EE569 Digital Image Processing: Homework #6

Name: Nazim N Shaikh
USC Id: 8711456229
Email Id: nshaikh@usc.edu
Submission Date: 04/28/2019

1) **Feedforward design convolutional neural networks (FF-CNNs):**
(Note: This section has no experimental results. All discussion questions have been included as part of Approach section)

Abstract and Motivation

- Convolutional Neural Networks (CNN) have proven effective in a wide range of applications involving image and/or video classification etc. They are being used in object detection, face recognition and have become increasingly powerful in helping self-driving cars figure out surrounding environment.
- However, how CNNs are so efficient at being able to perform such tasks, has been a long running question. Although, we know the math behind how network parameters get determined, for deep-networks this math is almost impossible to trace down.
- Over the years, various other methods have been examined to understand CNNs interpretability such as filter output visualization at various layers, purpose of using non-linear activation functions, knowledge representations, generative modelling etc.
- In this assignment, we work on understanding one novel method developed by our prof C.C Jay Kuo and his team in an effort to dig deep into CNN interpretability. This new design is called Interpretable Feedforward Convolutional Neural Networks. A special technique called Saab transform is used in cascade with Least Square Regressor for constructing layers of such network

Approach and Procedure

- Interpretable Feedforward Convolutional Neural Net is a data-centric based approach. Unlike Traditional CNNs, where network parameters are determined through an iterative backpropagation, here, the network parameters are determined by leveraging statistical knowledge behind input data. One of the main highlights of this method is that, all the processing happens in just single pass

- This methodology takes advantage of vector space terminology, in which images and its labels are viewed as high-dimensional vectors. An image classification task, is then considered as a sequence of vector space transformation for mapping some input in x dimension to some output in y dimension

- This entire transformation is performed through a sequence of operations explained below:
    1. Spatial-Spectral filtering

       This is the first step in the construction of convolutional layers. Here, the input image is divided spatially into a bunch of small windows using a filter patch of some size say 5x5. We consider a small patch since it gives more information than a single pixel. Also, for a small region feature comparison is translation and rotation invariant.
       For these series of patches, we then perform Principal Component Analysis to convert spatial representation into spectral representations by projecting window patches into PCA-based kernels. The gives us dimensions with most discriminant power.

    2. Saab Transform

       Once we have PCA-based kernels, the next transformation to be applied is called Subspace Approximation with adjusted bias. Saab transform is variant of PCA which helps in dimension reduction. It is a way of selecting weight and bias vector. To apply this, input needs to be a combination of 2 kernels – AC and DC kernels. AC component is basically PCA based kernels obtained from previous step. DC component is the orthogonal complement of AC and is obtained by projecting input data onto DC anchor vectors, calculated as below

$$X_{DC} = \frac{\sum_{n=0}^{N-1} X_n}{\sqrt{N}}$$

       Next, we consider adjusted bias term. This is added to solve the sign confusion problem resulting from non-linear activation function.

In traditional CNN, the output of previous layer can have positive or negative weights. As a result of this, the second layer is unable to differentiate between 2 similar yet different patterns (for example, 3 vertical stripes with black in between and vice-versa). This gives rises multiple scenarios with same output, such as,

a) positive output with negative incoming weights and
b) negative output with positive incoming weights.

Addition of RELU activation allows only positive output to get passed. However, there is a problem of doubling of spectral dimension. This is the reason for adding constant bias vector to transformed output. This leads to 2 criteria in bias selection

a. Bias is selected in such a way that response of previous layer is always non-negative
b. All bias terms are kept constant

In order to satisfy above, the bias term should meet following constraint:

$$b_k \geq \max\left||x|\right|, \qquad k = 0,1, \ldots . K-1$$

The max operation gives output similar to what RELU would give. The bais vector lies always in DC subspace to give shift in output response by some constant amount

3. <u>Max Pooling</u>

Similar to traditional CNN, pooling here is added to reduce spatial dimension and capture visually similar patterns. The operation is similar to filtering. The operation is considered as projection of patch of size say $32x32$ to a smaller subspace, usually half the original i.e $16x16$. Here, the output is the most significant pattern in that particular filter region.

