Visualizing Feature Maps in Deep Neural Networks using DeepResolve A Genomics Case Study

Ge Liu [1] **David Gifford** [1]

Abstract

Although many powerful visualization tools have been developed to interpret neural network decisions in input space, methods to interpret feature map space remain limited. Most existing tools examine a network's response to a specific input sample and thus are locally faithful to that sample. We introduce DeepResolve, a gradient ascent based method that visualizes intermediate layer feature maps in an input independent manner. We examine DeepResolve's capability to 1) discover network linear and non-linear combinatorial logic and summarize overall knowledge of a class, 2) reveal key features for a target class, 3) assess a network's activeness in pattern learning and network's vulnerability in feature space, and 4) analyze multi-task class similarity at high resolution. We demonstrate the value of DeepResolve on synthetic and experimental genomic datasets, and DeepResolve reveals biologically interesting observations from the experimental data.

1. Introduction

Deep learning has proven to be powerful on a wide range of tasks in computer vision and natural language processing (Krizhevsky et al.; Szegedy et al., a; Simonyan & Zisserman, 2015; Sutskever et al.; Bahdanau et al.), and these successes have motivated efforts to (1) characterize the individual input features that are detected and (2) interpret how input patterns are combined in a given network layer. Recently, several applications of deep learning in genomic data have shown deep learning's capability to outperform conventional methods across a variety of prediction tasks, such as transcription factor (TF) binding prediction (Alipanahi et al., 2015; Zeng et al.; Zhou & Troyanskaya, 2015), DNA methylation prediction (Zeng & Gifford, 2017; Angermueller et al.), and enhancer-promoter interaction prediction (Singh et al.). However, the construction of deep neural networks makes the interpretation of these trained models difficult (Castelvecchi, 2016), and thus limits model derived biological insight.

Recent computer vision studies have proposed effective ways to visualize and interpret deep neural networks by exposing their hidden model representations in input space with gradient based or activation map based methods, including de-convolutional networks (Zeiler et al., 2010; Zeiler & Fergus, 2014), saliency map (Simonyan et al., 2014), guided back-propagation (Springenberg et al.), layer-wise relevance propagation (Bach et al., 2015), DeepLIFT (Shrikumar et al.) and LIME (Ribeiro et al.). These methods aim to understand the response of a network to specific input examples, and provide a fine-grained visual interpretation of network response by highlighting relevant input data. While this work-flow is visually interpretable and can be validated by observation, it has the limitation of being only locally faithful to the model because it is based upon the selection of an input. The non-linearity and complex combinatorial logic in neural network may limit network interpretation from a single input. In order to extract generalized class knowledge multiple input samples need to be used with post-processing steps to get an overall understanding of a class.

Another class of methods for interpreting networks use only the network itself and do not start with reference inputs to annotate, but instead directly synthesize novel inputs that maximize the activation of the network. For example, (Simonyan et al., 2014), uses gradient ascent on the input space to maximize the predicted score of a class, and DeepMotif (Lanchantin et al., 2016) is an implementation of this method on genomic data. These gradient-ascent methods are different from input-dependent methods in that they explore the input space freely and are not biased towards a specific region of input space. Thus this type of interpretation may be more faithful to all aspects of the model because it generates a 'class model' that represents the activation of a whole class, instead of a response to a specific input. However, the stochastic nature of these meth-

[1]Massachusetts Institute of Technology. Correspondence to: <gifford@mit.edu>, <geliu@csail.mit.edu>.

Proceedings of Workshop on Visualization for Deep Learning at ICML 2017, Sydney, Australia, PMLR 70, 2017.

ods makes them less stable than input-dependent methods, and the images they generates are considered to be unnatural (Nguyen et al., 2015). Some even use them to generate adversarial noise (Goodfellow et al.) to study the vulnerability of a network. Potential reasons for this unnaturalness has been previously explored (Szegedy et al., b; Goodfellow et al.), and regularization methods have been studied to generate images that look more natural (Yosinski et al., 2015), however the improvement is still limited. One explanation is that when gradient ascent methods perform forward and backward propagation between a network's input layer and output layer, the many non-linear units that are present in the network can cause the methods to get stuck in a bad local maximum.

While many network interpretation methods provide visual summaries of desired inputs, few methods have focused on the interpretation of *feature maps* that encode how input patterns are combined in a given network layer. This is in part because feature maps are difficult to represent intuitively in visual form. However, feature maps can reveal key semantic information as well as the parts of a network that are important for a specific task for detector discovery and transfer learning. Thus, exposing key elements in feature space is essential for full understanding of a network. Moreover, directly visualizing input space is only partially informative in biological studies when compared computer vision studies, as in biological studies the combinatorial rules reveal mechanism. Previous work put limited effort in examining how to interpret feature map space in a way that relates to these combinatorial rules. Bolei et al. (Zhou et al.) proposed Class Activation Mapping (CAM) using a global average pooling layer which can learn a linear importance coefficient for each feature map in the last convolutional layer. Selvaraju et al. (Selvaraju et al., 2016) used gradient flow from output to the last convolutional layer to calculate a locally faithful feature map importance weight in their grad-CAM, which replace global average pooling. While these authors notice the class-specificity of a feature map importance pattern, they used it as a weight for an activation map to localize visual interpretation of a class, instead of viewing it as an independent representation of a class. Furthermore, these methods are either limited to specific network architectures or are input dependent.

