3 Approach

Previous computational methods failed to capture higher-order combinatorial effects among histone modifications, used bin-related strategies that cannot represent neighboring bins, or relied on multiple methods to separate prediction and combinatorial analysis. We utilize a deep convolutional neural network model for predicting gene expression from histone modification data. The network automatically learns both the combinatorial interactions and the classifier jointly in one unified discriminative framework, eliminating the need for human effort in feature engineering. Since the combinatorial effects are automatically learned through multiple layers of features, we present a visualization technique to extract those interactions and make the model interpretable.

3.1 Input generation

Aiming to systematically understand the relationship between gene regulation and histone modifications, we divided the 10 000 base-pair (bp) DNA region (±5000 bp) around the transcription start site (TSS) of each gene into bins of length 100 bp. Each bin includes 100 bp long adjacent positions flanking the TSS of a gene. In total, we consider five core histone modification marks from REMC database (Kundaje *et al.*, 2015), which are summarized in Table 2. These five histone modifications are selected as they are uniformly profiled across all cell-types considered in this study. This makes the input for each gene a 5×100 matrix, where columns represent different bins and rows represent histone modifications. For each bin, we report the value of all 5 histone signals as the input features for that bin (Fig. 1). We formulate the gene expression prediction as a binary classification task. Specifically, the outputs of DeepChrome are labels +1 and –1, representing gene expression level as high or low, respectively. Following Cheng *et al.* (2011), we use the median gene expression across all genes for a particular cell-type as a threshold to discretize the gene expression target. Figure 1 summarizes our input matrix generation strategy.

Our setup is similar to Cheng *et al.* (2011) and Dong *et al.* (2012), except that we primarily focus on the regions around TSS instead of also including regions from gene body or transcription termination site (TTS). This is based on the observations from Cheng *et al.* (2011) showing that signals close to the TSS are the most informative, therefore, eliminating the need to obtain bins from regions toward the end of the gene. In addition, due to the scalability of CNNs, we were able to use larger regions flanking TSS than previous studies in order to better capture effects of distal signals as well as to cover more regions. This, therefore, enhances the possibility to model long range interactions among histone modifications.

Table 2. Five core histone modification marks, as defined by Kundaje *et al.* (2015), along with their functional categories

Histone mark	Associated with	Functional category
H3K4me3	Promoter regions	Promoter mark
H3K4me1	Enhancer regions	Distal mark
H3K36me3	Transcribed regions	Structural mark
H3K9me3	Heterochromatin regions	Repressor mark
H3K27me3	Polycomb repression	Repressor mark

3.2 An end-to-end architecture based on convolutional neural network

Convolution Neural Networks (CNNs) were first popularized by LeCun *et al.* (1998) and have since been extensively used for a wide variety of applications. In this paper, we have implemented a CNN for gene expression classification task using the Torch7 (Collobert *et al.*, 2011) framework. Our DeepChrome model, summarized in Figure 2, is composed of five stages. We assume our training set contains N_{samp} gene samples of the labeled-pair form $(\mathbf{X}^{(n)}, y^{(n)})$, where $\mathbf{X}^{(n)}$ are matrices of size N_f (=5) $\times b$ (= 100) and $y^{(n)} \in \{-1, +1\}$ for $n \in \{1, \ldots, N_{\text{samp}}\}$.

1. Convolution: We use temporal convolution with N_{out} filters, each of length k. This performs a sliding window operation across all bin positions, which produces an output feature map of size $N_{\text{out}} \times (b - k + 1)$. Each sliding window operation applies N_{out} different linear filters on k consecutive input bins from position $p = 1$ to $(b - k + 1)$. In Figure 2, the red rectangle shows a sliding window operation with $k = 4$ and $p = 1$. Given an input sample $\mathbf{X}$ of size $N_f \times b$, the feature map, $\mathbf{Z}$, from convolution is computed as follows:

$$\begin{aligned} Z &= f_{conv}(\mathbf{X}) \\ \mathbf{Z}_{p,i} &= \mathbf{B}_i + \sum_{j=1}^{N_f} \sum_{r=1}^{k} \mathbf{W}_{i,j,r} \mathbf{X}_{p+r-1,j} \end{aligned} \tag{1}$$

This is generated for the pth sliding neighborhood window and the ith hidden filter, where $p \in \{1, \ldots, (b - k + 1)\}$ and $i \in \{1, \ldots, N_{\text{out}}\}$. $\mathbf{W}$, of size $N_{\text{out}} \times N_f \times k$, and $\mathbf{B}$, of size $N_{\text{out}} \times 1$, are the trainable parameters of the convolution layer and N_{out} denotes the number of filters.

