Efficient Discriminative Convolution Using Fisher Weight Map

Hideki Nakayama
http://www.nlab.ci.i.u-tokyo.ac.jp/

Graduate School of Information
Science and Technology
The University of Tokyo
Tokyo, JAPAN

Abstract

Convolutional neural networks (CNNs) have been studied for a long time, and recently gained increasingly more attention. Deep CNNs have especially achieved remarkably high performance on many visual recognition tasks due to their high levels of flexibility. However, since CNNs require numerous parameters to be tuned via iterative operations through layers, their computational cost is immense. Moreover, they often require a huge number of training samples and technical tricks, such as unsupervised pretraining and heuristic tuning, to successfully train the system.

In this work, we present a very simple method of layer-wise convolution. We can obtain discriminative filters by using a Fisher weight map, which well separates convolved images between categories. This operation can be deterministically solved as a simple eigenvalue problem and no back propagation or hyper-parameters are needed. Because our method is layer-wise and based on a simple eigenvalue problem, it is computationally efficient. Also, it is relatively stable with a moderate amount of training samples and is capable of learning densely from high-dimensional descriptors without dropping connections between neurons, which is a common practice in conventional implementations of CNNs. We demonstrated the promising performance of our method in extensive experiments with two datasets. Our network used together with appropriate pooling and rectification techniques achieved remarkably high performance that was distinctly comparable to those that were state-of-the-art.

1 Introduction

Deep neural networks are again gaining popularity in visual categorization tasks with the significant advances in computational power and the amount of data [11, 15]. Of these, convolutional neural networks (CNNs) [18, 22, 27] are one of the most successful approaches and have established state-of-the-art records in many benchmarks [4, 19, 20]. CNNs have limited the connections between layers just within nearby neurons in sub-windows (receptive fields) inspired by biological analysis of the visual cortex [17]. Moreover, the weight parameters for convolution have been shared by all sub-windows. Thus, CNNs have substantially reduced the number of free parameters compared to fully-connected networks.

However, CNNs still have a number of disadvantages. First, their computational cost is still immense and they usually require special implementation using graphics processing units (GPUs) or cluster computers to achieve state-of-the-art results [4, 19]. Since it is

occasionally difficult for even GPUs to train fully connected CNNs, connections are often randomly reduced so that training is feasible [9, 18]. CNNs are also prone to overfitting due to their high level capacities. We generally require a huge number of training samples to successfully train systems without the help of unsupervised pre-training [10, 24]. In addition, heuristic techniques such as dropouts [16] are often incorporated, although their theoretical background has not been fully investigated.

In this work, we present a very simple but efficient method of convolution using a weight learning method called the Fisher weight map (FWM). We can obtain weights for features that well separate convolved images between categories. This operation can be deterministically solved as a simple eigenvalue problem and no hyper-parameters for back propagation (*e.g.* weight decay) are needed. Because our method is layer-wise and based on a simple eigenvalue problem, it is computationally efficient. It does not need unsupervised pre-training and it is relatively stable with a moderate number of training samples.

We demonstrated in extensive experiments that our convolution method could reasonably improve the performance of the original descriptors. Moreover, we found that the key to achieving the best performance was to use it with appropriate methods of pooling and rectification, which is another contribution we made. Our overall network achieved surprisingly high-performance comparable to that of state-of-the-art ones, despite its simplicity.

2 Fisher weight map

The Fisher weight map (FWM) was originally proposed by Shinohara and Otsu [26] for computing spatial weights for individual pixels in images, and had its roots in Eigenface [28] and Fisherface [1]. While Eigenface and Fisherface simply perform principal component analysis (PCA) or Fisher linear discriminant analysis (FLDA) on image vectors, FWM is designed for a 2-dimensional (matrix) representation, where each pixel has multiple feature channels. FWM specifically computes weights for each pixel such that they maximize the Fisher criterion of the global feature vector. Its core algorithm is based on FLDA, and is a natural extension of the eigen weight map (EWM) [26], which is based on PCA. This idea has recently been revisited by Harada *et al.* [13, 14], who used FWM and other similar techniques to obtain weights for each region of spatial pyramids. Despite its substantially reduced dimensionality, their compact spatial pyramid representation was found to be as powerful as other state-of-the-art methods of pyramid matching.

