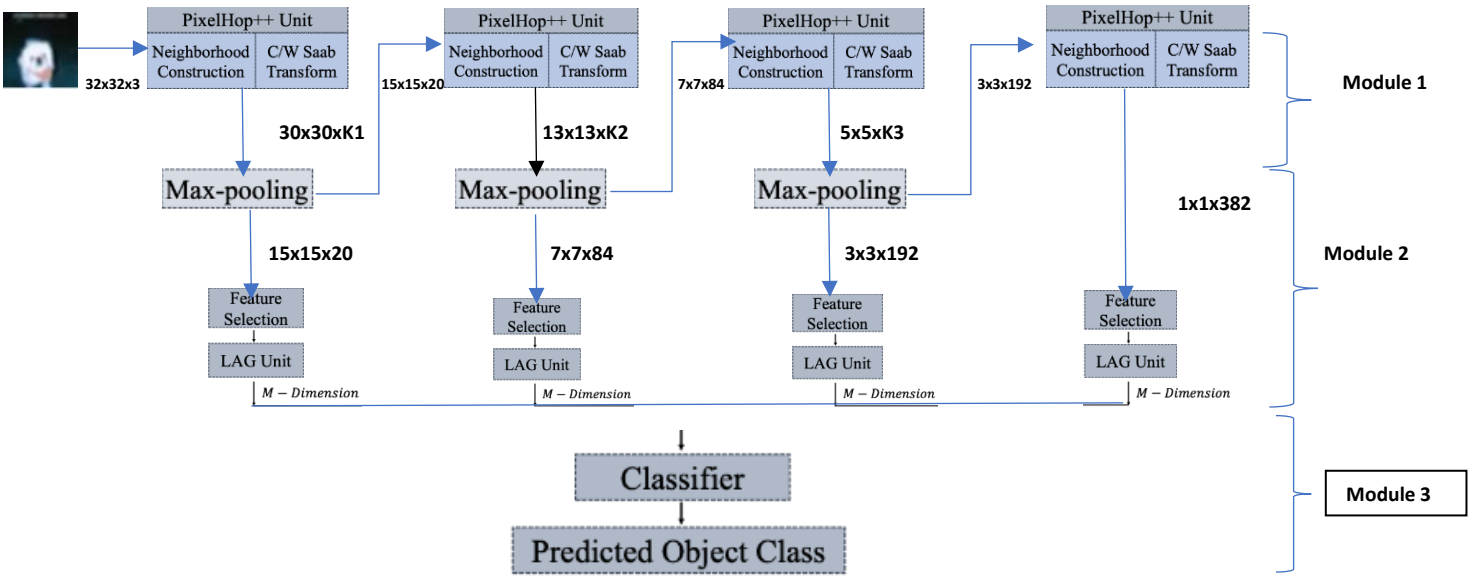
Name: Shreya Kate

Email: shreyak@usc.edu

USC ID: 2334973997

Homework 6 (P3)

My SSL system:



In this model, the input image is from CIFAR10 dataset which is a color image of size 32x32x3. I have cascaded 4 pixelhop++ units and done the max pooling in the shrink function to get the dimensions as shown in the figure. The feature selection is done using the kmeanscrossentropy or compute functions and the top 1000 features are selected and passed to the LAG unit and then the features from all 4 pixelhop++ units are concatenated and fed to the Random Forest classifier and we get the predicted object class from that.

Choosing hyperparameters

Hyperparameter	**Value**
Number of pixehop++ units	**4**
Spatial neighborhood size	**3x3**
stride	**1**
Pooling	**Max pooling**
Energy threshold (TH1)	**0.001**
Energy threshold (TH1)	**0.0001**
Number of selected features (Ns)	**1000**
Alpha	**5**
Number of centroids per LAG unit	**5**
Classifier	**Random Forest Classifier**

In my new SSL model, I changed the number of cascaded pixelhop++ units from 3 units to 4 units. This increase in depth of pixelhop++ units made the model more accurate.

I changed the spatial neighborhood size from 5x5 to 3x3. This led to a more detailed near-to-far view of the images and hence helped in improving accuracy. I changed the stride to 1 for a more intensive scan of the image.

I used max pooling in the shrink function to do pooling after each pixelhop++ unit.

I kept the energy threshold values as 0.001 and 0.0001 as these were giving highest accuracy. I tried to increase TH1 and decrease TH2 to see if that performed better, but I found that these threshold values worked best.

In P2(a) I had taken the top 50% number of selected features (Ns), but in this I took the top 1000 and that made a huge impact on the accuracy.

