Deep Learning in Computational Biology:

Predicting PBM binding from HT-SELEX data

Oded Yechiel
Yehu Sapir

Code available at:

https://github.com/odedyec/biological_dnn.git

06/08/2018

Introduction

High-throughput SELEX (HT-SELEX) technology, is a technique that attempts to filter oligos with strong protein bound, from non-bounding oligos. The SELEX process is performed by taking a pool of random oligos, and binding them with proteins. After filtering out the bounded oligos, which are sequenced and amplified, they are used as the new pool of the next cycle. This process is performed 4-6 times, and in each cycle oligos with stronger protein binds remain. The assumption is that the first cycle, with random oligos, are unbound, and the last cycle contain strong protein binding oligos. The oligoses' length are either 14, 20, 30 or 40 nucleotide long.

Protein Binding Microarrays (PBM) sequences are a different method for detecting TF binding to specific K-mers. The sequences in the PBM are 60 nucleotides long, and only the first 36 are of interest.

In this project, given 5-7 files of HT-SELEX and a PBM file with ~42K sequences, we attempt to rank the PBM sequences based on the SELEX information. Only the top ranked 100 sequences in the PBM are of interest, and should be retrieved from the ~42K sequences in the PBM file. The ranking is performed by using Deep Neural Network (DNN) that learns the binding strength of SELEX oligos. The dataset is comprised from 246 Trancription Factors (TF) tests: 123 tests were with already sorted PBM files, and 123 tests were with unsorted files.

This report contains a summary of our best performing DNNs, out of the numerous architectures with varying parameters that were explored. The DNN was developed in Python using Keras with Tensorflow backend. The models were trained on two computer systems:

1. Colfax's remote access program with Intel® Xeon Phi™ processors 7250, 64 cores, 96 GB RAM (DDR4) and 16 GB of high-bandwidth memory (MCDRAM).
2. A PC computer, with Intel Core i7, 4770, and NVidia Titan GPU, with 12GB memory and ~3000 Cuda cores.

Preprocessing

In order to reduce the total time of the program, not all of the data from each SELEX is loaded. Training the network with more data does not seem to improve the overall performance of the network. In fact, training on a small percentage of the data improves the classification of the SELEX. Since we do not know how the sequences were obtained, and if there is any difference between location of sequences in the SELEX file, we randomly pick T sequences from the file. In addition, in order to keep the trained dataset balanced, we take the same amount of data from each SELEX file.

All of the loaded sequences are OneHot encoded into a 4 category vector – each nucleotide is one category. The SELEX sequence is padded to a length of 36. The padding is uniform (i.e. a vector of [0.25 0.25 0.25 0.25]) and symmetrical around the SELEX sequence. The reverse complement vector is added to the dataset pool.

For each loaded sequence a label is generated. A label vector is a vector where all of the values are '0' and one value is '1', representing the class. For example, if only SELEX_0 and SELEX_4 are the input data, each sequence from SELEX_0 will have a label [1, 0], and each sequence from SELEX_4 will have a label [0, 1].

After generating the sequence vectors, i.e. the training data, and the label vector, i.e. the output, the data is shuffled and forward to the neural network for training and testing.

The PBM file contains approximately 42K sequences of length 36. These sequences are OneHot encoded and are not padded since they are already at the right size.

Note – no padding of the SELEX files was also explored, but did not show superior results over the padding method. In this case, each sequence in the PBM file was cut into $36 - L$ sub-sequences, where, 36 is the length of the PBM sequence and L is the length of the SELEX sequence.

Network architecture

We have explored numerous network architectures, including CpGenie [1] and DeepBind [2]. We have tried changing the architecture by making the network more or less deep, changing the kernel sizes of the network, changing the number of filters, the amount of fully connected layers and the overall hyper-parameters. Most of the changes, such as making the network deeper or increasing the number of filters, did not critically increase the performance of the network.
It has been well established that the k-mers are the cause for protein binding [3]. The kernel of the first convolutional layer is intended to find k-mers of known sizes, such as 3, 5, 6, 8, 10. The first DNN model that ranked 25 out of the first 100 sequences in the PBM file is shown in Fig. 1 and is referred to as PBM25.

