Mixed Pooling for Convolutional Neural Networks

Dingjun Yu, Hanli Wang*, Peiqiu Chen, and Zhihua Wei

Key Laboratory of Embedded System and Service Computing, Ministry of Education,
Tongji University, Shanghai, P.R. China
hanliwang@tongji.edu.cn

Abstract. Convolutional Neural Network (CNN) is a biologically inspired trainable architecture that can learn invariant features for a number of applications. In general, CNNs consist of alternating convolutional layers, non-linearity layers and feature pooling layers. In this work, a novel feature pooling method, named as mixed pooling, is proposed to regularize CNNs, which replaces the deterministic pooling operations with a stochastic procedure by randomly using the conventional max pooling and average pooling methods. The advantage of the proposed mixed pooling method lies in its wonderful ability to address the over-fitting problem encountered by CNN generation. Experimental results on three benchmark image classification datasets demonstrate that the proposed mixed pooling method is superior to max pooling, average pooling and some other state-of-the-art works known in the literature.

Keywords: Convolutional neural network, pooling, regularization, model average, over-fitting.

1 Introduction

Since its first introduction in the early 1980's [1], the Convolutional Neural Network (CNN) has demonstrated excellent performances for a number of applications such as hand-written digit recognition [2], face recognition [3], etc. With the advances of artificial intelligence, recent years have witnessed the growing popularity of deep learning with CNNs on more complicated visual perception tasks.

In [4], Fan *et al.* treat human tracking as a learning problem of estimating the location and the scale of objects and employ CNNs to reach this learning purpose. Cireşan *et al.* [5] propose an architecture of multi-column CNNs which

* Corresponding author (H. Wang, E-mail: hanliwang@tongji.edu.cn). This work was supported in part by the National Natural Science Foundation of China under Grants 61102059 and 61472281, the "Shu Guang" project of Shanghai Municipal Education Commission and Shanghai Education Development Foundation under Grant 12SG23, the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, the Fundamental Research Funds for the Central Universities under Grants 0800219158, 0800219270 and 1700219104, and the National Basic Research Program (973 Program) of China under Grant 2010CB328101.

D. Miao et al. (Eds.): RSKT 2014, LNAI 8818, pp. 364–375, 2014.
DOI: 10.1007/978-3-319-11740-9_34

can be accelerated by Graphics Processing Unit (GPU) for image classification and amazing performances are achieved on a number of benchmark datasets. In [6], a 3D CNN model is designed for human action recognition, in which both the spatial and temporal features are mined by performing 3D convolutions. krizhevsky *et al.* [7] train a very large CNN for the ImageNet visual recognition challenge [8] and achieve an astonishing record-breaking performance in 2012.

Despite the aforementioned encouraging progresses, there are still several problems encountered by CNNs such as the over-fitting problem due to the high capacity of CNNs. In order to address this issue, several regularization techniques have been proposed, such as weight decay, weight tying and augmentation of training sets [9]. These regularization methods allow the training of larger capacity models than would otherwise be possible, which are able to achieve superior test performances as compared with smaller un-regularized models [10].

Another promising regularization approach is Dropout which is proposed by Hinton *et al.* [11]. The idea of Dropout is to stochastically set half the activations in a hidden layer to zeros for each training sample. By doing this, the hidden units can not co-adapt to each other, and they must learn a better representation for the input in order to generalize well. Dropout acts like a form of model averaging over all possible instantiations of the model prototype, and it is shown to deliver significant gains in performance in a number of applications.

However, the shortcoming of Dropout is that it can not be generally employed for several kinds of CNN layers, such as the convolutional layer, non-linearity layer and feature pooling layer. To overcome this defect, a generalization of Dropout, called DropConnect, is proposed in [12]. Instead of randomly selecting activations within the network, DropConnect sets a randomly selected subset of weights to zeros. As compared to Dropout, better performances have been achieved by DropConnect in certain cases. In [10], another type of regularization for convolutional layers, named stochastic pooling, is proposed to enable the training of larger models for weakening over-fitting. The key idea of stochastic pooling is to make the pooling process in each convolutional layer a stochastic process based on multinomial distribution.

