EE569 Digital Image Processing

HOMEWORK # 6

Aagam Shah

USC ID-8791018480

USC Email – aagamman@usc.edu

Submission Date – April 26, 2020

Problem 1: Understanding Successive Subspace Learning (SSL)

(a) Feedforward-designed Convolutional Neural Networks (FF-CNNs):

1. ABSTRACT & MOTIVATION

As we already know that the Convolutional Neural Network is basically the network which is based on the mathematical operation known as Convolution which is nothing but a linear operation. So, basically Convolutional Neural Networks are nothing but the neural networks which will employ convolution operation in replacement of the basic matrix multiplication operation in minimum one of the layers.

But, the feedforward-designed Convolutional Neural Networks are basically the neural networks in which the information pass or the connections between the nodes are designed only in the forward direction, beginning from the input node and passing through the hidden nodes and the finally reaching it to the output nodes, which basically means that there will be no internal loops or the cycles in the network i.e., without back propagation. Convolutional Neural Network is basically the process which involves the feature extraction along-with the classification of the high dimensional data and then generates an output vector of lower dimension of class labels. The traditional CNNs use back-propagation method in order to train or optimize the model parameters to minimize the output classification cross entropy. Also, as we know that the nonlinear activation function is used when 2 Convolutional Neural Network layers are cascaded. So, in this novel technique a substitute of a nonlinear activation function is employed which is nothing but the Saak (subspace approximation via augmented kernels) transform and the Saab (subspace approximation via adjusted bias) transform.

So, basically to reduce the complex training and raise the understanding of the Convolutional Neural Network, the Feedforward design of Convolutional Neural Network was acquired. The FF-CNN is the data-centric approach. There are basically 2 cascade modules:

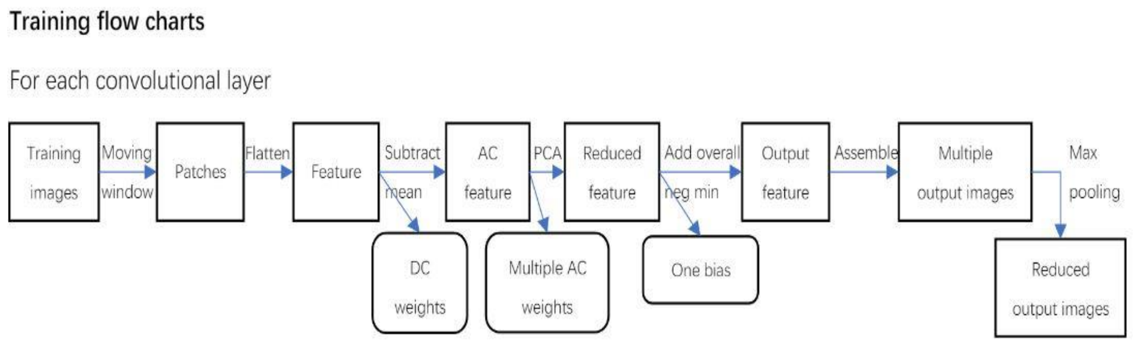
(1) Firstly, the series of convolutional layers through multi-stage Saab transforms.

(2) Secondly, the fully connected layers through the multi-stage linear least square regressors (LLSR).

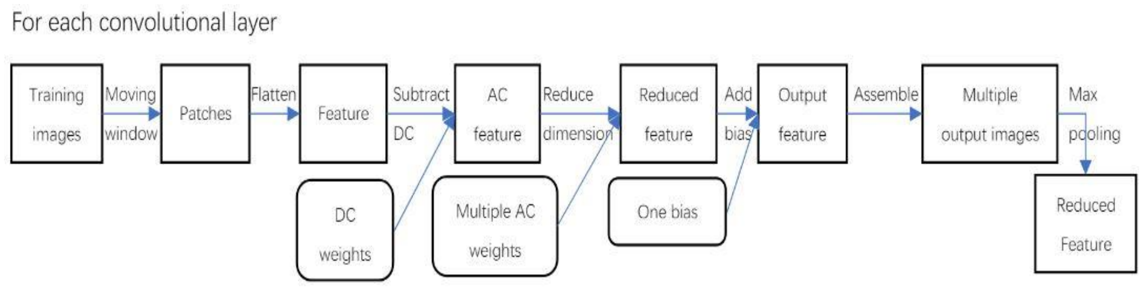
2. DISCUSSION

Answer 1:

Saab (subspace approximation with adjusted bias) is nothing but another version of Principal Component Analysis (PCA) along with the addition of the bias vector in order to eradicate the non-linearity of the activation function. So, in similar way in order to obtain multiple convolutional layers we cascade multiple Saab transforms. The Saab transform is basically used for the dimension reduction in the convolutional layer section of the Convolutional Neural Network. As we know that the trail of spatial-spectral filter operations are provided by the convolutional layers and gradually there is a loss of spatial resolution so in order to cover for that the spatial form is transformed to the spectral form by means of projection of pixels in window of already selected spatial patterns which are obtained by the PCA. Due to this the discriminant power is increased and there is larger viewing window for a larger receptive field. Now, in order to eradicate the nonlinearity of the activation function a new technique is developed known as Saab transform in which the bias vector is basically used to remove the nonlinearity of the activation function.
In order to avoid the back propagation technique for the weights of the filters in the convolutional layer, the anchor vector concept was derived which is developed by the statistical data of the input, which means that when the covariance matrix of the input vectors **x** is computed then the eigen vectors of that matrix can be chosen as the anchor vectors. Owing to the sign confusion problem a rectifier known as ReLU was employed in order to eliminate this issue. But, in this approach in order to overcome this sign problem an alternative PCA technique known as Saak transform was proposed, which basically augments the transform kernel with its negative transform kernel, but this results in doubling the number of kernels. So, to address this problem the Saab transform technique was proposed to eradicate this sign confusion issue at the same time avoid the spectral dimension doubling issue.



Testing flow charts



As we know that the several filters are used are used in the convolutional layer to carry out the convolution operation of the input images. All these input images are used to train the filter weights. Now further to obtain the transformation matrix for the conversion of represented data into the filter space, we perform PCA for all the training images on the small patches by sliding the kernel windows. Thus, the output obtained at this stage is the raw output data of the convolutional layer. Now as this raw output contains a lot of significant as well as unwanted information thereby increasing dimensionality of the original image but here in the original image in order to remove the unwanted information and reduce the dimensionality along with preserving the important information, we perform the dimensionality reduction step. The user can choose the number of dimensions wanted to be preserved by deciding the number of filters. This is basically employed by the novel technique known as Saak and Saab transform. Before, employing this transformation we need to compute the Direct current (DC) component for each input patch dimension and this DC component will later serve as the additional convolutional layer filter parameter. Apart from DC component the other component obtained is the AC component and then later on a bias component is added to each output dimension. The bias component is a scalar value which is adjusted to the same value as that of the minimum value of every outputs of the train dataset. The purpose of this bias is that all the output values are obtained as positive values especially for the testing data and thereby eradicate the sign confusion issue which is the case in the back-propagation method. The role of the max pooling layer is to basically just select the most prominent local dimension and then limit the dimensionality of the output data which might also be responsible for the non-linearity in the model.
So here in case of feed-forward CNN, the Saab transform is employed for deciding the filter coefficients. But as by just observing at the single pixel value we cannot determine the pattern information, so we need to look for a larger receptive field employed for convolution for feature identification. As the FF-CNN does perform over various epochs but has just one single pass or single epoch process, so hence the features must be selected with quite wisely which has the strongest discriminant power which can differentiate among various other classes. Thus, Saab transform technique is used for filter coefficients

(a_k) and bias term (b_k) selection by the anchor vectors. The affine transform equation is given by:

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

And the subspace which is spanned by anchor vectors is given as:

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

$$\mathcal{S} = \mathcal{S}_{DC} \oplus \mathcal{S}_{AC},$$

Where S (DC) is subspace spanned by DC anchor vector and similarly S (AC) is the subspace spanned by the AC anchor vectors.

