Graph Convolutional Networks for Epigenetic State Prediction Using Both Sequence and 3D Genome Data

Jack Lanchantin and Yanjun Qi
University of Virginia
`{jjl5sw,yq2h}@virginia.edu`

Abstract

Deep learning models have proven to be effective methods for classifying DNA segments into their respective chromatin state labels. However, current methods consider small segments of DNA independently, ignoring long range dependencies and the 3D shape of DNA which are influential in chromatin state labeling. In this work, we extend previous methods that classify segments independently by utilizing the 3D shape of DNA from Hi-C data and graph neural networks to model the interactions between segments. Our method leads to stronger classification performance, particularly for labels that have a high degree of interactions with other DNA segments, as indicated by the Hi-C graph.

1 Introduction

With over 3 billion base pairs (characters consisting of A,C,G,T) in human DNA, modeling the chromatin state, or regulatory labeling, of each base pair has been a long standing challenge due to the shear length and complexity of DNA. Deep neural networks have shown promise in extracting useful features from segments of DNA in order to predict each segment's chromatin state as a probability of state labels (e.g., transcription factor binding). However, these methods heuristically divide DNA into short windows (about 1000 base pairs long) and predict the state labels of each window individually, disregarding the effects of distant windows. Due to the spatial 3D organization of DNA, distal elements (potentially over 1 million base pairs away) have shown to have effects on chromatin state of DNA [12]. In this work, we extend current methods which predict the labels of each DNA window independently by taking into account long range interactions in DNA via window-to-window dependencies.

Fig. 1(a), illustrates an example of long range dependencies, where the colored shapes represent transcription factors (TFs). TFs are chromatin state elements which typically bind to specific patterns in DNA known as "motifs". However, if a TF binds to a segment that does not contain the motif needed, it is likely that it binds due to the presence of other TFs nearby in the 3D space in order to form a protein complex. This in turn is related to the motifs nearby in the 3D space, corresponding to dependencies between the DNA segments. This is illustrated by the circle TF in Fig. 1(a). The two segments interacting in the middle of the diagram are very far in the 1D sequence representation, but very close in the 3D representation. Similarly, a TF may not bind to a segment with its motif present due to interfering TFs nearby in the 3D space. These types of interactions are lost in models which only consider short segments independently, resulting in a need for effectively considering such long range dependencies.

Modeling long range, or non-local interactions, has had a long history in many areas such as natural language processing where the label of one particular segment depends on the label of a segment far away. Recurrent neural networks such as LSTMs [6] have been used to model non-local dependencies where the network relies on the hidden state of the network to remember the state of a token (e.g. a word) very far away. However, LSTMs have been known to only be able to remember the dependencies from a small number of tokens back [6, 14]. This has lead to an increasing interest in the direct modeling of non-local dependencies using pairwise interactions [14, 5, 4].

In a related line of work, graph neural networks [13, 9, 15, 18] have been proposed to model the pairwise dependencies of nodes in graph or 3D structured data such as citation networks and point clouds. While typically viewed in its 1D sequential form, DNA can can be represented as 3D graph structured data via Hi-C maps, as shown in Fig. 1(b). Hi-C maps are matrices which give the number of contacts between two segments of DNA, and normalized Hi-C maps tell us the likelihood of two locations interacting [2]. In other words, the nodes are segments of DNA and the edges are physical interactions between the two segments, relating to the 3D structure. Such interactions have been shown to be crucial in regulatory processes such as gene transcription [12]. In other words, Hi-C contacts are a reflection of the likelihood of regulatory elements interacting. Thus, we hypothesize that accounting for such contacts will allow us to better model the state of DNA, leading to improved chromatin label classification performance. Using graph neural networks, we leverage the 3D graph structure of DNA given from Hi-C maps and model such interactions for a better representation of chromatin state.

