Learning the Language of Genes: Representing Global Codon Bias with Deep Language Models

M. Stanley Fujimoto*, Paul M. Bodily, Cole A. Lyman, J. Andrew Jacobsen, Mark J. Clement
Computer Science Department
Brigham Young University
Provo, Utah 84604
*sfujimoto@gmail.com

***Abstract*—Codon bias, the usage patterns of synonymous codons for encoding a protein sequence as nucleotides, is a biological phenomenon that is not well understood. Current methods that measure and model the codon bias of an organism exist for usage in codon optimization. In synthetic biology, codon optimization is a task the involves selecting the appropriate codons to reverse translate a protein sequence into a nucleotide sequence to maximize expression in a vector. These features include codon adaptation index (CAI) [1], individual codon usage (ICU), hidden stop codons (HSC) [2] and codon context (CC) [3]. While explicitly modeling these features has helped us to engineer high synthesis yield proteins, it is unclear what other biological features should be taken into account during codon selection for protein synthesis maximization. In this article, we present a method for modeling global codon bias through deep language models that is more robust than current methods by providing more contextual information and long-range dependencies to be considered during codon selection.**

***Keywords*—IEEEtran, journal, LaTeX, paper, template.**

I. Introduction

Codon bias is the usage pattern of different synonymous codons, different codons that encode the same amino acids, in an organism. Studies have observed that differences exist in frequencies of synonymous codon usage across different organisms [4], [5], [6]. The genome hypothesis of codon bias is that different organisms have a distinct codon bias from other organisms [5]. The codon bias for a given organism is complex still as even within the same genome codon usage varies amongst genes [7], [5].

Codon bias influences many aspects of biology: RNA secondary structure [8], gene expression [9], speed of translation elongation and protein folding [10]. Thus, codon bias is an important phenomenon to understand. It influences such fields as synthetic biology where engineering is applied to biology. A protein of interest is usually presented as an amino acid sequence. The protein must be reverse-translated (backtranslated), converted from an amino acid sequence to a nucleotide sequence. The backtranslated sequence is then injected into a biological vector (e.g. *Escherchica coli*) to express the protein. Engineers seek to use codons that correspond to the amino acids of the protein of interest that maximize the expression of the protein to reduce production time and to use smaller culture volumes. A biologically unviable protein may result if the nucleotide sequence for a protein is transcribed into an RNA transcript that degrades prematurely, does not fold in the correct manner to confer correct function or is expressed in small quantities. Welch et al. found that proper selection of codons can result in 100-fold differences in expression of genes in *E. coli* [11].

The cause of codon bias is unclear. Hershberg and Petrov discuss that that there are two general classes of explanations of codon bias: selectionist and mutational [12]. The selectionist explanation states that codons are selected to maintain efficiency and/or accuracy of protein expression. This explanation is supported by the correspondence of *preferred codons*, frequently used codons, and tRNAs that occur more abundantly within an organism [13], [14]. The mutational explanation states that codon bias exists because mutational patterns are nonrandom.

A. Measuring Codon Bias

Many algorithms model codon bias with respect to different facets of biology. Some algorithms attempt to mimic the codon usage found throughout all, or a subset of, genes in an organism. Usually, a subset of genes is selected because they are genes that are highly-expressed. Welch et al. found that simply mimicking the host's codon usage does not always yield the highest expression [11]. Methods used to calculate codon usage to mimic host codon usage are very simple. We propose a method of modeling codon usage for a particular organism that is much more robust than current methods. These methods may be used in the continued development of codon optimization software packages and may show that mimicking host codon usage is sufficient so long as the model used for mimicking is sufficiently robust.

Several methods exist to empirically calculate analyze codon bias in different organisms. These measures can then be used to inform codon selection for proteins where the optimized nucleotide sequence is unknown. Frequencies for each set of synonymous codons can easily be calculated given the *coding sequence* (CDS) for a genome. Using the most frequently occurring codon of a set of synonymous codons can then be used to backtranslate amino acid sequences. This, however, is too simplistic. Not all proteins in an organism are highly expressed. Simply counting codon usage amongst all proteins allows for lowly-expressed proteins to influence codon selection for proteins we want to be highly-expressed. Additional studies also report that the choice of high-frequency codons contributes less than other factors when evaluating translational efficiency [15], [16], [17], [18].

