# Genome Dreaming

Akshay Maheshwari, Bohan Wu, and Oğuz H. Elibol

**Abstract**—We investigate and build the framework required to generate or "dream" of genomes that encode novel organisms. Importantly, we find that developing methods to validate the generated sequences is as important as building a good model to realize this task due to the lack of methods capable of interpreting and evaluating a generated sequence. We report on our results in building and optimizing a generative model (using Recurrent Neural Networks) and novel validation and visualization techniques that enable model evaluation and unsupervised discovery and comparison of gene properties to form a testable hypothesis for the generated sequence.

## 1 Introduction

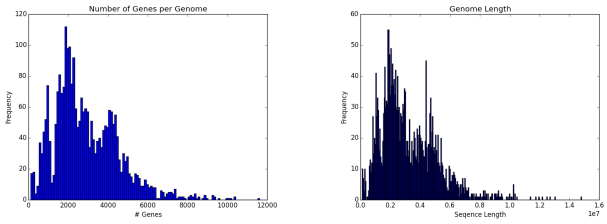
The engineering of biological organisms is limited by a lack of ability to rationally design DNA that functions in a predictable and desirable way. Synthetic biologists have attempted to overcome this over the last two decades by creating standardized genetic components with quantified dynamics that can be composed into genetic circuits [1]. This has led to a revolution in the engineering of biological organisms with applications ranging from memory storage and computation to environment and medicine. However, the scale and potential of these circuits is largely limited by a lack of understanding of genes and their complex interactions. To eventually enable the specification of genome-scale DNA and consequently the precise, large-scale forward engineering of entire novel organisms with human-desired functions, substantially greater integration of latent genome properties and intra-genome dependencies are needed in the design of genetic sequences.

Here we establish a preliminary framework for such an approach within the subspace of prokaryotic organisms. We focus on two tasks: 1) creating a model that captures the properties of prokaryotic genomes in an unbiased way and that can be used to generate new genetic sequences with predictable properties; and 2) interpreting and evaluating learned representations of genomes and generated sequences. For the first task, we trained nucleotide-level generative recurrent neural networks (RNNs) on whole-genomes of specific prokaryotic phyla and generated putative nucleotide sequences with the length of a typical gene. We assess the performance of the RNN models by evaluating their ability to accurately classify the phylogeny of an unseen genome and compare this performance to that of a baseline n-gram model. For the second task, we analyze and cluster labeled genes from Escherichia coli based on their hidden layer representations, which are generated by running sequences through an Escherichia-trained RNN model.

Most of the recent work in computational genomics focuses on discriminative models to classify particular features of the genome, such as transcription factor binding regions, and ab initio gene predictions [2], [3]. In contrast, this work investigates the feasibility of something different – learning a generative model capable of capturing genome properties with the goal of enabling the design of genetic sequences with desired characteristics.

## 2 Data Preprocessing and Statistics

We chose to use prokaryotic organisms because of their relatively small and simple genomes. One curated resource for genome sequences is the Kegg Database, which contains FASTA-format files with (a) complete unannotated prokaryotic genomes and (b) lists of annotated coding-sequences for each prokaryotic genome. We extracted all 4131 files corresponding to (a) and 4131 files corresponding to (b) and created json files that structure the genomes and their genes into proper phylogenetic categories with additional sequence metadata, such as location and orientation of genes. For simplicity, we removed organisms that contained multiple chromosomes or extrachromosomal DNA from the dataset. The median length of genomes in this subset was 2780346 nucleotides and the median number of genes per genome was 2558 genes/genome (Figure 4).




(a) Number of genes per genome (b) Genome length

Fig. 1. Statistical properties of the prokaryotic genome data

## 3 Modeling

Creating a novel sequence that is valid requires a robust generative model that can capture latent properties of

• *akshaym@standord.edu, bohanwu@stanford.edu, elibol@stanford.edu*

*We would like to thank the following people and others for their valuable discussions and guidance: Dr. Drew Endy, Dr. Anshul Kundaje, and Dr. James Zou; Namrata Anand, Jesse Zhang, Bo Wang, Volodymyr Kuleshov, and members of the Endy lab*

genomes. We use recurrent neural networks for this task and validate their abilities by gauging their performance in learning taxa-specific properties and by comparing this performance against the performance of a baseline n-gram model.

