EE 569 Homework VI

Royston Marian Mascarenhas

USC ID: 8286328166

Submission Date: April 28th, 2019

USC email: rmascare@usc.edu

Feedforward CNN Design

(a)

FF CNNs

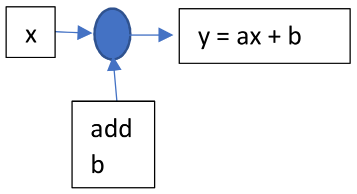
A Feed Forward Convolutional Neural Network (FF CNN) does not use back propagation to optimize the kernels of the convolutional layers. Therefore, it has less chance for error since the cost function of the output is not taken into consideration. Thus, the algorithm relies heavily on the information of the training data. In this rigorous data centric approach, the layer tries to extract not just spatial but spectral information too from the training data to gain a sharper insight into the features that connect that data point to a particular class. The network parameters are derived during the forward pass, from the statistics of the previous layer output.

Let's start building the FF CNN from the ground up.

Prerequisites:

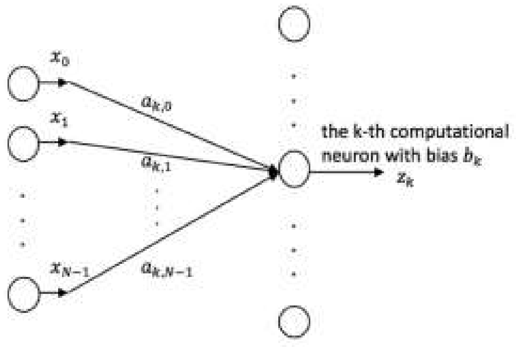
1. Neuron

The neuron is best explained by the following figure:



It can be divided into the affine component and the bias component. The input is a data point or a vector with N features or dimensions.

If N data points are taken into consideration, then we get the first layer of a neural network. It is visualized below:



Mathematically, it is expressed as

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

The output of this layer is fed to an activation function to enhance interpretability by introducing non linearity. ReLu output is expressed mathematically as the function which drops negative components and does nothing when it iterates over positive components.

$$z_k = \phi(y_k) = \max(0, y_k).$$

2. Non-linear activation:

Non linear activation helps differentiate between two patterns of opposite polarity. This is know as the **sign confusion problem.** Without non linear activation, the output from the convolution will nullify the difference in polarity. Since the negative outputs will be stemmed by ReLu, non linear activation helps eliminate lack of ambiguity between foreground and background of similar but complementary images. Another technique employed to override the sign confusion problem is the SAAK transform which splits positive/negative correlations into two channels. Since the computational cost increases due to the presence of two channels, the SAAB transform is used instead.

3. Span of anchor vectors:

Anchor vectors are those vectors which are determined during training and remain unchanged during the testing process. They can be thought of as correlators. If the feature and the weight are highly correlated, then the response is small and vice versa. These extract patterns from the input by thresholding. The span of the anchor vectors can be thought of as output vectors which are projections of input vectors into the weight space. If the subspace can be determined by PCA, the anchor vectors can in turn be calculated. Note that this prerequisite assumes zero bias term.

Feedforward design of convolutional layers:

The CONV layer design includes the following essential concepts:

Spatial Spectral Filtering:

1. Just spatial filtering is not enough. In CNNs, as the layers go deeper, the receptive field focuses on a particular spatial section of the input but the resolution is lost. The solution to this is to find a spectral representation of the neighbourhood through PCA to extract dominant stroke patterns. These are called anchor vectors.
2. Image representations and transforms are distinguished with the help of spectral aided spatial representations

Larger output:

Neglecting the boundary effect, if each pixel is involved in filtering, then the output is larger than the input by a factor of K. While this might hinder faster run time, it provides a more diverse feature set.

Saab Transform:

The SAAB transform is a process of selection of anchor vectors, which comprise of AC and DC components.

Selecting anchor vectors:

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

The DC and AC components of the anchor vectors are configured as above. As a result, the input vector space is a sum of the subspaces of the two components.

The relation between the AC an the DC components is:

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

where

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

The anchor vectors are obtained by selecting the first (k-1) principal components of the result of PCA applied on input vectors belonging to the AC subspace.

Bias Selection:

The conditions for selecting bias are that all bias terms should be equal and they should be more than the maximum component of the input vector so when the bias is added, all the terms in the input vector are positive. This second constraint simplifies the design and computation.

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

Spatial Pooling:

This helps in

1. picking the best filter responses
2. reducing computation
3. optimizing storage
4. deviant visual patterns are supressed

Together, convolution and pooling can be thought as projecting the input into the space spanned by the anchor vectors. Together, spatial filtering, spectral interpretation and max pooling can capture similar but spatially translated patterns.

Compound filters:

The collection of features extracted by the convolutional layers are the outputs of compound filters. In feed forward CNNs, the target is a linear combination of orthogonal PCA filter responses, which correspond to receptive field extractions.

Feed forward design of fully connected/dense layers.

- In FF CNNs, each FC output is a linear LSR. As a result, we get a one hot encoded output of the dense layer as that is the output of a linear least square regressor.
- Due to the lack of labels, we use k-means and classify the output space into a number of clusters. Once we have the new label and the original label, we further classify each cluster into a set of clusters. This is done to accommodate the diversity of each class. This process is known as **pseudo label generation.**
- In the Le-Net 5, we do k-means clustering to obtain a 120 one hot encoded vector with 10 final classes and 12 clusters within main clusters.
- The LSR ensures that all samples are mapped to the unit target vector itself and not any other target vector (another class with its own pseudolabels).
- Leakage of information happens when similar features lead to mapping from one pseudo space to the other, leading to faulty classification.
- The densely connected layers take the inputs from the conv layers and align the feature spaces corresponding to input and output. This is carried out to reduce the cross entropy and introduce more variance in each feature, resulting in more discriminating power.

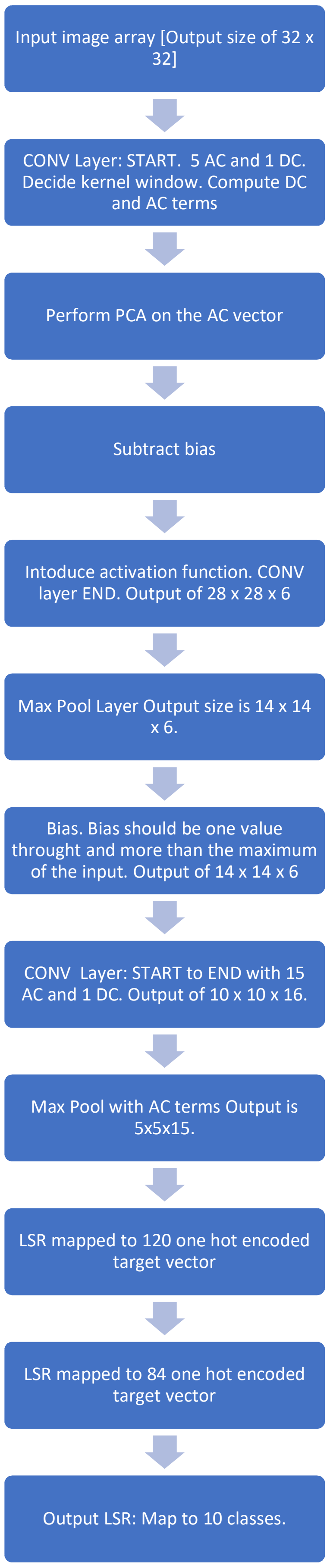
Now that we have explored the concepts, let us build the process.

Network architecture:

Layer	**Interpretation in FF CNN**	**Original LeNet output sizes and configurations**	**Modified Le-Net configurations**
CONV	SAAB (AC and DC) coefficient convolution	28 x 28 x 6 (5x 5 AC filters, totally 5 in number and 1 DC filter)	5 x 5 x 3
Max Pooling	Max Pooling	14 x 14 x 6	32
Bias addition	Bias addition	14 x 14 x 6	
CONV	SAAB (AC and DC) coefficient convolution	10 x 10 x 16 (5x 5 AC filters, totally 15 in number and 1 DC filter)	5 x 5 x 32
Max Pooling	Max Pooling of only AC responses	5 x 5 x 15	64
No Bias Addition	No need since the FC layers adopt a different approach.		
First FC Layer	Least Square Regressor	One hot encoded output with 120 columns	One hot encoded output with 200 columns

Second FC Layer	Least Square Regressor	One hot encoded output with 84 columns	One hot encoded output with 100 columns
Output layer	Least Square Regressor	One hot encoded output with 10 columns	One hot encoded output with 10 columns

Flow Chart:


Input image array [Output size of 32 x 32]
CONV Layer: START. 5 AC and 1 DC. Decide kernel window. Compute DC and AC terms
Perform PCA on the AC vector
Subtract bias
Intoduce activation function. CONV layer END. Output of 28 x 28 x 6
Max Pool Layer Output size is 14 x 14 x 6.
Bias. Bias should be one value throught and more than the maximum of the input. Output of 14 x 14 x 6
CONV Layer: START to END with 15 AC and 1 DC. Output of 10 x 10 x 16.
Max Pool with AC terms Output is 5x5x15.
LSR mapped to 120 one hot encoded target vector
LSR mapped to 84 one hot encoded target vector
Output LSR: Map to 10 classes.