Here we introduce DeepResolve, a new approach to feature map interpretation that works on all architectures and is reference input free. DeepResolve computes layer specific feature maps that summarize how a network combines elemental layer specific features to predict a specific class. The core approach of DeepResolve is to resolve the meaning of an intermediate layer by decomposing a network's upper and lower layer network components referenced from the intermediate layer and visualizing their function. DeepResolve inverts upper network layers with gradient ascent since they typically contain fewer non-linear units than lower network layers. We then use this visualization to guide the choice of neurons to invert in lower layers. The lower layers are then decomposed and analyzed using DeepResolve. Our proposed method requires no input dataset and is efficient.

Unlike images which contain complex features such as styles and scale, genomic sequences are simple strings of 4 letters, which are typically one-hot encoded into a 2-D array (see Figure 1). These 2-D arrays are used as inputs that are observed by 2-D convolutional filters. In genomic applications, lower level convolutional filters capture short patterns called motifs, which are represented as weights on each nucleotide in each position, while upper layers learn the combinatorial logic or 'grammar' of these motifs. (Zeng et al.) showed that for the purpose of predicting functionality of genomic sequences, deeper networks are not necessarily better than shallower ones. Empirically, one or two convolutional layers with two fully connected layers can achieve good performance. Since the focus of this report is the interpretation of intermediate layer feature maps, we focus on genomic datasets because of the simplicity of their lower level convolutional layers and the importance of discovering combinatorial logic in biology. To test DeepResolve we constructed synthetic genomic datasets with specific logic and validated that DeepResolve recovered this logic.

In summary, DeepResolve interprets network class based behavior in a input-free manner with gradient ascent on intermediate layers; it is efficient and requires no input dataset or post-processing steps; it reveals key features of a target class and discover both linear and non-linear combinatorial logic that a network learns; it assess a network's activeness in pattern learning, and it discovers a network's vulnerability in feature space. It can also be used to analyze class similarity and difference in high resolution. The remaining four sections of this paper discuss how DeepResolve computes class-specific feature importance maps and evaluate neuron importance (Section 2), DeepResolve's ability to recover combinatorial interactions from synthetic data and to assess the learning activeness and vulnerability in feature space (Section 3), a method to compute feature similarity across classes in a multi-class model (Section 4), and the benchmarking of DeepResolve on a TF binding prediction task and a DeepSEA network interpretation task in comparison to contemporary methods (Section 5).

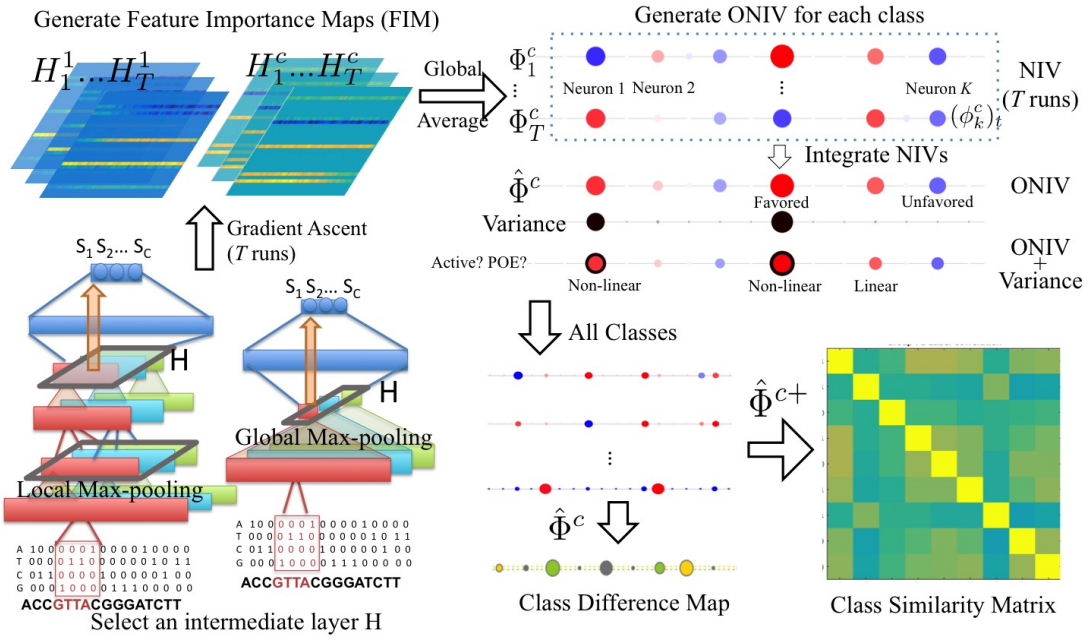

Figure 1. Illustration of DeepResolve's working flow. After a network is trained and an intermediate layer is selected, DeepResolve first computes the feature importance maps (FIM) using gradient ascent for multiple runs with different random initializations. After each run, it takes the global average of FIM scores for each neuron to compute Neuron Importance Vectors (NIV). Each position in the NIV represents a neuron. The variance in each position of the NIVs from different runs is then calculated, and used as an indicator for potential non-linear behavior of a neuron. The NIVs are then combined neuron-by-neuron with the reference to the magnitude of its variance, such that both non-linear and linear logic can be resolved. This process generates an Overall Neuron Importance Vector (ONIV) for each of the output classes, which summarize all 'favored' and 'unfavored' patterns of a class. We then explore ONIV's capability to propose key elements for a specific class, assess learning activeness and help interpret vulnerability of the network. Finally, we use the non-negative ONIVs to analyze class similarity and the ONIVs to analyze class differences.

