Parshvanath Charitable Trust's

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

(All Programs Accredited by NBA)

Department of Information Technology

Handwritten Digit Recognition

Ruchita Raut	19104033
Kushal Todi	19104047
Pratik Dhumal	19104031

Contents

- Introduction
- Technology Stack
- Implementation
- Conclusion

1. Introduction

- Handwritten Digit Recognition is the ability of computers to recognize human handwritten digits.
- It is not an easy task for the machines, because handwritten digits are not perfect and can be made with many different flavors.
- Handwritten digit recognition is the solution to this problem which uses the image of a digit and recognize the digit present in the image.

- **Problem Identified :**
  - Time was spent on recognizing digits manually, which could be automated with great accuracy.
- **Solution Proposed:**
  - A model can be developed to recognize the human handwritten digits effectively.
  - To predicts the actual handwritten digit considering the shape and the images of the same.

2. Technology Stack

Technologies Used: -

1. Python

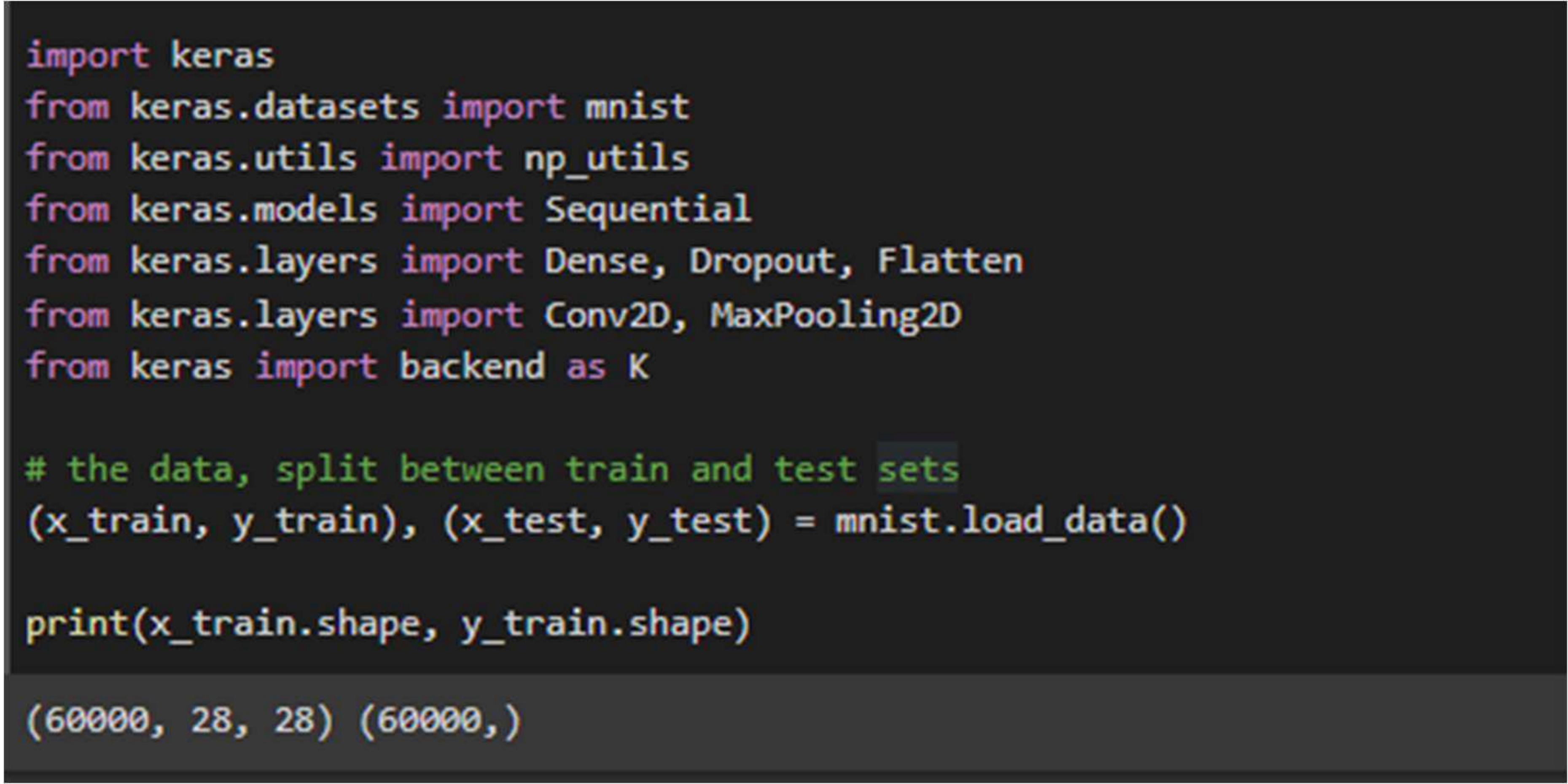
2. Tkinter

3. Numpy

4. Keras

3. Implementation

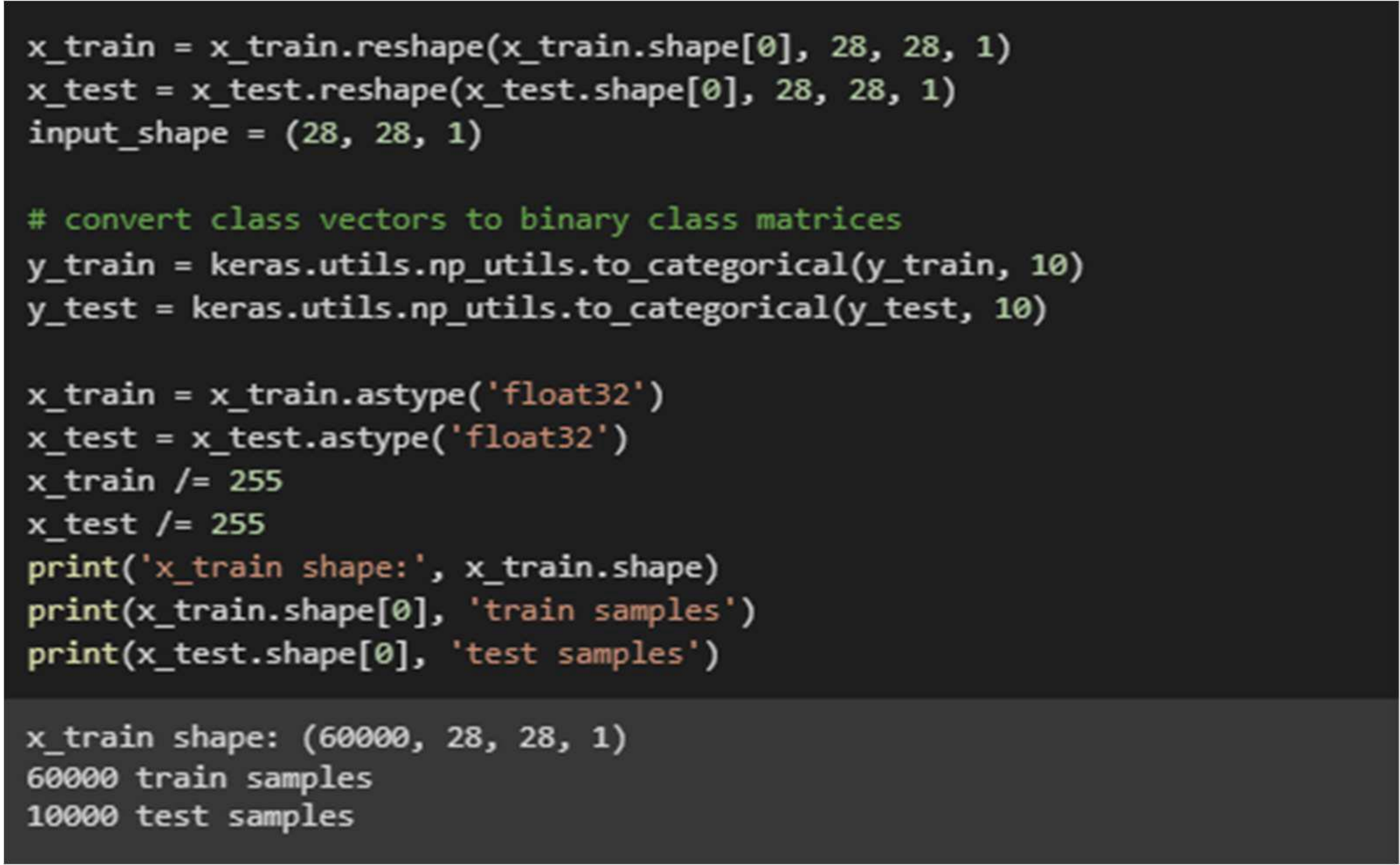
- **Import the Dataset**



```
import keras
from keras.datasets import mnist
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(x_train.shape, y_train.shape)
```

```
(60000, 28, 28) (60000,)
```

