Discovery of Transcription Factor Binding Sites with Deep Convolutional Neural Networks

Reesab Pathak
Dept. of Computer Science
Stanford University
rpathak@stanford.edu

Abstract

Transcription factors are key gene regulators, responsible for modulating the conversion of genetic information from DNA to RNA. Though these factors can be discovered experimentally, computational biologists have become increasingly interested in learning transcription factor binding sites from sequence data computationally. Though traditional machine learning architectures, including support vector machines and regression trees have shown moderately successful results in both simulated and experimental data sets, these models suffer from relatively low classification accuracy, typically measured by area under the receiver operating characteristic curve (auROC). Here we show that learning transcription factor binding sites from sequence data is feasible and can be done with high accuracy. We provide a sample CNN architecture that yields greater than 96% auROC. Additionally, we discuss key questions that need to be answered to improve hyperparamter search in this domain.

1 Introduction

Since the Human Genome Project, which concluded in the early 2000s, computational biologists have become interested in learning features of the genome from sequencing data. Revolutionized by the rapid advancement of second and third generation sequencing technologies, learning biologically relevant features from sequencing data has become possible. In the past 5 years, scientists have used traditional machine learning techniques and probabilistic graphical models to impute haplotypes, learn epigenetic markers, and much more. Though we cannot provide a full review here, this is done well in [5].

1.1 Problem and Related Work

This paper specifically considers the *transcription factor binding site discovery* problem. Though we cannot provide a review of transcription factors here, we refer the reader to Slattery et al, where the transcription factor binding site literature is well-reviewed [8]. Formally, our problem is a multi-task classification problem. We are given, as input, a training set with pairs $\{X^{(i)}, y^{(i)}\}_{i=1}^{n}$. In our problem, the input data, $X^{(i)}$ is a matrix, of dimension $4 \times N$, where N is the length of a DNA sequence. This matrix, which is referred to as the positional frequency matrix (PWM) has four rows corresponding to each channel of genetic alphabet, namely $\{A, C, T, G\}$. Our labels, $y^{(i)}$ are either scalar or vector, depending on the number of transcription factor binding sites that are being learned. Nonetheless, the dimension is equal to the number of classification tasks, and each element of $y^{(i)}$ is a binary label in the standard space, $\{0, 1\}$. The goal is then to accurately predict the labels from the training data, which is to accurately predict whether or not each of the transcription factors binds in a given sequence.

Recently, many groups have studied similar genome prediction problems, especially in the context of epigenetic features. Leung et al review the progress here in [5]. For this specific problem, support vector machines with a k-mer kernel has shown greater than 70 percent auROC [4]. Additionally, recent groups have discussed the applicability of neural network architectures to this problem [6][10].

1.2 Our contributions

Here, we provide a sample convolutional neural network (CNN, ConvNet) architecture which provides greater than 96 percent accuracy on a simulated data set. Additionally, we use a recently published interpretation package to show that the features learned by the CNN are roughly the transcription factor binding sites themselves, suggesting that CNNs are good models for this problem, despite the computational expense. Finally, we provide a discussion of areas of key difficulty that need exploration to improve the accuracy of CNNs for larger scale problems and for experimental datasets.

2 Methodology

Our work uses a simulated training set of sequencing data and labels. We note, however, that this is a standard practice within this domain, and many papers that evaluate model accuracy have taken similar approaches [1]. This simulated training set is then provided as input into our convolutional neural network, which requires hyperparamater tuning to get high accuracy.

2.1 Simulated data

We sampled sequence of length 1000 base-pairs (1000 symbols) from the genetic alphabet, A, C, T, G, with a fixed GC fraction, of 0.4, which is a biologically motivated result. This means that the probability assigned to each letter was 0.3, 0.2, 0.3, 0.2, respectively. With these probabilities we sampled a fixed number of reference sequences, depending on the problem. We then uniformly at random, sampled an index from which to embed a transcription factor motif. These motifs, which come from the Encylopedia for DNA Elements (ENCODE) project, are non-deterministic. Again, we are unable to review the results of the ENCODE project here, but this is done well in [2].