Differences:

BP CNN	**FF CNN**
Background	
Works on **optimization** of a cost function	Works on retrieving **data** statistics from previous layer to compute parameters of next layer
The mathematical core of BP CNN is **non convex optimization**. Once the cost function is computed, its gradient is plotted and an optimization algorithm is used to find the global minimum. Since this is non convex optimization, the update might get stuck at a local minimum and there might be a need to use momentum.	The mathematical flesh of an FF CNN relies heavily on **linear algebra and statistics**. The convolution is based on PCA and it aids discriminant analysis and reduction of the features. One advantage it has over the non convex optimization is that the vanishing gradient problem does not occur in this data centric approach.
Structure and Perception	
The BP CNN utilizes information from the input and the output. The input has an effect on the initial layers and the output decision space has an effect on the final layers. It is **end to end coupled**.	The FF CNN is split explicitly into two modules. The **data representation module** is formed by the PCA based convolutional layers. The **decision making module** is formed by the LSR densely connected layers.
BP CNNs **are hard to interpret** because of the mathematical density that develops with each layer.	FF CNNs are mathematically **clear and concise.** Therefore, they are easier to understand.
To ensure end to end connectivity and optimization, the BP design has a **constraint** on its layers. The FC layers have to be ANNs.	There is **no such constraint** in FF CNNs. The LSRs can be replaced by SVMs, random forests or another classifier.
Complexity	
The BP algorithm might have to run for several **epochs** for it to converge. With increase in data size or dimensionality, the **convergence might take a lot of time.**	The complexity can be mitigated in FF CNNs. The covariance might converge in a single epoch in some cases since the training image provides **several training patches**. Moreover, the **correlation** between the training images and the patches **can be statistically manipulated to further lower the complexity.**
Versatility	
For any task such as detection, tracking, segmentation, different CNNs will have to be designed because of differing cost functions.	Since this approach is data centric, the same CNN can be used for all tasks. The only part we might need to change is the setting for the FC layers. Information from different such techniques can be then combined to build one system rather than several networks.
Performance	
Optimization aids state of the art performance only if the network	The performance can be boosted by ensemble learning since multiple classifiers

configurations are directly related to the problem statement at hand.	can be constituted by several convolutional layers.
Overfitting	
The number of learning variables might lead to overfitting. However, there are several regularization methods such as dropout and early stopping which can help reduce overfitting.	In FF CNN, the degrees of freedom are not directly related to the model parameters. The weights in the deeper layers bank on those in the shallower layers. Since the weights of the CONV layers are determined by PCA of the input data to that layer, there is a direct correlation between the weights and the training data. Therefore, once the covariance matrices are identified, we can find the filter coefficients. i.e if the input of the first conv layer is 5 x 5 and the second conv layer is 5 x 5 x 6 the covariance matrices are of dimensions 25 x 25 and 150 x 150. For the FC layers, the dimensions of the LSR matrix is approximately the product between the input and the output dimensions.
Discriminating power using signal analysis	
Cross entropy is used to approximate the discriminating power of features. In BP CNN, the cross entropy of the first two CONV layers is much lower than that of the FF CNN. As the layers go deeper, the cross entropy decreases and the correlation in the information becomes more transparent.	The cross entropy for a FF CNN is higher than that of a BP CNN in the initial CNN layers but it decreases drastically in the FC layers. As a result, it has more orthogonality, less redundancy and wider information (broad band) flow.

Similarity:

FF CNNs and BP CNNs are similar in their core structure. They use different implementations of a convolutional, max pooling and densely connected layer but the types of layers are the same. Both use the basic neuron structure and activation functions. FF CNN has no back propagation but it derives information from the forward pass. Both the BP and the FF CNN use the forward pass. Each CNN layer corresponds to a vector a matrix transformation in both cases. Although FF CNN is more statistic based, BP CNN is also data centric in the sense that it relies on vast amount of input features for best discrimination rather than just mathematical processes. In both types of CNNs, the spatial resolution reduces as the network goes deeper. In both CNNs, each dimension of the output refers to a class label. There are similar tunable parameters in either. Both have adjusted filter weights or learning parameters. Both are vulnerable to adversarial attacks when the model is fixed.

(b)

Reconstruction using SAAB coefficients

1. **Abstract and Motivation:**
   The SAAB filters are used to obtain a statistical input to the CONV layer on the basis of its principal components and adjusted bias. The images, once feature encoded by SAAB can also be reversed.

2. **Approach and Procedures:**

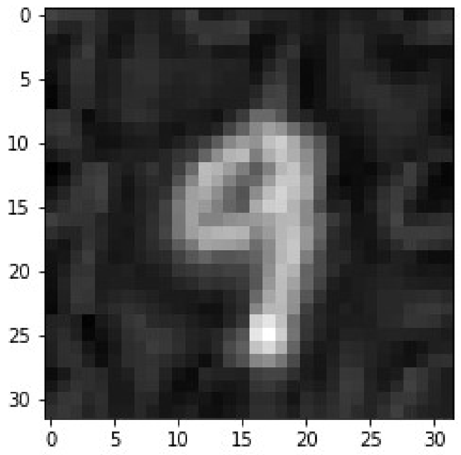
   The approach followed here is:

   1. Load the number of layers and iterate from the deeper layer to the shallower layer
   2. Obtain the kernel and the its inverse to resuscitate the transformed kernel. This is akin to the inverse PCA step.
   3. If outermost layer, the samples have bias. Remove the bias by subtraction.
   4. For both layers, revive the sample patches by reverse computation.
   5. Subtract the feature expectation from the sample patches.
   6. One last step is to perform dimension rearrangement to obtain the original patches.
   7. Reshape to input image.

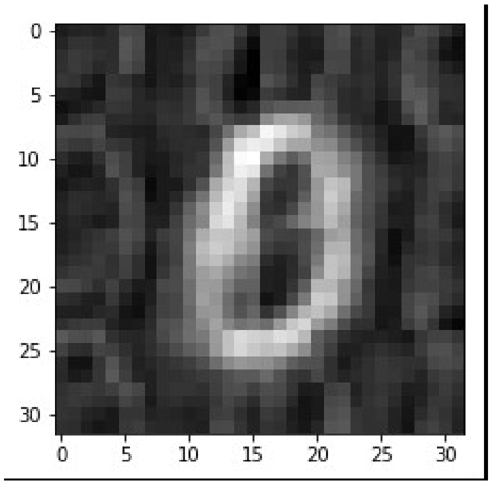
3. **Results:**
   Note that the dc component is discluded in the final number of filters.

PSNR values are given right below the image

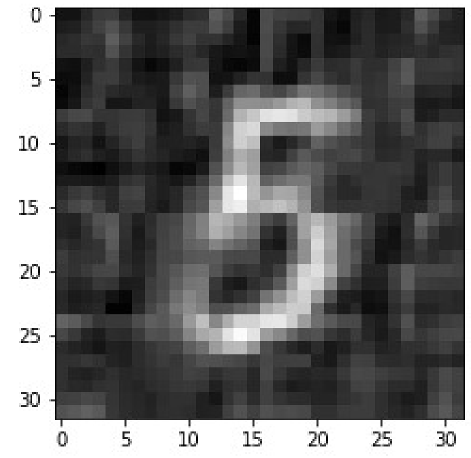
Setting 1:
Number of filters in first layer: 5
Number of filters in second layer: 15



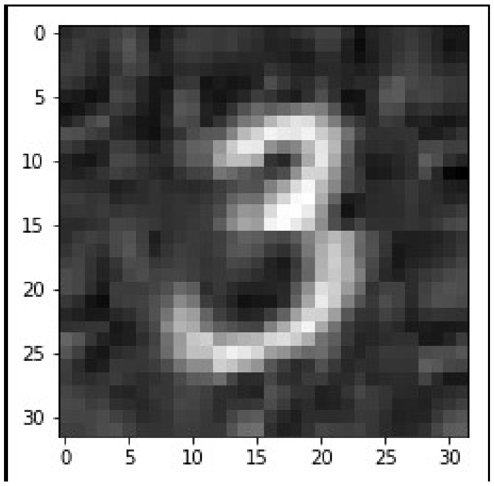
PSNR is 18.689149275274442 dB



PSNR is 16.63590590259514 dB

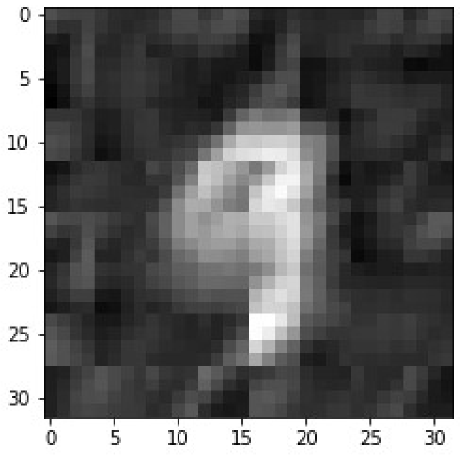


PSNR is 17.672289033793966 dB

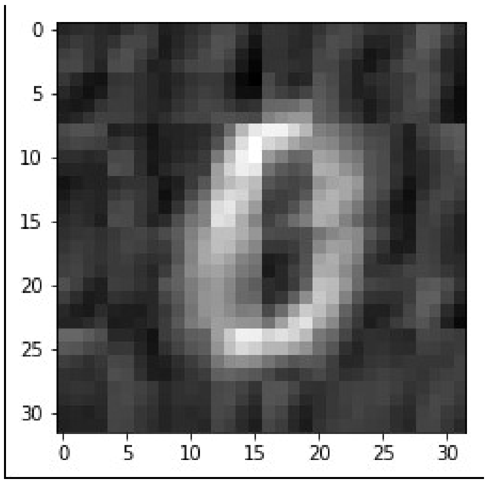


PSNR is 18.02334031191759 dB

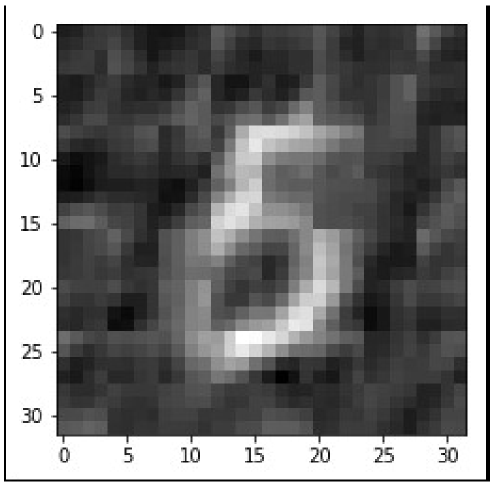
Setting 2:
Number of filters in first layer: 5
Number of filters in second layer: 11



