Evaluation of Multiple Deep-Learning Architectures for Transcription Factor Binding Prediction

Amish Suchak
Department of Electrical and Computer Engineering
University of Florida
Gainesville, Florida
Amish.suchak@ufl.edu

***Abstract*— The purpose of this experiment is to evaluate and compare different Deep-Learning architectures for transcription factor binding prediction. Different models are compared that incorporate fully-connected models, CNN and RNN based models and 2D-CNN models. These models are then compared by first fine-tuning the hyper-parameters to obtain the best possible prediction accuracy. The results of the paper that obtained the best prediction were then posted onto the Kaggle competition page to yield a prediction accuracy of 84%.**

Keywords—Deep Learning, RNN, CNN, LSTM

I. INTRODUCTION

Since the beginning of 2000, after the completion of the human genome project, the subject of human genome sequences have been studied in great detail to understand its biological influences on living organisms. One of the key challenges that scientists have faced is in revealing the dynamic regulatory systems by the transcription factors (TFs). TFs regulate the gene expression by interacting with DNA sequences called the transcription factor binding sites (TFBSs). In the process of transcription, the gene's DNA sequence is transcribed into an RNA molecule. This information is necessary for the gene to make a protein.

Chromatin immunoprecipitation (ChIP) is the most widely used method to recognize protein-DNA binding locations. In the ChIP protocol, TFs are crosslinked to their target DNA and then immunopurified from chromatin. The bound DNA is then purified and sequenced genome-wide by deep sequencing (ChIPseq). ChIP method only yields a set of statistically enriched high-occupancy binding regions. A substantial segment of this DNA may represent false positives and many other low-affinity sites be missed due to low sensitivity. Additionally, incongruent size of randomly clipped DNA fragments largely limits the resolution of ChIP-seq and thus the method fails to distinguish among neighboring binding events.

With the advent of improved machine learning methods over the past decade clubbed with computers being able to perform computations at a significantly faster rate using GPUs, computers have been able to predict biological sequences with a much higher accuracy [1][2]. In this research paper, one such method of obtaining a high prediction in Transcription Factor Binding is discussed.

Machine learning is a subfield of artificial intelligence, and aims to make predictions guided by data. The data set given to a machine learning algorithm can either be structured or unstructured. Structured data typically contains predefined labels for the features of the dataset (which is the case for the soccer game predictions). In the field of machine learning, neural networks are a subset of algorithms responsible for imitating a similar behavior to that of biological human-brain neurons. They are interconnected or partly connected and usually improve (learn) progressively depending on the algorithm. Within the core of each neuron lies an activation function used to connect all the inputs to yield one output. There can be multiple neurons in the layer between the input and the output layer. Deep learning, hence, is the inclusion of multiple hidden layers that are found between both the input and the output layers [3].

Currently, neural network architectures are selected depending on the type of application. For example, in image object classification and processing, Convolutional Neural Networks (CNNs) are generally applied due to their inherent structure [4]. Similarly, Reinforcement Learning-based architectures are typically used for video game and autonomous driving purposes [5]. Lastly, for applications with sequential-based data which depend on past and future inputs, Recurrent Neural Networks (RNNs) are most commonly applied [6]. Hence, in this paper, a combination of ANN, RNN and CNN architectures will be discussed and implemented.

The rest of the paper is organized as follows: The related work section - where existing research on Transcription Factor Binding predictions is compared - is presented in Section II. The system architecture and overall model of the system, modules and subsystems that will be implemented in this paper is presented in Section III. Lastly, the discussion and conclusion is presented in Section IV.

II. RELATED WORK

Currently, there are many existing deep-learning approaches that have been discussed in the literature that deal with the genome sequence classification problem. For the purpose of this paper, these architectures were considered based on their performance and their prediction accuracy.

In [6], the authors proposed a fully convolutional neural network based architecture. They implemented ReLU activation functions along with Max Pooling layers and a final fully connected layer. The main architecture was described as CL-MP-ML-MP.CL-MP-FC where CL was the convolutional layer, MP was the max pooling layer and fc was the fully connected layer. They converted their problem into that of a single motif problem with two labels for their output. From this paper, the most important aspects were the hyperparameter tuning that they implemented in which they received the best performance with a filter width of 10, pooling filter width of 25 and 10 convolution filters per convolutional layer. Hence, the important aspect that was obtained from this paper was the implementation of a convolutional based neural network for a 1D process along with the necessary hyperparameter selection.

