EE569- Introduction to Digital Image Processing

Homework #6 Project

Sourabh J. Tirodkar
3589406164
Submission Date- 3rd May 2020

Problem 3: EE569 Competition – CIFAR10 Classification using SSL

1. ABSTRACT AND MOTIVATION

   Successive Subspace Learning (SSL) was developed keeping in the mind the weakness of CNN method such as efficiency, scalability, etc. SSL is a new machine learning algorithm which is without backpropagation (BP) method and it is also scalable and mathematically explainable as well.

   Successive Subspace Learning (SSL) has been proven to a better implementation than the traditional CNN. The PixelHop and PixelHop++ have been implemented by Kuo *et al.* in which the implementation of the same is elaborated.

   PixelHop++ has been used in this Successive Subspace Learning (SSL) for this competition problem. The parameters mentioned in the problem 2 of homework 6 do not give us much higher accuracy. So, as to improve the performance for the same, the hyperparameters must be tuned.

   The parameters that I thought of have the capacity to improve the accuracy are increasing the depth, threshold values for Pixelhop++ units, aggregation in LAG units, number of clusters in LAG unit, changing classifier, number of selected features for each feature, etc.

   The PixelHop++ implementation by Kuo *et al.* [1] gives 2 Pixelhop++ model one with low model size with accuracy of 64.75% and other with large model size with accuracy of 66.81%.
   Efforts are made to combat the accuracy of 58.71% that I received in homework 6 problem 2 and reach the maximum accuracy by altering the parameters and reach nearer to the accuracy in the paper.

   My motivation to improve accuracy for this problem began soon after I found the vast applications of Pixelhop and Pixelhop++. As of the start of semester, Prof Kuo rightly said about a model better than Deep Learning which I was excited about.

   Checking the vast applications for the pixelhop++, it was quite necessary to keep improving the poxel hop unit for the best suiting model.

2. APPROACH

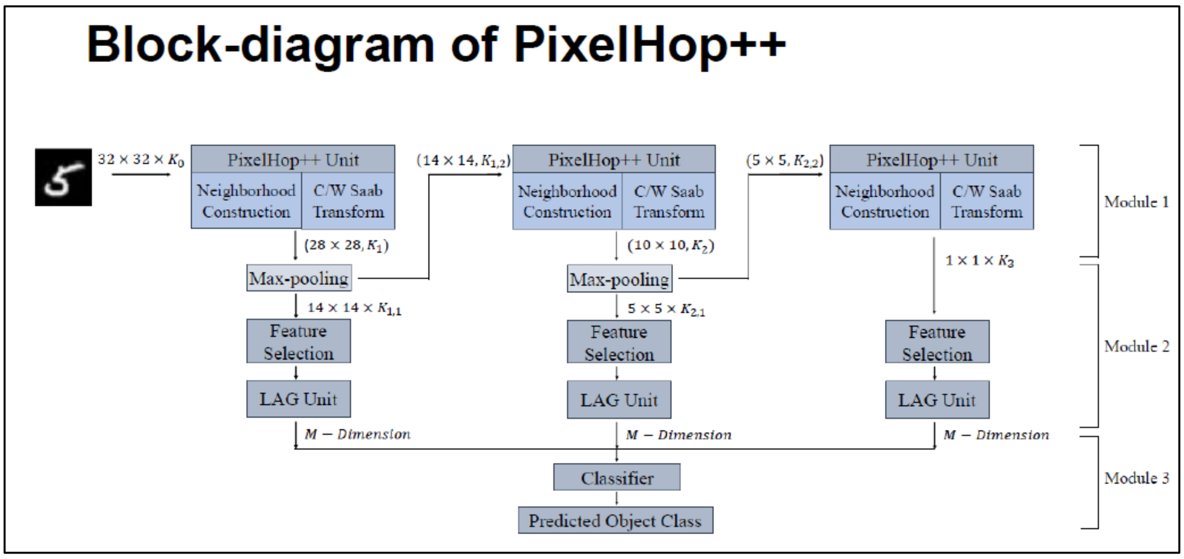


Fig. 2.1 PixelHop++ Model.

PixelHop++ Method:

PixelHop++ is an improved version of PixelHop in hich the model size is reduced and spatial-spectral features are hop jointly. It uses neighbourhood construction and channel wise Saab transform instead of basic Saab transform in the PixelHop method.

In the above diagram, my final parameters can be added and the model looks like this:

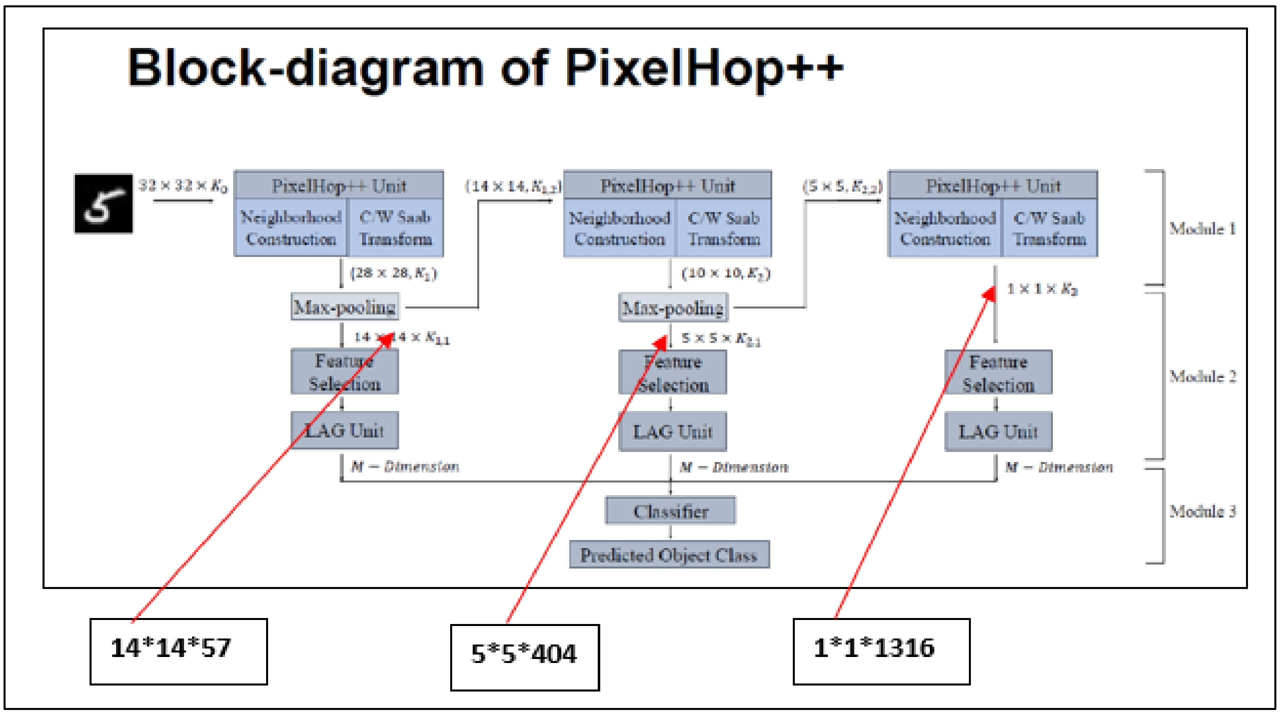


Fig. 2.2 PixelHop++ Model used for the implementation.

Reason choosing these parameters and Source of Improvement:

A. Depth: The paper at [1] has a depth of 3. The discussion of TAs helped me understand that depth does increase the accuracy to an extent but at the cost of model size. The model size goes up which isn't computionally effective. I tried the depth of 3 and 4. The depth at 4 was taking too much time to train and model size was also increasing. So, I sticked to chosing depth of 3.

B. Feature Selection: The number of selected features is any important factor while fitting the model. We cannot chose the whole features, of which some of them are of least important and won't contribute at all. So, it makes no sense to include all the features. I have selected top 1000 of the features for my implementation. The reason for the same is that chosing best features would make a difference.
Also, the paper at [1] clearly shows that if we select less features from each pixel hop unit, we tend to get lower accuracy. The graph below from the paper has X-axis with number of features while the test accuracy along the Y- axis. It can be concluded that more features, the better but would be bad idea to include all of them.

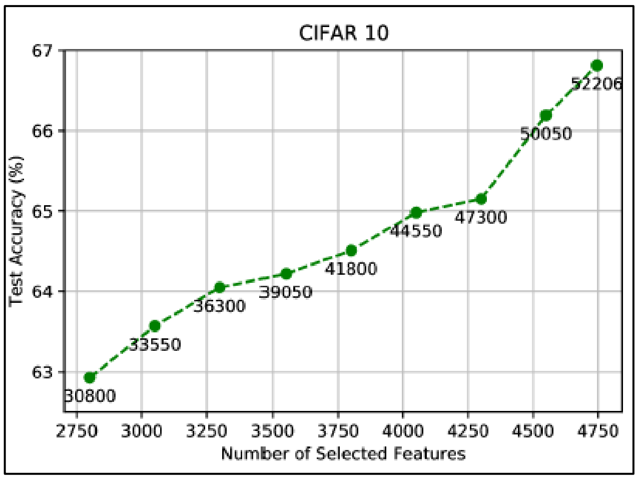


Fig. 2.3 Graph showing selected features vs test accuracy from paper [1].

C. Threshold:
We define 2 thresholds, TH1- Energy threshold for intermediate nodes and TH2- Energy threshold for discarded nodes. The threshold plays an important factor for deciding model size. A large threshold means the model size is large. The test accuracy decrease slightly as the threshold is decrease while model size is reduced significantly. The paper at [1] gives the graph for different threshold chosen for the CIFAR10 dataset. So, there is trade-off between threshold and the accuracy. I have chosen TH1= 0.0003 and TH2= 0.00003 for the same purpose.

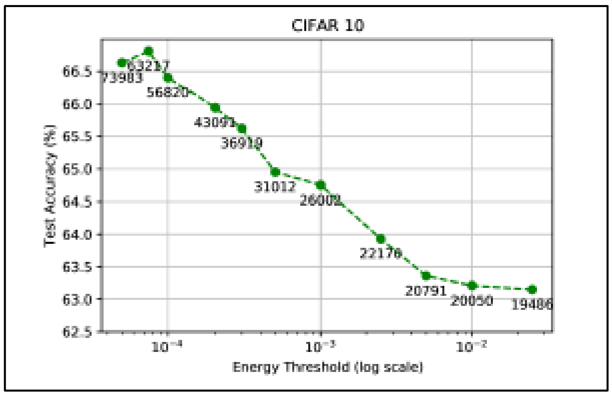
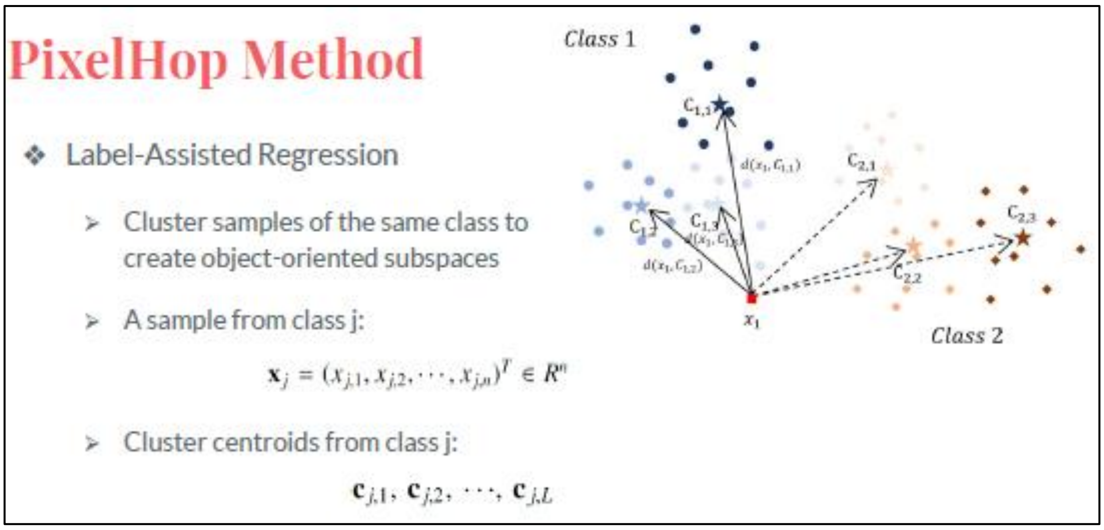


