# ICOR: Improving codon optimization

# with recurrent neural networks

Rishab Jain1, Aditya Jain2, Elizabeth Mauro1, Kevin LeShane1, Douglas Densmore1,3,4

1. Lattice Automation – address
2. Harvard Medical School, 25 Shattuck St, Boston, MA 02115
3. Department of Electrical and Computer Engineering, Boston University, 8 Saint Mary’s St. Boston MA, 02215
4. Biological Design Center, Boston University, 610 Commonwealth Ave., Boston MA, 02215

# Abstract

In protein sequences—as there are 61 sense codons but only 20 standard amino acids—most amino acids are encoded by more than one codon. Although such synonymous codons do not alter the encoded amino acid sequence, their selection can dramatically affect the production of the resulting protein. Codon optimization of synthetic DNA sequences for maximum expression is an important segment of heterologous expression. However, existing solutions are primarily based on choosing high-frequency codons only, neglecting the important effects of rare codons. In this paper, we propose a novel recurrent-neural-network (RNN) based codon optimization tool, ICOR, that aims to learn codon usage bias on a genomic dataset of *Escherichia coli*. We compile a dataset of over 42,000 non-redundant, robust genes that are used for deep learning. The model uses a bidirectional long short-term memory-based architecture, allowing for the sequential information of genes to be learnt. Our tool can predict synonymous codons for synthetic genes towards optimal expression in *E. coli*. We demonstrate that sequential context achieved via RNN may yield codon selection that is more similar to the host genome, therefore improving protein expression more than frequency-based approaches. On a benchmark set of over 40 select DNA sequences, ICOR tool improved the codon adaptation index by 41.69% compared to the original sequence. Our resulting algorithm is provided as an open-source software package along with the benchmark set of sequences.

# Introduction

Designing synthetic genes for heterologous expression is a keystone of synthetic biology. Expressing recombinant proteins in a heterologous host has applications from recombinant pharmaceuticals to vaccine manufacturing. For instance, producing malaria vaccine FALVAC-11 involves designing synthetic plasmids, transfection into the *Escherichia coli* (*E. coli*)host factory, growing out the cells, and harvesting the resulting protein2. To increase the efficacy of recombinant technology further, improving expression output is an area of particular interest.

Although the expression levels of synthetic genes when introduced recombinantly may be dependent on many factors, one important factor is the codon usage bias in comparison to the host3. During translation, complimentary tRNAs are used to read codons from the mRNA strand4. Due to the relative abundance of such tRNA, gene expression can be significantly impacted up to 1000 times4. The frequency of certain synonymous codons found in an organism’s genome is positively correlated with its tRNAs5,6. Thus, by choosing synonymous codons that are more frequently found in the host organism’s genome, there are more amino acids available during translation. Although such synonymous codons may code for the same amino acid, they are not redundant7. In a study by Gao et al.8, low expression levels of human immunodeficiency virus genes in mammalian cells are attributed to rare codon usage8. *E. coli* and the Chinese Hamster Ovary (CHO) cell are a few of many host cells that are used to produce recombinant technology-based biopharmaceuticals and vaccines9. However, it is important to understand the underlying codon usage bias of the host “factory” to maximize protein expression.

Codon optimization tools attempt to screen and filter sequences to improve their expression in a heterologous host and minimize secondary structures10. Today, various FDA-approved recombinant DNA products are ranging from insulin drugs to Hepatitis B therapeutics9. Codon optimization tools can be used to increase expression and therefore the efficiency of manufacturing for such products. Industry-standard codon optimization techniques based on biological indexes replace synonymous codons with the most abundant codon found in the host organism’s genome11. Our review shows that many tools such as Gene Designer7, OPTIMIZER12, and COOL13 are predisposed to favor high-frequency codons only. However, this may result in an imbalanced tRNA pool. If just one codon is being utilized, metabolic stress and translational error may be imposed on the cell7. Recent research has shown that some rare codons play an important role in protein folding14. Understanding the context by which synonymous codons are used in a gene is essential to truly unlock the full potential that evolution has instilled in genes.

By predicting synonymous codons based on sequential information found in the host organism’s genome, protein expression can be increased while preventing translational error and plasmid toxicity. To best learn sequential and contextual patterns of the host, deep learning can be utilized for its high level of abstraction on large datasets15. Deep learning systems are showing promise in bioinformatics because of their ability to analyze large sets of data efficiently16, potentially offering improvements over non-machine learning algorithms.

