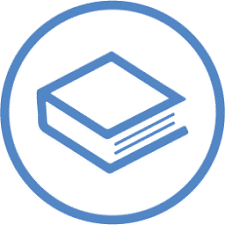
**IMAGE CLASSIFICATION ON CIFAR-10 DATASET**

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE INTERNSHIP CERTIFICATE

UNDER

ECKOVATION



**SUMMER INTERNSHIP REPORT**

MACHINE LEARING & ARTIFICIAL

INTELLIGENCE INTERN

GitHub: https://github.com/Priyaksheemahanta/image-classifier.git

SUBMITTED BY:

PRIYAKSHEE MAHANTA

180720014021

CANDIDATE’S DECLARATION

I, Priyakshee Mahanta, Roll No. 180720014021 of MCA,Jorhat Engineering College, hereby declare that the report titled **“**Image classification on Cifar-10 dataset**”** which is submitted by me to Eckovation in partial fulfillment of the requirement for the award of the internship certificate in machine learning and artificial intelligence,I also declared that the project is based on my original work except for quotations and citations which have been duly acknowledged.

**Place:** Jorhat,Assam Priyakshee Mahanta

**Date:** 21st of September, 2020

ACKNOWLEDGEMENT

I would like to express my deep gratitude to the Almighty, who bestowed ability and strength in me to complete this work. I also thank my parents and friends for their unceasing encouragement and support. They have always guided me to work on right path of life. I owe a profound gratitude to Eckovation for sincere and valuable guidance that helped me to develop a new insight into my project. I play on record, my sincere gratitude to the people of Assam Science And Technology University(ASTU) for providing me with this oppurtunity. Finally, I would like to express my gratitude to all the people who directly, or indirectly, have lent their helping hand in this project.

Date: 21st of September,2020 Priyakshee Mahanta

**CONTENTS**

* **Candidate’s Declaration**
* **Acknowledgement**
* **Table of Contents**
* **CHAPTER 1 INTRODUCTION**
* **CHAPTER 2 IMPLEMENTATION**
* **CHAPTER 3 THE CODE & OUTPUT**
* **CHAPTER 4 REFERENCES**

**CHAPTER 1**

**INTRODUCTION**

The cifar-10 dataset contains 32x32 sized 60,000 labelled images of 10 different classes. The goal is  to build a model that would accurately classify the images into the following classes.

0: Aeroplane, 1: Automobile, 2: Bird, 3: Cat, 4: Deer, 5: Dog, 6: Frog, 7: Horse, 8: Ship, 9: Truck

I have used Convolutional Neural Network (CNN) for this image classification  problem. I have used Keras with Tensor Flow backend.

Initially I was using Spyder as IDE on my local computer, but the performance was really low.  Then, I switched to Google Colab, and used the GPU provided by them to execute my program.

**CHAPTER 2**

**IMPLEMENTATION**

The dataset is imported and categorized into training data (train\_X, train\_y) and testing data (test\_X,  test\_y). The labels (train\_y, test\_y) contained integer values from 0-9. The labels column is one hot  encoded using keras.util.to\_ categorical () function. This transformed the column to 10-dimension  binary vector with all the categories being the columns of the vector and having 1 for the specific  class.

The feature vectors i.e. the pixel values (train\_X, test\_X) are having values from 0 – 255 with 0  being the lightest shade and 255 being the darkest. Each image is of size 32x32 with 3 -channels  (Red, Green, Blue). The pixel values are normalized to the range of 0 – 1.

Here I have used Convolutional Neural Network to build the model. CNN has the following steps:

**>> Convolution** :

Conv2D function adds a 2-d convolution layer. Here a feature detector or filter is  used over the input image to generate a feature map. The feature detector is of size 3x3. And 32 or  64 signifies the number of feature detectors. The output will retain some necessary features and get  rid of unwanted information.

**>> MaxPooling** :

MaxPooling2D is used for pooling. This reduces dimensionality of the feature  maps and retain the most important information. Here 2x2 matrix is used on the feature map to find  the maximum value from those particular pixels. This is done across all the pixels to generate a  Pooled feature map which preserved the important feature and ignore spatial differences.

**>> Flattening** :

Flatten is used to convert the feature map to a 1-D vector. The elements are taken  row by row and put it into one column. This vector is given as input to the Fully connected Neural  Network for further processing.

