# Illinois Institute of Technology

## CS-584 Machine Learning

# Deep Neural Networks - CIFAR10 Classification

# Intermediate Project Report

**Submitted By**

**Satwika Sriram (A20563950)**
**ssriram6@hawk.iit.edu**

**Fnu Anvika (A20556800)**
**aanvika@hawk.iit.edu**

# TABLE OF CONTENTS

## 1. INTRODUCTION

In the area of computer vision, CIFAR-10 is one of the most used benchmarks for image classification tasks. A difficult yet representative dataset for assessing how well different deep learning and machine learning models perform is provided by CIFAR-10.

Our main goal in this research is to create and train deep neural networks that can correctly categorize photos from the CIFAR-10 dataset into the appropriate groups. With this project, we want to investigate how well various neural network topologies, optimization methods, and regularization schemes perform when it comes to picture categorization.CIFAR-10 is a perfect testbed for evaluating the resilience and generalization abilities of deep learning models because of its small size and high picture quality.

In this introduction, we will provide a brief description of the CIFAR-10 dataset, emphasize its importance in the area of computer vision, and describe how we plan to use deep neural networks to tackle the picture classification job.

Our motivation to push the boundaries of deep learning and benefit the larger scientific community doesn't waver as we work with CIFAR-10 to further explore the field of image categorization. We cordially welcome you to go with us as we explore the nuances of neural network-based image classification and seek to outperform the CIFAR-10 benchmark.

## 2. PROBLEM DESCRIPTION

Our project's main goal is to categorize photos from the CIFAR-10 dataset into one of ten pre-established classes. 60,000 32x32 color pictures that fall into one of the following categories make up CIFAR-10: vehicles, trucks, birds, cats, deer, dogs, frogs, horses, and aircraft. The goal is to create a strong deep neural network model that can correctly classify and recognize these various animals and objects.

Several variables make the CIFAR-10 classification challenge extremely difficult:

1. **Low Resolution pictures:** Compared to datasets with greater resolution, the CIFAR-10 dataset's pictures have comparatively low resolution (32x32 pixels), which makes it challenging for models to identify minute details and features.

2. **Class Imbalance:** There may be discrepancies in the distribution of pictures across the various classes in CIFAR-10, which might make training and evaluating the model more difficult, particularly for the classes with less data.

3. **Variability in Object look:** There might be a lot of variation in the look of images in the same class, which makes it difficult to acquire discriminative characteristics that work well in various instances of the same class.

4. **Interclass Similarities:** A model must account for minute distinctions in order to prevent incorrect classifications in some CIFAR-10 classes, such as dogs and cats or trucks and cars.

Our study intends to use optimization methods, regularization techniques, and cutting edge deep neural network topologies to tackle these problems. To increase the model's resilience and performance, we will investigate a number of strategies, such as ensembling, data augmentation, transfer learning, and convolutional neural networks (CNNs).

We want to improve our knowledge of deep learning methods for image identification tasks by addressing the CIFAR-10 classification issue. We also hope to provide insights that may be used in more general computer vision applications, such object detection, picture segmentation, and scene comprehension. We want to create models that demonstrate excellent generalization capabilities across unknown data and real-world circumstances, in addition to achieving high accuracy on the CIFAR-10 benchmark via thorough testing and analysis.

.

## 3. DATASET DESCRIPTION

In the realm of computer vision, one benchmark dataset that is often used is CIFAR-10. There are 60,000 32x32 color pictures total, divided into 10 classes of 6,000 pictures each. Ten thousand test photos and fifty thousand training images make up the dataset.

The CIFAR-10 dataset has the following 10 classes:
1. Airplane
2. Automobile
3. Bird,
4. Cat,
5. Deer,
6. Dog,
7. Frog,
8. Horse,
9. Ship,
10. Truck

Every picture in the collection is an RGB image with three color channels (red, green, and blue), measuring 32 by 32 pixels. For every channel, the pixel values are integers between 0 and 255.

The dataset is extensively used for applications including object identification, picture categorization, and machine learning model benchmarking. Due to its tiny size and wide range of classes, it offers a difficult but doable challenge for testing different algorithms and models.

# 4. LITERATURE SURVEY

## Tensorflow

A free and open-source machine learning software library is called TensorFlow. Although it has many uses, its main emphasis is on deep neural network inference and training. Tensorflow is a symbolic math toolset based on dataflow and differentiable programming. It is used by Google for both production and research. TensorFlow was developed by the Google Brain team for usage inside the company. It was made available under the Apache License 2.0 in 2015. TensorFlow is the name of Google Brain's second-generation system. The release of version 1.0.0 occurred on February 11, 2017. While the standard version only operates on a single device, TensorFlow may run on many CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computation on graphics processing units). TensorFlow may be accessed on 64-bit operating systems such as Linux, macOS, Windows, and iOS and Android mobile platforms.

## Keras

Keras is a TensorFlow-based deep learning API written in Python. It was designed to make it as easy as possible for consumers to explore. Research success requires the ability to move as fast as possible from idea to result. A variety of implementations of fundamental neural-network building blocks, including as layers, goals, activation functions, optimizers, and other tools, are included in Keras to facilitate the handling of image and text input and to minimize the amount of coding needed to construct deep neural network code. Community help is available via a Slack channel and the code is posted on GitHub. In addition to standard neural networks, Keras also supports convolutional and recurrent neural networks. Other widely used utility layers that are supported include batch normalization, pooling, and dropout.

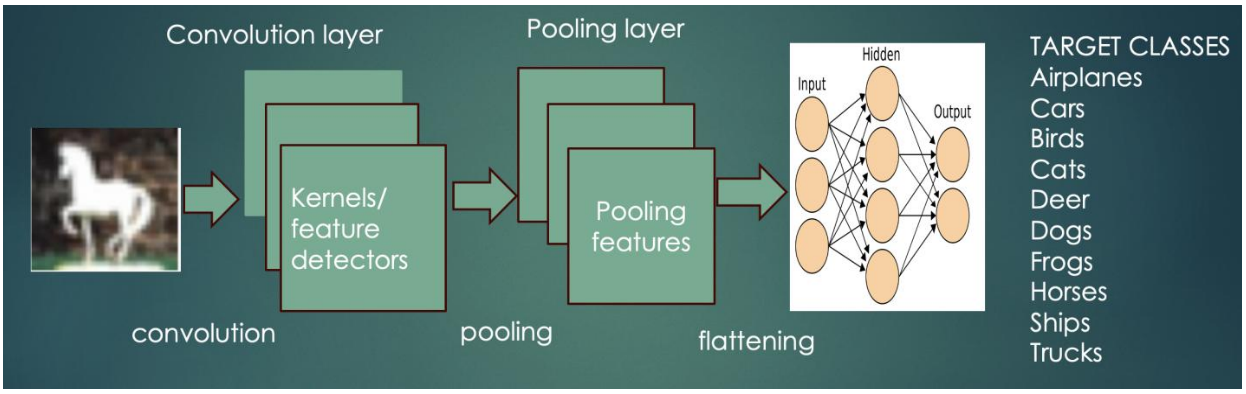
## Convolutional Neural Networks (CNN)

Deep Learning's ability to handle enormous amounts of data has made it an especially helpful method in recent years. The popularity of hidden layers has surpassed that of older methods, especially in pattern recognition. One of the most popular types of deep neural networks is the convolutional neural network. Since AI was still in its infancy in the 1950s, researchers have worked to develop a system. that has the ability to understand visual data Over time, this field's popularity increased. The name given to this kind of technology is computer vision. In the field of computer vision, the 2012 breakthrough was a quantum leap. An AI model developed by a team of academics at the University of Toronto significantly surpassed the top image recognition systems. The artificial intelligence system was called AlexNet, after its creator. In the 2012 ImageNet computer vision competition, Alex Krizhevsky won. It's amazing that 85 percent of the time is accurate. On the exam, the runner-up scored a creditable 74 percent. AlexNet was built on convolutional neural networks, a special kind of neural network. a neural network that attempts to closely mimic human vision. CNNs have changed over time. It's currently a vital component of many Computer Vision applications.

## Working of CNN Algorithm

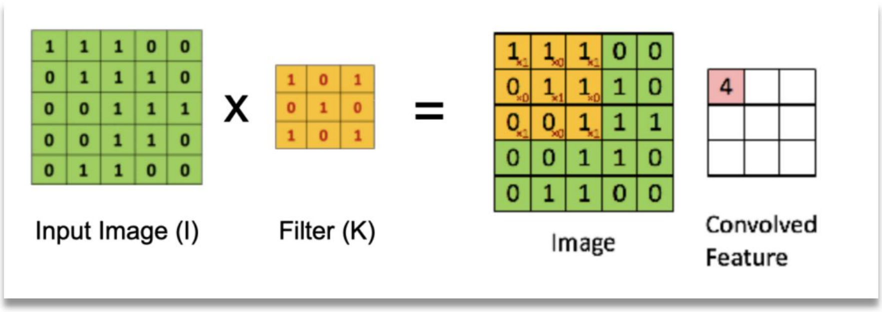
Convolutional neural networks feature several layers, which the input picture must travel through in order to get the intended result. Included are the following layers:

1. Convolutional layer
2. Pooling layer
3. Fully connected layer



### 1. The convolutional layer:

Here is where the convolution process is carried out. Filter or kernel: An array of weights learned by backpropagation that is used to extract features and edges from an input picture. Feature Map: the result of convolving a filter over an input picture.



