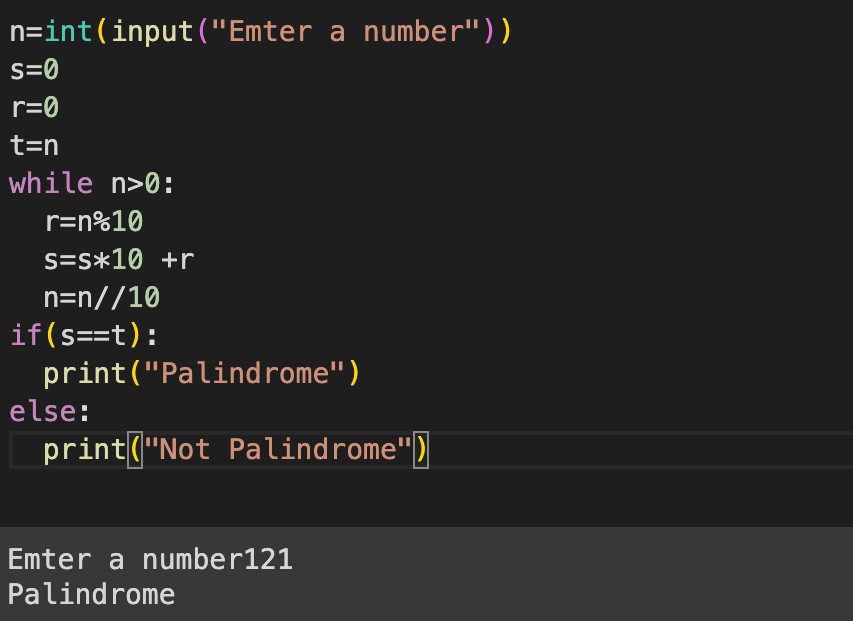


# Experiment 1

## Aim: Setting up the Jupyter IDE Environment and Executing a Python Program

### Procedure to Install Anaconda :

1. Go to the Anaconda Website and chose a Python 3.x graphical installer
2. Locate your download and double click it.
3. Read the license agreement and click on I Agree.
4. Note your installation location and then click Next
5. This is an important part of the installation process. The recommended approach is to not check the box to add Anaconda to your path. This means you will have to use Anaconda Navigator or the Anaconda Command Prompt when you wish to use Anaconda. If you want to be able to use Anaconda in your command prompt please use the alternative approach and check the box.
6. This is an optional step. This is for the case where you didn’t check the box in step 5 and now want to add Anaconda to your path in the environment variables

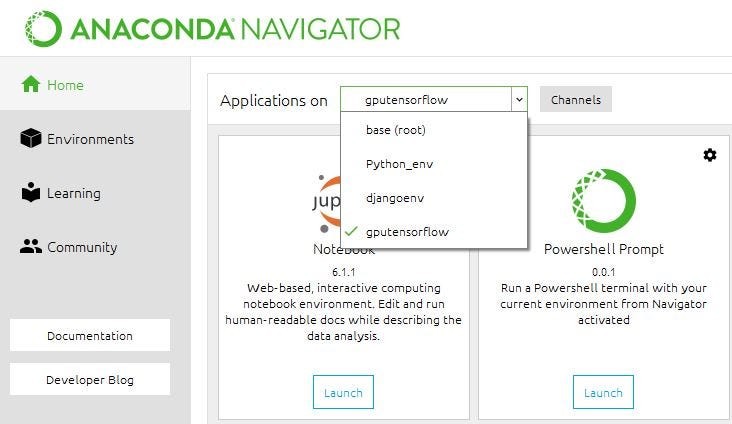


## Integrating Jupyter with Anaconda:

1. Find and open the Anaconda Prompt app using the search bar.
2. Once the Anaconda Prompt app opens, navigate to the desired folder, using the cd command.
3. Once in the desired folder, type Jupyter notebook followed by the Enter key.
4. The Jupyter server will start. You should see some server logs printed. You may be prompted to select an application to open Jupyter in. Firefox or Chrome are preferred.
5. Shortly after, a browser window should open, showing the files and folders located in the folder where you started the Jupyter server.