2. We compute a class specific Feature Importance Map and Neuron Importance Vector

DeepResolve uses a gradient-ascent based method in the similar manner as (Simonyan et al., 2014) to compute a class-specific optimal 'image' H in an intermediate layer that maximize the objective function:

$$H^c = \arg\max_H S_c(H) - \lambda||H||_2^2$$

S_c is the score of class C, which is the C-th output in the last layer before taking the sigmoid or soft-max, and H consists of feature maps $\{H^k\}$ for all the K neurons of the selected intermediate layer. We call H^c a feature importance map (FIM) for class C. As shown in Figure 1, for networks using genomic sequences as input, when the the relevant convolutional layer uses global max-pooling, the FIM is a 1-D vector directly indicating the importance of the motif each filter is capturing, whereas for convolutional layer with local max-pooling, the FIM is a 2-D map of the importance of each filter in different pooling position. The later provides extra information about the spatial distribution of a sequence pattern. In image classification networks, the FIM is 3-D for convolutional layers with local max-pooling, and 1-D for layers with global max/average pooling.

We then compute neuron importance score ϕ_k^c for each of the K neurons by taking the global average of the feature importance map $(H_k)^c$ of that neuron:

$$\phi_k^c = \frac{1}{Z}\sum_{i,j}(H_{i,j}^k)^c$$

Where (i, j) stands for different positions in a 2-D feature map of a neuron (it is $\phi_k^c = \frac{1}{Z}\sum_i(H_i^k)^c$ for neurons that output 1-D feature map, and $\phi_k^c = (H^k)^c$ for neurons that use global max/average pooling). We carefully pick the layer for feature importance map generation such that neurons in this layer capture local sequence patterns that are as short as possible (comparable with motifs). Meanwhile the layer selected should not be too far away from the output because additional subsequent non-linear layers increase uncertainty. For most of the classic networks for

predicting functional regulatory elements (Alipanahi et al., 2015; Zeng et al.), the optimal choice is the layer located in the border of fully connected layers and convolutional layers. For DeepSEA(Zhou & Troyanskaya, 2015) which has 3 convolutional layers, we chose the input to last convolutional layer.

For each target class C, we run gradient ascent multiple times (T times) using different random input initializations sampled from normal distribution. This generates a set of FIMs $\{H_t^c\}$ for each class. Note that although the natural domain of feature map space is $\mathbb{R}_0^+$, we allow FIMs to have negative values during gradient ascent. We found such that negative terms are interpretable and contain rich information about the reasoning of networks decisions. For each generated H_t^c of class C, we calculate neuron importance scores $\{(\phi_k^c)_t\}$ for every neuron which gives a *neuron importance vector* (NIV) $\Phi_t^c = ((\phi_1^c)_t, (\phi_2^c)_t, ..., (\phi_k^c)_t)$, from which we extract model faithful interpretations of the network.

3. DeepResolve helps interpret combinatorial logic in feature space

3.1. NIV recovers non-linear combinatorial logic

As mentioned in the previous section, we repeat gradient ascent multiple times and generate a set of FIMs $\{H_t^c\}$ and NIVs $\{\Phi_t^c\}$ for each class. We evaluate the variance of each position of NIVs, and observe that some neurons are much less stable than others with different initializations, which indicate that their contribution to the output can't be combined additively. We define this type of neurons as non-linear neuron and propose to use the variance of NIVs as an indicator of the non-linearity of a neuron. We further observe that the non-linear neurons can be interacting with each other following certain non-linear combinatorial logic, and the NIVs can be used to help discover the presence of non-linear logic.

To test the ability of this method to recover non-linear input feature relationships we created Synthetic Dataset I with XOR logic. This dataset contains 40000 synthetic DNA sequences that are strings containing only A,T,C,G. The sequences are 200 base pairs (bp) long, with each base randomly selected from the 4 nucleotides as background. We then placed 2 sequence patterns (`CGCTTG`, `CAGATG`) into random position in the 40000 sequences. We label a sequence 1 when only one of the patterns presents, and otherwise 0. We trained a convolutional neural network with 1 convolutional layer containing 32 filters of size 8, followed by local max-pooling with stride size 4. We used 2 fully connected layers with 64 hidden units, and the AUC on the held-out 20% test set is 0.909. To verify that DeepResolve can recover the non-linear logic in Synthetic Dataset

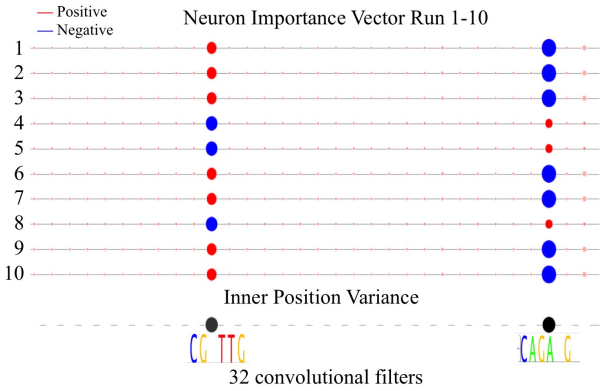

Figure 2. Neuron importance vectors (NIV) generated by all 10 runs of gradient ascent for dataset I. Neuron #11 and #30 always end up having opposite signs indicating that they are probably interacting in a XOR logic. The variance plot on the bottom highlight these neurons. For NIVs, each red circle on the X-axis represents a neuron with a positive NIV value and a blue circle represents a neuron with a negative NIV value. Circle size is proportional to the absolute NIV value, and to the variance value.

