Feature Analysis

Feature analysis was used to confirm that the CodOpt models learned to recognize evolutionary phenomena that affect codon usage. To understand the features learned by the

stacked LSTM for *Escherichia coli*, the activations of every unit were captured as the model performed prediction on the testing dataset. Heatmaps of these activations were created and inspected. These heatmaps demonstrated that the model had learned multiple discernible evolutionary phenomena that affect rare-codon usage.

4.2.1 Translation Initiation and Termination

In many organisms, rare codons cluster at both the 5' and 3' ends (the initial codons and the final codons) of genetic sequences. Clusters of rare codons that begin genes improve the efficiency of translation initiation, enhancing protein synthesis overall. Clusters of rare codons that end genes may provide proteins additional time to fold before being released from ribosomes. Thus, rare codon clusters that begin and end genes enhance the efficiency and safety of protein production.

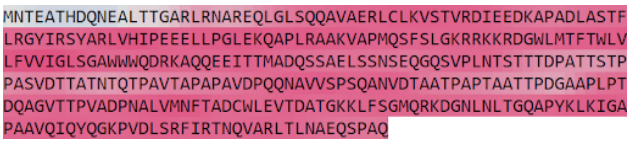


Figure 15: A unit identifying the amino acids toward the beginning of a sequence.

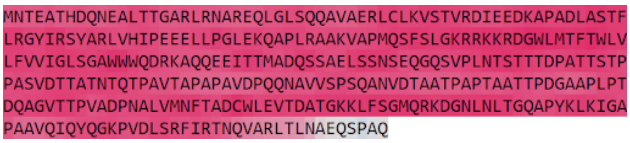


Figure 16: A unit identifying the amino acids toward the end of a sequence.

4.2.2 Kyte–Doolittle Hydropathy

The appearance of rare codons correlates with the Kyte–Doolittle hydropathy (a measure of hydrophobicity and hydrophilicity) of different protein regions. Rare codons cluster in hydrophobic areas where folding often must be slowed.

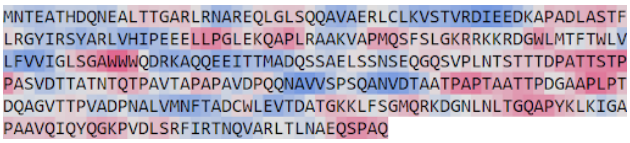


Figure 17: A unit whose values correlate with Kyte–Doolittle hydropathy.

4.2.3 Protein Disorder

Many proteins contain structured regions with uniquely defined three-dimensional arrangements of amino acids under typical conditions. *Intrinsically disordered regions*, however, lack fixed, unique structures. Rare codons correspond with intrinsically disordered protein regions, enhancing overall protein function and conformation.

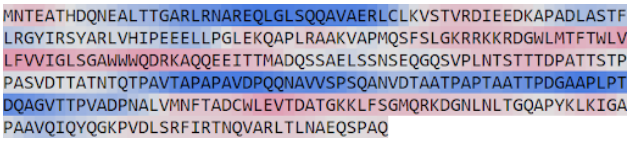


Figure 18: A unit whose values correlate with predicted protein disorder.

4.2.4 Transmembrane Proteins

Transmembrane proteins are proteins that span the cell membrane, often to facilitate the transfer of different substances across the membrane. The insertion of transmembrane proteins into the cell membrane often begins during translation. In genes for transmembrane proteins, rare codons often cluster in positions 50 to 70 to allow for unhindered cotranslational insertion by temporarily slowing translation.

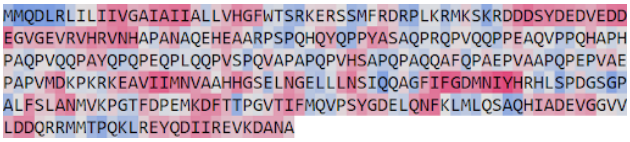


Figure 19: A unit activated at amino acids 50 to 70 (with some noise from connections with previous layers) for a transmembrane protein.

4.3 Optimized Recombinant Expression

For validating the stacked LSTM experimentally, a DNA sequence encoding the pA-Tn5 protein was optimized using the stacked LSTM for *Escherichia coli*. The optimized sequence was cloned into the same expression plasmid as the original sequence. Both the original and optimized expression plasmids were transformed into *E. coli*. After IPTG induction, the bacteria were lysed, and the protein content was separated by SDS-PAGE gel electrophoresis.

Expression levels for pA-Tn5 were visualized using Coomassie Blue staining. Compared to two colonies expressing the original plasmid, two colonies expressing the optimized plasmid had significantly greater recombinant protein expression. Therefore, codon optimization with CodOpt can significantly improve protein expression, accelerating the production of pA-Tn5 and other recombinant proteins such as vaccines and pharmaceuticals.