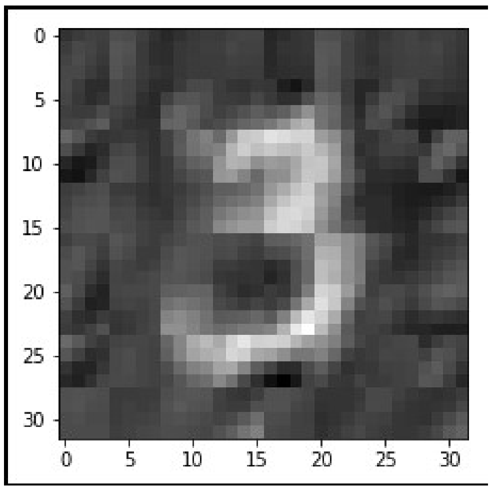
PSNR is 16.915136739446133 dB



PSNR is 15.856383840864002 dB



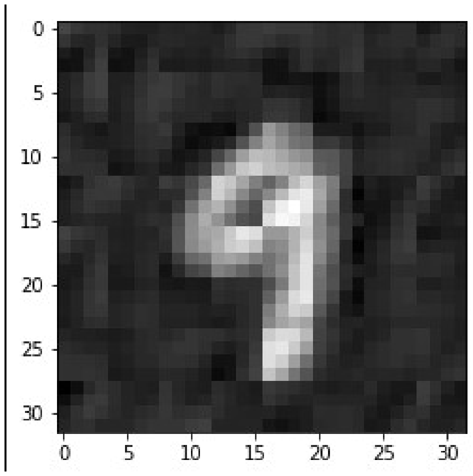
PSNR is 16.59938529803895 dB



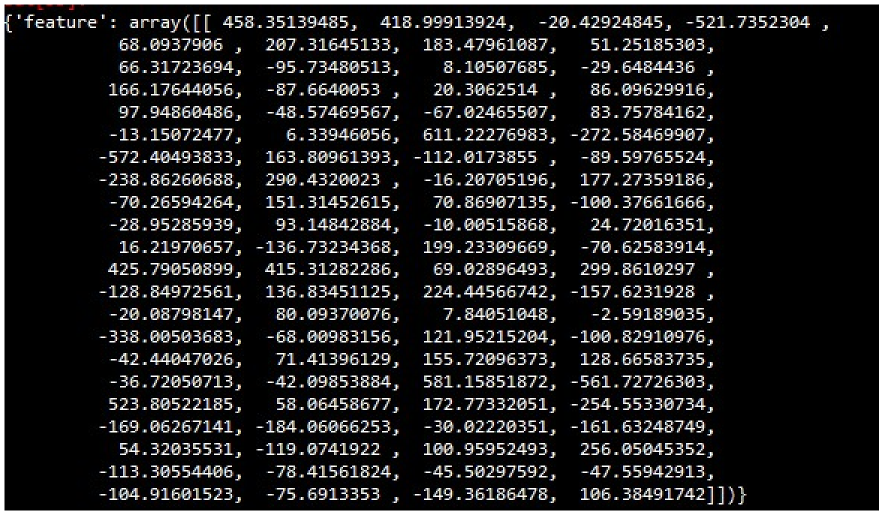
PSNR is 16.0894213600736 dB

Setting 3:
Number of filters in first layer: 3
Number of filters in second layer: 21

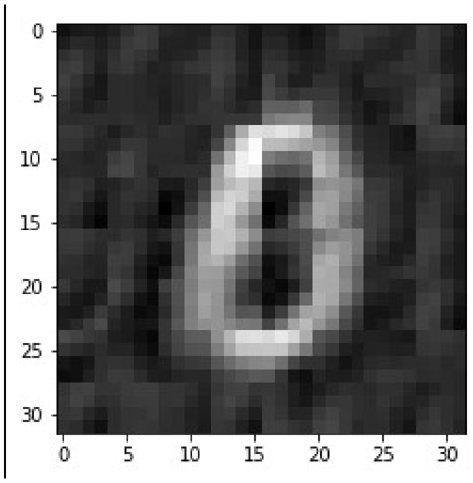
Feat refers to SAAB coefficients. They are in the shape 2x2xsecond stage kernel number but they are displayed in the form 1x(2x2xsecond stage kernel number)



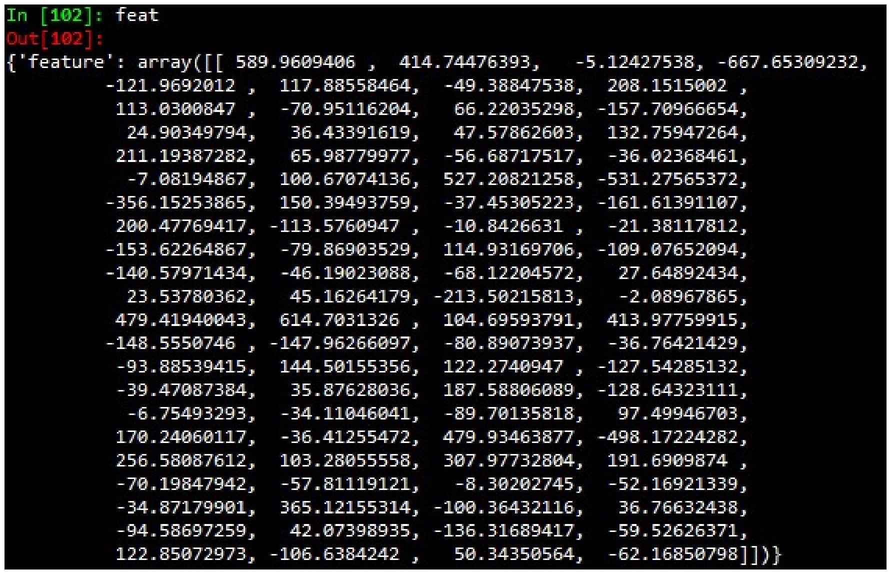
PSNR is 18.914979472919452 dB



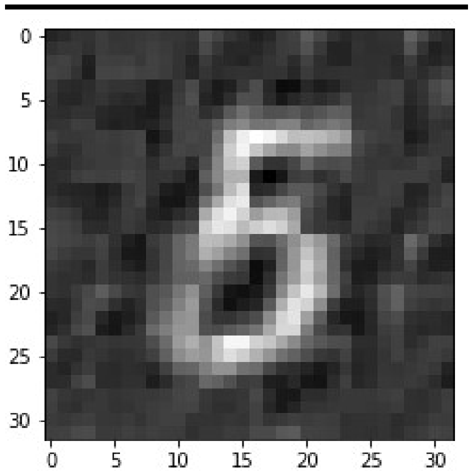
```
{'feature': array([[ 458.35139485,  418.99913924,  -20.42924845, -521.7352304 ,
           68.0937906 ,  207.31645133,  183.47961087,   51.25185303,
           66.31723694,  -95.73480513,    8.10507685,  -29.6484436 ,
          166.17644056,  -87.6640053 ,   20.3062514 ,   86.09629916,
           97.94860486,  -48.57469567,  -67.02465507,   83.75784162,
          -13.15072477,    6.33946056,  611.22276983, -272.58469907,
         -572.40493833,  163.80961393, -112.0173855 ,  -89.59765524,
         -238.86260688,  290.4320023 ,  -16.20705196,  177.27359186,
          -70.26594264,  151.31452615,   70.86907135, -100.37661666,
          -28.95285939,   93.14842884,  -10.00515868,   24.72016351,
           16.21970657, -136.73234368,  199.23309669,  -70.62583914,
          425.79050899,  415.31282286,   69.02896493,  299.8610297 ,
         -128.84972561,  136.83451125,  224.44566742, -157.6231928 ,
          -20.08798147,   80.09370076,    7.84051048,   -2.59189035,
         -338.00503683,  -68.00983156,  121.95215204, -100.82910976,
          -42.44047026,   71.41396129,  155.72096373,  128.66583735,
          -36.72050713,  -42.09853884,  581.15851872, -561.72726303,
          523.80522185,   58.06458677,  172.77332051, -254.55330734,
         -169.06267141, -184.06066253,  -30.02220351, -161.63248749,
           54.32035531, -119.0741922 ,  100.95952493,  256.05045352,
         -113.30554406,  -78.41561824,  -45.50297592,  -47.55942913,
         -104.91601523,  -75.6913353 , -149.36186478,  106.38491742]])}
```



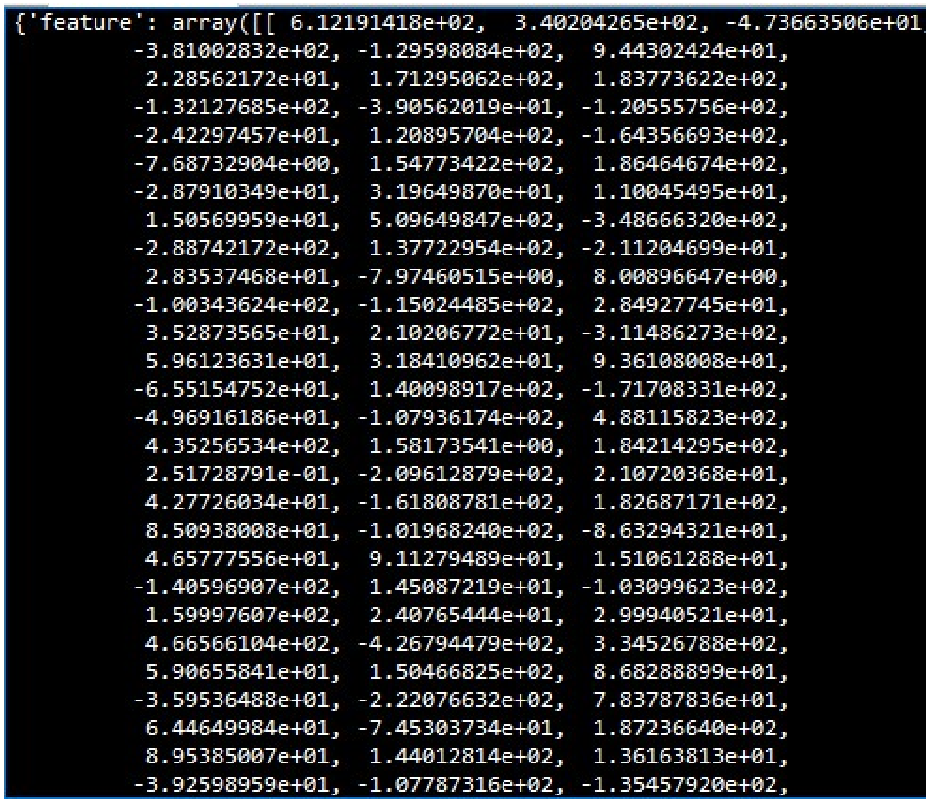
PSNR is 16.89226757315203 dB



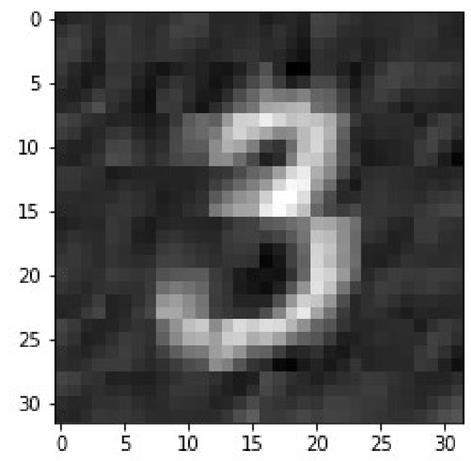
```
In [102]: feat
Out[102]:
{'feature': array([[ 589.9609406 ,  414.74476393,   -5.12427538, -667.65309232,
         -121.9692012 ,  117.88558464,  -49.38847538,  208.1515002 ,
          113.0300847 ,  -70.95116204,   66.22035298, -157.70966654,
           24.90349794,   36.43391619,   47.57862603,  132.75947264,
          211.19387282,   65.98779977,  -56.68717517,  -36.02368461,
           -7.08194867,  100.67074136,  527.20821258, -531.27565372,
         -356.15253865,  150.39493759,  -37.45305223, -161.61391107,
          200.47769417, -113.5760947 ,  -10.8426631 ,  -21.38117812,
         -153.62264867,  -79.86903529,  114.93169706, -109.07652094,
         -140.57971434,  -46.19023088,  -68.12204572,   27.64892434,
           23.53780362,   45.16264179, -213.50215813,   -2.08967865,
          479.41940043,  614.7031326 ,  104.69593791,  413.97759915,
         -148.5550746 , -147.96266097,  -80.89073937,  -36.76421429,
          -93.88539415,  144.50155356,  122.2740947 , -127.54285132,
          -39.47087384,   35.87628036,  187.58806089, -128.64323111,
           -6.75493293,  -34.11046041,  -89.70135818,   97.49946703,
          170.24060117,  -36.41255472,  479.93463877, -498.17224282,
          256.58087612,  103.28055558,  307.97732804,  191.6909874 ,
          -70.19847942,  -57.81119121,   -8.30202745,  -52.16921339,
          -34.87179901,  365.12155314, -100.36432116,   36.76632438,
          -94.58697259,   42.07398935, -136.31689417,  -59.52626371,
          122.85072973, -106.6384242 ,   50.34350564,  -62.16850798]])}
```



