Neurocomputing 321 (2018) 36–48

Contents lists available at ScienceDirect

Neurocomputing

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

AlphaMEX: A smarter global pooling method for convolutional neural networks

Boxue Zhang, Qi Zhao*, Wenquan Feng, Shuchang Lyu

School of Electronic and Information Engineering, Beihang University, Beijing, China

ARTICLE INFO

Article history:
Received 9 January 2018
Revised 11 June 2018
Accepted 24 July 2018
Available online 13 September 2018

Communicated by Dr Ruili Wang

Keywords:
CNN
Global Pooling
Feature-map sparsity
AlphaMEX
Network compression

ABSTRACT

Deep convolutional neural networks have achieved great success on image classification. A series of feature extractors learned from CNN have been used in many computer vision tasks. Global pooling layer plays a very important role in deep convolutional neural networks. It is found that the input feature-maps of global pooling become sparse, as the increasing use of Batch Normalization and ReLU layer combination, which makes the original global pooling low efficiency. In this paper, we proposed a novel end-to-end trainable global pooling operator AlphaMEX Global Pool for convolutional neural network. A nonlinear smooth log-mean-exp function is designed, called AlphaMEX, to extract features effectively and make networks smarter. Compared to the original global pooling layer, our proposed method can improve classification accuracy without increasing any layers or too much redundant parameters. Experimental results on CIFAR-10/CIFAR100, SVHN and ImageNet demonstrate the effectiveness of the proposed method. The AlphaMEX-ResNet outperforms original ResNet-110 by 8.3% on CIFAR10+, and the top-1 error rate of AlphaMEX-DenseNet ($k=12$) reaches 5.03% which outperforms original DenseNet ($k=12$) by 4.0%.

1. Introduction

With the fast development of high-performance hardware and big data technology, deep convolutional neural networks (CNN) have achieved great success in many visual tasks, such as object detection [1–6], image classification [7–13], and image segmentation [14–18]. It has been successfully implemented in various fields like digital microbiology imaging [19], sound event classification [20], early fire detection [21,22], deep representation of spatiotemporal event streams [23] and object recognition for smartphones [24]. Plain CNN mainly consists of four basic components: convolutional layers, non-linear activation layers, pooling layers and fully connected layers [25], while the ResNet [26] and DenseNet [27] structure has ResBlocks and DenseBlocks which is also constructed by these components.

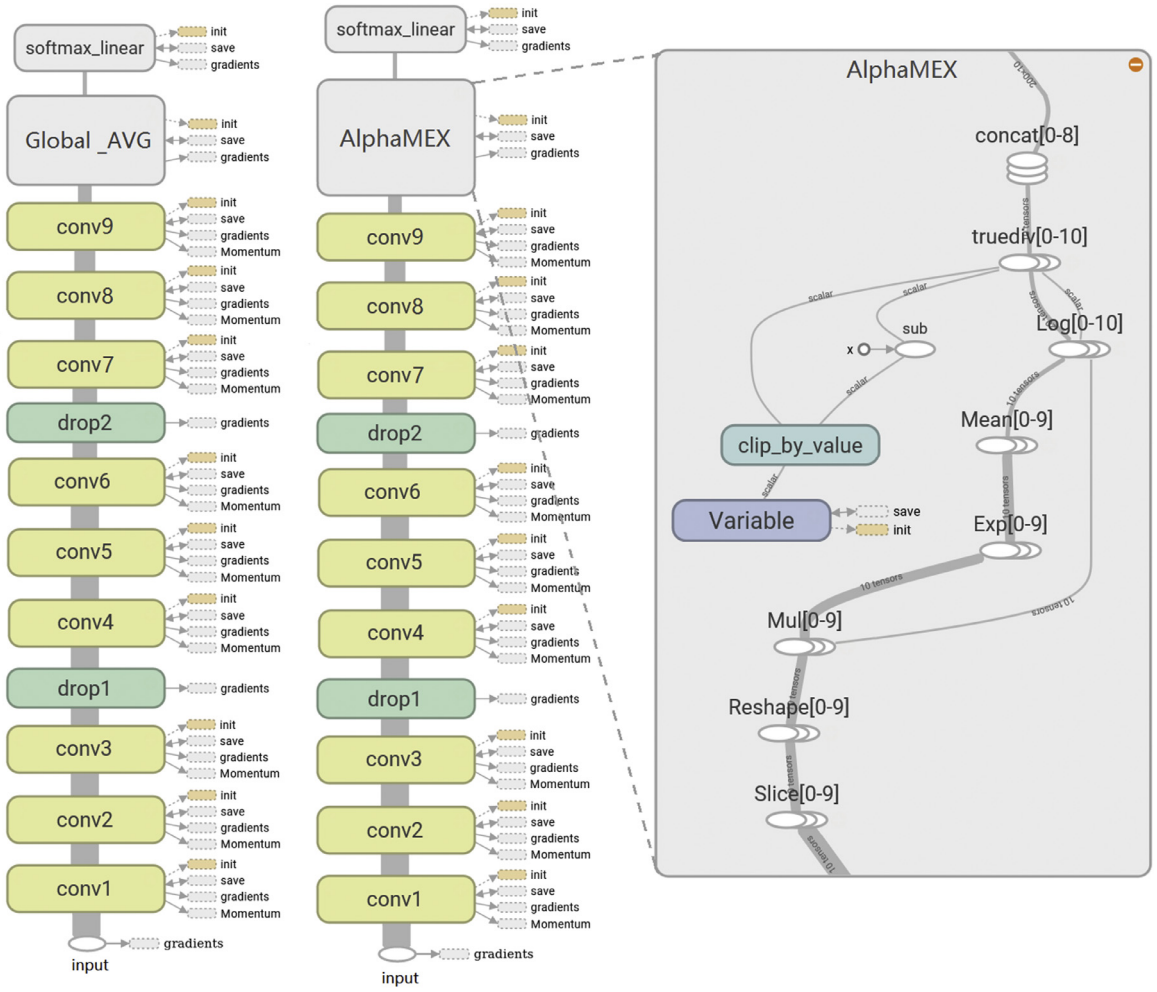
As CNNs become increasingly deep [28], a new research problem emerges: heavy structure and redundant parameters, which makes the network hard to train and transfer. Many publications address this or related problems. SqueezeNet [29] and Binary-Weight-Network [30] compress CNN structure and maintains accuracy. ShuffleNet [31] comes up with a novel channel shuffle operation to help the information flowing across feature channels. Highway Networks [32] and ResNet [26] bypass signal from one layer to the next via identity connections, which is also called "non-plain" CNNs [26].

However, the contributions of state-of-the-art "non-plain" CNNs are not only residual and dense structures, but also the global pooling layer inserted between feature-maps and fully connected layer. As most parameters are stacked in the gap between feature-maps and the fully connected layer, global pooling is one of the best methods to solve the problem of heavy structure and redundant parameters. The Network in Network [33] structure replaces the fully connected layers on top of feature-maps with a global average pooling layer, it takes the average of each feature-map, and the resulting vector is fed directly into the softmax layer. One advantage of global average pooling over the fully connected layers is that it is more native to the convolution structure by enforcing correspondences between feature maps and categories. Thus the feature maps can be easily interpreted as categories confidence maps.

As the combination of Batch Normalization (BN) [34] and ReLU [35] could achieve a great feature extracting performance, more and more state-of-the-art "non-plain" CNN structures [26–28,33] tend to use BN with ReLU as a common feature normalize and nonlinear transform operator. However, the combined operator will extract the most active features from previous feature-maps, which improves the efficiency of convolution layers, but makes

* Corresponding author.
E-mail address: zhaoqi@buaa.edu.cn (Q. Zhao).

https://doi.org/10.1016/j.neucom.2018.07.079



(a)All-CNN (b)AlphaMEX-All-CNN

Fig. 1. All-CNN and AlphaMEX-All-CNN architectures for CIFAR. (a) the original All-CNN for CIFAR with nine convolutional layers and a Global Average Pool layer; (b) the structure of AlphaMEX-All-CNN, AlphaMEX operator has been zoomed in to show details of the function.

feature maps sparse and is inappropriate for Global Average Pool. As the Global Average Pool layer will dilute the active features if the feature-maps are sparsity, which will influence the classification accuracy of the last fully connected layer.

In this paper, we proposed a smarter global pooling architecture called AlphaMEX Global Pool layer, which can learn better from sparse feature maps compared to traditional global pooling methods. Other pooling methods like Max Pooling [36] and Stochastic Pooling [37], are seldom used in global pooling layer, as both of them have low resistance to noise and low utilization of information when deal with large scale input. Unlike these pooling methods, the AlphaMEX Global Pool layer is an End-to-End trainable operator and it can be trained by most first order optimizer like SGD [38] and Ada [39] without adding too much redundant parameters. It has a better trade-off between the Average and Maximum Global Pool with a novel nonlinear function AlphaMEX. Fig. 1 shows the structure of AlphaMEX-All-CNN. The contributions of this paper can be summarized as follows:

- A novel mathematical nonlinear function AlphaMEX is proposed which could output any intermediate value between the average and extremum of inputs controlled by a single parameter α.
- Based on the AlphaMEX function, an end-to-end trainable global pooling layer is proposed, named AlphaMEX Global Pool. This novel global pooling layer has three advantages: It is smarter than any traditional global pooling method that it could be well trained end-to-end; It could improve the performance of state-of-the-art structures without adding any layers or too much redundant parameters; It is more appropriate in modern CNN structures, as the feature-maps become sparse after the Batch Normalization and ReLU combined layer which is commonly used in state-of-the-art "non-plain" CNNs.
- Experimental results on CIFAR10/CIFAR100 [40], SVHN [41] and ImageNet [42] demonstrate the effectiveness of our proposed method. The AlphaMEX-ResNet outperforms the original ResNet-110 by 8.3% on CIFAR10+, and the top-1 error rate of AlphaMEX-DenseNet reaches 5.03% which outperforms original DenseNet($k = 12$) by 4.0%.

