2015 IEEE International Conference on Systems, Man, and Cybernetics

Correlative Filters for Convolutional Neural Networks

Peiqiu Chen*†, Hanli Wang*†★, Jun Wu*†
*Department of Computer Science and Technology, Tongji University, Shanghai, China
† Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, China
E-mail: 14payenjoe@tongji.edu.cn, hanliwang@tongji.edu.cn, wujun@tongji.edu.cn

***Abstract*—This paper introduces a regularization method called Correlative Filter (CF) for Convolutional Neural Network (CNN), which takes advantage of the relevance between the convolutional kernels belonging to the same convolutional layer. During the process of training with the proposed CF method, several pairs of filters are designed in a manner of randomness to contain opposite weights in low-level layers. Regarding higher-level layers where synthetical features are processed, the relation between correlative filters is explored as translation of various directions. The proposed CF method attempts to optimize the inner structure of convolutional layers and it can work jointly with other regularization techniques, such as stochastic pooling, Dropout, etc. The experimental results on the competitive image classification benchmark dataset CIFAR-10 demonstrates the performance of the proposed CF method, additionally, it is also verified that the proposed CF method is wonderful to be employed to enhance several state-of-the-art regularization models.**

***Index Terms*—Correlative filter, convolutional neural network, convolutional kernel, image classification, regularization.**

I. Introduction

Convolutional Neural Network (CNN) [1], which is composed of several convolutional layers to extract various features, has attracted great attentions for its significant progress on visual tasks, such as image recognition [2] and video classification [3]. The neurons of a CNN are organized as feature maps, and the trainable weights for connecting neurons are called filters since the convolution occurs in each layer indicting the spatial filtering of feature maps in the previous layer. Through the inner product with the filters, each neuron in CNN only receives inputs from a set of units located in a small neighborhood of the previous layer. This is consistent with the discovery of locally-sensitive and orientation-selective neurons in the cat's visual system [4].

As stated in [4], the filters make the main effort of extracting visual features such as oriented edges, end-points and corners, which would gather in deeper layers to describe more complex visual features. Mostly, CNNs are trained with a stochastic gradient descent procedure in the sense that every weight is modulated by the partial derivative of a cost function with respect to itself, hence there is no explicit logical relation between the filters belonging to the same convolutional layer. Nevertheless, it is observed that the neurons of primate visual cortex could work cooperatively in the same area of cortex [5]. This phenomenon inspires us that the filters learnt in the same level of layer would have much more in common than those from different neural layers.

In this work, a novel regularization method called Correlative Filter (CF) is proposed to model the presupposed relation between filters in the same neural layer. Loosely speaking, during the whole training process of CNNs, the proposed CF method enforces that several filters should be the specific transformations of their original correlative filters. As the filters in shallow layers focus on generating simple elementary visual features while more complicated features are extracted in deeper layers, the proposed CF method abides by this guideline and operates differently according to the depth of the layers where applied. When utilized closely to the input layer, the proposed CF method would produce several pairs of filters in the way that they are opposite to each other. If the proposed CF method is applied to convolutional layers of deep level, the relation between the correlative filters becomes translational.

By employing the proposed CF method, the CNN structure can learn a more generalized model which preforms better on the test set, since these related filters follow the predetermined rules to process feature maps. Moreover, the proposed CF method only regularizes the filters of feature exactors, therefore it could also be applied with stochastic regularization methods to prevent over-fitting, such as the technique of Dropout [6], stochastic pooling [7], etc.

The rest of this paper is organized as follows. Section II reviews several related works about CNN regularization. Then, a brief introduction of CNN is given in Section III, which is helpful to elicit the proposed CF method that is detailed in Section IV. The experimental results are presented in Section V to verify the proposed CF method. At last, Section VI concludes this paper.

II. Related Work

Due to the excellent performance of CNNs, a plenty of research efforts are made to exploit the potentialities of such multi-layer models, particularly for image recognition tasks with big data set [2]. Generally speaking, the CNN based methods could be roughly divided into the following three

★Corresponding author (H. Wang). This work was supported in part by the National Natural Science Foundation of China under Grant 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 (No. GZ2015005), and the Fundamental Research Funds for the Central Universities under Grant 0800219270.

categories.

