EE569

HW 6

YAN JIAO

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

The conventional method to determine the parameters of the CNN is to use BP. When finish one time forward training, it needs to go backward to calculate the loss function to adjust the parameters, and keep doing forward and backward until the loss is very low. This procedure needs large amount of computing time and memory space. So a new way to determine the CNN’s parameter is introduced which only need forward training procedure, all the parameters can be determined through the one time forward training. The main idea is to use Saab signal transform, k-means and LSR in CONV layers and FC layers separately.

More details:

* Convolution layers:

For each CONV layer, our goal is to get the parameters of DC/AC anchor vectors and bias term.

DC anchor vector is all-one vector normalized by . For example, if we have kernel size of 5\*5, then the DC anchor vector is [0.2 0.2 … 0.2].

In order to calculate the parameters of the AC anchor vector, we can use the PCA.

First, we need to calculate the mean-removed input data, to make them have mean of zero.

Then, get the covariance of all the inputs and their corresponding eigenvectors. Choose the top eigenvectors with top eigenvalues as the AC anchor vectors (filters).

For bias selection, we should follow two constrains:

1. after add the bias term, all the output response should be none negative;
2. in order to simplify the computation, we set all bias terms to be the same.

Max-pooling can:

1. reduce our special size;
2. make the feature extraction more stable and robust.

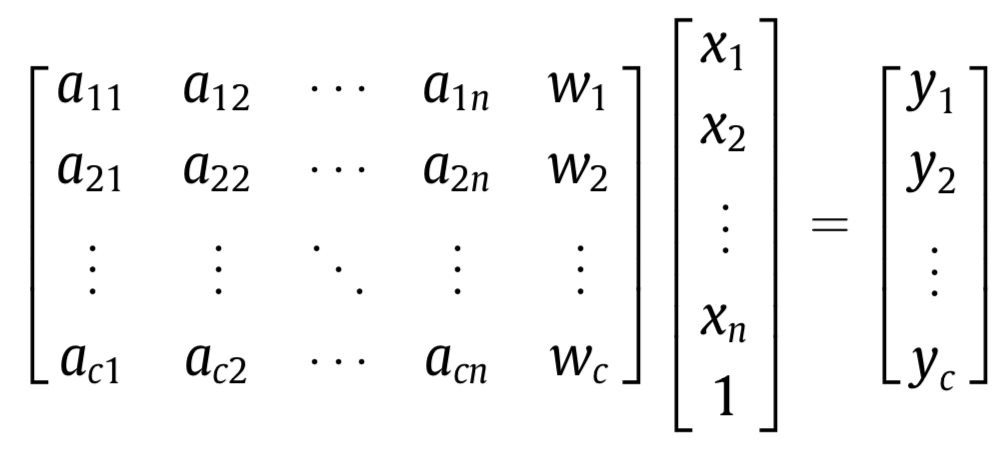
* FC layers (take the 1st FC layer as example):

For the first FC layer, we need to use K-means to cluster all the features to 120 classes.

We can simplify the K-means procedure by applying the K-means in each object classes (like in the MNIST, in each digit class, we need to generate 12 clusters, and these are the pseudo labels of the 12 different styles of one digit).

For each training input data, after the K-means, it will have a 120D one-hot vector (with one “1” and 119 “0”)

In the training procedure, we have to derive a linear least-squared regressor:

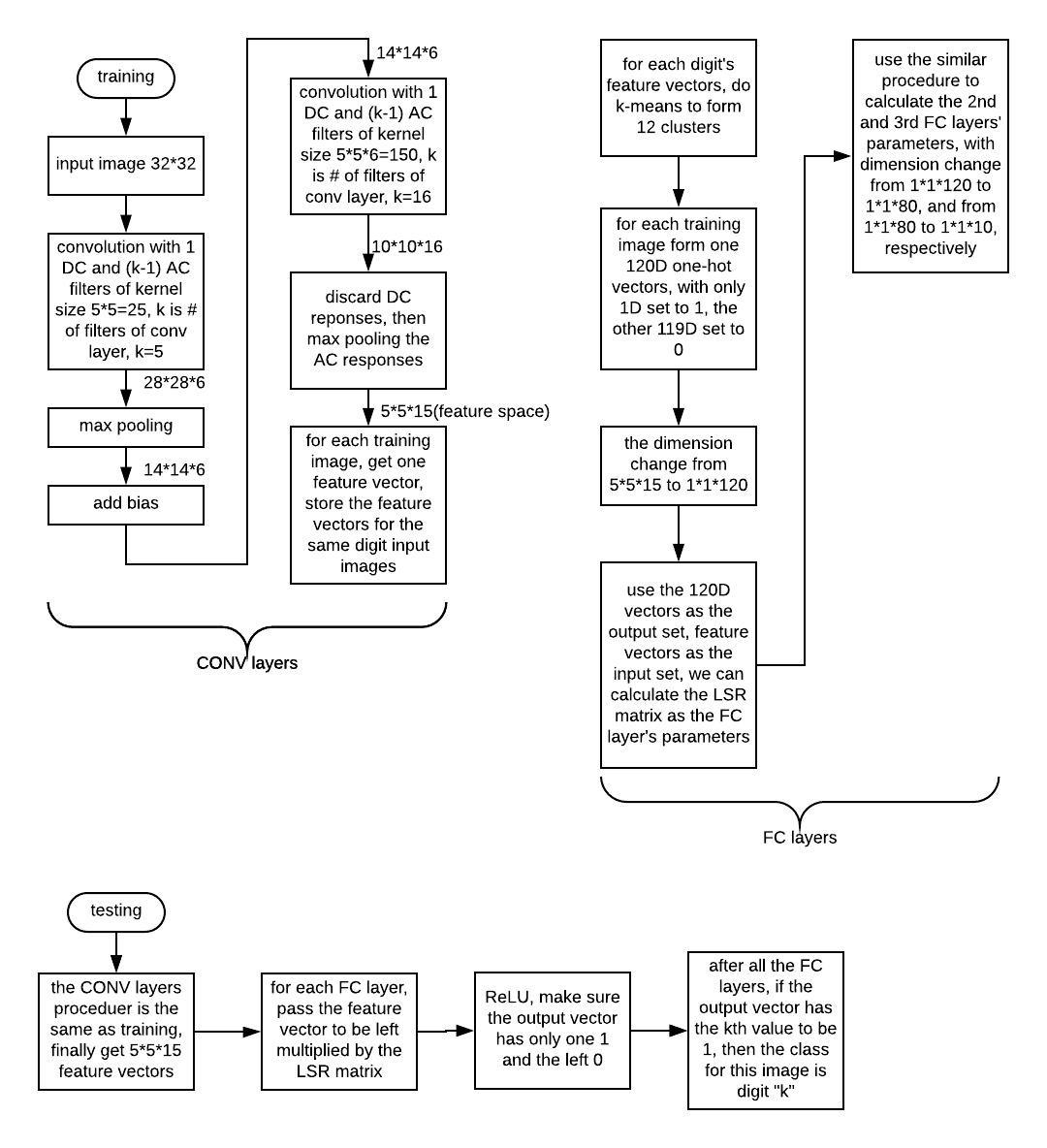


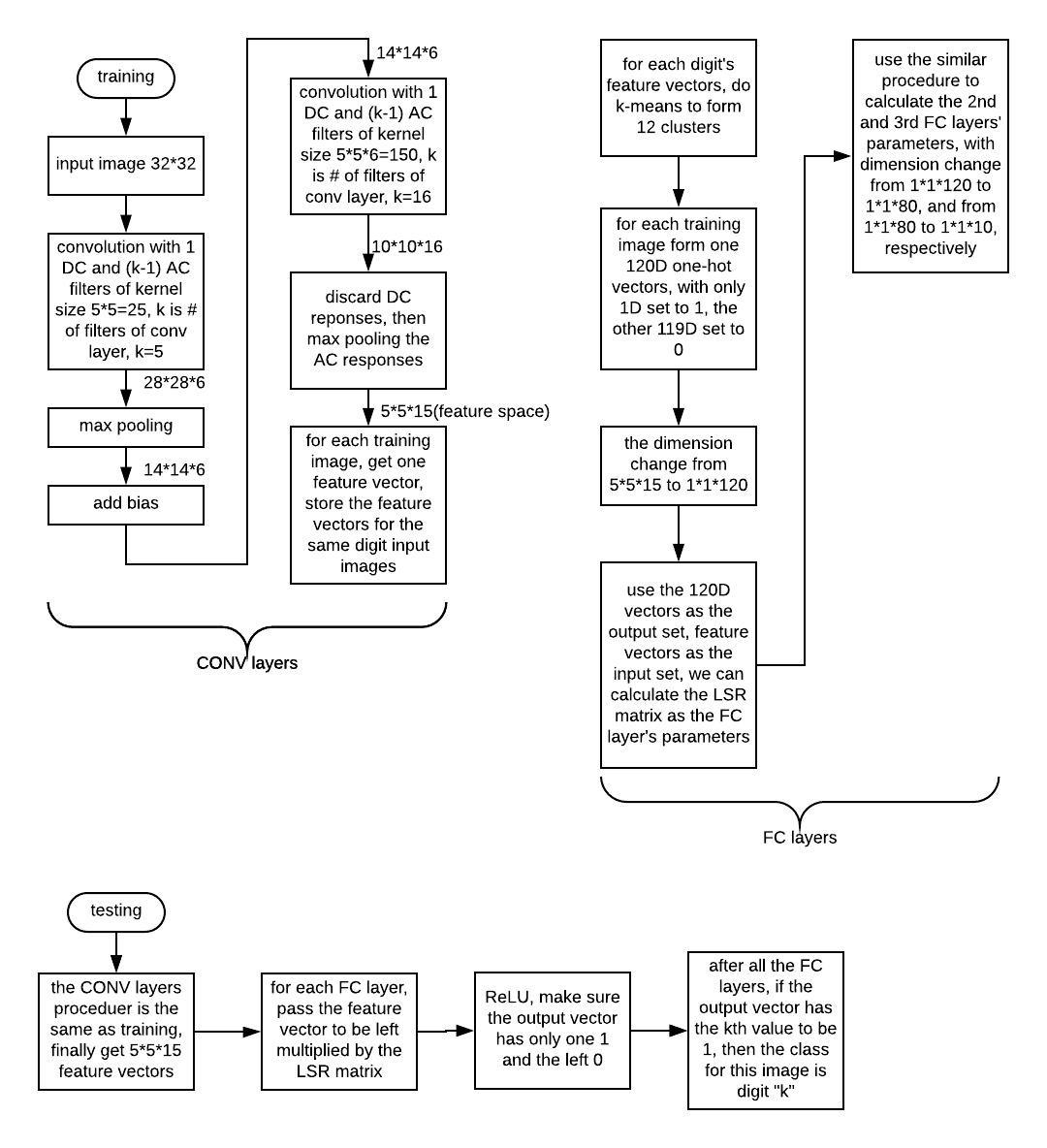
With a set of linear equations, we can solve the regressor’s parameters. The output y vector is the 120D vector, and the x vector is the feature vectors from the last convolution layer. The “w” is the bias.

The 2nd and 3rd FC layers have similar procedure.

During the testing procedure, we have to ReLU after each LSR to discard the negative output.

a) Flow chart:





b)

Similarities:

* The CNN procedure is similar with several CONV layers and FC layers, and also some activation function and max pooling layer.
* Both have patch filters in the CONV layer part.

Difference:

|  |  |
| --- | --- |
| BP | FF |
| Rely on stochastic gradient descent technique | Base more on linear algebra and statistics. Different mathematical tools. |
| Doesn’t have bias term | When compute the convolutional layers, add a bias term to make all the responses positive in order to fix the activation’s nonlinearity |
| System optimization centric (need lower cross entropy) | Different principle: Data-centric |
| Need to go backwards in training procedure to calculate the loss and adjust the CNN parameters | No need to go backwards when training |
| Need more time and space | Faster, need less memory space. Lower training complexity. |
| Have hidden layers which are hard to know the detailed mechanism inside it | Every layer can be mathematically explained very well. Higher interpretability. |
| Need labels | Independent from the label in CONV layer |
| Supervised model | Semi-(or un)supervised model |
| Lower robustness | Can ensemble several different FF CNN together to make the CNN more robust |
| Higher accuracy |  |
| Suitable for complicated (data set) problem. If used in simple problem, it may cause overfitting problem. | Suitable for simple problem |
| Still use convolution to get the FC layers’ output (when training) | Use k-means and one-hot vector to get the FC layers’ output |
| BP doesn’t have clear cut in different roles of CONV layers and FC layers. | Have modularity. Use PCA to help get filter kernels in CONV part.  Use LSR to get the FC layers’ parameters, this make the FC layer a continuous feature space alignment procedure. |
| Adjust the parameters after each forward training of all the images. End-to-end architecture. | Need to save all input images’(or of the same digit) the last convolution layers’ output to do the K-means. Finish calculating the parameters layer by layer. Because it can be separated layer by layer, so it can have more flexible architecture, like it can replace the FC layer with some other classifiers. |

**2) Image reconstructions from Saab coefficients**

If we want to reconstruct the original image from Saab coefficients, we need to do the inverse PCA.