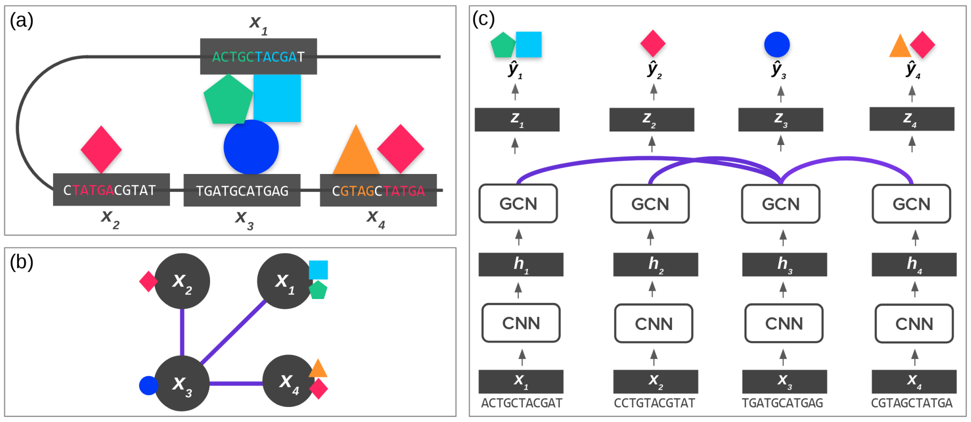

Figure 1: **(a)** The 3D shape of DNA can lead certain locations to be spatially close. These spatial interactions can influence chromatin state, such as TFs binding (as shown by the colored shapes). In most cases, chromatin state is determined by the DNA sequence, however it can also be influenced by interactions, such as the formation of TF complexes shown in the middle. **(b)** The graph representation of subfigure (a), where the orange lines are the edges indicated by Hi-C data. **(c)** Our GNN method which uses a graph neural network on top of convolutional outputs to consider dependencies between segment windows. The orange lines between windows correspond to edges in Hi-C data.

2 Experimental Setup

Problem Formulation The goal of chromatin state prediction is to tag each segment of DNA with the probability of each possible chromatin label, a multi-label classification task. Given an input DNA window $\mathbf{x}_i$ (a segment of length T), we want to predict the classification $y_i^l \in \{0, 1\}$ of each label l, where l ranges from 1 to L. In our formulation the labels are TF binding, DNase I-hypersensitivity, and histone modifications. Thus, we seek to learn a model f which takes in a sample $\mathbf{x}_i$ and predicts all labels $\hat{\mathbf{y}}_i$, where $\hat{\mathbf{y}}_i$ is an L dimensional vector.

Data Processing and Setup We follow a similar setup as in [17] where we bin the DNA into 1000 bp windows, and if any ChIP-seq peak overlaps with at least 100 bp of a particular window, we consider that a positive site for the ChIP-seq label. We then extract the 2000 bp sequence surrounding the center of the window for the input features, as done in [16] since the motif for a particular signal may not be contained fully in the 1000 bp window.

We then use Hi-C contact map data to represent interactions between the extracted windows. We use the 1000bp resolution GM12878 intra-chromosome Hi-C data from [12]. In other words, we extract the Hi-C contact map graph for each chromosome where the edges indicate number of contacts between two 1000bp windows. Since the full Hi-C contact for each chromosome is too dense, we rank each contact edge using the "SQRTVC" normalization method from [12], and set the top 500,000 Hi-C edges to 1, and all others to 0.

3 Method

Different from previous approaches, we follow a two-step method to predict the chromatin state labels of DNA segments. First, as previously done, we train a convolutional network (CNN) to predict the labels of each window independently. This step shuffles all the samples in the training set, and uses no known structure of pairwise window dependencies. Second, we use the pretrained feature vectors from the CNN and train a graph neural network to model the between window dependencies. This step uses the graph structure of each chromosome from the Hi-C contact maps and treats each chromosome as a single sample, where all windows are predicted simultaneously for a specific chromosome. We explain the details of our two step procedure in the following subsections.