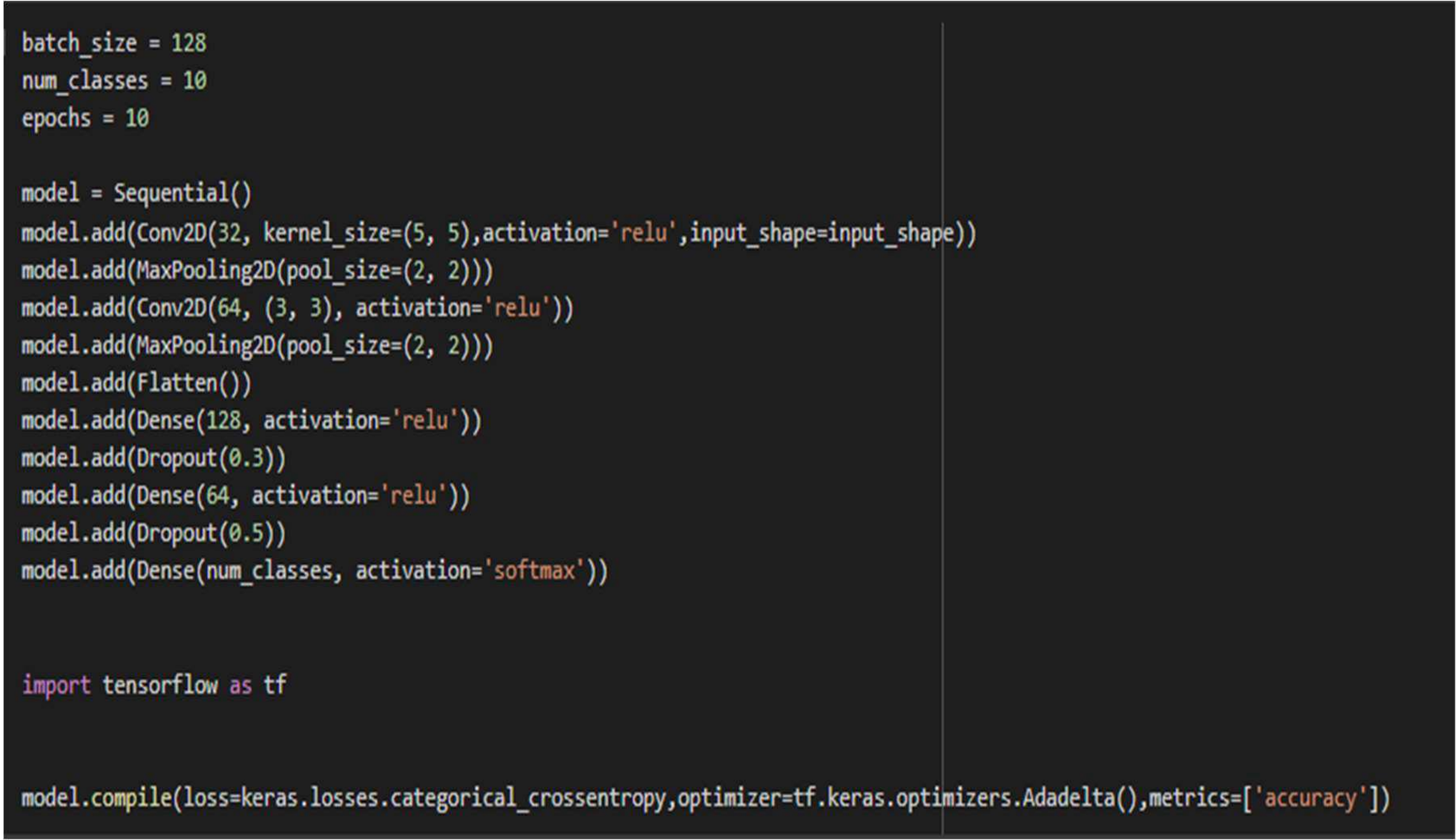
- **Pre-processing the data**



```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)

# convert class vectors to binary class matrices
y_train = keras.utils.np_utils.to_categorical(y_train, 10)
