Characterization of Human Genome Regulatory Regions using Convolutional Neural Networks

Kristina Stoyanova
Mentors: Professor Ron Weiss (MIT), PhD Candidate Sebastian Palacios (MIT), Professor Matt Thomson (Caltech)

Abstract

The Human Genome Project has led to the sequencing of 94% percent of human DNA[1]. However, there are still parts of the genome that have not been fully annotated. A recent multivariate Hidden Markov Model, chromHMM, has annotated 25 different chromatin states (each representing a genomic element such as a promoter or heterochromatin). They used epigenetic marks – such as ChIP-seq data of histone modifications. We used these labeled regions of chromatin states as input for a Convolutional Neural Network (CNN) model that we created for further categorizing regions of the human genome. We specifically chose to work with epigenome E003, which is the H1 cell line, due to it being a frequently used human Embryonic Stem Cell (hESC) line. The model is currently predicting with a 79.35% percent accuracy. By working on this new CNN, we anticipate finding new regulatory regions that could aid in synthetic biology genetic circuit development, which could then be experimentally verified.

Text

Introduction

There are still parts of the genome that are not fully annotated. For instance, not all human promoters are cataloged[2]. Promoters contain basal elements (TATA box, PSE, or small nuclear RNA[3]) where the transcriptional machinery, such as RNA polymerase II and general transcription factor, bind[3]. Additionally, the activation or inactivation of many genes depends on whether they reside in regions of heterochromatin[4]; However, while areas of histone modification are well characterized (The most common histone modifications are H3K4me3, H3K9me3, and H3K27me3)[5], the exact regions of DNA that cause these modifications remains unknown. Motifs and patterns of DNA could affect heterochromatin thousands or millions of base pairs away.

The epigenome is made of proteins and chemicals that regulate the genome and turn genes on and off. In order to store so much DNA we have proteins known as histones which DNA coils around. Thus condensing chromatin into chromosomes. The regions of coiled DNA are known as heterochromatin and the genes that reside there are unable to be transcribed, whereas DNA that is accessible to polymerases is known as Euchromatin and that contains genes that can be considered on.

A common problem biological engineers face is when designing a new genetic circuit the genes may contain regions that cause it to have un intentional silencing effects. Only when the exogenous DNA is inserted into the host can some of these effects be discovered leading to the genetic circuit being non-functional.

We have created a neural network with the intention of predicting epigenetic states of whether a DNA section is heterochromatin or not, with the only input being the DNA sequence.

Meaning you know information on the epigenetic states before even inserting the construct into the host. As opposed to other methods that rely on immunoprecipitation assay data that can only be obtained after inserting the sequences into the host.

Data Source

The Wang Lab at the Washington University in St. Louis hosts a database for 127 consolidated human cell epigenomes for which they have created chromatin state annotations (promoter regions, actively transcribed parts, enhancers, etc.) using chromHMM[6]. ChromHMM is a multivariate Hidden Markov Model that uses epigenomic data such as chip-seq and DNase data to label regions of the genome for chromatin states. Our project used data from their 25 state, 12 marks, 127 epigenetic genomes model.

Data Processing and Model

We used data from the E003 (H1 cell line) because it is a well-characterized embryonic stem cell line that is used in labs, and more annotations of the ES cell line would be helpful in synthetic biology. We extracted the genomic locations of the annotations from the BED files and paired them with the actual genomic sequence. Then, the DNA sequence was one-hot encoded to be inputted into the neural network. Data was split into two categories: heterochromatin and regions that are not heterochromatin. The not active enhancer data (negative examples) was made of genomic sequences of other chromatin states in the database. One concern with preparing the data was that CNN models have to take fixed length inputs because of the fully-connected convolutional neural network at the end[7]. We created three different input types to overcome the CNN input limitations splitting each sequence into 200 bp (base pairs), zero padding to the maximum sequence length, and zero padding to 1000 bp while truncating longer sequences (more about each type of data preparation is in the methods section.

We then created a CNN (described in the methods section) and trained it and used a validation set for adjusting parameters within the model and have reached an accuracy of 61% when fit using testing data.

Discussion

Members in the Weiss lab often design new genetic circuits, but with the introduction of new sequences into the genome oftentimes, new genes are silenced due to histone methylation[8]. If we could create a machine learning model that detects DNA regions that cause histone methylation sites, it could significantly contribute to design in synthetic biology.

