Neurocomputing 000 (2017) 1–7

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Improving deep neural networks with multi-layer maxout networks and a novel initialization method

Weichen Sun[a,*], Fei Su[a,b], Leiquan Wang[c]

[a] School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China, 100876
[b] Beijing Key Laboratory of Network System and Network Culture, Beijing University of Posts and Telecommunications, Beijing, China, 100876
[c] School of Computer and Communication Engineering, China University of Petroleum (Huadong), Qingdao, China, 266580

ARTICLE INFO

Article history:
Received 14 October 2016
Revised 26 March 2017
Accepted 13 May 2017
Available online xxx

Keywords:
Deep learning
Convolutional neural networks
Activation function
Image classification
Initialization

ABSTRACT

For the purpose of enhancing the discriminability of convolutional neural networks (CNNs) and facilitating the optimization, we investigate the activation function for a neural network and the corresponding initialization method in this paper. Firstly, a trainable activation function with a multi-layer structure (named "Multi-layer Maxout Network", MMN) is proposed. MMN is a multi-layer structured maxout, inheriting advantages of both a non-saturated activation function and a trainable activation function approximator. Secondly, we derive a robust initialization method specifically for the MMN activation with a theoretical proof, which works for the maxout activation as well. Our novel initialization method could reduce internal covariate shift when signals are propagated through layers and solve the so called "exploding/vanishing gradient" problem, which leads a more efficient training procedure of deep neural networks. Experimental results show that our proposed model yields better performance on three image classification benchmark datasets (CIFAR-10, CIFAR-100 and ImageNet) than quite a few state-of-the-art methods and our novel initialization method improves performance further. Furthermore, the influence of MMN in different hidden layers is analyzed, and a trade-off scheme between the accuracy and computing resources is given.

1. Introduction

A resurgence of neural networks (NN), also called deep learning (DL), has drawn much attention since 2006, mainly due to the significant performance gain in visual recognition tasks, such as recognizing objects [1–3], faces [4,5] and hand-written digits [6], especially in the presence of a large amount of training data. DL is a branch of machine learning based on a set of algorithms that attempt to learn representations of data using deep architectures [7–11].

The convolutional neural network (CNN) [12] is one of the most popular DL models applied in computer vision and its representational power increases dramatically with the depth [2,13]. However, when training a deep CNN employing the stochastic gradient decrease (SGD) algorithm, the error gradient weakens as it moves from the back of the network to the front. Consequently, higher layers could be trained well while lower layers often get stuck during training, almost stay the same as what they were initialized. Furthermore, the performance of our deep CNN may be not superior than that of a shallow network. Such arduousness in the training procedure is mainly due to the vanishing or exploding gradients problems [14,15] with the depth increasing. Several techniques, such as pre-training [16], data augmentation [1–3], regularization techniques [6,17,18], novel non-linear activation functions [1,19–25] and sophisticated initialization methods [14,25], have been proposed to solve difficulties of training deep neural networks.

Among recent advances of DL above, various activation functions and sophisticated initialization methods are two most effective ones that solve the vanishing or exploding gradients problems. On the one hand, whether the error gradient could propagate back to the lower layer depends on the activation function. The widely used strategy is to replace the saturated activation function (e.g. sigmoid or tanh) with the rectified linear unit (ReLU) [19], which is a non-saturated activation function and the error gradient could be passed back into the lower network. ReLU is a piecewise linear function which projects negative inputs to zeros, leading to a desirable property that activations of neural nodes are sparse. However, ReLU assigns a zero slope to the negative part. Hence back-propagation will be blocked when the unit is not activated and a vanishing error back flow has almost no effect on weight updates of lower layers. Authors in [21] presented a novel activation

* Corresponding author.
E-mail addresses: weichern.sun@gmail.com, weichen.sun@hotmail.com (W. Sun).