The first category aims at optimizing the training process. In deeply-supervised nets [8], the feature quality of hidden layers is calculated by a Support Vector Machine (SVM) which is added to the cost function and finally influences the update of weight/filter in CNNs. In another way, the Bayesian optimization method [9] could also be applied to find the most appropriate parameters for a CNN, *e.g.*, the learning rate and scale.

The second type consists of algorithms which utilize stochastic regularization techniques. The Dropout technique [6] is first proposed in this field. During every presentation of training, each neuron in a Dropout layer is randomly omitted from the network with a fixed probability of 0.5 [6]. Dropout is usually used with full connection layers just before the softmax layer, so that the whole model could be viewed as an average of several sub-networks which have half the neurons of a dropout layer and share the same convolutional layers ahead. Similarly, the DropConnect method [10] randomly drops neuron connections or weights instead of output units. In [7], another type of regularization for convolutional layers, named stochastic pooling, is proposed to enable the training of larger models to overcome over-fitting. The idea of stochastic pooling is to make the pooling process in each convolutional layer a stochastic process based on multinomial distribution.

The methods belonging to the last category seek for a better model structure based on CNNs. Inspired by microcolumns of neurons in the cerebral cortex, several Deep Neural Network (DNN) columns are combined to form a Multi-Column DNN (MCDNN) [11]. After each DNN is trained, the final predictions are the average of all DNN outputs. Another model called Recursive Convolutional Network (RCN) [12] is designed to consider convolutional layers whose weights are tied with higher layers. In order to enhance the model discriminability of local patches, the feature maps of Network In Network (NIN) [13] are obtained by sliding the micro networks, instead of simple linear filters, over the input in a similar manner to CNNs.

The proposed CF method introduced in this work is also a regularization technique for CNN structure refinement, which makes static connections between filters of the classic convolutional layers. As it only takes advantage of the inner information of each layer, it would also fit in models that use convolutional layers as components. As a consequence, the proposed CF method could be combined with other approaches such as Dropout and stochastic pooling to generate a better model.

III. Convolutional Neural Networks

In general, CNNs are representatives of the multi-stage Hubel-Wiesel architecture [14], which extract local features at a high resolution and successively combine these into more complex features at lower resolutions. At the top of a CNN, a softmax regression layer is applied to make prediction of input samples. In the convolution layer, neurons are organized as feature maps for convolution to extract local spatial information. Taking the square feature map that has only one filter for example, the formulation of executing convolution, which employs a half kernel width padding to retain information on the border, can be described as follows.

$$y(m,n) = \sum_{R} w_{i,j} \cdot I_A(m,n) \cdot x(m+i, n+j), \qquad (1)$$

where $y(m,n)$ is the pixel of the convolved output feature map y, x stands for the input patch of the previous layer, $w_{i,j}$ is the kernel weight at the location (i,j), and I is an indicator function. Regarding the region sets of R and A, they are given as

$$R = \{(i,j) | \lfloor k/2 \rfloor - (k-1) \le i,j \le \lfloor k/2 \rfloor\}, \qquad (2)$$

$$A = \{(m,n) | 0 \le m+i, n+j \le L-1\}, \qquad (3)$$

where k is the width of the convolution kernel and L is the width of the input patch. In practice, one feature map is usually the sum of multiple convolved results and each output neuron is processed by a nonlinear activation function, *e.g.*, the sigmoid function.

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, hence enhancing the invariance to distortions on the inputs. Since there are spatial redundancies about the data patches, the pooling operation will reduce the dimension of feature maps effectively while reserving enough information for the purpose of classification. In CNNs, there are two conventional pooling methods, including average pooling and max pooling. The average pooling method takes the arithmetic mean of the elements in each pooling region as the output, while the max pooling method chooses the largest value in the pooled patch as the single activation. As compared to average pooling, the max pooling is a kind of winner-take-all method that helps CNN converge to the best state faster while leading to over-fitting simultaneously. The stochastic pooling method [7], however, gives each neuron of the inputs a probability of being chosen as the only output pixel. In such a way, the over-fitting problem can be addressed to some extent.