In this work, we implement the FWM with the opposite approach that is suitable to convolute local features. Although the mathematical basis of FWM comes directly from [26], the objective here is essentially different and has not been investigated, making our contribution novel. More specifically, we compute weights for features of local neighboring pixels in receptive fields that maximize the discriminative ability of the convolved image map (Section 3.1), while the objective of [26] is to compute discriminative global features.

3 Network architecture

We consider a multilayer feedforward network for image classification (Figure 1). At the k-th layer (L_k) of the network, each image is represented by m_k feature arrays of $P_k \times P_k$ pixels. This 2-dimensional array and its elements are often called a "map" [1] and "neurons". For example, a raw color image (L_0) has three maps, each of which corresponds to R, G, and B channels. We extract local features from image patches with standard descriptors that constitute the L_1 layer. They are further followed by pooling and convolution layers. Finally,

[1] The definition of a "map" here is totally unrelated to that for the Fisher weight map or eigen weight map.

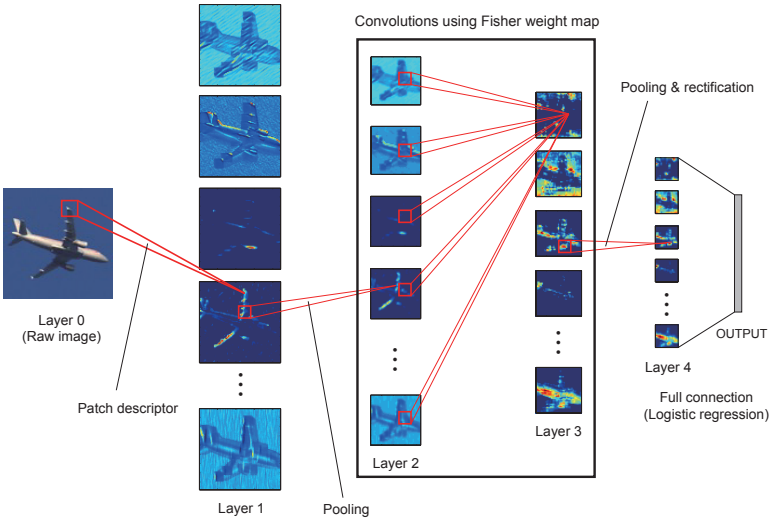

Figure 1: Illustration of our network. No back propagation is necessary for training.

a logistic regression classifier is trained using all neurons in the final layer as input. Our main contribution is the implementation of convolution layers using weight map techniques, which we describe in the following.

3.1 Convolution with weight maps

Figure 2 outlines our algorithm. Let $\boldsymbol{f}^{(k)}_{(x,y)} \in R^{m_k}$ denote the feature vector of coordinates (x, y) at the L_k-th layer, each element of which corresponds to the response of each map. We first stack neighboring feature vectors for convolution in a receptive field. For $n \times n$ convolution, we concatenate n^2 vectors as

$$\boldsymbol{x}^{(k)}_{(x',y')} = \left(\boldsymbol{f}^{(k)T}_{(x-\delta,y-\delta)} \;\; \boldsymbol{f}^{(k)T}_{(x-\delta+1,y-\delta)} \;\; \cdots \;\; \boldsymbol{f}^{(k)T}_{(x+\delta,y-\delta)} \;\; \cdots \;\; \boldsymbol{f}^{(k)T}_{(x,y)} \;\; \cdots \;\; \boldsymbol{f}^{(k)T}_{(x+\delta,y+\delta)}\right)^T, \tag{1}$$

where $\delta = \lfloor n/2 \rfloor$ and (x', y') are new coordinates for convolved images. After this operation is densely applied, we obtain $(P_k - n + 1) \times (P_k - n + 1)$ stacked vectors. We place them together as a matrix as

$$\boldsymbol{X} = \left(\boldsymbol{x}^{(k)}_{(1,1)} \; \boldsymbol{x}^{(k)}_{(2,1)} \; \cdots \; \boldsymbol{x}^{(k)}_{(P_k-n+1,P_k-n+1)}\right). \tag{2}$$