### 3.1 Deep Recurrent Neural Networks

Recurrent neural networks (RNNs) and their variations, such as Gated Recurrent Units (GRUs) and Long Short Term Memory networks (LSTMs), have become the standard for modeling and generating sequences because of their ability to capture context-dependent long-range sequence interactions. We trained nucleotide-level LSTMs on the genomes of prokaryotes within specific taxa with the goal of generating novel genomes that could encode an organism representative of the taxa the model was trained on.

We have set up a base level model as shown in Figure 2. A sequence unroll length of 50 was used for our RNN model.

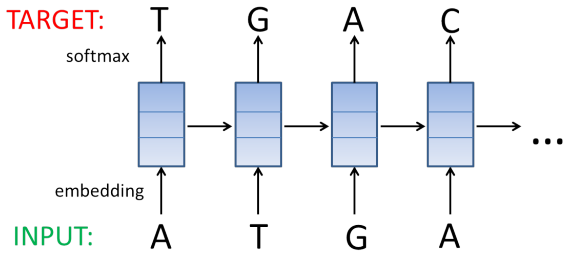



Fig. 2. Diagram for the base level recurrent neural network model used in this work

Inputs to each LSTM time block was a base embedding vector ($\mathbb{R}^4$). The number of hidden layers was limited to 2 or 3 depending on the experiment. Also the size of the hidden layer vector was chosen as either 128, 512 or 1024 depending on the experiments as explained later. The correct base output at each time step was predicted using a softmax layer (equation 1 ) connected to the final hidden layer h.

$$P(y=j|h) = \frac{e^{h^T w_j}}{\sum_{k=1}^{4} e^{h^T w_k}} \quad (1)$$

The network was trained by back-propagating the cross entropy error for the correct target base, allowing the model to learn a genome sequence model.

To validate that the model has learned genome and taxa-specific properties, we optimized and evaluated our model using two different methods: first, by using a validation set of whole genomes belonging to the specific taxa from which training genomes were derived; and second, by using an intragenome training-testing split, where 90% of a given genome was used in the training set and 10% in the validation set. We minimized the average cross entropy loss of genomes (equation 2) and used perplexity (equation 3) as a measure of model performance:

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N} \ln p_{\text{target}_i} \quad (2)$$

$$\text{perplexity} = e^{-\frac{1}{N}\sum_{i=1}^{N} \ln p_{\text{target}_i}} = e^{\text{loss}} \quad (3)$$

After experimenting with different model parameters we found that the model with 1024 neurons and 3 hidden layers achieved the best test perplexity with the intragenome training-test split. For these experiments multiple Azure and Amazon machines with GPUs were deployed which ran for multiple days (for complex models and large data). Figure 3 shows the model optimization and tuning process. We first started with a moderately complex model consisting of 2 hidden layers and a hidden vector size of 128 and trained it on a single genome (Figure 3 a and b). The model capacity was then increased to 512 or 1024 vector size while keeping the input data the same. This resulted in overfitting the data (Figure 3 c and d) relatively quickly as indicated by the validation curve diverging from the training curve. Next, more data was introduced to the model (7x genomes from the same family) to avoid over-fitting (Figure 3 e and f). We were able to reduce average validation perplexity to 3.5 by tuning the model. We have observed that using a 10% validation set may give overly optimistic results and result in not generalizing to other genomes, further highlighting the importance of choosing an appropriate validation and test set for the task in hand. Because we are mainly concerned about learning a model that will generalize to all genomes, we conclude that using a whole genome validation set makes more sense.

We next evaluated the ability of the model to capture genus-specific properties while still retaining the ability to generalize, by measuring perplexity of several Escherichia and Staphyloccus genomes fed forward into a tuned, Escherichia-trained RNN and a tuned, Staphylococcus-trained RNN (Table 1).