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

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

Now following this a covariance matrix of an input vector is calculated and then the eigen vectors of the largest eigen value are nothing but the vectors which will span the AC subspace. Later on, the PCA on the AC subspace is computed which will in turn determine the anchor vectors (k-1) of the strongest discriminant power. PCA can be basically thought of as the pooling operation in the spatial domain.

As the covariance matrix obtained for an input image of size NxN will be N^4, thereby increasing the computational complexity. So, owing to this reason and taking into consideration the computational complexity of covariance matrix the input of small window size is considered.

Now in order to avoid sign confusion issue we use Saab transform and along with 2 constraints on the bias terms i.e.,

- <u>Positive Response Constraint:</u>

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

- <u>Constant Bias Constraint:</u>

  The b_k is constant for all spectral filters. Anyways it is a DC component so can be easily removed from the succeeding layers to make it less computationally complex. Also, as this parameter is embedded into the DC subspace, it doesn't affect the choice of anchor vectors in the AC subspace. So, PCA is carried out only for the AC subspace, layer after layer in order to obtain the AC anchor vectors.

$$b_0 = b_1 = \cdots = b_{K-1} \equiv d\sqrt{K}.$$

As the response is positive as well as the bias term is positive which means the output remains the same whether we employ the non-linear activation function (ReLU) or not. This is the most important contribution as it greatly eases the design of the CNN network as the non-linearity incorporated due to the activation function is removed. The bias vector term added as constant to the output obtained after transformation is just the offset of every output by a particular value. Thus, the bias selection rule can be given as follows:

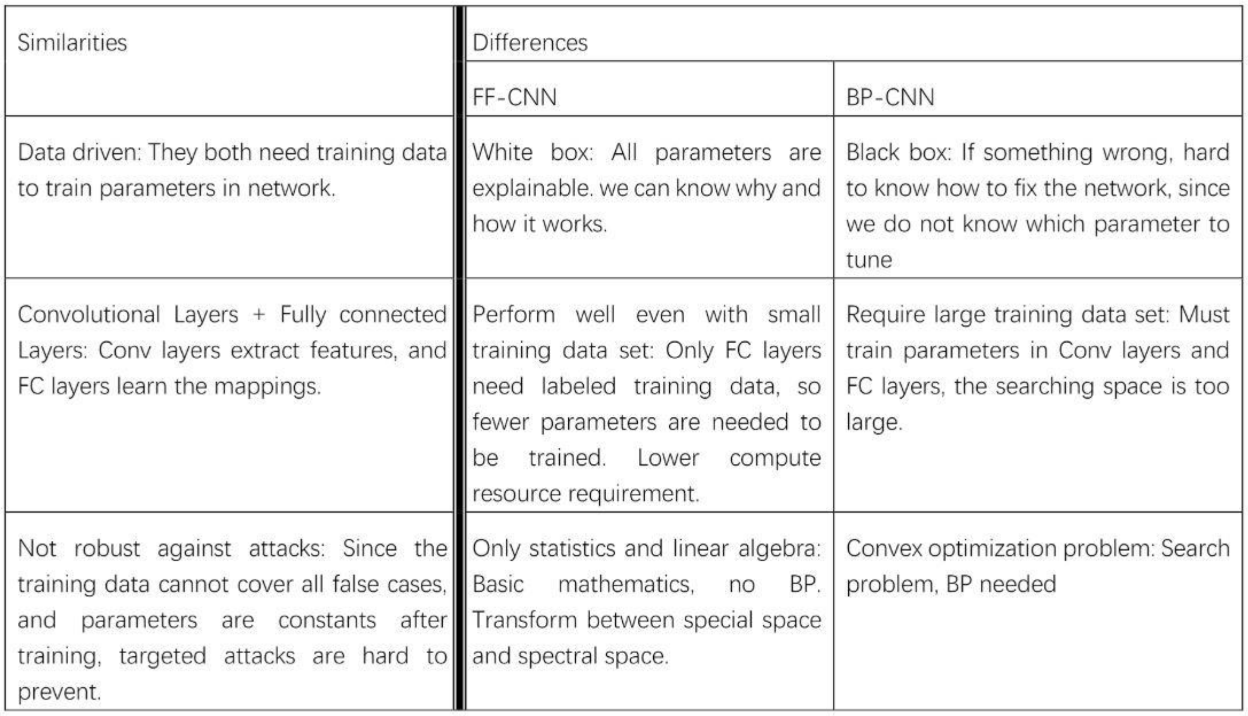
$$b_k \geqslant \max_{\mathbf{x}} ||\mathbf{x}||, \quad k = 0, \cdots, K-1.$$

where $\mathbf{x} \in R^N$ is an input vector and K is the dimension of the output space R^K.

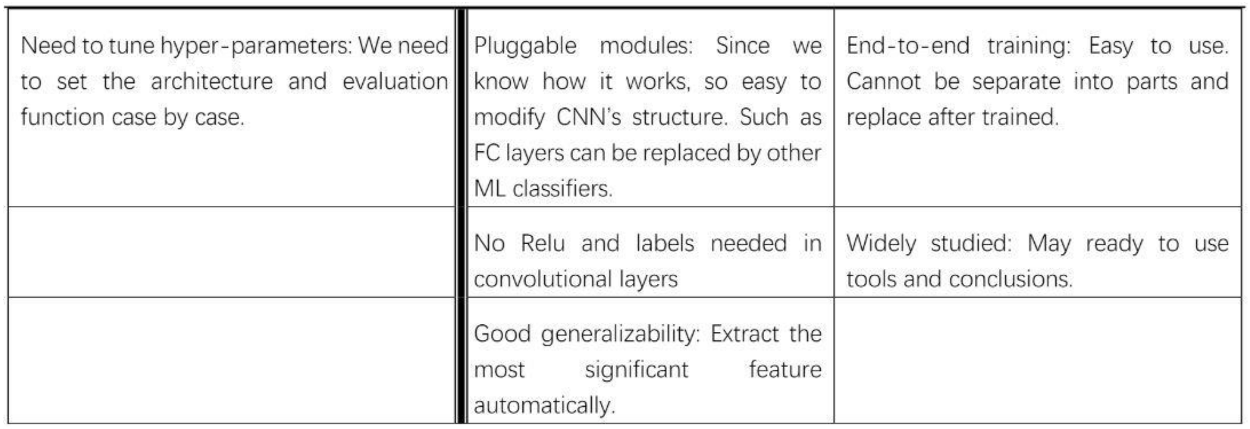
Hence, the Saab transform is superior than the Saak transform, the reason because the sign confusion issue is eradicated as well as there is no need for the addition of more spectral filters, thereby reducing the computational complexity.

<u>Answer 2:</u>

The similarities and differences between FF-CNN and the BP-CNN is as given in the tabular format:



Similarities	Differences	
	FF-CNN	BP-CNN
Data driven: They both need training data to train parameters in network.	White box: All parameters are explainable. we can know why and how it works.	Black box: If something wrong, hard to know how to fix the network, since we do not know which parameter to tune
Convolutional Layers + Fully connected Layers: Conv layers extract features, and FC layers learn the mappings.	Perform well even with small training data set: Only FC layers need labeled training data, so fewer parameters are needed to be trained. Lower compute resource requirement.	Require large training data set: Must train parameters in Conv layers and FC layers, the searching space is too large.
Not robust against attacks: Since the training data cannot cover all false cases, and parameters are constants after training, targeted attacks are hard to prevent.	Only statistics and linear algebra: Basic mathematics, no BP. Transform between special space and spectral space.	Convex optimization problem: Search problem, BP needed



Need to tune hyper-parameters: We need to set the architecture and evaluation function case by case.	Pluggable modules: Since we know how it works, so easy to modify CNN's structure. Such as FC layers can be replaced by other ML classifiers.	End-to-end training: Easy to use. Cannot be separate into parts and replace after trained.
	No Relu and labels needed in convolutional layers	Widely studied: May ready to use tools and conclusions.
	Good generalizability: Extract the most significant feature automatically.	

