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

Jack Lanchantin and Yanjun Qi

University of Virginia, Charlottesville VA 22903, USA
{jjl5sw,yq2h}@virginia.edu

Abstract. Predictive models of DNA epigenetic state such as transcription factor binding are essential for understanding regulatory processes and developing gene therapies. It is known that the 3D genome, or spatial structure of DNA, is highly influential in the epigenetic state. Deep neural networks have achieved state of the art performance on epigenetic state prediction by using short windows of DNA sequences independently. These methods, however, ignore the long-range dependencies when predicting the epigenetic states because modeling the 3D genome is challenging. In this work, we introduce ChromeGCN, a graph convolutional network for epigenetic state prediction by fusing both local sequence and long-range 3D genome information. By incorporating the 3D genome, we relax the i.i.d. assumption of local windows for a better representation of DNA. ChromeGCN explicitly incorporates known long-range interactions into the modeling, allowing us to identify and interpret those important long-range dependencies in influencing epigenetic states. We show experimentally that by fusing sequential and 3D genome data using ChromeGCN, we get a significant improvement over the state-of-the-art deep learning methods as indicated by three metrics. Importantly, we show that ChromeGCN is particularly useful for identifying epigenetic effects in those DNA windows that have a high degree of interactions with other DNA windows.

Keywords: Genomics · Hi-C · Deep Learning · Graph Convolutional Networks

1 Introduction

The human genome includes over 3 billion base pairs (bp), each being described as A,C,G, or T. Chromatin (DNA and its organizing proteins) is responsible for many regulatory processes such as controlling the expression of a certain gene. Active chromatin elements such as transcription factors proteins binding at particular location in DNA or histone modifications are what constitute that location's "epigenetic state" [1]. Understanding the epigenetic state of a local region of DNA is a step toward understanding how that region influences relevant gene regulation since epigenetic state is a direct factor in regulating expression. Since biological experiments are time-consuming and expensive, computational methods that can accurately simulate and predict the epigenetic state are crucial. Modeling the epigenetic state of each base pair has been a long standing challenge due to the sheer length and complexity of genome DNA. Deep neural networks have shown state-of-the-art performance in extracting useful features from segments of DNA to predict the epigenetic state (e.g., if a transcription factor protein binds to that location or not) [36, 1]. However, these methods heuristically divide DNA into local "windows" (e.g., about 1000bp long) and predict the states of each window independently, disregarding the effects of distant windows. Due to the spatial 3D organization of chromatin in a cell, distal DNA elements (potentially over 1 million bp away) have shown to have effects on epigenetic states [23, 21].

Fig. 1(a) illustrates the importance of using both sequence and 3D genome data. This figure shows long-range dependencies between chromatin windows, where the colored shapes represent multiple transcription factors (TF) proteins. TFs typically bind to specific sequence patterns in DNA known as motifs [29]. However, a TF may also bind to a DNA window due to the presence of other TFs nearby in the 3D space because they form a protein complex [5, 21]. Such a case will result in motifs far away in the 1D genome coordinate space, but nearby in the 3D space. The corresponding dependencies between the chromatin windows are illustrated by the triangle, square, and circle TFs. The two segments interacting in the middle of the diagram are very far in the 1D sequence representation (represented by the grey line), but very close in the 3D representation.

[1] We use the terms epigenetic state and chromatin state interchangeably.

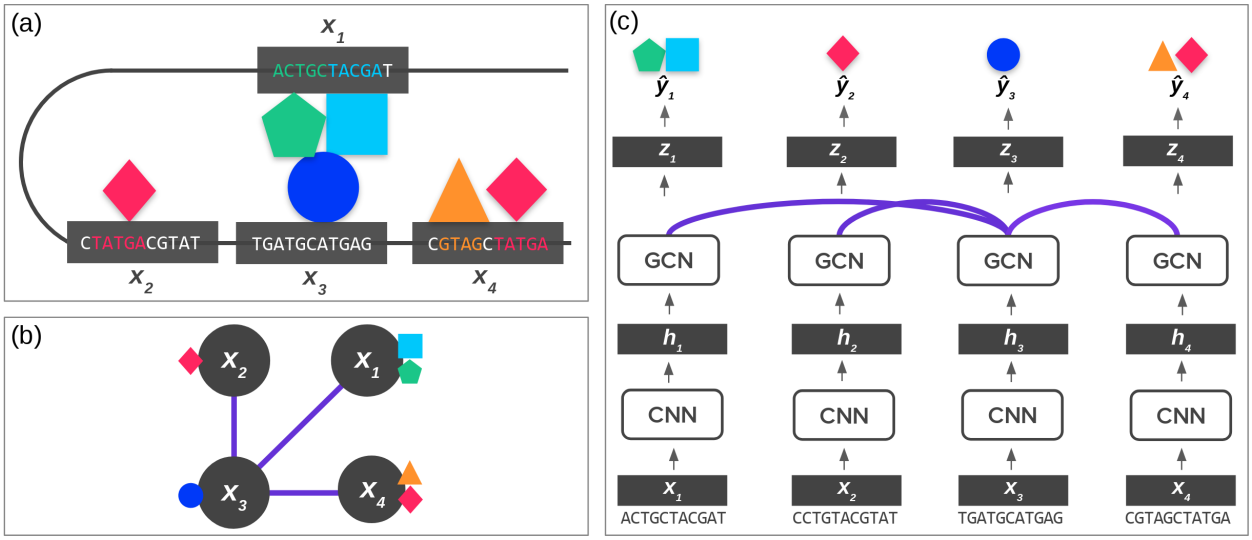

Fig. 1: **(a) 3D Genome.** The 3D shape of chromatin can lead DNA "windows" (shown in grey boxes) far apart in the 1D genome space to be spatially close. These spatial interactions can influence epigenetic states, such as TFs binding (as shown by the colored shapes). In most cases, the DNA sequence determines the epigenetic state. However, it can also be influenced by interactions, such as the formation of TF complexes shown in the middle. **(b) Graph Representation of DNA.** Using Hi-C data, we can represent subfigure (a) using a graph, where the lines between windows are the edges indicated by Hi-C data. **(c) ChromeGCN.** By using a graph convolutional network on top of convolutional outputs the model considers the known dependencies between long-range DNA windows. The lines between windows correspond to edges in Hi-C data.

Similarly, a TF may not bind to a segment with its motif present due to another interfering TF nearby in the 3D space. These types of interactions are lost in data-driven prediction models that only consider local DNA segments independently.

However, modeling these known long-range interactions between windows is difficult. Local sequence window-based prediction methods assume data samples are independent according to the commonly used independent and identically distributed (IID) assumption. Yet, the long-range dependencies existing in DNA make windows not IID.

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 [15] have been used to model non-local dependencies where the model relies on the hidden state to remember the state of a token (e.g., a word) very far away. However, LSTMs are known to only remember a small number of tokens back, leading to rather "local" relationship learning [15, 30]. This drawback has lead to an increasing interest in the explicit modeling of non-local dependencies via pairwise interaction models such as transformers [30, 9, 8].

In a related line of work, graph convolutional networks (GCNs) have been proposed to model the pairwise dependencies of nodes in graph or 3D structured data such as citation networks and point clouds [24, 17, 32, 37]. This direct modeling of edges allows the network to learn non-local relationships. While typically viewed in its 1D sequential form, DNA can be represented as 3D genome structured data via Hi-C maps, as shown in Fig. 1(b). Hi-C maps are matrices that give the number of contacts between two segments of DNA, and normalized Hi-C maps tell us the likelihood of two locations interacting [2]. Using Hi-C data, segments of DNA can be represented as nodes on a graph, and edges are interactions between segements. Such interactions can be crucial in regulatory processes such as gene transcription [23]. That is, Hi-C contacts are a direct reflection of how distant epigenetic elements interact. We hypothesize that accounting for such interactions will lead to improved epigenetic state prediction accuracy.

In this work, propose ChromeGCN, a novel method that uses a fusion of both sequence and 3D genome data (in the form of Hi-C maps) to predict the epigenetic state of DNA segments. To the best of our knowledge, ChromeGCN is the first deep learning framework that successfully combines sequence and 3D genome data to model both local sequence features and long-range dependencies for epigenetic state prediction.

ChromeGCN works by first representing DNA windows as a d-dimensional vector with a convolutional neural network on the local window sequence. We then revise the window vector using a graph convolutional network on all window relationships from Hi-C 3d genome data. We test ChromeGCN on datasets from two cell lines where we compare against the previous state of the art epigenetic state prediction methods. We demonstrate that ChromeGCN outperforms previous methods, especially for labels that are highly correlated with long-range chromatin interactions.

