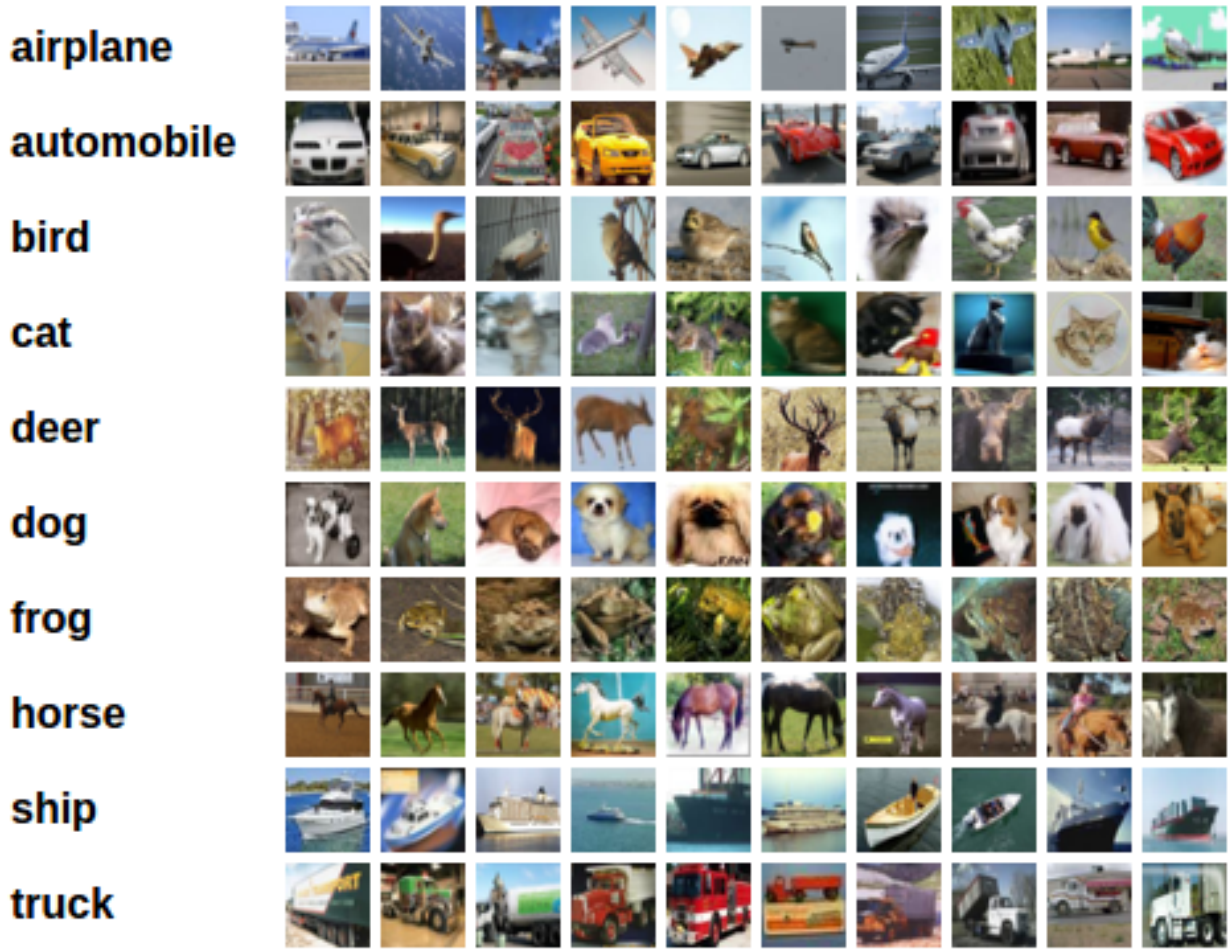
FINAL PROJECT REPORT

Author: Aleti Sai Subhash, NUID: 002986943

INTRODUCTION: In this project, we have tried to classify the CIFAR-10 Dataset using Machine Learning algorithms. I have used Logistic Regression and Deep Neural Network with CNN for classification.
CIFAR-10 is a dataset of 50,000 32x32 colored training images and 10,000 test images, labeled with 10 categories

Here are the classes in the dataset, as well as 10 random images from each class:



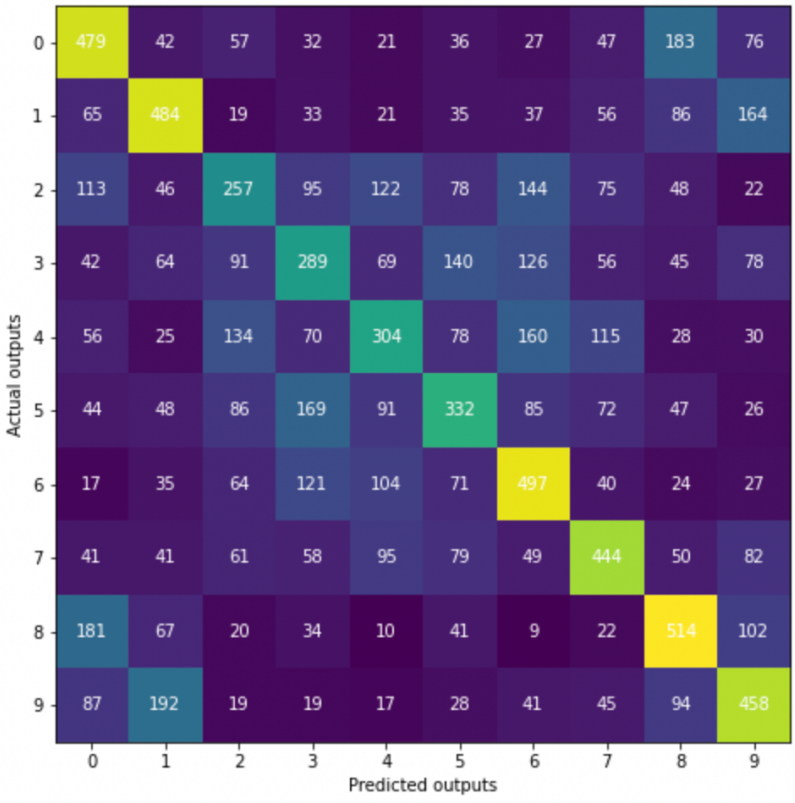
I have used Python to write my code and execute both the algorithms.

ALGORITHMS & RESULTS:

Logistic Regression

This type of statistical analysis is often used for predictive analytics, modeling, and extends to applications in machine learning. In this analytics approach, the dependent variable is finite or categorical. It is used in statistical software to understand the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic regression equation.
In our project, I have used Multinomial Logistic Regression as the classification method.

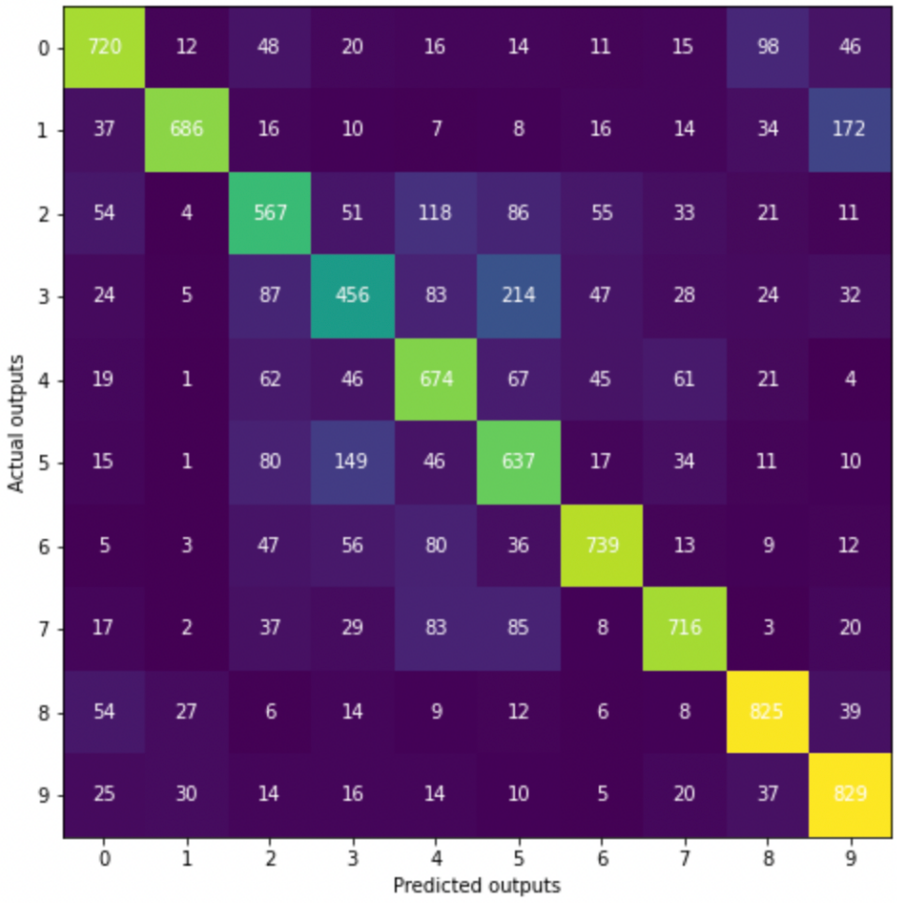
Iterations : 100
Accuracy : 0.4058
Time taken to run : 97 Seconds
Confusion Matrix Graph:



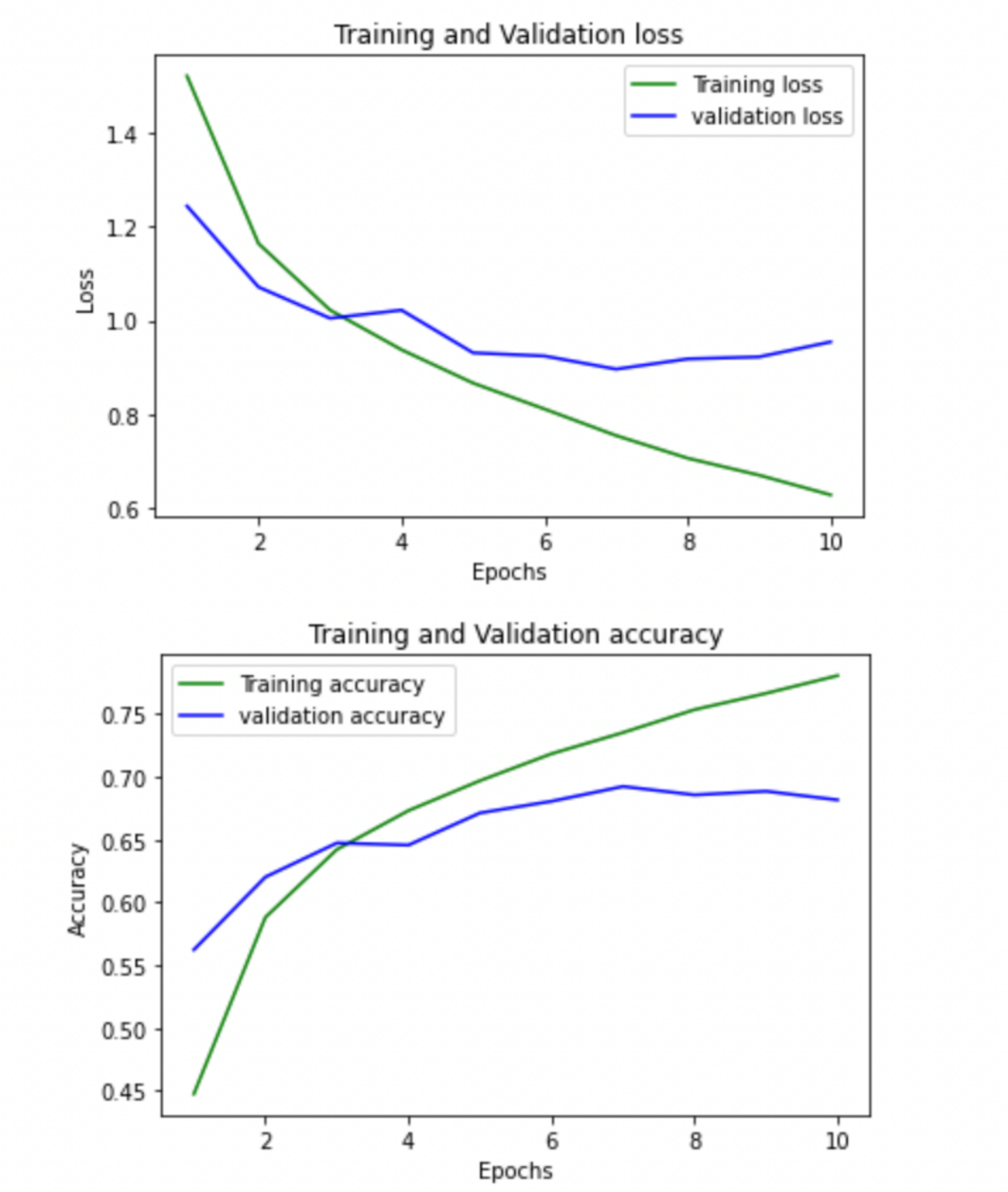
Deep Neural Networks with CNN

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery.
Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

