Homework 6: EE569

Ishaan Vasant

6989-5065-37

ivasant@usc.edu

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

Abstract

Convolutional Neural Networks are very similar to ordinary Neural Networks architectures but make the explicit assumption that the inputs are images, which allows to encode certain properties into the architecture. The Feed Forward – CNN and Saab coefficients were explored in this problem.

Approach

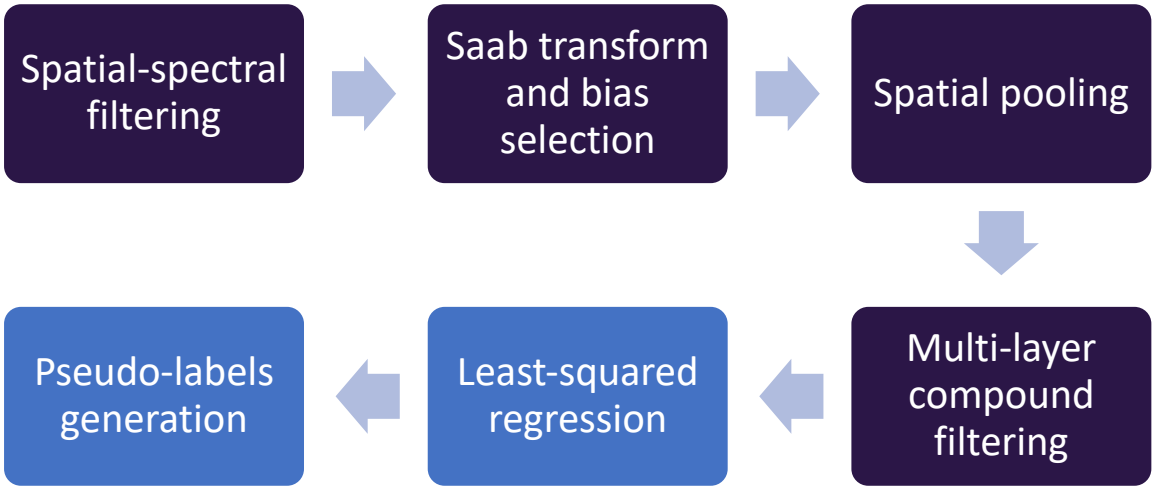
For the first part, the paper on the feed-forward design of convolutional neural networks was read and summarized with a flowchart, before explaining each of its components. For the second part, the Saab coefficients of the handwritten digit images were computed, and the images were reconstructed from them. PSNR was used to evaluate them under different settings. For the third part, the FF-CNN was trained and tested on the MNIST dataset. An ensemble model of 10 different FF-CNNs was constructed and its accuracies were reported. A comparison was made between the FF-CNN and the BP-CNN on the basis of their error rates.

Results and Discussion (merged for ease of readability)

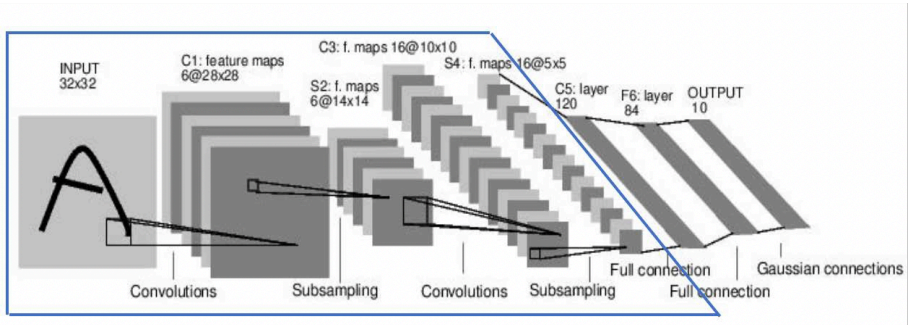
a)

1. Flowchart and Summary of Feed Forward - Convolutional Neural Network: -

   The flowchart below summarizes the processes that define the Feed Forward – Convolutional Neural Network. The purple boxes illustrate the process that take place in the convolutional layers whereas the blue boxes illustrate the process that take place in the fully connected layers. These processes are explained in brief below.