y_test = keras.utils.np_utils.to_categorical(y_test, 10)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

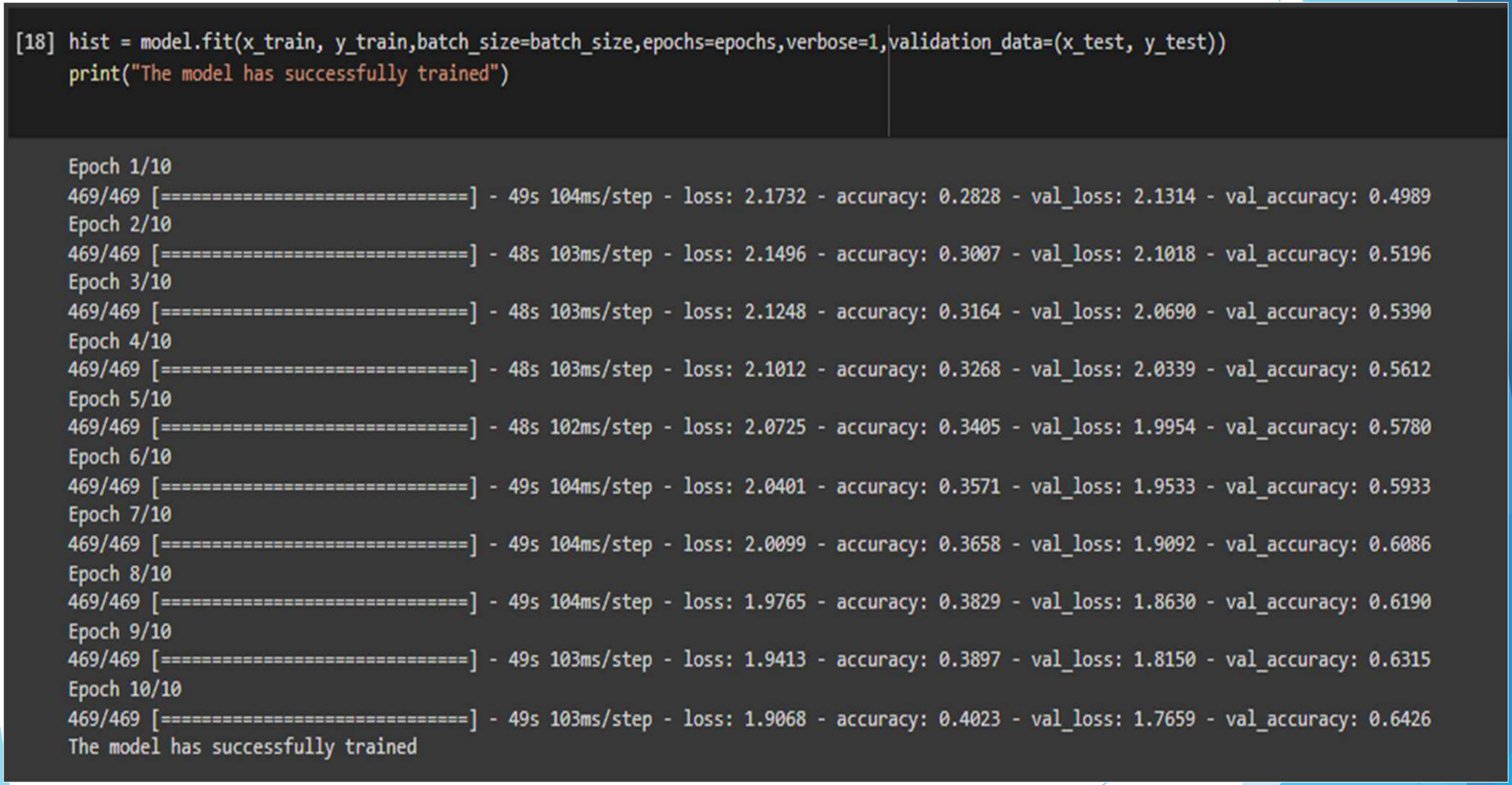
```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