- **Convolution operation:**

To create a feature map, this method entails convolving a specified filter over an input picture.

- **Padding:**

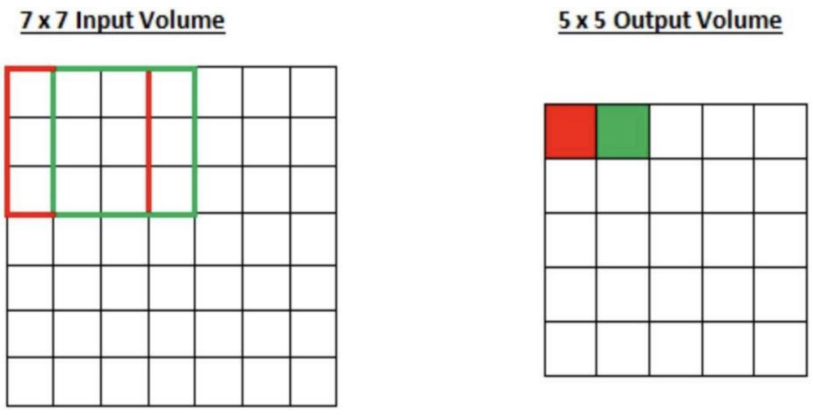
By adding a second layer around the edge of our original image, we preserve its details without reducing them. Traditionally, we pad using zeros. It's a hyperparameter that requires adjustment. Some of the reasons we use padding are as follows:

1. Our input image will shrink and lose its original size until it reaches the last stage since the convolution technique is repeated.

2. Information is lost from the corners when a filter is applied to an input picture because the top left corner appears just once while pixels in the center appear several times.

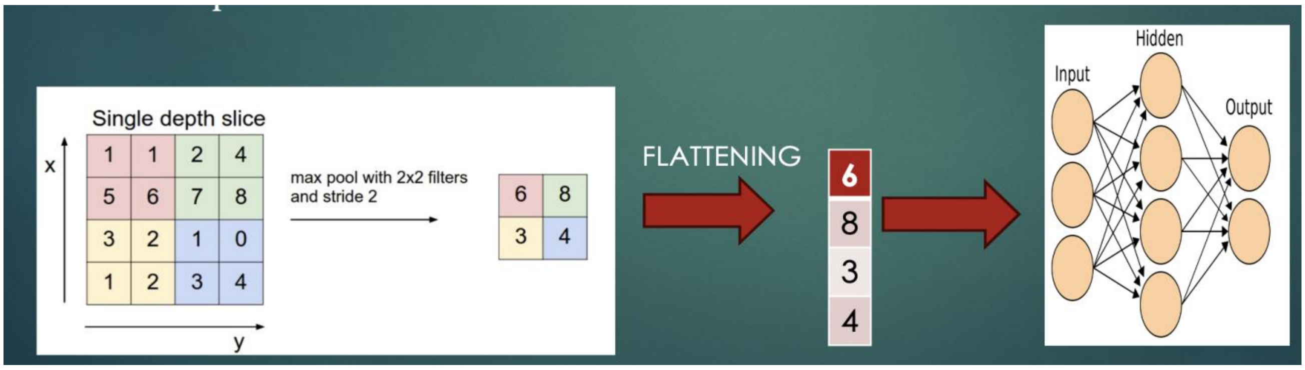
- **Stride:**

The stride is the amount of movement the kernel makes as it advances over the image. Stride is a hyperparameter in this context that describes the step of the convolution procedure.



**2. The Pooling Layer :**

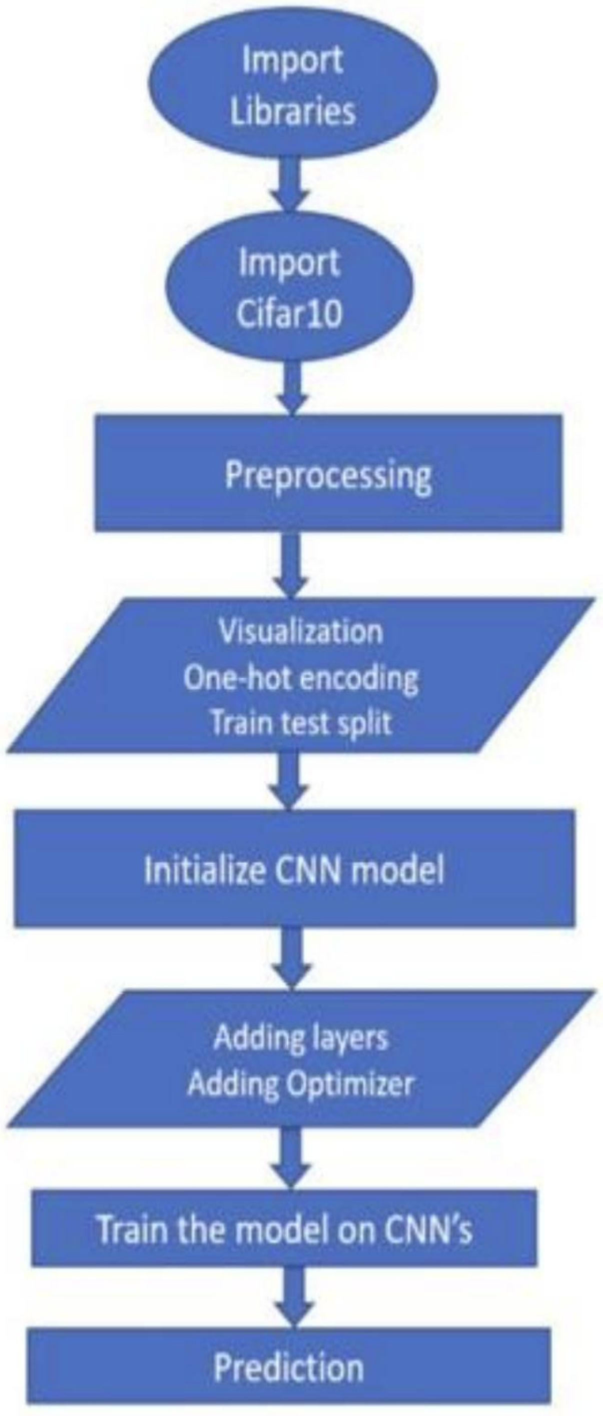
ConvNets usually use pooling layers in addition to convolutional layers to reduce the size of the picture, speed up computation, and somewhat strengthen some of the detecting properties. Although there are many other types of pooling, these two are the most well-known: There are two kinds of pooling: average pooling and maximum pooling.



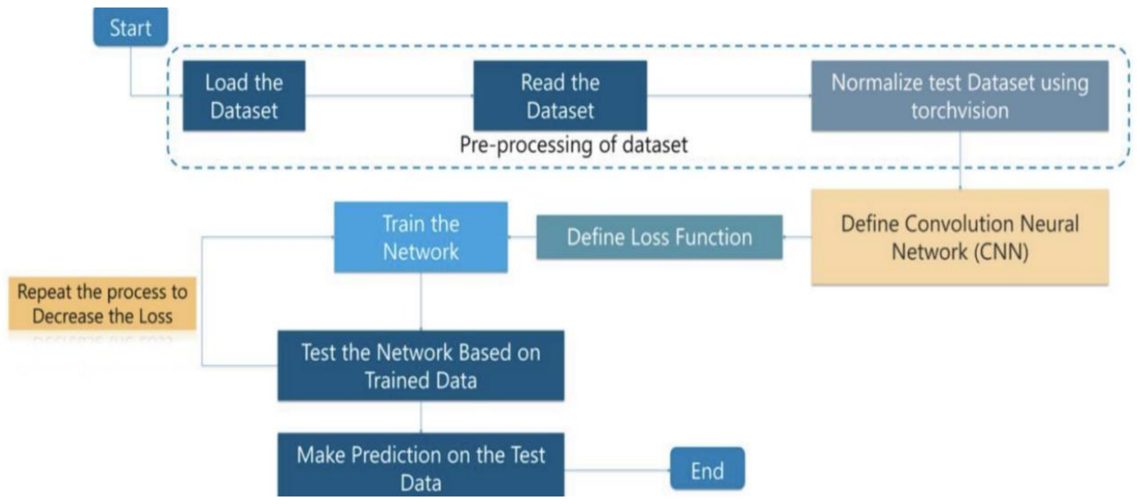
**3. Fully Connected Layer/ Dense Layer :**

In order to generate a prediction, the dense layer—a fully connected layer—feeds the convolutional layers' outputs via one or more neural layers. When a layer is fully connected, every element from every feature in the previous layer is computed for every element of every output feature, which also includes an activation function based on our business requirements. Between the convolutional and fully connected layers comes a layer known as "Flatten." A fully connected neural network classifier may receive a vector that is created by flattening a two-dimensional matrix of attributes.