>> **Full Connection Layer** :

Dense function is used to initialize a fully connected network. In CNN,  the hidden layers must be fully connected. The final layer contains 10 neurons to signify 10 output  classes.

>> **Dropout**:

Dropout is a regularization technique that is used to prevent overfitting. This layer  randomly sets some data points to 0. I’ve used Dropout(0.3), means this layer will randomly assign  0 to 30% of the data.

An Activation function is used to introduce non-linearity in the data. This is a function which maps  the input data to the output of that neuron. This output is given as input to another neuron, and so on  until the desired output is received. The activation function helps the network to learn the complex  pattern in the data.

Here ReLU (Rectified Linear Unit) is mainly used. ReLU function is defined as y = max (0, x). This  function returns 0 if it receives any negative value and for any positive value x, it returns the value.  Softmax function is used in the output layer. This function adds the value of output to a form that  the sum of the output becomes 1. It takes input vector of K-dimensions and normalize it into a  probability distribution consisting of K probabilities proportional to the exponents of the input  numbers.

For learning the parameters Stochastic Gradient Descent(SGD) is used. This is a classical  optimization algorithm to increase the model accuracy by minimizing the loss using standard weight  update formula of: Wt+1 = Wt + n\*gradient (n: learning rate).

In SGD, the parameter update for an epoch is done by using just one or a small subset (mini-batch)  data point from the batch of training data, unlike vanilla gradient descent, where update is done by  calculating all the data points of the training data for one epoch.

The learning rate chosen is 0.01 and a momentum of 0.9, which is an ideal situation. A decay is  used (decay = learning rate/epoch) to keep on decreasing the learning rate so that smaller steps are  taken as we move towards finding the optimal weights so that the loss function becomes minimum  (approaching 0).

The loss function used is Categorical Cross Entropy. This is used for multi-class classification  problems where we have to choose 1 class out of k classes.

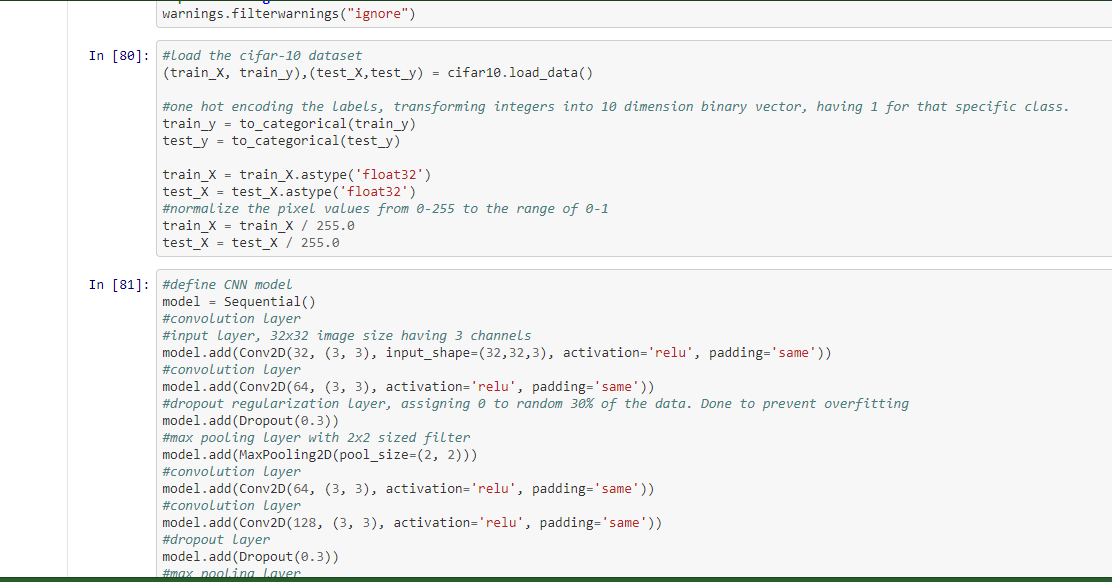
Through many trial and error experiments, by changing different parameters of the layers of CNN,  trying different combinations of the layers of CNN, like adding a convolution layer having different  number of filter, adding Dropout layers in between, trying out different learning rates and different  optimization algorithms, trying various activation functions, and many more such manipulation,  I’ve finally reached the Accuracy of 80.96%.

