EE569: Introduction to Digital Image Processing

ASSIGNMENT 6

FEED FORWARD DESIGN NEURAL NETWORKS

Suchismita Sahu | 28th April 2019 | USCID: 7688176370

PROBLEM: FEEDFORWARD-DESIGNED CNNs

ABSTRACT AND MOTIVATION

As we know, in a traditional CNN, we determine the network parameters as a solution to non-convex optimization problem, solving using back-propagation. Also, non-convex optimization of deep networks is intractable mathematically, making it challenging to provide an end-to-end analysis of the working principle of deep CNNs. Thus, we need to develop an interpretable feedforward design CNN.

UNDERSTANDING OF FEEDFORWARD-DESIGNED CNNs

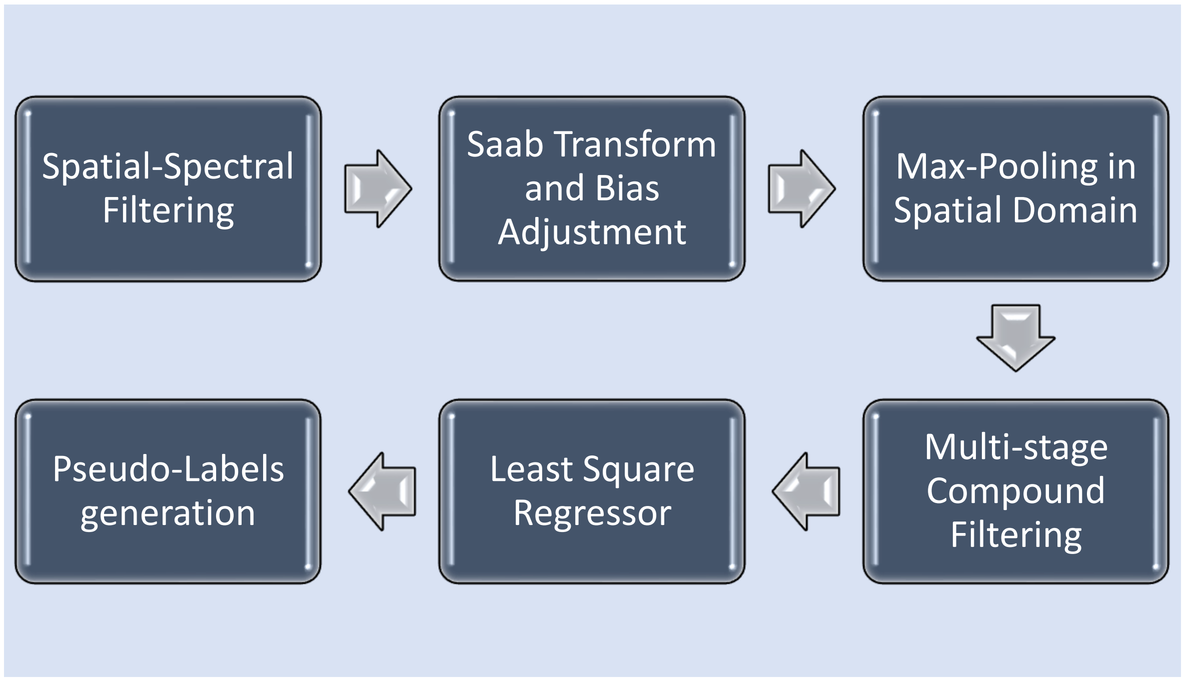
Feedforward-designed Convolutional Neural Networks use a data-centric approach for training instead of the traditional Back-propagation training method of training CNNs. The network parameters of a layer are calculated based on the data statistics from the output obtained from the previous layer in a single-pass manner. The convolutional layers are developed using Saab (Subspace approximation with adjusted bias) transform signal transform technique which is a variant of the principal component analysis (PCA) technique. Saab transform technique has an added bias vector on top of PCA to annihilate the nonlinearity of the activation. Hence, convolutional layers are defined by cascading multiple Saab transforms. Also, Fully-Connected (FC) layers are constructed by cascading multi-stage linear least squared regressors (LSRs).

So, overall, FF-CNN has two modules:

1. Convolutional layers constructed by using the Saab transforms
    a. Spatial-Spectral Filtering
    b. Saab Transform and Bias Adjustment
    c. Max-Pooling in Spatial Domain
    d. Multi-stage Compound Filtering
2. Fully-Connected layers using the multi-stage linear Least Square Regressor

a. Least Square Regressor
b. Pseudo-Labels Generation

Flow Chart of FF-CNN with the two modules:


Spatial-Spectral Filtering
Saab Transform and Bias Adjustment
Max-Pooling in Spatial Domain
Multi-stage Compound Filtering
Least Square Regressor
Pseudo-Labels generation

For achieving optimal subspace approximation, Saab transform selects best eigenvectors from the set of transform kernels of covariance matrix. When we cascade multiple stages of saab transform, we must use ReLu activation function to limit the confusion of signs that arises in the stages. Hence, due to the non-linearity, we face information loss that we tackle by adjusting the bias. We save both the original and adjusted bias terms after every stage.

Multi-stage saab transform is able to provide a full spectral and spatial domain representation. We use the extracted saab coefficients as training and testing features into any classifier, for example SVM. Important features are selected after performing PCA making saab transform robust to small perturbation.

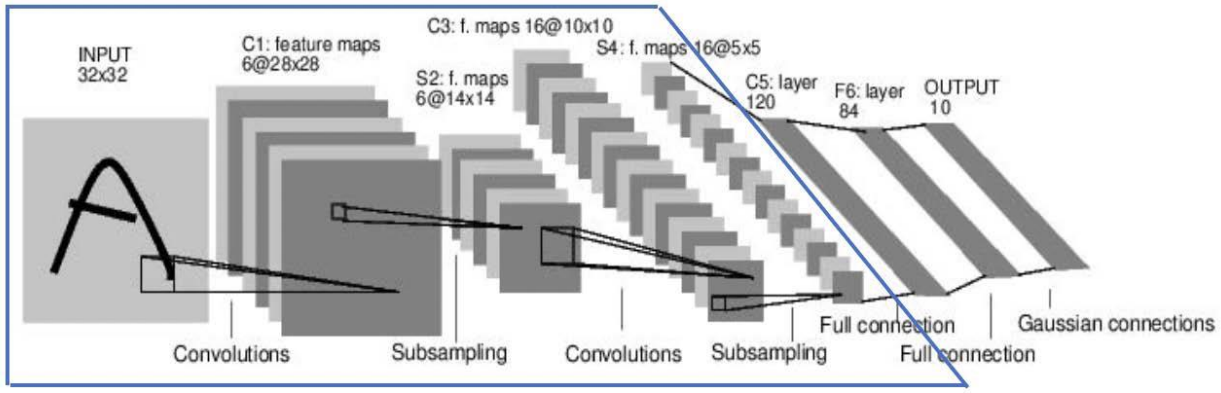


Figure 1 FF-CNN Convolutional layer Module

Spatial-Spectral Filtering:

Spatial-spectral filtering when cascaded with pooling, provides an effective way to extract the important discriminant dimensions. Since here, we consider the neighborhood of each pixel to find it's spatial representation, we get robustness to rotation and translation. Also, implementing PCA on the features helps us to find the dominant stroke patterns. Also, PCA output helps us represent a pattern and it's neighborhood pixels as a linear combination of those dominant patterns. Therefore, the convolutional layers transform an image into it's spatial-spectral form. As we implement the transform in a non-overlapping window, we get richer feature set.

Saab Transform and Bias Selection:

The saab transform selects bias and anchor vector terms for affine transform as follows:

$$y_k = \sum_{n=0}^{N-1} a_{k,n} x_n + b_k = \mathbf{a}_k^T \mathbf{x} + b_k, \quad k = 0, 1, \cdots, K-1,$$

The anchor vectors are divided into two categories that sum up to represent the input vector space.

- DC anchor vector $\mathbf{a}_0 = \frac{1}{\sqrt{N}}(1, \cdots, 1)^T$.
- AC anchor vectors $\mathbf{a}_k$, $k = 1, \cdots K-1$.

For any vector x, we can represent AC component as:

$$\mathbf{x}_{AC} = \mathbf{x} - \mathbf{x}_{DC}.$$

Also, DC component is:

$$\mathbf{x}_{DC} = \mathbf{x}^T \mathbf{a}_0 = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n.$$

Bias selection is done considering the two constraints:

1. Bias bk is chosen such that kth response is a non-negative value: Positive response constraint
2. All bias terms are equal: Constant Bias constraint

Mathematically expressed as :

- None negative response constraint $y_k = \sum_{n=0}^{N-1} a_{k,n} x_n + b_k = \mathbf{a}_k^T \mathbf{x} + b_k \geq 0,$
- Constant bias constraint $b_1 = b_2 = \cdots = b_K \equiv d\sqrt{K}.$

Because of the above, the output is same with or without using ReLu function.

