DeepCodon: An LSTM Transducer for DNA codon optimization

Samuel L. Goldman David K. Yang Michael Zhang

Abstract

Optimizing foreign DNA sequences for maximal protein production in a specified host organism is an important problem for synthetic biology and biomanufacturing. Experimental results have demonstrated that simply interchanging *codons*, triplets of three DNA bases, with "synonymous" alternatives can in fact amplify protein production several-fold while holding the produced protein constant. Previous methods for codon optimization are frequency based, which cannot consider factors such as RNA secondary structure that contribute to protein expression. Here, we apply a deep learning framework to model the distribution of codons in the genome of bacteria *E. coli*. We show that our LSTM-Transducer model is able to predict the next codon of a genetic sequence with an accuracy of 0.5766 and perplexity of 2.54 on a held out test set of the *E. coli* genome, outperforming the previously state of the art frequency-based approach.

1. Introduction

The ability to express recombinant proteins, proteins artificially designed or extracted from a different species, has been pivotal in all of biology. Expression of recombinant proteins in different host organisms has applications from basic research to industrial production of commercial therapeutics (Xiao et al., 2014). Naturally, increasing the expression level of these recombinant proteins in a host organism such as *E.coli* is highly relevant as it can allow for more robust experimentation and more efficient biomanufacturing. Traditional biological approaches to increase the expression of recombinant proteins have included selecting strong promoters, enhanced cell proliferation, and codon optimization (Xiao et al., 2014; Sivashanmugam et al., 2009; Rosano & Ceccarelli, 2014; Zhou et al., 2016). In this work, we focus on codon optimization.

Despite the importance of codon optimization, computational methods for this task remain naive and *ad hoc*, unable to capture deeper complexity of a given DNA sequence such as three dimensional folding that have recently been shown to influence the optimal set of codons (Kudla et al., 2009; Cambray et al., 2018; Goodman et al., 2013).

In this work, we propose DeepCodon, a neural language model (NLM) that captures the native distribution of codon choice in a host organism with higher accuracy and lower perplexity than the previously state of the art frequency-based approach. DeepCodon allows for codon optimization of recombinant proteins, and we are optimistic that this method will lead to experimental boosts in *heterologous* expression.

In doing so, we build on a growing trend of applying deep neural models to biological sequences. While several different authors have attempted to train large neural models at the protein level, whether to produce meaningful representations of proteins (Bepler & Berger, 2019) or *de novo* designs of functionally active proteins (Brookes et al., 2019; Gupta & Zou, 2019; Riesselman et al., 2018; Sinai et al., 2017), to our knowledge this is the first work to apply neural techniques to biological sequences at the codon level.

2. Background

The central dogma describes how information is transferred from DNA to proteins. The DNA, formed of nucleotide bases $\{a, t, c, g\}$, is *transcribed* into messenger RNA (mRNA), which is more mobile and transient. This mRNA, with the help of ribosomes, is *translated* into a sequence of amino acids that becomes the protein. There are 20 different amino acids, which allows for the combinatorially large number of proteins in nature.

Each amino acid is encoded in the DNA as nucleotide triplets called codons. Because there are $4^3 = 64$ nucleotide triplets but only 20 amino acids, the same amino acid is encoded by multiple triplets with the exception of tryptophan and methionine.

To express recombinant proteins "heterologously" in organisms that do not naturally have the protein, the researcher can choose between a variety of DNA sequences that encodes the same protein. Although codon choice rarely influences the function of the protein itself, it can modulate the expression of the protein (Brule & Grayhack, 2017).

Curiously, the distribution of codons is not uniform across the genome. Instead, the distributions of codons for a given amino acid tends to be biased, with some codons favored and other codons appearing more rarely. For instance, humans encode amino acid arginine with aga 21.79% of the time, whereas *E.coli* encodes arginine with aga 2.02% of the time (Table 2). Research attributes this to the abundance of codon-specific tRNA's, molecules that selectively translate codons into their corresponding amino acid. Codon usage frequency is correlated with tRNA abundance, and thus offers an explanation for this native bias in different genomes (Rosano & Ceccarelli, 2014). Due to these differences in native distribution, a sequence of DNA often requires codon optimiziation before insertion into a new host organism, especially if the goal is to maximize expression.

Amino Acid	**Codon**	***Homo Sapiens***	***E. Coli***
Arginine (R)	CGT	8.15%	38.12%
	CGC	15.69%	40.08%
	CGA	11.43%	6.39%
	CGG	19.13%	9.75%
	AGA	23.81%	3.65%
	AGG	21.79%	2.02%

Table 1. The distribution of codon choice for amino acid Arginine shown for Human and *E.coli* genomes. This is constructed from coding sequences retrieved from NCBI on May 11, 2019.

