Project Report

Course: IE 7615 Neural Networks & Deep Learning Fall 2022

Professor: Dr. Braun Jerome

CIFAR - 10

Project Title: To classify the various categories(or classes)

Author:

Ashish Sharma, sharma.ashis@northeastern.edu

1. Abstract

In this case study, I will use Python language to classify the various categories (or classes) present in the given dataset having been converted into integer values. The main objective of the dataset is to classify the 10 classes having RGB values of size 32x32. The CIFAR – 10 is a collection of images which are commonly used for training various machine learning or computer vision algorithms.

2. Introduction

2.1 Overview of Project

The dataset which is CIFAR 10 is a subset of 80 million tiny images dataset, all the attributes or data is in the form of an array of RGB values of the given 60000 images which have the image size of 32x32. This data has various classes such as Airplane, Truck, Frogs, Automobile, Bird, Cat, Deer, Dog, Horse, Ship. I need to classify these classes using the image recognition techniques and using various models which can help me achieve the target and correctly classify the images among various classes. This data is a standard dataset which is often used in computer vision and deep learning.

2.2 Motivation

The motivation behind working on this dataset is to deep dive into the image classification techniques and be able to differentiate between the images through various machine learning or deep learning models. This data will help me achieve the ability to explore the image data and to classify images with the help of deep learning and using the neural networks.

2.3 Approach

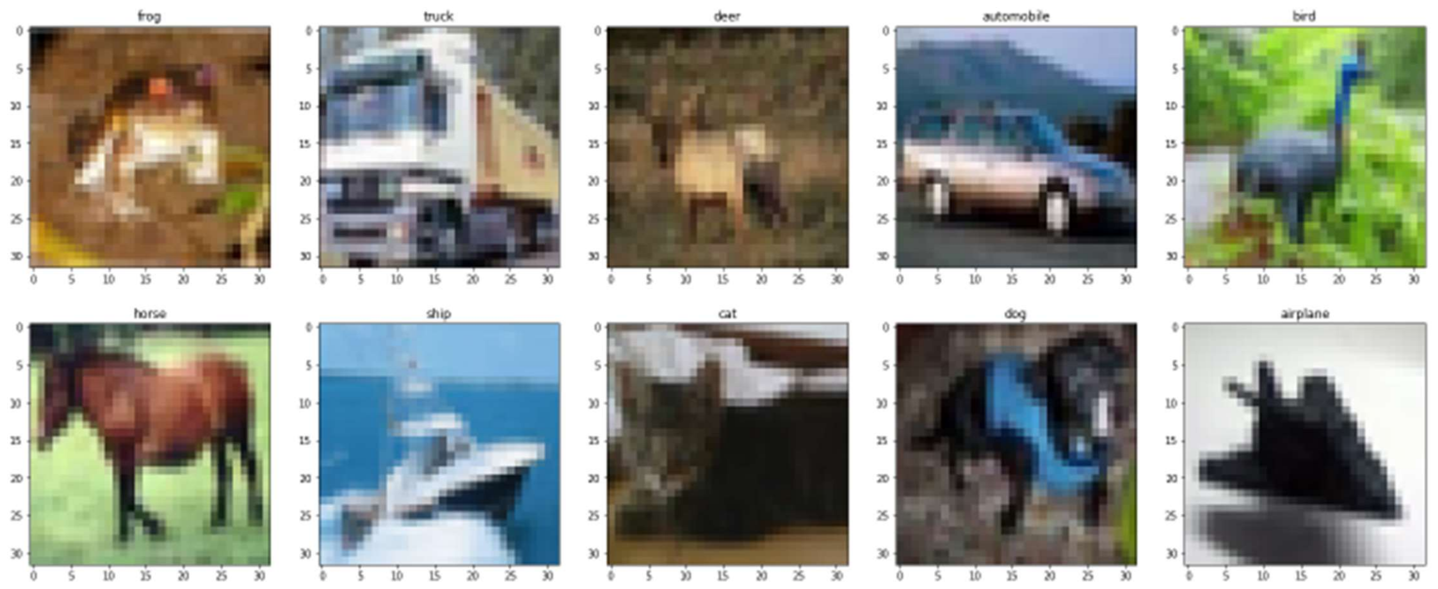
This project will be based on performing exploratory data analysis on the dataset and getting ideas which then can be used to apply various deep learning models to classify the various categories and improvising the results using various hyperparameter tuning. I will estimate the performance of the model using various techniques and explore the various ways in which the model could be improvised. I will also be focusing on decreasing the error rate in order to get best results with high F1, recall and precision scores. As per different research done by the researchers, I was able to gather that the Convolutional Neural Networks would be the best to implement the model on.

2.4 About the Dataset

This data was extracted from the University of Toronto's website. This dataset is a subset of the 80 million tiny images dataset. This dataset is popularly called the "CIFAR - 10" data set. This was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The dataset provides 60000 RGB values of size 32x32 in numerical data types. Every class consists of 6000 images and the following dataset is split into 50000 training images and 10000 testing images for us to work on.

Data: Downloaded from (https://www.cs.toronto.edu/~kriz/cifar.html)
This dataset is widely publicly available and is an unrestricted dataset.