Executing a Python Program:

**Result: We have successfully installed Anaconda and set up the Jupyter IDE and have executed a Python program to check whether an input number is Palindrome or not.**



# Experiment 2

## Aim: Installing Tensor flow and PyTorch Libraries and make use of them

### Procedure to install Tensor-flow in Anaconda:

Tensor-flow with conda is supported on 64-bit Windows 7 or later, 64-bit Ubuntu, Linux 14.04 or later, 64 bit CentOS Linux 6 or later and macOs 10.10 or later.

1. On Windows open the Start menu and open Anaconda Command Prompt.
2. Choose a name for your TensorFlow environment, such as “tf"
3. To install the current release of CPU-only TensorFlow, recommended for beginners. conda create -n tf tensorflow

conda activate tf

1. Or, to install the current release of GPU TensorFlow on Linux or conda create Windows: conda create-n tf-gpu tensorflow-gpu

conda activate tf-gpu

1. Now go to Anaconda Navigator and change the environment to tf-gpu from base.
2. Install Jupyter notebook and launch Jupyter in the new environment
3. Install numpy using pip install numpy==1.23.4
4. Now import tensorflow and check the version import tensorflow as tf

print(tf. version )

2.6.0

1. Check the keras version using the following command

!pip show keras

Name: keras Version: 2.13.1

Summary: Deep learning for humans. Home -page: https://keras.io/

Author: Keras team

Author-email: keras [-users@googlegroups.com](mailto:-users@googlegroups.com) License: Apache 2.0

Location: c:\users\mgit\anaconda3\envs\tf- gpu\lib\site-packages Requires:

Required-by: tensorflow Example program for Tensorflow basics

import tensorflow as tf

x= tf.constant ([[1., 2., 3.],[4., 5., 6.]])

print(x) print(x.shape) print (x.dtype)

**Output:** tf.Tensor([[1., 2., 3.][4., 5., 6.]],

shape=(2, 3), dtype=float32)

(2,3)

<dtype: ‘float32'>

Installing Pytorch and importing it in Jupyter notebook

1. Use the command pip3 instal torch torch vision torch audio in anaconda command prompt to install pytorch.
2. Now import torch in Jupyter notebook
3. Write an example program in Jupyter

import torch

x = torch.rand(5, 3) print (x)

**Output:** tensor([[0.8338, 0.2921, 0.2501],

[0.8172, .9531, 0.9061],

[0.4925, .0952, 0.3532],

[0.3888, 0.7118, 0.3312],

[0.4027, 0.3560, 0.8726]])

**Result: We have successfully installed Tensorflow and Keras and executed simple programs**

# Experiment 3

## Aim: Applying Convolutional Neural Network on Computer Vision Algorithms

#### Importing The Libraries

import os

import numpy as np import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D from tensorflow.keras.optimizers import Adam

#### Loading and Resizing the training datasets of dogs and cats

import PIL import os import os.path

from PIL import Image

f= r'C:\Users MGIT\Desktop\cd dataset\train\dog’ for file in os.listdir(f):

f\_img= f+ “/" +file img=Image.open(f\_img) img=img.resize((112, 112)) img.save(f\_img)

import PIL import os import os.path

from PIL import Image

f= r'C:\Users MGIT\Desktop\cd dataset\train\cat’

for file in os.listdir(f): f\_img= f+ “/" +file

img=Image.open(f\_img) img=img.resize((112, 112)) img.save(f\_img)

#### Loading and Resizing the testing datasets of dogs and cats

import PIL

import os import os.path

from PIL import Image

f= r'C:\Users MGIT\Desktop\cd dataset\test\dog’ for file in os.listdir(f):

f\_img= f+ “/" +file img=Image.open(f\_img) img=img.resize((112, 112)) img.save(f\_img)

import PIL import os import os.path

from PIL import Image

f= r'C:\Users MGIT\Desktop\cd dataset\test\cat’ for file in os.listdir(f):

f\_img= f+ “/" +file img=Image.open(f\_img) img=img.resize((112, 112)) img.save(f\_img)

#### Image Preprocessing

IMAGE\_SIZE = 112

BATCH\_SIZE= 32

train\_data\_size = 180

test\_data = 20

train= tf.keras.preprocessing.image. ImageDataGenerator(rescale=1./255, rotation\_range = 90, shear\_range =0.2, zoom\_range = 0.2, horizontal\_flip = True,)

**Output**: Found 180 images belonging to 2 classes

test= tf.keras.preprocessing.image. ImageDataGenerator(rescale=1./255, rotation\_range = 90, shear\_range =0.2, zoom\_range = 0.2, horizontal\_flip = True,)