An important aspect of ChromeGCN is that it allows us to better understand Hi-C data in the context of epigenetic state. Hi-C maps tell us where the contacts are in the genome, but they don't tell us important contacts for epigenetic labels. Using ChromeGCN, we propose Hi-C saliency maps to understand which Hi-C contacts are most important for epigenetic state labeling. Since ChromeGCN uses explicit long-range relationships from Hi-C data (as opposed to implicit long-range relationships using a recurrent neural network), we can easily understand the important relationships for greater interpretability.

The main contributions of this paper are:

1. We propose ChromeGCN, a novel framework that incorporates both local sequence and long-range 3D genome data for epigenetic state prediction.
2. We experimentally validate the importance of ChromeGCN on two cell lines from ENCODE, showing that modeling long range genome dependencies is critically important.
3. We introduce Hi-C saliency maps, a method to identify the important long range interactions for epigenetic state prediction from Hi-C data.

2 Background and Related Work

2.1 Predicting Epigenetic State Using Machine Learning

Computational models for accurately predicting epigenetic state labels from DNA sequence have gained popularity in recent years due to the urgency of the task for many applications. For instance, predicting how epigenetic effects vary when variants in DNA occur. The importance of computational modeling arises from the low cost and high speed in comparison to biological lab experiments.

One class of methods for state prediction used generative models in the form of position weight matrices (PWM) [29]. These methods construct motifs, or short contiguous sequences (often 8-20bp in length), which are representative of a particular epigenetic state label such as a transcription factor binding. A new sequence can then be classified according to how well it matches the motif. A significant drawback of using predefined motif features is that it is difficult to find the correct motifs for predicting unseen sequences [1]. Another class of methods use string kernels (SK) [10, 28], where some kernel function is built to capture the similarity between DNA segments according to substring patterns. However, these methods suffer from the issue of a predefined feature engineering. Moreover, these methods do not scale to a large number of sequences [36].

To overcome the issues of PWM and SK methods, researchers turned to automatic feature extraction using deep neural networks which have outperformed both generative PWM and SK methods [1, 36]. Convolutional neural networks (CNNs) were the first deep learning method to outperform previous methods. CNNs have been used extensively to learn features of DNA for sequence-based prediction [36, 1, 14, 19, 35]. The benefit of convolutional models is that they have an inductive bias for modeling translation invariant features in DNA sequences. This allows CNNs to effectively learn the correct "motifs" or kernels for chromatin labeling. There has since been several revisions to the original CNN models for marginally better feature extraction, such as adding a recurrent network on top of the CNN motif features [22, 20].

However, current state-of-the-art models only learn the features from the sequences of individual local windows and not between windows (i.e., longer-range interactions). Since DNA interacts with itself in the form of long-range 3D contacts, labeling the epigenetic states of a window can be affected by another distant window due. [16] use longer input sequences (131kbp), but the dependencies are modeled implicitly using dilated convolution across 128bp windows. Accordingly, methods that account for explicit long range 3D chromatin contacts are needed to model the true interactions in DNA.

2.2 DNA Interactions via Hi-C Maps

Hi-C experiments, and 3C experiments in general, are biological methods used to analyze the spatial organization of chromatin in a cell. These methods quantify the number of interactions between genomic loci. Two loci that are close in 3D space due to chromatin folding may be separated by up to millions of nucleotides in

the sequential genome. Such interactions may result from biological functions, such as protein interactions [12]. Appendix Fig. 9 shows an example Hi-C map from the GM12878 cell line. The darker the lines indicate more DNA-DNA interactions.

Since the first Hi-C maps were generated, many works have been introduced to analyze the maps. [21] investigated the spatial relationships of co-localized TF binding sites within the 3D genome. They show that for certain TFs, there is a positive correlation of occupied binding sites with their spatial proximity in the 3D space. This is especially apparent for weak TF binding sites and at enhancer regions. [3] identified that the ZNF143 TF motif in the promoter regions provides sequence specificity for long range promoter-enhancer interactions. [34] identified coupling DNA motif pairs on long-range chromatin interactions. [25] use convolutional neural networks to predict Hi-C interactions from sequence inputs. None of the previous methods, however, use known Hi-C data to learn better feature representations of genomic sequences for epigenetic state prediction.

2.3 Graph Convolutional Networks

Graph convolutional networks (GCNs) were recently introduced to model non-local or non-smooth data [24, 7, 17, 13, 11, 31]. For the task of node classification, GCNs can learn useful node representations which encode both node-level features and relationships between connected nodes. Essentially, GCNs learn node representations by encoding local graph structures and node attributes, and the whole framework can be trained in an end-to-end fashion. Because of their effectiveness in learning graph representations, they achieve state-of-the-art results in node classification. The main assumption is that the input samples (in our case, individual DNA windows) are not independent. By modeling the graph dependency between samples, we can obtain a better representation of each of the samples. Non-local neural networks [32] are an instantiation of graph convolution, which was designed to model the long-range interactions in video frames.

3 Problem Formulation and Data Processing

The objective of epigenetic state prediction (i.e., chromatin effect prediction) is to tag segments of DNA with the probability of how likely a certain chromatin effect (aka epigenetic state label) is present. In our formulation, we define epigenetic state labels to include transcription factor (TF) binding, histone modifications (HM), and accessibility (DNase I). This is known as a multi-label classification task, where multiple labels can be positive at once (different from multi-class tasks where only one label can be positive). Formally, Given an input DNA window $\mathbf{x}_i$ (a segment of length T), we want to predict $y_i^l \in \{0, 1\}$ for a label l, where l ranges from 1 to L.

Cell Line	Train Windows	Valid Windows	Test Windows	TFs	HMs	DNase I
GM12878	368,082	89,911	79,731	90	11	2
K562	457,609	106,777	117,815	150	12	2

Table 1: **Datasets Summary.** GM12878 contains 103 total epigenetic state labels, and K562 contains 164 total labels. We use the same chromosomes for training, validation, and testing for both datasets.

Sequence Data We derive epigenetic labels using ChIP-seq data from ENCODE [6]. We use the cell lines GM12878 and K562, two of the most widely used from ENCODE and Roadmap [6, 18]. For each cell line, we use all windows which have at least one positive epigenetic ChIP-seq peak. We consider any peak from ENCODE to be a positive peak. We follow a similar setup as in [36] where we bin the DNA into 1000bp windows. If any ChIP-seq peak overlaps with at least 100bp of a particular window, we consider that a positive window for that chromatin label. We then extract the 2000bp sequence surrounding the center of each window as the input features, as done in [35], since the motif for a particular signal may not be contained fully in the 1000bp length window. Although we use the 2000bp sequence, we consider each window to be the original non-overlapping 1000bp for notation purposes. An illustration of how sequences are extracted is shown in Appendix Fig. 10 (a). Following [36], we use chromosome 8 for testing and also add chromosomes 1 and 21. Chromosomes 3, 12, and 17 are used for validation, and all other chromosomes (excluding X and Y) are used for training. The datasets are summarized in Table 1.

3D Genome Data We then use 3D genome data from Hi-C contact maps to extract interaction evidence between the DNA windows. We use 1000bp resolution intra-chromosome Hi-C data from [23] (for K562, the

lowest resolution is 5000bp, so we upsample to get 1000bp resolution). We convert the Hi-C contact map for each chromosome into a graph whose nodes are 1000bp DNA windows and whose edges represent contact between two 1000bp windows. Since the full Hi-C contact for each chromosome is too dense, we rank each contact edge, and use the top 500,000 Hi-C contacts as edges per chromosome (each chromosome maps to a Hi-C graph). Contacts are ranked using the $SQRTVC$ normalization from [23], which normalizes for the distance between two positions so that long-range contacts are included in the top 500k.

4 Method

Our goal is to learn a model f which takes in a DNA subsequence window $\mathbf{x}_i$ and predicts the probability of a set of epigenetic labels $\hat{\mathbf{y}}_i = f(\mathbf{x}_i)$, where $\hat{\mathbf{y}}_i$ is an L dimensional vector. Our method, ChromeGCN uses three submodules for f: f_{CNN}, f_{GCN}, and f_{Pred}. The first module, f_{CNN}, models local sequence patterns from each window using a convolutional neural network. This module takes as input $\mathbf{x}_i$ and outputs a vector representation $\mathbf{h}_i = f_{CNN}(\mathbf{x}_i)$. The second module, f_{GCN}, models long range 3D genome dependencies between windows using a graph convolutional network. This module takes as input all window vectors $\mathbf{h}_i$ concatenated as $\mathbf{H}$, as well as their Hi-C relationships represented by adjacency matrix $\mathbf{A}$, and outputs refined representations of all windows $\mathbf{Z} = f_{GCN}(\mathbf{H}, \mathbf{A})$. The $\mathbf{z}_i$ of each window now encodes both window sequence patterns and the relationships between windows. We can then predict the epigenetic labels using a classifier function on each $\mathbf{z}_i$ using $\hat{\mathbf{y}}_i = f_{Pred}(\mathbf{z}_i)$. An overview of ChromeGCN is shown in Fig. 1(c). The following subsections explain each submodule in detail.