In this work, similar to [10], a novel type of pooling method, termed as mixed pooling, is proposed in order to boost the regularization performance for training larger CNN models. Inspired by Dropout (that randomly sets half the activations to zeros), the proposed mixed pooling method replaces the conventional deterministic pooling operations with a stochastic procedure, randomly employing the max pooling and average pooling methods during the training of CNNs. Such a stochastic nature of the proposed mixed pooling method helps prevent over-fitting to some extent. Experiments are performed to verify the superiority of the proposed mixed pooling method over the traditional max pooling and average pooling methods.

The rest of this paper is organized as follows. Section 2 provides a background review of CNNs. The proposed mixed pooling method is introduced in Section 3. In Section 4, the comparative experimental results are presented. Finally, Section 5 concludes this paper.

2 Review of Convolutional Neural Networks

A brief review of CNNs is presented herein which is useful to elicit the proposed mixed pooling method. In general, CNNs are representatives of the multi-stage Hubel-Wiesel architecture [13], which extract local features at a high resolution and successively combine these into more complex features at lower resolutions. The loss of spatial information is compensated by an increasing number of feature maps in higher layers.

A powerful CNN is composed of several feature extraction stages, and each stage consists of a convolutional layer, a non-linear transformation layer and a feature pooling layer. The convolutional layer takes inner product of the linear filter and the underlying receptive field followed by a nonlinear activation function at every local portion of the input. Then, the non-linear transformation layer performs normalization among nearby feature maps. Finally, the feature pooling layer combines local neighborhoods using an average or maximum operation, aiming to achieve invariance to small distortions. An example of a two-stage CNN with the aforementioned three layers is shown in Fig. 1 for illustration.

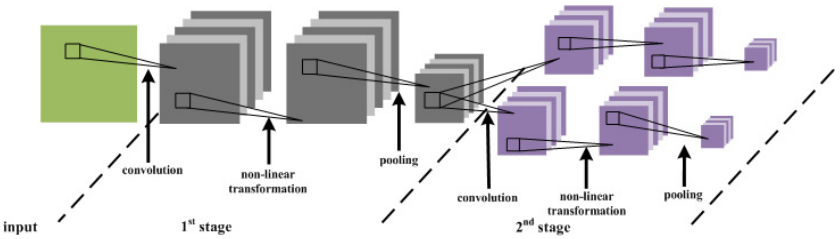

Fig. 1. An example of a two-stage CNN. An input image is passed through a convolutional layer, followed by non-linear transformation layer and pooling layer to extract low-level features in the first stage. Then, these three layers are applied again in the second stage to extract high-level features.

2.1 Convolutional Layer

The aim of the convolutional layer is to extract patterns found within local regions of the input images that are quite common in natural images [10]. Generally speaking, the convolutional layer generates feature maps by linear convolutional filters followed by nonlinear activation functions, such as ReLU [14], sigmoid, tanh, etc. In this layer, the kth output feature map y_k can be calculated as follows:

$$y_k = f(w_k * x), \tag{1}$$

where x denotes the input image, w_k stands for the convolutional filter associated with the kth feature map, $*$ indicates the 2D convolution operator which is used to calculate the inner product of the filter template at every location in the input image, and $f(\cdot)$ is the nonlinear activation function.

2.2 Non-linear Transformation Layer

It has been shown in [15] that using a rectifying non-linear transformation layer is an effective way to further improve the CNN performance for visual recognition tasks. This layer usually performs local subtractive or divisive operations for normalization, enforcing a kind of local competition between features at the same spatial location in different feature maps. There are usually two kinds of non-linear transformations. One is the local response normalization [11], which yields the normalized output y_{kij} at the position (i, j) in feature map k as

$$y_{kij} = \frac{x_{kij}}{\left(1 + \frac{\alpha}{N} \cdot \sum\limits_{l=k-\frac{N}{2}}^{k+\frac{N}{2}} (x_{lij})^2\right)^{\beta}}, \tag{2}$$

where the sum runs over N adjacent feature maps at the same spatial location, and the parameters of α and β can be determined using a validation set.