(b) Successive Subspace Learning (SSL):

1. ABSTRACT & MOTIVATION

The subspace techniques are extensively used in the field of signal and image processing, computer vision and mathematical pattern recognition. It offers powerful tool for signal analysis, modeling as well as processing. Subspace learning is basically used to represent data and make accurate decision for the training samples. SSL is based on the 4 key parameters as described below:

a. Expansion of near to far neighborhood successively
b. Unsupervised reduction of dimension through the subspace approximation technique
c. Supervised reduction of dimension through the Label-assisted regression (LAG)
d. Decision forming and concatenation of features.

2. DISCUSSION

Answer 1:

The Successive Subspace Learning (SSL) technique basically consists of the 4 key parameters stated as:
(1) Expansion of near to far neighborhood successively
(2) Unsupervised reduction of dimension through the subspace approximation technique
(3) Supervised reduction of dimension through the Label-assisted regression (LAG)
(4) Decision forming and concatenation of features.

Taking (1) into consideration, the computation of the local to global neighborhoods attributes of the chosen pixels is performed successively through the multiple stages. The advantage of this particular approach is that the features of the near as well as the far neighboring pixels are carried to the target pixel via local communication. The attributes

can either be gray scale image or RGB image or spatial coordinates, etc. The attributes can be passed on in single hop if they are near neighbors while it may take multiple hops for transferring attributes or features if the neighbors are far away. The neighborhood is larger as the value of number of hops increases which basically leads to the increase in the feature dimension of the neighborhood rapidly as it is directly proportional to the number of pixels present in the neighborhood. So, in order to control the growth rate of dimensions without compromising the accuracy, we opt for the 2nd key parameter of the approximate subspace. This parameter basically deals with the statistical properties and correlations among the feature vectors of the neighborhoods. Here we go for the subspace approximation technique. PCA is single stage subspace approximation technique but when we refer to the successive subspace learning we employ unsupervised Saak/Saab transform for dimensionality reduction and eliminating the sign confusion issue. In the initial steps the feature vector dimension is small then the neighborhood dimension is also small but the amount of independent neighbors are big and vice-versa, providing the classification accuracy. Now in the upcoming stages to get the dimensionality reduction we employ the label information via Label assisted regression unit (LAG) unit which is nothing but the 3rd key parameter of the SSL technique. Last but not the least, the 4th parameter plays a role in concatenation of all the dimension reduced features from all the stages in order to generate the final feature vector and then it is fed to the classifier for training purpose of the multi class classifier for performing the classification job.
Comparison of the DL and SSL can be done as follows:
The similarities of DL and SSL can be given as in the tabular form as:

Sr. No.	**Key points**	**SSL**	**DL**
1.	Collection of attributes	The neighborhoods are grown successively.	The receptive fields are grown slowly.
2.	Processing of the attributes	Spatial dimensions are traded in turn of spectral dimensions.	Spatial dimensions are traded in turn of spectral dimensions.
3.	Pruning of spatial dimension	The pruning of spatial dimension is done by the spatial pooling.	The pruning of spatial dimension is done by the spatial pooling.

The differences between the SSL and DL are given as follows:

Sr.No.	**Key Points**	**SSL**	**DL**
1.	Expandability of model	It is a non-parametric model with the flexibility of add/removal of filters which makes it capable to handle large as well as small datasets.	It is a parametric learning process with fixed network architecture which will lead to over-parameterized model.

2.	Progressive Learning	It is easy to adapt to progressive learning by SSL as employs a non-parametric model.	It is difficult to adapt to the incremental learning for DL as it employs parametric model.
3.	Architecture of model	While the SSL has more flexible architecture which allows the extraction valuable features at multiple stages.	It is based on the network architecture which demands for the last node where the loss function can be defined, especially to train the network parameters via back propagation.
4.	Interpretation of model	SSL model is a white box which means that it is mathematically transparent and easily understood.	DL model is a black box which means that most of the properties are not properly understood.
5.	Search of model parameter	SSL technique uses the feed-forward design approach which consists of unsupervised as well as the supervised dimensionality reduction approaches for feature extraction.	DL model uses back propagation technique to decide the parameters of the model employing end to end optimization approach.
6.	Complexity of training and testing dataset	The complexity of the training data in SSL is less as it is just a single feed forward design approach and the testing complexity depends on the number of stages.	The complexity of the DL model is higher as it has a huge amount of training parameters due to the back-propagation approach which makes the model deep and highly complex.
7.	Reduction of spectral dimension	Here in SSL the convolutional operations are employed in order to find the projections on the principal components of the subspace.	While in DL model the convolutional operations are used for transformation of one form to other for the end to end optimization with the help of BP.
8.	Features which are task independent	SSL consists of both types of features i.e., task independent and task dependent which	DL model makes use of both the parameters which are input images as well as output labels

		are obtained through unsupervised and supervised learning respectively.	and hence they are task dependent.
9.	Ability of multi-tasking	It is easy to perform multi-tasking by SSL model.	It is difficult for the DL model to perform multi-tasking due to integration of the multiple loss functions to obtain a new joint function.
10.	Addition of priors & constants	It is easy in SSL model to incorporate the priors and constants in order to clip the features of large as well as small neighbors in previous to feeding it to the classifier.	It is difficult to add the priors and constants in the DL model to the loss function which basically relates to the priors and constants.
11.	Weak supervision	It is easy in SSL model due to the reason that labels are not required for the unsupervised dimension reduction. It is only required for the LAG unit as well as for the training of the classifier.	It is difficult in the DL model as huge number of labeled data is required for the training process which is obtained with the help of the data augmentation.
12.	Adversarial attacks	It is difficult for the SSL model to have adversarial attacks as it has less disturbances which are removed by PCA which makes it difficult for the attackers to carry out similar kind of attacks.	It is easy to have adversarial attacks for the DL model because it is easy to determine the path from the output decision to the input space, which can easily be corrupted by adding small disturbances to the input data which is difficult to get identified by the human eye.

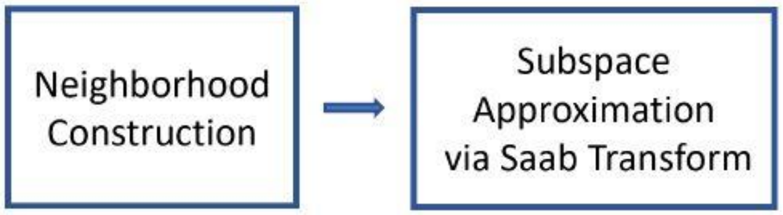
Answer 2:

The function of each module can be described in detail as follows:

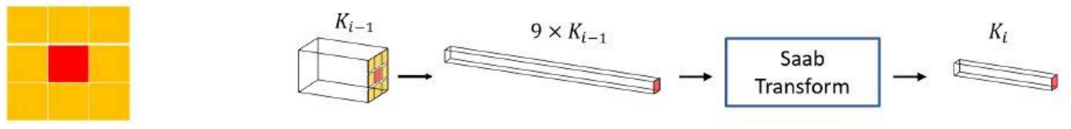
EE569 Digital Image Processing

Module 1: The cascading of the Pixel Hop units.

The main role of this module is basically to determine the features of the near to far neighborhoods of the chosen elements of pixels via Pixel Hop blocks.