I we generated feature importance maps and neuron importance vectors $\{\Phi_t^c\}$ for the last convolutional layer. Figure 2 visualize the neuron importance vectors $\{\Phi_t^c\}$ of all 10 runs, and the variance per position. The filter weight matrix is converted into position weight matrix (PWM), a commonly used representation of motifs in biology studies, by setting all the negative terms to zero and rescale each column to have inner column sum of 1. The interesting motifs are visualized using `seqlogo`, a commonly used visualization tool in motif study. Each position in a `seqlogo` corresponds to a column in the PWM and each letter corresponds to a nucleotide. The heights of the letters are proportional to the weight assigned to that nucleotide within each column, and the total height of the letters in a position is scaled by the information content (entropy) of that position. As shown in Figure 2, with different initialization, the filter 30 capturing `CAGATG` and filter 11 capturing `CGCTTG` always end up having opposite sign. This is suggesting that these two neurons are likely to interact with each other such that they cancel each other out during gradient ascent, as in XOR logic only one variable should be TRUE to make the outcome to be TRUE. The large variance in position 14 and 21 successfully indicates the non-linear behavior of these two neurons, and their interaction pattern suggests non-linear XOR logic.

3.2. NIV recovers linear combinatory logic

The inner position variance of NIVs also reveal neurons that are invariant with different initializations. These neurons are contributing to the output either positively, nega-

tively or not contributing at all. We define this type of neuron as linear neuron because their contribution can be combined additively. To test for the ability of NIV to discover linear combinatory logic (AND, OR, NOT) of the network we created Synthetic Dataset II, which has 4 classes of sequences, each containing different pre-defined patterns. Class 1 has both CAGGTC and AGATT, Class 2 has both AGATT and GCTCAT, Class 3 has CAGGTC or GCTCAT , and Class 4 has CGCTTG or CAAGCG. We trained a multi-task convolutional neural network with the same set-up as the one used for dataset I except that this network has 4 outputs, and calculate FIM and NIV for each class. The test AUC is 0.991,0.990,0.944,0.972 for Class 1 to 4 correspondingly, where 20% of the dataset was held out as test set.

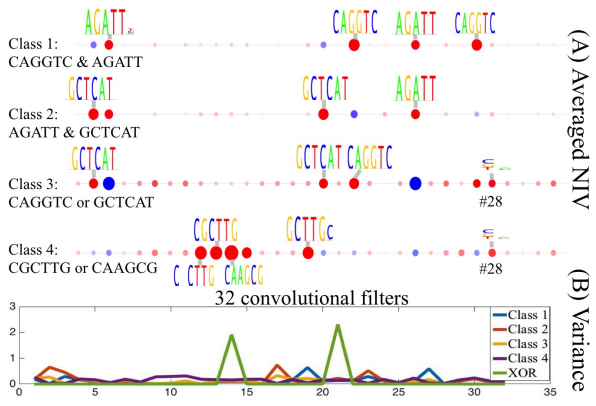

Figure 3. Averaged NIV generated by all 10 runs of gradient ascent for each class. Each red circle on the X-axis represents a neuron with a positive NIV value and a blue circle represents a neuron with a negative NIV value. Circle size is proportional to the absolute NIV value. A motif visualization of the sequence feature that corresponding neuron is capturing is placed above the circle. The NIV successfully ranks predefined sequence features as the most important features for each of the classes, while the difference between AND logic and OR logic is not obvious. The variance plot on the bottom shows that in the linear case the inner position variance is small. The filter #28 contains very little information content, and does not accord with any of the predefined patterns, even though ranked with high score.

Figure 3 shows the variance of the NIVs of each class. Unlike NIVs in XOR logic, the gradient ascent result is much more stable for the linear logic, and thus the variances are small. We generated an averaged neuron importance vector from the average of the NIVs for each class. Figure 3 shows the joint visualization of this map of all classes. All expected patterns in each class are highlighted by strong positive scores, no matter if AND or OR logic is being employed by the class. This is reasonable because the network will prefer to see as many as possible of both patterns to get a higher output activation.

3.3. Integrating the NIVs to generate overall neuron importance visualization (ONIV)

As discussed above, for a network containing linear logic the averaged NIV can be used directly to evaluate the importance of neurons. However, in the non-linear case, a neuron importance score might get canceled out if we take the average of the NIVs. Thus we propose a better way to integrate the NIVs from different runs to generate an *overall neuron importance vector* (ONIV) $\hat{\Phi}^c$ with the following steps:
1) Calculate the variance for each position of the NIVs.
2) For positions that have a small variance (possibly linear, or unimportant), we take the average of all its scores in all NIVs.
3) For positions with a large variance (possibly non-linear), we take the maximum of its scores in all NIVs.

By visualizing ONIV together with the variance in each position, we are able to recover both the importance level of different features and the presence of non-linear logic. We created Synthetic Dataset III which contains both linear and non-linear logic, and tested ONIV's capability to highlight all important features as well as to identify both types of logic. This dataset contains 100,000 synthetic DNA sequences each containing patterns chosen from CGCTTG, CAGGTC and GCTCAT in random positions. We label a sequence 1 only when CAGGTC and one of (GCTCAT,CGCTTG) present, and otherwise 0. This is a combination of AND logic and XOR logic. We also include 20,000 sequences that are totally random and label them as 0. We held out 20% of the dataset and the test AUC was 0.983.