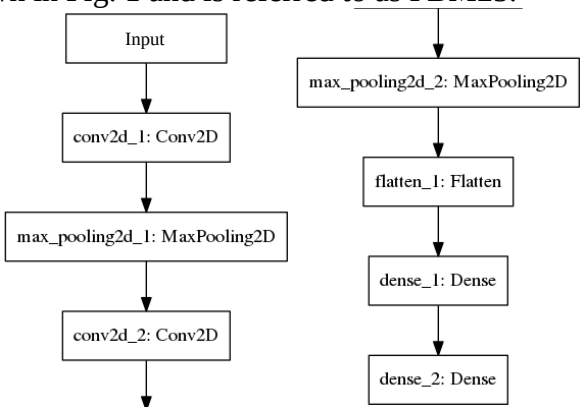

Fig. 1 – DeepBind like model. PBM25.

PBM25 has an architecture similar to the one presented in DeepBind. PBM25, however, has an additional convolution layer over the one presented in DeepBind. The PBM25 architecture has 6 layers. The first layer is a convolutional layer with 32 filters of kernels of size 6x4. The second layer is a MaxPooling layer with size of 5x1 and step size 1x1. The third layer is a convolutional layer with 16 filters of kernels of size 8x4. The fourth layer is a MaxPooling layer with size of 5x1 and step size 1x1. The fifth layer is a fully connected layer with 128 neurons in the output. The final layer is a fully connected layer, with either 2 or 5 neurons, depending on the amount of SELEX files used.

The second model explored is an inception model [4] with 3 different kernel sizes, (3,4), (5,4), (10,4), that work in parallel. This way it is possible to directly learn 3 k-mers. Fig. 2 shows the entire DNN model.

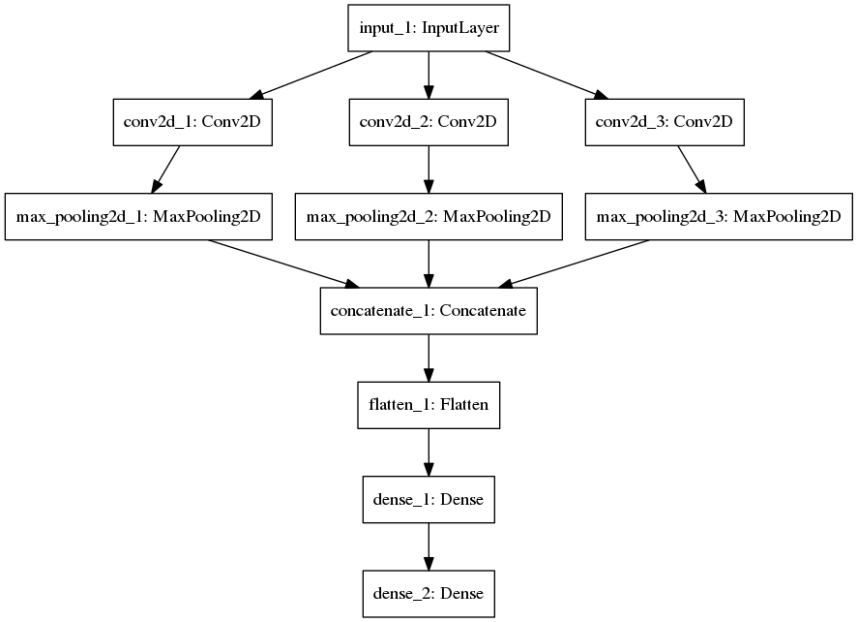

Fig. 2 – Inception model.

Conv2d_1 has 16 filters and a 3x4 kernel size which is forward to a maxpool layer of size 3x4. Conv2d_2 has 16 filters and a 5x4 kernel size which is forward to a maxpool layer of size 5x4. Conv2d_1 has 8 filters and a 10x4 kernel size which is forward to a maxpool layer of size 10x4. These layers are concatenated and flattened and inserted into a fully connected 128 neuron layer. The output of the network is the number of SELEX files used, i.e., 2 – for SELEX_0 and SELEX_4/5/6; or 5 – for SELEX_0 and the last 4 SELEX files.

Hyper parameters

Several optimization methods, such as batch Normalization [4], setting of initialization of weights and values of dropouts were explored. All of the layers have a ReLU activation function, except for the final layer that uses sigmoid activation function.
The loss function used is categorical cross entropy, and the metric used is accuracy. We have used different optimizers such as, Adam, Adagared and Adadelta were used and showed the same training results. SGD was also explored but showed inferior results. The default values (learning rate, momentum, epsilon, etc.) gave best results.