In the deep learning field, recurrent neural networks should theoretically offer the best approach. Recurrent neural networks (RNNs) are a class of artificial neural networks that can grasp temporal data thus being able to learn sequential information18. For example, speech recognition models that utilize the long short-term memory (LSTM) architecture take advantage of the memory built-in to the LSTM module, thus allowing the model to interpret speech based on the contest of the words surrounding it. For codon optimization, RNNs may offer improved synonymous codon selection because the RNN is designed to understand data’s characteristics and patterns to make its next synonymous codon prediction. For each timestep (amino acid) in the sequence, the RNN would evaluate its prediction in the context of the amino acids before and after it.

In this study, a deep learning tool, ICOR, is trained on a large, robust, non-redundant dataset of *E. coli* genomes. *E. coli* is the most frequently used host for recombinant protein production due to its genetic simplicity and compatibility with codon optimization tools. This “big data” approach allows our model to learn codon usage across multitudinous genes of *E. coli* which may contain valuable lessons in rare codon usage and advantages instilled through evolution. The Bidirectional Long-Short-Term Memory (BiLSTM) architecture is adopted because of its ability to preserve temporal information from both the past and future. In other words, the amino acids present in the timesteps before and after the synonymous codon are taken into account.

40 reference DNA sequences as shown in Table S1 were used to establish a benchmark set used for testing and evaluation. The resultant optimized sequences from the ICOR model were compared to 5 approaches (original, super naïve, naïve, brute-force, GenSmart) as outlined in S1 Appendix. Along with codon adaptation index (CAI), the GC content, CFD, negative repeat elements, and negative CIS elements are used as metrics to evaluate the performance of the algorithms. The ICOR tool is open-source and can be accessed at: <https://github.com/Lattice-Automation/icor-codon-optimization>.

# Materials and Methods

## Dataset

We use the National Center for Biotechnology Information’s *E. coli* genomes19 which included 6,877,000 genes. Genes that were under 90 amino acids in length were removed due to their hypothetical or specialized nature. Further, CD-HIT-EST20 was utilized to cluster similar nucleotide sequences by sequence identity of over 90%. After curation and removal of redundant genes, 42,266 sequences were left of which 7,000 high-expression sequences were used for the training of the model. Approximately 70% of the dataset was used for training, 10% was used for development or validation, and 20% for testing.

## Benchmarks

40 DNA sequences were established as a benchmark set, extracted from studies conducted on codon optimization and evaluating the gene expression of plasmids in *E. coli.* The resultant sequences can be accessed at <https://github.com/Lattice-Automation/icor-codon-optimization/> and their descriptions in Table S1.

## Encoding

Encodings were created for our entire dataset using the “one-hot encoding” technique. Amino acid sequences were converted into integers and then placed into vectors that are 26 features long. At each timestep, the present amino acid is encoded into the vector as “1” while all other features are set to “0”. For example, the amino acid Alanine can be represented by 1, and will thus be set to 1 in the encoding while all other features are set to 0. Features were based on Table S2.

## Deep Learning

Sequential information may yield synonymous codon selection that is more similar to the host organism. Deep learning is a technique that may be able to capture underlying patterns found in the host genome. Our model uses the BiLSTM architecture which predicts synonymous codons given the input amino acid sequence. The model hyperparameters were tuned iteratively when trained on the train and validation datasets. We used L2 regularization and dropout to finetune our model and prevent overfitting.

**Fig 1. The ICOR model’s architecture consists of a 12-layer recurrent neural network.** Data is fed forward from the first layer—Sequence Input—to the last layer—Classification—from top to bottom.

The model was trained in the MATLAB r2020b25 environment on 50 epochs which took 138 minutes during training on the Tesla V100 graphics card.

**Hyperparameter Tuning**

We experimented with many variables such as mini-batch size, gradient decay, and model architecture as displayed in Table 1.

**Table 1. Hyperparameter variables are used to finetune the results of the model.** (A) In the left-hand column, 8 hyperparameters are listed. (B) In the right-hand column, their respective settings are given based on the testing and finetuning conducted in this research. Our iterative tuning methods reach the following hyperparameters which prevent the possibility of over/underfitting while maintaining high accuracies.

The need to make trade-offs when it came to hidden units became evident. Initially, we found our model to be underfitting. In order to make it more complex, we made our model architecture more complex, and also increased the number of hidden units. Further, when the learning rate was too high, the model very quickly plateaued. In order to force the model to train for a longer period of time, this value was adjusted along with the epochs. Changing the value of the mini-batch-size was a matter of balancing memory for performance. The final model was trained across 50 epochs, with a minibatch size of 512, and 256 hidden units.

