Homework 6: Feedforward CNN Design and Its Application to the MNIST Dataset

Motivation:

Convolution Neural Network uses non-convex optimization and back propagation to classify images. Even though it has given very good accuracy compared to classic image processing algorithms, it has some issues like giving complete different classification of slightly perturbed input with very high confidence. To tackle these issues, Dr. Kuo came up with a more transparent model called feedforward CNN inspired by BP-CNN.

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

Training:

Construction of Conv layer:

- For a 32x32 input , overlapping or non-overlapping sample patches are created with a kernel size, 4x4 in our case.
- These patches are stacked one below other, increasing the spectral dimension
- Patch mean and feature mean are removed from each patch and all patches respectively.
- Apply PCA on this spectral dimension and define AC anchor vectors
- Define DC anchor vector by 1/sqrt(size of kernel)x$(1, 1, 1, 1 \ldots . 1)^T$
- Concatenate DC and AC anchor vectors.
- Find SAAB coefficients of next layer by multiplying kernel with flattened sampled patches and add bias

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

- Bias is selected to follow the constraint:

$$b_k \geqslant \max_{\mathbf{x}} ||\mathbf{x}||, \quad k = 0, \ldots, K-1.$$

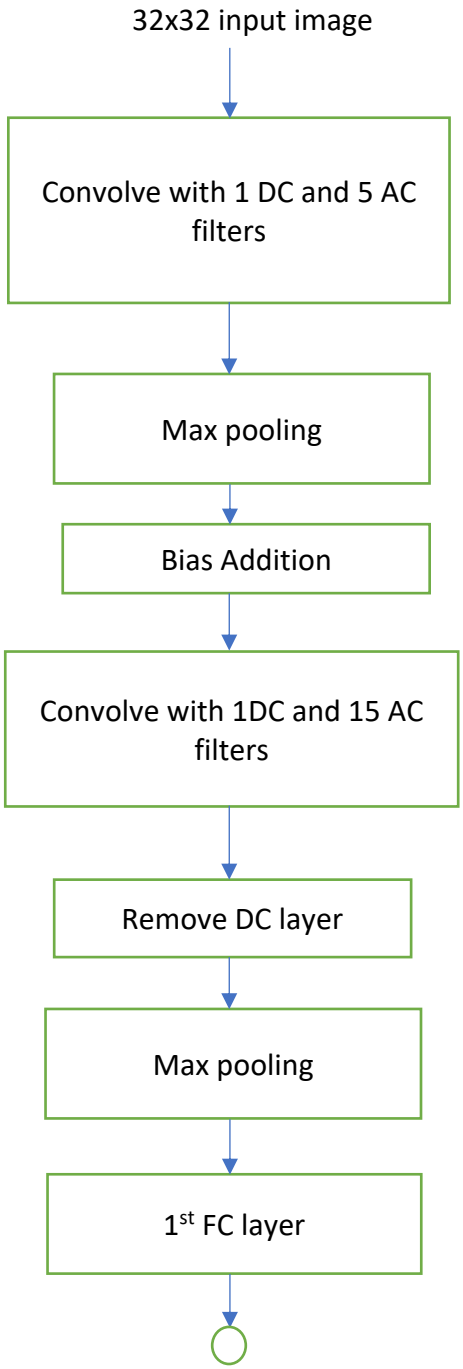
  A constant bias is used for all ks.
- Reshape in 4D blocks from flattened and do max-pooling
- Repeat for number of convolutional layers

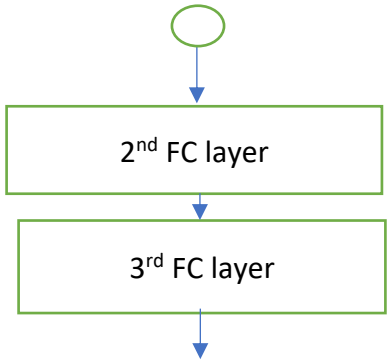
Construction of FC layers: (for LeNet5 architecture)(can be generalized)

- The SAAB coefficients of last conv layer are clustered into 120 clusters.
- Only images of a certain class are trained once, and clustered into 12 clusters. 10 classes have 120 clusters in total.

