2019 IEEE Symposium Series on Computational Intelligence (SSCI)
December 6-9 2019, Xiamen, China

A Metric to Measure Contribution of Nodes in Neural Networks

Jay Hoon Jung
Department of Computer Science,
Stony Brook University
Incheon, Korea
Email: jay.jung@stonybrook.edu

Yousun Shin
Department of Computer Science,
Stony Brook University
Incheon, Korea
Email: yousun.shin@stonybrook.edu

Youngmin Kwon
Department of Computer Science,
The State University of New York at Korea
Incheon, Korea
Email: youngmin.kwon@sunykorea.ac.kr

***Abstract*—We propose a metric that can measure the influence of a node in a layer of a neural network to a node in subsequent layers. The metric, called *influential scores*, enables us to interpret the effect of elements in the input layer on the output layer. Furthermore, it provides us a way to inspect hidden layers of a neural network. When the former usage is applied to an image classification task, one can see which parts of an image contribute positively and which parts work negatively to the classification result. That is, the metric provides an interpretable explanation about the input image on the classification result. For the latter, we could explain how hidden layers control the output layer values.**

***Keywords*—deep neural network; interpretability**

I. Introduction

The hidden layers inside deep neural networks have been a *black box* to humans. Even with tremendous success of deep learning, we still do not understand exactly how a *Neural Network* (NN) works. We are able to create a network to classify images successfully, but we cannot describe why and how these images can be classified. Furthermore, we do not know how to optimize a huge number of these parameters in a NN model neither. It is likely that unnecessary parameters are abundant in a certain design, but we cannot even identify these parameters, still less know how to eliminate them properly. Thus, understanding the prediction-making processes of NN models can have fundamental impacts on deep learning from designing an optimal NN to its numerous applications.

The studies related to computer vision often shed light on the *black boxed* processes. NNs can extract rich mid-level features from images. Zeiler et al. visualized features of images found inside hidden layers [1], and Oquab et al. demonstrated mid-level image descriptors could be transferred to a different neural network [2]. Although neural networks seem to generalize well on object recognition tasks because of rich mid-level image descriptors, nonlinearity of neural networks is not clearly understood since imperceptible non-random perturbations lead to misclassification at the same time [3].

Meanwhile, the interpretation over the predictions of NN models is an important task and many researchers have been attempting to explain the predictions of NN. Several well-known methods, based on the backpropagation have been published. Simonyen et al. derived internal structures, the saliency map, of a given model by calculating the gradient of the target class, and back-propagation is used to find the gradient [4]. Bach et al. presented Layerwise Relevance Propagation (LRP), a method to calculate relative importance through layers [5]. Shrikumar et al. showed LRP rules are equivalent to elementwise products of the gradient and input. Furthermore, they showed that this feature type is corresponding to the first-order Taylor series approximation with the all-zero reference [6]. Lundberg et al. suggested unified approaches to interpret models and they applied Shapley values to Shrikumar et al's DeepLIFT [7], [6]. Chen et al. introduced an explainer, or a feature selector, to select a subset of most informative pixels from a given input. The explainer, named as L2X (Learning to eXplain) is to maximize the mutual information between response variables and selected features. They executed their L2X model works on synthetic and real data [8].

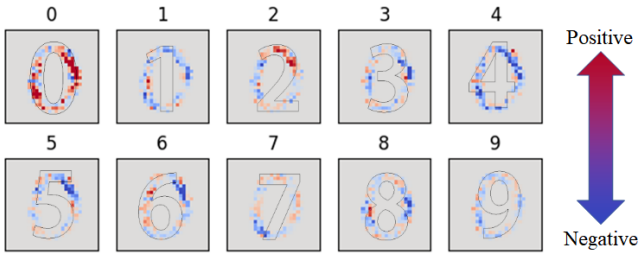

Fig. 1. The influential score images for a number zero (Outlines were overlaid to help to understand). The red and blue pixels depict positive and negative scores. The first image shows the informative pixels why the network classifies the input as number zero. The blue pixels of the other images suggest why the network does not classify a number as the others.

In this paper, we develop a tool to inspect hidden layers directly and to reveal the most informative subset of features for a given instance. We propose a new metric to estimate influences between any two nodes within a network. In other words, we create a score, called an, *influential score*, which shows the proportional contribution of a node with respect to other nodes. The influential scores can be used in several areas: analyzing how hidden layers work in neural networks, providing interpretable explanations about a prediction, and selecting subsets of features from an instance.

In Section II, we will define the influential score mathe-

978-1-7281-2485-8/19/$31.00 ©2019 IEEE

matically. In the following sections, we will demonstrate its application to synthetic, MNIST and fake face data sets. We will show the influential scores-based instance-wise feature selection and describe how a neural network model controls output values for the binary classification with synthetic data. Combining with the hybrid convolutional network, we will suggest explanations for a classification, and which features are important in the classification using MNIST data sets. Lastly, we will describe a NN model to detect GAN-generated fake faces and analyze informative regions for a given input image.

II. Proposed Method

In this section, we describe the basic mathematical definition of the influential scores. The scores are based on the trained weights of a NN and each node value for a specific input. The influential scores for the input layer with respect to the output layer show how informative the elements of an input are compared to others.