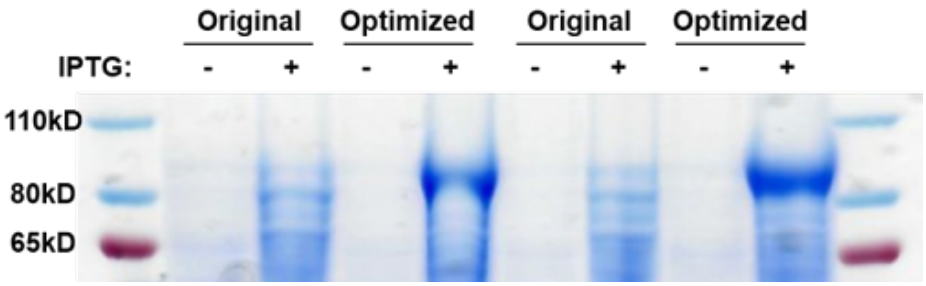


Figure 20: Coomassie Blue staining of SDS-PAGE separated lysates from *E. coli* cultures transformed with either original or optimized plasmids expressing pA-Tn5. IPTG was added to the cultures to induce recombinant protein expression. The gene optimized by CodOpt achieved significantly greater expression than the original gene.

5 Web Application

A web application was built to provide researchers around the globe with the functionality of the CodOpt models. The application is hosted through Amazon Web Services (AWS) for scalable and reliable global hosting. Using the app, researchers developing vaccines and pharmaceuticals can accelerate their work, broadening the impact of these crucial treatments.

After entering and logging into the application, researchers can:

- Select the host species they are using for recombinant expression (*Escherichia coli*, *Saccharomyces cerevisiae*, or *Cricetulus griseus*).
- Input the recombinant DNA sequence for the vaccine, pharmaceutical, or other recombinant protein they intend to manufacture.
- Generate a DNA sequence optimized by the stacked LSTM model for maximal expression within the selected host organism.

The stacked LSTM model trained for each host species was saved in the Open Neural Network Exchange (ONNX) format. ONNX is an extensible, cross-platform file format for saving machine learning and deep learning models. Since ONNX is used for integrating the models, future updates to the models can utilize other deep-learning frameworks if needed.

After a researcher submits an input sequence, the application generates an optimized sequence as follows.

- The amino acids for the corresponding protein are determined using the genetic code.
- The saved model is loaded with the ONNX Runtime for Python and used to predict a codon probability distribution for each amino acid.
- The codons with the highest probabilities are returned as the output DNA sequence.
- The input and output sequences are stored in the application's database, so that researchers can return and review previous results.

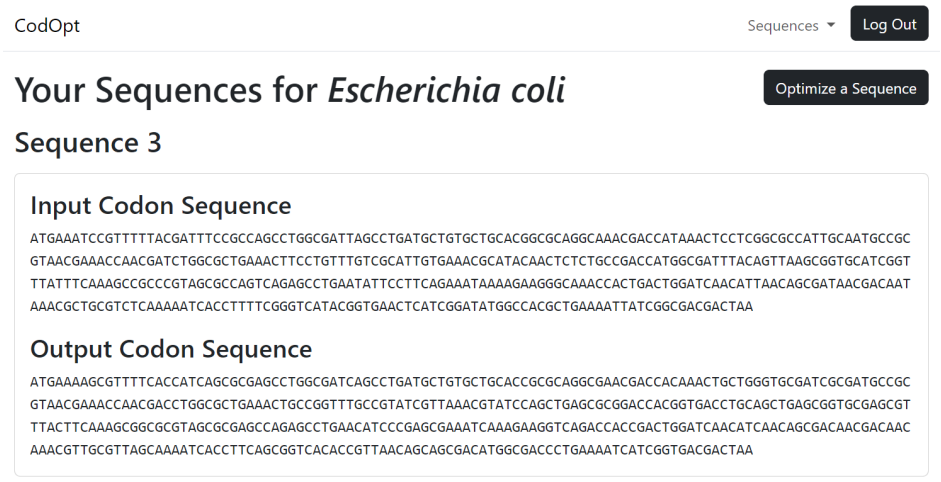
5.1 API and Publishing

Some researchers or organizations may seek to integrate the CodOpt into their pipelines. A JSON API was developed for such integrations to allow other programs and servers to interact with the models. Developers can programmatically submit DNA sequences with JSON and forward the optimized sequences to other applications or databases through the API. Therefore, developers for labs or companies developing recombinant proteins can build applications that meet organizational needs while utilizing the powerful CodOpt models.

Furthermore, the CodOpt models can be published to model networks such as TensorFlow Hub. Through these networks, other developers could download the models' weights and reuse or repurpose the models as needed.

