<u>Analysis part</u>

Question 1

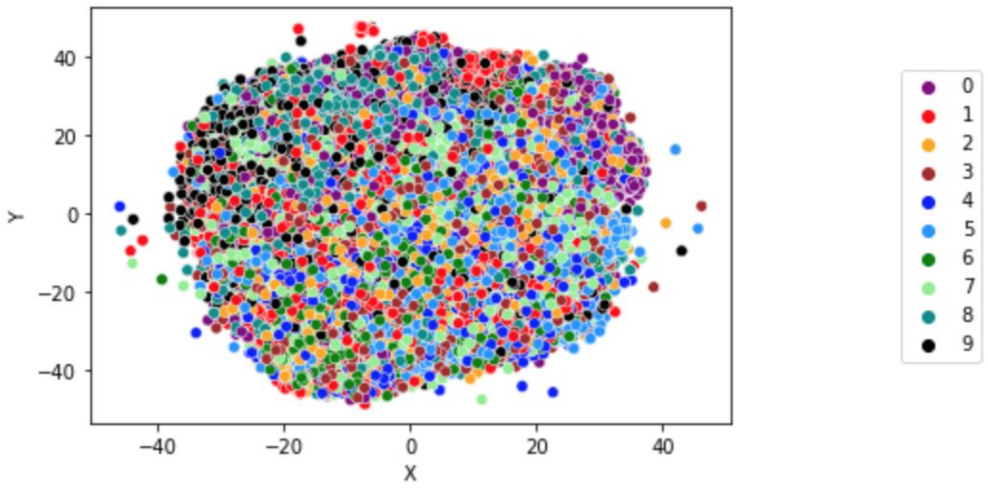
1) a)PCA that is Principal Component Analysis is used to reduce the dimension of the data.Here we have to retain 90% of total variance of the data.This helps in the speeding up of the training process.

b) HOG is used to extract features from the images.It extracts information about the edges and orientation of the edges.

- We have taken orientation of 8 that is number of bins in the histogram.
- The pixel per cell is 8x8 that is size of every cell.
- Cells per block 1x1. Is the the cells per block.

Color histogram gives the presentation of components of color which can be used as features for the model.