4.1 Modeling Local Sequence Representations Using Convolutional Neural Networks

Following the recent successes in many epigenetic label prediction tasks [36, 1, 22, 20, 35], we learn a representation of each genomic window sequence $\mathbf{x}_i$ using a convolutional neural network (CNN). CNNs have become the de facto standard for encoding short DNA windows due to their properties, which effectively capture local sequence structure. Each learned kernel, or filter, in CNNs effectively learns a DNA "motif", or short contiguous sequence representative of a particular output label [1]. Since many epigenetic processes are hypothesized to be dependent on motifs [3], CNNs are a good choice for encoding DNA.

This module, f_{CNN}, takes an input genomic sequence window $\mathbf{x}_i$, and outputs an embedding representation vector $\mathbf{h}_i$. We represent window $\mathbf{x}_i$ of length τ as a one-hot encoded matrix $\mathbf{X}_i \in \mathbb{R}^{\tau \times n_{in}}$, where n_{in} is 4, representing the base-pair characters A,C,G, and T. Convolution with filters (i.e. learned motifs) of length $k < \tau$ takes an input data matrix $\mathbf{X}_i$ of size $\tau \times n_{in}$, and outputs a matrix $\mathbf{X'}_i$ of size $\tau \times n_{out}$, where n_{out} is the chosen dimension of the learned hidden representations:

$$\mathbf{X'}_{i\,t,u} = \sigma\left(\sum_{j=1}^{n_{in}}\sum_{z=1}^{k}\mathbf{W}_{u,j,z}\mathbf{X}_{i\,t+z-1,j}\right), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{n_{out} \times n_{in} \times k}$ are the trainable weights, and σ is a function enforcing element-wise nonlinearity.

Eq. 1 can then be repeated for several layers where each successive layer uses a new $\mathbf{W}$ and n_{in} is replaced with n_{out} from the previous layer. In our implementation, we use six layers of convolution where each successive layer learns higher-order motifs of the window. After the convolutional layers, the output of the last layer is flattened into a vector and then linearly transformed into a lower-dimensional vector of size d, which we denote $\mathbf{h}_i$. Succinctly, the CNN module computes the following: $\mathbf{h}_i = f_{CNN}(\mathbf{x}_i)$ for each window.

4.2 Modeling Long-Range 3D Genome Relationships Using Graph Convolutional Networks

While CNN models work well on independent local window sequences, they disregard known long-range relationships between windows that are influential in the epigenetic state. One option would be to extend the window size. However, due to the 3D shape of DNA, long-range contact dependencies may be located millions of base-pairs apart, making current convolutional models infeasible. In this subsection, we introduce the f_{GCN} module, a method to explicitly and efficiently model such long-range interactions using graph convolutional networks.

Known long-range relationships in the 3D genome are available in the form of Hi-C contact maps. A Hi-C map can be represented as an adjacency matrix $\mathbf{A}$, where the nonzero elements indicate contacts in the 3D space between two DNA windows[2]. In our formulation, we represent sequence windows $\mathbf{x}_i$ as nodes on a graph, and $\mathbf{A}$ are the edges between the windows. We can then use a modified version of graph convolutional networks [17] (GCN) to update each $\mathbf{x}_i$ with its neighboring windows $\mathbf{x}_j, j \in \mathcal{N}(i)$, where $\mathcal{N}(i)$ denotes the neighbors of node i obtained from $\mathbf{A}$. The GCN works by revising a window's representation $\mathbf{h}_i^t$, where $\mathbf{h}_i^0$ is from the output of the first module, f_{CNN}. Specifically, each $\mathbf{h}_i^t$ is revised using a parameterized summation of neighbors, $\mathbf{h}_j^t, j \in \mathcal{N}(i)$:

$$\mathbf{h}_i^{t+1} = \sigma\left(\frac{1}{|\mathcal{N}(i)|}\sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^t \mathbf{W}^t\right), \tag{2}$$

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 the GCN layer t. Importantly, using the summation in Eq. 2, the representation of each DNA window $\mathbf{x}_i$ is updated based on the representation of its neighbors (windows that interact with $\mathbf{x}_i$ in the 3D genome). We can compute the simultaneous update of all windows together by concatenating all $\mathbf{h}_i$ denoted $\mathbf{H}^t \in \mathbb{R}^{N \times d}$ where N is the number of DNA windows and d is the dimension of each $\mathbf{h}_i$. The simultaneous update can then be written as:

$$\mathbf{H}^{t+1} = \sigma(\mathbf{A}'\mathbf{H}^t\mathbf{W}^t). \tag{3}$$

where $\mathbf{A}' = \hat{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\hat{\mathbf{D}}^{-\frac{1}{2}}$, is the normalized adjacency matrix and $\hat{\mathbf{D}}$ is the diagonal degree matrix of $(\mathbf{A} + \mathbf{I})$.

In our experiments, we use a variant of the graph convolutional network, which uses a gating function allowing the model to use or not use neighboring windows to update each $\mathbf{h}_i$:

$$\widetilde{\mathbf{H}}^t = \tanh\left(\mathbf{A}'\mathbf{H}^t\mathbf{W}_z^t\right) \tag{4}$$

$$\mathbf{G}^t = \text{sigmoid}\left(\widetilde{\mathbf{H}}^t\mathbf{w}_g^t\right) \tag{5}$$

$$\mathbf{H}^{t+1} = \mathbf{G}^t \odot \widetilde{\mathbf{H}}^t + (1 - \mathbf{G}^t) \odot \mathbf{H}^t \tag{6}$$

where $\mathbf{W}_z^t$ is a linear transformation matrix, and $\mathbf{w}_g^t \in \mathbb{R}^d$ is used to compute the gate. $\mathbf{G}^t$ 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 Hadamard product. Equations 3-6 indicate one layer update of GCN window embeddings $\mathbf{H}$. In our experiments, we use a 2-layer gated GCN, and we denote the final output of all $\mathbf{H}^t$ as $\mathbf{Z}$, where vector $\mathbf{z}_i$ of $\mathbf{Z}$ represents the output of window i. In summary, the GCN module computes the following: $\mathbf{Z} = f_{GCN}(\mathbf{H}, \mathbf{A})$.

4.3 Predicting Label Probabilities for Each Window

After $\mathbf{Z}$ is computed, we then use a linear classifier layer, f_{Pred} to classify each $\mathbf{z}_i$ into its output space (a set of epigenetic labels): $\hat{\mathbf{y}}_i = \sigma(\mathbf{z}_i^\top \mathbf{W} + \mathbf{b})$. In summary, the prediction $\hat{\mathbf{y}}_i$ for a particular input sample $\mathbf{x}_i$ can be decomposed as three steps:

$$\begin{aligned}\hat{\mathbf{y}}_i &= f_{Pred}(\mathbf{z}_i)\\ \mathbf{Z} &= f_{GCN}(\mathbf{H}, \mathbf{A})\\ \mathbf{h}_i &= f_{CNN}(\mathbf{x}_i)\end{aligned}$$

4.4 Identifying and Interpreting Important Hi-C Edges via Hi-C Saliency Maps

One benefit of the ChromeGCN formulation is that we use explicit long-range window dependencies for epigenetic state prediction. As a result, we propose a method to identify the important dependencies for ChromeGCN's predictions. We call our proposed method Hi-C saliency maps. Saliency maps were introduced by [27] to understand the importance of each pixel in an input $\mathbf{x}_i$ for the prediction of the image's true class. We instead are trying to understand the importance of each edge in $\mathbf{A}'$ for the prediction of the epigenetic

[2] In our experiments, we use a different adjacency matrix $\mathbf{A}$ for each chromosome (intra-chromosome Hi-C maps). However, we generalize a $\mathbf{A}$ to represent all possible window interactions (i.e., including inter-chromosome maps).

state over all windows. The $\mathbf{A}'$ saliency map is defined as the absolute value gradient of a true class prediction $\hat{y}_i^\ell$ with respect to $\mathbf{A}'$, where ℓ is a true class for sample i. The absolute value gradient is then element-wise multiplied by $\mathbf{A}'$ to zero out "non-edges". Since there are N samples, and there can be multiple true labels ℓ for a particular sample $\hat{\mathbf{y}}_\mathbf{i}$, we define the Hi-C saliency map, $S_{Hi\text{-}C}$, as the accumulated absolute gradient over all samples and true labels w.r.t $\mathbf{A}'$:

$$S_{Hi\text{-}C} = \sum_{i=1}^{N} \sum_{\ell \in \hat{\mathbf{y}}_\mathbf{i}} \mathbf{A}' \circ \left| \frac{\partial \hat{y}_i^\ell}{\partial \mathbf{A}'} \right|, \tag{7}$$

where $\circ$ is the Hadamard product. Since the saliency map of all windows N are accumulated, we normalize $S_{Hi\text{-}C}$ across each row to a 0-1 range so that we can interpret the edges at each window.