This ends convolutional layer construction.

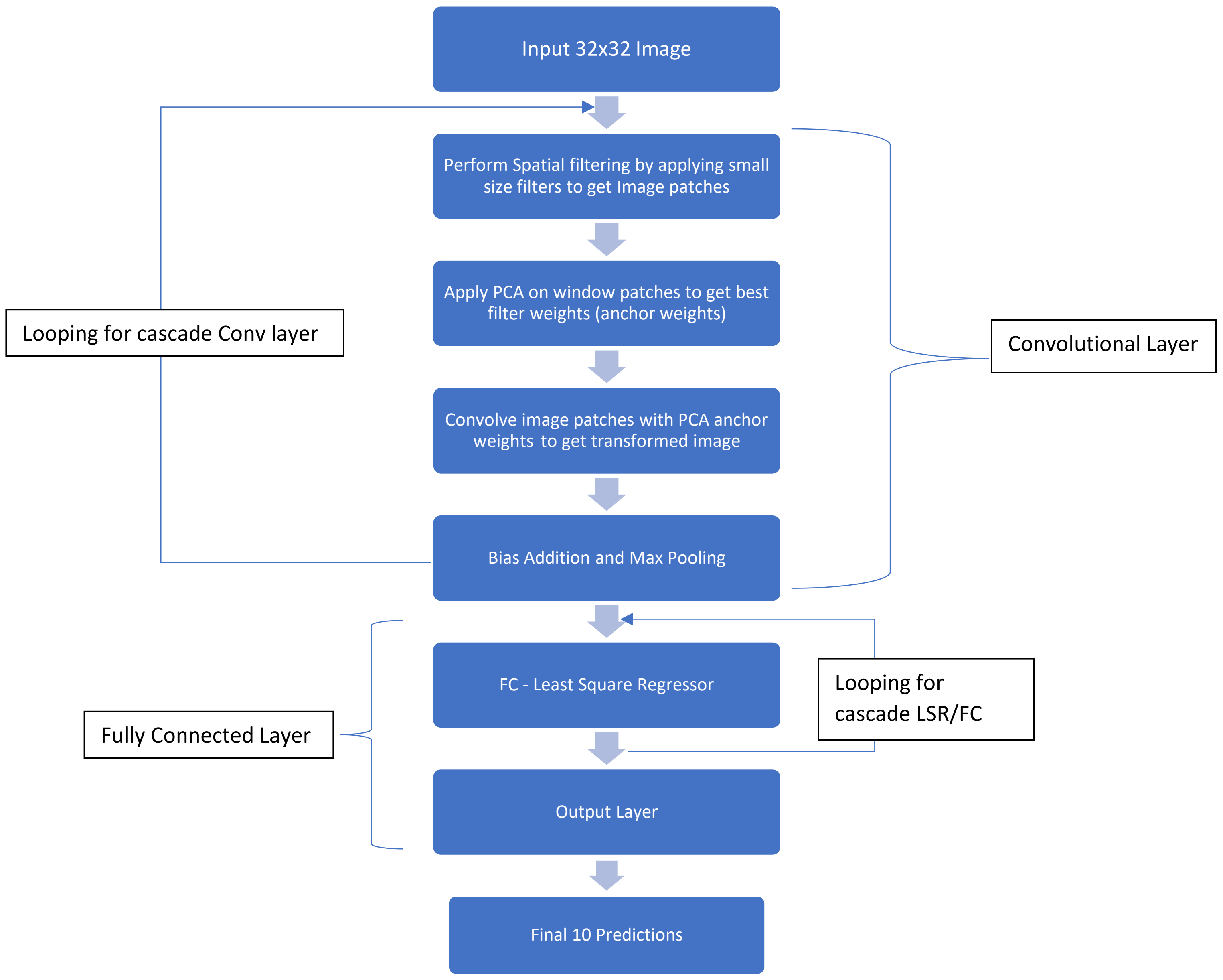
Similar to LeNet5, convolution layers are cascaded to result in more discriminant image features. Next, we proceed with Fully Connected Architecture