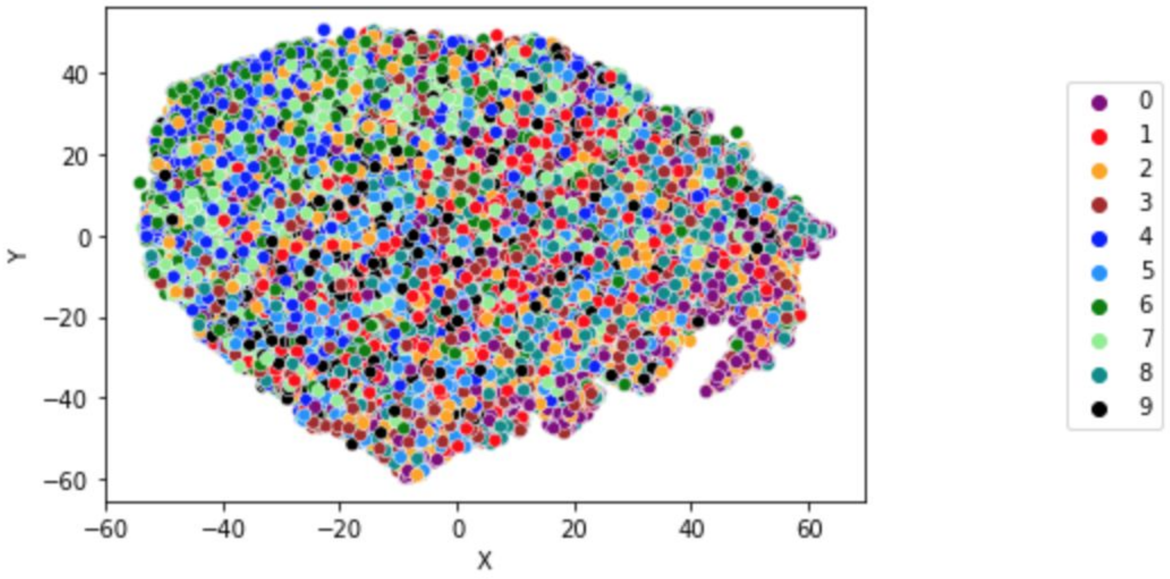
2) Using PCA TSNE



- Loaded the data using unpickle and numpy libraries and converted it to numpy array.
- Given data was preprocessed by PCA with 90% variance retained.
- Dataset was huge i.e 50000,3072 which converted to 50000,99
- Here airplane : 0, automobile : 1, bird : 2, cat : 3, deer : 4, dog : 5, frog : 6, horse : 7, ship : 8, truck : 9
- All data points are almost close to each other due to dimension reduction.
- Similar data points are forming small clusters internally.

- Dimension of data has been reduced very drastically into 2.

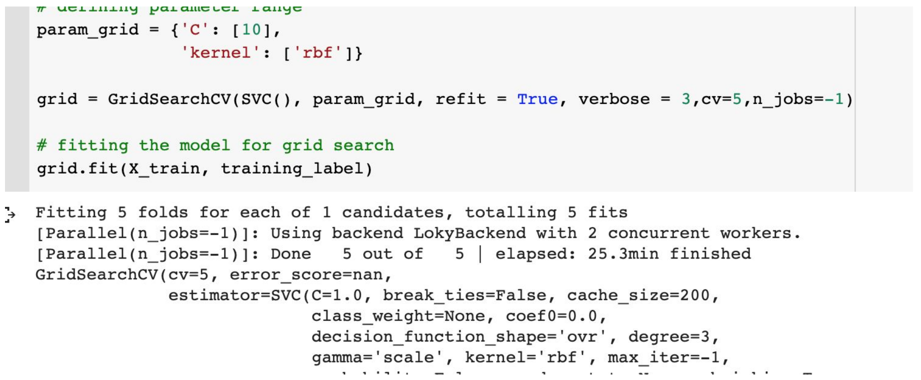
Using HOG+Color Histogram TSNE



- Loaded the data using unpickle and numpy libraries and converted it to numpy array.
- Given data was preprocessed by HOG and color histogram.
- Dataset was huge i.e 50000,3072 which converted to 50000,138
- Here airplane : 0, automobile : 1, bird : 2, cat : 3, deer : 4, dog : 5, frog : 6, horse : 7, ship : 8, truck : 9
- All data points are almost close to each other due to dimension reduction.
- Similar data points are forming small clusters internally.
- Dimension of data has been reduced very drastically.

3) Grid Search CV PCA

- Loaded the data using unpickle and numpy libraries and converted it to numpy array.
- Given data was preprocessed by PCA with 90% variance retained.
- Dataset was huge i.e 50000,3072 which converted to 50000,99.
- Further GridsearchCV was used to get the best model for SVM.
- Best Accuracy was obtained on Radial Basis Function(RBF) SVM kernel with C=10 and gamma= scale. We didn’t give gamma explicitly.
- Train Accuracy obtained was 92% , Test Accuracy obtained was 56%
- Running time was alleast 25.3 min.