While Hi-C contact maps tell us *where* the contacts are, Hi-C saliency maps show us *how important* each contact is for the epigenetic states. We define Eq. 7 to be over all labels, but we can easily visualize the Hi-C saliency, or important edges for one particular label. We show both the full Hi-C saliency across all labels, as well as for one specific label (YY1) in the experiments.

4.5 Model Variations

To test the effectiveness of the long range dependencies, we use the following ChromeGCN variations. Each variation uses the same model, with different edge dependencies in the form of $\mathbf{A}$.

ChromeGCN$_{const}$: Instead of using Hi-C edges we use a constant set of nearby neighbors according to the 1D sequential DNA representation. We define each window $\mathbf{x}_i$'s neighbors to be the windows surrounding $\mathbf{x}_i$ (7 on each side: $\mathbf{x}_{i-7}, ..., \mathbf{x}_{i+7}$) which we denote as $\mathbf{A}_{const}$. $\mathbf{Z} = GCN(\mathbf{A}_{const}, \mathbf{H})$. This variant allows us to see whether the very long range interactions from the normalized Hi-C maps are useful.

ChromeGCN$_{Hi\text{-}C}$: This variation uses the original Hi-C adjacency matrix, $\mathbf{A}_{Hi\text{-}C}$. $\mathbf{Z} = GCN(\mathbf{A}_{Hi\text{-}C}, \mathbf{H})$. Since we used use the top 500k Hi-C edges using a the SQRTVC normalization, many of these edges are far away in the 1D space.

ChromeGCN$_{const+Hi\text{-}C}$: Lastly, we use a combination of the constant neighborhood around each window and the Hi-C adjacency matrix, which integrates close and far windows for each window. This varition uses the following function: $\mathbf{Z} = GCN(\mathbf{A}_{const+Hi\text{-}C}, \mathbf{H})$.

4.6 Model Details and Training

To circumvent GPU memory constraints of training end-to-end, we pretrain the f_{CNN} model by classifying each $\mathbf{h}_i$ with the classification function $\hat{\mathbf{y}}_i = f_{Pred}(\mathbf{h}_i)$. Once the pretraining converges on the training set, we use the trained weights $\mathbf{h}_i$ for each sample as fixed inputs to f_{GCN}. While we pretrain f_{CNN}, ChromeGCN is still end-to-end differentiable, making it possible to use sequence visualization methods such as DeepLIFT [26] for a particular window.

For all model predictions, we run the forward and the reverse complement through simultaneously and average the output of the two. All DNA window inputs are encoded using a lookup table that maps each character A, C, G, T, and N (unknown) to a d-dimensional vector. The output of the encoding is a $d \times \tau$ matrix, whereτ denotes sequence length ($\tau = 2000$ in our experiments).

All of our models are trained using stochastic gradient descent with momentum of 0.9 and a learning rate of 0.25. The CNN model is trained using a batch size of 64, and the GCN and RNN models are trained using an entire chromosome as a batch (since each is modeling the between window dependencies of an entire chromosome at once). The CNN model projects each window to a vector of dimension 128. The GCN uses two layers of feature dimension 128 at each layer.

ChromeGCN predicts the probabilities of all labels for each window: $\hat{\mathbf{y}}_i \in \mathbb{R}^L$, where L is the total number of labels. For our loss function, we use the mean binary cross-entropy across all samples N and labels L:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{M} \sum_{i=1}^{N} \frac{1}{L} \sum_{l=1}^{L} - \left(y_i^l \log \left(\hat{y}_i^l \right) + \left(1 - y_i^l \right) \log \left(1 - \hat{y}_i^l \right) \right) \tag{8}$$

	GM12878			K562		
	Mean AUROC	Mean AUPR	Mean Recall at 50% FDR	Mean AUROC	Mean AUPR	Mean Recall at 50% FDR
CNN [35]	0.895	0.350	0.293	0.894	0.325	0.265
DanQ [22]	0.886	0.348	0.290	0.900	0.343	0.290
ChromeRNN	0.906	0.384	0.342	0.910	0.365	0.327
ChromeGCN$_{const}$	0.904	0.377	0.331	0.904	0.358	0.321
ChromeGCN$_{Hi\text{-}C}$	0.904	0.385	0.341	0.903	0.358	0.319
ChromeGCN$_{const+Hi\text{-}C}$	**0.909**	**0.395**	**0.356**	**0.912**	**0.372**	**0.338**

Table 2: Performance results. For both cell lines, GM12878 and K562, we show the average across all labels for three different metrics. Our method, using a graph convolutional network (GCN) to model long range dependencies helps improve performance over the baseline CNN model which assumes all DNA segments are independent. Detailed results of all cell lines and label types are shown in the Appendix.

5 Experiments and Results

5.1 Baselines

We compare against the state-of-the-art epigenetic state prediction model from [35], referred to as the CNN baseline, as well as the recurrent model from [22]. Since our model outputs labels for TFBS, HMs, and accessibility, motif-based methods [4] aren't applicable. Since we have 368,082 training samples, kernel-based methods such as [10] aren't applicable. [36] compared their CNN to a modified version of [10], which only used a small number of training samples, and the CNN model was significantly better. Our CNN baseline, [35] is an improved version from [36]. Furthermore, the focus of our study is to show that state-of-the-art deep learning models are missing important long range dependencies in the genome.

CNN [35]: To illustrate the importance of the GCN, we compare against the outputs from the f_{CNN} module: $\hat{\mathbf{y}}_i = f_{Pred}(\mathbf{h}_i)$. This is the 6-layer CNN model from [35] (we modify the last layer in order to extract a d dimensional feature vector output). This is the same CNN that is pretrained for ChromeGCN to produce each $\mathbf{h}_i$.

DanQ [22]: This model uses a recurrent neural network (RNN) on top of CNN outputs within a window. It still uses local sequence window inputs, but models relationships between sequence patterns via an LSTM.

ChromeRNN: As a baseline to compare against using GCNs for long range dependency modelling, we construct an RNN model on the window embeddings $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N$. After pretraining the CNN module f_{CNN}, The RNN model takes in all window embeddings at once and models the sequential dependencies among windows: $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, ..., \mathbf{z}_N) = f_{RNN}(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, ..., \mathbf{h}_N)$. As with ChromeGCN, the RNN is shared across chromosomes, but does not cross chromosomes. In other words, the embeddings are updated one chromsome at a time $f_{RNN}(\mathbf{h}_i, \mathbf{h}_{i+1}, ..., \mathbf{h}_C)$ where C is the total number of windows for a chromosome. We note this is different from the DanQ baseline [22] which uses an RNN *within* windows [22]. ChromeRNN instead is for modeling dependencies *between* windows. In our experiments, we use an LSTM [15] with the same number of layers and hidden units as the GCN.

