STREAM Project 2

Handwritten digit classification

In this project, you will be focusing on:

Science: understand the function the nerves in the human brain

Technology: use of Google Colab to implement the Python program

Reading: use different types of research to aid in completing the project successfully

Engineering: the engineering design process employed through computer engineering skills to program neural networks using Python programming skills

Art: neural network architecture and required learning curves for measuring the performance of neural networks

Maths: defining parameters such as the bias and weight, for the neural network and learning activation functions basic calculations

Entrepreneurial Design Process

Stage 1	The design brief
Stage 2	Analysing the brief
Stage 3	Research and investigation
Stage 4	Possible solutions
Stage 5	Final solution
Stage 6	Design realisation
Stage 7	Evaluation

STREAM Project 2: Handwritten digit classification

Aim

This section of the workbook aims to provide activities that assess the understanding of the topics covered in the student book and skills guide. It will provide activities to evaluate your understanding of neural networks and their design using Python. In this project, you will build an ANN to classify handwritten digits based on the MNIST dataset. The project will be completed with an entrepreneurial outlook that combines all areas of STREAM and the engineering design process to complete the project successfully.

Learning outcomes

- Develop a simple neural network algorithm to solve a problem.
- Solve a classification problem using NNs.
- Use training and testing datasets to build a model.
- Use learning curves to diagnose the performance of a machine learning model.
- Analyse the main sections of an advanced design brief.
- Address the constraints and requirements of an advanced design problem.
- Apply different methods of research to solve an advanced design problem.
- Transform research ideas into possible solutions for an advanced design problem.
- Construct a prototype to solve an advanced design problem through various prototyping means.
- Test a design prototype for operational expectations solving an advanced design problem.
- Evaluate the success of implementing the proposed design idea solving an advanced design problem.
- Improve a design prototype to overcome identified design faults.
- Create a business plan and work within a team to achieve it.

Prior Knowledge

- Programming
- Neural network concepts

MNIST digit dataset

The MNIST digit dataset is developed by Yann LeCun, Corinna Cortes and Christopher Burges for evaluating machine learning models on the handwritten digit classification problem. The dataset was constructed from a number of scanned document and is available from the National Institute of Standards and Technology (NIST). This is where the name for the dataset comes from, as the Modified NIST or MNIST dataset. Images of digits were taken from a variety of scanned documents, normalised in size and centred. This makes it an excellent dataset for evaluating models, allowing the developer to focus on machine learning with very little data cleaning or preparation required.

It consists of 70,000 labelled 28x28 pixel grayscale images of handwritten digits. The dataset is split into 60,000 training images and 10,000 test images. There are ten classes (one for each of the ten digits).

STAGE 1

The design brief

Modern technology and new concepts have increased the reliability of machines over humans. We rely on machines to make our lives safer and easier, whether it is making calculations or scanning our retinas for recognising people's identities. It is a fact that has been proven by the amount of research and development undertaken in the field of AI, particularly deep learning and machine learning.

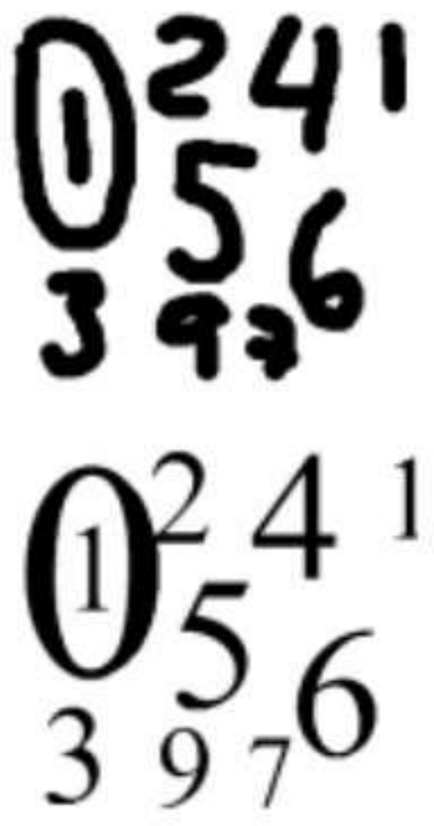
The amount of research work going on in the field of deep learning and machine learning completes the task of object classification in photographs to adding sound to silent movies. Similarly, handwritten text recognition is one of the significant areas of research and development with a lot of posibilities. Handwriting recognition (HWR), also known as Handwritten Text Recognition (HTR), is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. Handwritten Text recognition is an important application of deep learning and machine learning. This technique is helpful in detecting forgeries while verifying signatures, interpreting postal addresses, processing bank cheques, etc.

