Deep Learning in Genomics Final Project Report

By WakaWaka: Isaac Nathoo and Haider Asif

Introduction:

Using histone mark data and DNA sequence data, our goal was to develop a deep learning model that could predict gene expression in a total of 56 cell types across 17447 genes. The training set contained the histone marks and expression values for 50 cell types and 16000 genes. We also had a 10kb DNA sequence, ± 5kb from the transcription start site, for each gene.

Our final model included a histone modification (HM) module, DNA sequence module, and combined module three. The HM module included two 1D convolutional layers and a fully connected layer. The DNA module included one 1D convolutional layer and a fully connected layer. The combined module included a concatenation layer to combine the outputs of the HM and DNA modules and two fully connected layers. When we trained our model on the entire dataset, we achieved a final training loss (mean squared error) of 2.13606 and a Pearson correlation coefficient of 0.66539. Furthermore, we ran k-fold cross-validation for 4 folds using our model and found that the validation loss generally decreased across the epochs. Our final testing mean squared error on the evaluation set with data for 6 held-out cell types across all genes and 1447 held-out genes was 3.15962. Our final testing was fairly large compared to the final training loss, which indicates that our model was overfitting the training data despite the various regularization methods we attempted.

Method:

Our final model architecture was a Keras function model with three modules, a histone modification module, a DNA sequence module, and a combined module (Figure 1), defined as follows:

Histone Modification Module

- Conv1D: 50 filters of width 10, LeakyReLU with alpha=0.05
- MaxPooling1D with pool size of 5
- Conv1D: 50 filters of width 5, LeakyReLU with alpha=0.05
- MaxPooling1D with pool size of 3, Flatten then Dropout with rate=0.5
- Dense layer with 82 units, LeakyReLU with alpha=0.05

DNA Sequence Module

- Conv1D: 132 filters of width 6, ReLU activation
- MaxPooling1D with pool size of 32, Flatten then Dropout 0.5
- Dense layer with 82 units, LeakyReLU with alpha=0.05

Combined Models

- Concatenation of Histone and DNA Module Final Layers
- Dense layer with 100 units, LeakyReLU with alpha=0.05
- Dense layer with 1 unit, no activation

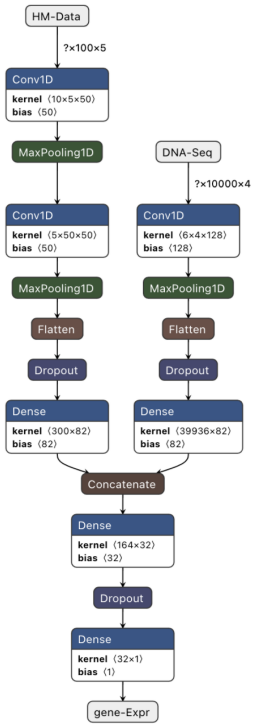

Figure 1: WakaWaka final model architecture.

The loss function that we used was Mean Squared Error because this was a regression task where we wanted to predict continuous values for gene expression. We also used the Adam optimizer with a learning rate of 0.0005 and a batch size of 100, and trained the model for 10 epochs.

We decided to use separate modules for the histone modification data and DNA sequence data based on the DeepCpG model and the ability to use what the model learns about each of these types of data to predict gene expression. For the histone modification module, we used two 1D convolution layers through biological reasoning about the nature of histone modifications and based on our experimental results. The 1D convolution operation allowed the model to look at all 5 histone marks across a certain number of bins and try to find patterns in the combination of

histone marks. We used multiple convolutional layers so we could extract both local features from the data in earlier convolutions as well as more global features in later convolutions. For the DNA sequence module, we used one 1D convolution layer so the model could look at the one-hot encoded representation across a fixed window length and try to learn important motifs for predicting gene expression. At the end of both of these modules, half of the values in the output of the convolutions were dropped out to help prevent overfitting and then fed through a fully connected layer for the model to learn how to combine these different weights into lower dimensional representation of the genomic signal, either histone marks or DNA sequences.

Next, we concatenated the outputs of the dense layers in the histone modification module and DNA sequence module and fed them through two dense layers. Between these two dense layers, we dropped out 10% of the weights to reduce any final overfitting. The number of filters for each layer and the width of these filters and the size of the dense layers was determined through k-fold cross validation and model experimentation/hyperparameter tuning.

For the interpretation aspect of our model, we used two different approaches. For the histone modification module, we plotted a heat map showing the number of reads in each of the 100 bins across all 5 histone marks to visualize the input. Then, to see what our model was learning, we calculated the gradient of the output of the model with respect to the HM inputs using Tensorflow's gradient tape and multiplied this gradient by the input, and then plotted the corresponding heat map. The regions with the highest intensity would show us which features of the input and combination of HM marks were the most important for predicting gene expression. For the DNA sequence module, we calculated the gradient of the output of the model with respect to the DNA sequence input and then plotted gradients as a line graph across the length of the sequence for each of the 4 base pairs. This would show us which regions of the DNA sequence had the highest gradient and thus were the most important for the regression task.