The rest of the paper reviews related works in Section 2, describes in detail our AlphaMEX structure in Section 3. After that, experimental results and discussion are shown in Section 4. Finally, we conclude this paper and give the future work in Section 5.

2. Related work

Conventional neural networks perform convolution in the lower layers of the network. For classification, the feature maps of the last convolutional layer are vectorized and fed into fully connected

layers followed by a softmax logistic regression layer [43–45]. This structure bridges the convolutional structure with traditional neural network classifiers. It treats the convolutional layers as feature extractors, and the resulting feature is classified in a traditional way.

Pooling layers between convolutional layers aim to strengthen the translational invariance and reduce the dimension of the feature maps. Max pooling [36] is one of the typical pooling methods, which computes the maximum of a local patch. Another commonly used pooling method is average pooling [46], which outputs the mean value of a local patch. Zeiler and Fergus [37] proposed stochastic pooling. It randomly selects the activation unit from a local patch according to a multinomial distribution.

The fully connected layer at the end of CNNs is treated as a classifier. However, it is prone to overfitting, thus hampering the generalization ability of the overall network. Dropout is proposed by Hinton et al. [47] as a regularizer which randomly sets half of the activations to the fully connected layers to zero during training. It has improved the generalization ability and largely prevents overfitting [48].

Global average pooling is firstly proposed by NIN [33] to replace the traditional fully connected layers. The idea is to generate one feature map for each corresponding category of the classification task in the last convolutional layer. Global average pooling is widely used in modern CNN structures for its advantages. Highway Networks [32] is amongst the first architectures that provided a means to effectively train end-to-end networks with more than 100 layers. Using bypassing paths along with gating units, Highway Networks with hundreds of layers can be optimized without difficulty. The bypassing paths are presumed to be the key factor that eases the training of these very deep networks. This point is further supported by ResNet [26], in which pure identity mappings are used as bypassing paths. ResNet have achieved impressive, record-breaking performance on many challenging image recognition, localization, and detection tasks, such as ImageNet [42] and COCO object detection [26]. Instead of drawing representational power from extremely deep or wide architectures, DenseNet [27] exploits the potential of the network through feature reuse, yielding condensed models that are easy to train and highly parameter efficient. Concatenating feature-maps learned by different layers increase variation in the input of subsequent layers and improve efficiency.

There are other notable network architecture innovations which have yielded competitive results in image classification. All-CNN [49], replaces max-pooling with a convolutional layer with increased stride and yields competitive or state-of-the-art performance on several image recognition datasets. Deep SimNets [50], contains a higher abstraction level compared to a traditional CNN and shows a significant gain in accuracy over CNN when computational resources at run-time are limited. The SimNets architecture is driven by two operators: (i) a similarity function that generalizes inner-product, and (ii) a log-mean-exp function called MEX that generalizes maximum and average. The two operators applied in succession give rise to a standard neuron but in "feature space", which makes the SimNets contain a higher abstraction level compared to a traditional CNN. In Deeply Supervised Network (DSN) [51], internal layers are directly supervised by auxiliary classifiers, which can strengthen the gradients received by earlier layers.

3. AlphaMEX

In this section, we firstly take a look at the MEX function, then introduce the proposed AlphaMEX function and AlphaMEX Global Pool layer.

3.1. MEX

The MEX operator [50], a log-mean-exp function, has a softmax-like structure:

$$\underset{i=1,\ldots,n}{MEX_{\beta}}\{c_i\} := \frac{1}{\beta}\log\left(\frac{1}{n}\sum_{i=1}^{n}\exp\{\beta\cdot c_i\}\right) \tag{1}$$

where c_i represents input elements and n represents the number of c_i. The parameter $\beta \in \mathbb{R}$ spans a continuum between maximum ($\beta \to +\infty$), average ($\beta \to 0$) and minimum ($\beta \to -\infty$). Furthermore, the MEX operator implements a soft trade-off between maximum and average which is depended on different values of β. The MEX function has the following "collapsing" property [50]:

$$MEX_{\beta}\left\{MEX_{\beta}\left\{c_{ij}\right\}_{1\le j\le m}\right\}_{1\le i\le n} = MEX_{\beta}\left\{c_{ij}\right\}_{1\le j\le m,1\le i\le n} \tag{2}$$

which is very helpful when dealing with the feature-map extracted from the convolutional layer in CNN.

Another practical property is that the MEX function is derivable. The partial derivative of β and input c_i are given as follows:

$$\begin{aligned}\frac{\partial(MEX_{\beta}\{c_i\})}{\partial(\beta)} &= \frac{\partial\left(\frac{1}{\beta}\cdot\log\left(\frac{1}{n}\sum_{i=1}^{n}e^{\beta\cdot c_i}\right)\right)}{\partial(\beta)}\\ &= -\beta^{-2}\cdot\log\left(\frac{1}{n}\sum_{i=1}^{n}e^{\beta\cdot c_i}\right)+\frac{\frac{1}{n}\left(\sum_{i=1}^{n}c_i e^{\beta\cdot c_i}\right)}{\frac{1}{n}\left(\sum_{i=1}^{n}e^{\beta\cdot c_i}\right)\cdot\beta}\\ &= \frac{1}{\beta}\cdot\left(\frac{\sum_{i=1}^{n}c_i e^{\beta\cdot c_i}}{\sum_{i=1}^{n}e^{\beta\cdot c_i}}-\frac{1}{\beta}\cdot\log\left(\frac{1}{n}\sum_{i=1}^{n}e^{\beta\cdot c_i}\right)\right)\end{aligned} \tag{3}$$

$$\begin{aligned}\frac{\partial(MEX_{\beta}\{c_i\})}{\partial(c_i)} &= \frac{\partial\left(\frac{1}{\beta}\cdot\log\left(\frac{1}{n}\sum_{i=1}^{n}e^{\beta\cdot c_i}\right)\right)}{\partial(c_i)}\\ &= \frac{1}{\beta}\cdot\frac{\frac{1}{n}\cdot\beta\cdot e^{\beta\cdot c_i}}{\frac{1}{n}\sum_{i=1}^{n}e^{\beta\cdot c_i}}\\ &= \frac{e^{\beta\cdot c_i}}{\sum_{i=1}^{n}e^{\beta\cdot c_i}}\end{aligned} \tag{4}$$

3.2. AlphaMEX

Although the MEX function has some nice properties, it is hard to set the initial value or train the parameter β which ranges from 0 to $+\infty$. Actually, it is impractical to set a number infinity as to the upper limit of numerical value. So the max output from the MEX function is only theoretical. The performance of the function to approximate the max output depends on the base data types and the computing power.

To solve this problem, we propose a novel log-mean-exp function, called AlphaMEX function:

$$\underset{i=1,\ldots,n}{AlphaMEX_{\alpha}}\{c_i\} := \frac{1}{\log\left(\frac{\alpha}{1-\alpha}\right)}\log\left(\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\alpha}{1-\alpha}\right)^{c_i}\right) \tag{5}$$

where α is limited from 0 to 1. In this continuous function, α is a trainable parameter which has more numerical stability and higher efficiency. AlphaMEX function has these followed excellent properties:

(1) when α right-sided limits to 0, function outputs the minimum value:

$$\lim_{\alpha\to 0^{+}}\underset{i=1,\ldots,n}{AlphaMEX_{\alpha}}\{c_i\} \equiv \min_{i=1,\ldots,n}\{c_i\} \tag{6}$$

(2) when α limits to $\frac{1}{2}$, function outputs the average value:

$$\lim_{\alpha\to\frac{1}{2}}\underset{i=1,\ldots,n}{AlphaMEX_{\alpha}}\{c_i\} \equiv \operatorname{mean}_{i=1,\ldots,n}\{c_i\} \tag{7}$$

(3) when α left-sided limits to 1, function outputs the maximum value:

$$\lim_{\alpha\to 1^-} AlphaMEX_\alpha\{c_i\} \equiv \max_{i=1,\ldots,n}\{c_i\} \tag{8}$$

(4) "collapsing" property:

$$AlphaMEX_\alpha\left\{AlphaMEX_\alpha\left\{c_{ij}\right\}_{1\le j\le m}\right\}_{1\le i\le n} = AlphaMEX_\alpha\left\{c_{ij}\right\}_{1\le j\le m,1\le i\le n} \tag{9}$$