Neural Networks $\longrightarrow$ "Neural Networks"

Likewise, handwritten digit recognition is the ability of a computer to recognise the human handwritten digits and classify them into ten predefined classes (0-9). This is one of the famous topics of boundless-research in the field of deep learning. Digit recognition has many applications like number plate recognition, postal mail sorting, bank check processing, etc. In handwritten digit recognition, many challenges are faced because of different styles of writing.

As a grade 12 CDI student, you must now apply the skills you have learned in CDI to design an ANN model to solve a problem.

Imagine you are a trainee AI engineer who needs to build an ANN model to classify handwritten digits in order to be able to use it in ddifferent applications (like banks, RTA, police investigation, electronic assessments, etc). This will be your task for the next few weeks. Having done so, you should develop a business model for an application of your own choice and prepare to present your idea to possible investors.



You will now complete your STREAM project to create an ANN model to recognise handwritten digits for your chosen application (RTA, Bank, Police investigation etc.), using Python programming language.

STAGE 2

Analysis of brief

Activity 2.2.1

In this space below, list and explain at least 4 keywords from the design brief

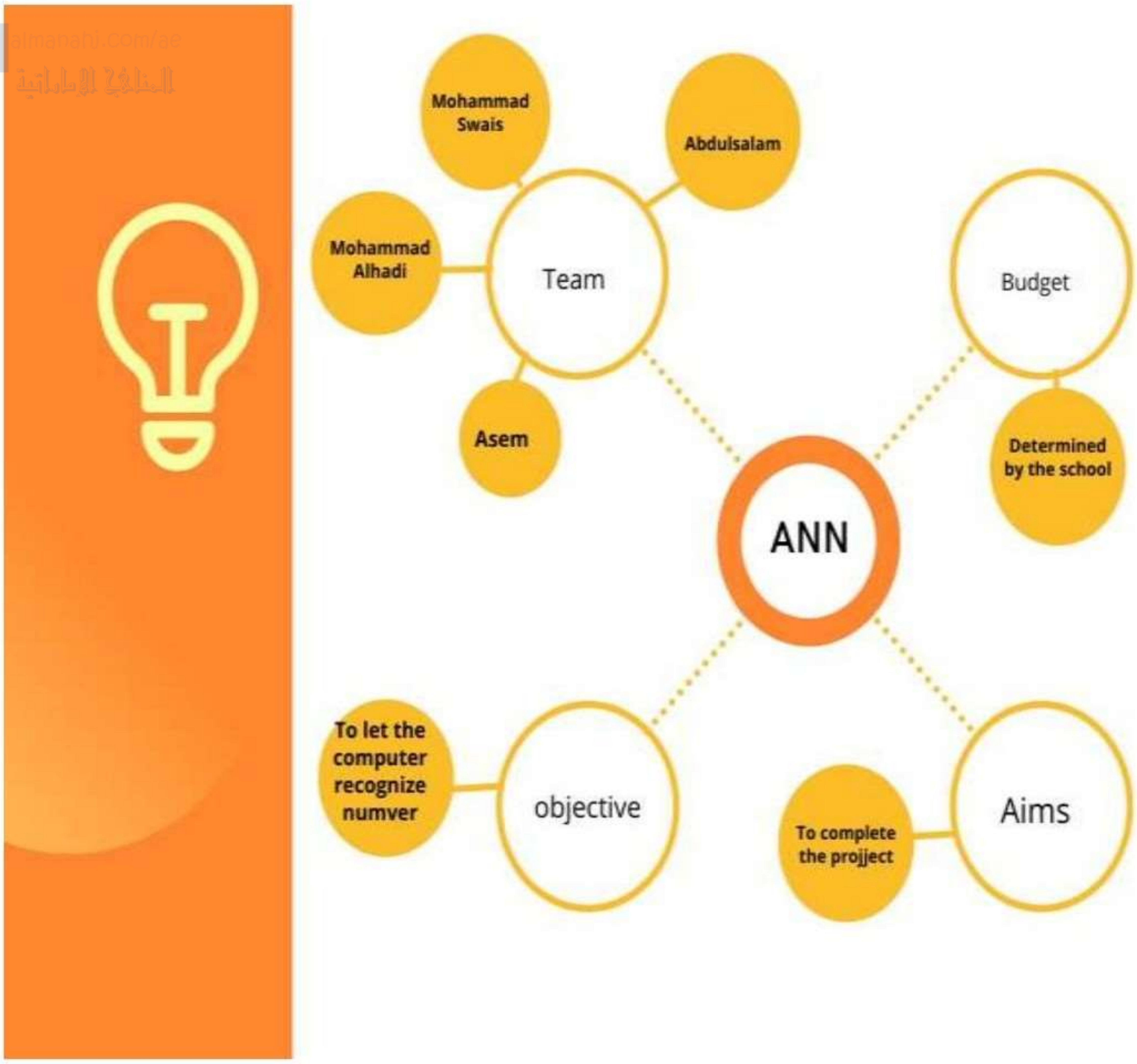
Keyword	Meaning:
RTA	The roads and transport authority website is an online gate for all online services for Dubai traffic, fines, licensing, public transport, nol and transport
Machine learning	Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.
Recognition (HTR)	is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices.
ANN	Artificial neural networks, usually simply called neural networks, are computing systems inspired by the biological neural networks that constitute animal brains.
Modern technology	Modern technology is all about efficiency and speed; it is about ensuring face-to-face communication, connecting you to your healthcare provider, and empowering you by giving you more access and control to the kind of care you get as well as service you receive.

Activity 2.2.2

Key areas of brief:	Possible questions	Explain the key areas in the brief.
Aims and objectives	What is the overall aim? What steps will you take to meet this aim?	Handwritten digit recognition by using python.
Budget and schedule	Do you have a budget? When must your project be completed?	Free budget 22 February
Target audience	Who is this project aimed at?	Special students and organizations

Activity 2.2.3

Create your own Mind Map. Add key information you have taken from activity 2.2.2.