Fig. 2.4 Graph showing threshold vs test accuracy from paper [1].

D. Spatial Neighbourhood Size: The kernel window for the pixelhop units for the problem 2 was 5. When the depth is 4, we need to take window less than 5, which is for example 3. But as I have chosen depth of 3, I tried using window size 3 and 5, and found 5 as the better option for my implementation.

E. Number of centroid per class in LAG unit:
The functionality of LAG unit is to create clusters samples of the same class. The homework 6 problem mentioned the numberof centroid per class to be 5. I tried different values which were greater and lower to 5. I tried 5 and 10 for my implemenation and found out that there was not much difference in the accuracy for test images. So, I kept the centroid as 5.


PixelHop Method

❖ Label-Assisted Regression

➢ Cluster samples of the same class to create object-oriented subspaces

➢ A sample from class j:

$$\mathbf{x}_j = (x_{j,1}, x_{j,2}, \cdots, x_{j,n})^T \in R^n$$

➢ Cluster centroids from class j:

$$\mathbf{c}_{j,1}, \mathbf{c}_{j,2}, \cdots, \mathbf{c}_{j,L}$$

Fig. 2.5 Figure explaining LAG clustering

The model is trained using 10K images for the fit function. The transform function has been used for the batch processing of 5k images per batch.

Steps followed:

1. Load CIFAR10 dataset
2. Load saab, cwsaab, pixelhop2, crossentropy, lag, llsr
3. Choose hops (here 3) for pixelhop
4. Specifying the thresholds for the pixelhop++ units.
5. Fitting pixelhop.fit for the randomly chosen 10k images each has 1000 images for 1 class
6. Pixelhop.transform on 50k images each in the batch size of 5k
7. Display Module 1, output for each hop.
8. Module2: Reshaping output of module 1 into useable form
9. Cross-entropy for feature selection- using K-means (bins method can be used)
10. Selecting top 1000 features for each hop which are of lowest cross entropy.
11. Using lag.fit for the selected top 1000 features for each hop.
12. Lag.transform to get the all hops output in the same dimension (here- 50k,50)
13. Concatenate all 3 hops output from lag unit (Ouput- 50k, 150).
14. Using Random Forest as a classifier for predicting the training accuracy.
15. Taking testing data (here 10k images)
16. Doing pixelhop.transform, reshaping, lag.transform onto it.
17. Using same RF classifier to predict the accuracy.