Generally speaking, a CNN can be trained with the back propagation algorithm using the stochastic gradient descent. During each iteration of training, a cost function is defined to minimize the average sum-of-square error of predicted labels across all the training samples. After the i^{th} iteration, the j^{th} weight could be updated as

$$\triangle w_{j,i+1} = m \cdot \triangle w_{j,i} + \lambda \cdot E(\frac{\partial J}{\partial w_{j,i}}), \qquad (4)$$

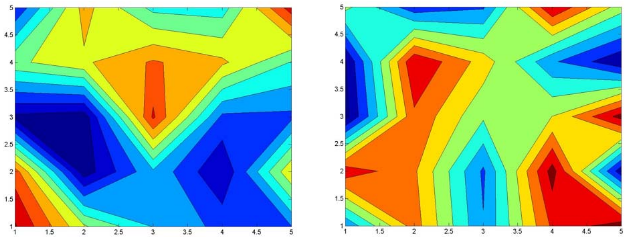
$$w_{j,i+1} = w_{j,i} + \triangle w_{j,i+1}, \qquad (5)$$

where m is the momentum parameter, λ is the learning rare and $E(\frac{\partial J}{\partial w_{j,i}})$ is the average of the partial derivative of the cost function J with respect to the j^{th} weight over the samples trained in the i^{th} iteration.

IV. Proposed Correlative Filters

A. *Motivation*

In the primate subcortical vision system, there are luminance sensitive cells with a center-surround receptive field [5]. These cells can be classified into two types: (1) the on-center/off-surround cells, which are sensitive to a bright spot on a dark background, and (2) the off-center/on-surround cells, which are sensitive to the inverse pattern. Located in the same early stage of visual processing, these two types of cells emphasize to spatial change in luminance. This kind of opposite relation between filters also exists in CNNs. In order to illustrate this phenomenon, a simple CNN model is trained which consists of two convolutional layers on the dataset of Caltech101. After this CNN model is generated, it is observed that some pairs of filters are just the opposite to the others as shown in Fig. 1, even though these filters are randomly initialized completely independent from each other.


(a) Two filter examples which are randomly initialized before training

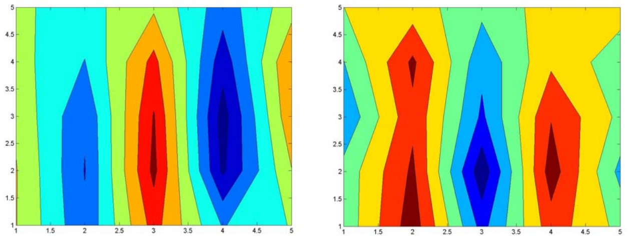

(b) The above two filters after training are almost opposite to each other

Fig. 1. A toy example of opposite filters generated after training a CNN.

In addition to this observation, such an opposite relation is only found in the first convolutional layer right behind the original input image, which is similar to the case that the center-surround receptive field appears. Moreover, we seek the cooperative relationships between the filters in higher layers, and the translational relation is frequently recognized between the convolutional kernels which belong to the same feature map, as illustrated in Fig. 2. As seen in Fig. 2, there are four examples of the filters with the translational relation to each other.

As a consequence, we come out with the idea of correlative filters to preset these relations between filters artificially, which we thought would at least speed up the convergence of CNN structure learning. Currently, two kinds of CFs are designed to simulate the discovered relevances of opposite and translation. As shown in the experiments in Section V, the proposed CF method is able to promote the classification accuracy while reducing the quantity of CNN parameters.

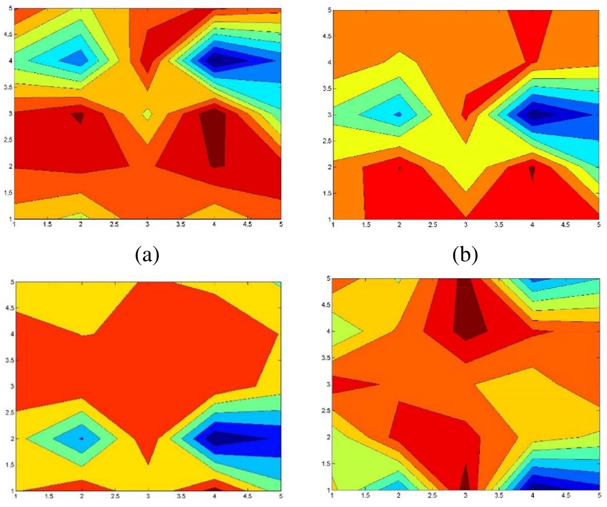


Fig. 2. A toy example of translational filters generated after training a CNN.