The motifs are accompanied by a position weight matrix (PWM), which is a $N \times 4$ representation of the odds of each symbol of the genetic alphabet at each position over the background frequency of a given letter. We binomially sampled motifs from the position weight matrix, and replaced our sequence data with the sequence of the motifs. For our studies, some experiments tested the number of training examples, in which case, we typically kept balanced classes of negative and training examples. Negative samples had randomly embedded sequences, which provides a more biologically relevant negative class of data. Then, from these reference sequences, we sampled sequence reads, which are 100 base pair substrings. Since sequencing data contains irreducible noise, we added noise by flipping some symbols, with probability 0.001 at each index, based on the approximate error rate of a standard high-throughput sequencing technology. Our labels were constructed by creating vectors with dimension equal to the number of transcription factors that were embedded. We placed a 1 in the ith index of our vector if transcription factor i was embedded in the sequence, otherwise we placed 0. Array data is achieved through the one-hot encoding of our reads. Our sampling methodology, though not identical to any previous study, was informed by previous machine learning studies in genomics [1].

2.2 Convolutional Neural Network

Convolutional neural networks have three features: (1) convolutional layers, (2) pooling or subsampling layers, and (3) fully connected layers. The convolutional layer conducts an affine transformation over its input. Multiple activation maps are created by multiple convolutional lters. These pass to an activation function, which is a non-linearity such as a rectified linear unit (ReLU). The output is then passed to the pooling layer, which subsamples the convolutional output and takes the maximum entries within the domain of a subsampling unit (called Max Pooling). This process (convolution to max pool) is repeated. Unlike previous layers, the final layer is fully connected and a matrix-multiply is done to get output predictions. Our architecture is CL-MP-CL-MP-CL-MP-FC, where CL is a convolution layer, MP is a max pool layer, and FC is a fully connected layer. Between a convolution and max pool layer, data always passes through the ReLU. Additionally, convolutional neural networks like other feedforward archictures use backpropogation to update weights during each epoch during training. We cannot fully review CNNs here, so again we refer the reader to [3].

We implemented our convolutional neural network using Keras, a deep learning library that wraps around the deep learning software in Theano. We discuss hyperparamaters used in **Results** and **Discussion**. The input data was the training set as described previously. The testing and validation data were similarly simulated sets of data.

3 Results and Discussion

We conducted two classes of experiments: single-motif classification and multi-motif classification. The first task is formulated as a single-task, binary classification problem, with scalar labels. The second task is formulated as a multi-task, binary classification problem, with vector labels.

3.1 Single motif embedding task

In the single motif embedding task, we first embedded motif TAL1, which encodes the transcription factor T-cell acute lymphocytic leukemia protein 1. This transcription factor has the sequence (see right), based on the second known motif from the ENCODE project.

This figure is related to the position weight matrix (PWM) of each motif, which is of dimension $4 \times L$ (L is the length of the motif). The ith column of the matrix represents the probability of each of the four states (A, C, T, G) at base-pair i in the motif. The sequence logo above is generated by calculating for each base,

Figure 1: $TAL1$ (ENCODE) transcription factor motif, depicted as sequence logo

the information content, which is $\log_2(4) - (H_i + e_n)$, where H_i is the (Shannon) entropy at each base pair, and e_n is the small sample collection factor, which is reviewed in [7].

3.1.1 Data needs

We investigated how many training examples in the form of position-specific scoring matrices (PSSMs) were necessary to get near optimal results. We measured an upper bound on testing accuracy by providing to a Random Forest the exact locations in the PSSM where the embedded motifs were. Thus, this represents an upper bound on performance, since we do not limit CNNs from learning these decision boundaries.

From Figure 2, around 4000 training examples is optimal to reach near testing accuracy achieved by a Random Forest model (auROC = 0.9502, n = 4000).

