Batch-Normalized Maxout Network in Network

Jia-Ren Chang FOLLOWWAR.CS00G@NCTU.EDU.TW
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

Yong-Sheng Chen YSCHEN@CS.NCTU.EDU.TW
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

Abstract

This paper presents a novel deep architecture, Maxout network In Network (MIN), which can enhance model discriminability and facilitate the process of information abstraction within the receptive field. The proposed network slides maxout multilayer perceptron to learn a variety of piecewise linear activation functions and to mediate the problem of vanishing gradients that can occur when using rectifier units. Moreover, batch normalization is applied to reduce the saturation of maxout units by pre-conditioning the model and dropout is applied to prevent overfitting. Finally, average pooling is used in all pooling layers, which can regularize the batch-normalized MIN architecture and aggregate its resultant stable features in order to facilitate information abstraction while accommodating different object positions. Our experiments demonstrated the improved or comparable classification performance when the MIN model was applied to MNIST, CIFAR-10, CIFAR-100, and SVHN datasets.

1. Introduction

Deep convolutional neural networks (CNNs) (Krizhevsky et al., 2012) have recently been applied to large image datasets, such as MNIST (LeCun et al., 1998), CIFAR-10/100 (Krizhevsky & Hinton, 2009), SVHN (Netzer et al., 2011), and ImageNet (Deng et al., 2009) for image classification (Hinton et al., 2012). A deep CNN is able to learn basic filters automatically and combine them hierarchically to enable the description of latent concepts for pattern recognition. In (Zeiler & Fergus, 2014), Zeiler et al. illustrated how deep CNN organizes feature maps and the discrimination among classes.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

Despite the advances that have been made in the development of this technology, many issues related to deep learning remain, including: (1) model discriminability and the robustness of learned features in early layers (Zeiler & Fergus, 2014); (2) the vanishing gradients and saturation of activation units during training (Glorot & Bengio, 2010); and (3) limited training data, which may lead to overfitting (Srivastava et al., 2014).

Because data are usually distributed on nonlinear manifolds, they are not separable by linear filters. For enhancing model discriminability, the Network In Network (NIN) (Lin et al., 2014) model uses a sliding micro neural network, multilayer perceptron (MLP), to increase the nonlinearity of local patches in order to enable the abstraction of greater quantities of information within the receptive fields. Similarly, Deeply Supervised Nets (DSN) (Lee et al., 2015) provides companion objective functions to constrain hidden layers, such that robust features can be learned in the early layers of a deep CNN structure.

The problem of vanishing gradients is essentially the shrinking of gradients backward through hidden layers. Some activation functions, such as sigmoid, are susceptible to vanishing gradients and saturation during the training of deep networks, due to the fact that higher hidden units become saturated at -1 or 1. Current solutions involve the use of rectified linear units (ReLU) to prevent vanishing gradients (Krizhevsky et al., 2012; Maas et al., 2013; Nair & Hinton, 2010) because ReLU activates above 0 and its partial derivative is 1. Thus gradients flow through while ReLU activates. Unfortunately, ReLU has a potential disadvantage. The constant 0 will block the gradient flowing through inactivated ReLUs, such that some units may never activate. Recently, the maxout network (Goodfellow et al., 2013) provided a remedy to this problem. Even when maxout output is 0, this value is from a maxout hidden unit and this unit may be adjusted to become positive afterwards. Another issue involves changes of data distribution during the training of deep networks that are likely to saturate the activation function. This changed data distribution can move input data into the saturated regime of

the activation function and slow down the training process. This phenomenon is referred to as internal covariate shift (Shimodaira, 2000). Ioffe et al. (Ioffe & Szegedy, 2015) addressed this problem by applying batch normalization to the input of each layer.

In this study, we aimed to increase nonlinearity within local patches and alleviate the problem of vanishing gradients. Based on the NIN (Lin et al., 2014) structure, we employ a maxout MLP for feature extraction and refer to the proposed model as Maxout network In Network (MIN). The MIN model uses batch normalization to reduce saturation and uses dropout to prevent overfitting. To increase the robustness to spatial translation of objects, furthermore, average pooling is applied in all pooling layers to aggregate the essential features obtained by maxout MLP.

2. Design of Deep Architecture

This section presents previous works related to the proposed MIN structure, including NIN, Maxout Network, and batch normalization, followed by the design of the MIN architecture.

2.1. NIN

The NIN model (Lin et al., 2014) uses the universal approximator MLP for the extraction of features from local patches. Compared to CNN, MLP, wherein an ReLU is used as the activation function, enables the abstraction of information that is more representative for the latent concepts. The NIN model introduced the $mlpconv$ layer which consists of a linear convolutional layer and a two-layer MLP. The calculation performed by the $mlpconv$ layer is as follows:

$$
\begin{aligned}
f^1_{i,j,n_1} &= \max\left({\mathbf{w}^1_{n_1}}^{\mathrm{T}}\mathbf{x}_{i,j} + b_{n_1}, 0\right) \ , \\
f^2_{i,j,n_2} &= \max\left({\mathbf{w}^2_{n_2}}^{\mathrm{T}}\mathbf{f}^1_{i,j} + b_{n_2}, 0\right) \ , \\
f^3_{i,j,n_3} &= \max\left({\mathbf{w}^3_{n_3}}^{\mathrm{T}}\mathbf{f}^2_{i,j} + b_{n_3}, 0\right) \ , \qquad (1)
\end{aligned}
$$

where (i, j) is the pixel index in the feature maps, $\mathbf{x}_{i,j}$ represents the input patch centered at location (i, j), and $n_1, n_2,$ and n_3 are used to index the channels of the feature maps. From another perspective, the $mlpconv$ layer can be viewed as equivalent to a cascaded cross-channel parametric pooling layer on a convolutional layer. The cascaded cross-channel parametric pooling layer linearly combines feature maps and then passes through ReLUs, thereby allowing the cross-channel flow of information.

