

Artificial Intelligence

Lab final

Syed Faizan Haider | Sp21-Bse-001 | 15/1/2024

**LAB FINAL**

**Syed Faizan Haider**

**SP21-BSE-001**

**Code Explained:**

This code focuses on the following aspect:

Implement a robust image classification system using CNN to accurately classify images into predefined categories

The data set being used is CIFAR-10. A quick overview about the CIFAR-10 data set. This data set contains 60,000 32x32 images with predefined categories such as airplane, cats and frogs etc.

The model focuses on training its self to recognize a certain image that is provided to it, and provide the class/category to which that image belongs to in the data set.

The model Is trained for 100 epochs (an epoch is one complete pass through the entire training dataset during the training of a model). But to maintain accuracy and save time the model stops the iterations or the epochs when the accuracy value remains constant.

The mode works by using CNN and Adam for the optimization and training. In CNN it utilizes both conv layers and max pool layers with fully connected layers to end with, using RELU AND SOFTMAX as the activation functions.

I’ve made the model in such a way that you can provide it with any image of a class that is mentioned in CIFAR-10 and it will predict the class of the image that you have provided.

The code uses tensor flow so the method Verbose keeps telling the progress of each epoch as it progresses and scans the whole dataset.

That’s the overview of the code. I will try to write down more details with the code.

**CODE:**

**The following code is of the model that is trained and saved.**

import tensorflow as tf

from tensorflow.keras.datasets import cifar10

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.losses import categorical\_crossentropy

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.callbacks import EarlyStopping

import numpy as np

import matplotlib.pyplot as plt

# The following LOC loads the data set that we have to check

(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()

#To normalize the pixel values:

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

# I converted it to binary class matrice because im using softmax.

# activation function for the output layer thats why

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

data\_generator = ImageDataGenerator(

    rotation\_range=20,

    width\_shift\_range=0.2,

    height\_shift\_range=0.2,

    horizontal\_flip=True

)

# CNN Model Architecture

model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 3)),

    MaxPooling2D(2, 2),

    Conv2D(64, (3, 3), activation='relu'),

    MaxPooling2D(2, 2),

    Conv2D(64, (3, 3), activation='relu'),

    Flatten(),

    Dense(64, activation='relu'),

    Dense(10, activation='softmax')

])

# Compile the model

model.compile(optimizer=Adam(),

              loss=categorical\_crossentropy,

              metrics=['accuracy'])

model.summary()

# Define early stopping callback

# This bit of code is to stop the model if we are getting the same accuracy at the end of each epoch

early\_stopping = EarlyStopping(monitor='val\_loss', patience=3, restore\_best\_weights=True)

# The following lines will train the model with early stopping

history = model.fit(data\_generator.flow(x\_train, y\_train, batch\_size=32),

                    epochs=100,

                    validation\_data=(x\_test, y\_test),

                    callbacks=[early\_stopping])

# Evaluate the model to check the accuracy of our model

# the term verbose=2 is just to show the progress while running the code

#verbose 2 means a progress bar for each epoch

test\_loss, test\_acc = model.evaluate(x\_test, y\_test, verbose=2)

print('\nTest accuracy:', test\_acc)

# Finallt we will save the mode with the name

model.save('LAB\_FINAL\_TRANED\_MODEL\_SP21\_BSE\_001.h5')

**This next part is the code where model Is used to predict the class:**

from tensorflow.keras.preprocessing import image

from tensorflow.keras.models import load\_model

import numpy as np

# First we will load the model that we trained earlier

model = load\_model('/content/cifar10\_cnn\_model.h5')

# we load and pre process the image that we want to predict the class of!

img\_path = '/content/trk.jpg'

img = image.load\_img(img\_path, target\_size=(32, 32))

img\_array = image.img\_to\_array(img)

img\_array = np.expand\_dims(img\_array, axis=0)

img\_array /= 255.0

# feed the image to the mode to make the prediction

prediction = model.predict(img\_array)