Neighborhood Construction
Subspace Approximation via Saab Transform

The basic process of this unit is that the *ith* unit of PointHop basically connects the features of the previous i.e., (i-1) neighborhood and the dimension is given by K(i-1), of the pixel in target and the Ni pixels in the neighboring region in order to create a union of the neighboring pixels. Via this procedure, the expanded neighborhood dimension is equal to K(i-1) x (Ni + 1). As we know that it is somewhat cumbersome to control the growth of the dimension, but the dimensionality reduction is achieved by the subspace approximation technique known as Saab transform.

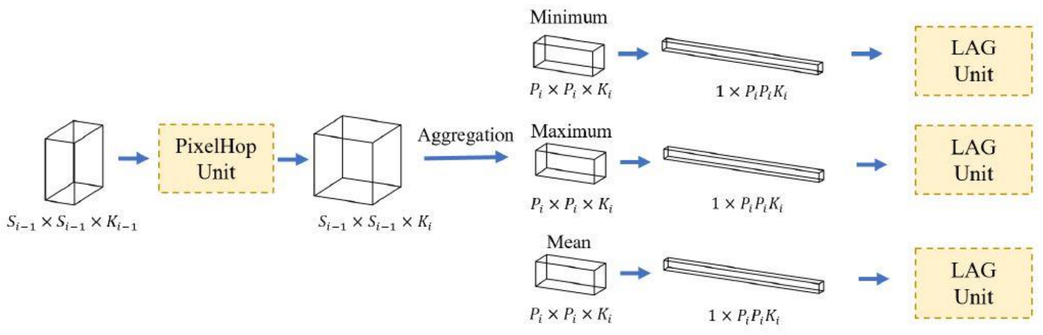


(a) (b)

The output of each pixel hop unit gives us the representation of the neighborhood which indicates the index of the stage and the size of the input neighborhood. As we can observe that at a particular pixel hop unit the there is a reduction of the spectral dimension, but the spatial dimension remains as it is, due to the reason that the 2 adjacent neighbors are overlapping with each other. Due to which there is an incorporation of the (2x2) to (1x1) max pooling layer between 2 Pixel Hop units to reduce the spatial redundancy.

Module 2: Accumulation and Supervised Dimensionality reduction using LAG unit

The dimension of the output obtained from the previous ith layer of the Pixel Hop unit is basically S(i-1) x S(i-1) x Ki. In order to reduce the spatial dimension prior to give the output to the succeeding Pixel Hop layer max pooling operation is performed. In order to get the variety of features at a particular layer, various accumulation techniques are like max, min or mean value response of the small patch of the non-overlapping regions are adopted. The spatial dimension of the attributes after accumulation is given by Pi x Pi, where Pi is nothing but the hyper-parameter. Followed by this, the supervised learning of feature reduction is carried out. Considering the neighborhood, all the various classes will have various different kinds of distributions and owing to this property it helps to clearly distinguish and give a briefer representation.


Minimum
Maximum
Mean
PixelHop Unit
Aggregation
LAG Unit
LAG Unit
LAG Unit
$S_{i-1} \times S_{i-1} \times K_{i-1}$
$S_{i-1} \times S_{i-1} \times K_i$
$P_i \times P_i \times K_i$
$1 \times P_i P_i K_i$

<u>Module 3:</u> Concatenation of the features among Pixel Hop Unit & Classification.

The N features obtained from the K Pixel Hop units are concatenated in order to get the NxK attributes in this module. Then later we basically train the multi-class classifier taking into consideration of the features obtained and then perform the standard pattern recognition process.

<u>Answer 3:</u>

<u>Neighborhood Construction:</u>

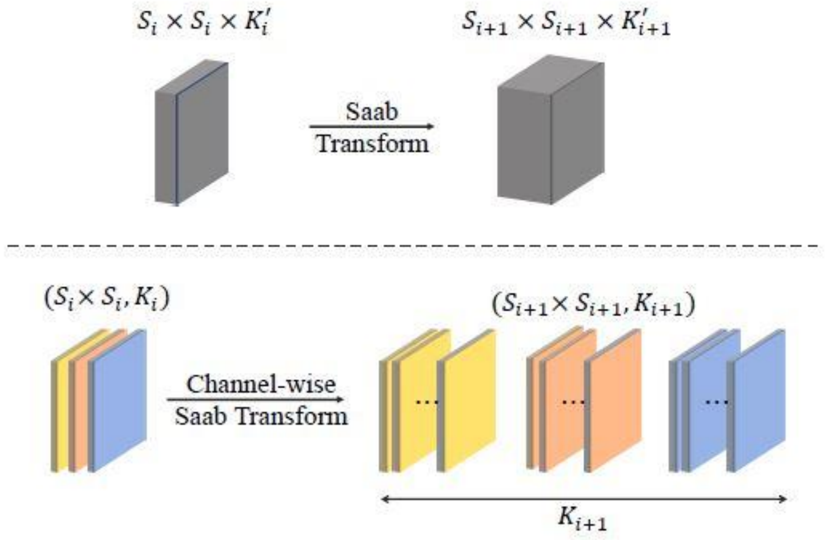
In Pixel Hop unit the neighborhood construction is basically done by performing the union of the centermost pixel and the 8 nearest neighborhood pixels. Basically, in this process the *ith* unit of PointHop basically connects the features of the previous i.e., (i-1) neighborhood and the dimension is given by K(i-1), of the pixel in target and the Ni pixels in the neighboring region in order to create a union of the neighboring pixels. Via this procedure, the expanded neighborhood dimension is equal to K(i-1) x (Ni + 1).

In Pixel Hop ++ Unit the neighborhood construction is basically done by the successive expansion of near-to-far neighbors which is very similar to the Pixel Hop unit. The computation of the local to global neighborhoods attributes of the chosen pixels is performed successively through the multiple stages. The advantage of this particular approach is that the features of the near as well as the far neighboring pixels are carried to the target pixel via local communication. The attributes can either be gray scale image or RGB image or spatial coordinates, etc. The attributes can be passed on in single hop if they are near neighbors while it may take multiple hops for transferring attributes or features if the neighbors are far away. The neighborhood is larger as the value of number of hops increases which basically leads to the increase in the feature dimension of the neighborhood rapidly as it is directly proportional to the number of pixels present in the neighborhood.

Subspace Approximation

Subspace approximation in the Pixel Hop unit is basically due to the fact that as we know that it is somewhat cumbersome to control the growth of the dimension, but the dimensionality reduction is achieved by the subspace approximation technique known as Saab transform. Saab (subspace approximation with adjusted bias) is nothing but another version of Principal Component Analysis (PCA) along with the addition of the bias vector in order to eradicate the non-linearity of the activation function. So, in similar way in order to obtain multiple convolutional layers we cascade multiple Saab transforms. The Saab transform is basically used for the dimension reduction in the convolutional layer section of the Convolutional Neural Network. As we know that the trail of spatial-spectral filter operations are provided by the convolutional layers and gradually there is a loss of spatial resolution so in order to cover for that the spatial form is transformed to the spectral form by means of projection of pixels in window of already selected spatial patterns which are obtained by the PCA. Due to this the discriminant power is increased and there is larger viewing window for a larger receptive field. Now, in order to eradicate the nonlinearity of the activation function a new technique is developed known as Saab transform in which the bias vector is basically used to remove the nonlinearity of the activation function.

