An Interpretation of Forward-Propagation and Back-Propagation of DNN

Guotian Xie[1,2] and Jianhuang Lai[1,2,3(✉)]

[1] The School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China
guotian.xgt@alibaba-inc.com
[2] Guangdong Key Laboratory of Information Security Technology, Guangzhou 510006, China
[3] The School of Information Science and Technology, Xinhua College, Sun Yat-sen University, Guangzhou, People's Republic of China
stsljh@mail.sysu.edu.cn

Abstract. Deep neural network (DNN) is hard to understand because the objective loss function is defined on the last layer, not directly on the hidden layers. To best understand DNN, we interpret the forward-propagation and back-propagation of DNN as two network structures, fp-DNN and bp-DNN. Then we introduce the direct loss function for hidden layers of fp-DNN and bp-DNN, which gives a way to interpret the fp-DNN as an encoder and bp-DNN as a decoder. Using this interpretation of DNN, we do experiments to analyze that fp-DNN learns to encode discriminant features in the hidden layers with the supervision of bp-DNN. Further, we use bp-DNN to visualize and explain DNN. Our experiments and analyses show the proposed interpretation of DNN is a good tool to understand and analyze the DNN.

Keywords: Forward-propagation · Back-propagation · Encoder Decoder

1 Introduction

In recent years, with the assist of hard-ware development, more and more applications are based on Deep learning, e.g., Compute vision [18], Audio Analysis [2], Nature Language Processing [8], Robots [13] and so on. Encouraging by its successes in widespread applications, Deep Learning (DL) and Deep Neural Networks (DNN) become a hot research topic among researchers and show its power comparing to other machine learning model, e.g., in these years, DL wins the first place on machine learning competitions on real data challenges [6,10] and even surpasses human beings on some tasks [7]. Despite such successes, we still know little about DNN, though it bases on a simple optimization techniques, Gradient Back-Propagation [11]. Although there are some feedback mechanisms

J.-H. Lai et al. (Eds.): PRCV 2018, LNCS 11257, pp. 3–15, 2018.
https://doi.org/10.1007/978-3-030-03335-4_1

proposed to train DNN [14,16,19], BP is still a popular way and DNN is still a black box [9,17,20].

The major obstacle of understanding how DNN learns knowledge is that, the objective function of the DNN is defined on last layer of DNN. It's not directly defined on the hidden layers. A loss function directly defining on the hidden layers could help us to understand what knowledge will be learned. For example, researchers know that the objective loss function of SVM [5] is to learn a hyperplane to segment the dataset into two classes. That's because the objective loss function is defined directly on the linear model $y = \mathbf{W}\mathbf{x} + b$.

Inspire by that, we define a direct loss function for the hidden layers of DNN. First, we interpret the forward-propagation and back-propagation of DNN as two network structures, denoted as fp-DNN and bp-DNN. Then we define direct loss function for the hidden layers of fp-DNN and bp-DNN respectively, which uncovers the fact that fp-DNN acts as a role of encoder, while bp-DNN acts as a role of decoder. In this interpretation, fp-DNN and bp-DNN generate targets to supervise the training of each other, as showing in Fig. 2. Here, the word "targets", which is also used in [3,12], represents the signal to supervise the training.

In experiments, we use the proposed interpretation of fp-DNN and bp-DNN to analyze DNN. First, we analyze the bp-DNN generate discriminant targets to supervise the fp-DNN to learn to encode the discriminant features. Then, we visualize the distribution of the encoded features of fp-DNN to verify that fp-DNN does learn to encode discriminant features. Finally, we use bp-DNN to visualize the DNN to do some analyses. The experimental results show that the proposed interpretation is a good tool for analyzing DNN.

Our contributions in this paper are as follows,

1. We interpret the forward-propagation and the backward-propagation of DNN as two network structures, fp-DNN and bp-DNN, respectively. By defining the direct loss function for hidden layers of fp-DNN and bp-DNN, we think that the fp-DNN acts as a role of an encoder, supervising the training of the bp-DNN, while the bp-DNN acts as a role of a decoder, supervising the training of the fp-DNN. This interpretation helps us to understand the DNN. For example, we could explain why the hidden layers of fp-DNN learns to encode the discriminant features.
2. Since the bp-DNN acts as a decoder, It could be used to visualized what knowledge the DNN have learned. However, the bp-DNN has some disadvantages for visualization. So we propose the guided-bp-DNN for knowledge visualization of DNN, and our experiments show that the visualization using guided-bp-DNN could focus on the important patterns for recognition.

Notation: We use bold lower case letters to represent a column based vector, e.g., $\mathbf{x}$, bold capital letters to represent a matrix, e.g., $\mathbf{W}$. we denote a function as $f_{\mathbf{\Theta}}(\mathbf{x})$, where $\mathbf{\Theta}$ is the parameters of this function and $\mathbf{x}$ is the input.