Another is the local contrast normalization [15] with the normalized output y_{kij} produced with the following formula.

$$y_{kij} = \frac{x_{kij}}{\left(1 + \frac{\alpha}{M_1 \cdot M_2} \sum\limits_{p=i-\frac{M_1}{2}}^{i+\frac{M_1}{2}} \sum\limits_{q=j-\frac{M_2}{2}}^{j+\frac{M_2}{2}} (x_{kpq} - m_{kij})^2\right)^{\beta}}, \tag{3}$$

where the local contrast is computed within a local $M_1 \times M_2$ region with the center at (i, j), and m_{kij} is the mean of all x values within the above $M_1 \times M_2$ region in the kth feature map as computed as

$$m_{kij} = \frac{1}{M_1 \cdot M_2} \cdot \sum_{p=i-\frac{M_1}{2}}^{i+\frac{M_1}{2}} \sum_{q=j-\frac{M_2}{2}}^{j+\frac{M_2}{2}} x_{kpq}. \tag{4}$$

2.3 Feature Pooling Layer

The purpose of pooling is to transform the joint feature representation into a more usable one that preserves important information while discarding irrelevant details. The employment of pooling layer in CNNs aims to achieve invariance to changes in position or lighting conditions, robustness to clutter, and compactness of representation. In general, the pooling layer summarizes the outputs of neighboring groups of neurons in the same kernel map [7]. In the pooling layer, the resolution of the feature maps is reduced by pooling over local neighborhood on the feature maps of the previous layer, thereby enhancing the invariance to distortions on the inputs.

In CNNs, there are two conventional pooling methods, including max pooling and average pooling. The max pooling method selects the largest element in each pooling region as

$$y_{kij} = \max_{(p,q) \in \mathcal{R}_{ij}} x_{kpq}, \tag{5}$$

where y_{kij} is the output of the pooling operator related to the kth feature map, x_{kpq} is the element at (p, q) within the pooling region $\mathcal{R}_{ij}$ which represents a local neighborhood around the position (i, j). Regarding the average pooling method, it takes the arithmetic mean of the elements in each pooling region as

$$y_{kij} = \frac{1}{|\mathcal{R}_{ij}|} \sum_{(p,q)\in\mathcal{R}_{ij}} x_{kpq}, \tag{6}$$

where $|\mathcal{R}_{ij}|$ stands for the size of the pooling region $\mathcal{R}_{ij}$.

3 Proposed Mixed Pooling

3.1 Motivation

As mentioned before, the max pooling and average pooling methods are two popular choices employed by CNNs due to their computational efficiency. For instance, the average pooling method is used in [15] which obtains an excellent image classification accuracy on the Caltech101 dataset. In [7], the max pooling method is successfully applied to train a deep 'convnet' for the ImageNet competition. Although these two kinds of pooling operators can work very well on some datasets, it is still unknown which will work better for addressing a new problem. In another word, it is a kind of empiricism to choose the pooling operator.

On the other hand, both the max pooling and average pooling operators have their own drawbacks. About max pooling, it only considers the maximum element