A. Framework

For a given neural network model M, $w^l_{i,j}$ denotes the weight between the i^{th} node in l^{th} layer and the j^{th} node in $(l+1)^{st}$ layer, and L^l_i and b^l_i denote i^{th} node (or its value) and i^{th} bias in l^{th} layer respectively.

If rectified linear units function, *ReLU*(x), is used as the activation function for a NN M, then the value of a node L^{l+1}_j is

$$L^{l+1}_j = ReLU\left(\sum_i (w^l_{i,j} \cdot L^l_i) + b^{l+1}_j\right). \quad (1)$$

For a trained NN model, the weights $w^l_{i,j}$ and the bias b^l_j are constants, but the node value L^l_i is a variable that depends on the input I.

Definition. When an input I is fed into a trained model M, its *snapshot* M_I can be represented by a tuple $M_I = (\mathcal{W}, \mathcal{L})$, where $\mathcal{W} = \{w^0_{0,0}, w^0_{0,1}, w^0_{0,2}, ..., w^l_{i,j}\}$, and $\mathcal{L} = \{L^0_0, L^0_1, L^0_2, ..., L^l_i\}$.

The difference between M and M_I is that while M is a structure, M_I is a snapshot of M with the nodes of L^l_i loaded with their values from the input I.

Definition. For a given M_I, the *influential matrix* $\mathcal{J}^{l,l+1}$ is defined as

$$\mathcal{J}^{l,l+1}_{i,j} = \frac{w^l_{i,j} \cdot L^l_i}{\sum_i |w^l_{i,j} \cdot L^l_i|}$$

An influential matrix $\mathcal{J}^{l,l+1}$ has a dimension of n by m, where n and m are the number of nodes in L^l and in L^{l+1} respectively. The influential matrix implies how much a node L^l_i influences in L^{l+1}_j. Since weight values can be negative values, $\mathcal{J}^{l,l+1}_{i,j}$ can be negative; a value of a node L^l_i reduces a value of a node L^{l+1}_j.

Unlike the proposed method that uses the absolute values in the denominator, Bach et al. suggested to simply sum over all the $w^l_{i,j} \cdot L^l_i$ [5]. Their methods are turned out to be equal to the element-wise product of inputs and gradients as previously described in Introduction.

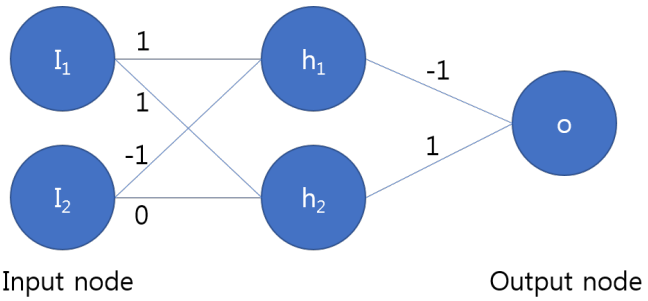

Fig. 2. Toy network computing $o = \min(i_1, i_2)$. The influential scores can quantify ignored nodes in gradient-based method. Weight values are indicated in numbers with edges. Biases are all set to 0. At the point where $i_1 < i_2$, $\frac{\partial o}{\partial i_2} = 0$. When $i_1 > i_2$, then $\frac{\partial o}{\partial i_1} = 0$. The gradient-based approaches assign scores either exclusively to i_1, or i_2 The their influential scores are assigned based on the distance of two inputs.

B. Calculating Influential Scores from Matrices

Definition. For a given M_I and a positive integer $\delta \in \mathbb{N}^+$, the matrix multiplication between two influential matrices, $\mathcal{J}^{l,l+\delta}$ and $\mathcal{J}^{l+\delta,l+\delta+1}$, defines an influential matrix $\mathcal{J}^{l,l+\delta+1}$.

For a given M_I, let $\mathcal{J}^{l,l+1}$ and $\mathcal{J}^{l+1,l+2}$ be $n \times m$ and $m \times o$ matrices respectively. Then, $\mathcal{J}^{l,l+2}$ has $n \times o$ dimension by definition. An element $\mathcal{J}^{l,l+2}_{i,k}$ is a measure of how much influence L^l_i has in L^{l+2}_k.

Definition. For a given M_I and a $\delta \in \mathbb{N}^+$, the elements of an influential matrix $\mathcal{J}^{l,l+\delta}$ are defined as *influential scores*, i.e., $\mathcal{J}^{l,l+\delta}_{i,j}$ is an influential score.

Axiom. For any $L^l_i \in \mathcal{L}$ and $L^m_j \in \mathcal{L}$, there is an influential score $\mathcal{J}^{l,m}_{i,j}$ where $l < m$.

Therefore, the influential scores of a node with respect to any node placed in one of the subsequent layers can be calculated. Especially, the influential scores for a specific node with respect to nodes in output layers show a specific contribution of that node to the prediction of a model.