As a trainable parameter, α can be optimized by most first order learning algorithms like SGD [38] or Adam [39]. In back-propagation, the gradient of α is calculated as:

$$\begin{aligned}\frac{\partial(AlphaMEX_\alpha\{c_i\})}{\partial(\alpha)} &= \frac{\partial\left(\frac{1}{\log\left(\frac{\alpha}{1-\alpha}\right)}\log\left(\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\alpha}{1-\alpha}\right)^{c_i}\right)\right)}{\partial(\alpha)}\\ &= -\frac{\log^{-2}\left(\frac{\alpha}{1-\alpha}\right)}{\alpha(1-\alpha)}\cdot\log\left(\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\alpha}{1-\alpha}\right)^{c_i}\right)\\ &\quad+\frac{\frac{1}{n}\left(\sum_{i=1}^{n}c_i\left(\frac{\alpha}{1-\alpha}\right)^{c_i}\right)}{\frac{1}{n}\left(\sum_{i=1}^{n}\left(\frac{\alpha}{1-\alpha}\right)^{c_i}\right)\cdot\log\left(\frac{\alpha}{1-\alpha}\right)\cdot\alpha(1-\alpha)}\\ &= \frac{1}{\log\left(\frac{\alpha}{1-\alpha}\right)\cdot\alpha(1-\alpha)}\cdot\left(\frac{\sum_{i=1}^{n}c_i\left(\frac{\alpha}{1-\alpha}\right)^{c_i}}{\sum_{i=1}^{n}\left(\frac{\alpha}{1-\alpha}\right)^{c_i}}\right.\\ &\quad\left.-\frac{1}{\log\left(\frac{\alpha}{1-\alpha}\right)}\cdot\log\left(\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\alpha}{1-\alpha}\right)^{c_i}\right)\right)\\ &= \frac{1}{\Gamma(\alpha)}\cdot\left(\frac{\Phi'_\alpha\{c_i\}}{\Phi_\alpha\{c_i\}}-AlphaMEX_\alpha\{c_i\}\right)\end{aligned} \tag{10}$$

Also when the chain rule is used to propagate the gradient from the output to the input of AlphaMEX, the gradient of input element c_i is given as follows:

$$\begin{aligned}\frac{\partial(AlphaMEX_\alpha\{c_i\})}{\partial(c_i)} &= \frac{\partial\left(\frac{1}{\log\left(\frac{\alpha}{1-\alpha}\right)}\log\left(\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\alpha}{1-\alpha}\right)^{c_i}\right)\right)}{\partial(c_i)}\\ &= \frac{1}{\log\left(\frac{\alpha}{1-\alpha}\right)}\cdot\frac{\frac{1}{n}\cdot\log\left(\frac{\alpha}{1-\alpha}\right)\cdot\left(\frac{\alpha}{1-\alpha}\right)^{c_i}}{\frac{1}{n}\cdot\sum_{i=1}^{n}\left(\frac{\alpha}{1-\alpha}\right)^{c_i}}\\ &= \frac{\left(\frac{\alpha}{1-\alpha}\right)^{c_i}}{\sum_{i=1}^{n}\left(\frac{\alpha}{1-\alpha}\right)^{c_i}}\\ &= \frac{\Phi_\alpha(c_i)}{\Phi_\alpha\{c_i\}}\end{aligned} \tag{11}$$

where $\Phi(X)$ and $\Gamma(X)$ operators are defined as follows:

$$\Phi_\alpha\{c_i\} = \sum_{i=1}^{n}\left(\frac{\alpha}{1-\alpha}\right)^{c_i} \tag{12}$$

$$\Phi'_\alpha\{c_i\} = \sum_{i=1}^{n}c_i\left(\frac{\alpha}{1-\alpha}\right)^{c_i} \tag{13}$$

$$\Phi_\alpha(c_i) = \left(\frac{\alpha}{1-\alpha}\right)^{c_i} \tag{14}$$

$$\Gamma(\alpha) = \log\left(\frac{\alpha}{1-\alpha}\right)\cdot\alpha(1-\alpha) \tag{15}$$

The distribution graph of $\Gamma(\alpha)$ function is shown in Fig. 2.

As shown in Fig. 2, the extremum of $\Gamma(\alpha)$ is under 0.25 while α is from 0 to 1. Actually, the interval of α is generally set (0.5,1), as we hope to output the value between average and maximum of input in AlphaMEX pooling.

3.3. Network compression

In a typical convolutional neural network, the fully connected layer takes 70∼80% parameters, which makes it the heaviest part of the network. Fig. 3(a) shows the connections of the fully connected layer.

To compress the parameters and connections of the fully connected layer, three kinds of methods are proposed, as shown in Fig. 3(b)–(d). The first method is total Global Pool, which extracts only one output from each feature-maps instead of a full connection to compress most parameters. Another method combines the Global Pool with full connections in a mix structure, as shown in Fig. 3(c). This method keeps both advantages of these two operations but compresses less parameters. The last method also combines Global Pool with full connection, called insert structure, as shown in Fig. 3(d), but it separates the original layer into two layers which will increase the depth of the network.

According to the above analysis, the total Global Pool method compresses most parameters and connections, and it reduces the overfitting of network. The most common Global Pool is Global Average Pool layer, which is widely used in state-of-the-art CNNs, like NIN [33], ResNet [26], DenseNet [27], etc. To avoid the weakness (see Section 4.4) of Global Average Pool and improve the accuracy, AlphaMEX Global Pool, a novel Global Pool operator, has been proposed, which transforms the AlphaMEX function into Global Pool. As it could be well trained end-to-end by most first order learning algorithms (see Section 3.2), it is smarter than any traditional global pooling methods.

3.4. Nonlinear analysis

In AlphaMEX Global Pool layer, each feature-map is fed into independent AlphaMEX operator, as shown in Fig. 3(b). The output extracted from each feature-map is given as follows:

$$y_k = AlphaMEX_{\alpha_k}\left\{x_{ij}^k\right\}_{1\le j\le m,1\le i\le n} \tag{16}$$

where x_{ij}^k is the ijth element of kth feature-map, m and n represent the size of feature-maps. Each feature-map could have its independent α_k, or all the feature-maps share a single α.

As the AlphaMEX function has a nonlinear characteristic (see Section 3.2), the degree of its nonlinearity will influence the performance and training efficiency. To analysis the nonlinear characteristic of AlphaMEX, six different distributions of x_{ij} have been tested as input, and the output curves are plotted in Fig. 4 with α ranges from −1 to 1.

As shown in Fig. 4, the nonlinearity of AlphaMEX is influenced by different input distributions. The larger range of the input distribution become, the more nonlinearity the output curve has. When the range changes to [10,100], it becomes over nonlinear, which will cause exploding gradient. However, the commonly used operator, Batch Normalization [34], solved this problem, which could normalize feature-maps, and confirm a stable nonlinear working state for AlphaMEX. Give consideration to both nonlinearity and computational complexity, [−1,1] and [0,1] are optimum ranges of input distribution, which make the AlphaMEX Global Pool a better trade-off between average and maximum global pool.

4. Experimental results and discussion

In this section, several benchmark datasets, CIFAR-10/CIFAR-100 [40], SVHN [41] and ImageNet [42] are used for performance evaluation. CIFAR-10/CIFAR-100 are the tiny images dataset. CIFAR-10 dataset has 60,000 images in 10 classes, where there are 6000 images per class. It is split into the training set and the test set. Specifically, there are 50,000 images in the training set and 10,000

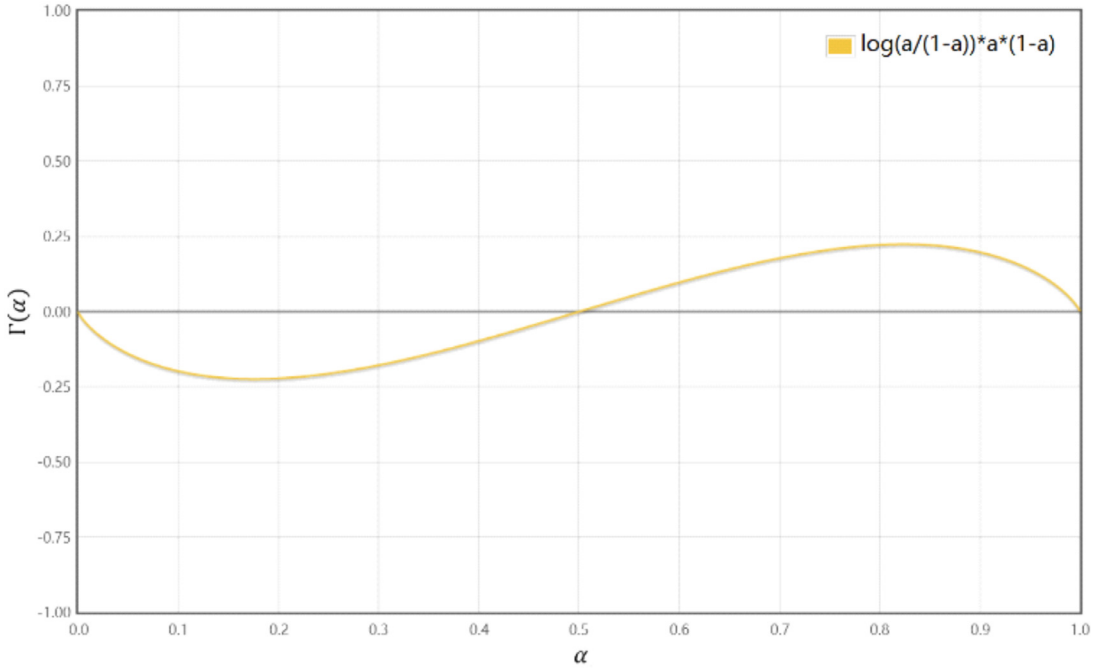


Fig. 2. Distribution graph of $\Gamma(\alpha)$ function. The extremum of $\Gamma(\alpha)$ is under 0.25 while α is from 0 to 1. Actually, the interval of α is generally set (0.5,1) in AlphaMEX Global Pool.

images in the test set. The size of the image is 32 × 32. CIFAR-100 dataset is the same as CIFAR-10, except it has 100 classes where each class contains 600 images. A standard data augmentation scheme (mirroring/shifting) is adopted which is widely used for CIFAR datasets [40]. SVHN [41] is the street view house numbers dataset. It consists of 10 classes for digit '0′ to digit '9′, where there are 73,257 images for training, 26,032 images for test, and 531,131 additional images. The size of the image in SVHN is also 32 × 32. The large-scale dataset ImageNet consists 1.2 million images for training, and 50,000 for validation, from 1,000 classes.