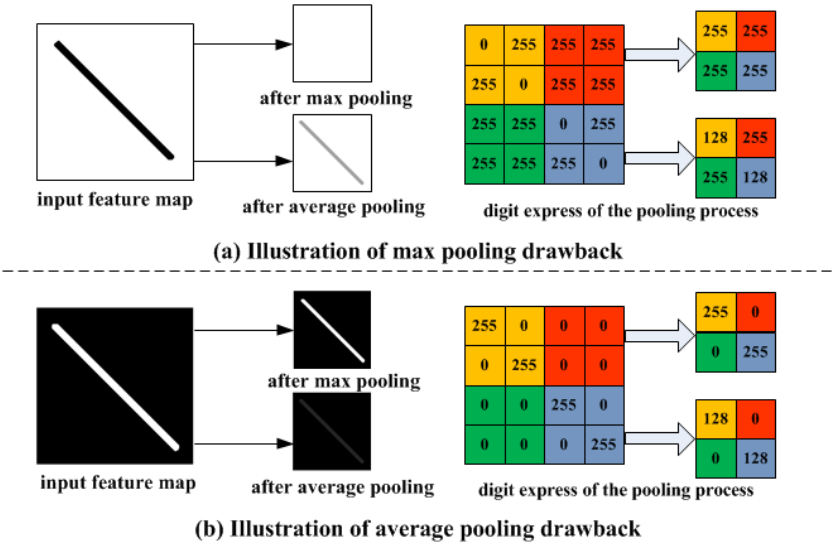

Fig. 2. Toy example illustrating the drawbacks of max pooling and average pooling.

and ignores the others in the pooling region. Sometimes, this will lead to an unacceptable result. For example, if most of the elements in the pooling region are of high magnitudes, the distinguishing feature vanishes after max pooling as shown in Fig. 2(a). Regarding average pooling, it calculates the mean of all the elements within the pooling region. This operator will take all the low magnitudes into consideration and the contrast of the new feature map after pooling will be reduced. Even worse, if there are many zero elements, the characteristic of the feature map will be reduced largely, as illustrated in Fig. 2(b).

It is well known that images in the nature world are ever-changing, and it is of high possibility that the defective aspects of max pooling and average pooling (as shown in Fig. 2) will have negative effects in applying pooling layers to CNNs. Therefore, as a solution, we consider to replace the deterministic pooling operation with a stochastic procedure, which randomly employs the local max pooling and average pooling methods when training CNNs. This is the proposed mixed pooling method to be introduced next.

3.2 Pooling Scheme

The proposed mixed pooling is inspired by the random Dropout [11] and DropConnect [12] methods. As mentioned before, when training with Dropout, a randomly selected subset of activations are set to zeros within each layer. While for DropConnect, it instead sets a randomly selected subset of weights within the network to zeros. Both of these two techniques have been proved to be powerful for regularizing neural networks.

In this work, the proposed mixed pooling method generates the pooled output with the following formula.

$$y_{kij} = \lambda \cdot \max_{(p,q)\in\mathcal{R}_{ij}} x_{kpq} + (1-\lambda) \cdot \frac{1}{|\mathcal{R}_{ij}|} \sum_{(p,q)\in\mathcal{R}_{ij}} x_{kpq}, \tag{7}$$

where λ is a random value being either 0 or 1, indicating the choice of using the max pooling or average pooling. In another word, the proposed method changes the pooling regulation scheme in a stochastic manner which will address the problems encountered by max pooling and average pooling to some extent.

3.3 Back Propagation

As usual, CNN layers are trained using the back propagation algorithm. For error propagation and weight adaptation in fully connected layers and convolutional layers, the standard back propagation procedure is employed. For the pooling layer, the procedure is a little bit different. As noted in [2], the pooling layers do not actually do any learning themselves. Instead, they just reduce the dimension of the networks. During forward propagation, an $N \times N$ pooling block is reduced to a single value. Then, this single value acquires an error computed from back propagation. For max pooling, this error is just forwarded to where it comes from because other units in the previous layer's pooling blocks do not contribute

to it. For average pooling, this error is forwarded to the whole pooling block by dividing $N \times N$ as all units in the block affect its value.

In mixed pooling, it is also needed to locate where the error comes from so that it can modify the weights correctly. The proposed mixed pooling randomly apply the max pooling and average pooling during forward propagation. For this reason, the pooling history about the random value λ in Eq. (7) must be recorded during forward propagation. Then, for back propagation, the operation is performed depending on the records. Specifically, if $\lambda = 1$, then the error signals are only propagated to the position of the maximum element in the previous layer; otherwise, the error signals will be equally divided and propagated to the whole pooling region in the previous layer.

3.4 Pooling at Test Time

When the proposed mixed pooling is applied for test, some noises will be introduced into CNNs' predictions, which is also found in [10]. In order to reduce this kind of noise, a statistical pooling method is used. During the training of CNNs, the frequencies of using the max pooling and average pooling methods related to the kth feature map are counted as F^k_{max} and F^k_{avg}. If $F^k_{max} \geq F^k_{avg}$, then the max pooling method is applied in the kth feature map; otherwise, the average pooling method is used. In this sense, the proposed statistical pooling at the test time can be viewed as a form of model averaging.

