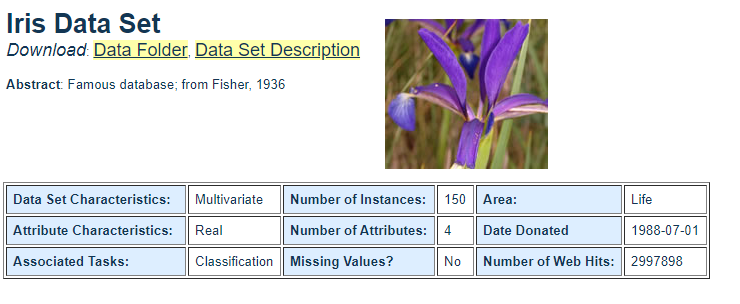
GRADUATE PROJECT

202010-CSCI/STAT- Statistical/Machine Learning and Data Mining

IRIS DATSET

Dataset Description:

Citations: <https://archive.ics.uci.edu/ml/datasets/iris>



I have chosen this dataset because, I find this as the standard Dataset that is being used in most of the examples for explaining the Machine learning algorithms

I have applied Standard algorithm like:

* [Support Vector Machines](https://en.wikipedia.org/wiki/Support_Vector_Machines)
* Quadratic Discriminant Analysis
* [Naive Bayes](https://en.wikipedia.org/wiki/Naive_Bayes_classifier)
* [linear discriminant analysis](https://en.wikipedia.org/wiki/Linear_discriminant_analysis)
* [Decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning)
* Random Forest classifier
* [k-nearest neighbor algorithm](https://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm)

Following is the Result that is obtained after applying PCA technique along with Meshgrid Plot

I would like to show clearly the boundary differentiation and prediction when multiple algorithms are being used and calculate the accuracy and error

The following is the parameters that is used in the Algorithms

#Citations: <http://uc-r.github.io/discriminant_analysis>

#geeks for geeks referenced for Machine learning definitions

If it’s a two-class classification, we use Linear classification Algorithm

If there are many dimensions, we go with LDA and QDA

**LDA()**

I have tried Linear discriminant Analysis applying Basic algorithm which is used for Dimensionality reduction

However, I am trying to use this for prediction

**QDA()**

I have tried Quadratic Discriminant analysis applying Basic algorithm which is used for Dimensionality reduction

However, I am trying to use this for prediction

**NB()**

Naive Bayes is based on probabilistic learning algorithm which is Intuitive

**RandomForestClassifier (n\_estimators=10, criterion='entropy',random\_state=0)**

Random forest is an Ensemble learning algorithm capable of performing both regression and classification

I have used this algorithm with n\_estimators of 10

**SVC(kernel='rbf',random\_state=0)**

SVM is a discriminative classifier formed by separating hyperplane divided by a planes and support vector points

I have used the rbf kernel here

**DecisionTreeClassifier(criterion='entropy',random\_state=0)**

Decision Tree classifier is a that is tree like structure which sorts out the data from the root

i=1,5,10

**KNeighborsClassifier (n\_neighbors=i,n\_jobs=-1)**

Knn is used for both classification and regression that tries to find near neighbors that belongs to there group

Kmeans is an unsupervised learning algorithm that forms category based on K samples/cluster

**KMeans (n\_clusters =10,init='k-means++',max\_iter =300,n\_init=10,random\_state=0)**

I have initialized Kmeans ++ that is used for that is sensitive for initialization points