http://dx.doi.org/10.1016/j.neucom.2017.05.103
0925-2312/

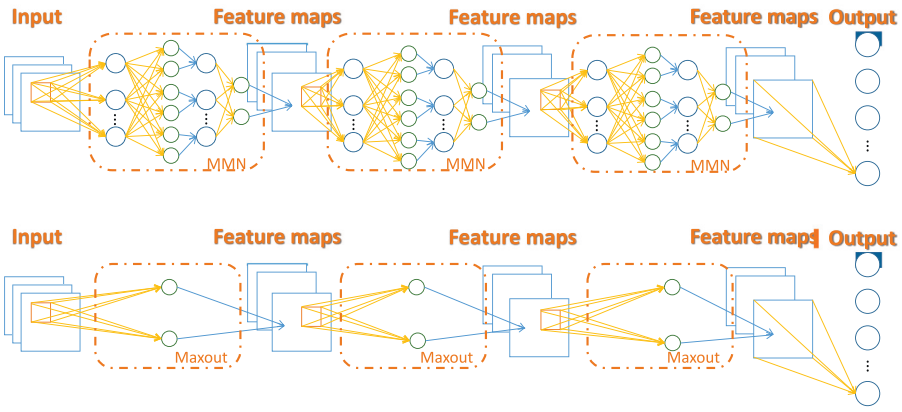

Fig. 1. Comparison of our model and a convolution neural network with maxout units.

function called maxout which assigns a non-zero slope to both the positive part and the negative part. Maxout facilitates the optimization procedure by partly preventing hidden units from transiting to inactive. Although the maxout unit is a trainable activation function, it is not a arbitrary function approximator. On the other hand, a suitable initialization method plays an crucial role on training a deep neural networks. Otherwise, a unfavorable initialization may lead to a long learning stage and a poor solution. As we all know, deep learning is non-convex optimization and the solution to a non-convex optimization algorithm depends on the initial values of parameters. Approaches such as the unsupervised pre-training [26], the transfer learning [27] and various initialization methods [14,25] are proposed for the sake of superior initial values for the optimization procedure. As pointed out in Glorot et al. [14], a properly scaled uniform distribution for initialization could solve the vanishing or exploding gradients problems, which is called "Xavier" initialization method in [28]. Nevertheless, the theoretical derivation is based on the assumption that the activations are linear, which is obviously not valid in a deep neural network. Afterward, He et al. [25] derived a robust initialization method particularly considering the rectifier nonlinearities. By rescaling the distribution of each layer's inputs at the initial state, lower layers in a extremely deep architecture could be efficiently trained.

Inspired by replacing a single conventional convolutional layer by a more complicated micro neural network, called "Network In Network (NIN)" [23], we proposed a joint supervised training framework that a micro deep network called MMN is trained as the nonlinear activation function together with parameters of each filter in convolutional layers. The primary advantage is that a micro deep network with increasingly complex structures could compactly represent a larger set of activation functions more efficiently than a shallow network. In other words, there are activation functions which could be compactly represented by a k-layer network, but could not be represented by a (k-1)-layer network unless it has an exponentially larger number of hidden units. In our proposed model, the activation function is replaced with the MMN, which is a more general nonlinear activation function approximator than the maxout unit. Here, weights of MMNs are shared as those of the maxout unit. By sliding convolutional filters with the MMN activation over the local patch, feature maps are obtained and then fed into higher layers. We stacked several convolutional layers with such MMNs and pooling layers to compose a deep CNN, which is trained by supervised back-propagation algorithm [29]. Since each hidden unit in MMN is a maxout unit, MMN can be treated as a multi-layer generalization of the maxout unit, which preserves the properties of the maxout unit while improving the capability in modelling various distributions of latent concepts. Furthermore, because existing initialization methods [14,25] do not make sense for MMN and maxout, we proposed a novel initialization method specialized for our MMN (valid for the maxout activation function as well) and provided a theoretical proof. In addition, the influence of MMN in different hidden layers is considered to show a trade-off scheme between the accuracy and the computational cost. The comparison of our model and a convolutional neural network with maxout units is shown in Fig. 1. To verify the performance of our proposed model and the novel initialization method, experiments are conducted on CIFAR-10, CIFAR-100 [30] and ImageNet [31] datasets. Experimental results show better performance of our model with dropout than those of current state-of-the-art methods.

2. Related work

2.1. Rectified linear unit

Rectified Linear Unit(ReLU) is first applied in Restricted Boltzmann Machines (RBM) [19]. It is defined as

$$y_i = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases} \tag{1}$$

where x_i is the input to a neuron and y_i is the output. ReLU has a desirable property that activations of neural nodes are sparse.

