HOMEWORK #6

University of Southern California

Lecturer: C.-C.JayKuo | Issued: 4/08/2019 | Due:4/28/2019

Report

1 Problem 1: Feedforward CNN Design and Its Application to the MNIST Dataset

1.1 Understanding of feedforward-designed convolutional neural networks (FF-CNNs)

1.1.1 Abstract and Motivation

FF-CNN aim at optimize the parameters in one pass feedforward computation rather than take plenty of BP steps. The FF-CNN model is consist of two stage: The first stage is the construction of convolution layers use saab transformation. The second stage is to mimic fully-connected layers using multi-stage linear least squared regressor. The details are as follows.

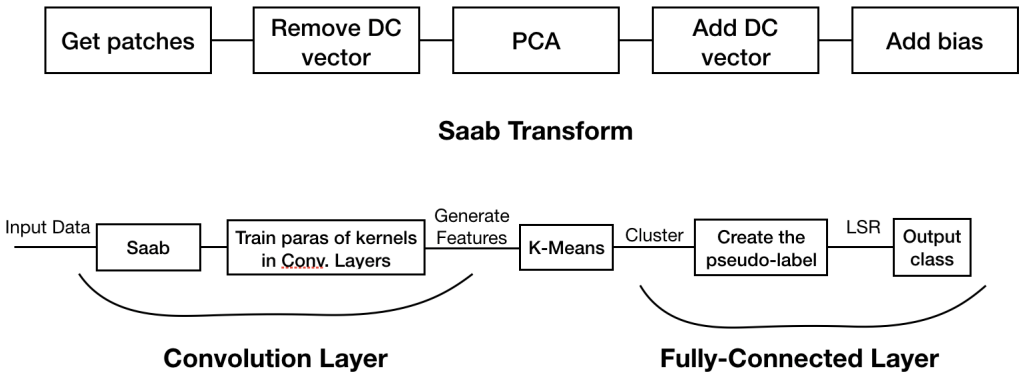

Figure 1: Flow Chart

1.1.2 Approach and Procedures

1. Convolution Layes
   The first part of FF-CNN is to construct convolution layers. Taking the LeNet-5 as an example, The forward part is similar – use 6 kernel of size 5×5 to do the convolution. the convolutional layers provide a sequence of spatial-spectral transformations that convert an input image to its joint spatial-spectral representations layer by layer. The training data give a sample distribute in the input data space. The parameter of the convolution is determined by saab transformation. Since the input data distribution is provided by mass input image, we could probably find the potential domain feature via dimension reduction. That

is, use principal component analysis(PCA) to reduce the dataset into subspace with lower dimensions.

Spatial-spectral filtering

We first use a patch of size 5×5 to scan the image and derive the pattern of the images. Thus we get a set of patterns. Then use PCA to find the dominant patterns which called anchor vectors. The anchor vector is That is, similar to the AC/DC concept in electricity, we split anchor vector into AC/DC vector. The DC anchor vector is a norm 1 vector with equal entries.

$$\boldsymbol{a}_0 = \frac{1}{\sqrt{N}}[1, 1, 1, 1, 1...1]^T_{N\times 1} \tag{1}$$

AC anchor vector is anchor vector minus DC vector. In a word, AC anchor vector is nothing but a zero-mean vector. Doing PCA on AC anchor vector and get 6 eigenvector corrsponding to the largest eigenvalue. These eigenvectors are considered as AC anchor vectors. Then combine AC and DC part to form the anchor vectors.

$$p_k = \boldsymbol{a}_k^T \boldsymbol{f}, k = 0, 1, ..K \tag{2}$$

$$\tilde{f} = f - \boldsymbol{p}_0 \boldsymbol{a}_0 \tag{3}$$

Before continue to the next stage, the bias should be added to the anchor vector to make all entried non-negtive. Bias is the current feature with the largest norm. This selection satisfied two constrain: It's a constant and None negative response.

This is called saab transformation. After several saab transformations, we do maxpooling and train the parameters of the convolution kernel properly.

