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

Abstract

For Dream Challenge 2022, we propose our deep neural network model named DeepGXP (Deep learning model for the prediction of Gene eXpression using Promoter sequence) for predicting gene expression through random promoter sequences. The model consists of four main components: a convolutional neural network (CNN), bidirectional long-short term memory (Bi-LSTM), another CNN block, and fully-connected (FC) layers, primarily inspired by a hybrid convolutional and recurrent neural network proposed in Quang & Xie (2016). Considering common motif lengths of training data obtained from ProSampler, we used two different kernel sizes (k = 9, 15) for 1d-CNN based on the DeepFam model. The extracted information of local motifs is then fed to Bi-LSTM, which learns long-term forward and backward dependencies of motifs. Another CNN block computes the output matrix after this process which is then finally led to the FC layers. We preprocessed the training data by matching their lengths to test data sequences' length (110bp). In order to avoid disrupting overall data integrity, sequences longer than 110bp are trimmed to the right. Sequences shorter than 110bp are padded with original vector sequence on both sides randomly to give the effect of data augmentation. To give reverse complement (RC) equivariance to our model, we used post-hoc conjoined setting suggested in Zhou et al., 2020. Additionally, at inference time, we applied test time augmentation by averaging the predictions of the original sequence, the shifted sequences, and reverse complement of those sequences. Lastly, to be as unbiased as possible for the test predictions, each test prediction was quantile-transformed using the distribution of standardized measured expressions of train sequences as a post-processing. Altogether, we suggest a highly effective prediction model that can predict the expression level based on promoter sequences independent of test data distribution.

1. Description of data usage

Following the length of test data and majority of training data sequences, we preprocessed training data by adjusting data size unanimously into 110 bp. Sequences longer than 110 bp were trimmed from right to maintain data integrity during CNN motif finding. In order to reflect the actual experimental environment to the furthest extent, shorter sequences were padded with original vector sequences. We used three ways to pad short sequences for train data: padding only at 5' end, only at 3' end, or at both ends with respective ratios of 4:3:2. We expected such strategy to serve as a form of data augmentation during the training. To accelerate model training, we standardized the measured expression in training data by subtracting 11, which is the most commonly appeared bin value, and then dividing them by their standard deviation. The bases (A, C, G, and T) were one-hot encoded while N was encoded as [0, 0, 0, 0]. For further augmentation of input sequences, both forward sequences and corresponding RC sequences were simultaneously used for training. For the evaluation of our model, we split training data into five chunks and used 5-fold cross-validation scheme.

2. Description of the model

As illustrated in **Figure 1**, our model consists of four main components: CNN, Bi-LSTM, another CNN block, and FC layers. Note that our model is mainly based on DanQ, a hybrid convolutional and recurrent neural network, proposed in Quang & Xie (2016). The one-hot

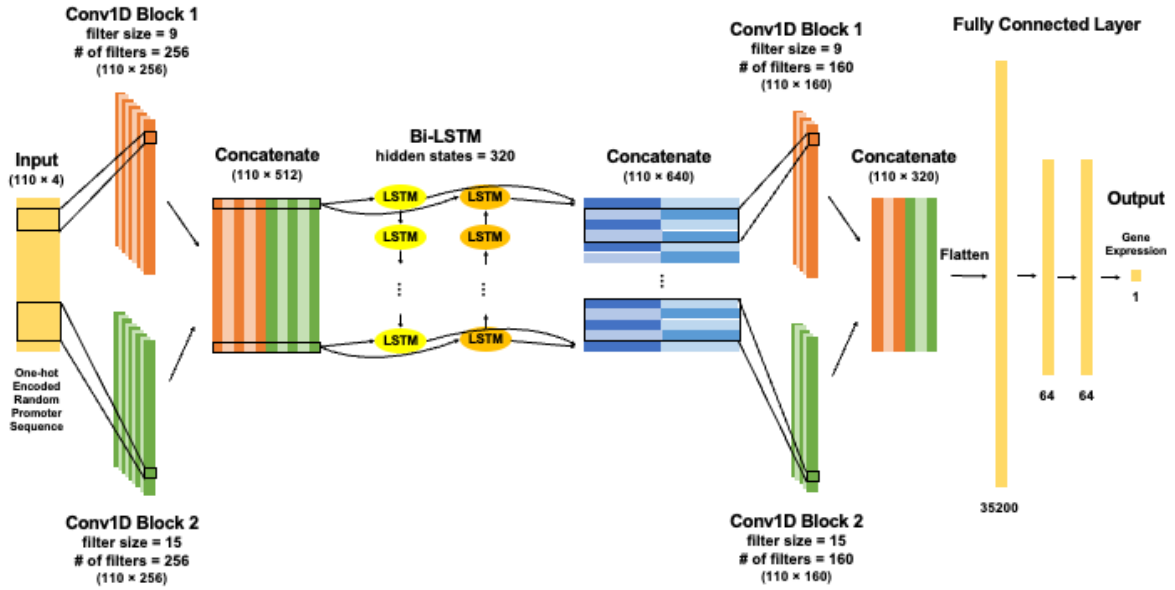

Figure 1. DeepGXP model architecture. It predicts the gene expression level based on one-hot encoded random promoter sequence.