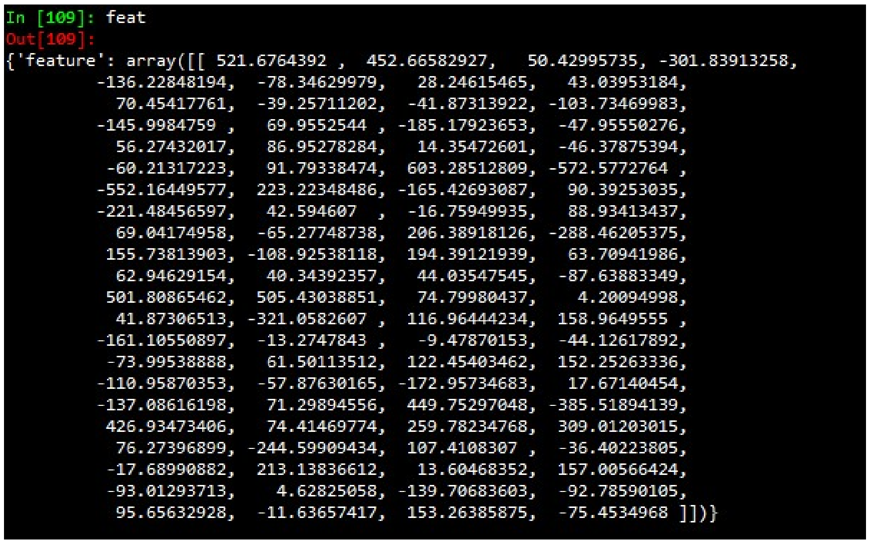
PSNR is 18.086554001146062 dB



```
{'feature': array([[ 6.12191418e+02,  3.40204265e+02, -4.73663506e+01
         -3.81002832e+02, -1.29598084e+02,  9.44302424e+01,
          2.28562172e+01,  1.71295062e+02,  1.83773622e+02,
         -1.32127685e+02, -3.90562019e+01, -1.20555756e+02,
         -2.42297457e+01,  1.20895704e+02, -1.64356693e+02,
         -7.68732904e+00,  1.54773422e+02,  1.86464674e+02,
         -2.87910349e+01,  3.19649870e+01,  1.10045495e+01,
          1.50569959e+01,  5.09649847e+02, -3.48666320e+02,
         -2.88742172e+02,  1.37722954e+02, -2.11204699e+01,
          2.83537468e+01, -7.97460515e+00,  8.00896647e+00,
         -1.00343624e+02, -1.15024485e+02,  2.84927745e+01,
          3.52873565e+01,  2.10206772e+01, -3.11486273e+02,
          5.96123631e+01,  3.18410962e+01,  9.36108008e+01,
         -6.55154752e+01,  1.40098917e+02, -1.71708331e+02,
         -4.96916186e+01, -1.07936174e+02,  4.88115823e+02,
          4.35256534e+02,  1.58173541e+00,  1.84214295e+02,
          2.51728791e-01, -2.09612879e+02,  2.10720368e+01,
          4.27726034e+01, -1.61808781e+02,  1.82687171e+02,
          8.50938008e+01, -1.01968240e+02, -8.63294321e+01,
          4.65777556e+01,  9.11279489e+01,  1.51061288e+01,
         -1.40596907e+02,  1.45087219e+01, -1.03099623e+02,
          1.59997607e+02,  2.40765444e+01,  2.99940521e+01,
          4.66566104e+02, -4.26794479e+02,  3.34526788e+02,
          5.90655841e+01,  1.50466825e+02,  8.68288899e+01,
         -3.59536488e+01, -2.22076632e+02,  7.83787836e+01,
          6.44649984e+01, -7.45303734e+01,  1.87236640e+02,
          8.95385007e+01,  1.44012814e+02,  1.36163813e+01,
         -3.92598959e+01, -1.07787316e+02, -1.35457920e+02,
```



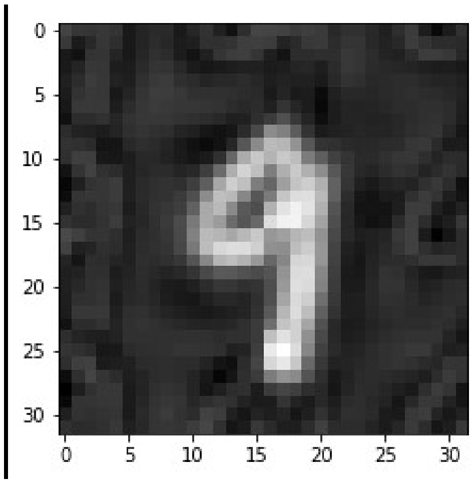
PSNR is 17.763986201525718 dB



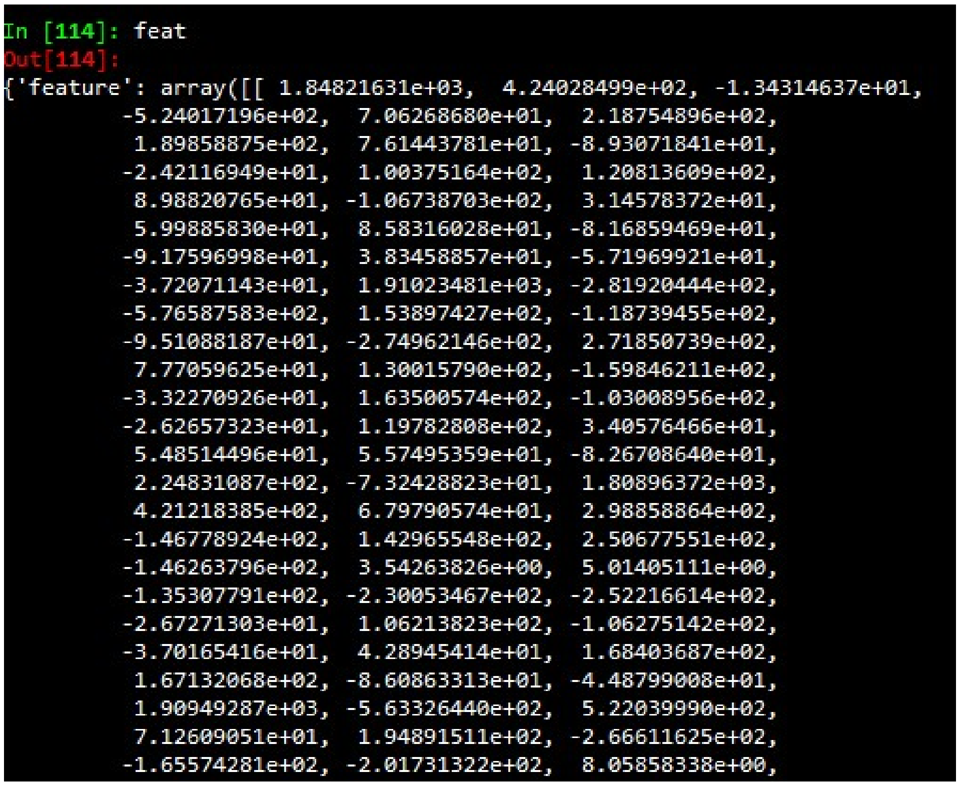
```
In [109]: feat
Out[109]:
{'feature': array([[ 521.6764392 ,  452.66582927,   50.42995735, -301.83913258,
        -136.22848194,  -78.34629979,   28.24615465,   43.03953184,
          70.45417761,  -39.25711202,  -41.87313922, -103.73469983,
        -145.9984759 ,   69.9552544 , -185.17923653,  -47.95550276,
          56.27432017,   86.95278284,   14.35472601,  -46.37875394,
         -60.21317223,   91.79338474,  603.28512809, -572.5772764 ,
        -552.16449577,  223.22348486, -165.42693087,   90.39253035,
        -221.48456597,   42.594607  ,  -16.75949935,   88.93413437,
          69.04174958,  -65.27748738,  206.38918126, -288.46205375,
         155.73813903, -108.92538118,  194.39121939,   63.70941986,
          62.94629154,   40.34392357,   44.03547545,  -87.63883349,
         501.80865462,  505.43038851,   74.79980437,    4.20094998,
          41.87306513, -321.0582607 ,  116.96444234,  158.9649555 ,
        -161.10550897,  -13.2747843 ,   -9.47870153,  -44.12617892,
         -73.99538888,   61.50113512,  122.45403462,  152.25263336,
        -110.95870353,  -57.87630165, -172.95734683,   17.67140454,
        -137.08616198,   71.29894556,  449.75297048, -385.51894139,
         426.93473406,   74.41469774,  259.78234768,  309.01203015,
          76.27396899, -244.59909434,  107.4108307 ,  -36.40223805,
         -17.68990882,  213.13836612,   13.60468352,  157.00566424,
         -93.01293713,    4.62825058, -139.70683603,  -92.78590105,
          95.65632928,  -11.63657417,  153.26385875,  -75.4534968 ]])}
```

Setting 4:
Number of filters in first layer: 14
Number of filters in second layer: 22

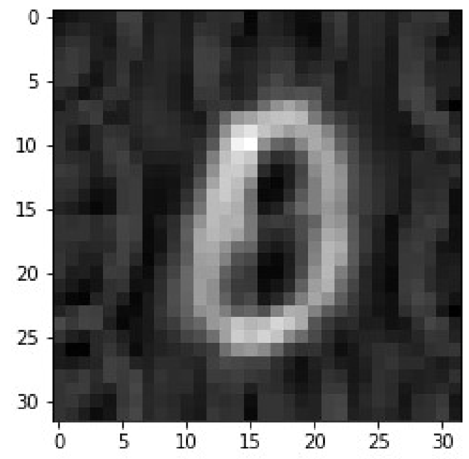
Feat refers to SAAB coefficients. They are in the shape 2x2xsecond stage kernel number but they are displayed in the form 1x(2x2xsecond stage kernel number). Here, the numbers include the dc components of the filters.