In future versions, we intend to change out more parameters of the model, including number of layers, optimizers, loss functions, and nodes in each layer. We hope to try methods that will allow us to use inputs of varying lengths for the CNN. We also look forward to expanding the model to other chromatin states, such as promoter regions.

Methods

Preparing the Data for the Model

Chrom HMM had 29975 sequences that were heterochromatin from their 25 chromatin state model[9]. We then randomly took another 29975 sequences from the other 24 chromatin states to use as negative data. Next, we added the reverse complement to avoid the model from learning directional patterns[9]. We then one-hot encoded the sequences into one of four-vectors ([1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]) representing nucleotides A, T, C, G ([0, 0, 0, 0] was used for zero padding sequences, if necessary). The data was split 20% for testing, 54% for training and 16% for validation data.

CNN 200bp Model

We took the sequences from chromHMM and split them into sections of 200 bp and input them by 200 sequence batches into the CNN. If the end of a sequence did not round to 200 bp we added zeros to the end. The main issue was this did not conserve the full length of a sequence. Seeking to preserve more spatial information of the DNA, we attempted other methods of splitting the data.

CNN Max Sequence Length Model

We took the dataset and zero-padded all the smaller sequences to the length of the largest sequence, which was 223836 bp. However, the model did not seem to learn as only 1019 sequences were larger than 100000, out of a total of 751466 sequences (Figure 1), and only 302702 sequences were larger than 1000. Thus, the model was given data with a too high percentage of zero-padding for it to learn from the significant parts.

CNN 1000 bp Model

Realizing approximately 59.7% of the data was 1000 bp or fewer, we decided to train the model only on the sequences 1000 bp or fewer, with the smaller sequences being padded with zeros. After running a few models and adjusting the parameters using the validation dataset (Figure 3), we picked the best performing model and ran it on the test data only on that model and recorded a 79.35% accuracy (Figure 4). The model summary is in Figure 2.

Optimizers and Loss

All models used the Adam optimizer, binary_crossentropy loss, and trained to 500 epochs with early stopping with a patience of 10.