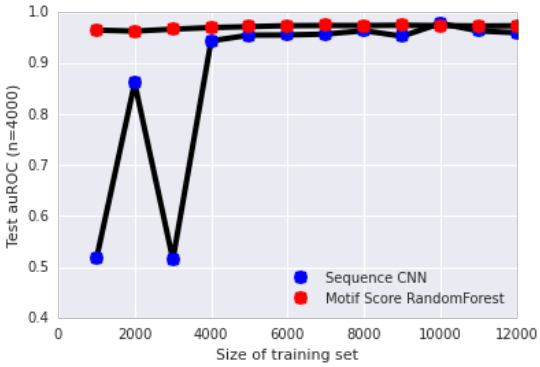
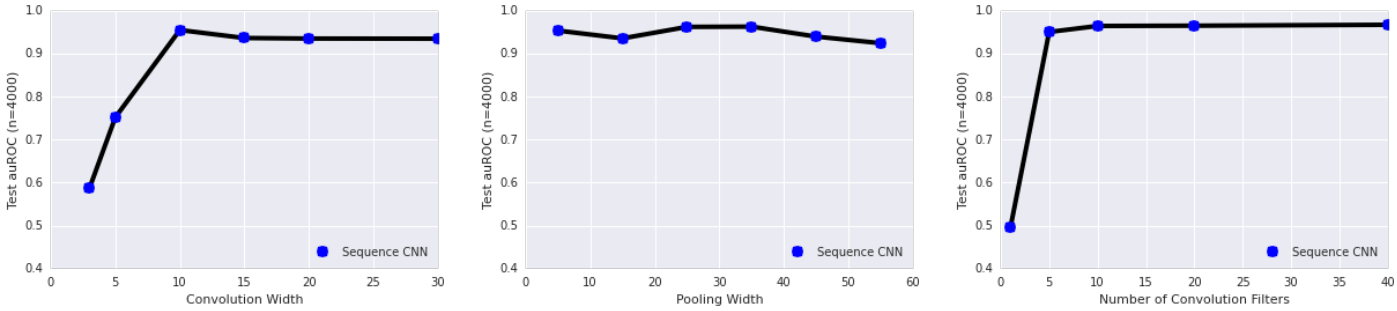


Figure 2: Single motif embedding, data needs

3.1.2 Hyperparameter tuning

The hyperparameters for the single motif task needed to be tuned to get optimal results. Basedo on Figure 3, with a convolution filter width of 10, pooling filter width of 25, and 10 convolution filters per convolutional layer, we achieved auROC of 0.975, with $n = 4000$ training examples.



(a) Convolution filter width

(b) Pooling filter width

(c) Number of filters per layer

Figure 3: Example hyperparameter tuning plots for single motif embedding

3.1.3 Dropout

To prevent overfitting, we investigated whether dropout was a feasible strategy to improve training accuracy. To assess this, we used a motif with greater heterogeneity than TAL1. The motif we chose was CTCF, discovered motif 5, which displayed more entropy in its position weight matrix. Based on the figure 4, as dropout increases, the test auROC substantially increases, suggesting that adding dropout to the model may improve its robustness.

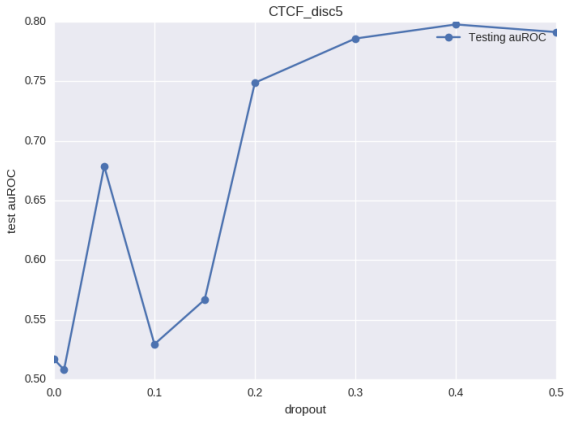
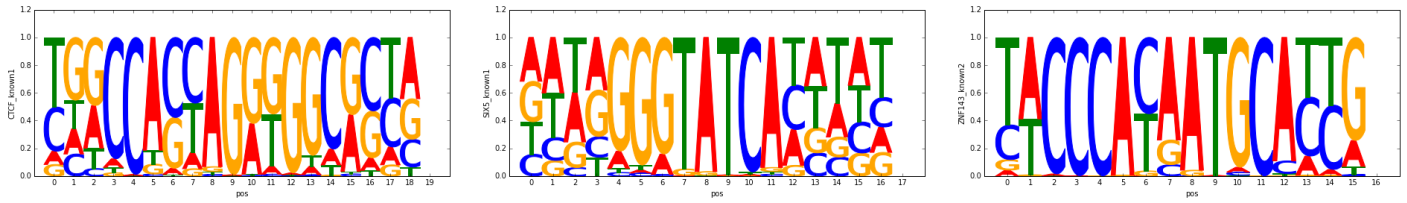


Figure 4: Dropout probability versus test auROC

3.2 Multi motif embedding task

For the multi motif embedding task, we embedded three motifs known to co-bind [9]. CTCF known motif 1 is responsible for transcripition regulation, V(D)J recombination, and chromatin architecture regulation. ZNF143, known motif 2, is a transcription activator, and SIX5, known motif 1, recruits many DNA-binding proteins. The sequence logos are shown in Figure 5.

