Predicting Gene Expression from Histone Modifications and Gene Sequences

Team Jamaica
Sean Hastings, Stefan Stanojevic

***Abstract*— We incorporated cell type statistics and genomic sequence information via kmer counts into our model by adding them as additional inputs. After some experimentation with RNNs and convolutional autoencoders, we settled on our model from the midterm project, which eventually resulted in a mean squared error of around 2.90.**

I. Introduction

Understanding which factors contribute to gene expression is one of the most pressing questions in genetics. While understanding the mechanisms that underlie cell differentiation is a fascinating scientific question in its own right, it could also open the door to numerous clinical applications.

One of the factors relevant for gene regulation are modifications to the histone proteins. The problem of predicting gene expression from histone modifications has been attacked by a variety of traditional machine learning methods, such as linear regression, random forests, support vector machines and Bayesian networks [3]–[5]. The first application of deep learning to this task was the DeepChrome model [2], which successfully applied a CNN architecture to the task of classifying genes into weakly and strongly expressed.

In our previous report, we built on this work to perform a regression analysis predicting the amount of gene expression. We used a fully convolutional neural network familiar from the field of computer vision.

Here, we offer an extension of this work that incorporates both the genetic sequence and histone modifications in order to predict the amount of gene expression.

II. Method

Our best performing model ended up being very similar to the model we used for histone-only predictions. It consisted of an ensemble of fully convolutional sub-models. The sub-model architecture is very simple, consisting of the following:

- 9 one-dimensional convolutional layers with filter size 3 and a varying number of filters
- 2 convolutional layers with filter size 1, effectively serving as a dense layer

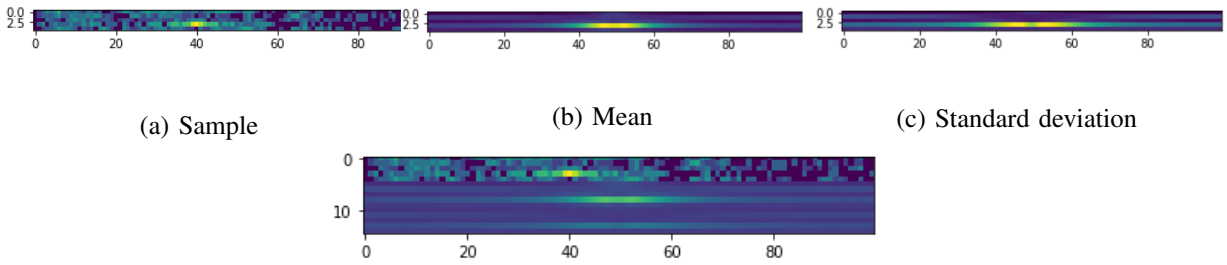
Each of the hidden layers is followed by batch normalization, dropout, and leaky ReLU in that order. Every other layer (layers 1, 3, 5, and 7) has a stride of 2 and doubles the number of channels, up to 512.

Our model was heavily inspired by the vision module from [1]. Despite the model being quite deep, the vanishing gradients problem is taken care of by the use of batch normalization and leaky ReLU activations. Dropouts are the simplest and least computationally intensive method of regularization, yet proved after testing to be very effective in our model. While max pools are commonly used to reduce the number of trainable parameters, we have opted for strided convolutions instead due to their ability to learn something akin to custom pooling kernels.

III. Experimental Setup

Like in the previous project, we were given gene expression data over 50 cell types and 16000 genes, for a total of 800 000 data points. For each input, we were given histone modification data for 5 different histones, over 100 bins spanning the region of interest, as well as the genetic sequence.

In order to improve generalization to unseen cell types, we came up with the following way to embed cell type information in the inputs to our neural network. For each cell type, we calculated mean and standard deviation of histone bin values over all data points belonging to that particular



(a) Sample (b) Mean (c) Standard deviation

(d) Sample combined input for the machine learning model

Fig. 1: Preprocessing of sample histone modification entry. The figures show: (a) histone sample, (b) mean and (c) standard deviation of histone modifications of a given cell type, and (d) sample combined input obtained by vertically stacking (a), (b) and (c)

cell. Then, we attached this information to all histone data, in a manner shown in Fig. 1. This way, our model was able to use the cell type information to make predictions in a way that easily generalizes to new cell types.

We encoded the genetic data by counting the number of appearances of 3-letter kmers across the gene sequences. Besides base nucleotides A,C,T,G, our data also contained entries N, which serve as a stand in for any of the 4 nucleotides, leading to 125 possible 3-letter motifs. Out of them, we picked only the kmers that appeared more than 10 000 times total across the dataset, which came out to 64 kmers, and counted the number of their appearances among 100 bins across the genetic sequence. This resulted in a 64 x 100 matrix of genetic data.

For each example, we thus had histone modification data in a form of 15 x 100 matrix, and genetic sequence data in a form of 64 x 100 matrix. Those were concatenated and fed into our model for a full input shape of 79 x 100.

Like in the case of histone-only predictions, we obtained the best results from an ensemble of 3 models, each trained on overlapping two thirds of our training dataset, and validated on the remaining one third, in a setup similar to the one used for 3-fold cross-validation.