However, the constant 0 will block the gradients flowing through the inactivated ReLUs and these ReLUs will not be updated during the training process. In this work, we adopted another universal approximator, maxout MLP, to overcome this problem.

2.2. Proposed MIN Architecture

The NIN (Lin et al., 2014) has capability to abstract representative features within the receptive field and thereby achieve good results in image classification. As described in Section 2.1, NIN uses ReLU as the activation function in $mlpconv$ layer. In this study, we replaced the ReLU activation functions in the two-layer MLP in NIN with the maxout units to overcome the vanishing gradient problem commonly encountered when using ReLU. Furthermore, we applied batch normalization immediately after convolutional calculation to avoid the covariate shift problem caused by the changes of data distribution. Specifically, we removed the activation function of the convolutional layer, thereby rendering it a pure feature extractor. The architecture of the proposed MIN model is presented in Figure 1. Feature maps in a MIN block are calculated as follows:

$$
\begin{aligned}
f^1_{i,j,n_1} &= \mathrm{BN}\left({\mathbf{w}^1_{n_1}}^{\mathrm{T}}\mathbf{x}_{i,j} + b^1_{n_j}\right) \ , \\
f^2_{i,j,n_2} &= \max_{m\in[1,k_1]}\left(\mathrm{BN}\left({\mathbf{w}^2_{n_m}}^{\mathrm{T}}\mathbf{f}^1_{i,j} + b^2_{n_m}\right)\right) \ , \\
f^3_{i,j,n_3} &= \max_{m\in[1,k_2]}\left(\mathrm{BN}\left({\mathbf{w}^3_{n_m}}^{\mathrm{T}}\mathbf{f}^2_{i,j} + b^3_{n_m}\right)\right) \ , \qquad (2)
\end{aligned}
$$

where $\mathrm{BN}(\cdot)$ denotes the batch normalization layer, (i, j) is the pixel index in the feature map, $\mathbf{x}_{i,j}$ represents the input patch centered at location (i, j), and *n* is used to index the channels of the feature maps that are constructed by taking the maximum across *k* maxout hidden pieces. Montufar et al. [18] demonstrated that the complexity of maxout networks increases with the number of maxout pieces or layers. By increasing the number of maxout pieces, the proposed model expands the ability to capture the latent concepts for various inputs.

From another perspective, a MIN block is equivalent to a cascaded cross-channel parametric pooling layer and a cross-channel max pooling on a convolutional layer. The MIN block linearly combines feature maps and selects the combinations that are the most informational to be fed into the next layer. The MIN block reduces saturation by applying batch normalization and makes it possible to encode information on pathways or in the activation patterns of maxout pieces (Wang & JaJa, 2014). This makes it possible to enhance the discrimination capability of deep architectures.

3. Experiments

In the following experiments, the proposed method was evaluated using four benchmark datasets: MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky & Hinton, 2009),