# in CIFAR-10 the classes are labled with integer values from 0-9 so i've provided the names of each

#class for better understanding!!!

predicted\_class = np.argmax(prediction)

class\_names = ['Airplane', 'Automobile', 'Bird', 'Cat', 'Deer', 'Dog', 'Frog', 'Horse', 'Ship', 'Truck']

# Assuming predicted\_class is the index predicted by the model

if 0 <= predicted\_class < len(class\_names):

    predicted\_class\_name = class\_names[predicted\_class]

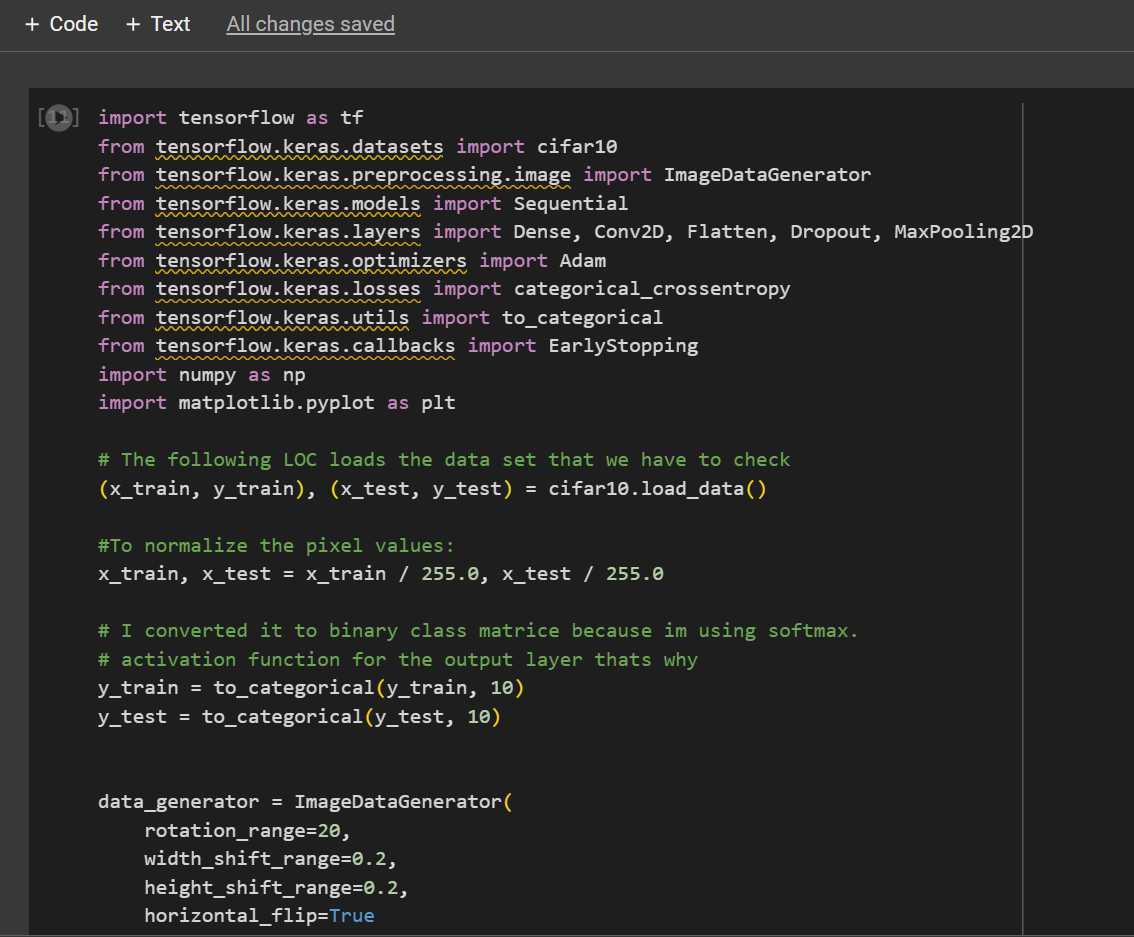
    print(f'Predicted Class: {predicted\_class\_name}')