3.1 Modeling Local Sequence Dependencies Using Convolutional Neural Networks

To encode the short windows of DNA, we feed the raw sequence of each window through a convolutional network (CNN). CNNs have become the de facto standard for encoding short DNA windows due to their inductive bias of encoding local dependencies in DNA. Each learned kernel, or filter in CNNs learns a DNA "motif", or short contiguous sequence representative of a particular output label [1].

In our implementation, we use several layers of convolution where each successive layer learns higher order motifs of the window. After several layers of convolution, the output of the last convolution layer is flattened into a vector and then linearly transformed to a lower dimensional vector of size d, which we denote $\mathbf{h}$. Thus, for a particular window $\mathbf{x}_i$, the output of the CNN model is a single vector $\mathbf{h}_i$. Each $\mathbf{h}_i$ is then fed through a classification layer which outputs a probability for each label: $\hat{\mathbf{y}}_i = \sigma(\mathbf{h}_i^\top \mathbf{W}_o + \mathbf{b}_o)$ where $\mathbf{W}_o$ is a $d \times L$ matrix, $\mathbf{b}_o$ is an L dimensional bias vector, and σ is a sigmoid function. The final output, $\hat{\mathbf{y}}_i$ is an L dimensional vector.

3.2 Modeling Long Range Sequence Dependencies Using Graph Neural Networks

While CNN models work well on individual windows, they disregard neighboring windows which may be influential in the chromatin state labelling. One option would be to extend the window size. The problem is that these long range contact dependencies are up to millions of base-pairs apart, making traditional convolutional models infeasible. [7] showed that dilated convolutions can implicitly capture long range dependencies, which we plan to compare to in future work. In this subsection, we introduce our proposal for efficiently modeling such long range interactions using graph neural networks on known interactions from Hi-C data. An overview of our method is shown in Figure 1(c).

In order to model relations *between* DNA windows (i.e. between each $\mathbf{x}_i$), we utilize DNA Hi-C contact maps. A Hi-C map can be represented as an adjacency matrix $\mathbf{A}$, where the nonzero elements indicate physical contact between two DNA windows. We then use a modified version of graph neural networks [9] (GNN) to update each $\mathbf{x}_i$ using known physical contacts, or 3D shape of DNA. Specifically, the GNN works by taking as input the intermediate output of a CNN for each window, $\mathbf{h}_i$, which we denote $\mathbf{h}_i^0$, and uses a parameterized summation of its neighbors in the Hi-C map, denoted $\mathcal{N}(i)$:

$$\mathbf{h}_i^{t+1} = \sigma\left(\tfrac{1}{|\mathcal{N}(i)|} \sum_{\mathbf{h}_j^t \in \mathcal{N}(i)} \mathbf{h}_j^t \mathbf{W}^t\right), \tag{1}$$

where $\sigma(\cdot)$ is a non-linear activation function such as tanh and $\mathbf{W}^t \in \mathbb{R}^{d\times d}$ is a linear feature transformation matrix for GNN layer t. Importantly, pairwise interactions are modeled using the summation in Eq. 1, where the representation of each DNA segment $\mathbf{x}_i$ is updated based on the representation of its neighbors which are segments that physically interact with $\mathbf{x}_i$. We can represent all window vectors $\mathbf{h}_i$ together by concatenating them, resulting in $\mathbf{H}$ of size $N \times d$ where N is the number of DNA windows, and d is the dimension of each $\mathbf{h}_i$. We can then represent the simultaneous update of all windows together using $\mathbf{A}'$, which is a normalized adjacency matrix:

$$\mathbf{H}^{t+1} = \sigma(\mathbf{A}'\mathbf{H}^t\mathbf{W}^t) \tag{2}$$

In our experiments, we use a variant of the graph neural network, which uses a gating function allowing the model to use the neighboring windows to update the vector for a particular window if needed:

$$\widetilde{\mathbf{H}}^t = \tanh\left(\mathbf{A}'\mathbf{H}^t\mathbf{W}_z^t\right) \tag{3}$$

$$\mathbf{G}^t = \text{sigmoid}\left(\widetilde{\mathbf{H}}^t\mathbf{w}_g^t\right) \tag{4}$$

$$\mathbf{H}^{t+1} = \mathbf{G}^t \odot \widetilde{\mathbf{H}}^t + (1 - \mathbf{G}^t) \odot \mathbf{H}^t \tag{5}$$

where $\mathbf{w}_g^t \in \mathbb{R}^d$ is used to compute the gate, $\mathbf{G}^t$, which allows the model to selectively choose between using the neighborhood representation of nodes, $\widetilde{\mathbf{H}}^t$, or the independent representation, $\mathbf{H}^t$. $\odot$ is an element-wise product. Equations 2-5 indicate one layer update of the window embeddings. In our experiments, we use a 2-layer GNN, and we denote the final output, $\mathbf{H}^2$ to be $\mathbf{Z}$. We then use the same linear classifier from the CNN, to now classify each $\mathbf{z}_i$ into its label probabilities: $\hat{\mathbf{y}}_i = \sigma(\mathbf{z}_i^\top \mathbf{W}_o + \mathbf{b}_o)$.

4 Model Variations

Local Dependency CNN Model We use the 6-layer CNN model from [16] as a baseline model to learn the features for each window (we modify the last layer in order to extract a lower dimensional feature vector). The CNN takes input DNA sequence $\mathbf{X}_i$, and outputs feature vector $\mathbf{h}_i$: $\mathbf{h}_i = f_{CNN}(\mathbf{X}_i)$. Each $\mathbf{h}_i$ is then fed to a linear classification layer, f_{out} which classifies $\mathbf{h}_i$ into probabilities for each label l: $\hat{\mathbf{y}}_i = \sigma(\mathbf{h}_i^\top \mathbf{W}_o + \mathbf{b}_o)$.

Long Range Dependency Models After pretraining the CNN model, the following models use all CNN window embeddings $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N$ from a particular chromosome (where N is the total number of windows in the chromosome) as input to model dependencies between windows. Each method outputs vectors $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N$, which are then all classified using $\hat{\mathbf{y}}_i = \sigma(\mathbf{z}_i^\top \mathbf{W}_o + \mathbf{b}_o)$. Model and training details are presented in appendix section 6.3.

LSTM: As a baseline to compare against using GNNs, we use a bi-directional LSTM [6]. The LSTM model takes in all window embeddings at once and models the sequential dependencies among windows: $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, ..., \mathbf{z}_N) = f_{LSTM}(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, ..., \mathbf{h}_N)$. We note this is different from other methods using LSTMs for classification which are used on top of CNNs *within* windows [11]. This instead is for modeling dependencies *between* windows.

GNN$_{const}$: To compare against the LSTM, this GNN variation uses a constant local set of neighbors, surrounding each window (7 on each side) which we denote as $\mathbf{A}_{const}$. $\mathbf{Z} = f_{GNN}(\mathbf{A}_{const}, \mathbf{H})$.

GNN$_{Hi\text{-}C}$: This GNN variation takes in all embeddings at once and models the long range dependencies among windows using the contact map matrix from Hi-C data, denoted $\mathbf{A}_{Hi\text{-}C}$. $\mathbf{Z} = f_{GNN}(\mathbf{A}_{Hi\text{-}C}, \mathbf{H})$.

GNN$_{const+Hi\text{-}C}$: Lastly, we use a combination of the constant neighborhood around each window (local) and Hi-C contact map neighbors (long-range). $\mathbf{Z} = f_{GNN}(\mathbf{A}_{const+Hi\text{-}C}, \mathbf{H})$.