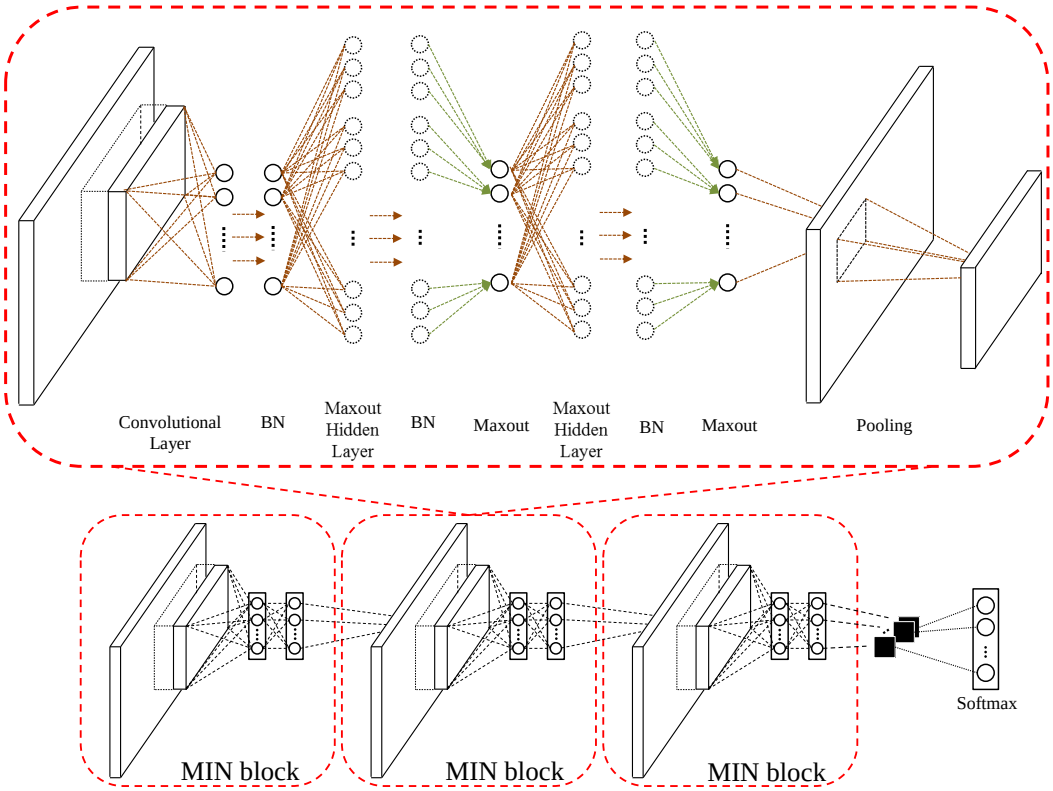


Figure 1. The architecture of the proposed MIN model.

CIFAR-100 (Krizhevsky & Hinton, 2009), and SVHN (Netzer et al., 2011). The proposed model consists of three stacked MIN blocks followed by a softmax layer. A MIN block includes a convolutional layer, a two-layer maxout MLP, and a spatial pooling layer. Dropout is applied between MIN blocks for regularization. Table 1 details the parameter settings which, for the sake of a fair comparison, are the same as those used in NIN (Lin et al., 2014). The network was implemented using the MatConvNet (Vedaldi & Lenc, 2014) toolbox in the Matlab environment. All models were trained with stochastic gradient descent. The momentum for all datasets were fixed to 0.9. The weight decay for all datasets were fixed to 0.0005. The mini-batch size for all datasets were set to 100.

3.1. MNIST

The MNIST dataset (LeCun et al., 1998) consists of handwritten digit images, 28 $\times$ 28 pixels in size, organized into 10 classes (0 to 9) with 60,000 training and 10,000 test samples. Testing on this dataset was performed without data augmentation. The model was trained for 79 epochs (5 hours on a GeForce Titan X). The first column of Table 1 indicates that our model has 0.45 million parameters. Table 2 compares the results obtained in this study with those obtained in previous works. Despite the fact that many methods can achieve very low error rates for MNIST dataset, we achieved a test error rate of 0.24%, which set a new state-of-the-art performance without data augmentation.

3.2. CIFAR-10

The CIFAR-10 dataset (Krizhevsky & Hinton, 2009) consists of color natural images, 32 $\times$ 32 pixels in size, from 10 classes with 50,000 training and 10,000 test images. For this dataset, we applied global contrast normalization and whitening in accordance with the methods outlined in (Goodfellow et al., 2013). To enable a comparison with previous works, the dataset was augmented by zero-padding 2 pixels on each side, which resulted in images 36 $\times$ 36 pixels in size. We then performed random corner cropping back to 32 $\times$ 32 pixels as well as random flipping on the fly during training. The model was trained for 222 epochs (10 hours on a GeForce Titan X). The middle column of Table 1 indicates that our model has 1.6 million parameters. Table 3 compares our results with those obtained in previous works. We obtained an error rate of 7.85% without data augmentation and 6.75% with data augmentation. These are the improved or comparable results. We also trained the model (0.95 millons parameters as shown

Table 1. Parameter settings of the proposed MIN architecture used in the experiments. The convolutional kernel is defined as (height) $\times$ (width) $\times$ (number of units). Below, we present the stride (st.), padding (pad) and batch normalization (BN) of the convolution kernel. In maxout MLP layers (MMLP), k indicates the number of maxout pieces used in one maxout unit. A softmax layer is applied to the last layer in the model (not shown here). The first column lists the parameters used in MNIST, the middle column lists those used in CIFAR-10/100 and SVHN, where as the last column lists that the model has same number of parameters with original NIN.