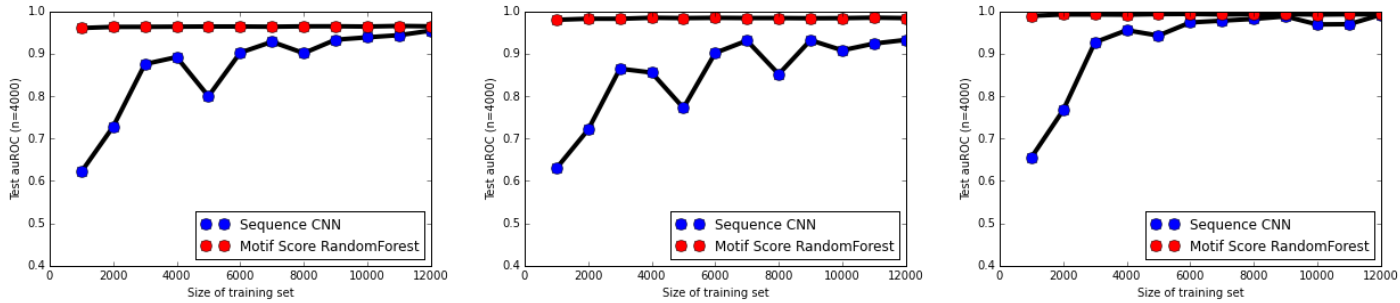


(a) CTCF, known motif 1 (b) SIX5, known motif 1 (c) ZNF143, known motif 2

Figure 5: Multi motif embedding, ENCODE motif sequence logos

3.2.1 Data needs

We again investigated the number of training examples to reach the accuracy of the Random Forest model, which is given the optimal decision boundaries by location of motifs in the sampled sequence. Based on figure 6, we needed around 12000 training examples



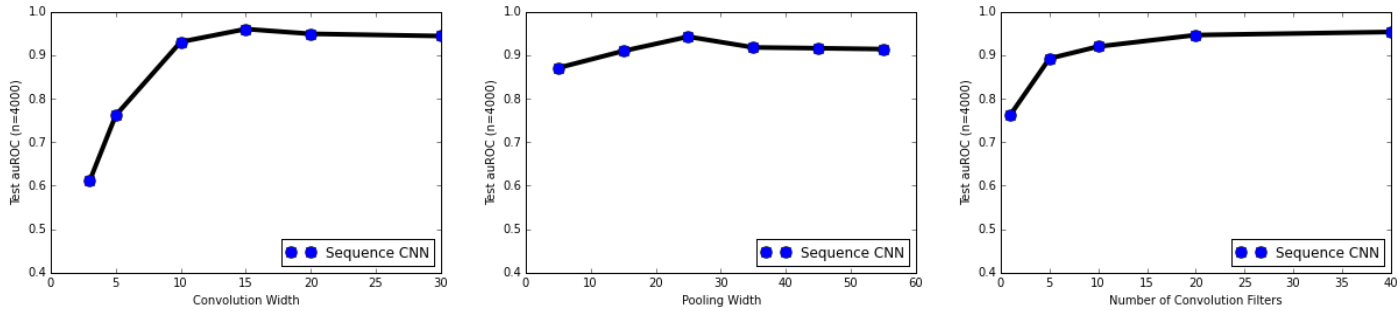
(a) Data size, overall task (b) Data size, motif 1 (c) Data size, motif 2

Figure 6: Data size needs by task, multiple motif embedding

to get near the Random Forest accuracy. Additionally, for single motifs, within the task, the same size of data also nearly achieved the Random Forest model's accuracy.

3.2.2 Hyperparameter tuning

The hyperparameters for the multiple motif task needed to be tuned to get optimal results. Based on figure 7, using a convolution filter