C. Usefulness of Influential Score

This definition of influential scores turns out to be very useful. The scores represent normalized metrics to quantify the influences between two connected nodes. They are normalized between -1 to 1 depending on their influences on subsequent nodes. For example, assume there are two strongly correlated nodes between two connected layers with no bias; the node value in the latter layer is only positively proportional to the node value in the previous one. That is, the weight values between these two layers are all 0s except for the one. The influential scores for the latter node are all 0s for but only one has a value of 1. When two nodes have negatively proportional to each other, the influential score becomes -1. If the latter node is also related to other nodes in the previous layer, the influential scores will be between -1 and 1. In addition, we can keep track of important nodes using influential matrices. In the neural networks, nodes are complicatedly connected each other so that it is not easy to see how their influences are passed onto others. The influential scores show these connections with simple matrix products.

In Figure 2, the output of the toy network is the smaller of i_1 and i_2. Specifically, $h_1 = i_1 - i_2$ if $i_1 > i_2$ or $h_1 = 0$ if $i_1 < i_2$. The actual parameter o' to the ReLU function in the output layer is $o' = -i_1 + i_2 + i_1 = i_2$ if $i_1 > i_2$ or $o' = i_1$ if $i_1 < i_2$. Then, after applying the ReLU, $o = \min(o', 0)$. $o = i_2$ if $i_1 > i_2$ and $o = i_1$ otherwise.

How do we know which values more important than others when $i_1 > i_2$? The gradient-based methods claims that i_2 is solely responsible for $i_1 > i_2$ [4]. For example, using the *saliency map* and *gradient* $\times$ *input* method [4], [5], [6], the effects of the the larger side of the input are ignored. That is, $\nabla \min(i_1, i_2) = [\frac{\partial i_1}{\partial i_1}, \frac{\partial i_1}{\partial i_2}] = [1, 0]$ when $i_1 < i_2$ and $\nabla \min(i_1, i_2) = [\frac{\partial i_2}{\partial i_1}, \frac{\partial i_2}{\partial i_2}] = [0, 1]$ when $i_1 > i_2$. However, the output values are also dependent on the larger of the two as it affects the min function. In other words, the gradient-based methods do not reveal the fact that there is a min function that selects the smaller of the two inputs.

Skrikumar et al. suggested that the contribution should be $0.5 \min(i_1, i_2)$ for both i_1 and i_2 nodes using the RevealCancel rule in DeepLIFT [6]. They claimed that two inputs should have the same importance of $0.5 \min(i_1, i_2)$. However, is it unnatural to assign the same scores when two numbers are different? The importance score should be differentiated when two inputs are far apart, e.g. 1000 and 1, and when two inputs are close to each other, e.g. 2 and 1. We believe that assignments of importance scores should be considered how actually one node influence the other.

Therefore, our approaches can quantify relative importances even when the gradient-based and other methods fail to explain. The influential scores disclose the relative importance scores for the larger values since the scores are defined in terms of ratios to the absolute summation. For example, let I_1 and I_2 be 2 and 1. h_1 node in Figure 2 has influential scores for I_1 and I_2 are $2/3$ and $-1/3$ respectively. h_2 node has influential scores for I_1 and I_2 are 1 and 0 respectively. Similarly, the scores for h_1 and h_2 are $-1/2$ and $1/2$ since h_1 and h_2 are both 1. A node of I_1 have more influence to the output layer even if $o = i_2$.

Moreover, as indicated in the definition of the influential matrix, there are no terms for biases. However, the influence of biases is reflected in the value of a node, L_j^{l+1}, since it includes biases terms in Definition 1. Therefore, even with no inputs, no activated nodes in the previous layer, information can still be passed to the output. That is, the influential scores reveal important nodes due to these biases. In gradient-based methods, these biases are completely ignored. In DeepLIFT, biases values are considered by adding references. However, DeepLIFT only assigns scores respect to reference so that it may be ignored the stored information inside of neural networks.

D. Obtaining Influential Tensor for Convolutional Neural Network

Building an influential matrix for a fully connected layer is explained in the previous section. However, when there are convolutional layers in a NN, a tensor operator is required to compute an influential matrix.

For a given neural network model M, let $L^l_{x,y,d}$ be the node at (x, y)-position of depth d in l^{th} convolutional layer and let D^l be the total depth of layer L^l. w^l_{i,j,d_1,d_2} denotes the $(i, j)^{th}$ weight of the filter from depth d_1 of l^{th} convolutional layer to depth d_2 of $(l+1)^{st}$ layer, where i, $j \in \{-1, 0, 1\}$ in case of 3×3 filters, and d_1, d_2 are depth of l^{th} and $(l+1)^{st}$ convolutional layers.

$$L^{l+1}_{x,y,d_2} = ReLU\left(\sum_{-1\le i,j\le 1}\sum_{d_1=1}^{D^l}(w^l_{i,j,d_1,d_2}\cdot L^l_{x+i,y+j,d_1})\right)$$

Definition. When an input I is fed into a trained convolutional network model M, its *snapshot* M_I can be represented by a quadruple M_I = ($\mathcal{W}_c$, $\mathcal{L}_c$, $\mathcal{W}$, $\mathcal{L}$), where $\mathcal{W}_c$ = $\{w^0_{-1,-1,0,0},\ w^0_{0,-1,0,0},\ w^0_{0,0,0,0},\ ...,\ w^l_{i,j,d_1,d_2}\}$ and $\mathcal{L}_c$ = $\{L^0_{0,0,0},\ L^0_{0,0,1},\ L^0_{0,0,2},\ ...\ , L^l_{x,y,d}\}$ are the weights and the layers for the convolutional layers respectively, and $\mathcal{W}$ = $\{w^0_{0,0}, w^0_{0,1}, w^0_{0,2}, ..., w^l_{i,j}\}$ and $\mathcal{L}$ = $\{L^0_0, L^0_1, L^0_2, ..., L^l_i\}$ are the weights and the layers for the fully connected layers respectively.