**------------------LDA-K-FOLD Mean Accuracy------------------------------**

**0.973**

**------------------QDA-K-FOLD Mean Accuracy------------------------------**

**0.98**

**------------------NB-K-FOLD Mean Accuracy--------------------------------**

**0.893**

**------------------KNN- 1-K-FOLD Mean Accuracy----------------------------**

**0.953**

**------------------KNN- 5-K-FOLD Mean Accuracy----------------------------**

**0.946**

**------------------KNN- 10-K-FOLD Mean Accuracy----------------------------**

**0.98**

**------------------Random forest-K-FOLD Mean Accuracy--------------------**

**0.953**

**------------------Decision Tree -K-FOLD Mean Accuracy---------------------**

**0.98**

**------------------SVM -K-FOLD Mean Accuracy---------------------------------**

**0.973**

**------------------LDA-K-FOLD Mean Error----------------------------**

**0.026**

**------------------QDA-K-FOLD Mean Error----------------------------**

**0.02**

**------------------NB-K-FOLD Mean Error----------------------------**

**0.106**

**------------------KNN-K-FOLD Mean Error----------------------------**

**0.046**

**------------------KNN-5-K-FOLD Mean Error----------------------------**

**0.053**

**------------------KNN-10-K-FOLD Mean Error----------------------------**

**0.02**

**------------------RF -K-FOLD Mean Error----------------------------**

**0.046**

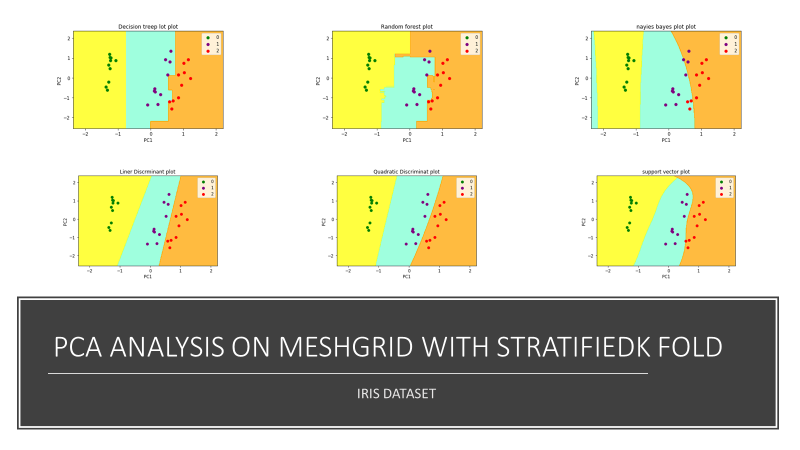
**------------------DT -K-FOLD Mean Error----------------------------**

**0.02**

**------------------SVM-K-FOLD Mean Error----------------------------**

**0.026**

As per the results we see that basic learning algorithm can perform well on a limited dataset and QDA performs well here with an accuracy rate of **98 %**



Note: Generated Plots are attached in folder IRIS\_DATSET\_PLOT

DATASET 2

WHEN DATSET IS LARGER?

CIFAR DATSET

Whenever the Dataset changes as per the example of CIFAR dataset, which is the Image dataset, I find the it needs an algorithm like Convolution Neural Network that is powerful for computing

I have tried applying K-fold stratified shuffle shift with value of 5 and plotted the graph with PCA and mesh grid analysis which clearly shows Basic Supervised learning algorithm is not efficient in getting the predictions compared to IRIS DATSET that clearly demonstrate that as dataset changes the computation algorithm also changes

Step 1:

I have tried to apply the all supervised algorithm on the RAW SAMPLES of data (ice Not applied PCA reduction but I see the following challenges of computation of dataset and running the complete algorithms on dataset ,Doing this we understand the challenges of dimensionality reduction that is needed for preprocessing of data before applying to any machine learning Algorithms

Note: **Total No of samples =60000**

WITHOUT PCA (10000 samples)