5.2 Prediction Performance

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 2 shows the mean metric results across all epigenetic labels for each cell line. Modeling long-range dependencies results in significant improvements over the baseline CNN model, which does not account for such long range interactions. For instance, with respect to AUPR for GM12878, ChromeRNN improves upon the CNN from 0.350 to 0.384, and ChromeGCN outperforms ChromeRNN to achieve a mean AUPR of 0.395. Also, we see that the ChromeRNN outperforms DanQ, indicating that using an RNN to model between window dependencies is more important than within window features. Moreover, we can see that the ChromeGCN$_{Hi\text{-}C}$ models outperform ChromeRNN, indicating that not only the closely neighboring windows in the 1D space contribute to the improvements, but also the close neighbors in the 3D space, as indicated by the used Hi-C maps. ChromeGCN outperforms the baselines on the TF and DNase I labels, and ChromeRNN outperforms all other methods on the HM labels. This indicates that non-local modeling is particularly important for TF

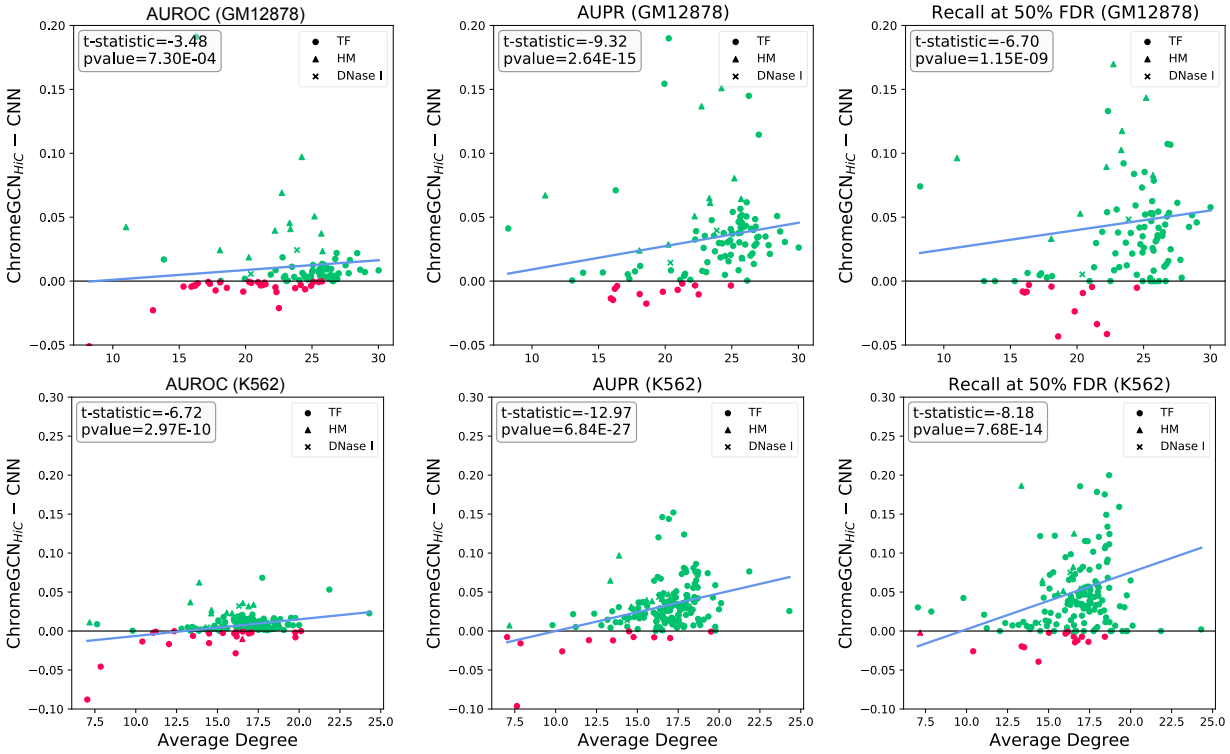

Fig. 2: Comparison of our ChromeGCN$_{Hi\text{-}C}$ method vs the baseline CNN [35] for 3 Metrics. Each point represents one epigenetic state label. The labels are sorted in the x-axis by the average degree of their positive windows. The y-axis indicates absolute increase of the ChromeGCN$_{Hi\text{-}C}$ over the CNN model. As the average degree increases, the improvement of the ChromeGCN$_{Hi\text{-}C}$ model increases over the CNN. Green points indicate ChromeGCN$_{Hi\text{-}C}$ performed better, red indicate the CNN performed better. The blue line shows the linear trend line. The ChromeGCN$_{Hi\text{-}C}$ is significantly better, as demsonstrated by the pvalues from a pairwise t-test.

binding and accessibility. We provide the performance results of each label type (TF, HM, DNase I) are shown in Appendix Tables 3 and 4. We also provide detailed plots of ROC and Precision-Recall curves in Appendix Figures 5, 6, 7, and 8.

Furthermore, since ChromeGCN models the window relationships explicitly and not recurrently, we obtain a significant speedup at test time over the ChromeRNN baseline. ChromeGCN achieves a 6.3x speedup on the three GM12878 test chromosomes, and 6.8x speedup at test time on the three K562 test chromosomes.

5.3 Analysis of Using Hi-C Data

Fig. 2 shows a a detailed comparison of ChromeGCN$_{Hi\text{-}C}$ vs the baseline CNN model across three different metrics for both GM12878 and K562. Each point represents a label, and the y-axis shows the absolute improvement of the ChromeGCN$_{Hi\text{-}C}$ model over the CNN. The labels are sorted on the x-axis by the average degree of the label's positive samples (i.e., windows where the label is positive) on the Hi-C map. We can see that for all three metrics, the improvements of the ChromeGCN over the CNN increase as the average degree of the labels increase. This indicates that the ChromeGCN is important for labels that have many neighbors in the Hi-C graph (i.e., those that are frequently in contact with other segments in the 3D space). Two of the transcription factors which obtain the highest performance increase (in the top 5) from using ChromeGCN over CNN, CEBPB STAT3, are validated by [21], which show that these two TFs commonly co-occur with other TFs in the 3D space when binding.

The p-values shown are computed by a pairwise t-test across all labels. The ChromeGCN$_{Hi\text{-}C}$ 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.

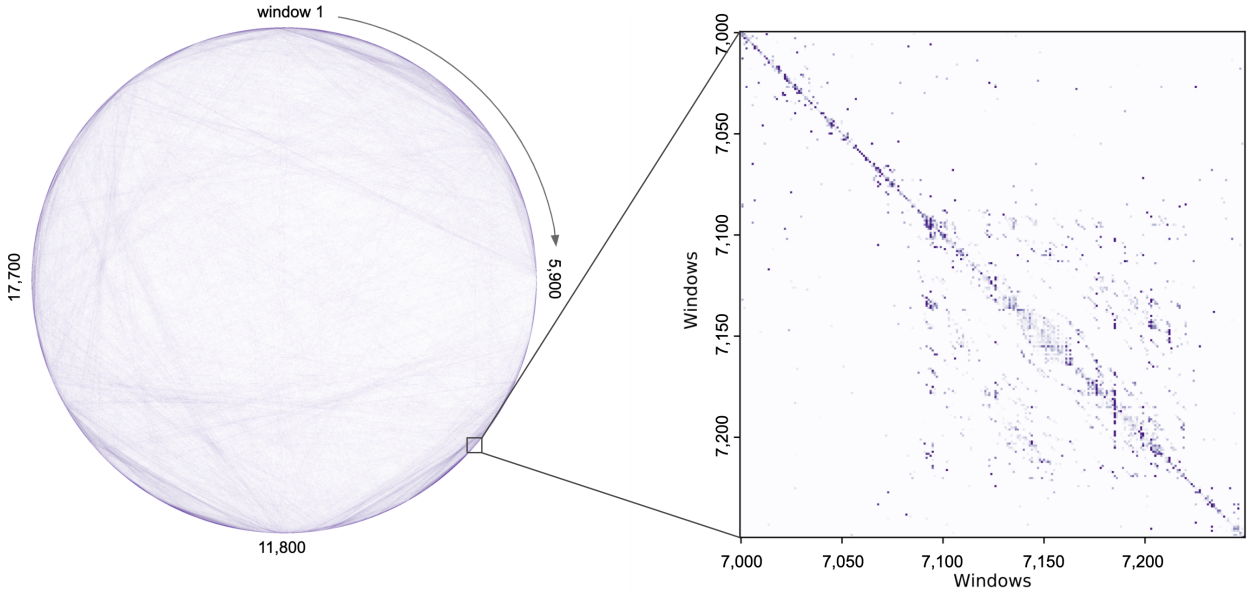

Fig. 3: **Hi-C Saliency Map Visualization.** *Left:* Saliency Map for the all 500k edges in $\mathbf{A}_{Hi\text{-}C}$ for GM12878 Chromosome 8 (total of 23,600 windows). The darker the line, the more important that edge was for predicting the correct Epigenetic state, indicating that the Hi-C data was used by the GNN for that particular interaction. *Right:* Fine grained analysis of the Chromosome 8 Saliency Map. This figure shows the normalized Saliency Map values for for 250 windows (total of 250kbp input) in chromosome 8.

5.4 Long Range Interaction Visualization

A significant merit of ChromeGCN is that by using known 3D genome relationships, we can find and visualize the critical relationships for epigenetic state prediction. To understand how important the Hi-C edges are for the predictions of ChromeGCN, we visualize the saliency map of $\mathbf{A}'$, as explained in Section 4.4.

Fig. 3 shows the Hi-C saliency map for chromosome 8 in GM12878. Fig. 3 (left) shows all 500k Hi-C contacts used chromosome 8. Windows are represented as points along the circle, with a total of 23,600 windows. Lines between the windows represent Hi-C edges, and the darkness of the line represents the saliency, or importance of that edge for chromatin state prediction across all windows in chromosome 8. Fig. 3 (right) shows the saliency map for 250 windows (total of 250kbp input) in chromosome 3. Cell (i, j) tells us the importance of window column j for the prediction of window row i labels.