- Outputs are converted into one hot vector of 120D.
- Weights for FC layer are calculated using Least Square Regressor which maps previous layer outputs to current layer output found in the last step.
- These 120D vectors are fed as input to next FC layer which clusters them into 80D vectors by repeating the same clustering method here, 8 clusters for each class.
- These 80D are then converted to 10D in the last layer.
- Weights of each FC layer are calculated using Least Square Regressor.

Once the Feedforward model is generated by training data,
Following flow chart shows how FF-CNN is implemented for any test input:





Similarities:

- FF-CNN is inspired from BP-CNN and has a lot of architectural similarities.
  Both BP-CNN and FF-CNN have convolution layers followed by fully connected layers.
- Both learn from data. Weights for convolution and FC layers are found and optimized by Back propagation while in FF-CNN, weights for convolution layer are computed from the statistics of input image. We convert from spatial to spectral domain and use PCA to find most discriminative features and use them as features.
- They are both domain specific and we need to train new models for new domain.
- They are both vulnerable to adversarial attacks.

Differences:

- BP-CNN does feature extraction and classification using back propagation. It learns all the parameters on it's own. Even though it is very powerful to be able to do this, it acts very poorly in some cases eg. on slight perturbations in input data and we cannot change much in the architecture as it is end-to-end.
  FF-CNN is a feed forward method. Every layer learns it's parameters from output of previous layer and the first layer learns from the statistics of image itself. So it's much easier to understand when the model behaves poorly for some kind of input and change it accordingly. It's much more transparent.
- Even though both are data driven, FF-CNN learns from much less data compared to BP-CNN. BP-CNN needs large dataset to perform well. FF-CNN doesn't have any such requirement. Hence, for weakly supervised data or medical applications or any other field where labelling data is very expensive, FF-CNN has an upper hand over BP-CNN.
- Also, BP-CNN needs to run for some epochs till it optimizes all weights and hence takes much longer to converge than FF-CNN which is one pass algorithm.
- BP-CNN filters have much more discriminative power than FF-CNN filters. We can use ensemble methods to increase accuracy of FF-CNN to capture variability.

Explanation and Discussion:

- Conv layers are used to capture features and FC layers are used for classification just like in BP-CNN but in a one-pass manner.
- We convert spatial dimensions to spectral dimensions in FF-CNN.
- We use a DC component. So that we can ignore that during PCA.

- Bias is chosen such that it is the max of norm of input matrix x. So that, we have all positive features. This makes sure we have all positive features to avoid sign confusion in cascade system. Thus, we have replaced ReLU as ReLU keeps the positive values as it is and cuts of all negative values. Thus, we have captured both positive and negative correlation in our data.
- We use PCA to truncate the less discriminant spectral dimensions.
- Training class-wise and generating pseudo labels extract both inter-class and intra-class variability.

2) Image reconstructions from Saab coefficients

Approach:

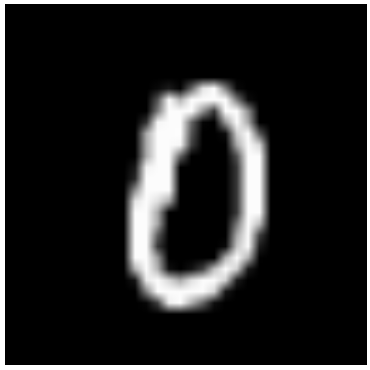
1. Define convolutional layer using following architectural settings:
   Kernel size= 4x4
   Non-overlapping with stride 4
   And train our SAAB model (Steps in part 1) using only 10000 training inputs. This function returns the model parameters.
2. Getfeature- Load four images and generate features for the model trained in above step.
3. Inverse_saab- For image reconstruction:
   1. Add bias of 2nd conv layer to its features (SAAB coefficients)
   2. Get features for 1st layer by getting inverse of kernel and image patch
   3. Add patch mean and feature mean back to the image patches
   4. Invert window_process and feed the patches to 1st conv layer.
   5. Repeat the same for 1st conv layer.
   6. Reconstruct the image from these patches.
   7. Calculate PSNR for 4 different kernel numbers:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

