Progress report: A simple PCANet-CNN for Image classification

Yin Zhang
Department of Statistics
University of Virginia
Charlottesville, VA 22903
yz4an@virginia.edu

Haoran Liu
Department of Computer Science
University of Virginia
Charlottesville, VA 22903
hl4fb@virginia.edu

Abstract

In this paper, we are going to propose a simple deep learning architecture for image classification which is based on the PCANet [1]. In the proposed architecture, we extend the PCANet by connecting it with a convolutional network [2], and the convolutional filters can be learned from back propagation. We will test the proposed deep network in various image datasets for different visual tasks such as CIFAR10 dataset for objection recognition, MNIST dataset for digit recognition and LFW dataset for face verification. We are going to compare the our result with the state of the art and the baselines in [1].

1 Motivation and Overview

This work is mainly motivated by the PCANet, which served as a simple but competitive deep learning baseline. Typically the deep learning networks (DNN) contains stacked trainable stages and is then followed by a supervised loss function. Each stage comprises a fully-connected layer or convolutional layer together followed by a nonlinear neuron function. DNN usually needs time-consuming training via back propagation.

In contrast, PCANet comprises only very basic data processing components. The multi-stage filters are learned by a simple principal component analysis, and no linear operation is involved until its very last layer, where binary hashing and histogram are employed to compute the features. Thus, back propagation is not needed for parameters updating and this results in a very efficient model training. The architecture of PCANet is shown in Figure 1.

Our goal is to design a novel deep learning architecture, which starts with a PCANet, without binary hashing or histogram, and then is followed by a convolutional network and a supervised loss such as SVM loss or cross entropy loss. Our idea is as follows: the first two stages of PCANet already have learned a very good representation of each image, however, the binary hashing and histogram may lose some information. Therefore, we connect it with convolutional layer(s) and hope the information can be retained as much as possible, thereby leading to a better performance. Since we believe the input features of the convolutional layer(s) are well learned (by PCANet), so we guess it will not take too much time to train the these convolutional filters.

2 Model

Suppose that we are given N input training images $\{\mathcal{I}_i\}_{i=1}^N$ of size $m \times n$, and assume that the patch size is $k_1 \times k_2$ at all stages in PCANet. The architecture of PCANet is shown in Figure 1. Here we discard the binary hashing and histogram in the output stage and connect the remaining PCANet with regular convolution layers. The PCA filters only need to be learned from the data one time, but the convolutional filters require to be learned by back propagation.

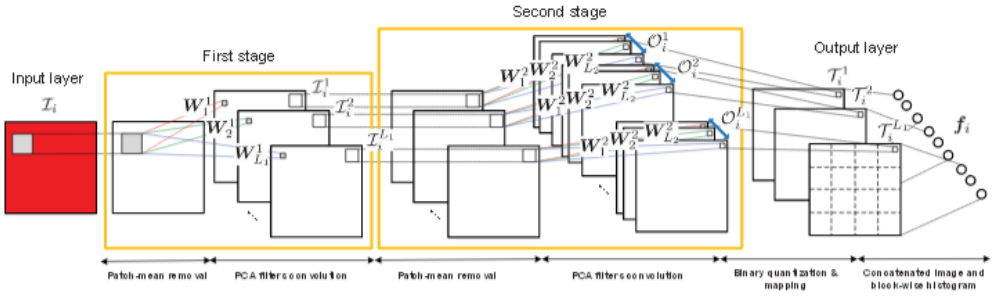

Fig. 2. The detailed block diagram of the proposed (two-stage) PCANet.