**Output**: Found 20 images belonging to 2 classes

#### Model Building

model= Sequential([

Conv2D(32,(3,3),activation='relu', input\_shape=(112,112,3)), MaxPool2D(2,2),

Conv2D(32,(3,3),activation=‘relu' ,input\_shape=(112, 112, 3)), MaxPool2D(2,2),

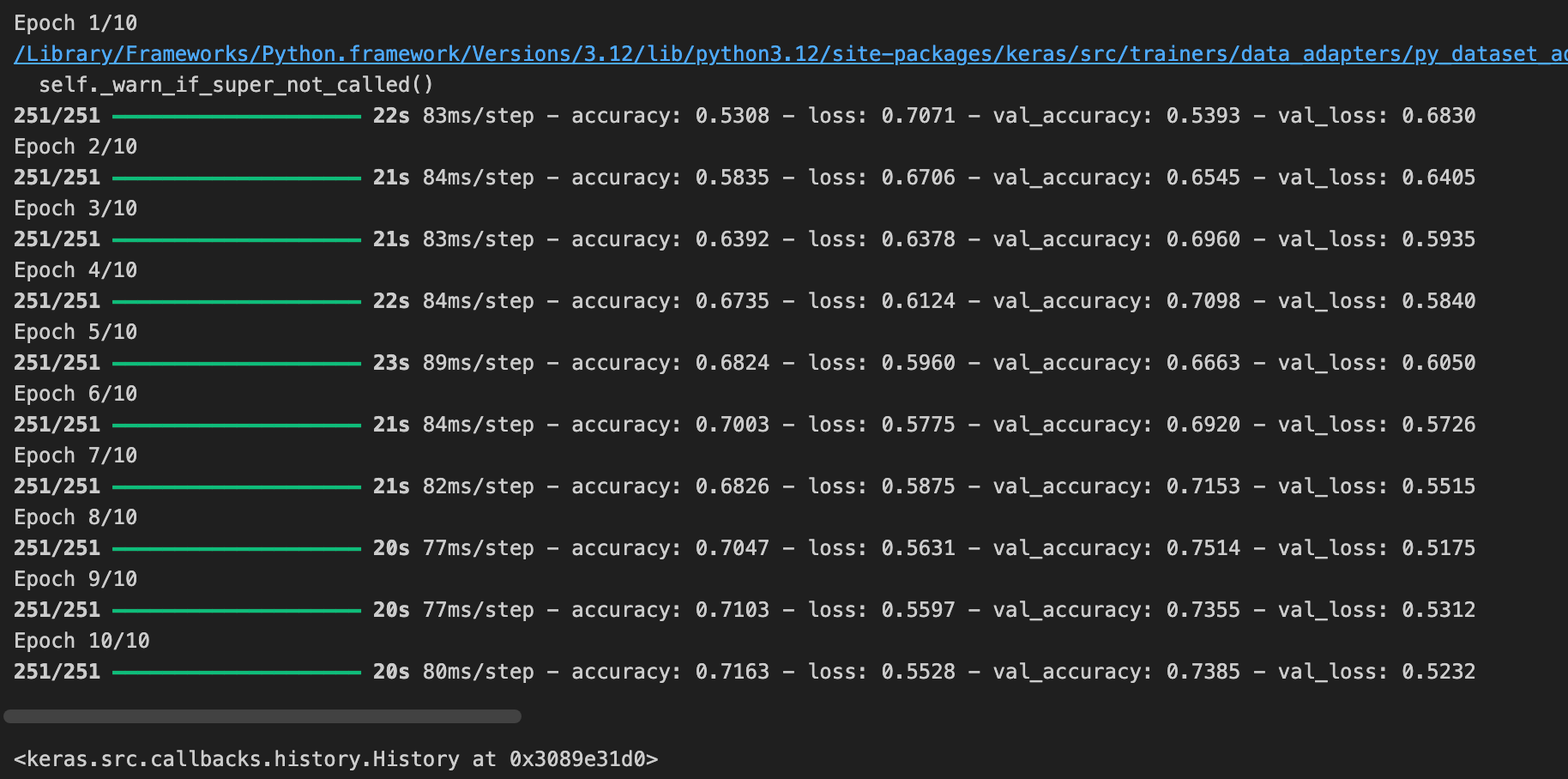
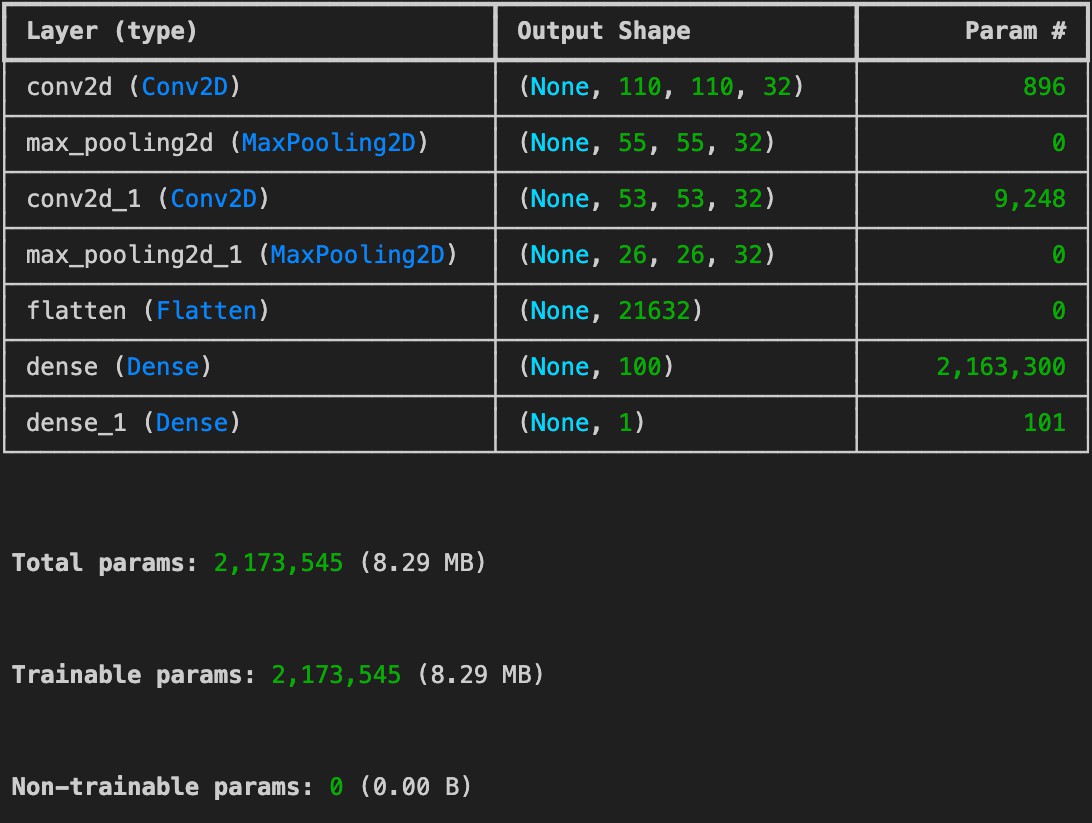
Flatten(),