4. <u>FC layer</u>

   Unlike traditional CNN, here, the FC layer is constructed using sequence of least square regressors. Output of each layer is a one-hot vector. Until now, no labelling was needed, however for FC layers we need label for each input. This is achieved using K-Means clustering. The output of convolutional layer is grouped into K clusters where K being number of filters of that particular FC layer. The input samples of same class /digit are usually evenly divided into $\frac{K}{\#\, real\ class}$ clusters to generate labels. The class and cluster labels are combined to create output labels. This are also known as pseudo labels.
   This is done in order to the diversity of a single class with more samples

   Multiple LSRs are connected in cascade to get better performance, with the last LSR having final 10D desired output labels.

   The entire transformation process can be shown in a flowchart as below:


Input 32x32 Image
Perform Spatial filtering by applying small size filters to get Image patches
Apply PCA on window patches to get best filter weights (anchor weights)
Convolve image patches with PCA anchor weights to get transformed image
Bias Addition and Max Pooling
FC - Least Square Regressor
Output Layer
Final 10 Predictions
Looping for cascade Conv layer
Convolutional Layer
Fully Connected Layer
Looping for cascade LSR/FC

Training Phase:

- In training phase, we first generate PCA kernels as below:

  For each stage of conv layer:

  a) For input image, generate window patches (overlapping or non-overlapping)
  b) Subtract Feature mean as well as patch mean from generated image patches in order to center input image.
  c) Perform PCA on patches to get AC kernel. Number of components in PCA is equal to number of filters for each layer.
  d) DC kernel is computed using equation I mentioned above
  e) Combine AC and DC kernels to get PCA kernel weight matrix
  f) Add bias to input image to avoid for non-linearity
  g) Apply PCA kernel to input images to generate transformed images
  h) Apply max Pooling to get final Saab coefficients

- Once we have PCA kernels, we then apply trained PCA kernels to train 60K images through steps mentioned in first half of flowchart. This will give us best feature matrices

- Next, For each stage of FC layer:
  a) Take in best features obtained from conv layer and cluster the input into some specific clusters which is equivalent to number of filters of that particular layer using K means clustering. This will generate cluster labels
  b) Generate Pseudo labels by combining class labels and cluster labels which are one hot encoded
  c) Perform Least Square Regression to generate output labels
  d) Save weights and biases at each stage

  The output of final layer is 10D prediction vector

Testing Phase:

- For testing, we generate best test features or (Saab coefficients) by applying trained PCA kernels to test images through steps mentioned in first half of flowchart.

- Pass obtained test features through each FC layer to least square regression. Here we used trained weights and biases to apply LSR.
- The final output is 10D test prediction vector
- The index of highest value in prediction vector is the final classification label for that particular image

FFCNN and BP-CNN comparison

- The 2 CNN designs can be compared based on different properties as below:

1. Principle: BP-CNN generates best features by optimizing some particular cost function, while FC-CNN uses input data statistics. In BP-CNN, convolution is performed through matched filtering. FF-CNN uses principal component projection. Features extracted are less discriminant compared to BP-CNN design. Unlike BP-CNN, FC-CNN avoids use of non-linear function at convolutional layers by incorporating additional bias term. BP-CNN requires labelled data since the start. No data labelling is needed for FC-CNN until convolutional layer stage. FC layer labels are generated using k-means clustering approach.
2. Modularity: BP-CNN design is end-to-end while FF design is divided into 2 modules – convolutional network and fully connected work. Convolutional network is unsupervised while FC network is supervised. Additionally, since FC network does classification task, it can be replaced with SVM or Random Forest Classifiers.
3. Architecture: Both BP-CNN and FC-CNN uses same concept of convolution-FC type architecture (although approach is different). Max Pooling is used in both for dimension reduction and to get visually dominant features
4. Training complexity: BP-CNN is an iterative approach while FC-CNN is a single pass methodology. Training can be done on small set of 10K images. This makes FF- CNN design training faster than BP-CNN
5. Performance: BP-CNN achieves state of the art performance with proper choice of network and optimization function. This is better than current FF-CNN design, although more work and evaluation is yet to come for FF design.