However, experimental results from Goodman et al. show that using rare codons instead of common ones at the beginning of the sequence can lead to roughly 14-fold increase in expression (Goodman et al., 2013). Downstream analyses suggest that rare codons can alter secondary structure at the N terminus (start of sequence), which can affect expression levels. This is consistent with previous work that found that three-dimensional secondary structure of the mRNA particularly in the beginning of the sequence, can affect protein expression (Kudla et al., 2009). These experimental results suggest that considering only the frequency of codons when redesigning a sequence for heterologous expression is insufficient, as modeling secondary structure requires long-term dependencies in the RNA.

Given protein expression data, a supervised approach can identify important motifs or features that can lead to high expression in a given host organism. However, large-scale labeled data of this kind is often unavailable for different host organisms, preventing such approaches from directly generating sequences with desired properties.

Several models rely on the assumption that the endogenous genes of a prospective host organism have already been codon optimized. In this work, we seek to use this assumption and apply a more sophisticated unsupervised approach that can learn the underlying distribution of an organism's native codon choice. Using these rules, our model can, in

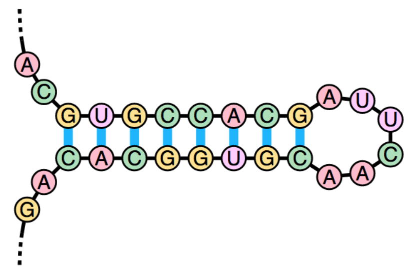

Figure 1. Diagram of a stem-loop, an example of an RNA secondary structure. Modeling this would require knowledge of nearby residues (Wikipedia, 2019).

turn, reverse translate a protein sequence of amino acids to a sequence of codons that more closely matches the underlying distribution of the target host organism. In this work, we focus on modeling the next codon choice for *Escherichia coli* (*E. coli*), a popular expression vector for heterologous expression.

3. Related Work

Related computational methods to perform codon optimization rely primarily upon the natural codon usage bias of the target organism. Sharp and Li pioneered this effort by using the codon frequencies present in a subset of genes to create a "codon adaptation index" (CAI), which they claimed would lead to high amounts of heterologous expression (Sharp & Li, 1987). Further platforms for codon optimization such as "JCat" build upon the CAI approach, offering slight *ad hoc* measures to avoid certain domains in the sequence (Grote et al., 2005). Another web-based application, "Optimizer" offers three different optimization methods: substitution with the most common codon for the target organism, sampling from the natural frequency distribution, and selective substitution of certain user-defined codons in their input sequence (Puigbo et al., 2007). Codon optimization tool "COOL" combines previous variations and offers users the choice between the codon adaptation index, the codon usage table, and considering codon choice pairings (Chin et al., 2014). While these methods add incremental boosts in improvement, but do not add new modeling ideas.

A recent platform, "Presyncodon" claims to use machine learning in order to incorporate local amino acid context to guide codon choice (Tian et al., 2017). Each amino acid window of sizes 3, 5, and 7 is mapped to the codon of the middle amino acid. Using this mapping as a look-up table, the model trains a random forest classifier to learn how *E. coli* would select a codon for an exogenous protein. The authors report a 97% accuracy in predicting the correct codon choice when trained with 64 available *E. coli* genomes and tested on another held-out, *E. coli* genome. Despite their high accuracy, this evaluation metric is significantly flawed as the held-out strain likely is very similar to the strains in the train set.

Parallel to this computational line of work, significant experimental efforts have demonstrated the huge boost in expression that is possible with successful codon optimization. Specifically, Goodman et al. measure the expression of over $14,000$ different DNA sequences, choosing to primarily vary the first 11 codons of the sequence. They demonstrate that using rare codons at the start of the sequence, codons that would almost never be selected by the above methods, can lead to roughly 14 fold increase in expression (Goodman et al., 2013). Following up on this work, Napolitano et al. try to systematically replace only arginine-coding codon sequences with rare alternatives, demonstrating the degree to which mutations to codons in the sequence are possible (Napolitano et al., 2016). Even more recently, in a massively parallel assay, Cambray et al. test the expression of $244,000$ different sequences in *E. coli* and attempt to explain the expression of sequences using a combination of the protein being translated, codon choice, and different sequence properties such as the content of DNA bases $\{g, c\}$ in each sequence. While they do implicate codon usage and the role of secondary structures, they are surprisingly unsucessful at explaining the total variance in protein expression between sequences, capturing only 53% of the variation with their features (Cambray et al., 2018).

Current computational models fail to account for the secondary structure or position of the codon within the sequence, despite the clear importance of these features in protein expression. In order to more completely take these non-linear factors into consideration, we build an end-to-end neural language model that can generate highly expressive DNA for input proteins.

We train our next-codon model with the genes of *E. coli* and test on a held-out set of genes, ensuring that there is sufficient difference between the train and test set genes. We examine the model's output patterns to see whether it can capture E. coli's codon usage bias, as well as its tendency to place rare codons in the beginning of the sequence. Furthermore, we evaluate our model's ability to generate structurally similar sequences to the true target.