While Fig 3 shows the Hi-C saliency map for all epigenetic state labels, we can also visualize the Hi-C saliency map for individual labels. The inner loop of Eq. 7 changes to only use the label of interest. In Appendix Fig 4, we show the Hi-C saliency map for the YY1 transcription factor on GM12878 chromosome 8. This gives us insight into the important 3D contacts for YY1 binding.

6 Conclusion

In this work, we present ChromeGCN, a novel framework that combines both local sequence and long-range 3D genome data (via Hi-C data) for epigenetic state prediction. We show experimentally that ChromeGCN outperforms previous state-of-the-art methods that only use local sequence data. Additionally, we show that we can identify and visualize the important 3D genome dependencies using the proposed Hi-C saliency maps. We plan to further investigate the value of Hi-C saliency maps in future work.

While we demonstrate the importance of ChromeGCN on the task of epigenetic state prediction, ChromeGCN is a generic model for incorporating 3D genome structure into any genome sequence prediction task. We hope that this work encourages researchers to use known long-range relationships from 3D genome data when constructing machine learning models. We plan to release our data and code for greater visibility. ChromeGCN introduces an effective and efficient framework to model such relationships for better chromatin modeling, as well as an easy way to interpret important relationships.

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. Swneke D Bailey, Xiaoyang Zhang, Kinjal Desai, Malika Aid, Olivia Corradin, Richard Cowper-Sal, Batool Akhtar-Zaidi, Peter C Scacheri, Benjamin Haibe-Kains, Mathieu Lupien, et al. Znf143 provides sequence specificity to secure chromatin interactions at gene promoters. *Nature communications*, 6:6186, 2015.
4. Timothy L Bailey, Mikael Boden, Fabian A Buske, Martin Frith, Charles E Grant, Luca Clementi, Jingyuan Ren, Wilfred W Li, and William S Noble. Meme suite: tools for motif discovery and searching. *Nucleic acids research*, 37(suppl_2):W202–W208, 2009.
5. Chris A Brackley, Mike E Cates, and Davide Marenduzzo. Facilitated diffusion on mobile dna: configurational traps and sequence heterogeneity. *Physical review letters*, 109(16):168103, 2012.
6. ENCODE Project Consortium et al. The encode (encyclopedia of dna elements) project. *Science*, 306(5696):636–640, 2004.
7. Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning*, pages 2702–2711, 2016.
8. 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.
9. 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.
10. Mahmoud Ghandi, Dongwon Lee, Morteza Mohammad-Noori, and Michael A Beer. Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS computational biology*, 10(7):e1003711, 2014.
11. Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.
12. Ofir Hakim and Tom Misteli. Snapshot: chromosome conformation capture. *Cell*, 148(5):1068–e1, 2012.
13. William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.
14. Hamid Reza Hassanzadeh and May D Wang. Deeperbind: Enhancing prediction of sequence specificities of dna binding proteins. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 178–183. IEEE, 2016.
15. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
16. 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.
17. Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
18. 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.
19. Jack Lanchantin, Ritambhara Singh, Zeming Lin, and Yanjun Qi. Deep motif: Visualizing genomic sequence classifications. *arXiv preprint arXiv:1605.01133*, 2016.
20. Jack Lanchantin, Ritambhara Singh, Beilun Wang, and Yanjun Qi. Deep motif dashboard: Visualizing and understanding genomic sequences using deep neural networks. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017*, pages 254–265. World Scientific, 2017.
21. Xiaoyan Ma, Daphne Ezer, Boris Adryan, and Tim J Stevens. Canonical and single-cell hi-c reveal distinct chromatin interaction sub-networks of mammalian transcription factors. *Genome biology*, 19(1):174, 2018.
22. 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.
23. 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.
24. 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.
25. Jacob Schreiber, Maxwell Libbrecht, Jeffrey Bilmes, and William Noble. Nucleotide sequence and dnasei sensitivity are predictive of 3d chromatin architecture. *bioRxiv*, page 103614, 2018.

26. Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3145–3153. JMLR. org, 2017.
27. Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
28. Ritambhara Singh, Arshdeep Sekhon, Kamran Kowsari, Jack Lanchantin, Beilun Wang, and Yanjun Qi. Gakco: a fast gapped k-mer string kernel using counting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 356–373. Springer, 2017.
29. Gary D Stormo. Dna binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, 2000.
30. 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.
31. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
32. 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.
33. Wikipedia contributors. Chromosome conformation capture — Wikipedia, the free encyclopedia, 2019. [Online; accessed 5-November-2019].
34. Ka-Chun Wong. Motifhyades: expectation maximization for de novo dna motif pair discovery on paired sequences. *Bioinformatics*, 33(19):3028–3035, 2017.
35. 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.
36. Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931, 2015.
37. Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

7 Appendix

7.1 YY1 Hi-C Saliency Map

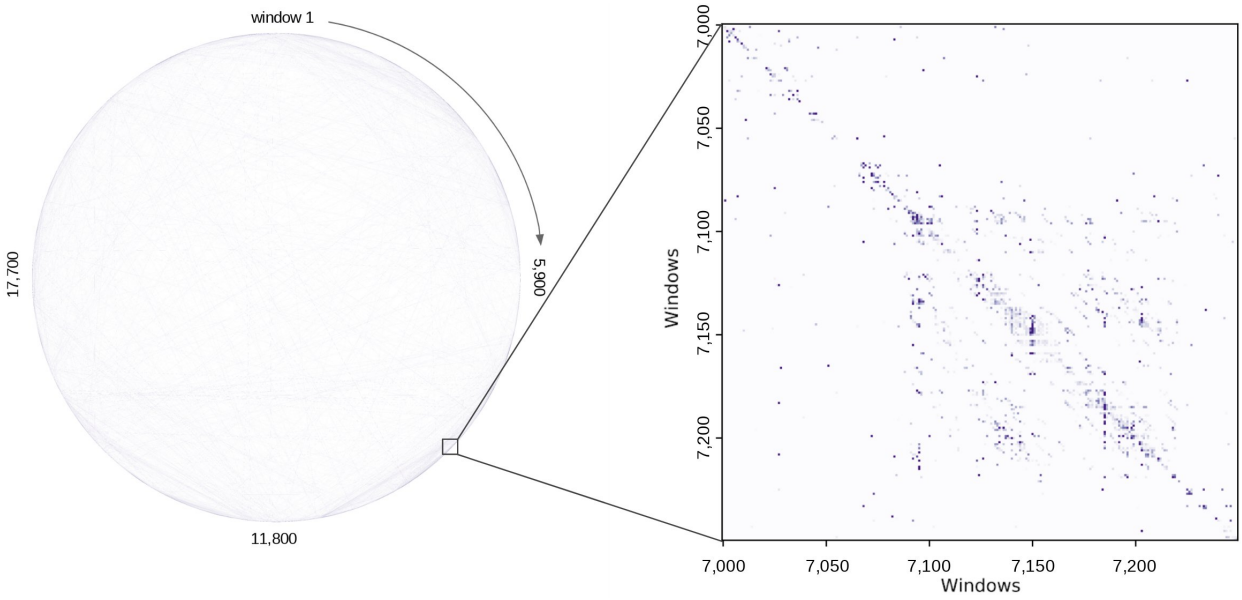

Fig. 4: Hi-C Saliency Map for the YY1 transcription factor on GM12878 Chromosome 8.