6. Interpretability: BP CNN design is difficult to interpret as far as math's is concerned. However, as seen from explanation in previous section, FF designs are mathematically transparent
7. Generalizability: In case of BP design, we need to build different CNN networks based on tasks such as segmentation, classification, object detection, etc. However, in case of FF design, same conv layer can be used for different tasks and only FC layer can be changed accordingly. FF design also allows fusing of features obtained through other techniques such as laws filtering, SIFT, etc.
8. Robustness: As proven in the paper, both the designs are vulnerable to adversarial attacks.
9. Ensemble Design: Th ensemble of FF-CNN is proved to be giving better results. Also, the computation cost of building ensemble for BP-CNN is too high due to iterative optimization process

2) **Image reconstruction from Saab coefficients:**

Abstract and Motivation

10. As learned from previous section, the convolutional layers of FF-CNN design are constructed using subspace approximation followed by max pooling. Subspace approximation is achieved using Saab Transform. The output of each convolutional transform is called Saab coefficients.
11. Image reconstruction is one way to check how your convolutional layer performs with given settings and further help in fine tuning the same. This can be done using Saab coefficients to get back original image.

Approach and Procedure

12. Here, we consider 2-stage Saab transform with filter size as (4,4), stride of 4 (non-overlapping) in both stages. Number of filters in both stages are free to choose. We

start with N1 = 10 and N2 = 40. For sake of simplicity of construction and reconstruction we skip max-pooling operation.

13. Initially, train a set of 10K image samples using Getkernel.py with above settings to generate PCA kernel params (DC and AC anchor weights). Getkernel.py implements convolutional layers construction with PCA training. Once we have trained PCA kernel weights for both stages, the construction of Saab coefficients can be explained as below.

For each stage:

1. Create image patches for input image using non-overlapping windows with configurations mentioned above
2. Remove feature mean and patch mean from image patches. (Save feature mean in for purpose of reconstruction)
3. We add bias to image patches in order to annihilate non-linearity. (Save bias for purpose of reconstruction
4. Apply PCA kernel to mean removed image patches to get transformed image - this step is Saab transform. The output is the required Saab coefficients
5. We discard added dc bias since it is constant element vector relying in DC space and does not contribute further in feature extraction.

14. The reconstruction starts from last stage and is explained as follows:

1. From the Saab coefficients obtained at the last stage, we first add bias to compensate for bias removal at the end of last stage.
2. Apply inverse PCA kernel to this output to get back image patches.
3. Add back saved feature mean and patch mean to the image patches.
4. Revert the window process applied to image patches in order to get Saab coefficients for previous stage.
5. Repeat above steps until we reach first stage, the output of which would be the reconstructed input image
6. Finally, we calculate PSNR between reconstructed and original image to evaluate the reconstructed result.

Experimental Results

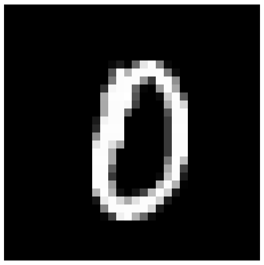
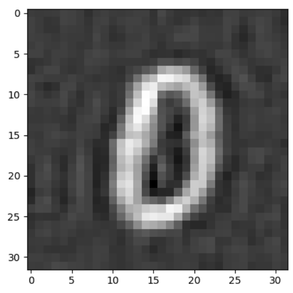
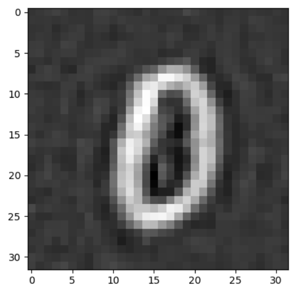
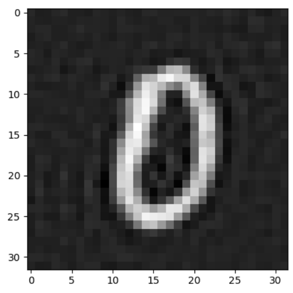
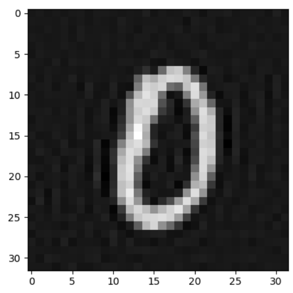
 Figure 1 Original Digit 0	
 Figure 2 Reconstruct Digit 0 (N1 =10, N2 = 40)	 Figure 3 Reconstruct Digit 0 (N1 =15, N2 = 45)
 Figure 4 Reconstruct Digit 0 (N1 =10, N2 = 80)	 Figure 5 Reconstruct Digit 0 (N1 =15, N2 = 120)