2. Rectification: In this stage, we apply a non-linearity function called rectified linear unit (ReLU). The ReLU is an elementwise operation that clamps all negative values to zero:

$$f_{\text{relu}}(z) = \text{relu}(z) = \max(0, z) \tag{2}$$

3. Pooling: Next, in order to learn translational invariant features, we use temporal maxpooling on the output from the first two steps. Maxpooling simply selects the max values in a certain range, which forms a smaller representation of a large TSS-proximal region for a given gene. Maxpooling is applied on an input $\mathbf{Z}$ of size $N_{\text{out}} \times P$, where $P = (b - k + 1)$. With a pooling size of m, we obtain an output $\mathbf{V}$ of size $N_{\text{out}} \times \lfloor \frac{P}{m} \rfloor$:

$$\begin{aligned} \mathbf{V} &= f_{\text{maxpool}}(\mathbf{Z}) \\ \mathbf{V}_{i,p} &= \max_{j=1}^{m} \mathbf{Z}_{i,m(p-1)+j} \end{aligned} \tag{3}$$

where $p \in \{1, \ldots, \lfloor \frac{P}{m} \rfloor\}$ and $i \in \{1, \ldots, N_{\text{out}}\}$. In Figure 2, the blue rectangle shows the result of a maxpooling operation on the feature map where $m = 3$.

4. Dropout: The output is then passed through a dropout layer (Srivastava *et al.*, 2014), which randomly zeroes the inputs to the next layer during training with a chosen probability of 0.5. This regularizes the network and prevents over-fitting. It resembles

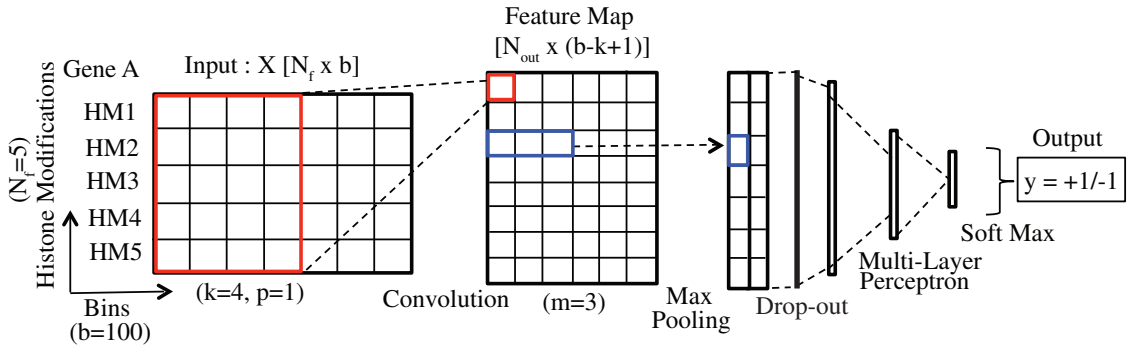

Fig. 2. DeepChrome convolution neural network (CNN) model. The input matrix X, comprising of 100 bins with signals from five histone modifications, goes through different CNN stages. These stages are: convolution, pooling followed by dropout, and MLP with alternating linear and non-linear layers. Softmax function, in the end, maps the output from the model into classification prediction

ensemble techniques, like bagging or model averaging, which are very popular in bioinformatics.

5. Classical feed-forward neural network layers: Next, the learnt region representation is fed into a MLP classifier to learn a classification function mapping to gene expression labels. This standard and fully connected MLP network has multiple alternating linear and non-linear layers. Each layer learns to map its input to a hidden feature space, and the last output layer learns the mapping from a hidden space to the output class label space $(+1/-1)$ through a softmax function. Figure 2, shows a MLP with 2 hidden layers and a softmax function at the end. This stage is represented as $f_{mlp}(.)$.