<u>Max-Pooling in Spatial Domain:</u>

We implement pooling to implement reduction in storage and computation because it basically filters out insignificant patterns and preserves the significant patterns. In max-pooling, small windows are identified and the window is reduced to a single value that is the maximum value of the window.

<u>Multi-Stage Compound Filtering:</u>

On cascading multiple Saab transform convolutional layers we obtain a rich set of features called compound filters. Target pattern is a linear combination of PCA component eigen vectors that represent the receptive field.

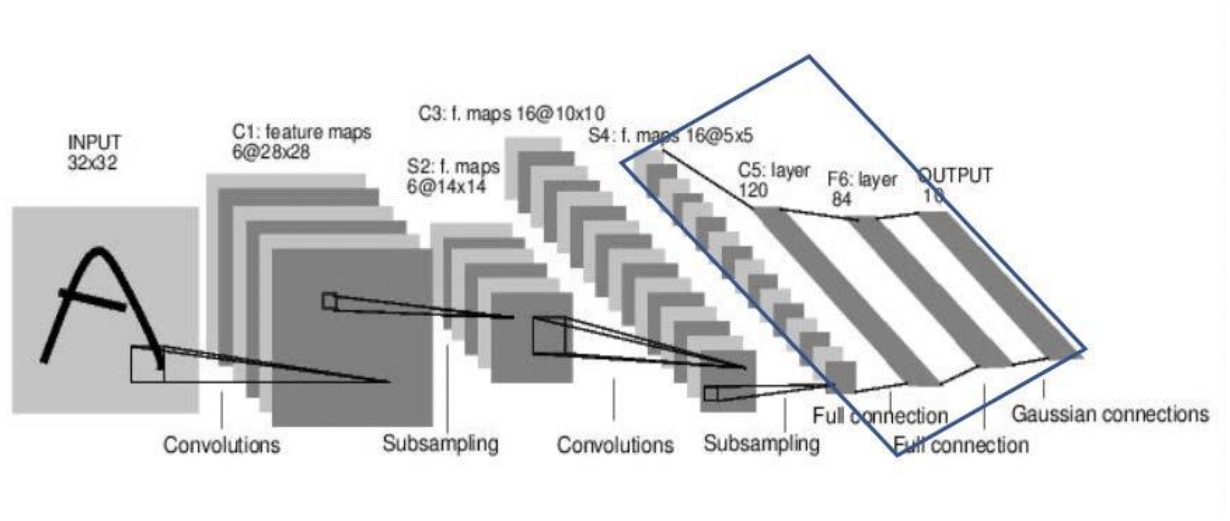


Figure 2 FF-CNN FC layer Module

Least Squared Regressor (LSR) and Pseudo-Labels Generation

LSR represents a FC layer and it's output is a one-hot vector denoting the label of the digit to be classified. Also, k-means clustering is performed on the input to this layer and the number of clusters is the number of filters in the FC layer output nodes. Then each input has the new cluster labels as well as the original class label. These are combined for obtaining pseudo-labels. These pseudo categories capture the diversity of a single class. We implement ReLu after each LSR to finally transform the samples to final decision space gradually.

Similarities and Differences between FF-CNNs and BP-CNNs:

1. **Overall Principle of Operation:**
   A BP-CNN can be divided overall into 3 modules: Data (Test and Train), Network Architecture (Conv Layers, Pooling Layers, Activation Function, Filters, Batch Normalization etc.) and Loss Function (Cost function to be optimized). The process then follows back-propagation to train the network weights and establish the important features for classification. Whereas, FF-CNN uses spatial-spectral transformations using covariance matrices to build the initial convolutional layer module. Also, it uses multi-stage LSR models to build the FC layer which needs the data labels.
2. **Basic Mathematical Principle:**
   In BP-CNN, we implement stochastic gradient descent algorithm to optimize the pre-defined cost-function of the network. Whereas, FF-

CNN uses basic linear algebra and statistics, and hence is more interpretable.

3. **Interpretability:**
   BP-CNN is basically a black-box like many Deep Neural Network Architectures whose performance cannot be interpreted in simple mathematical manner. Like basic Deep Neural Networks, the performance cannot be measured with respect to the parameters exactly to understand the effect of each parameter towards the performance of the network. In contrast, FF-CNN is mathematically simple to interpret.
4. **Modularity, Model Design and Architectural Constraints:**
   In BP-CNN, the architecture depends on the input data and the output labels and the parameters of the initial layers are more dependent on the input data and weights of the deep layers are more dependent on the output data labels. It's an end-to-end Model in whole. Whereas the FF-CNN converts this network into two modules – convolutional layers module and the FC module which take care of the feature extraction and classification respectively. Also, Design of BP-CNN depends on the data input and output labels and involves an end-to-end sequential model having connected layers and cost function. But, the design of FF-CNN has no architectural constraints and the FC layer can be replaced by any classifier for example SVM or Random Forest classifier. Classification accuracy of FF-CNN is based on the first block.
5. **Performance:**
   The performance of BP-CNN is state of the art when we chose the cost function and architecture properly and relevant to the performance metric otherwise we cannot guarantee high performance. FF-CNN is still being explored and using the ensemble method might help to boost performance. Also, efficiency of saab transform is better with minimal data for training.
6. **Robustness:**
   CNNs are sensitive to adversarial attacks when the model is fixed. Thus the ensemble model for FF-CNN works better.
7. **Complexity of Network:**

BP-CNN is an iterative optimization process in which we process all training samples in each training epoch and we must train the network for many epochs until the network converges. Whereas FF-CNN is basically a statistical method and the complexity can be reduced directly by performing PCA and LSR on a smaller set of data. FF-CNN is computationally efficient.

8. **Network Generalizability:**
   The data space and the decision space are strongly related in a BP-CNN model and hence as our data changes, so does our network architecture. For example, we use different CNN architectures for different tasks even with same data. Whereas in FF-CNN we can share the same convolutional layer module for different tasks and then design the Fully connected module considering the different tasks.
9. **Design Flexibility:**
   For any changes in the input data, since BP-CNN has learnt weights, it needs to be trained all over again. But for FF-CNN design using saab coefficients, we have more robustness for small changes in input data as they don't affect the last-stage saab coefficients and we don't have to find features again.
   When we change the number of output classes, we need to retrain the BP-CNN again but with Saab transform, the feature extraction module doesn't change, and we don't need to retrain/redesign.

IMAGE RECONSTRUCTION FROM SAAB COEFFICIENTS

Here, the task is to apply Saab transform to the images in the MNIST dataset.

APPROACH AND PROCEDURE

Generating Saab Coefficients:

1. Train convolutional layers using 10,000 train images to get the kernels in the convolutional layer. Kernel Size = 4*4, with non-overlapping stride.

2. Generate input data matrix by taking input image of 32*32 and creating patches of 4*4.
3. For all given input images, subtract the pixel mean from all patches created above.
4. Saab coefficients are generated for the 1st convolutional layer by multiplying the input data matrix to 1st convolutional layer kernels.
5. Then again 4*4 patches are generated from the above output and patch mean is subtracted from all patches
6. Bias term is added
7. This output is then multiplied to the kernels in the 2nd convolutional layer
8. Bias is subtracted from the output of 2nd conv layer to obtain the saab coefficients of the second convolutional layer.
9. Weights and features are saved with features having dimension 2*2*N for every input image, N being the number of kernels in the 2nd conv layer.

Image Reconstruction from Saab Coefficients Algorithm:

1. Load the features as saved in calculating the saab coefficients.
2. Inverse transformation is applied using the saved weights for each corresponding layer and bias is added/removed if needed.
3. Original Image patches are created from the AC and DC kernels by adding mean to Ac and DC and reshaping
4. The patches obtained are then rearranged to create the original image.
5. PSNR is calculated between the original and reconstructed image.