Another method would be to choose only codons that

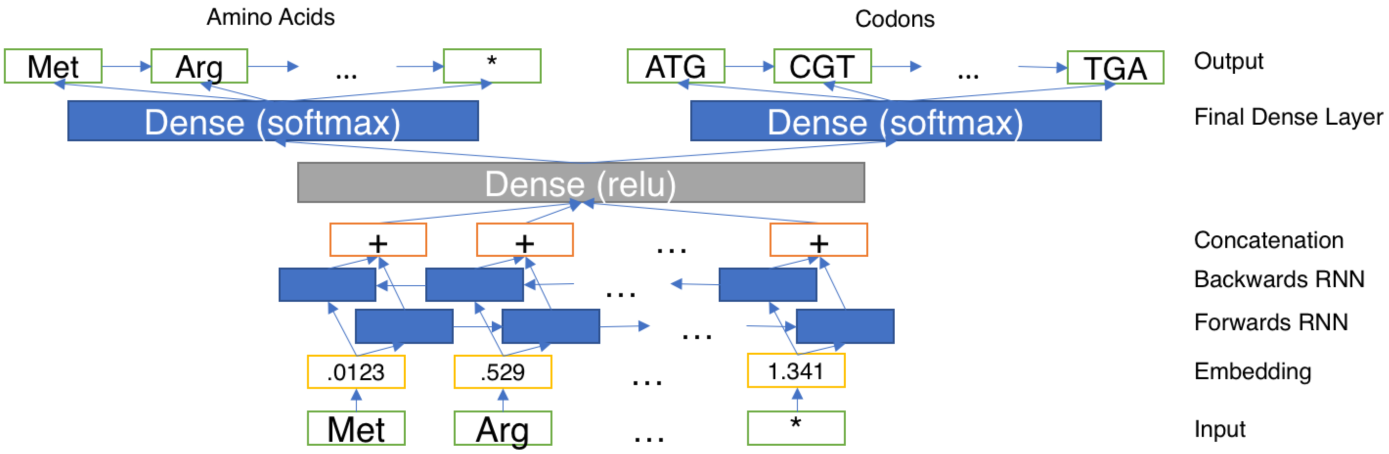

Fig. 1. Model architecture. The model is very similar to that used in Neural Machine Translation with the exception of tighter coupling between the encoder and decoder. Our model also features two target outputs: amino acid sequences and codon sequences.

correspond to tRNAs that are abundant in the host organism. Intuitively, this method seems appropriate because it would allow for the rapid translation of a protein. Studies have found, however, that there are circumstances in which translation should slow down. This may be the case because during translation the protein is folding and by slowing down translation the correct structures are able to form. If translation occurs too quickly, the protein may not fold in the correct manner.

While other methods exist for determining the correct codon usage for backtranslation (F_{of} [19], RCA [20]) the Codon Adaptation Index (CAI) [1], according to Graf et al. [21], is the most widely accepted index for codon usage determination in backtranslation. Welch et al. finds, however, that CAI usage is a poor predictor of gene expression levels [11].

DNAWorks is a tool used in synthetic gene engineering for backtranslating amino acid sequences given a target vector for gene expression [22]. It is designed to take an amino acid sequence as input and a table that specifies the usage frequency of each codon in a particular organism. DNAWorks' method for optimizing the codons in the nucleotide sequences relies on choosing amongst a set of synonymous codons the most frequently observed codon as annotated in the input table. Post-processing can modify the sequence further to reduce unwanted structures (hairpins, etc.) from forming during PCR. Their main method, however, of modeling the codon bias of a particular organism is limited as it is represented solely as a table of synonymous codon usage frequencies and contradicts what is now known about codon bias.

Another application that exists is GeneGPS by ATUM. GeneGPS is based on the work done by Welch et al. [11] where they found that, in *E. coli*, that the expression levels of proteins were most strongly dependent on codons with corresponding tRNAs that are highly charged during amino acid starvation and not the most commonly used codons in highly-expressed *E. coli* proteins. Codon selection for high expression should maintain high levels of charged tRNAs and minimize levels of uncharged tRNAs. The results from Welch et al. show that mimicking the host's codon bias or using high CAI does not result in optimal expression levels.