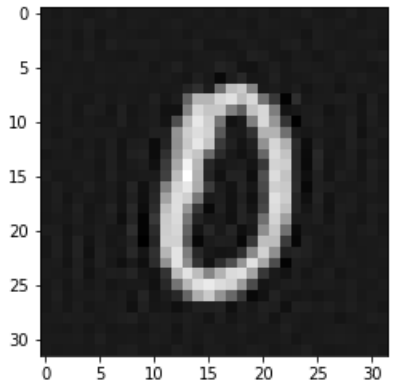
$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

Experimental Results:

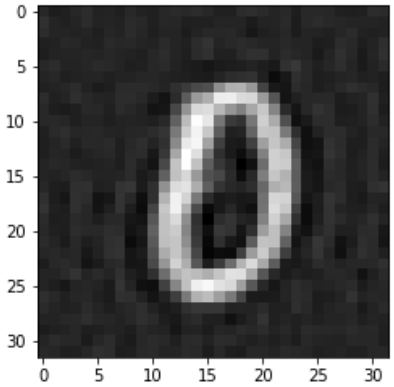
Original:



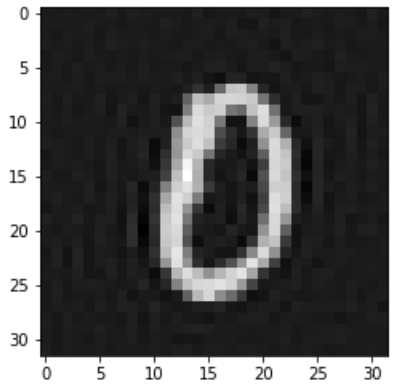
Kernel numbers :(10,120)
PSNR = 27.24207429231005



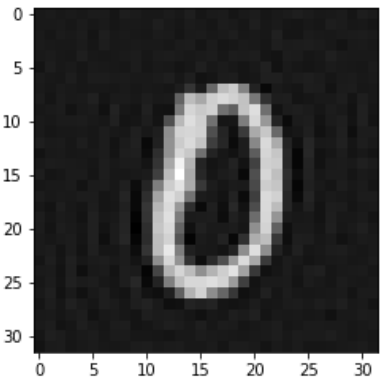
(10, 60)
22.111406774006678



(12, 120)
27.876777111598486

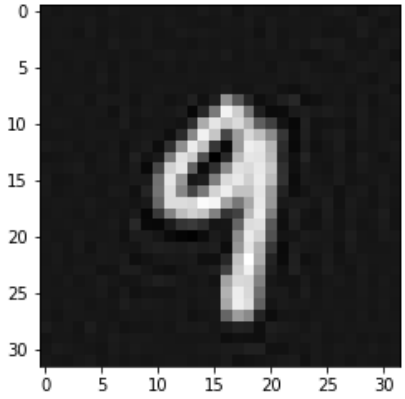


(14,120)
27.65259928234713

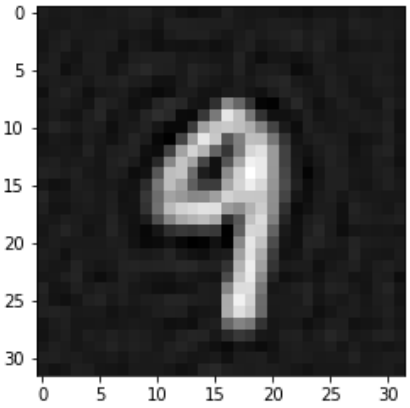


Original:

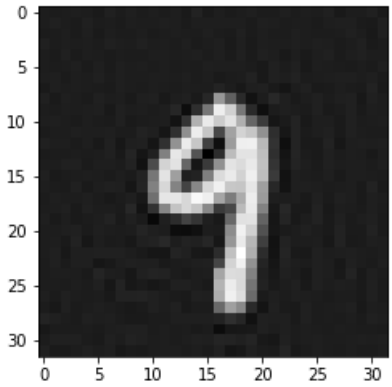
(10,120)
29.641964706799364



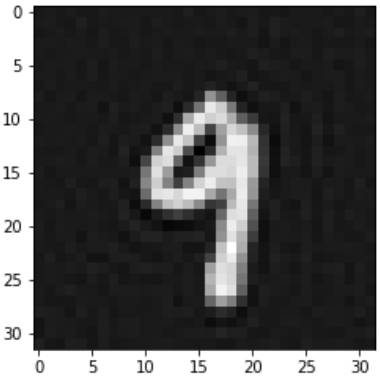
(10, 60)
24.125276090644512



(12, 120)
29.790325939556304

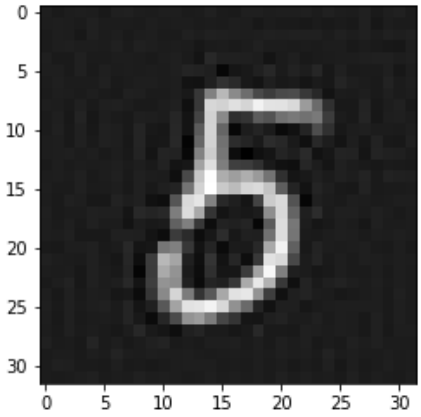


(14, 120)
29.731235126397312

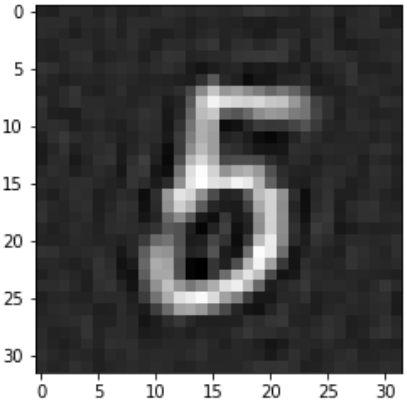


Original:

(10,120)
27.154410966514007

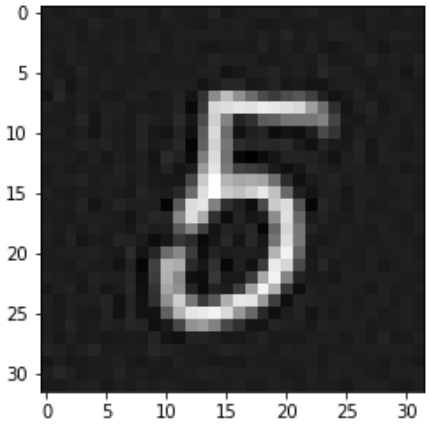


(10, 60)
22.023190284720574

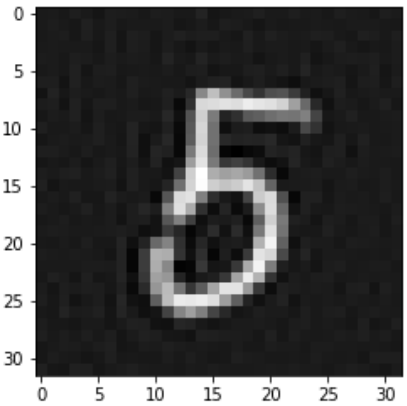


`

(12, 120)
27.651643806916613

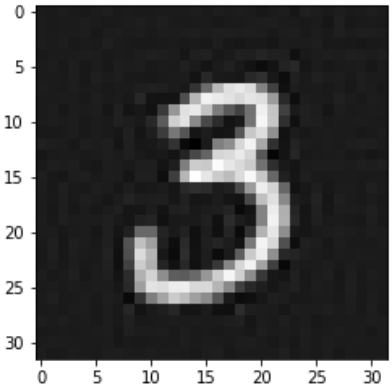


(14, 120)
27.741153250556934

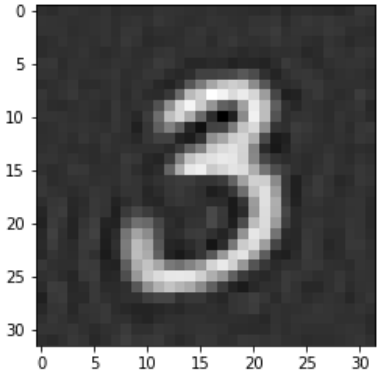


Original:

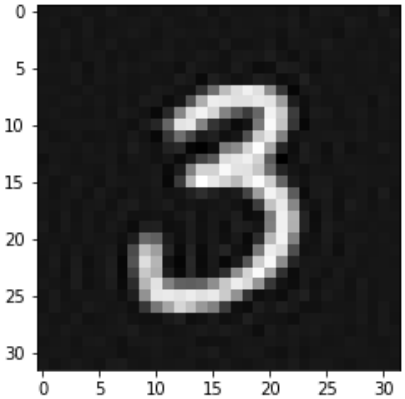
(10,120)
28.26805525598527



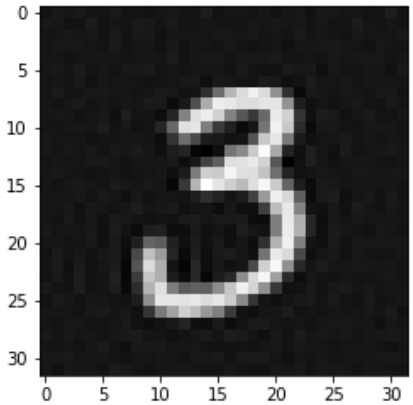
(10, 60)
23.89032239615566



(12, 120)
29.43175541342709



(14, 120)
29.629524155096902



Discussion:

- In image reconstruction, we invert the saab transform. When we reduce dimension using PCA, we don't get a loss less image reconstruction.
- While doing dimensionality reduction in SAAB, we change the number of kernels. When we use less number of kernels, we are reducing more dimensions. Thus, the image reconstructed is not so good quality. As the number of kernels increase, we are considering more dimensions and don't reduce many dimensions and inverse PCA works better and we see better output.
- Increasing the kernel size is equivalent to increasing number of components while doing PCA. Thus, less dimensionality reduction and hence we can see that the PSNR value increases with kernel number.
- If we keep no of kernels in any one layer constant and increase the number of kernels in other layer, the PSNR value increases. Thus, keeping more information at any stage of convolution helps increase PSNR value for reconstruction.

3) Handwritten digits recognition using ensembles of feedforward design

Approach:

1. Getkernel- Here we specify architectural settings:
   Kernel size: 3x3, 3x5, 5x3, 5x5 (4 different settings)
   Overlapping kernel with stride 1
   And train our SAAB model (Steps in part 1). This function returns the model parameters.
2 Getfeature- Loads MNIST dataset and does generates features for the model trained in above step.
3 Getweight- Training of the FC layer is done by this function. It does clustering according to LeNet5 architecture, with 120, 80, 10 clusters and hence does classification. (Steps in part 1)

Apart from the 4 different settings above, I have implemented 6 Law's filters,

For Law's filters:
Instead of passing image from training data, we filter it through 6 different Law's filters and feed it to Getkernel. Also, we have memory storage issue using this. Hence, we train using batch size of 200 samples for 10000 filtered training images.

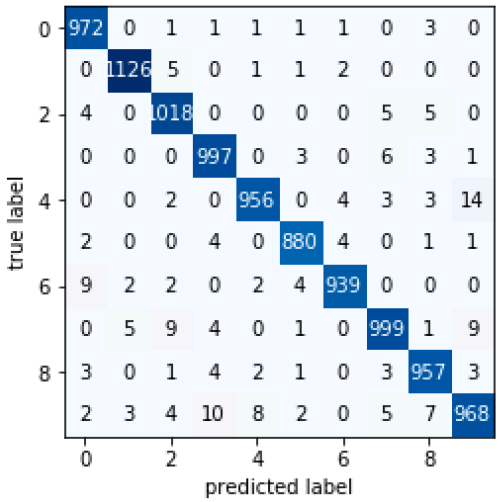
4 Now, we have outputs for each setting(60000x10)
5 We concatenate these to form a 60000x100 matrix.
6 We do PCA to reduce the dimension to 60000x10
7 Now use SVM or any other multi-class classifier using training samples from above step and find training accuracy.
8 Also find test accuracy for 10000 test samples.