EXPERIMENTAL RESULTS

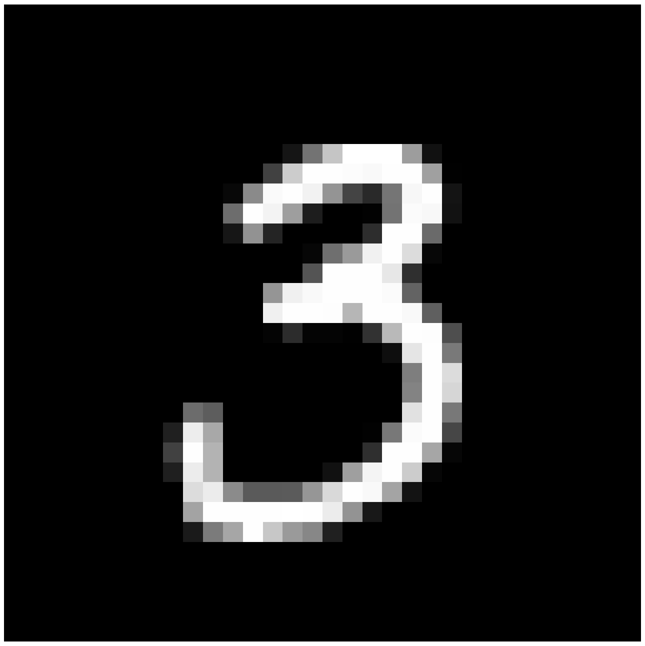


Figure 3 Original Image 3

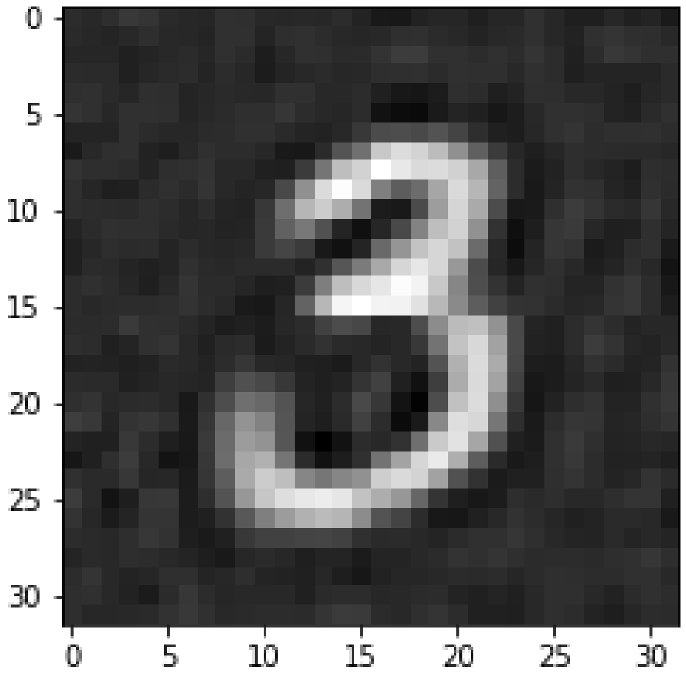


Figure 4 Reconstructed Image for Setting 1, Conv1 Filters = 5, Conv2 Filters = 20

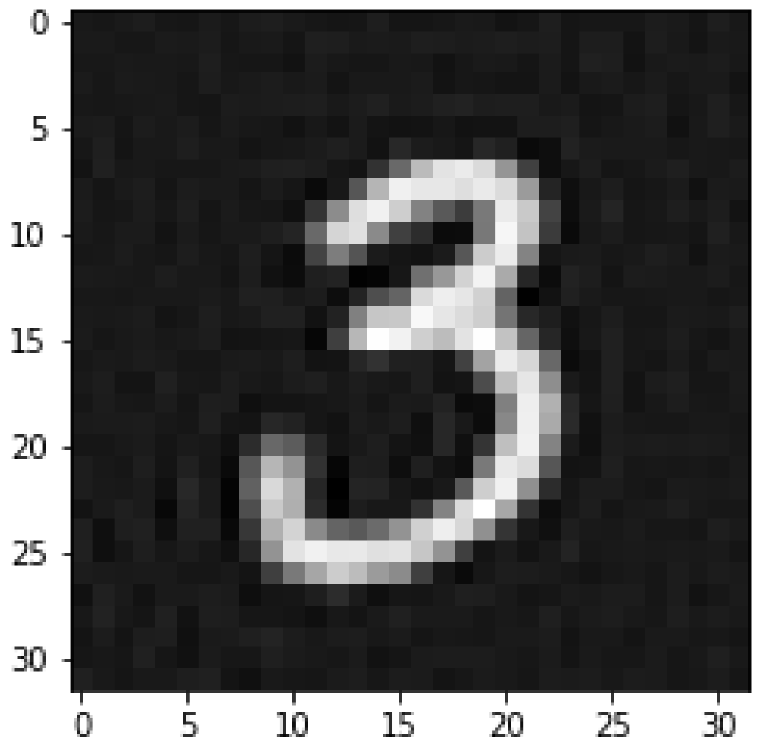


Figure 5 Reconstructed Image for Setting 2, Conv1 Filters = 10, Conv2 Filters = 40

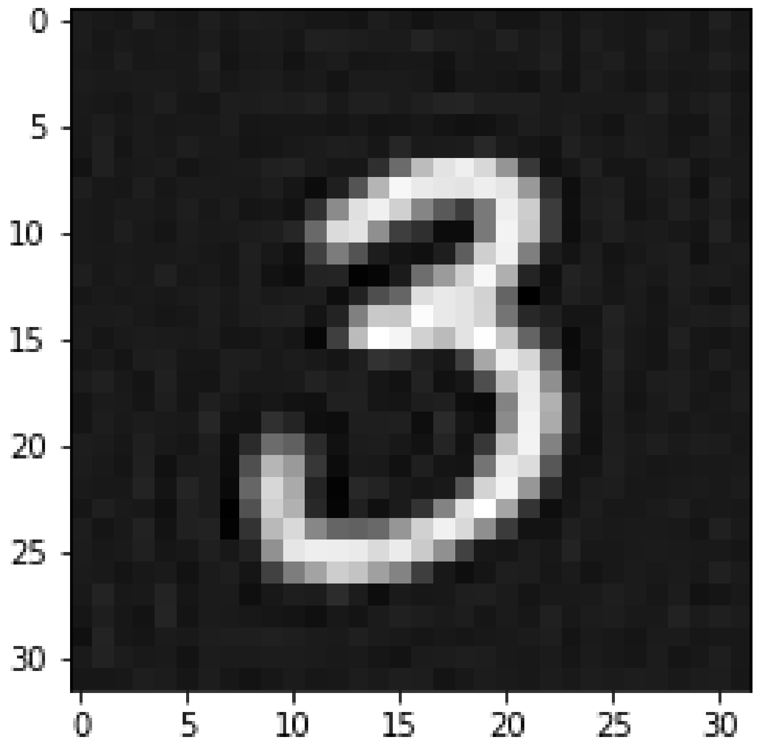


Figure 6 Reconstructed Image for Setting 3, Conv1 Filters = 20, Conv2 Filters = 100

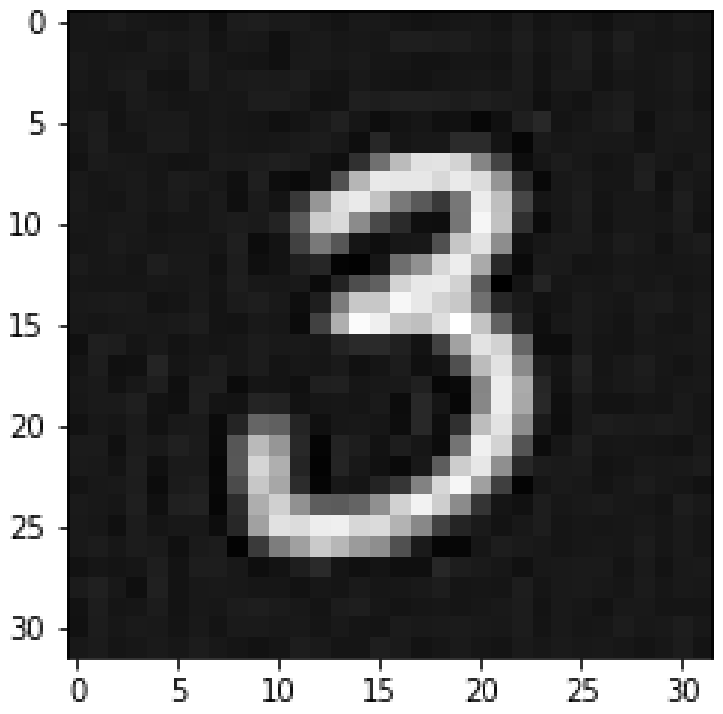


Figure 7 Reconstructed Image for Setting 4, Conv1 Filters = 10, Conv2 Filters = 120