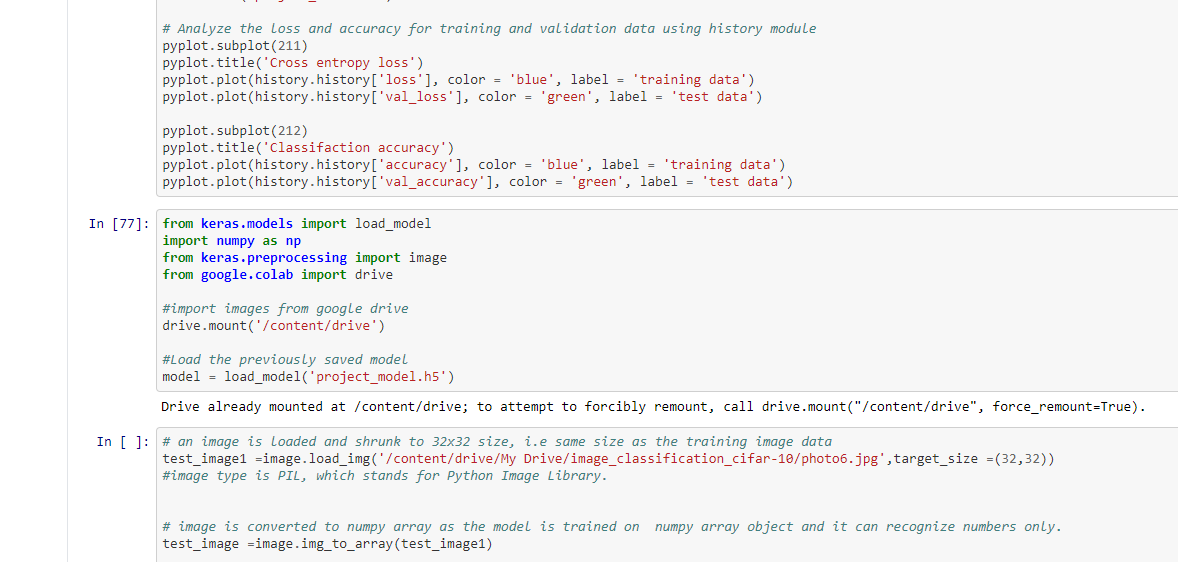
The CNN model predicted the correct class for most of the image during testing phase. The images  used for testing are downloaded from Internet