Model	GM12878 Mean AUC	GM12878 Mean AUPR	GM12878 Mean Recall at 50% FDR	K562 Mean AUC	K562 Mean AUPR	K562 Mean Recall at 50% FDR
CNN [16]	0.895	0.350	0.293	0.894	0.325	0.265
LSTM	0.906	0.384	0.342	0.907	**0.363**	**0.326**
GNN$_{const}$	0.904	0.377	0.331	0.905	0.358	0.321
GNN$_{Hi\text{-}C}$	0.906	0.383	0.338	0.906	0.355	0.314
GNN$_{const+Hi\text{-}C}$	**0.909**	**0.388**	**0.345**	**0.908**	**0.363**	0.324

Table 1: Performance results. For both cell types, GM12878 and K562, we show the average across all labels for three different metrics. Our method, using a graph neural network (GNN) to model long range dependencies helps improve performance over the baseline CNN model which assumes all DNA segments are independent.

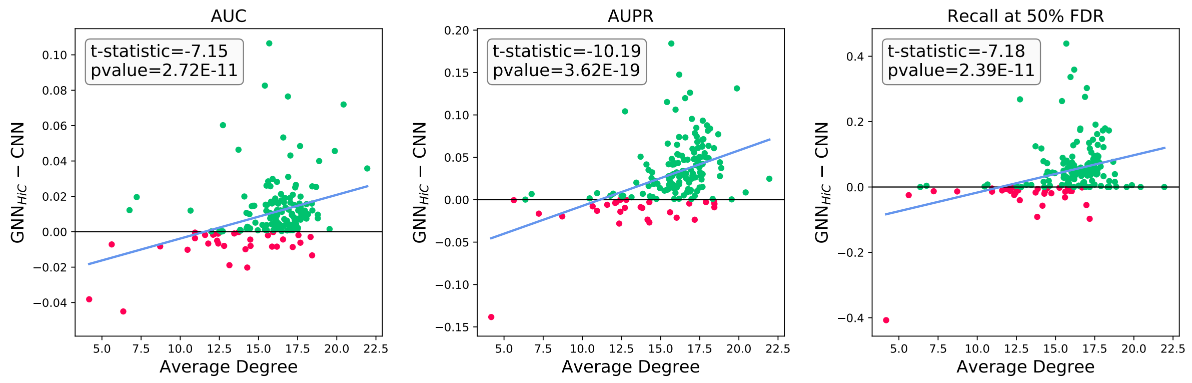

Figure 2: Comparison of our GNN$_{Hi\text{-}C}$ method vs the baseline CNN [16] for 3 Metrics. Each point represents one ChIP-seq label in K562. The labels are sorted in the x-axis by the average degree of their positive windows. The y-axis indicates absolute increase of the GNN$_{Hi\text{-}C}$ over the CNN model. As the average degree increases, the improvement of the GNN$_{Hi\text{-}C}$ model increases over the CNN. Green points indicate GNN$_{Hi\text{-}C}$ performed better, red indicate the CNN performed better. The blue line shows the linear trend line. The GNN$_{Hi\text{-}C}$ is significantly better, as demsonstrated by the pvalues from a pairwise t-test.

5 Experiments and Results

We use chromatin label ChIP-seq data from two of the most widely used cell types from ENCODE [3] and Roadmap [10], GM12878 and K562. For GM12878, there are 103 ChIP-seq labels, including 90 transcription factors, 2 DNase experiments, and 11 histone modifications. For K562, there are 164 ChIP-seq labels, including 150 transcription factors, 1 DNase experiment, and 13 histone modifications. For each cell type, we use all windows which have at least one positive label from the respective cell type. In our experiments, we use chromosomes 3, 12, and 17 as validation and chromosomes 1, 8, and 21 as testing. All other chromosomes (excluding X and Y) are used for training. To evaluate the methods, we use Area under the ROC curve (AUROC), Area under the Precision-Recall curve (AUPR), and the Mean Recall at 50% False Discovery Rate (FDR) cutoff.