- Feedforward design of convolutional layers



Spatial-spectral filtering

In convolutional neural networks, each dimension of intermediate space can be interpreted as a feature or as a representation. The process of spatial-spectral filtering in cascade with pooling, can provide an effective way of extracting the discriminant dimensions. Conducting pixel-wise comparisons in images to recognize digits is not a great idea since there are many varieties in each digit's spatial domain. It is also sensitive to rotation and translation. Considering the neighborhood of each pixel and finding its spatial representations is a better idea. PCA is used to find the dominant stroke patterns and represent any neighborhood pattern as a linear combination of

those dominant patterns. The convolutional layers include spatial-spectral transformations that transform an image into its spatial-spectral representations layer by layer. Spatial resolution lowers gradually and is traded for spectral representations by projecting them onto PCA kernels. The advantage of PCA-based subspace approximation is that it does not need image labels. The transforms are conducted in non-overlapping windows to provide a richer feature set for feature selection even though it increases computation and storage costs.

Saab transform and bias selection

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

The Saab transform is a particular way of selecting the bias term b_k and the anchor vector a_k in the above affine transform.

Anchor vectors are divided into 2 categories:

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

The input vector space is the sum of the two vector spaces created by the DC and AC anchor vectors. For any vector x, we can express its DC component as:

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

and its AC component as:

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

Bias selection is done with 2 constraints imposed:

1. Positive response constraint, where b_k is chosen such that the k^{th} response is a non-negative value.
2. Constant bias constraint, where all bias terms are equal

Therefore, the output with or without the ReLU activation function is the same and the nonlinearity introduced by it is eliminated.

Spatial pooling

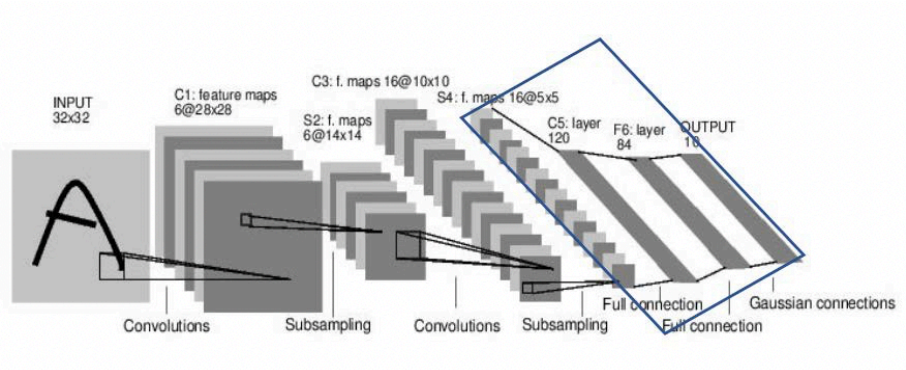
Spatial pooling is a process that allows a reduction in storage and computational resources. It filters out insignificant patterns while preserving significant patterns. It is common practice to include a pooling layer after a convolutional layer in the network architecture. The pooling layer reduces the

size of the representation in order to speed up computation. The pooling layer works on each of the dimensions of the representation coming in to it. It does do by identifying small 2x2 windows and depending on the type of pooling layer, it reduces the window to a single value, therefore reducing the size by half. In max pooling, the maximum value out of the 4 in the 2x2 window is chosen. It is performed at each spectral component in an independent fashion and due to the ReLU activation function, all responses are non-negative. In max pooling, for a given pattern the maximum response value is used to indicate the best match and other visual patterns are suppressed.