I experimented with various alpha values ranging from 5-15 and found that the values of alpha=5 worked best for my model. I kept the number of centroids per LAG unit as 5.

I used Random Forest Classifier for classification and kept the max_depth in that as 15 to get the most accurate predictions.

By implementing the above changes in hyperparameters I was able to get the test accuracy to 60.33%.

Accuracy for full and weak supervision:

50000 training images:

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
model = RandomForestClassifier(max_depth = 18, n_jobs=-1)

model.fit(lag_concat,y_train.ravel())

rfc_predict = model.predict(lag_concat)
print('train accuracy:',accuracy_score(y_train, rfc_predict))
pred = model.predict(lag_concat_t)
print('test accuracy:',accuracy_score(y_test, pred))
# model.fit(lag_concat, y_train.ravel())

# pred = model.predict(lag_concat_t)
```

```
train accuracy: 0.94038
test accuracy: 0.6033
```

I got a test accuracy of 60.33% and training accuracy of 94.038%

12500 training images:

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
model = RandomForestClassifier( max_depth = 15, random_state=2, n_jobs=-1)

model.fit(lag_concat,y_train[0:nti].ravel())

rfc_predict = model.predict(lag_concat)
print('train accuracy:',accuracy_score(y_train[0:nti].ravel(), rfc_predict))
pred = model.predict(lag_concat_t)
print('test accuracy:',accuracy_score(y_test, pred))

end = time.time()
print('Module 3:',(end - start))
```

```
pixelhop2 transform
(10000, 15, 15, 20) (10000, 7, 7, 84) (10000, 3, 3, 192) (10000, 1, 1, 382)
(10000, 200)
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:34: DataConversi
train accuracy: 0.98216
test accuracy: 0.4495
Module 3: 28.221314907073975
```

I got a test accuracy of 44.95% and training accuracy of 98.216%

6250 training images:

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
model = RandomForestClassifier( max_depth = 15, random_state=2, n_jobs=-1)

model.fit(lag_concat,y_train[0:nti].ravel())

rfc_predict = model.predict(lag_concat)
print('train accuracy:',accuracy_score(y_train[0:nti].ravel(), rfc_predict))
pred = model.predict(lag_concat_t)
print('test accuracy:',accuracy_score(y_test, pred))

end = time.time()
print('Module 3:',(end - start))
```

```
pixelhop2 transform
(10000, 15, 15, 20) (10000, 7, 7, 84) (10000, 3, 3, 192) (10000, 1, 1, 382)
(10000, 200)
train accuracy: 0.99952
test accuracy: 0.289
Module 3: 23.977113246917725
```

I got a test accuracy of 28.9% and training accuracy of 99.952%

3125 training images:

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
model = RandomForestClassifier( max_depth = 15, random_state=2, n_jobs=-1)

model.fit(lag_concat,y_train[0:nti].ravel())

rfc_predict = model.predict(lag_concat)
print('train accuracy:',accuracy_score(y_train[0:nti].ravel(), rfc_predict))
pred = model.predict(lag_concat_t)
print('test accuracy:',accuracy_score(y_test, pred))

end = time.time()
print('Module 3:',(end - start))
```

```
pixelhop2 transform
(10000, 15, 15, 20) (10000, 7, 7, 84) (10000, 3, 3, 192) (10000, 1, 1, 382)
(10000, 200)
train accuracy: 1.0
test accuracy: 0.1135
Module 3: 20.57886290550232
```

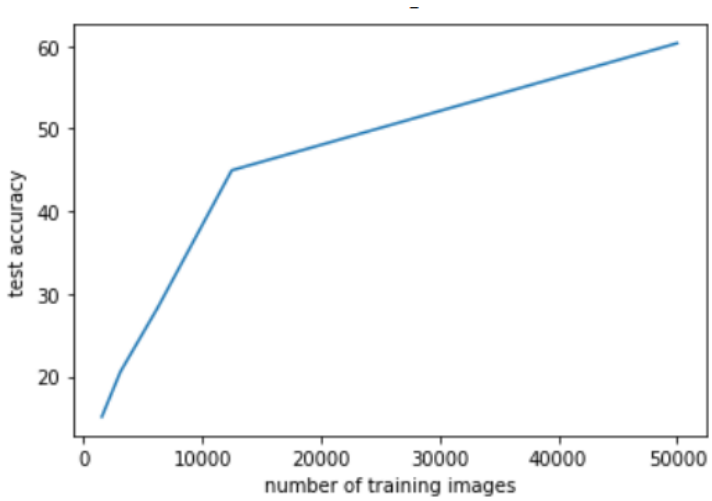
I got a test accuracy of 20.57% and training accuracy of 100%