Table 1 shows the mean metric results across all chromatin labels for each cell type. We can see that modeling between window dependencies results in improvements over the baseline CNN model which does not account for such long range interactions. More importantly, we can see that the GNN$_{Hi\text{-}C}$ models outperform the LSTM model, indicating that it is not only the closely neighboring windows in the 1D space, but also the close neighbors in the 3D space, as indicated by the used Hi-C maps.

Figure 2 shows a fine-grained comparison of the GNN$_{Hi\text{-}C}$ vs the baseline CNN model across three different metrics for K562 (GM12878 is shown in appendix). Each point represents a label, and the y-axis shows the improvement of the GNN$_{Hi\text{-}C}$ model over the CNN. The labels are sorted on the x-axis by the average degree of their positive windows across all samples. We can see that for all three metrics, the GNN improvements over the CNN increase as the average degree of the labels increase. This indicates that the GNN is important for labels which have many neighbors in the graph (are frequently in contact with other segments in the 3D space). Thus, we demonstrate the importance of modeling such long range interactions, especially for DNA segments which are close to many other segments in the 3D space. The p-values shown are computed by a pairwise t-test across all labels. The GNN$_{Hi\text{-}C}$ model significantly outperforms the CNN model in all three metrics.

Importantly, our results indicate that by using the long range interactions given by Hi-C data, we can obtain improvements in modeling the chromatin state labeling, resulting in better classification accuracy. We plan to further validate our method using real biological evidence from wet lab experiments.

References

[1] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831, 2015.

[2] Ferhat Ay, Timothy L Bailey, and William Stafford Noble. Statistical confidence estimation for hi-c data reveals regulatory chromatin contacts. *Genome research*, 24(6):999–1011, 2014.

[3] ENCODE Project Consortium et al. An integrated encyclopedia of dna elements in the human genome. *Nature*, 489(7414):57, 2012.

[4] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] David R Kelley, Yakir A Reshef, Maxwell Bileschi, David Belanger, Cory Y McLean, and Jasper Snoek. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome research*, 28(5):739–750, 2018.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[10] Anshul Kundaje, Wouter Meuleman, Jason Ernst, Misha Bilenky, Angela Yen, Alireza Heravi-Moussavi, Pouya Kheradpour, Zhizhuo Zhang, Jianrong Wang, Michael J Ziller, et al. Integrative analysis of 111 reference human epigenomes. *Nature*, 518(7539):317, 2015.

[11] Daniel Quang and Xiaohui Xie. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic acids research*, 44(11):e107–e107, 2016.

[12] Suhas SP Rao, Miriam H Huntley, Neva C Durand, Elena K Stamenova, Ivan D Bochkov, James T Robinson, Adrian L Sanborn, Ido Machol, Arina D Omer, Eric S Lander, et al. A 3d map of the human genome at kilobase resolution reveals principles of chromatin looping. *Cell*, 159(7):1665–1680, 2014.

[13] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[15] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[16] Jian Zhou, Chandra L Theesfeld, Kevin Yao, Kathleen M Chen, Aaron K Wong, and Olga G Troyanskaya. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nature genetics*, 50(8):1171, 2018.

[17] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931, 2015.

