DREAM Challenge 2022
Predicting gene expression using millions of random promoter sequences
by
Noisy Chardonnay

Abstract

We trained a CNN-BiLSTM network on a training set of nearly 5 million promoter sequences without any data augmentation to predict expression levels measured in a separate experiment for more than 70,000 promoter sequences. Our overall approach leveraged several insights from previous work on sequence to expression levels. Specifically, we used deep convolutional layers to capture distributed representations while the following bidirectional LSTM layer aims to capture positional encoding information. Our extensive experimentation over the key model parameters did not significantly improve the model performance on a randomly split validation set.

1. Description of data usage

We removed any training sequences with poor genotype calls (i.e., low-confidence genotypes) or containing indels such that the input sequence is not equal to 110-nt long. We split the remaining training set randomly into train (n=4,959,901; 90%) and validation (n=551,101; 10%) sets. We performed one-hot encoding on all the sequences in our analysis. We did not include reverse complement sequences. We standardized the expression values in our training data using StandardScaler. Scaling was applied to the entire training set (train and validation), and the final prediction values were transformed using the same scaler.

2. Description of the model

We used a simple CNN-BiLSTM architecture with three convolutional layers followed by a bidirectional LSTM layer. Prediction layer was preceded by two fully-connected layers with ReLU activation functions (**Figure 1**). Our model included 1,007,169 parameters.

3. Training procedure

We trained our neural network model on Google Colab using available GPU resources (NVIDIA Tesla P100) to predict the scaled expression levels by minimizing the loss as measured by mean squared error. We used Weights & Biases platform to register each training run to track experiments (different architectures, parameters, design choices) and later for hyperparameter tuning. Extensive experimentation regarding optimizers (Adam, SGD, learning schedule with exponential decay), kernel initialization (He normal, Glorot uniform) and kernel regularization (varying L2 penalty) did not have any impact on the training performance or predictive accuracy.

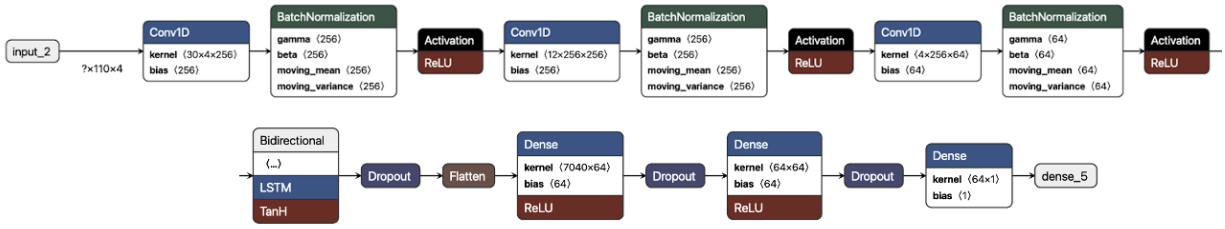

Figure 1. Schematic representation of the final network architecture using Netron app.

We performed a random grid search for a selected subset of parameters and corresponding values for hyperparameter tuning using Weights & Biases platform (**Table 1**). We selected the parameter combination with the lowest validation loss and trained our model for 200 epochs with an early stopping rule after 20 epochs of no improvement in validation R-square.Training completed after 44 epochs with validation mean squared error of 0.47 and validation R-square of 0.52. **Figure 2** shows the training and validation loss during training.

Parameter	**Value range**
Batch size	[128, 256, **512**, 1024]
Dropout rate (Dense layer)	[0.05, 0.1, **0.2**, 0.3, 0.4, 0.5]
Dropout rate (LSTM layer)	[0.05, **0.1**, 0.2, 0.3, 0.4, 0.5]
Learning rate	Uniform(0.0001, 0.01) (**0.003**)
Number of units (Dense layer)	[32, **64**, 128, 512, 1024]
Number of units (LSTM layer)	[4,8,16,**32**,64,96]

Table 1. Hyperparameter tuning. Parameters used in the final model are highlighted in **red**.

4. Other important features

We picked the final model architecture after extensive experimentation across various layers, parameters and preprocessing strategies. While there are promising attribution methods to benchmark competing models for binary prediction tasks (e.g., motif discovery), it is very challenging to relate model parameters and design choices to meaningful biological features or representations for sequence-to-expression models. Indeed, we have not been able to identify a set of design choices that consistently improved prediction accuracy (validation loss).

