Predicting Gene Expression from Histone Modifications

Team Jamaica
Sean Hastings, Stefan Stanojevic

Abstract— **Predicting gene expression from histone modifications is one of the open questions in the field of genomics. We attacked this question through an application of deep learning. An ensemble of convolutional neural networks familiar from the field of robotics has produced good results, with a mean squared error of 3.26 on a held-out test set.**

I. Introduction

The rich and rapidly growing field of epigenetics studies the factors governing gene expression. It has been suggested that one such factor has to do with modifications of the histone proteins. This is perhaps not surprising, considering that histone is one of the main building blocks of chromatin, and that chromatin structures play a leading role in many cellular processes including transcription.

This problem has previously been attacked by a variety of statistics and machine learning methods, including linear regression [4], Support Vector Machines and Bayesian networks [3], and Random Forests [5].

Most notably, convolutional neural networks (CNNs) have previously been applied to study this problem [2] with excellent results. A major advantage of neural networks over the traditional machine learning methods is their ability to do automatic feature extraction, and capture regularities at several orders of abstraction.

While DeepChrome performed a classification task over weakly versus strongly expressed genes, our project performed a regression analysis over a normalized expression value. We have done so using a slightly different CNN architecture than those we have seen in the literature.

II. Method

Our best performing model consisted of an ensemble of fully convolutional sub-models. The sub-model architecture is very simple, consisting of the following:

- 8 one-dimensional convolutional layers with filter size of 3 and a varying number of filters
- a convolutional layer with a filter size of 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 a 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 poolings 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.

We ended up obtaining the best results from an ensemble of 3 models, each one of which was trained on two thirds of the training set selected by dividing the data by cell types and taking the inverse of one of three folds. While this kind of setup resembles the one used for 3-fold cross validation, we opt to use the data in each fold as the validation set and the data outside the fold as the training set, leaving each pair of trained models with 50% unique and 50% shared training data.

III. Experimental Setup

For the purposes of counting histone modifications, the gene region of interest was divided into 100 bins, for which the amount of modifications was recorded for 5 different histones. We were provided with data for 56 different cell types, for a total of 800 000 data points.

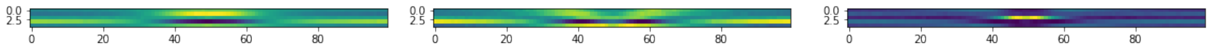


Fig. 1: Heatmaps of input distributions for 5 different histone modifications and 3 representative cell types

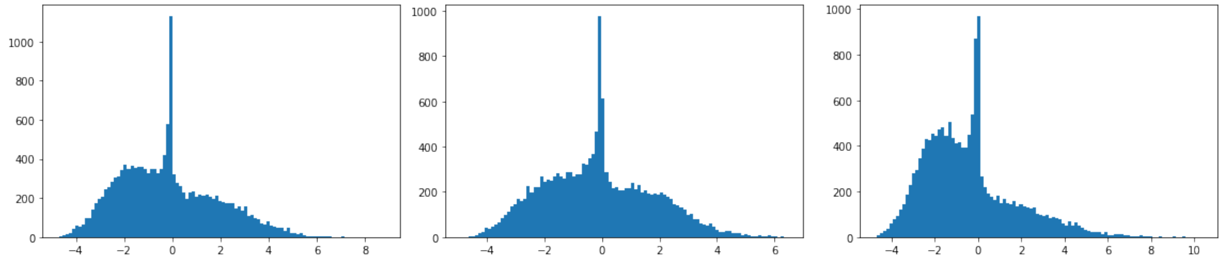


Fig. 2: Histograms of gene expression values for three representative cell types

Distribution of our input values for three representative cell types is shown in Fig. 1, while histograms of gene expression values are shown in Fig. 2. One may note that both the input and output data looks significantly different for different cell types.

The three models in our ensemble were trained for 1000 epochs each, taking around 2 days in total to train on a machine with a single GPU. Training and validation losses were tracked throughout the training, but because our validation set was over different cell types than the train set we found it useless for tracking overfitting and did not stop training preemptively despite high evaluation losses.

We also developed a model to classify the cell type of data by changing the last layer in the sub-model neural network to have an output size of 50 and a softmax activation. The training set consisted of all data, leading the classifier to predict "probabilities" that could be used as weights for sub-models trained on individual cell types. Putting these two parts together results in a learned-weighting ensemble of cell-type specific sub-models. Unfortunately, this turned out to be not as effective as the 3-model ensemble, and resulted in a test MSE of 3.47.

Linear regression analysis resulted in an MSE of around 4.70 on the testing set, and served as a useful benchmark for other tests. In order to see whether neural networks are really the optimal solution for this task, we've also experimented with feeding the training set to a random forest regression model from Scipy's library. However, we were unable to get a much better accuracy than linear regression this way, even after computing the r - values and training the random forest only on the bins that are highly correlated with the outputs, inspired by [5].

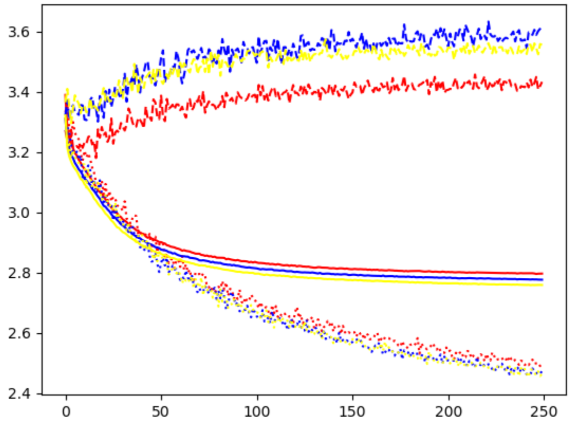
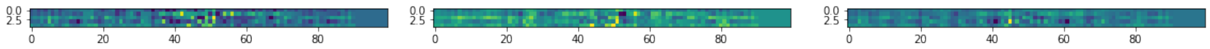


Fig. 3: Loss curves for a fewer-epochs replication of our best model due to the loss of the original loss curves. The three colors represent the three sub-models of the ensemble while the solid, dashed, and dotted lines represent the train, eval, and "trainval" (evaluation over training set) losses. Note how even when the training loss largely appears to have converged by 250 epochs the trainval loss continues to drop significantly.



(a) E012 (b) E028 (c) E065

Fig. 4: Heatmaps of the average gradient of the output of our neural network with respect to its input over 3 representative cell types

IV. Results

Fig. 4 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 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].

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 3.26 on the test set. It is interesting that a neural network trained on data from a particular set of cells can learn to decently predict gene expression in cells never seen before. It is also curious that our slightly more sophisticated model employing a cell-type classifier and an ensemble of models trained on single cell types did not perform as well, despite what one might expect a-priori.

References

[1] Codevilla, F., Miiller, M., Lpez, 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. i639i648., 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] Karli R. et al. (2010) Histone modification levels are predictive for gene expression. Proc. Natl. Acad. Sci. U. S. A., 107, 29262931.

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