2. Fully-Connected Layes
   After convolution layers, features are extracted by anchor vector. Use K-Means cluster to classify the feature into 20 sub-classes. Label the feature dataset by their centroid.This is called pseudo-label. Then using Linear Least Square Regressor (LLSR) to classify the data into the one-hot vector. Repeat this process, weights for fully connected layer can be derived from least square regression's weight. After each LSR, FF-CNN use ReLU and samples to final decision space gradually.

3. similarities and differences between FF-CNNs and BP-CNNs
   Overall, FF-CNN does not need to do back propagation to train the parameters. All the parameters are obtained by one-pass feedforward operation.

   For tradational CNN, it is hard to understand the insight of the model, especially with large parameters. On the other hand, the architecture of FF-CNN is straight forward. In addition, CNN could use SGD with mini-batch to train the parameters. Thus, it can handle a large amount of dataset. However, FF-CNN have to train the entire datasets at once. This operation require a very large RAM. Thus, it may not easily to handle a large dataset.

1.2 Image reconstructions from Saab coefficients

1.2.1 Abstract and Motivation

In this section, we first use FF-CNN to compute the saab coefficient, then use the matrix inverse method to compute the original image.

1.2.2 Approach and Procedures

As we know, saab coefficient is compute via PCA method. I use two-stage Saab transforms with 4×4 kernel and the stirde is 4, which means the patches are not overlapping with others. I use k_1 kernels in the first stage and k_2 kernels in the next stage. In the first stage, total $8 \times 8 = 64$ patches are extract from the original image. Use saab transform to choose k_1 patches ($k_1 \leq 4 \times 4$). This is achieved by reduce the patches demension to $(64, k_1)$. Then input this dataset to the second stage. Before convolution, the dataset is reshaped into $(8, 8, k_1)$. Do the saab transformation again and result in a $(4, k_2$ patches set. In the first stage, demension of patches is reduced from 4×4 to k_1, while in the second stage, the demension is reduced from $4 \times 4 \times k_1$ to k_2. Thus, in order to reconstruct the image, we should compute the pseudo-inverse matrix of the projection matrix and reshape the extracted features back to the original image.

Firstly, we need to compute the reconstruct image of the second stage. We compute the dot product of the pseudo-inverse of saak coefficient and extract feature. Then minus the bias which corresponding to the add bias part in saab transformation. This result in a 4-dimension matrix with the forth dimension equal to k_1. Then we need to reshape the first three dimension into 8×8 matrix. This is corrsponding to the features patches extract by the first stage. Then repeat this until we finally get a 32×32 reconstruct image.

1.2.3 Experimental Result

The result of the reconstruct image are show in figure 2 to 10.