3. Background

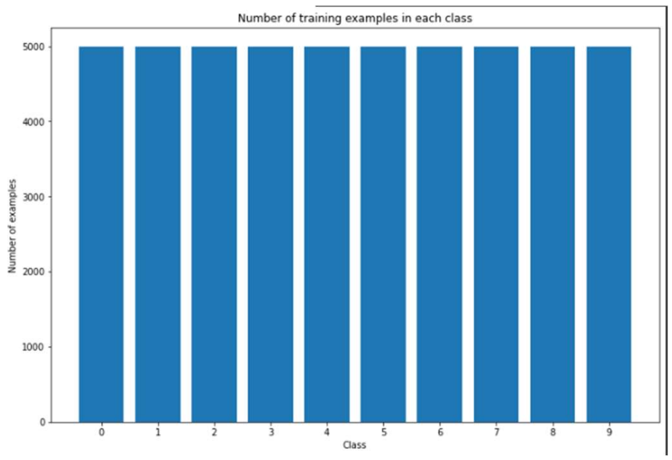
Various research papers have been written which claims to have achieved state of the art results on this dataset which were mainly focused on reducing the error percentage, as the papers are not standardized based on their techniques used during the pre-processing phase, so there might be a case that the paper's error percentage could higher than the other state of the art results but as they have been published, they are valid. These are various results which have been mentioned in various papers. These results have been taken from the Wikipedia page of CIFAR -10.

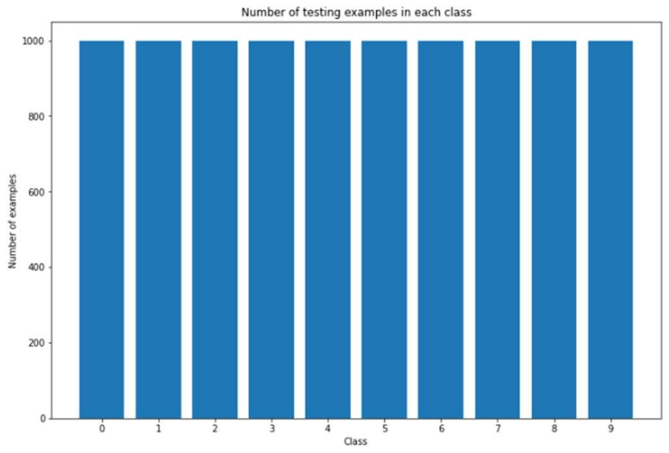
Paper title	Error rate (%)	Publication date
Convolutional Deep Belief Networks on CIFAR-10[6]	21.1	August, 2010
Maxout Networks[7]	9.38	February 13, 2013
Wide Residual Networks[8]	4.0	May 23, 2016
Neural Architecture Search with Reinforcement Learning[9]	3.65	November 4, 2016
Fractional Max-Pooling[10]	3.47	December 18, 2014
Densely Connected Convolutional Networks[11]	3.46	August 24, 2016
Shake-Shake regularization[12]	2.86	May 21, 2017
Coupled Ensembles of Neural Networks[13]	2.68	September 18, 2017
ShakeDrop regularization[14]	2.67	Feb 7, 2018
Improved Regularization of Convolutional Neural Networks with Cutout[15]	2.56	Aug 15, 2017
Regularized Evolution for Image Classifier Architecture Search[16]	2.13	Feb 6, 2018
Rethinking Recurrent Neural Networks and other Improvements for Image Classification[17]	1.64	July 31, 2020
AutoAugment: Learning Augmentation Policies from Data[18]	1.48	May 24, 2018
A Survey on Neural Architecture Search[19]	1.33	May 4, 2019

4. Approach

4.1 Data Analysis

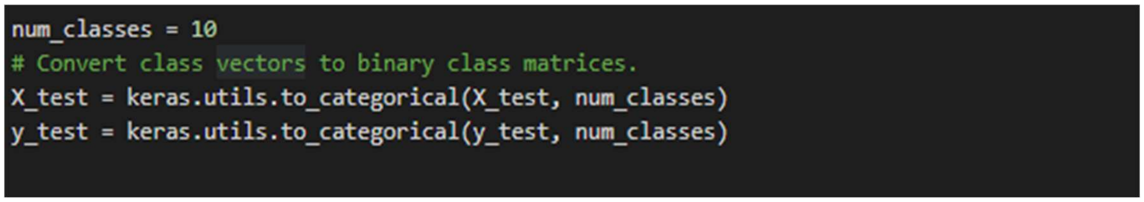
Starting with the analysis, first the data is loaded onto the notebook and named into training and testing data. In the analysis I checked the data count of every class in both training and testing dataset.





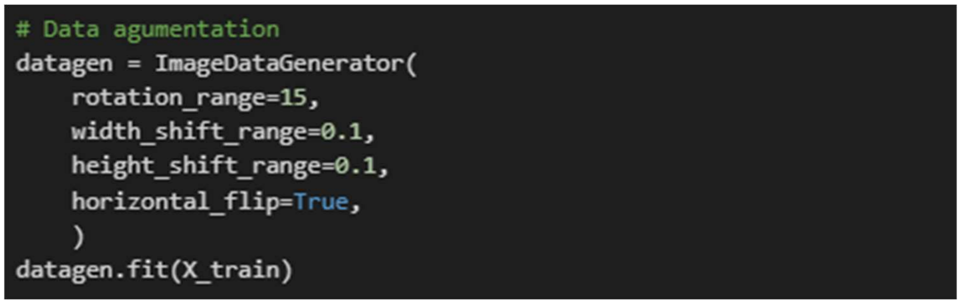
Here classes 0 to 9 represent Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck respectively. So we confirm using the plots that they have perfectly balanced classes both in training as well as testing datasets.

Further, we move on to one hot encoding as the classes are in the character data type and we need it to convert into integer data type for applying it into the models. With the help of this encoding, the number may be converted into a binary vector with 10 elements and an index for the class value of 1. The to_categorical() utility function allows us to accomplish this.