Our objective is to find a linear projection, $f^{(k+1)}_{(x',y')} = \boldsymbol{w}^T \boldsymbol{x}^{(k)}_{(x',y')} (\boldsymbol{w} \in R^{m_k \times n^2})$, that convolves all local features in a receptive field to derive a new map that embeds informative local structures. Both EWM and FWM are formulated as an eigenvalue problem to achieve this end. We obtain convolving projection that defines the next maps by using the top m_{k+1} eigenvectors as

$$\left(\boldsymbol{f}^{(k+1)}_{(1,1)} \;\; \boldsymbol{f}^{(k+1)}_{(2,1)} \;\; \cdots \;\; \boldsymbol{f}^{(k+1)}_{(P_k-n+1,P_k-n+1)}\right) = \left(\boldsymbol{w}_1 \;\; \boldsymbol{w}_2 \;\; \cdots \;\; \boldsymbol{w}_{m_{k+1}}\right)^T (\boldsymbol{X} - \bar{\boldsymbol{X}}), \tag{3}$$

where $\bar{\boldsymbol{X}}$ is the mean matrix of $\boldsymbol{X}$ in the entire training dataset. Below, we describe two weight map techniques that we tested in the experiments.

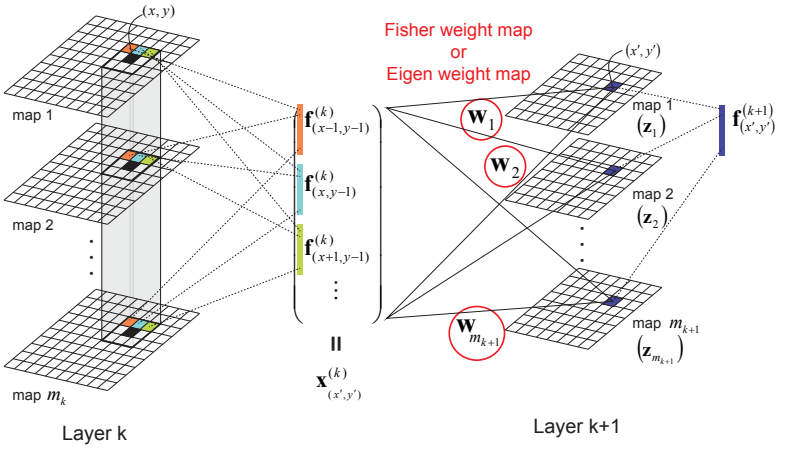

Figure 2: Convolution with weight maps (FWM or EWM).

Eigen weight map

Let $\boldsymbol{z} = \boldsymbol{X}^T\boldsymbol{w}$ denote a convolved map vector via projection $\boldsymbol{w}$. EWM finds the projection that maximizes the variance of $\boldsymbol{z}$. The variance criterion, $J_E(\boldsymbol{w})$, is written as

$$J_E(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{z}_i - \bar{\boldsymbol{z}})^T(\boldsymbol{z}_i - \bar{\boldsymbol{z}}) \tag{4}$$

$$= \boldsymbol{w}^T\left\{\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{X}_i - \bar{\boldsymbol{X}})(\boldsymbol{X}_i - \bar{\boldsymbol{X}})^T\right\}\boldsymbol{w} \tag{5}$$

$$= \boldsymbol{w}^T\Sigma_X\boldsymbol{w}, \tag{6}$$

where N is the total number of training samples and $\bar{\boldsymbol{z}}$ is the mean vector of $\boldsymbol{z}_i$. The optimal projection that maximizes $J_E(\boldsymbol{w})$ under constraint $\boldsymbol{w}^T\boldsymbol{w} = 1$ can be obtained as the eigenvector of the following eigenvalue problem