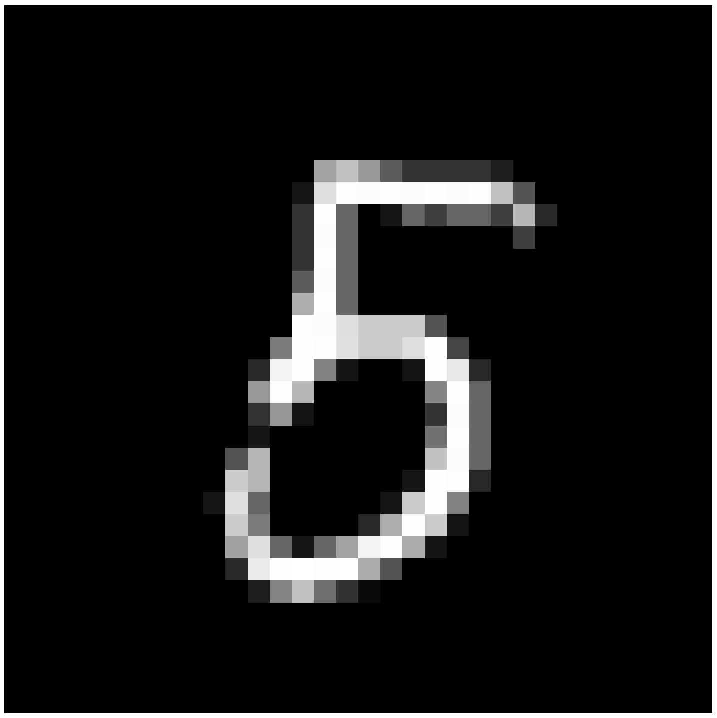


Figure 8 Original Image 5

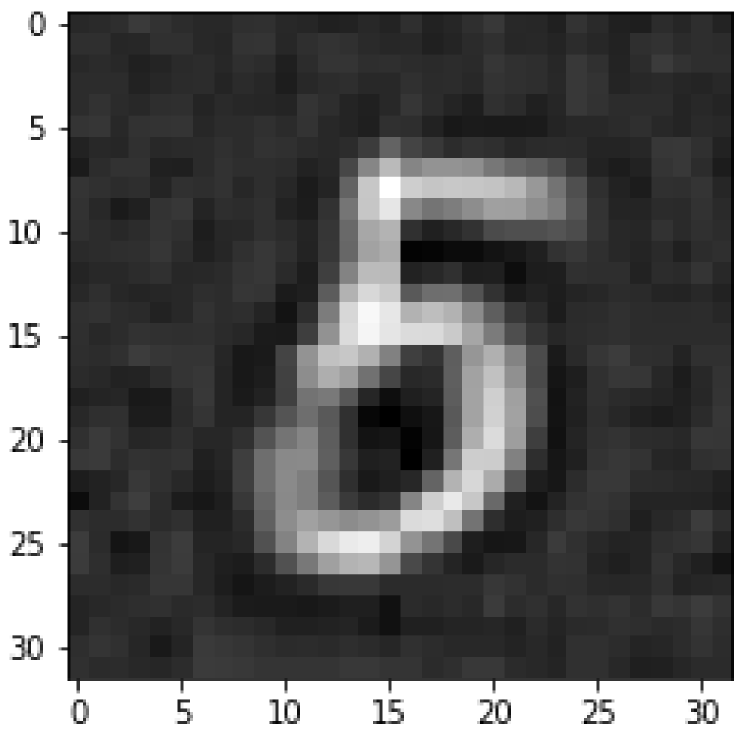


Figure 9 Reconstructed Image for Setting 1, Conv1 Filters = 5, Conv2 Filters = 20

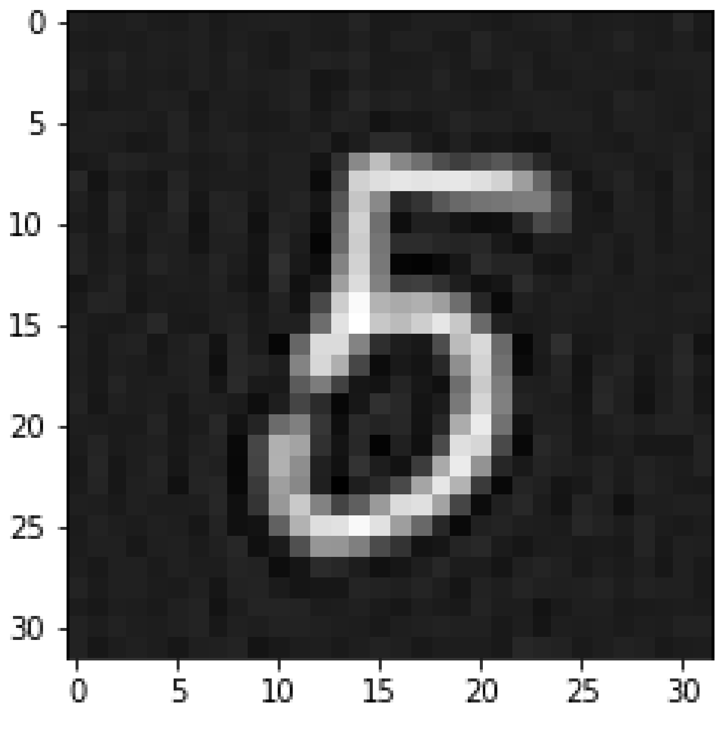


Figure 10 Reconstructed Image for Setting 2, Conv1 Filters = 10, Conv2 Filters = 40

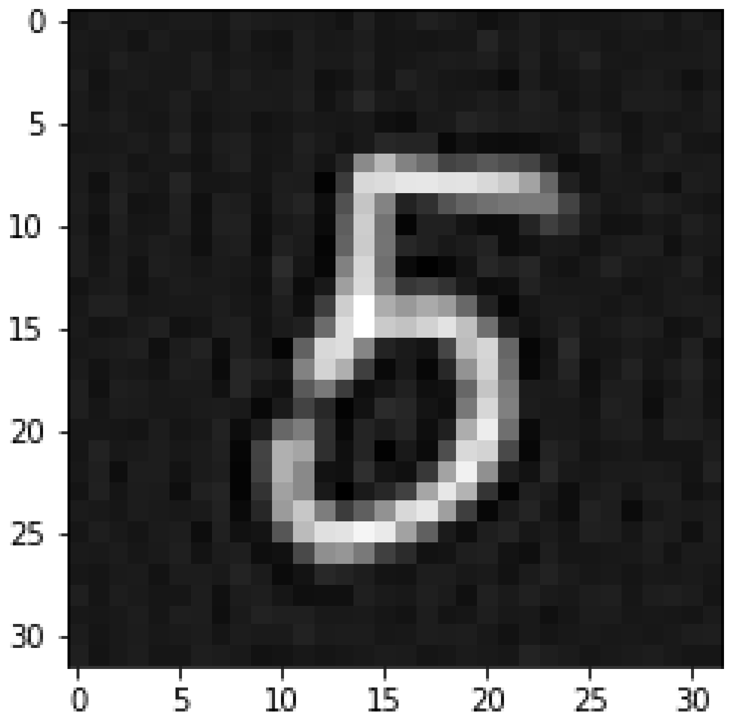


Figure 11 Reconstructed Image for Setting 3, Conv1 Filters = 20, Conv2 Filters = 100

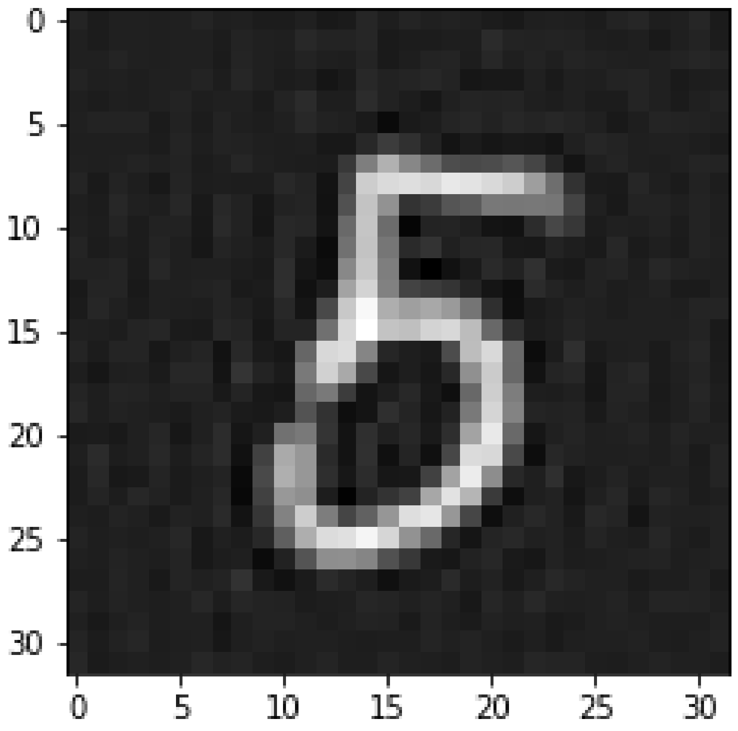


Figure 12 Reconstructed Image for Setting 4, Conv1 Filters = 10, Conv2 Filters = 120

