Imputing chromatin landscape from a single assay

Quinton Wessells **Rafael Rafailov** **Elena Tuzhilina** **Ulugbek Baymuradov**

Stanford University

Abstract

Chromatin landscapes provide critical insight into the transcriptional regulation of the genome. Current approaches for profiling chromatin landscape require multiple high-throughput sequencing assays, creating the desire for a single cost-effective assay. Here we assess the ability of nascent transcription assay - Global Run-On and sequencing (GRO-seq) and Precision Run-On and sequencing (PRO-seq) -- to impute H3K4me3, H3K27ac, H3K27me3, and DNase-seq using XGBoost, Dense Neural Network, and Convolutional Neural Network models.

Introduction

Chromatin landscapes provide critical insight into the transcriptional regulation of the genome. Transcriptional regulatory elements, such as promoters, enhancers, and insulators can alter gene expression by promoting or inhibiting chromatin compaction, transcription initiation, and RNA polymerase binding. Epigenetic changes such as DNA methylation and histone modifications also contribute significantly to transcriptional regulation.

Current methods to characterize chromatin landscapes involve multiple high-throughput genomic assays. Chromatin immunoprecipitation and sequencing (ChIP-seq) assays of DNA-bound transcription factors (TF) and histone modifications is a common approach, but is limited in that a separate assay is required for each target since the approach relies on a high-affinity antibody for the targeted TF or histone modification. ChIP assays are commonly accompanied by DNase-I hypersensitivity and sequencing (DNase-seq), which measures open chromatin regions, to characterize chromatin landscapes. However, performing multiple assays on a sample is both time intensive and expensive; therefore, the ability to profile chromatin state from a single, cost-effective assay would be extremely valuable.

Methods for measuring the production of nascent RNA provide direct evidence of local transcription. Two such methods are Global Run-On and sequencing (GRO-seq) and its successor, Precision Run-On and sequencing (PRO-seq). These assays can map RNA polymerase II active sites by isolating and sequencing RNA that is being actively transcribed. Benefit of capturing active transcription is that it can capture expression that may be too short

lived to show up in steady state expression assays. Since chromatin landscapes are highly associated with local transcription and PRO-seq directly measures local transcription, we hypothesize that multiple chromatin landscape profiles can be imputed from a single PRO-seq assay.

In this paper we introduce deep learning methods for imputing histone modification (H3K4me3, H3K27ac, H3K27me3) and DNase-seq profiles from PRO-seq data. We benchmark this approach against XGBoost model.

Related Work

There has been previous research both into computational methods for imputing chromatin landscape from other sequencing assays and from GRO/PRO-seq. However, the methods described in this paper describe the first known research into deep learning for imputing chromatin regulatory profiles from GRO/PRO-seq data.

The most relevant deep learning approaches for predicting sequential regulatory activity were published by Kelley et al. in 2016 and 2018. In 2016, the authors published the Basset model to predict DNase 1 hypersensitivity from a window of 500-1000 base pairs. In 2018 the authors improved on Basset by developing the Basenji model to predict cell type-specific epigenetic and transcriptomic profiles from DNA sequence.The Basenji architecture uses dilated convolutions to incorporate a significantly larger base pair receptive field, 131 kilobases (kb), which allows for modeling of distal regulatory interactions. Specifically the architecture consists of 4 standard convolution layers, pooling in between layers by 2, 4, 4, and 4 to a multiplicative total of 128, 7 dilated convolution layers, and a final convolution layer. They tested the model on a dataset consisting of reads for 593 ENCODE DNase-seq, 356 Roadmap DNase-seq, 1704 ENCODE histone modification ChIP-seq, 603 Roadmap ChIP-seq, and 973 FANTOM5 CAGE experiments. Compared to Basset, the Basenji quantitative predictions achieved a greater AUPRC for all DNase-seq experiments, increasing the average from 0.435 to 0.577 and median from 0.449 to 0.591.