Experimental Results:

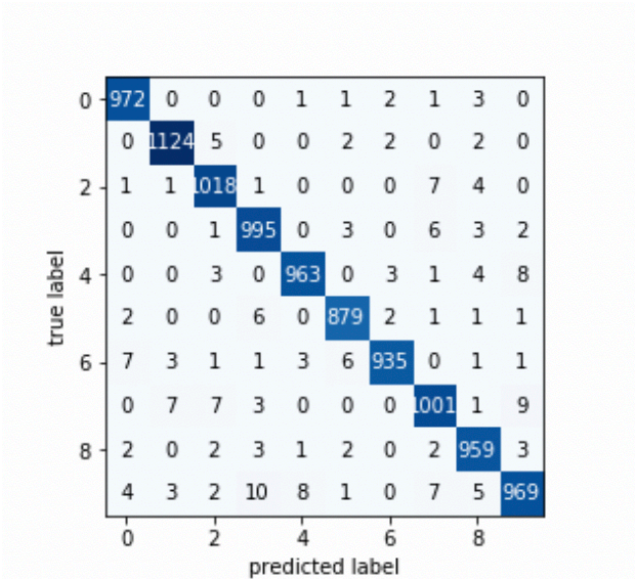
Training and testing classification accuracy:

	Kernel Size 5x5	Kernel Size 5x3	Kernel Size 3x5	Kernel Size 3x3	Law's Filter S5S5	Law's Filter S5W5	Law's Filter S5E5	Law's Filter S5L5	Law's Filter S5R5	Law's Filter W5L5	Ensemble
Train accuracy	0.9693	0.9720	0.9714	0.9684	0.9708	0.9611	0.9701	0.9712	0.9711	0.9677	0.9812
Test accuracy	0.9697	0.9727	0.9714	0.9658	0.9680	0.9658	0.9689	0.9707	0.9705	0.9671	0.9814

FF-CNN confusion matrix:



BP-CNN confusion matrix



Discussion:

- We have adopted ensembles method to increase the accuracy of our FF-CNN model. Use of ensemble helps capture the variance in our data thus allowing classification with better accuracy.
- We have used different kernel sizes and Law's filtered images to capture this variability. Law's filters are designed to capture specific patters and cascading them with our FF-CNN helps capture features. Different kernel sizes project images into different spectral dimensions.
- Confusion matrix show(in blue) show correct classification. Thus, we can see that misclassification is very less for both FF-CNN and BP-CNN.
- train_acc 0.9812166666666666, test_acc 0.9814 of FF-CNN on MNIST.
- Best Test_accuracy of BP-CNN was 0.9906. This value keeps on fluctuating by a small margin due to non-convex optimization. But, BP-CNN is slightly better than ensembled FF-CNN accuracy-wise.
- The error in both the models is about 1 and 1/2 percent. Their performance is very close to each other.

Certain ideas to improve FF-CNN and BP-CNN:

- Increase training data to improve BP-CNN accuracy.
- Train both models by augmenting data with some added random noise in it.
- Try adding unseen data and data with different capture conditions (like during day and night) for both.
- Try implementing more complex BP-CNN models than LeNet5 for better accuracy in FF-CNN.
- Try using different multiclass classification methods other than SVC.
- Try different filters from Signal and image processing to train different ensembles to capture the variability to increase performance of FF-CNN.

References:

- 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.
- Chen, Y., Yang, Y., Wang, W., & Kuo, C. C. J. (2019). Ensembles of feedforward-designed convolutional neural networks. arXiv preprint arXiv:1901.02154.
- https://github.com/davidsonic/Interpretable_CNNs_via_Feedforward_Design
- http://rasbt.github.io/mlxtend/api_subpackages/mlxtend.evaluate/

- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
-