	MNIST	**CIFAR-10(100) SVHN**	**CIFAR-10 same # params**
Conv-1	5x5x128 / st. 1 / pad 2 BN	5x5x192 / st. 1/ pad 2 BN	5x5x96 / st. 1/ pad 2 BN
MMLP-1-1	1x1x96 / st. 1 / pad 0 k=5 / BN	1x1x160 / st. 1 / pad 0 k=5 / BN	1x1x96 / st. 1 / pad 0 k=5 / BN
MMLP-1-2	1x1x48 / st. 1 / pad 0 k=5 / BN	1x1x96 / st. 1 / pad 0 k=5 / BN	1x1x96 / st. 1 / pad 0 k=5 / BN
	3x3 avg. pool / st.2 dropout 0.5	3x3 avg. pool / st.2 dropout 0.5	3x3 avg. pool / st.2 dropout 0.5
Conv-2	5x5x128 / st. 1 / pad 2 BN	5x5x192 / st. 1 / pad 2 BN	5x5x108 / st. 1 / pad 2 BN
MMLP-2-1	1x1x96 / st. 1 / pad 0 k=5 / BN	1x1x192 / st. 1 / pad 0 k=5 / BN	1x1x192 / st. 1 / pad 0 k=5 / BN
MMLP-2-2	1x1x48 / st. 1 / pad 0 k=5 / BN	1x1x192 / st. 1 / pad 0 k=5 / BN	1x1x192 / st. 1 / pad 0 k=5 / BN
	3x3 avg. pool / st.2 dropout 0.5	3x3 avg. pool / st.2 dropout 0.5	3x3 avg. pool / st.2 dropout 0.5
Conv-3	3x3x128 / st. 1 / pad 1 BN	3x3x192 / st. 1 / pad 1 BN	3x3x108 / st. 1 / pad 1 BN
MMLP-3-1	1x1x96 / st. 1 / pad 0 k=5 / BN	1x1x192 / st. 1 / pad 0 k=5 / BN	1x1x192 / st. 1 / pad 0 k=5 / BN
MMLP-3-2	1x1x10 / st. 1 / pad 0 k=5 / BN	1x1x10(100) / st. 1 / pad 0 k=5 / BN	1x1x10 / st. 1 / pad 0 k=5 / BN
	7x7 avg. pool	8x8 avg. pool	8x8 avg. pool

in the third column of Table 1) having the same number of parameters with the NIN method (0.97 million parameters), and obtained the test error of 8.39%.

3.3. CIFAR-100

The CIFAR-100 dataset (Krizhevsky & Hinton, 2009) is the same size and format as the CIFAR-10; however, it contains 100 classes. Thus, the number of images in each class is only one tenth of that of CIFAR-10. As a result, this dataset is far more challenging. The model was trained for 341 epochs (15 hours on a GeForce Titan X). The middle column of Table 1 indicates that our model has 1.69 million parameters. This resulted in an error rate of 28.86% without data augmentation, which represents the state-of-the-art performance. Table 4 presents a summary of the best results obtained in previous works and the current work.

Table 2. Comparison of test errors on MNIST without data augmentation, in which k denotes the number of maxout pieces.

Method	# params	Error (%)
NIN (Lin et al., 2014)	0.35 M	0.45
DSN (Lee et al., 2015)	0.35 M	0.39
RCNN-96 (Liang & Hu, 2015)	0.67 M	0.31
MIN (k=5)	0.45 M	**0.24**

Table 3. Comparison of test errors on CIFAR-10 dataset. * denote that the model has same number of parameters with NIN.

Method	# params	Error (%)
No data augmentation		
NIN (Lin et al., 2014)	0.97 M	10.41
DSN (Lee et al., 2015)	0.97 M	9.69
RCNN-160 (Liang & Hu, 2015)	1.86 M	8.69
MIN (k=5)*	0.95 M	8.39
MIN (k=5)	1.60 M	**7.85**
Data augmentation		
NIN (Lin et al., 2014)	0.97 M	8.81
DSN (Lee et al., 2015)	0.97 M	8.22
RCNN-160 (Liang & Hu, 2015)	1.86 M	7.09
MIN (k=5)	1.60 M	6.75
Fractional MP (1 test) (Graham, 2014)	$>$ 5 M	**4.5**

3.4. SVHN

The SVHN dataset consists of color images of house numbers (32 $\times$ 32 pixels) collected by Google Street View. There are 73,257 and 531,131 digits in the training and additional sets, respectively. In accordance with previous works (Goodfellow et al., 2013), we selected 400 samples per class from the training set and 200 samples per class from the additional set for validation. The remaining 598,388 images were used for training. Moreover, there are 26,032 digits in the test set. We preprocessed the dataset using local contrast normalization, in accordance with the method outlined by Goodfellow et al. (Goodfellow et al., 2013). For this dataset, the initial learning rate was set to be 0.1. The model was trained for 27 epochs (14 hours on a GeForce Titan X). The middle column of Table 1 indicates that our model has 1.6 million parameters. Without data augmentation, we achieved a test error rate of 1.81%, which is comparable to the best result obtained in previous works. Table 5 presents a comparison of our test results with those obtained in recent studies.

Table 4. Comparison of test errors on CIFAR-100 dataset

Method	# params	Error (%)
NIN (Lin et al., 2014)	0.98 M	35.68
DSN (Lee et al., 2015)	0.98 M	34.57
RCNN-160 (Liang & Hu, 2015)	1.87 M	31.75
Fractional MP (1 test) (Graham, 2014)	> 5 M	31.2
MIN (k=5)	1.69 M	**28.86**

Table 5. Comparison of test errors on SVHN. Note that Dropconnet (Wan et al., 2013) uses data augmentation and multiple model voting

Method	# params	Error (%)
NIN (Lin et al., 2014)	1.98 M	2.35
Dropconnect (Wan et al., 2013)	-	1.94
DSN (Lee et al., 2015)	1.98 M	1.92
MIN (k=5)	1.60 M	1.81
RCNN-192 (Liang & Hu, 2015)	2.67 M	**1.77**

3.5. Model capacity

We tested the proposed method on CIFAR-10 dataset using various numbers of maxout pieces. The left panel of Figure 2 illustrates how increasing the number of maxout pieces can improve the performance of our method, by which point the MIM model has already reached saturation. This figure also shows the saturation of maxout units due to the growing number of maxout pieces without batch normalization. We also compared the results of our method with those of batch normalized NIN using the same number of parameters. The right panel of Figure 2 illustrates the training and test error curves under three conditions: MIN with the same number of hidden units (7.85%, final test error), MIN with the same number of parameters (8.39%, final test error), and the batch normalized NIN method (8.94%, final test error). Using the same number of parameters, the proposed method performs better than NIN.