## Software Functionalities

The development of ICOR has two major software components for the user: ICORnet architecture and runtime scripts. The ICORnet architecture is a BiLSTM type of RNN. It serves as the brain for the codon optimization tool. By providing the amino acid sequence as an input, ICORnet can output a nucleotide codon sequence that would ideally match the codon biases of the host genome. The key steps of development are as follows:

1. Fastalator loads DNA sequences from FASTA files and converts them into structures containing amino acid sequences and the original DNA sequence. It can account for DNA sequences that do not have 100% confidence (occurs due to sequencing) by using IUPAC probability estimators.
2. CreateTrainingData & CreateTrainingDataNLF are two scripts that encode the amino acid data into vectors for the ICORnet RNN model. One uses One Hot Encoding while the other uses Non-Linear Fisher Transform.
3. trainNet trains ICORnet on the data imported. It provides options to adjust hyper parameterization and the network architecture. It outputs a network object which can then be used for classification using the classify function or through the ICORnet user application.

Given an input of sequences they would like to optimize, a user would receive optimized codon sequences using the runtime scripts integrated with the ICOR model. The runtime scripts utilize the ONNX runtime to run inference on the trained MATLAB model. This system was designed modularly so improvements to the ICORnet model can be easily accessed by the user without the need to have a packaged execution. Rather, the ICORnet model can be downloaded and executed using the runtime workflow.

## Statistical Analysis

We use the CAI, the mutational rate, rare codon usage, CFD, negative repeat elements, and negative CIS elements to conduct statistical analysis. The CAI is calculated using the formulae described in Fig S2. The mutational rate is quantified by conducting optimization on the test dataset, converting the optimized codons back to amino acids, and then counting the number of amino acids that varied between them. Rare codon usage was qualitatively and quantitatively compared to reference tables27.

During validation, codon accuracy, amino acid accuracy, and base errors were measured. By comparing the similarity of the optimized sequence to the input, these accuracies can be calculated.

# Results

**Codon Prediction with Deep Learning**

We use primarily to quantify the performance of our tool. As noted in previous studies, CAI is highly correlated with real-world expression. On the test dataset of 8,000 genes, we find that ICOR offered an improvement in CAI from 0.73 to 0.889 ± 0.012, or about 29.1% compared to the original sequences (Fig 2).

**Fig 2. ICOR significantly improves CAI when compared to original sequences on a test dataset of 8,000 genes.** Box and Whisker Plot (n=8000) comparing CAI (left: original sequences, right: ICOR optimized sequences). The y-axis is the codon adaptation index on a scale from 0 to 1.0. The open points outside of the boxes are outliers that are beyond 1.5 times the interquartile range. The horizontal divisions present in each box (from top to bottom) are the upper whisker, 3rd quartile, median, 1st quartile, and lower whisker.

In order to properly contextualize the performance of the developed model, 4 algorithms from S1 Appendix were used to generate optimized sequences from the original benchmark sequences. These proteins came from a variety of origin organisms and had a mean CAI of 0.638 with a standard deviation of 0.0386. ICOR optimization resulted in a mean CAI of 0.904 with a standard deviation of about 0.016, signifying a ~41.692% increase. The super naive approach had a mean CAI of 0.602 and standard deviation of about 0.022. ICOR offered a ~50.21% increase compared to this approach. Finally, the naive approach offered a mean CAI of 0.699 and a standard deviation of 0.0158. ICOR offered a ~29.32% increase in CAI compared to the naive approach. These comparisons were statistically significant (p<0.0001) using a two-sample t-test. The mean CAI for all approaches is shown in Fig 3.

**Fig 3. ICOR significantly improves CAI when compared to the original, naïve, super naïve, and GenSmart techniques.** Box and Whisker Plot (n=40) comparing CAI with legend indicating the color for each optimization method.

When extrapolating such improvements to findings by dos Reis et al., real-world mRNA expression could improve by up to 236%30. ICOR codon optimization offers a significant improvement when compared to the original gene, and that the tool is competitive.

**Table 2. Codon optimization approaches are scored using metrics on the benchmark set**. The first column lists the genes used in the analysis. The second column lists the number of base pairs for each gene. The third column has the Codon Adaptation Index of the original genes. The fourth column shows the Codon Adaptation Index of the Genewiz31 (accessed March 10th, 2020) optimized genes. The fifth (right-hand most) column lists the Codon Adaptation Index for the ICOR optimized genes. This comparison was made on December 25th, 2020.