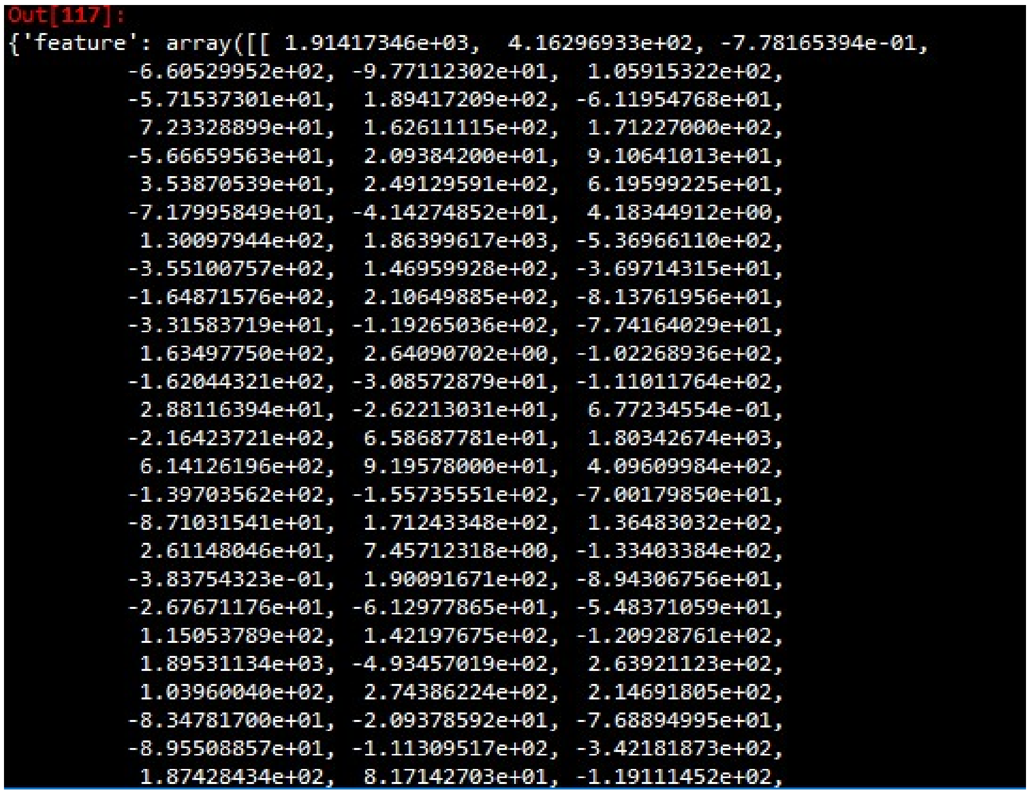
PSNR is 20.608254115070906 dB



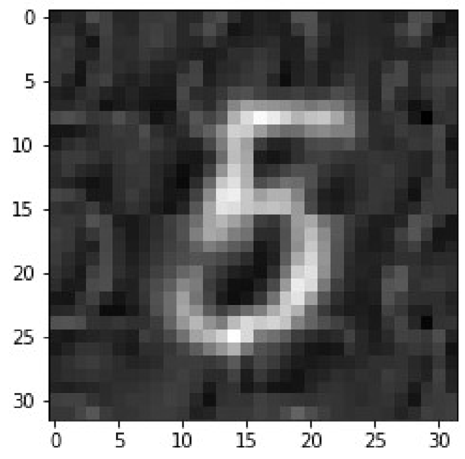
```
In [114]: feat
Out[114]:
{'feature': array([[ 1.84821631e+03,  4.24028499e+02, -1.34314637e+01,
         -5.24017196e+02,  7.06268680e+01,  2.18754896e+02,
          1.89858875e+02,  7.61443781e+01, -8.93071841e+01,
         -2.42116949e+01,  1.00375164e+02,  1.20813609e+02,
          8.98820765e+01, -1.06738703e+02,  3.14578372e+01,
          5.99885830e+01,  8.58316028e+01, -8.16859469e+01,
         -9.17596998e+01,  3.83458857e+01, -5.71969921e+01,
         -3.72071143e+01,  1.91023481e+03, -2.81920444e+02,
         -5.76587583e+02,  1.53897427e+02, -1.18739455e+02,
         -9.51088187e+01, -2.74962146e+02,  2.71850739e+02,
          7.77059625e+01,  1.30015790e+02, -1.59846211e+02,
         -3.32270926e+01,  1.63500574e+02, -1.03008956e+02,
         -2.62657323e+01,  1.19782808e+02,  3.40576466e+01,
          5.48514496e+01,  5.57495359e+01, -8.26708640e+01,
          2.24831087e+02, -7.32428823e+01,  1.80896372e+03,
          4.21218385e+02,  6.79790574e+01,  2.98858864e+02,
         -1.46778924e+02,  1.42965548e+02,  2.50677551e+02,
         -1.46263796e+02,  3.54263826e+00,  5.01405111e+00,
         -1.35307791e+02, -2.30053467e+02, -2.52216614e+02,
         -2.67271303e+01,  1.06213823e+02, -1.06275142e+02,
         -3.70165416e+01,  4.28945414e+01,  1.68403687e+02,
          1.67132068e+02, -8.60863313e+01, -4.48799008e+01,
          1.90949287e+03, -5.63326440e+02,  5.22039990e+02,
          7.12609051e+01,  1.94891511e+02, -2.66611625e+02,
         -1.65574281e+02, -2.01731322e+02,  8.05858338e+00,
```



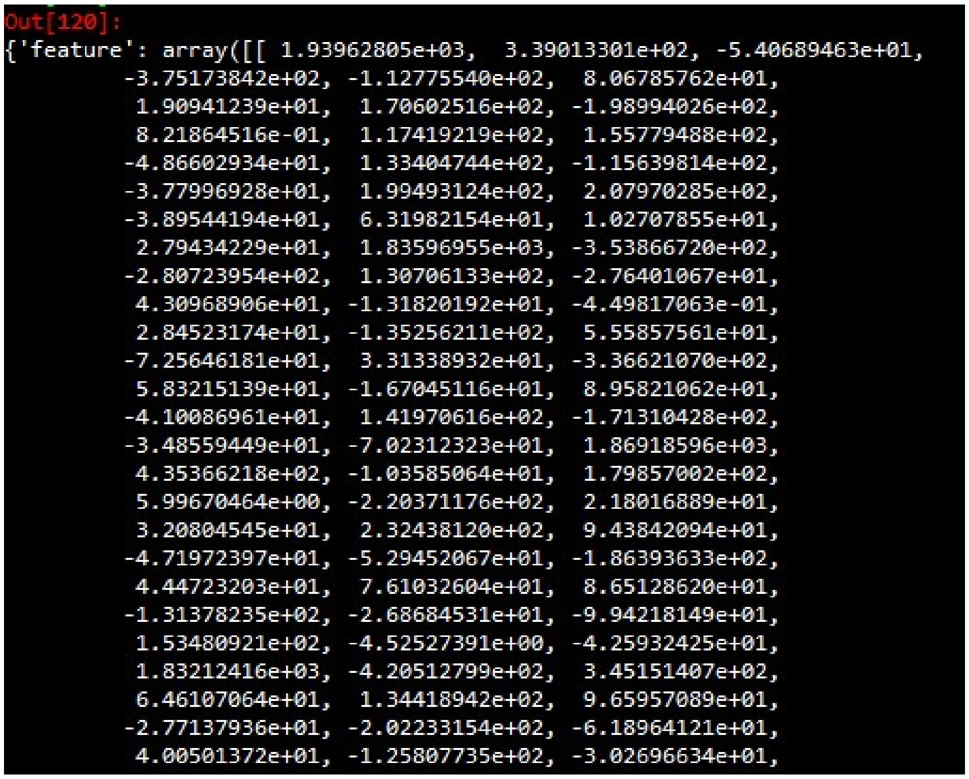
PSNR is 17.773702314861374 dB



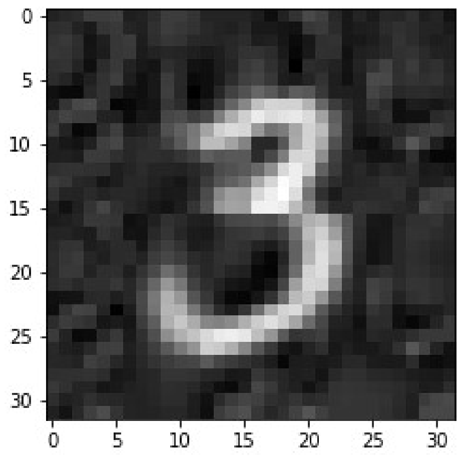
```
Out[117]:
{'feature': array([[ 1.91417346e+03,  4.16296933e+02, -7.78165394e-01,
         -6.60529952e+02, -9.77112302e+01,  1.05915322e+02,
         -5.71537301e+01,  1.89417209e+02, -6.11954768e+01,
          7.23328899e+01,  1.62611115e+02,  1.71227000e+02,
         -5.66659563e+01,  2.09384200e+01,  9.10641013e+01,
          3.53870539e+01,  2.49129591e+02,  6.19599225e+01,
         -7.17995849e+01, -4.14274852e+01,  4.18344912e+00,
          1.30097944e+02,  1.86399617e+03, -5.36966110e+02,
         -3.55100757e+02,  1.46959928e+02, -3.69714315e+01,
         -1.64871576e+02,  2.10649885e+02, -8.13761956e+01,
         -3.31583719e+01, -1.19265036e+02, -7.74164029e+01,
          1.63497750e+02,  2.64090702e+00, -1.02268936e+02,
         -1.62044321e+02, -3.08572879e+01, -1.11011764e+02,
          2.88116394e+01, -2.62213031e+01,  6.77234554e-01,
         -2.16423721e+02,  6.58687781e+01,  1.80342674e+03,
          6.14126196e+02,  9.19578000e+01,  4.09609984e+02,
         -1.39703562e+02, -1.55735551e+02, -7.00179850e+01,
         -8.71031541e+01,  1.71243348e+02,  1.36483032e+02,
          2.61148046e+01,  7.45712318e+00, -1.33403384e+02,
         -3.83754323e-01,  1.90091671e+02, -8.94306756e+01,
         -2.67671176e+01, -6.12977865e+01, -5.48371059e+01,
          1.15053789e+02,  1.42197675e+02, -1.20928761e+02,
          1.89531134e+03, -4.93457019e+02,  2.63921123e+02,
          1.03960040e+02,  2.74386224e+02,  2.14691805e+02,
         -8.34781700e+01, -2.09378592e+01, -7.68894995e+01,
         -8.95508857e+01, -1.11309517e+02, -3.42181873e+02,
          1.87428434e+02,  8.17142703e+01, -1.19111452e+02,
```



