Brown University

Transcription Factor Binding Site Prediction with Convolution and Attention Model

CS2470 Final Report

Suchen Zheng

Xiling Zhang

Table of Contents

1. Introduction 1
2. Methodology 2
    2.1 Data 2
    2.2 Models 2
        2.2.1 Three Convolutional Layers 2
        2.2.2 Transformer Attention After All Convolutional Layers 3
        2.2.3 Transformer Attention After First Convolutional Layer 5
        2.2.4 Transformer Attention Using K-mers Before Convolutional Layers 5
3. Challenges 6
4. Results 8
        4.1 Convolutional Layers with Transformer 8
        4.2 K-mers Transformer Before Convolutional Layers 9
5. Reflection 9
        5.1 Problems and Thoughts 9
        5.2 Future Work 10
6. Link to GitHub 10
References 11
Tables: 12
Figures: 12

1. Introduction

The transcription factor (TF) binding is a biological process that regulate the gene expression. Transcription Factors are proteins that bind to DNA. When they bind, they influence the probability of nearby gene being transcribed into RNA. Every TF has a specific DNA sequence called its binding site motif. The binding site motifs are different for every TF and tends to be very short (10 bases or less). By recognizing its own motif, a TF is then able to bind itself to the DNA sequence. Wherever the motif exists in the DNA sequence, the TF will bind to it.

The function of TF binding process is to regulate gene expression, to activate or inactivate the gene to make sure the gene is expressed in the right cells at the right time. Therefore, identifying the TF binding site discovery is the first step in understanding the regulatory of the gene expression.

DeepSea [1] is a state-of-the-art solution to TF binding site finding which already bring about accuracy above 95%. It made use of a 3-layer convolutional layers. The convolution has its limitation that the long-range information cannot be aggregated before the network reaches a very deep layer. Based on this, the DeapSea works on the DNA sequence information only.

However, according to biology experimental study, there is a relationship between different motifs from the distance between each other to the arrangement of different motifs among the DNA sequence. Apart from the sequence, other aspects of the DNA also affect the motif. Many motifs are influencing by the physical shape of DNA and it is worth to mention that most DNA is wound around histones. Based on these facts, there is a complicated relationship among motifs. To predict their existence, more factors should be taken into our consideration.

In this project, we would like to predict the TF binding sites by capturing the dependency information among different motifs. To achieve this, the transformer model [2] with self-attention will be adopted. The transformer model is designed to be able to capture the relationship among the whole sequence regardless of the long range. Comparing with pure convolutional method, it is expected to be more capable in aggregating a long-range information without waiting for several rounds of information passing. Details of our attention model with self-attention will be discussed later.

2. Methodology

2.1 Data

The dataset we used in this project is retrieved from the DeepSea project. The dataset of the DeepSea project is from human GRCh37 reference. The training data consist of 440,000 samples and the test data consist of 45,000 samples. Every sample is a a 1,000-bp sequence represented by a 1000*4 matrix. The last dimension of size 4 corresponding bases A, T, C, and G in DNA. Each data sample is paired with a label vector of length 919. Each value in the label is representing a chromatin feature.

In our experiments, two subsets from the 440,000 samples were retrieved to train models. One is a 10,000-sample training dataset and another is a 60,000-samples training dataset. From the testing data, we get a 10,000-samples testing dataset as our consistent dataset. The original labels consist of 125 DNase features, 690 TF features, and 104 histone features. We select features from position 125 to position 815 of each label. The chromatin features in this range are representing the TF binding motifs.

2.2 Models

2.2.1 Three Convolutional Layers

We first implement the three-convolution-layer model with output channel: 320, 480, 960 for each convolutional layer respectively. First two convolutional layers have Maxpool1D and 0.2 dropout afterward. The third convolutional layer has dropout 0.5 without any maxpooling process. Details for convolutional layers are listed in table below. The output of the three convolutional layers will be feed into two Dense layers with the first one using relu as activation and the second one using sigmoid. Both output layer has output size 690 which is the same as the number of TF bind site in this dataset. We keep those three convolutional-layer setting for all our models in the project.