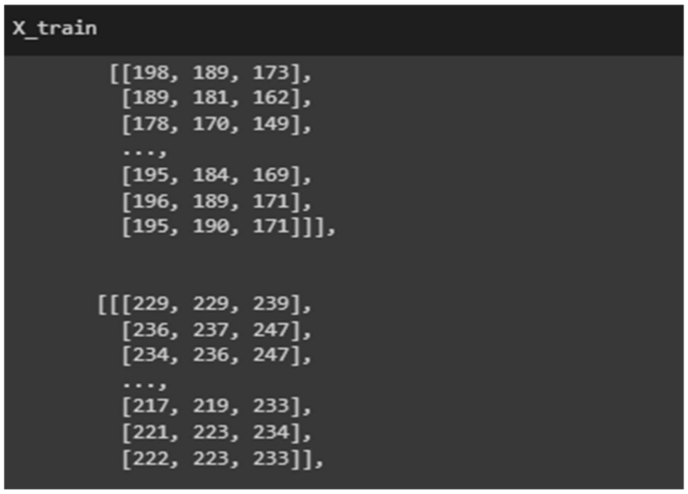


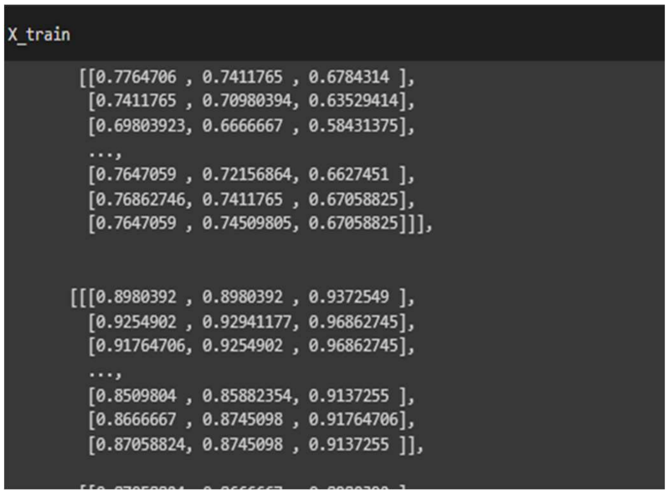
```
num_classes = 10
# Convert class vectors to binary class matrices.
X_test = keras.utils.to_categorical(X_test, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

Then we perform data augmentation which makes copies of examples in the dataset with small modifications. This has a regularizing effect which can help in performance and we have various augmentations to perform such as rotating the image, shifting the width or height of the image, or flipping the image.


```
# Data agumentation
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    )
datagen.fit(X_train)
```

It is known that each image has been assigned to a value from range 0 to 255, as we are not aware of the best way to scale the values, but we know scaling is needed, so to achieve that the data or values are normalized in the range from 0 to 1. This approach is done by first converting the integer value to float value and then dividing it by 255. Now the data is ready to be implemented on different models.


```
X_train
        [[198, 189, 173],
         [189, 181, 162],
         [178, 170, 149],
         ...,
         [195, 184, 169],
         [196, 189, 171],
         [195, 190, 171]]],


       [[[229, 229, 239],
         [236, 237, 247],
         [234, 236, 247],
         ...,
         [217, 219, 233],
         [221, 223, 234],
         [222, 223, 233]],
```


```
X_train
        [[0.7764706 , 0.7411765 , 0.6784314 ],
         [0.7411765 , 0.70980394, 0.63529414],
         [0.69803923, 0.6666667 , 0.58431375],
         ...,
         [0.7647059 , 0.72156864, 0.6627451 ],
         [0.76862746, 0.7411765 , 0.67058825],
         [0.7647059 , 0.74509805, 0.67058825]]],