[18] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

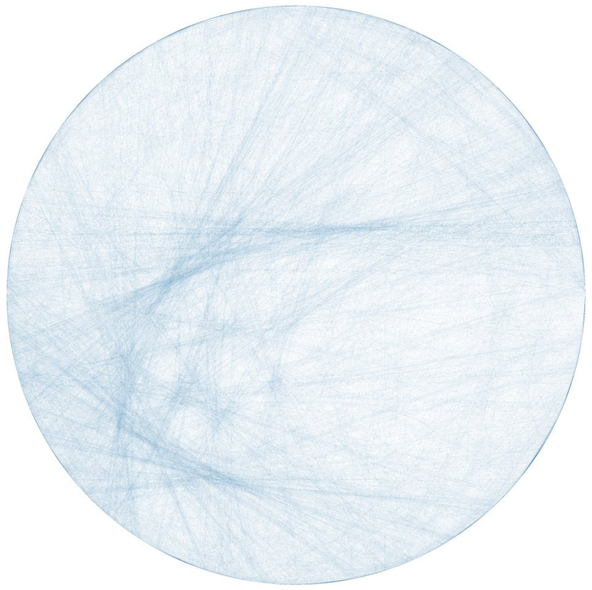

Figure 3: GM12878 Chromosome 3 Hi-C contact map edges. The edges (lines) are weighted weighted by the gating function, **G** (Eq. 4). The darker the line, the more the edge was used, indicating that the Hi-C data was used by the GNN for that particular interaction.

6 Supplementary

6.1 Long Range Interaction Visualization

Since we use a gating function which allows a particular window to selectively use its neighbors, we seek to visualize which edges are used. Figure 3 shows an example from Chromosome 3 in GM12878, where the darkness of the line indicates how the amount those edges were used, as computed by the gating function G.

6.2 Additional Performance Figures

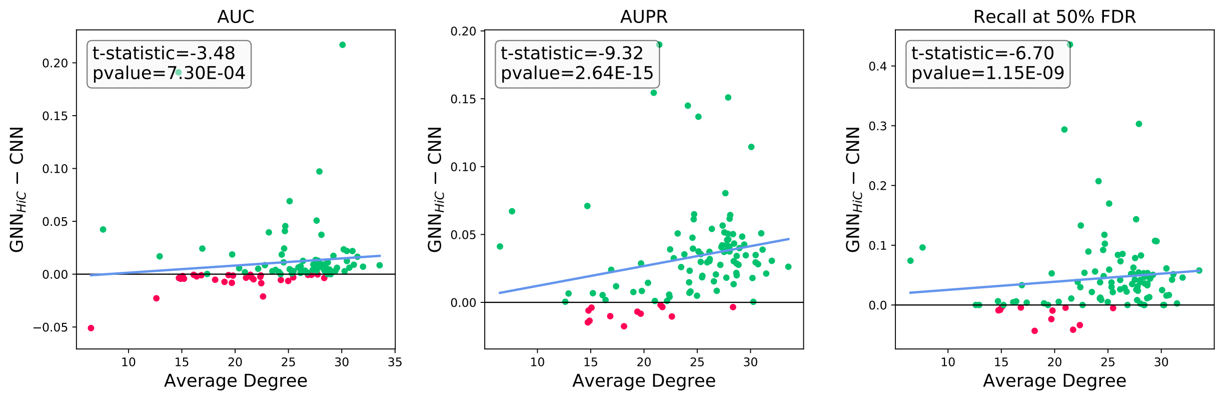

Figure 4: Comparison of our GCN (Hi-C) method vs the baseline CNN [16] for 3 Metrics. Each point represents one ChIP-seq label in GM12878. The labels are sorted in the x-axis by the average degree of their positive windows. We can see that as the average degree increases, the performance of the improvement of the GNN model increases over the CNN model. The blue line shows the linear trend line.

6.3 Model Details

For all model predictions, we run the forward and the reverse complement through simultaneously and average the output of the two. All DNA segment inputs are a $4 \times T$ one-hot encoded matrix, where the 4 represents the characters A,C,G,T, and T represents sequence length, which in our case is 2000.

All of our models are trained using Adam [8] using a learning rate of 0.0002. The CNN model is trained using a batch size of 64, and the GNN and LSTM models are trained using an entire chromosome as a batch (since each are modelling the between window dependencies of an entire chromosome at once). The CNN model projects each window to a vector of dimension 128. The LSTM and GNN both use two layers of feature dimension 128 at each layer.