The whole network output form can be written as:

$$f(\mathbf{X}^{(n)}) = f_{mlp}(f_{\text{maxpool}}(f_{\text{relu}}(f_{\text{conv}}(\mathbf{X}^{(n)})))) \tag{4}$$

All the above stages are effective techniques that are widely practiced in the field of deep learning. All parameters, denoted as Θ, are learned during training in order to minimize a loss function which captures the difference between true labels y and predicted scores from $f(.)$ (When training this deep model, parameters, at first, are randomly initialized and input samples are fed through the network. The output of this network is a score prediction associated with a sample. The difference between a prediction output $f(\mathbf{X})$ and its true label y is fed back into the network through a 'back-propagation' step.). The loss function L, on the entire training set of size n, is defined:

$$L = \sum_{n=1}^{N_{\text{samp}}} \text{loss}(f(\mathbf{X}^{(n)}), y^{(n)}) \tag{5}$$

We use stochastic gradient descent (SGD) (Bottou, 2004) to train our model via backpropagation. For a set of training samples, instead of calculating the true gradient of the objective using all training samples, SGD calculates the gradient per sample and updates accordingly on each training sample. For our objective function, the loss $L(.)$ [Equation (5)] is minimized by the gradient descent step that is applied to update network parameters Θ as follows:

$$\Theta \leftarrow \Theta - \eta \frac{\partial L}{\partial \Theta} \tag{6}$$

where η is the learning rate (set to 0.001).

3.3 Visualizing combinatorial effect through optimization

In addition to being able to make high accuracy predictions on the gene expression task, an important contribution of DeepChrome is that it allows us to discover and visualize the combinatorial relationships between different histone modifications which lead to such predictions. Until recently, deep neural networks were viewed as 'black boxes' due to the automatically learned features spanning multiple layers. Since gene expression is dependent on the combinatorial interactions among histone modifications, it is critical to understand how the network extracts features and makes its predictions. In other words, we wish to understand the combinatorial patterns of histone modifications which lead to either a high or low gene expression prediction by the network. We attempt to do this by extracting a map of feature patterns that are highly influential in predicting gene expression directly from the trained network. This approach, called a network-centric approach (Yosinski *et al.*, 2015), finds the *class specific* features from the trained model and is independent of specific testing samples.

The technique we use to generate this visualization was inspired from works by Simonyan *et al.* (2013) and Yosinski *et al.* (2015), which seek to understand how a convolutional neural network interprets a certain image class on the task of object detection. We, instead, seek to find how our network interprets a gene expression class (high or low). Given a trained CNN model and a label of interest (+1 or −1) in our case, we perform a numerical optimization procedure on the model to generate a feature pattern map which best represents the given class. This optimization includes four major steps:

1. Randomly initialize an input $\mathbf{X}_c$ (of size $N_f(=5) \times b(=100)$).
2. Find the best values of entries in $\mathbf{X}_c$ by optimizing the following equation (7). We search for $\mathbf{X}_c$ so that the loss function is minimized with respect to the desired labels + 1 (gene expression = high) or −1 (gene expression = low). Using equation (4), $f(\mathbf{X}_c)$ provides the predicted label using the trained DeepChrome model on an input $\mathbf{X}_c$. We would like to find an optimal feature pattern, $\mathbf{X}_c$, so that its predicted label $f(\mathbf{X}_c)$ is close to the desired class label c:

$$\arg\min_{X_c} L_{\text{visual}} = \arg\min_{X_c} \{L(f(\mathbf{X}_c), y = c) + \lambda \|\mathbf{X}_c\|_2^2\} \tag{7}$$

where $c = +1$ or −1, $L(.)$ is the loss function defined in equation (5). L_2 regularization, $\|\mathbf{X}_c\|_2^2$, is applied to scale the signal values in $\mathbf{X}_c$, and λ is the regularization parameter. A locally optimal X_c can be found by the back-propagation method. This step is similar to the CNN training procedure, where back-propagation is used to minimize the loss function by optimizing the *network* parameters Θ. However, in this case, the optimization is performed with respect to the *input values* ($\mathbf{X}_c$) and the network parameters are fixed to the values obtained from the classification training. $\mathbf{X}_c$ is optimized in the following manner:

$$\mathbf{X}_c^{t+1} \leftarrow \mathbf{X}_c^t - \alpha \frac{\partial L_{\text{visual}}}{\partial \mathbf{X}_c} \quad (8)$$

where α is the learning rate parameter and t represents the iteration step of the optimization.