More robust and expressive models of codon bias are necessary to generate high synthesis yield nucleotide sequences. We present a method for modeling the codon bias of an organism by using a *bi-directional recurrent neural network* (BRNN) with either *long short-term memory* (LSTM) or *gated recurrent unit* (GRU) memory cells [23], [24], [25].

This method seeks to capture the global codon bias of an organism in a much richer manner than CAI and similar methods. CAI computes a frequency matrix of codon usage. This simplistic representation does not account for complex phenomena that we are aware of such as the preferences of infrequent codons when translation should be slowed or sped up during translation. The method that we have developed is much richer by allowing it to select codons based on contextual information instead of relying solely on a single amino acid and an overall matrix of codon usage for an organism. By using the known nucleotide sequence of existing proteins, our model learns which codon to use while considering the entirety of the protein sequence. Using contextual information allows our model to learn when to use certain codons as selection may differ when forming different types of protein structures or how the nucleotide sequence will influence mRNA secondary structure formation. This contextual information gives a much richer model of codon bias for an organism.

II. Methods

A. Datasets

We trained base models using the 159 different *Escherchica coli* strains. These data were gathered from NCBI using the Entrez API through BioPython [26] which allows for easy automated querying and retrievel of both CDS and corresponding amino acid sequences. The model was then fine-tuned for 4 of the *E. coli* strains and for the bacterias *Staphylococcus aureus* and *Yersinia enterocolitica* (see Table III for strain NCBI accessions).

B. Deep Language Model

To model the codon bias of a specific organism we use a method that has been used in Natural Language Processing

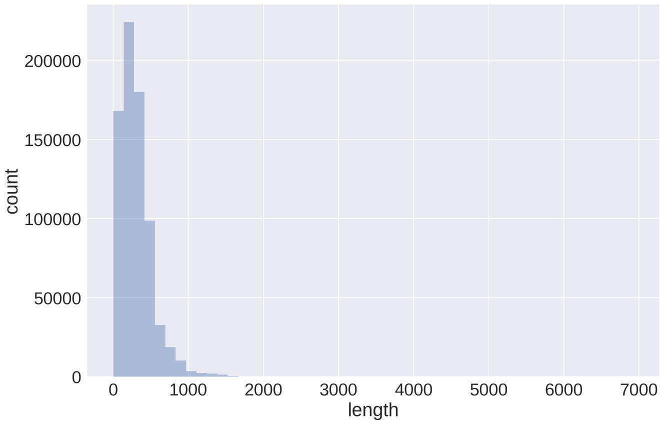

Fig. 2. Histogram of protein sizes used from the 159 pooled *E. coli* genomes. Most proteins used fall under 1,000 amino acids in length with the longest protein 6,926 amino acids and the shortest with 15 amino acids.

TABLE I. SEQUENCE LENGTH FILTERING. THE DATASET SIZES AFTER SEQUENCE LENGTH FILTERING HAS BEEN APPLIED

Seq Len	Train/ Validate	Test
100	83531	20883
250	278690	69673
500	516151	129038
1000	586227	146557

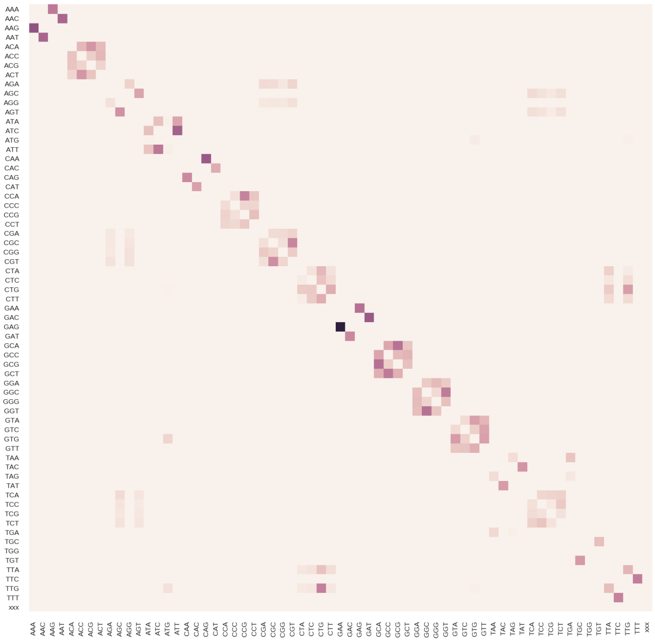

Fig. 3. Heat map of erroneous codon substitutions by the sequence length 100, no dropout model. The x-axis represents the correct codon and the y-axis represents the codon incorrectly substituted for by the model.

