DREAM Challenge 2022
Predicting gene expression using millions of random promoter sequences by
[Peppa]

Abstract

Numerous deep learning methods have been developed to predict gene expression from pure DNA sequences. Therefore, we decided to just use the state-of-the-art method, i.e., the Enformer model from DeepMind [1]. The Enformer model consists of several convolution-and-pooling layers, multiple transformer layers, and a final pointwise convolution layer. The loss function is a poisson loss between predicted values and true values. One important thing to note is that the Enformer model is specifically designed for large DNA sequences (e.g., 200kb) so that it can learn enhancer-promoter interactions to accurately predict gene expression. Due to the GPU memory size and the time I have, I decided to make the model smaller by using less number of filters (from 1536 to 192), smaller conv1D width (from 15 to 5 for the stem layer and from 5 to 4 for the conv_tower layer), and fewer number of transformer layers (from 11 to 4). The original Enformer model has ~200M parameters and I reduced the number to less than 2M. I found that data augmentation (shift the input DNA sequence by +/- 1bp) only increased the model performance a little bit. Overall, I am very satisfied with the result I have (0.717 for ScorePearsonR in the open leaderboard stage) considering the time I spent.

1. Description of data usage

Both training and testing DNA sequences were converted to one-hot encoding using the janggu [2] library with a fixed sequence length of 110bp and zero padding if the sequence is shorter (truncated if the sequence is longer). The one-hot encoded DNA sequences and target values were stored as numpy array. Training data were randomly divided into training and evaluation with evaluation set fixed to be 10,000. Training sequence length is also truncated into 85bp (i.e., the first 15bp and the last 10bp of the original sequence were removed, "train_X[train_index,15:100,:]").

For data augmentation, I shifted the training sequence by +/- 1bp (i.e., train_X[train_index,14:99,:] and train_X[train_index,16:101,:]).

2. Description of the model

I didn't invent any wheels. I basically modified the original Enformer model and reduced the number of filters from 1536 to 192, conv1D width from 15 to 5 for the stem layer and from 5 to 4 for the conv_tower layer, and the number of transformer layers from 11 to 4. I think such modifications are important so that I can train the model in a reasonable time. The model structure is shown below:

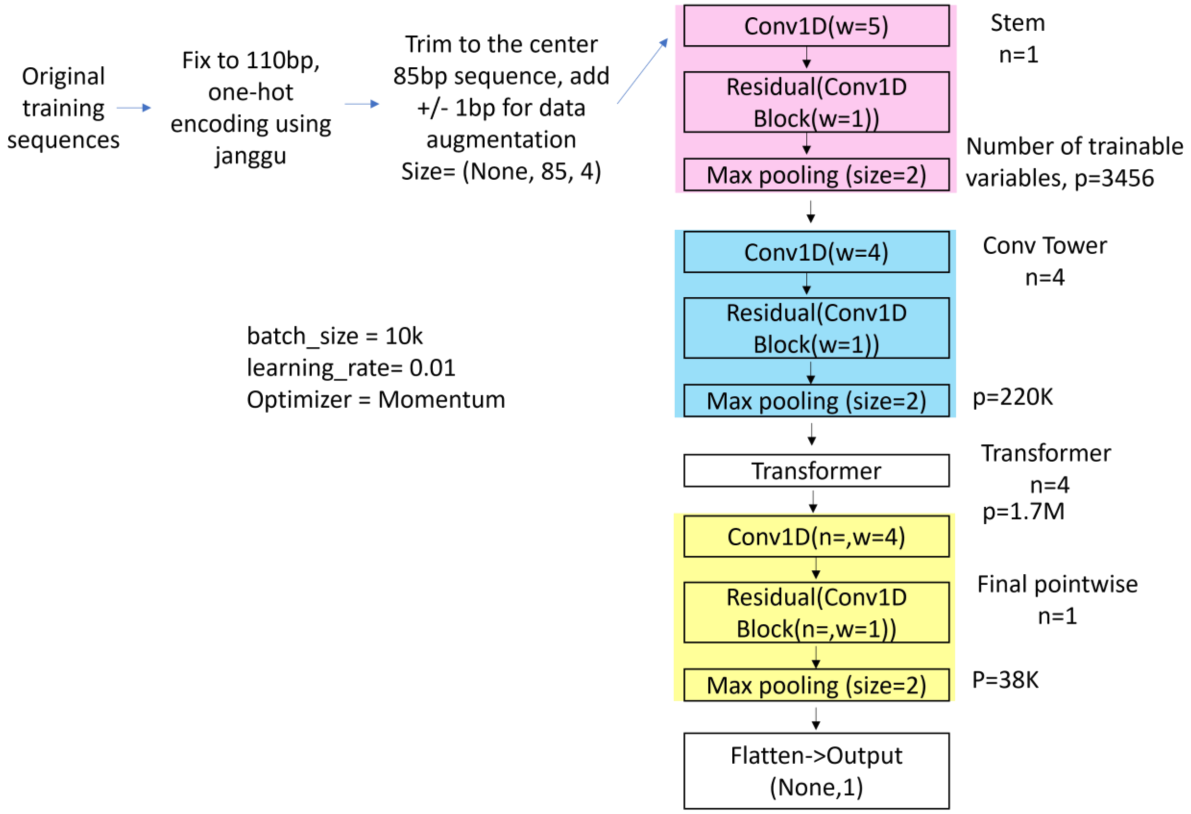
Original training sequences
Fix to 110bp, one-hot encoding using janggu
Trim to the center 85bp sequence, add +/- 1bp for data augmentation Size= (None, 85, 4)
Conv1D(w=5)
Residual(Conv1D Block(w=1))
Max pooling (size=2)
Stem n=1
Number of trainable variables, p=3456
Conv1D(w=4)
Residual(Conv1D Block(w=1))
Max pooling (size=2)
Conv Tower n=4
p=220K
batch_size = 10k
learning_rate= 0.01
Optimizer = Momentum
Transformer
Transformer n=4
p=1.7M
Conv1D(n=,w=4)
Residual(Conv1D Block(n=,w=1))
Max pooling (size=2)
Final pointwise n=1
P=38K
Flatten->Output (None,1)