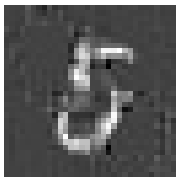

Figure 2: 1st stage = 10, 2nd stage = 100, PSNR = 16.96

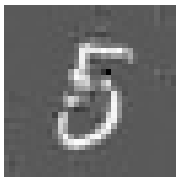

Figure 3: 1st stage = 10, 2nd stage = 140, PSNR = 20.38

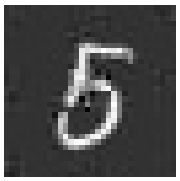

Figure 4: 1st stage = 10, 2nd stage = 160, PSNR = 23.58

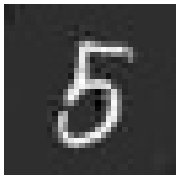

Figure 5: 1st stage = 10, 2nd stage = 170, PSNR = 27.44

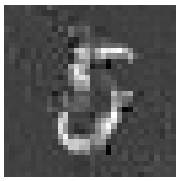

Figure 6: 1st stage = 15, 2nd stage = 100, PSNR = 16.56

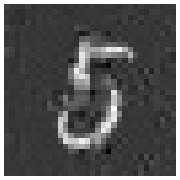

Figure 7: 1st stage = 15, 2nd stage = 140, PSNR = 18.19

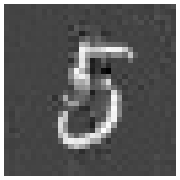

Figure 8: 1st stage = 15, 2nd stage = 160, PSNR = 19.42

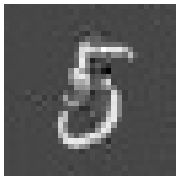

Figure 9: 1st stage = 15, 2nd stage = 170, PSNR = 19.77

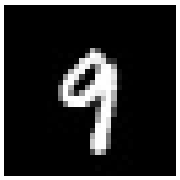

Figure 10: 1st stage = 16, 2nd stage = 256, PSNR = 202.77

1.2.4 Discussion

It is easy to observe that if we preserve all the kernel in every stage to do reconstruct, the PSNR could reach over 200. The image could perfectly be recovered. This is because all the features are preserve during the saab transformation. Besides, smaller number of kernels in the 1st stages and larger numbers of the 2nd stage tends to preserve more information.

1.3 Handwritten digits recognition using ensembles of feedforward design

1.3.1 Abstract and Motivation

In this section, I will train the FF-CNN on MNIST and report the training and testing classification accuracy. Then I building the ensemble systems to improve the performance of FF-CNN.

1.3.2 Approach and Procedures

First train the FF-CNN with MNIST. The FF-CNN cannot deal with 60000 training example. Thus, I use 20000 training set as the input. In the saak transform part, the first stage has 6 kernels and the second has 16 kernels, with kernel size 5×5 and stride 1. In the fully-connection part, the filters are 120 and 80.

I do five times training and testing on the model and compute the mean and variance.

1.3.3 Experimental Result

The mean of training accuracy 97.21% and the variance is 6.7×10^{-8}. The mean of testing accuracy is 96.99 % and the variance is 6.9×10^{-8}.

Error analysis: The best training result in HW # 5 is 99.12% and the testing result is 98.67 %. We can see the test error of FF-CNN is about the same as the BF-CNN while the training error is higher. That might because the feature extract by FF-CNN model is more general while the BP-CNN is more specific. FF-CNN aim at the entire dataset while it loss some confident on each specific data. BF-CNN deals with each dataset and extract feature from the specific image. So the training accuracy of BP-CNN is higher. Another difference come from the Fully-Connection part. Different initialization of K-Means might result in large difference in cluster result.

To ensemble the model, we need to train 10 FF-CNN model seperately. Use the result of 10 FF-CNN model to do the channel-wized PCA and get the final result.

The first idea came up with bagging. Choose the dataset and form 10 small dataset with replacement and train seperately. Each sub-dataset contains 5000 images. Then use majority vote method to combine the result. The result are shown in table 1.

Table 1: FF-CNN with Bagging Method on MNIST

	ES1	ES2	ES3	ES4	ES5	ES6	ES7	ES8	ES9	ES10	Total
acc	97.52	97.40	97.24	97.01	97.32	96.89	97.44	97.25	97.64	97.59	97.30

Test accuracy for each network is around 0.9640, while train accuracy is around 0.98. The result does not see much improvement. That might because the FF-CNN tend to extract the general features among the training set. If the dataset does not give a nice probability distribution, it will under perform. Then I tried different Convolutional-layer parameter settings. I set the size of two-stage filters with size equal to 3,5,7. The result are show in Table 2.

Table 2: FF-CNN Ensemble Method with Different kernel size on MNIST

	BP-CNN	ES1	ES2	ES3	ES4	ES5	ES6	ES7	ES8	ES9	Total
kernel size	5,5	5,5	5,3	3,5	3,3	5,7	7,7	7,5	7,3	3,7	
acc	98.97	97.34	97.04	96.43	96.72	96.12	96.22	96.93	95.91	97.35	97.33

From the tabel 2 we can see the test accuracy increase from 96.99 to 97.33. By enlarging the respect field, the kernel could extract more information, while small kernels could preserve the texture and edges. Combine the advantage of different size kernel by majoruty voting. This is similar to random forest, which also ensemble at the feature aspect. It could not only improve the accuracy, but also prevent overfitting.