Results

A surprising result received is that as we lower the number of training data, the accuracy of the training and validation data was raised. Our first intuition was that we are dealing with a case of overfitting, however, the accuracy of the testing data was raised as well. We have seen that when training with approximately 30,000 sequences optimal results are achieved. We have tested correct sequence classification on 60,000 different sequences.
Fig. 3 shows the accuracy, loss and AUPR of correctly classifying a SELEX sequence to its correct origin, i.e., SELEX_0 or SELEX_4. The Figure shows overlaid results of 6 unrelated TF experiments. The figure shows that for 5/6 of these experiments the accuracy is above 80%. When running on all 123 TF experiments (not shown in figure for clarity) only 7 experiments got accuracy below 70%, as opposed to almost 100% when trained on 300,000 data sequences.

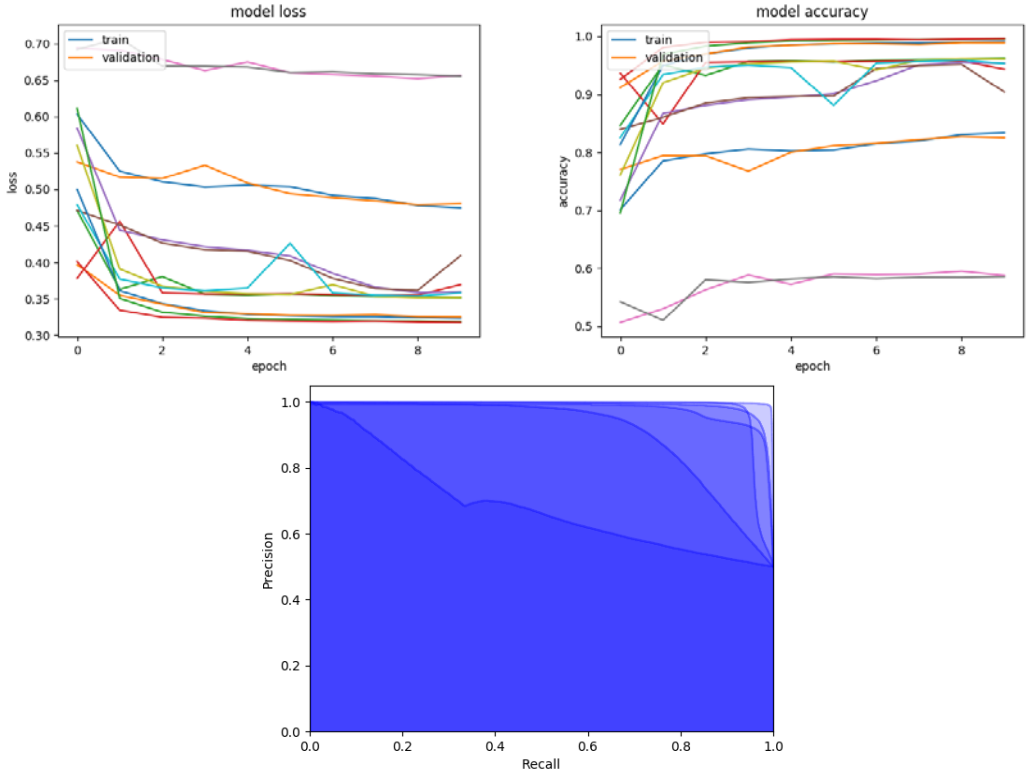

Fig. 3 – Loss (top left), accuracy (top right) and AUPR of testing data, for PBM25 model with two SELEX files. The charts show overlaid results of 6 unrelated TF experiments.

Unfortunately, the main purpose of this project and paper is to rank the PBM file associated with the SELEX files. It seemed that most of the changes discussed in this report had small or no effect on the AUPR of the ranking the PBM.

Fig. 4 compares the results of the 4 DNN models used to rank the PBM files. There are mainly two models: the PBM25 model and the inception model named tower. Each of these models were experimented using either two SELEX files (PBM25_2 and tower2) or five SELEX files (PBM25_5 and tower5). The results show here were performed on the TitanX GPU, however, the performance on the Colfax was almost the same, just longer process time which we shall address.

Reducing the number sequences on which to perform the training process clearly had an excellent effect on the time for the entire process. All of the models finished training the model, perform testing of SELEX classification and rank the PBM in under 20 seconds. The PBM25_2 was naturally the fastest. Even on the Colfax server, the training time took on average 43 seconds for an entire run.

The number of PBM sequences that were actually retrieved from the top 100 sequences in the PBM file was relatively similar to all of the models with an average of 6, but the best was the PBM25 with 5 SELEX files with an average of 9 and maximum value of 46. The minimum is 0 for all models. Unfortunately, none of the models were able to maintain a high retrieval value throughout the 123 TF experiments. Moreover, the sequences retrieved, which are from the top 100, were not ranked first resulting in poor AUPR results, as shown in the top figure of Fig. 4. The best average AUPR was 0.025 and even lower in some cases, even though the AUPR of some experiments reached 0.25. This is due to a large amount of experiments that retrieved 0 PBM sequences.