2.2. Maxout

The maxout is a more general version of ReLU, which takes the max operation on $k(k = 2)$ trainable linear functions. Given an input $x \in R^d$ (x is either the raw input vector or the state vector of a hidden layer), the output of a maxout unit is formulized as follows:

$$h_i(x) = \max_{j \in [1,k]} z_{ij} \tag{2}$$

Here, $z_{ij} = x^T W_{\cdots ij} + b_{ij}$, $W \in R^{d \times m \times k}$ and $b \in R^{m \times k}$ are trainable parameters. k is the number of linear sub-hidden units where taking the maximum Fig. 2 across. In a CNN, the activation of a maxout unit equals the maximum over the k feature maps. Though the maxout unit is similar to the commonly used spatial max-pooling in CNNs, it takes the maximum value over a subspace of k trainable linear transformations over the same input, whereas the spatial max-pooling is corresponding to k different input.

2.3. "Xavier" Initialization

During the forward propagation, there are two kinds of unfavorable conditions. On one hand, if the weights in a network start too

Please cite this article as: W. Sun et al., Improving deep neural networks with multi-layer maxout networks and a novel initialization method, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.05.103

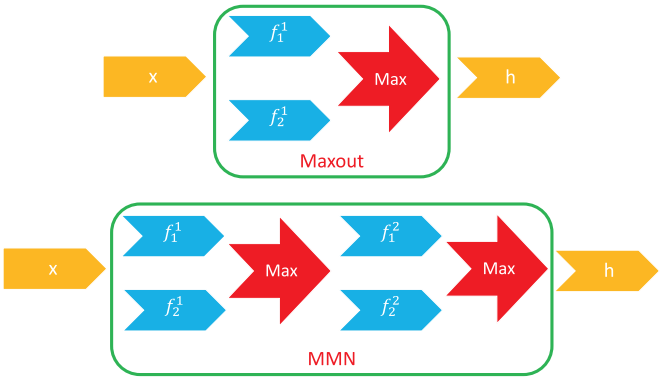

Fig. 2. Comparison of maxout and MMN.

small, then the signal shrinks as it passes through each layer until it is too tiny to be useful. On the other hand, if the weights in a network start too large, then the signal grows as it passes through each layer until it is too massive to be useful. In either condition, it slows down the training process. Jia et al. [28] proposed a novel initialization method called "Xavier" to keep the signal in a reasonable range of values through layers. For each weigh filler, the Xavier algorithm automatically determines the scale of initialization based on the number of input and output neurons.

3. Multi-layer maxout networks (MMN)

MMN is a kind of trainable activation function with a multi-layer structure. Given an input $x \in R^d$ (x is either the raw input vector or the state vector of a hidden layer), the activation of a hidden unit is calculated as follows:

$$f_{i,j}^1 = \max_{j\in[1,k_1]} x^T W_{\cdots ij} + b_{ij} \tag{3}$$

$$f_{i,j}^2 = \max_{j\in[1,k_2]} {f_{i,j}^1}^T W_{\cdots ij} + b_{ij} \tag{4}$$

$$\vdots$$

$$f_{i,j}^m = \max_{j\in[1,k_m]} {f_{i,j}^{m-1}}^T W_{\cdots ij} + b_{ij} \tag{5}$$

$$\vdots$$

$$f_{i,j}^n = \max_{j\in[1,k_n]} {f_{i,j}^{n-1}}^T W_{\cdots ij} + b_{ij} \tag{6}$$

$$h_i = \max_{j\in[1,k_n]} f_{i,j}^n \tag{7}$$

Here k_m is the number of units in the mth layer and n is the total number of layers in MMN. The MMN activation is powerful enough to approximate an arbitrary continuous activation function, even a non-convex activation function such as a w-shaped curve depicted in Fig. 3. The proof is given by Theorem 4.3 in [21]. Traditional nonlinear activation functions, such as the rectified linear and absolute value rectifier, can be approximated well by a MMN when k is not less than two. Even if the potential feature extraction requires a more complex nonlinear activation function, a MMN could approximate arbitrary activation functions by increasing the parameter k [21]. Meanwhile, a side effect is that the computation complexity increases dramatically.

4. Initialization of filter weights for MMN

Glorot et al. [14] proposed an initialization method called "Xavier" based on the assumption that the activation function is linear. Obviously, it is invalid for non-linear activation functions such as ReLU, maxout and MMN. Authors in [25] derived a theoretically sound initialization method by taking ReLU into account, which accelerates the training of deep neural nets. In the following, we derive a theoretically effective initialization method specialized for both maxout and the MMN activation function.

4.1. Forward propagation case

Inspired by [14] and [25], we also focused on the variance of the responses in each layer. We denote l to index the layer. For the lth convolutional layer in a maxout network, the response is $z_l = x_l^T W_l + b_l$, where $x_l \in R^d$ (x_l is either the raw input vector or the state vector of a hidden layer). Here, $d = p^2 c$ denotes the number of connections of a response, with that p is the kernel size and c is the number of the input channel. Then the output of each maxout unit is formulized as Eq. (2).