Figure 1. My own Enformer model structure. The color scheme follows the original paper.

3. Training procedure

Loss function is possion loss. No regularization was used. Optimizer is Momentum with a learning rate of 0.01 and momentum of 0.9. The training and evaluation score per epoch is shown below. Overall, I found the pearson correlation I got from my own evaluation set is similar to the one I got from leaderboard. For final submission, I use the model from epoch 282 (blue vertical line in the figure below).

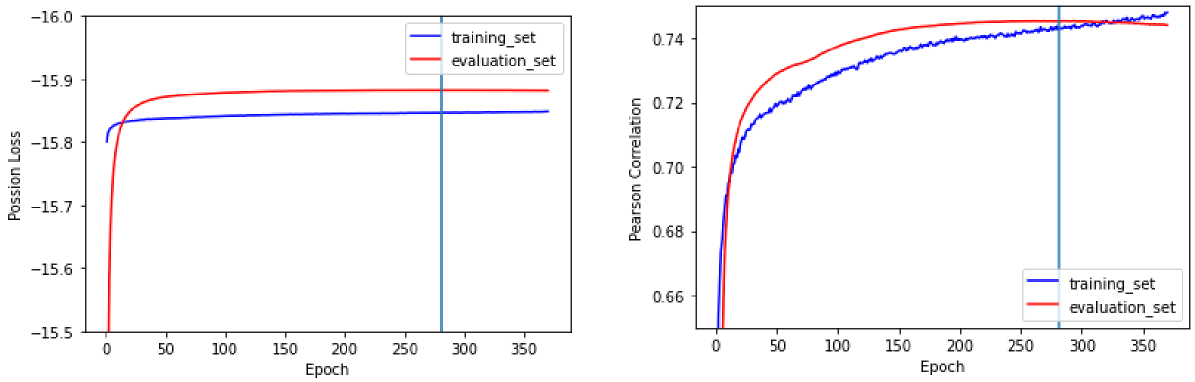

Figure 2. Training and evaluation score per epoch.

4. Other important features

I think the number of convolution filters and the model depth are very important features to have a good performance (but training time is increased a lot, I didn’t spend time waiting for them to be finished).
I also think reducing the model complexity and trying to fit the model with more data helps in some extent.

5. Contributions and Acknowledgement

Name	Affiliation	Email
Yichao Li	Department of Hematology, St. Jude Children’s Research Hospital, Memphis, TN, USA	Yichao.Li@stjude.org

6. References

[1] Avsec, Žiga, Vikram Agarwal, Daniel Visentin, Joseph R. Ledsam, Agnieszka Grabska-Barwinska, Kyle R. Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R. Kelley. "Effective gene expression prediction from sequence by integrating long-range interactions." Nature methods 18, no. 10 (2021): 1196-1203.

[2] Kopp, Wolfgang, Remo Monti, Annalaura Tamburrini, Uwe Ohler, and Altuna Akalin. "Deep learning for genomics using Janggu." Nature communications 11, no. 1 (2020): 1-7.