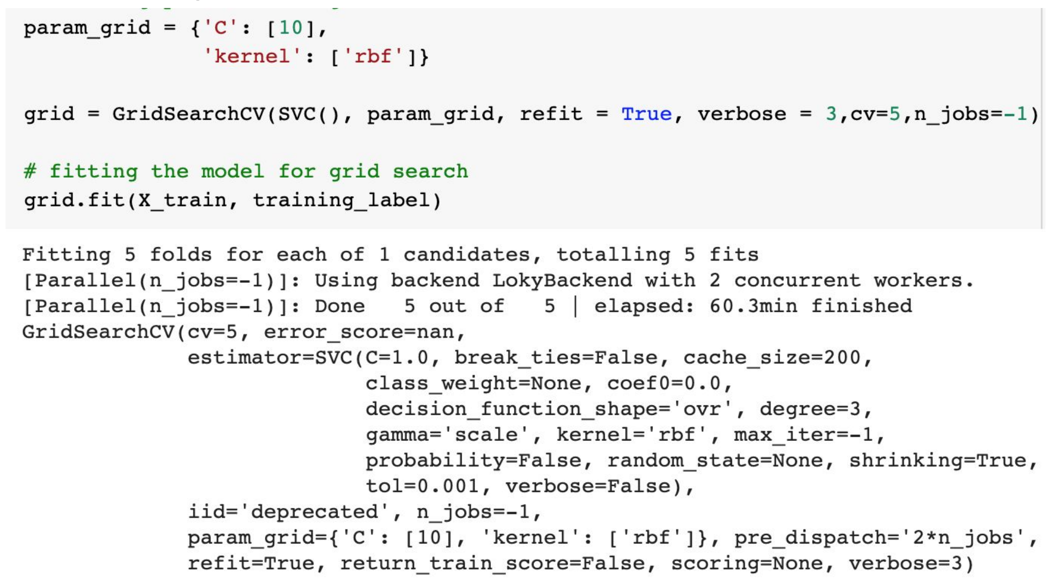


```
# defining parameter range
param_grid = {'C': [10],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3,cv=5,n_jobs=-1)

# fitting the model for grid search
grid.fit(X_train, training_label)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed: 25.3min finished
GridSearchCV(cv=5, error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
```

Grid Search CV (HOG+Histogram)

- Loaded the data using unpickle and numpy libraries and converted it to numpy array.
- Given data was preprocessed by HOG and color histogram to reduce the dimension.
- Dataset was huge i.e 50000,3072 which converted to 50000,138.
- Further GridsearchCV was used to get the best model for SVM.
- Best Accuracy was obtained on Radial Basis Function(RBF) SVM kernel with C=10 and gamma= scale. We didn’t give gamma explicitly.
- Preprocessed the data using preprocessing.scale.
- Train Accuracy obtained was 96%
- Test Accuracy obtained was 35%
- Running time was alleast 60.3 min.



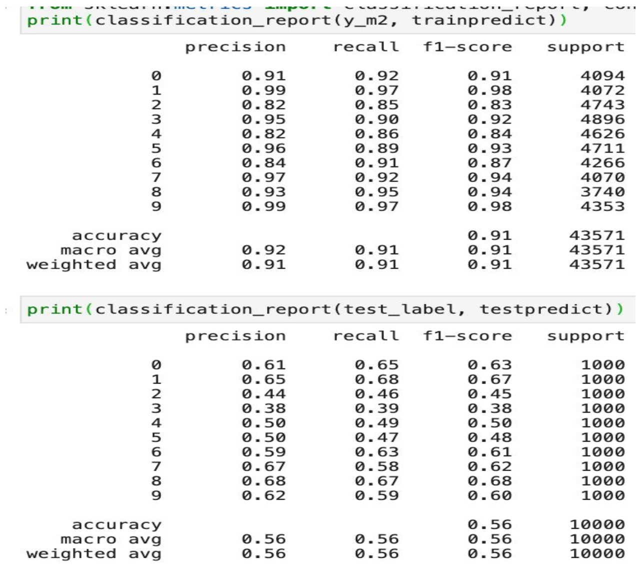
```
param_grid = {'C': [10],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3,cv=5,n_jobs=-1)

# fitting the model for grid search
grid.fit(X_train, training_label)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed: 60.3min finished
GridSearchCV(cv=5, error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'C': [10], 'kernel': ['rbf']}, pre_dispatch='2*n_jobs',
             refit=True, return_train_score=False, scoring=None, verbose=3)
```

4) SVM extract Support vector PCA

- Load the best model from previous grid search.
- Used svm.support_ to extract the support vector indices of the model.
- Then took the data from the original dataset at given indices.

- Used the same support vector for Xtrain and yTrain in another SVM of the same parameter.
- The accuracies are almost the same i.e Train Accuracy is 91% and Test Accuracy is 56%.
- Their is reduction in 6429 points in training set i.e they aren't in support vectors of previous SVM.

