DREAM Challenge 2022
Predicting gene expression using millions of random promoter sequences
by
Wen Group

Abstract

To predict the gene expression with millions of random promoter sequences, we built a deep neural network that adopted some latest concepts in deep learning, such as UNet, Transformer and Squeeze Excitation. We train our model from end to end without any data augmentation and model ensemble. Our model can achieve a better performance than Vaishnav et al.'s reference model in the withheld test data [1]. Considering we have not done any hyperparameter fine-tuning and only used the original training data without any additional operation such as augmentation or enhancement, we believe our model has the potential for further improvement.

1. Description of data usage

We first use *tf.keras.preprocessing.pad_sequence* function to pad the original sequence to a length equal to 112nt since the original input sequences don't have a fixed length. For the sequence longer than 112nt, we will truncate the sequence from the end; for the sequence shorter than 112nt, we will pad the sequence from the end with "N". Then, we use one-hot encoding to encode the nucleotides based on the order of "A", "C", "G", and "T". Specifically, we use *tf.keras.layer.StringLookup* function to encode the input sequences and define the vocabulary as ['A', 'C', 'G', 'T'] while characters not in the vocabulary (i.e., "N") are encoded in the fifth dimension. Therefore, "A", "C", "G", "T" and "N" are encoded to [1,0,0,0,0], [0,1,0,0,0], [0,0,1,0,0], [0,0,0,1,0] and [0,0,0,0,1], respectively. After transferring data into one-hot encoding, we randomly shuffle the original dataset using *tf.dataset.shuffle* function, where the parameter *reshuffle_each_iteration* is set to False to avoid the same data being sampled multiple times into the training and validation dataset. We randomly sample 95% of the original data as the training dataset and use the left 5% as the validation dataset to avoid overfitting. Since we are training over the TPU v2-8 VM, we set the global batch size to 1024, where each TPU core will be assigned 128 samples. To accelerate the efficiency of feeding data into the model, we prefetch the training data into the memory using *tf.data.Dataset.prefetch()* function. This operation will reduce the latency and improve the data pipeline throughput. And we set the buffer size for prefetching the data equal to *tf.data.AUTOTUNE*, so it will optimise the number of data been prefetched automatically. For data augmentation, we have tried to incorporate the reverse complement of the training sequence by one-hot encoding the original sequence with the order of "T", "G", "C", and "A". But we could not observe improvement based on the leader board's result, so we decided not to adopt this data augmentation in the final model.