B. *Opposite Correlative Filter*

As the proposed opposite CF works with a normal convolutional layer, several couples of convolutional kernels are randomly chosen to be the correlative pairs. Under the condition that the opposite filters are applied to the convolutional layer right after the input layer, the filters determined to be connected should be attached to disparate output feature maps. If the opposite filters contribute to the same feature map, their convolved results are counteracted because the input channels *e.g.*, the RGB image signal, have a lot in common. In each pair of the opposite CFs, one is assigned as the master filter while the other is recognized as the dependent filter since its value is exactly the opposite to that of the master, which is illustrated in Fig. 3.

Nevertheless, it is not enough to just reset the dependent filter to the opposite of its master after each iteration, since it may not meet the requirement of back propagation. In order to learn this structure by the stochastic gradient descent progressively, the updating rule for the CFs of opposite relation is designed as follows.

$$\begin{aligned}\triangle\vec{w}_{mas,i+1} =& m\cdot\triangle\vec{w}_{mas,i}\\ &+\lambda\cdot(E(\frac{\partial J}{\partial\vec{w}_{mas,i}})-E(\frac{\partial J}{\partial\vec{w}_{dep,i}})),\end{aligned} \tag{6}$$

$$\begin{aligned}\triangle\vec{w}_{dep,i+1} =& m\cdot\triangle\vec{w}_{dep,i}\\ &+\lambda\cdot(-E(\frac{\partial J}{\partial\vec{w}_{mas,i}})+E(\frac{\partial J}{\partial\vec{w}_{dep,i}})),\end{aligned} \tag{7}$$

$$\vec{w}_{mas,i+1}=\vec{w}_{mas,i}+\triangle\vec{w}_{mas,i+1}, \tag{8}$$

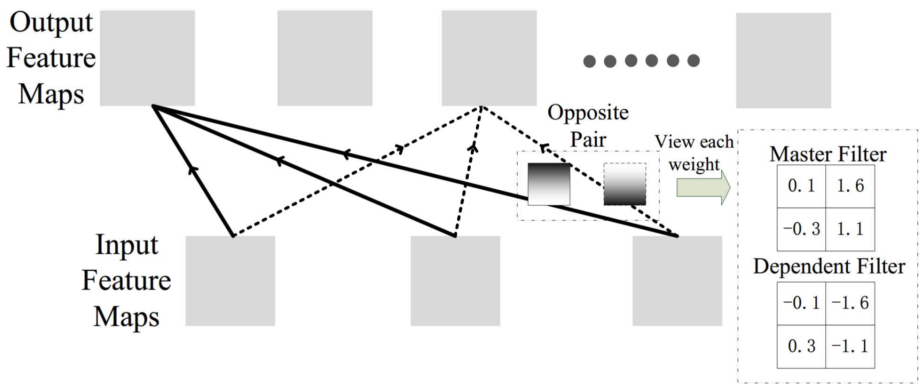


Fig. 3. Illustration of opposite correlative filters.

$$\vec{w}_{dep,i+1} = \vec{w}_{dep,i} + \triangle\vec{w}_{dep,i+1}, \tag{9}$$

where $\vec{w}_{mas,i}$ and $\vec{w}_{dep,i}$ denote the weight vectors of the master filter and dependent filter, respectively, $E(\frac{\partial J}{\partial \vec{w}_{mas,i}})$ and $E(\frac{\partial J}{\partial \vec{w}_{dep,i}})$ are similar to that of Eq. (4) except they are vectors of the partial derivatives to all the weights. In addition, $E(\frac{\partial J}{\partial \vec{w}_{mas,i}})$ and $E(\frac{\partial J}{\partial \vec{w}_{dep,i}})$ will be averaged over samples within each iteration. It should be pointed out that both $\triangle\vec{w}_{mas,0}$ and $\triangle\vec{w}_{dep,0}$ are zero vectors. Accordingly, if we have initialized the dependent filter to the negative values of master filter before training, they would always stick to the opposite of each other during every training iteration, following the adjustment strategy mentioned above. Furthermore, this adjustment only affects those correlative kernels so that the others are trained freely with normal back propagation.