Figures

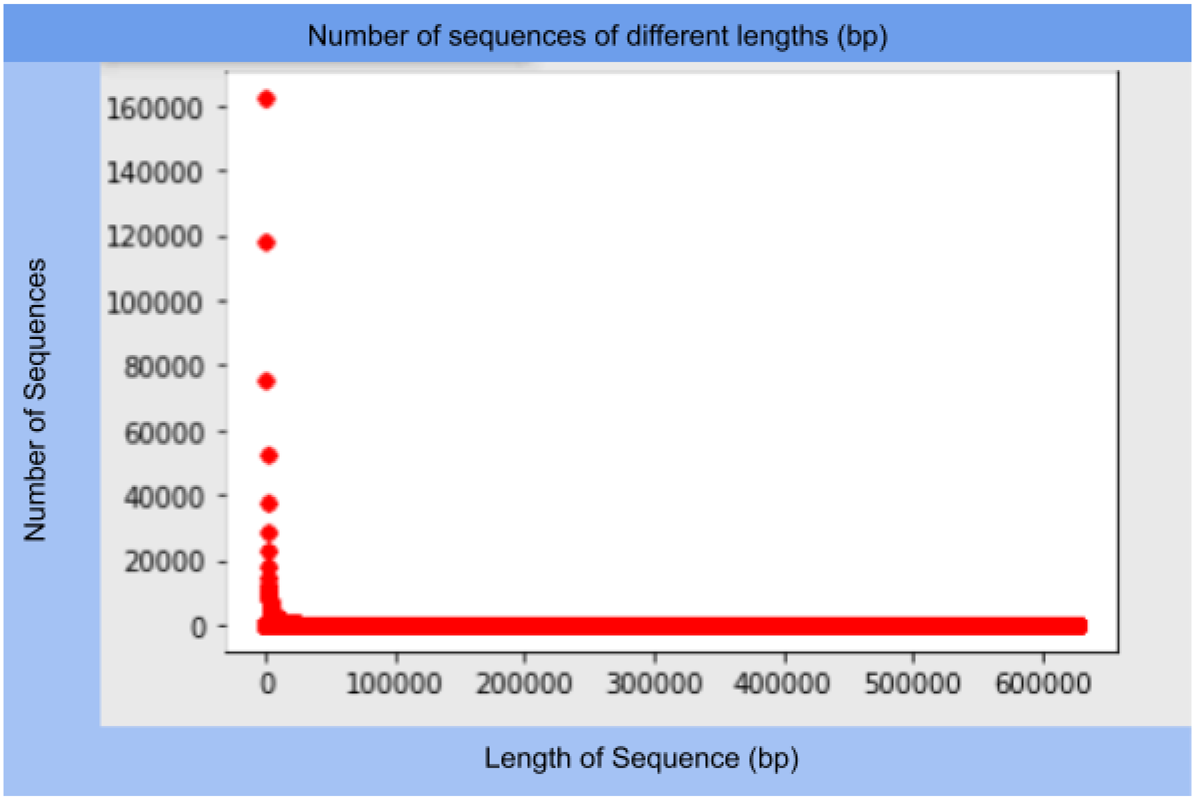
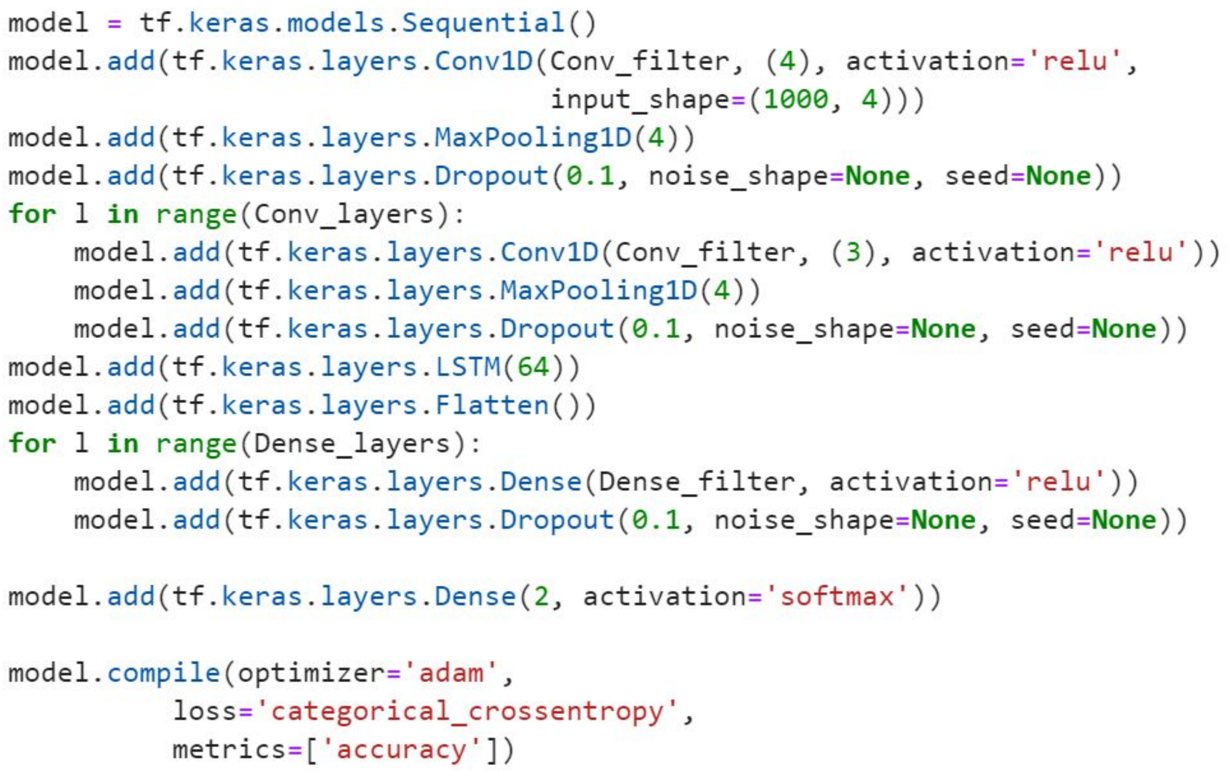


Figure 1: Graph of the number of sequences of each length (bp). 302702 sequences are longer than 1000 bp, 69574 sequences are longer than 10000 bp, and 1019 sequences are longer than 100000 bp, out of a total of 751466 sequences.