4. Model

We propose codon optimization through the lens of an NLM problem. Let X be a sequence of N amino acids $x_1, \ldots, x_N$ defining a single gene. We are interested in producing codons $Y = [y_1, \ldots, y_N]$ corresponding to each position, such that the sequence Y is functionally preserving for X. Furthermore, we specify that we want to produce Y that leads to a certain amount of protein expression z in target organism s. Thus, we try to model $P(Y|X, z, s)$. In this work, we make the simplifying assumption that all native genes have been evolutionary optimized, such that if some set of codons appears in a genome $Y \in genome(s)$, then we assume for our training set that this codon has maximal expression, z_{max}. Therefore, we consider the set of an organism's genes to be a training set of X, Y, z_{max} and we try to model $p(Y_i|X, Y_{<i}, s, z_{max})$.

To model this distribution, we employ and ultimately compare several different architectures operating on both the codon and individual DNA base level.

4.1. n-gram

We first consider the use of a simple n-gram language model (LM), introducing notation for language modeling at the codon level. Inspired by the "Presyncodon" approach that claims 97% accuracy, we attempt to model the codon choice for amino acid x_i using the context of $\lfloor n - 1 \rfloor$ amino acids to both sides. Specifically, given a sequence of amino acids X and corresponding codons $y_{1:N} = [y_1, \ldots, y_N]$, we model the the joint probability of the sequence as $p(y_{1:N})$ as

$$p(y_{1:N}) = \prod_{i=1}^{N} p(y_i | x_{i-(n/2):i+(n/2)})$$

However, while n-grams are computationally simple, we face potential issues with respect to sparsity and modeling constraints. Due to the counts-based nature of n-gram modeling, during evaluation new combinations of n amino acids not encountered in training would be assigned probability 0. In addition, we are limited to sequence dependencies within size n windows. Still, this provides a benchmark that we consider to be state of the art based upon the reviewed literature, as it only considers local frequency based measurements to model the rest of the codon choices.

4.2. Transducer Architecture

In our neural models, we want to consider both information at the codon level and at the amino acid level. While knowledge of previously predicted codons to inform the next codon prediction is well motivated by language models, we also posit that our model should always be aware of amino acid x_i when predicting codon y_i. If the model has no information about amino acid, x_i, the model will not only be tasked with modeling the distribution of next codon, but also the distribution of next amino acid, which is far less constrained. By transducing information about the current amino acid, we disentangle our problem from that of amino acid language modeling. Furthermore, we can design more complex models that pass in contextual information from across the amino acid $[x_1, ..., x_N]$. This can allow our model types to incorporate long range dependencies of the sequence.

To give our models these desired properties, we use a variant of the transducer architecture primarily developed for converting speech to text (Graves, 2012). While traditionally motivated by problems such as audio transcription which rely on continuous or streaming possibly variable-length outputs, our variation creates a positional encoding at the amino acid sequence level X and uses this in combination with a decoder that possesses information at the predicted codon-level, that can have information about all codons predicted so far. We use a joint network, a single linear layer, in order to produce the prediction at the i^{th} time step taking into account our encoding of the amino acids and encoding of the previously predicted codons. We visualize this in Figure 2. More formally, if we refer to $Enc(\cdot)_t$ as the t-th output encoding among a vector of N encodings produced by our encoder network Enc, our prediction network as $Pred$, and our joint network as G, then we get:

$$p(y_t|X, y_{1:t-1}) = \text{softmax}(G(Pred(y_{1:t-1}), Enc(X)_t))$$

We describe our different Encoder and Prediction network architectures below, differentiating between them by referring to encoder networks as Amino Acid (AA) networks and Prediction networks as Codon networks.

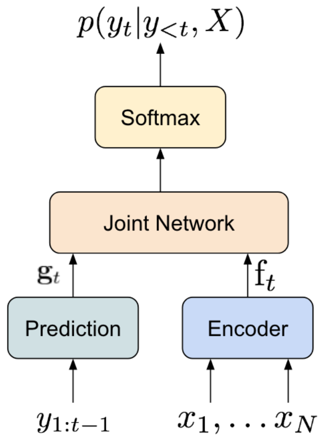

Figure 2. The transducer architecture, which also serves as the architecture for DeepCodon. An encoder network (LSTM) maps input amino acids $x_1, \ldots x_{,N}$ to produce transcription vector $\mathbf{f}_n$, which gets concatenated to outputs from a separate prediction network taking in previous outputs $y_{1:t-1}$ as input for language modeling.