Dense(100, activation=‘relu' ), Dense(1, activation=‘sigmoid')

]

)

model.summary()



model.compile(‘Adam’, ‘binary\_crossentropy’, [‘accuracy’]) model.fit(train\_data, epochs=10, validation\_data=test\_data)

**Result:Trained a neural network model to classify the dogs and cats images**

# Experiment 4

## Aim: Image Classification on MNIST dataset (CNN model with Fully connected Layer)

#### Importing The Libraries

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D from tensorflow.keras.optimizers import Adam

#### PreProcessing and Loading Images

image\_size = 64

batch\_size = 32

train = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255, rotation\_range=90, shear\_range=0.2, zoom\_range=0.2, horizontal\_flip=True)

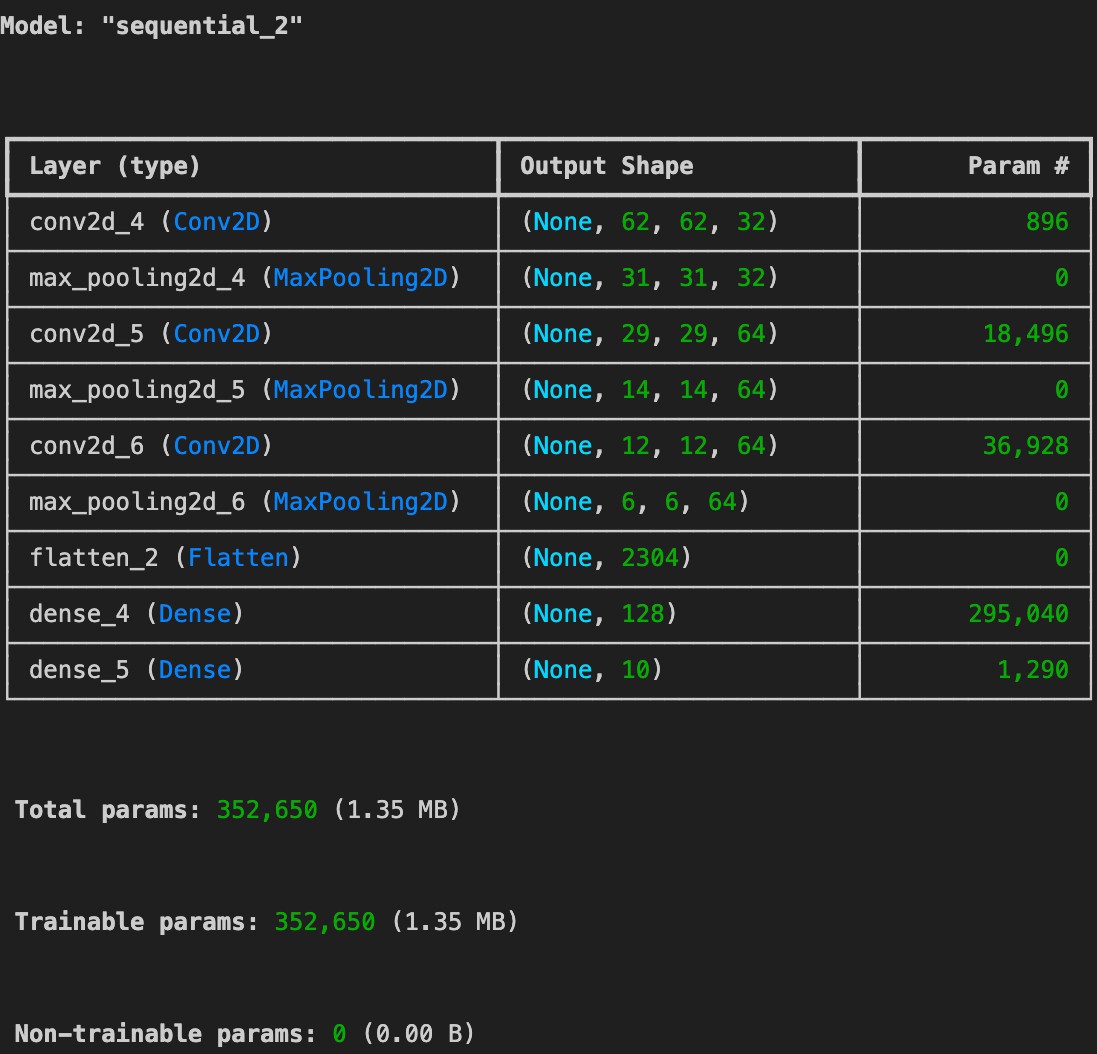
test = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255) train\_set = train.flow\_from\_directory(r’ ./dataset\_mnist/train’,

target\_size=(image\_size, image\_size), batch\_size= batch\_size, class\_mode=‘categorical’)

test\_set= test.flow\_from\_directory( r’ ./dataset\_mnist/test’, target\_size=(image\_size, image\_size), batch\_size= batch\_size,

class\_mode=‘categorical’)

**Output:** Found 100 images belonging to 10 classes Found 100 images belonging to 10 classes



#### Model Building

model= Sequential([

Conv2D(32,(3,3),activation='relu', input\_shape=(112,112,3)), MaxPool2D(2,2),

Conv2D(64,(3,3),activation=‘relu'), MaxPool2D(2,2), Conv2D(64,(3,3),activation=‘relu'), MaxPool2D(2,2),