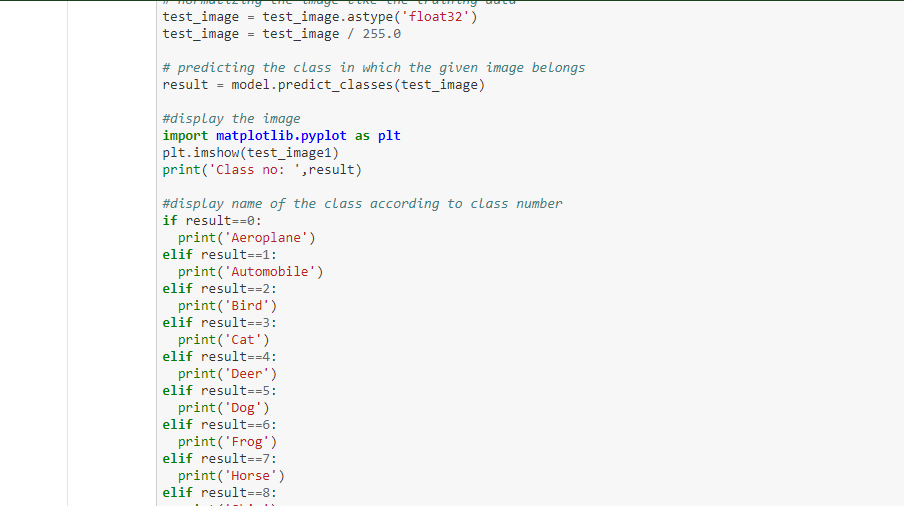
**CHAPTER 5**

**CODING**

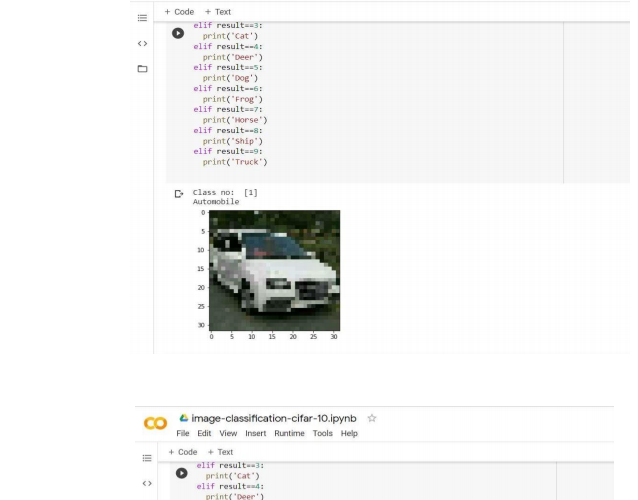
****

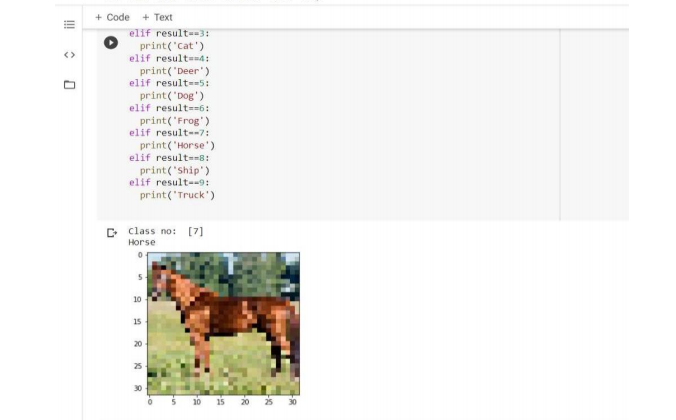


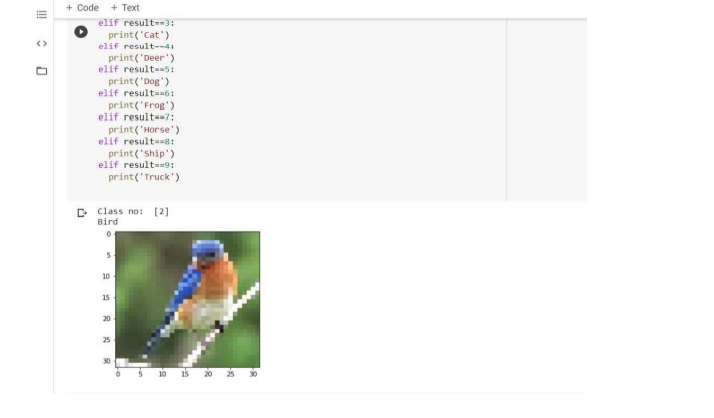


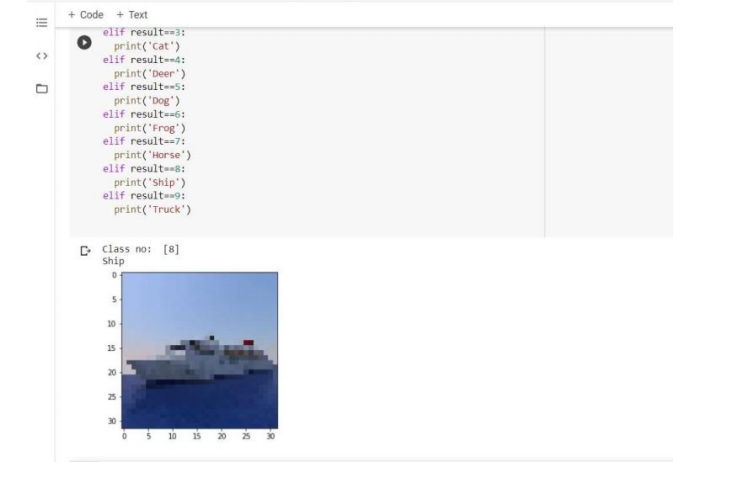


**OUTPUT**









**CHAPTER 4**

**REFERENCES**

* https://en.wikipedia.org/wiki
* <https://en.wikipedia.org/wiki/Machine_learning>
* https://en.wikipedia.org/wiki/language\_processing
* https://github.com
* https://github.com/python/omkar
* https://github.com/BuzzFeedNews
* <https://youtube.com>
* https:// www.tutorialspoint.com