that uses deep neural networks for natural language translation and for language modeling. We treat amino acid sequences as a source language and the corresponding codon sequences as the target language. A base model is trained using an aggregate of amino acid sequences from many different organisms and is later fine-tuned for a specific organism. This process is analogous to processes used in object recognition in computer vision, style transfer and other deep learning applications.

Neural machine translation (NMT) is a method used for natural language translation that uses neural networks [27], [28]. The translation task is to take a sentence in one language as input and output the sentence in another language. Neural machine translation relied on an *encoder/decoder* scheme to solve the task. The input sentence is encoded using a neural network, the *encoder* into a vector representation of float values. A separate neural network, the *decoder*, then takes the encoded values as its input and outputs a sentence in the target language.

C. Network Architecture

Our basic neural network architecture is a BRNN trained with amino acid sequences as inputs and the known codons as target outputs (see Figure 1). This architecture was extended to increase codon prediction accuracy. LSTM or GRU cells are specified at run-time by the user for use in the BRNN. Each separate amino acid in the input sequence is regarded as a time-step in the overall sequence by the network. Our model architecture differs from those used in NMT where NMT does a complete encoding of the input and then passes that complete encoding to the decoder. This results in outputs that can have lengths that are independent of the input (as can be the case in natural language translation). For the amino acid detranslation problem, we know that there will be as many codons as there are input amino acids. Thus, we built our network to produce an output at every time-step instead of encoding the entire sequence first.

In order to help the network to output only codons that encode the correct amino acid, we have two output targets for the network: correct amino acid and correct codon. Using the correct amino acid as one of the targets should help the network where an incorrect codon is selected at least it encodes the correct amino acid.

D. Fine-Tuning

After training the base model, we then fine-tuned models for a specific strains. Fine-tuning involves taking the base model and continuing training using a more specific dataset [29]. Different variations of fine-tuning exist. To fine-tune our implementation, we freeze the weights in the recurrent layers and leave the last neural network layer weights unfrozen during the additional training. This results in the final fully-connected layer being modified and the recurrent layers remaining the same as the base model during fine-tuning.

Fine-tuning can also be accomplished by replacing the last layer of the network and completely retraining the last layer while the other layer weights remain frozen. This may be beneficial if the sequences used for fine-tuning vary greatly from the original dataset used to train the network. This still leverages the learned features from the network but allows it to learn final codon selection in a manner much more specific to the specified sequences.

Fine-tuning allows us to build networks that are specific to a particular species or even for a specific strain of a bacteria. We perform fine-tuning instead of re-training a new network to reduce training time. The base model is trained with all data

TABLE II. Model parameters and testing accuracy. Note that the training accuracy is usually lower than the validation accuracy when dropout is used because a portion of the network is unused during training while the full network is used on the validation dataset while.

Network Architecture								Training		Validation		Testing	
RNN Cell Type	# RNN Layers	Embedding Size	Network Width	Batch Size	Epochs	Dropout	Seq Len	CDS Acc	AAS Acc	CDS Acc	AAS Acc	CDS Acc	AAS Acc
LSTM	2	64	512	128	10	0%	100	98.63%	100.00%	97.48%	100.00%	97.35%	100.00%
LSTM	2	64	512	128	10	0%	250	97.60%	100.00%	96.98%	100.00%	97.04%	99.99%
LSTM	2	64	512	128	10	0%	500	96.78%	100.00%	96.65%	100.00%	96.70%	99.99%
LSTM	2	64	512	32	2	0%	1000	94.05%	100.00%	94.47%	100.00%	-	-
LSTM	2	64	512	128	10	20%	100	97.07%	100.00%	97.31%	100.00%	97.15%	99.99%
LSTM	2	64	512	128	10	20%	250	94.02%	100.00%	95.51%	100.00%	95.55%	100.00%
LSTM	2	64	512	96	10	20%	500	88.65%	100.00%	93.79%	100.00%	93.79%	99.99%
LSTM	2	64	512	32	2	20%	1000	79.86%	100.00%	91.08%	100.00%	-	-

from different species/strains pooled. After the base model is trained, we then select a specific species or strain to fine-tune with resulting in a model that can train quicker by leveraging previous work.