3. Next, we set all the negative output values to 0 and normalize $\mathbf{X}_c \in [0, 1]$:

$$\mathbf{X}_{c(\text{norm})} = \frac{\mathbf{X}_c}{\max(\mathbf{X}_c)} \quad (9)$$

4. Finally, we set a threshold of 0.25 to define 'active' bins. Bins in $\mathbf{X}_{c(\text{norm})}$ with values >0.25 are considered important since they indicate that such histone modification signals are important for predicting particular class. We count the frequency of these active bins for a particular histone modification mark. A high frequency count (>mean frequency count across all histone modification marks) of active bins indicates the important influence of these histone modification signals on target gene expression level.

This visualization technique represents the notion of a class that is learnt by the DeepChrome model and is not specific to a particular gene. The optimized feature pattern map $\mathbf{X}_{c(\text{norm})}$ is representative for a particular gene expression label of +1 (high) or −1 (low). In Figure 5, DeepChrome visualizes $\mathbf{X}_{c(\text{norm})}$ as heat-maps. Through these maps, we obtain intuitive outputs for understanding the combinatorial effects of histone modifications on gene regulation.

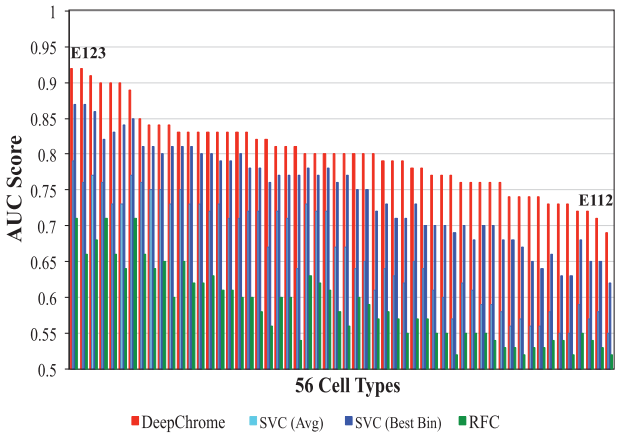

Fig. 3. Performance Evaluation on Test Set. (Best viewed in color) The bar graph represents AUC scores of DeepChrome versus state-of-the-art baseline models for 56 cell types (i.e. 56 different classification tasks). The results have been arranged from best performing cell type (E123) to the worst performing cell type (E112) for the test set (6600 genes). DeepChrome (Average AUC = 0.80) consistently outperforms both SVM (Average AUC: SVM Best Bin = 0.75 and SVM Avg = 0.66) and Random Forest Classifier (Average AUC = 0.59) for the task of binary classification of gene expression. SVM based baseline has a separate model for each bin (bin specific model), thus, results for both average AUC scores across all bins (SVM Avg) and best performing AUC score among the bins (SVM Best Bin) are presented

4 Experiment setup

4.1 Dataset

We downloaded gene expression levels and signal data for five core histone modification signals for 56 different cell types from the REMC database (Kundaje *et al.*, 2015). REMC is a public resource of human epigenomic data produced from hundreds of cell-types. Core histone modification marks, as defined by Kundaje *et al.* (2015), have been listed in Table 2 and are known to play important roles in gene regulation. We focus on these 'core' histone modifications as they have been uniformly profiled for all 56 cell types through sequencing technologies. The gene expression data has been quantified for all annotated genes in the human genome and has been normalized for all 56 cell types in the REMC database. As mentioned before, the target problem has been formulated as a binary classification task. Thus, each gene sample is associated with a label $+1/-1$ indicating whether gene expression is high or low, respectively. The gene expression values were discretized using the median of gene expressions across all genes for a particular cell-type.