## SYSTEM ARCHITECTURE:



**DATA FLOW DIAGRAM:**


Start
Load the Dataset
Read the Dataset
Normalize test Dataset using torchvision
Pre-processing of dataset
Define Convolution Neural Network (CNN)
Define Loss Function
Train the Network
Repeat the process to Decrease the Loss
Test the Network Based on Trained Data
Make Prediction on the Test Data
End

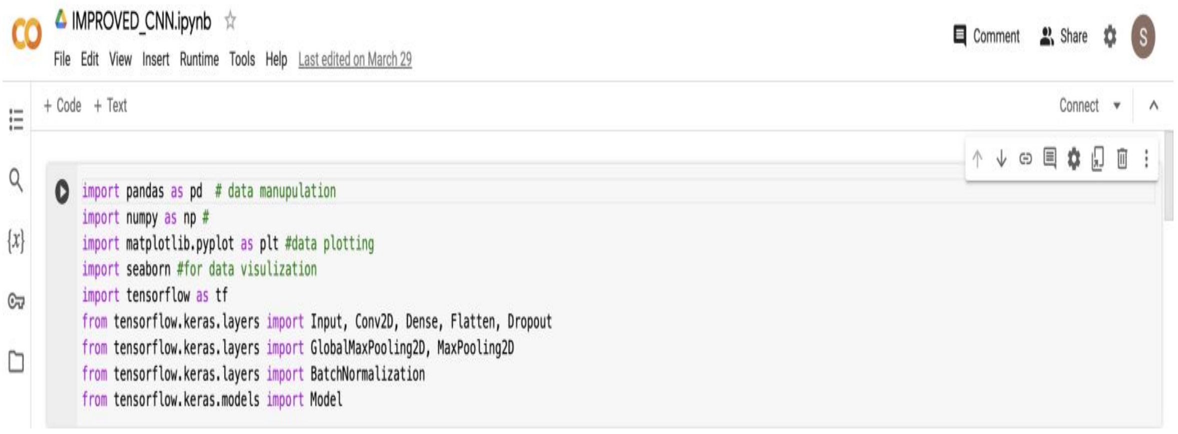
## 5 . SYSTEM MODULES AND ACCOMPLISHMENTS

The modules in this project comprise of:

1. Importing libraries
2. Importing Dataset
3. Data Visualization
4. Data Preparation
5. Training the model
6. Model Evaluation

### 5.1 Importing the Libraries

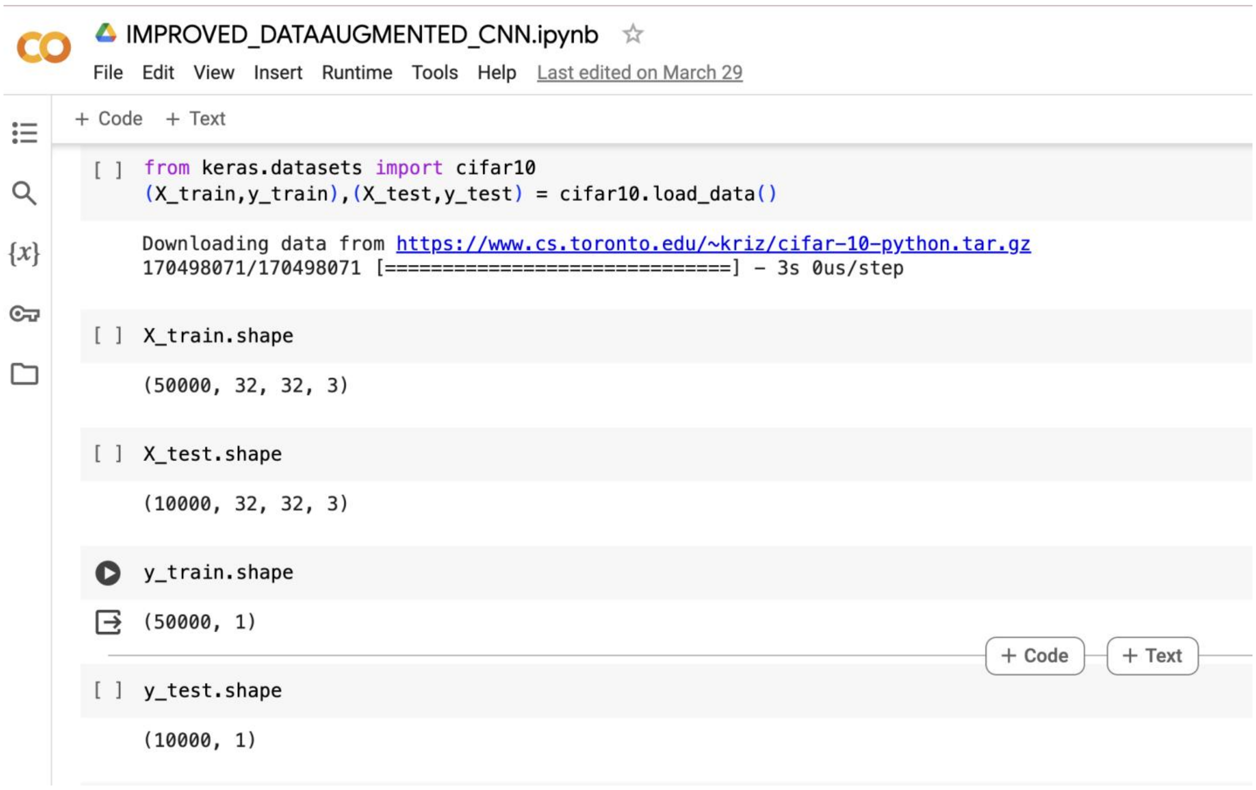
Importing the libraries as reqired


IMPROVED_CNN.ipynb
File Edit View Insert Runtime Tools Help Last edited on March 29
Comment Share
+ Code + Text
Connect

```
import pandas as pd  # data manupulation
import numpy as np #
import matplotlib.pyplot as plt #data plotting
import seaborn #for data visulization
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout
from tensorflow.keras.layers import GlobalMaxPooling2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.models import Model
```

## 5.2 Importing the Dataset

In this step, we will feed the cifar10 data into our model and separate it into train and test sets. We now have 50000 photos in the train set and 10,000 images in the test set out of our total of 60000 shots. The collection's photographs have dimensions of (32,32,3) since they are colored.


IMPROVED_DATAAUGMENTED_CNN.ipynb
File Edit View Insert Runtime Tools Help Last edited on March 29
+ Code + Text

```
from keras.datasets import cifar10
(X_train,y_train),(X_test,y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 3s 0us/step
```

```
X_train.shape
```

```
(50000, 32, 32, 3)
```

```
X_test.shape
```

```
(10000, 32, 32, 3)
```

```
y_train.shape
```

```
(50000, 1)
```

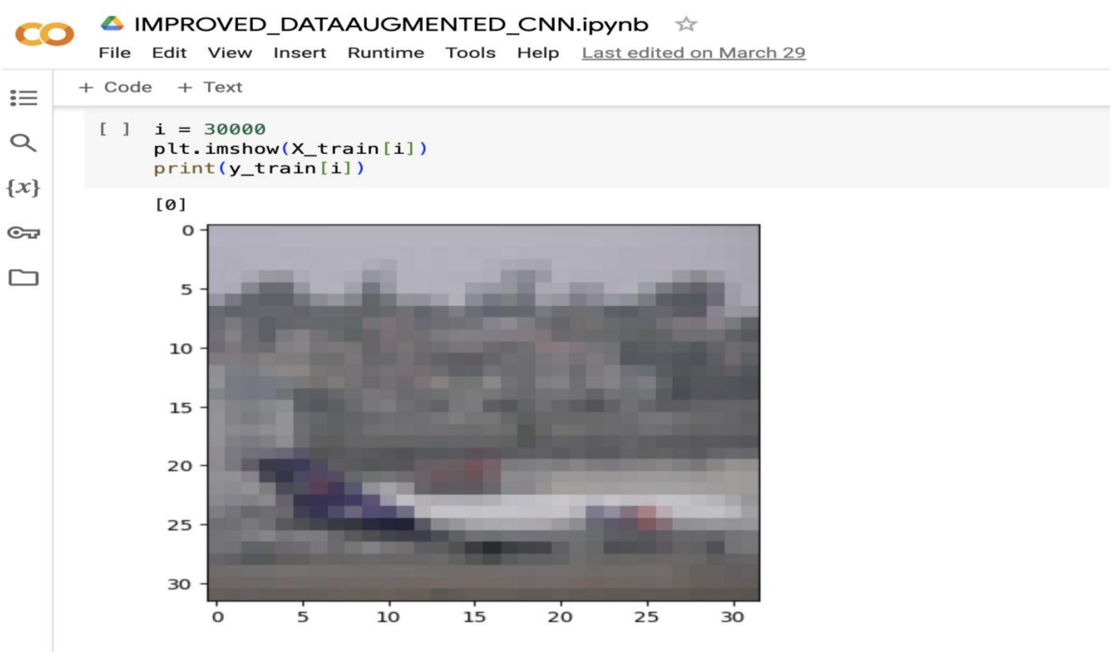
```
y_test.shape
```

```
(10000, 1)
```