       [[[0.8980392 , 0.8980392 , 0.9372549 ],
         [0.9254902 , 0.92941177, 0.96862745],
         [0.91764706, 0.9254902 , 0.96862745],
         ...,
         [0.8509804 , 0.85882354, 0.9137255 ],
         [0.8666667 , 0.8745098 , 0.91764706],
         [0.87058824, 0.8745098 , 0.9137255 ]],
```

4.2 Modeling

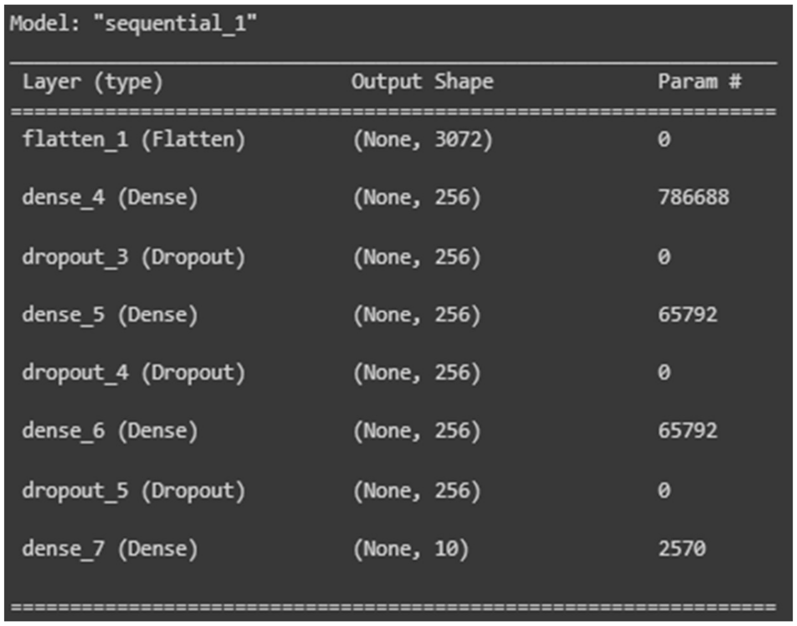
Modeling was done using various parameters and also using the neural networks with or without the Convolutional Layers. I tried to work with every perspective possible right from basic network to the best possible network. The models which I worked on were:

1. Basic Neural Network
2. Simple Model with Convolutional Layers
3. Complex Model with Increasing dropout, Batch Normalization
4. Complex Model with batch normalization, Stochastic Gradient Descent, L2 norm Regularizers

5. Results

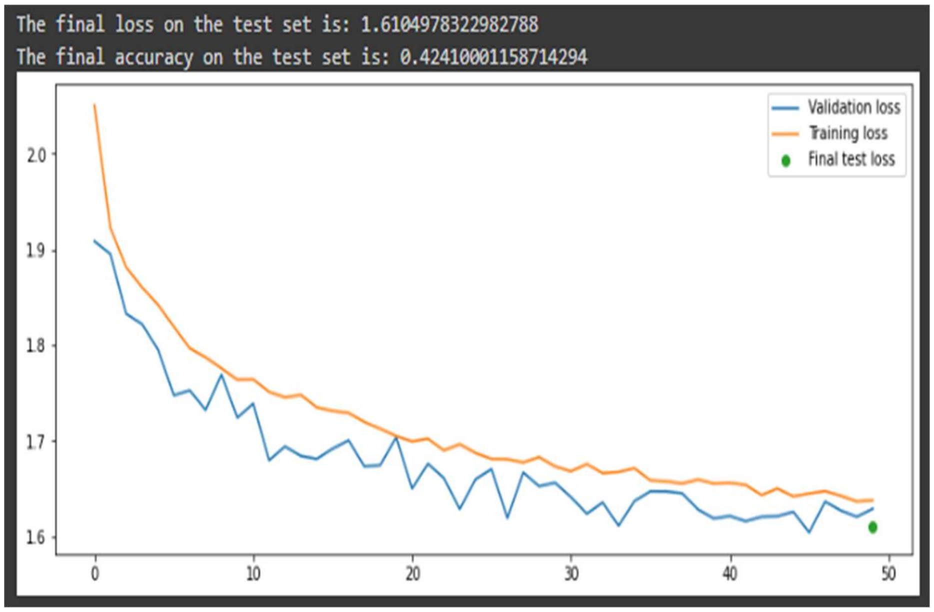
5.1 Basic Neural Network

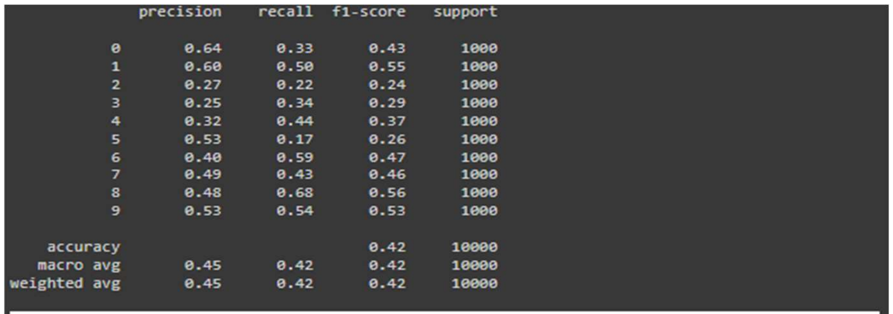
In this model I tried to implement a network without the convolution layer, as expected there were no great results and the accuracy score turned out to be 42.41%. SoftMax Activation function has been used as it works very well for multi class classification. Also, Adam optimizer is used for better performance in compiling the model. It can be seen that the model is a good fit when it ran for 50 epochs, but the test loss is still in high areas. For that, I will add convolutional layers to the model to achieve better performance. Some of the classes are struggling with the model as they are complex in nature. Also, from ROC - AUC we can interpret that the model is not classifying the categories very well. The model is clearly not complex enough to extract the features from the images.



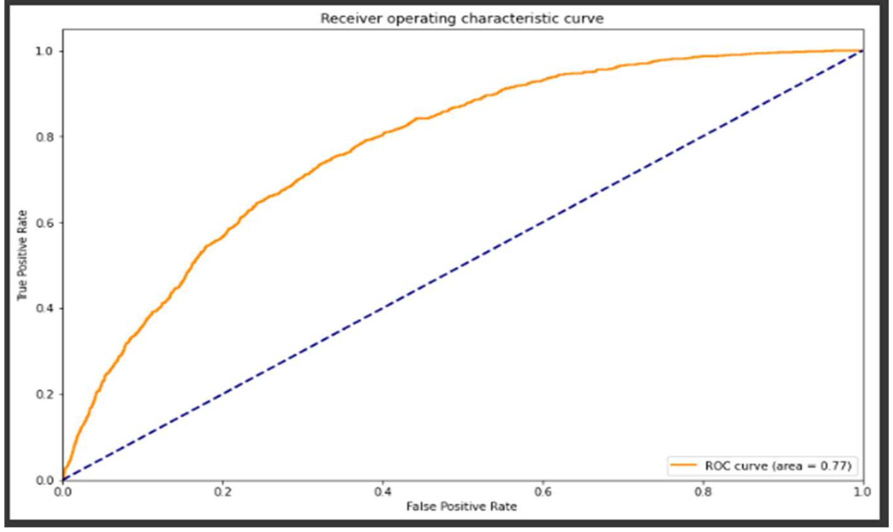
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 3072)              0

 dense_4 (Dense)             (None, 256)               786688

 dropout_3 (Dropout)         (None, 256)               0

 dense_5 (Dense)             (None, 256)               65792

 dropout_4 (Dropout)         (None, 256)               0

 dense_6 (Dense)             (None, 256)               65792

 dropout_5 (Dropout)         (None, 256)               0

 dense_7 (Dense)             (None, 10)                2570

=================================================================
```

```
The final loss on the test set is: 1.6104978322982788
The final accuracy on the test set is: 0.42410001158714294
```