4.2 Baselines

We compare DeepChrome to two baseline studies, Cheng *et al.* (2011) which uses Support Vector Machines (SVM) and Dong *et al.* (2012) which uses a Random Forest Classifier. Their implementation strategies are as follows:

- *SVM (Cheng et al., 2011):* The authors selected 160 bins from regions flanking the gene TSS and TTS. Each bin position uses a separate SVM classification model, resulting in 160 different models in total. This gave insights into important bin positions for classifying gene expression as high or low. Following this bin-specific model strategy, we provide results for performance of the best bin (SVM Best Bin) along with average performance across all bins (SVM Avg) in Section 5.1 and in Figure 3.
- *Random Forest Classifier (Dong et al., 2012):* In this study, bins were selected from regions flanking the TSS, TTS and gene body. This study selected the bin values having the highest correlation with gene expression as 'best bins'. A matrix with all genes and best bins for each histone modification signal was used as input into the model to predict gene labels $(+1/-1)$ as output. Since this baseline performs feature selection using the best bin strategy, our experiment uses the best-bin Random Forest performance as a baseline in Figure 3.

We implemented these baselines using the python-based scikit-learn (Pedregosa *et al.*, 2011) package.

4.3 Hyperparameter tuning

For each cell type, our sample set of total 19 802 genes was divided into 3 separate, but equal size folds: training (6601 genes), validation (6601 genes) and test (6600 genes) sets. We trained DeepChrome using the following hyperparameters: filter size ($k = \{10, 5\}$), number of convolution filters ($N_{\text{out}} = \{20, 50, 100\}$) and pool size for maxpooling ($m = \{2, 5\}$). Table 3 presents the validation set results for tuning different combinations of kernel size k and pool size m. k denotes the local neighborhood representations of flanking bins. m represents selected whole regions in our CNN model. We report the maximum, minimum and mean AUC scores obtained across 56 cell types. Performances of models using these different hyperparameter values did not vary significantly (P-value$\sim$0.92) from each other. We also trained a deeper model with 2 convolution layers and observed no significant (P-value= 0.939) increase in performance.

- We selected $k = 10$, $N_{\text{out}} = 50$ and $m = 5$ for training the final CNN models based on highest Max. and Min. AUC scores in Table 3. The number of hidden units chosen for the two MLP layers were 625 and 125, respectively. We trained the model for 100 epochs and observed that it converged early (around 15–20 epochs).
- For the SVM implementation, an RBF kernel was selected and the model was trained on varying hyperparameter values

Table 3. Results on **validation set** (6601 genes) during tuning across different combinations of kernel size *k* and pool size *m*

Kernel size, pool size (k, m)	AUC scores (validation set)		
	Max	Min	Mean
(5,2)	0.94	0.65	0.77
(5,5)	0.94	0.65	0.77
(10,2)	0.94	0.65	0.76
(10,5)	0.94	0.66	0.77

k captures the local neighborhood representations of bins and *m* combines the important representations across whole regions for our CNN model. We report the maximum, minimum and mean AUC score obtained across 56 cell types (or tasks). The best performing values of $k = 10$ and $m = 5$ (highest Max. and Min. AUC scores) were selected for evaluating test performance of DeepChrome.

of $C \in \{0.01, 0.1, 1, 10, 100, 1000\}$ and $\gamma \in \{0.01, 0.1, 1, 2\}$. The C parameter balances the tradeoff between misclassification of training examples and simplicity of the decision surface, while the γ parameter defines the extent of influence of a single training sample.

- For the Random Forest Classifier implementation, we varied the number of decision trees, $n_d \in \{10, 20, \ldots, 200\}$ trained in each model.

All the above models were trained on the training set, and the parameters for testing were selected based on their results on the validation set. We then applied the selected models on the test dataset. The AUC scores [Area Under Curve (AUC) score from Receiver Operating Characteristic (ROC) curve is interpreted as the probability that a randomly selected 'event' will be regarded with greater suspicion (in terms of its continuous measurement) than a randomly selected 'non-event'. AUC score ranges between 0 and 1, where values closer to 1 indicate more successful predictions.] (performance metric) are reported in Section 5.1.