Definition. For a given M_I, an *intermediate tensor* $\mathcal{I}^{l,l+1}$ is

$$\mathcal{I}^{l,l+1}_{(x_1,y_1,d_1),(i,j,d_2)} = \frac{w^l_{i,j,d_1,d_2}\cdot L^l_{x_1,y_1,d_1}}{\sum_{-1\le i',j'\le 1}\sum_{d1=1}^{D^l}|w^l_{i',j',d_1,d_2}\cdot L^l_{x_1+i',y_1+j',d_1}|}$$

For high-dimensional arrays like $\mathcal{I}$ in the equation above, we group the related subscripts together using parentheses to make them easier to read. The *influential tensor* $\mathcal{J}^{l,l+1}$ is defined as

$$\mathcal{J}^{l,l+1}_{(x_1,y_1,d_1),(x_2,y_2,d_2)} = \mathcal{I}^{l,l+1}_{(x_1,y_1,d_1),(x_2-x_1,y_2-y_1,d_2)}$$

where i, $j \in \{-1, 0, 1\}$ in case of 3×3 filters. x_2 and y_2 are equal to $x_1 + i$ and $y_1 + j$ respectively. It simply changes indices to match the position of $(l+1)^{st}$ layer.

The basic idea is the same as a fully connected influential matrix, but an influential tensor for convolutional layers contains the position and the depth dimensions for each layer. However, the multiplication of tensors has to be defined as it is not like the matrix products.

Definition. For a given M_I, the *influential tensor multiplication*[1], $\odot$, is defined as

$$(\mathcal{J}^{l,l+1}\odot\mathcal{J}^{l+1,l+2})_{(x_1,y_1,d_1),(x_2,y_2,d_2)} = \sum_i\sum_j\sum_k \mathcal{J}^{l,l+1}_{(x_1,y_1,d_1),(i,j,k)}\cdot\mathcal{J}^{l+1,l+2}_{(i,j,k),(x_2,y_2,d_2)}$$

Definition. For a given M_I and any $\delta \in \mathbb{N}^+$, the multiplication between two influential tensors, $\mathcal{J}^{l,l+\delta} \odot \mathcal{J}^{l+\delta,l+\delta+1}$, defines the *influential tensor* $\mathcal{J}^{l,l+\delta+1}$.

Definition. For a given M_I and any $\delta \in \mathbb{N}^+$, the elements of an influential tensor $\mathcal{J}^{l,l+\delta}$ are defined as *influential scores*, i.e., $\mathcal{J}^{l,l+\delta}_{i,j}$ is an influential score.

Axiom. For any $L^l_{x_1,y_1,d_1} \in \mathcal{L}$ and $L^m_{x_2,y_2,d_2} \in \mathcal{L}$, there is an influential score $\mathcal{J}^{l,m}_{(x_1,y_1,d_1),(x_2,y_2,d_2)}$ where $l < m$.

[1]The influential tensor multiplication is not the Kronecker product nor the Kronecker sum, but is the Einstein summation with regard to i, j, k.

III. Max-pooling

Conventionally, each convolutional layer is followed by a max-pooling layer. Mathematically, a pooling layer simply selects the maximum value among the filtered results in a sub-region as the representative of the region. For an influential tensor, $\mathcal{J}^{l,l+1}_{(x_1,y_1,d_1),(x_2,y_2,d_2)}$, the dimensions of x_1 and y_1 are equal to x_2 and y_2 with a stride of 1 in the convolutional neural network. Selecting the representative, a max-pooling layer reduces the dimensions of x_2 and y_2 by half or less depending on the size of pooling filter. In case of 2×2 filters, the indexes of influential tensors becomes

$$\mathcal{J}^{l,l+1}_{(x_1,y_1,d_1),(x_2,y_2,d_2)} \rightarrow \mathcal{J}^{l,l+2}_{(x_1,y_1,d_1),(x_2/2,y_2/2,d_2)}$$

after the max-pooling layer.

As a max-pooling layer samples only the maximum value among filtered results and we know their indices from L^{l+1}_{x,y,d_2}, we can eliminate unnecessary tensor elements using these indices.

Therefore, for a given quadruple M_I, we can calculate any influential scores. In other words, we can monitor each node with respect to any nodes in subsequent layers.

IV. Experiment

We performed experiments on synthetic and real data sets. Using synthetic data, we showed the usefulness of the influential scores in selecting features locally and in analyzing hidden layers. In Section IV-A we explained how we generated inputs and labels for the synthetic data. For the real data sets, we use MNIST and GAN-generated fake face data sets. Using them we interpreted how NNs work.