There was a statistically insignificant difference (p=0.726) between ICOR and GenSmart in the

number of negative CIS elements in the optimized sequences using the Mann-Whitney U Test for non-parametric distributions. When computing the mean change in negative CIS elements between the optimization tool – original sequence, there is a statistically insignificant difference between ICOR and GenSmart. This suggests that ICOR maintains equally low negative CIS elements as GenSmart while achieving a higher CAI. There was a trending difference (p=0.1826) between ICOR and GenSmart in the number of negative repeat elements in the optimized sequences using the Mann-Whitney U Test for non-parametric distributions. This suggests that although ICOR may have higher negative repeat elements, the difference is minimal as compared to GenSmart.

**Optimization Run Time**

The run time was calculated for the approaches where inference time could be isolated. Using a testing system as described in S1 Supporting Information, the algorithms were evaluated for run time on the benchmark set with an average length of 1687.65 nucleotides per sequence. The scores normalized to the super naïve approach are displayed in Table 3.

**Table 3. Optimization run time was calculated for four approaches normalized to the super naïve approach.** Three trials were conducted with the score being an average of these three times measured to the third significant figure in milliseconds.

**Gene Mutations**

Deep learning has the capability to learn, classify and predict large amounts of data. However, as with any algorithm, it may not be perfect. In the case that the current amino acid (i.e., Glutamine) is replaced with a codon for an entirely different acid, this would result in a point mutation. When introduced inside of the host, detrimental effects to gene expression would ensue.

During our testing, it was found that encoding techniques made a significant difference. The One-Hot Encoding technique offered approximately a 10% improvement over Non-Linear Fisher Transform. Theoretically, the One-Hot Encoding technique will be able to learn the amino acid context faster, because the input data is simpler: a 1x26 array with a “1” in the position of where the amino acid is. On the test dataset of 8,000 genes, our model yielded a 0.00% mutational rate. Thus, it was found that gene mutations are not present in our codon optimization technique.

**Discussion**

In this research, we propose a codon optimization tool named ICOR that uses recurrent neural networks towards improving heterologous expression for synthetic genes. We find that deep learning is a particularly effective method in the synonymous codon optimization area because it can contextualize the entire host genome to understand synonymous codon usage bias. While previous research focuses on machine learning models and the convolutional neural network architectures, we use the RNN architecture which has the ability to undertake temporal data. By understanding the underlying patterns in the host genome through the use of an RNN, codon selection may be more similar. Using this approach, we train the ICOR model on a large dataset of 42,266 non-redundant genes from the *E. coli* genome. Having a non-redundant dataset was of vital importance in this study as many *E. coli* genomes will contain extremely similar genes. We used the CD-HIT-EST server to overcome this issue, as training a model on redundant data would yield codon selection that is biased to frequent genes only. Further, this helped reduce the necessary compute resources required for deep learning.

This research also demonstrates that encoding gene sequences using Natural Language Processing techniques is of particular interest. We use the one-hot encoding method in our final model. However, we also tested the Non-Linear Fisher Transform (NLFT) based on the physicochemical properties of amino acids. One-hot encoding offered approximately a 10% improvement over NLFT. Theoretically, this is because when training a model using one-hot encoded data, it can immediately identify the amino acid labels. Thus, it can very quickly start to work on the problem of predicting codons from the simple amino acid labels. On the other hand, NLFT may take a longer time to do so due to its unique feature set.

The statistical data analysis primarily uses the Codon Adaptation Index as a way to directly compare ICOR to other solutions in addition to quantifying the accuracy of the tool on the test dataset. Although increases in CAI are strongly correlated with increased protein expression, it is not the only comprehensive statistic to quantify gene expression. Recent research has shown that creating models for measuring translation dynamics is possible, and such research can be applied to the results of our study. Modeling elongation rate, tRNA adaptation index, and other metrics may provide valuable insight into the results of the tool.

ICOR can be applied directly in synthetic gene design. Synonymous codons can be optimized to increase expression further. Thus, the efficiency of production improves, potentially decreasing the cost of the product. Currently, the ICOR tool can be installed in the Windows platform, however, we would like to build an API to improve accessibility in the future.

Although our model is based on the *E. coli* bacterium, it may be plausible to apply our methodology to other organisms such as yeast and mammalian cells in future research. A transfer learning approach may allow us to preserve our pre-trained model, and adapt it to other host cells. Additionally, we would like to add the ability for our model to optimize other regions of a gene such as promoter sequences. Finally, to further verify confidence in the results, future research may consider testing our model in the lab setting.