We have also tried applying an autoencoder to the task of compressing the genetic information, rather than using counts of common kmers, and experimented with an architecture containing both convolutional filters and recurrent units. Those experiments failed to yield better results in reasonable time, mostly due to the fact that they were too computationally intensive.

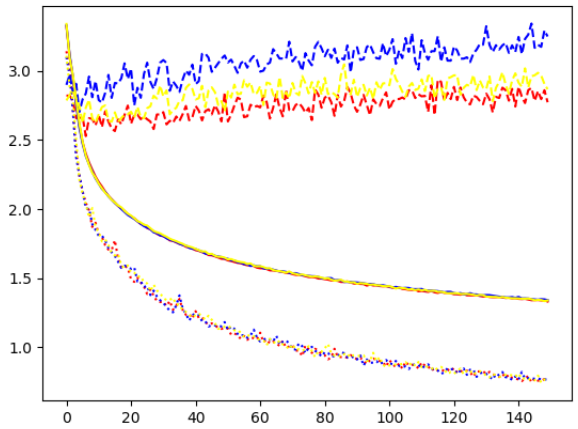
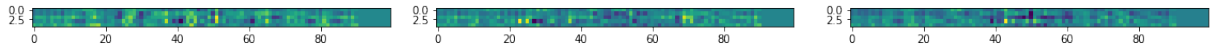


Fig. 2: Loss curves for our best model. The three colors represent the three sub-models of the ensemble while the solid, dashed, and dotted lines represent the train, eval, and "trainval" (evaluation over training set) losses

IV. Results

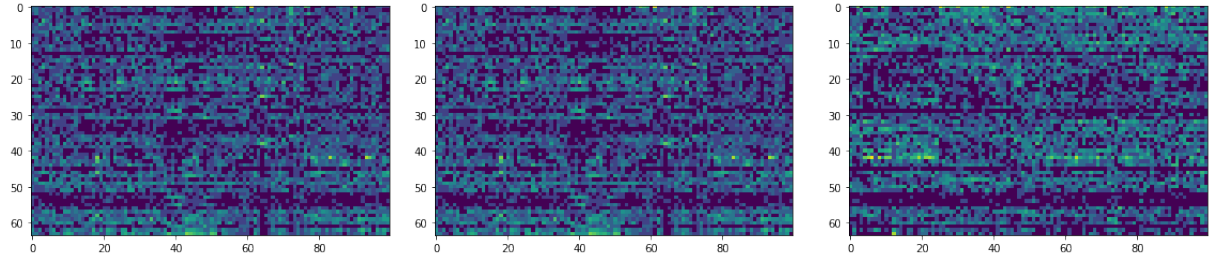
Our best model obtained a Kaggle mean squared error of slightly under 2.90.

Fig. 3 shows the gradients of the output of our neural network versus its input for several representative cell types, and may provide some insight into which bins are contributing positively or negatively to the gene expression, and which bins are not contributing that much. Note that this is not quite a *saliency map*, since we are not taking the absolute values of the gradients. By looking at



(a) E004 (b) E005 (c) E053

Fig. 3: Heatmaps of the average gradient of the output of our neural network with respect to histone modifications over 3 representative cell types



(a) 3 (b) 5 (c) 457

Fig. 4: Heatmaps of the average gradient of the output of our neural network with respect to motiff data for 3 representative genes, labeled by their indices in the Kaggle dataset

the absolute values of the gradients, we haven't been able to extract that much useful information, but a further analysis might be able to identify promoter and repressor regions by an analysis reminiscent of that done in [2]. Average gradients of model outputs with respect to kmer information corresponding to several different genes are shown in Fig. 4. We are including those figures for completeness, though we were unable to discern much useful information from them. Another common interpretation method used for neural network classification tasks looks at input configurations that maximize output probabilities. This turned out to be inapplicable for our regression task, since model outputs are unbounded from above and below, and using gradient ascent/descent to search for inputs that would maximize or minimize gene expression naturally lead to solutions that blow up.

Overall, we have had a chance to implement and compare several machine learning methods and neural network architectures, and have found a simple CNN architecture to be best suited for the task, achieving a mean squared error of 2.86 on the held-out kaggle test set. We believe that most of these gains from our midterm model (MSE 3.26) are due to our cell-type statistics, with the sequence information providing some assitance as well.

REFERENCES

[1] Codevilla, F., Miiller, M., López, A., Koltun, V., & Dosovitskiy, A. (2018, May). End-to-end driving via conditional imitation learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1-9). IEEE.

[2] Singh, Ritambhara, et al. (2016) DeepChrome: Deep-Learning for Predicting Gene Expression from Histone Modifications. Bioinformatics, vol. 32, no. 17, Jan. 2016, pp. i639–i648., doi:10.1093/bioinformatics/btw427.

[3] Cheng C. et al. (2011) A statistical framework for modeling gene expression using chromatin features and application to modENCODE datasets. Genome Biol., 12, R15.

[4] Karlić R. et al. (2010) Histone modification levels are predictive for gene expression. Proc. Natl. Acad. Sci. U. S. A., 107, 2926–2931.

[5] Dong X. et al. (2012) Modeling gene expression using chromatin features in various cellular contexts. Genome Biol.,13,R53.

[6] Our midterm report. Since we ended up using the same model, there is significant overlap between the two papers.