A. Synthetic Data

Data: We generated three different synthetic data sets: two-features, plus-one, and orange skin [8]. For these synthetic data sets, 10-dimensional random input vector Xs are created by sampling from a uniform distribution of certain ranges. For two-features, the samples are in between 0 and 1 range. For plus-one and orange skin, the range is from 1 to 2. The label vector Y is defined as follows:

- **Two-features**: A 2-dimensional label Y is created from X_0 and X_1, which are the first two elements of X.

$$Y_0 = 1 - Y_1, \quad Y_1 = 1/(1 + e^{X_0 \times X_1})$$

- **Plus-one**: Similar to Two-features except for the exponent term in Y_1.

$$Y_0 = 1 - Y_1, \quad Y_1 = 1/(1 + e^{X_0 \times X_1 - 2})$$

- **Orange skin:** A 2-dimensional label Y is created from X_0, X_1, X_2, and X_3.

$$Y_0 = 1 - Y_1, \quad Y_1 = 1/(1 + e^{X_0^2 + X_1^2 + X_2^2 + X_3^2 - 10})$$

We modified these data sets from commonly used ones in the feature selection literature. Especially, our data sets are similar to Chen's data sets of XOR and orange skin [8]. Chen et al. used the Gaussian distribution of mean 0 and variance 1 instead of the uniform distribution. Observe that the label Y depends only on X_0 and X_1 in two-features and plus-one and only on X_0, X_1, X_2 and X_3; the rest elements of X do not have any effects on the label Y. In this experiment, we will validate that the influence score can pick those and only those relevant inputs that have effects on the output.

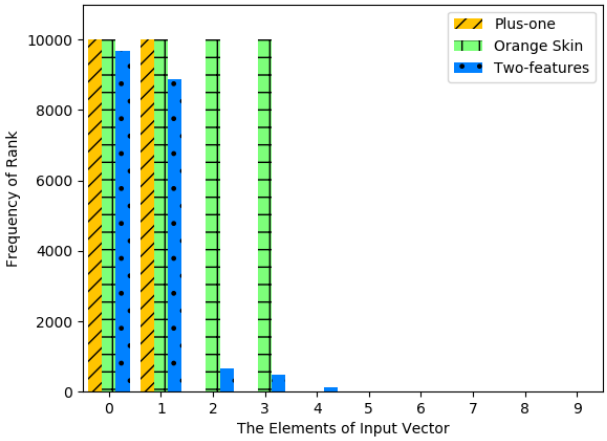

Fig. 3. The distribution of rank of X_0 and X_1 for plus-one and two-features and that of X_0, X_1, X_2 and X_3 for orange skin. Rank 0 means the most informative elements, i.e., the highest influential scores

Architecture: 10-dimensional inputs X are fed into the network model with three fully-connected hidden layers: L1, L2, and L3. Each layer has 200 nodes. We trained the model with 80,000 samples and successfully gained more than 99.3% accuracy for all data sets.

Ranks: In this experiment, we check how many times the relevant elements get large influential scores. We define the rank of an element X_i as the index of X_i when the input vector X is sorted by the influential score in the descending order. We counted how many times the ranks of X_0 and X_1 for plus-one and two-features are 0 or 1, and how many times those of X_0, X_2, X_2 and X_4 for orange skin are 0, 1, 2, or 3 when the influential scores are measured from X_i to the output layer over 10,000 samples. Figure 3 shows the influential scores successfully found most informative elements in all data sets.

Local and Global Feature Selection: Feature selection is the process of selecting a subset of features for improving the prediction performance of the model. It returns the subset of features that are useful for classifying the entire labeled samples, i.e., feature selection is "globally" informative. On the other hand, the influential scores are based on values of weights and nodes at a specific input. The most informative subsets are calculated from the scores. However, these scores have only information about the specific input so that these subsets only reflect "local" information.

Local and global feature selection can be very different depending on an input. For example, if one element of an input is 0, then its influential score is 0 because multiplying any weights to 0 will be 0. Even though being 0 may contain information, which can be reflected by a global feature selection,

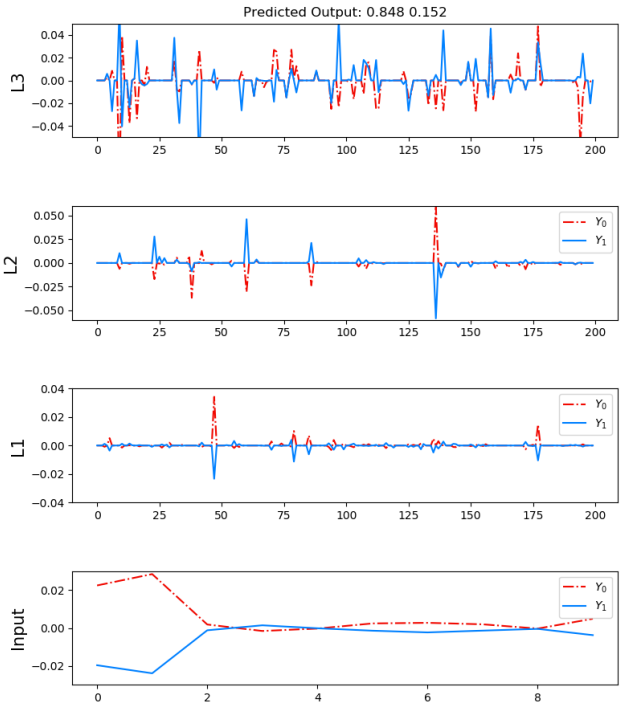

Fig. 4. The influential scores with respect to two output layer, Y_0 and Y_1, for input and 3 different hidden layers. Predicted output layer value is (0.848, 0.152). As it is a binary classification and the sum of Y_0 and Y_1 are equal to 1, the sign of high scores are different so that if Y_0 increases, it leads to decrease Y_1.

its importance should be limited in a neural network model locally.