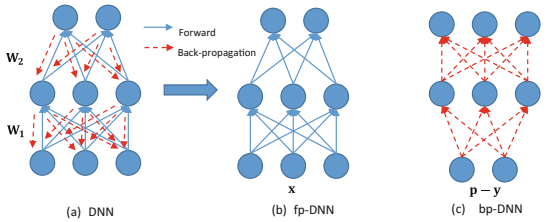

Fig. 1. (a) The normal DNN training procedure contains two steps, forward(blue) and backward(red), which forms two network sharing weights. (b) The network fp-DNN represents the forward pass, extracting features (c) The network bp-DNN has the inverted structure of fp-DNN, but sharing the same parameters, which is for transporting the gradients (or label information) from top to the bottom. (Color figure online)

2 Formulation of Deep Neural Networks

Classification is a basic task for Machine Learning. In this paper, we use DNN to model the classification task and analyze how DNN is trained. We assume a classification task with C classes, with a training data set $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ that contains N training samples. Where $\mathbf{x} \in R^S$ is the input signal and $\mathbf{y} \in \{0,1\}^C$ is the class label of $\mathbf{x}$, with $y_c = 1$ if $\mathbf{x}$ belongs to the cth class and otherwise $y_i = 0, i \neq c$. The classification task for this data set is to train a DNN to predict the conditional distribution $p(\mathbf{y}|\mathbf{x}) = f_{\mathbf{\Theta}}(\mathbf{x})$, where $f_{\mathbf{\Theta}}(\mathbf{x})$ is the function of DNN. We denote $\mathbf{p} = [p_0, p_1, \ldots, p_C]^T \in R^C$ as the output of $f_{\mathbf{\Theta}}(\mathbf{x})$ for convenience, with $p_i = p(y_i|\mathbf{x})$.

To solve this classification task, we construct a model of deep neural network with L hidden layers, and formulate it as [1], (Fig. 1(a))

$$\text{DNN} = \begin{cases} \ell_{\mathbf{\Theta}}(\mathbf{x}, \mathbf{y}) = \sum_i^C y_i \log p_i \\ p_i = \frac{e^{z_{L,i}}}{\sum_j^C e^{z_{L,j}}} \\ \mathbf{z}_1 = \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1 \\ \mathbf{z}_l = \mathbf{W}_l \sigma(\mathbf{z}_{l-1}) + \mathbf{b}_l, 2 \leq l \leq L \end{cases} \tag{1}$$

where $\mathbf{W}_l \in R^{C_l \times C_{l-1}}$ and $\mathbf{b}_l \in R^{C_l}$ is the parameters of the lth layer of DNN, and $\ell_{\mathbf{\Theta}}(\mathbf{x}, \mathbf{y})$ is the softmax loss function. $\mathbf{\Theta}$ is all the parameters of DNN. $\mathbf{z}_L = [z_{L,1}, z_{L,2}, \ldots, z_{L,C}]^T$ is the linear output of DNN. $\mathbf{z}_l \in R^{C_l}$ is the linear output of the lth hidden layer, and $\mathbf{p} = [p_1, p_2, \cdots, p_C]^T$ is the final prediction of this DNN.

We define two network structures corresponding to the training process with forward and back-propagation,

$$\text{fp-DNN} = \begin{cases} \mathbf{z}_0 = \mathbf{x} \\ \mathbf{z}_1 = \mathbf{W}_1 \mathbf{z}_0 + \mathbf{b}_1 \\ \mathbf{z}_l = \mathbf{W}_l \sigma(\mathbf{z}_{l-1}) + \mathbf{b}_l, 2 \leq l \leq L \end{cases} \tag{2}$$

$$\text{bp-DNN} = \begin{cases} \tilde{\mathbf{z}}_L = \mathbf{p} - \mathbf{y} \\ \tilde{\mathbf{z}}_{L-1} = \mathbf{W}_L^T \tilde{\mathbf{z}}_L \\ \tilde{\mathbf{z}}_{l-1} = \mathbf{W}_l^T (\tilde{\mathbf{z}}_l \odot \tilde{\mathbf{b}}_l), L-1 \geq l \geq 1 \end{cases} \tag{3}$$

where $\tilde{\mathbf{b}}_l$ is the derivation of non-linear activation function of the lth layer, $\tilde{\mathbf{b}}_l = \frac{\partial \sigma(\mathbf{z}_l)}{\partial \mathbf{z}_l}$. Since bp-DNN has an inverted structure, showing in Fig. 1, we use an inverted order to number the layers of bp-DNN, i.e., the Lth layer is the first layer of bp-DNN, while the 0th layer is the last layer of bp-DNN.

As the definition of fp-DNN and bp-DNN shows, they share the parameters $\mathbf{W}_l, l = 1, \cdots, L$. See Fig. 1(b) and (c) for specification. Compare (b) and (c), the structure of fp-DNN and bp-DNN is similar but inverted. For convenience, we denoted $\mathbf{o}_l$ and $\tilde{\mathbf{o}}_l$ as the non-linear output of the l^{th} layer of fp-DNN and bp-DNN, respectively,

$$\mathbf{o}_l = \begin{cases} \mathbf{z}_0, l = 0 \\ \sigma(\mathbf{z}_l), \text{otherwise} \end{cases}, \tilde{\mathbf{o}}_l = \begin{cases} \tilde{\mathbf{z}}_L, l = L \\ \tilde{\mathbf{z}}_l \odot \tilde{\mathbf{b}}_l, \text{otherwise} \end{cases} \tag{4}$$

Since bp-DNN is an interpretation of back-propagation of DNN, we have $\frac{\partial \ell_{\Theta}}{\partial \mathbf{z}_l} = \tilde{\mathbf{o}}_l$.