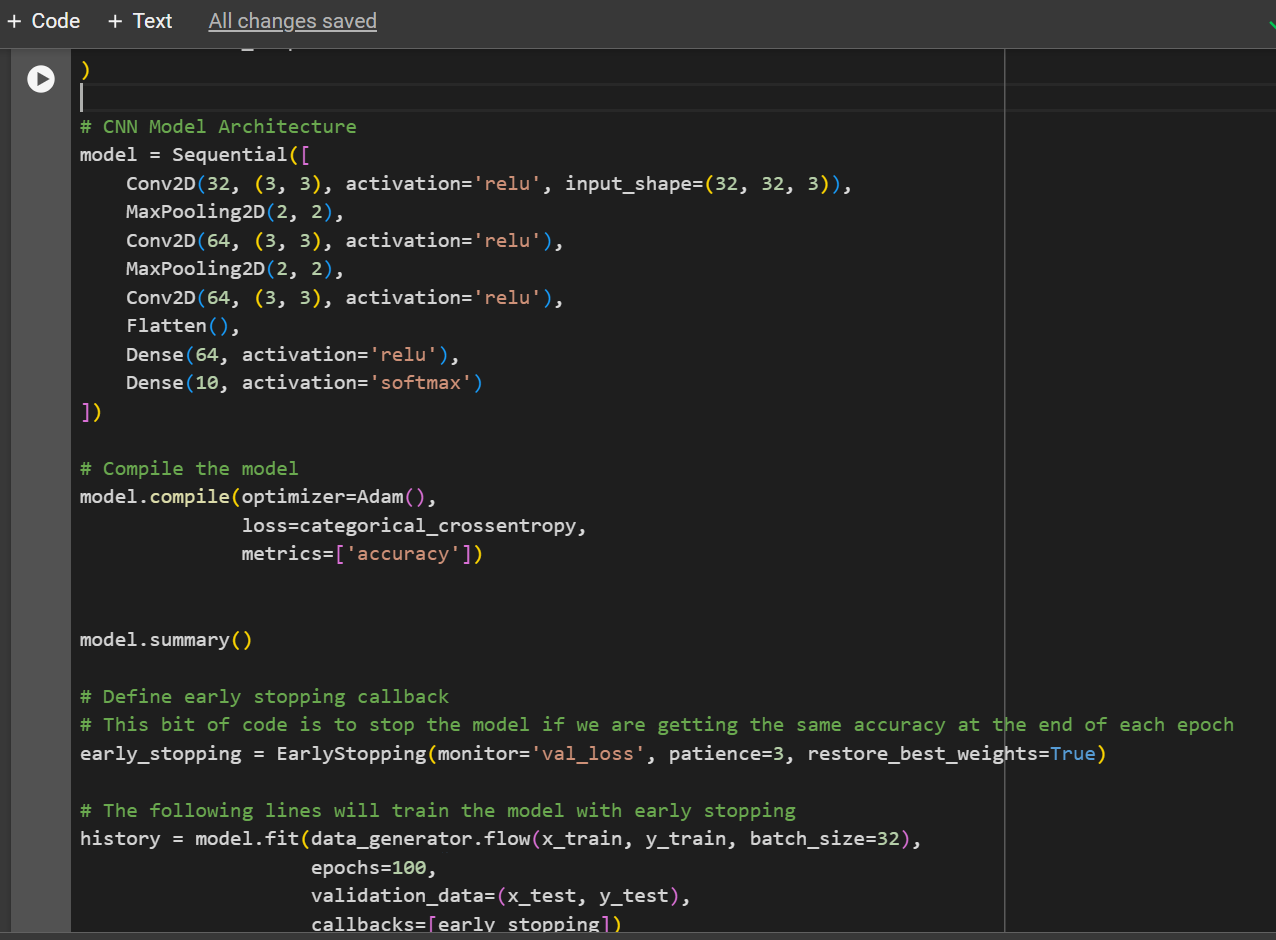
else:

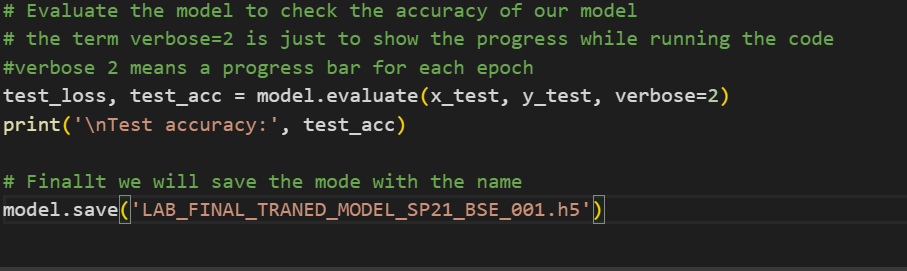
    print(f'Invalid Predicted Class Index: {predicted\_class}')

In this code, two paths are provided the first one is of the Model and the second one is of the input image.

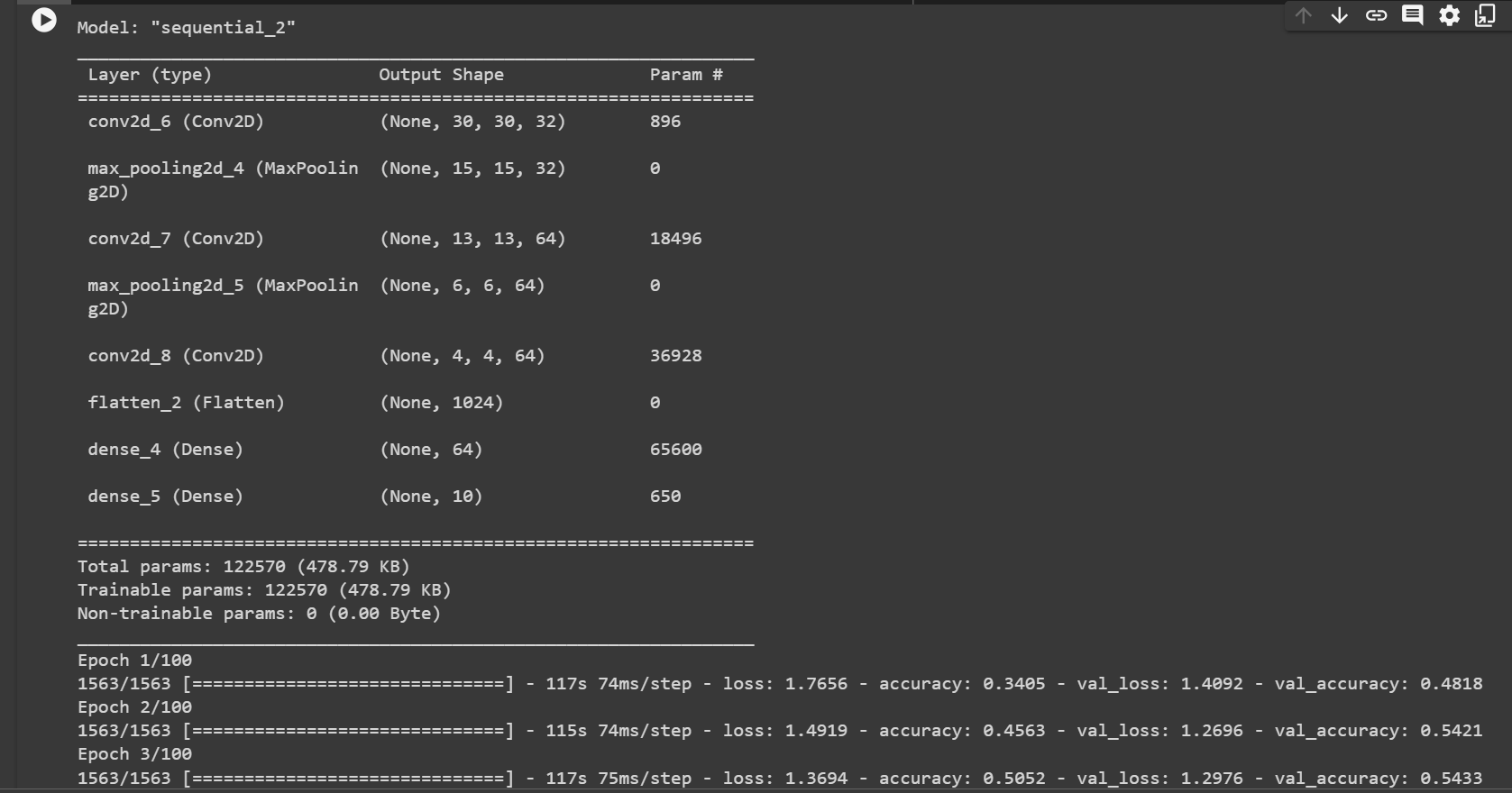
As CIFAR-10 has 0-9 index-based results and each number representing a different category I wrote the code to interpret the number and give us the name of the category using the if statements.