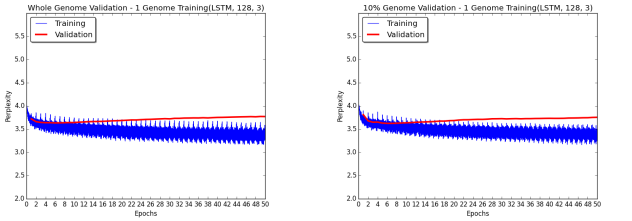
TABLE 1
Perplexities obtained for two different models, one trained on Escherichia genomes (ECO, ENA, ELC, ELX, ECOJ, EFE, ECOL) and the other on Streptococcus genomes (SSIF, SHU, SEPP, SAUM, SAO, SAU, SAUT). Each row is a genome that the model has not seen, but is from the same family of either Escherichia or Streptococcus. For reference - since there are four possible nucleotides, the perplexity of a model that does random guessing would return a perplexity of 4.

| | Escherichia Model | Streptococcus Model |
|---|---|---|
| ECL (E) | 3.59998 | 4.17365 |
| ESO (E) | 3.62222 | 4.1728 |
| EBL (E) | 3.56904 | 4.17352 |
| EAL (E) | 3.63504 | 4.15488 |
| SAH (S) | 3.67812 | 3.42402 |
| SXY (S) | 3.68177 | 3.47255 |
| SEQO (S) | 3.68418 | 3.47396 |
| SAUR (S) | 3.67919 | 3.43899 |

Based on these results, we see that the model generalizes better for the genomes that are in the same family than for genomes in a different family.

### 3.2 N-gram model

For baseline comparison, we used a (1-5)-gram representation of genomes. The (1-5)-gram model characterizes genomes based on the frequency of all possible subsequences of at most 5 nucleotides in length (a $4^5 + 4^4 + 4^3 + 4^2 + 4$ feature embedding). An SVM using the (1-5)-gram genome representation was able to achieve 89% classification accuracy of Staphylococcus, Escherichia, and Mycoplasma; and was




(a) Model Capacity: Medium (128x2), Training Data: Medium (1 genome), Validation Data: different genome

(b) Model Capacity: Medium (128x2), Training Data: Medium (1 genome), Validation Data: Holdout 10%

(c) Model Capacity: High (512x3), Training Data: Medium (1 genome), Validation Data: different genome

(d) Model Capacity: High (512x3), Training Data: Medium (1 genome), Validation Data: Holdout 10%

(e) Model Capacity: High (512x3), Training Data: High (7 genomes), Validation Data: different genome

(f) Model Capacity: High (1024x3), Training Data: High (7 genomes), Validation Data: Holdout 10%

Fig. 3. Models with varying capacity and varying amount of data (rows) and using different validation sets (columns). Left column: separate genome from the same family as a validation set. Right column: 10% holdout of the genomes as a validation set. Top row: medium capacity model with single genome training data. Middle row: High capacity model with single genome training data. Bottom Row: High capacity model with 7 genomes training data

able to achieve 34% classification accuracy of alpha, beta, gamma, gamma-enterobacteriae, delta, epsilon, and "other" members of the Protobacteria phylum.

### 3.3 Sequence Generation

Finally, we used the Escherichia-trained RNNs to generate sequences 1000-2000 nucleotides in length. These sequences were constrained to start with a start codon and only contain a stop codon at its end to make them gene-like. Properties of the generated proteins from the generated genes are shown below in Figure 4 (using Open Predict Protein [4]). During generation, instead of selecting the base with the highest probability, we selected a base with a probability that is proportional to the probability of the base to allow some variance in the generated sequences.

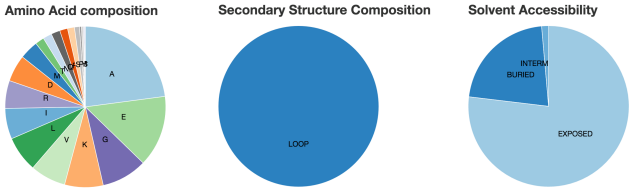



Fig. 4. Properties of the protein from the generated gene

## 4 Visualization and Validation

Verifying the quality and functionality of a generated sequence is difficult because no standard in silico gene simulator or compiler exists. Here we establish a series of methods to evaluate the quality, and properties of generated sequences.