Our initialization method is based on assumptions below:

- All x and W are mutually independent of each other and share the same distribution
- The initialized distribution of W is symmetric around zero
- The bias of each layer b equals zero

Then we have

$$Var[z_l] = d_l Var[W_l x_l] \tag{8}$$

Since W_l is drawn from a Gaussian distributions with zero mean, and W_l and x_l are independent, then

$$Var[z_l] = d_l Var[W_l] E[x_l^2] \tag{9}$$

where $E[x_l^2]$ is the expectation of x_l^2. In the case of maxout activation function, we take max operations over two linear functions for simplicity here, i.e. $x_l = h_{l-1}(x) = \max(z_{l-1,1}, z_{l-1,2})$, and x_l is not a symmetric distribution around zero. Because b_{l-1} equals zero and W_l has the mean of zero, then z_{l-1} has zero mean and its distribution is symmetric around zero. In order to discover the relationship between $E[x_l^2]$ and $Var[z_{l-1}]$, we define

$$x_l = \frac{z_{l-1,1} + z_{l-1,2} + |z_{l-1,1} - z_{l-1,2}|}{2} \tag{10}$$

Then we obtain

$$\begin{aligned} E[x_l^2] &= \frac{1}{4} E[(z_{l-1,1} + z_{l-1,2} + |z_{l-1,1} - z_{l-1,2}|)^2] \\ &= \frac{1}{2} E[z_{l-1,1}^2 + z_{l-1,2}^2 + (z_{l-1,1} + z_{l-1,2})|z_{l-1,1} - z_{l-1,2}|] \\ &= \frac{1}{2}(E[z_{l-1,1}^2] + E[z_{l-1,2}^2] \\ &\quad + (E[z_{l-1,1}] + E[z_{l-1,2}])E[|z_{l-1,1} - z_{l-1,2}|]) \\ &= \frac{1}{2}(Var[z_{l-1,1}] + Var[z_{l-1,2}]) \end{aligned}$$

Because $z_{l-1,1}$ and $z_{l-1,2}$ share the same distribution, we could define z_{l-1} as $Var[z_{l-1}] = Var[z_{l-1,1}] = Var[z_{l-1,2}]$. Then we have

$$E[x_l^2] = Var[z_{l-1}] \tag{11}$$

Putting Eq. (11) into Eq. (9), we could obtain

$$Var[z_l] = d_l Var[W_l] Var[z_{l-1}] \tag{12}$$

When there are L hidden layers, the relationship between the first hidden layer and the last one is as follows:

$$Var[z_L] = \left(\prod_{l=2}^{L} d_l Var[W_l]\right) Var[z_1] \tag{13}$$

For the purpose of reducing the internal covariate shift, a proper initialization method should meet the following sufficient condition:

$$d_l Var[W_l] = 1, \forall l \tag{14}$$

Please cite this article as: W. Sun et al., Improving deep neural networks with multi-layer maxout networks and a novel initialization method, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.05.103

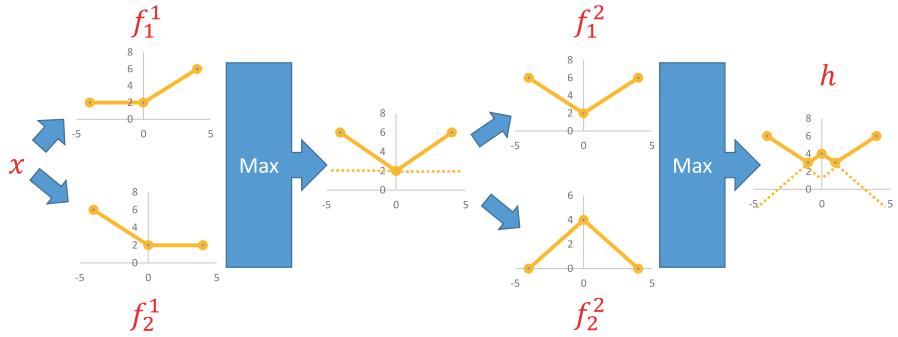

Fig. 3. A MMN could approximate arbitrary activation functions.