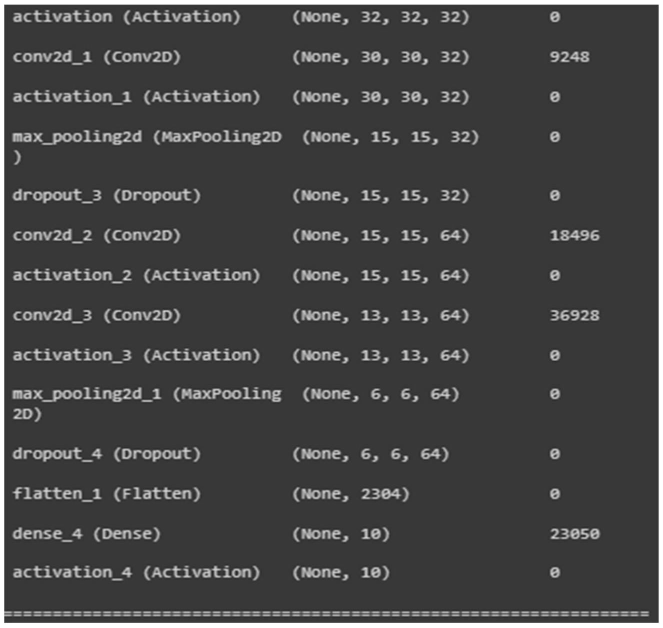


```
              precision    recall  f1-score   support

           0       0.64      0.33      0.43      1000
           1       0.60      0.50      0.55      1000
           2       0.27      0.22      0.24      1000
           3       0.25      0.34      0.29      1000
           4       0.32      0.44      0.37      1000
           5       0.53      0.17      0.26      1000
           6       0.40      0.59      0.47      1000
           7       0.49      0.43      0.46      1000
           8       0.48      0.68      0.56      1000
           9       0.53      0.54      0.53      1000

    accuracy                           0.42     10000
   macro avg       0.45      0.42      0.42     10000
weighted avg       0.45      0.42      0.42     10000
```

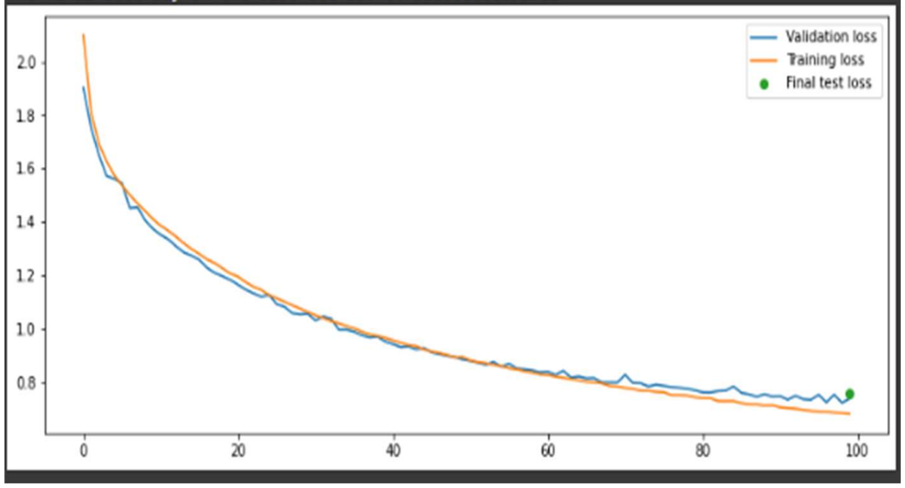


5.2 Simple Model with Convolutional Layers

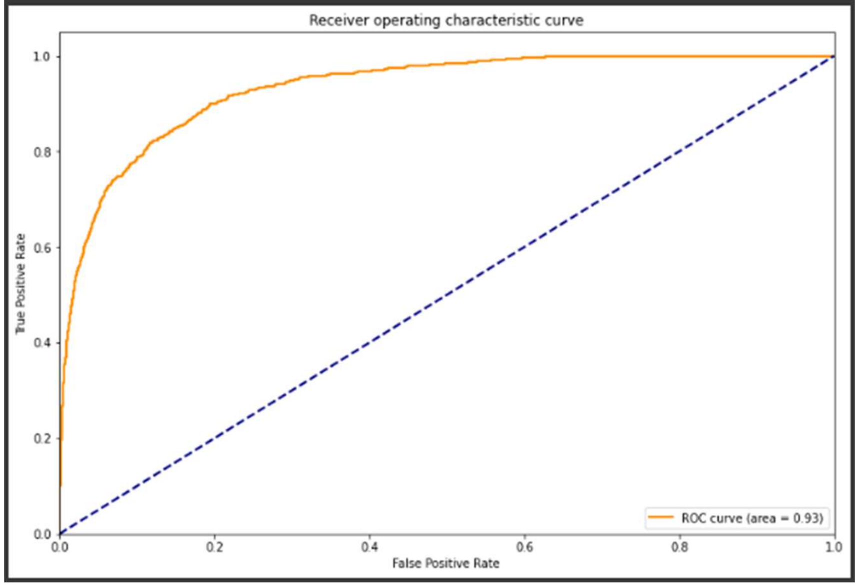
This model was the combination of the basic model with convolutional layers, also the learning rate was changed to 0.0001 and with the RMS optimizer. There are 4 convolution layers using the 3x3 max pooling. With the help of the following parameters 73.82% accuracy was achieved with significant reduction in test loss, so there is definitely a huge improvement. The model is a good fit even running for 100 epochs which means there are less chances of overfitting and underfitting.



```
activation (Activation)     (None, 32, 32, 32)        0

conv2d_1 (Conv2D)           (None, 30, 30, 32)        9248

activation_1 (Activation)   (None, 30, 30, 32)        0

max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
)

dropout_3 (Dropout)         (None, 15, 15, 32)        0

conv2d_2 (Conv2D)           (None, 15, 15, 64)        18496

activation_2 (Activation)   (None, 15, 15, 64)        0

conv2d_3 (Conv2D)           (None, 13, 13, 64)        36928

activation_3 (Activation)   (None, 13, 13, 64)        0

max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
2D)

dropout_4 (Dropout)         (None, 6, 6, 64)          0

flatten_1 (Flatten)         (None, 2304)              0

dense_4 (Dense)             (None, 10)                23050

activation_4 (Activation)   (None, 10)                0

=================================================================
The final loss on the test set is: 0.7549781799316406
The final accuracy on the test set is: 0.7382000088691711
```