3.6. Regularization of average pooling in MIN

Most of the previous methods used max pooling for down sampling. Max pooling extracts the features within local patches that are the most representative of the class. In this study, the MIN block is able to abstract representative information from every local patch such that more discriminable information is embedded in the feature map. Thus, we are able to use spatial average pooling in each pooling layer to aggregate local spatial information. We compared the results using average pooling in the first two pooling layers with those using max pooling, whereas the last pooling layer was fixed to global average pooling. Figure 3 presents the training and test error curves associated with different pooling methods using the NIN and MIN methods. The use of average pooling in all pooling layers was shown particularly effective with batch normalized NIN architecture. Both batch normalized NIN and MIN methods performed better with average pooling than with max pooling. But the original NIN method performed better with max pooling than with average pooling. The reason is that batch normalization not only can reduce the internal covariance shift but also can regularize models. The features in the same channel are more stable and thus can benefit average pooling.

3.7. Visualization of learned features

Average pooling was applied in all pooling layers to facilitate the abstraction of input images. In (Zhou et al., 2015), they showed that global average pooling can facilitate weakly-supervised object localization. We followed (Zhou et al., 2015) to extract feature maps from models trained using CIFAR-10 to illustrate these effects. Figure 4 presents examplar images and their corresponding feature maps, which were selected from the CIFAR-10 test set. For each method, the first column illustrates the selected feature maps related to the objects per se, whereas the second column shows the selected feature maps for the background, and the last column show the corresponding class activation map (CAM). Note that only the top 50% of the data are shown in this figure while only top 10% are represented in CAM. The learned feature maps produced using the proposed MIN method appear to be more intuitive than the NIN method when dealing with both foreground and background. Compared to NIN, moreover, the MIN method revealed more obviously class-specific units, showing large activation for a specific class. This finding demonstrates the effectiveness of MIN and its potential for object segmentation.

Figure 5 presents examplar images selected from SVHN test set and their corresponding feature maps extracted in the last convolutional layer by using the MIN and NIN models. Only the top 10% of the data are presented. Note that the NIN model in this experiment had the same number of hidden units as the MIN model and achieved the test error of 2.36%. One of the major difficulties in the classification on SVHN dataset is that there are a large portion of images containing distractors, which may confuse the model during training. After all, the distractors are also digits and should be recognized by the model during testing as well as the targeted digit in the center. Therefore, the model should recognize the distractors as the runners-up, besides classifying the targeted digit as the first candidate. In Figure 5, we presented the images containing targeted digits from 0

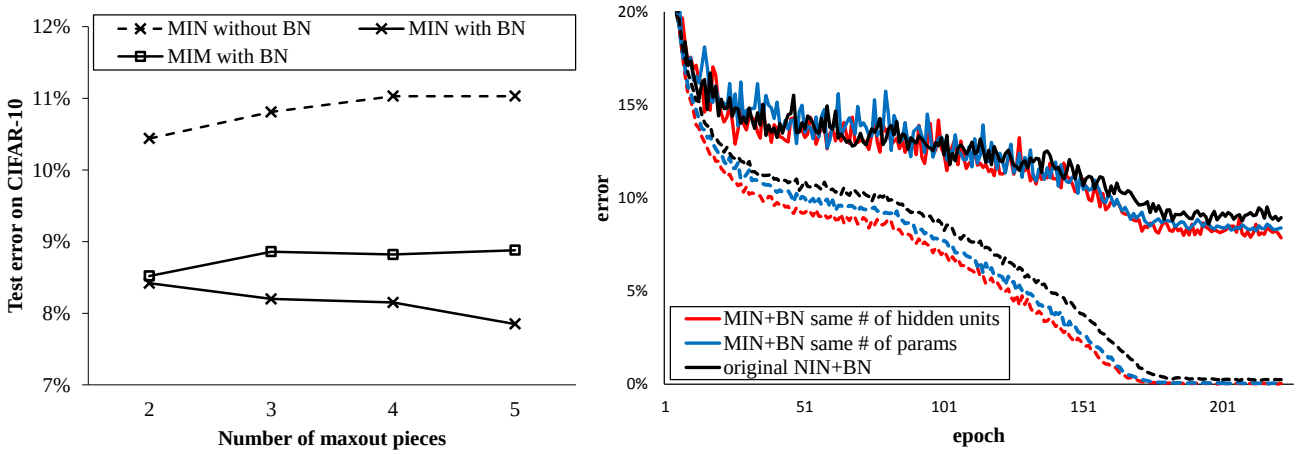


Figure 2. **Left)** Performance related to the number of maxout pieces. We fixed the hyper-parameters when training the MIN model with different maxout pieces. Our method dramatically reduces test error of CIFAR-10 dataset with increasing the number of maxout pieces. **Right)** The training (dotted lines) and test (solid lines) curves of MIN and NIN with batch normalization. With the same number of parameters, our method (8.39% test error) performs better than the batch normalized NIN method (8.94% test error). Errors greater than 20% are not shown here.