As shown in Figure 4, neurons capturing CGCTTG, CAGGTC and GCTCAT are all assigned significantly high ONIV scores and neurons capturing GCTCAT and CGCTTG are highlighted with large variance, showing that they are contributing to the output non-linearly. The NIVs suggest potential non-linear interaction between GCTCAT and CGCTTG as the corresponding neurons tend to have opposite signs.

3.4. ONIVs permit the activeness of learning to be evaluated

Interestingly, we observe that there are strong negative terms in ONIVs for all classes. Neurons with negative ONIV score are constantly repressed in all runs of gradient ascent. These neurons are associated with 'NOT' logic even though we didn't intentionally add that logic. An ONIV's positive and negative terms can be seen as an indicator of whether a feature is 'favored' or 'unfavored'

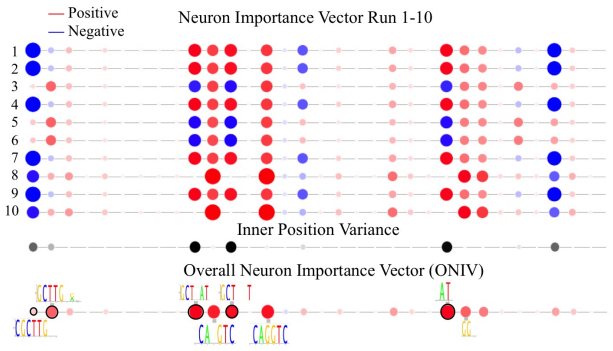

Figure 4. Illustration of the generation of ONIV from NIVs generated by all 10 runs of gradient ascent on the model from Dataset III. Each red circle on the X-axis represents a neuron with a positive NIV value and a blue circle represents a neuron with a negative NIV value. Circle size is proportional to the absolute NIV value. Neurons that are potentially involved in non-linear logic are highlighted with black edge in ONIV.

respectively by the target class. We observed that features highlighted by strong negative ONIV values in a target class are usually patterns 'favored' by other non-target classes, and when the network detects these patterns, it drastically reduce its confidence in classifying the input into the target class. This is suggesting that a network could potentially perform well on a classification task just by using the process of elimination, without actively learning patterns from a target class. In another word, a 'lazy' network could use a process of elimination. In this situation, we should adjust the training set by adding more random negative samples that do not contain any of the patterns in 'NOT' logic to increase the activeness of pattern learning from the positive samples.

3.5. ONIVs discover network's vulnerability in feature space

As shown in Figure 3, neuron 28 contains very little information content (entropy) and does not accord with any of the patterns we designed, even though it is given a high ONIV score. This is an example of a feature that is trusted by the model but not relevant to real world ground truth. Since the only goal of a deep neural network is to maximize the objective function and to get better AUC scores on the validation set, it learns whatever patterns increase its performance. Ground truth irrelevant neurons are potentially dangerous and could be one reason why gradient ascent in input space can produce unrecognizable images. (Szegedy et al., b) showed that when generating adversarial examples for an input, the minimum necessary perturbations are relatively robust across different subsets of the training data, indicating that the adversarial noise may not be random. By looking at the irregular neurons in intermediate layers using ONIV, we can discover neurons that capture patterns of adversarial noise, and analyze a network's vulnerability more systematically.

4. The correlation of non-negative ONIVs reveals lower level feature sharing

We use a novel method to discover lower-level feature sharing, even when the class labels are not correlated, using the overall neuron importance vector proposed in Section 3.3.

The weight sharing mechanism of multi-task neural networks allows the reuse of features among classes that share similar patterns. In past studies, the weight matrix in the last layer before output is used to examine class similarity. However, this is potentially problematic because the features in a network's last layer are already high-level and thus tend to be class-specific. This method also fails to discover lower level feature sharing between classes that are rarely labeled positive together. Moreover, adding special sparsity constraints during training is usually needed to enforce feature sharing in last layer, which could be inefficient. As discussed in Section 3.4, the negative ONIV terms are usually associated with 'NOT' logic that helps the model to obtain improved classification accuracy using the process of elimination. Only the features highlighted by positive ONIV scores represent patterns truly exist in the target class. Thus we use the non-negative ONIV $\hat{\Phi}^{c+}$ to analyze similarity between classes. Below we introduce a class similarity matrix Φ by taking pair-wise Pearson correlation of non-negative ONIV of all the classes.

$$A_{C_iC_j} = \frac{\mathrm{Cov}(\hat{\Phi}^{c_i+}, \hat{\Phi}^{c_j+})}{\sigma_{\hat{\Phi}^{c_i+}}\sigma_{\hat{\Phi}^{c_j+}}}$$

$\hat{\Phi}^{c+}$ encodes the composition of all favored features for a given class in intermediate layer. Therefore it describes the structure of a class with higher resolution than the weight matrix in the last layer. By taking the difference of ONIV of a pair of classes, we can also generate a class difference map by

$$D_{C_iC_j} = \hat{\Phi}^{c_i} - \hat{\Phi}^{c_j}.$$

This map highlights features that are favored by one class but not favored by the other. This is especially helpful when studying cell-type specific problems where a key feature deciding differential expression or binding in different cell type might be crucial.