Multi-layer compound filtering

The result of cascading multiple convolutional layers is a rich set of image patterns which are interpreted as object signatures. These are called compound filters. In the Feed Forwards design, a target pattern is represented as the linear combination of PCA filters and these responses are signatures of the receptive fields in the image. A bias term is not needed in the final convolutional layer as a different method is used for the fully connected layers.

- Feedforward design of Fully Connected layers



Least-squared regressor (LSR)

Each fully connected layer in the Feed Forward design is a least squared regressor and its output is a one-hot vector – a vector whose elements are all zero barring one whose values is set to one. This vector is usually used in the output layer of convolutional neural networks. Here it is the output of each fully connected layer. The challenge is no label is connected to the input vector when the output is a hidden layer. To overcome this, k-means clustering is performed on the input and grouped into clusters equal to the number of output nodes. At this point, each input has the new cluster label as well as the original class label. These are combined to generate pseudo-labels which sets

up a linear least squared regression problem. Every one-hot vector denotes the label of the digit to be classified.

Pseudo-labels generation

In order to build least squared regressors in the fully connected layers, output labels need to be defined. In this case, a combination of the original class label and the auxiliary label is considered. Each input sample has its own original label. In order to create auxiliary labels, k-means clustering is performed on samples of the same class. The representative sample is provided by the centroid of the class. This is done to capture diversity of a single class with more samples. The output of the last convolutional layer is a spatial-spectral transform of an input image that provides a feature vector for the classifier. A mathematical model is used for the alignment of input and output feature spaces in order to lower cross-entropy value and produce more discriminant features. Least squared regression in cascade is therefore a sequence of feature space alignment processes.

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

- Principle

  The BP-CNN is based on 3 factors – data (training and testing), network architecture and an optimizing cost function. Thus, the procedure here is straightforward once all 3 factors are established. On the other hand, the FF-CNN design uses covariance matrices to determine spatial-spectral transformations in the convolutional layers. Data labels are not needed in the conv layer but in the FC layer. LSR models are built in the FC layer to enable a multi-stage decision process.

- Mathematical Tools

  The BP-CNN design is dependent on the technique of stochastic gradient descent to optimize the cost function that is pre-defined. This can give rise to the vanishing gradient problem when the CNN architecture becomes deep. In FF-CNN, the mathematical tool is linear algebra and statistics.

- Interpretability

  FF-CNN is mathematically transparent and the learnings from the FF-CNN model cannot be easily transferred to the BP-CNN design. The BP-CNN design is difficult to interpret whereas the FF-CNN is easy to interpret.

- Modularity

  The BP-CNN design depends on input data and output labels. In the BP-CNN design, the input data has a stronger influence on parameters of shallow layers and the output labels has a strong influence on the parameters of the deep layers. The whole network design is coupled end to end. The FF-CNN design separates this couple into two parts – convolutional layer network and the fully connected network. This is parallel to the pattern recognition process of feature extraction and classification.

- Robustness

  Both BP-CNN and FF-CNN are susceptible to attacks that are adversarial in nature when the model is known to the attackers and is fixed. These lead to performance degradation of extreme measures for the target network. This issue gave rise to the suggestion of using ensemble models by combining multiple FF-CNN networks.

- Training Complexity

  The BP-CNN is an optimization process that is iterative where all training samples are processed by the network once in each epoch. Therefore, tens or even hundreds of epochs are required for the network to converge. The FF-CNN is much faster than the BP-CNN and it is possible to reduce its complexity using statistical methods i.e. both least squared regression and principal component analysis can be performed on a smaller set of training data.

- Architecture

  The design of the BP-CNN has a constraint on the network architecture, in the sense that it has to ensure that the BP optimization is carried out end to end. On the other hand, the FF-CNN design does not require any architectural constraint. The fully connected layers can be replaced by a random forest or support vector machine classifier.

- Generalizability

  The cohesion between the data space and the decision space is very strong in BP-CNN. Therefore, different tasks require different networks even with the same input data. This is because the cost function for each task is different. On the other hand, in the FF-CNN, design, the same convolutional layers can be shared by all tasks since they only depend on the input data. The fully connected layers then take care of the individual task requirements. Different FC layers can be designed for different tasks.