The symmetrical but inverted structures of fp-DNN and bp-DNN remind us the auto-encoder network structures, which contain a part of network acting as a role of encoder, and another part of network acting as a role of decoder. In analyses of the next section, we will found that fp-DNN does act as a role of encoder, while bp-DNN does act as a role of decoder.

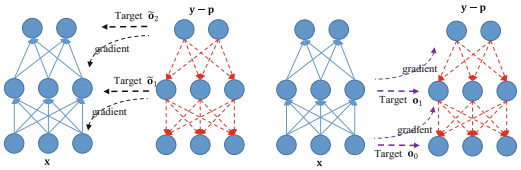

Fig. 2. fp-DNN and bp-DNN are supervisor for each other. (a) From the view of fp-DNN, bp-DNN generates targets to supervise the training of each hidden layers of fp-DNN. (b) From the view of bp-DNN, fp-DNN generates targets to supervise the training of each hidden layers of bp-DNN.

3 fp-DNN and bp-DNN vs Encoder and Decoder

In this section, we will give an interpretation that bp-DNN acts as a role of decoder and fp-DNN acts as a role of encoder, by defining the direct loss function for hidden layers of fp-DNN and bp-DNN respectively.

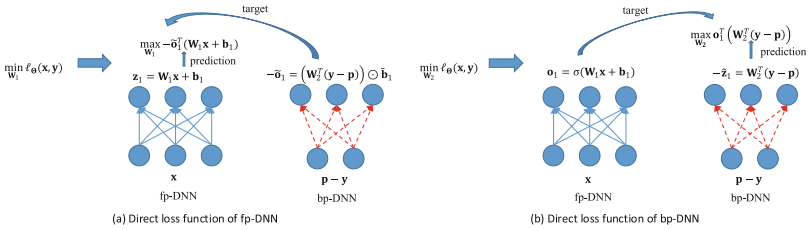

Fig. 3. The direct loss function for fp-DNN is $\tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, -\tilde{\mathbf{o}}_l) = -\tilde{\mathbf{o}}_l^T(\mathbf{W}_l\mathbf{o}_{l-1} + \mathbf{b}_l)$, while the direct loss function for bp-DNN is $\tilde{\ell}^{bp}_{\mathbf{W}_l}(-\tilde{\mathbf{o}}_l, \mathbf{o}_{l-1}) = -\mathbf{o}_{l-1}^T(\mathbf{W}_l^T\tilde{\mathbf{o}}_l)$. Here we show an instance of the direct loss function of the DNN network described in Fig. 1. The fp-DNN and bp-DNN in both (a) and (b) are shown only part of the fp-DNN and bp-DNN in Fig. 1. (a) The bp-DNN generates the targets $\tilde{\mathbf{o}}_1$ to supervise the fp-DNN to learn the $\mathbf{W}_1$. (b) The fp-DNN generates the targets $\mathbf{o}_1$ to supervise the bp-DNN to learn the $\mathbf{W}_2$.

3.1 Direct Loss Function for Hidden Layer of fp-DNN

The DNN is optimized on the loss function $\ell_{\mathbf{\Theta}}(\mathbf{x}, \mathbf{y}) = \sum_i^C y_i \log p_i$. If we interpret $\ell_{\mathbf{\Theta}}(\mathbf{x}, \mathbf{y})$ into a direct loss function of hidden layer, we can understand easily how the hidden layer is trained. We define the direct loss function $\ell^{fp}_{\mathbf{W}_l}$ of lth hidden layer, so that it has the same gradient w.r.t $\mathbf{W}_l$, i.e., $\frac{\partial \ell^{fp}_{\mathbf{W}_l}}{\partial \mathbf{W}_l} = \frac{\partial \ell_{\mathbf{\Theta}}(\mathbf{x},\mathbf{y})}{\partial \mathbf{W}_l}$

When we want to optimize the parameters $\mathbf{W}_l$ of l_{th} hidden layer, we use the gradient descent, $\mathbf{W}_l = \mathbf{W}_l - \eta \Delta \mathbf{W}_l$, where η is the learning rate and $\Delta\mathbf{W}_l$ is the gradient back-propagated from the top layer, and has the form as follows,

$$\begin{aligned} \Delta \mathbf{W}_l &= \frac{\partial \ell_{\mathbf{\Theta}}}{\partial \mathbf{z}_l}\frac{\partial \mathbf{z}_l^T}{\partial \mathbf{W}_l} = \frac{\partial \ell_{\mathbf{\Theta}}}{\partial \mathbf{z}_l}\sigma(\mathbf{z}_{l-1})^T = \tilde{\mathbf{o}}_l\mathbf{o}_{l-1}^T \\ \Delta \mathbf{b}_l &= \frac{\partial \ell_{\mathbf{\Theta}}}{\partial \mathbf{z}_l}\frac{\partial \mathbf{z}_l^T}{\partial \mathbf{b}_l} = \frac{\partial \ell_{\mathbf{\Theta}}}{\partial \mathbf{z}_l} = \tilde{\mathbf{o}}_l \end{aligned} \tag{5}$$