Generally, the convolutional layer using opposite CFs operates with the following steps. First, once the number of the pairs of opposite filters is decided, a number of convolutional kernels are randomly selected to be correlative filters. Second, all the weights are initialized usually with the normal distribution, and the weights of the dependent filter are set as opposite to that of the corresponding master. Third, the weights are learned through forward and back propagation just like that of normal CNN structure learning. Finally, all the weights are renewed and the operation goes to the next iteration if further training is required.

C. Translational Correlative Filter

In a few aspects, the translational CF differs from the opposite filters. For each master filter, there are two dependent filters which present the translational relation to the corresponding master filter in reverse directions. Moreover, two types of batches related to filters are constructed to characterize vertical translation and horizontal translation, respectively. While the filters of the same opposite pair are recommended to come from different output feature maps, the convolutional kernels of the same translation batch will perform better if they are owned by the same output feature, which conforms to our observation as shown in Fig. 2.

As shown in Fig. 4, only half of the weights of the dependent translational filter are set to the same value as those of shifted in the corresponding master filter, and the rest half of activations are trained freely. Since these tied weights locate differently from the original master filter to the related dependent filter, the relation of translation is presented. Similar to the opposite filters, the translational filters are also trained effectively with the classic back propagation algorithm after refining the related weight tuning method. Taking the vertical batch as an example, the updating rule for translational CF is given as follows.

$$\begin{cases} \text{Given} \quad r \in \{r | 0 \le r < k - \lfloor \frac{k}{2} \rfloor\} \\ \vec{d}_{down} = E(\frac{\partial J}{\vec{w}_{M,r,i}}) + E(\frac{\partial J}{\partial \vec{w}_{D_1,r+\lfloor k/2 \rfloor,i}}) \\ \triangle\vec{w}_{M,r,i+1} = m \cdot \triangle\vec{w}_{M,r,i} + \lambda \cdot \vec{d}_{down} \\ \triangle\vec{w}_{D_1,r+\lfloor k/2 \rfloor,i+1} = \triangle\vec{w}_{M,r,i+1} \end{cases}, \tag{10}$$

$$\begin{cases} \text{Given} \quad r \in \{r | k - \lfloor \frac{k}{2} \rfloor \le r < k\} \\ \vec{d}_{up} = E(\frac{\partial J}{\vec{w}_{M,r,i}}) + E(\frac{\partial J}{\partial \vec{w}_{D_2,r-(k-\lfloor k/2 \rfloor),i}}) \\ \triangle\vec{w}_{M,r,i+1} = m \cdot \triangle\vec{w}_{M,r,i} + \lambda \cdot \vec{d}_{up} \\ \triangle\vec{w}_{D_2,r-(k-\lfloor k/2 \rfloor),i+1} = \triangle\vec{w}_{M,r,i+1} \end{cases}, \tag{11}$$

where $\vec{w}_{M,r,i}$ stands for the r^{th} row of master filter, $\vec{w}_{D_1,r+\lfloor k/2 \rfloor,i+1}$ is the row of $r + \lfloor k/2 \rfloor$ in the first dependent filter, and k is the filter width. Therefore, the first dependent filter has the downward shifting relation to its master and the second one represents the upward shift on the contrary. The updating rule for horizontal batches is nearly the same as that of vertical batches, except that the master filter is fed with several columns of weights in dependent filters, instead of diverse rows. Briefly speaking, as compared to a normal filter, half of the activations in a dependent filter is tied with those of its master to get trained simultaneously.

The training steps for translational CF are almost the same as the opposite CF. First, the amount of vertical batches and

