DREAM Challenge 2022
Predicting gene expression using millions of random promoter sequences
by
Team Wan&Barton_BBK

Abstract

We adopted a modified Temporal Convolutional Networks (TCN) to predict the expression profiles of random promoter sequences. Using TCN's capacity on dealing with long sequential inputs and our newly proposed loss function, our model successfully outperformed two benchmark methods according to the leaderboard results.

1. Description of data usage

The data preprocessing consisted of two steps. First, all training sequences were grouped into different bins, where each bin holds sequences of the same length. The bins that hold less than 1,000 sequences were removed from the final training dataset, leading to 6,728,720 sequences in total. 23 out of 64 bins were included in the final training dataset, whilst the lengths of the selected sequences range from 89 to 111, and 110 is the dominant sequence length. Second, the remaining 6,728,720 sequences were broken into characters with each character encoded using a dictionary holding five characters, i.e., A, C, G, N and T. Note due to the highly noisy distribution retained in the original training dataset (The Dream Challenges consortium, 2022), we exploit all 6,728,720 sequences to train our final prediction model for the competition.

2. Description of the model

We used a modified Temporal Convolutional Networks (TCN) and a newly proposed loss function to train our predictive model for this competition. TCN (Bai, et al., 2018) is a variant of 1D convolutional neural networks for coping with time-series and sequence data. Recent work has revealed TCN to outperform other popular sequence models, e.g., LSTM (Hochreiter & Schmidhuber, 1997) and Transformer (Vaswani et al., 2017), on a variety of predictive tasks (Bai, et al., 2018).

In general, TCN is constructed as a type of stacked neural networks, where each hidden layer has the same length as the input layer in order to guarantee that the prediction for the target time point t_i depends on all previous time points' information (i.e., t_0, t_1, t_2, …, t_{i-1}). The well-known dilated convolutional operation (Yu & Koltun, 2016) was also exploited in order to expand the receptive field when coping with long input sequence. The residual connection (He et al., 2015), weight normalization (Salimans & Kingma, 2016), and spatial dropout (Srivastava et al., 2014) mechanisms were also used to construct the convolutional layer. In our TCN model, the input promoter sequence's nucleotides were encoded as a type of character-level embeddings, which are parameterized as the learnable first level of TCN. The final output is generated by a linear layer that was added on top of the last residual block in order to predict the expression profiles by using the learned feature representation of the entire target promoter sequence's nucleotide combinations.

3. Training procedure

In general, we used the preprocessed training dataset (including 6,728,720 sequences) to train our model. As TCN can be trained on variable length inputs, our model is trained on 105,123 batches that were sequentially created by 23 bins, where each batch holds sequences of the same length. We used the weekly leaderboard testing sequences to briefly estimate the predictive performance of our model. As mentioned in Section 1, due to the highly noisy distribution of the training dataset, we believe using all training sequences would lead to the best model generalizability w.r.t. the testing sequences. Note that, due to the potential bias of the leaderboard testing sequences (since they were randomly sampled as ∼13% of the entire testing dataset), we didn't adopt any early-stopping strategy based on the model

performance evaluated by using the leaderboard testing sequences, which were nevertheless used to conduct a hyperparameters optimization. The detailed hyperparameters for our final model are listed in Table 1.

Table 1. The values of the hyperparameters for our TCN model.

Hyperparameters	**Values**
Number of levels	5
Batch size	64
Embidding dim	100
Channel size	384
kernel size	3
Stride size	1
Temporal block dropout rate	0.1
Embedding dropout rate	0.1
Gradient clipping	0.15
Learning rate	0.05
Output size	1
Step size to measure training performance	2
Number of epochs	100

We also proposed a new loss function *L* for training our TCN model for this competition. As demonstrated in Equation 1, the Mean Squared Error (MSE), Pearson correlation, and Spearman correlation values are used to compute the value of *L*, whilst a weight for the MSE is applied in order to reflect the importance of those two types of correlations. Due to the time constraint, we only tried 0.1 as the weight for training our model, since we assumed that those two types of correlation coefficients values are more important for this competition. Figure 1 shows the curves of the loss function and its different elements over 100 epochs training. It is obvious that the value of the loss function dramatically decreased due to the increased values of *Pearson*^{2} (~0.58), and *Spearman* correlation coefficient (~0.76), with the increased number of training epochs.