The computation time take to compute the above algorithm is

**-- 2869.274126291275 seconds ---**

------------------LDA-K-FOLD Mean Accuracy------------------------------

0.2454

------------------QDA-K-FOLD Mean Accuracy------------------------------

0.1029

------------------NB-K-FOLD Mean Accuracy--------------------------------

0.2853

------------------KNN- 1-K-FOLD Mean Accuracy----------------------------

0.281

------------------KNN- 5-K-FOLD Mean Accuracy----------------------------

0.2831

------------------KNN- 10-K-FOLD Mean Accuracy----------------------------

0.2859

------------------Random forest-K-FOLD Mean Accuracy--------------------

0.3096

------------------Decision Tree -K-FOLD Mean Accuracy--------------------

0.2304

------------------SVM -K-FOLD Mean Accuracy---------------------------------

0.103

------------------LDA-K-FOLD Mean Error----------------------------

0.7546

------------------QDA-K-FOLD Mean Error----------------------------

0.8971

------------------NB-K-FOLD Mean Error----------------------------

0.7147

------------------KNN-K-FOLD Mean Error----------------------------

0.719

------------------KNN-5-K-FOLD Mean Error----------------------------

0.7169

------------------KNN-10-K-FOLD Mean Error----------------------------

0.7141

------------------RF -K-FOLD Mean Error----------------------------

0.6904

------------------DT -K-FOLD Mean Error----------------------------

0.7696

------------------SVM-K-FOLD Mean Error----------------------------

0.897

**WITH PCA (10000 SAMPLES)**

**Hence applied PCA that reduces the Computation time retaining the features, however there are differences in the accuracy for first 10,000 samples as below**

The computation time is reduced to

**---75.55178427696228 seconds---**

------------------LDA-K-FOLD Mean Accuracy------------------------------

0.21409999999999998

------------------QDA-K-FOLD Mean Accuracy------------------------------

0.2135

------------------NB-K-FOLD Mean Accuracy--------------------------------

0.2109

------------------KNN- 1-K-FOLD Mean Accuracy----------------------------

0.1479

------------------KNN- 5-K-FOLD Mean Accuracy----------------------------

0.1628

------------------KNN- 10-K-FOLD Mean Accuracy----------------------------

0.1612

------------------Random forest-K-FOLD Mean Accuracy-------------------------

0.1508

------------------Decision Tree -K-FOLD Mean Accuracy---------------------------

0.1418

------------------SVM -K-FOLD Mean Accuracy--------------------------------

0.1032

------------------LDA-K-FOLD Mean Error----------------------------

0.7859

------------------QDA-K-FOLD Mean Error----------------------------

0.7865

------------------NB-K-FOLD Mean Error----------------------------

0.7891

------------------KNN-K-FOLD Mean Error----------------------------

0.8521

------------------KNN-5-K-FOLD Mean Error----------------------------

0.8371999999999999

------------------KNN-10-K-FOLD Mean Error----------------------------

0.8388

------------------RF -K-FOLD Mean Error----------------------------

0.8492

------------------DT -K-FOLD Mean Error----------------------------

0.8582

------------------SVM-K-FOLD Mean Error----------------------------

0.8968

**From the results (with and without PCA) reduction we see that PCA helps in computation and dimensionality reduction How ever there is a compromise in the accuracy that results in 50 % drop from 30% to 15 % in Random forest classifiers and fair reduction in rest of the algorithm**

Applying PCA +MESH GRID (Boundary analysis compared to IRIS DATASET)

This helps to compute on dataset without losing features:

The following Figures represent the plot for CIFAR DATASET with PCA and MeshGrid with one KFold:

As the DATASET is too large for computation, I have tried using 10000 samples and calculated the accuracy error due to computational issues which is as follows