4.3. Encoder Networks

Identity AA: In the most simple case for our encoder network, we create an identity mapping of our input amino acids. That is, our amino acid encoder outputs one hot vectors such that f_i, the encoding used to predict the i^{th} codon is onehot(x_i). In the absence of a more complicated encoder, this simple encoding ensures that our model can learn to model the codon distribution at y_i, rather than the distribution of possible amino acids at position i. We describe this:

$$Enc(X) = [\text{onehot}(x_1), \text{onehot}(x_2) ... \text{onehot}(x_N)]$$

In the absence of a sophisticated prediction network, this model is equivalent in power to the unigram, frequency based model.

Conv AA: To use further contextual amino acid information, we take convolutions over a sequence of amino acids to produce local encodings f_i.

In this setup, we take convolutions, or sliding windows of width w with stride 1 over our amino acid sequence X, which has 23 input channels, corresponding to our size 23 one-hot encoding of the amino acids such that the output of this encoding network is a d-dimensional dense vector, and each local encoding $f_i \in \mathbb{R}^d$. In the absence of a sophisticated prediction network, this model is equivalent in power to the n-gram frequency based model, where n is defined by the receptive field, w of the local convolutions. We note that in this case, we pad our input sequence to ensure we get an output encoding vector of length N $[f_1, f_2, \ldots, f_N]$.

$$Enc(X) = [Conv(x_{1-w/2:1+w/2}), \ldots, Conv(x_{N-w/2:N+w/2})]$$

BiLSTM AA Next, we define a Bidirectional LSTM that can model long-term dependencies, and tackle the limitations of n-gram models. Specifically, we consider a BiLSTM that explicitly takes into account information of local and distant amino acids. LSTM Networks are an especially promising model choice for biological sequence modeling, which may include components such as transcription factor binding sites that modulate gene expression and involve the interaction of DNA base-pairs several tens of codons away (Sønderby et al., 2015). We choose to use bidirectional models such that the encoding produced by this architecture can capture sequence level information at the whole sequence for a single encoding position $Enc([x_1, x_2, ..., x_N])_i$.

4.4. Prediction network

LSTM Codon Next, we consider different models for the prediction network, that can take into account information about all previously outputted codons. First, we consider a standard one-directional LSTM that is able to produce hidden hidden representations:

$$\begin{aligned} Pred([y_0, y_1, .., y_{n-1}]) &= LSTM([y_0, y_1, .., y_{n-1}]) \\ &= [h_0, h_1, .., h_{n-1}] \end{aligned}$$

Here, each output of the prediction network is concatenated with the output of the encoder, which is fed as the input to predict the next codon. We show a schematic of this LSTM network in Figure 3.

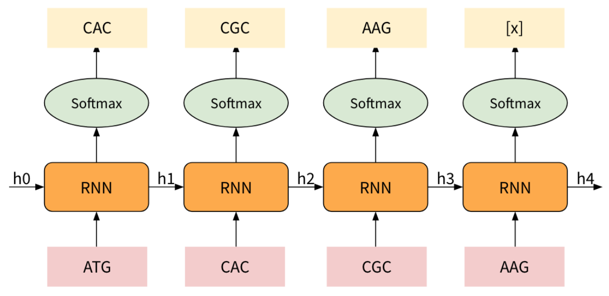

Figure 3. Schematic of our neural language model applied to predicting codons. To predict the next codon, the entire history of predicted codons is incorporated.

LSTM Codon (Char) In addition to the above codon-level language models in the predictive network, we looked into the analogous character-aware predictive network models. Inspired by (Kim et al., 2016), we desire a model that is able to take in character-level information in order to predict the next word. In this case, words are the codons, what we want to predict, and characters are the nucleotide base pairs $\{a, t, g, c\}$, that compose each predicted codons. In doing so, we want our model to be able to capture structural similarities between codons that share nucleotides.

Using part of the model proposed by Kim et al., we decompose the previously predicted codons $[y_1, .., y_{i-1}]$ into their respective size 3 characters, such that this sequence becomes $[y_{1,1}, y_{1,2}, y_{1,3}, y_{2,1} .., y_{i-1,3}]$, where $y_{ij} \in \mathbb{R}^4$. For this new model, we run a CNN over character level inputs with fixed width 3 and stride 3 with hidden dimension h, such that we compress our sequence into a hidden representation $H \times (i-1)$. We note that unlike Kim et al., because each word has a constant number of characters, we need not pool over time. We directly pass this sequence into an LSTM level model, similar to the LSTM Codon model, which then produces the hidden state h_{i-1} passed into the joint prediction network.

4.5. Training constraints for optimization

Apart from our model architectures, we also employ teacher-forcing for recurrent models at test time and codon-level masking at both test and prediction in order to constrain our prevent biologically implausible outputs. We briefly describe these techniques.