5 Results

5.1 Performance evaluation

The bar graph in Figure 3 compares the performance of DeepChrome and three baselines on test dataset for gene expression classification across 56 different cell-types (or tasks). DeepChrome (Average AUC = 0.80) outperforms the baselines for all the cell types shown along the *X*-axis. As discussed earlier, Cheng *et al.* (2011) implement a different SVM model for each bin position. Therefore, we report both average AUC score for all the bins (SVM Avg) as well as the best AUC score among all bins (SVM Best Bin). 'SVM Best Bin' (Average AUC = 0.75) gives better results than 'SVM Avg' (Average AUC = 0.66). However, its AUC scores are still lower than those of DeepChrome. Random Forest gives the worst performance (Average AUC = 0.59). Additionally, we observe that the performances of all three models vary across different cell types and follow a similar trend. For some cell types, like E123, the prediction task resulted in higher AUC scores among all models than other cell types.

5.2 Validating the influence of bin positions on prediction

Cheng *et al.* (2011) obtained predictions for each bin (due to bin-specific strategy) and reported that, on average, the best AUC scores were obtained from bins that are close to the TSS. Figure 4(a) shows that our implementation of this SVM baseline confirms this observation. Since our convolutional network makes a prediction on the entire flanking region (i.e. all the bins at once), we cannot evaluate the AUC for each individual bin. However, we can roughly determine which bins are the most influential for a specific gene prediction. To do this, we look at the strongest activations among the output of the convolution step (the feature map, as shown in Fig. 2). Since the column in the feature map corresponds to the bins in the input region, we can simply look at the feature map values to determine which bin positions are most influential for that prediction. To validate our model, we ran all of our test samples through a trained deep network, and took the average of all the feature maps across all 56 models. Figure 4(b) shows that bins near the center, closer to TSS, are assigned with higher values by the convolution layers. This indicates that DeepChrome maintains similar trends as observed by Cheng *et al.* (2011). This trend indicates histone modification signals of bins that are closer to TSS are more influential in gene predictions.

5.3 Visualizing combinatorial interactions among histone modifications

In order to interpret the combinatorial interactions among histone modifications, we present a visualization technique in Section 3.3.

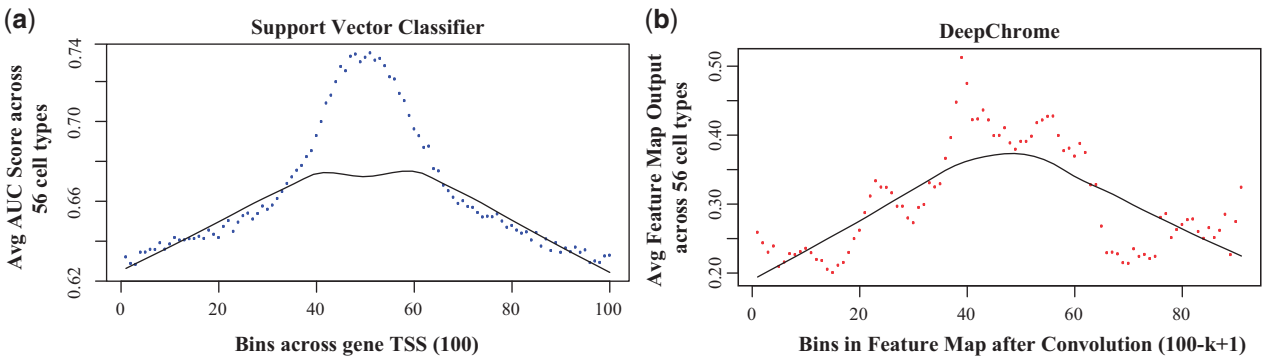

Fig. 4. Validating the influence of positions for gene expression classification. Cheng *et al.* (2011) reported that the bin positions closer to the transcription start site (TSS) of each gene are more important when predicting gene expression. This is confirmed by our implementation of this bin-specific baseline model in (**a**). For each bin position, it shows the mean AUC score across all the cell types. In (**b**), we plot the filter outputs from the convolution layer of DeepChrome model. For each bin, its value has been averaged across all filters and cell-types. The solid lines represent the best-curve fit to the data points plotted in the figures. The trends for both (a) and (b) are similar