Epochs: 10
Accuracy: 0.6825 – Testing
0.7607 – Training
Time Taken to run: 51 seconds
Confusion Matrix Graph:



Training vs Validation Graphs:



COMPARISION:

CLASSIFIER	EPOCHS	ACCURACY	TIME TAKEN
Logistic Regression	100	40.5%	91 seconds
DNN with CNN	10	68.25%	51 seconds

CONCLUSION: From the above tabular values, it is obvious that the Deep Neural Network has performed better when compared to Logistic regression for the CIFAR-10 dataset.

The accuracy of the DNN with CNN will improve to almost 80% if the number of epochs is increased while running the model fitting.

CODE

Original file is located at
https://colab.research.google.com/drive/1VQ_fGW0AlSEkEtgIMzYet-tb939WDSvR?usp=sharing

```
import keras
from keras.datasets import cifar10
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np

# define num_class
```

```
num_classes = 10

# load dataset keras will download cifar-10 datset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

classes =
["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]

import matplotlib.pyplot as plt

plt.imshow(x_train[1])

plt.imshow(x_train[19])

x_train[0]

"""**Logistic Regression**"""

x_train=x_train.reshape(x_train.shape[0],-1)
x_test=x_test.reshape(x_test.shape[0],-1)
y_train = y_train.reshape(-1,)
y_test = y_test.reshape(-1,)

from sklearn.linear_model import LogisticRegression
import time
lr = LogisticRegression();
tic = time.time()
losses_lr=lr.fit(x_train, y_train)
toc = time.time()

print('testing accuracy= '+ str(lr.score(x_test,y_test)))

print ('Traning time for logistic regression : %f \n' % (toc - tic))
```

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, lr.predict(x_test))
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cm)
ax.grid(False)
ax.set_xlabel('Predicted outputs', color='black')
ax.set_ylabel('Actual outputs', color='black')
ax.xaxis.set(ticks=range(10))
ax.yaxis.set(ticks=range(10))
ax.set_ylim(9.5, -0.5)
for i in range(10):
   for j in range(10):
      ax.text(j, i, cm[i, j], ha='center', va='center', color='white')
plt.show()

"""**Deep Neural Network with CNN**"""

(x_train, y_train), (x_test, y_test) = cifar10.load_data()

y_train = y_train.reshape(-1,)
y_test = y_test.reshape(-1,)

x_train = x_train / 255.0
x_test = x_test / 255.0

cnn = models.Sequential([
   layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(32,
32, 3)),
   layers.MaxPooling2D((2, 2)),

   layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
   layers.MaxPooling2D((2, 2)),

   layers.Flatten(),
   layers.Dense(64, activation='relu'),
```

```
   layers.Dense(10, activation='softmax')
])

cnn.compile(optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])

from sklearn.model_selection import train_test_split

x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2,
random_state=1)

history=cnn.fit(x_train, y_train, epochs=10,validation_data=(x_val, y_val))

history2=cnn.evaluate(x_test,y_test)

loss_train = history.history['loss']
loss_val = history.history['val_loss']
epochs = range(1,11)
plt.plot(epochs, loss_train, 'g', label='Training loss')
plt.plot(epochs, loss_val, 'b', label='validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

loss_train = history.history['accuracy']
loss_val = history.history['val_accuracy']
epochs = range(1,11)
plt.plot(epochs, loss_train, 'g', label='Training accuracy')
plt.plot(epochs, loss_val, 'b', label='validation accuracy')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
cm = confusion_matrix(y_test, [np.argmax(element) for element in
cnn.predict(x_test)])
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cm)
ax.grid(False)
ax.set_xlabel('Predicted outputs', color='black')
ax.set_ylabel('Actual outputs', color='black')
ax.xaxis.set(ticks=range(10))
ax.yaxis.set(ticks=range(10))
ax.set_ylim(9.5, -0.5)
for i in range(10):
   for j in range(10):
      ax.text(j, i, cm[i, j], ha='center', va='center', color='white')
plt.show()
```