Three typical and state-of-the-art CNN architectures (i.e., ALL-CNN [49], ResNet [26], and DenseNet [27]) are adopted to demonstrate the effectiveness of the proposed method. Although the structures of these three architectures are different, like ALL-CNN is a kind of simple and classic plain CNN while ResNet and DenseNet has residual and dense-block structures, they all have a same layer between the last feature-maps and the fully-connected or softmax layer, the Global Average Pooling layer, which is used as the baseline of experiments.

4.1. Experiments based on ALL-CNN architecture

In this section, the traditional Global Average Pool [33] and the proposed AlphaMEX Global Pool are firstly compared based on the ALL-CNN-like architecture, which is a simple but typical plain convolutional neural network. As the original ALL-CNN [49] has its own image preprocessing strategy, for simplification, the baseline model in Table 1 is called ALL-CNN-like model, which follows the base structure of All-CNN–C [49].

In our experiment, each feature-map is fed into an independent AlphaMEX operator, which means each feature-map has its own α. Fig. 1 shows the structure of All-CNN and AlphaMEX-All-CNN. As there are ten feature-maps in the original All-CNN-like structure, ten αs are used in AlphaMEX-All-CNN. Fig. 5 shows the trajectories of ten αs in training, initialized 0.65. The clip-by-value function of TensorFlow [52] is used to keep αs staying in (0.5,1). Values of αs are clipped to 0.5 and 1.0 during the training process.

Table 2 shows the computational complexity and error rates of baseline structures and the proposed AlphaMEX structure on CIFAR-10 dataset (with standard data augmentation). FLOPs represents the number of floating-points operations required to classify an image. To circumvent the computational price of exp and log functions included in AlphaMEX and MEX, we used approximations that require up to 10 FLOPs per operation [50]. The execution time is also shown in Table 2, including training and inference. As the MEX function is not designed for CNN, but for SimNet [50], we followed the base structure of MEX and designed global MEX pooling to demonstrate the effectiveness of the AlphaMEX Global Pool. To avoid the NaN (Not a Number) error in programming, βs of MEX are initialized 5.0 and clipped to 0–50 during the training process.

All of the structures have the same training strategy. The training consists of 400 epochs, with the initial learning rate 0.05 and 0.1 times decay at epoch 200 and 300. The batch size in training is 256 while it is set 200 in testing. αs are initialized 0.65. The weights of each layer are initialized by Xavier initialization [53] with a scale for L2 regularization [54] 0.001 and the momentum is set to 0.9 in momentum optimization [55].

The best performance of the All-CNN-like and ConvPool-CNN structures reach 7.61% and 7.90% error rates on CIFAR-10 dataset with standard data augmentation in the experiment, while the AlphaMEX-All-CNN reaches 7.07%. The proposed AlphaMEX global pooling method improved the performance of All-CNN-like structure by 7.1% without adding any layers or too much execution time, which demonstrates its effectiveness.

4.2. Experiments based on resnet architecture

In this section, the structure of Global Average Pool and AlphaMEX Global Pool are compared based on ResNet [26] architecture. In the training stage, the parameters are set as follows. It consists of 200 epochs, with the initial learning rate 0.1 and 0.1 times decay at epoch 100 and 150. The batch size in training is 128 while it is set 250 in testing. The weights of each layer are initialized by Xavier initialization [53] with a scale for L2

Table 1
Three base CNN structures for evaluating the performance of AlphaMEX Global Pool.

ConvPool-CNN	All-CNN-like	AlphaMEX-All-CNN
Input 32 × 32 RGB image		
3 × 3 conv. 96 ReLU	3 × 3 conv. 96 ReLU	3 × 3 conv. 96 ReLU
3 × 3 conv. 96 ReLU	3 × 3 conv. 96 ReLU	3 × 3 conv. 96 ReLU
3 × 3 conv. 96 ReLU		
3 × 3 max-pooling stride2	3 × 3 conv. 96 ReLU with stride r = 2	3 × 3 conv. 96 ReLU with stride r = 2
3 × 3 conv. 192 ReLU	3 × 3 conv. 192 ReLU	3 × 3 conv. 192 ReLU
3 × 3 conv. 192 ReLU	3 × 3 conv. 192 ReLU	3 × 3 conv. 192 ReLU
3 × 3 conv. 192 ReLU		
3 × 3 max-pooling stride2	3 × 3 conv. 192 ReLU with stride r = 2	3 × 3 conv. 192 ReLU with stride r = 2
	3 × 3 conv. 192 ReLU	
	1 × 1 conv. 192 ReLU	
	1 × 1 conv. 10 ReLU	
	global averaging	AlphaMEX global pooling
	10 or 100-way softmax	

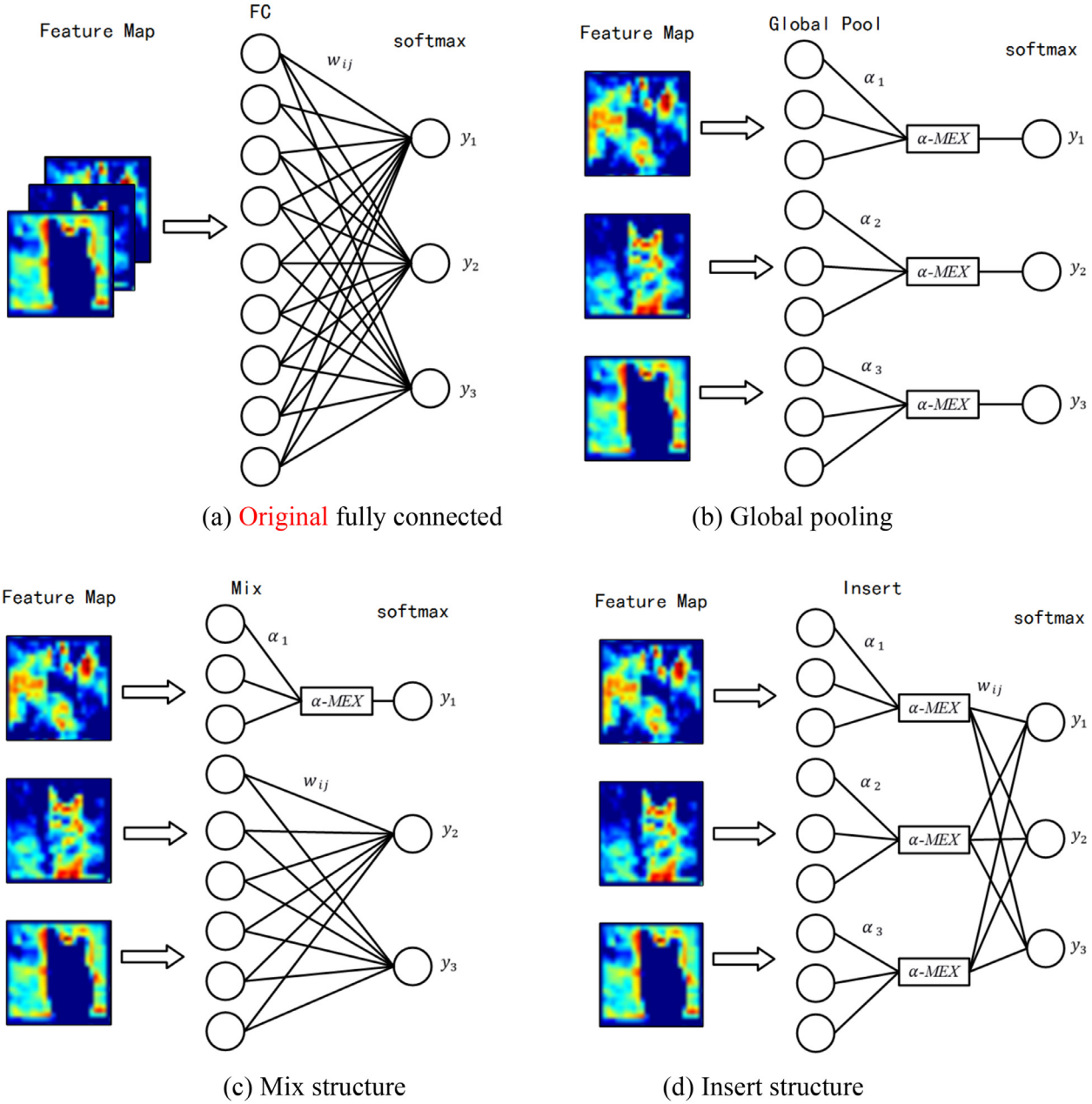


Fig. 3. Network compression by AlphaMEX. (a) the structure of original fully connected layer in CNN; (b) the global pooling layer, which extracts one output from each feature-maps and compresses most parameters; (c) the mix structure of Global Pool operator and fully connection; (d) the insert structure which separates original layer into two layers and increases the depth.