Experiment Setup and Model Development:

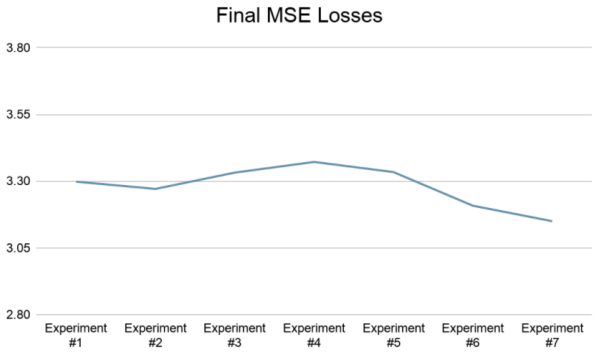

Figure 2: The final testing MSE loss across 7 main experiments.

Before we arrived at our final model, we went through an iterative process of developing and tuning our model. We started with a more complex model than we had last time, this time making our way from a more complex architecture to a less complex one in order to make better predictions (Figure 2). Our first model had two modules like our final model, one for the histone modifications data and one for the DNA sequence data which in this case was the entire 10 kbp sequence. Our histone module had 3 Conv1D layers with the latter 2 being dilated convolution layers followed by leaky ReLU activation and our sequence model had 1 Conv1D layer followed by ReLU activation. The outputs from both these layers were concatenated before dropping out 50% of the weights after which we passed the output through three Dense layers. The first two were of size 50 and 10 with leaky ReLU activation and the third was of size 1 with no activation. This model (Experiment #1) achieved a testing accuracy of 3.298, which was an improvement on our results from the midterm. This sparked our experimentation process in which we conducted a grid search for the sizes of the dense layer(s), the filter size, kernel size, and the dilation rates for the convolution layer(s), and the pool size of the max-pool layer(s). We also searched for the most optimal learning rate, activation function, dropout, padding type, batch size, along with trying out regularization methods like batch normalization. We also tried

using different sections of the DNA sequence, especially taking sections ± a certain length from the TSS.

From our initial experiment it was evident to us that our model was not learning very well, so the goal of our experimental approach was to improve its performance (Figure 2), while making sure that it does not overfit by using k-fold cross validation. To accomplish this we tried adding another Conv1D layer to the DNA module (Experiment #2), based on our hypothesis that increasing the number of convolutional layers would make the network more complex and enable it to capture the more complex/global patterns in the data. This turned out to be the correct hypothesis as our results showed we had a lower testing loss of 3.271. Next, we wanted to try using only part of the DNA sequence data, ±2kbp from the TSS, to see if we can capture more valuable relationships by using a smaller subset of the data. This ended up increasing our testing loss to 3.33 (Experiment #3), which led us to believe that there are important regions of the DNA sequence both up- and down-stream from the TSS, so we went back to the full 10kbp sequence.

In an effort to do even better, we hypothesized that removing dilation and a convolution layer, and adding a Dense layer of size 100 to the joint module would make the model do better, but as it turned out, that made the model overfit even more, giving us a testing accuracy of 3.372 (Experiment #4). After that, in an effort to learn better relationships but not overfit the model so much, we tried a pre-training approach. In this we pre-trained the histone modifications and sequencing modules and then passed their flattened and 50% dropped output to the joint module which was not trained in the first round. After this, we trained the model again, this time not training the HM module and the DNA modules but only training the joint module. This model took almost 10 hours to run for us, and even after that overfit by a lot giving us a testing accuracy of 3.334 (Experiment #5). After learning from our initial approaches, we decided to make the model less complex, so we kept only 2 Conv1D layers in the HM module, and only 1 Conv1D layer in the DNA module, each followed by a 0.5 dropout, along with a joint module which now only contained 2 Dense layers, one of size 100, and the other of size 1 to output expression values. This improved our testing accuracy by fair amount, and brought it down to 3.207 (Experiment #6). Then, we hypothesized learning a lower dimensionality representation from each output might result in less overfitting and give us better results. This model was our final model, and the only difference was that both the HM module and the sequence module included dropout (instead of the joint module) and 1 Dense layer before concatenation was performed. This gave us our best test accuracy of 3.15 (Experiment #8) on the test dataset.