Danko et al. are have previously published work on identifying regulatory elements from nascent transcription with the machine learning tool, discriminative Regulatory Element detection from GRO-seq (dREG). The dREG model was first published in 2015 and employs support vector regression (SVR) to detect predict transcriptional regulatory elements from GRO/PRO-seq data. The authors have published a new, optimized implementation of dREG on bioRxiv. They trained the model using 3.3 million sites from five independent PRO-seq or GRO-seq experiments in K562 cells and evaluated the performance on two held out datasets, a sixth PRO-seq experiment in K562 and a GRO-seq experiment in GM12878. dREG predicted 34,677 and 71,131 TIRS in the K562 and GM12878 test datasets, respectively. The authors compared these predicted TIRS to DNase-seq and H3K27ac ChIP-seq data as orthogonal validation of TIR recovery, and found overlap with 81.3% or 96.1% of the DNase-seq and 58.4% or 71.8% of the H3K27ac sites.

Data and Preprocessing

Data for training the model comes from the multiple genomic assays done on human biosample GM12878. CHip-seq assays for histone modifications H3K4me3, H3K27ac, and H3K27me3, and DNase-seq come from the ENCODE Project. GRO-seq data for the same biosample comes from the Dowell Lab and BioFrontiers at the University of Colorado.

All the data was binned by 50 base pairs (bp) using the pyBigWig stats function, which takes the average value over the 50bp range. In keeping with the dREG model that made predictions based on a 5 kb window around each base, we created a dataset consisting of window of 200 bins (200 bins * 50bp = 5000bp) around each bin. We also created a dataset using a 128 kb (2560 bin) window around each bin, as this was around the window size used in the Basenji architecture. The output label was a single 50bp bin from the fold-change-over-control bigWig file in all four prediction tasks (H3K4me3, H3K27ac, and H3K27me3, and DNase-seq). We explored different levels of smoothing for the binned outputs using the baseline XGBoost model, and found that gaussian filtering with a sigma around 10 yields the best results.

Baseline Model

Our baseline model was XGBoost gradient boosting. We ran XGBoost separately on each of the four predictive tasks and for both linear regression and binary classification. In linear regression, we used root mean squared error as the evaluation metric. In the classification task, we set the objective to binary classification using hinge loss with area under the precision-recall curve as the evaluation metric. The models were trained on the first quarter of chromosome 1 and tested using the second quarter of chromosome 1 using a window size of 5000bp. Results are shown in Table 1 and Figure 1.

Table 1: XGBoost evaluation

Model	H3K4me3	H3K27ac	H3K27me3	DNase
Binary Classification				
Recall	0.340	0.221	0.000	0.001
Precision	0.752	0.735	0.000	0.531
F measure	0.470	0.328	0.000	0.002
Accuracy	0.993	0.986	0.961	0.941
Regression				
Pearson Correlation				
No Smoothing	0.550	0.504	0.092	0.062
Gaussian Filter, σ=10	0.633	0.635	0.157	0.094

The H3K4me3 and H3K27ac models were performing relatively well on both regression and binary classification. As regression is the more valuable task, we decided to focus on regression for the deep learning models. In addition, based on the results of this model we only used the histone mark and DNase data that was smoothed using gaussian filtering of sigma 10 for the deep learning models.

a)