Figure 6 shows the class similarity matrix of the Synthetic Dataset IV with 4 classes of DNA sequences. Class 1 contains `GATA` and `CAGATG`, class 2 contains `TCAT` and `CAGATG`, Class3 contains `GATA` and `TCAT`, while class 4 only contains `CGCTTG`. We trained a multi-task CNN with one convolutional layer that has 32 x 8bp long filters, and

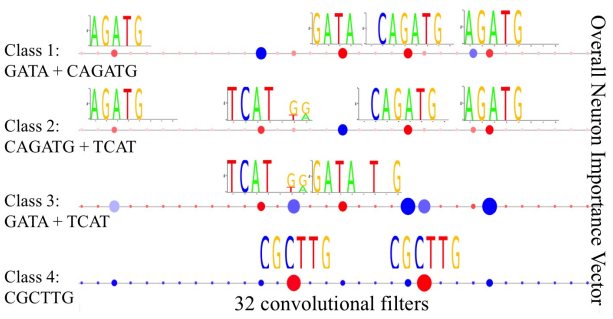

Figure 5. ONIV for Synthetic Dataset IV class 1 - 4. Each circle on the X-axis represents a neuron, with red representing positive ONIV values and blue representing negative values. The size of the circle is proportional to the absolute value of its ONIV score. ONIV successfully ranks predefined sequence features as the most important features for each of the classes.

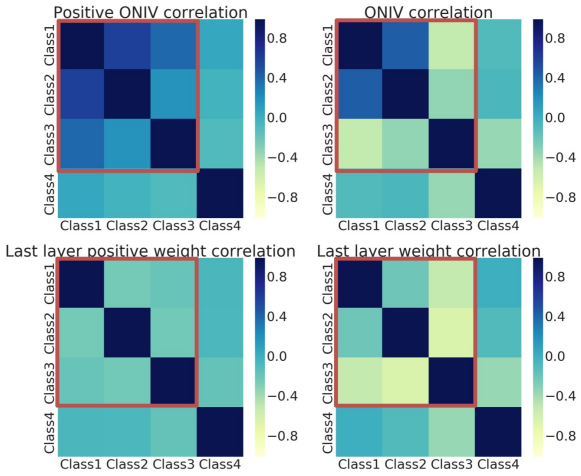

Figure 6. Correlation matrix using non-negative ONIV, ONIV, non-negative last layer weight and last layer weight. X-axis and y-axis stands for Class 1 - 4 from left to right, and from top to bottom. The box on the left-top corner highlights the 3 pairs of classes that share half of the sequence patterns and thus should be assigned higher score than the classes located in the L-shape area in right-bottom corner. The predefined sequence pattern for each class is stated in Figure 5. Matrix in the left-top corner is our proposed Class Similarity Matrix, which successfully assign high correlation to (Class1, Class2), (Class2, Class3) and (Class1, Class3) and low correlation to all pairs with Class 4 which is totally different from the other three. The top right corner uses ONIV without setting negative terms to zero. It fails to propose feature sharing between Class1 and Class3, Class 2 and Class3. The bottom two matrices are calculated by taking the Pearson correlation of the corresponding rows in last layer positive weight (left) and last layer weight (right). They both fail to assign correct scores to some combinations of classes that share sequence features.

1 fully connected layers with 64 hidden neurons. The test AUC is 0.968, 0.967, 0.979, 0.994 for class 1 to 4. The introduced sequence patterns are deliberately selected such that 3 pairs of the classes share half of the patterns, while class 4 is totally different. These four classes are never labeled as 1 at the same time, thus the labels yield zero information about their structural similarities. We compared our method to class correlation analysis using only the last layer weight matrix. The non-negative ONIV correlation matrix successfully assigned higher similarity score to class 1+2, class 1+3 and class 2+3, while the other methods failed to do so. Note that method using last layer weight assigned lowest score to class 2+3, even though they share the pattern `TCAT`.

5. DeepResolve produces superior results on experimental data

We analyzed two experimental biological data sets to compare DeepResolve to other methods.

5.1. Identifying key sequence features in models of TF binding

We applied DeepResolve to convolutional neural networks trained on 422 Transcription Factor ChIP-Seq experiments which have a known motif as ground truth. The inputs are 200 bp long sequences embedded into 2-D matrices using one-hot encoding, where the positive set contains sequences that have peaks of TF binding signal and are matched with the known motif. The negative set consists of bi-nucleotide shuffled sequences. We train a single-class CNN for each experiment using 1 convolutional layer with 128 filters of size 24 with global max-pooling, and 2 fully connected layers with 32 hidden units. We then generate FIMs and ONIVs for each experiment on the last convolutional layer, and rank the filters using ONIV scores. We convert the top filters into position weight matrices (PWMs) and match them with known motif for the target TF using TOMTOM (Gupta et al., 2007), a tool for comparing PWM of different motifs, and count how many times we hit the known motif in top 1, top 3 and top 5 filters with p-value less than 0.5 and 0.05. We compare our method to DeepMotif(Lanchantin et al., 2016), a similar visualization tool that generates important sequence features by conducting gradient ascent directly on the input layer. We modified DeepMotif's initialization strategy to allow multiple random initializations instead of using all 0.25 matrix (naming it enhanced-DeepMotif), and take the most informative 24bp fragment of generated sequences with top 5 class score. We also compare with two gradient-based method: Grad-CAM and the direct multiplication of gradients and neuron activation. We sample 5000 sequences

Table 1. Top-1, top-3, top-5 accuracy in identifying matching motif for TF binding (out of 422 experiments). DR stands for DeepResolve (out method), eDM stands for enhanced-DeepMotif

	TOP 1		TOP 3		TOP 5	
P-VALUE	0.5	0.05	0.5	0.05	0.5	0.05
DR (OURS)	**368**	**345**	416	405	**421**	**419**
GRAD-CAM	353	329	407	394	421	417
GRAD*INPUT	350	338	**417**	**412**	**421**	**419**
EDM	120	42	212	62	249	69

from the positive set, and calculate the Grad-CAM feature map importance weight for each input by taking the global average of the gradient flowing back from output to last convolutional layer. For the second method, we multiply the gradient by neuron's activation in the last convolutional layer for each input, and take the global average. We take the average of the scores assigned to all the inputs as an indication of the importance of the neurons.