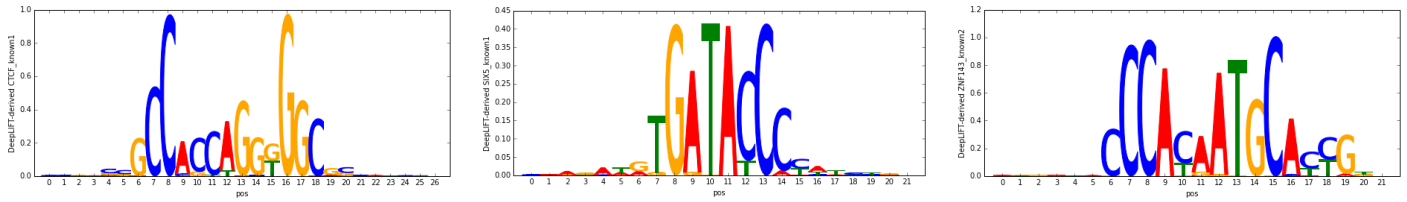
(a) Convolution filter width (b) Pooling filter width (c) Number of filters per layer

Figure 7: Hyperparameter tuning plots, combined for all 3 tasks

width of 15, a pooling filter width of 25, and 45 convolutional filters per layer, we achieved auROC = 0.967 with $n = 12000$ training examples.

3.2.3 Feature importance and model interpretation

Recently, a group at Stanford published a method called DeepLIFT (Deep Linear Importance Feature Tracker), which is able to identify feature importance in the input to a CNN (CITE!). We ran the software for our multi-task classification to see what features the CNN



(a) CTCF, (b) SIX5 (c) ZNF

Figure 8: DeepLIFT interpretation sequence logos, multi-task classification problem

found most important to predicting the labels. These figures are encouraging because, if compared against Figure 5, it is clear that the features learned by the model correspond with parts of the ENCODE motifs. Thus, not only is the model learning the correct labels, but it does so by identifying structure of motifs within the sequence data.

4 Further Questions and Outlook

Despite good accuracy on the learning tasks described above, a number of questions need to be answered before this model can be extended to experimental data. This work shows that it is indeed possible to learn transcription factor binding sites based on sequencing data, after sufficient data and hyperparameter tuning. Nonetheless, it is still unclear how the optimal width of the convolution filter scales with the entropy and structure of the position weight matrix. Additionally, we were not able to investigate how density localization impacts hyperparameters and how penalty coefficients for regularization and dropout probabilities may need alteration as more motifs are embedded. Though our methodology is similar to that adopted by previous computational biology papers, we acknowledge that testing this model on experimental data is important.

5 Conclusion

Here, we have demonstrated the feasibility of learning transcription factor binding sites from sequencing data. We note that there are an number of key questions (see **Further Questions and Outlook**) that should be explore prior to using these models on experimental data. Nonetheless, deep learning through CNNs provides high accuracy for learning genomic transcription factor binding loci.

Acknowledgments

We thank Stanford Research Computing for allowing access to the Sherlock computing cluster for access to GPU compute nodes on which to run Theano and Keras code. Additionally, we thank Anshul Kundaje, Johnny Israeli, and Avanti Shrikumar (Stanford, Computer Science) for providing access to their recently published DeepLIFT code, which provided interpretation figures for our CNNs.

References

[1] Babak Alipanahi et al. “Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning.” In: *Nat Biotechnol* 33.8 (2015), pp. 831–838.

[2] Roadmap Epigenomics Consortium et al. “Integrative analysis of 111 reference human epigenomes”. In: *Nature* 518.7539 (2015), pp. 317–330.

[3] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521 (2015), pp. 436–444.

[4] Dongwon Lee et al. “A method to predict the impact of regulatory variants from DNA sequence”. In: *Nat. Genet.* 47.8 (2015), pp. 955–61.

[5] Michael K. K. Leung et al. “Machine Learning in Genomic Medicine: A Review of Computational Problems and Data Sets”. In: *Proc. IEEE* 104.1 (2016), pp. 176–197.

[6] Daniel Quang and Xiaohui Xie. “DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences”. In: *bioRxiv* (2015), p. 032821.

[7] Thomas D Schneider and R Michael Stephens. “Sequence logos: a new way to display consensus sequences”. In: 18.20 (1990), pp. 6097–6100.

[8] Matthew Slattery et al. “Absence of a simple code: How transcription factors read the genome”. In: *Trends Biochem. Sci.* 39.9 (2014), pp. 381–399. arXiv: NIHMS150003.

[9] Jie Wang et al. “Sequence features and chromatin structure around the genomic regions bound by 119 human transcription factors”. In: *Genome Res.* 22.9 (2012), pp. 1798–1812.

[10] Jian Zhou and Olga G Troyanskaya. “Predicting effects of noncoding variants with deep learning-based sequence model.” In: *Nat. Methods* 12.10 (2015), pp. 931–4.