7.2 Additional Performance Tables and Figures

	TFs			HMs			DNase I		
	Mean AUROC	Mean AUPR	Mean Recall at 50% FDR	Mean AUROC	Mean AUPR	Mean Recall at 50% FDR	Mean AUROC	Mean AUPR	Mean Recall at 50% FDR
CNN [35]	0.909	0.328	0.262	0.796	0.478	0.469	0.801	0.657	0.750
DanQ [22]	0.899	0.325	0.260	0.794	0.479	0.463	0.792	0.650	0.723
ChromeRNN	0.914	0.354	0.299	**0.859**	**0.574**	**0.610**	**0.821**	0.689	**0.787**
ChromeGCN$_{const}$	0.913	0.348	0.291	0.849	0.554	0.58	0.815	0.68	0.777
ChromeGCN$_{Hi\text{-}C}$	0.916	0.362	0.309	0.819	0.516	0.528	0.814	0.683	0.774
ChromeGCN$_{const+Hi\text{-}C}$	**0.918**	**0.369**	**0.319**	0.845	0.552	0.584	0.820	**0.692**	0.783

Table 3: Performance on GM12878 for each label category

	TFs			HMs			DNase I		
	Mean AUROC	Mean AUPR	Mean Recall at 50% FDR	Mean AUROC	Mean AUPR	Mean Recall at 50% FDR	Mean AUROC	Mean AUPR	Mean Recall at 50% FDR
CNN [35]	0.905	0.314	0.256	0.78	0.410	0.317	0.792	0.625	0.683
DanQ [22]	0.909	0.331	0.279	0.796	0.438	0.365	0.797	0.638	0.693
ChromeRNN	0.917	0.349	0.305	**0.848**	**0.520**	**0.528**	0.817	0.664	0.742
ChromeGCN$_{const}$	0.911	0.342	0.301	0.84	0.507	0.507	0.810	0.657	0.729
ChromeGCN$_{Hi\text{-}C}$	0.912	0.346	0.306	0.807	0.454	0.412	0.812	0.657	0.730
ChromeGCN$_{const+Hi\text{-}C}$	**0.919**	**0.358**	**0.320**	0.837	0.495	0.484	**0.821**	**0.67**	**0.752**

Table 4: Performance on K562 for each label category

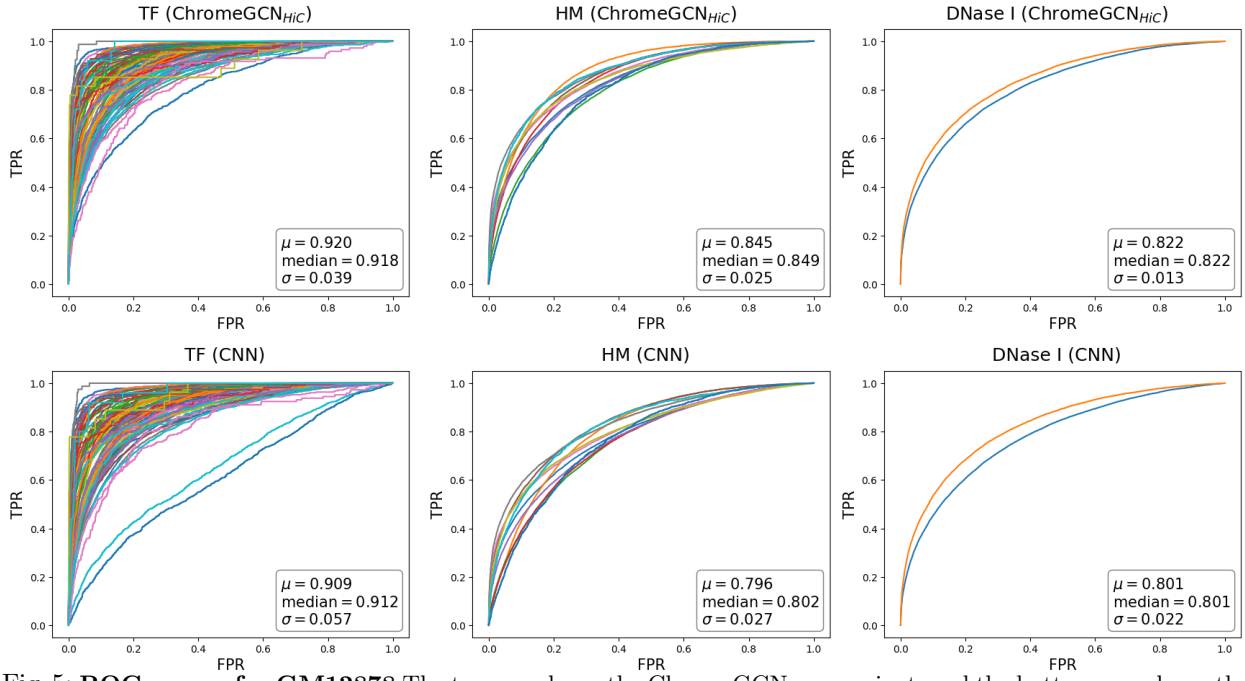

Fig. 5: **ROC curves for GM12878** The top row shows the ChromeGCN$_{Hi\text{-}C}$ variant, and the bottom row shows the CNN[35]. The columns are divided into the 3 types of labels: transcription factors (TFs), histone modificiations (HMs), and DNA accesibility (DNase I). The color of each curve represents a different label, where they are consistent across columns. The box in each plot shows the statistics of the area under the curves (AUC). ChromeGCN outperforms the CNN for all Epigenetic state labels.

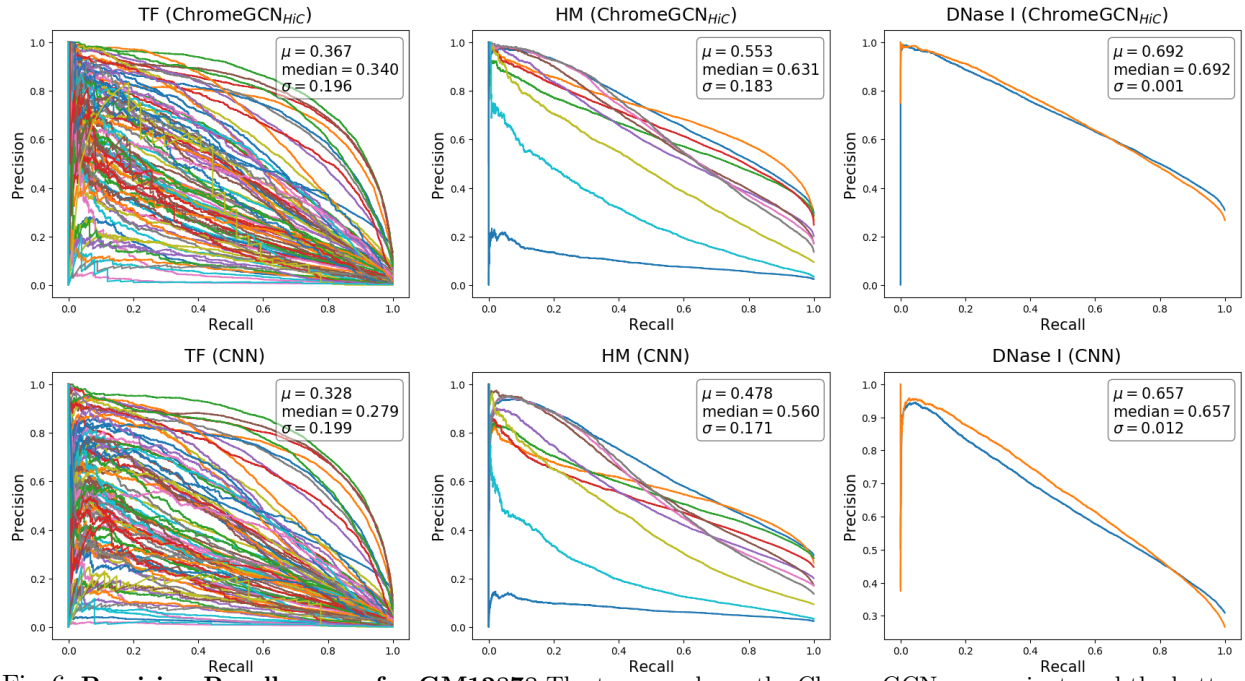

Fig. 6: **Precision-Recall curves for GM12878** The top row shows the ChromeGCN$_{Hi\text{-}C}$ variant, and the bottom row shows the CNN[35]. The columns are divided into the 3 types of labels: transcription factors (TFs), histone modificiations (HMs), and DNA accesibility (DNase I). The color of each curve represents a different label, where they are consistent across columns. The box in each plot shows the statistics of the area under the curves (AUC). ChromeGCN outperforms the CNN for all Epigenetic state labels.