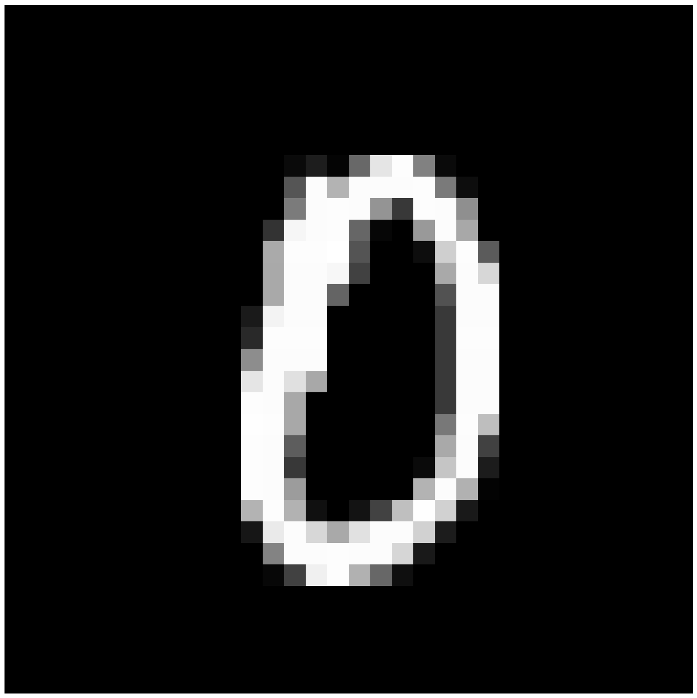


Figure 13 Original 0 Image

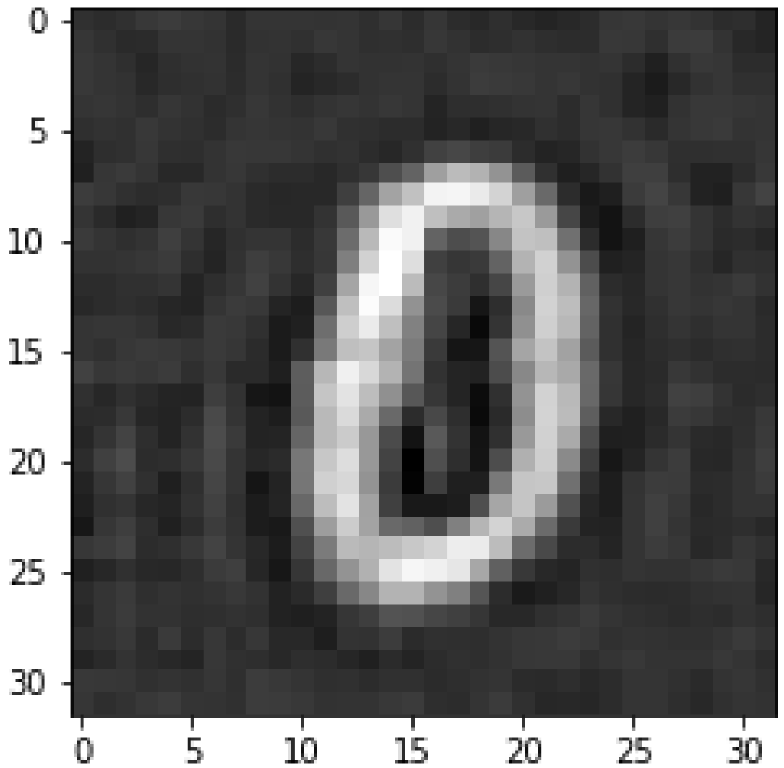


Figure 14 Reconstructed Image for Setting 1, Conv1 Filters = 5, Conv2 Filters = 20

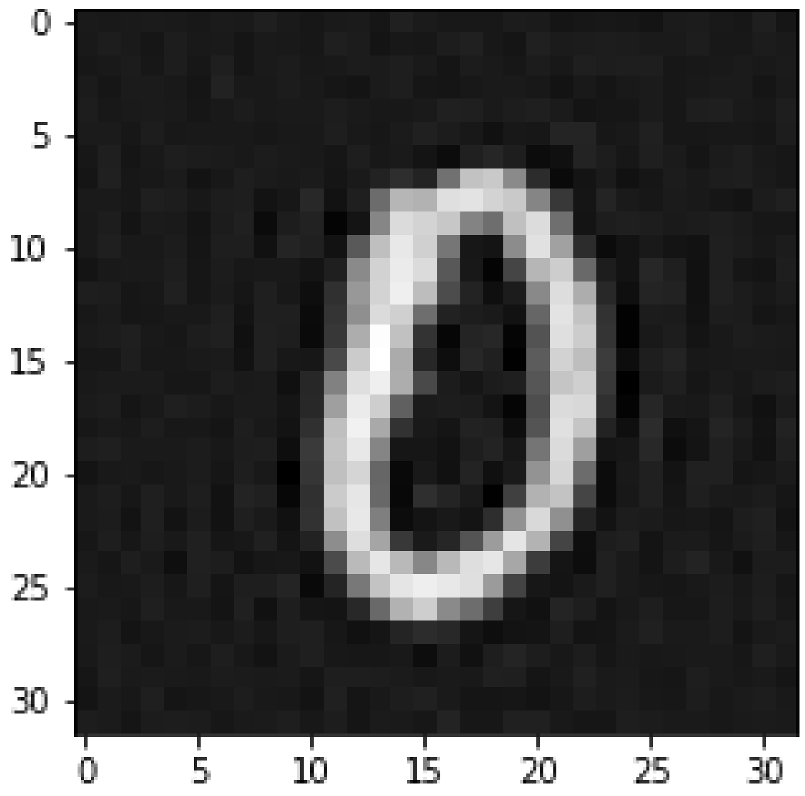


Figure 15 Reconstructed Image for Setting 2, Conv1 Filters = 10, Conv2 Filters = 40

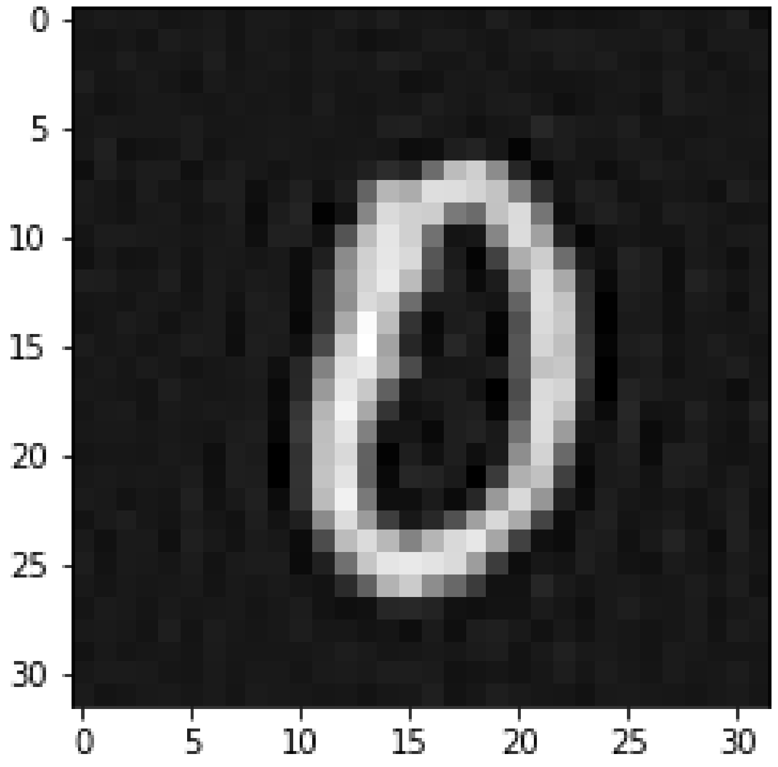


Figure 16 Reconstructed Image for Setting 3, Conv1 Filters = 20, Conv2 Filters = 100

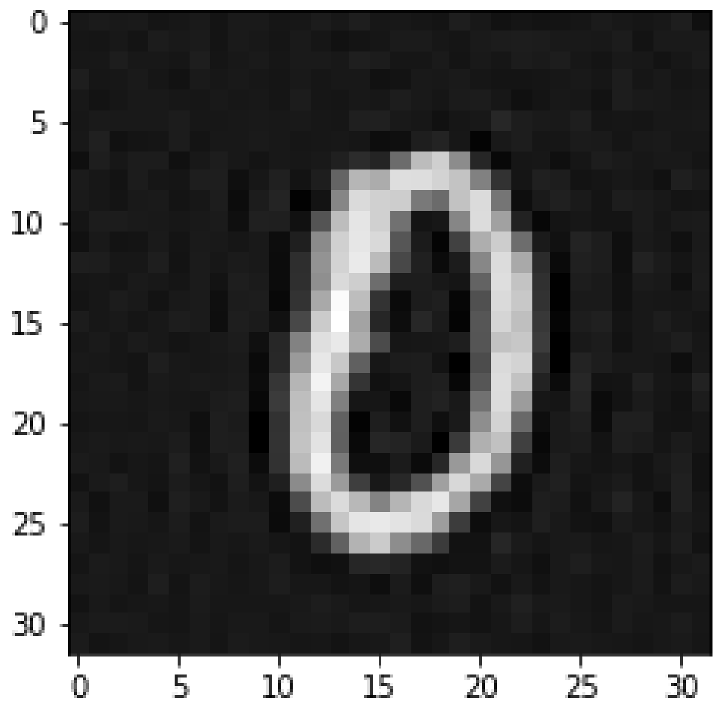


Figure 17 Reconstructed Image for Setting 3, Conv1 Filters = 20, Conv2 Filters = 100