## 5.3 Data Visualization

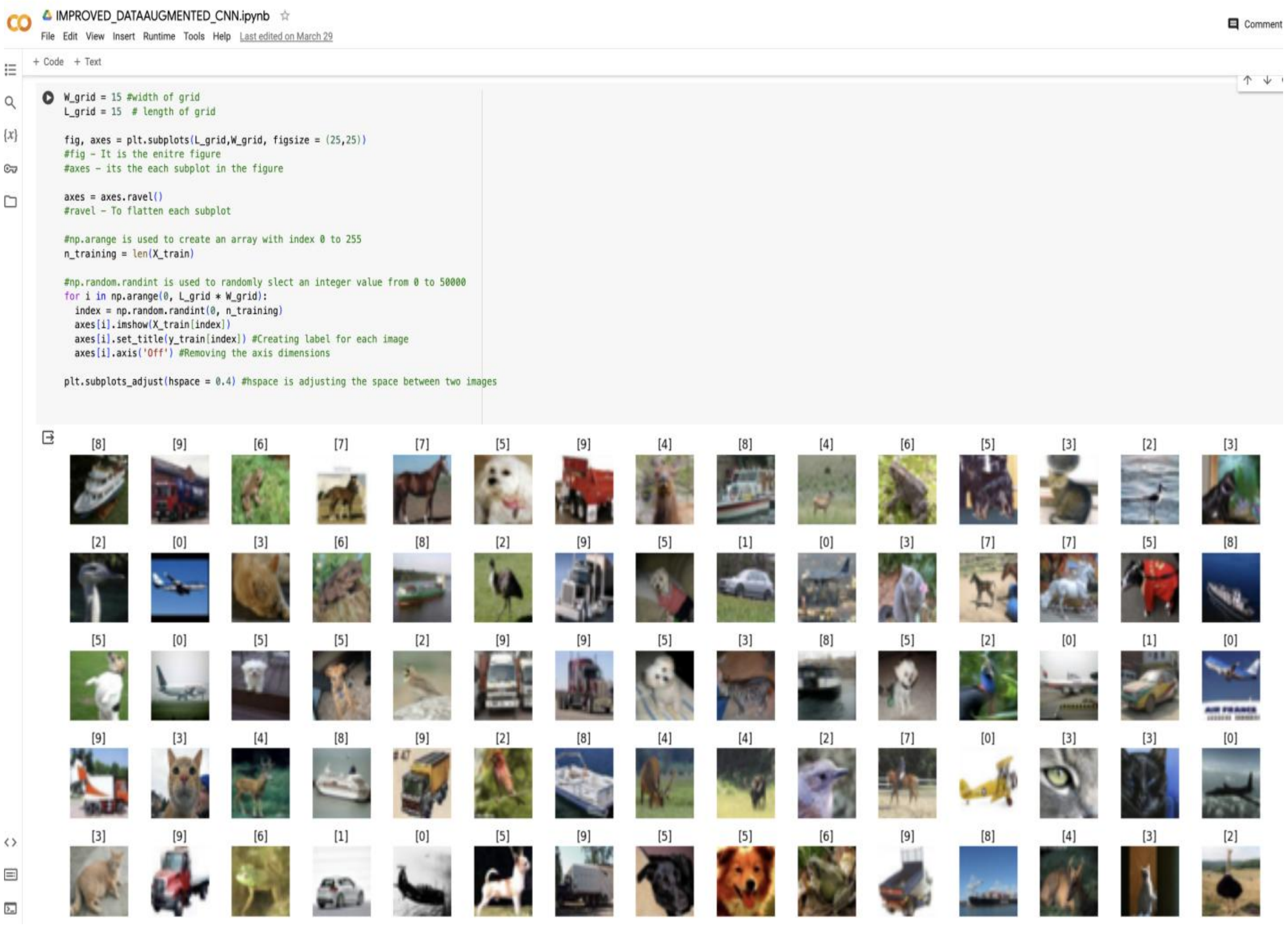
Information and data are represented graphically in data visualization. Visualizations are an effective tool for conveying complex concepts, patterns, and trends in an understandable and straightforward manner. We may examine data, spot trends and outliers, and share our conclusions with others by utilizing visualizations. With the CIFAR 10 dataset, we created a few visualizations.

In data analysis, visualizations are a crucial tool that may reveal insights that would otherwise be hard to perceive. They may also be used to disseminate research results to a larger audience, increasing the accessibility and interest level of data. Effective visualization skills are becoming more and more crucial due to the daily generation of ever-increasing amounts of data.

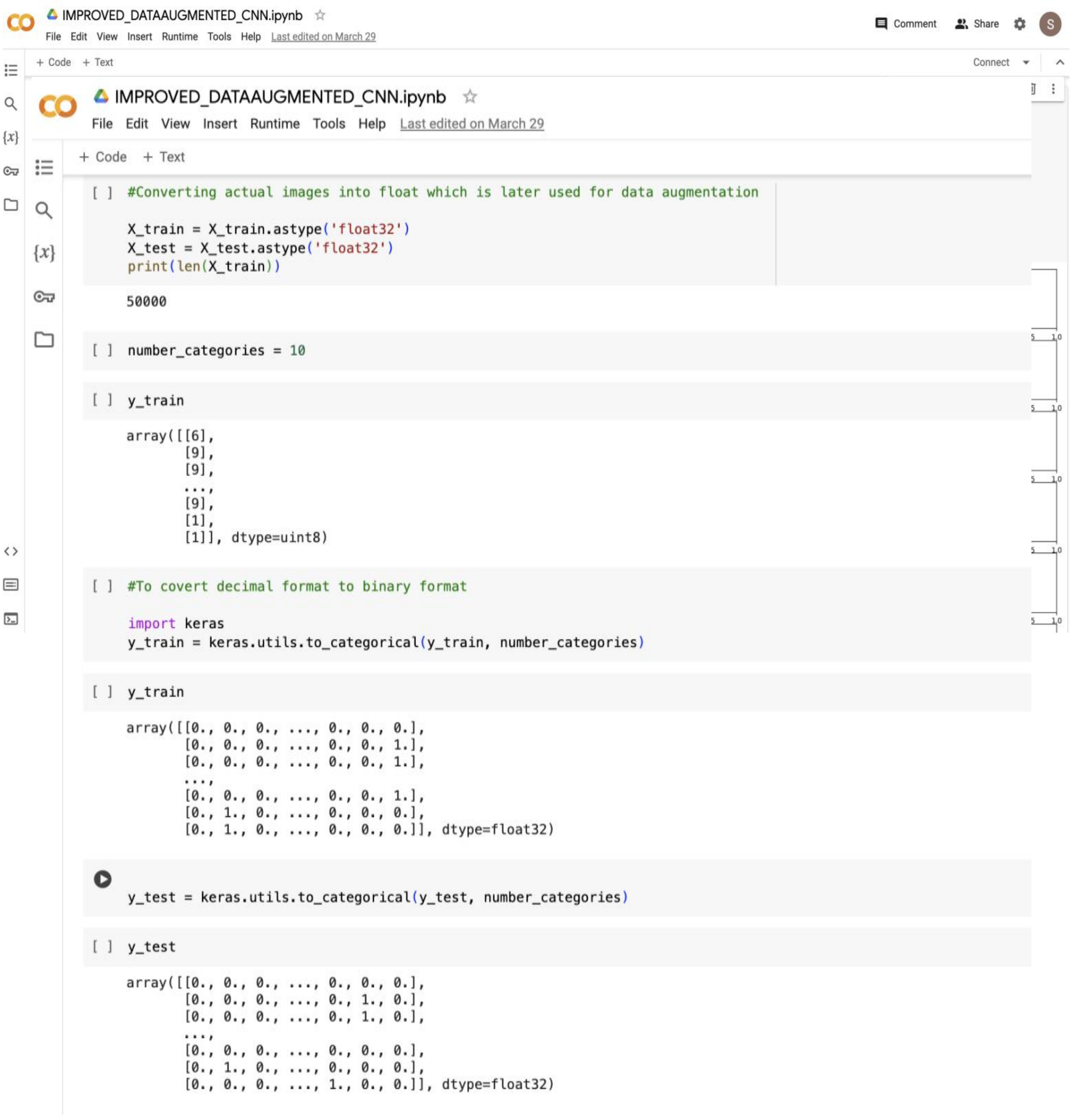

IMPROVED_DATAAUGMENTED_CNN.ipynb
File Edit View Insert Runtime Tools Help Last edited on March 29
+ Code + Text

```
i = 30000
plt.imshow(X_train[i])
print(y_train[i])
```

```
[0]
```