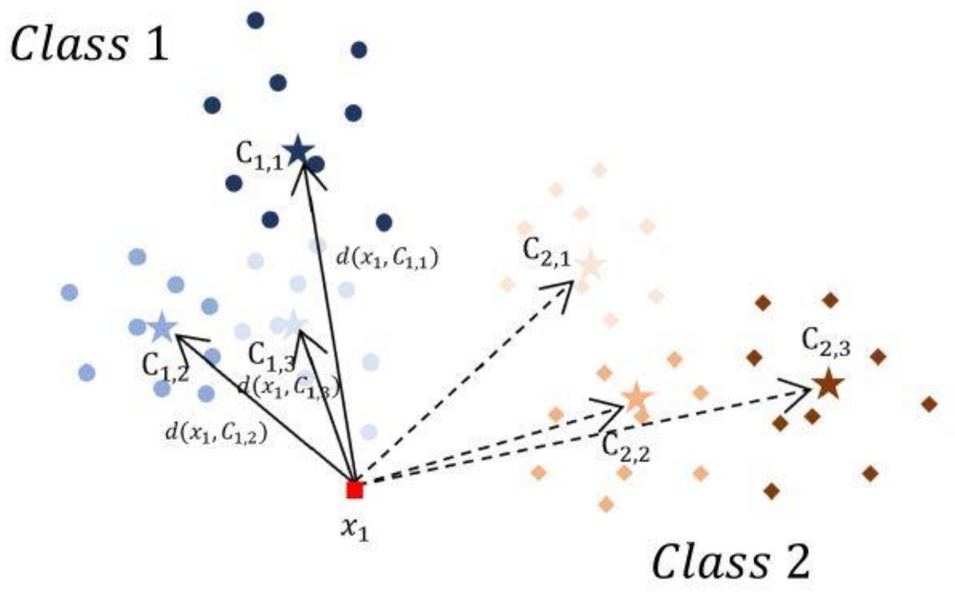
Subspace approximation in the Pixel Hop++ unit somewhat different than the one traditional Pixel Hop unit owing to the replacement of the Saab transform with the channel-wise Saab transform.
Basically, the differences between the Pixel Hop and the Pixel Hop ++ Units are explained in detail below:
As we know that the module 1 is basically an unsupervised learning module and we know that Saab transform is nothing but the tensor product of spatial domain transform and the spectral domain transform. Basically, Saab transform is nothing but a similar type of PCA (Principal Component Analysis) transform. Although PCA can un-correlate the covariance matrix to the diagonal matrix but the same is not possible to do with the Saab transform the reason being that the Saab coefficients are loosely correlated in spectral domain. Also, the spatial correlations are stronger than the spectral correlations which become more stronger as we go deep into the Pixel Hop ++ units. Due to the reason of employing instead the traditional Saab transform for the high dimensional dataset, we apply Ki channel wise (c/w) Saab Transforms to each and every spatial tensors of the Pixel Hop++. Basically, if we compare the traditional Saab transform used in the Pixel Hop and the channel wise (c/w) Saab Transform employed in the Pixel Hop++ we can observe that in the traditional Saab transform the dimension of an input image is Si x Si x Ki’ which in turn will produce an output of dimension S(i+1) x S(i+1) x K(i+1)’ and after that pooling layer will pool via grid of size Si x Si to the grid size of S(i+1) x S(i+1). Whereas, in the channel wise (c/w) Saab transform the Ki number of channel images are chosen which have the input dimension of Si x Si which in turn generates the K(i+1) output images which will have the dimension S(i+1) x S(i+1) after max pooling layer operation.


$S_i \times S_i \times K_i'$
$S_{i+1} \times S_{i+1} \times K_{i+1}'$
Saab Transform
$(S_i \times S_i, K_i)$
$(S_{i+1} \times S_{i+1}, K_{i+1})$
Channel-wise Saab Transform
K_{i+1}

Answer 4:

As we know that both the Pixel Hop and Pixel Hop ++ use the Label Assisted Gradients for the Supervised learning process for the feature reduction. So basically, the motivation for the LAG unit is based on the 2 observations. Firstly, the representation of the neighbor of a particular dimension with the target pixel at center is presented, whose size may vary from small to the large through previously discussed successive neighborhood expansion as well as the subspace approximation which are known as attributes. Thus in order to solve the image classification issue we basically integrate the attributes (local to global) via the multiple Pixel Hop units, which is nothing but the concatenation of all the attributes in order to generate a long feature vector, but still the dimension of the concatenated attributes is too high. So, in order to reduce the dimension of the concatenated attributes we go for the second option which is nothing but usage of labels for the Successive Subspace Learning (SSL). In this technique the attributes of the similar object class are found in the small subspace of the high dimensional feature space. Thus, in this technique it is proposed that the search for the subspace generated through the samples of similar class is carried out for the dimension reduction. This requires the supervised dimension reduction technique.

Similar to CNN technique it is proposed that the sample of each class are partitioned to generate pseudo classes resembling to the 1st FC layer which in turn will assign one hot vector in the dimension space. Then later on the Linear Least Square Regressor (LLSR) system is used in order to map the samples in the input feature space to the output space created by the hot vectors. Hot vector is basically nothing, but hard label used to indicate the cluster. But we use the soft labels for the LLSR setup. The training samples are used to learn the values of the regression matrix. Next the already learned regression matrix is applied to the testing data samples in order to perform dimension reduction.



The steps followed are basically as given below:

(i) In the step 1 we use the k-means clustering, applying it to the identical class to cluster the samples in order to create the subspaces which are object oriented and then later on find centroid of each and every subspace.

(ii) Now for the Step 2, as discussed earlier that we use the soft labels to associate the samples and the centroids instead of the hard labels, which is nothing but done by defining the probability vector along with the Euclidean distances between two vectors and the alpha parameter to assert the relation between likelihood and the Euclidean distance of a particular cluster. The greater the alpha value the decaying of probability occurs at a faster pace along with the distance. Similarly, the likelihood is larger for the shorter Euclidean distances. So, as a result the target vector of the output is changed from the one hot vector to the probability vector.

$$\text{Prob}(\mathbf{x}_j, \mathbf{c}_{j',l}) = 0 \quad \text{if} j \neq j',$$

$$\text{Prob}(\mathbf{x}_j, \mathbf{c}_{j,l}) = \frac{\exp(-\alpha d(\mathbf{x}_j, \mathbf{c}_{j,l}))}{\sum_{l=1}^{L} \exp(-\alpha d(\mathbf{x}_j, \mathbf{c}_{j,l}))},$$

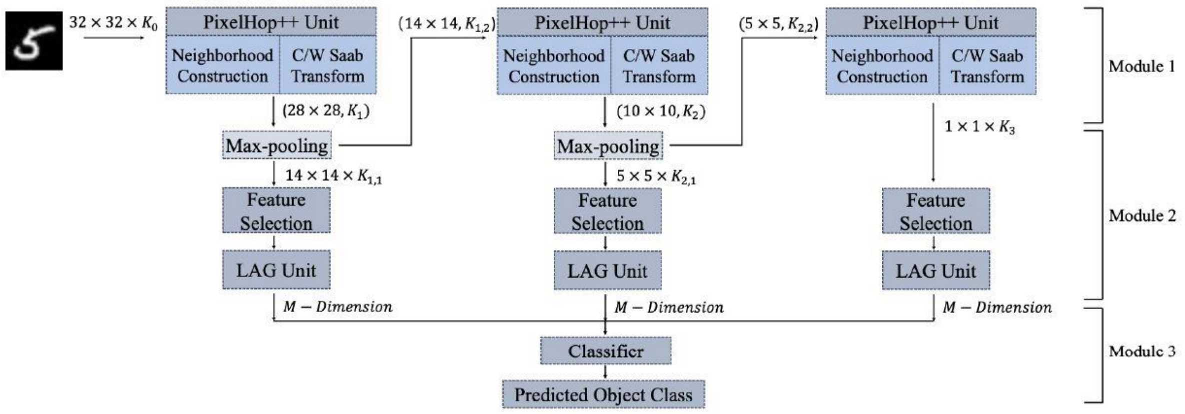
(iii) Lastly, the equations of the Linear Least Square regressor are formulated and solved in order to achieve the probability vectors. *LSR equations basically create a relation between the output probability vector and the input attribute vector.* [4]

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & w_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & w_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{Mn} & w_M \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1(\mathbf{x}) \\ \vdots \\ \mathbf{p}_j(\mathbf{x}) \\ \vdots \\ \mathbf{p}_J(\mathbf{x}) \end{bmatrix},$$