In the case $l = 1$, it still satisfies Eq. (14), since there is no activation function directly applied on the input. In summary, our initialization method requires that each W_l should obey a zero-mean Gaussian distribution with a standard deviation of $\sqrt{\frac{1}{\hat{d}_l}}$.

4.2. Back propagation case

In the back-propagation case, what we focused is the gradient of each convolutional layer

$$\Delta x_l = \hat{W}_l \Delta h_l \tag{15}$$

Here Δx_l and Δh_l denote gradients $\frac{\partial Loss}{\partial x}$ and $\frac{\partial Loss}{\partial h}$ separately. Δx_l is a c-by-1 vector and Δh_l is a $\hat{d}$-by-1 vector where $\hat{d} = k^2 e$. Here e represents the number of filters. Notice that W and $\hat{W}_l$ could be reshaped from each other by reason that $\hat{W}_l$ is a c-by-$\hat{d}$ matrix. As what we mentioned above, we assume

- Δh_l and W (or $\hat{W}_l$) are independent from each other
- The initialized distribution of W (or $\hat{W}_l$) is symmetric around zero
- The expectation of Δx_l equals zero for all l

Taking the maxout activation function ($h_l = max(z_{l,1}, z_{l,2})$) into account, we obtain

$$\Delta z_{l,k} = f'(z_{l,k}) \Delta x_{l+1}, k \in \{1, 2\} \tag{16}$$

Here, $f'(z_{l,k}) = 1$ or $f'(z_{l,k}) = 0$ with the same probability. Since $f'(z_{l,k})$ and Δx_{l+1} are independent, we have $E[\Delta h_l] = E[\Delta z_{l,k}] = \frac{E[\Delta x_{l+1}]}{2} = 0$ for any $k \in \{1, 2\}$ and $E[(\Delta h_l)^2] = Var[\Delta h_l] = \frac{1}{2} Var[\Delta x_{l+1}]$. Thus the variance of the gradient of Eq. (15) is

$$Var[\Delta x_l] = \frac{1}{2} \hat{d}_l Var[W_l] Var[\Delta x_{l+1}] \tag{17}$$

With L layers, we have

$$Var[\Delta x_2] = Var[\Delta x_{L+1}] \left(\prod_{l=2}^{L} \frac{1}{2} \hat{d}_l Var[W_l] \right) \tag{18}$$

The sufficient condition that makes the gradient back propagated smoothly is

$$\frac{1}{2} \hat{d}_l Var[W_l] = 1, \forall l \tag{19}$$

This leads to apply a zero-mean Gaussian distribution with a standard deviation of $\sqrt{2/\hat{d}_l}$ for initialization. When initializing the first layer, we still adopt Eq. (19) for the same reason mentioned in the forward propagation case.

Note that both sufficient conditions could not satisfied simultaneously. As a compromise between them, we propose a new sufficient condition

$$Var[W_l] = \frac{4}{2d_l + \hat{d}_l}, \forall l \tag{20}$$

Since the MMN activation is a multi-layer version of the maxout activation, hence the initialization method based on the maxout is valid for MMN as well.

5. Experiments

5.1. Overview

As we described before, our model is a stack of three alternating convolutional layers and pooling layers, and an objective logistic cost layer followed by a global average pooling layer over 10 feature maps on the top. There are 192 filters in each convolutional layer, which is the same size as the NIN. We also apply the dropout(50%), which remained essential even after removing the fully connected layers. The input size of the receptive field in our network is 32×32 taking RGB channels. All convolutional layers use MMN as the nonlinear activation function. In each MMN, there are two hidden layers, and each unit is a maxout unit. Weights of all convolutional layers are initialized by our proposed initialization method satisfying Eq. (20). Our models are trained by the Stochastic Gradient Descent (SGD) method with mini-batching. In all the followed experiments, we operate out model using the cuda-convnet code developed by Alex Krizhevsky [1]. To verify the performance of the proposed model, we conduct experiments on three benchmark datasets (CIFAR-10, CIFAR-100 [30] and ImageNet [31], and give the comparing results with those methods reported in literatures [6,21–23,25,32–35]. In addition, we provide the comparison between our initialization method and "Xavier" initialization method based on the same MMN model. Furthermore, the influences of using MMNs as activation functions on different feature extraction layers are analyzed.