IMPROVED_DATAAUGMENTED_CNN.ipynb
File Edit View Insert Runtime Tools Help Last edited on March 29
Comment
+ Code + Text

```
W_grid = 15 #width of grid
L_grid = 15  # length of grid

fig, axes = plt.subplots(L_grid,W_grid, figsize = (25,25))
#fig - It is the enitre figure
#axes - its the each subplot in the figure

axes = axes.ravel()
#ravel - To flatten each subplot

#np.arange is used to create an array with index 0 to 255
n_training = len(X_train)

#np.random.randint is used to randomly slect an integer value from 0 to 50000
for i in np.arange(0, L_grid * W_grid):
  index = np.random.randint(0, n_training)
  axes[i].imshow(X_train[index])
  axes[i].set_title(y_train[index]) #Creating label for each image
  axes[i].axis('Off') #Removing the axis dimensions

plt.subplots_adjust(hspace = 0.4) #hspace is adjusting the space between two images
```


IMPROVED_DATAAUGMENTED_CNN.ipynb
File Edit View Insert Runtime Tools Help Last edited on March 29
Comment Share
+ Code + Text
Connect

```
#Converting actual images into float which is later used for data augmentation

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
print(len(X_train))

50000
```

```
number_categories = 10
```

```
y_train

array([[6],
       [9],
       [9],
       ...,
       [9],
       [1],
       [1]], dtype=uint8)
```

```
#To covert decimal format to binary format

import keras
y_train = keras.utils.to_categorical(y_train, number_categories)
```

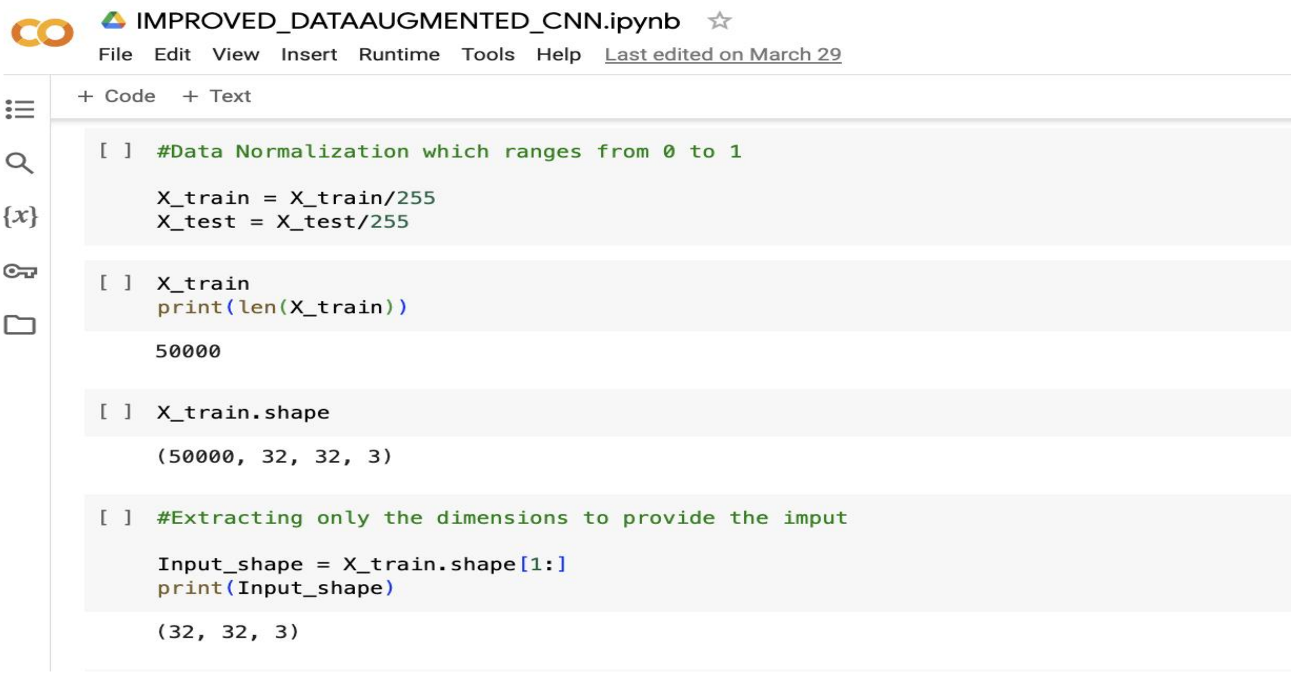
```
y_train

array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 1.],
       ...,
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.]], dtype=float32)
```

```
y_test = keras.utils.to_categorical(y_test, number_categories)
```

```
y_test

array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.]], dtype=float32)
```

## 5.4 Data Preparation

Hot encoding is one method of altering data to make it more predictive and ready for an algorithm. Each category value is transformed into a new categorical column and assigned a binary value of 1 or 0 using one-hot. Every integer value is represented as a binary vector.


IMPROVED_DATAAUGMENTED_CNN.ipynb
File Edit View Insert Runtime Tools Help Last edited on March 29
+ Code + Text

```
#Data Normalization which ranges from 0 to 1

X_train = X_train/255
X_test = X_test/255
```

```
X_train
print(len(X_train))
```

```
50000
```

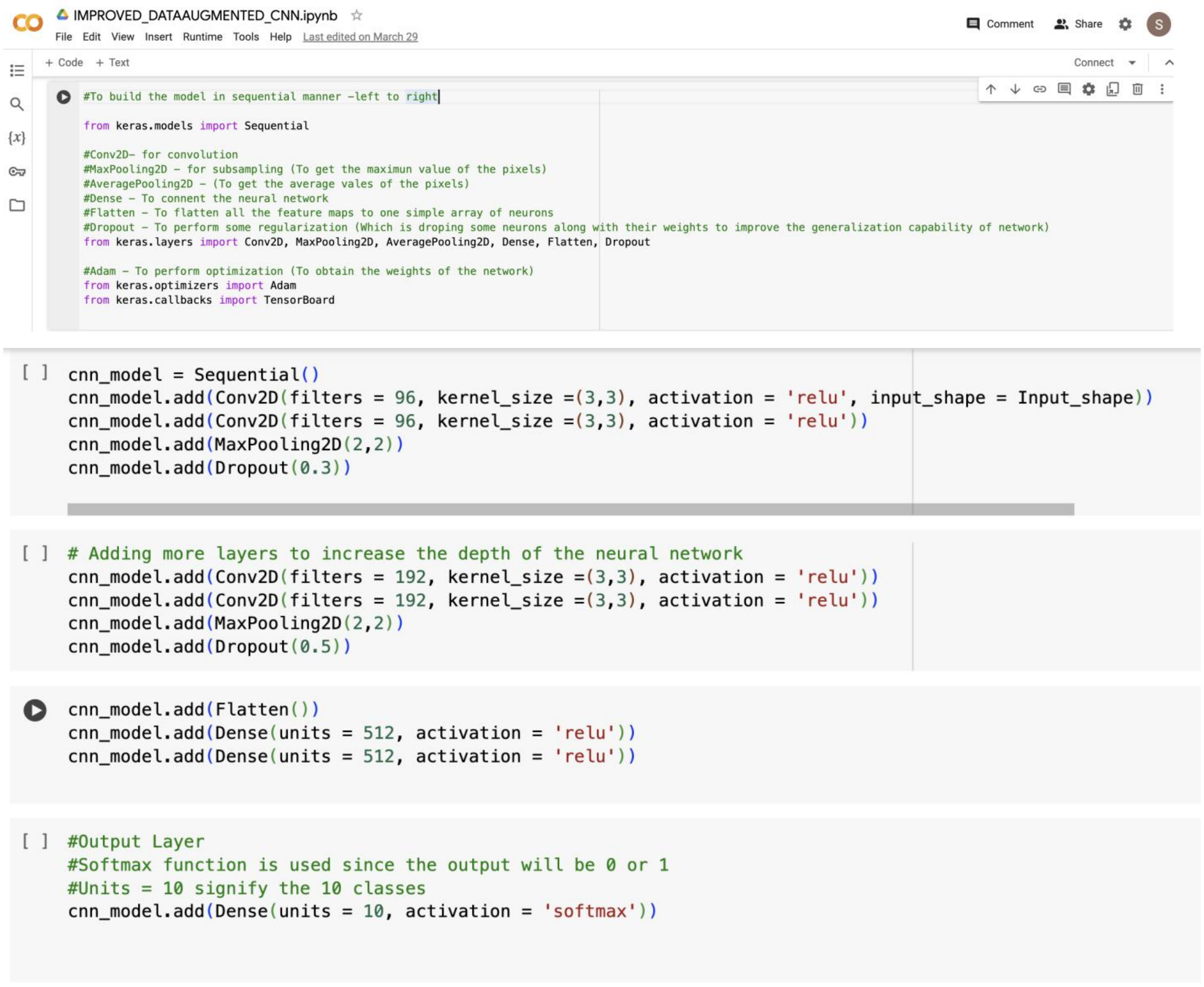
```
X_train.shape
```

```
(50000, 32, 32, 3)
```

```
#Extracting only the dimensions to provide the imput

Input_shape = X_train.shape[1:]
print(Input_shape)
```

```
(32, 32, 3)
```