PCA first calculate the eigenvectors of the covariance matrix of the input matrix. Then, sorts the eigenvectors by their eigenvalues. And we pick the first k (assume we need to reduce the dimension from p to k) eigenvector to be the kernels.

Let the input data be Xn×p (n rows with p features each row), k eigenvectors be Vp×k. The result of the PCA projection is Z = XV.

VVT is identity matrix, so we can reconstruct the input data X using ZVT (Xre = ZVT = XVVT)

In python, we can use “np.dot(pca.transform(Z)[:,:k], pca.components\_[:k,:])” or “pca.inverse\_transform(Z)” to reconstruct the X (may need to add the mean of X).

The reconstruction output and the corresponding PSNR value (N1, N2 are kernel # of two CONV layers):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | output | N1 | N2 | PSNR | # | output | N1 | N2 | PSNR |
| 1 |  | 5 | 15 | 17.63 | 19 |  | 5 | 15 | 16.71 |
| 2 |  | 5 | 50 | 21.33 | 20 |  | 5 | 50 | 20.11 |
| 3 |  | 10 | 15 | 17.92 | 21 |  | 10 | 15 | 16.91 |
| 4 |  | 10 | 50 | 22.86 | 22 |  | 10 | 50 | 20.83 |
| 5 |  | 10 | 100 | 27.82 | 23 |  | 10 | 100 | 25.20 |
| 6 |  | 10 | 150 | 30.95 | 24 |  | 10 | 150 | 27.86 |
| 7 |  | 15 | 50 | 22.95 | 25 |  | 15 | 50 | 20.80 |
| 8 |  | 15 | 100 | 28.35 | 26 |  | 15 | 100 | 26.07 |
| 9 |  | 15 | 150 | 32.58 | 27 |  | 15 | 150 | 30.02 |
| 10 |  | 5 | 15 | 15.64 | 28 |  | 5 | 15 | 16.25 |
| 11 |  | 5 | 50 | 19.44 | 29 |  | 5 | 50 | 20.67 |
| 12 |  | 10 | 15 | 15.73 | 30 |  | 10 | 15 | 16.44 |
| 13 |  | 10 | 50 | 20.76 | 31 |  | 10 | 50 | 22.28 |
| 14 |  | 10 | 100 | 25.43 | 32 |  | 10 | 100 | 26.57 |
| 15 |  | 10 | 150 | 28.54 | 33 |  | 10 | 150 | 28.42 |
| 16 |  | 15 | 50 | 20.85 | 34 |  | 15 | 50 | 22.71 |
| 17 |  | 15 | 100 | 25.70 | 35 |  | 15 | 100 | 27.84 |
| 18 |  | 15 | 150 | 30.25 | 36 |  | 15 | 150 | 30.44 |

From the table above, we can see if the kernel number is lager the output PSNR will be higher.

|  |  |  |
| --- | --- | --- |
| Compare picture # | Difference | conclusion |
| 2,4,7 | N1 # | Larger N1 # ==>larger PSNR  But only a small PSNRincrease |
| 12,13,14,15 | N2 # | Larger N2 # ==>larger PSNR  Big PSNR increase |
| 8,17,26,35 | Original image | With the same N1 and N2 #, the PSNR is quite similar |

Inverse PCA procedure will cause the loss of the original information. If the Kernel # is small, the inverse output will loss more original information.

Larger kernel # means that we can reserve more features of the original image. But this will cost more computing time and memory and may cause over-fitting. And we cannot let the Kernel # be too high, because the purpose of PCA is to get the main feature and reduce the data size.

**3) Handwritten digits recognition using ensembles of feedforward design (50%)**

training and testing classification accuracy for individual FF-CNN on the MNIST dataset:

|  |  |  |
| --- | --- | --- |
| Test # | training | testing |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

The accuracy of both training and testing are lower than the BP-CNN (regular accuracy is higher than 98%). One way to improve the accuracy of FF-CNN is to ensemble several FF-CNNs with different settings together to get the final class label. We can use different kernel size, randomly pick the feature values or use different color spaces and spectral space. When training, we combine the different output decision vectors (each setting has a 10\*1) together and put them into the SVM to get the parameters of the classifier.