3. RESULTS

<u>Best Choice of Parameter:</u>

Depth = 3

TH1 = 0.0003

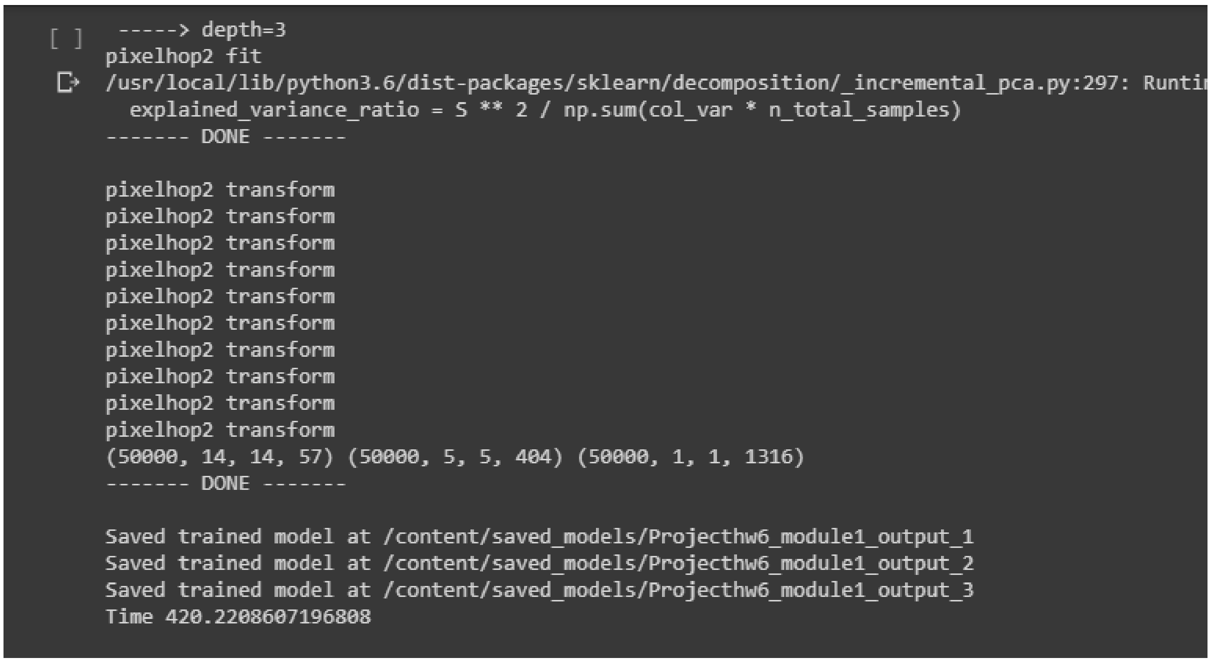
TH2 = 0.00003

Alpha = 10

Clusters = 5

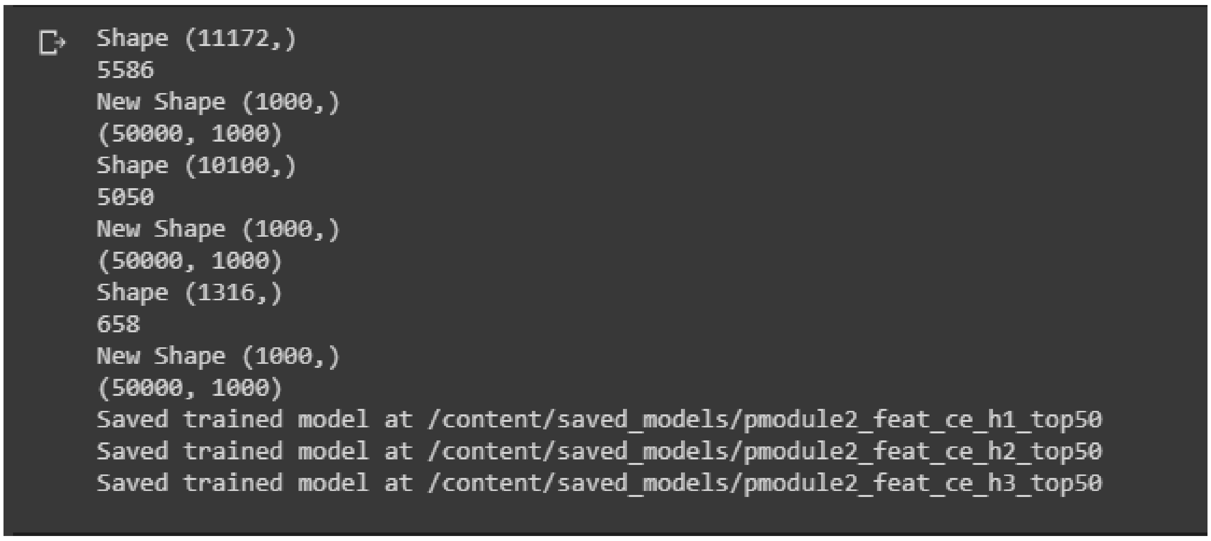
Random Forest Estimators = 500

The module 1 output is as shown:
I have put 10k images for the fit function. The transform function have been put in the batches of 5k. So, p2.transform has been used 10 times as shown in the screenshot below.



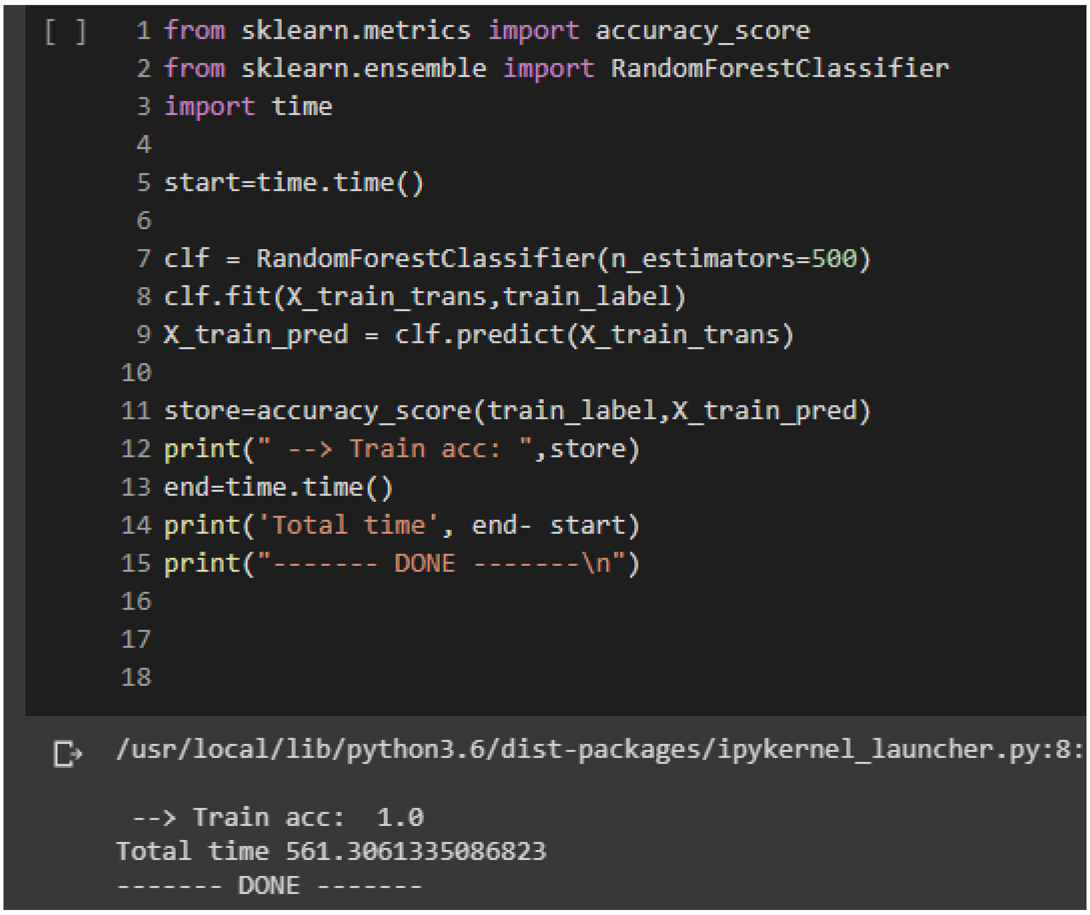
```
[ ]  -----> depth=3
    pixelhop2 fit
    /usr/local/lib/python3.6/dist-packages/sklearn/decomposition/_incremental_pca.py:297: Runti
      explained_variance_ratio = S ** 2 / np.sum(col_var * n_total_samples)
    ------- DONE -------

    pixelhop2 transform
    pixelhop2 transform
    pixelhop2 transform
    pixelhop2 transform
    pixelhop2 transform
    pixelhop2 transform
    pixelhop2 transform
    pixelhop2 transform
    pixelhop2 transform
    pixelhop2 transform
    (50000, 14, 14, 57) (50000, 5, 5, 404) (50000, 1, 1, 1316)
    ------- DONE -------

    Saved trained model at /content/saved_models/Projecthw6_module1_output_1
    Saved trained model at /content/saved_models/Projecthw6_module1_output_2
    Saved trained model at /content/saved_models/Projecthw6_module1_output_3
    Time 420.2208607196808
```