E. Additional Heuristics

While this approach showed high codon selection accuracy, there were still instances where the model would choose codons that encoded the wrong amino acid even after the addition of the amino acid target function to the network (see Section II-C). We correct for this by taking the softmax probabilities for the codons, mask probabilities for any codon that does not encode the correct amino acid and select from the remaining codons the highest probability. Amino acid accuracy is guaranteed to be perfectly accurate with this post-processing measure but does not make a significant difference in overall codon selection accuracy. This is most likely because the network almost always selects a codon that codes for the correct amino acid.

This heuristic can be applied to all positions in the predicted sequence except the start codon. While methionine is usually the first amino acid, it is not always encoded by AUG. Alternative start codons may be used and still encode a methionine. This is due to a special initiator tRNA that is used during translation [30]. Recently, more alternative start codons have been discovered [31]. Thus, we leave the network with the ability to choose any codon as a start codon as it is unclear which codons are valid start codons as there may be yet undiscovered start codons.

F. Implementation

Model training was done using NVIDIA K40, Titan X Maxwell and Titan X Pascal GPUs. Our models were built using the Keras [32] neural network library. All our models were developed using Theano [33] as the backend for Keras. The model can be run using different parameters for the type of recurrent cell, dropout in the recurrrent cells, the number of weights for different layers and number of recurrent layers (see Table II for tested parameters). All models were trained using the Adam optimizer [34].

III. Results and Discussion

A. Base Model

The results of training the base model using the pooled data from 159 *E. coli* genomes can be seen in Table II. Data was pooled to provide enough data to train the model where a single strain did not. In total, the pooled dataset consisted of 742,494 sequences. Characteristics of the dataset can be seen in Figure 2.

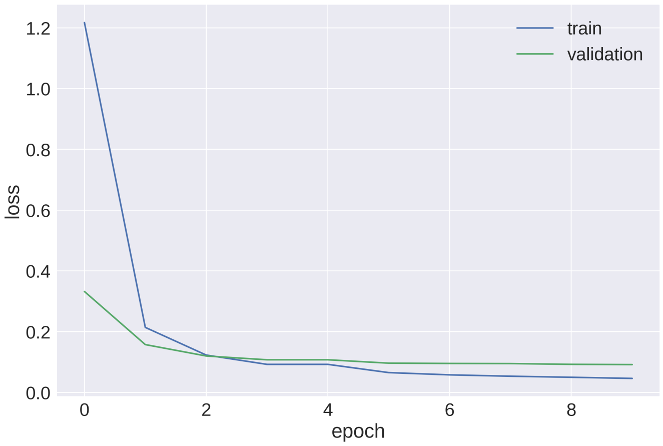

Fig. 4. Loss over epochs for the SEQ LEN 100 dataset. The same loss trend is observed for all training parameters tested.

A number of different models were trained with different parameters. Selecting for different maximum sequence lengths can greatly affect training time as long sequences take longer to process. We tested different sequence lengths (100, 250, 500 and 1,000) and achieved high accuracy that decreased as sequence length increased. The testing accuracy of the sequence length 1,000 models were unavailable due to their long run-times. Filtering the dataset based on sequence length resulted in different dataset sizes used for training, validation and test (see Table I). Although the number of instances does not grow dramatically bhen filtering sequences of 500 and 1,000 amino acids, the overall amount of data increases dramatically.

Reported training accuracy is lower than validation accuracy for for models using dropout. This is because during training with dropout, a portion of the network is unused. This results in lower performance. When calculating validation accuracy, the entire network is used.