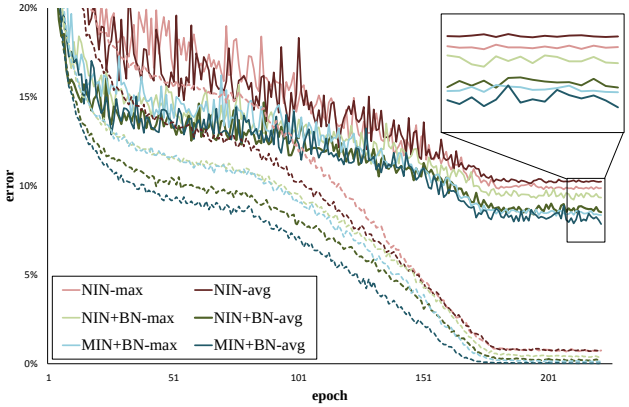


Figure 3. Comparison of training (dotted lines) and test (solid lines) errors on CIFAR-10 dataset without data augmentation using max/average pooling in the first two pooling layers using three models: original NIN, batch normalized NIN, and MIN. Errors great than 20% are not shown here.

to 9 and distractors on the side and highlighted the first and second candidates of the output determined by the softmax layer. These results show that the proposed approach is able to recognize distractors in input images with high accuracy. To demonstrate the localizability of the MIN method, some images selected from SVHN test set as well as their corresponding CAMs are illustrated in Figure 6. We showed that the proposed MIN is better than NIN in the localization of the discriminative image regions.

This indicates that the MIN can robustly preserve information of each category because of the pathway encoding in maxout MLP and spatial average pooling. When convolutional filters slide onto the distractor, the MIN model can extract features of the distractor along its own pathway. The MIN model downscale the feature maps by using spatial average pooling and this pooling method keeps all information of a local patch, whereas max pooling only passes the maximal part. These results suggest the possibility of applying the MIN method to multiple object recognition using a more comprehensive image dataset, such as ImageNet.

4. Conclusions

This paper presents a novel deep architecture, MIN. A MIN block, consisting of a convolutional layer and a two-layer maxout MLP, is used to convolve the input and average pooling is applied in all pooling layers. The proposed method outperforms other methods because of the following improvements: the MIN block facilitates the information abstraction of local patches, batch normalization prevents covariate shift, and average pooling acts as a spatial regularizer tolerating changes of object positions. Our experiments showed that the MIN method achieves improved or comparable performance on the MNIST, CIFAR-10, CIFAR-100, and SVHN datasets. Moreover, the extracted feature maps demonstrate the efficacy of categorical representation by using the MIN method, as well as its potential to multiple object recognition. Source code of the proposed MIN method can be found on GitHub.