Problem 2: CIFAR-10 Classification using SSL

(a) Building a Pixel Hop++ Model:

(1)



The training time = 83.7 minutes
The train accuracy = 99.8%
The model size in terms of total number of parameters are as follows:

Module	**Hop 1**	**Hop 2**	**Hop 3**	**Total**
Module 1	3075	16575	39300	58950
Module 2	200950	134400	12550	347900

Total Model Size = 406850

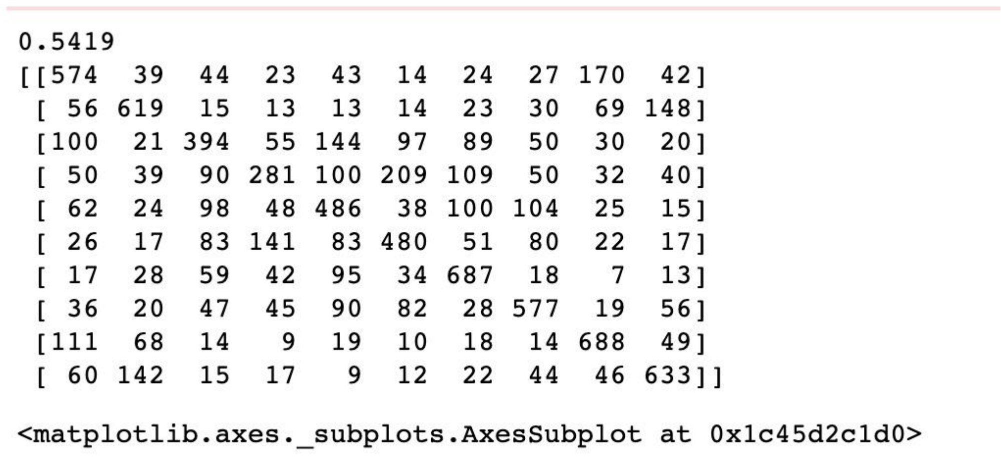
(2) Applying the trained model to the 10,000 testing images,

The Testing Accuracy = 59.18%

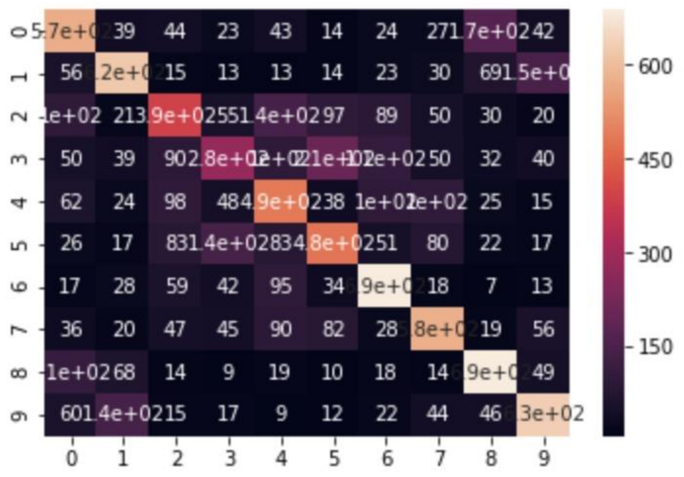
(3) Unchanging the trained Module 1 and then decreasing the number of the labeled training samples of Module 2 and 3 to the various cases and then applying the trained model to the 10,000 testing images. Finally, we obtain the curve of the test accuracy w.r.t to the number of training images for each case as follows.

<u>Case 1:</u> (1/4) of 50,000 training images.

Testing accuracy = 54.19%



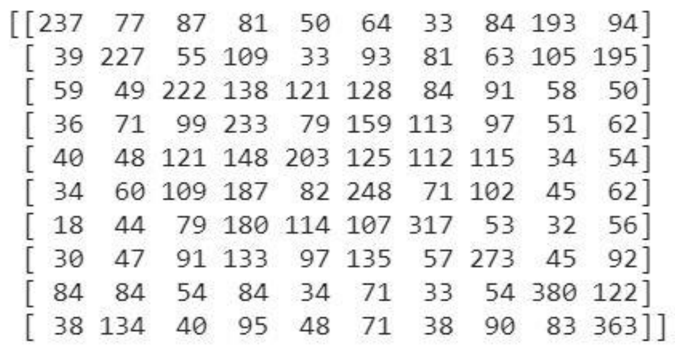
```
0.5419
[[574  39  44  23  43  14  24  27 170  42]
 [ 56 619  15  13  13  14  23  30  69 148]
 [100  21 394  55 144  97  89  50  30  20]
 [ 50  39  90 281 100 209 109  50  32  40]
 [ 62  24  98  48 486  38 100 104  25  15]
 [ 26  17  83 141  83 480  51  80  22  17]
 [ 17  28  59  42  95  34 687  18   7  13]
 [ 36  20  47  45  90  82  28 577  19  56]
 [111  68  14   9  19  10  18  14 688  49]
 [ 60 142  15  17   9  12  22  44  46 633]]

t[190]: <matplotlib.axes._subplots.AxesSubplot at 0x1c45d2c1d0>
```



<u>Case 2:</u> (1/8) of 50,000 training images.

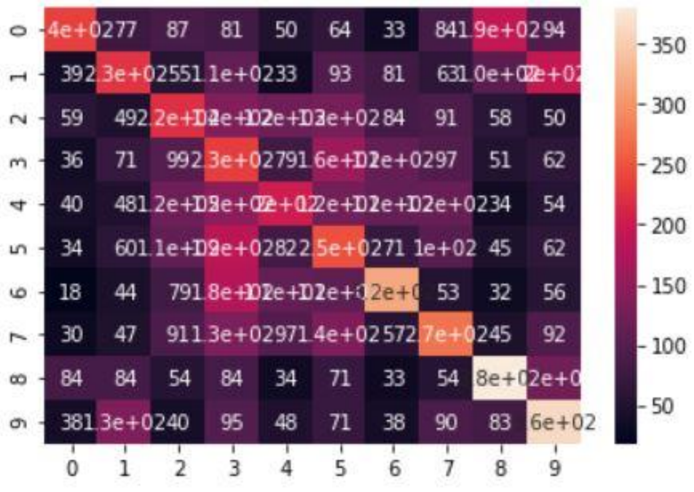
Testing accuracy = 27.03%

Confusion Matrix:



```
[[237  77  87  81  50  64  33  84 193  94]
 [ 39 227  55 109  33  93  81  63 105 195]
 [ 59  49 222 138 121 128  84  91  58  50]
 [ 36  71  99 233  79 159 113  97  51  62]
 [ 40  48 121 148 203 125 112 115  34  54]
 [ 34  60 109 187  82 248  71 102  45  62]
 [ 18  44  79 180 114 107 317  53  32  56]
 [ 30  47  91 133  97 135  57 273  45  92]
 [ 84  84  54  84  34  71  33  54 380 122]
 [ 38 134  40  95  48  71  38  90  83 363]]
```

Heat Map:

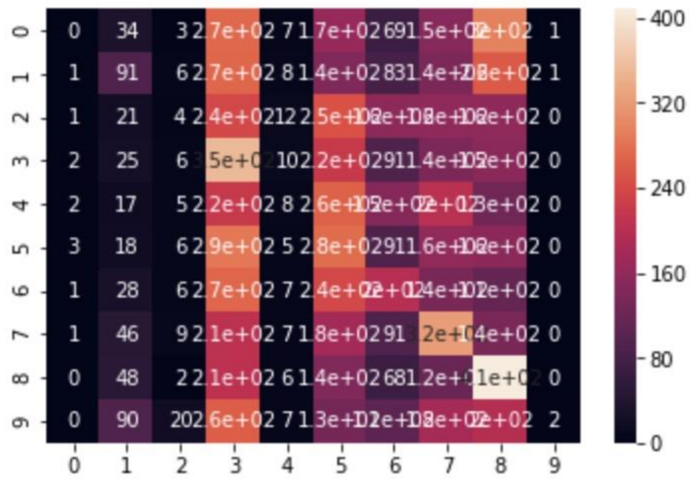


<u>Case 3:</u> (1/16) of 50,000 training images.