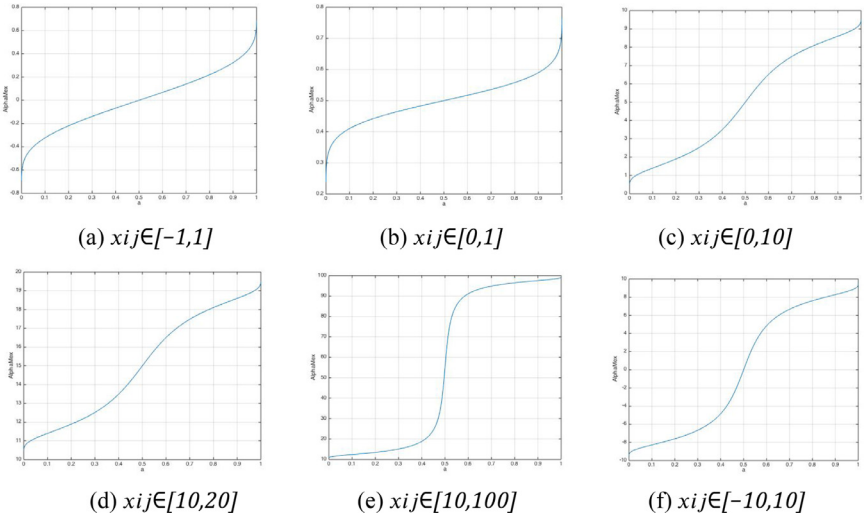


Fig. 4. Different nonlinearity of AlphaMEX function. It is influenced by distributions of input x_{ij}, (a) $x_{ij} \in [-1, 1]$; (b) $x_{ij} \in [0, 1]$; (c) $x_{ij} \in [0, 10]$; (d) $x_{ij} \in [10, 20]$; (e) $x_{ij} \in [10, 100]$; (f) $x_{ij} \in [-10, 10]$, the larger range of the input distribution become, the more nonlinearity the output has. (a)(b) show better nonlinearity.

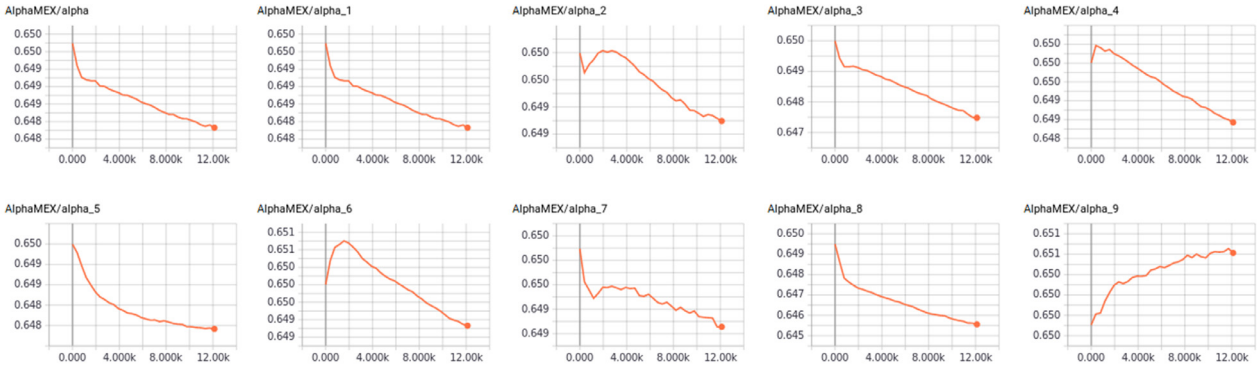


Fig. 5. Trajectories of αs in training. Ten αs have different trajectories, which means AlphaMEX operator could be well trained and each feature-map channel has its own optimum α.

Table 2
The computational complexity and error rates of four base CNN structures for evaluating the performance of AlphaMEX Global Pool. FLOPs represents the number of floating-points operations required to classify an image.

Method	FLOPs (million)	Training (sec/batch)	Inference (msec/image)	Error rate (%)
All-CNN-like	542.29	0.078	0.114	7.61
ConvPool-CNN	797.61	0.078	0.114	7.90
AlphaMEX-All-CNN	542.29	0.079	0.117	7.07
MEX-All-CNN	542.29	0.079	0.117	8.32

regularization [54] 0.0002 and the momentum is set to 0.9 in momentum optimization [55].

Table 3 compares the performance of the original ResNet-32 structure on CIFAR10+(CIFAR-10 with standard data augmentation) and the purposed AlphaMEX structure in different hyper-parameter settings. The second column represents the number of α in each network. As there are 64 feature-maps before the fully-connected layer of ResNet-32, three types of α has been test. First, different feature-maps have their own α, which makes the number of α 64. Second, only one α is used in each different feature-maps. Furthermore, each feature-map has been segmented into four equal parts and each tiny part has its own α, which takes 256 αs. To better reflect the different performance between the Global Average Pool and AlphaMEX Global Pool in ResNet structure, the optimization and learning rate are set independently to the AlphaMEX layer while the other parts of the original ResNet-32 remain unchanged. The error rate of ResNet-32 original is 7.44% and the best performance of AlphaMEX reaches 7.06% with Adadelta [56] optimization and 1×10^{-4} learning rate, which improved the performance by 5.1%.

As shown in Table 3, different learning rates will influence the performance of AlphaMEX. A higher learning rate will shake parameters too much to reach the optimal point while a lower learning rate slows down the learning step. To solve this problem, the learning rate decay strategy is used in the test. Table 4 compares the performance of the state-of-the-art ResNet-110 structure and our purposed AlphaMEX-ResNet-110 with learning rate decay. The AlphaMEX Global Pool layer outperforms the origin global pooling layer in all of the learning rates with Adadelta optimization. The best performance of AlphaMEX Global Pool is 5.84% with 0.001 learning rate and three times 0.1 learning rate decay, which outperforms the origin ResNet-110 by 8.3% on CIFAR10+.

To further demonstrate the effectiveness and generality of our proposed AlphaMEX Global Pool, Fig. 6 shows the accuracy curve

Table 3
Comparison of the original ResNet-32 structure and the purposed AlphaMEX structure in different hyper-parameter settings on CIFAR10+. Top-1 error rate (%) is used.

ResNet-32	Number of α	Initial α	Optimization	Learning rate of α	Error rate (%)
1	64	0.5	SGD	1×10^{-4}	7.57
2	64	0.5	SGD	1×10^{-3}	7.38
3	64	0.5	SGD	1×10^{-2}	8.14
4	64	0.5	Ada	1×10^{-3}	7.33
5	64	0.8	Ada	1×10^{-4}	7.06
6	1	0.8	Ada	1×10^{-3}	7.18
7	64	0.8	Ada	1×10^{-3}	7.48
8	256	0.8	Ada	1×10^{-3}	7.82
Origin	–	–	–	–	7.44

Table 4
Comparison of the origin ResNet-110 structure and the purposed AlphaMEX structure in different learning rate strategy on CIFAR10+. Top-1 error rate (%) is used.

ResNet-110	Initial α	Optimization	Learning rate of α	Learning rate decay	Error (%)
1	0.8	Ada	1×10^{-4}	No decay	6.04
2	0.8	Ada	1×10^{-3}	No decay	6.46
3	0.8	Ada	1×10^{-3}	0.1 Three times	5.84
Origin	–	–	–	–	6.37