Results:

After experimentation, the final structure of the model consisted of 3 modules, a histone modifications module with 2 Conv1D layers, a Dense layer and a dropout layer, a sequence module with 1 Conv1D layer, a Dense layer and a dropout layer, and a joint module that took the concatenated output of the two modules and passed it through two Dense layers of size 100,1 to return the expression values. We used the Adam optimizer with a learning rate of 0.0005 and a batch size of 100, and trained for 10 epochs. When we ran k-fold cross validation for 4 folds with a 75/25 split in the training data (Figure 3), we found that the training loss was constantly decreasing and the validation loss was also generally decreasing and relatively smooth. The validation

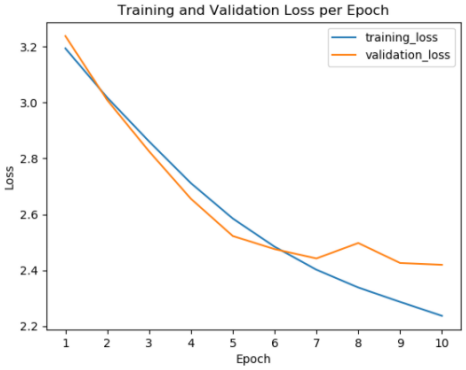

Figure 3: The training and average validation losses across 10 epochs using k-fold cross validation with 4 folds.

loss shows some patterns corresponding to overfitting as it bounces up and down. This suggests that our convolutional layers may have been learning features that are slightly too specific to the training data. We decreased the size of our model to have much less parameters than what we began with to try to prevent overfitting, but this was the best result we could obtain. Training this model on 100% of the training data, histone marks and the 10kbp gene sequence resulted in a training MSE loss of 1.55 and on Kaggle, we achieved a final testing mean squared error (MSE) of 3.15. Throughout the course of training, the model's loss was consistently decreasing across the epochs (Figure 4). We also chose to cut off the training at 10 epochs because of computational and time constraints, on average a single epoch took almost 30 minutes to train on GCP, and along with running the validation, the total time we had to run the model for was significantly high. The Pearson correlation coefficient was 0.6654 between the predicted expression values from the training data and the true labels.

When interpreting our model to determine whether it was learning biologically relevant features, we found that the features with the highest gradient corresponded to some underlying biology of the system. For the histone modifications, the input data includes the highest number of read counts in the center of the gene body around the TSS as shown in Figure 4A. The map in Figure 4B calculated by multiplying the input by the gradient of the output with respect to the HM input shows that this central region has the largest absolute value of gradients and that this region is the most important for predicting gene expression. The highest density is in column 3, which is an enhancer mark (H3K4me1), and column 4, which is a promoter mark (H3K4me1). For the DNA sequences, the gradients of the output with respect to the sequence input were calculated and plotted for each of the four nucleotides across the 10kb sequence as shown in Figure 5. This analysis showed specific regions across this sequence that are particularly important for predicting gene expression, which is corroborated by the underlying biology of gene regulation and the presence of regions such as promoters and enhancers. For this specific gene, there are several regions between bases 0 and 2500 with large gradients which signify that they are important and biologically these may be distal promoters or nearby enhancers. There is also a lot of activity close to the transcription start site around base 5000, which is good to see because this is where most transcription factors probably bind so this region should be important for gene expression. Interestingly, our results also show regions at the beginning of the gene body which are important for prediction.

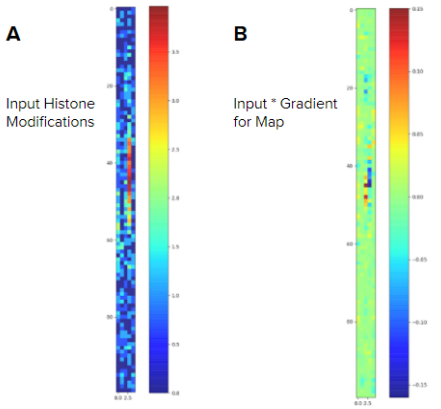

*Figure 4: (A) Heatmap of the 5 histone marks across the all the bins for an example gene (B) gradients*input for each of the 5 histone marks across the all the bins*

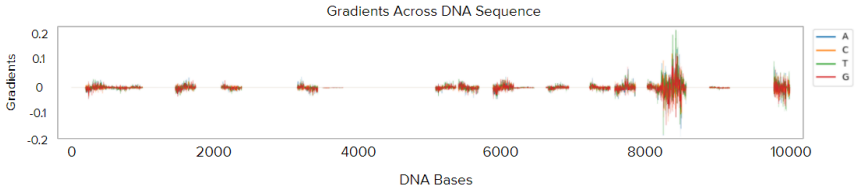

Figure 5: The gradients for each of the four nucleotides plotted across the DNA sequence for an example gene.

Some of the innovation in this work was the combination of histone modification data and DNA sequence data in a joint module and the attempt at developing a model that would pre-train the HM and DNA modules then later retrain the combined module. We also applied the idea of saliency maps and gradients to interpret what our model was learning, which showed biologically relevant results. We also demonstrated that decreasing the complexity of the model could improve performance likely because it had less parameters and thus overfit less.

Citations:

1. Singh, R., Lanchantin, J., Robins, G., & Qi, Y. (2016). DeepChrome: deep-learning for predicting gene expression from histone modifications. Bioinformatics, 32(17), i639-i648. [1]
2. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958. [2]
3. Angermueller, C., Lee, H. J., Reik, W., & Stegle, O. (2017). DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning. Genome biology, 18(1), 1-13. [3]