5.2. Experiments on CIFAR-10

CIFAR-10 is an established computer-vision dataset for the sake of object recognition. It is a subset of 80 million tiny images dataset and composed of 10 object classes of natural images with 5000 training images and 1000 testing images per class [30]. Each image is a colour image of size 32×32. In order to adjust model parameters, such as the learning rate and the number of iterations, we random select 1000 samples per class as the validation set from those training images. Before training, the same preprocessing steps including global contrast normalization and ZCA whitening are performed like in the maxout network [21]. All the compared models use the same number of feature maps such as MMN, NIN [23] and maxout networks [21]. Test error is defined as the proportion between the number of incorrectly classified instances and the total number of classified instances in the test set. Details

Please cite this article as: W. Sun et al., Improving deep neural networks with multi-layer maxout networks and a novel initialization method, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.05.103

Table 1
Test set error rates for CIFAR-10 of different methods.

Method	Test error (%)
Stochastic pooling [32]	15.13
CNN + Spearmint [33]	14.98
Conv. maxout [21]	11.68
Conv. probout [22]	11.35
NIN [23]	10.41
PReLU [25]	9.41
MMN	9.17

Table 2
Test set error rates for CIFAR-10 of different methods with data augmentation.

Method	Test error (%)
CNN + Spearmint + [33]	9.50
Conv. Maxout [21]	9.38
DropConnect + 12 networks [6]	9.32
Conv. probout [22]	9.39
NIN [23]	8.81
PReLU [25]	7.68
MMN	7.66

Table 3
Test set error rates for CIFAR-100 of different methods.

Method	Test error (%)
Learned pooling [34]	43.71
Stochastic pooling [32]	42.51
Conv. maxout [21]	38.57
Conv.probout [22]	38.14
Tree based priors [35]	36.85
NIN [23]	35.68
PReLU [25]	35.32
MMN + Dropout	35.29

Table 4
Test set error rates for ImageNet of different methods.

Method	Top-1 test error (%)	Top-5 test error (%)
AlexNet [1]	40.7	18.2
NIN [23]	40.9	18.5
MMN	39.7	17.6

Table 5
Test set error rates for CIFAR-10 ofdifferent initialization methods.

Method	Test error (%)
MMN (Xavier)	9.63
MMN (ours)	9.17
MMN + Data augmentation (Xavier)	7.81
MMN + Data augmentation (ours)	7.66

Table 6
Test set error rates for CIFAR-100 of different initialization methods.

Method	Test error (%)
MMN (Xavier)	35.40
MMN (ours)	35.29
MMN + Data augmentation (Xavier)	33.49
MMN + Data augmentation (ours)	33.24

Table 7
Test set error rates for ImageNet of different initialization methods.

Method	Top-1 Test error (%)	Top-5 Test error (%)
MMN (Xavier)	39.7	17.6
MMN (ours)	39.4	17.4

of the performance comparison are illustrated in Table 1, and results show that our model gives the best performance in accuracy.

Furthermore, we evaluated the performance of our model in case of data augmentation. We augmented the training data by randomly cropping a 24×24 patches of the 32×32 original training images. This will cause the model to train on random 24×24 patches and test on the center 24×24 patch. The comparison of the state-of-the-art methods with data augmentation are given in Table 2. Data augmentation helps to reduce overfitting and increase the performance of the algorithm through generating additional examples. Meanwhile, it significantly increases time required to converge. A trade-off scheme between the performance and the computational cost should be considered. The experimental results demonstrate MMN is a trainable activation function which can give rise to more interesting invariances in deep convolution neural networks.

5.3. Experiments on CIFAR-100

The CIFAR-100 dataset is similar to the CIFAR-10 dataset, except it has 100 classes containing 600 images each [30]. There are 500 images for training, 100 images for testing per class. For CIFAR-100, we applied the same model and hyper-parameters as in CIFAR-10, except the last MMN layer outputs 100 feature maps instead of 10 feature maps corresponding to 100 different classes. A summary in Table 3 shows the performance comparison of different models on the CIFAR-100 database.

5.4. Experiments on ImageNet

ImageNet is a dataset of over 15 million labeled images belonging to 22,000 categories. In our experiments, we used a subset of ImageNet which has about 1000 images in each 1000 categories. There are roughly 1.2 million training images, 50,000 validation images and 150,000 testing images. Each image is resized to a fixed resolution of 256×256 and subtracting the mean activity over the training set from each pixel. Considering ImageNet is a more complicated classification task, we employed a deeper neural networks which has 4 convolutional layers. A summary in Table 4 shows the performance comparison of different models on the CIFAR-100 database.

