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, 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 (FWM) [8], 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. We demonstrate the promising performance of our method in extensive experiments with two datasets.

Network architecture

Let $\boldsymbol{x}^{(k)}_{(x',y')}$ denote a vector of stacked features in a receptive filed in the k-th layer (see Figure 1), and $\boldsymbol{X}$ denote a matrix that contains all $\boldsymbol{x}^{(k)}$ from an instance in its columns. Also, we let $\boldsymbol{z} = \boldsymbol{X}^T\boldsymbol{w}$ denote a convolved image vector via projection $\boldsymbol{w}$. FWM finds discriminative projections by maximizing between-class distance of $\boldsymbol{z}$. The optimal weights can be analytically obtained by solving the following eigenvalue problem.

$$\Sigma_B \boldsymbol{w} = \lambda \Sigma_W \boldsymbol{w}. \tag{1}$$

Here, Σ_W and Σ_B are within- and between-class scatter matrices, namely

$$\Sigma_W = \sum_{j=1}^{C}\sum_{i=1}^{N_j}(\boldsymbol{X}_i^{(j)} - \bar{\boldsymbol{X}}^{(j)})(\boldsymbol{X}_i^{(j)} - \bar{\boldsymbol{X}}^{(j)})^T, \tag{2}$$

$$\Sigma_B = \sum_{j=1}^{C} N_j(\bar{\boldsymbol{X}}^{(j)} - \bar{\boldsymbol{X}})(\bar{\boldsymbol{X}}^{(j)} - \bar{\boldsymbol{X}})^T, \tag{3}$$

where C is the number of categories and N_j is the number of training samples in class j.

Moreover, we found that the key to achieving the best performance was to use the above method with appropriate methods of pooling and rectification, which is another contribution we made. More specifically, it is crucial to use our convolutional layer with subsequent average-pooling (AP) and rectified linear units (ReLU) operations. We observed that ReLU exploiting both positive and negative activations are particularly effective.

$$R_2(x) = \begin{pmatrix} \max(0, x) \\ \max(0, -x) \end{pmatrix}. \tag{4}$$

Results and discussion

We tested our methods on two benchmarks, *i.e.*, the STL-10 [2] and MNIST [7] datasets (Figure 2). The experimental results revealed that our convolution layer could reasonably improve the performance of the original descriptors. Moreover, our method used together with appropriate pooling methods and ReLU operations achieved remarkably high levels of performance on both datasets that were comparable or better than those of state-of-the-art networks.

Tables 1 and 2 summarize the scores of our method and those from previous work for STL-10 and MNIST, respectively. We implemented K-means features for STL-10 and random features for MNIST as the low-level inputs of the networks. Our best model outperformed all previously published scores in the literature. Superior results on STL-10 especially indicated that our method could stably learn from a limited number of training examples (100 per class). 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.

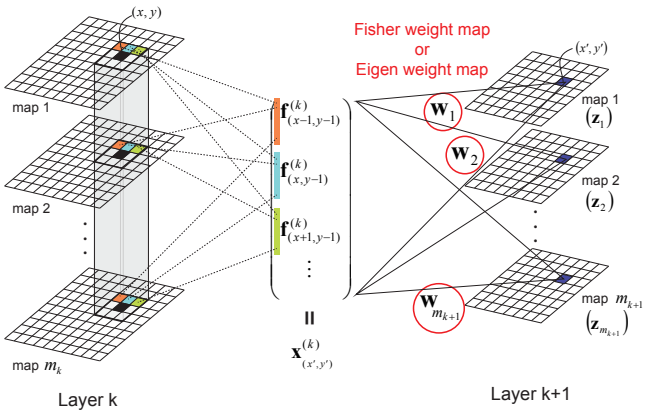

Figure 1: Convolution with weight map techniques.

Figure 2: Images from the STL-10 (top) and MNIST (bottom) datasets.

Table 1: Comparison of classification rates on STL-10 (%).

1-layer Vector Quantization [4]	54.9 ± 0.4
1-layer Sparse Coding [4]	59.0 ± 0.8
3-layer Learned Receptive Field [3]	60.1 ± 1.0
Discriminative Sum-Product Network [5]	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**

Table 2: Comparison of classification errors on MNIST (%). We compared our method with previous work using raw training dataset.

Large CNN (unsup. pretraining) [6]	0.53
3-layer CNN + Stochastic Pooling [9]	0.47
Multi-Column Deep Neural Network [1]	0.46*
Ours, $Rand(5, 1000)$-R-$AP(3, 2)$-$C_F(3, 500)$-R_2-AP_q	**0.44**

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

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

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

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

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

[6] 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.

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

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

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