**------------------LDA-K-FOLD Mean Accuracy------------------------------**

**0.2140**

**------------------QDA-K-FOLD Mean Accuracy------------------------------**

**0.2135**

**------------------NB-K-FOLD Mean Accuracy--------------------------------**

**0.2109**

**------------------KNN- 1-K-FOLD Mean Accuracy----------------------------**

**0.1524**

**------------------KNN- 5-K-FOLD Mean Accuracy----------------------------**

**0.1591**

**------------------KNN- 10-K-FOLD Mean Accuracy----------------------------**

**0.1653**

**------------------Random forest-K-FOLD Mean Accuracy-------------------**

**0.1509**

**------------------Decision Tree -K-FOLD Mean Accuracy--------------------**

**0.1418**

**------------------SVM -K-FOLD Mean Accuracy--------------------------------**

**0.2168**

**------------------LDA-K-FOLD Mean Error-----------------------------**

**0.7859**

**------------------QDA-K-FOLD Mean Error----------------------------**

**0.7865**

**------------------NB-K-FOLD Mean Error------------------------------**

**0.7891**

**------------------KNN-K-FOLD Mean Error----------------------------**

**0.8476**

**------------------KNN-5-K-FOLD Mean Error-------------------------**

**0.8409**

**------------------KNN-10-K-FOLD Mean Error------------------------**

**0.8347**

**------------------RF -K-FOLD Mean Error------------------------------**

**0.8491**

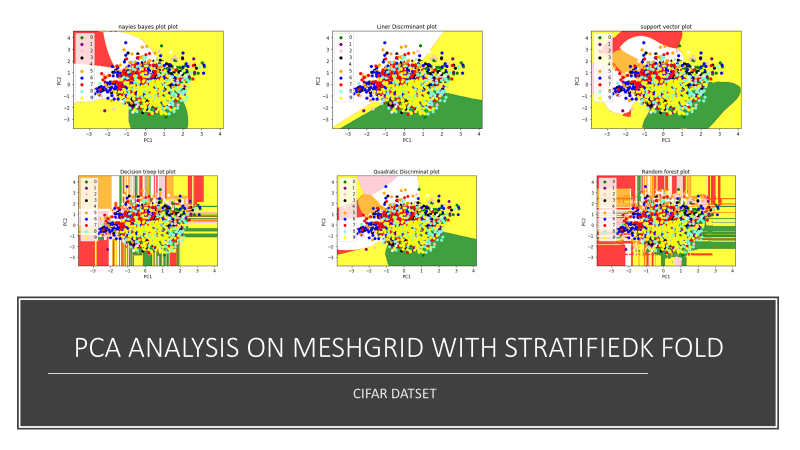
**------------------DT -K-FOLD Mean Error------------------------------**