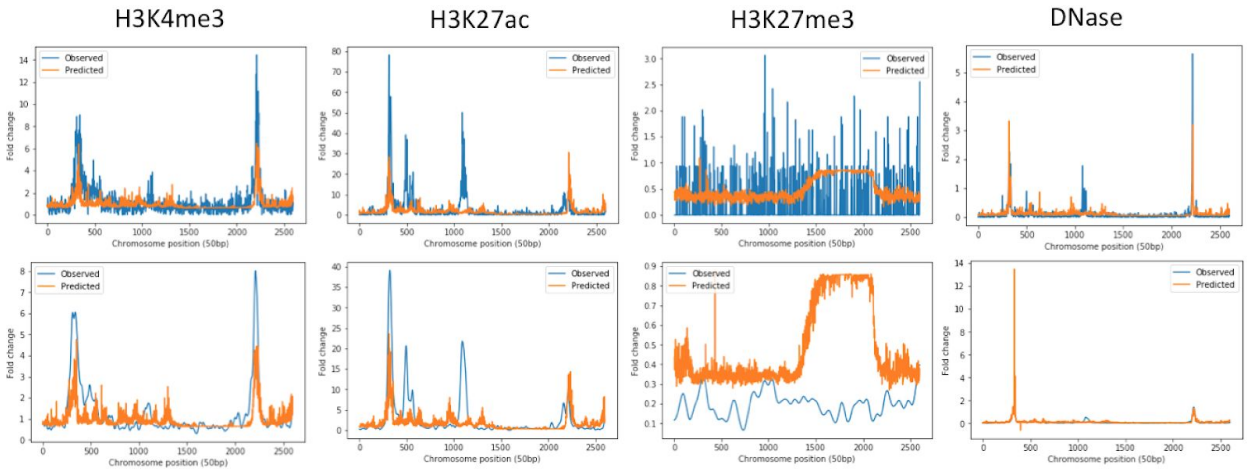


Figure 1: XGBoost Model Performance and Comparison of Smoothing Levels
a) Representative regions of XGBoost regression prediction vs. observed profiles for the four prediction tasks. The top panels were produced without smoothing the binned labels and the bottom panels were produced using gaussian filtering with sigma of 10.

Dense Neural Network

We next built a simple Dense Neural Network and achieved slightly higher performance than XGBoost (Table 2). The architecture consisted of five sequential dense layers with relu activations. We used the adam optimizer and root mean squared error as the loss function. We trained on chromosomes 1 and 3 and validated on chromosome 10 using the 5000bp window.

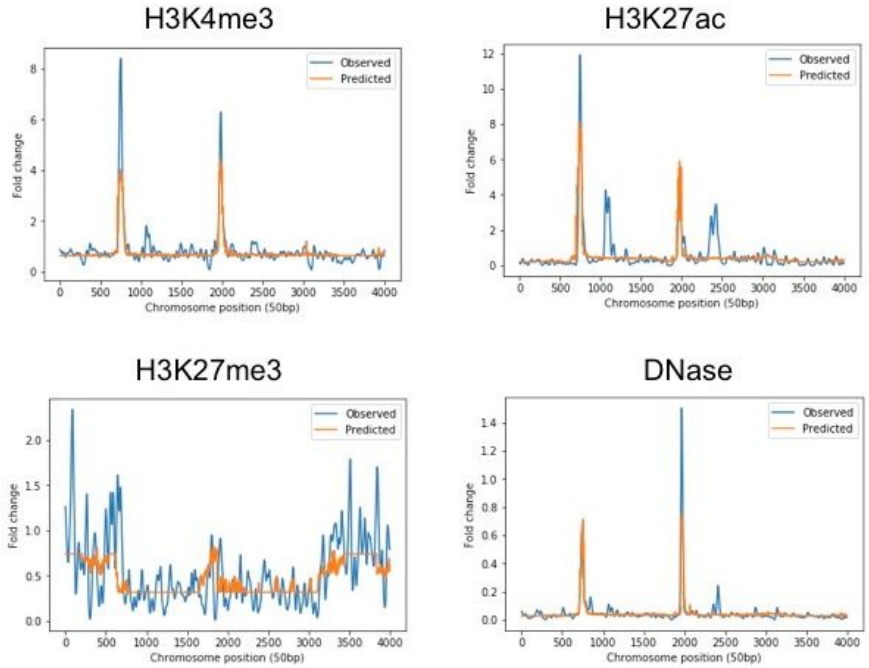


Figure 2: DNN Model Performance
Representative regions from the validation chromosome for each of the four tasks. Predicted profile in orange and observed profile in blue.

Table 2: DNN Model Evaluation

Model	H3K4me3	H3K27ac	H3K27me3	DNase
Regression				
Pearson Correlation	0.673	0.616	0.223	0.156
Root Mean Squared Error	0.155	0.341	0.381	0.024

Convolutional Neural Network

As there could be distal regulatory interactions that we're missing using the 5 kb window, we also wanted to build a convolutional neural network that took in a window of 128 kb. Our final CNN model has 6 convolutional layers, each followed by max-pooling, followed by a single fully connected layer. The structure is displayed in the appendix. While we explored multi-task models, we found out that we could achieve better performance with separate single-task models for each target. For the training done for H3K4me3 mark, our prediction had 0.678 Pearson correlation score with true labels, outperforming the DNN model both on the Pearson Correlation and Root Mean Squared Error.