Teacher-forcing As described in (Williams & Zipser, 1989), teacher-forcing is a technique used in recurrent models that replaces the output generated by the training model at previous timesteps with the actual or expected output from the training dataset. Such strategies may improve model performance in the face of model instability and slow convergence, where we avoid propagating a possibly incorrect early output as input to prediction outputs down the line. In the context of codon optimization, specifically our prediction network, we often condition on previously predicted codons $[y_1, y_2, .., y_{i-1}]$ in order to predict the i^{th} codon y_i. At test time, we replace our previously output codons $[y_1, y_2, .., y_{i-1}]$ with the codon observed in our ground-truth dataset.

Codon-masking In addition to teacher-forcing, we further incorporate knowledge that only a subset of all possible DNA codons is biologically plausible for each target amino acid. While we hope that our transducer architecture prompts our model to quickly learn this, we also mask all biologically impossible codon choices. For example, if we encounter the amino acid Proline, then our model should only consider the subset {CCT, CCC, CCC, CCG} as outputs, avoiding all other potential DNA triplets due to knowledge that no other codons can code for Proline. We hardcode these associations using a reference to a standard genetic code table at both test and prediction time.

5. Methods

5.1. Dataset

We model the codons of Escherichia coli strain K-12 substrain MG1655 (Accession: NZ_CP032667) (Earl et al., 2018). Since we are only interested in protein-coding regions, we extract the coding sequences (CDS), freely available from the NCBI database. We filter by sequences with length divisible by 3, which retrieved 4388 CDS. This provides a dataset with 4067532 nucleotides and 1355844 codons total, with variable sized sequences (Figure 4).

5.2. Data Pre-processing

Each sequence was tokenized into triplets of nucleotides (codons) with a START token added into the beginning of each sequence. A corresponding amino acid sequence was generated for each sequence by converting the DNA triplet into an amino acid token. For downstream modeling, each

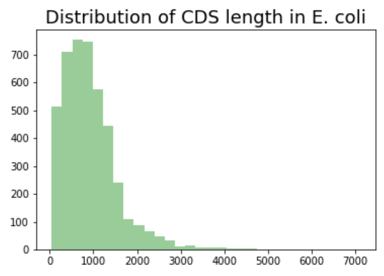

Figure 4. Histogram depicting length of 4388 exracted CDS sequences.

codon was encoded in a 67-size one hot encoding and the amino acids were encoded into a 23-size one hot encoding, where the extra three tokens come from the addition of pad, $start$, and unk tokens.

The sequences were separated into 70/30 split, resulting in 3072 genes in the training set and 1316 genes in the test set.

5.3. Implementation Details

All models were implemented in PyTorch and trained models using Google Colaboratory. Our code can be viewed here. We implemented unigram, trigram and 5-gram models as baselines. Each model was trained with a batch size of 10 for a maximum of 30 epochs. Teacher forcing was used for both training and testing. Loss was calculated by taking the cross entropy loss between the predicted codons and the target codons, and trained using Adam with learning rate $1e-3$. We describe further model parameters in the Appendix.

5.4. Evaluation Metrics

To evaluate performance, we compared both accuracy and perplexity (PPL) across all trained models. Accuracy was calculated by generating a reverse translated DNA sequence given a protein, and examining how many of the codons were predicted correctly. We note that in a biological context, we imagine sampling from this distribution, giving credence to our metric of perplexity.

5.5. Secondary structure prediction

We further evaluate full sequence outputs of our models on the secondary structure of the predicted sequences. Previous experimental studies have shown that the secondary structure of the beginning of sequences influences expression levels. More specifically, Goodman et al. found high correlation between the structure of the first 120 bases of the RNA and the expression level (Goodman et al., 2013). Thus, we predict the secondary structure, measured by the minimum free energy (MFE), of the first 120 bases of the generated sequence by our models. MFE prediction is done through the ViennaRNA package (Lorenz et al., 2011).

6. Results

We present our results with respect to accuracy and perplexity (PPL) in Table 6. Across all models, our models that use LSTM at the codon level and a BiLSTM at the amino acid level perform best, producing the highest accuracy and lowest PPL on the test split. We achieve marginal accuracy improvements over existing n-gram models, with DeepCodon (LSTM Codon + BiLSTM-T) achieving best in class performance. All neural models achieved competitive perplexity against n-gram models, suggesting advances in capturing evolutionary “optimized” codon synonyms in E. coli.

We also note that character-aware codon models do not significantly improve evaluation on test, which could indicate that our word level models are able to learn the relationships between nucleotides despite performing at a codon level.

Additionally, we test our LSTM Codon + BiLSTM-T model without teacher forcing in order to evaluate the effect of long term dependencies within predicted codons. We find that DeepCodon still achieves 0.5486 test accuracy when tested without teacher forcing, indicating that dependencies on previously produced codons do not dominate model predictions.

Lastly, we interrogate the necessary receptive field of the BiLSTM over the amino acid sequence. We do this by comparing the LSTM Codon + BiLSTM-T model perplexities with those of the LSTM Codon + Conv-T model with a receptive field of 5 amino acids around the predicted codon. We see that this receptive field of size 5 is not enough to match performance of the BiLSTM, motivating the need for capturing long term information over the amino acid sequence.