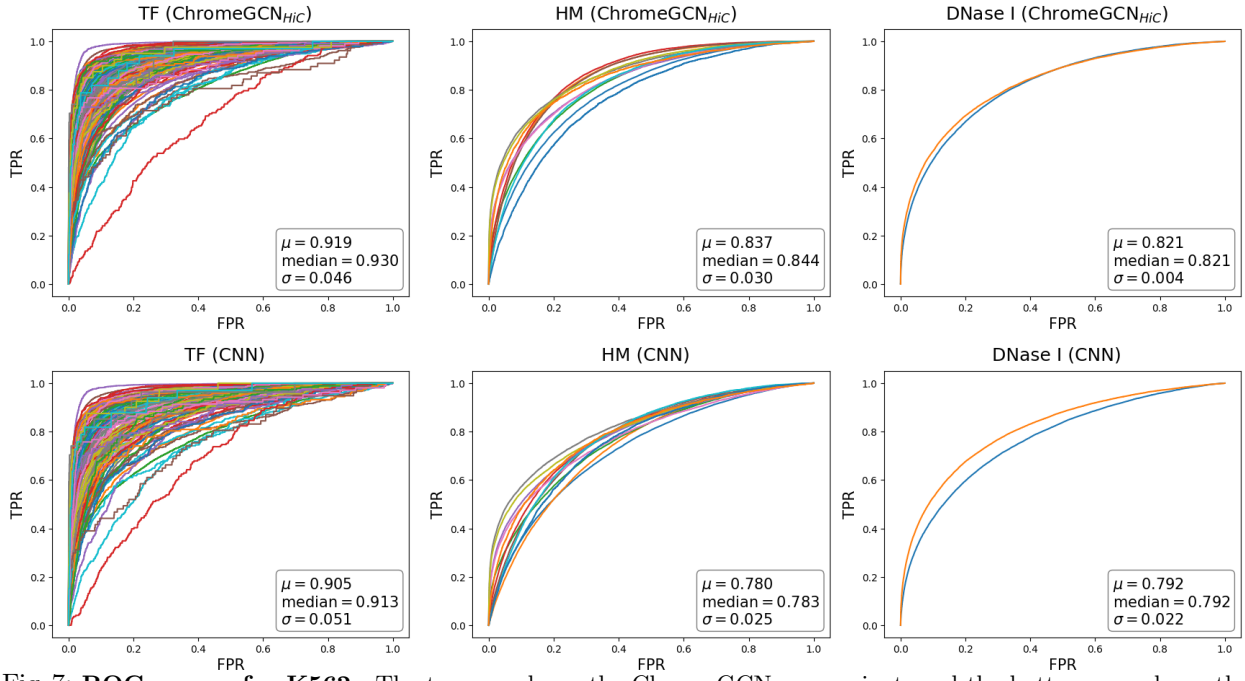

Fig. 7: **ROC curves for K562** . The top row shows the ChromeGCN$_{Hi\text{-}C}$ variant, and the bottom row shows the CNN[35]. The columns are divided into the 3 types of labels: transcription factors (TFs), histone modificiations (HMs), and DNA accesibility (DNase I). The color of each curve represents a different label, where they are consistent across columns. The box in each plot shows the statistics of the area under the curves (AUC). ChromeGCN outperforms the CNN for all Epigenetic state labels.

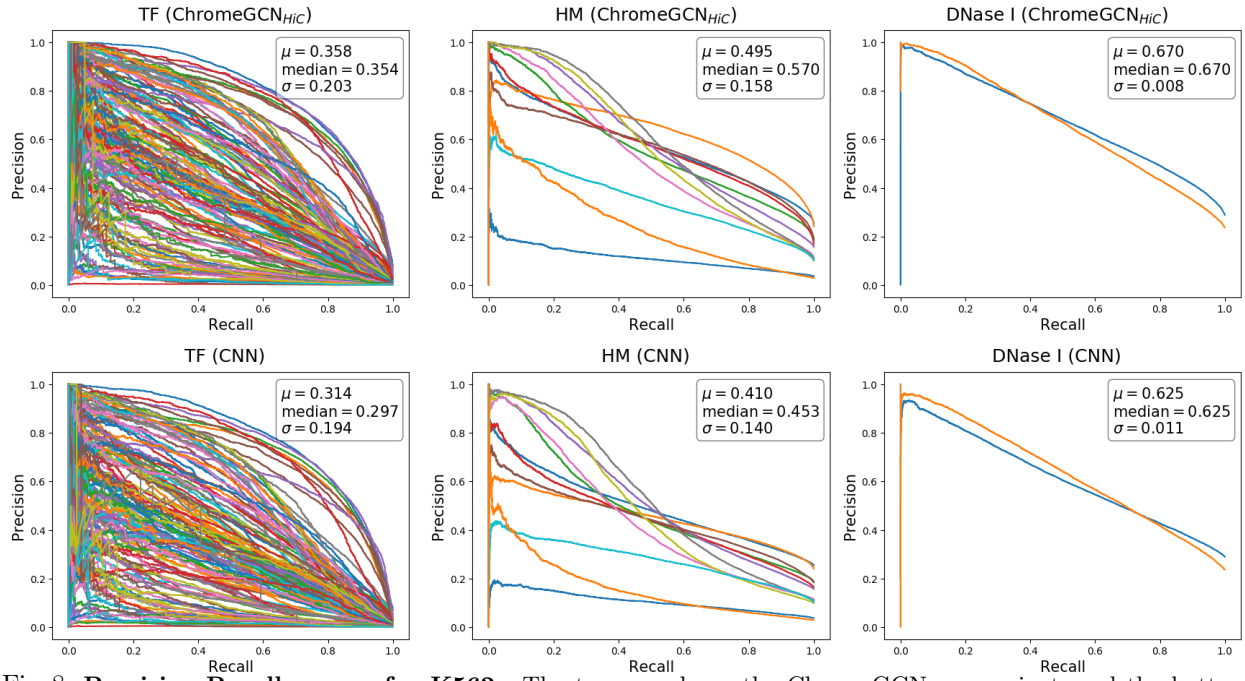

Fig. 8: **Precision-Recall curves for K562** . The top row shows the ChromeGCN$_{Hi\text{-}C}$ variant, and the bottom row shows the CNN[35]. The columns are divided into the 3 types of labels: transcription factors (TFs), histone modificiations (HMs), and DNA accesibility (DNase I). The color of each curve represents a different label, where they are consistent across columns. The box in each plot shows the statistics of the area under the curves (AUC). ChromeGCN outperforms the CNN for all Epigenetic state labels.

7.3 Data Details

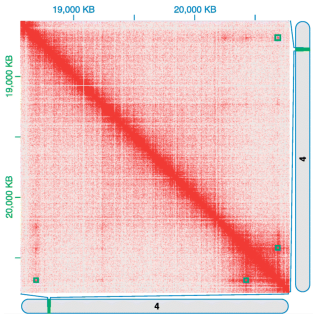

Fig. 9: Example Hi-C map showing the (chr4:18000000-21000000) region from GM12878 cell line [33]. Red blocks indicate DNA interactions, the darker red indicates more interactions. Our Hi-C saliency maps differ from Hi-C maps in that saliency maps tell us which contacts are *important*.

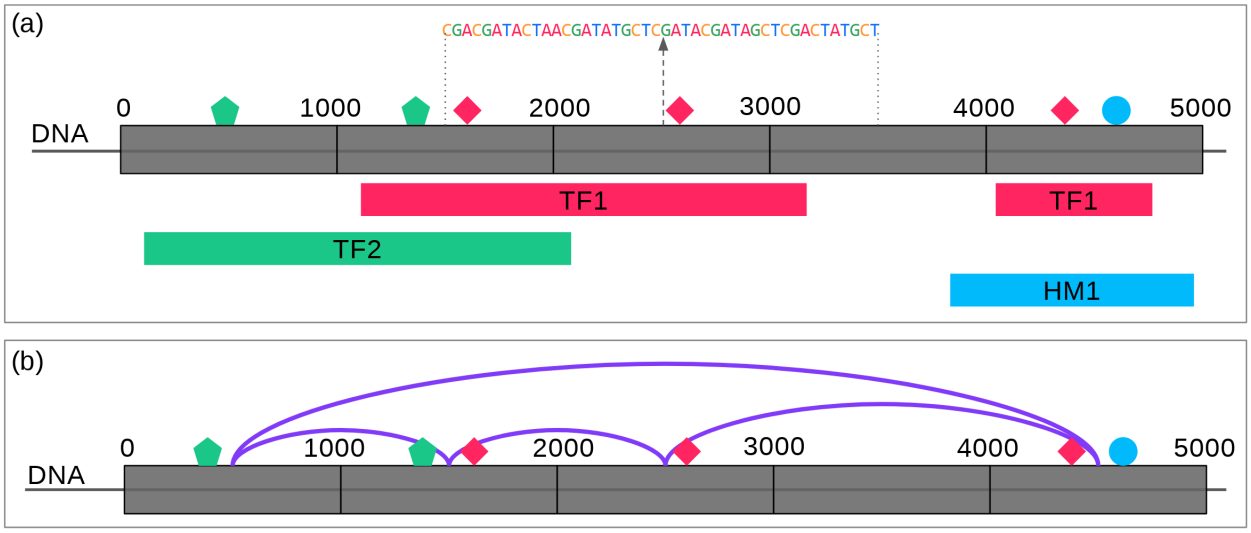

Fig. 10: **(a) Sequence Data Processing** We extract 2000bp sequences surround 1000bp windows for any window that has an overlapping ChIP-seq peak. **(b) 3D Genome Data Processing** we use Hi-C contacts between the 1000bp windows from [23] as edges in our 3D genome graph.