encoded input sequence is first scanned by 1d-CNN. Inspired by DeepFam (Seo et al., 2018), we used convolutional layers with two different kernel sizes to search motifs of various lengths and concatenated the outputs in channel dimension. We set kernel sizes to 9 and 15, corresponding to the common motif lengths of training data sequences searched through ProSampler (Li et al., 2019). The channel size of each convolutional layer is 256 and we subsequently applied ReLU activation and dropout with probability of 0.2. The output matrix from the CNN block contains sequential information of local motifs. Such information is then fed to Bi-LSTM, which captures long-term forward and backward dependencies of motifs. The two dimensions of LSTM hidden states were set to 320 each, resulting in 640 dimensions after concatenation. After the LSTM block, another CNN block, which has the similar structure as the first CNN block, is applied. The output then goes through the fully-connected layers to make a final prediction of expression value.

3. Training procedure

Our model is implemented in PyTorch (Paszke et al., 2019) and optimized with the AdamW optimizer. Huber Loss was used for the loss function of the model. We used Optuna (Akiba et al., 2019) to optimize our hyperparameters, and as a result, the initial learning rate of 0.0015, weight decay of 0.01, and batch size of 512 were used for training. Cosine annealing learning rate scheduler was used to decrease learning rate progressively, reaching 0 at the end of the training. We set the random seed as 40 for model training and 42 for prediction. The essential package versions of the environment are CUDA=11.2, python=3.9.12, cuDNN=8200(8.2.0), torch=1.11.0. Environment requirements are exported into a yaml file, which is stored in the submitted github repository. Among various strategies to make model RC-equivariant, we found the “post-hoc conjoined” setting (Zhou et al., 2020) worked best. In training time, losses calculated from both forward and RC strand are averaged for backpropagation, which is equivalent to data augmentation with RC strand. At test time, we used test time augmentation for prediction. When the test sequence was given, the sequence was shifted 1 and 2 bp from both right and left side, and padded with vector sequence to maintain the original length (110bp) while shifting. As a result, 4 shifted and 1 original sequences for each forward and

reverse complement were used for predicting expression. The average of the 10 augmented sequences' results was used for final expression prediction of the given test sequence.
We trained the model for 15 epochs and conducted 5-fold cross-validation with given train data. We used a wandb logger (Biewald, 2020) to track the training procedure of the model, and the results (Huber loss, Pearson correlation, and Spearman correlation) were as below (**Table 1, Figure 2 and 3**).

Table 1. Huber loss, Pearson correlation, and Spearman correlation of train and validation data. Mean and standard deviation of 5-fold cross-validation results are written. (Random state 123456789 was used for data split in 5-folds.)

Data	Loss (Huber)	Pearson Correlation	Spearman Correlation
Train	0.1853(0.001215)	0.7688(0.001468)	0.786(0.003371)
Validation	0.1932(0.0005587)	0.7536(0.0009638)	0.7701(0.0007393)

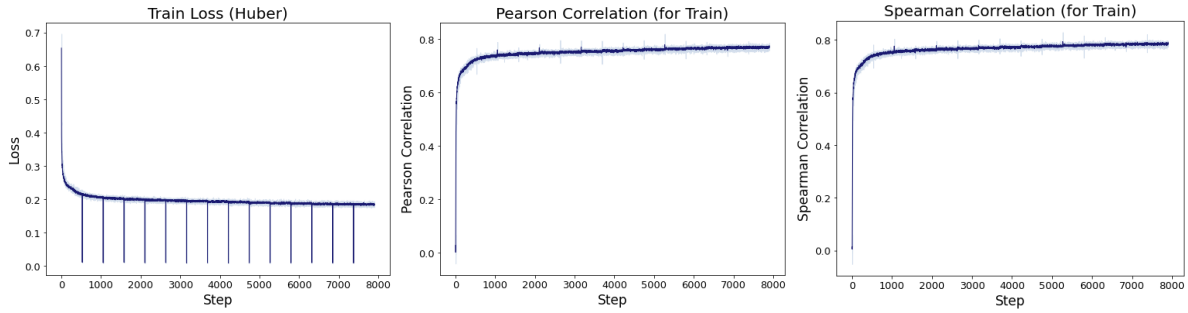

Figure 2. Huber loss, Pearson correlation, Spearman correlation of train data while training 15 epochs (average graph of 5-fold cross-validation).