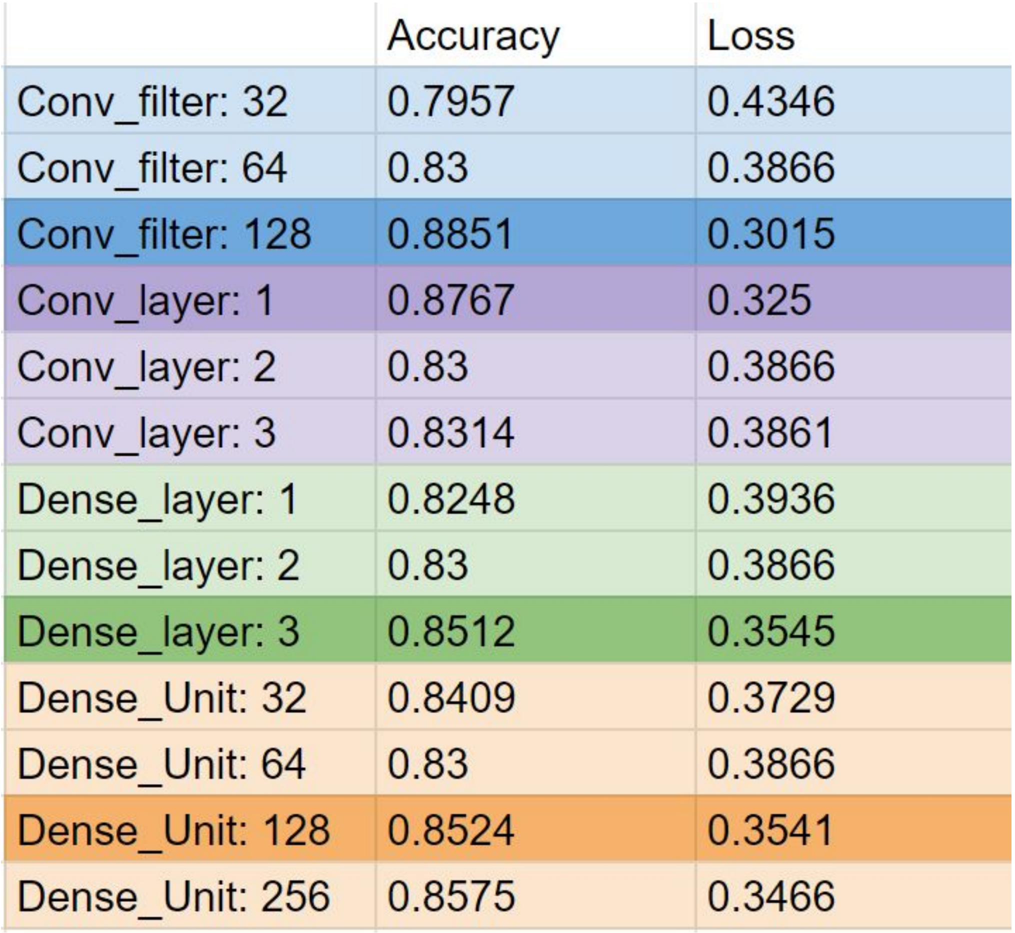


```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv1D(Conv_filter, (4), activation='relu',
                                 input_shape=(1000, 4)))
model.add(tf.keras.layers.MaxPooling1D(4))
model.add(tf.keras.layers.Dropout(0.1, noise_shape=None, seed=None))
for l in range(Conv_layers):
    model.add(tf.keras.layers.Conv1D(Conv_filter, (3), activation='relu'))
    model.add(tf.keras.layers.MaxPooling1D(4))
    model.add(tf.keras.layers.Dropout(0.1, noise_shape=None, seed=None))
model.add(tf.keras.layers.LSTM(64))
model.add(tf.keras.layers.Flatten())
for l in range(Dense_layers):
    model.add(tf.keras.layers.Dense(Dense_filter, activation='relu'))
    model.add(tf.keras.layers.Dropout(0.1, noise_shape=None, seed=None))

model.add(tf.keras.layers.Dense(2, activation='softmax'))

model.compile(optimizer='adam',
          loss='categorical_crossentropy',
          metrics=['accuracy'])
```

Figure 2: CNN for Model 1000bp with convolutional and max pooling layers in the beginning, followed by a flattening layer, and fully-connected dense layers with a sigmoid activation.



	Accuracy	Loss
Conv_filter: 32	0.7957	0.4346
Conv_filter: 64	0.83	0.3866
Conv_filter: 128	0.8851	0.3015
Conv_layer: 1	0.8767	0.325
Conv_layer: 2	0.83	0.3866
Conv_layer: 3	0.8314	0.3861
Dense_layer: 1	0.8248	0.3936
Dense_layer: 2	0.83	0.3866
Dense_layer: 3	0.8512	0.3545
Dense_Unit: 32	0.8409	0.3729
Dense_Unit: 64	0.83	0.3866
Dense_Unit: 128	0.8524	0.3541
Dense_Unit: 256	0.8575	0.3466

Figure 3: Accuracy and loss of model with different hyperparameter changes. We chose to have 128 convolutional filters, 1 convolutional layer, 3 dense layers, and 128 dense units, as they had the best accuracy on the validation.