STAGE 3

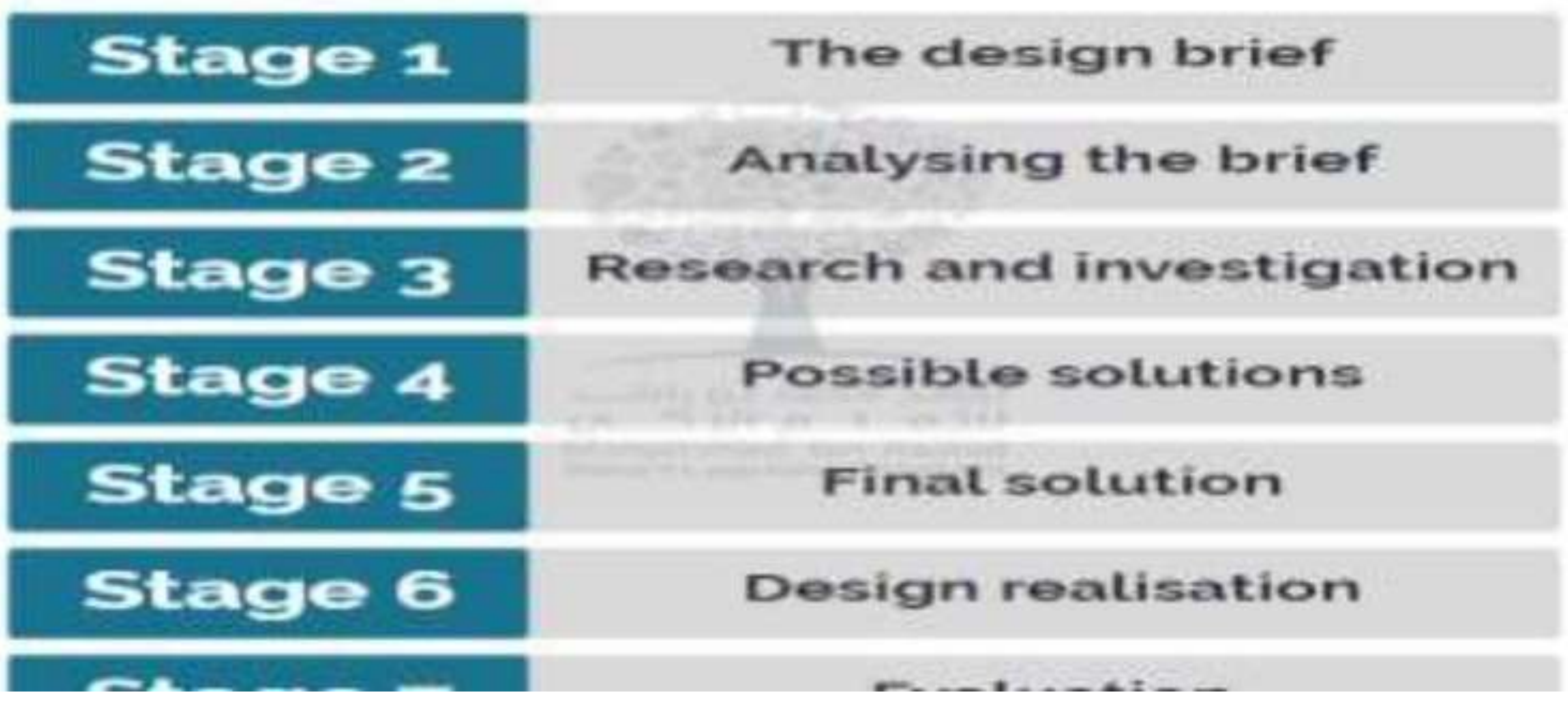
Research and investigation

Activity 2.3.1

Consider the questions below to help you carry your research.

- How can machines learn in general?
  a machine "learns" by looking for patterns among massive data loads, and when it sees one, it adjusts the program to reflect the "truth" of what it found.
- What is Deep Q learning?
  Critically, Deep Q-Learning replaces the regular Q-table with a neural network.
- What can I learn about the MNIST dataset?
  It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.
- What is meant by handwritten digit recognition?
  Handwritten digit recognition is the ability of a computer to recognize the human handwritten digits from different sources like images, papers, touch screens, etc, and classify them into 10 predefined classes (0-9)
- What are the different AI / ML/ DL techniques that can be used for handwritten digit recognition? There are a number of ways and algorithms to recognize handwritten digits, including Deep Learning/CNN, SVM, Gaussian Naive Bayes, KNN, Decision Trees, Random Forests, etc.
- What are the different real-life applications for handwritten digit recognition? Handwritten Digit Recognition has various real-life time uses. It is used in the detection of vehicle number, banks for reading cheques, post offices for arranging letter, and many other tasks.
- How do neural networks help in handwritten digit recognition? Compare it with different methods. Compare it with different methods. The idea is to take a large number of handwritten digits, known as training examples, and then develop a system which can learn from those training examples. In other words, the neural network uses the examples to automatically infer rules for recognizing handwritten digits.