B. Start Codon Selection

While AUG is the predominant start codon, there are several others that can be used [31]. We analyzed the predicted start codon for all nucleotide sequences and found that all sequences were found to use a known start codon; namely: AUG, GUG, UUG, CUG, AUU, AUC and AUA.

C. Predicting the Wrong Codon

Prediction of the wrong codon occurs more frequently as sequences become longer. This could be mitigated by a larger network at the cost of increased training time. The occurrence of incorrectly selecting codons can be seen in Figure 3. The axes are sorted lexicographically and, as expected, a strong band along the diagonal emerges. This is because synonymous codons often share the first two bases with the third position in the codon acting as the wobble base pair.

D. Codons that Encode the Wrong Amino Acid

After training, the models will still choose codons that encode the wrong amino acid. It is unclear why the model sometimes selects a codon that encodes a different amino acid than what was originally provided as input. One reason may be that the network has seen so many examples in the training set of a particular motif that it has high confidence that another amino acid is more appropriate. Using the added heuristics (see Section II-E), the network is able to eliminate the selection of codons that encode the wrong amino acid and providing increased overall codon selection accuracy (see Table II).

The impact of these mis-substitutions is also unclear. Further analysis should be done to see if protein function is impacted or if changes modify the translation of this gene.

E. Fine-Tuned Models

We fine-tuned our base model on 4 different strains of *E. coli* as well as for two other bacterias: *Staphylococcus aureus* and *Yersinia enterocolitica*. Fine-tuning was done by replacing the CDS and amino acid sequence prediction layers and completely training them for 10 epochs. Results from fine-tuning can be seen in Table III.

The results of the fine-tuning show good performance on the *E. coli* bacterias and poor performance on the two other bacterias. The general many-to-one relationship of codons to amino acids was learned as evidenced by the $> 99\%$ amino acid prediction accuracy but the codon prediction accuracy can be as low as 59.28% in the case of *Y. enterocolitica*. This suggests that the recurrent portion of our network is tied specifically to *E. coli* and does not generalize well. We had hoped that we could create a network that would generalize well but this behavior can be expected as we trained exclusively on *E. coli* data.

During fine-tuning, the network overfits to the specific strains as can be seen by the high codon prediction accuracies (100% for *E. coli* strains). This could be mitigated by decreasing the number of epochs that are run for fine-tuning. The appropriate number of epochs used in fine-tuning is unknown especially as the fine-tuning datasets are significantly smaller than the datasets used to train the base models (see Table III).

Our next goal is to train our network on a variety of different bacterias. We have selected 17 genera that include 301 bacterial strains. Using this data, we believe that we can construct a network that is able to model the codon bias of a variety of different bacteria with minimal fine-tuning by training on the mixed data and by adding augmenting our network with additional prediction tasks.

IV. Conclusion

Here we have presented a method for representing the global codon bias of an organism. Previous methods, such as CAI, have provided a statistical overview of codon usage and a measure of how much a particular protein deviates from a reference set of proteins. Often, these approaches are used in order to backtranslate protein sequences such that the output nucleotide sequence matches the codon bias of a reference set of proteins. Methods such as CAI, however, are too coarse and give a very high-level picture of codon usage that does not account for protein structures that may be forming.

Our method, leveraging deep language models and translation techniques used in machine-learning, provides a rich representation of codon bias by taking looking at local and long-range contexts to inform codon selection. Using this method, we can predict the correct nucleotide sequence for given proteins with high accuracy. While we have only demonstrated this ability on *E. coli*, we believe that this approach can be generalized and applied to other species allowing for more advanced methods of backtranslation and for alternative methods for phylogenetic tree reconstruction.

Acknowledgment

The authors would like to thank Dr. Dan Ventura, Dr. Ryan Farrell, and Dr. Amin Salehi, Nozomu Okuda, Christopher Tensemeyer, Derrall Heath, Aaron Dennis, Shayla Draper and Michael Whatkins.

References

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

[2] H. Seligmann and D. D. Pollock, "The ambush hypothesis: hidden stop codons prevent off-frame gene reading," *DNA and cell biology*, vol. 23, no. 10, pp. 701–705, 2004.