By observation, we found that the gradient $\Delta\mathbf{W}_l$ and $\Delta\mathbf{b}_l$ is equivalent to the gradient of the loss function $\tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, \tilde{\mathbf{o}}_l)$ w.r.t. $\mathbf{W}_l$ and $\mathbf{b}_l$,

$$\tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, \tilde{\mathbf{o}}_l) = \tilde{\mathbf{o}}_l^T(\mathbf{W}_l\mathbf{o}_{l-1} + \mathbf{b}_l) \tag{6}$$

where $\mathbf{o}_{l-1} = \sigma(\mathbf{z}_{l-1})$ is the input signal and $\tilde{\mathbf{o}}_l = \frac{\partial \ell_{\mathbf{\Theta}}(\mathbf{x},\mathbf{y})}{\partial \mathbf{z}_l}$ is the new target of the lth hidden layer, and we view $\mathbf{o}_{l-1}$ and $\tilde{\mathbf{o}}_l$ as constants w.r.t. $\mathbf{W}_l$ and $\mathbf{b}_l$. Obviously,

$$\begin{aligned} \Delta \mathbf{W}_l &= \frac{\partial \tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, \tilde{\mathbf{o}}_l)}{\partial \mathbf{W}_l} = \frac{\partial \ell_{\mathbf{\Theta}}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{W}_l} \\ \Delta \mathbf{b}_l &= \frac{\partial \tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, \tilde{\mathbf{o}}_l)}{\partial \mathbf{b}_l} = \frac{\partial \ell_{\mathbf{\Theta}}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{b}_l} \end{aligned} \tag{7}$$

$$\min_{\mathbf{W}_l, \mathbf{b}_l} \ell_{\mathbf{\Theta}}(\mathbf{x}, \mathbf{y}) \equiv \min_{\mathbf{W}_l, \mathbf{b}_l} \tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, \tilde{\mathbf{o}}_l) \equiv \max_{\mathbf{W}_l, \mathbf{b}_l} \tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, -\tilde{\mathbf{o}}_l) \tag{8}$$

Please note that, $\tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, -\tilde{\mathbf{o}}_l)$ defines a metric based on the cosine distance between $-\tilde{\mathbf{o}}_l$ and $\mathbf{z}_l = \mathbf{W_l}\mathbf{o}_{l-1} + \mathbf{b}_l$. Maximizing the cosine distance between the vectors $-\tilde{\mathbf{o}}_l$ and $\mathbf{z}_l$ is to minimize the angle of these two vectors. One example is shown in Fig. 3 (a).

In conclusion, the direct loss function for the lth hidden layer of fp-DNN is $\tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, -\tilde{\mathbf{o}}_l)$, which use a target $-\tilde{\mathbf{o}}_l$ generated by the bp-DNN. The objective function $\tilde{\ell}^{fp}_{\mathbf{W}_l}$ trains the $\mathbf{W}_l$ so that the output $\mathbf{z}_l$ of this hidden layer is close to the target $-\tilde{\mathbf{o}}_l$ in cosine distance metric. The fp-DNN is going to map to information of the output label $\mathbf{y}$ layer by layer during the training process, with the supervision of bp-DNN. As a result, fp-DNN could be interpreted as acting the role of the encoder, encoding the information of input $\mathbf{x}$ into the information of label signal $\mathbf{y}$.

3.2 Direct Loss Function for Hidden Layer of bp-DNN

Similar to the definition of direct loss function of fp-DNN, we define the direct loss function of bp-DNN to have the same gradient w.r.t. $\mathbf{W}_l$.

$$\tilde{\ell}^{bp}_{\mathbf{W}_l}(-\tilde{\mathbf{o}}_l, \mathbf{o}_{l-1}) = -\mathbf{o}^T_{l-1}(\mathbf{W}^T_l \tilde{\mathbf{o}}_l) \tag{9}$$

$$\min_{\mathbf{W}_l} \ell_{\mathbf{\Theta}}(\mathbf{x}, \mathbf{y}) \equiv \max_{\mathbf{W}_l} \tilde{\ell}^{bp}_{\mathbf{W}_l}(-\tilde{\mathbf{o}}_l, \mathbf{o}_{l-1}) \tag{10}$$

Comparing to Eq. (6), It's a symmetric form of $\tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, -\tilde{\mathbf{o}}_l)$. $\tilde{\ell}^{bp}_{\mathbf{W}_l}(-\tilde{\mathbf{o}}_l, \mathbf{o}_{l-1})$ also defines a metric based on the cosine distance between $\mathbf{o}_{l-1}$ and $-\tilde{\mathbf{z}}_{l-1} = -\mathbf{W}_l\tilde{\mathbf{o}}_l$. It minimizes the angle of these two vectors. On example is shown in Fig. 3(b).

In conclusion, the direct loss function for the lth hidden layer of bp-DNN is $\tilde{\ell}^{bp}_{\mathbf{W}_l}(-\tilde{\mathbf{o}}_l, \mathbf{o}_{l-1})$, which use a target $\mathbf{o}_{l-1}$ generated by the fp-DNN. The objective function $\tilde{\ell}^{bp}_{\mathbf{W}_l}$ trains the $\mathbf{W}_l$ so that the output $-\tilde{\mathbf{z}}_{l-1}$ of this hidden layer is close to the target $\mathbf{o}_{l-1}$ in cosine distance metric. The bp-DNN is going to reconstruct the information of the input signal $\mathbf{x}$ layer by layer during the training process, with the supervision of fp-DNN. As a result, bp-DNN could be interpreted as acting the role of decoder, decoding the information of label $\mathbf{y}$ into the information of input signal $\mathbf{x}$.