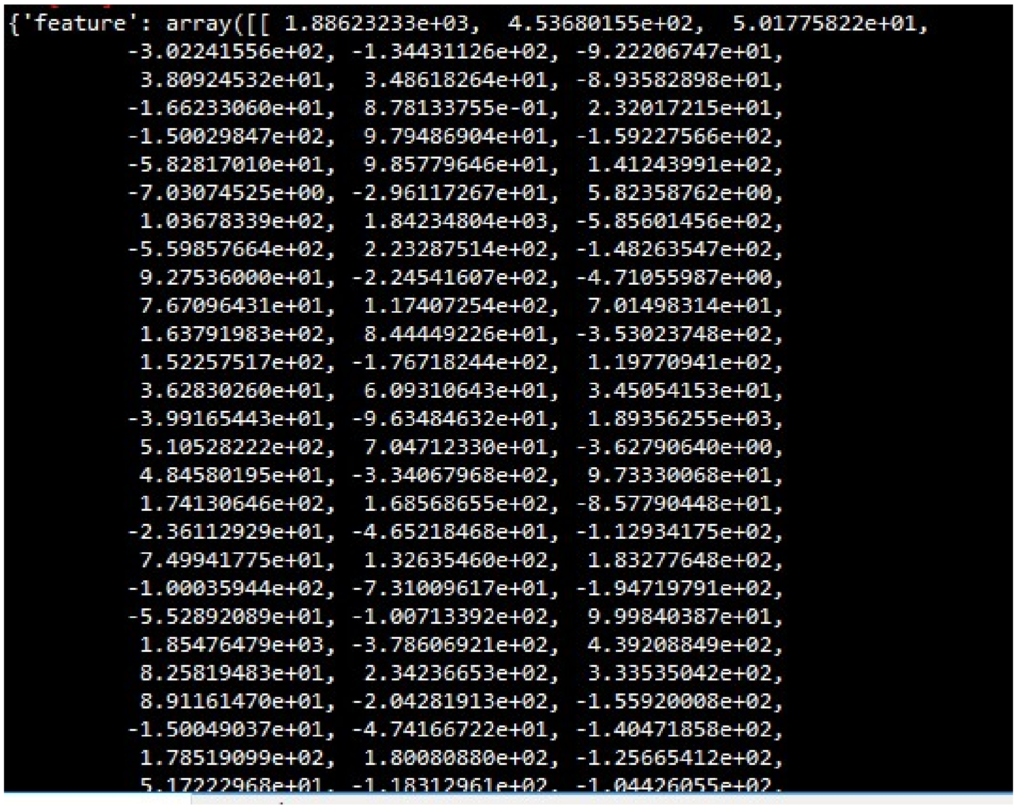
PSNR is 19.017636874705914 dB



```
Out[120]:
{'feature': array([[ 1.93962805e+03,  3.39013301e+02, -5.40689463e+01,
         -3.75173842e+02, -1.12775540e+02,  8.06785762e+01,
          1.90941239e+01,  1.70602516e+02, -1.98994026e+02,
          8.21864516e-01,  1.17419219e+02,  1.55779488e+02,
         -4.86602934e+01,  1.33404744e+02, -1.15639814e+02,
         -3.77996928e+01,  1.99493124e+02,  2.07970285e+02,
         -3.89544194e+01,  6.31982154e+01,  1.02707855e+01,
          2.79434229e+01,  1.83596955e+03, -3.53866720e+02,
         -2.80723954e+02,  1.30706133e+02, -2.76401067e+01,
          4.30968906e+01, -1.31820192e+01, -4.49817063e-01,
          2.84523174e+01, -1.35256211e+02,  5.55857561e+01,
         -7.25646181e+01,  3.31338932e+01, -3.36621070e+02,
          5.83215139e+01, -1.67045116e+01,  8.95821062e+01,
         -4.10086961e+01,  1.41970616e+02, -1.71310428e+02,
         -3.48559449e+01, -7.02312323e+01,  1.86918596e+03,
          4.35366218e+02, -1.03585064e+01,  1.79857002e+02,
          5.99670464e+00, -2.20371176e+02,  2.18016889e+01,
          3.20804545e+01,  2.32438120e+02,  9.43842094e+01,
         -4.71972397e+01, -5.29452067e+01, -1.86393633e+02,
          4.44723203e+01,  7.61032604e+01,  8.65128620e+01,
         -1.31378235e+02, -2.68684531e+01, -9.94218149e+01,
          1.53480921e+02, -4.52527391e+00, -4.25932425e+01,
          1.83212416e+03, -4.20512799e+02,  3.45151407e+02,
          6.46107064e+01,  1.34418942e+02,  9.65957089e+01,
         -2.77137936e+01, -2.02233154e+02, -6.18964121e+01,
          4.00501372e+01, -1.25807735e+02, -3.02696634e+01,
```



PSNR is 19.389316286452946 dB



```
{'feature': array([[ 1.88623233e+03,  4.53680155e+02,  5.01775822e+01,
        -3.02241556e+02, -1.34431126e+02, -9.22206747e+01,
         3.80924532e+01,  3.48618264e+01, -8.93582898e+01,
        -1.66233060e+01,  8.78133755e-01,  2.32017215e+01,
        -1.50029847e+02,  9.79486904e+01, -1.59227566e+02,
        -5.82817010e+01,  9.85779646e+01,  1.41243991e+02,
        -7.03074525e+00, -2.96117267e+01,  5.82358762e+00,
         1.03678339e+02,  1.84234804e+03, -5.85601456e+02,
        -5.59857664e+02,  2.23287514e+02, -1.48263547e+02,
         9.27536000e+01, -2.24541607e+02, -4.71055987e+00,
         7.67096431e+01,  1.17407254e+02,  7.01498314e+01,
         1.63791983e+02,  8.44449226e+01, -3.53023748e+02,
         1.52257517e+02, -1.76718244e+02,  1.19770941e+02,
         3.62830260e+01,  6.09310643e+01,  3.45054153e+01,
        -3.99165443e+01, -9.63484632e+01,  1.89356255e+03,
         5.10528222e+02,  7.04712330e+01, -3.62790640e+00,
         4.84580195e+01, -3.34067968e+02,  9.73330068e+01,
         1.74130646e+02,  1.68568655e+02, -8.57790448e+01,
        -2.36112929e+01, -4.65218468e+01, -1.12934175e+02,
         7.49941775e+01,  1.32635460e+02,  1.83277648e+02,
        -1.00035944e+02, -7.31009617e+01, -1.94719791e+02,
        -5.52892089e+01, -1.00713392e+02,  9.99840387e+01,
         1.85476479e+03, -3.78606921e+02,  4.39208849e+02,
         8.25819483e+01,  2.34236653e+02,  3.33535042e+02,
         8.91161470e+01, -2.04281913e+02, -1.55920008e+02,
        -1.50049037e+01, -4.74166722e+01, -1.40471858e+02,
         1.78519099e+02,  1.80080880e+02, -1.25665412e+02,
         5.17222968e+01, -1.18312961e+02, -1.04426055e+02,
```

4. Discussion:

- Note that the reconstruction offers a good approximation to the original image.
- If the number of kernels filters are decreased in both the layers, then there is loss of information for sufficient statistics and therefore, there is a drop in the PSNR as can be observed from setting 2.
- In setting 1, the standard for the PSNR value is set. The other settings are compared to this setting.
- In setting 3, we can see that the number of second layer filters has been increased, which is supposed to increase the fine detail information encoded in the features because it is the deeper layer but the results are not very encouraging as the increase in second layer CONV filters is of not much use as the general features are not exploited by the first layer filters which are only 3 in number. As a result, we observe stagnant PSNR growth over this promising setting.
- In setting 4, the increase in filters in both the layers boosts the PSNR by about 2.5 dB as more information is extracted by the filters. As a result, the principle components are richer and the reconstructed images are more faithful to their original constitution.

(c)

Ensembles of FF CNN Design

1. Abstract and Motivation:

Since FF CNN is more data centric than optimization based, more statistics will help augment the accuracy of the system. Therefore, the ensemble model is used to include model diversity so that better predictions can be made.

2. Approach:

The algorithm used in this case is :

1. Run the FF CNN algorithm for different values of number of filters, kernel sizes and different law filters.
2. In each case, record the training accuracy and the testing accuracy.
3. Save the weights and features produced by the algorithm.
4. Once all ten settings are complete, merge all features horizontally so that a lot of discriminating power is created.
5. Use PCA to reduce the dimensionality of the data stack in order to speed up the system.
6. Feed the data and the features to a classification algorithm, in this case, SVM.
7. Note the output scores of the ensemble algorithm.

3. Results:

Section 1 :Individual accuracies for training and testing:

Settings adopted (The stride is taken 1 in all cases):

Note that the number of filters in each CONV layer as mentioned below ignores the (+1) filter (DC component). In actuality, in setting 1 for example, the number of filters in each layer are 6,16. In some cases, like the later settings, lower number of filters are taken because of strain on computational resources.