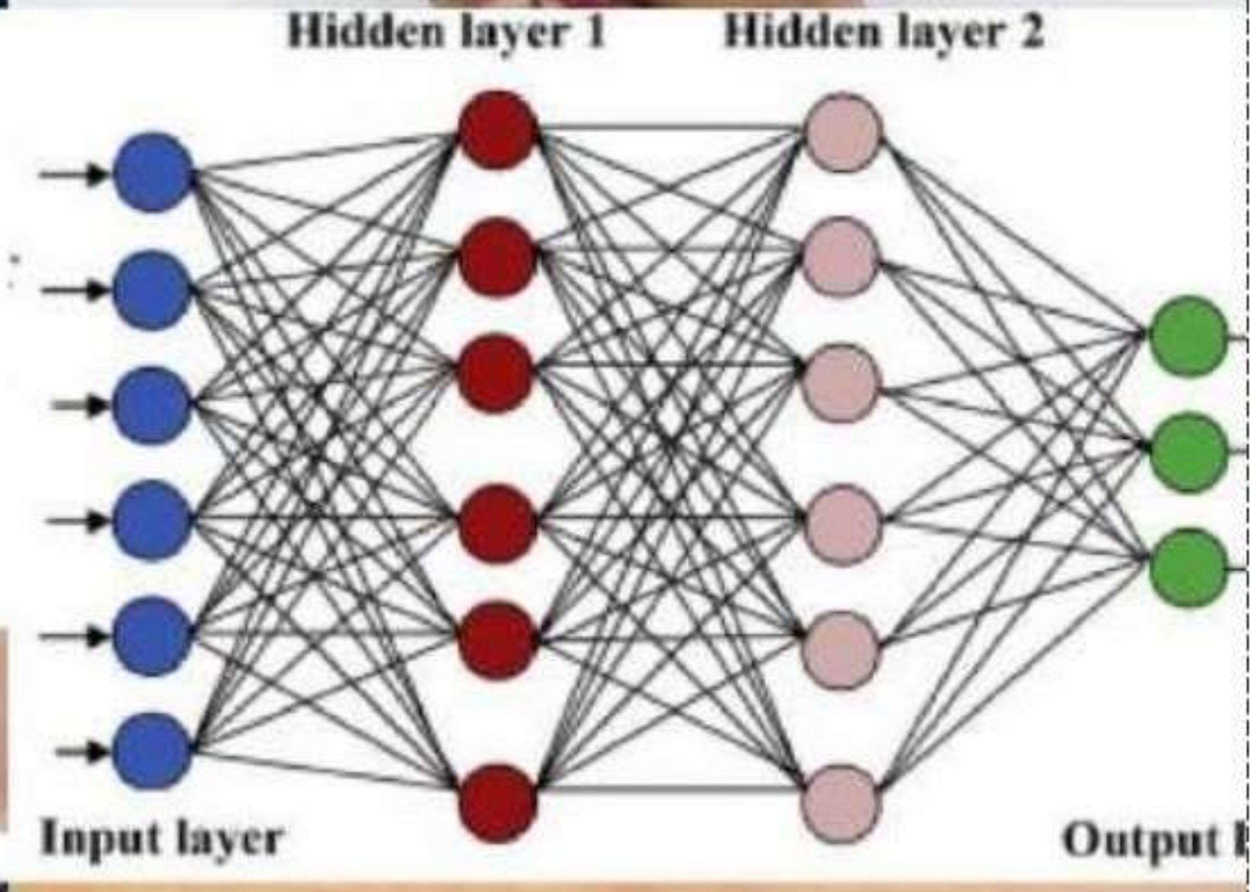
Show clear research of all the aspects of the design brief. You may use a combination of text and images to explain the research carried out.