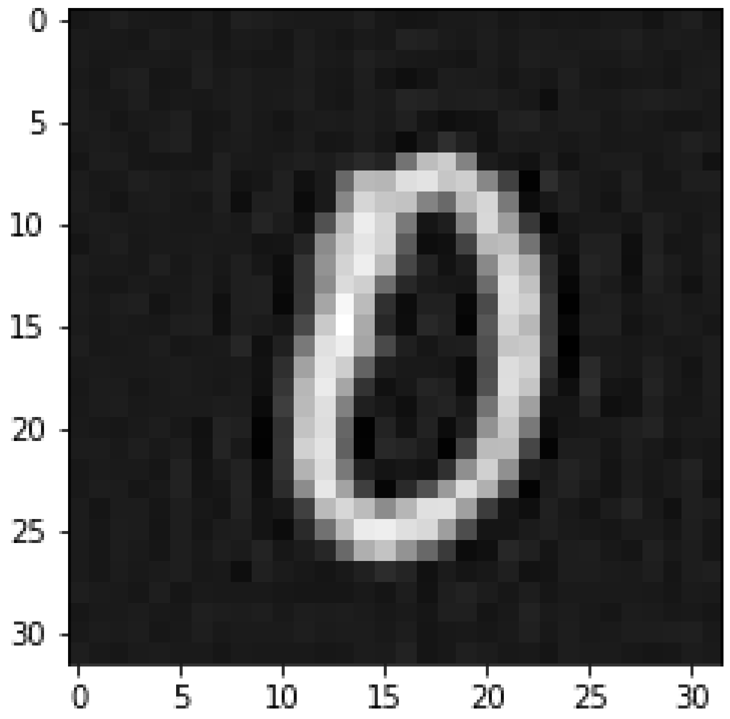


Figure 18 Reconstructed Image for Setting 4, Conv1 Filters = 10, Conv2 Filters = 120

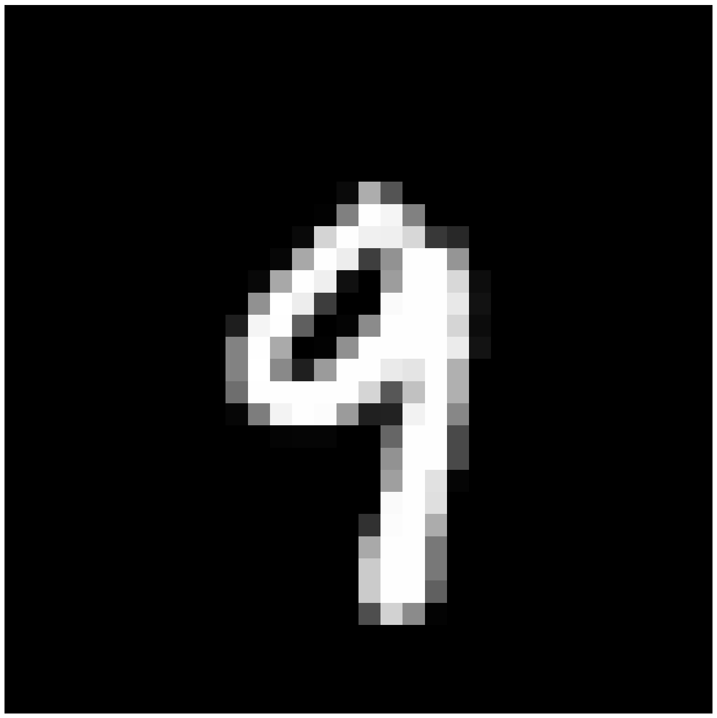


Figure 19 Original 9 Image

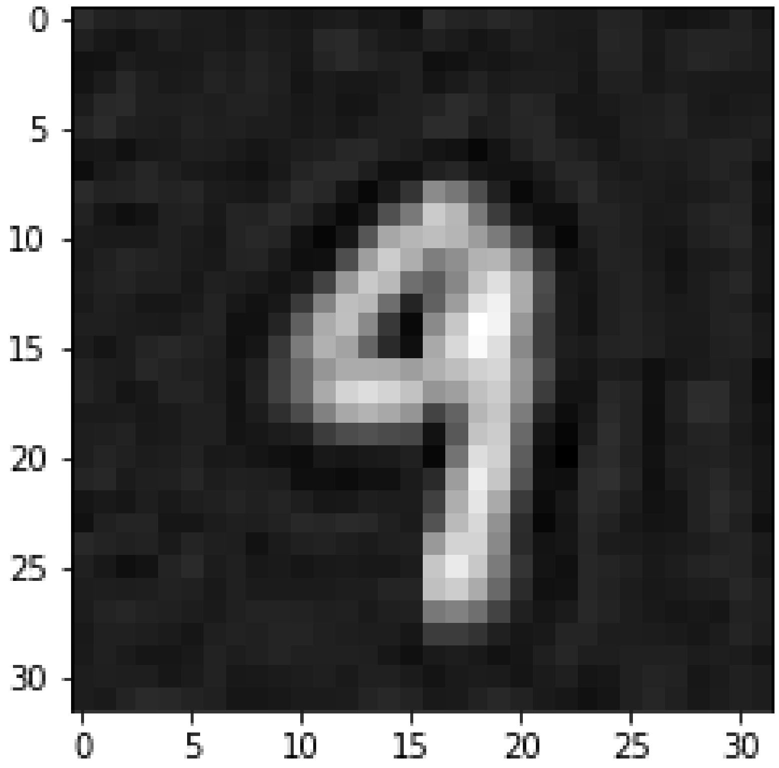


Figure 20 Reconstructed Image for Setting 1, Conv1 Filters = 5, Conv2 Filters = 20

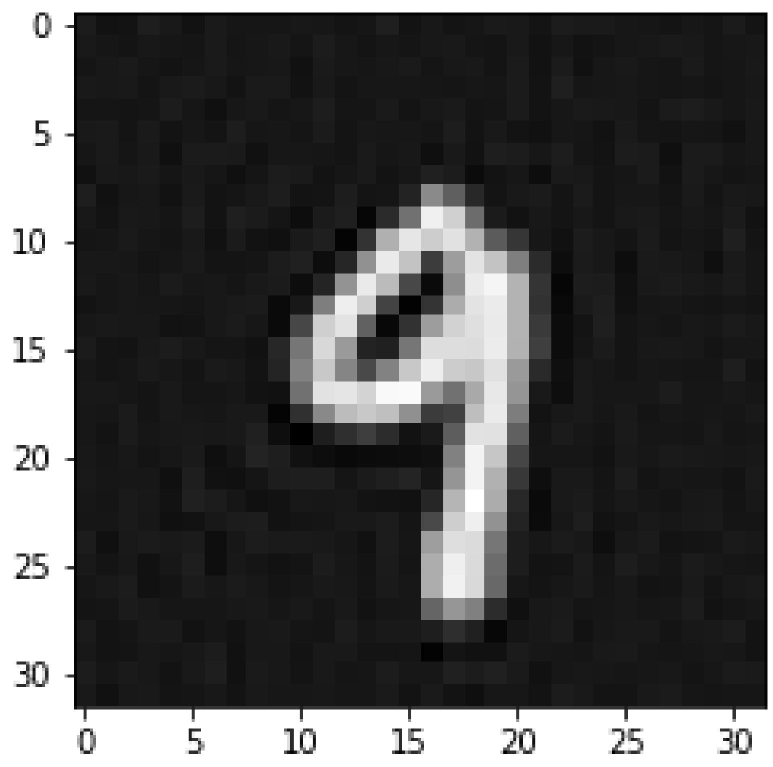


Figure 21 Reconstructed Image for Setting 2, Conv1 Filters = 10, Conv2 Filters = 40

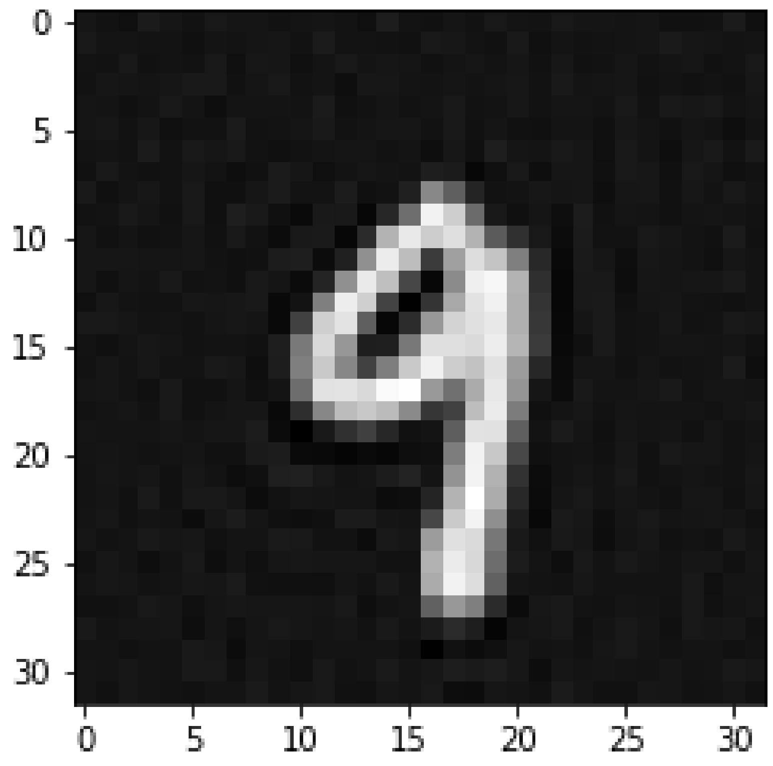


Figure 22 Reconstructed Image for Setting 3, Conv1 Filters = 20, Conv2 Filters = 100