1. Kernel Size (layer 1, Layer2): 5 x 5, 5 x 5
   Number of filters in CONV layer 1: 5
   Number of filters in CONV layer 2: 15

2. Kernel Size: (layer 1, Layer2): 5 x 5, 5 x 5
   Number of filters in CONV layer 1: 3
   Number of filters in CONV layer 2: 13

3. Kernel Size: (layer 1, Layer2): 5 x 5, 5 x 5
   Number of filters in CONV layer 1: 5
   Number of filters in CONV layer 2: 11

4. Kernel Size: (layer 1, Layer2): 5 x 5, 3 x 3

Number of filters in CONV layer 1: 5
Number of filters in CONV layer 2: 15

5. Kernel Size: (layer 1, Layer2): 3 x 3, 3 x 3
   Number of filters in CONV layer 1: 5
   Number of filters in CONV layer 2: 15

6. Kernel Size: (layer 1, Layer2): 3 x 3, 5 x 5
   Number of filters in CONV layer 1: 5
   Number of filters in CONV layer 2: 15

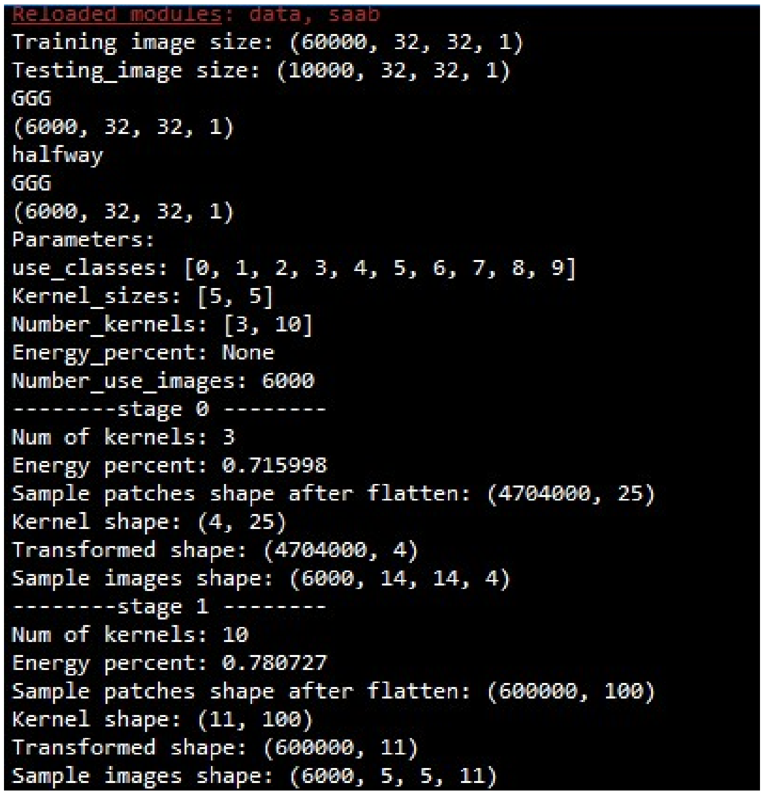
7. Kernel Size: (layer 1, Layer2): 3 x 3, 3 x 3
   Number of filters in CONV layer 1: 7
   Number of filters in CONV layer 2: 17

8. Kernel Size: (layer 1, Layer2): 5 x 5, 5 x 5
   Number of filters in CONV layer 1: 3
   Number of filters in CONV layer 2: 11
   E5S5 Law filter application

9. Kernel Size: (layer 1, Layer2): 5 x 5, 3 x 3
   Number of filters in CONV layer 1: 3
   Number of filters in CONV layer 2: 11
   Neighbourhood addition

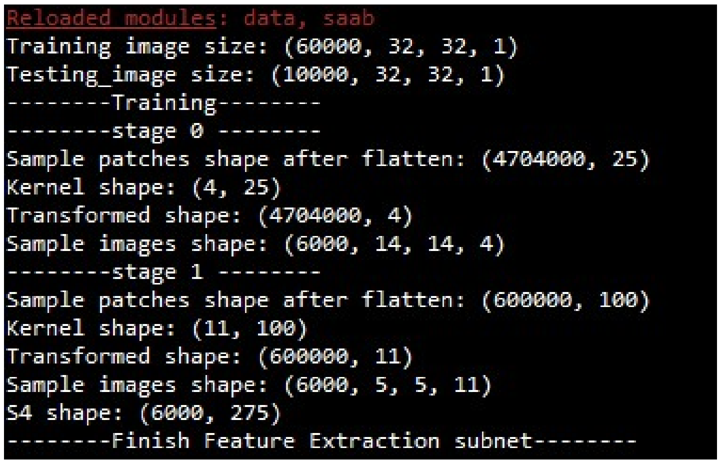
10. Kernel Size: (layer 1, Layer2): 5 x 5, 3 x 3
    Number of filters in CONV layer 1: 3
    Number of filters in CONV layer 2: 10
    E5E5 Law filter application

Setting	Training Accuracy	Testing Accuracy
1	98.23	96.33
2	95.84	95.2
3	97.76	96.2
4	98.51	96.24
5	98.3	96.55
6	98.6	96.47
7	98.68	95.37
8	97.55	95.7
9	97.51	96.31
10	97.58	95.6



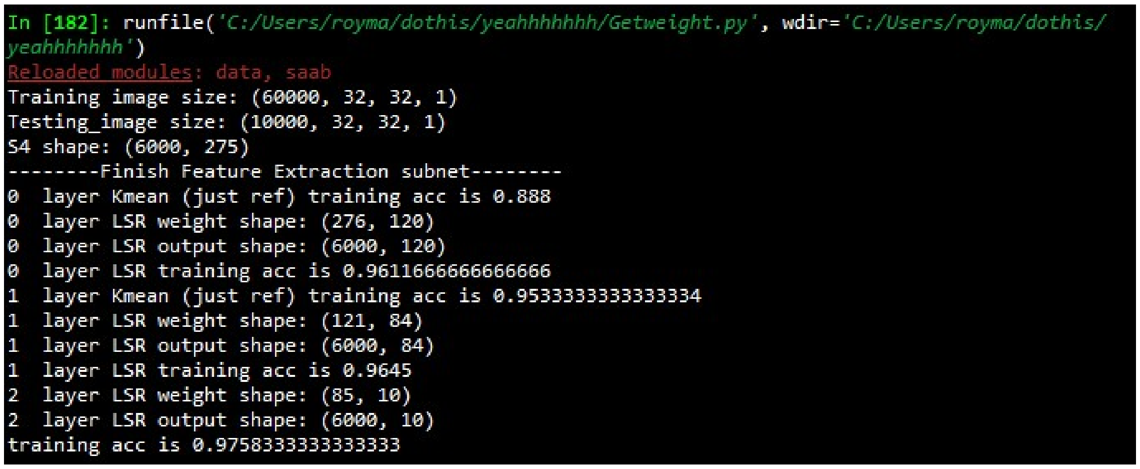
```
Reloaded modules: data, saab
Training image size: (60000, 32, 32, 1)
Testing_image size: (10000, 32, 32, 1)
GGG
(6000, 32, 32, 1)
halfway
GGG
(6000, 32, 32, 1)
Parameters:
use_classes: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Kernel_sizes: [5, 5]
Number_kernels: [3, 10]
Energy_percent: None
Number_use_images: 6000
--------stage 0 --------
Num of kernels: 3
Energy percent: 0.715998
Sample patches shape after flatten: (4704000, 25)
Kernel shape: (4, 25)
Transformed shape: (4704000, 4)
Sample images shape: (6000, 14, 14, 4)
--------stage 1 --------
Num of kernels: 10
Energy percent: 0.780727
Sample patches shape after flatten: (600000, 100)
Kernel shape: (11, 100)
Transformed shape: (600000, 11)
Sample images shape: (6000, 5, 5, 11)
```

Sample run of setting 10 in GetKernel



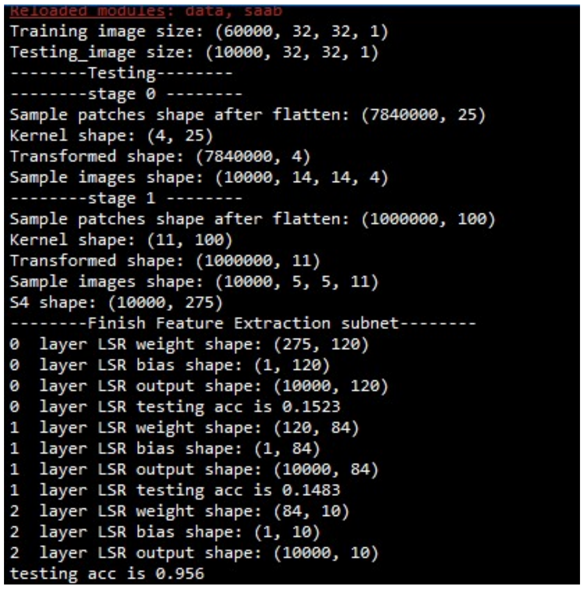
```
Reloaded modules: data, saab
Training image size: (60000, 32, 32, 1)
Testing_image size: (10000, 32, 32, 1)
--------Training--------
--------stage 0 --------
Sample patches shape after flatten: (4704000, 25)
Kernel shape: (4, 25)
Transformed shape: (4704000, 4)
Sample images shape: (6000, 14, 14, 4)
--------stage 1 --------
Sample patches shape after flatten: (600000, 100)
Kernel shape: (11, 100)
Transformed shape: (600000, 11)
Sample images shape: (6000, 5, 5, 11)
S4 shape: (6000, 275)
--------Finish Feature Extraction subnet--------
```

Sample run of setting 10 in GetFeature



```
In [182]: runfile('C:/Users/royma/dothis/yeahhhhhhh/Getweight.py', wdir='C:/Users/royma/dothis/
yeahhhhhhh')
Reloaded modules: data, saab
Training image size: (60000, 32, 32, 1)
Testing_image size: (10000, 32, 32, 1)
S4 shape: (6000, 275)
--------Finish Feature Extraction subnet--------
0  layer Kmean (just ref) training acc is 0.888
0  layer LSR weight shape: (276, 120)
0  layer LSR output shape: (6000, 120)
0  layer LSR training acc is 0.9611666666666666
1  layer Kmean (just ref) training acc is 0.9533333333333334
1  layer LSR weight shape: (121, 84)
1  layer LSR output shape: (6000, 84)
1  layer LSR training acc is 0.9645
2  layer LSR weight shape: (85, 10)
2  layer LSR output shape: (6000, 10)
training acc is 0.9758333333333333
```