Testing accuracy = 16.64%

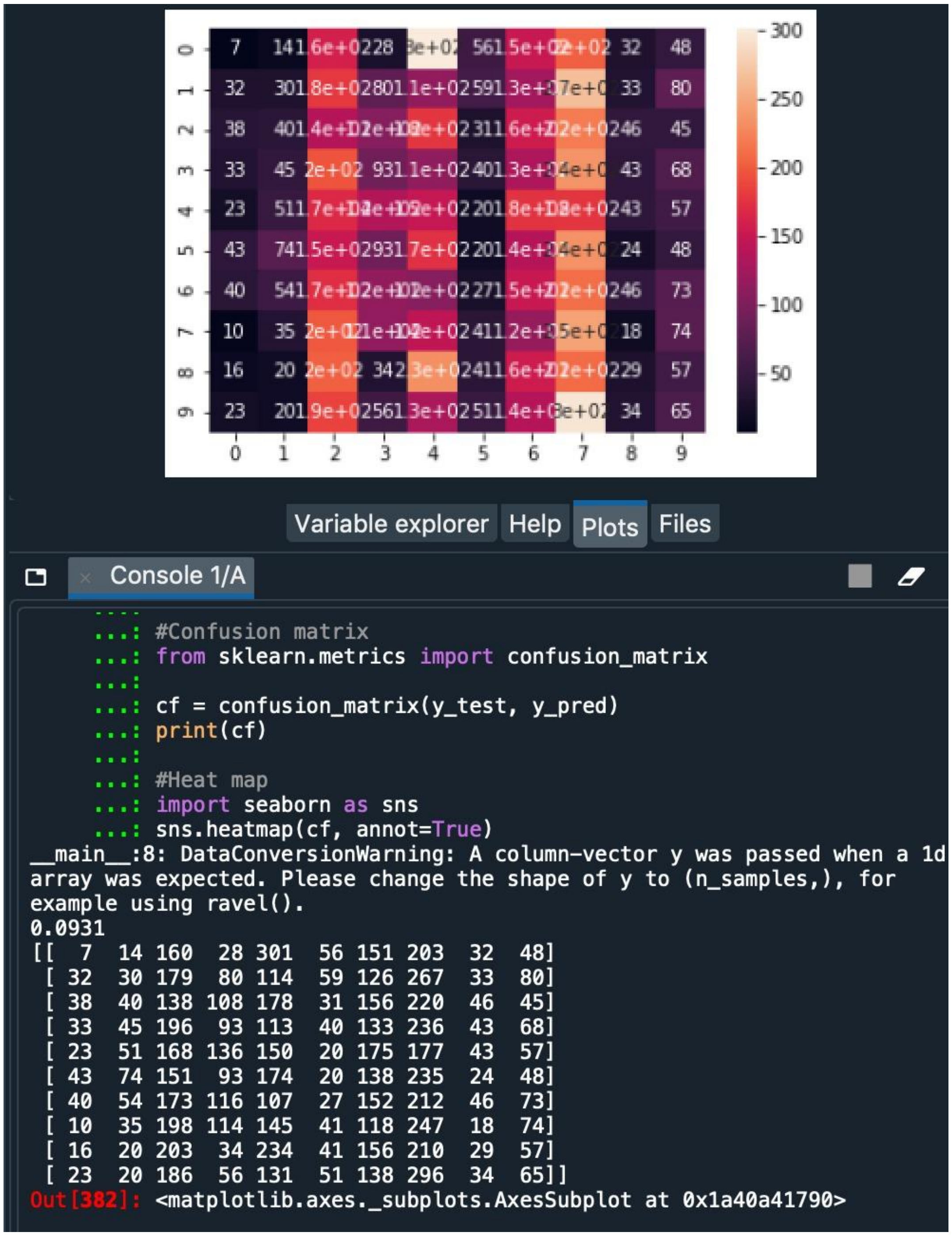
```
0.1664
[[  0  34   3 272   7 167  69 151 296   1]
 [  1  91   6 268   8 145  83 139 258   1]
 [  1  21   4 239  12 250 157 158 158   0]
 [  2  25   6 353  10 220  91 139 154   0]
 [  2  17   5 221   8 265 149 204 129   0]
 [  3  18   6 286   5 275  91 161 155   0]
 [  1  28   6 268   7 239 200 137 114   0]
 [  1  46   9 206   7 180  91 323 137   0]
 [  0  48   2 206   6 144  68 118 408   0]
 [  0  90  20 262   7 131 108 182 198   2]]
```

```
t[198]: <matplotlib.axes._subplots.AxesSubplot at 0x1cef6ece10>
```



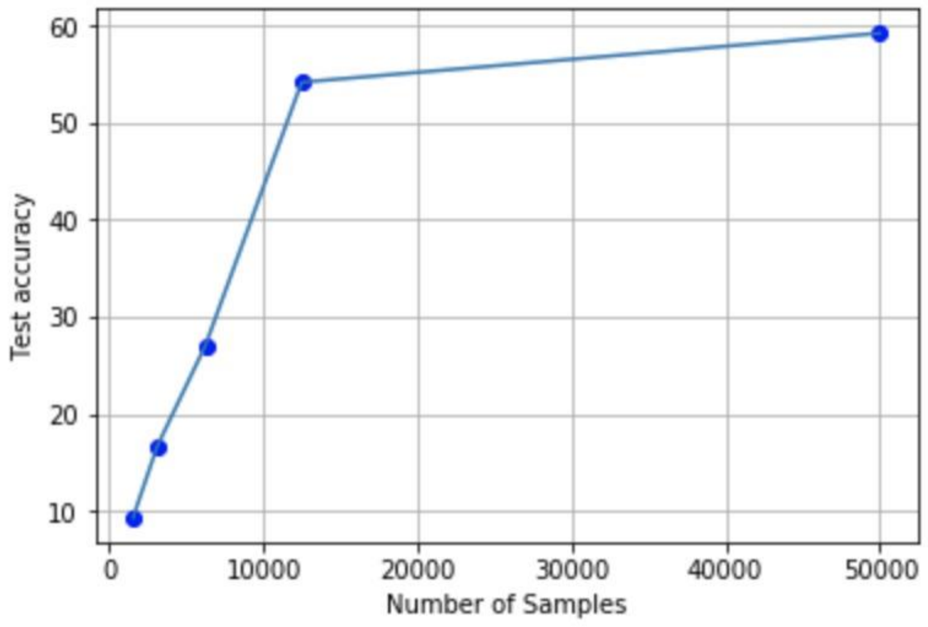
Case 4: (1/32) of 50,000 training images.

Testing accuracy = 9.31%



```
   ...: #Confusion matrix
   ...: from sklearn.metrics import confusion_matrix
   ...:
   ...: cf = confusion_matrix(y_test, y_pred)
   ...: print(cf)
   ...:
   ...: #Heat map
   ...: import seaborn as sns
   ...: sns.heatmap(cf, annot=True)
__main__:8: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,), for
example using ravel().
0.0931
[[  7  14 160  28 301  56 151 203  32  48]
 [ 32  30 179  80 114  59 126 267  33  80]
 [ 38  40 138 108 178  31 156 220  46  45]
 [ 33  45 196  93 113  40 133 236  43  68]
 [ 23  51 168 136 150  20 175 177  43  57]
 [ 43  74 151  93 174  20 138 235  24  48]
 [ 40  54 173 116 107  27 152 212  46  73]
 [ 10  35 198 114 145  41 118 247  18  74]
 [ 16  20 203  34 234  41 156 210  29  57]
 [ 23  20 186  56 131  51 138 296  34  65]]
Out[382]: <matplotlib.axes._subplots.AxesSubplot at 0x1a40a41790>
```

The curve of the test accuracy w.r.t to the number of training images for each case as follows.