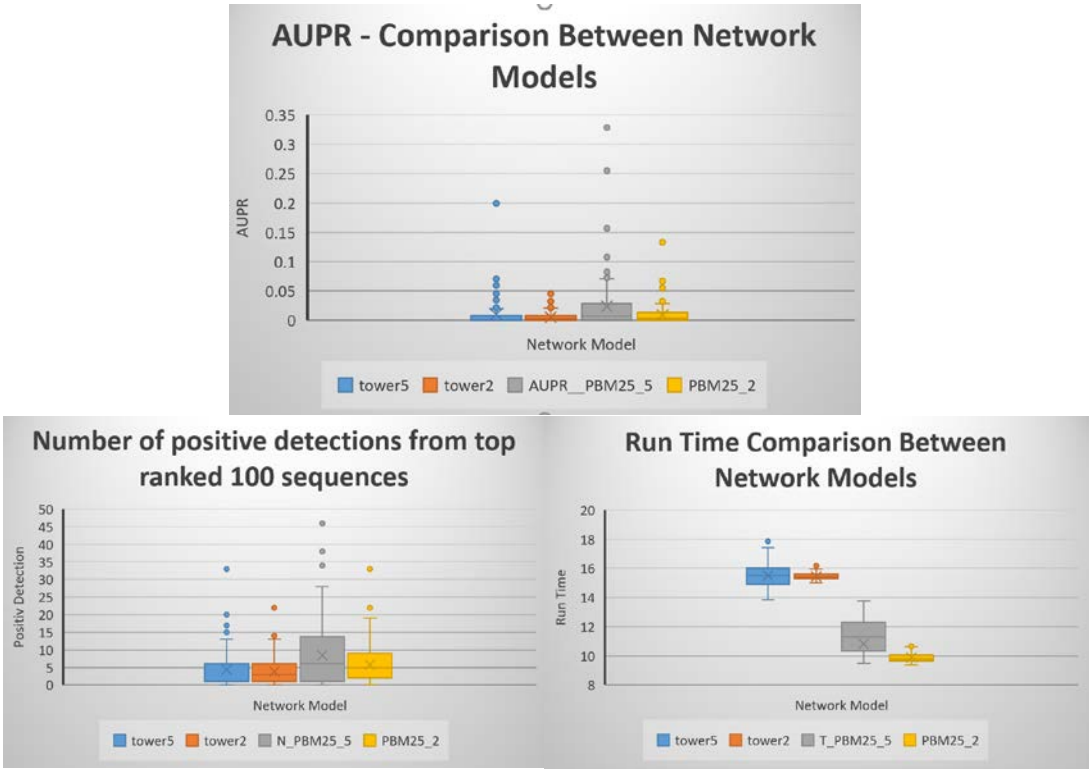

Fig. 4 – PBM results on all DNN model. Top: AUPR; left bottom: number of sequences correctly ranked in the top 100 sequences; right bottom: run time.

Conclusions

Predicting the Binding intensity of a protein to a DNA chain is not trivial, let alone when comparing two methods of intensity binding detection, e.g. SELEX vs PBM methods. It seems that the data has an integral role in the success of the ranking procedure, as there is very little preprocessing performed on the input data itself. It was extremely surprising to see that making the network deeper, and other optimization and hyper parameters have almost no impact on the overall result of the network. This might indicate mishandling the input data. Although, the overall results were mediocre, retrieving up to 35 out of 100 PBM sequences is much better than randomly choosing 100 sequences, which proves some sort of learning has occurred.

Reference:

[1] Zeng H, Gifford DK., "Predicting the impact of non-coding variants on DNA methylation", Nucleic Acids Res, 45 (11): e99.

[2] Alipanahi B, Delong A, Weirauch MT, Frey BJ., "Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning", Nat. Biotechnol. [Internet]. Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.; 2015 [cited 2015 Jul 27];33:831–8. Available from: https://doi.org/10.1038/nbt.3300

[3] Orenstein, Yaron, and Ron Shamir. "HTS-IBIS: fast and accurate inference of binding site motifs from HT-SELEX data." bioRxiv (2015): 022277.

[4] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[5] Sergey I, Christian S, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", arXiv:1502.03167v3 [cs.LG] 2 Mar 2015.