4 Experimental Results

4.1 Overview

The proposed mixed pooling method is evaluated on three benchmark image classification datasets, including CIFAR-10 [16], CIFAR-100 [16] and the Street View House Number (SVHN) dataset [17], with a selection of images from CIFAR-10 and SVHN as shown in Fig. 3. The proposed method is compared with the max pooling and average pooling methods for demonstrating the performance improvement. In the experiments, the CNNs are generated from the raw RGB values of the image pixels. As a regularizer, the data augmentation technique [18] is applied for CNN training, which is performed by extracting 24×24 sized images as well as their horizontal reflections from the original 32×32 image and then training CNNs on these extracted images. Another regularizer applied in this work is the weight decay technique as used in [7].

In this work, the publicly available cuda-convnet [19] package is used to perform experiments with a single NVIDIA GTX 560TI GPU. Currently, the CNNs are trained using stochastic gradient descent approach with a batch size of 128 images and momentum of 0.9. Therefore, the update rule for weight w is

$$v_{i+1} = 0.9v_i + \epsilon \left\langle \frac{\partial L}{\partial w}|_{w_i} \right\rangle_i , \tag{8}$$

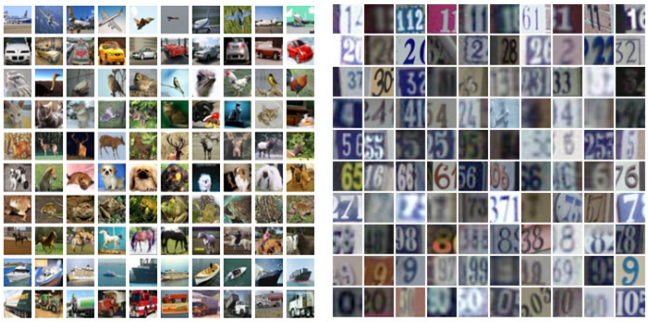

Fig. 3. A selection of images we evaluated. (a) CIFAR-10 [16]. (b) SVHN [17].

$$w_{i+1} = w_i + v_{i+1}, \tag{9}$$

where i is the iteration index, v is the momentum variable, ϵ is the learning rate, and $\left\langle \frac{\partial L}{\partial w}|_{w_i} \right\rangle_i$ is the average over the ith batch of the derivative of the objective with respect to w_i.

4.2 CIFAR-10

CIFAR-10 [16] is a collection of natural color images of 32×32 pixels. It contains 10 classes, each of them having 5,000 samples for training and 1,000 for testing. The CIFAR-10 images are highly varied, and there is no standard viewpoint or scale at which the objects appear. Except for subtracting the mean activity of the training set, the CIFAR-10 images are not preprocessed.

A two-stage CNN model is trained in this work, with each stage consisting of a convolutional layer, a local response normalization layer and a pooling layer. All the convolutional layers have 64 filter banks and use a filter size of 5×5. Local response normalization layers follow the convolutional layers, with $N = 9$, $\alpha = 0.001$ and $\beta = 0.75$ (as used in Eq.(2)), which normalize the output at each location over a subset of neighboring feature maps. This typically helps training by suppressing extremely large outputs allowed by the rectified linear units and helps neighboring features communicate with each other. Additionally, all of the pooling layers that follow local response normalization layers summarize a 3×3 neighborhood and use a stride of 2. Finally, two locally connected layers and a softmax layer are used as classifier at the end of the entire network.

We follow the common experimental protocol for CIFAR-10, which is to choose 50,000 images for training and 10,000 images for testing. The network parameters are selected by minimizing the error on a validation set consisting of the last 10,000 training examples.