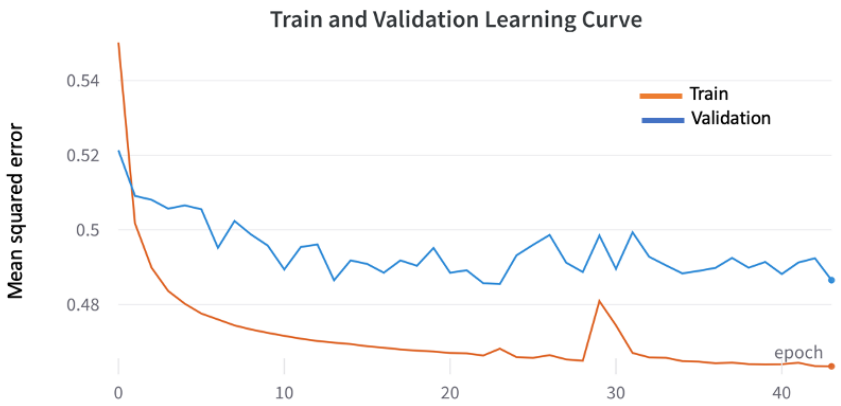

Figure 2. Mean squared error for Train and Validation loss.

Koo and Ploenzke (2021) showed that using exponential activating function in the first layer (and first layer alone) is helpful for better model interpretation. This strategy did not help with prediction accuracy on this dataset. Similar to the motivation for using exponential activation in the first layer (i.e., better discrimination btween distributed vs local representation), we tried different pooling strategies. Notably, we tested whether a popular design strategy of concatenating 'max pooling' and 'average pooling' together (**fastai's AdaptiveConcatPool2d layer**) would help. This strategy did not help the training.

It is conceivable to think that most promoters strongly favor one strand over the other. However, it is not trivial to identify which reverse complement sequences are predictive of expression levels. We considered augmenting the training sequences by stacking their reverse complement sequences (instead of, for example, using a reverse complement aware convolutional layers to process forward and reverse strand separately). Stacking strategy did not improve the predictive accuracy as measured by validation loss during training. We trained our model without any reverse complement sequences included.

We are currently experimenting with three additional strategies. First, one of the participants suggested that providing longer flanking sequences may be helpful. We would like to use attention-based models to quantify the relative importance of distal and proximal regions. Second, k-mer encoding instead of one-hot encoding has been shown to improve prediction accuracy in certain settings (**Gunasekaran et al 2021**). Given the relatively short input sequences in our dataset, it was not clear whether choice of encoding scheme would make a significant difference. We plan to compare our model's performance on k-mer encoded input sequences. Finally, we did not perform any test time augmentation. In principle, averaging the predictions for test sequences and their reverse complements may reduce prediction error, however, this requires accurate identification of subsets of promoters with informative strand-specific behavior. In a two-stage experimental setup, deep learning models with particularly strong performance for motif discovery (or related binary tasks) can be leveraged for choosing the most informative reverse complement sequences to include as part of the training set.

5. Contributions and Acknowledgements

5.1. Contributions

Name	Affiliation	Email
Onuralp Soylemez	Global Blood Therapeutics	onuralp@gmail.com

5.2. Acknowledgement

I would like to acknowledge the Google TPU Research Team for providing state-of-the-art computational resources and their generous support.

6. References

FastAI - AdaptiveConcatPool2d layer
https://forums.fast.ai/t/what-is-the-distinct-usage-of-the-adaptiveconcatpool2d-layer/7600

Hemalatha Gunasekaran, K. Ramalakshmi, A. Rex Macedo Arokiaraj, S. Deepa Kanmani, Chandran Venkatesan, C. Suresh Gnana Dhas, "Analysis of DNA Sequence Classification Using CNN and Hybrid Models", *Computational and Mathematical Methods in Medicine*, vol. 2021, Article ID 1835056, 12 pages, 2021.

Koo, P.K., Ploenzke, M. Improving representations of genomic sequence motifs in convolutional networks with exponential activations. *Nat Mach Intell* 3, 258–266 (2021).

Netron app for network visualization. https://netron.app

Weights and Biases. https://wandb.ai

7. Feedback

Thank you! I learned a lot, and I think this competition provides a strong foundation & community to develop better benchmark methods & strategies. Experimentation with TPU was quite challenging primarily due to lack of best practices around data storage and custom data generator to process large files. I am really looking forward to learning from other participants how they leveraged TPU resources.