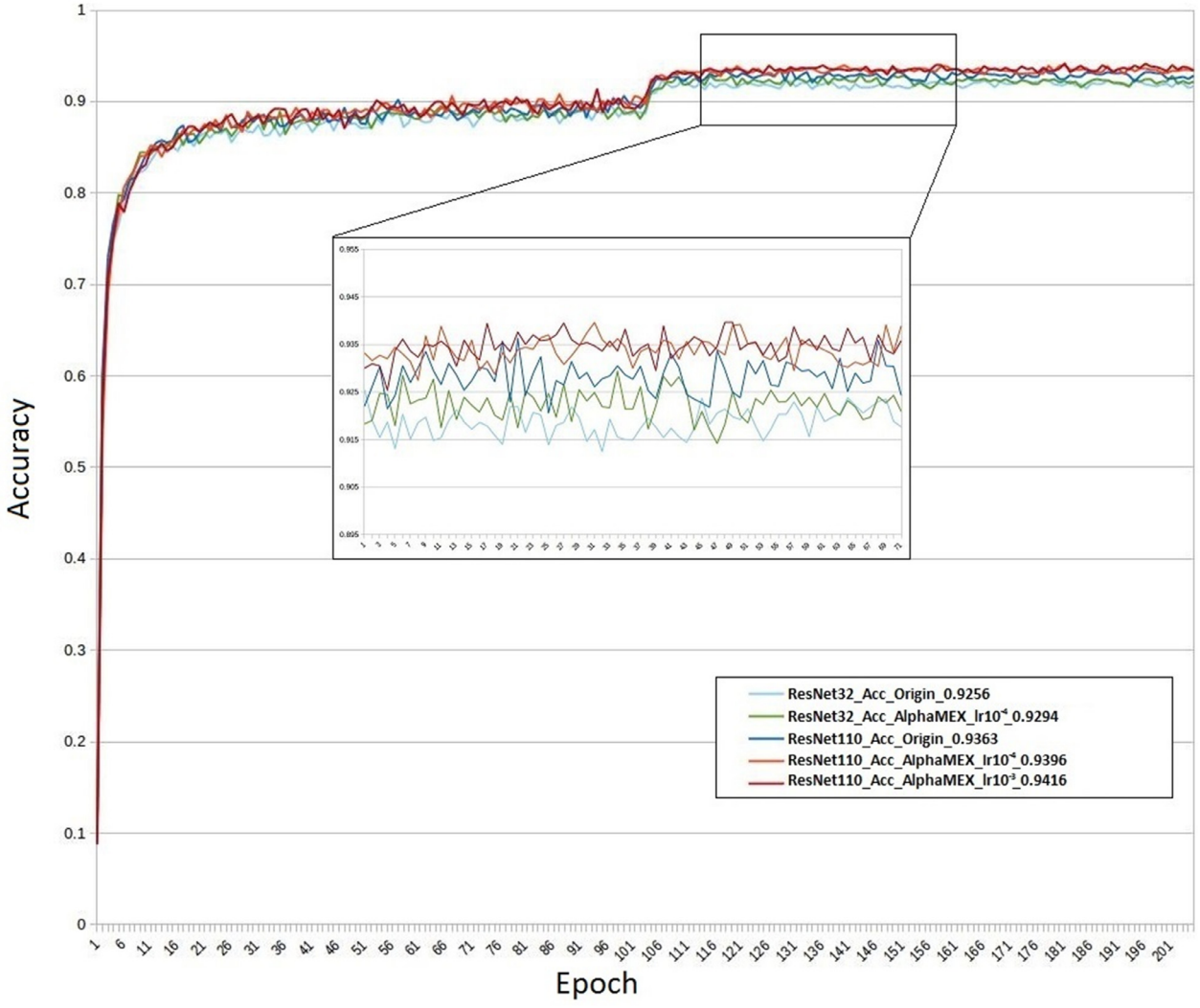


Fig. 6. The accuracy curve of ResNet and AlphaMEX structures. It is tested on the whole test set of CIFAR10+at each epoch. The AlphaMEX structure is in red, which shows a better performance on both the convergence and maximum accuracy. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 5
DenseNet40 and AlphaMEX-DenseNet40 architectures for CIFAR.

Layers	Output Size	Feature-maps	DenseNet-40 ($k = 12$)	AlphaMEX-DenseNet-40($k = 12$)
Convolution	32 × 32	24	3 × 3 conv, stride 1	
Dense Block (1)	32 × 32	268	[3 × 3 conv] × 12	
Transition Layer (1)	32 × 32	268	1 × 1 conv	
	16 × 16	268	2 × 2 average pool, stride 2	
Dense Block (2)	16 × 16	312	[3 × 3 conv] × 12	
Transition Layer (2)	16 × 16	312	1 × 1 conv	
	8 × 8	312	2 × 2 average pool, stride 2	
Dense Block (3)	8 × 8	456	[3 × 3 conv] × 12	
Classification Layer	1 × 1	456	8 × 8 global average pool	AlphaMEX global pool
	–	–	456 × 10 fully-connected, softmax	
Error rate	–	–	5.24%	5.03%

of original ResNet and the proposed structure. It is tested on the whole test set of CIFAR10 + at each epoch. The AlphaMEX Global Pool shows a better performance on both the convergence and maximum accuracy.

4.3. Experiments based on densenet architecture

In this section, the proposed structure AlphaMex Global Pool is tested based on DenseNet [27] architecture, which outperforms the current state-of-the-art results on most of the benchmark tasks.

Traditional convolutional neural network pass a single image through non-linear transformations $H_l(\cdot)$ layer by layer, where l indexes the layer, $H_l(\cdot)$ can be a composite function of operations like Convolution [25], Pooling [46], ReLU [35], or Batch Normalization [34]. ResNet [26] add a skip-connection that bypasses the non-linear transformation:

$$X_l = H_l(X_{l-1}) + X_{l-1} \tag{17}$$

where X_l represents the output of the l^{th} layer.

To further improve the information flow between layers, DenseNet add connections from any layer to all subsequent layers [27], the l^{th} layer receives the feature-maps of all preceding layers:

$$X_l = H_l([X_0, X_1, \ldots, X_{l-1}]) \tag{18}$$

where [.] refers to concatenations of the feature-maps.

The DenseNet structure used in our experiments has three dense blocks that each has an equal number of layers which has been shown in Fig. 7(a). As shown in Fig. 7(a), a Global Average Pool layer is performed at the end of the last dense block in the original DenseNet40 [27], which has been replaced by the proposed AlphaMEX Global Pool layer in AlphaMEX-DenseNet without changing any other layers.

Details of the DenseNet40 and AlphaMEX-DenseNet are shown in Table 5. The number of feature-maps increased while images flow through DenseBlocks. At the end of the last DenseBlock, 456 feature-maps are fed into the global pooling layer with each size 8 × 8. Both of the DenseNet-40 and the AlphaMEX-DenseNet-40 have the same training parameters, the total training epoch is 300 while the initial learning rate is 0.1 and decay at 150 and 225 epoch by 0.1 times. αs are initialized 0.75. The batch size is 64 in both training and testing stage and 0.9 momentum is used in optimization.

As shown in Table 5, the error rate of proposed AlphaMEX-DenseNet on CIFAR10 + outperforms origin DenseNet(k = 12) by 4.0% without adding any layers or too much redundant parameters, which demonstrate the effectiveness of the AlphaMEX Global Pool layer.

In Table 6, some state-of-the-art methods (i.e., NIN [33], SimNet [50], All-CNN [49], DSN [51], Hightway [32], ResNet [26][57], and DenseNet [27]) are compared to our proposed AlphaMEX Global Pool method on CIFAR-10/CIFAR100 [40] and SVHN [41]. In the training stage of SVHN, the parameters are set as follows. It consists of 40 epochs, with the initial learning rate 0.1 and 0.1 times decay at epoch 20 and 30. The batch size in training and testing is 64. αs are initialized 0.75. The parameters are given in million while the top-1 error rate is used. It can be seen that our proposed method achieves the competitive classification accuracy.

4.4. Experiments on imagenet

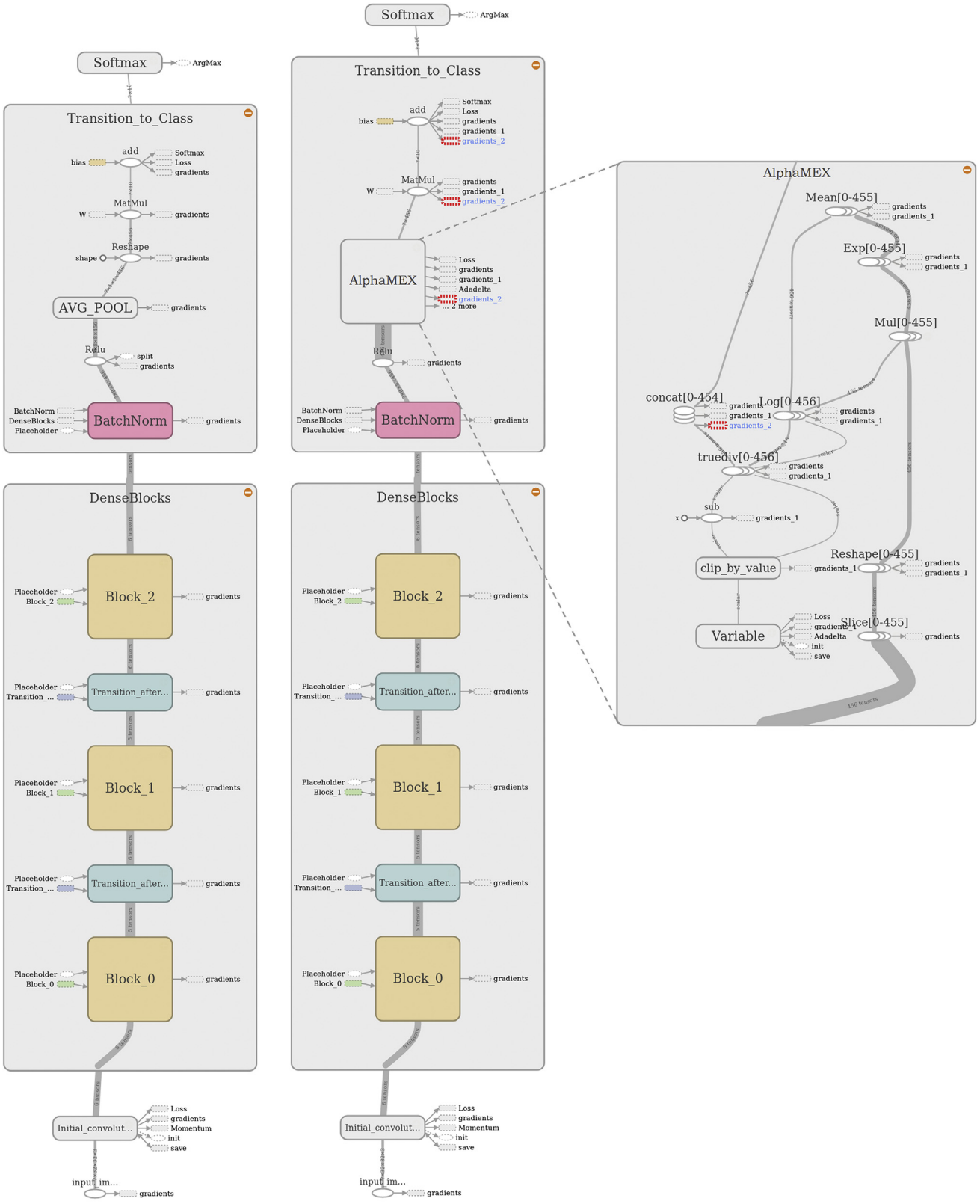
To further demonstrate the effectiveness of proposed method on the large-scale dataset, the proposed AlphaMEX Global Pool layer is tested on ImageNet [42]. The Global Average Pool and AlphaMEX Global Pool are compared based on the ResNet [26] architecture. The training parameters are set as follows. It consists of 55 epochs, where the learning rate decreases four times exponentially from 10^{-2} to 10^{-4}. The batch size in training is 64 while it is set 512 in testing. Table 7 shows the top-1 error rate and top-5 error rate. AlphaMEX Global Pool outperforms Global Average Pool by 1.7% on top-1 error rate and 1.4% on top-5 error rate. It can be concluded that our proposed AlphaMEX method achieves better accuracy on large-scale dataset.