The comparative results are shown in Table 1, where the test accuracy results of several state-of-the-art approaches are cited for illustration besides the max pooling, average pooling and mixed pooling methods. From the results, it can

Table 1. Comparative classification performances with various pooling methods on the CIFAR-10 dataset

Method	Training error (%)	Accuracy (%)
3-layer Convnet [11]	-	83.4%
10-layer DNN [5]	-	88.79%
Stochastic pooling [20]	-	84.87%
Max pooling	3.01%	88.64%
Average pooling	4.52%	86.25%
Mixed pooling	6.25%	**89.20%**

be seen that the proposed mixed pooling method is superior to other methods in terms of the test accuracy although it produces larger training errors than that of max pooling and average pooling. This indicates that the proposed mixed pooling outperforms max pooling and average pooling to address the over-fitting problem. As observed from the results, a test accuracy of 89.20% is achieved by the proposed mixed pooling method which is the best result which we are aware of without using Dropout. In addition, the features which are learnt in the first convolutional layer by using different pooling methods are shown in Fig. 4, where it can observed that the features learnt with the proposed mixed pooling method contains more information than that of max pooling and average pooling.

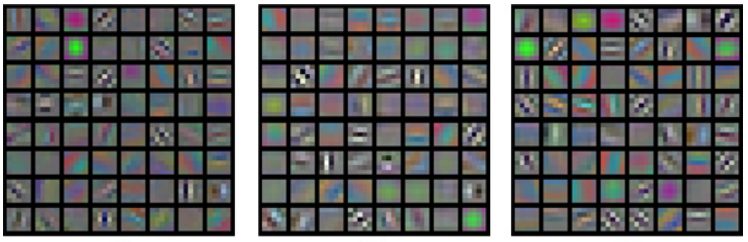

Fig. 4. Visualization of 64 features learnt in the first convolutional layer on the CIFAR-10 dataset. The size of each feature is 5×5×3. (a) Features learnt with max pooling. (b) Features learnt with average pooling. (c) Features learnt with mixed pooling.

4.3 CIFAR-100

The CIFAR-100 dataset [16] is the same in size and format as the CIFAR-10 dataset, but it contains 100 classes. That is to say, each class in CIFAR-100 has 500 images to train and 100 images to test. We preprocess the data just like the way we have done for the CIFAR-10 dataset, and the same CNN structure as used for CIFAR-10 is applied to CIFAR-100. The only difference is that the last softmax layer outputs 100 feature maps. The comparative results are shown

Table 2. Comparative classification performances with various pooling methods on the CIFAR-100 dataset

Method	Training error (%)	Accuracy (%)
Learnable pooling regions [21]	-	56.29%
Stochastic pooling [20]	-	57.49%
Max pooling	5.42%	59.91%
Average pooling	14.61%	55.99%
Mixed pooling	25.71%	**61.93%**

in Table 2, where it can be observed that the proposed mixed pooling method outperforms the other methods in terms of test accuracy.

4.4 SVHN

Finally, we also perform experiments on the SVHN dataset [17]. SVHN consists of images of house numbers collected by Google Street View. There are 73,257 digits in the training set, 26,032 digits in the test set and 531,131 additional examples as an extra training set. We follow [22] to build a validation set which contains 400 samples per class from the training set and 200 samples per class from the extra set. The remaining digits of the training and extra sets are used for training. The local contrast normalization operator is applied in the same way as used in [20]. The comparative results are presented in Table 3, which demonstrate the superiority of the proposed mixed pooling method over the others.

Table 3. Comparative classification performances with various pooling methods on the SVHN dataset

Method	Training error (%)	Accuracy (%)
Lp-pooling Convnet [22]	-	95.10%
64-64-64 Stochastic pooling [20]	-	96.87%
64-64-64 Max pooling	2.03%	96.61%
64-64-64 Average pooling	2.41%	96.14%
64-64-64 Mixed pooling	3.54%	**96.90%**

4.5 Time Performance

To further illustrate the advantage of the proposed mixed pooling method, the time consumption performances are illustrated in Fig. 5 with two testing scenarios evaluated for max pooling, average pooling, stochastic pooling [10] and mixed pooling, where nine epoches are tested. From Fig. 5, it can be seen that the computational complexity of mixed pooling is almost the same as that of average pooling and max pooling, and far lower than that of stochastic pooling.