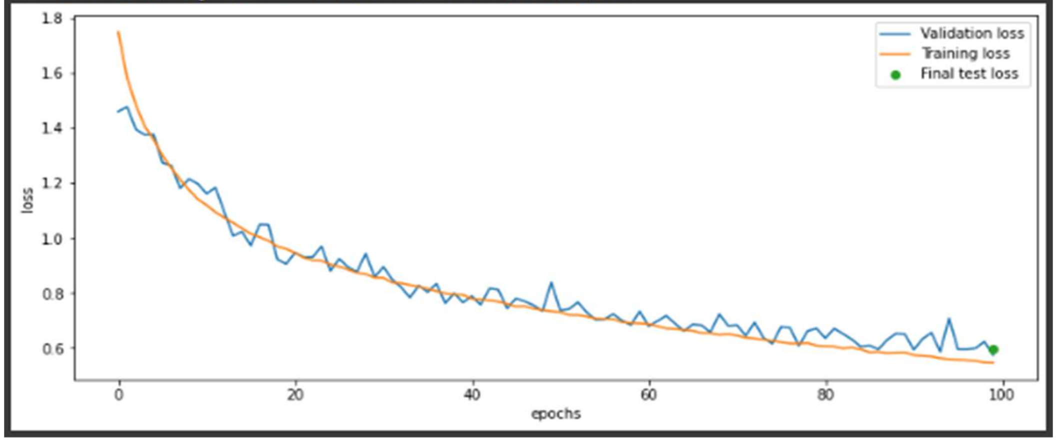
```
              precision    recall  f1-score   support

           0       0.74      0.81      0.77      1000
           1       0.87      0.84      0.85      1000
           2       0.63      0.63      0.63      1000
           3       0.61      0.47      0.53      1000
           4       0.64      0.75      0.69      1000
           5       0.64      0.67      0.65      1000
           6       0.72      0.86      0.79      1000
           7       0.84      0.74      0.79      1000
           8       0.89      0.79      0.84      1000
           9       0.83      0.82      0.82      1000

    accuracy                           0.74     10000
   macro avg       0.74      0.74      0.74     10000
weighted avg       0.74      0.74      0.74     10000
```



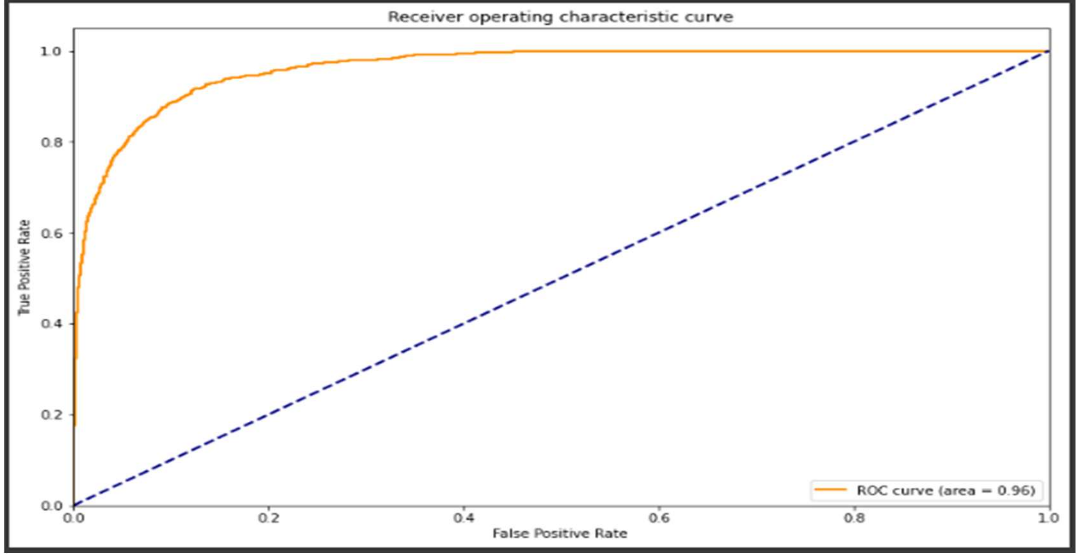
5.3 Complex Model with Increasing dropout, Batch Normalization