Figure 3 shows this difference between global and local feature selection. As mentioned early, the inputs of two-features data sets are generated by a uniform distribution between 0 and 1. Some elements of an input have 0 or very close to it for X_0 and X_1. For these cases, the ranks of X_0 and X_1 can be higher than rank 0 and 1. The frequency of higher ranks above rank 2 is about $\approx 13\%$, which is approximately equal to the probability of getting a small number (< 0.06). On the other hand, inputs of plus-one data set do not contain any 0s. Therefore, the influential scores successfully select features locally and show the intrinsic characteristics of data sets.

Furthermore, the influential scores can have negative values as well as positive ones. The negative values imply that the input elements corresponding to them degrade the output layer values. For example, as X_0 is getting larger negative values, the influential score for Y_1 should be decreased. Although negative scores are informative, the rank of the score does not show their contribution to the output layer. Especially, when the input Xs have the standard Gaussian distribution, the possibilities for including 0s or negative values are non-trivial and it becomes difficult to select features using the rank of the influential scores.

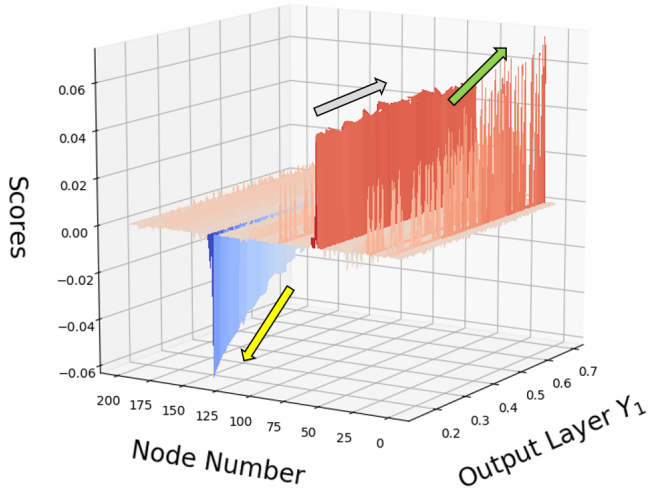

Fig. 5. The influential scores for L2 layer with respect to Y_1 while increasing Y_1. The negative and positive scores are depicted in blue and red colors respectively. The green, yellow and gray arrows are added to indicate the increasing, decreasing, and stationary peaks.

Hidden Layers: Figure 4 shows the influential scores for the input, L1, L2 and L3 layers with respect to the output layer. Most of the nodes are not activated and only some of them are contributed to the output layer. Interestingly, the number of significant peaks increases as layers are getting close to the output layer.

The sum of Y_0 and Y_1 is equal to 1, i.e., increasing Y_0 should decrease Y_1 to make their sum a constant. In the neural network models for binary classification, this highly correlated relations between two output nodes can be achieved in two different ways: sharing relevant nodes but having opposite effects on the output layer or having two entirely different routes. As you can see in Figure 4, the signs of important peaks in L1, L2, L3 are different. Thus, for the synthetic data, two elements, Y_0 and Y_1, of the output layer have similar routes to be activated by an input and they share the significant nodes.

Figure 5 shows how the model controls its output layer values. The 100 influential scores for L2 with respect to Y_1 are drawn as Y_1 increases from 0.1 to 0.7 to emphasize changes of peaks. Stationary peaks guarantee basic values for Y_1 and the negative peak (136^{th} node) is deep when Y_1 is small. On the other hand, the positive peak(9^{th} node) is high when Y_1 is large.

Not surprisingly, we can easily understand how the hidden layer, L1, controls the predictions. If the 136^{th} and the 9^{th} nodes have smaller influential scores, then Y_1 becomes small. If they have larger scores, Y_1 becomes large. The other peaks have less effect on Y_1 value, which implies that we do not need the whole 200 nodes in L2 layer.

B. MNIST

Data: The Modified National Institute of Standards and Technology database (MNIST) is a database of 28×28 size images of handwritten digits [9]. We use 60,000 images for training and 10,000 images for testing.

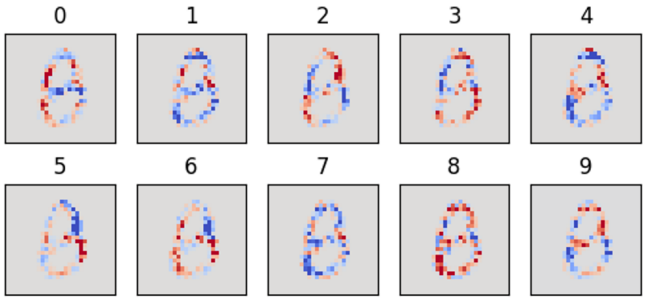

Fig. 6. The influential score images for a number eight. The red and blue pixels depict the positive and the negative scores respectively. The ninth image, with label 8, shows the informative pixels for the NN in classifying the input as number eight. The blue pixels in the other images suggest why the network does not classify the image as the label number above.