The Module 2 output is as shown:
From module 1, 14*14*57= 11172
5*5*404=10100
1*1*1316=1316
These are the matrix which has the features in it using k-means.
Selecting top 1000 of those features from all 3 hops The new feature dimension is of the size 1000 which are displayed below.



```
Shape (11172,)
5586
New Shape (1000,)
(50000, 1000)
Shape (10100,)
5050
New Shape (1000,)
(50000, 1000)
Shape (1316,)
658
New Shape (1000,)
(50000, 1000)
Saved trained model at /content/saved_models/pmodule2_feat_ce_h1_top50
Saved trained model at /content/saved_models/pmodule2_feat_ce_h2_top50
Saved trained model at /content/saved_models/pmodule2_feat_ce_h3_top50
```

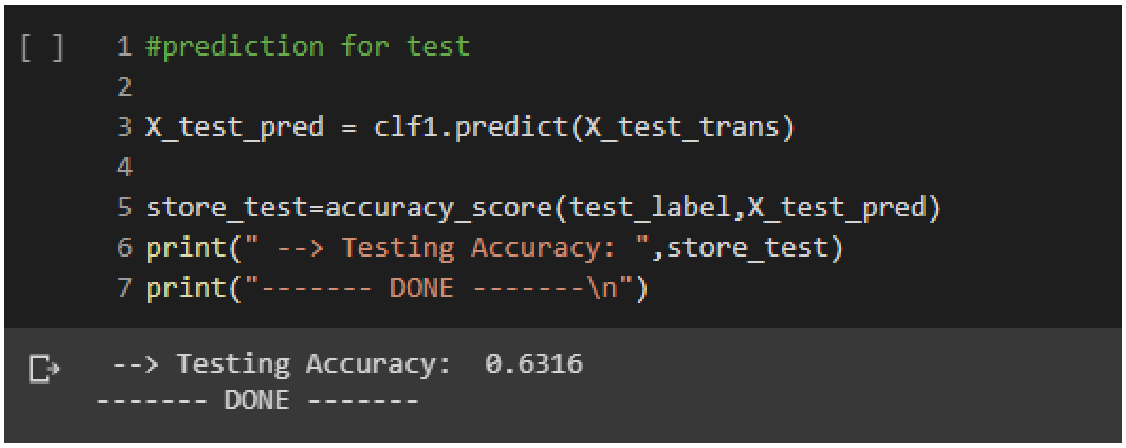
The Module 3 output is as shown:
Using RF classifer and trying on fit and predict onto the training data
Training Accuracy: 100%



```
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
import time

start=time.time()

clf = RandomForestClassifier(n_estimators=500)
clf.fit(X_train_trans,train_label)
X_train_pred = clf.predict(X_train_trans)

store=accuracy_score(train_label,X_train_pred)
print(" --> Train acc: ",store)
end=time.time()
print('Total time', end- start)
print("------- DONE -------\n")
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8:

 --> Train acc:  1.0
Total time 561.3061335086823
------- DONE -------
```

Taking testing data and using RF classifier onto it.



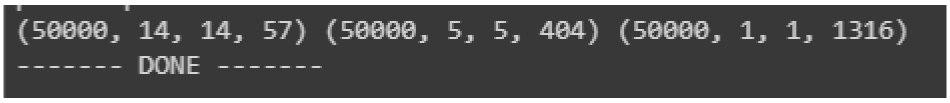
```
#prediction for test

X_test_pred = clf1.predict(X_test_trans)

store_test=accuracy_score(test_label,X_test_pred)
print(" --> Testing Accuracy: ",store_test)
print("------- DONE -------\n")
```

```
 --> Testing Accuracy:  0.6316
------- DONE -------
```

Model Size:

Module 1: Window size is 5*5*3 for hop 1 and 5*5*1 for hop 2 and 3

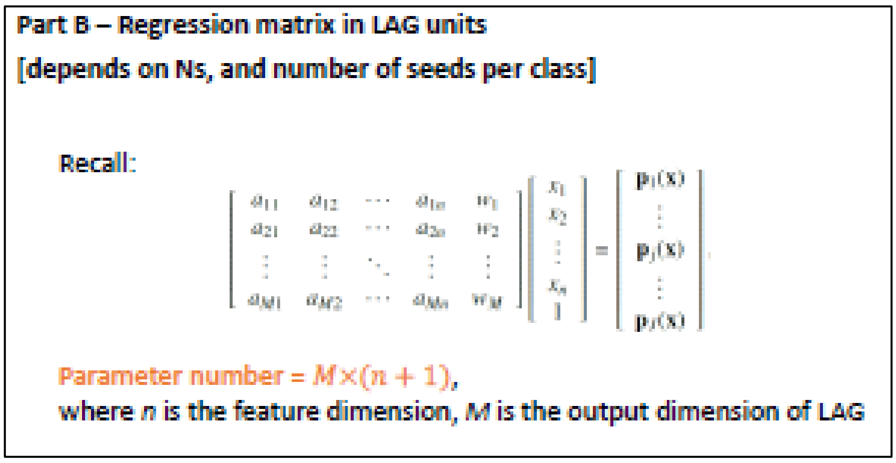
Output of module 1:



```
(50000, 14, 14, 57) (50000, 5, 5, 404) (50000, 1, 1, 1316)
------- DONE -------
```

So, Hop1: 5*5*3*57= 4275
Hop2: 5*5*1*404= 10100
Hop3: 5*5*1*1316= 32900
Thus, total model size of module 1: 4275+10100+32900= 47275.

Module 2:



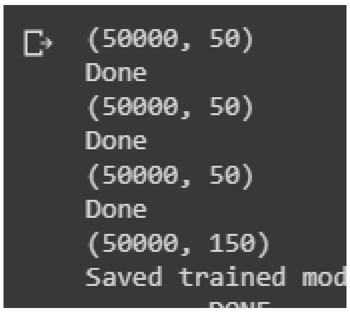
Part B – Regression matrix in LAG units
[depends on Ns, and number of seeds per class]

Recall:

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

Parameter number = $M \times (n+1)$,
where n is the feature dimension, M is the output dimension of LAG

Fig. 2.6 Model size for module 2



```
(50000, 50)
Done
(50000, 50)
Done
(50000, 50)
Done
(50000, 150)
Saved trained mod
```

Here, M= 50 for all hops
N1 = 1000
N2 = 1000
N3 = 1000
So hop 1, 50*(1000)= 50000
hop 2, 50*(1000)= 50000
hop 3, 50*(1000)= 50000
Total parameters= 150000

Module 3: Random Forest Classifier
N_estimators= 500
Max_depth= None
RF classifier model size isn't included in the total model size of the pixelhop module.

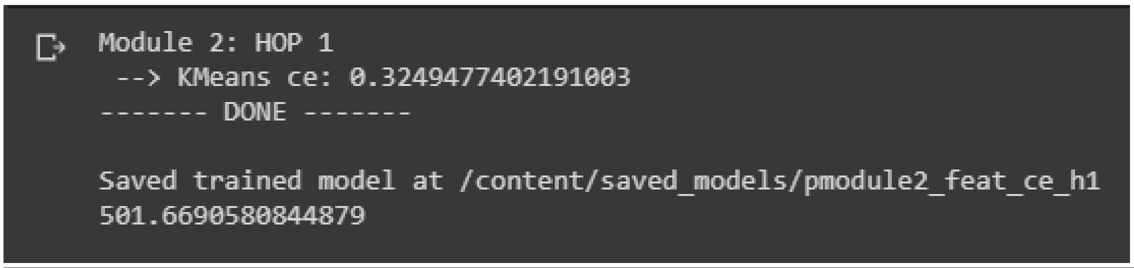
Total Model Size:
Module 1+ Module 2 = 197275

Model Time:

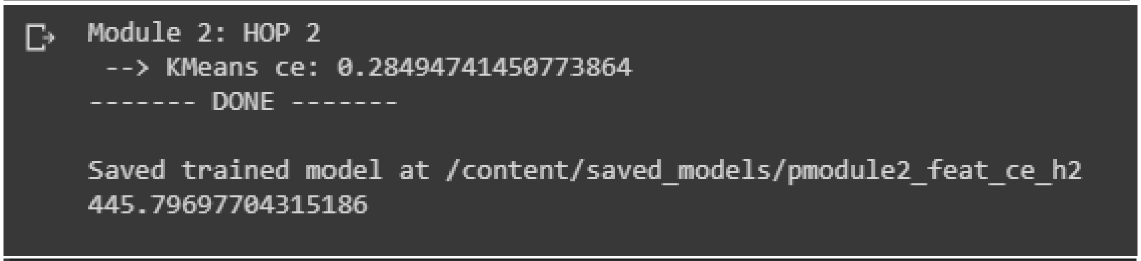
A. Training Time:

Module 1 : 420 sec (Around 7 mins)