3.3 Interpretation of Forward-Propagation and Back-Propagation

As discussion above, we first interpret the forward-propagation and back-propagation of DNN to be corresponding to two network structures, fp-DNN and bp-DNN. Using the direct loss function for hidden layers of fp-DNN and bp-DNN, we further interpret training process of DNN as the process of collaboration of training the encoder and decoder, i.e., the targets generated by bp-DNN (decoder) supervises the training of fp-DNN (encoder), while the targets generated by fp-DNN supervises the training of bp-DNN, see Fig. 2 for specification. The structures of this interpretation shows a similar form as the stacked

auto-encoder [4], both with the encoder and decoder network structures. However, the major difference between our interpretation and stacked auto-encoder is that the input signal for the decoder. The input signal for decoder of stacked auto-encoder is the code $\mathbf{p}$ generated by encoder, while the input signal for the decoder (bp-DNN) of our interpretation contains the label information, $\mathbf{p} - \mathbf{y}$. With the supervision of label information, the encoder (fp-DNN) of our interpretation learns to encode the discriminant features, while encoder of stacked auto-encoder learns to encode features with maximum information for reconstruction.

This interpretation could help us understand the training process of DNN. For example, by analyzing the distribution of the targets $\tilde{\mathbf{o}}_l$ generated by bp-DNN, we can know how the bp-DNN guides the fp-DNN to encode discriminant features layer by layer; by analyzing the reconstruction of input signal $\mathbf{x}$ from the bp-DNN, we can learn what the network have learned. In next section, we use this interpretation of DNN to do some analyses.

4 Experiments

We designed a DNN with six layers, denoted as DNN-6, of which all layers are fully connection layers. The number of output channels of each hidden layers of DNN-6 is set as 100, and we use ReLU as the non-linear activation function. We train DNN-6 on MNIST [11] digits dataset and use it to show the collaboration of training process of the encoder and decoder. We train DNN-6 for totally 10000 iterations, with batch size as 64. The final accuracy of DNN-6 on the test set of MNIST is 97.51%.

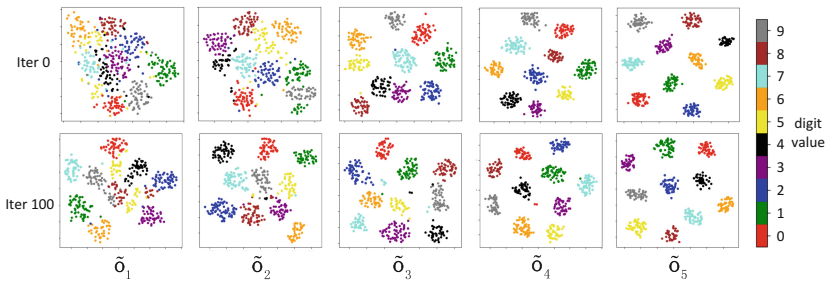

Fig. 4. In the 0th and 100th iteration, the targets $\tilde{\mathbf{o}}_l$ generated by bp-DNN have the discriminant property. This shows that targets generated by bp-DNN preserve the discriminant property of label information during training. Here, points with the same color come from the same category, e.g., red points belong to the category of digit 0. (Color figure online)

4.1 Explanation of the Encoder (fp-DNN)

Why fp-DNN learns to encode discriminant features? We think the reason is that the targets generated by bp-DNN, which supervise the training of fp-DNN, are discriminant. In this discussion, we conduct experiments to verify the targets generated by bp-DNN are discriminant, even when the weights of bp-DNN are initialized randomly.

The direct loss function for fp-DNN is $\tilde{\ell}^{fp}_{\mathbf{W}_l}(\mathbf{o}_{l-1}, -\tilde{\mathbf{o}}_l) = -\tilde{\mathbf{o}}_l^T(\mathbf{W}_l\mathbf{o}_{l-1} + \mathbf{b}_l)$, where $\tilde{\mathbf{o}}_l$ is the supervised information generated by bp-DNN and $\mathbf{z}_l = \mathbf{W}_l\mathbf{o}_{l-1} + \mathbf{b}_l$ is the learned encoded features of fp-DNN. Since the fp-DNN is supervised by the bn-DNN, analyzing the properties of the target $\tilde{\mathbf{o}}_l$ generated by the bp-DNN can tell us what a kind of fp-DNN will be trained using bp-DNN as supervisor. The input of bp-DNN is the label information $\mathbf{p}-\mathbf{y}$, so the target $\tilde{\mathbf{o}}_l$ generated by bp-DNN is a function of the label information. As we know, the most important property of label information is the discriminant property. Here we want to use the t-SNE [15] technique to visualize the target $\tilde{\mathbf{o}}_l$ to explore whether $\tilde{\mathbf{o}}_l$ preserve the discriminant property of label information $\mathbf{p}-\mathbf{y}$. We use the DNN-6 to generate the corresponding bp-DNN and fp-DNN, and use them to extract the target $\tilde{\mathbf{o}}_l$ and the encoded feature $\mathbf{z}_l$.