Table 1 Reconstruction results for Digit 0

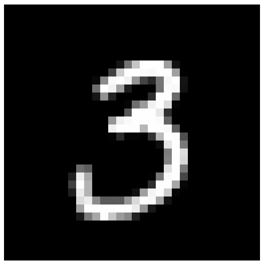
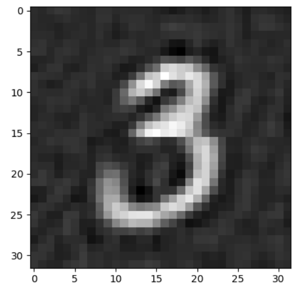
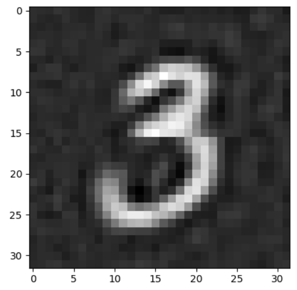
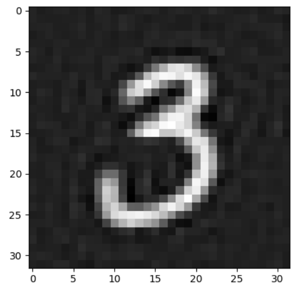
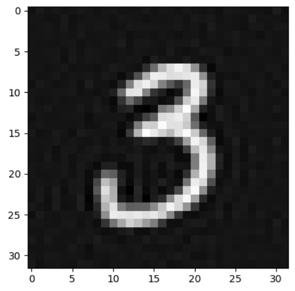
 *Figure 6 Original Digit 3*	
 *Figure 7 Reconstruct Digit 3 (N1 =10, N2 = 40)*	 *Figure 8 Reconstruct Digit 3 (N1 =15, N2 = 45)*
 *Figure 9 Reconstruct Digit 3 (N1 =10, N2 = 80)*	 *Figure 10 Reconstruct Digit 3 (N1 =15, N2 = 120)*

Table 2 Reconstruction results for Digit 3

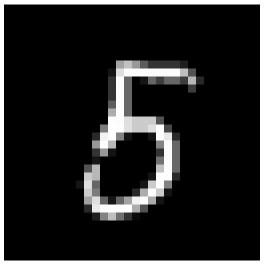
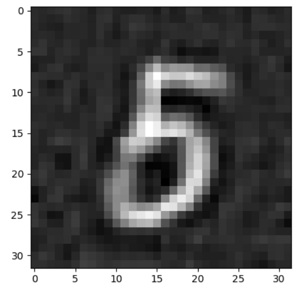
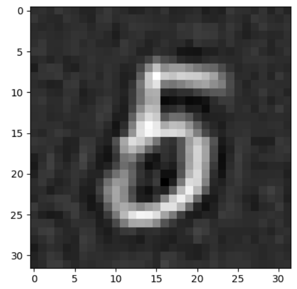
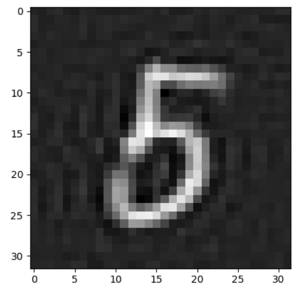
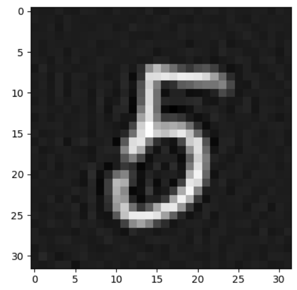
 *Figure 11 Original Digit 5*	
 *Figure 12 Reconstruct Digit 5 (N1 =10, N2 = 40)*	 *Figure 13 Reconstruct Digit 5 (N1 =15, N2 = 45)*
 *Figure 14 Reconstruct Digit 5 (N1 =10, N2 = 80)*	 *Figure 15 Reconstruct Digit 5 (N1 =15, N2 = 120)*