5.5. Comparisons with "Xavier" Initializations

In order to validate our initialization method specialized for networks with MMN activations, we compare our initialization method meeting Eq. (20) with the "Xavier" initialization [14] on the same model mentioned in the former section over CIFAR-10, CIFAR-100 and ImageNet datasets. Here, we take into account the data augmentation and better performance is obtained with more expensive computational cost. A comparison of our model with those state-of-the-art methods using convolutional structures are shown inTables 5–7. Our initialization method has been theoretically proved in this paper to be specialized for neural networks with MMN activations and the experimental results still verify our conclusion.

5.6. Influence of using MMN in different layers

As we mentioned before, this new type of activation function has greater capability than the maxout. However, it is computationally expensive for training. In case of the limitation of computing resources, replacing a portion of activation functions with MMNs is an effective way to improve performance of deep models. In order to identify the influence of MMN in different layers of

Please cite this article as: W. Sun et al., Improving deep neural networks with multi-layer maxout networks and a novel initialization method, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.05.103

Table 8
Test set error rates and computational costs for CIFAR-10 of different methods.

Method	Test error (%)	Time (h)
3-Conv.NIN [23]	10.41	5.93
Conv.MMN + 2-Conv.NIN	9.99	7.15
Conv.NIN + Conv.MMN + Conv.NIN	10.29	6.38
2-Conv.NIN + Conv.MMN	10.34	5.97

Table 9
Test set error rates and computational costs for CIFAR-100 of different methods.

Method	Test error (%)	Time(h)
3-Conv.NIN [23]	35.68	5.83
Conv.MMN + 2-Conv.NIN	34.74	7.08
Conv.NIN + Conv.MMN + Conv.NIN	35.13	6.04
2-Conv.NIN + Conv.MMN	35.25	5.90

a deep model, we designed extensive experiments on CIFAR-10 and CIFAR-100 datasets by replacing only one NIN of the current state-of-the-art method proposed in [23], with a MMN of the same size. All experiments are run on a NVidia GeForce GTX Titan Black(6GB). The experimental results show that modeling with activation functions in lower layer replacing by MMNs, has lower error rate and simultaneously a more expensive computational cost we have to afford. Details of experimental results are given in Tables 8 and 9.

Our MMN activation could be applied in any layer of deep neural networks flexibly and improve performance of models. The only factor we should take into account is a trade-off between the performance and the computational complexity.

6. Conclusions

A novel nonlinear activation function named "Multi-layer Maxout Network (MMN)" for feature extractions and correspondingly a theoretically sound initialization method are proposed in this paper. MMN is a trainable universal approximator and has advantages of both the maxout unit and the deep neural network. Furthermore, our theoretically sound initialization method, which is suitable for activation functions such as MMN and maxout, helps with training deep models by reducing problems of vanishing or exploding gradients and yields competitive performance on three datasets (CIFAR-10, CIFAR-100 and ImageNet). In addition, since deep neural networks with MMN are more time-consuming than those with conventional activation functions, replacing a portion of activation functions with MMNs is proposed as a trade-off scheme between the accuracy and computing resources. We demonstrate the influence of MMN in different hidden layers and recommend that it is more effective to replace activation functions of lower layers with MMNs than those of the others in case of computing resource limitation.

Acknowledgment

This work is supported by Chinese National Natural Science Foundation of China (61372169, 61532018, 61471049).