Figure 1: The Architecture of PCANet

2.1 The first stage of PCA Network (PCANet)

Around each pixel, we take a $k_1 \times k_2$ patch, and we collect all (overlapping) patches of the ith image; i.e., $\boldsymbol{x}_{i,1}, ..., \boldsymbol{x}_{i,mn} \in \mathbb{R}^{k_1 \times k_2}$. We then subtract mean from each patch and obtain $\bar{\boldsymbol{X}} = [\bar{\boldsymbol{x}}_{i,1}, ..., \bar{\boldsymbol{x}}_{i,mn}]$. By constructing the same matrix for all input images and putting them together, we get

$$\boldsymbol{X} = [\bar{\boldsymbol{X}}_1, ..., \bar{\boldsymbol{X}}_N] \in \mathbb{R}^{k_1 k_2 \times Nmn} \tag{1}$$

In layer 1, we can obtain the first L_1 PCA filters by minimizing the reconstruction error,i.e.

$$\hat{\boldsymbol{V}} = \arg \min_{\boldsymbol{V} \in \mathbb{R}^{k_1 K_2 \times L_1}} ||\boldsymbol{X} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{X}||_F^2 \quad \text{s.t. } \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I} \tag{2}$$

The solution is known as the L_1 principal vectors of $\boldsymbol{X}\boldsymbol{X}^T$. Then we can obtain the PCA filters by reshaping column l of $\hat{\boldsymbol{V}}$ to a matrix $\boldsymbol{W}_l^1$ of dimension $k_1 \times k_2$, for $l = 1, ..., L_1$. The lth filter output of the first stage is

$$\mathcal{I}_i^l = \mathcal{I} * \boldsymbol{W}_l^1 \tag{3}$$

where $*$ denotes the 2D convolution. Therefore, the number of output of first stage is L_1, but the size of each output depends on the strides and number of zero paddings.

2.2 The second stage of PCA Network (PCANet)

Almost repeating the same process as the first stage. We can collect all the overlapping patches $\mathcal{I}_i^l$ from the first stage, and form $\bar{\boldsymbol{Y}}^l = [\bar{\boldsymbol{y}}_{i,l,1}, ..., \bar{\boldsymbol{y}}_{i,l,mn}] \in \mathbb{R}^{k_1 K_2 \times mn}$, where $\bar{\boldsymbol{y}}_{i,l,j}$ is the jth mean-removed patch in $\mathcal{I}_i^l$. We further define $\boldsymbol{Y}^l = [\bar{\boldsymbol{Y}}_1^l, ..., \bar{\boldsymbol{Y}}_N^l]$ for the matrix collecting all the mean-removed patches of the lth filter, and concatenate $\boldsymbol{Y}^l$ for all the filter outputs as

$$\boldsymbol{Y} = [\boldsymbol{Y}^1, ..., \boldsymbol{Y}^{L_1}] \in \mathbb{R}^{k_1 k_2 \times L_1 Nmn} \tag{4}$$

The PCA filters of second stage are the eigenvectors of matrix $\boldsymbol{Y}\boldsymbol{Y}^T$. The number of outputs of the second stage is $L_1 L_2$.

2.3 Convolutional Network

We are going to design a 2 layer CNN following the PCANet, and the loss function can be either hinge loss (SVM) or cross entropy loss (logistic regression).

3 Experiments

We will evaluate the proposed network in various tasks, and compare the results with the state of the art and the baselines in [1]. At this moment, we just finished the code, and the program is still running. The code is released at https://github.com/tonyzhang1231/PCANet-CNN-torch. Unfortunately, we haven't developed the GPU mode for this program, and only CPU mode is available at this moment.

For each task, we plan to set up different cases in terms of varying PCA filter size, number of PCA filters, number of convolutional filters.

3.1 Object Recognition on CIFAR10

- Impact of the filter size
- Impact of number of PCA filters
- Impact of number of convolutional filters
- comparison with state of the art

3.2 Digit recognition on MNIST

- Impact of the filter size
- Impact of number of PCA filters
- Impact of number of convolutional filters
- comparison with state of the art

3.3 Face Verification on LFW

- Impact of the filter size
- Impact of number of PCA filters
- Impact of number of convolutional filters
- comparison with state of the art

4 Future work

To develop a nn.Module class for PCANet, which inherit the basic nn.Module class. Then it can be integrated into a larger network. The problem we can see so far is that the PCANet can not be placed in the middle of the network. Since if the input of the PCA layers change, we need to retrain the PCA filters, which is super computationally expensive.

References

[1] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.