6.1. Secondary structure of predicted mRNA

While accuracy and perplexity test our models at a single codon level, we are ultimately interested in designing full codon sequences. To assay our performance on this, we estimate the minimum free energy (MFE) of the sequences outputted by our model. Specifically, we compare the MFE between our model outputs and the true target in the test set. We do this comparison with Spearman rank correlation (Spearman’s rho) between the target MFE and the predicted sequence MFE values (Table 3).

We find that LSTM Codon + BiLSTM-T AA model has similar rank correlation to the true targets as the unigram model. When the LSTM model was predicted with teacher

Model	**Train Acc**	**Test Acc**	**Train PPL**	**Test PPL**
Unigram AA	0.5177	0.5139	3.0375	3.0321
Trigram AA	0.5581	0.5488	2.9387	2.9482
5-gram AA	**0.8449**	0.3540	**2.1416**	3.1120
1, 3, 5-gram AA	0.8325	0.5272	2.6332	3.0074
BiLSTM AA	0.5885	0.5626	2.5333	2.6214
LSTM Codon	0.5521	0.5492	2.6592	2.6675
LSTM Codon + BiLSTM-T AA	0.6125	0.5766	2.4043	**2.5400**
LSTM Codon + Conv-T AA	0.5672	0.5648	2.5687	2.5838
LSTM B	0.6144	0.5384	2.3968	2.7409
LSTM B + BiLSTM-T AA	0.6129	**0.5777**	2.3965	2.5459

Table 2. Evaluation metrics for our neural and n-gram models. AA denotes amino acid-level input, B denotes DNA base-level input. We note that transducer architectures (T) achieved lowest perplexity (PPL, the lower the better) and highest accuracy during testing. We include full model parameters and descriptions in the Appendix.

Model	**Spearman's rho**	**p-value**
Unigram	0.573	5.60e-119
LSTM without TF	0.580	8.39e-116
LSTM with TF	**0.658**	**3.85e-164**

Table 3. Rank correlation between the MFE values of target sequences and model predicted sequences in the test set. LSTM refers to LSTM Codon + BiLSTM-T AA model, while TF refers to whether teacher forcing was used during test time.

forcing at test time, the correlation increased significantly.

Increased MFEs are associated with increased expression. We compare the distribution of the MFE of the model outputs in Figure 5. Similar to the correlation, we find that LSTM and unigram models outputs have comparable mean MFE outputs, while LSTM with teacher forcing has a higher MFE value than the two.

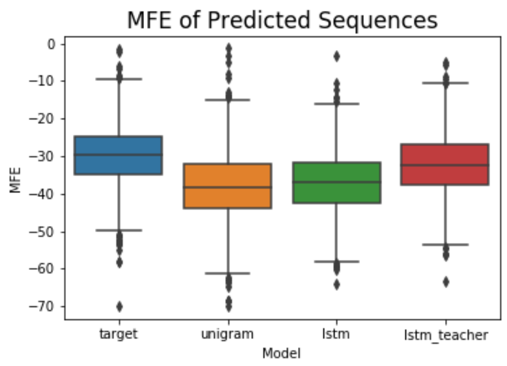

Figure 5. MFE of model outputs, represented through a boxplot. Lstm_teacher refers to teacher forcing used at test time.

7. Discussion

We introduce a neural machine translation approach to codon optimization, and present DeepCodon, a transducer model which achieves best-in-class performance on a left out test set of the *E.coli* genome. Comparing a variety of NLM architectures, we observe that all of our neural models achieve better accuracy and perplexity than n-gram and frequency baselines. Accordingly, our models provide a competitive alternative to those making up the backbone of existing codon optimization methods. Nevertheless, our improvements over trigram accuracy are surprisingly low, leading us to be cautiously optimistic about the potential for neural models in this space.

Neural language models We explore several different neural language models, being the first as far as our knowledge to frame codon optimization in the lens of a neural language model problem. Our preliminary results suggest that such methods are well-suited to the task, and we have the luxury of a variety of models developed for other NLP application domains such as in the traditional context of human language to motivate follow-on work with regard to increasing model performance. We suspect that further parameter tuning and applying a similar set of approaches to a more complex genome, such as the human genome, may lead to more stark differences in performance between frequency and neural based methods.

Biological assumptions As a critical and potential flaw of our model, we make the assumption that sequences in the *E.coli* genome are codon-optimized. While this assumption undergirds previous approaches to codon optimization, we highlight this as one potential pitfall of our model if output sequences were experimentally tested. Accordingly, utilizing biological data and real expression measurements for future models, whether for evaluation or train, could improve confidence in our codon predictions. Furthermore, we choose to model the *E.coli* genome beginning at the coding sequence. Optimizing the region shortly preceding the protein-coding sequence, the UTR, has been experimentally demonstrated to also significantly impact protein expression, and it is possible that incorporating this region could improve signal and prediction in the early parts of the protein sequence.