2. Description of the model

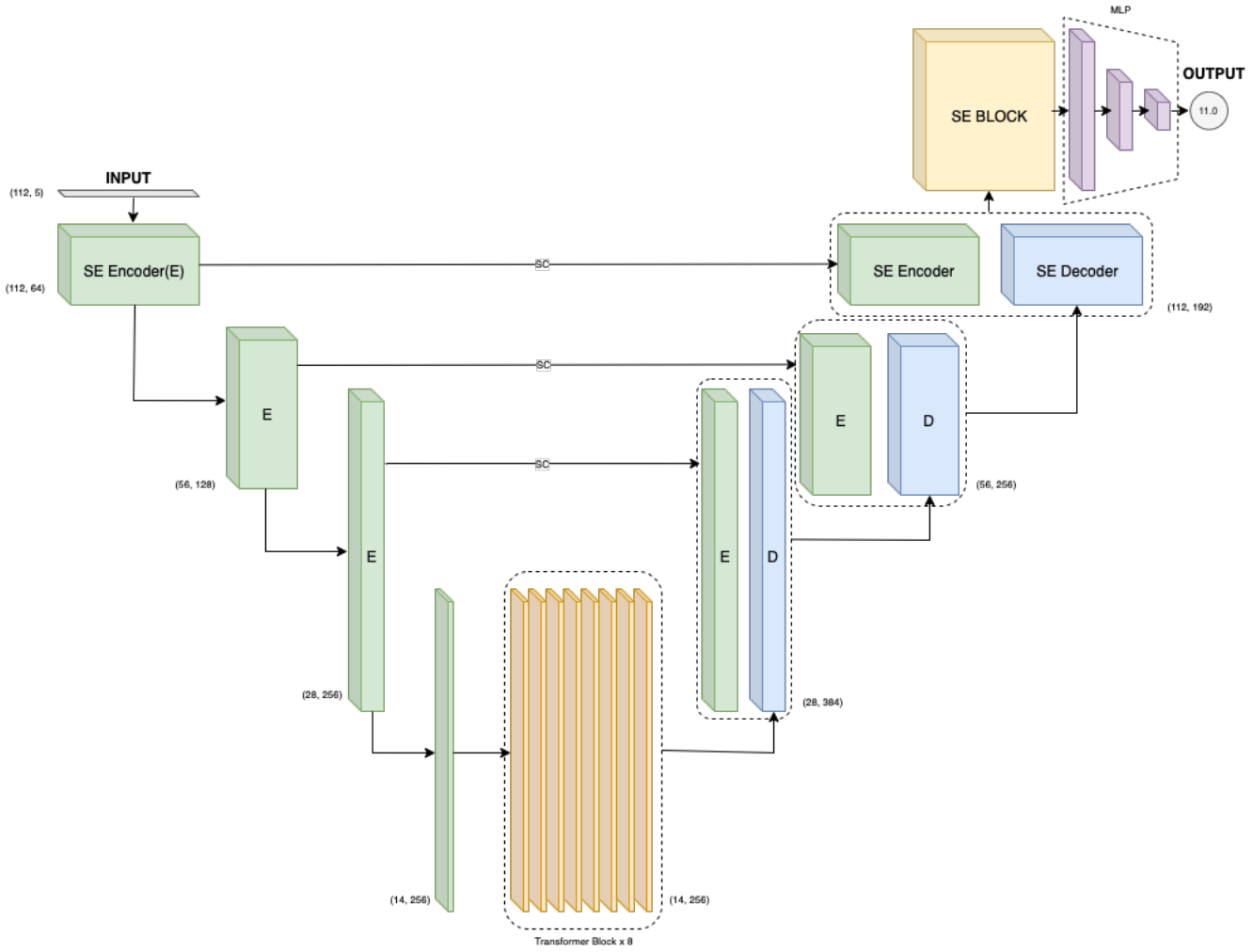

Figure 1. Model's Architecture

We train a deep neural network to predict the expression level of the promoter sequences. For the neural network model, we adopt the architecture of UNet since we believe the encode-and-decode operation will help the neural network to learn a better representation of the original data [2]. Figure 1 is our model's architecture. Our customised UNet consists of multiple encoder blocks and decoder blocks, where the encoder and decoder in different levels have skip connections between them. Inside each encoder block, we replace the 2D convolution operation with 1D convolution since our input data is genomic sequences rather than images. Also, we add a squeeze excitation layer after each 1D convolutional layer because the squeeze and excitation operation has shown to improve the overall performance of the CNN-based model by assigning importance scores to different feature maps [3]. Another modification to the original UNet encoder block is replacing the max pooling operation with a convolution operation whose stride equals two. This modification considers the differences between images and genomic sequences, where max pooling over genomic sequences may potentially lose information. The following change we add to the model uses the transformer encoder to further encode the feature maps before feeding them to the decoder [4]. We use eight transformer encoder blocks, each with eight heads. We believe the self-attention module in the transformer will help the model to focus on the critical section of the feature maps

[5]. The decoder block is very similar to the vanilla UNet decoder. The main difference is using a 1D transpose convolution operation and adding a squeeze excitation layer after every convolutional layer. Our model will learn the projection from the “annotated” promoter sequence to the related expression level in the next stage. Specifically, the output from our customised UNet is fed into a fully convolutional block where a squeeze excitation operation follows each convolutional process to highlight the critical feature maps. The final output layer is a dense neuron with a linear activation function to predict the expression score.

3. Training procedure

We use Tensorflow built-in Adam optimizer for the training procedure with default parameters whose learning rate equals 0.001. For the loss function, we use the Tensorflow built-in Huber loss, which is less sensitive to outlier compared to mean squared error loss in regression problems. To avoid our model overfitting the training data, we adopt the idea of early stopping. So, we monitor the Huber loss over validation data. And if our model’s performance over the validation data is not improved in the recent twenty epochs, we will stop the training and restore the weight that our model achieves the best performance in terms of the Huber loss. By implementing the above procedure, our end-to-end model completes the training process in thirty epochs and achieves a Huber loss of 0.767 and an r-square of 0.549 in the validation dataset (5% of the given examples). In our training dataset (95% of the given examples), our model has a Huber loss of 0.718 and r-square of 0.582 at the last epoch. Our model rank 11^{th} on the final leader board with ScorePearsonR^2: 0.742, ScoreSpearman: 0.786, PearsonR^2: 0.908 and Spearman: 0.956.

4. Other essential features

There are two essential features in our model’s implementation. The first one is adding transformer blocks in the encoder part of the UNet. And in our ablation experiment, a UNet model with transformer blocks can achieve a much better score on the leader board than vanilla fully convolutional UNet. The second essential modification is adding the squeeze excitation (SE) layer after the convolutional layer. By adopting the SE block, our model can pay “attention” to different feature maps generated by convolutional layers. Since this modification is relatively lightweight, it can improve CNN based model’s performance efficiently.

5. Contributions and Acknowledgement

5.1 Contributions

Name	Affiliation	Email
Ke Ding	Australia National University	Ke.Ding@anu.edu.au
Gunjan Dixit	Australia National University	Gunjan.Dixit@anu.edu.au
Jiayu Wen	Australia National University	Jiayu.Wen@anu.edu.au

5.2 Acknowledgement

This work is supported by Cloud TPUs from Google's TPU Research Cloud (TRC) and computational resources provided by the Australian Government through the National Computational Infrastructure (NCI) under the ANU Merit Allocation Scheme. We would like to thank de Boer Lab and DREAM Challenge committee for organising the challenge. We would like to thank Jiayu Wen and Jingbo Wang for their support and feedback.

6. References

[1] E. D. Vaishnav *et al.*, "The evolution, evolvability and engineering of gene regulatory DNA," *Nature,* vol. 603, no. 7901, pp. 455-463, 2022.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015: Springer, pp. 234-241.

[3] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132-7141.

[4] J. Chen *et al.*, "Transunet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306,* 2021.

[5] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems,* vol. 30, 2017.

7. Feedback (optional)

We enjoy participating in this challenge and appreciate the opportunity to use the TPU to accelerate our model's training process. We think it would be great if we could have more time to fine-tune our model, mainly because we are unfamiliar with google TPU and working on this challenge in our spare time.