Sample run of GetWeights

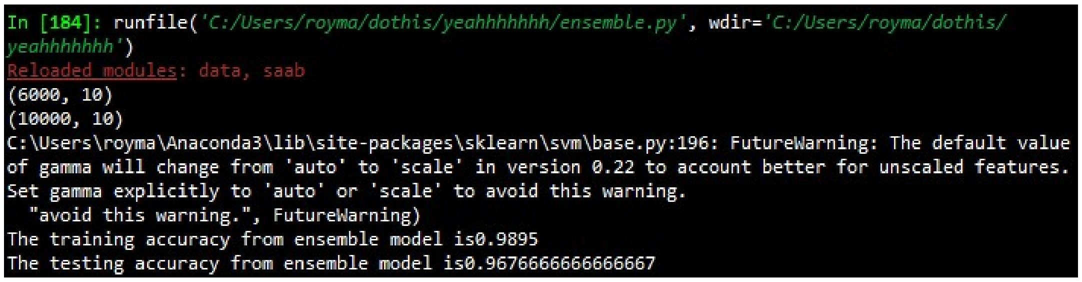


```
Reloaded modules: data, saab
Training image size: (60000, 32, 32, 1)
Testing_image size: (10000, 32, 32, 1)
--------Testing--------
--------stage 0 --------
Sample patches shape after flatten: (7840000, 25)
Kernel shape: (4, 25)
Transformed shape: (7840000, 4)
Sample images shape: (10000, 14, 14, 4)
--------stage 1 --------
Sample patches shape after flatten: (1000000, 100)
Kernel shape: (11, 100)
Transformed shape: (1000000, 11)
Sample images shape: (10000, 5, 5, 11)
S4 shape: (10000, 275)
--------Finish Feature Extraction subnet--------
0  layer LSR weight shape: (275, 120)
0  layer LSR bias shape: (1, 120)
0  layer LSR output shape: (10000, 120)
0  layer LSR testing acc is 0.1523
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 0.1483
2  layer LSR weight shape: (84, 10)
2  layer LSR bias shape: (1, 10)
2  layer LSR output shape: (10000, 10)
testing acc is 0.956
```

Sample run of MNIST test

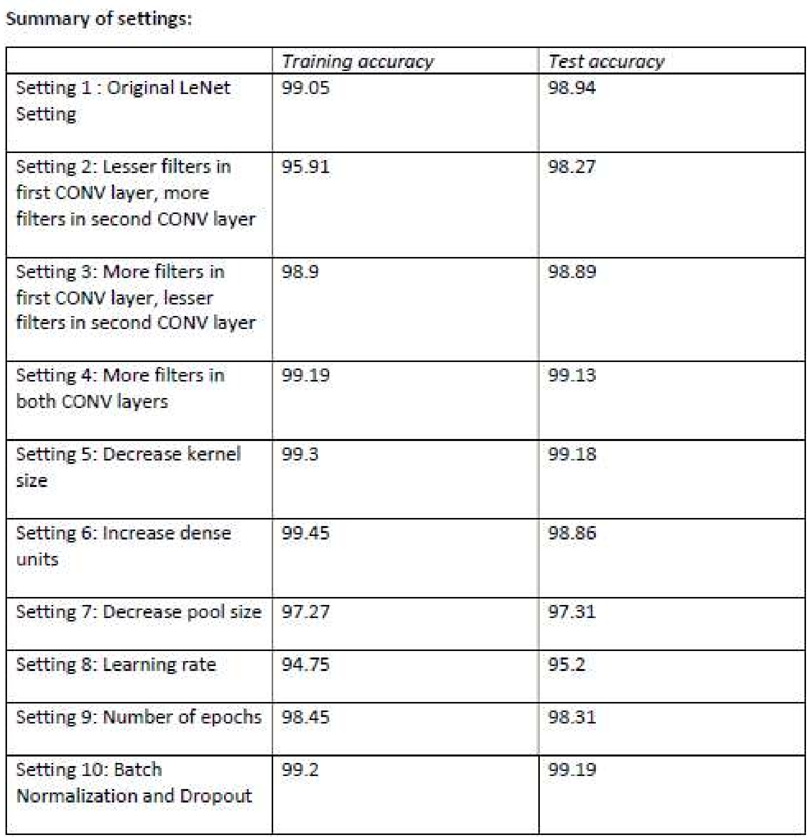
Section 2: Ensemble Results

Addition of different settings to ensemble model	1	2	3	4	5	6	7	8	9	10
Training accuracy	87.83	97.6	97.76	98.51	98.38	98.43	98.58	98.66	98.9	98.95
Testing accuracy	82.43	93.8	94.31	96.24	95.21	95.48	95.6	95.99	96.58	96.76



```
In [184]: runfile('C:/Users/royma/dothis/yeahhhhhhh/ensemble.py', wdir='C:/Users/royma/dothis/
yeahhhhhhh')
Reloaded modules: data, saab
(6000, 10)
(10000, 10)
C:\Users\royma\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value
of gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features.
Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
The training accuracy from ensemble model is0.9895
The testing accuracy from ensemble model is0.9676666666666667
```

Section 3: BP CNN and FF CNN performance:



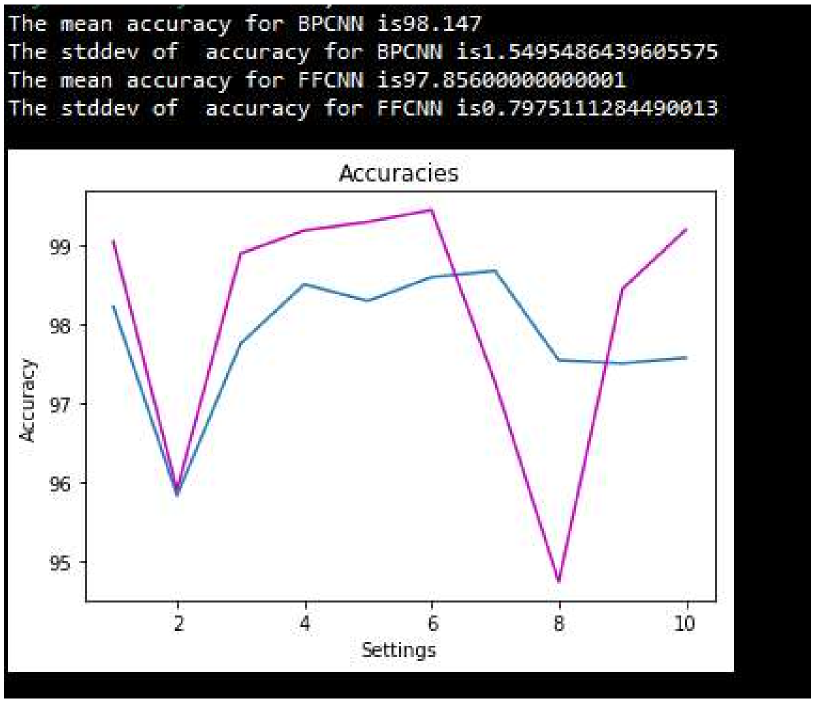
Summary of settings:

	Training accuracy	*Test accuracy*
Setting 1 : Original LeNet Setting	99.05	98.94
Setting 2: Lesser filters in first CONV layer, more filters in second CONV layer	95.91	98.27
Setting 3: More filters in first CONV layer, lesser filters in second CONV layer	98.9	98.89
Setting 4: More filters in both CONV layers	99.19	99.13
Setting 5: Decrease kernel size	99.3	99.18
Setting 6: Increase dense units	99.45	98.86
Setting 7: Decrease pool size	97.27	97.31
Setting 8: Learning rate	94.75	95.2
Setting 9: Number of epochs	98.45	98.31
Setting 10: Batch Normalization and Dropout	99.2	99.19

Accuracies from BP CNN HW 5

Setting	Training Accuracy	Testing Accuracy
1	98.23	96.33
2	95.84	95.2
3	97.76	96.2
4	98.51	96.24
5	98.3	96.55
6	98.6	96.47
7	98.68	95.37
8	97.55	95.7
9	97.51	96.31
10	97.58	95.6

Accuracies from FF CNN



```
The mean accuracy for BPCNN is98.147
The stddev of  accuracy for BPCNN is1.5495486439605575
The mean accuracy for FFCNN is97.85600000000001
The stddev of  accuracy for FFCNN is0.7975111284490013
```

The magenta line is BP CNN and the blue line is FF CNN

4.Discussion:

1. Note that the complete set of 60000 images could not be handed in for training. Fewer images were input due to heavy strain on computational resources (the system crashed eleven times). Since FF CNN does not work on the basis of optimization and as is mentioned in the official paper, it is mentioned by experiment and proof that the convergence is much faster due to abundance of training patches and statistical correlation. Therefore, a very certain, direct line cannot be drawn between BP CNN executed in the past project since it had the luxury of several epochs and the complete dataset.
2. The accuracy for the FF CNN follows a similar pattern as it corresponded to BP CNN.

3. With increase in number of kernels, the accuracy increases.
4. With decrease in initial kernel filters and increase in final kernel filters, the accuracy varies depending on the task. If the generalizability is more, such as different types of animals, the initial number of kernels will not affect the accuracy as opposed to the case when the task is specific such as different types of cat. That is when the number of second layer kernel filters is more.
5. The ensemble model is used to boost the diversity of the model. Since FF CNNs rely heavily on statistics, more discriminability helps in increasing accuracy.
6. As the number of settings added increases, as can be noted from the table in the results section, the features added from the settings enhances the accuracy of the overall model.
7. Computation wise, the FF CNN requires fewer epochs to converge but the computations are on the large scale (huge matrix operations) as opposed to a multitude of small scale neuron level calculations in BP CNNs. With increase in the number of filters in both the CONV layers, the decrease in kernel size and stride, the algorithm becomes computationally more complex and therefore requires much more time to run.
8. As can be noted from the table of accuracies, setting 1 had optimal parameters, resulting in good accuracy.
9. Setting 2 had less number of filters, which had lesser run time, but costed information.
10. Similarly, the other settings correspond to the explanation above.
11. Laws filter application was used to enhance the diversity of the model. However, in the individual case, it did not result in the best accuracy, as opposed to the ensemble case, where it boosted the model.
12. Performance wise, we can see that BP CNN has a lot of constraints on its implementation whereas in this homework, I implemented FF CNN FC layers using Least Square Regressors and also SVM.
13. The accuracies were more for BP CNN since it had access to the complete dataset and is end to end optimized. The FF CNN has only used a certain set of statistical methods but if more were to be leveraged, it would reduce the run time and increase the efficiency.
14. The FF CNN is not task based since it depends on the statistics. The same model can be used for another problem. However, a different BP CNN will have to be designed.
15. The performance of the BP CNN, in the fine scale, depends on random initializations and the momentum of optimization, therefore, it has more standard deviation from an FF CNN since the statistics of the data helps center the accuracy.
16. **Improvements** that can be made to BP CNN include adding additional layers in between the traditional CONV and the FC layers which help select or boost the features more depending on the statistics of the dataset such as selective dropout based on performance of a neuron or batch normalization based on performance of the previous layer. Additions to the layer must be made optional depending on the efficiency of the previous layer.
17. **Improvements** that can be made to FF CNN include using more statistical techniques such as rejection sampling to monitor information gain, MCMC for

statistical optimization in the forward pass since it is more data centric than non convex optimization.

References:

1. Interpretable convolutional neural networks via feedforward design. Journal of Visual Communication and Image Representation. C.C. Jay Kuo
2. Kuo, C. C. J. (2019). Ensembles of feedforward-designed convolutional neural networks.

Thank you!