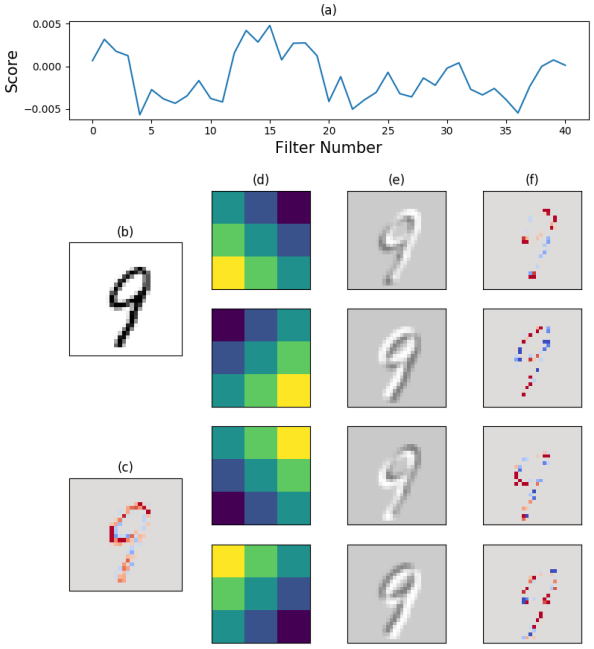

Fig. 7. (a) Influential scores for 41 different filters with respect to 10^{th} output node (filter scores). (b) the original image. (c) influential score for the input with respect to 10^{th} output node. (d) schematic images of $7^{th}, 15^{th}, 23^{rd}$, and 31^{st} filter. Filters are chosen to show the effects of varying degree of importances. (e) each filter is convoluted onto the original image. (f) influential scores for each filter with respect to 10^{th} output node.

Architecture: Jung et al. published a hybrid neural network model with a convolutional neural network and 41 general image kernels [10]. They substituted filters of the first convolutional layer with 41 pre-defined and commonly used image kernels such as sharpening, embossing, gradient operators, etc. Three max-pooling and two convolutional layers are followed: 64 3×3 filters, and 128 3×3 filters are used for each convolution layer respectively with appropriate paddings to keep the dimensions of images. We used 2×2 max pooling after convolutional layers. The first and second dense layers have 2048×625 and 625×10 dimensions respectively. We achieved

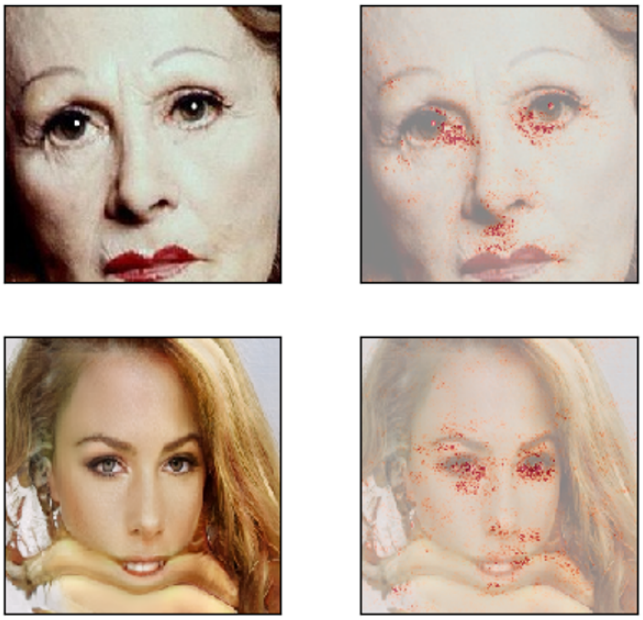

Fig. 8. Left images are original images. Upper images are real and bottom images are fake. Right images show the important position for the classification generated by the influential scores. Even though the fake image has no jaw, but the network model does not recognize.

$> 99.5\%$ accuracy with the test data sets.

Interpretability: We calculated the influential scores for the input layer with respect to the output layer. Each dot of the digit image has 10 different influential scores since there are 10 nodes in the output layer. We rearranged them into a 28×28 matrix, the same as the input dimension, for each output nodes. The 10 matrices, named as *input scores*, are shown in Figure 1 and Figure 6. The input scores for a digit 0 in Figure 1 have two types of dots: blue and red. The blue and the red dots depict the positive and the negative influential scores respectively. The positive score means that such dots positively contribute in classifying the input image to the given number. For example, the red dots in the first upper image in Figure 1 indicate why the input image should be classified as the number 0. The blue dots mean why the image should not be classified as the number 0. Similarly, the score images with respect to the second output node, i.e. the number 1, show why and why not the digit image should be the number 1.

The input scores explain why a digit image is classified as the number. At the same time, it shows why a digit image is *not* classified as the number. It is clearly shown in Figure 1 that there are blue dots outside of a number outline. Especially, red dots are located in only overlaid regions between an input digit image and number outline in all the images; it is more clear in $2^{nd}, 5^{th}, 6^{th}$, and 8^{th} images . Most of the dots are red for a number 0, which explains why the NN chooses a number 0 in this case.

Role of filters: We used the special hybrid architecture for classification of MNIST dataset. The reason we choose this model is to investigate how the NNs work with the influential scores. We can determine which filters are more important than

the others by simply summing the influential scores related to each filter.