[3] J. R. Coleman, D. Papamichail, S. Skiena, B. Futcher, E. Wimmer, and S. Mueller, "Virus attenuation by genome-scale changes in codon pair bias," *Science*, vol. 320, no. 5884, pp. 1784–1787, 2008.

[4] S. L. Chen, W. Lee, A. K. Hottes, L. Shapiro, and H. H. McAdams, "Codon usage between genomes is constrained by genome-wide mutational processes," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 10, pp. 3480–3485, 2004.

[5] P. M. Sharp, L. R. Emery, and K. Zeng, "Forces that influence the evolution of codon bias," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 365, no. 1544, pp. 1203–1212, 2010.

[6] A. Hilterbrand, J. Saelens, and C. Putonti, "Cbdb: the codon bias database," *BMC bioinformatics*, vol. 13, no. 1, p. 62, 2012.

[7] H. M. Salim and A. R. Cavalcanti, "Factors influencing codon usage bias in genomes," *Journal of the Brazilian Chemical Society*, vol. 19, no. 2, pp. 257–262, 2008.

[8] E. M. Novoa and L. R. de Pouplana, "Speeding with control: codon usage, trnas, and ribosomes," *Trends in Genetics*, vol. 28, no. 11, pp. 574–581, 2012.

[9] J. B. Plotkin and G. Kudla, "Synonymous but not the same: the causes and consequences of codon bias," *Nature Reviews Genetics*, vol. 12, no. 1, pp. 32–42, 2011.

[10] P. S. Spencer and J. M. Barral, "Genetic code redundancy and its influence on the encoded polypeptides," *Computational and structural biotechnology journal*, vol. 1, no. 1, pp. 1–8, 2012.

TABLE III. Fine-tuning results using the sequence length 500, no dropout model with all weights frozen except for the prediction layers. Prediction layer weights are not reset. Reported accuracies are for codon selection accuracies only. All amino acid accuracies were $> 99\%$. The datasets for fine-tuning are dramatically smaller than those used to train the base model. The number of sequences for each dataset is reported, 80% of the dataset was used for training/validation and the other 20% used for testing.

Name	# seqs	Training	Validation	Testing
E. coli str. K-12 substr. MG1655 (NC_000913.3)	3570	100.00%	95.64%	95.73%
E. coli UTI89 (NC_007946.1)	4243	99.98%	91.66%	91.99%
E. coli 536 (NC_008253.1)	4012	99.98%	90.18%	89.43%
E. coli APEC O1 (NC_008563.1)	4307	99.98%	91.91%	91.59%
Y. enterocolitica subsp. enterocolitica 8081 (NC_008800.1)	3385	87.74%	45.22%	46.78%
S. aureus RF122 (NC_007622.1)	2318	81.19%	59.09%	59.28%

[11] M. Welch, S. Govindarajan, J. E. Ness, A. Villalobos, A. Gurney, J. Minshull, and C. Gustafsson, “Design parameters to control synthetic gene expression in escherichia coli,” *PloS one*, vol. 4, no. 9, p. e7002, 2009.

[12] R. Hershberg and D. A. Petrov, “Selection on codon bias,” *Annual review of genetics*, vol. 42, pp. 287–299, 2008.

[13] T. Ikemura, “Codon usage and trna content in unicellular and multicellular organisms.,” *Molecular biology and evolution*, vol. 2, no. 1, pp. 13–34, 1985.

[14] F. Yamao, Y. Andachi, A. Muto, T. Ikemura, and S. Osawa, “Levels of trnas in bacterial cells as affected by amino acid usage in proteins,” *Nucleic acids research*, vol. 19, no. 22, pp. 6119–6122, 1991.

[15] M. Allert, J. C. Cox, and H. W. Hellinga, “Multifactorial determinants of protein expression in prokaryotic open reading frames,” *Journal of molecular biology*, vol. 402, no. 5, pp. 905–918, 2010.

[16] S. Pechmann and J. Frydman, “Evolutionary conservation of codon optimality reveals hidden signatures of cotranslational folding,” *Nature structural & molecular biology*, vol. 20, no. 2, pp. 237–243, 2013.

[17] D. B. Goodman, G. M. Church, and S. Kosuri, “Causes and effects of n-terminal codon bias in bacterial genes,” *Science*, vol. 342, no. 6157, pp. 475–479, 2013.