1562 training images:

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
model = RandomForestClassifier( max_depth = 15, random_state=2, n_jobs=-1)

model.fit(lag_concat,y_train[0:nti].ravel())

rfc_predict = model.predict(lag_concat)
print('train accuracy:',accuracy_score(y_train[0:nti].ravel(), rfc_predict))
pred = model.predict(lag_concat_t)
print('test accuracy:',accuracy_score(y_test, pred))

end = time.time()
print('Module 3:',(end - start))
```

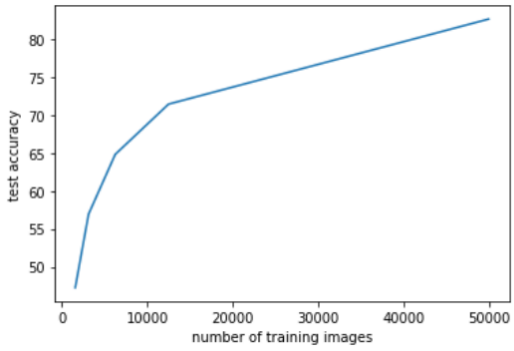
```
pixelhop2 transform
(10000, 15, 15, 20) (10000, 7, 7, 84) (10000, 3, 3, 192) (10000, 1, 1, 382)
(10000, 200)
train accuracy: 1.0
test accuracy: 0.1513
Module 3: 20.064886808395386
```

I got a test accuracy of 15.13% and training accuracy of 100%

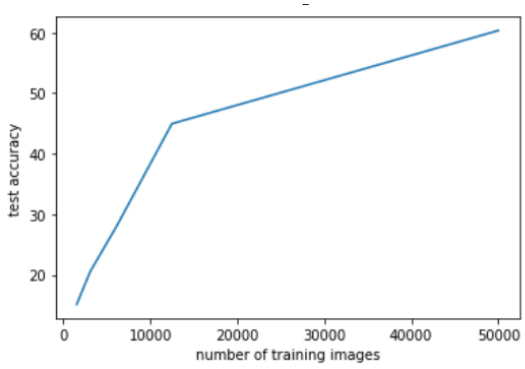


The plot shows the test accuracy vs number of training images. As the number of training images gets lesser, the test accuracy drops.

Now, let us compare the curves for BP-CNN and SSL.



BP-CNN



SSL

For both the models, in weak supervision the training numbers are reduced to 12500, 6250, 3125 and 1562. The number of training images vs test accuracy is plotted in the graphs above for both BP-CNN and SSL. We can see that in BP-CNN the test accuracy drops gradually as the number of training images are reduced,

whereas for SSL, the drop in test accuracy is quite steep. Also, the lowest test accuracy in SSL drops to 15% whereas in BP-CNN it is 47%. So, BP-CNN performs better in weak supervision too.

The test accuracy for 50,000 training images in BP-CNN is 83% whereas in SSL, it is 60-61%

Training and Inference time for 50000 training images and Ns = 1000:

I have used Google colab and used their GPU to speed up the process.

Time required for execution of:

Module 1: 123.27 sec

Module 2: 1034.52 sec

Module 3: 58.125 sec

Therefore, training time = Module 1 + Module 2 = 1157 sec = 19.28 minutes

Inference time = Module 3 = 129.117 sec

I was using stratifiedkfold earlier but I noticed that it wasn't helping much with the accuracy so I decided to drop that.

Model size:

Model size:

Part A

Hop 1: 15x15x20 = 4500
Hop 2: 7x7x84 = 4116
Hop 3: 3x3x192 = 1728
Hop 4: 1x1x382 = 382

Part B

$M\times(n+1)$
Hop 1: 50 x ((15x15x20)+1) = 2,25,050
Hop 2: 50 x ((7x7x84)+1) = 2,05,850
Hop 3: 50 x ((3x3x192)+1) = 86,450
Hop 4: 50 x ((1x1x382)+1) = 19,150

Part C (Classifier)

(number of hops)x(M)x(number of classes)

4x50x10 = 2000

Module 1: 3x3x3x20+ 3x3x3x84 + 3x3x3x192 + 3x3x3x382 = 18306

Module 2: 1000x50 + 1000x50 + 1000x50 + 382x50 = 169100

Total number of parameters = 18306 + 169100 = 187406