In [7], the authors proposed DeepSNR, a deep learning algorithm implementing a DeconvNet to train on top of an existing DeepBind architecture. The authors were able to achieve an precision of 87% overall based on their architecture. The authors implemented the exact same convolutional architecture as DeepBind, but incorporated the deconv network with a series of unpooling, deconvolution and Z rectification layers. The authors implemented a sigmoidal cross entropy loss function. However, there was no mention of the type of hyperparameters that were implemented or any additional information on the hyperparameter tuning.

In [8], the authors proposed another convolutional based architecture called DeepEnhancer. The important aspects of this paper was understanding how the authors were able to obtain the data in a 4 x 1 x L size matrix where L was the length of the input sequence (14 in our case). The four necleotides, A,C,G and T were encoded using one hot encoding to form 4 channels and therefore the length L was thought of as an image of 4 channels with height 1 and width L. The authors showed how they were able to achieve a prediction accuracy close to 91% in their predictions.

In [9], the authors presented a network based on a combination of convolutional, and LSTM layers called DanQ. The first convolutional layers were designed to scan sequences for motif sites through convolution filtering. As compared to the DeepSEA model where they implemented three convolution layers and two max pooling layers, the convolution layer of the DanQ algorithm only contained the one convolution layer with one max pooling layer. The layers were then followed by a bidirectional LSTM structure.

In [10], the authors presented a CNN that outperformed conventional methods in sequence specificity. They implemented the adaptation of convolutional neural networks by considering a window of genome sequence as an image. Instead of processing 2-D images with three color channels, (R,G,B) they considered a genome sequence as a fixed length 1-D sequence of window with four channels (A,C,G,T). Based on their architecture, they implemented convolutional layers with a window size of 24 at first followed by global max pooling layers, a fully connected layer followed by an output layer. The authors achieved a significantly high prediction accuracy, and hence their algorithm was also tested and implemented in our experiment.

III. Prediction model architecture and discussion

For the purpose of this experiment, I tried to include a combination of several architectures that were mentioned in the literature. The point was to obtain a model that would yield the best prediction accuracy. Hence, the overall goal was to incorporate simple fully connected ANN models, CNN and also LSTM based models.

The experiment was conducted on Google Colab, using the Keras library.

A. Using an ANN architecture

The first part of the code was to obtain the information about the training data and the testing data. From the training data columns, I obtained the 'sequence' column which was a 14 character sequence consisting of alphabets (A,C,G,T). Hence, for the ANN structure, I could simply convert these sequences into numbers using a python dictionary specifying each character as a particular number. Once these characters were defined, I could convert the sequence into a 2000x14 array of input numbers. For the ANN architecture, I realized that there was no need to convert the input sequence into a one-hot encoded vector. Once the data was selected, I used the test_train_split function from the sklearn library to divide my data with 1800 training and 200 validation.

After doing the initial data processing, the next step was to try out different architectures. I began by implementing a 1 layer ANN fully connected architecture using ReLU activation function and sigmoid functions as the output layer. The loss function was a binary cross entropy implementing an adam optimizer. The hyperparameters were continually changed along with the number of hidden layers. The best performance that I had received was using a batch size of 32 and training for 120 epochs. Longer epochs meant that the data began overtraining. However, using only the ANN architecture, the best performance I could obtain was with a 73% accuracy on 3 hidden layers.

B. Using CNN and LSTM architecture

For the CNN architecture, the model could be implemented similar to the ANN. However, instead of the input being a 2000x14 matrix, I had to convert the input sequence into a one-hot encoded matrix. I could do that with the help of the "to_categorical" function provided by keras.utils. Everything else in the process remained the same. The convolutional layer received a 2000x14x4 sized model as shown in the literature. I continuously changed the 1D convolutional layers incorporating a combination of Max Pooling, Dropout and Flatten layers in between the architecture. I also added a fully connected layer towards the end of the architecture followed by the output layer. I realized that for the CNN type architecture, I could compile the model much sooner. After 8 epochs, My loss was at 0.3511 which was a lot better than the performance of the ANN type architecture.