4.5. Sparsity analysis

To help understand why the AlphaMEX Global Pool has better performance than Global Average Pool, Fig. 8 shows ten feature-maps before the global pooling layer in DenseNet [27]. As the output size of the last Dense Block layer is 8*8, each feature-map has 64 tiny blocks, and the brighter the feature is, the more activation the feature has. It can be seen from the feature-maps that, the features, extracted by previous layers and as the input of the global pooling layer, has a kind of sparsity.

The same sparse feature-maps condition also appears in ResNet. The reason why feature-maps become sparse lies in the Batch Normalization (BN) [34] and ReLU [35] layers. As the combination of BN and ReLU could extract features efficiently, more and more CNN structures tend to use BN with ReLU as a common feature normalize and nonlinear transform operator. As shown in Fig. 9, the combined operator will extract the most active features from previous feature-maps, which makes the convolution layer more efficient, but is inappropriate for Global Average Pool.

A Global Average Pool will dilute the active features if the feature-maps are sparse, which will influence the classification accuracy of the last fully connected layer. Compared to other methods, our proposed AlphaMEX Global Pool method has the better trade-off between the average and maximum active feature, which will learn a best way to pass and extract information through global pooling. That makes the convolutional neural network smarter and more accurate.



(a)DenseNet40 (b)AlphaMEX-DenseNet40

Fig. 7. DenseNet40 and AlphaMEX-DenseNet40 architectures for CIFAR. (a) the original DenseNet40 for CIFAR with three dense blocks and a Global Average Pool layer; (b) the structure of AlphaMEX-DenseNet40, AlphaMEX operator has been zoomed in to show details of the function.

Table 6
Error rates (%) on CIFAR and SVHN datasets. "+" indicates standard data augmentation (translation and/or mirroring). AlphaMEX Global Pool based structures achieve lower error rates than each compared state-of-the-art structures without adding any layers or too much redundant parameters.

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network[33]	–	0.97M	10.41	8.81	35.68	–	2.35
SimNet[50]	–	–	–	7.82	–	–	–
Deeply Supervised Net[51]	–	0.97M	9.69	7.97	–	34.57	1.92
Highway Network[32]	–	–	–	7.72	–	32.39	–
All-CNN[49]	–	1.3M	9.08	7.25	–	33.71	–
AlphaMEX-All-CNN	–	1.3M	**8.99**	**7.07**	–	**29.10**	**–**
ResNet[26]	110	1.7M	–	6.61	–	–	–
ResNet(reported by [57])	110	1.7M	13.63	6.41	44.74	27.22	2.01
AlphaMEX-ResNet	110	1.7M	**8.41**	**5.84**	**32.87**	27.71	**1.97**
DenseNet(k = 12)[27]	40	1.0M	7.00	5.24	27.55	24.42	1.79
AlphaMEX-DenseNet(k = 12)	40	1.0M	**6.54**	**5.03**	**27.24**	**23.71**	**1.73**

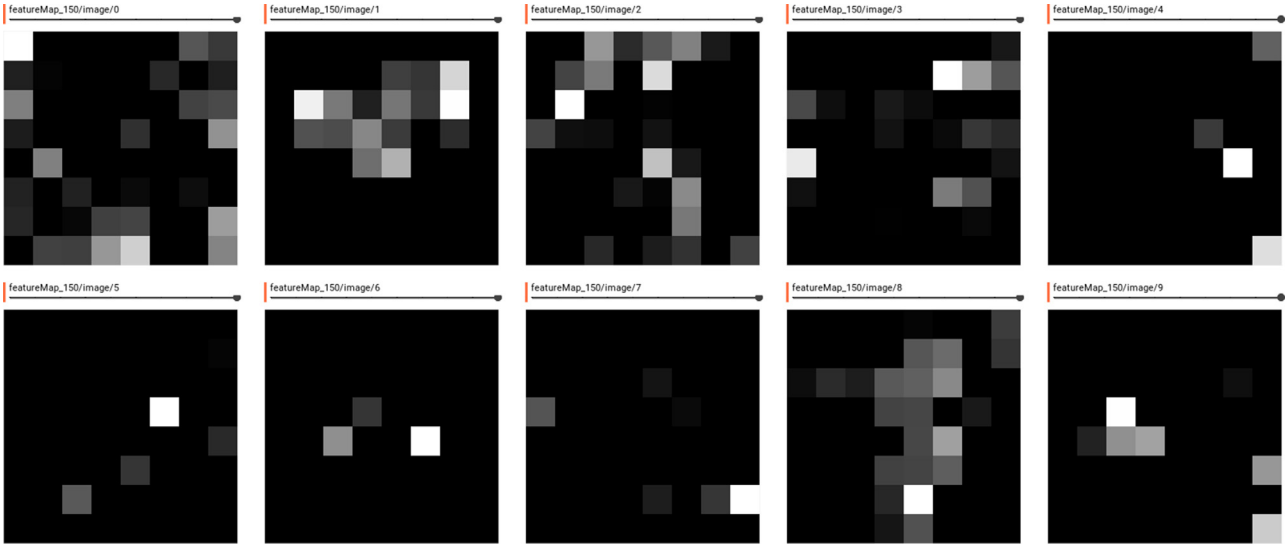


Fig. 8. Feature-maps fed into Global Average Pool in DenseNet. Each feature-map has 64 tiny blocks (features), the brighter the block is, the more activation the feature has. The feature-maps show a kind of sparsity.

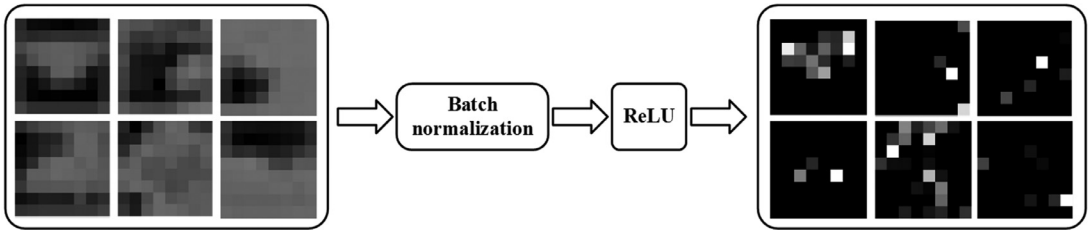


Fig. 9. Sketch map of feature extraction by BN and ReLU. The combined operator will extract the most active features from previous feature-maps, which makes the convolution layer more efficient, but also makes the feature-map sparse.

Tables 7
Experiments on ImageNet. ResNet architecture is used to compare AlphaMEX Global Pool and Global Average Pool. Top-1 error rate (%) and top-5 error rate (%) are both shown.

Method	Top-1	Top-5
Global Average Pool	29.8	10.5
AlphaMEX Global Pool	28.1	9.1
Difference	1.7	1.4

5. Conclusions and future work

In this paper, we proposed a novel log-mean-exp function AlphaMEX and an end-to-end trainable global pooling operator AlphaMEX Global Pool for convolutional neural network. Comparing to traditional global pooling method, the proposed AlphaMEX Global Pool avoids data dilution from the sparse feature-maps and transfers information more effectively. Without adding any layers or too much redundant parameters, our proposed method achieves superior performance compared with state-of-the-arts methods. Experimental results on CIFAR-10/CIFAR-100 and SVHN demonstrate the effectiveness of our proposed method. In the future work, we will explore our global pooling operator to other deep learning neural networks like RNNs and R-CNNs. Also we will analyse the visualization and attention of the AlphaMEX Global Pool.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61772052).