- **Creating the Model**



```
batch_size = 128
num_classes = 10
epochs = 10

model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))


import tensorflow as tf


model.compile(loss=keras.losses.categorical_crossentropy,optimizer=tf.keras.optimizers.Adadelta(),metrics=['accuracy'])
```

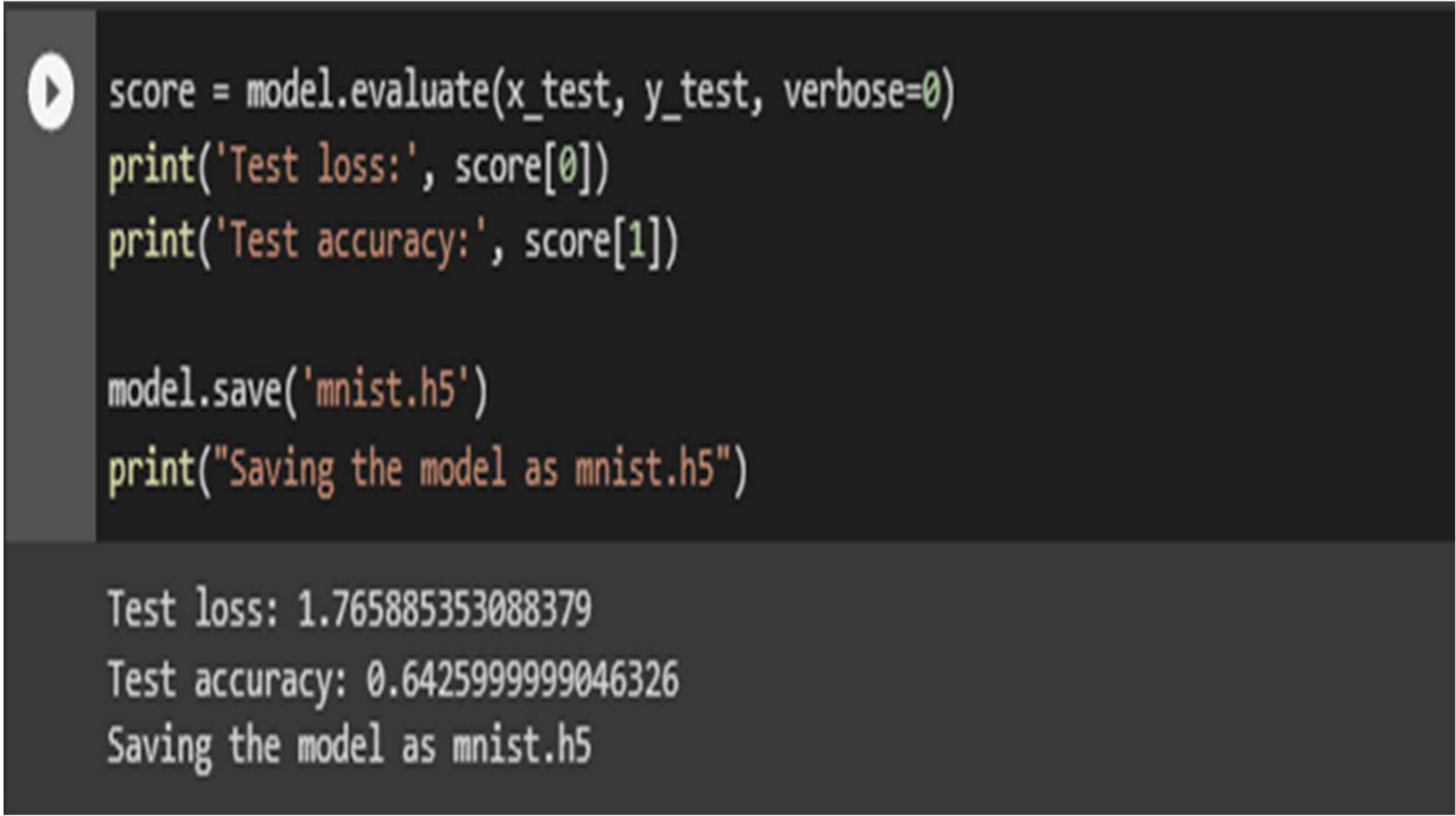
- **Training the Model**



```
[18] hist = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(x_test, y_test))
     print("The model has successfully trained")
```

```
Epoch 1/10
469/469 [==============================] - 49s 104ms/step - loss: 2.1732 - accuracy: 0.2828 - val_loss: 2.1314 - val_accuracy: 0.4989
Epoch 2/10
469/469 [==============================] - 48s 103ms/step - loss: 2.1496 - accuracy: 0.3007 - val_loss: 2.1018 - val_accuracy: 0.5196
Epoch 3/10
469/469 [==============================] - 48s 103ms/step - loss: 2.1248 - accuracy: 0.3164 - val_loss: 2.0690 - val_accuracy: 0.5390
Epoch 4/10
469/469 [==============================] - 48s 103ms/step - loss: 2.1012 - accuracy: 0.3268 - val_loss: 2.0339 - val_accuracy: 0.5612
Epoch 5/10
469/469 [==============================] - 48s 102ms/step - loss: 2.0725 - accuracy: 0.3405 - val_loss: 1.9954 - val_accuracy: 0.5780
Epoch 6/10
469/469 [==============================] - 49s 104ms/step - loss: 2.0401 - accuracy: 0.3571 - val_loss: 1.9533 - val_accuracy: 0.5933
Epoch 7/10
469/469 [==============================] - 49s 104ms/step - loss: 2.0099 - accuracy: 0.3658 - val_loss: 1.9092 - val_accuracy: 0.6086
Epoch 8/10
469/469 [==============================] - 49s 104ms/step - loss: 1.9765 - accuracy: 0.3829 - val_loss: 1.8630 - val_accuracy: 0.6190
Epoch 9/10
469/469 [==============================] - 49s 103ms/step - loss: 1.9413 - accuracy: 0.3897 - val_loss: 1.8150 - val_accuracy: 0.6315
Epoch 10/10
469/469 [==============================] - 49s 103ms/step - loss: 1.9068 - accuracy: 0.4023 - val_loss: 1.7659 - val_accuracy: 0.6426
The model has successfully trained
```