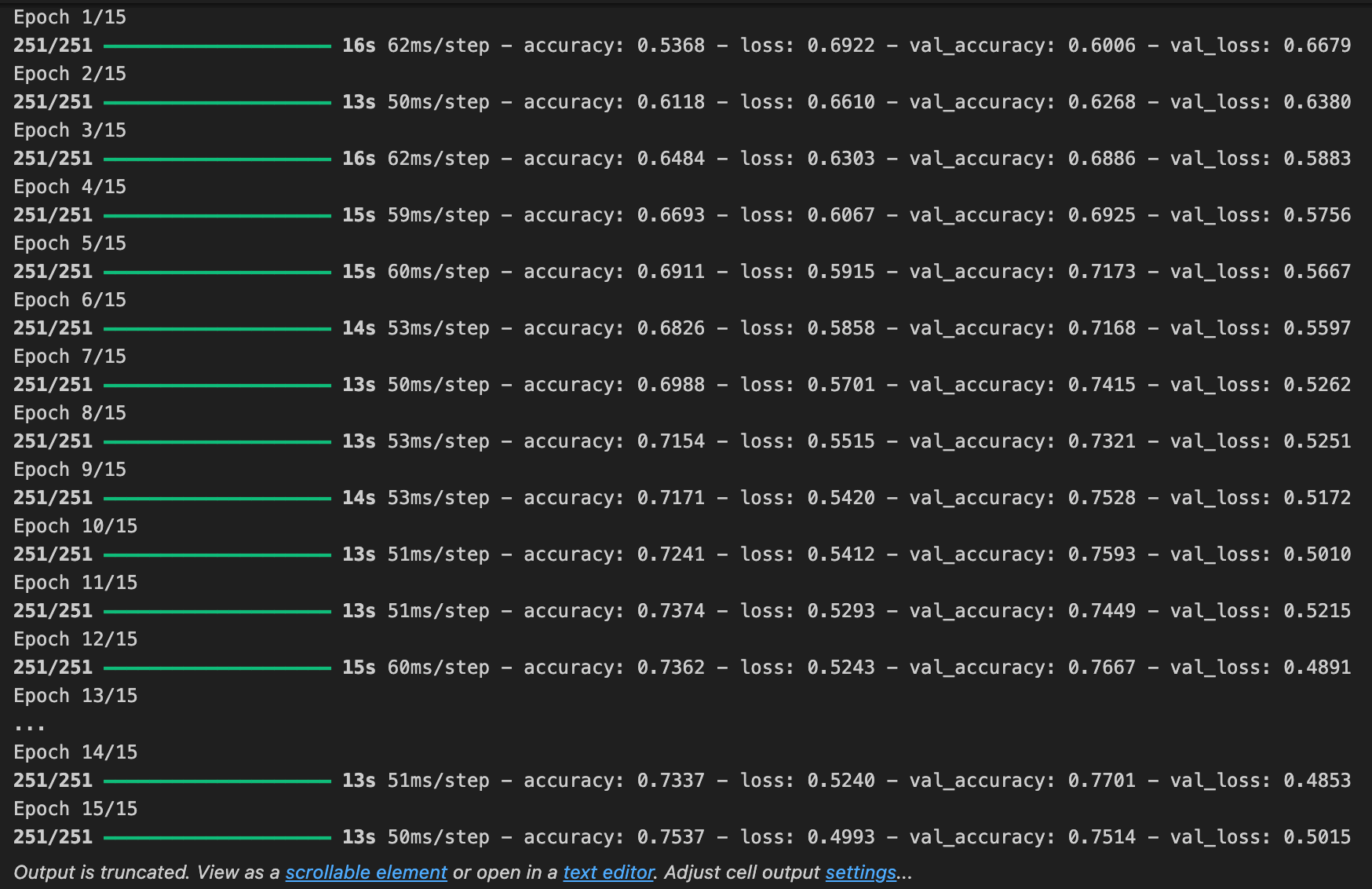
Flatten(),

Dense(100, activation=‘relu' ), Dense(1, activation=‘sigmoid')

]

)

model.summary()



model.compile(optimizer=Adam(), loss='binary\_crossentropy', metrics=[‘accuracy']) model.fit(train\_data, epochs=15, validation\_data=test\_data)

**Result: Performed Image classification on MNIST Dataset for numeric digits from 0 to 9**

# Experiment 5

## Aim: Applying the pre-trained model VGG16 for MNIST Dataset Classification

#### Importing The Libraries

from tensorflow import keras

from keras.models import Sequential, Model

from keras. layers import Input, Dense, Dropout, Flatten,Conv2D, MaxPoo12D from keras.layers import BatchNormalization