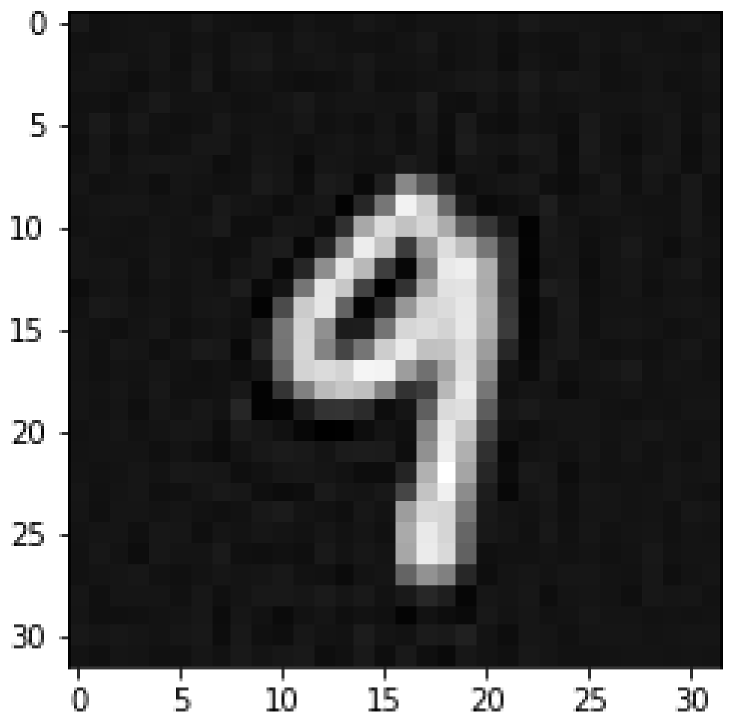


Figure 23 Reconstructed Image for Setting 4, Conv1 Filters = 10, Conv2 Filters = 120

PSNR:

Settings	Zero image	Three Image	Five Image	Nine Image
1	20.2	21.8	20.1	22.3
2	25.1	26.9	25.7	26.4
3	26.2	28.1	24.4	28.5
4	25.7	27.4	26.1	26.3

DISCUSSION

- As the kernel size increases, the PSNR value also increases, i.e. the quality of the reconstructed image is better.
- Low kernel size gives a more blurry reconstructed image.
- This might be happening because as we take larger kernels, we are capturing more important features that, when reversed, give a better reconstructed image.
- On the other hand, using small kernel size means throwing away more important feature eigen vectors that are necessary for image reconstruction and thus reconstructed image is very blurry.

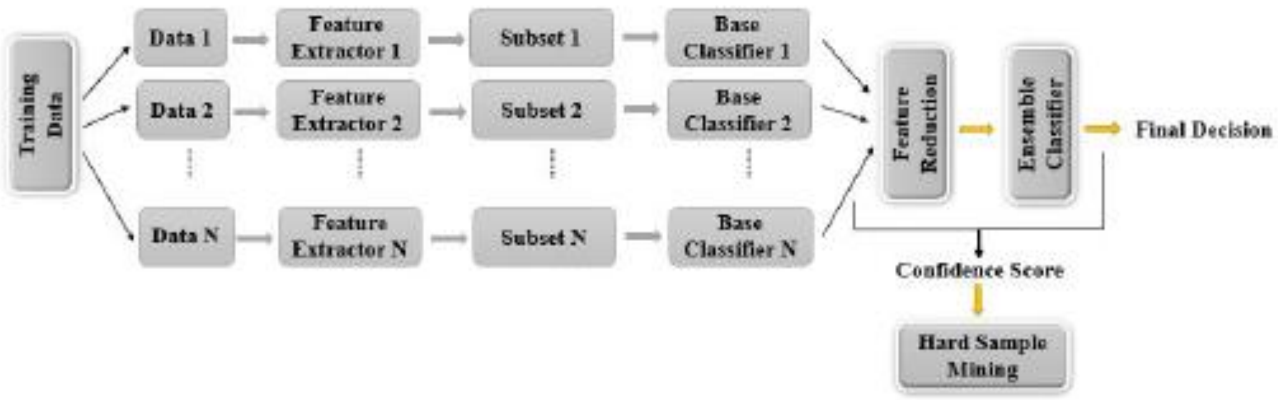
- Saab transform is a lossy transform, therefore the reconstructed image is not an exact replica of the input image.
- One of the main advantages of Saab transform is that it performs well with a very small train and test data set also.
- Saab Transform offers robustness and scalability compared to BP-CNN.
- When exposed to noisy images for training and testing, Saab transform has better performance than CNN.

Improving the performance of Saab-transform-based decision systems:

1. Removing the spatial-dimension redundancy of Saab-transform-based decision system by implementing Channel-wise PCA.
   a. This removes the feature redundancy in spatial dimensions
   b. Shows stronger correlation in low frequency components.
   c. PCA is applied to spatial dimensions on each filter
   d. This preserves more energy for low frequency components.
2. Developing an ensemble system that combines the predictions of multiple saab-transform-based-decision systems.
3. Extend the saab-transform based decision system to a semi-supervised decision system.

HANDWRITTEN DIGITS RECOGNITION USING ENSEMBLES OF FEEDFORWARD DESIGN

Ensemble system is effective for improving performance of saab-transform based decision system. The base classifier can be FF-CNNs having different parameter settings and the Ensemble final classifier can be SVM or random Forest classifier.



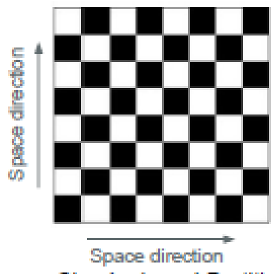
Training and testing Classification Accuracy of Individual FF-CNNs on MNIST dataset.

Strategies implemented to increase diversities in an ensemble of FF-CNNs:

1. Different Filter Size settings : (5x5, 5x5), (3x3, 5x5), (5x5, 3x3), and (3x3, 3x3)
2. Selecting Subsets of features from the available features.
    a. Select subset of features from Conv 1 randomly and from conv 2 as well at random

$$\mathbf{F} = [\mathbf{F}_{conv1}, \mathbf{F}_{conv2}]$$

    b. Implement checkerboard partition of features in the spatial dimension



Checkerboard Partition

3. LAWS Filter Bank:
    a. Changing representation of the input images
    b. RGB to YCbCr color space transformation
    c. 3*3 Laws Filter Bank
4. Hard Examples Mining
    a. Calculate the confidence score of each selection of hard examples:

Confidence score

$$CS1_i = max(P_{final}(\mathbf{y}|\mathbf{x_i}))$$

$$CS2_i = N_i/N_{all}$$

$$\mathbf{X_{hard}} = \{\mathbf{x_i} : CS1_i < th_1 \quad and \quad CS2_i < th_2\},$$

APPROACH AND PROCEDURE:

1. I trained 10 different FF-CNNs with different settings. The settings are:
    a. Filter Size settings:
        i. 5*5, 5*5
        ii. 3*3, 5*5
        iii. 5*5, 3*3
        iv. 3*3, 3*3
    b. 3*3 Laws Filters:
        i. L3L3
        ii. E3E3
        iii. S3S3
        iv. L3S3
        v. S3L3
        vi. L3E3
2. The weights and features are saved as pickle files.
3. The predictions of each setting FF-CNN is concatenated to form a prediction matrix of dimension 1000.
4. PCA is applied on this matrix to reduce the dimension from 1000 to 10.
5. This PCA output is then fed into a SVM classifier and accuracy is calculated with respect to the true output labels.