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems (NIPS), Neural Information Processing Systems (NIPS) Foundation, 2012, pp. 1097–1105.
[2] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, ICLR, 2015.
[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: CVPR, 2015.
[4] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al., Deepid-net: Deformable deep convolutional neural networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 2403–2412.
[5] Y. Sun, D. Liang, X. Wang, X. Tang. Deepid3: Face recognition with very deep neural networks. arXiv preprint arXiv:1502.00873, 2015.
[6] L. Wan, M. Zeiler, S. Zhang, Y.L. Cun, R. Fergus, Regularization of neural networks using dropconnect, in: Proceedings of the 30th International Conference on Machine Learning (ICML), 2013, pp. 1058–1066.
[7] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.
[8] J. Taghia, Z. Ma, A. Leijon, Bayesian estimation of the von-Mises fisher mixture model with variational inference, IEEE Trans. Pattern Anal. Mach. Intell. 36 (9) (2014) 1701–1715.
[9] Z. Ma, A.E. Teschendorff, A. Leijon, Y. Qiao, H. Zhang, J. Guo, Variational Bayesian matrix factorization for bounded support data, IEEE Trans. Pattern Anal. Mach. Intell. 37 (4) (2015) 876–889.
[10] Z. Ma, Z.-H. Tan, J. Guo, Feature selection for neutral vector in EEG signal classification, Neurocomputing 174 (2016) 937–945.
[11] Z. Ma, J.-H. Xue, A. Leijon, Z.-H. Tan, Z. Yang, J. Guo, Decorrelation of neutral vector variables: theory and applications, IEEE Trans. Neural Netw. Learn. Syst. PP (99) (2016).
[12] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324, doi:10.1109/5.726791.
[13] R.K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, J. Schmidhuber, Compete to compute, in: Advances in Neural Information Processing Systems (NIPS), Neural Information Processing Systems (NIPS) Foundation, 2013, pp. 2310–2318.
[14] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), 2010, pp. 249–256.
[15] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: Proceedings of the 30th International Conference on Machine Learning, ICM-L 2013, Atlanta, GA, USA, 16-21 June 2013, 2013, pp. 1310–1318.
[16] G.E. Dahl, D. Yu, L. Deng, A. Acero, Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition, IEEE Trans. Audio Speech Lang. Process. 20 (1) (2012) 30–42.
[17] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv preprint arXiv:1207.0580 (2012).
[18] L. Garrido, S. Gómez, V. Gaitán, M. Serra-Ricart, A regularization term to avoid the saturation of the sigmoids in multilayer neural networks, Int. J. Neural Syst. 7 (03) (1996) 257–262.
[19] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML), 2010, pp. 807–814.
[20] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS), 2011, pp. 315–323.
[21] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio. Maxout networks. arXiv:1302.4389, 2013.
[22] J.T. Springenberg, M. Riedmiller, Improving deep neural networks with probabilistic maxout units, in: International Conference on Learning Representations (ICLR), 2014.
[23] M. Lin, Q. Chen, S. Yan, Network in network, in: In- ternational Conference on Learning Representations (ICLR), 2014.
[24] W. Sun, F. Su, L. Wang, Improving deep neural networks with multilayer maxout networks, in: Proceedings of Visual Communications and Image Processing Conference (VCIP), IEEE, 2014, pp. 334–337.
[25] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: ICCV, 2015.
[26] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[27] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM International Conference on Multimedia, ACM, 2014, pp. 675–678.
[29] Y. Chauvin, D.E. Rumelhart, Backpropagation: Theory, Architectures, and Applications, Psychology Press, 1995.
[30] A. Krizhevsky, G. Hinton, Learning Multiple Layers of Features from Tiny Images, Computer Science Department, University of Toronto, 2009.
[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
[32] M.D. Zeiler, R. Fergus, Stochastic pooling for regularization of deep convolutional neural networks, in: ICLR, 2013.
[33] J. Snoek, H. Larochelle, R.P. Adams, Practical bayesian optimization of machine learning algorithms, in: Advances in Neural Information Processing Systems (NIPS), Neural Information Processing Systems (NIPS) Foundation, 2012, pp. 2951–2959.
[34] Mateusz Malinowski, Mario Fritz, Learnable pooling regions for image classification, in: International Conference on Learning Representations (ICLR): Workshop track, 2013.

Please cite this article as: W. Sun et al., Improving deep neural networks with multi-layer maxout networks and a novel initialization method, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.05.103

[35] N. Srivastava, R. Salakhutdinov, Discriminative transfer learning with tree-based priors, in: Advances in Neural Information Processing Systems (NIPS), Neural Information Processing Systems (NIPS) Foundation, 2013, pp. 2094–2102.

Weichen Sun is a Ph.D. candidate in School of Information and Telecommunication Engineering, Beijing University of Posts and Telecommunications. His current research interests include deep learning, image classification and object detection.

Fei Su is a female professor in the multimedia communication and pattern recognition lab, School of Information and Telecommunication Engineering, Beijing University of Posts and Telecommunications. She received the Ph.D. degree majoring in Communication and Electrical Systems from BUPT in 2000. She was a visiting scholar at electrical computer engineering department, Carnegie Mellon University from 2008 to 2009, Her current interests include pattern recognition, image and video processing and biometrics. She has authored and co-authored more than 70 journal and conference papers and some textbooks.

Leiquan Wang is an experimenter in college of computer and communication engineering, China University of Petroleum. His current research interests include multimodal fusion, cross modal retrieval and social media analysis.

Please cite this article as: W. Sun et al., Improving deep neural networks with multi-layer maxout networks and a novel initialization method, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.05.103