Table 3 Reconstruction results for Digit 5

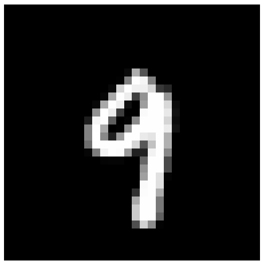


Figure 16 Original Digit 9

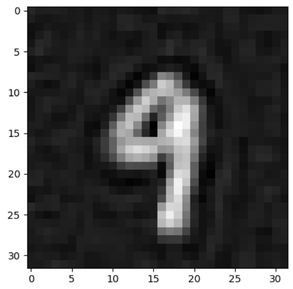


Figure 17 Reconstruct Digit 9 (N1 =10, N2 = 40)

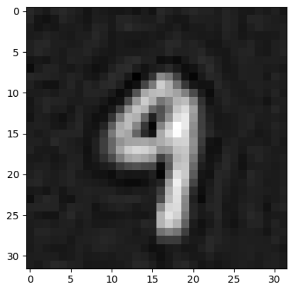


Figure 18 Reconstruct Digit 9 (N1 =15, N2 = 45)

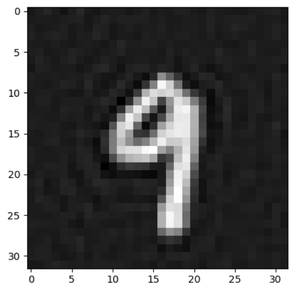


Figure 19 Reconstruct Digit 9 (N1 =10, N2 = 80)

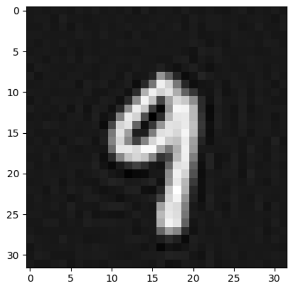


Figure 20 Reconstruct Digit 9 (N1 =15, N2 = 120)

Table 4 Reconstruction results for Digit 9

PSNR Values

	Setting 1 (10/40)	**Setting 2** (15/45)	**Setting 3** (10/80)	**Setting 4** (15/120)
PSNR for digit 0	20.09	20.81	24.28	28.06
PSNR for digit 3	21.66	22.25	25.81	29.55
PSNR for digit 5	20.11	20.63	23.35	27.92
PSNR for digit 9	22.28	22.61	26.74	29.69

Table 5 PSNR values for test images on different settings

Discussion

15. Reconstruction was performed on a set of 4 MNIST images using varied number of filters in both stages and PSNR values was calculated between original and reconstructed image. High PSNR value is an indication of good reconstruction
16. The output for different MNIST digits is shown in table 1 - 4. N1, N2 indicates number of filters in 1st stage and 2nd stage respectively.
17. Consider PSNR values (table 5) of first setting, we see that the values are fairly fine (not so high and not so low as well). The reconstruction is kind of blur. This implies that Saab transform is a lossy process. This is expected as the Saab transform involves dimension reduction through PCA. Although PCA selects best features, the features extracted might not have enough discriminative power
18. In Setting 2 we Increase number of filters in first stage. This gives a little improvement in the reconstruction results. This is reasonable as early stages are only responsible for extracting low level features. This implies even though we try to increase filter size in 1st stage, the improvement won't be significant.
19. In Setting 3 and 4, we try increasing the number of filters in second stage. From the output, we observe that reconstruction gets much better with less blurring. This is also evident from PSNR values which are fairly good compared to initial settings.
20. This is because (1) 2nd stage helps capturing high level features (2) increase in filters allows us to capture more features across spectral dimension resulting in increase in Saab coefficients which are responsible for reconstruction

21. Filter Settings with best PSNR value indicates that convolutional layer construction chosen with those filter numbers results in best feature extraction, further helping in accurate classification