### 4.1 Information Extraction from Hidden Layers

We hypothesized that the final hidden layer in trained RNNs should contain high-level latent genome properties that can be used to infer the properties of unlabeled sequences. The final hidden layer is used to generate softmax outputs to predict the next nucleotide in a sequence and so should contain information about the properties of sequences sufficient to perform this prediction task. Indeed, the ability of final hidden layers to capture high-level representations of the information they were trained on has been shown in a variety of image and language tasks [5].

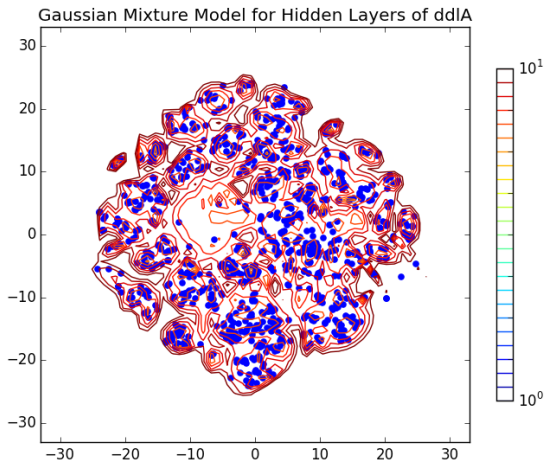
#### *4.1.1 Hidden Layer Matrix*

Hidden layer activations change each time a new nucleotide is passed through the model, so we collected hidden layer activation vectors ($\mathbb{R}^h$ where h is the size of hidden layer, 128 for medium capacity and 512 or 1024 for high capacity networks) for all nucleotides, resulting in a matrix of all hidden layers ($\mathbb{R}^{g \times h}$ where g is the number of bases in the specific genome). Thus there is a unique matrix for each gene or sequence.

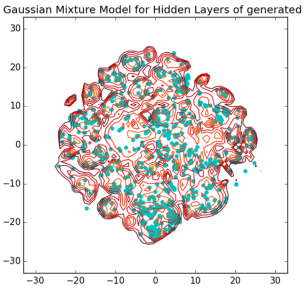
#### *4.1.2 Sequence Embedding*

We next created an embedding of gene properties using the extracted hidden layer matrix described above. First, we used t-Distributed Stochastic Neighbor Embedding (tSNE) [6] to reduce the dimensions of hidden layer activations of subsequences corresponding to labeled genes in E. coli from $g$ to 2 resulting in a $\mathbb{R}^{g \times 2}$ matrix for each gene. Thus, now each gene can be represented as a series of points (of length g) on a 2 dimensional surface (Figure 5 a). Taking all of the points for all genes, we then clustered these points using a Mixture of Gaussians model (100 gaussians with independent covariance matrices). Unique points for the gene ddlA and the Mixture of Gaussians likelihood contours from clustering are shown in Figure 5. Each gene forms a unique representation occupying certain clusters.

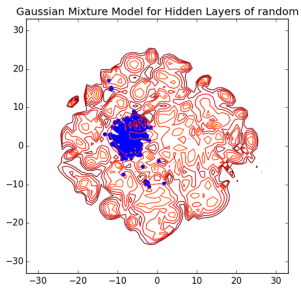
We hypothesized that each of the Gaussians in the mixture of Gaussians model captured specific properties of genes. These properties were quantified by individually projecting each of the reduced hidden layer representations of genes ($\mathbb{R}^{g \times 2}$) onto the 100 mixture of Gaussians model




(a) ddlA from E. coli




(b) Sequence generated by the model




(c) random sequence of length 1372

Fig. 5. Reduced dimension hidden layer representation for different genes overlaid with the Mixture of Gaussians models representing all the clusters formed with the genes. Each point represents a hidden layer in time for the forward pass of the specific sequence

