DREAM Challenge 2022
Predicting gene expression using millions of random promoter sequences by The Dream Team

Nicholas Abad[1], Cindy Körner[2], Lars Feuerbach[1]
[1]Division of Applied Bioinformatics, German Cancer Research Center (DKFZ), Heidelberg, Germany
[2]Division of Molecular Genome Analysis, German Cancer Research Center (DKFZ), Heidelberg, Germany

***Abstract* – Decoding how gene expression is regulated is essential to understanding the origins of many diseases. Within regulatory DNA, proteins called Transcription Factors (TFs) bind to specific DNA sequences within the genome and potentially work together to produce a level of gene expression for its downstream gene. Because this process is exceedingly complex to model within humans, participants of this DREAM Challenge were given randomly millions of randomly generated promoter sequences and their gene expression in order to train a machine learning model that predicts gene expression from solely the sequence. We propose a new dataset augmentation method as a well as a neural network architecture that incorporates convolutional layers, multi-head attention layers, and long-short term memory layers. Throughout the challenge, we were able to obtain a Pearson R Correlation Coefficient of** 0.720**.**

I. Description of Data Usage

Initially, the dataset consisted of 6,739,258 random promoter sequences with their corresponding downstream gene expression. As a result, we decided to use a training set of 5,151,744 (roughly 75%), a validation set of 673,925 (1%), and the remaining 913,589 as the testing set.

Although multiple different methods for encoding were attempted such as kmer embedding, a basic one-hot encoding of the sequences was used in the final sets such that within a sequence, $A = [1,0,0,0]$, $T = [0,1,0,0]$, $C = [0,0,1,0]$ and $G = [0,0,0,1]$.

Additionally, due to the distributions of the actual given training set did not mirror the distributions of the competition testing set, we decided to augment the entire dataset. In order to do so, we first realized that a high majority of observations had integer expression values (i.e. 10, 11, 12, etc.). Rather than using these exact gene expression values, we decided to replace these integer values with values coming from a $Normal(i, 0.3)$ distribution such that i represents the integer value that it was originally given. An example of the original and augmented distribution for a small subset of the dataset can be seen in Figure 1.

As a result, the cumulative distribution functions (CDF) of the original and augmented/new sets can be seen in Figure 2, which better emulates the CDF of the testing distribution that was given to participants of the challenge.

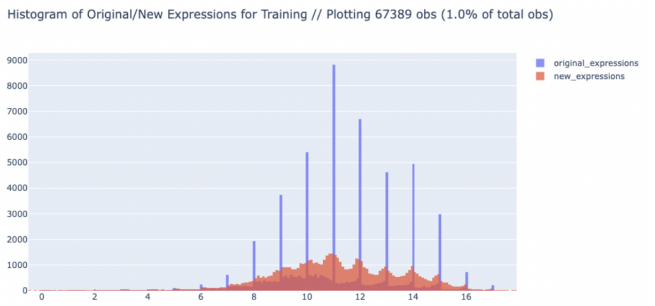

Fig. 1. Distribution of the original (blue) and augmented (red) gene expression

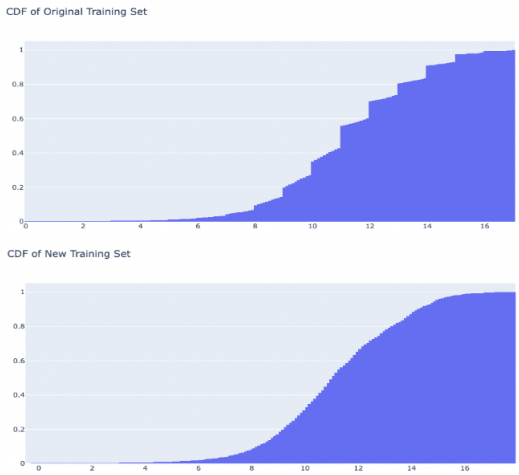

Fig. 2. Cumulative distribution functions of the old and new training sets.

II. Description of the Model

In order to learn the non-linear dependencies of the model, a deep neural network was used. As a final submission, our deep neural network was inspired by the ideas of Vashinav et al. [1] and Li et al. [2] in which we sequentially perform the following operations:

1. Split the original sequence into its forward and reverse strand, which can account for positional effects of the sequence on the gene expression
2. Perform multiple convolutions on each of the strands, which can be analogous to scanning both of the strands for a single or multiple TFBSs.
3. Utilize a multi-head attention layer and LSTM layer in both strands in order to learn the most important features of the output of the convolutional layers.