**0.8582**

**------------------SVM-K-FOLD Mean Error----------------------------**

**0.7832**

**This clearly explains that we are not able to achieve accuracy with the above algorithms**



**Note: Complete plots included in folder CIFAR+MESH+GRID+PLOT\_10000SAMPLES**

WITH PCA (60,0000) samples

The computation time is

**2297.1568989753723 seconds**

------------------LDA-K-FOLD Mean Accuracy------------------------------

0.20608333333333334

------------------QDA-K-FOLD Mean Accuracy------------------------------

0.20801666666666666

------------------NB-K-FOLD Mean Accuracy--------------------------------

0.20338333333333333

------------------KNN- 1-K-FOLD Mean Accuracy----------------------------

0.1539

------------------KNN- 5-K-FOLD Mean Accuracy----------------------------

0.15456666666666666

------------------KNN- 10-K-FOLD Mean Accuracy----------------------------

0.15818333333333334

------------------Random forest-K-FOLD Mean Accuracy--------------------

0.15183333333333332

------------------Decision Tree -K-FOLD Mean Accuracy---------------------

0.14243333333333333

------------------SVM -K-FOLD Mean Accuracy--------------------------------

0.10268333333333333

------------------LDA-K-FOLD Mean Error--------------------------------------

0.7939166666666667

------------------QDA-K-FOLD Mean Error-------------------------------------

0.7919833333333334

------------------NB-K-FOLD Mean Error---------------------------------------

0.7966166666666666

------------------KNN-K-FOLD Mean Error-------------------------------------

0.8461

------------------KNN-5-K-FOLD Mean Error----------------------------------

0.8454333333333334

------------------KNN-10-K-FOLD Mean Error--------------------------------

0.8418166666666667

------------------RF -K-FOLD Mean Error--------------------------------------

0.8481666666666666

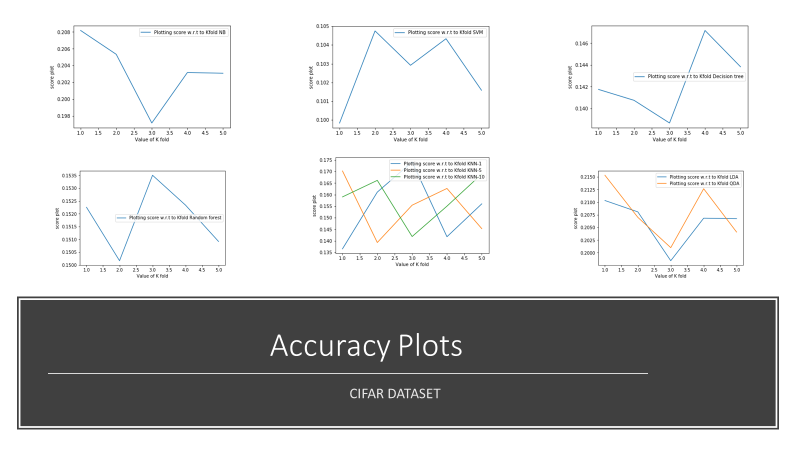
------------------DT -K-FOLD Mean Error--------------------------------------

0.8575666666666666

------------------SVM-K-FOLD Mean Error------------------------------------

0.8973166666666667

Accuracy Plot



Error Plot

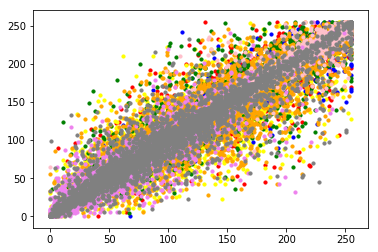


**Note: Complete plots included in folder CIFAR+PCA+60000\_SAMPLES**

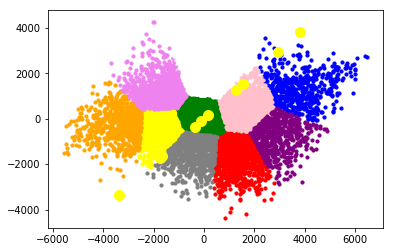
K Means Clustering (Unsupervised learning Algorithm)

I have tried applying the K means clustering to CIFAR Dataset I find that as the Number of classes and Data sample are High