First, we will conduct experiments that to show the $\tilde{\mathbf{o}}_l$ preserve the discriminant property of label information during training.

Then we further visualize the distribution of the encoded feature $\mathbf{z}_l$, to verify that the encoded features are discriminant under the supervision of the discriminant target $\tilde{\mathbf{o}}_l$ generated by bp-DNN.

Since the direct loss function is based on the cosine distance metric, we also set the distance metric of t-SNE as cosine distance metric for reducing dimension.

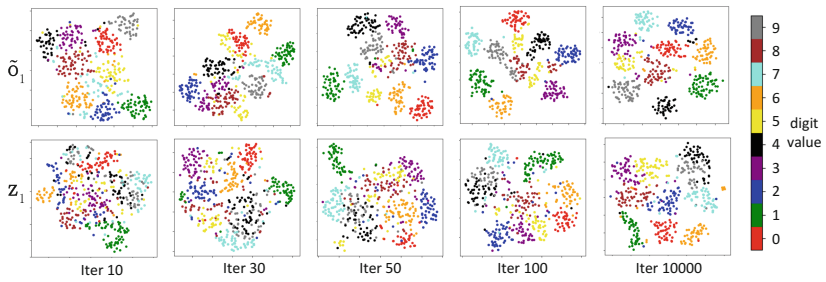

Fig. 5. The learned feature $\mathbf{z}_1$ of fp-DNN is going to be more discriminant during the training, under the supervision of the discriminant target $\tilde{\mathbf{o}}_1$. Finally, the distribution of $\mathbf{z}_1$ is close to the distribution of $\tilde{\mathbf{o}}_1$ in the iteration 10000.

Visualize the Discriminant Target $\tilde{\mathbf{o}}_l$. This experiment is to show that the target $\tilde{\mathbf{o}}_l$ generated by bp-DNN is discriminant in varying degrees. We visualize the $\tilde{\mathbf{o}}_l, l = 1, \cdots, 5$ in the 2D plane, using t-SNE to reduce dimension, as showing

in Fig. 4. In Fig. 4, we show the distribution of the target of 5 layers, with $l = 1, \cdots, 5$, with the first column showing the distribution of the target of $l = 1$th layer, the second column showing the distribution of the target of the $l = 2$th layer and so on. We also show the distribution of the target in different training iteration, and specifically, we show the distribution in 0th iteration and the 100th iteration. More distributions in other iterations are not shown for the limitation of paper length, but we are sure that distribution in other iterations shows consistent agreements with those in the 0th iteration and 100th iteration.

In Fig. 4, points with the same colour are in the same category. The points in the same cluster are more close to each other, then the distribution of these points are more discriminant. From the visualization in Fig. 4, we have such observation,

First, the target $\tilde{\mathbf{o}}_l$ generated from bp-DNN does have the discriminant property for all layers $l = 1, \cdots, 5$. But the degree of discriminant properties of each layer are varying, where the distribution of target in higher layers ($l \geq 3$) seems more discriminant than that of bottom layers ($l \leq 3$).

Second, the discriminant property is preserved during training, for example, in the iteration 100, $\tilde{\mathbf{o}}_l$ preserve the discriminant property as well, as Fig. 4 showing. Specifically, after training, the target has the trends to be more discriminant. For example, the targets $\tilde{\mathbf{o}}_1$ and $\tilde{\mathbf{o}}_2$ in 100th iteration are more discriminant than those in the 0th iteration.

In conclusion, the targets generated by bp-DNN preserve the discriminant property of label information in varying degrees. Naturally, the encoded feature $\mathbf{z}_l$ of fp-DNN is expected to be discriminant after training, since it's supervisor is discriminant. Next, we conduct experiments to visualize the encoded features to verify this assumption.

Visualize the Discriminant Feature $\mathbf{z}_l$. First, we show the encoded features $\mathbf{z}_1$ in the first layer during the training iteration 10, 30, 50, 100 and 10000, as showing in the second row of Fig. 5. For comparison, we show the corresponding targets $\tilde{\mathbf{o}}_1$ for supervision in the first row of Fig. 5. In Fig. 5, the distribution of the learned features $\mathbf{z}_1$ becomes more and more discriminant during the training. When the training iteration is finished, the learned feature $\mathbf{z}_1$ run into a distribution similar to the target $\tilde{\mathbf{o}}_l$. Specifically, in iteration 10000, the distribution of $\mathbf{z}_1$ achieves a similar degree of discriminant property as the distribution of $\tilde{\mathbf{o}}_1$. The fp-DNN does learn to encode the input signal to a feature space with discriminant property.