$$\Sigma_X\boldsymbol{w} = \lambda\boldsymbol{w}. \tag{7}$$

Fisher weight map

FWM finds discriminative projections by maximizing between-class distance of $\boldsymbol{z}$. While EWM is an unsupervised method, FWM maximizes the Fisher criterion, $J_F(\boldsymbol{w})$, in a supervised manner to separate categories. Therefore, it is expected to work better than EWM in terms of classification. Let $\tilde{\Sigma}_W$ and $\tilde{\Sigma}_B$ correspond to the within-class and between-class covariance matrices of $\boldsymbol{z}$, respectively. More specifically,

$$\tilde{\Sigma}_W = \frac{1}{N}\sum_{j=1}^{C}\sum_{i=1}^{N_j}(\boldsymbol{z}_i^{(j)} - \bar{\boldsymbol{z}}^{(j)})(\boldsymbol{z}_i^{(j)} - \bar{\boldsymbol{z}}^{(j)})^T, \tag{8}$$

$$\tilde{\Sigma}_B = \frac{1}{N}\sum_{j=1}^{C}N_j(\bar{\boldsymbol{z}}^{(j)} - \bar{\boldsymbol{z}})(\bar{\boldsymbol{z}}^{(j)} - \bar{\boldsymbol{z}})^T, \tag{9}$$

where C is the number of categories, N_j is the number of training samples in class j, $z_i^{(j)}$ is the i-th sample in class j, and $\bar{z}^{(j)}$ is their mean. The traces of $\tilde{\Sigma}_W$ and $\tilde{\Sigma}_B$ can be written as

$$tr\tilde{\Sigma}_W = \frac{1}{N}\sum_{j=1}^{C}\sum_{i=1}^{N_j}(z_i^{(j)} - \bar{z}^{(j)})^T(z_i^{(j)} - \bar{z}^{(j)}) \tag{10}$$

$$= \boldsymbol{w}^T\left\{\frac{1}{N}\sum_{j=1}^{C}\sum_{i=1}^{N_j}(\boldsymbol{X}_i^{(j)} - \bar{\boldsymbol{X}}^{(j)})(\boldsymbol{X}_i^{(j)} - \bar{\boldsymbol{X}}^{(j)})^T\right\}\boldsymbol{w} \tag{11}$$

$$= \boldsymbol{w}^T\Sigma_W\boldsymbol{w}. \tag{12}$$

$$tr\tilde{\Sigma}_B = \frac{1}{N}\sum_{j=1}^{C}N_j(\bar{z}^{(j)} - \bar{z})^T(\bar{z}^{(j)} - \bar{z}) \tag{13}$$

$$= \boldsymbol{w}^T\left\{\frac{1}{N}\sum_{j=1}^{C}N_j(\bar{\boldsymbol{X}}^{(j)} - \bar{\boldsymbol{X}})(\bar{\boldsymbol{X}}^{(j)} - \bar{\boldsymbol{X}})^T\right\}\boldsymbol{w} \tag{14}$$

$$= \boldsymbol{w}^T\Sigma_B\boldsymbol{w}. \tag{15}$$

Therefore, the Fisher criterion is defined as

$$J_F(\boldsymbol{w}) = \frac{tr\tilde{\Sigma}_B}{tr\tilde{\Sigma}_W} = \frac{\boldsymbol{w}^T\Sigma_B\boldsymbol{w}}{\boldsymbol{w}^T\Sigma_W\boldsymbol{w}}. \tag{16}$$

The optimal weights that maximize the above can be analytically obtained by solving the following eigenvalue problem.

$$\Sigma_B\boldsymbol{w} = \lambda\Sigma_W\boldsymbol{w}. \tag{17}$$

If the dimensions of Σ_W are too large, we can apply EWM before FWM to reduce dimensionality to cope with the singularity problem [26], although we have not done this in this study.