4. Concatenate the output of these two strands.
5. Perform additional convolutions on the concatenated output in order to learn joint interactions
6. Feed this output through multiple dense layers, which is a necessity in order to obtain the final gene expression.

The specifics of this architecture can be seen in Figure 3.

III. Training Procedure

For the training procedure, we used the optimizer Nadam, mean squared error as the loss function, and all layers fully trainable. Additionally, we used the following hyperparameters:

- Number of Epochs: 30
- Batch Size: 1024
- Learning Rate: 0.001
- Dropout Rate: 0.1
- Number of Multi-Head Attention Heads: 8
- Number of Attention Layers: 2
- Number of Convolutional Layers per block: 2
- Number of Dense Layers per block: 2

IV. Contributions and Acknowledgements

Authors:

- **Nicholas Abad:** Division of Applied Bioinformatics, German Cancer Research Center (DKFZ), Heidelberg, Germany
- **Cindy Körner:** Division of Molecular Genome Analysis, German Cancer Research Center (DKFZ), Heidelberg, Germany
- **Lars Feuerbach:** Division of Applied Bioinformatics, German Cancer Research Center (DKFZ), Heidelberg, Germany

REFERENCES

[1] E. D. Vaishnav, C. G. de Boer, J. Molinet, M. Yassour, L. Fan, X. Adiconis, D. A. Thompson, J. Z. Levin, F. A. Cubillos, A. Regev, "The evolution, evolvability and engineering of gene regulatory DNA", *Nature*, vol. 603, no. 7901, pp. 455–463, 2022.

[2] J. Li, Y. Pu, J. Tang, Q. Zou, F. Guo, "DeepATT: a hybrid category attention neural network for identifying functional effects of DNA sequences", *Briefings in Bioinformatics*, vol. 22, no. 3, p. bbaa159, May 2021, doi:10.1093/bib/bbaa159.

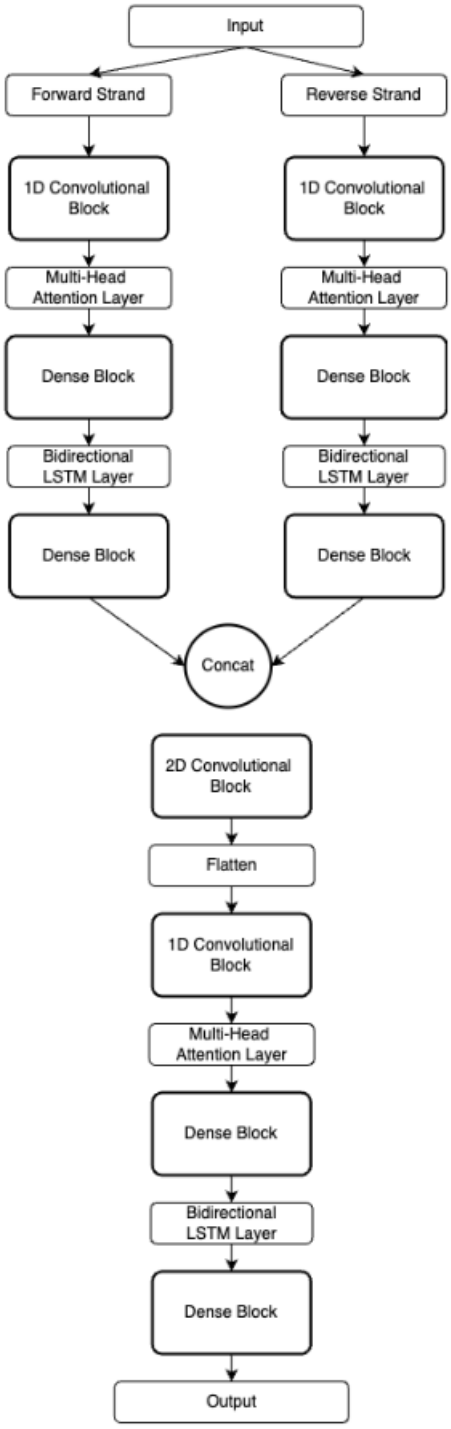

Fig. 3. Overview of our neural network architecture.