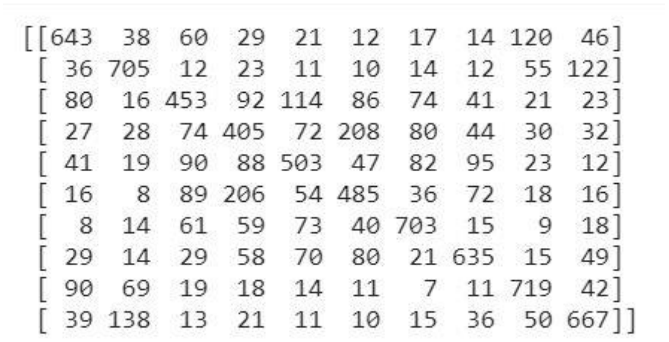


(b) Error Analysis:

Answers:

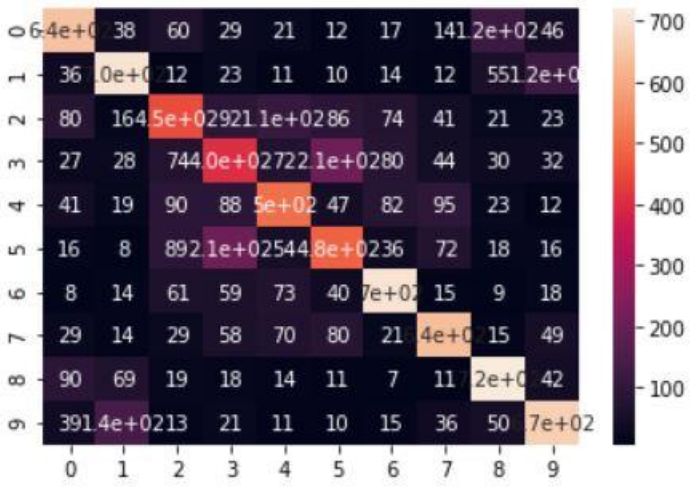
(1) The confusion matrix and the Heat Map is as follows:

Confusion Matrix:



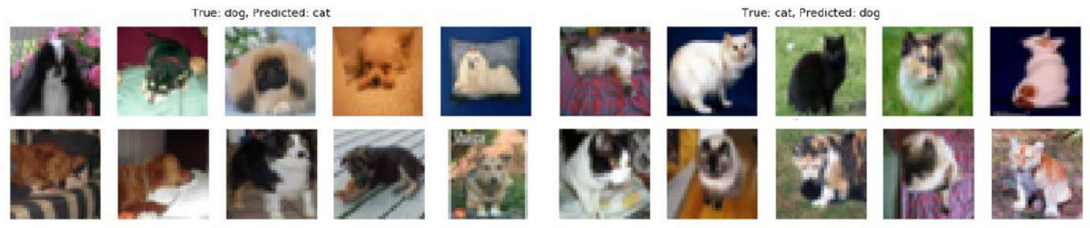
```
[[643  38  60  29  21  12  17  14 120  46]
 [ 36 705  12  23  11  10  14  12  55 122]
 [ 80  16 453  92 114  86  74  41  21  23]
 [ 27  28  74 405  72 208  80  44  30  32]
 [ 41  19  90  88 503  47  82  95  23  12]
 [ 16   8  89 206  54 485  36  72  18  16]
 [  8  14  61  59  73  40 703  15   9  18]
 [ 29  14  29  58  70  80  21 635  15  49]
 [ 90  69  19  18  14  11   7  11 719  42]
 [ 39 138  13  21  11  10  15  36  50 667]]
```

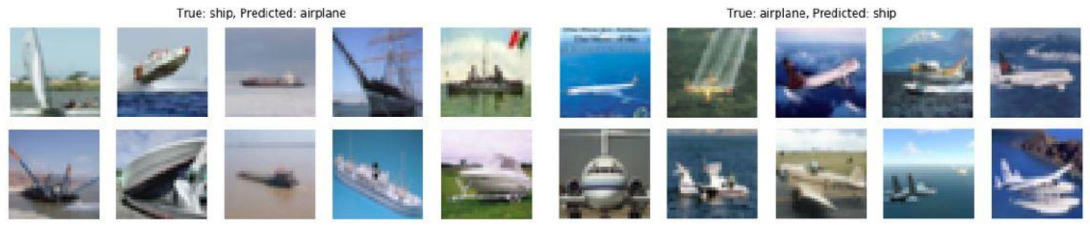
Heat Map:



As we can make out from the diagonal elements Confusion Matrix that the object class 8 i.e., the object class of **Ship** yields the lowest error rate as it has the highest value among the diagonal elements indicating that it is more clearly classified object class as compared to other while the object class 3 i.e., the object class of **Cat** has the highest error rate and it is one of the most difficult one to get properly classified among the various other object classes.

(2) The most confusing class groups for the CIFAR-10 dataset is the object class 3 and 5 i.e., the **Dog** and the **Cat** & object class 0 and 8 i.e., the **Airplane** and the **Ship.** The reason for these classes getting easily confused with each other can be explained below along with the examples as follows.
The Dog and the Cat object class can get easily confused with each other due to the appearance similarities and the facial appearance similarities, while the other category which can be easily to get confused with is the Ship and the Airplane object class, the reason being the similarity in the background i.e., the blue color of the sky and the ocean makes it difficult to distinguish between two object classes. The other reason may be due to the poor resolution of the images which makes it difficult to distinguish them which requires the features to be distinct.




True: ship, Predicted: airplane
True: airplane, Predicted: ship

(3) The ideas in order to improve the accuracy of the hard classes of the Pixel Hop++ technique can be discussed as follows:

- It can be improved by invoking the SSL-based contour/edge detection as well as image segmentation methods in order to improve the accuracy of the poor resolution images of the CIFAR-10 dataset. This might improve the accuracy due to the reason as the images can be now segmented using segmentation techniques and region-based classification can be performed to improve upon the results.
- Another technique can be Generative Adversarial Networks (GAN) which take into consideration of the data augmentation as well as the distribution of the attribute vectors of various kinds of neighbors, is shared similarly between the source as well as the target domain images.
- By changing various parameter such as the kernel size, strides, the seeds of the LAG units etc. has the ability to improve upon the testing accuracy of the hard or difficult object class of the Pixel Hop++.
- The increase in the number of hops with various different hyper-parameters can also support in improving the accuracy of the difficult object classes by increasing the number of attributes dimension.
- The different kind of pooling methods such as min, max, even and mean pooling techniques also has the ability to classify the object classes more accurately by considering only the features with the strongest discriminant power and having the low error rates.

<u>References:</u>

1. CNN as Guided Multi-layer RECOS Transform, C.-C. Jay Kuo, University of Southern California
2. On Data-Driven Saak Transform, C.-C. Jay Kuo and Yueru Chen, Ming-Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564, USA
3. Interpretable Convolutional Neural Networks via Feedforward Design C.-C. Jay Kuo, Min Zhang, Siyang Li, Jiali Duan and Yueru Chen, University of Southern California, Los Angeles, CA 90089-2564, USA
4. PIXELHOP: A SUCCESSIVE SUBSPACE LEARNING (SSL) METHOD FOR OBJECT CLASSIFICATION, Yueru Chen, C.-C. Jay Kuo, University of Southern California
5. PIXELHOP++: A SMALL SUCCESSIVE-SUBSPACE-LEARNING-BASED (SSL-BASED) MODEL FOR IMAGE CLASSIFICATION Yueru Chen_, Mozhdeh Rouhsedaghat_, Suya You+, Raghuveer Rao+ and C.-C. Jay Kuo_, University of Southern California, Los Angeles, California, USA