## 5.5 Training the model


IMPROVED_DATAAUGMENTED_CNN.ipynb
File Edit View Insert Runtime Tools Help Last edited on March 29
Comment
Share
S
+ Code + Text
Connect

```
#To build the model in sequential manner -left to right

from keras.models import Sequential

#Conv2D- for convolution
#MaxPooling2D - for subsampling (To get the maximun value of the pixels)
#AveragePooling2D - (To get the average vales of the pixels)
#Dense - To connent the neural network
#Flatten - To flatten all the feature maps to one simple array of neurons
#Dropout - To perform some regularization (Which is droping some neurons along with their weights to improve the generalization capability of network)
from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D, Dense, Flatten, Dropout

#Adam - To perform optimization (To obtain the weights of the network)
from keras.optimizers import Adam
from keras.callbacks import TensorBoard
```

```
cnn_model = Sequential()
cnn_model.add(Conv2D(filters = 96, kernel_size =(3,3), activation = 'relu', input_shape = Input_shape))
cnn_model.add(Conv2D(filters = 96, kernel_size =(3,3), activation = 'relu'))
cnn_model.add(MaxPooling2D(2,2))
cnn_model.add(Dropout(0.3))
```

```
# Adding more layers to increase the depth of the neural network
cnn_model.add(Conv2D(filters = 192, kernel_size =(3,3), activation = 'relu'))
cnn_model.add(Conv2D(filters = 192, kernel_size =(3,3), activation = 'relu'))
cnn_model.add(MaxPooling2D(2,2))
cnn_model.add(Dropout(0.5))
```

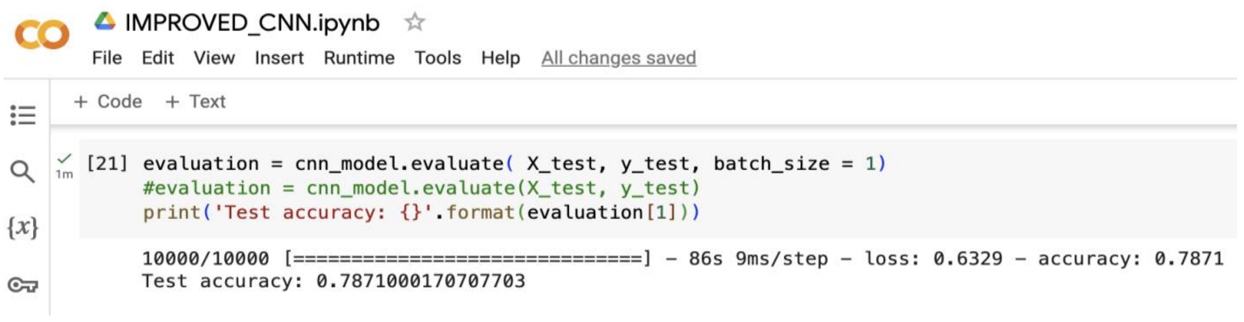
```
cnn_model.add(Flatten())
cnn_model.add(Dense(units = 512, activation = 'relu'))
cnn_model.add(Dense(units = 512, activation = 'relu'))
```

```
#Output Layer
#Softmax function is used since the output will be 0 or 1
#Units = 10 signify the 10 classes
cnn_model.add(Dense(units = 10, activation = 'softmax'))
cnn_model.compile(loss= 'categorical_crossentropy', optimizer=Adam(learning_rate=1.0e-4), metrics = ['accuracy'])
```

```
history = cnn_model.fit( X_train, y_train, batch_size = 32, epochs = 10, validation_data=(X_test, y_test), shuffle = True)
```

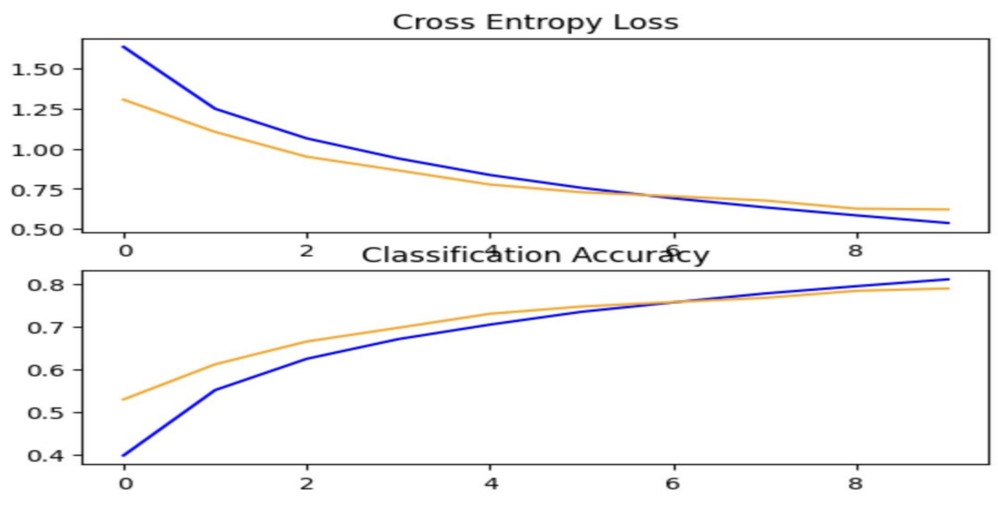
```
Epoch 1/10
1563/1563 [==============================] - 997s 637ms/step - loss: 1.6350 - accuracy: 0.3980 - val_loss: 1.3059 - val_accuracy: 0.5294
Epoch 2/10
1563/1563 [==============================] - 986s 631ms/step - loss: 1.2491 - accuracy: 0.5514 - val_loss: 1.1047 - val_accuracy: 0.6118
Epoch 3/10
1563/1563 [==============================] - 947s 606ms/step - loss: 1.0646 - accuracy: 0.6247 - val_loss: 0.9504 - val_accuracy: 0.6653
Epoch 4/10
1563/1563 [==============================] - 982s 628ms/step - loss: 0.9395 - accuracy: 0.6709 - val_loss: 0.8649 - val_accuracy: 0.6975
Epoch 5/10
1563/1563 [==============================] - 995s 637ms/step - loss: 0.8366 - accuracy: 0.7049 - val_loss: 0.7771 - val_accuracy: 0.7303
Epoch 6/10
1563/1563 [==============================] - 984s 629ms/step - loss: 0.7564 - accuracy: 0.7349 - val_loss: 0.7291 - val_accuracy: 0.7473
Epoch 7/10
1563/1563 [==============================] - 985s 630ms/step - loss: 0.6915 - accuracy: 0.7569 - val_loss: 0.7039 - val_accuracy: 0.7577
Epoch 8/10
1563/1563 [==============================] - 946s 605ms/step - loss: 0.6349 - accuracy: 0.7778 - val_loss: 0.6776 - val_accuracy: 0.7676
Epoch 9/10
1563/1563 [==============================] - 983s 629ms/step - loss: 0.5848 - accuracy: 0.7949 - val_loss: 0.6266 - val_accuracy: 0.7841
Epoch 10/10
1563/1563 [==============================] - 981s 628ms/step - loss: 0.5374 - accuracy: 0.8110 - val_loss: 0.6216 - val_accuracy: 0.7896
```