5.2 Web Application Examples

5.2.1 Species Pages



6 Conclusions

This research improves the efficiency of heterologous expression for developing vaccines, pharmaceuticals, and other recombinant proteins. Codon optimization algorithms revise recombinant sequences to increase expression levels. However, the common strategy of removing all rare codons ignores evolutionary details embedded in DNA sequences, such as the contextual usage of rare codons for protein folding. This standard technique can induce metabolic stress in host cells and produce misfolded proteins that harm patients. Therefore, novel codon optimization techniques must increase expression levels while considering the significance of rare codons and other sequence properties.

The CodOpt networks optimize codon sequences by emulating the highly expressed genes of host species. Evolution has tuned these genes to ensure high protein expression while avoiding protein misfolding and other dangerous side effects. Since deep learning can learn at multiple levels of abstraction, the CodOpt networks can emulate these highly expressed genes by comprehending their patterns and structures. By applying this understanding to recombinant sequences, the CodOpt networks can improve expression levels without causing detrimental side effects such as protein misfolding.

The CodOpt networks were compared by their capability to enhance protein production. The stacked LSTM architecture achieved the highest global codon adaptation index (CAI), which predicts protein expression quantitatively using codon bias. Since the models were trained to predict natural codon sequences, they successfully avoided standard optimization techniques, mitigating consequences such as metabolic stress and protein misfolding. Instead, as confirmed by feature analysis, the stacked LSTM model learned several evolutionary features that affect where rare codons are used in natural sequences. Finally, the CodOpt web application enables global access to codon-optimization tools based on deep learning, accelerating the development of safe vaccines and pharmaceuticals.

The CodOpt models can accelerate the development of safe vaccines and pharmaceuticals by addressing the drawbacks of current solutions for codon optimization. This enhancement could save lives, especially during outbreaks that require the rapid design of therapeutics to fight disease.

7 Future Research

Further research can extend and improve the CodOpt models and deploy the host-independent pipeline publicly.

- *Further evolutionary feature analysis.* The CodOpt networks were trained to emulate highly expressed natural genes from host organisms. By visualizing the activations of the recurrent units within the stacked LSTM, five discernible features learned by the model from those genes were identified. Future research could investigate the other activation heatmaps to decipher the properties learned by the model entirely. Neural networks can discern patterns unnoticeable to humans, so further feature analysis may reveal unknown or ignored sequence properties that improve protein expression.

- *Further experimental trials.* Experimental evidence demonstrates that CodOpt-optimized sequences generate significantly more protein than unoptimized sequences. The lab trial conducted with the CodOpt models analyzed the expression of pA-Tn5, a crucial recombinant protein used in the CUT&Tag procedure for analyzing DNA–protein interactions. Further lab trials could establish this increase for multiple proteins and compare the CodOpt models to commercially available optimization techniques on metrics such as expression, protein folding, and protein solubility.

- *Transfer learning for less common hosts.* The CodOpt models are trained for popular heterologous hosts with extensive, public sequencing data. For example, *Escherichia coli* is the most studied model organism, and the sequencing data available online for *E. coli* included over two thousand genomes with tens of millions of genes. However, training models for less common hosts may require more data than is publicly available. Transfer learning, which tunes neural networks for new tasks, could solve this data disparity. Future research could apply transfer learning to build networks for less common hosts.

- *Optimizing promoter sequences and other sequence regions.* The CodOpt models were built to optimize genetic coding regions that define proteins. However, many auxiliary sequences, such as promoter sequences that initiate transcription, also contribute to gene expression. Future research could use deep learning to optimize these regions, further improving recombinant expression.

References

[1] Mohammed N. Baeshen, Ahmed M. Al-Hejin, Roop S. Bora, Mohamed M. M. Ahmed, Hassan A. I. Ramadan, Kulvinder S. Saini, Nabih A. Baeshen, and Elrashdy M. Redwan. Production of Biopharmaceuticals in E. coli: Current Scenario and Future Perspectives. *Journal of Microbiology and Biotechnology*, 25(7):953–962, 2015. Publisher: The Korean Society for Microbiology and Biotechnology.

[2] Suliman Khan, Muhammad Wajid Ullah, Rabeea Siddique, Ghulam Nabi, Sehrish Manan, Muhammad Yousaf, and Hongwei Hou. Role of Recombinant DNA Technology to Improve Life. *International Journal of Genomics*, 2016:2405954, 2016.

[3] WHO. COVID-19 Vaccines with WHO Emergency Use Listing, November 2021.

[4] UK. One year anniversary of UK deploying Oxford-AstraZeneca vaccine, January 2022.

[5] Arnold L. Demain and Preeti Vaishnav. Production of recombinant proteins by microbes and higher organisms. *Biotechnology Advances*, 27(3):297–306, May 2009.

[6] ADA. The History of a Wonderful Thing We Call Insulin, July 2019.