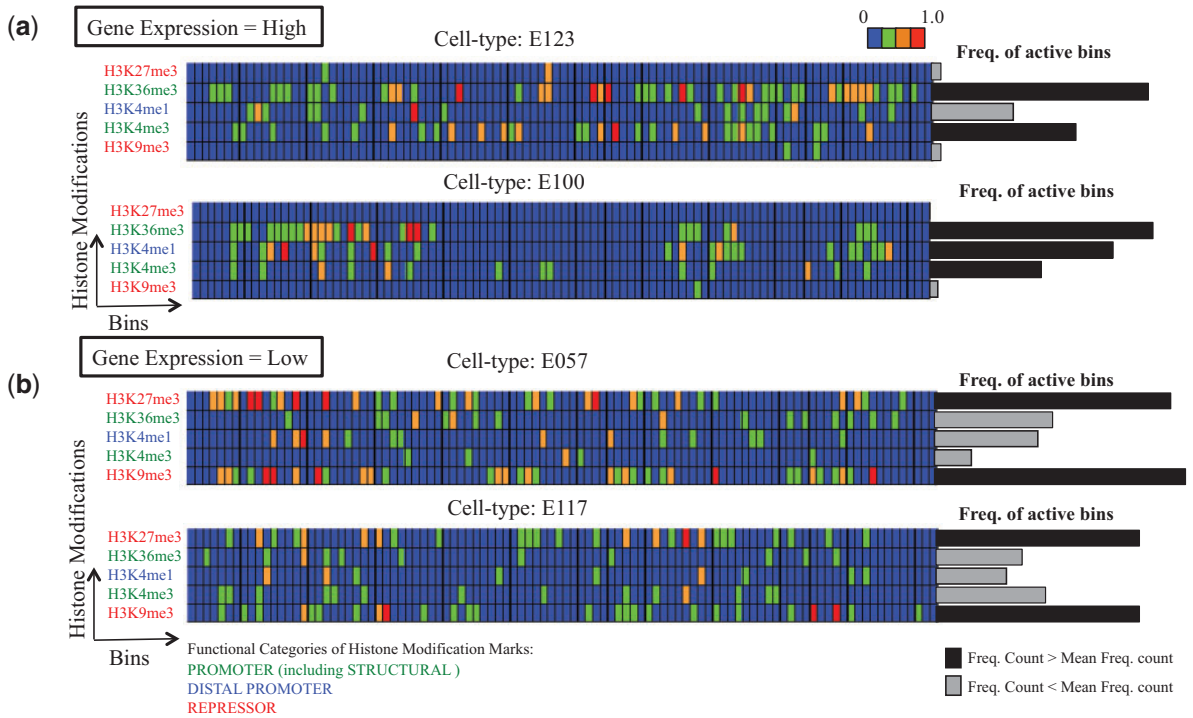

Fig. 5. DeepChrome visualization. (Best viewed in color) Four examples of feature maps generated by our optimization technique from four trained models. The scores in these feature maps are $\in [0, 1]$ and a threshold of 0.25 was selected to indicate 'active' (or important) bins. The bar graph represents the count of active bins for each histone modification. Higher frequency count (> mean frequency count across all histone marks) indicates greater influence of the histone modification mark in prediction of gene expression labels. Multiple marks with high frequency count are considered to be combinatorially affecting the gene expression to become high or low. (**a**) As expected, we observe a relationship among promoter and structural histone modification marks (H3K4me3 and H3K36me3) when gene expression is high. (**b**) Similarly, we observe an opposite trend with repressor marks (H3K9me3 and H3K27me3) showing combinatorial relationship, when gene expression is low. These pattern maps not only support previous quantitative observations in Cheng *et al.* (2011) and Dong *et al.* (2012), but also provide novel insights that are supported by recent biological studies. For example, a recent study by Boros *et al.* (2014) has reported evidence of coexistence of H3K27me3 and H3K9me3 modifications in gene silencing