Index	**Layer**	**Feature/ Output Size**	**Kernel**	**Maxpool**	**Dropout**
0	Convolutional	320	8	4	0.2
1	Convolutional	480	8	4	0.2
2	Convolutional	960	8	N/A	0.5

Table 1: Three Convolutional Structure

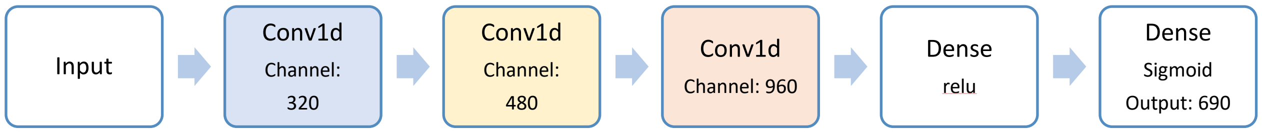

Figure 1: Pure Three Convolutional Layer Model

2.2.2 Transformer Attention After All Convolutional Layers

Now we start implement transformer model after all convolutional layers to help build positional relationship with convoluted data. The transformer model uses classic design as Ashish Vaswani mentioned [2]. Instead of building embedding matrix, we use the output of convolutional layers to work like embedding lookup process. It convert the input into an embedding matrix which has size: (batch_size, steps, output_channel_size). In our case, we start building dependency between each feature with all other features, where output_channel_size has the same functionality as embedding size.

After using convoluted output to multiple weight matrixes in order to get Queries, Keys and Values matrix, we apply the Equation 1 to calculate the output matrix Z. The input size and output size stay the same during Transformer model processing.

$$softmax\left(\frac{Q\ \times transposed(K)}{\sqrt{d_k}}\right)V = \ Z$$

Equation 1: Transformer Model Process

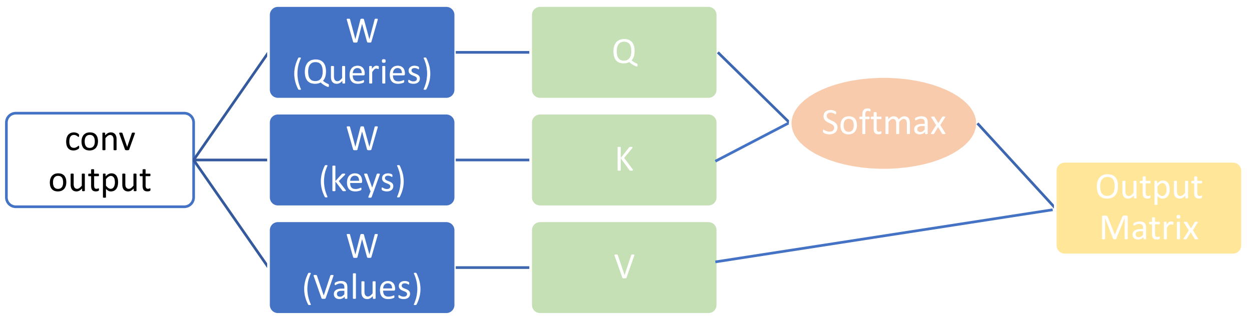

Figure 2: Transformer Model [3]

Therefore, our model now looks like in figure 3 in which transformer is placed after three convolutional layers. However, the output after training is as good as our expectation. We think that is because three convolutional layers put data relationship in high dimension and consequently makes the transformer failed to study from positional information. Since TF biding site is normally smaller than 10, after three convolutional layers, the positional details between TF binding site might be too complicated to extract.

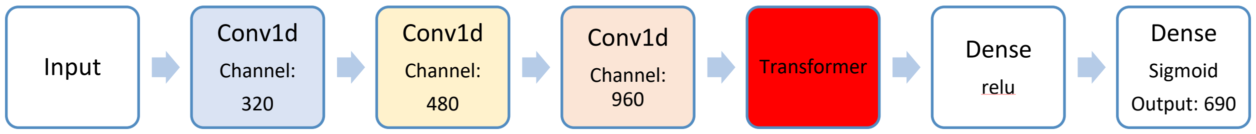

Figure 3: Transformer After All Conv layers

2.2.3 Transformer Attention After First Convolutional Layer

To get more location details about TF binding site, then we move transformer to be after first convolutional layer (figure 4). The AUROC output is correspondingly getting better but still not exceeds the three convolutional model.