[7] CDC. How Influenza (Flu) Vaccines Are Made, November 2022.

[8] Terese Winslow. Transcription and Translation, December 2021.

[9] Connie Rye, Robert Wise, Vladimir Jurukovski, Jean DeSaix, Jung Choi, and Yael Avissar. The Genetic Code. In *Biology*. OpenStax, October 2016.

[10] Eric Lyons and Michael Freeling. How to usefully compare homologous plant genes and chromosomes as DNA sequences. *The Plant Journal*, 53(4):661–673, 2008. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-313X.2007.03326.x.

[11] Evelina Angov. Codon usage: Nature's roadmap to expression and folding of proteins. *Biotechnology Journal*, 6(6):650–659, 2011.

[12] John Athey, Aikaterini Alexaki, Ekaterina Osipova, Alexandre Rostovtsev, Luis V. Santana-Quintero, Upendra Katneni, Vahan Simonyan, and Chava Kimchi-Sarfaty. A new and updated resource for codon usage tables. *BMC Bioinformatics*, 18(1):391, September 2017.

[13] 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*, 15(3):1281–1295, February 1987.

[14] Gila Lithwick and Hanah Margalit. Hierarchy of Sequence-Dependent Features Associated With Prokaryotic Translation. *Genome Research*, 13(12):2665–2673, December 2003.

[15] EurekAlert! It's all in the code: protein production efficiency can be predicted by gene sequence, February 2019.

[16] Vincent P. Mauro and Stephen A. Chappell. A critical analysis of codon optimization in human therapeutics. *Trends in molecular medicine*, 20(11):604–613, November 2014.

[17] Ming Gong, Feng Gong, and Charles Yanofsky. Overexpression of tnaC of Escherichia coli Inhibits Growth by Depleting tRNA2Pro Availability. *Journal of Bacteriology*, 188(5):1892–1898, March 2006.

[18] Vincent P. Mauro. Codon Optimization in the Production of Recombinant Biotherapeutics: Potential Risks and Considerations. *BioDrugs*, 32(1):69–81, February 2018.

[19] Vincent P. Mauro. Codon Optimization of Therapeutic Proteins: Suggested Criteria for Increased Efficacy and Safety. In Zuben E. Sauna and Chava Kimchi-Sarfaty, editors, *Single Nucleotide Polymorphisms: Human Variation and a Coming Revolution in Biology and Medicine*, pages 197–224. Springer International Publishing, Cham, 2022.

[20] Florian Buhr, Sujata Jha, Michael Thommen, Joerg Mittelstaet, Felicitas Kutz, Harald Schwalbe, Marina V. Rodnina, and Anton A. Komar. Synonymous Codons Direct Cotranslational Folding toward Different Protein Conformations. *Molecular Cell*, 61(3):341–351, February 2016.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. Number: 7553 Publisher: Nature Publishing Group.

[22] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6):1236–1246, November 2018.

[23] Afshine Amidi and Shervine Amidi. Deep Learning, October 2018.

[24] Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal Processing Magazine*, 29(6):141–142, November 2012. Conference Name: IEEE Signal Processing Magazine.

[25] Afshine Amidi and Shervine Amidi. Deep Learning Tips and Tricks, January 2019.

[26] Paul A. Kitts, Deanna M. Church, Françoise Thibaud-Nissen, Jinna Choi, Vichet Hem, Victor Sapojnikov, Robert G. Smith, Tatiana Tatusova, Charlie Xiang, Andrey Zherikov, Michael DiCuccio, Terence D. Murphy, Kim D. Pruitt, and Avi Kimchi. Assembly: a resource for assembled genomes at NCBI. *Nucleic Acids Research*, 44(Database issue):D73–D80, January 2016.

[27] Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J. L. de Hoon. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, June 2009.

[28] John D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(03):90–95, May 2007. Publisher: IEEE Computer Society.

[29] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A System for Large-Scale Machine Learning. pages 265–283, 2016.

[30] Torbjørn Rognes, Tomáš Flouri, Ben Nichols, Christopher Quince, and Frédéric Mahé. VSEARCH: a versatile open source tool for metagenomics. *PeerJ*, 4:e2584, October 2016. Publisher: PeerJ Inc.

[31] Robert C. Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461, October 2010.

[32] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, July 2006.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. arXiv:1505.04597 [cs].

[34] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into Deep Learning. June 2021.

[35] Hatice S. Kaya-Okur, Steven J. Wu, Christine A. Codomo, Erica S. Pledger, Terri D. Bryson, Jorja G. Henikoff, Kami Ahmad, and Steven Henikoff. CUT&Tag for efficient epigenomic profiling of small samples and single cells. *Nature Communications*, 10(1):1930, April 2019. Number: 1 Publisher: Nature Publishing Group.