Shown in Table 1, our method successfully proposes known matching motifs as top 5 features in 421 out of 422 experiments with p-value less than 0.5, and 419 out of 422 experiments with p-value less than 0.05, which outperforms DeepMotif by six times. Our method also outperforms Grad-CAM in top-1, top-3, top-5 accuracy and gradient*input in top-1 accuracy, and is compatible to both of them in the rest of the tasks even though we do not refer to any input dataset when calculating the ONIVs.

These results demonstrate that convolutional neural networks are suitable for predicting genome function because the resulting convolutional filters can capture motifs accurately.

5.2. Identifying class correlation in the DeepSEA network

We validate DeepResolve's class similarity analysis on DeepSEA (Zhou & Troyanskaya, 2015), a classic multi-class convolutional network trained on whole genome data to predict 919 different features including chromatin accessibility, TF binding and histone marks across a variety of cell types. This network learns abundant information about the functionality of genomic sequences, and thus could yield potentially important biological insights. We analyze the class similarity of each pair of its 919 classes and use it as a metric to evaluate functional correlation between the classes. We discover strong positive correlations between DNase accessibility data and TF/Histone marks known to be actively regulating DNase hypersensitivity, and strong negative correlations between DNase accessibility data and TF/Histone marks known to repress the DNase hypersensitivity.

In DeepSEA, input sequences are 1000bp long, and the labels are 919 long binary vectors. Each bit of the vector correspond to one experiment and it is labeled as 1 if there is peak inside the center 200bp of the input sequence. The network has 3 convolutional layers with 320, 480, 960 filters, and 1 fully connected layer. We chose the input layer to the $3rd$ convolutional layer as H to generate a feature importance map, where the neurons encodes 480 x 51bp long local sequence features which can be easily visualized using gradient-based methods with small number of inputs. We then generate our class similarity matrix by calculating the Pearson correlation of the non-negative ONIVs. Given that DeepSEA takes in 1000bp long sequences around the biological event, it captures upstream and downstream sequence context. Therefore our proposed metric is measuring similarities between the contextual structures of a pair of regulators, which could suggest interesting correlations in functionality and mechanism. Figure 7 shows our class similarity matrix and label correlation of DeepSEA. Our matrix revealed strong correlations between pairs of targets that do not necessarily co-appear within 200 bp, but are similar in functionality. We then examined the correlation pattern between selected TF/histone marks and DNase I hypersensitivity across different cell types. Figure 8 shows the bi-clustering result on the TF-histone mark/DNase correlation matrix (the upper part of the DeepSEA class similarity matrix). We observed clusters of TFs and histone marks sharing similar correlation patterns, and some of them exhibit cell-type specific effect on DNase hypersensitivity. We collapsed the map into 1-D by calculating number of strong positive and negative correlations with DNase experiments for each TF/Chromatin mark. As shown in Figure 8, we can characterize each TF and histone mark's association with chromatin accessibil-

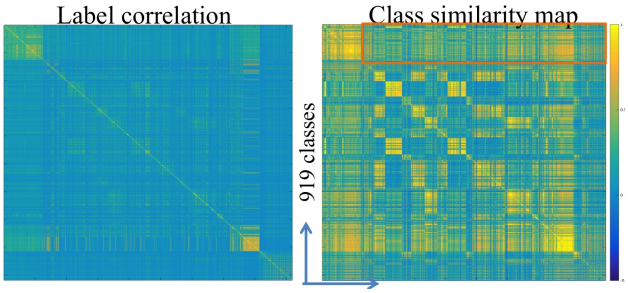

Figure 7. Class similarity map for DeepSEA. X and Y axis represents 919 different experiments including DNase I hypersensitivity, TF binding and Histone marks across different cell types. The sub-matrix highlighted by the red box is used for DNase correlation pattern analysis in Figure 8.

ity using these indexes. We identify groups of TFs/histone marks that are positively correlated with DNase hypersensitivity (located in most left of the histogram), and most of them are known to be chromatin regulators. We also identify groups of TFs/histone marks that are negatively correlate with DNase hypersensitivity and observe that most of them are well-known transcription repressors.

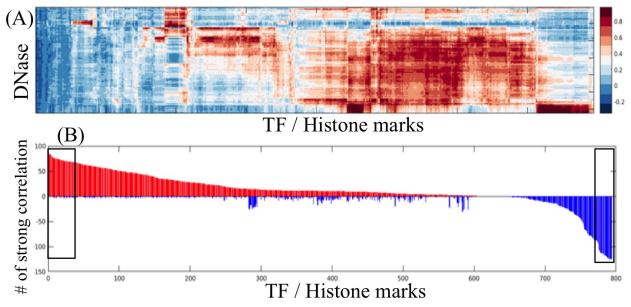

Figure 8. (A)Bi-clustering of TF/Histone mark - DNase hypersensitivity correlation map (right-top corner of Figure 7), x-axis are the TF/Histone mark experiments and y-axis are DNase hypersensitivity experiments across 125 different cell types. (B)Bar-plot of number of strong positive correlation (red) and strong negative correlation (blue) with DNase experiments for each of the TFs and Histone marks. Majority of the TF/histone marks in the left box are known chromatin regulators, and majority of TF/histone marks in the right box are known transcription repressor.