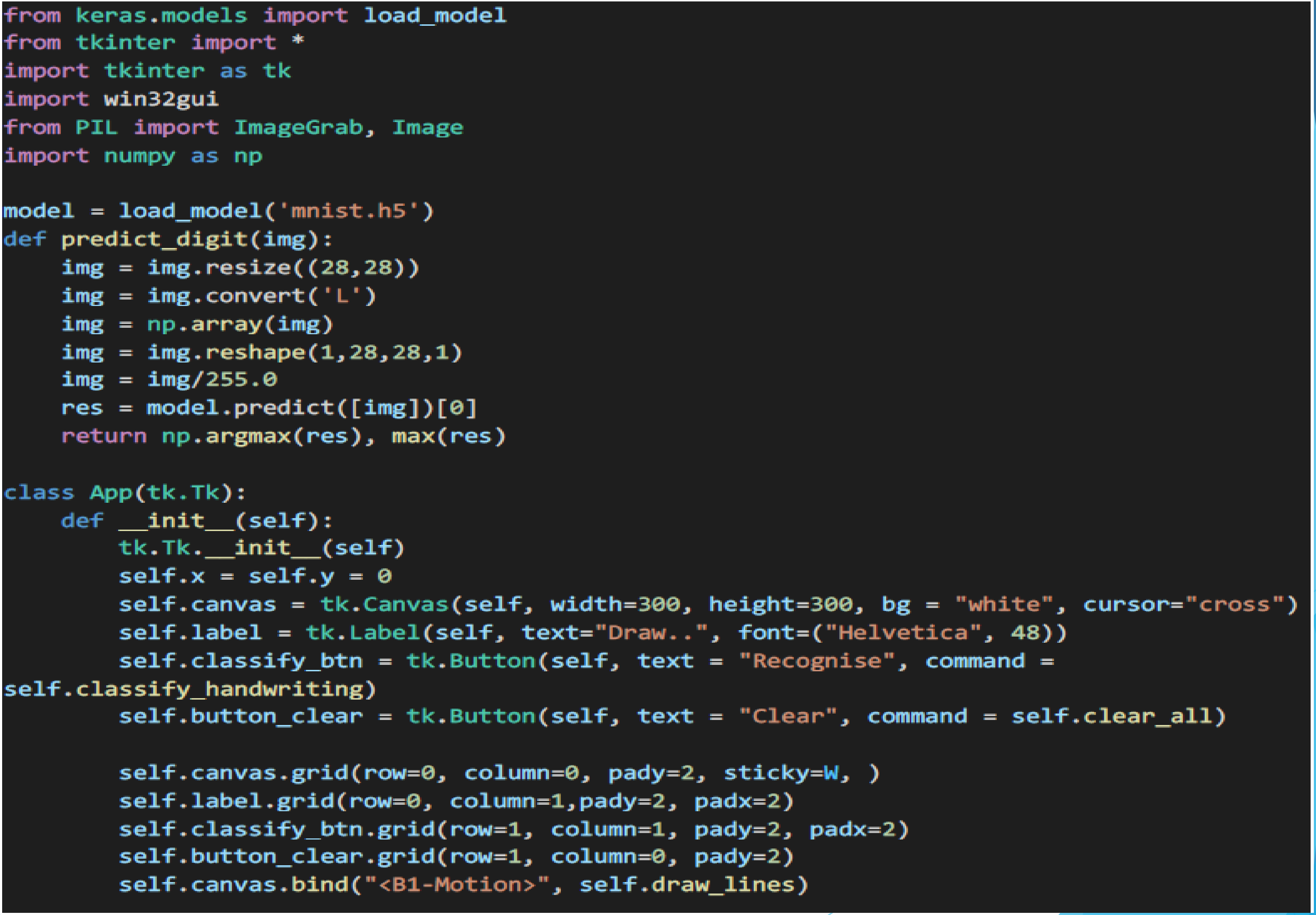
- **Evaluating the Model**



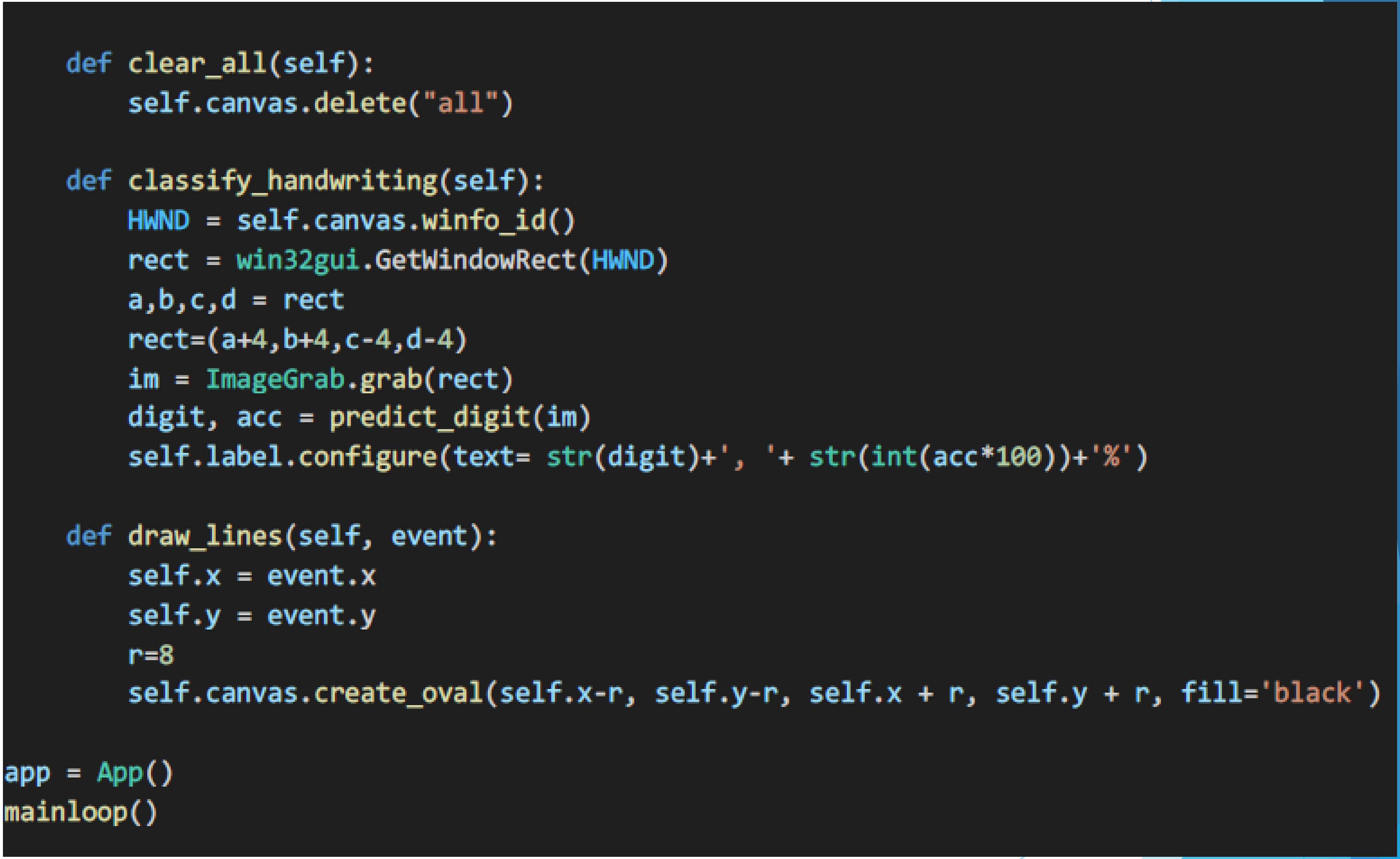
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