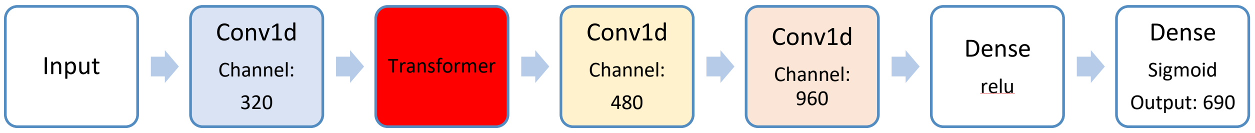

Figure 4: Transformer After First Conv

2.2.4 Transformer Attention Using K-mers Before Convolutional Layers

This time, we would like to have the transformer model at the beginning of the CNN before all three convolutional layers. To achieve this, we need to have an embedding representation of the input sequence. The raw DNA sequence contains only 4 characters (A, C, G and T) which makes the vocabulary size of the raw DNA sequence is too small for us to train an informative embedding matrix. Therefore, we adopt the k-mers method to enrich the vocabulary size of the input sequence. With a chosen k value, we will have a vocabulary size of 4^k. In our project, considering that the motif length is usually 10 or even less, we choose 4 as the k value. This provides us with a vocabulary size of 246 and keep a sufficient number of features for each motif site.

The k-mers was applied during the preprocess of the data. Before the k-mers process, each 1,000*4 matrix sample is converted to a vector of length 1,000 with 4 characters (A, C, G and T). After the k-mers process, we give id to each unique k-mer and convert the 1,000 length sequence to a sequence of ids with length 997. When every data sample is a list of ids, we are now able to train the embedding matrix for the ids and feed the embeddings into the transformer mode.

Index	**Layer**	**Feature/ Output Size**	**Kernel**	**Pool**	**Dropout**
0	Embedding	20	N/A	N/A	N/A
1	Transformer-Kmer	20	N/A	N/A	N/A
2	Convolutional	320	8	4	0.2
3	Convolutional	480	8	4	0.2
4	Convolutional	960	8	N/A	0.5
5	Fully-connected	690	N/A	N/A	N/A
6	Fully-connected	690	N/A	N/A	N/A

Table 2: Configuration of 7-layer CNN in our experiments, with an embeddinglayer at the beginning of the network and a transformer block right after theembedding layer. Layers that do not present the use of convolutional, poolingor dropout layers are omitted

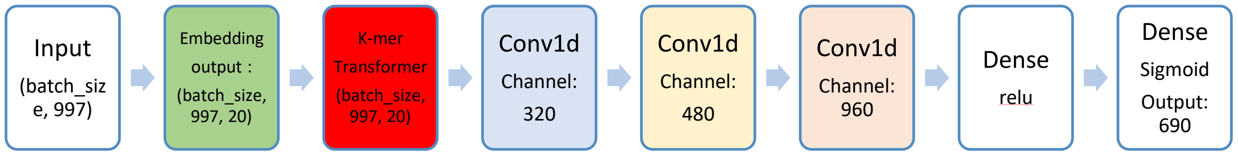
Input
(batch_size, 997)
Embedding output : (batch_size, 997, 20)
K-mer Transformer (batch_size, 997, 20)
Conv1d Channel: 320
Conv1d Channel: 480
Conv1d Channel: 960
Dense relu
Dense Sigmoid Output: 690

Figure 5: K-mer Transformer Before Conv

3. Challenges

The motive of using attention is that we can build positional relationship between all features as a complimentary of convolutional layers. However, we realize that when and where to use transformer attention model is highly important. Using attention after high convoluted data in this case only dampen our system. Even moving transformer model before convolutional layer using k-mers still fails to meet our expectation.

There might be some problems with our parameters for building embedding matrix. Since the data to train is very large, we tested some learning rate and output channel size, but we still did not explore more in model tuning.

Another challenge we have is to extract certain data from Encode project instead of using all shuffled cell type data from DeepSea project. The data is also too big for us to modified and use for our model. If we can separate cell type, it will be helpful to train and test on one or a few cell types to see how transformer model works based on known cell type. We can further train one some cell types and test on others to see if transformer improves the model with potential structural TF binding information.

4. Results

Model	Details		Epoch	AUROC	
				10,000 training data	**60,000 training data**
Pure CNN	3 convolutional layer		2	0.8244	0.8430
			5	0.8333	0.8517
Attention after 1st CNN layer	Single-head		2	0.8226	0.8266
			5	0.8257	0.7929
	Muti-head		2	0.8095	0.8274
			5	0.8259	0.7922
Attention after 3rd CNN layer	Single-head		2	0.7728	0.7867
			5	0.7822	0.7924
	Muti-head		2	0.7716	0.7892
			5	0.7866	0.7940
K-mers + attention	k=4	Single-head	2	0.7750	N/A
			5	0.7800	N/A
		Muti-head	2	0.7790	N/A
			5	0.7860	N/A

Table 3: Result Table

4.1 Convolutional Layers with Transformer

If using 10000 training data, generally speaking, the earlier we use transformer model, the better result we achieve. Using transformer after three convolutional layers all give AUROC less than 80. If we train using 60000 data, transformer after first convolutional layer has strange result that the more epochs it trains, the AUROC starts going down. However, the basic trend follows 10000 training tendency which means using transformer earlier achieves better result.