Another way of utilizing the class similarity matrix is directly use it as a metric of distance for clustering. We performed hierarchical clustering of the 919 biological targets and identified meaningful clusters where targets within the same cluster are known to be related to each other, including groups of the same TF across different cell types, groups of different TFs in same cell type, and groups of targets that are known to be co-binder or co-regulating elements.

We also make use of the feature importance maps to discover novel sequence features, and explore novel functionalities of some known motifs. We ranked the 480 sequence features by its ONIV score in each experiment. For each sequence feature, we extracted a group of experiments in which it is ranked as among the top 5 most important features, and visualized the experiment matrix. We recognize some motifs that might be relevant to cell-type specific binding, or mediating the binding of multiple targets. (Please contact the authors for more analysis on DeepSEA if interested.)

6. Conclusion

DeepResolve is a gradient ascent based method for visualizing and interpreting network's behavior in feature space that is reference input free. DeepResolve provides a summary of a deep neural network's decision making process, is capable of extracting both linear and non-linear combinatorial logic of a network, reveals key features for a target class, assesses a network's activeness in pattern learning, discovers network's vulnerability in feature space, and analyzes class similarities at high resolution.

References

Alipanahi, Babak, Delong, Andrew, Weirauch, Matthew T, and Frey, Brendan J. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature Biotechnology*, 33(8):3–1, 2015. doi: 10.1038/nbt.3300.

Angermueller, Christof, Lee, Heather J, Reik, Wolf, and Stegle, Oliver. DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning. doi: 10.1186/s13059-017-1189-z.

Bach, Sebastian, Binder, Alexander, Montavon, Grégoire, Klauschen, Frederick, Müller, Klaus Robert, and Samek, Wojciech. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7), 2015. ISSN 19326203. doi: 10.1371/journal.pone.0130140.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE.

Castelvecchi, Davide. THE BLACK BOX OF. *Nature*, 538:20–23, 2016. ISSN 0028-0836. doi: 10.1038/538020a.

Goodfellow, Ian J, Shlens, Jonathon, and Szegedy, Christian. EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES.

Gupta, Shobhit, Stamatoyannopoulos, John A, Bailey, Timothy L, and Noble, William Stafford. Quantifying similarity between motifs. *Genome biology*, 8(2):R24, 2007. ISSN 1474-760X. doi: 10.1186/gb-2007-8-2-r24.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks.

Lanchantin, Jack, Singh, Ritambhara, Lin, Zeming, and Qi, Yanjun. Deep Motif: Visualizing Genomic Sequence Classifications. pp. 1–5, 2016.

Nguyen, Anh, Yosinski, Jason, and Clune, Jeff. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436, 2015.

Ribeiro, Marco Tulio, Singh, Sameer, and Guestrin, Carlos. " Why Should I Trust You? " Explaining the Predictions of Any Classifier. doi: 10.1145/2939672.2939778.

Selvaraju, Ramprasaath R., Cogswell, Michael, Das, Abhishek, Vedantam, Ramakrishna, Parikh, Devi, and Batra, Dhruv. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. oct 2016.

Shrikumar, Avanti, Greenside, Peyton, Shcherbina, Anna Y, and Kundaje, Anshul. Not Just A Black Box: Learning Important Features Through Propagating Activation Differences. 1.

Simonyan, Karen and Zisserman, Andrew. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. 2015.

Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *Iclr*, pp. 1–, 2014.

Singh, Shashank, Yang, Yang, Póczos, Barnabás, and Ma, Jian. Predicting Enhancer-Promoter Interaction from Genomic Sequence with Deep Learning.

Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin. STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to Sequence Learning with Neural Networks.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going Deeper with Convolutions. a.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. b.

Yosinski, Jason, Clune, Jeff, Nguyen, Anh, Fuchs, Thomas, and Lipson, Hod. Understanding Neural Networks Through Deep Visualization. 2015.

Zeiler, Matthew D., Krishnan, Dilip, Taylor, Graham W., and Fergus, Rob. Deconvolutional networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2528–2535, 2010. ISSN 10636919. doi: 10.1109/CVPR.2010.5539957.

Zeiler, Md and Fergus, Rob. Visualizing and understanding convolutional networks. *Computer Vision–ECCV 2014*, 8689:818–833, 2014. ISSN 978-3-319-10589-5. doi: 10.1007/978-3-319-10590-1_53.

Zeng, Haoyang and Gifford, David K. Predicting the impact of non-coding variants on DNA methylation. *Nucleic Acids Research*, 2017. doi: 10.1093/nar/gkx177.

Zeng, Haoyang, Edwards, Matthew D, and Gifford, David K. Convolutional Neural Network Architectures for Predicting DNA-Protein Binding. pp. 1–10.

Zhou, Bolei, Khosla, Aditya, Lapedriza, Agata, Oliva, Aude, and Torralba, Antonio. Learning Deep Features for Discriminative Localization.

Zhou, Jian and Troyanskaya, Olga G. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature methods*, 12(10):931–4, 2015. ISSN 1548-7105. doi: 10.1038/nmeth.3547.