Then, in the second row of Fig. 6, we show the distribution of $\mathbf{z}_l, l = 1, \cdots, 5$ after training for 10000 iteration. For comparison, the corresponding distribution of $\mathbf{z}_l, l = 1, \cdots, 5$ in the iteration 0 is shown in the first row of Fig. 6, which are far from discriminant. Comparing to the distribution of $\mathbf{z}_l, l = 1, \cdots, 5$ in iteration 0, the distribution of $\mathbf{z}_l, l = 1, \cdots, 5$ in iteration 10000 does show the discriminant property as Fig. 6 showing. And we think that the reason of fp-DNN learning discriminant features in hidden layer is the target generated by

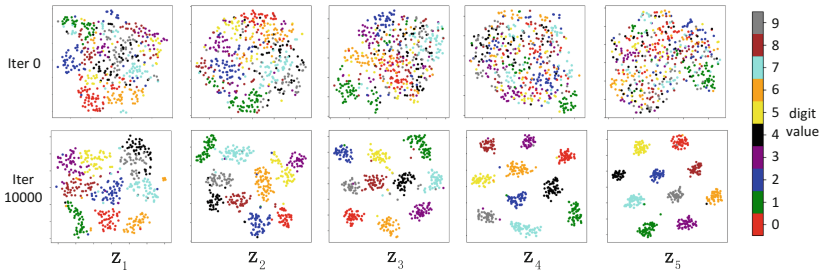

Fig. 6. Before training, features of fp-DNN don't have discriminant property. After training, features of all hidden layers of fp-DNN have varying degrees of discriminant property. This is because fp-DNN learns to encode features under the supervision of discriminant targets generated by bp-DNN.

bp-DNN, which preserve the discriminant property and supervise the fp-DNN to learn discriminant features.

4.2 Explanation of the Decoder (bp-DNN)

Since bp-DNN is a decoder network, we can use bp-DNN to map $\mathbf{y}$ back into the input signal space $\hat{\mathbf{x}}$. By visualizing the reconstruction $\hat{\mathbf{x}}$, we can know which part of $\mathbf{x}$ is important for the DNN to recognize that it belongs to the category $\mathbf{y}$. However, the direct reconstruction loss function of bp-DNN is to train $\mathbf{W}_l$ to reconstruct all training samples $\mathbf{o}_{l-1}$ as much as possible. This leads the reconstruction to be an average version of $\mathbf{o}_{l-1}$ and not only relative to the specific input signal $\mathbf{x}$. An average version of $\mathbf{o}_{l-1}$ could be a bad visualization for human.

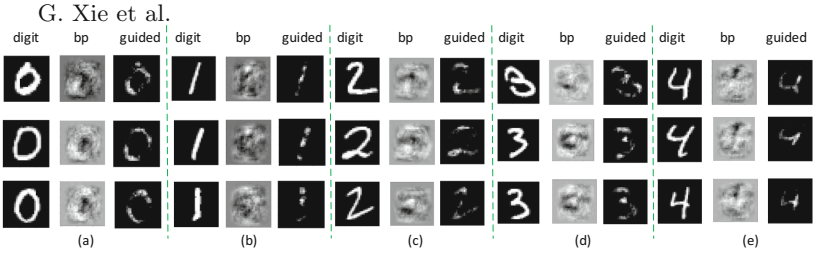

Fig. 7. Visualization using guided-bp-DNN could show clearly which parts of the digit images are important for DNN to recognize, while visualization of bp-DNN is not clear and hard to read.

To only visualize those reconstructions that are relative to a specific signal $\mathbf{x}_i$, we need to add some constrains. Here, we select the reconstructions

that are mostly matched with the input. To achieve this objective, we introduce an indicator to measure the degree of matching between the reconstructed feature $\tilde{\mathbf{z}}_{l-1}$ and the feature $\mathbf{z}_{l-1}$, that is, $\mathbf{1}_{sign(\mathbf{z}_l)=sign(\tilde{\mathbf{z}}_l)}$. Let's give an example. If $\mathbf{z}_l = [0.5, 0.3, -0.1, -0.2]^T$ and $\tilde{\mathbf{z}}_l = [0.5, -0.3, -0.1, 0.2]^T$, the $\mathbf{1}_{sign(\mathbf{z}_l)=sign(\tilde{\mathbf{z}}_l)} = [1, 0, 1, 0]^T$. With the $\mathbf{1}_{sign(\mathbf{z}_l)=sign(\tilde{\mathbf{z}}_l)}$ to measure the degree of matching, we select the reconstructed feature as $\hat{\mathbf{z}}_l = \mathbf{1}_{sign(\mathbf{z}_l)=sign(\tilde{\mathbf{z}}_l)} \odot \tilde{\mathbf{z}}_l$. The bp-DNN could be modified as, denoted as guided-bp-DNN,

$$\text{Guided-bp-DNN} = \begin{cases} \tilde{\mathbf{z}}_L = \mathbf{y} \\ \tilde{\mathbf{z}}_{L-1} = \mathbf{W}_L^T \tilde{\mathbf{z}}_L \\ \hat{\mathbf{z}}_l = \mathbf{1}_{sign(\mathbf{z}_l)=sign(\tilde{\mathbf{z}}_l)} \odot \tilde{\mathbf{z}}_l, l \geq 0 \\ \tilde{\mathbf{z}}_{l-1} = \mathbf{W}_l^T(\hat{\mathbf{z}}_l \odot \tilde{\mathbf{b}}_l), L-1 \geq l \geq 1 \end{cases} \tag{11}$$

We will give experiments to show the advantages of guided-bp-DNN for visualization. By the way, when using ReLU as non-linear activation function, guided-bp-DNN is the same as the guided-Back-Propagation proposed in paper [21] for visualization. That is, guided-Back-Propagation is a specific case of our guided-bp-DNN.