For different size of training data, single-head and muti-head result does not make much difference disregarding the position of applying transformer.

4.2 K-mers Transformer Before Convolutional Layers

The k-mers transformer model with k equals to 4, gives us a AUROC score from 0.77 to 0.79 when using different number of heads and training the model with different number of epochs.

With 10,000 training data, when we are using the single head self-attention, the AUROC score for the testing result is 0.7750 when running 2 epochs and 0.7800 when running 5 epochs. When we use a multi-head self-attention model with 3 heads, the AUROC score for the testing result is 0.7790 when running 2 epochs and 0.7860 when running 5 epochs. The 3 heads transformer model gives a result that is a little bit better than the single head model.

5. Reflection

5.1 Problems and Thoughts

The models we designed in this project did not work out in our expectations. Using the transformer model with self-attention seems not to help the prediction task. But the experiments still bring us some results and thoughts.

Our first thought is about the biology facts of this project. The idea of this project is coming from the biology experiment results saying that the motif binding is not only a result of the sequence but also affected by the distance and arrangement of other motifs. However, it is not known in what range of distance the existence of a motif can be influenced by other motifs. If this range of distance is relatively small, which means motifs would not affect each other, if they are of a very long distance, then it is meaningless trying to capture the relationship between all motifs in a sequence. If this is the case, then we are adding many redundant information to our model by putting the whole sequence into the transformer model.

Moreover, in this project, we are mostly focused on model architecture design. We tried various positions in the pure CNN model to add the transformer and use various methods to implement the model. However, the results don't have much difference among different methods and different architecture. One possible reason for this problem may be the hyperparameter. If we are not using the best set of hyperparameters, even the architecture of the model is good, the training result may not be that satisfying.

5.2 Future Work

Considering the first thought on the biology fact, we can reduce the length of the DNA sequence as the model input. Using a larger k to do the k-mers processing on the raw data sample, we will get a larger number of training data with shorter sequence length. It would be a huge success if we can figure out what length of the input sequence the transformer can work on. If the transformer can improve the prediction result within some maximum sequence length, then we will be able to inference what is the influencing range between motifs. It is possible that the range to too small that CNN is already able to process within the range, then we also get to know that it is useless to apply the transformer here.

Regarding the second thought on parameter tuning, there is one difficulty in this project that hampers us from finish this part. The running time of training this model takes too long, it is not possible to finish the parameter tuning within the project time. This would be one of the future work, we'd love to find out the best parameter setting that can improve the performance of our models.

On the other hand, instead of applying transformer to train data, we can try using it for learning labels data. There are some research trying to pre-train the label to extract the positional information [3]. In this label training research, for each label, they learn an embedding for the positive, negative, and unknown values. This can be another direction to dig into in future work.

6. Link to GitHub

https://github.com/xiling42/DL_Final_RedPandaGo

References

[1] O. G. T. Jian Zhou, "Predicting effects of noncoding variants with deep learning–based sequence model," *Nature Methods volume ,* 2015.

[2] N. S. N. P. J. U. L. J. A. N. G. L. K. I. P. Ashish Vaswani, "Attention Is All You Need," 2017.

[3] G. Baid, "An Attention-Based Model for Transcription Factor Binding," 2018.

[4] I. G. D. M. A. O. Han Zhang, "Self-Attention Generative Adversarial Networks," 2018.

[5] B. Ramsundar, "Transcription Factor Binding," in *Deep Learning for the Life Sciences: Applying Deep Learning to Genomics, Microscopy, Drug Discovery, and More*, 2019.

Tables:

Table 1: Three Convolutional Structure 3
Table 2: Configuration of 7-layer CNN in our experiments, with an embeddinglayer at the beginning of the network and a transformer block right after theembedding layer. Layers that do not present the use of convolutional, poolingor dropout layers are omitted 6
Table 3: Result Table 8

Figures:

Figure 1: Pure Three Convolutional Layer Model 3
Figure 2: Transformer Model [3] 4
Figure 3: Transformer After All Conv layers 4
Figure 4: Transformer After First Conv 5
Figure 5: K-mer Transformer Before Conv 6