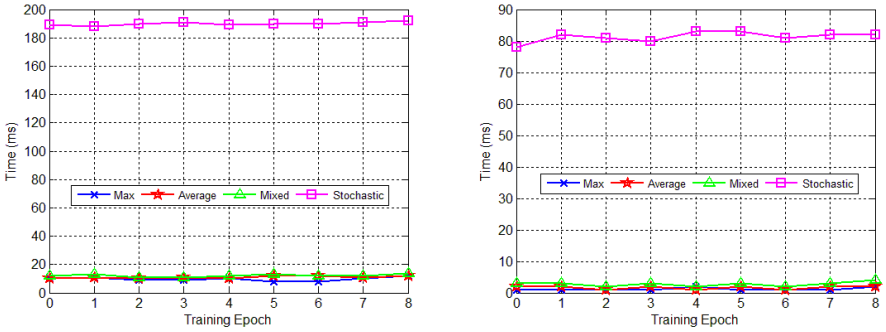

Fig. 5. Time performance comparison among max, average, stochastic and mixed pooling. (a) Time consumption when feature map size is 28×28 and pooling size is 2×2. (b) Time consumption when feature map size is 14×14 and pooling size is 2×2.

5 Conclusion

In this paper, a novel pooling method called mixed pooling is proposed, which can be combined with any other forms of regularization such as weight decay, Dropout, data augmentation, and so on. Comparative experimental results demonstrate that the proposed mixed pooling method is superior to the traditional max pooling and average pooling methods to address the over-fitting problem and improve the classification accuracy. With the proposed method, we achieve the start-of-the-art performances on the CIFAR-10, CIFAR-100 and SVHN datasets as compared with other approaches that do not employ Dropout. Furthermore, the proposed method requires negligible computational overheads and no hyper-parameters to tune, thus can be widely applied to CNNs.

References

1. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36(4), 193–202 (1980)
2. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation 1(4), 541–551 (1989)
3. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: A convolutional neural-network approach. IEEE Transactions on Neural Networks 8(1), 98–113 (1997)
4. Fan, J., Xu, W., Wu, Y., Gong, Y.: Human tracking using convolutional neural networks. IEEE Transactions on Neural Networks 21(10), 1610–1623 (2010)
5. Cireşan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: CVPR, pp. 3642–3649 (2012)

6. Ji, S., Xu, W., Yang, M., Yu, K.: 3D convolutional neural networks for human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(1), 221–231 (2013)
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS, vol. 1 (2012)
8. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: CVPR, pp. 248–255 (2012)
9. Montavon, G., Orr, G.B., Müller, K.R. (eds.): Neural networks: tricks of the trade, 2nd edn. Spinger, San Francisco (2012)
10. Zeiler, M.D.: Hierarchical convolutional deep learning in computer vision. PhD thesis, ch. 6, New York University (2014)
11. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)
12. Wan, L., Zeiler, M.D., Zhang, S., LeCun, Y., Fergus, R.: Regularization of neural networks using DropConnect. In: ICML, pp. 1058–1066 (2013)
13. LeCun, Y., Kavukcuoglu, K., Farabet, C.: Convolutional networks and applications in vision. In: ISCAS, pp. 253–256 (2010)
14. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML, pp. 807–814 (2010)
15. Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: ICCV, pp. 2146–2153 (2009)
16. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto (2009)
17. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning, vol. 2011 (2011)
18. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: ICDAR, pp. 958–962 (2003)
19. Krizhevsky, A.: cuda-convnet., `http://code.google.com/p/cuda-convnet/`
20. Zeiler, M.D., Fergus, R.: Stochastic pooling for regularization of deep convolutional neural networks. arXiv preprint arXiv:1301.3557 (2013)
21. Malinowski, M., Fritz, M.: Learnable pooling regions for image classification. arXiv preprint arXiv:1301.3516 (2013)
22. Sermanet, P., Chintala, S., LeCun, Y.: Convolutional neural networks applied to house numbers digit classification. In: ICPR, pp. 3288–3291 (2012)