#### Loading the dataset

image\_size = 64

batch\_size = 32

train = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255, rotation\_range=90, shear\_range=0.2, zoom\_range=0.2, horizontal\_flip=True)

test = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

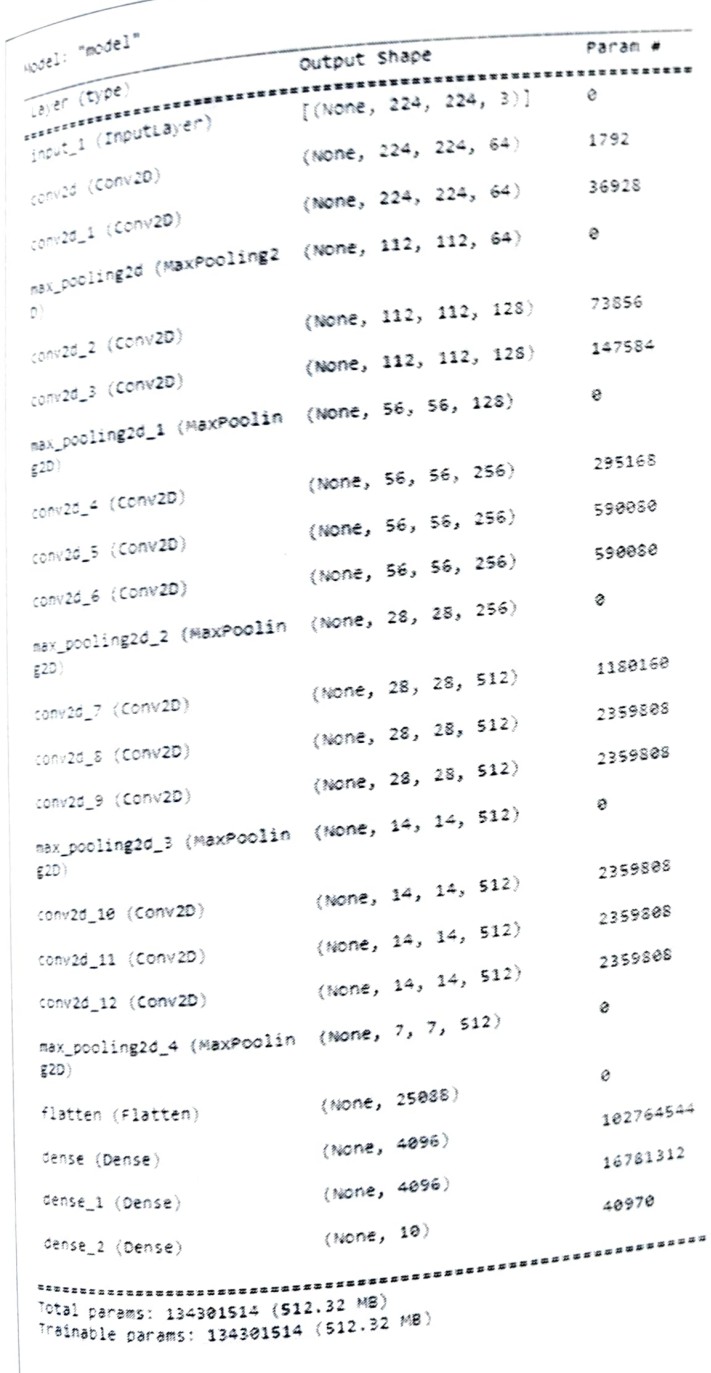
**Output:** Found 100 images belonging to 10 classes Found 100 images belonging to 10 classes

input = Input (shape (224, 224,3))

x=Conv2D (filters=64, kernel\_size=3, padding=‘same’ , activation=‘relu’)(input) x=Conv2D (filters=64, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=MaxPool2D (pool\_size=2, strides=2 ,padding=‘same’ )(x)

x=Conv2D (filters=128, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=Conv2D (filters=128, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=MaxPool2D (pool\_size=2, strides=2 ,padding=‘same’ )(x)

x=Conv2D (filters=256, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=Conv2D (filters=256, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=Conv2D (filters=256, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=MaxPool2D (pool\_size=2, strides=2 ,padding=‘same’ )(x)