3.2 Descriptors

We test two descriptors to extract layer-1 features from raw images.

The first is the **Random Filter**, which convolves small image patches (*e.g.*, 3x3 and 5x5) with random weights. The weights are set randomly between -0.05 to 0.05 in our experiments. Previous work has shown that deep networks with random weights achieve reasonable performance despite their simplicity [13]. Also, they are frequently used as the initial weights for deep CNNs.

The second is the **K-means** encoding proposed by Coates *et al.* [5, 8]. We first apply contrast normalization and zero component analysis (ZCA) whitening to image patches as preprocessing. We then generate a codebook using the K-means as in the standard bag-of-words approach [9]. We use triangular encoding to encode patch features using the codebook.

3.3 Pooling

Pooling is an operation to summarize the responses of neighboring neurons in a map by obtaining their statistical properties. In this way, we can systematically incorporate local translation invariance, which is crucially important for visual object recognition. This was

often done in early work via simple sub-sampling [27]. Recent studies have commonly been based on dense pooling operations [4, 5, 29]. Here, we use average-pooling (AP) and max-pooling (MP), which have recently been theoretically analyzed by Boureau *et al*. [2].

3.4 Rectification

One of the keys to deep networks is to incorporate nonlinearity by applying an appropriate rectification function to the output of neurons. Many functions have been tested for this purpose such as the tanh and sigmoid [18, 22]. Recent state-of-the-art research has demonstrated that the Rectified Linear Units (ReLU) function, which simply takes $R(x) = \max(0, x)$ for input x, is surprisingly effective for deep networks, leading to high levels of performance and fast convergence [19, 23]. Although our architecture does not need backward propagation, we implemented ReLU to check its effectiveness. Motivated by Coates and Ng [7], we also tested the following two-dimensional rectification units to similarly exploit minus activation obtained from convolution.

$$R_2(x) = \begin{pmatrix} \max(0, x) \\ \max(0, -x) \end{pmatrix}. \tag{18}$$

4 Experiment

We compared our methods using various pooling and rectification strategies on two benchmarks, *i.e*., the STL-10 [5] and MNIST [21] datasets. The notations of network architectures are

- $Rand(n, d)$: a d-dimensional random filter applied to $n \times n$ image patches,
- $K_m(n, d)$: a K-means descriptor extracted from $n \times n$ image patches with d visual words,
- $C(n, m)$: a convolutional layer with m maps and $n \times n$ receptive fields. C_E and C_F correspond to the convolution with EWM and FWM,
- R, R_2: Rectification unit (ReLU, Section 3.4),
- $AP[MP](n, s)$: Average-pooling [Max-pooling] over $n \times n$ neighborhood spacing s pixels apart, and
- $AP[MP]_q$: Average-pooling [Max-pooling] such that pooled regions correspond to the quadrants (2×2 spatial regions) of original raw images.

For example, $Rand(200, 5)$-R-$AP(4, 4)$-$C_F(3, 100)$-R-AP_q represents the following architecture: (1) 200 random filters, followed by (2) ReLU, (3) average pooling, (4) 100 maps of 3×3 convolution using FWM, (5) ReLU, and (6) average pooling. We can carry out a fair comparison to check the effectiveness of each step by using quadrants (AP_q, MP_q) as the final layer. All the experiments were done on a single desktop PC (Core-i7 3930K with 32 GB of RAM).

4.1 STL-10 dataset

The STL-10 [5] dataset is composed of 96×96 color images of 10 object classes such as airplanes, birds, and cars. This dataset has especially been designed to investigate problems with unsupervised feature learning. While 100,000 unlabeled images are available for this purpose, there are only 100 labeled examples per class for each predefined fold. We ignored

Table 1: Classification rates on STL-10 (%) using various network architectures.