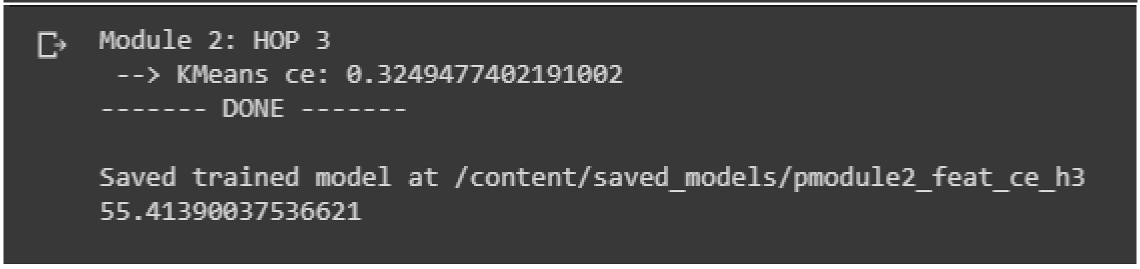
Module 2 : Hop1- 501 sec (8 mins) (K-means)
Hop 2- 445 sec (7 mins) (K-means)
Hop 3- 55 sec (1 min) (K-means)
Lag. Fit for all 3 hops: Around 10 mins



```
Module 2: HOP 1
 --> KMeans ce: 0.3249477402191003
------- DONE -------

Saved trained model at /content/saved_models/pmodule2_feat_ce_h1
501.6690580844879
```



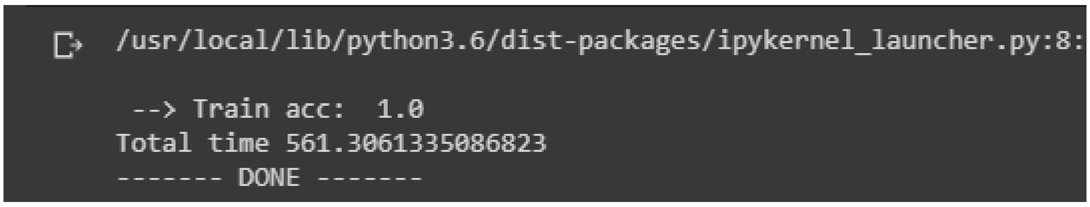
```
Module 2: HOP 2
 --> KMeans ce: 0.28494741450773864
------- DONE -------

Saved trained model at /content/saved_models/pmodule2_feat_ce_h2
445.79697704315186
```



```
Module 2: HOP 3
 --> KMeans ce: 0.3249477402191002
------- DONE -------

Saved trained model at /content/saved_models/pmodule2_feat_ce_h3
55.41390037536621
```

Total time for module 2: 8+7+1+10= 26 mins

Module 3 : RF Classifier- Around 10 mins



```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8:

 --> Train acc:  1.0
Total time 561.3061335086823
------- DONE -------
```

Total Time: 43 mins approximately (0.75 hrs)
Tuning RF classifier parameters and checking again for the training accuracy, not included.

B. <u>Inference Time:</u>

The inference time is the time which is done for the testing data. We first do pixelhop transform for the test images. The time required for the same is 1-2 mins. The predicting time using classifer is also 1 min. Thus the inference time for the model is approximately 4-5 mins.
If RF classifier training is included in the inference time, then it would be around 10-14 mins.

<u>Weak Supervision Graph</u>:

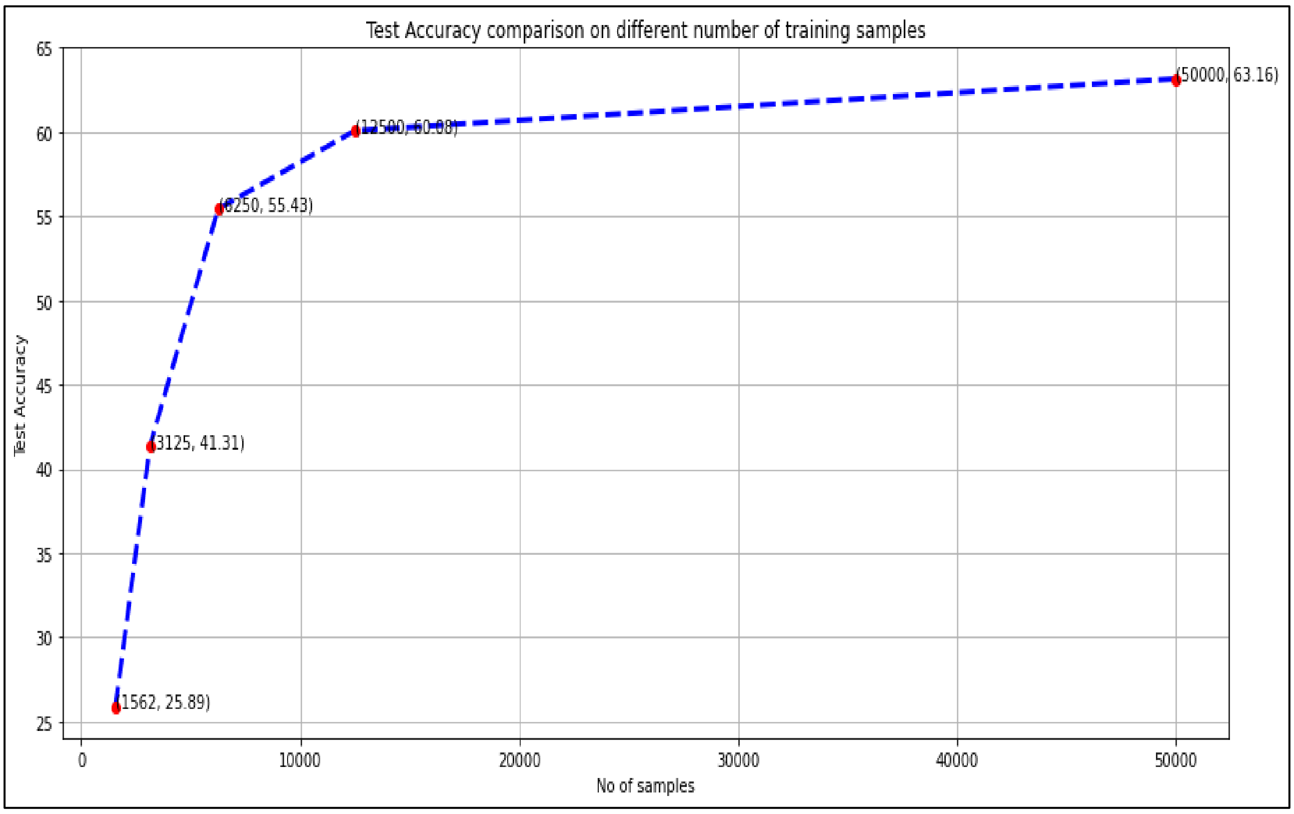


Fig. 2.7 Graph explaining weak supervision

The above graph shows that if the size of the training images is decreased, the testing images also decreases. It is quite understandable that more the images on the training dataset to be fit, the more model would be perfect. The graph is a weak supervision since as the training images get reduced the accuracy get reduced.