In this model dropout was increased and batch normalization was introduced in order to improve the model accuracy and performance. There is a significant increase from 74 to 79.5% also there is reduction in loss from 0.75 to 0.59. All classes are performing pretty well except a couple of classes due to their complexity. The ROC curve is pretty strong which states that our model is working well and classifying the categories properly.

```
The final loss on the test set is: 0.5948849320411682
The final accuracy on the test set is: 0.7939000129699707
```

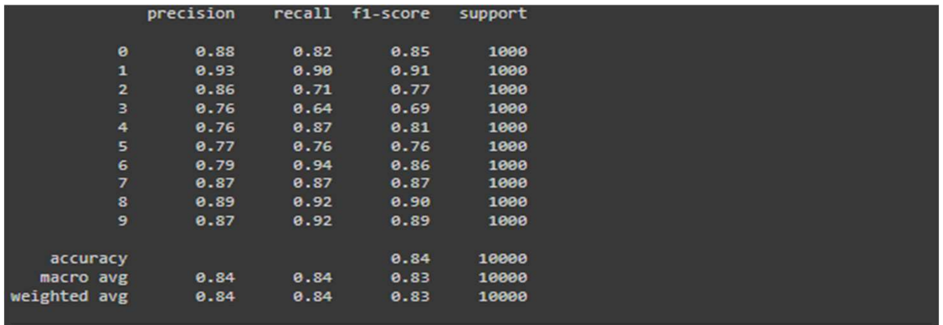


```
              precision    recall  f1-score   support

           0       0.80      0.81      0.80      1000
           1       0.93      0.83      0.88      1000
           2       0.79      0.64      0.70      1000
           3       0.64      0.64      0.64      1000
           4       0.73      0.82      0.77      1000
           5       0.74      0.70      0.72      1000
           6       0.82      0.88      0.85      1000
           7       0.90      0.79      0.84      1000
           8       0.80      0.92      0.86      1000
           9       0.82      0.90      0.86      1000

    accuracy                           0.79     10000
   macro avg       0.80      0.79      0.79     10000
weighted avg       0.80      0.79      0.79     10000
```

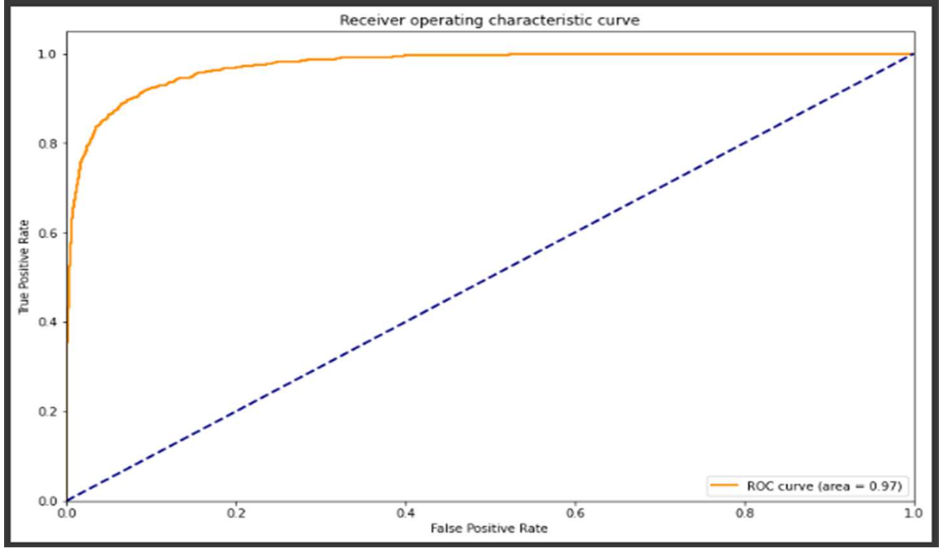


5.4 Complex Model with batch normalization, Stochastic Gradient Descent, L2 norm Regularizers

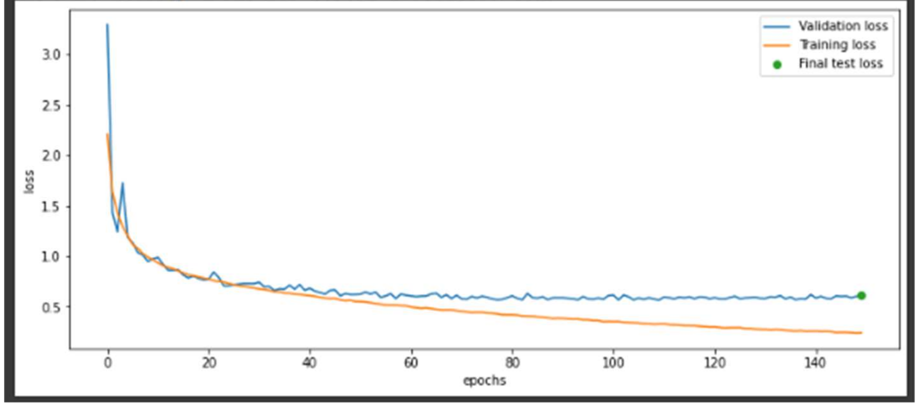
This model is the CNN network with batch normalization, SGD, weight decay and regularizers to improve the model further, to achieve 84% accuracy with low losses to 0.57 and without any overfitting. There is an increase in accuracy from 79.5 to 84%. Data augmentation also helped in improving the model. Batch Normalization helped in stabilizing the learning and increasing the rate of learning process. Here ‘elu’ function is used which is an improvement of ‘ReLu’ function. Also, L2 norm regularizers were used to prevent overfitting in the model and adjusting the loss function accordingly and hence increase the accuracy. The ROC - AUC curve is also very promising which suggests that the model is correctly classifying the images into categories.