Architecture	Accuracy	
$K_m(9,200)$-AP_q	46.7 ± 1.7	
$K_m(9,200)$-MP_q	**53.4** ± **0.7**	
	EWM	FWM
$K_m(9,200)$-$AP(4,2)$-$C(3,100)$-MP_q	45.2 ± 0.9	45.6 ± 1.6
$K_m(9,200)$-$MP(4,2)$-$C(3,100)$-MP_q	49.3 ± 0.6	48.0 ± 1.3
$K_m(9,200)$-$AP(4,2)$-$C(3,100)$-AP_q	45.2 ± 1.8	55.8 ± 1.1
$K_m(9,200)$-$MP(4,2)$-$C(3,100)$-AP_q	**54.6** ± **0.9**	**58.0** ± **0.8**
$K_m(9,200)$-$AP(4,2)$-$C(3,100)$-R-AP_q	50.9 ± 0.9	57.7 ± 0.7
$K_m(9,200)$-$MP(4,2)$-$C(3,100)$-R-AP_q	54.9 ± 0.6	59.0 ± 0.5
$K_m(9,200)$-$MP(4,2)$-$C(3,100)$-R_2-AP_q	**56.9** ± **0.4**	**61.2** ± **0.6**

Table 2: Classification rates on STL-10 (%) using FWM with different numbers of features and network configurations.

Architecture \ Dictionary size (d)	200	500	1000
$K_m(9,d)$-MP_q	53.4 ± 0.7	56.5 ± 0.7	57.9 ± 0.6
$K_m(9,d)$-$MP(4,2)$-$C_F(3,100)$-R-AP_q	59.0 ± 0.5	60.9 ± 0.9	61.9 ± 0.6
$K_m(9,d)$-$MP(4,2)$-$C_F(3,100)$-R_2-AP_q	61.2 ± 0.6	62.5 ± 0.7	63.3 ± 0.4
$K_m(9,d)$-$MP(8,4)$-$C_F(3,100)$-R-AP_q	60.4 ± 0.9	62.0 ± 0.9	63.1 ± 0.7
$K_m(9,d)$-$MP(8,4)$-$C_F(3,100)$-R_2-AP_q	62.2 ± 0.8	63.6 ± 0.7	64.6 ± 0.4
$K_m(9,d)$-$MP(8,4)$-$C_F(3,100)$-R_2-$AP(4,3)$	**64.2** ± **0.7**	**65.4** ± **0.7**	**66.0** ± **0.7**

Table 3: Comparison of classification rates on STL-10 (%).

1-layer Vector Quantization [7]	54.9 ± 0.4
1-layer Sparse Coding [7]	59.0 ± 0.8
3-layer Learned Receptive Field [6]	60.1 ± 1.0
Discriminative Sum-Product Network [12]	62.3 ± 1.0
Ours, $K_m(9,1000)$-$MP(8,4)$-$C_F(3,100)$-R_2-$AP(4,3)$	**66.0** ± **0.7**

unlabeled samples in this experiment and simply trained our system with labeled examples as in Gens and Domingos [12].

We first tested various combinations of convolution and pooling operations with K-means features using 200 visual words. Table 1 summarizes the results. We could obtain a standard bag-of-words representation ($K_m(9,200)$-$AP[MP]_q$) by simply pooling the descriptors. Max-pooling achieved better performance than average-pooling for this dataset. After the convolution layer, however, average-pooling substantially improved performance from non-convolution models, while max-pooling had no effect at all. Moreover, we found that ReLU after convolution could consistently improve performance; the R_2 filter was found to be especially effective. These results indicated that it was crucial to use our methods of convolution with adequate pooling and rectification techniques. Both EWM and FWM outperformed the $K_m(9,200)$-MP_q model when used with average-pooling and the R_2 filter. As expected, FWM, which is a discriminative convolution layer, achieved the best results.

We next fine-tuned our model with more features as in Table 2. We found that our method

Table 4: Classification errors on MNIST (%) using various network architectures.