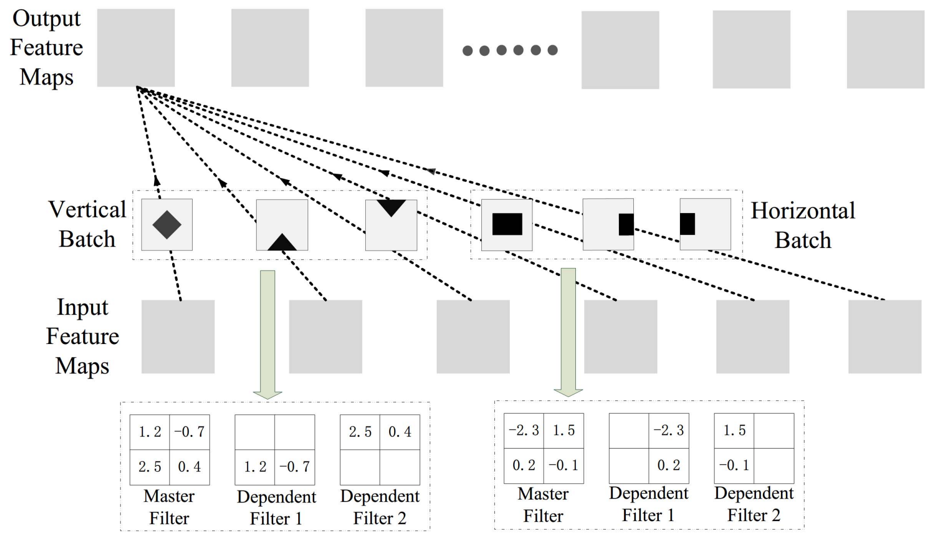


Fig. 4. Two kinds of batches for translational correlative filters.

horizontal batches are decided to build connections between correlative filters. Right after all the weights are initialized, the related weights are reset using the values of masters. For example, if one kernel is made the right forward translation of its master, as a result of that, this filter's right half would be set to the same as the left half of the original master filter. Then, the whole layer is trained by back propagation with the revised weight updating approach as given in Eqs. (10-11).

V. Experimental Results

In order to evaluate the performance of the proposed CF method, the competitive benchmark dataset of CIFAR-10 [15] is employed for experiments, with a selection of images as shown in Fig. 5. The CIFAR-10 dataset 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.

All of our experiments are based on the widely used Caffe infrastructure [16] that is among the fastest implementations of CNNs. In order to analyze the performance of the proposed CF method and its compatibility with other regularization techniques, a basic model applying CF to CNN is tested and a more complex structure is obtained by combining the proposed CF method with the stochastic pooling and Dropout techniques. Note that the data augmentation technique, which can be achieved by randomly extracting smaller patches from the original picture as well as their horizontal reflections, is currently not applied.

The basic model is mainly composed of three convolutional layers, each of which is followed by a pooling layer. These three pooling layers share the same kernel size of 3 and stride of 2, with the first one utilizing the max pooling and the rest performing average pooling. In addition, two local normalization layers are placed before the last two convolutional layers according to [6]. The first convolutional layer has 100 feature maps, and 50 pairs of opposite filters are randomly generated. On the other hand, the second convolution layer is normal to have 68 feature maps whereas the third is made up of 96 feature maps and uses the proposed translational CF.

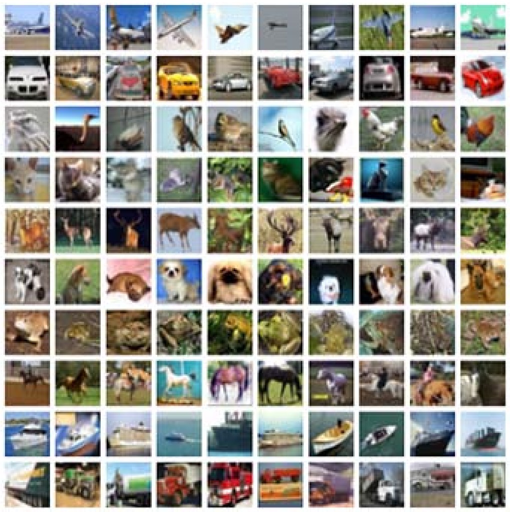


Fig. 5. A selection of images from CIFAR-10 [15].

Each feature map of the last convolutional layer owns two vertical batches and two horizontal batches as well. All the feature maps set the convolution kernel's width to 5 and are processed with the activation function of rectification. After the last pooling layer, a softmax layer of 10 units is set to do the classification. Using the momentum of 0.9, this model is trained with the learning rate (λ) of 0.001 for 240 epochs and then a decreased λ of 0.0001 for 20 epochs and at last 10 epochs with λ of 0.00001.