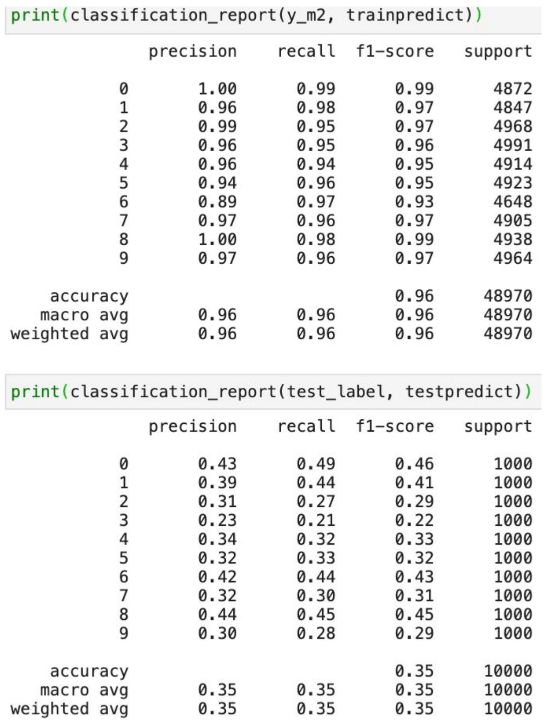


```
print(classification_report(y_m2, trainpredict))

              precision    recall  f1-score   support

           0       0.91      0.92      0.91      4094
           1       0.99      0.97      0.98      4072
           2       0.82      0.85      0.83      4743
           3       0.95      0.90      0.92      4896
           4       0.82      0.86      0.84      4626
           5       0.96      0.89      0.93      4711
           6       0.84      0.91      0.87      4266
           7       0.97      0.92      0.94      4070
           8       0.93      0.95      0.94      3740
           9       0.99      0.97      0.98      4353

    accuracy                           0.91     43571
   macro avg       0.92      0.91      0.91     43571
weighted avg       0.91      0.91      0.91     43571
```

```
print(classification_report(test_label, testpredict))

              precision    recall  f1-score   support

           0       0.61      0.65      0.63      1000
           1       0.65      0.68      0.67      1000
           2       0.44      0.46      0.45      1000
           3       0.38      0.39      0.38      1000
           4       0.50      0.49      0.50      1000
           5       0.50      0.47      0.48      1000
           6       0.59      0.63      0.61      1000
           7       0.67      0.58      0.62      1000
           8       0.68      0.67      0.68      1000
           9       0.62      0.59      0.60      1000

    accuracy                           0.56     10000
   macro avg       0.56      0.56      0.56     10000
weighted avg       0.56      0.56      0.56     10000
```

SVM extract Support vector (HOG+Color Histogram)

- Loaded the best model from previous grid search.
- Used svm.support_ to extract the support vector indices of the model.
- Then took the data from the original dataset at given indices.
- Used the same support vector for Xtrain and yTrain in another SVM of the same parameter.
- The accuracies are almost the same i.e Train Accuracy is 97% and Test Accuracy is 35%.
- There is reduction in 1030 points in the training set i.e they aren't in support vectors of previous SVM.



```
print(classification_report(y_m2, trainpredict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      4872
           1       0.96      0.98      0.97      4847
           2       0.99      0.95      0.97      4968
           3       0.96      0.95      0.96      4991
           4       0.96      0.94      0.95      4914
           5       0.94      0.96      0.95      4923
           6       0.89      0.97      0.93      4648
           7       0.97      0.96      0.97      4905
           8       1.00      0.98      0.99      4938
           9       0.97      0.96      0.97      4964

    accuracy                           0.96     48970
   macro avg       0.96      0.96      0.96     48970
weighted avg       0.96      0.96      0.96     48970
```

```
print(classification_report(test_label, testpredict))
```

```
              precision    recall  f1-score   support

           0       0.43      0.49      0.46      1000
           1       0.39      0.44      0.41      1000
           2       0.31      0.27      0.29      1000
           3       0.23      0.21      0.22      1000
           4       0.34      0.32      0.33      1000
           5       0.32      0.33      0.32      1000
           6       0.42      0.44      0.43      1000
           7       0.32      0.30      0.31      1000
           8       0.44      0.45      0.45      1000
           9       0.30      0.28      0.29      1000

    accuracy                           0.35     10000
   macro avg       0.35      0.35      0.35     10000
weighted avg       0.35      0.35      0.35     10000
```