model.save('mnist.h5')
print("Saving the model as mnist.h5")
```

```
Test loss: 1.765885353088379
Test accuracy: 0.6425999999046326
Saving the model as mnist.h5
```

- # Creating the GUI

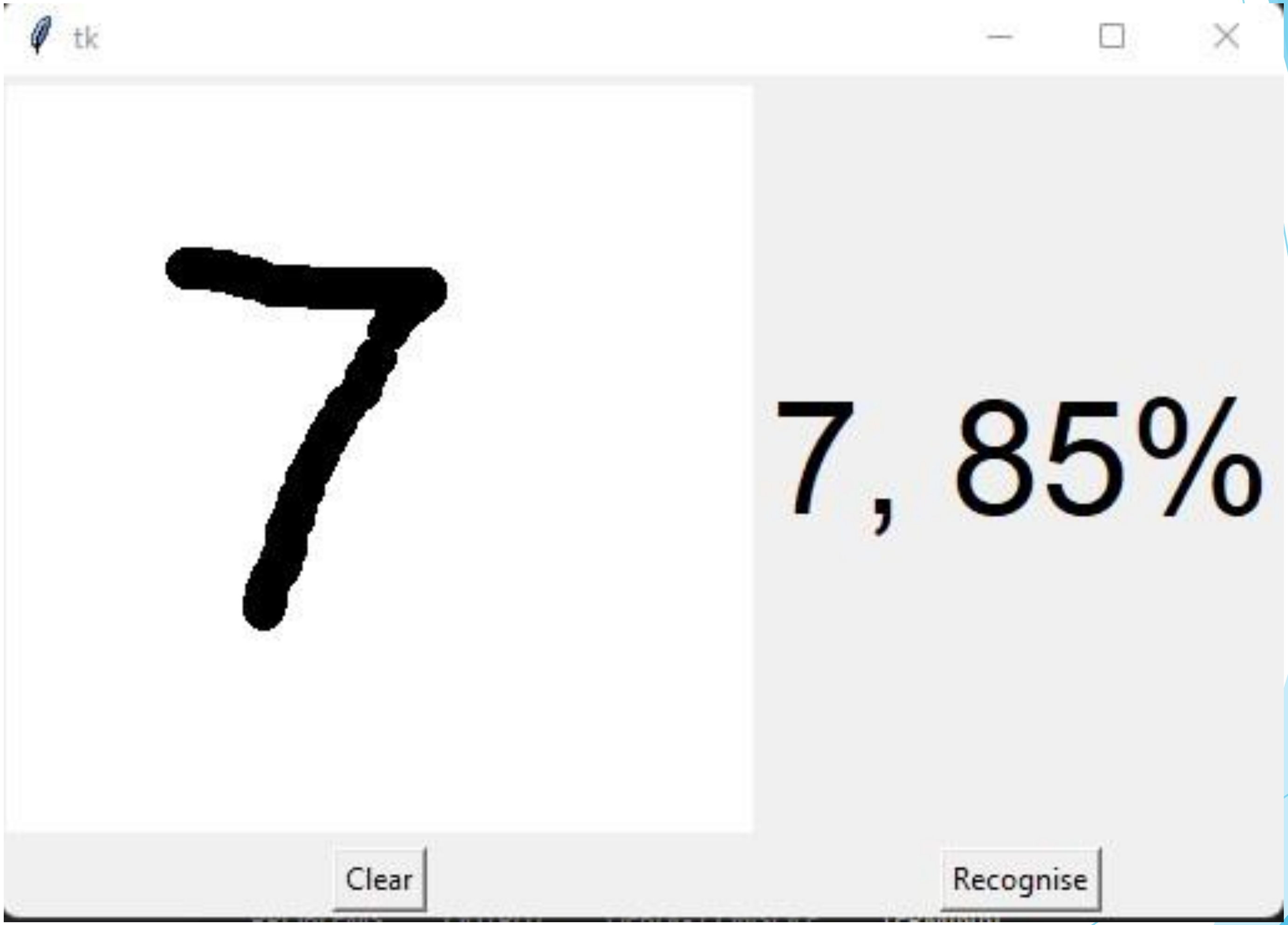


```
from keras.models import load_model
from tkinter import *
import tkinter as tk
import win32gui
from PIL import ImageGrab, Image
import numpy as np

model = load_model('mnist.h5')
def predict_digit(img):
    img = img.resize((28,28))
    img = img.convert('L')
    img = np.array(img)
    img = img.reshape(1,28,28,1)
    img = img/255.0
    res = model.predict([img])[0]
    return np.argmax(res), max(res)

class App(tk.Tk):
    def __init__(self):
        tk.Tk.__init__(self)
        self.x = self.y = 0
        self.canvas = tk.Canvas(self, width=300, height=300, bg = "white", cursor="cross")
        self.label = tk.Label(self, text="Draw..", font=("Helvetica", 48))
        self.classify_btn = tk.Button(self, text = "Recognise", command =
self.classify_handwriting)
        self.button_clear = tk.Button(self, text = "Clear", command = self.clear_all)