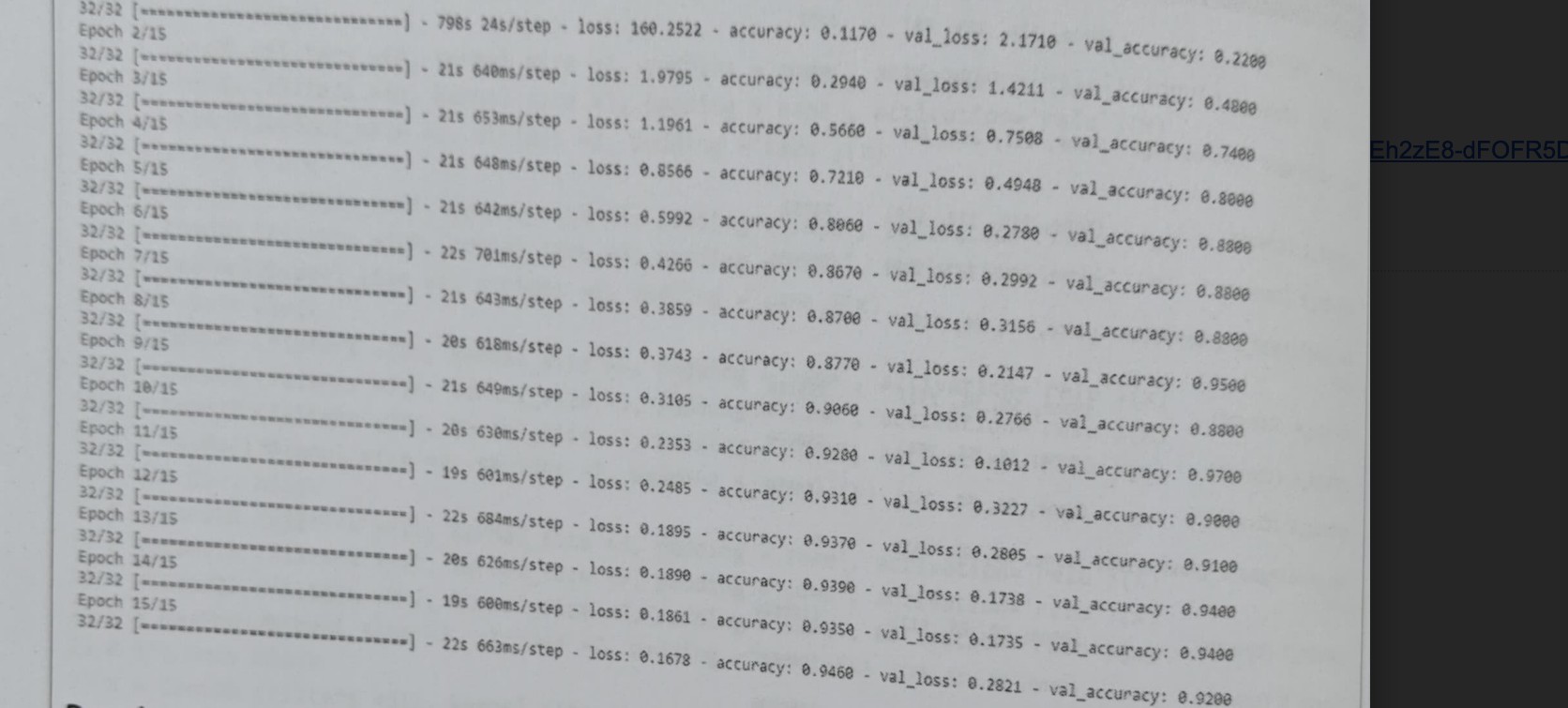


x=Conv2D (filters=512, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=Conv2D (filters=512, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=Conv2D (filters=512, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=MaxPool2D (pool\_size=2, strides=2 ,padding=‘same’ )(x)

x=Conv2D (filters=256, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=Conv2D (filters=256, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=Conv2D (filters=256, kernel\_size=3, padding=‘same’ , activation=‘relu’)(x) x=MaxPool2D (pool\_size=2, strides=2 ,padding=‘same’ )(x)

x = Flatten() (x)

x = Dense (units = 4096, activation =‘relu')(x) output=Dense(units=10, activation=‘softmax’)(x) model = Model (inputs=input, outputs =output) model.summary()



model.compile(loss=“categorical\_crossentropy”, optimizer=‘Adam’,metrics=[‘accuracy’]) history= model.fit(train\_data, epochs=15, validation\_data=test\_data)

**Result:Successfully implemented the pre-trained CNN model VGG16 for MNIST Dataset Classification**