Secondary structure predictions One major goal of the project was to improve the prediction in the beginning of the sequence, as secondary structure in the region can significantly influence expression. We find that DeepCodon performs only marginally better than the frequency based model in predicting sequences with similar levels of secondary structure as the target.

Most interestingly, we find that teacher forcing during testing leads to a significant improvement in the MFE task. This is surprising given that DeepCodon's accuracy with the use of teacher forcing is 0.5766 and 0.5487 without. Despite the small difference in the codon level accuracy, we find that rank correlation score improves from 0.580 to 0.658.

7.1. Future work

Tuning a more global model One limitation in this work is that our training set is limited by the genes of the target organism, *E. coli*. We hypothesize that there may be global rules of codon selection that is universal between species, along with species-specific rules due to factors like codon usage bias. One strategy is to train DeepCodon through DNA sequences across domains of life first, and then fine-tuning it to the specific organism that we want heterologous expression in.

Application to other organisms While our work was focused on E. coli, mammalian protein production is often done through Chinese Hamster Ovary (CHO) cells. Relative performance of DeepCodon will likely improve as well, since CHO cells have a much larger genome which will offer more data for model training, in comparison to E. coli genome. Additionally, different species have different entropies over codon choice, and we suspect that with more complex genomes, DeepCodon may perform better with respect to a frequency baseline on that same species.

Incorporating experimental findings We hope to incorporate previous experimental findings into our model (Goodman et al., 2013; Cambray et al., 2018). Since secondary structure in the beginning seem to play a significant role, we could train a model for generating a similar structure in addition to next codon prediction.

Acknowledgments

We would like to thank Yuntian Deng and Justin Chiu for helpful discussions regarding the model setup and evaluation metrics, Eli Weinstein and Debora Marks for the initial problem suggestion as well as constant feedback, and Sasha Rush for the amazing course that allowed us to do this work.

References

Bepler, Tristan and Berger, Bonnie. Learning protein sequence embeddings using information from structure. *arXiv preprint arXiv:1902.08661*, 2019.

Brookes, David H, Park, Hahnbeom, and Listgarten, Jennifer. Conditioning by adaptive sampling for robust design. *arXiv preprint arXiv:1901.10060*, 2019.

Brule, Christina E and Grayhack, Elizabeth J. Synonymous codons: choose wisely for expression. *Trends in Genetics*, 33(4):283–297, 2017.

Cambray, Guillaume, Guimaraes, Joao C, and Arkin, Adam Paul. Evaluation of 244,000 synthetic sequences reveals design principles to optimize translation in escherichia coli. *Nature biotechnology*, 36(10):1005, 2018.

Chin, Ju Xin, Chung, Bevan Kai-Sheng, and Lee, Dong-Yup. Codon optimization online (cool): a web-based multi-objective optimization platform for synthetic gene design. *Bioinformatics*, 30(15):2210–2212, 2014.

Earl, Joshua P, Adappa, Nithin D, Krol, Jaroslaw, Bhat, Archana S, Balashov, Sergey, Ehrlich, Rachel L, Palmer, James N, Workman, Alan D, Blasetti, Mariel, Sen, Bhaswati, et al. Species-level bacterial community profiling of the healthy sinonasal microbiome using pacific biosciences sequencing of full-length 16s rrna genes. *Microbiome*, 6(1):190, 2018.

Goodman, Daniel B, Church, George M, and Kosuri, Sriram. Causes and effects of n-terminal codon bias in bacterial genes. *Science*, 342(6157):475–479, 2013.

Graves, Alex. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012. URL `http://arxiv.org/abs/1211.3711`.

Grote, Andreas, Hiller, Karsten, Scheer, Maurice, Münch, Richard, Nörtemann, Bernd, Hempel, Dietmar C, and Jahn, Dieter. Jcat: a novel tool to adapt codon usage of a target gene to its potential expression host. *Nucleic acids research*, 33(suppl_2):W526–W531, 2005.

Gupta, Anvita and Zou, James. Feedback gan for dna optimizes protein functions. *Nature Machine Intelligence*, 1 (2):105, 2019.

Kim, Yoon, Jernite, Yacine, Sontag, David, and Rush, Alexander M. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Kudla, Grzegorz, Murray, Andrew W, Tollervey, David, and Plotkin, Joshua B. Coding-sequence determinants of gene expression in escherichia coli. *science*, 324(5924): 255–258, 2009.