22. Reconstruction can further be improved by increasing filter size or receptive field while performing Saab transform. The result of increasing filter size is increase in feature discriminant power

3) **Handwritten digits recognition using ensembles of feedforward design:**

Abstract and Motivation

23. Ensemble design is a machine learning technique of combining several baseline models in order to produce optimal model. Methods such as bagging, boosting, random forest, stacking etc. are examples of ensemble techniques. Ensemble technique helps improving overall system predictions and/or also at times decrease any bias-variance.
24. The performance of ensemble design highly depends on the choice of different settings used in generating various baseline models.
25. Here, we apply FF-CNN design to classify handwritten digits using LeNeT5 architecture configurations and improve the performance using ensemble systems of 10 FF-CNNs.

Approach and Procedure

- Ensemble approach is just a combination of individual predictions obtained through different FF-CNNs settings to get best predictions. The approach is as follow:

  1. Train FF-CNN individually with 10 different settings on MNIST dataset in order to learn PCA kernels and LSR weights

2. Save 10D train prediction vector along with learned PCA kernels and LSR weights from each setting.
3. Now, test each of FF-CNN on test labels using learned kernels and LSR weights to generate 10D test prediction vector. The output shape for each setting will be10000 x 10
4. Concatenate all ten 10D train prediction vectors to get new train feature vectors.
5. Perform PCA on combined train predictions to get best 10D train feature vector. We apply PCA with no. of components as 10.
6. Similarly, concatenate all ten 10D test predictions to get new test feature vectors.
7. Apply PCA on this to get best 10D test feature vector.
8. Train an ensemble classifier such as a support vector machine or any basic classifier with input as 10D train feature vector and output label as the original train labels
9. Perform prediction on 10D test feature vector
10. Calculate test accuracy
11. Generate Confusion matrix plot

Experimental Results

	Filter size (5,5)	**Filter size** (5,3)	**Filter size** (3,3)	**Filter size** (3,3)	**L1**	**L2**	**L3**	**L4**	**L5**	**L6**	**ED**	**BP CNN**
Train Acc	96.98	97.11	97.11	96.73	96.77	96.47	96.39	96.32	96.99	96.67	98.17	99.69
Test Acc	97.11	96.96	97.17	96.72	96.84	96.52	96.54	96.43	96.97	96.54	98.14	99.08

Table 6 Comparison of Classification accuracy using different filter sizes and laws filters. L1-L6 denotes filtered maps by L5E5, L5S5, L5W5, L5R5, E5S5, S5R5, ED indicates ensemble results