$$\mathcal{L} = MSE * w - (PearsonR^2 + SpearmanR) \tag{1}$$

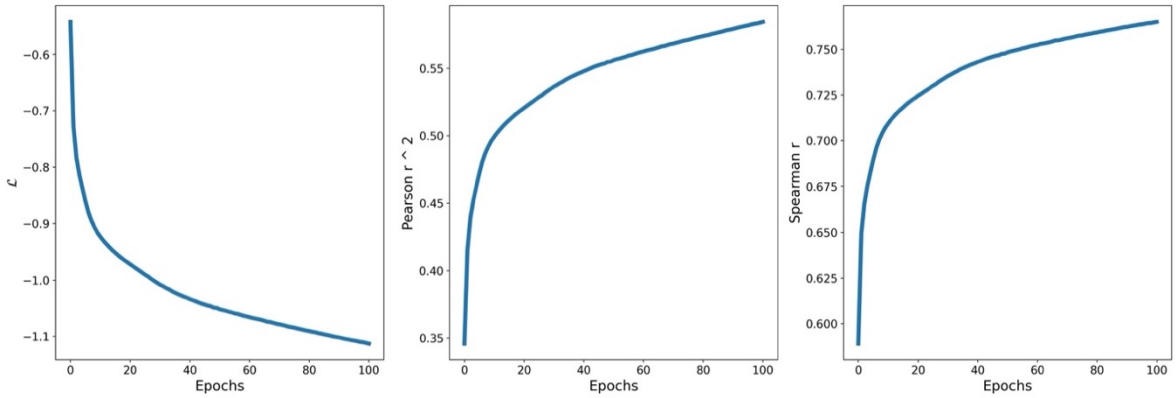

Figure 1. The training curves for the loss function and its different components.

4. Other important features

Regarding the final predictions, we don't specify weights for different evaluation metrics. In addition, due to the large number of training sequences, we briefly estimated the computational training cost – the actual training time for using one of the provided Google cloud TPUs. For 100 epochs training, it took ~120 TPU hours, which encourage larger scale

of hyperparameters (including the weight *w* in our newly proposed loss function) optimization and model selection in the future research.

5. Contributions and Acknowledgement

5.1 Contributions

Name	Affiliation	Email
Ibrahim Alsaggaf	Birkbeck, Univerity of London	i.alsaggaf@bbk.ac.uk
Patrick Greaves	Birkbeck, Univerity of London	p.greaves@bbk.ac.uk
Carl Barton	Birkbeck, Univerity of London	c.barton@bbk.ac.uk
Cen Wan (corresponding author)	Birkbeck, Univerity of London	cen.wan@bbk.ac.uk

5.2 Acknowledgement

We thank DREAM Challenges for organizing this competition, and Google Research for providing TPU resources to make training the model possible. We also acknowledge the support by the Department of Computer Science and Information System, and the Birkbeck GTA programme.

6. References

Bai, S., Kolter, J. Z., and Koltun, V. (2018). *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*. arXiv, 1803.01271.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). *Deep residual learning for image recognition*. arXiv, 1512.03385.

Hochreiter, S. and Schmidhuber, J. (1997). *Long short-term memory*. Neural computation, 9(8), 1735–1780.

Salimans, T. and Kingma, D. P. (2016). *Weight normalization: A simple reparameterization to accelerate training of deep neural networks*. arXiv, 1602.07868.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). *Dropout: A simple way to prevent neural networks from overfitting*. Journal of Machine Learning Research, 15(56), 1929–1958.

The Dream Challenges consortium. (2022). *An introduction to the DREAM challenges - Predicting gene expression using millions of random promoter sequences*. Retrieved from https://drive.google.com/file/d/150eHgy-x3R9ZMBENoDT6zfq4KLgKcmfn/view

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ..., and Polosukhin, I. (2017). *Attention is all you need*. arXiv, 1706.03762.

Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. arXiv, 1511.071223.