Lorenz, Ronny, Bernhart, Stephan H, Zu Siederdissen, Christian Hoener, Tafer, Hakim, Flamm, Christoph, Stadler, Peter F, and Hofacker, Ivo L. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

Napolitano, Michael G, Landon, Matthieu, Gregg, Christopher J, Lajoie, Marc J, Govindarajan, Lakshmi, Mosberg, Joshua A, Kuznetsov, Gleb, Goodman, Daniel B, Vargas-Rodriguez, Oscar, Isaacs, Farren J, et al. Emergent rules for codon choice elucidated by editing rare arginine codons in escherichia coli. *Proceedings of the National Academy of Sciences*, 113(38):E5588–E5597, 2016.

Puigbo, Pere, Guzman, Eduard, Romeu, Antoni, and Garcia-Vallve, Santiago. Optimizer: a web server for optimizing the codon usage of dna sequences. *Nucleic acids research*, 35(suppl_2):W126–W131, 2007.

Riesselman, Adam J, Ingraham, John B, and Marks, Debora S. Deep generative models of genetic variation capture the effects of mutations. *Nat. Methods*, 15:816–822, 2018.

Rosano, Germán L and Ceccarelli, Eduardo A. Recombinant protein expression in escherichia coli: advances and challenges. *Frontiers in microbiology*, 5:172, 2014.

Sharp, Paul M and Li, Wen-Hsiung. The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic acids research*, 15(3):1281–1295, 1987.

Sinai, Sam, Kelsic, Eric, Church, George M, and Nowak, Martin A. Variational auto-encoding of protein sequences. *arXiv preprint arXiv:1712.03346*, 2017.

Sivashanmugam, Arun, Murray, Victoria, Cui, Chunxian, Zhang, Yonghong, Wang, Jianjun, and Li, Qianqian. Practical protocols for production of very high yields of recombinant proteins using escherichia coli. *Protein Science*, 18(5):936–948, 2009.

Sønderby, Søren Kaae, Sønderby, Casper Kaae, Nielsen, Henrik, and Winther, Ole. Convolutional lstm networks for subcellular localization of proteins. In *International Conference on Algorithms for Computational Biology*, pp. 68–80. Springer, 2015.

Tian, Jian, Yan, Yaru, Yue, Qingxia, Liu, Xiaoqing, Chu, Xiaoyu, Wu, Ningfeng, and Fan, Yunliu. Predicting synonymous codon usage and optimizing the heterologous gene for expression in e. coli. *Scientific reports*, 7(1): 9926, 2017.

Wikipedia. Stem-loop — Wikipedia, the free encyclopedia. `https://upload.wikimedia.org/wikipedia/\commons/thumb/3/3f/Stem-loop.svg/\440px-Stem-loop.svg.png`, 2019. [Online; accessed 11-May-2019].

Williams, Ronald J and Zipser, David. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Xiao, Su, Shiloach, Joseph, and Betenbaugh, Michael J. Engineering cells to improve protein expression. *Current opinion in structural biology*, 26:32–38, 2014.

Zhou, Zhipeng, Dang, Yunkun, Zhou, Mian, Li, Lin, Yu, Chien-hung, Fu, Jingjing, Chen, She, and Liu, Yi. Codon usage is an important determinant of gene expression levels largely through its effects on transcription. *Proceedings of the National Academy of Sciences*, 113(41): E6117–E6125, 2016.

8. Appendix

8.1. Model-specific implementation

Here we further describe specific parameters and implementation details of our models.

n-grams For each n-gram model, we calculate the full distribution of the middle codon for ever n gram set of amino acids. The $1, 3, 5$-gram denotes an ensemble method with the above, weighting the predictions of a unigram, trigram, and 5-gram model equally for final output.

BiLSTM AA We used a standard bidirectional LSTM architecture, with 3 hidden layers of 200 hidden nodes each and 0.2 dropout.

LSTM Codon We used a standard bidirectional LSTM architecture, with 3 hidden layers of 200 hidden nodes each and 0.2 dropout.

LSTM Codon + BiLSTM-T AA We used a 2-layer 300-hidden dimension LSTM with 0.2 dropout for both the codon input model and the bidirectional LSTM on the amino acid level.

LSTM Codon + Conv-T AA We used the same model architecture as above for the codon model. For our convolutional model, we used a CNN with kernel size 5, 50 outchannels, and 0.1 dropout, converting this to 100 hidden dimensions with a feedforward network before concatenating with the prediction network.

LSTM B We employed a character-level LSTM with 2 hidden layers and 200 hidden dimensions. Here we used teacher forcing, linear dropout 0.2, LSTM dropout 0.3, and convolutional dropout 0.3.

LSTM B + BiLSTM-T AA The DNA base input model used a 1-layer 200-dimension LSTM with linear dropout 0.2, LSTM dropout 0.3, and convolutional dropout 0.3. In addition, we used a 2-layer 50-dimensional LSTM with dropout 0.3 for the BiLSTM.

Joint Network For the joint network, we use a single linear layer and apply softmax to produce codon prediction.