To improve the work done on the CNN architecture, I tried to do what the authors in [9] did by incorporating RNNs in the process as well. At first I tried adding an LSTM layer after the MaxPooling layer, but only to receive the same prediction accuracy as the CNN architecture. A GRU layer surprisingly was able to get me a better prediction than the LSTM layer. I also incorporated a bi-directional LSTM similar to [9] to see if I was able to get a better performance, but the best performance that I got was using the GRU with an accuracy of 80%. It was not the best in the leaderboard but was significantly more accurate than the ANN architecture.

C. Using a 2-D Convolutional network

The last thing I tried to do was to implement the architecture that the authors had done so in [10]. Since their architecture outperformed all other convolution based methods, I was curious to see if I would obtain the same result as them. Similar to their architecture, Instead of processing 2-D images with three color channels, (R,G,B) I considered a genome sequence as a fixed length 1-D sequence of window with four channels (A,C,G,T). The model began with a 2D convolutional network followed by a global max pooling layer and then a fully connected layer and the output layer. I kept changing the optimizer between ‘adam’, ‘rmsprop’, and ‘SGD’ to evaluate the performance of all of them. I noticed that the adam optimizer gave me the best results in terms of prediction accuracy. Similarly, the batch size and epochs were also hyparameters that were fine-tuned after every iteration to obtain the best possible accuracy.

After running the algorithm multiple times, I came to realize that since there are only 2000 values for the data, it meant that there were inconsistencies. Sometimes, I would receive a prediction rate of 92% on the validation data and other times I would receive a prediction rate of 85% on the same prediction data. Had there been a larger dataset, then maybe I would have had more room to experiment and figure out the best possible hyperparameters. Overall, based on my experimentation, I found the best accuracy of 91.5% on the validation using an adam optimizer and 23 epochs with a batch size of 64.

IV. Conclusion

In conclusion, the experiment was conducted to obtain the transcription factor binding predictions. Several architectures were tested (The results of these architectures are provided in the excel sheet as part of the assignment submission). Overall, the best performance was received using the 2-D convolutional network. When tested on the official Kaggle test-set, I received a prediction accuracy of ~ 84%.

Kaggle Username : Amish Suchak
Github: https://github.com/Amish1234/Transcription-Factor-Binding-Prediction

V. Bibliography

[1] S. D, I. Buck and P. Simard, "Using GPUs for machine learning algorithms," in *Eight International Conference on Document Analysis and Recognition*, Seoul, 2005.

[2] X. Du and Y. Cai, "Overview of Deep Learning," in *Youth Academic Annual Conference of Chinese Association of Automation,* Wuhan, 2016.

[3] T. Guo and J. Dong, "Simple convolutional neural network on image classification," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, Beijing, 2017.

[4] M. Emigh and E. Kriminger, "Reinforcement Learning in Video Games Using Nearest Neighbor Interpolation and Metric Learning," *IEEE Transactions on Computational Intelligence and AI in Games* , vol. 8, no. 1, pp. 56-66, 2014.

[5] A. Graves, G. Hinton and A.-R. Mohamed, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Vancouver , 2013.

[6] R. Pathak, "Discovery of Transcription Factor Binding Sites with Deep Convolutional Neural Networks".

[7] S. Salekin, "A deep learning model for predicting transcription factor binding location at single nucleotide resolution," in *Conference: IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), 2017,* 2017.

[8] X. Min and W. Zeng, "Predicting enhancers with deep convolutional neural networks," in *BMC Bioinformatics* , 2017.

[9] D. Quang and X. Xie, "DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences," *Nucleic Acids Res,* vol. 44, no. 11, pp. 107-116, 2016.

[10] H. Zeng, M. Edwards and G. Liu, "Convolutional neural network architectures for predicting DNA–protein binding," *Bioinformatics,* vol. 32, no. 12, pp. 121-127, 2017.