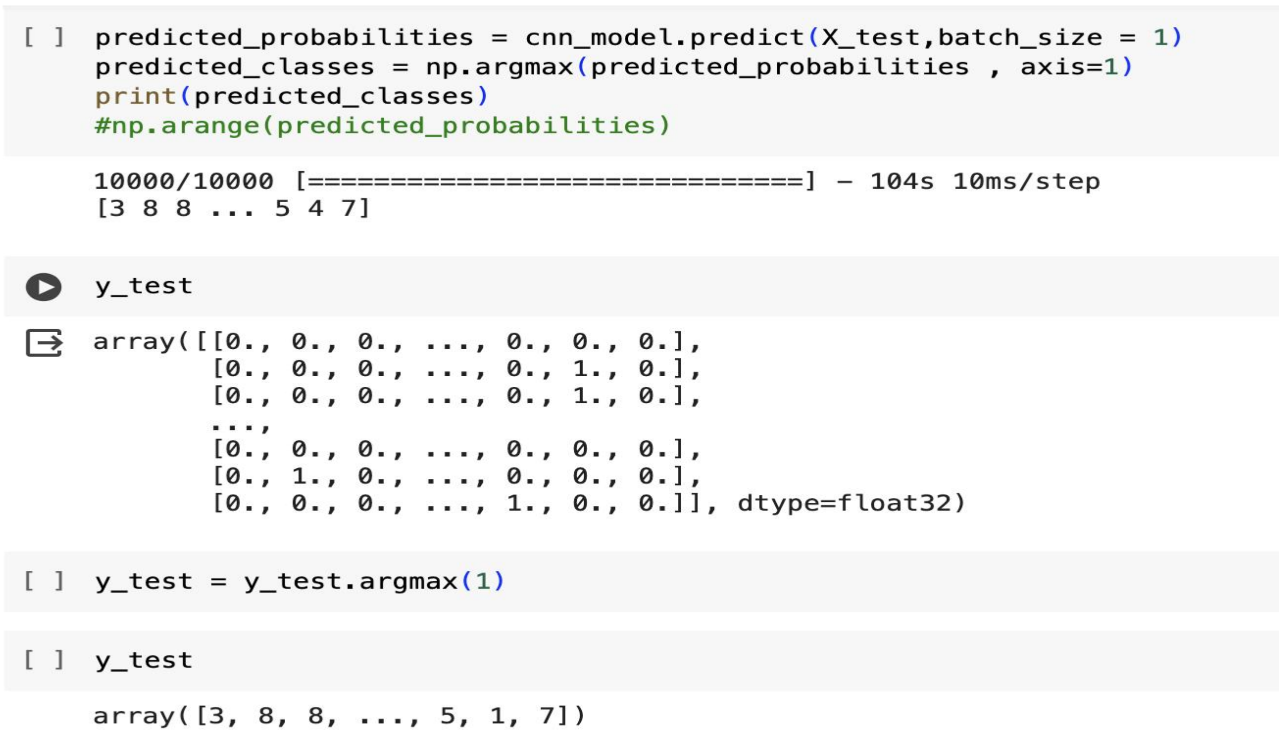
## 5.6 Model Evaluation



```
[21] evaluation = cnn_model.evaluate( X_test, y_test, batch_size = 1)
     #evaluation = cnn_model.evaluate(X_test, y_test)
     print('Test accuracy: {}'.format(evaluation[1]))

     10000/10000 [==============================] - 86s 9ms/step - loss: 0.6329 - accuracy: 0.7871
     Test accuracy: 0.7871000170707703
```

```
# plot loss
from matplotlib import pyplot

def summarize_diagnostics(history):
  pyplot.subplot(211)
  pyplot.title('Cross Entropy Loss')
  pyplot.plot(history.history['loss'], color='blue',label='train')
  pyplot.plot(history.history['val_loss'], color='orange',label='test')
  # plot accuracy
  pyplot.subplot(212)
  pyplot.title('Classification Accuracy')
  pyplot.plot(history.history['accuracy'], color='blue',label='train')
  pyplot.plot(history.history['val_accuracy'], color='orange',label='test')

summarize_diagnostics(history)
```





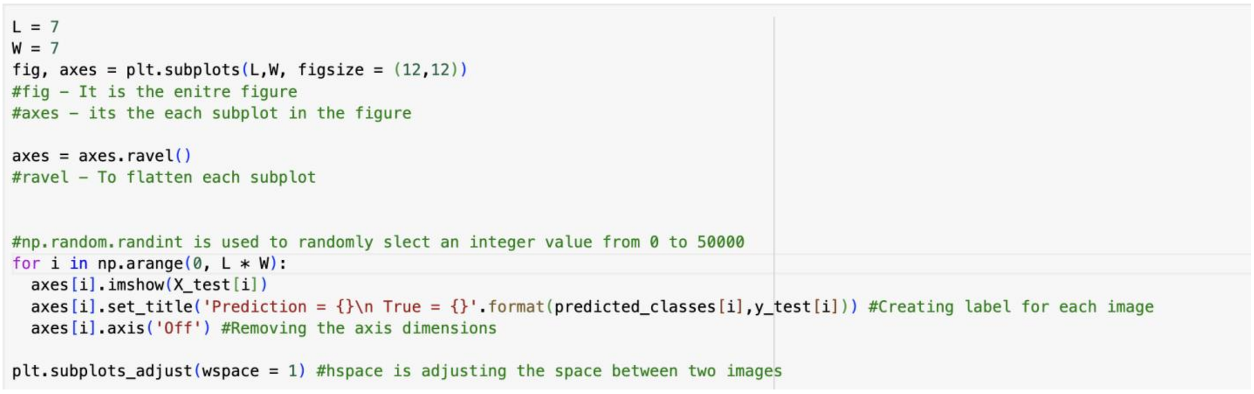
```
[ ]  predicted_probabilities = cnn_model.predict(X_test,batch_size = 1)
     predicted_classes = np.argmax(predicted_probabilities , axis=1)
     print(predicted_classes)
     #np.arange(predicted_probabilities)

     10000/10000 [==============================] - 104s 10ms/step
     [3 8 8 ... 5 4 7]
```

```
y_test
```

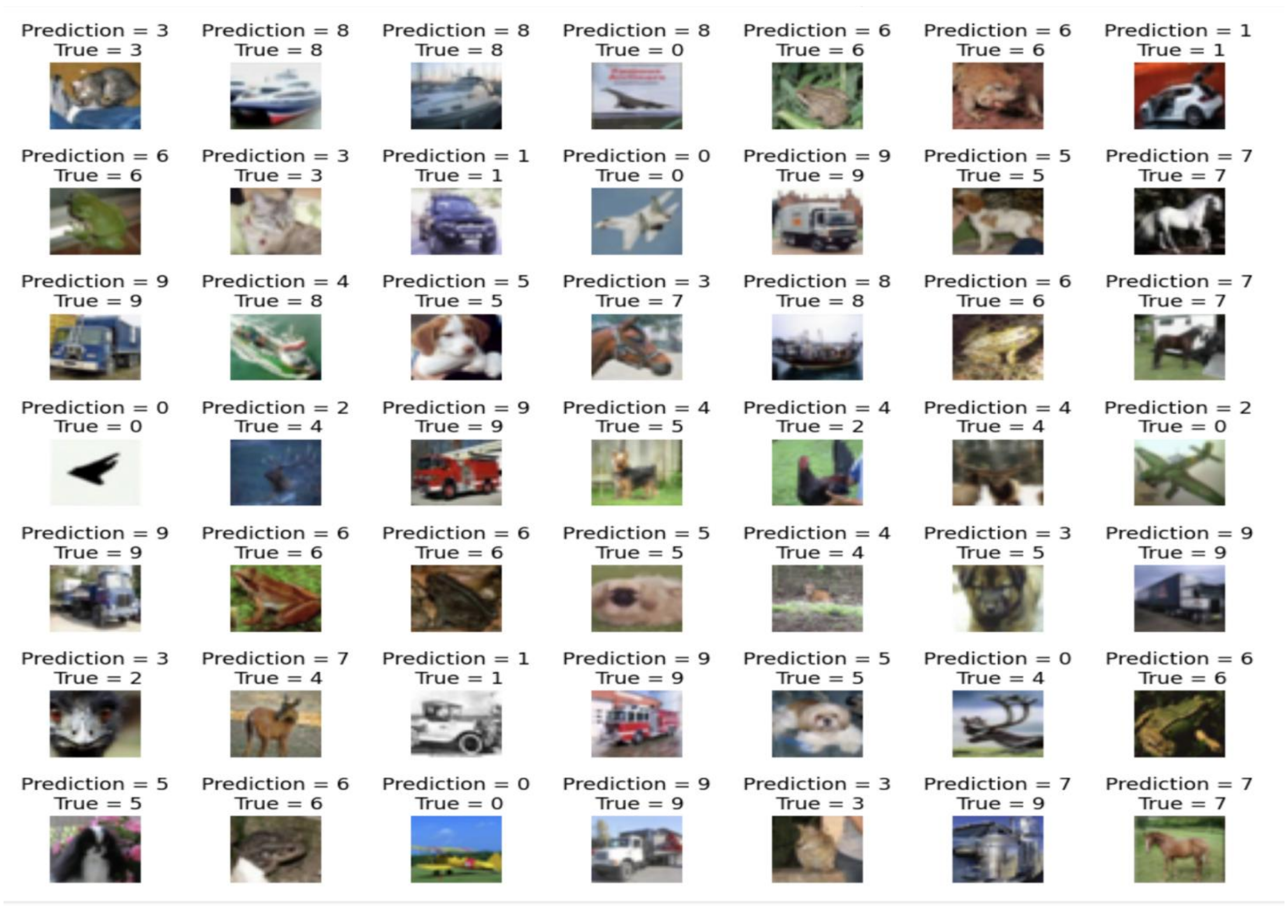
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.]], dtype=float32)
```