Figure 5 presents four visualization results from DeepChrome on four cell types with high AUC scores. Each visualization result is a heat-map which shows the histone modification combinatorial pattern that is best representative of high (label = +1) or low (label = –1) gene expression. Note that this is different than Section 5.2 where we validated the importance of bin positions in general, rather than the combinatorial interactions for a specific class. The values in the heatmaps are within the range $[0, 1]$, representing how important a particular bin is for prediction of the class of interest. A threshold of 0.25 was selected to filter 'active', or important, bins that are most influential for a particular classification. We calculated the frequency count of active bins for each histone modification. Histone marks with high frequency counts (>mean frequency count across all histone marks) are considered to be strongly affecting the gene expression to become high or low. As expected, we observe a relationship among promoter and structural histone modification marks (H3K4me3 and H3K36me3) for 47 out of 56 (84%) cell-types when gene expression is high. Similarly, we observe an opposite trend with repressor marks (H3K9me3 and H3K27me3) showing combinatorial relationship for 50 out of 56 (89%) cell-types, when gene expression is low. In other words, our model automatically learns that in order to classify a high or low gene expression, there needs to be high counts among promoter marks, or repressor marks, respectively.

Next, we validated our visualization results with the findings in previous studies. Both of our baseline papers, Cheng *et al.* (2011) and Dong *et al.* (2012), showed that there is a combinatorial correlation between H3K4me3 (promoter mark) and H3K36me3 (structural mark). This pattern can be seen in Figure 5 for high gene expression cases. Similarly, Dong *et al.* (2012) also reported a combinatorial correlation between promoter mark (H3K4me3) and distal promoter mark (H3K4me1), which is also validated by the DeepChrome visualization for 35 out of 56 cell-types (62.5%). In addition, experimental studies have shown that these promoter marks play a role in the activation of genes, and this trend is seen in our visualization when the assigned label is +1.

Another combinatorial pattern that we noticed in the majority of cell-types (89%, i.e 50 out of 56 cell-types) was that of H3K9me3 (heterochromatin repressor) and H3K27me3 (polycomb repressor) for low gene expression case (label = –1). We found this observation in multiple recent biological studies such as Boros *et al.* (2014). This study reported that these two repressor marks coexist and cooperate in gene silencing. With almost no expert knowledge, we were able to find and visualize this combination through DeepChrome. To our knowledge, none of the previous computational studies have reported this combinatorial effect between H3K9me3 and H3K27me3. In short, the DeepChrome visualization technique provides the potential to learn novel insights

into combinatorial relationships among histone modifications for gene regulation.

6 Discussion

We have presented DeepChrome, a deep learning framework that not only accurately classifies gene expression levels using histone modifications as input, but also learns combinatorial relationships among these modification marks that regulate genes. We implement a Convolutional Neural Network based model, inspired from deep learning work in image recognition applications, and evaluate its performance on 56 cell types from the latest REMC dataset. To our knowledge, we are the first to implement deep learning on the task of gene expression classification using histone modification signals.

DeepChrome outperforms state-of-the-art models using SVM and Random Forests for the target task over 56 cell-types (or tasks). In addition, we propose an optimization strategy to extract combinatorial relationships among histone modifications from the trained models. Our findings not only validate previous observations but also provide new insights for underlying gene regulation mechanisms that have been observed in recent experimental studies. We note that these insights are, for now, restricted to our literature search. Therefore, we provide the optimized histone pattern maps from the DeepChrome models for all 56 cell types for both cases of gene expression being classified as high and low online (www.deepchrome.org). We hope that biologists are able to utilize these results for drawing significant hypotheses on histone modification interactions that lead to gene activation or silencing.

For future work, we would like to observe DeepChrome's performance on adding histone modifications to understand their combinatorial effects as well. We would also perform cross-cell predictions, where one model is trained on data from one cell type and predictions are made on the other cell types. Previous studies have reported that the correlations among histone modifications remain consistent across cell types. However, the decrease in performance (right tail in Fig. 3) for some cell types in our results suggests the potential to explore this further. Another plausible direction is to understand the effect of relationships among histone modifications for regulating individual genes. This can help biologists in designing 'epigenetic drugs' that can manipulate histone modification marks and control the expression of a particular gene target.

In summary, DeepChrome opens multiple new avenues for studying and exploration of genetic regulation via epigenetic factors. This is made possible due to deep learning's ability to handle a large amount of existing data as well as to automatically extract important features and complex interactions, providing us with important insights. Techniques like DeepChrome hold the potential to bring us one step closer to properly investigating gene regulation mechanisms, which in turn can lead to understanding of genetic diseases.

Conflict of Interest: none declared.