Comparison Between bp-DNN and Guided-bp-DNN. We use DNN-6 for visualization on the MNIST. In Fig. 7, we show the visualization results using bp-DNN and guided-bp-DNN respectively, showing the visualization results of five digits $(0, 1, 2, 3, 4)$, each digit with three instances. The first column shows the origin images of the digit, while the second column and third column shows the visualization results using bp-DNN and guided-bp-DNN respectively. From Fig. 7, we can found that the visualization results of bp-DNN could be a mess and hard for us to understand. Only the visualization of digit 1 and digit 3 could be recognized to has a shape of digit 1 and digit 3. As mentioned previously, this mess is caused by the fact that bp-DNN is trained to reconstruct an average version of all training samples.

On the contrary, guided-bp-DNN use $\mathbf{1}_{sign(\mathbf{z}_l)=sign(\tilde{\mathbf{z}}_l)}$ to guide the reconstruction to be close to the specific input signal. The visualization of guided-bp-DNN in Fig. 7 could be readably for human. In Fig. 7, The highlight in visualization of guided-bp-DNN shows that the part of the digit covering by the highlight is an important pattern for the DNN to recognize this digit. In next section, we use guided-bp-DNN to analyze what DNN have learned to recognize the digit.

What Patterns are Important for DNN to Recognize? In Fig. 8, we use guided-bp-DNN to visualize what part of the digit the DNN think is important to recognize. In Fig. 8, the column 'digit' is the origin image of those digits, and the column 'guided' is the visualization results of guided-bp-DNN. In the origin images of digits, we use a green/red box to mark out the part that is highlighted by the guided-bp-DNN visualization. In Fig. 8, we show two types of patterns that is important to recognize digits. One is the intersection part where two or more strokes intersect, which is marked with red box in Fig. 8. The other is the

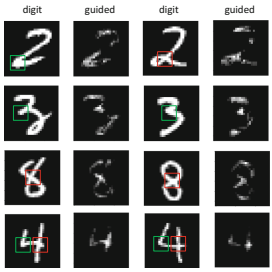

Fig. 8. Through the visualization using guided-bp-DNN, we found that two types of patterns are important for DNN to recognize digits, i.e., the intersection parts of strokes (red box) and the turning parts of strokes (green box). These two types of patterns contain rich information for recognition. (Color figure online)

turning part where the strokes change the direction suddenly, which is marked with green box. The DNN learns these two types of patterns to recognize digits is reasonable, because there are rich information in the intersection parts and the turning parts.

5 Conclusion

We proposed an interpretation of DNN into two networks structures, fp-DNN and bp-DNN. By introducing direct loss function for hidden layers of fp-DNN and bp-DNN, fp-DNN could be interpreted to act as a role of encoder while bp-DNN acts as a role of decoder. Using this interpretation, we could explain how DNN learn discriminant features in the hidden layer. We also use the proposed guided-bp-DNN to analyze what have learned in DNN.

Acknowledgments. This project is supported by the Natural Science Foundation of China (61573387) and Guangdong Project (2017B030306018).

References

1. An, S., Boussaid, F., Bennamoun, M., Hu, J.: From deep to shallow: transformations of deep rectifier networks. arXiv preprint arXiv:1703.10355 (2017)
2. Arik, S.O., et al.: Deep voice: real-time neural text-to-speech. arXiv preprint arXiv:1702.07825 (2017)
3. Bengio, Y.: How auto-encoders could provide credit assignment in deep networks via target propagation. arXiv preprint arXiv:1407.7906 (2014)
4. Bengio, Y., et al.: Learning deep architectures for Ai. Found. Trends® Mach. Learn. **2**(1), 1–127 (2009)
5. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)

7. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
8. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T.: Fasttext.zip: compressing text classification models. arXiv preprint arXiv:1612.03651 (2016)
9. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. arXiv preprint arXiv:1703.04730 (2017)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
12. Lee, D.-H., Zhang, S., Fischer, A., Bengio, Y.: Difference target propagation. In: Appice, A., Rodrigues, P.P., Santos Costa, V., Soares, C., Gama, J., Jorge, A. (eds.) ECML PKDD 2015. LNCS (LNAI), vol. 9284, pp. 498–515. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23528-8_31
13. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. Int. J. Robot. Res. **37**, 421–436 (2016). https://doi.org/10.1177/0278364917710318
14. Lillicrap, T.P., Cownden, D., Tweed, D.B., Akerman, C.J.: Random feedback weights support learning in deep neural networks. arXiv preprint arXiv:1411.0247 (2014)
15. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(Nov), 2579–2605 (2008)
16. Nøkland, A.: Direct feedback alignment provides learning in deep neural networks. In: Advances in Neural Information Processing Systems, pp. 1037–1045 (2016)
17. Pei, K., Cao, Y., Yang, J., Jana, S.: Deepxplore: automated whitebox testing of deep learning systems. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 1–18. ACM (2017)
18. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. arXiv preprint arXiv:1612.08242 (2016)
19. Scellier, B., Bengio, Y.: Equilibrium propagation: bridging the gap between energy-based models and backpropagation. Front. Comput. Neurosci. **11**, 24 (2017)
20. Shwartz-Ziv, R., Tishby, N.: Opening the black box of deep neural networks via information. arXiv preprint arXiv:1703.00810 (2017)
21. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: the all convolutional net. arXiv preprint arXiv:1412.6806 (2014)