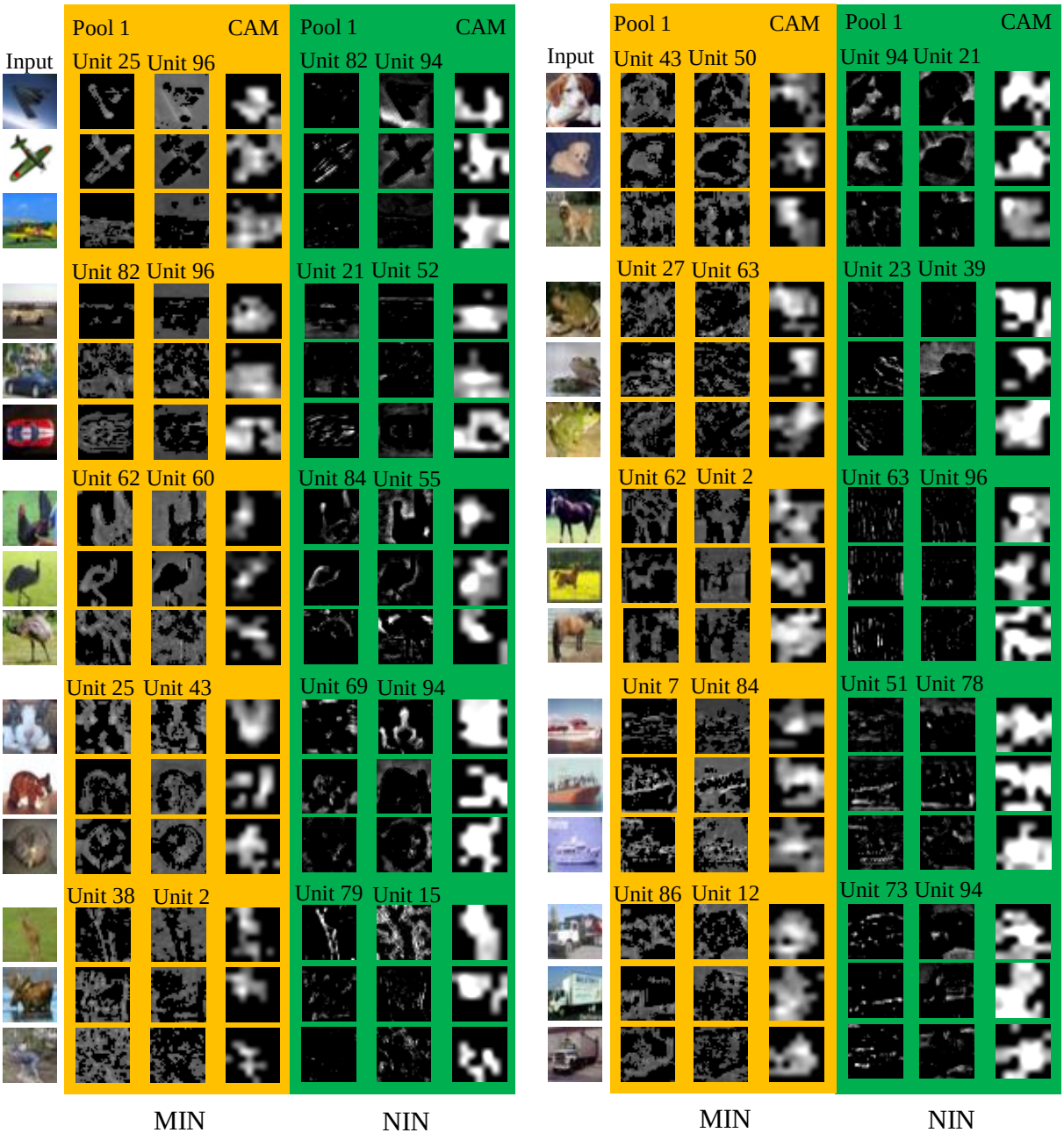
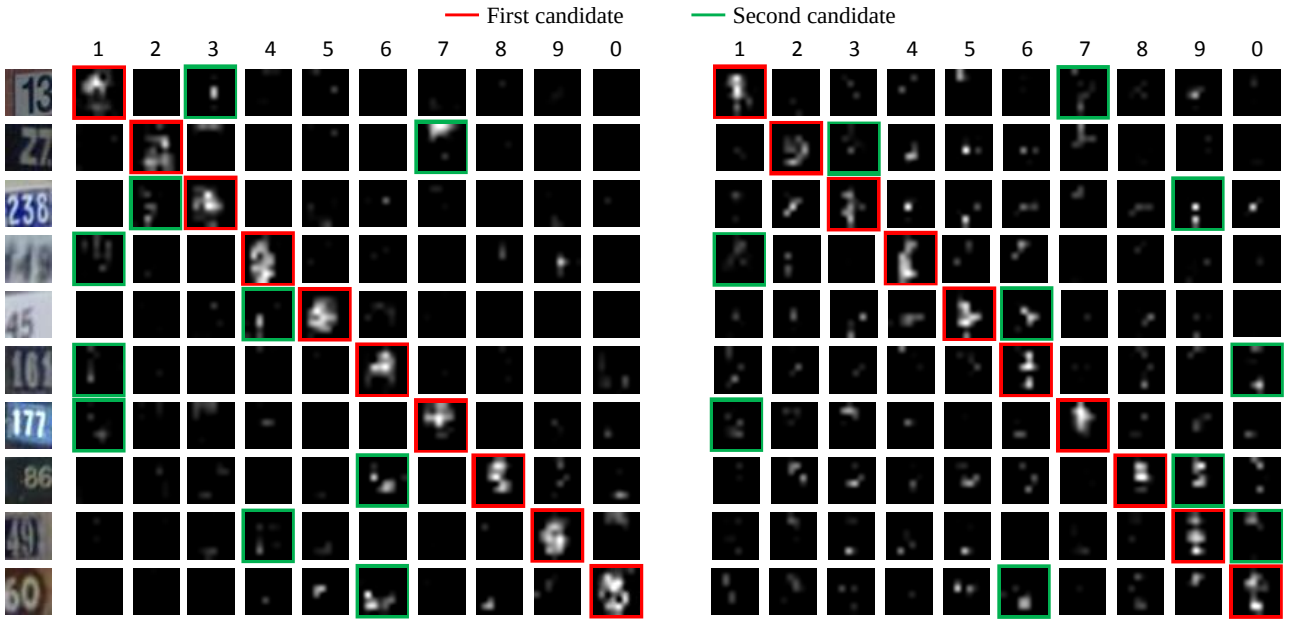


Figure 4. Visualization of learned feature maps before the first pooling layer obtained using the MIN and NIN methods. Those two method were both batch normalized. Only the top 50% of the data are presented. Compared to NIN, the MIN method revealed more obviously class-specific units, showing large activation for a specific class.



(a) Batch normalized MIN (b) Batch normalized NIN

Figure 5. Visualization of the learned feature maps before the global average pooling layer obtained using the MIN (1.81% test error) and NIN (2.36% test error) methods in our training. Only the top 10% of the data are presented. The first and second candidates of the output are highlighted in red and green boxes.