Data represented before applying Kmeans algorithm

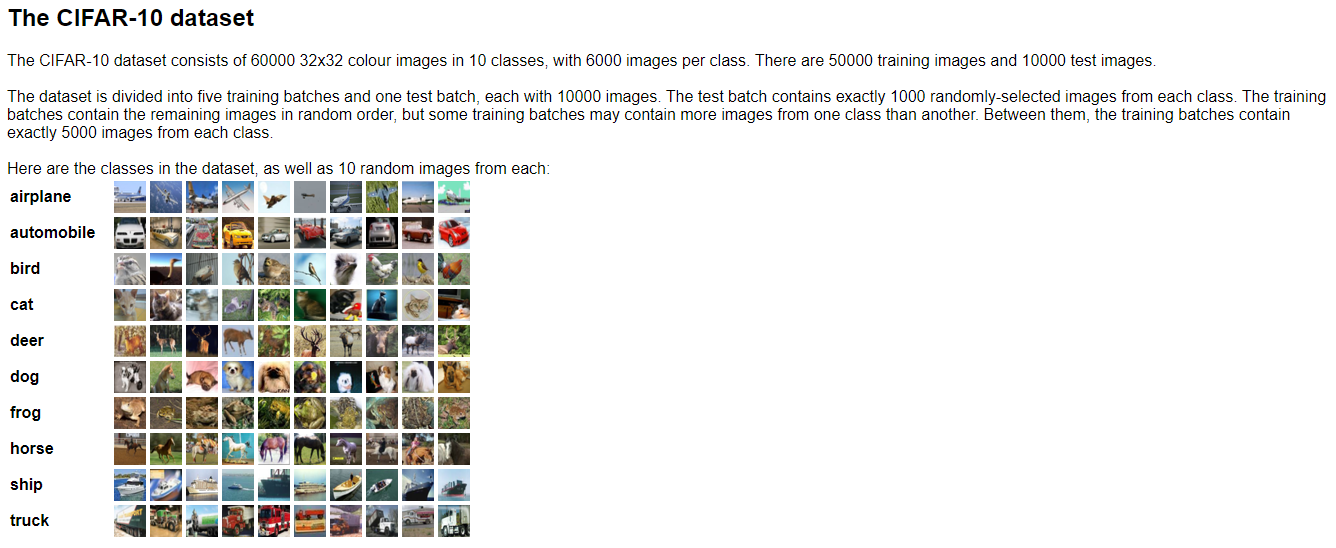


I have applied Kmeans algorithms that represents the clusters as below, However, there is a mismatch in the centroids Due to dataset being high and not able to predict the classes properly for all 60000 samples



CNN FOR CIFAR

<https://www.cs.toronto.edu/~kriz/cifar.html>



While building the CNN model I have taken complete dataset 50000 samples for Training and 10000 samples for testing

The following is the description of the model that is being taken and the accuracy

Just the reference Not taken any code from these links

#Citations: https://keras.io/initializers/

#https://appliedmachinelearning.blog/2018/03/24/achieving-90-accuracy-in-object-recognition-task-on-cifar-10-dataset-with-keras-convolutional-neural-networks/

https://blog.goodaudience.com/visualizing-various-filter-initializers-in-keras-ca14c996db22

Experiment 1

Wanted to increase the depth of the layer and also the no of filter size in each layer with multiple of 32

model.add(BatchNormalization())

model.add(Conv2D(32,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(Conv2D(64,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(Conv2D(128,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(Conv2D(64,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(Conv2D(64,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(MaxPool2D(pool\_size=(2,2)))

model.add(Conv2D(64,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(Flatten())

model.add(Dense(units=10, activation="softmax"))

model.compile(optimizer='adam',loss='categorical\_crossentropy',metrics=["accuracy"])

I have applied 7 layers of convolution2d which each is a combination of 32 and 64 and 128, 1 Batch Normalization

and the activation function being used here is relu that introduces Nonlinearity to system compared to other function like sigmoid

and tahn the input dimension for the layer is 32,32,3 as this is RGB image and filters size 3\*3 and output activation function being used is

softmax that is used for multiple categorical labels

Instructions for updating:

Use tf.cast instead.

Epoch 1/15

50000/50000 [==============================] - 915s 18ms/step - loss: 1.4699 - accuracy: 0.4694

Epoch 2/15

50000/50000 [==============================] - 884s 18ms/step - loss: 1.1330 - accuracy: 0.5940

Epoch 3/15

50000/50000 [==============================] - 916s 18ms/step - loss: 0.9847 - accuracy: 0.6486

Epoch 4/15

50000/50000 [==============================] - 952s 19ms/step - loss: 0.8690 - accuracy: 0.6901

Epoch 5/15

50000/50000 [==============================] - 955s 19ms/step - loss: 0.7701 - accuracy: 0.7235

Epoch 6/15

50000/50000 [==============================] - 877s 18ms/step - loss: 0.6852 - accuracy: 0.7555

Epoch 7/15

50000/50000 [==============================] - 945s 19ms/step - loss: 0.5977 - accuracy: 0.7872

Epoch 8/15

50000/50000 [==============================] - 942s 19ms/step - loss: 0.5255 - accuracy: 0.8138

Epoch 9/15

50000/50000 [==============================] - 946s 19ms/step - loss: 0.4632 - accuracy: 0.8325