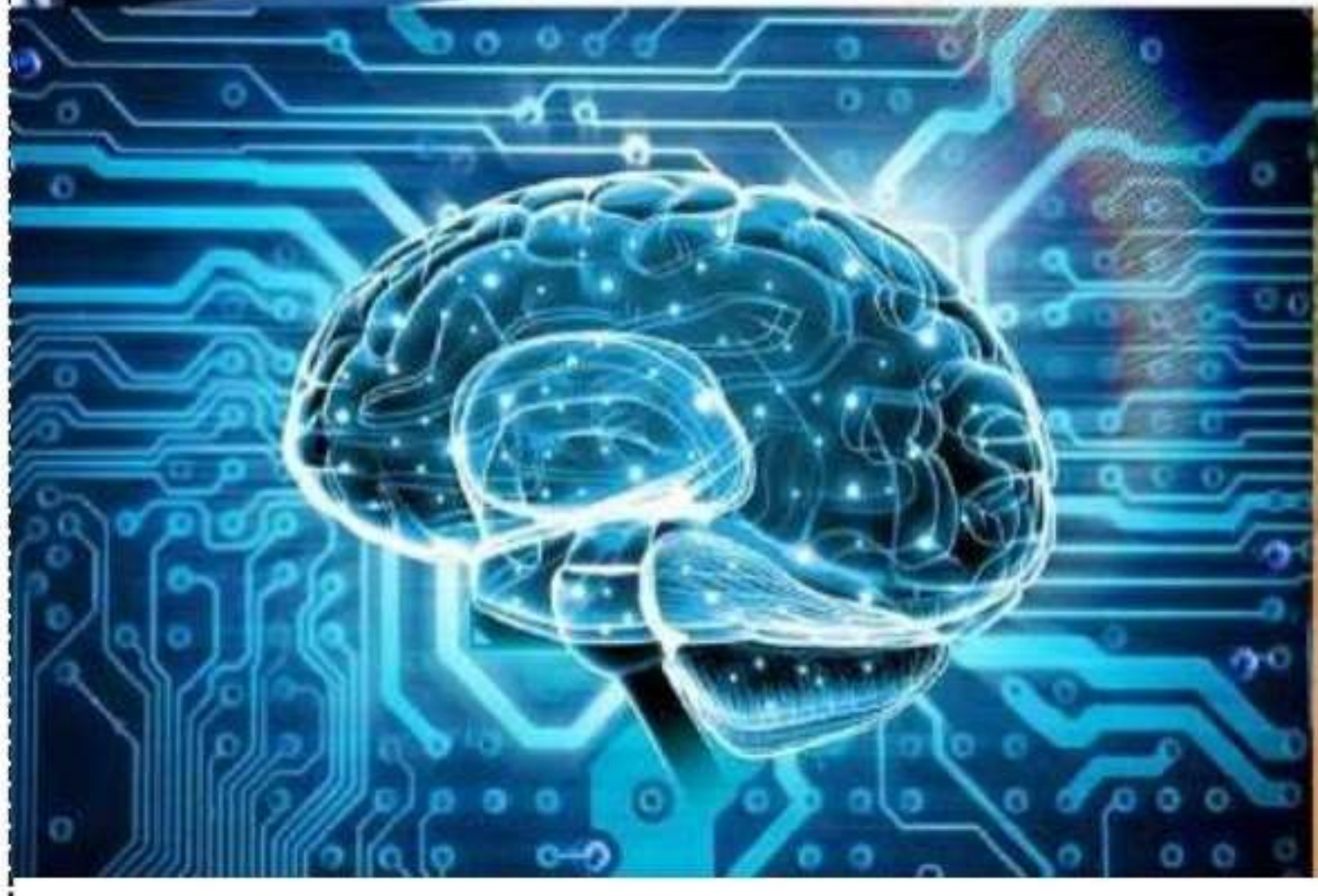


Stage 1	The design brief
Stage 2	Analysing the brief
Stage 3	Research and investigation
Stage 4	Possible solutions
Stage 5	Final solution
Stage 6	Design realisation

Activity 2.3.2

Create a mood board/research page to display the information gathered in activities 2.2.1-2.3.1. You may use a combination of images, sketches, and notes.