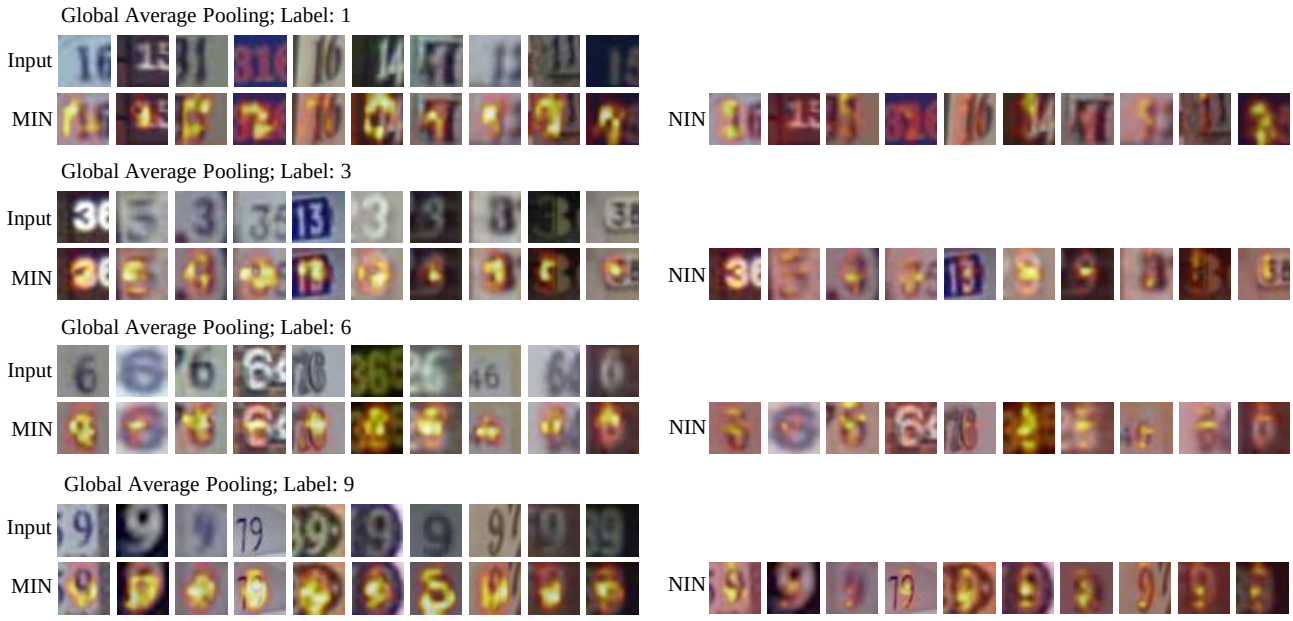


Figure 6. The potential of object detection by using the MIN method. We illustrated the results of global average pooling layer, as well as their corresponding class labels. Only the top 10% of the data are presented. These results demonstrate the possibility of applying the proposed MIN method to multiple object detection.

References

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In *CVPR09*, pp. 248–255. IEEE, 2009.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Goodfellow, Ian J., Warde-Farley, David, Mirza, Mehdi, Courville, Aaron C., and Bengio, Yoshua. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, volume 28 of *JMLR Proceedings*, pp. 1319–1327. JMLR.org, 2013. URL http://dblp.uni-trier.de/db/conf/icml/icml2013.html#GoodfellowWMCB13.

Graham, Benjamin. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

Hinton, Geoffrey E., Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37 of *JMLR Proceedings*, pp. 448–456. JMLR.org, 2015. URL http://dblp.uni-trier.de/db/conf/icml/icml2015.html#IoffeS15.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images, 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, Chen-Yu, Xie, Saining, Gallagher, Patrick, Zhang, Zhengyou, and Tu, Zhuowen. Deeply-supervised nets. *In Proceedings of AISTATS 2015*, 2015.

Liang, Ming and Hu, Xiaolin. Recurrent convolutional neural network for object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. *International Conference on Learning Representations*, abs/1312.4400, 2014. URL http://arxiv.org/abs/1312.4400.

Maas, Andrew L, Hannun, Awni Y, and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, pp. 807–814, 2010.

Netzer, Yuval, Wang, Tao, Coates, Adam, Bissacco, Alessandro, Wu, Bo, and Ng, Andrew Y. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 5. Granada, Spain, 2011.

Shimodaira, Hidetoshi. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90 (2):227–244, 2000.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Vedaldi, Andrea and Lenc, Karel. Matconvnet-convolutional neural networks for matlab. *arXiv preprint arXiv:1412.4564*, 2014.

Wan, Li, Zeiler, Matthew, Zhang, Sixin, Cun, Yann L, and Fergus, Rob. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, pp. 1058–1066, 2013.

Wang, Qi and JaJa, Joseph. From maxout to channel-out: Encoding information on sparse pathways. In *Artificial Neural Networks and Machine Learning–ICANN 2014*, pp. 273–280. Springer, 2014.

Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pp. 818–833. Springer, 2014.

Zhou, Bolei, Khosla, Aditya, Lapedriza, Agata, Oliva, Aude, and Torralba, Antonio. Learning deep features for discriminative localization. *arXiv preprint arXiv:1512.04150*, 2015.