Architecture	Error	
$Rand(5,500)$-AP_q	14.57	
$Rand(5,500)$-MP_q	1.02	
$Rand(5,500)$-R-AP_q	**0.89**	
$Rand(5,500)$-R-MP_q	1.02	
	EWM	FWM
$Rand(5,500)$-R-$MP(3,2)$-$C(3,200)$-AP_q	4.13	2.49
$Rand(5,500)$-R-$AP(3,2)$-$C(3,200)$-AP_q	3.33	2.06
$Rand(5,500)$-R-$AP(3,2)$-$C(3,200)$-R-AP_q	0.95	0.67
$Rand(5,500)$-R-$AP(3,2)$-$C(3,200)$-R_2-AP_q	**0.90**	**0.54**

Table 5: Classification errors on MNIST (%) using different size and feature dimensionality of receptive fields.

$Rand(5,d)$-R-$AP(3,2)$-$C_F(n,100)$-R_2-AP_q			
n \ d	100	200	300
1	1.09 (100)	0.86 (200)	0.73 (300)
2	0.68 (400)	0.68 (800)	0.63 (1200)
3	0.69 (900)	0.66 (1800)	0.61 (2700)
4	0.71 (1600)	0.60 (3200)	0.56 (4800)
5	0.62 (2500)	0.56 (5000)	0.52 (7500)

of convolution could successfully learn weight parameters from more features without overfitting, although there were only 100 labeled examples per class. Moreover, we could further improve performance by using a finer grid for the last pooling layer ($AP(4,3)$) instead of quadrants. We have compared the scores for our method and previous ones in Table 3. Our best model outperformed all previously published scores in the literature by a large margin.

4.2 MNIST dataset

MNIST is a dataset of tiny (28×28 pixels[2]) handwritten digits, and has been a de-facto standard for the study of deep networks. There are 60,000 training samples and 10,000 testing samples in this dataset. Approximately 6,000 training and 1,000 testing samples are specified per class (0 to 9). We found that random filters worked well for this dataset. Table 4 summarizes classification error using different combinations of convolution and pooling methods. It was somewhat surprising that just applying ReLU and average-pooling could greatly improve performance ($Rand(5,500)$-R-AP_q). After convolution, however, performance fell without the help of ReLU. The results basically correspond to those for STL-10, but ReLU appears to be more important for this dataset. Also, EWM, unlike in STL-10, does not improve performance from the original random features.

We next investigated what effects the size of the receptive field and dimensionality of features had on convolution. Table 5 lists the results. The numbers in parentheses represent the number of features in a receptive field for each case. We observed that the key factor was

[2]We append one column and one row with zeros so that images are 29×29.

Table 6: Classification errors on MNIST (%) with best configurations.

Architecture \ Filter dimensions (d)	200	500	1000
$Rand(5, d)$-R-AP_q	1.02	0.89	0.87
$Rand(5, d)$-R-$AP(3, 2)$-$C_F(3, 200)$-R-AP_q	0.67	0.66	0.67
$Rand(5, d)$-R-$AP(3, 2)$-$C_F(3, 200)$-R_2-AP_q	0.55	0.54	0.47
$Rand(5, d)$-R-$AP(3, 2)$-$C_F(3, 500)$-R_2-AP_q	0.57	0.47	**0.44**

Table 7: Comparison of classification errors on MNIST (%). We compared our method with previous work using raw training dataset. (*) Ciresan *et al.* [4] achieved 0.23% using elastic distortions.

Large CNN (unsup. pretraining) [25]	0.60
Large CNN (unsup. pretraining) [18]	0.53
3-layer CNN + Stochastic Pooling [29]	0.47
Multi-Column Deep Neural Network [4]	0.46*
Ours, $Rand(5, 1000)$-R-$AP(3, 2)$-$C_F(3, 500)$-R_2-AP_q	**0.44**

the number of features in a receptive field, rather than its size. Based on this, we increased the dimensions of features fixing $n = 3$ (Table 6) and effectively obtained better scores.