4. DISCUSSION

A. Comparing HW6 problem 2 with the current improved model

The parameters in HW6 problem 2 was given and with those parameters the accuracy of test images was low. In my current implementation, I have decided by myself to chose the hyperparameters by research and also trial and error.
The parameters difference between two are as below:

	HW6 Problem 2	**Current Implementation**
Depth	3	3
TH1	0.001	0.0003
TH2	0.0001	0.00003
Kernel Size	5	5
No of features	Top 50%	Top 1000
Alpha in LAG	10	10
Cluster in LAG	5	5
Classifier estimators- RF	300	500

Due to all these parameters, the test accuracy and the model size changes. We can see the test accuracy is improved in the current implementation.
The values of threshold reducing has significant impact on my testing accuracy to increase. The reason is that it increased my features for the each pixel hop unit.

B. Comparing BP-CNN of HW 5 Problem 2 with current SSL Model

The PixelHop++ paper by Kuo *et al.* [1] gives the comparsion between the Convolutional Neural Network (CNN) with the proposed SSL model. The author does gives comparison between the classification accuracy on test images and also on the model size.

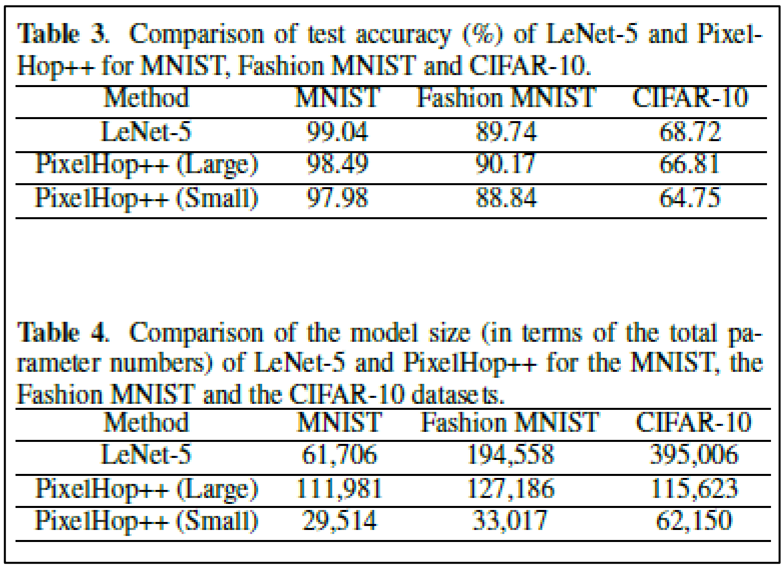


Table 3. Comparison of test accuracy (%) of LeNet-5 and PixelHop++ for MNIST, Fashion MNIST and CIFAR-10.

Method	MNIST	Fashion MNIST	CIFAR-10
LeNet-5	99.04	89.74	68.72
PixelHop++ (Large)	98.49	90.17	66.81
PixelHop++ (Small)	97.98	88.84	64.75

Table 4. Comparison of the model size (in terms of the total parameter numbers) of LeNet-5 and PixelHop++ for the MNIST, the Fashion MNIST and the CIFAR-10 datasets.

Method	MNIST	Fashion MNIST	CIFAR-10
LeNet-5	61,706	194,558	395,006
PixelHop++ (Large)	111,981	127,186	115,623
PixelHop++ (Small)	29,514	33,017	62,150

Fig. 2.8 Table from paper [1] giving evidence of test accuracy and model size.

Above is the table from the paper[1] where one can see the test accuracy on CIFAR10 dataset by the LeNet5 model and also by PixelHop++ model. Also, the model size comparsion is made where we can see there is much difference in the model size for the same.

My implementaion for the HW5 problem 2 and the current SSL model are as shown below:

	LeNet-5 (HW5 problem 2)	SSL (Current Implementation)
Test Accuracy	80.46	63.16
Model Size	315,972	197,275

As seen, there is much difference in the model size of CNN and SSL model. If the model size is increased as similar to CNN, we may get accuracy as same as the CNN one.
There is also the influence of training images used in the Module 1 of the pixelhop unit. As the images in fit function reducces, the accuracy would drop.
As the current SSL implementation can be extended by increasing the hops (depth) and the parameters, I feel SSL can give higher accuracy than LeNet5. Also, the model size is smaller than CNN architecture.
The model size is lower in SSL model as I have set threshold values higher.

REFERENCES:

1. Yueru Chen, Mozhdeh Rouhsedaghat, Suya You, Raghuveer Rao, C.-C. Jay Kuo, "PixelHop++: A Small Successive-Subspace-Learning-Based (SSL-based) Model for Image Classification," *https://arxiv.org/abs/2002.03141*, 2020

2. Yueru Chen, Yijing Yang, Wei Wang, C.-C. Jay Kuo, "Ensembles of Feedforward-designed Convolutional Neural Networks", in *International Conference on Image Processing*, 2019