        self.canvas.grid(row=0, column=0, pady=2, sticky=W, )
        self.label.grid(row=0, column=1,pady=2, padx=2)
        self.classify_btn.grid(row=1, column=1, pady=2, padx=2)
        self.button_clear.grid(row=1, column=0, pady=2)
        self.canvas.bind("<B1-Motion>", self.draw_lines)
```

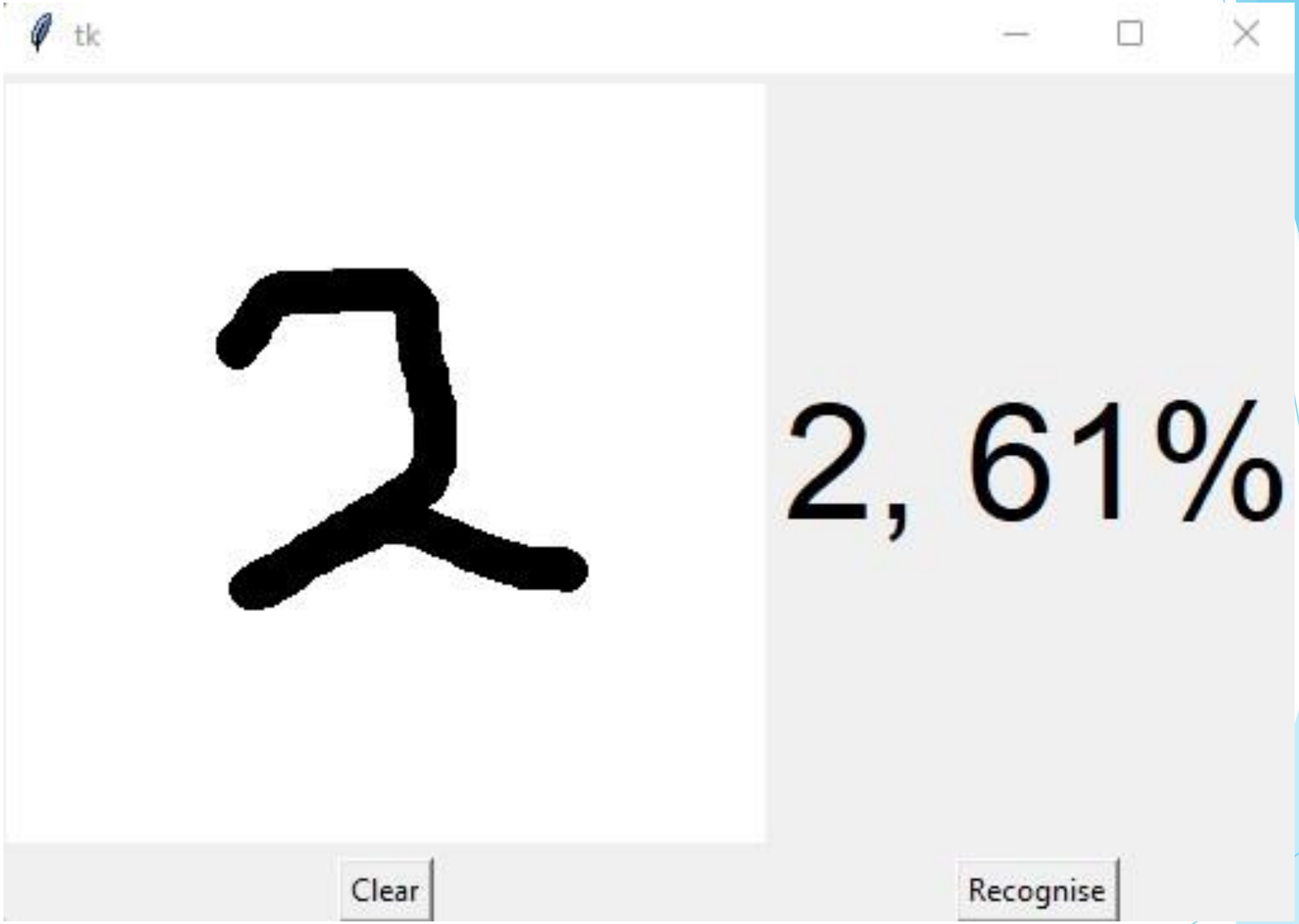


```
    def clear_all(self):
        self.canvas.delete("all")

    def classify_handwriting(self):
        HWND = self.canvas.winfo_id()
        rect = win32gui.GetWindowRect(HWND)
        a,b,c,d = rect
        rect=(a+4,b+4,c-4,d-4)
        im = ImageGrab.grab(rect)
        digit, acc = predict_digit(im)
        self.label.configure(text= str(digit)+', '+ str(int(acc*100))+'%')

    def draw_lines(self, event):
        self.x = event.x
        self.y = event.y
        r=8
        self.canvas.create_oval(self.x-r, self.y-r, self.x + r, self.y + r, fill='black')

app = App()
mainloop()
```

Result



7, 85%



2, 61%

Algorithm Used

Convolutional Neural Network (CNN)

- A Convolutional Neural Network is a Deep Learning algorithm that can take in an input image, assign importance to various aspects/objects in the image and be able to differentiate one from the other.
- Convolutional neural networks are more complex than standard multi-layer perceptrons, so we will start by using a simple structure, to begin with, that uses all of the elements for state-of-the-art results. Below summarizes the network architecture.

Conclusion

- Convolutional Neural Network is very effective for image classification purposes.
- For GUI we used Tkinter, wherein we drew a digit on canvas, then classified the digit with it’s accuracy.
- Hence, we have created a Model in Python to detect human handwritten digits.

Thank You...!!