Table 7 summarizes our comparison with previous work. State-of-the-art work on this dataset used augmented training data with various distortions on original images. Because this was not of interest in this work, we compared our method with previous scores for the original raw dataset. To the best of our knowledge, the best previous score with this setup (no distortion) was that by Ciresan *et al.* [4], which was based on the combination of five CNNs. We achieved 0.44% using a single network with a FWM convolution layer with 500 maps. Also Zeiler and Fergus [29] achieved 0.47% using a novel technique of pooling called stochastic pooling with CNNs. It would be interesting to implement stochastic pooling in our network, which we intend to do in future work.

5 Conclusion

We presented a simple but efficient method of layer-wise discriminative convolution based on a Fisher weight map. Our method of convolution can be analytically solved and no hyper-parameters for backward learning are needed. The experimental results revealed that our convolution layer could reasonably improve the performance of the original descriptors. However, we also found that just using FWM for convolution was insufficient to obtain the best performance. Our method, used together with appropriate pooling methods and ReLU operations, achieved remarkably high levels of performance on STL-10 and MNIST datasets that were comparable or better than those of state-of-the-art networks. Superior results on STL-10 especially indicated that our method could stably learn from a limited number of training examples. This is probably because our method is based on a simple eigenvalue problem and is capable of densely learning from high-dimensional descriptors without dropping connections between neurons. This fact indicates the importance of fully-connected convolutional networks, considering that connections are often reduced in many other CNN methods.

We intend to extend our method to handle hierarchical convolution in future work. Moreover, it would be interesting to initialize deep networks with our method and fine tune them with state-of-the-art methods of gradient descent.

Acknowledgement

This work is partially supported by Hoso Bunka Foundation.

References

[1] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Trans. PAMI*, 19(7):711–720, 1997.

[2] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proc. ICML*, 2010.

[3] D. Ciresan, U. Meier, J. Masci, L. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proc. IJCAI*, 2011.

[4] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proc. IEEE CVPR*, 2012.

[5] A. Coates, H. Lee, and A. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proc. AISTATS*, 2011.

[6] A. Coates and A. Ng. Selecting receptive fields in deep networks. In *Proc. NIPS*, 2011.

[7] A. Coates and A. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proc. ICML*, 2011.

[8] A. Coates and A. Ng. Learning feature representations with K-means. *Neural Networks: Tricks of the Trade*, 2012.

[9] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proc. ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

[10] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, and P. Vincent. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11: 625–660, 2010.

[11] K. Fukushima. Neocognitron for handwritten digit recognition. *Neurocomputing*, 51: 161–180, 2003.

[12] R. Gens and P. Domingos. Discriminative learning of sum-product networks. In *Proc. NIPS*, 2012.

[13] T. Harada, H. Nakayama, and Y. Kuniyoshi. Improving local descriptors by embedding global and local spatial information. In *Proc. ECCV*, 2010.

[14] T. Harada, Y. Ushiku, Y. Yamashita, and Y. Kuniyoshi. Discriminative spatial pyramid. In *Proc. IEEE CVPR*, pages 1617–1624, 2011.

[15] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

[16] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint*, 2012.

[17] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148:574–591, 1959.

[18] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. Lecun. What is the best multi-stage architecture for object recognition? In *Proc. IEEE ICCV*, 2009.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. NIPS*, 2012.

[20] Q. V Le, M. A. Ranzato, M. Devin, G. S. Corrado, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *Proc. ICML*, 2012.

[21] Y. LeCun. The MNIST database of handwritten digits. URL http://yann.lecun.com/exdb/mnist/.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 1998.

[23] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. ICML*, 2010.

[24] M. A. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. IEEE CVPR*, 2007.

[25] M. A. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *Proc. NIPS*, 2006.

[26] Y. Shinohara and N. Otsu. Facial expression recognition using Fisher weight maps. In *IEEE FG*, pages 499–504, 2004.

[27] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proc. ICDAR*, 2003.

[28] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proc. IEEE CVPR*, 1991.

[29] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *arXiv preprint*, 2013.