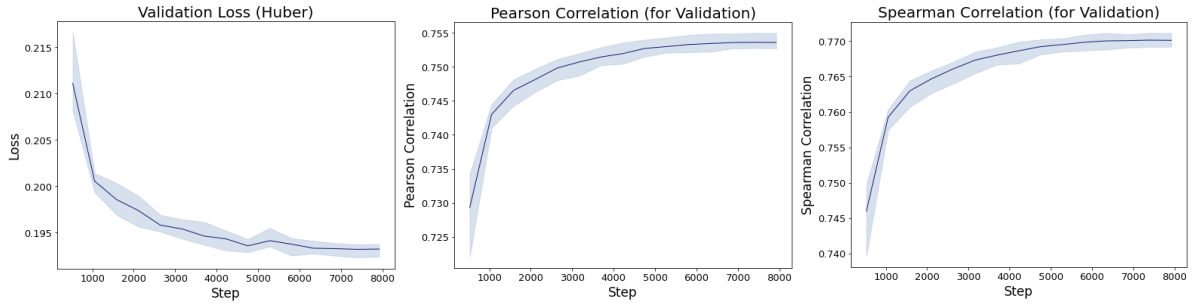

Figure 3. Huber loss, Pearson correlation, Spearman correlation of validation data while training 15 epochs (average graph of 5-fold cross-validation).

4. Other important features

As our training data follows Gaussian distribution, our trained model naturally performs prediction in such form. To be as unbiased as possible for the distribution of test sequences, we transformed each test prediction into a corresponding quantile value among the standardized measured expressions in training set. Please note that as this post-processing only requires the distribution of the measured expression of training sequences, it can be independently applied for the prediction of individual test sequence.

※ We observed that the model does not converge with certain random seeds. As it can be detected at the initial phase of training, so please monitor the loss (or metrics) at the beginning of the training to assure that the model is being properly trained when reproducing the model.

5. Contributions and Acknowledgement

5.1 Contributions

Name	Affiliation	Email
Dohoon Lee	(Team lead) Bioinformatics Institute, Seoul National University, Seoul, South Korea BK21 FOUR Intelligence Computing, Seoul National University, Seoul, South Korea	apap7@snu.ac.kr
Danyeong Lee	Interdisciplinary Program in Bioinformatics, Seoul National University, Seoul, South Korea	ldy9381@snu.ac.kr
Dohyeon Kim	Interdisciplinary Program in Artificial Intelligence, Seoul National University, Seoul, South Korea	scottkdh@snu.ac.kr
Nayeon Kim	Interdisciplinary Program in Artificial Intelligence, Seoul National University, Seoul, South Korea	ny_1031@snu.ac.kr
Sangyeup Kim	Department of Computer Science and Engineering, Seoul National University, Seoul, South Korea	sang2668@snu.ac.kr
Yeojin Shin	Department of Computer Science and Engineering, Seoul National University, Seoul, South Korea	syjssj95@snu.ac.kr
Sun Kim	(Principal Investigator) Interdisciplinary Program in Bioinformatics, Seoul National University, Seoul, South Korea Department of Computer Science and Engineering, Seoul National University, Seoul, South Korea	sunkim.bioinfo@snu.ac.kr

5.2 Acknowledgement

6. References

1) Akiba, Takuya, et al. “Optuna.” *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, https://doi.org/10.1145/3292500.3330701.
2) Biewald, Lukas. “Weights & Biases – Developer Tools for ML.” *Weights & Biases – Developer Tools for ML*, https://wandb.ai/, 2020.
3) Li, Yang, et al. “Prosampler: An Ultrafast and Accurate Motif Finder in Large Chip-Seq Datasets for Combinatory Motif Discovery.” *Bioinformatics*, vol. 35, no. 22, 2019, pp. 4632–4639., https://doi.org/10.1093/bioinformatics/btz290

4) Paszke, Adam, et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” *Advances in Neural Information Processing Systems,* vol. 32, 2019, pp.8024-8035.
5) Quang, Daniel, and Xiaohui Xie. “DanQ: A Hybrid Convolutional and Recurrent Deep Neural Network for Quantifying the Function of DNA Sequences.” *Nucleic Acids Research*, vol. 44, no. 11, 2016, doi:10.1093/nar/gkw226.
6) Seo, Seokjun, et al. “DeepFam: Deep Learning Based Alignment-Free Method for Protein Family Modeling and Prediction.” *Bioinformatics*, vol. 34, no. 13, 2018, pp. i254–i262., doi:10.1093/bioinformatics/bty275.
7) Zhou, Hannah, et al. “Towards a Better Understanding of Reverse-Complement Equivariance for Deep Learning Models in Regulatory Genomics.” *Proceedings of Machine Learning Research*, vol. 165, no. 1-33, 2020, https://doi.org/10.1101/2020.11.04.368803.

7. Feedback (optional)