Declarations of interest

None.

References

[1] J. Cao, Y. Pang, X. Li, Learning multilayer channel features for pedestrian detection, IEEE Trans. Image Process. 26 (7) (2016) 3210–3220.
[2] Y. Chen, Y. Ma, D. Kim, S. Park, Region-based object recognition by color segmentation using a simplified PCNN, IEEE Trans. Neural Netw. Learn. Syst. 26 (8) (2015) 1682–1697.
[3] R. Girshick, Fast r-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2015.
[4] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: Proceedings of the European Conference on Computer Vision, 2016.
[5] H. Li, J. Chen, H. Lu, Z. Chi, CNN for saliency detection with low-level feature integration, Neurocomputing 226 (2017) 212–220.
[6] M. Xin, H. Zhang, H. Wang, ARCH: adaptive recurrent-convolutional hybrid networks for long-term action recognition, Neurocomputing 178 (2016) 87–102.
[7] Q. Yu, J. Wang, S. Zhang, Y. Gong, J. Zhao, Combining local and global hypotheses in deep neural network for multi-label image classification, Neurocomputing 235 (2017) 38–45.
[8] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: a deep convolutional encoder–decoder architecture for image segmentation, in: Proceedings of the CoRR, 2015 arXiv: abs/1511.00561.
[9] T. Chen, L. Lin, L. Liu, X. Luo, X. Li, DISC: deep image saliency computing via progressive representation learning, IEEE Trans. Neural Netw. Learn. Syst. 27 (6) (2016) 1135–1149.
[10] J. Dai, K. He, J. Sun, Instance-aware semantic segmentation via multi-task network cascades, in: Proceedings of the IEEE Conference on Computer Visual Pattern Recognition, 2016.
[11] J. Redmon, S. Divvala, R. Girshick, You only look once: unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
[12] J. Dai, Y. Li, K. He, R-FCN: object detection via region-based fully convolutional networks, in: Proceedings of the Advances in Neural Information Processing systems, 2016.
[13] R. Girshick, J. Donahue, T. Darrell, Region-based convolutional networks for accurate object detection and segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2016) 142–158.
[14] J. Wang, Y. Yang, J. Mao J, CNN-RNN: a unified framework for multi-label image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
[15] T. Liang, X. Xu, P. Xiao, A new image classification method based on modified condensed nearest neighbor and convolutional neural networks, Pattern Recognit. Lett. 94 (2017) 105–111.
[16] E. Maggiori, Y. Tarabalka, G. Charpiat, Convolutional neural networks for large-scale remote-sensing image classification, IEEE Trans. Geosci. Remote Sens. 55 (2) (2017) 645–657.
[17] L.C. Chen, J.T. Barron, G. Papandreou, Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
[18] L.C. Chen, G. Papandreou, I. Kokkinos, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs, in: Proceedings of the Computer Vision and Pattern Recognition, 2016 arXiv: 1606.00915.
[19] A. Ferrari, S. Lombardi, A. Signoroni, Bacterial colony counting with convolutional neural networks in digital microbiology imaging, Pattern Recognit. 61 (2017) 629–640.
[20] I. Ozer, Z. Ozer, O. Findik, Noise robust sound event classification with convolutional neural network, Neurocomputing 272 (2018) 505–512.
[21] K. Muhammad, J. Ahmad, S.W. Baik, Early fire detection using convolutional neural networks during surveillance for effective disaster management, Neurocomputing 288 (2018) 30–42.
[22] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho, S.W. Baik, Convolutional neural networks based fire detection in surveillance videos, IEEE Access 6 (2018) 18174–18183.
[23] H. Li, G. Li, X. Ji, L. Shi, Deep representation via convolutional neural network for classification of spatiotemporal event streams, Neurocomputing 299 (2018) 1–9.
[24] A. Pertusa, A.J. Gallego, M. Bernabeu, MirBot, a collaborative object recognition system for smartphones using convolutional neural networks, Neurocomputing 293 (2018) 87–99.
[25] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2012.
[26] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Visual Pattern Recognition, 2016.
[27] G. Huang, Z. Liu, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the Computer Vision and Pattern Recognition, 2016 arXiv:1608.06993.
[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Visual Pattern Recognition, 2015.
[29] F.N. Iandola, S. Han, M.W. Moskewicz, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size, in: Proceedings of the ICLR, 2016 arXiv: 1602.07360.
[30] M. Rastegari, V. Ordonez, J. Redmon, Xnor-net: Imagenet classification using binary convolutional neural networks, in: Proceedings of the European Conference on Computer Vision, 2016.
[31] X. Zhang, X. Zhou, M. Lin, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the Computer Visual Pattern Recognition, 2017 arXiv: 1707.01083.
[32] R.K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, in: Proceedings of the NIPS, 2015.
[33] M. Lin, Q. Chen, S. Yan, Network in network, in: Proceedings of the ICLR, 2014.
[34] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the ICML, 2015.
[35] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the AISTATS, 2011.
[36] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the EMNLP, 2014 arXiv:1408.5882.
[37] M.D. Zeiler, R. Fergus, Stochastic pooling for regularization of deep convolutional neural networks, in: Proceedings of the International Conference on Learning Representations, 2013.
[38] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of the COMPSTAT'2010. Physica-Verlag HD, 2010, pp. 177–186.
[39] D. Kingma, J. Ba. Adam, A method for stochastic optimization, in: Proceedings of the ICLR, 2014 arXiv: 1412.6980.
[40] A. Krizhevsky, Learning multiple layers of features from tiny images, 2009, Tech report.
[41] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in: Proceedings of the Advances in Neural Information Processing Systems Workshop on Deep Learning and Unsupervised Feature Learning, 2011.
[42] J. Deng, W. Dong, R. Socher, L. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: Proceedings of the IEEE Conference on Computer Visual Pattern Recognition, 2009.
[43] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proceedings of the CoRR, 2015 arXiv: abs/1409.1556.
[44] Y. Pang, M. Sun, X. Jiang, X. Li, Convolution in convolution for network in network, IEEE Trans. Neural Netw. Learn. Syst (2017) 2676130, doi:10.1109/TNNLS.2017.
[45] D. Zhang, J. Han, J. Han, L. Shao, Cosaliency detection based on intrasaliency prior transfer and deep intersaliency mining, IEEE Trans. Neural Netw. Learn. Syst. 27 (6) (2016) 1163–1176.
[46] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradientbased learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.
[48] I.V. Tetko, D.J. Livingstone, A.I. Luik. Neural network studies. 1. Comparison of overfitting and overtraining, J. Chem. Inf. Comput. Sci., 35(1995) 826–833.
[49] J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: the all convolutional net, in: Proceedings of the ICLR, 2014 arXiv: 1412.6806.
[50] N. Cohen, O. Sharir, A. Shashua, Deep simnets, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
[51] C. Lee, S. Xie, P. Gallagher, Z. Zhang, Z. Tu, Deeply-supervised nets, in: Proceedings of the Advances in Neural Information Processing Systems Workshop on Deep Learning and Representation Learning, 2014.
[52] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Kudlur, TensorFlow: a system for large-scale machine learning, in: Proceedings of the OSDI, 2016.
[53] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010.
[54] A.Y. Ng, Feature selection, L 1 vs. L 2 regularization, and rotational invariance, in: Proceedings of the Twenty-First International Conference on Machine Learning, ACM, 2004.
[55] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: Proceedings of the International Conference on Machine Learning, 2013.
[56] M.D. Zeiler, ADADELTA: an adaptive learning rate method, Computer Science, 2012, arXiv:1212.5701.
[57] G. Huang, Y. Sun, Z. Liu, D. Sedra, K.Q. Weinberger, Deep networks with stochastic depth, in: Proceedings of the ECCV, 2016.

Boxue Zhang received the B.S. degree in Mathematics and System science from Beihang University, Beijing, China, in 2014. Since 2016, he has been with the School of Electronic Information Engineering, Beihang University, where he is a Ph.D. candidate. His current interests include the key technologies in the image processing, big data processing and Deep Learning technology.

Qi Zhao received Ph.D in communication and information system from Beihang University, Beijing, China, in 2002. From 2002 to date, she is an associate professor and works in Beihang University. She was in the Department of Electrical and Computer Engineering at the University of Pittsburgh as a visiting scholar from 2014 to 2015. Her current research interests include image recognition and processing, communication signal processing, and target tracking.

Wenquan Feng received Ph.D. degree in communication and information system from Beihang University, Beijing, China. He is a professor and works in Beihang University. His current research interests include image target tracking, big data processing, satellite navigation and satellite communication.

Shuchang Lyu received the B.S. degree in Communication and Information from Shanghai University, Shanghai, China, in 2016. He has been with the School of Electronic Information Engineering, Beihang University, where he is a master. His current interests include the key technologies in the image processing, wearable device research, big data processing and Deep Learning technology.