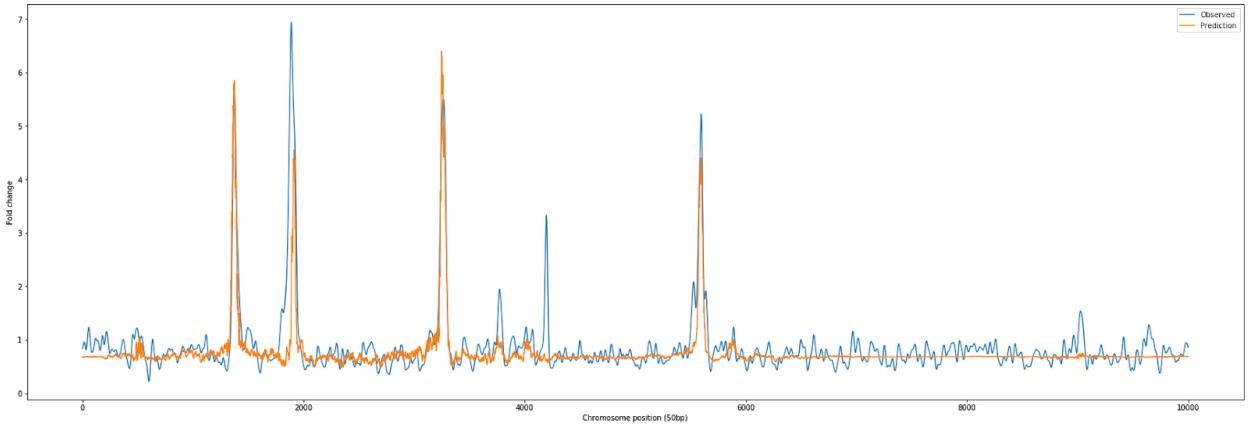


Figure 3: CNN Model Performance
a) Representative regions of CNN prediction vs. observed profiles for the H3K4me3 prediction task.

Discussion

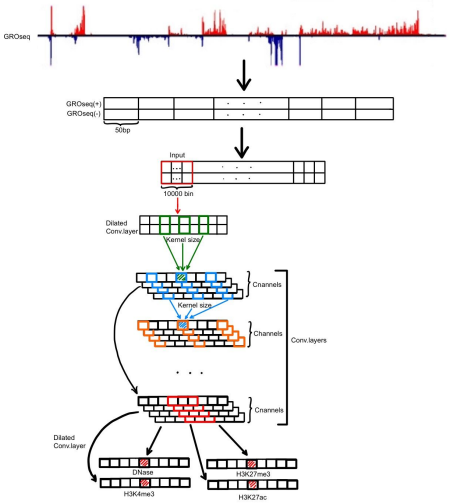
We achieved similar performance with all models what we tried. The XGBoost baseline performed the worst, but only marginally worse than the DNN and CNN. The models performed better on H3K4me3 and H3K27ac compared to H3K27me3. There could be a biological explanation for this difference, since H3K4me3 and H3K27ac are associated with transcriptional activity whereas H3K27me3 is associated with transcriptional repression. Another possible explanation is that the H3K27me3 data contains more noise than the other two histone marks and therefore requires additional data cleaning to achieve optimal performance. The models also don't perform as well on DNase. This is most likely due to the fact that only approximately 3% of DNase-hypersensitivity sites localize to transcriptional start sites (TSSs) and only 5% lie

within 2.5 kb of a TSS (Thurman, R. E. et al., 2012). The models trained using a 5 kb window are therefore unable to capture the vast majority of regulatory interactions with DNase-hypersensitivity sites.

While the pearson correlations between the predicted profiles and the observed profiles weren't extremely high, the correlations for H3K4me3 and H3K27ac were comparable to or even better than the results published in the Basenji paper; we achieved test set correlations above 0.6 for both H3K4me3 and H3K27ac whereas the median correlations published in the Basenji paper for the two histone marks were 0.501 and 0.502, respectively. However, the Basenji paper did achieve better results for DNase and H3K27me3. This dichotomy could be due to the different architectures or the different input data types; perhaps DNase and H3K27me3 are better captured using sequence information than with PRO-seq profile. Future work using different architectures could help answer that question.