EXPERIMENTAL RESULTS

Accuracy Metrics for Individual FF-CNNs of 10 Settings:

Setting	Training Accuracy	Testing Accuracy
1. 5*5, 5*5	99.22	97.23
2. 3*3, 5*5	98.87	96.20

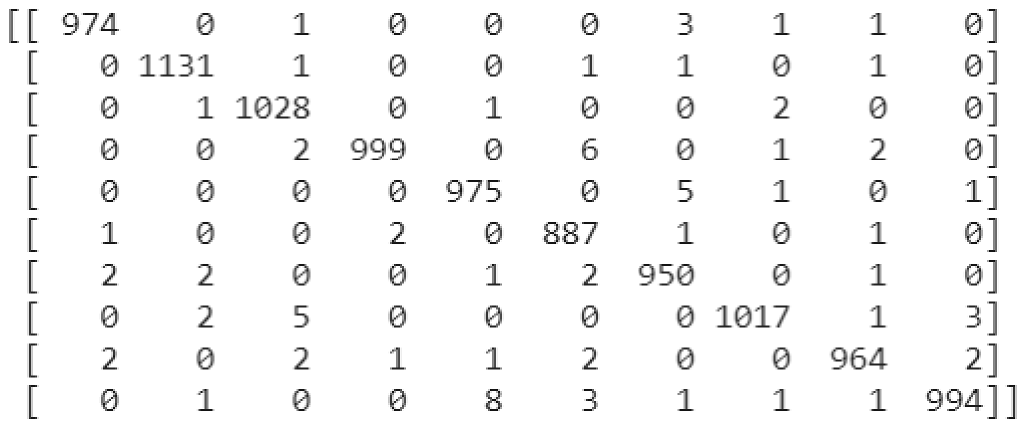
3. 5*5, 3*3	98.02	97.60
4. 3*3, 3*3	99.07	96.69
5. L3L3	99.10	96.20
6. E3E3	99.20	97.14
7. S3S3	99.23	97.63
8. L3S3	99.22	97.01
9. S3L3	99.07	97.30
10. L3E3	99.03	97.00

Accuracy Metric of Final Ensemble FF-CNN Model:

Training Accuracy: 99.31

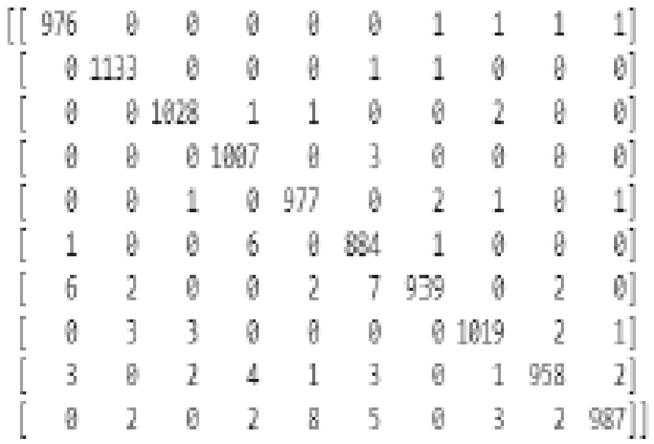
Test Accuracy: 97.91

ERROR ANALYSIS



```
[[ 974    0    1    0    0    0    3    1    1    0]
 [    0 1131    1    0    0    1    1    0    1    0]
 [    0    1 1028    0    1    0    0    2    0    0]
 [    0    0    2  999    0    6    0    1    2    0]
 [    0    0    0    0  975    0    5    1    0    1]
 [    1    0    0    2    0  887    1    0    1    0]
 [    2    2    0    0    1    2  950    0    1    0]
 [    0    2    5    0    0    0    0 1017    1    3]
 [    2    0    2    1    1    2    0    0  964    2]
 [    0    1    0    0    8    3    1    1    1  994]]
```

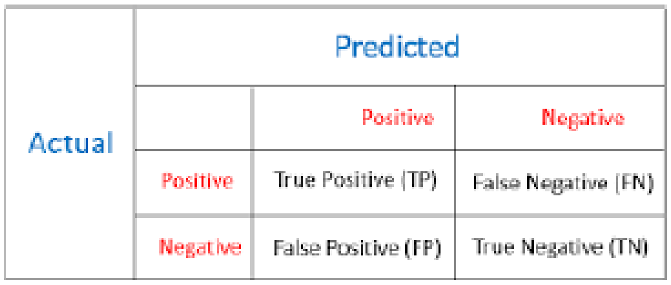
Figure 24 Confusion Matrix for BP-CNN Best Setting (From HW 5)



```
[[ 976    0    0    0    0    0    1    1    1    1]
 [   0 1133    0    0    0    1    1    0    0    0]
 [   0    0 1028    1    1    0    0    2    0    0]
 [   0    0    0 1007    0    3    0    0    0    0]
 [   0    0    1    0  977    0    2    1    0    1]
 [   1    0    0    6    0  884    1    0    0    0]
 [   6    2    0    0    2    7  939    0    2    0]
 [   0    3    3    0    0    0    0 1019    2    1]
 [   3    0    2    4    1    3    0    1  958    2]
 [   0    2    0    2    8    5    0    3    2  987]]
```

Figure 25 Confusion Matrix for FF-CNN Ensemble

Classification error is the ratio of the number of samples that are incorrectly classified to the total number of samples.



Actual	Predicted		
		Positive	Negative
	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

From the above confusion matrices, we can see that they are very similar. Diagonal elements are the correctly classified numbers and the off-diagonal elements are misclassified.

Classification error is usually caused due to data imbalance and/or over-fitting.

DISCUSSION:

MNIST dataset doesn't have data imbalance and is basically an ideally distributed dataset that gives good classification accuracy even for less training data.

Thus, the performance of BP-CNN and FF-CNN using Saab transform is comparable to each other on MNIST dataset. If there was data-imbalance present, Saab transform wouldn't have performed as well.

BP-CNN still has slightly better performance in terms of classification accuracy compared to Saab transform using SVM classifier as seen in the Confusion matrices.

Classification error of BP-CNN and FF-CNN is not that different. The percentage of error for both is around 2% which is good.

Thus, both models perform well.

PERFORMANCE IMPROVEMENT

We can basically improve the performance of BP-CNN in three ways:

1. Data Improvement:
   Tinkering with data for example: Data Augmentation, Collecting More Data, Feature selection based on correlation.

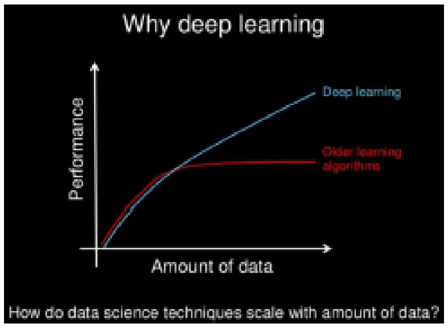


Figure 26 Performance vs Data Curve

2. Algorithm Improvement:
   Since different algorithms perform differently with different data, we must perform trial and error with available algorithms, evaluate performance and chose the best performing algorithm.
3. Resampling Methods:
   To reduce the impact of data imbalance and get better classification accuracy, we can resample the data.

Other Points on improvement of BP-CNN:

1. Learning rate determines the convergence time of the network. A smaller learning rate converges very slowly. But very high learning rate may result in no convergence at all. Thus, we must define try different setting of learning rate.

2. We tend to use small filters in the CONV layers to capture the low-level features properly and implement padding to conserve the spatial dimensions of the input. Bigger filter sizes than 5*5, if needed should be used only in the first convolution layer.
3. Pool layers with 2*2 receptive field and stride 2 discard exactly 75% of the input activations. Larger pooling receptive fields are uncommon as they are too lossy and aggressive leading to worse performance.
4. Stride 1 in CONV layer ensures transforming input depth wise in conv layer and leaves down-sampling work for the pooling layers.
5. Padding keeps the spatial sizes constant as well as improves the performance. If padding wasn't present, the size of the input volumes will keep decreasing throughout the CONV layers as well making the border data vanish.
6. Filter Numbers: on increasing the number of filters, basically we get activation maps for more features, in turn helping the network learn better, increasing the performance of classification. 64 filters gave substantially good output.
7. Filter Size: Comparing between filter sizes of 3*3, 5*5 and 7*7, best observation is seen for 3*3 filter size.
8. Activation Function: tanH vs ReLu function comparison, ReLu works better in improving receptiveness of network. It helped improve accuracy.
9. Number of Layers: As number of conv layers are increased, it increases the number of trainable weights of the network substantially, offsetting the tradeoff between the number of available data points vs parameters.
10. Incorporating Dropout and Batch Normalization layers helped reduce overfitting issues in the network, making network converge faster.

We can improve the performance of FF-CNN as:

1. FF-CNN method using saab coefficients can be generalized by combining the feature extraction and classification modules end to end.
2. Since we select best features at every stage using PCA, but this may not always guarantee best features. We can implement more effective

methods to extract important features from all the available features for example correlation matrix or chi-square values.

3. Removing the spatial-dimension redundancy of Saab-transform-based decision system by implementing Channel-wise PCA.
    a. This removes the feature redundancy in spatial dimensions
    b. Shows stronger correlation in low frequency components.
    c. PCA is applied to spatial dimensions on each filter
    d. This preserves more energy for low frequency components.
4. Developing an ensemble system that combines the predictions of multiple saab-transform-based-decision systems.
5. Extend the saab-transform based decision system to a semi-supervised decision system.

REFERENCES:

1. Class Notes and assignments
2. Digital Image Processing, Second Edition, Rafael Gonzalez, Richard E Woods, Pearson Education
3. Digital Image Processing, Fourth Edition, William K. Pratt, Wiley-Interscience Publication
4. [LeNet-5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.
5. [MNIST] http://yann.lecun.com/exdb/mnist/
6. [ReLU] https://en.wikipedia.org/wiki/Rectifier_(neural_networks).
7. [Softmax] https://en.wikipedia.org/wiki/Softmax_function
8. https://machinelearningmastery.com/improve-deep-learning-performance/
9. 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.
10. https://github.com/davidsonic/Interpretable_CNN
11. Chen, Y., Yang, Y., Wang, W., & Kuo, C. C. J. (2019). Ensembles of feedforward-designed convolutional neural networks. arXiv preprint arXiv:1901.02154.