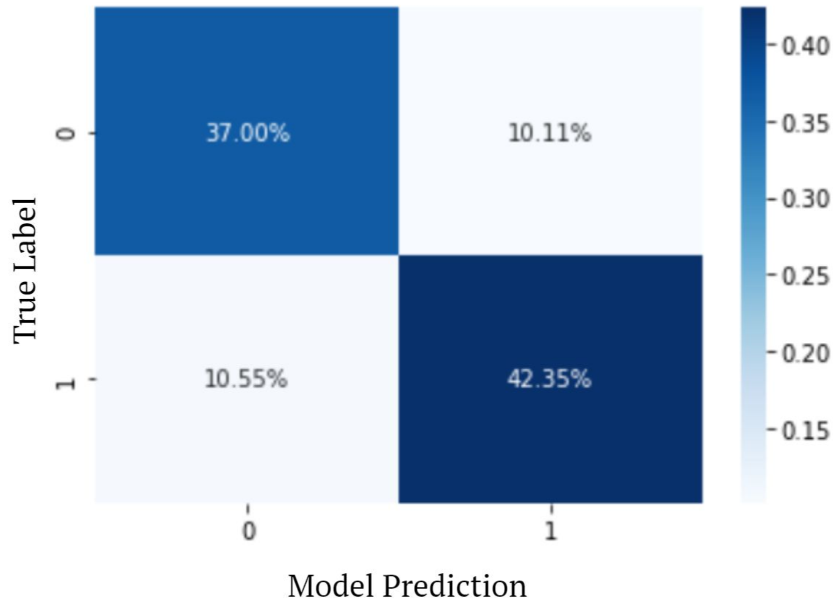


Figure 4: Confusion matrix of the final model we chose. Key: 1 is for heterochromatin, 0 for non-heterochromatin. The results were True Positive - 42.35%, True Negative - 37.00%, False Negative - 10.55%, False Positive - 10.11%. The model ran for 34 Epochs with a loss of 0.4552 and an Accuracy of 79.35%.

References

1. Lander, E., & Birren, B. (2001). Initial sequencing and analysis of the human genome. Nature, 409(6822), 860–921. doi: 10.1038/35057062
2. Rothman, J., & Singson, A. (Eds.). (2011). Major Caenorhabditis elegans Web Resources. Methods in Cell Biology Caenorhabditis Elegans: Molecular Genetics and Development, 461–462. doi: 10.1016/b978-0-12-544172-8.00017-7
3. Das, G., Hinkley, C. S., & Herr, W. (1995). Basal promoter elements as a selective determinant of transcriptional activator function. Nature, 374(6523), 657–660. doi: 10.1038/374657a0
4. Katan-Khaykovich, Y., & Struhl, K. (2005). Heterochromatin formation involves changes in histone modifications over multiple cell generations. *The EMBO Journal, 24*(12), 2138-2149. doi:10.1038/sj.emboj.7600692
5. Histone modifications. (2020, February 14). Retrieved from https://www.abcam.com/epigenetics/histone-modifications

6. Primary data processing and quality control. (n.d.). Retrieved September 28, 2020, from https://egg2.wustl.edu/roadmap/web_portal/processed_data.html
7. He, K., Zhang, X., Ren, S., & Sun, J. (2015, April 23). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. Retrieved September 28, 2020, from https://arxiv.org/abs/1406.4729
8. Allshire, R. C., & Madhani, H. D. (2017). Ten principles of heterochromatin formation and function. Nature Reviews Molecular Cell Biology, 19(4), 229-244. doi:10.1038/nrm.2017.119
9. ChromHMM: Chromatin state discovery and characterization. (n.d.). Retrieved from (n.d.). Retrieved September 28, 2020, from http://compbio.mit.edu/ChromHMM/
10. Nielsen, A.A.K., Voigt, C.A. Deep learning to predict the lab-of-origin of engineered DNA. Nat Commun 9, 3135 (2018). https://doi.org/10.1038/s41467-018-05378-z

Acknowledgments

Thanks to the Caltech SFP Office for providing the funding for the SURF grant, to Sebastian Palacios and Ron Weiss for providing such an exciting research experience, and to Matt Thomson for being my Caltech mentor.