Implementing Feedback

Initial Work



The first model we attempted in this project was a multi-task convolution neural network taking a 10,000 bin range and using two-dimensional positive and negative GROseq input to predict four corresponding 10,000 bin regions for H3K4me3, H3K27ac, H3K27me3 and DNase (see Figure). During the multiple experiments with this CNN architecture we faced with the following difficulties:

1. The model did not learn a lot with epochs.
   Explanation and solution: this phenomenon can be explained by the fact that the data is highly imbalanced, meaning that only small percent (around 1%) of output values is non-zero. As a solution we implemented mask MSE that calculates the mean square error only for bins with non-zero output values and used it as a loss function training the CNN.
2. The training loss was decreasing, whereas, the test loss did not improve.
   Explanation and solution: the possible explanation of this problem is that the multi-task model was training to predict all histone modifications simultaneously. It seems that there is a weak correlation between these four outputs and, therefore, adjusting CNN model parameters to predict one output can spoil the prediction for another output making the total loss function fluctuating around the same average spoiled value. We concluded that multi-task CNN is not a correct approach to this problem and, as a result, used the separate single-output CNN model for each histone modification.
3. The training loss values were higher than the test loss values.
   Explanation and solution: the reason for this loss performance is that the activity regions are not uniformly distributed across the DNA. For example, the first and the last 10% of

the bins usually do not contain a lot of histone activity, therefore, splitting the data into 90% and 10% of train and test data respectively the mask loss value could be much lower on a test set comparing to the one on a training set. One of the possible solutions is to shuffle the data before the split.

First Draft Feedback

1. We received consistent feedback to first train a simpler baseline model, before deploying a deep learning approach in order to gauge the complexity and scope of the task. We did so by running the XGBoost algorithm, which helped us pinpoint the data processing that we needed to enhance.
2. Once we settled on the correct data processing techniques we made significant progress in training the neural net models, which was also a suggested weakness of our first draft.
3. Another question that we saw was why we modified the original Basenji architecture and whether that was causing our issues with model training. We in fact iterated over a significant number of CNN models, but found out that once the data was processed correctly they had similar performance. We iterated through models with different kernel size, number of layers, strides, filter numbers, skip connections and fully connected layers.
4. Another suggested weakness of our approach was only training on a limited amount of data (1-5 chromosomes). This still remains as a work in progress, as we focused on iterating through multiple models and focused on speed. Now that we have finalised a lot of our approach we can definitely expand our training and test sets to include all available chromosomes.
5. We had originally proposed a RNN based architecture as well, which we had unsuccessfully tried to train. We believe that RNN based models are appropriate for sequence based data, such as ours. Unfortunately, since we focused on iterating through convolutional based models, we did not have enough time to circle back to training RNN models.

References

Danko, C.G. et al. (2015) Identification of active transcriptional regulatory elements from GRO-seq data. Nat. Methods 12, 433–438

Danko, C.G. et al. (2018) Identification of regulatory elements from nascent transcription using dREG. BioRxiv.

Kelley DR, Snoek J, Rinn JL. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. Genome Res. 26(7), 990–999 (2016).

Kelley DR, Reshef Y, Bileschi M, Belanger D, McLean CY, Snoek J. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. Genome Res. 2018 Mar.

Thurman, R. E. et al. (2012) The accessible chromatin landscape of the human genome. Nature http://dx.doi.org/10.1038/nature11232

T. Chen and C. Guestrin. (2016) XGBoost: A Scalable Tree Boosting System. SIGKDD.

Appendix

Convolutional Neural Network Structure

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputs (InputLayer)          (None, 2561, 1)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 2557, 16)          96
_________________________________________________________________
dropout_1 (Dropout)          (None, 2557, 16)          0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 1278, 16)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 1274, 16)          1296
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 637, 16)           0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 633, 16)           1296
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 316, 16)           0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 312, 16)           1296
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 156, 16)           0
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 152, 16)           1296
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 76, 16)            0
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 72, 16)            1296
_________________________________________________________________
max_pooling1d_6 (MaxPooling1 (None, 36, 16)            0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 32, 16)            1296
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
H3K4me3 (Dense)              (None, 1)                 513
=================================================================
Total params: 8,385
Trainable params: 8,385
Non-trainable params: 0
_________________________________________________________________
```