Confusion Matrix

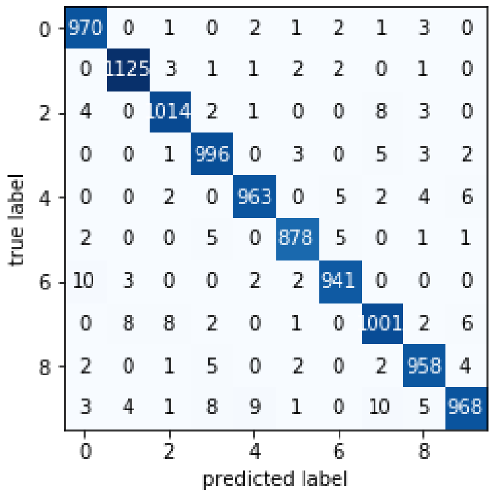


Figure 21 Ensemble FF-CNN

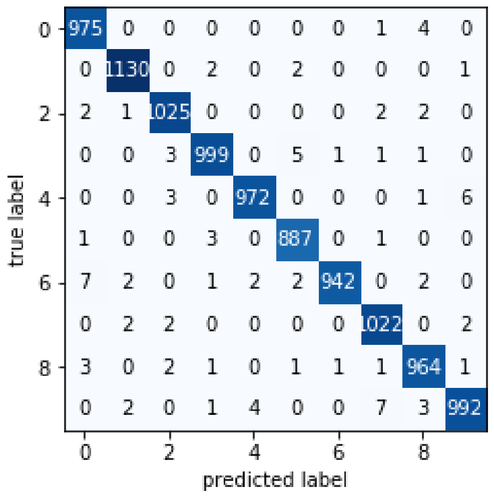


Figure 22 BP-CNN

Discussion

- Ensemble FF-CNN was built using 10 different settings. Since diversity is an important factor contributing to success of ensemble model, 2 different diversity types were fused in an effort to build better ensemble system.
- First diversity type involved using filter of varied sizes in both stages of convolutional layer - (5,5), (3,5), (5,3), (3,3). Different filter sizes help in yielding different features at convolutional layer output. This contributed to first 4 settings
- Second diversity type involved changing input to FF-CNN by application of laws filters. 6 laws filters (L5E5, L5S5, L5W5, L5R5, E5S5, S5R5) of size 5x5 were applied to input images to generate images with different spectral characteristics. This contributed for remaining 6 settings
- The training and testing accuracy for individual as well as ensemble FF-CNN and BP-CNN is shown in table 6. BP-CNN was evaluated using best settings obtained in homework 5
- From the output, we see that ensemble design results in approximately 1-1.5% increase in the train as well as test accuracy compared to individual ones.

Classification Error Analysis

- **a.** To understand classification error between BP-CNN and Ensemble FF-CNN, we plot confusion matrix for test data - shown in figure 21 and 22
- **b.** From the plot, diagonal values represent number of correct predictions in each class and off diagonal values represents number of incorrect predictions in each class. High diagonal values indicate good classification
- **c.** Classification error can be divided into cases - False Positive (Type I error) and False Negative (Type II error). Type I error is the number of samples which are incorrect but have been predicted as correct. So, False Positive Rate = False Positives / (True Negatives + False Positives). Type II error is the number of samples which are true but have been predicted incorrect. So, False Negative Rate = False Negatives / (True Positives + False Negatives)
- **d.** To understand this, let's take example of classifying digit as 0. This is represented by first row and column of confusion matrix. We need to look at off diagonal values. Off diagonal values along row represent False Negatives and off diagonal values along column represents False Positive.
- **e.** Consider confusion matrix of Ensemble FF-CNN. From first row, total False Negatives = 10. Similarly, False Positives = 21. So False Negative Rate $=10 / (970 + 10)$ ~ 0.01 and False Positive Rate = $21/(970 + 21)$ ~ 0.021. Total error rate is 0.031.
- **f.** Similarly, for BP-CNN, False Negative Rate is ~ 0.005 and False Positive Rate is ~ 0.013. Total error rate is 0.018.
- **g.** We see that FPR is almost same for BP-CNN and Ensemble FC-CNN, however, there is slight .5% difference in False Negative Rate while classifying digit 0
- **h.** Although, at individual class levels these errors are little different, on an average I conclude the errors would be almost same and total error would be low. This implies that both models perform well.

Performance Improvement

- One way to improve performance of BP CNN model would be to consider deeper networks. However, as network gets deeper, the number of parameters increases which might result in overfitting. So, additionally we perform data augmentation to improve data size to parameter ratio.

- Improvement of FF-CNN can be done on convolutional layer as well as FC section. As part of convolution layer, we can use Image reconstruction algorithm as a cross validation technique in which we run construction and reconstruction for different filter settings on a small set of 10K images and the find the best filter settings based on PSNR value.
- We can then train the convolutional layer using these filter settings before passing output to FC Layer.
- Since FC layer performs the task for classification, we can replace LSR with much better algorithms such as SVM or Random Forest.
- Additionally, we can perform FC layer classification task using back propagation technique as it proven to give optimized results

References

[1] Kuo, C. C. J., Zhang, M., Li, S., Duan, J., & Chen, Y. (2019). Interpretable convolutional neural networks via feedforward design. Journal of Visual Communication and Image Representation.

[2] https://github.com/davidsonic/Interpretable_CNNs_via_Feedforward_Design (Most code files provided in this repo has been used as it as for 3rd part – further instruction in readme)

[3] Chen, Y., Yang, Y., Wang, W., & Kuo, C. C. J. (2019). Ensembles of feedforward-designed convolutional neural networks. arXiv preprint arXiv:1901.02154.

[4] Discussion Slides and Class notes