Epoch 10/15

50000/50000 [==============================] - 924s 18ms/step - loss: 0.4037 - accuracy: 0.8560

Epoch 11/15

50000/50000 [==============================] - 828s 17ms/step - loss: 0.3652 - accuracy: 0.8682

Epoch 12/15

50000/50000 [==============================] - 2799s 56ms/step - loss: 0.3296 - accuracy: 0.8825

Epoch 13/15

50000/50000 [==============================] - 856s 17ms/step - loss: 0.2959 - accuracy: 0.8937

Epoch 14/15

50000/50000 [==============================] - 854s 17ms/step - loss: 0.2836 - accuracy: 0.8985

Epoch 15/15

50000/50000 [==============================] - 827s 17ms/step - loss: 0.2709 - accuracy: 0.9030

10000/10000 [==============================] - 54s 5ms/step

**CNN ACCURACY**

**0.2493000030517578**

**CNN LOSS**

**10.497021226882934**

**CNN ERROR**

**0.7506999969482422**

Improvised model

Experiment 2 CNN

Train samples :50000

Test samples :10000

model = Sequential()

model.add(BatchNormalization())

model.add(Conv2D(32,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(Conv2D(64,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(Conv2D(64,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(Conv2D(64,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(MaxPool2D(pool\_size=(2,2)))

model.add(Conv2D(64,kernel\_size=(3,3),input\_shape=(32, 32, 3),activation='relu'))

model.add(Flatten())

model.add(Dense(units=10, activation="softmax"))

model.compile(optimizer='adam',loss='categorical\_crossentropy',metrics=["accuracy"])

I have applied 6 layers of convolution2d which each is a combination of 32 and 64 1 Batch Normalization

and the activation function being used here is relu that introduces Nonlinearity to system compared to other function like sigmoid

and tahn the input dimension for the layer is 32,32,3 as this is RGB image and filters size 3\*3 and output activation function being used is

softmax that is used for multiple categorical labels

2019-12-02 10:47:24.237526: I TensorFlow/core/platform/cpu\_feature\_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2

Epoch 1/10

50000/50000 [==============================] - 161s 3ms/step - loss: 1.4662 - accuracy: 0.4791

Epoch 2/10

50000/50000 [==============================] - 163s 3ms/step - loss: 1.0925 - accuracy: 0.6181

Epoch 3/10

50000/50000 [==============================] - 162s 3ms/step - loss: 0.9231 - accuracy: 0.6768

Epoch 4/10

50000/50000 [==============================] - 162s 3ms/step - loss: 0.7983 - accuracy: 0.7225

Epoch 5/10

50000/50000 [==============================] - 161s 3ms/step - loss: 0.6876 - accuracy: 0.7609

Epoch 6/10

50000/50000 [==============================] - 161s 3ms/step - loss: 0.5871 - accuracy: 0.7940

Epoch 7/10

50000/50000 [==============================] - 162s 3ms/step - loss: 0.4988 - accuracy: 0.8246

Epoch 8/10

50000/50000 [==============================] - 161s 3ms/step - loss: 0.4188 - accuracy: 0.8518

Epoch 9/10

50000/50000 [==============================] - 161s 3ms/step - loss: 0.3577 - accuracy: 0.8737

Epoch 10/10

50000/50000 [==============================] - 161s 3ms/step - loss: 0.3120 - accuracy: 0.8903

10000/10000 [==============================] - 8s 838us/step

**CNN ACCURACY**

**0.6305000185966492**

**CNN LOSS**

**1.5805373914718628**

**CNN ERROR**

**0.36949998140335083**

**Acknowledgments:**

I would Like to Thank Prof, Dr. Jacob Levman for providing complete guidance throughout the course and helping us to understand the concepts and have overall knowledge on Machine learning concepts

Also, Derek Berger for providing support throughout the assignments for python Implementations