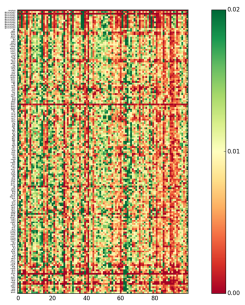
and then measuring the effective proportion of each gene's hidden layers that fell into each Gaussian (equation 4).

$$\text{for each cluster j} : \frac{1}{g}\sum_{i=1}^{g} P_{ij} \tag{4}$$

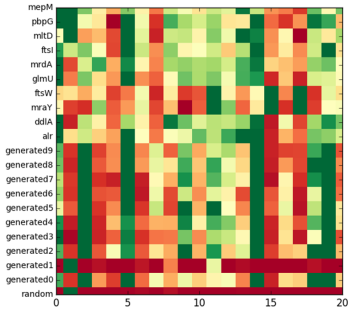
This resulted in a 100-dimensional embedded representation of genes (Figure 6). The generation of this embedding is similar to forming a single vector using an encoder-decoder sequence-to-sequence model [7]. However, while in these sequence-to-sequence models long-term information is largely lost, using the novel method presented here maximizes retention of long-term information in the hidden layers prior to dimensionality reduction. This theoretically leads to better performance at the cost of being more computationally expensive.

#### 4.1.3 *Sequence Clustering*

We compared the properties of genes using heatmaps of 100-dimensional vector gene embeddings (Figure 6). These heatmaps allow for easy understanding of how model-generated sequences relate to different genes and to random sequences.



(a) Embedding represenation for all genes




(b) Zoomed version for clarity focusing on the first 20 properties of generated genes versus real E. coli genes

Fig. 6. Embedding representation of each gene in E.coli, along with the generated sequences and a random sequence. Each row in the figure is the embedding for a specific gene, and each column represents a property of a gene found in an unsupervised fashion. Red signifies the lack of the property and green signifies strong presence of the property

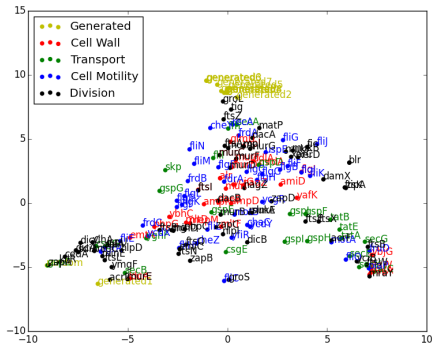
Embedded representations of genes can also further be visualized using tSNE to interpret the closeness of gene properties (Figure 7). Here, genes that correspond to different organism functions have different colors. The significant overlap of clusters of genes with different functions could be due to overlapping structure or function between the different classes of genes. Since more separation of genes with different function was achieved with the RNN model than with the baseline 5-gram model, we conclude that the RNN embedding used here likely captures important properties of genes beyond nucleotide frequency.

It is important to note that the clusters do detect that the generated sequences are perhaps not functional, because they are outside of the clusters formed by real genes. This may mean that the model is producing non-coding sequences, and we would have to improve the model to generate higher quality sequences. Given that we have optimized the model, there may be a need to try a different model that captures long term dependencies better than an LSTM.

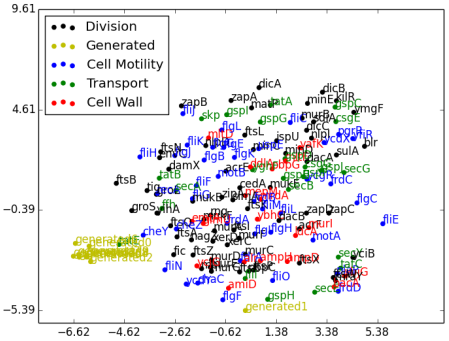
## 5 Discussion

We described a pipeline to create genome and gene property embeddings, generate novel sequences, and visualize differences between genes and generated sequences. While the sequences we generated do not fall within the space occupied by real genes in our tSNE visualization of embedded gene representations, the pipeline is highly modular, and better models can be engineered to replace the LSTM used here and generate better sequences. Importantly, the ability to describe the properties of a sequence and evaluate its legitimacy is in itself substantial and provides a way to analyze newly sequenced and unlabeled genes and genomes as well as artificial sequences generated by any means. Additionally, the methods used here capture a sufficiently good representation of genomes to predict the taxonomy of unlabeled genomes, indicating the potential to design sequences that have properties similar to genomes in desired taxonomic groups.