```
[ ]  y_test = y_test.argmax(1)
```

```
[ ]  y_test

     array([3, 8, 8, ..., 5, 1, 7])
```



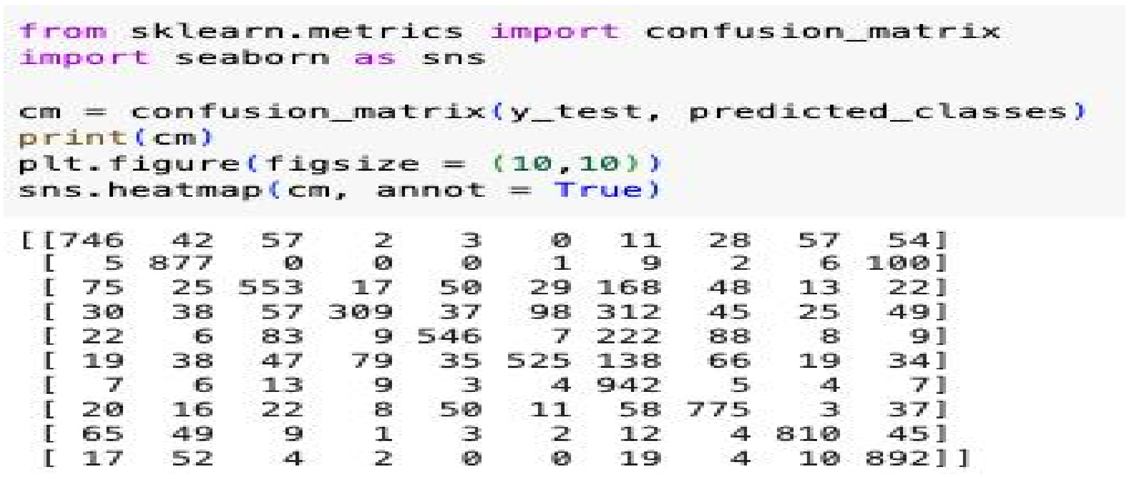
```
L = 7
W = 7
fig, axes = plt.subplots(L,W, figsize = (12,12))
#fig - It is the enitre figure
#axes - its the each subplot in the figure

axes = axes.ravel()
#ravel - To flatten each subplot


#np.random.randint is used to randomly slect an integer value from 0 to 50000
for i in np.arange(0, L * W):
  axes[i].imshow(X_test[i])
  axes[i].set_title('Prediction = {}\n True = {}'.format(predicted_classes[i],y_test[i])) #Creating label for each image
  axes[i].axis('Off') #Removing the axis dimensions

plt.subplots_adjust(wspace = 1) #hspace is adjusting the space between two images
```



## CONFUSION MATRIX AND HEAT MAP :

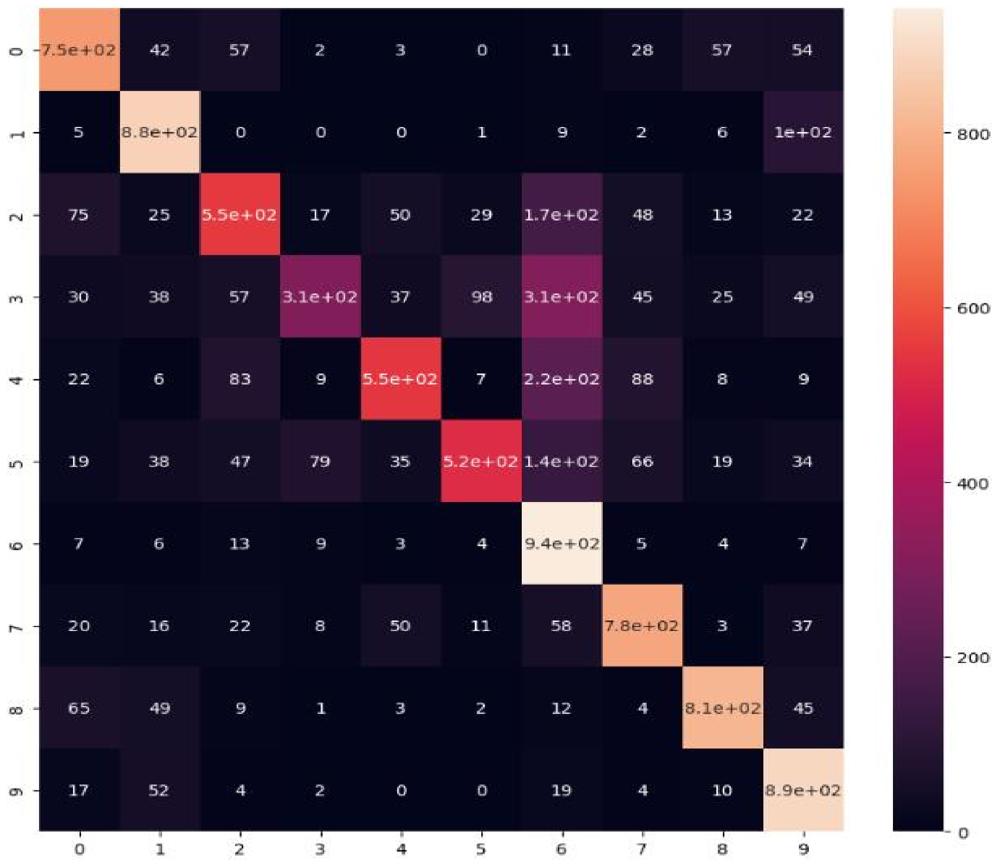
A confusion matrix is a table that compares the predicted labels of a piece of data with the actual labels in order to assess how well a machine learning algorithm is doing. It is also referred to as a contingency table or an error matrix.

A machine learning model's efficacy may be assessed using a variety of performance measures, including accuracy, precision, recall, and F1 score, which can be computed by examining the values in the confusion matrix.



```python
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(y_test, predicted_classes)
print(cm)
plt.figure(figsize = (10,10))
sns.heatmap(cm, annot = True)
```

```
[[746  42  57   2   3   0  11  28  57  54]
 [  5 877   0   0   0   1   9   2   6 100]
 [ 75  25 553  17  50  29 168  48  13  22]
 [ 30  38  57 309  37  98 312  45  25  49]
 [ 22   6  83   9 546   7 222  88   8   9]
 [ 19  38  47  79  35 525 138  66  19  34]
 [  7   6  13   9   3   4 942   5   4   7]
 [ 20  16  22   8  50  11  58 775   3  37]
 [ 65  49   9   1   3   2  12   4 810  45]
 [ 17  52   4   2   0   0  19   4  10 892]]
```



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 7.5e+02 | 42 | 57 | 2 | 3 | 0 | 11 | 28 | 57 | 54 |
| 1 | 5 | 8.8e+02 | 0 | 0 | 0 | 1 | 9 | 2 | 6 | 1e+02 |
| 2 | 75 | 25 | 5.5e+02 | 17 | 50 | 29 | 1.7e+02 | 48 | 13 | 22 |
| 3 | 30 | 38 | 57 | 3.1e+02 | 37 | 98 | 3.1e+02 | 45 | 25 | 49 |
| 4 | 22 | 6 | 83 | 9 | 5.5e+02 | 7 | 2.2e+02 | 88 | 8 | 9 |
| 5 | 19 | 38 | 47 | 79 | 35 | 5.2e+02 | 1.4e+02 | 66 | 19 | 34 |
| 6 | 7 | 6 | 13 | 9 | 3 | 4 | 9.4e+02 | 5 | 4 | 7 |
| 7 | 20 | 16 | 22 | 8 | 50 | 11 | 58 | 7.8e+02 | 3 | 37 |
| 8 | 65 | 49 | 9 | 1 | 3 | 2 | 12 | 4 | 8.1e+02 | 45 |
| 9 | 17 | 52 | 4 | 2 | 0 | 0 | 19 | 4 | 10 | 8.9e+02 |

## 6. REMAINING TASKS

- Perform data augmentation on the model to improve the accuracy of the model.

## 7. REFERENCES

1. Doon, R., Kumar Rawat, T., & Gautam, S. (2018). Cifar-10 Classification using Deep Convolutional Neural Network. In 2018 IEEE Punecon. 2018 IEEE Punecon. IEEE. https://doi.org/10.1109/punecon.2018.8745428

2. Aslam, S., & Nassif, A. B. (2023). Deep learning based CIFAR-10 classification. In 2023 Advances in Science and Engineering Technology International Conferences (ASET). . https://doi.org/10.1109/aset56582.2023.10180767

3. Vinay, S. B., & Balasubramanian, S. (n.d.). A COMPARATIVE STUDY OF CONVOLUTIONAL NEURAL NETWORKS AND CYBERNETIC APPROACHES ON CIFAR-10 DATASET. In International Journal of Machine Learning and Cybernetics (IJMLC) (Vol. 1, Issue 1). https://iaeme.com

4. Emonds, Y., Xi, K., & Fröning, H. (2024). Implications of Noise in Resistive Memory on Deep Neural Networks for Image Classification. http://arxiv.org/abs/2401.05820

5. Seetharamarao, V., Deepak, S. N., Srihari, N., & Kumar, S. (n.d.). IMAGE CLASSIFICATION OF CIFAR10 USING CNN. www.irjmets.com