- Performance

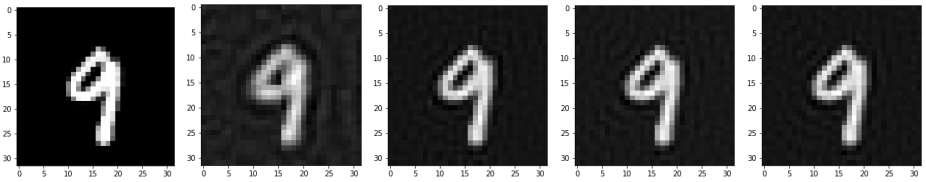
  Since the BP-CNN optimization process is end to end, it produces state of the art performances and is very difficult to beat. This is only the case when the cost function and the network architecture chosen is very relevant to the performance metric. If not, high performance is not guaranteed. The FF-CNN design is yet to be explored thoroughly. However, it can be said that one way to boost performance is to use the ensemble learning method which suits the FF-CNN design. This is because it is easy to adopt multiple classifiers in the fully connected stage.

b)

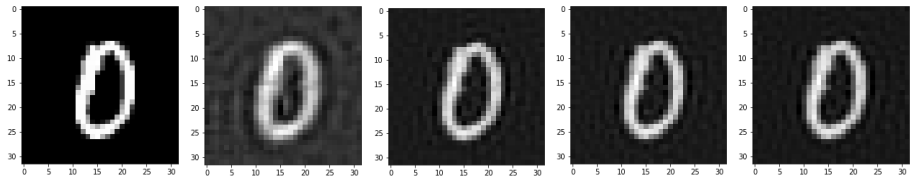
- The Saab coefficients for each of the four images were computed and were transformed back to images using a reconstruction algorithm. Four different settings were used by tuning the number of AC filters in the 2 convolutional layers.

  Setting 1: Conv Layer 1= 12 AC filters, Conv Layer 2 = 40 AC filters
  Setting 2: Conv Layer 1= 10 AC filters, Conv Layer 2 = 100 AC filters
  Setting 3: Conv Layer 1= 12 AC filters, Conv Layer 2 = 100 AC filters
  Setting 4: Conv Layer 1= 15 AC filters, Conv Layer 2 = 100 AC filters

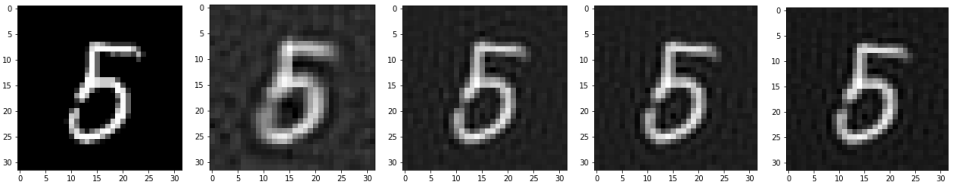
  The reconstructed images for each image and each setting are reported below.



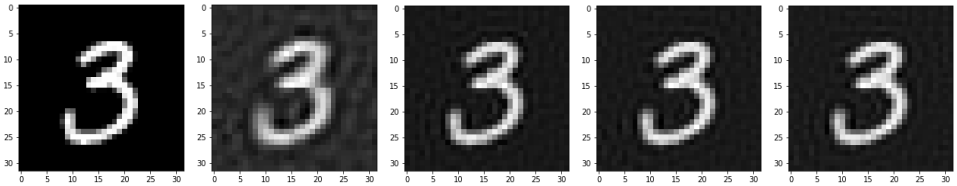
Original Setting 1 Setting 2 Setting 3 Setting 4



Original Setting 1 Setting 2 Setting 3 Setting 4



Original Setting 1 Setting 2 Setting 3 Setting 4



Original Setting 1 Setting 2 Setting 3 Setting 4

- The PSNR scores for each reconstructed image with respect to their originals were calculated as follows: -