Having established this baseline framework, now the community is in a position to iterate quickly between forming




(a) Similarity of genes based on the formed gene embeddings




(b) Similarity of genes based on 5-gram representation of genes

Fig. 7. Comparison of reduced dimension representation of gene embeddings and 5-gram representation of genes

better models and evaluating the generated sequence quality. Having generated a way to predict the gene properties in an unbiased fashion also provides the opportunity to form testable hypotheses for experimental work. We also note that the RNN visualization techniques developed in this work can be potentially applied to other fields working with RNNs.

## 6 Conclusion and Future Directions

The generative modeling of unannotated genome sequences presents several unique challenges not typically encountered in traditional language or vision problems and warrants discussion.

Firstly, constructing an explicit objective function and evaluating success is difficult because of the lack of easily measurable metrics for what a "good" or even functioning sequence looks like. Surrogates such as secondary structure, sequence homology, and phylogenetic classification can be used to create a traversable fitness landscape, but large-scale functional assays of a variety of sequences and automated in vitro validation studies are likely necessary to enable better scoring.

Secondly, genomes are not robust to particular types of small changes, which can make categorization difficult. Indeed, a single shift in reading frame or displacement of a small promoter motif can lead to a nonviable organism. To ever enable the generation of novel genes or genomes, further work must be done specifically towards creating custom models suited for genomics; for example, better genomics-oriented models could be more rapidly developed by focusing on low to high level machine code translation problems that are similar to the genome to function problem (e.g., machine code is also not robust to particularly small changes), but are substantially easier to test using a decompiler. In this faux-genome setting, models other than LSTMs such as Generative Adversarial Networks with attention or Neural Turing Machines, which may be better suited for genome dreaming than LSTMs, could be rapidly prototyped.

Thirdly, the range and variety of interactions across the genome is mostly unknown, which can make choosing and tuning models difficult. Measuring changes in hidden layer activations across a sequence could be a good way to discover new relationships within genomes. Tools have recently been published that do just this [8]. A clear next step to the work presented here is to demonstrate that the hidden layers of neural networks trained with unlabeled genomes can capture human-defined DNA annotations, such as promoters, ribosome binding sites, terminators, or even operons, and then explore what unknown motifs the model has learned.

Finally, because current phylogenetic methods take into account many variables other than whole-genome sequence similarity into their classification scheme (e.g., phenotype or 16s rRNA sequence similarity), current taxonomic groupings of prokaryotic organisms are likely not a sufficient standard by which to evaluate whole-genome predictive tasks. For example, when using whole genomes as individual samples, choosing a proper training and test set is difficult because it's unclear how related genome sequences are to each other. One potential way to address this is to reclassify organisms based only on whole-genome similarity measures.

In sum, these methods can be used for discovering new genome features, predicting the taxonomy and annotations of newly sequenced genomes, and designing genetic sequences with desired properties.

## References

[1] Drew Endy Barry Canton, Anna Labno. Refinement and standardization of synthetic biological parts and devices.

[2] Xiaohui Xie Daniel Quang. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences.

[3] Matthew T Weirauch Brendan J Frey Babak Alipanahi, Andrew Delong. Predicting the sequence specificities of dna- and rna-binding proteins by deep learning.

[4] Guy Yachdav, Edda Kloppmann, Laszlo Kajan, Maximilian Hecht, Tatyana Goldberg, Tobias Hamp, Peter Hönigschmid, Andrea Schafferhans, Manfred Roos, Michael Bernhofer, et al. Predictprotein—an open resource for online prediction of protein structural and functional features. *Nucleic acids research*, page gku366, 2014.

[5] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, June 2009. Also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada.

[6] Geoffrey Hinton Laurens van der Maaten. Visualizing data using t-sne.

[7] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[8] Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M. Rush. Visual analysis of hidden state dynamics in recurrent neural networks. *CoRR*, abs/1606.07461, 2016.