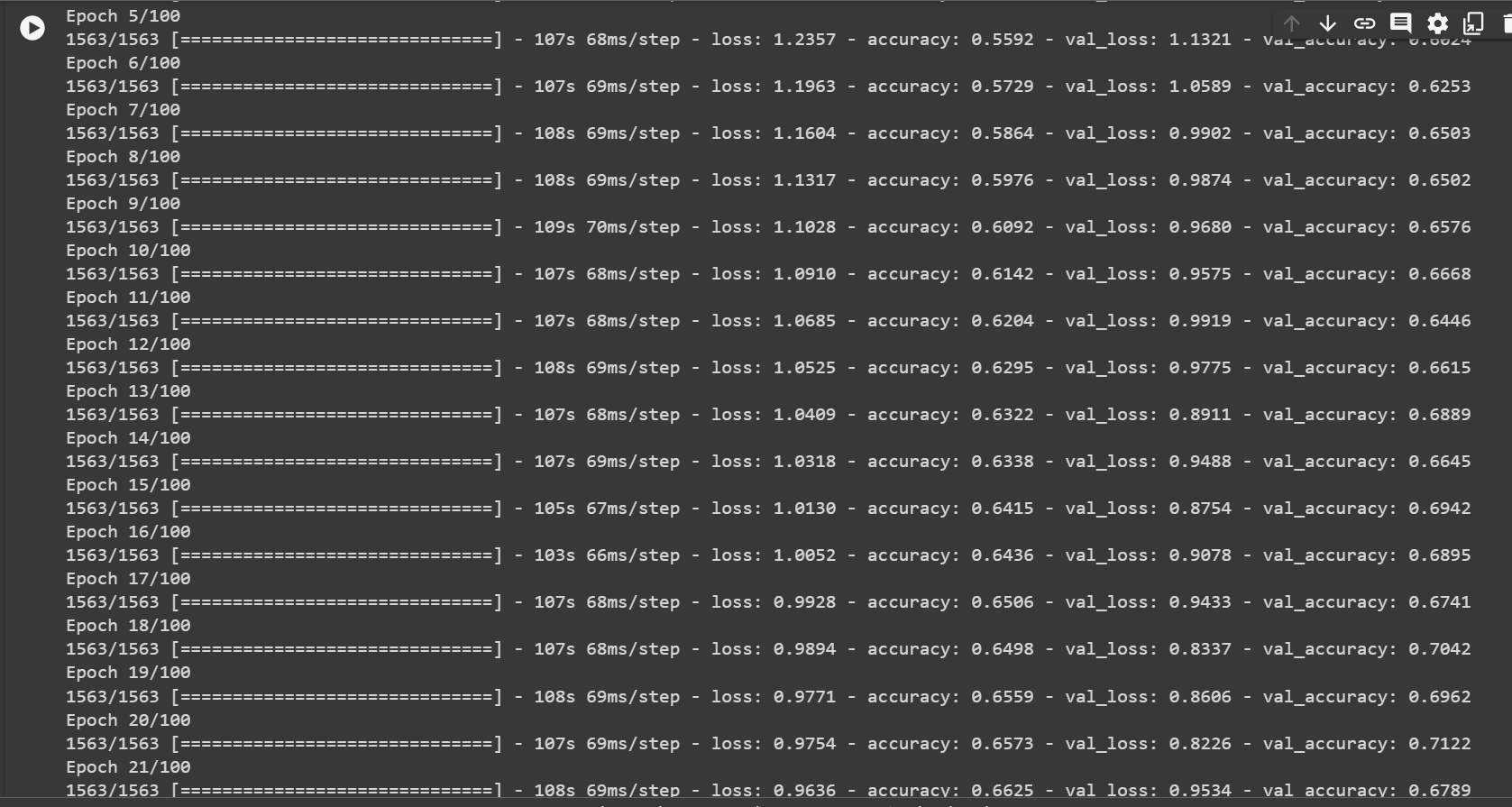
**Screen Shots of Code:** ****

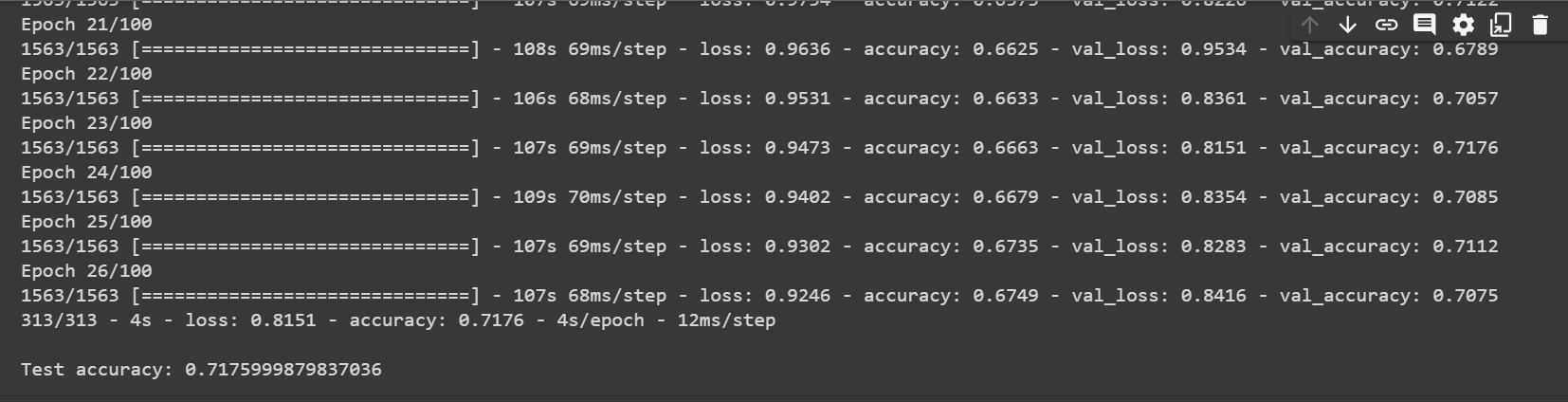
****

****

**OUTPUT:**

****

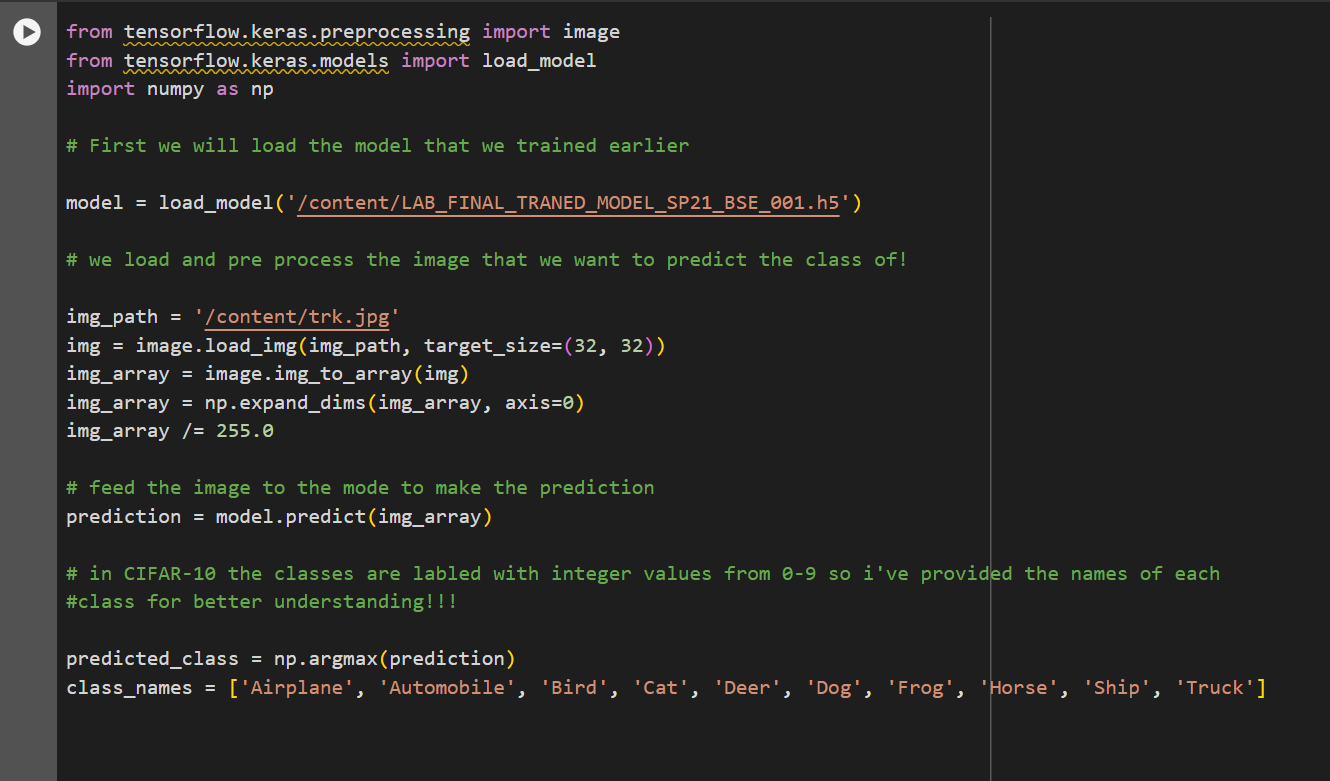
****

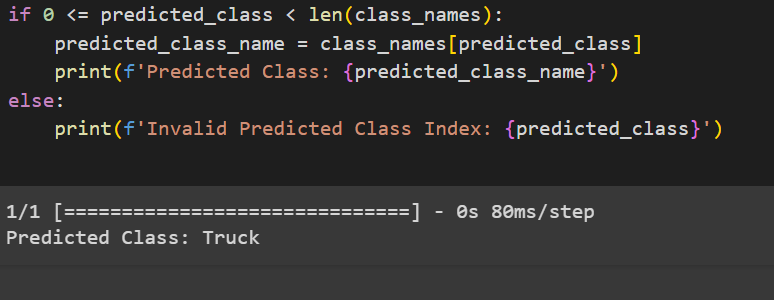
****

As we can see in the above screenshots the model goes to 26 epochs and predicts that no progress is being made and stops further epochs.

With test accuracy of 71% with more epochs or using other Optimizing functions like Stochastic Gradient Descent we can further improve the accuracy of the test. But the question demanded it to be Adam so I’ve used Adam.

The following code is of the part where the mode is used to make predictions.

****

****

In the above screenshot we can see that the model predicted the image to be of a Truck. Which means that the image belongs to the Truck Category.

I provided this image as the input.

**Image**