We evaluated influential scores for the first convolutional layer, named as L1 scores, with respect to each output layer. We computed L1 scores for the 41 filters and named them as filter scores. As mentioned previously, these 41 filters are predefined. The filter scores indicate how important these filters are in the classification of a specific input image. Figure 7 (a) shows the filter scores. The 15^{th} filter, the first filter in (d), is the most important filter in classifying the original image (b) as the number nine. The 13^{th} and the 14^{th} filters are also important, meanwhile, the 4^{th} filter is the least important.

A role of the filters is to divide a digit image into parts depicted in Figure 7 (f). For example, a handwriting image of 9 is divided into four different parts by the four filters. Each part is weighted differently and reflected in the input score images. Comparing the image in Figure 7 (c) and (f), we can understand why red dots in (c) have higher influential scores. Red dots in (f) are features for the number nine.

C. GAN-generated Fake Image Detection

Data: 114,000 GAN generated fake face and 73,877 real face images are used for training and testing. Real face images are cropped from UMD face data set [11], [12]. We trained the model with 150,000 samples and tested with 37,877 samples.

Architecture: The input dimension is 224×224×3. Three convolution layers, followed by max-pooling layers, are used: 32 3×3 filters, 64 3×3 filters, and 128 3×3 filters are used for each convolution layer respectively with appropriate paddings to keep the dimensions of the images. We used 3x3 max-pooling for the first two pooling layers and 2×2 for the last layer after each convolution layer respectively. The dense layer has 625×2 dimensions.

Results: Even though GAN-generated images are passed through the discriminators, our model successfully detects fake images from real ones. However, we do not explain how the model can discriminate fake images. Therefore, we calculate influential scores for input layer with respect to output layers. The influential scores for each pixel with respect to real and fake nodes have the opposite sign as we have seen in the binary classification of the synthetic data set (data not shown here). Additionally, unlike MNIST data sets, the images have red, green, and blue channels. These channels have their own influential scores to infer color influences on the classification.

If specific regions of a fake image contribute to the fake output node, unnatural regions are most likely to be the regions. If the models find features that are not related to any regions in images, then the influential scores for the unnatural regions of fake images should be relatively small. The important pixels from input images are highlighted with red dots in Figure 8. The fake image at the bottom clearly has strange regions for humans eyes; the girl has no jaw. However, the important pixels based on the influential scores show somewhat unexpected results. For both real and fake images, the model investigates regions around eyes and wrinkles and makes decisions. That is, the model does not consider the shape of the face as an important feature. Instead, it seems to find other features that are useful to detect fake faces.

V. Conclusion

In case of the binary classification, the nodes in hidden layers have complementary influential scores with respect to output nodes as shown in the synthetic and GAN-generated face image data sets. The complementary influential scores have two implications: important nodes are shared by all instances and shifting these values eventually results in changes of outcomes.

Moreover, influential scores are an indicator of which features are crucial for a prediction. The 41 different predefined filters decompose an initial input into several parts as shown with MNIST data set. Interestingly, each piece has different importance, i.e., influential scores with respect to different digits. We suggested that the networks are searching the right parts in the right positions.

Finally, the prediction of networks can be interpreted using influential scores. Unlike traditional model interpretation methods, the influential scores provide both supporting and opposing reasons for the prediction as shown in Figure 1. The scores do not provide overall characteristics of networks but knowing how each instance behaves we can understand the global tendency. For example, there were only handful of important nodes for each instance in synthetic data. Tracing these nodes, we showed that small changes in an input cause enhancing or reducing the values of the output nodes.

Influential scores are a guiding tool that can shed light on the roles of hidden layers and help us optimizing an NN architecture and initializing its weights. Knowing important nodes inside layers and their connections, unnecessary nodes and weights can possibly be eliminated and a systematic initialization of the NN might be achievable.

In conclusion, we have introduced the influential scores and applied to interpret an outcome of the models. The influential scores are based on weights and node values calculated for a specific instance. We trained networks with synthetic, MNIST, and GAN-generated fake face data sets and interpreted them with the influential scores. We showed that the influential scores are practical tools to analyze any layers including hidden layers of neural networks.

Acknowledgment

This work was supported by MSIT, Korea under the ITCCP program (IITP-2019-2011-1-00783), by NRF of MSIT, Korea (2019R1F1A1058770), and by KEIT under the GATC program (10077300).

References

[1] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *Comput. VisionECCV 2014*, 2014.

[2] "Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1717–1724, 2014.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2013.

[4] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," pp. 1–8, 2013.
[5] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, pp. 1–46, 2015.
[6] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning Important Features Through Propagating Activation Differences," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, 2017, pp. 3145–3153.
[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*.
[8] J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2018, pp. 882–891.
[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.
[10] J. H. Jung, Y. Shin, and Y. Kwon, "Extension of convolutional neural network with general image processing kernels," in *TENCON*, 2018.
[11] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *CoRR*, 2017.
[12] A. Bansal, A. Nanduri, C. D. Castillo, R. Ranjan, and R. Chellappa, "Umdfaces: An annotated face dataset for training deep networks," *arXiv preprint arXiv:1611.01484v2*, 2016.