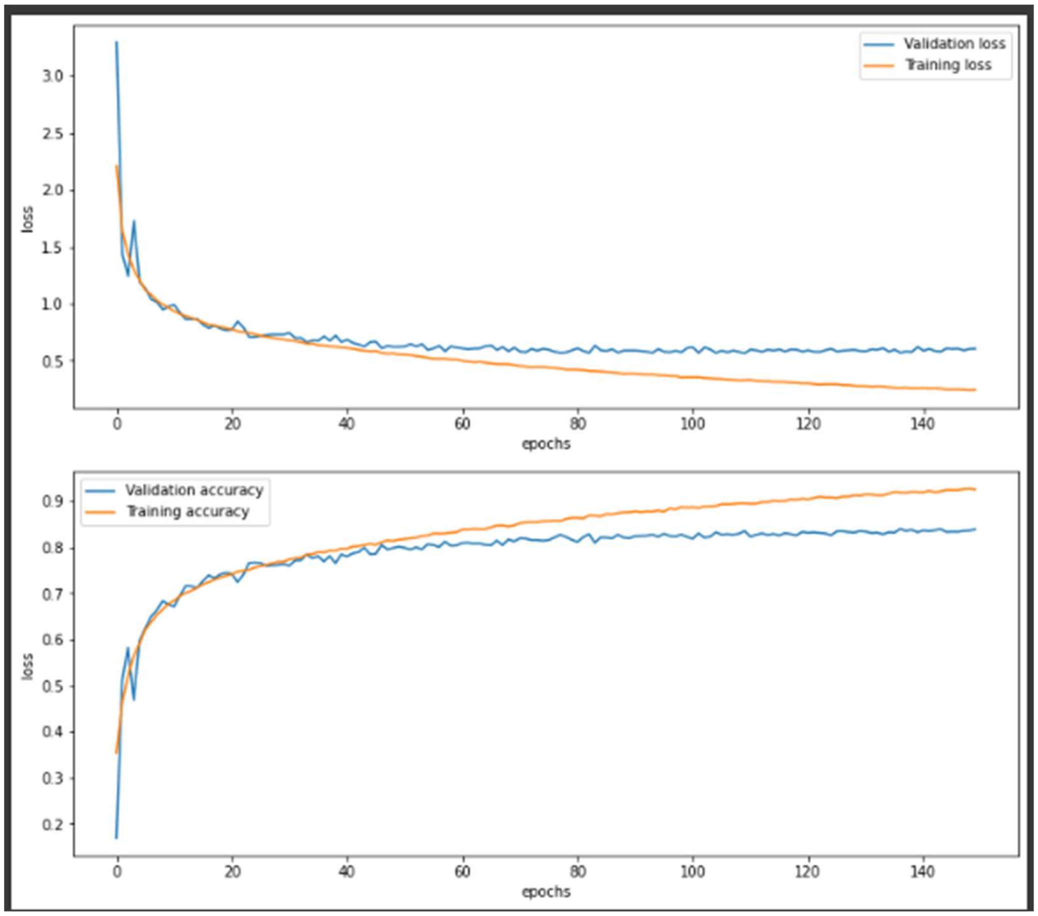


```
              precision    recall  f1-score   support

           0       0.88      0.82      0.85      1000
           1       0.93      0.90      0.91      1000
           2       0.86      0.71      0.77      1000
           3       0.76      0.64      0.69      1000
           4       0.76      0.87      0.81      1000
           5       0.77      0.76      0.76      1000
           6       0.79      0.94      0.86      1000
           7       0.87      0.87      0.87      1000
           8       0.89      0.92      0.90      1000
           9       0.87      0.92      0.89      1000

    accuracy                           0.84     10000
   macro avg       0.84      0.84      0.83     10000
weighted avg       0.84      0.84      0.83     10000
```



```
The final accuracy on the test set is: 0.8353000283241272
```





Overall, the best performing model was the CNN with batch normalization, Stochastic Gradient Descent, Weight Decay in order to improve the model which gave 84% accuracy with low test loss around 0.57. As working on SGD, the results might slightly differ on every runtime, but the difference will be very small. Model has learnt well, and the rate of learning is pretty controlled as well without any case of overfitting.

There is always room for improvements and in this case, we can further tune the hyperparameters of the learning algorithm such as the learning rate which is the most important hyperparameter. Also, further there can be changes in the learning rate technique such using different types of optimizers like Adam. Such changes might help the model to converge more.

Following are the results and performance of the different models trained and tested.

Model	Accuracy	Test Loss
Basic Neural Network	42.41%	1.61
Simple CNN	73%	0.75
Complex Model with increased dropout, batch normalization	79.5%	0.59
Complex Model with batch normalization, Stochastic Gradient Descent, L2 norm	84%	0.57

6. Conclusions

We went through studying the dataset to implement the classification model and understood the nuances faced with multivariate classifications. There are few conclusions which we can get from our study:

- The data which we had was perfectly balanced when tested for the same, hence there were less signs of overfitting and underfitting.
- CNN is the best performing model for image recognition and classifications.
- Worked on 4 different improving models with varying the parameters increasing the accuracy and performance metrics.
- Overall, the model with data augmentation, hyperparameter tuning and batch normalization was the best model to work with giving the highest accuracy.
- Each algorithm used has been tested with a proper confusion matrix and classification report has been provided establishing the accuracy. The F1 scores are in relation to the precision and accuracy for each algorithm.

7. References

7.1 CIFAR-10 dataset gathered from https://www.cs.toronto.edu/~kriz/cifar.html

7.2 Information about CIFAR -10 https://en.wikipedia.org/wiki/CIFAR-10

7.3 Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.