	Digit 9	Digit 0	Digit 5	Digit 3
Setting 1	22.11	20.24	20.07	21.77
Setting 2	28.36	25.73	25.69	27.29
Setting 3	28.42	26.04	26.31	27.93
Setting 4	28.53	26.12	26.43	28.08

It is observed that the PSNR scores increase for all handwritten digit images as the number of AC filters in the convolutional layers increase.

c)

- The training and testing classification accuracy for the individual FF-CNN on the MNIST dataset is as follows: -

```
initiali dtype:  float32
Training image size: (60000, 32, 32, 1)
Testing_image size: (10000, 32, 32, 1)
S4 shape: (60000, 400)
--------Finish Feature Extraction subnet--------
0  layer Kmean (just ref) training acc is 0.8735166666666667
0  layer LSR weight shape: (401, 120)
0  layer LSR output shape: (60000, 120)
0  layer LSR training acc is 0.9524166666666667
1  layer Kmean (just ref) training acc is 0.9458333333333333
1  layer LSR weight shape: (121, 84)
1  layer LSR output shape: (60000, 84)
1  layer LSR training acc is 0.95475
2  layer LSR weight shape: (85, 10)
2  layer LSR output shape: (60000, 10)
training acc is 0.9702833333333334
```

```
--------Finish Feature Extraction subnet--------
0  layer LSR weight shape: (400, 120)
0  layer LSR bias shape: (1, 120)
0  layer LSR output shape: (10000, 120)
0  layer LSR testing acc is 0.1533
1  layer LSR weight shape: (120, 84)
1  layer LSR bias shape: (1, 84)
1  layer LSR output shape: (10000, 84)
1  layer LSR testing acc is 0.1461
2  layer LSR weight shape: (84, 10)
2  layer LSR bias shape: (1, 10)
2  layer LSR output shape: (10000, 10)
testing acc is 0.9717
Ishaans-Air:MNIST_FF ishaan$
```

Training Accuracy = 0.9702
Testing Accuracy = 0.9717

- Ten different FF-CNNs were trained and were ensembled. The ten FF-CNNs were distinguished based on the filter size and the use of Laws filter. Different filter sizes directly affect the receptive field size of each convolutional layer, thus introducing diversity. Laws filter utilizes various input representations as a diversity source.

  FF-1 – (5,5) Filter Size
  FF-2 – (5,3) Filter Size
  FF-3 – (3,5) Filter Size
  FF-4 – (3,3) Filter Size
  FF-5 – L3L3 Laws Filter
  FF-6 – E3E3 Laws Filter
  FF-7 – S3S3 Laws Filter
  FF-8 – L3S3 Laws Filter
  FF-9 – S3L3 Laws Filter
  FF-10 – L3E3 Laws Filter

  The training and testing accuracies of the above along with the ensemble accuracies are reported below: -

	Training Accuracy	Testing Accuracy
FF-1	0.9702	0.9717
FF-2	0.9768	0.9705
FF-3	0.9845	0.9719
FF-4	0.9861	0.9734
FF-5	0.9725	0.9701
FF-6	0.9622	0.9497
FF-7	0.8936	0.8823
FF-8	0.9337	0.9248
FF-9	0.9564	0.9326
FF-10	0.9601	0.9464
Ensemble	**0.9890**	**0.9786**

It is observed that the ensemble model has the highest training and testing accuracy among all models.

- The best performing BP-CNN had a training error of 0.2233% and a test error of 1.18% while the best performing FF-CNN (ensemble) has a training error of 1.10% and a test error of 2.14%. It is observed that the percentages are not the same. The FF-CNN has a better training error whereas the BP-CNN has a better test error. This gives rise to the proposal that an ensemble model of both FF-CNN as well as BP-CNN could result in a better training and testing classification error rate.