The comparison of the proposed CF method with some state-of-the-art methods is shown in Table I, where the test error is evaluated and no data augmentation or stochastic regularization methods are employed. As indicated from the results in Table I, the proposed CF method surpasses the classic CNN method by 1.05% in terms of test error, and is also better than the recursive CNN [12] which shares weights between different layers and demonstrates that the filters from the same layer are more correlated than those belonging to different layers. Regarding the the superior methods of Network In Network [13] and Bayesian Optimization [9], they are likely to be improved with the proposed CF as none of them takes use of the relation between filters.

TABLE I
COMPARISON OF TEST ERROR BETWEEN THE PROPOSED CF METHOD AND OTHER STATE-OF-THE-ART APPROACHES WITHOUT DATA TRANSFORMATIONS OR STOCHASTIC REGULARIZATION METHODS.

Method	Test error
Network In Network [13]	**14.51%**
CNN + Bayesian Optimization [9]	14.98%
CNN + CF (**Ours**)	15.55%
CNN + Recursive Convolution [12]	16.00%
CNN [6]	16.60%
Sparse Coding Over VQ [17]	18.50%

Moreover, through changing the method of the last two pooling layers into stochastic pooling and adding a fully connected layer before the softmax layer to apply Dropout, the over-fitting problem of our basic model can be further addressed. The inserted fully connected layer has 128 neurons and uses a drop ratio of 0.5. This complicated model is also trained by three phases with different learning rates. The numbers of epochs for these three steps are 150, 20 and 10, respectively. The comparative results are summarized in Table II. When implemented together with stochastic pooling and Dropout, the test error rate of CNN utilizing CF reduces from 15.55% to 14.61%. Moreover, we disable the relations of filters and retrain the whole net normally, the performance drops down to 14.93%. In other words, by introducing related weights which actually decreases the capacity of the CNN structure, the proposed CF method is able to enhance the CNN's ability of learning, even though both the number of feature maps and iterations stay the same. As a result, the CNN structure with CF is better than the classic one. In fact, the proposed CF method can also be employed by the state-of-the-art method DSN [8] since DSN only takes use of the discriminativeness of feature maps.

TABLE II
COMPARISON OF TEST ERROR BETWEEN THE PROPOSED CF METHOD AND OTHER STATE-OF-THE-ART APPROACHES IMPLEMENTED WITH STOCHASTIC REGULARIZATION.

Method	Test error
DSN + Dropout [8]	**9.78%**
Network In Network + Dropout [13]	10.14%
CF (**Ours**) + Stochastic Pooling + Dropout	14.61%
Stochastic Pooling + Dropout	14.93%
Stochastic Pooling [7]	15.13%
Dropout [6]	18.50%

VI. CONCLUSION

In this paper, a novel regulation method called Correlative Filter (CF) is proposed, which attempts to optimize the structure of classic CNNs by binding several filters belonging to the same convolutional layer. Different from other existing approaches, the amount of trainable weights in CNNs decreases when presented with the proposed CF. The experimental results demonstrate the effectiveness and efficiency of the proposed CF method in improving CNN's learning ability. Under the condition that alternative methods seldom take the relevance of filters into considerations, CF is promising to improve their learning abilities.

REFERENCES

[1] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *CVPR'09*, Sept.-Oct. 2009, pp. 2146–2153.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS'12*, Dec. 2012, pp. 1097–1105.
[3] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR'14*, Jun. 2014, pp. 1725–1732.
[4] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, 1995.
[5] N. Kruger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. Piater, A. J. Rodriguez-Sanchez, and L. Wiskott, "Deep hierarchies in the primate visual cortex: What can we learn for computer vision?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1847–1871, Aug. 2013.
[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.
[7] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv:1301.3557*, 2013.
[8] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," *arXiv preprint arXiv:1409.5185*, 2014.
[9] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *NIPS'12*, Dec. 2012, pp. 2951–2959.
[10] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *ICML'13*, Jul. 2013, pp. 1058–1066.
[11] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *CVPR'12*, Jun. 2012, pp. 3642–3649.
[12] D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun, "Understanding deep architectures using a recursive convolutional network," *arXiv preprint arXiv:1312.1847*, 2013.
[13] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.
[14] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *ISCAS*, May-Jun. 2010, pp. 253–256.
[15] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical Report TR-2009, University of Toronto*, 2009.
[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.
[17] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *ICML'11*, Jun.-Jul. 2011, pp. 921–928.