[18] K. Bentele, P. Saffert, R. Rauscher, Z. Ignatova, and N. Blüthgen, “Efficient translation initiation dictates codon usage at gene start,” *Molecular systems biology*, vol. 9, no. 1, p. 675, 2013.

[19] T. Ikemura, “Correlation between the abundance of escherichia coli transfer rnas and the occurrence of the respective codons in its protein genes: a proposal for a synonymous codon choice that is optimal for the e. coli translational system,” *Journal of molecular biology*, vol. 151, no. 3, pp. 389–409, 1981.

[20] J. M. Fox and I. Erill, “Relative codon adaptation: a generic codon bias index for prediction of gene expression,” *DNA research*, vol. 17, no. 3, pp. 185–196, 2010.

[21] M. Graf, T. Schoedl, and R. Wagner, “Rationales of gene design and de novo gene construction,” *Systems Biology and Synthetic Biology*, pp. 411–438, 2009.

[22] D. M. Hoover and J. Lubkowski, “Dnaworks: an automated method for designing oligonucleotides for pcr-based gene synthesis,” *Nucleic acids research*, vol. 30, no. 10, pp. e43–e43, 2002.

[23] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

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

[25] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[26] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, *et al.*, “Biopython: freely available python tools for computational molecular biology and bioinformatics,” *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[27] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[28] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[29] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition,” *CoRR*, vol. abs/1310.1531, 2013.

[30] A. V. Lobanov, A. A. Turanov, D. L. Hatfield, and V. N. Gladyshev, “Dual functions of codons in the genetic code,” *Critical Reviews in Biochemistry and Molecular Biology*, vol. 45, no. 4, pp. 257–265, 2010.

[31] A. Hecht, J. Glasgow, P. R. Jaschke, L. Bawazer, M. S. Munson, J. Cochran, D. Endy, and M. Salit, “Measurements of translation initiation from all 64 codons in e. coli,” *bioRxiv*, p. 063800, 2016.

[32] F. Chollet, “Keras.” https://github.com/fchollet/keras, 2015.

[33] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, Y. Bengio, A. Bergeron, J. Bergstra, V. Bisson, J. Bleecher Snyder, N. Bouchard, N. Boulanger-Lewandowski, X. Bouthillier, A. de Brébisson, O. Breuleux, P.-L. Carrier, K. Cho, J. Chorowski, P. Christiano, T. Cooijmans, M.-A. Côté, M. Côté, A. Courville, Y. N. Dauphin, O. Delalleau, J. Demouth, G. Desjardins, S. Dieleman, L. Dinh, M. Ducoffe, V. Dumoulin, S. Ebrahimi Kahou, D. Erhan, Z. Fan, O. Firat, M. Germain, X. Glorot, I. Goodfellow, M. Graham, C. Gulcehre, P. Hamel, I. Harlouchet, J.-P. Heng, B. Hidasi, S. Honari, A. Jain, S. Jean, K. Jia, M. Korobov, V. Kulkarni, A. Lamb, P. Lamblin, E. Larsen, C. Laurent, S. Lee, S. Lefrancois, S. Lemieux, N. Léonard, Z. Lin, J. A. Livezey, C. Lorenz, J. Lowin, Q. Ma, P.-A. Manzagol, O. Mastropietro, R. T. McGibbon, R. Memisevic, B. van Merriënboer, V. Michalski, M. Mirza, A. Orlandi, C. Pal, R. Pascanu, M. Pezeshki, C. Raffel, D. Renshaw, M. Rocklin, A. Romero, M. Roth, P. Sadowski, J. Salvatier, F. Savard, J. Schlüter, J. Schulman, G. Schwartz, I. V. Serban, D. Serdyuk, S. Shabanian, E. Simon, S. Spieckermann, S. R. Subramanyam, J. Sygnowski, J. Tanguay, G. van Tulder, J. Turian, S. Urban, P. Vincent, F. Visin, H. de Vries, D. Warde-Farley, D. J. Webb, M. Willson, K. Xu, L. Xue, L. Yao, S. Zhang, and Y. Zhang, “Theano: A Python framework for fast computation of mathematical expressions,” *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[34] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.