STAGE 4

Possible solutions

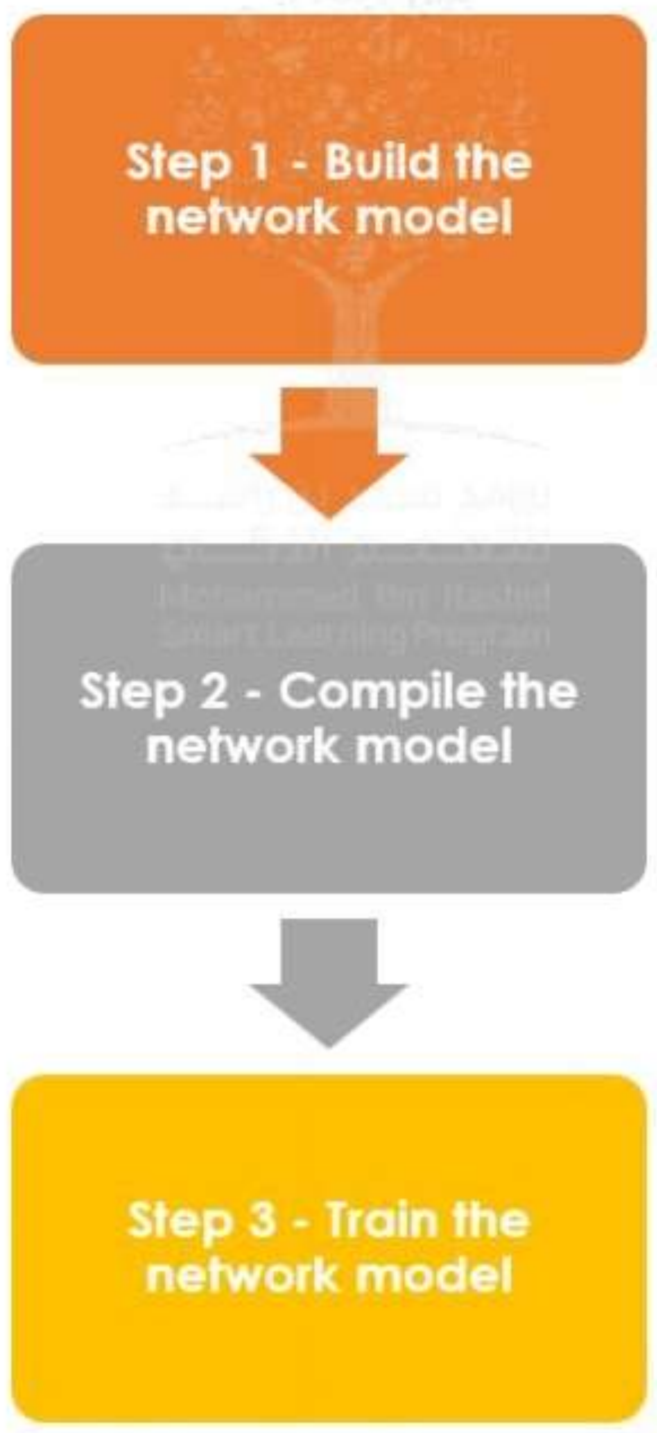
Possible ANN model scenarios

Creating the model possibilities

Creating an ANN model depends heavily on the number of layers in the network. To construct the network, you will need to determine a number of parameters, including the number of layers, the number of neurons in each layer, the activation function required, and the number of outputs.

Remember that the number of layers in an ANN, and the number of neurons inside each layer it can be decided by you based on the task type.

As you have learned in the student book, follow the steps below to design an ANN model.


Step 1 - Build the network model
Step 2 - Compile the network model
Step 3 - Train the network model

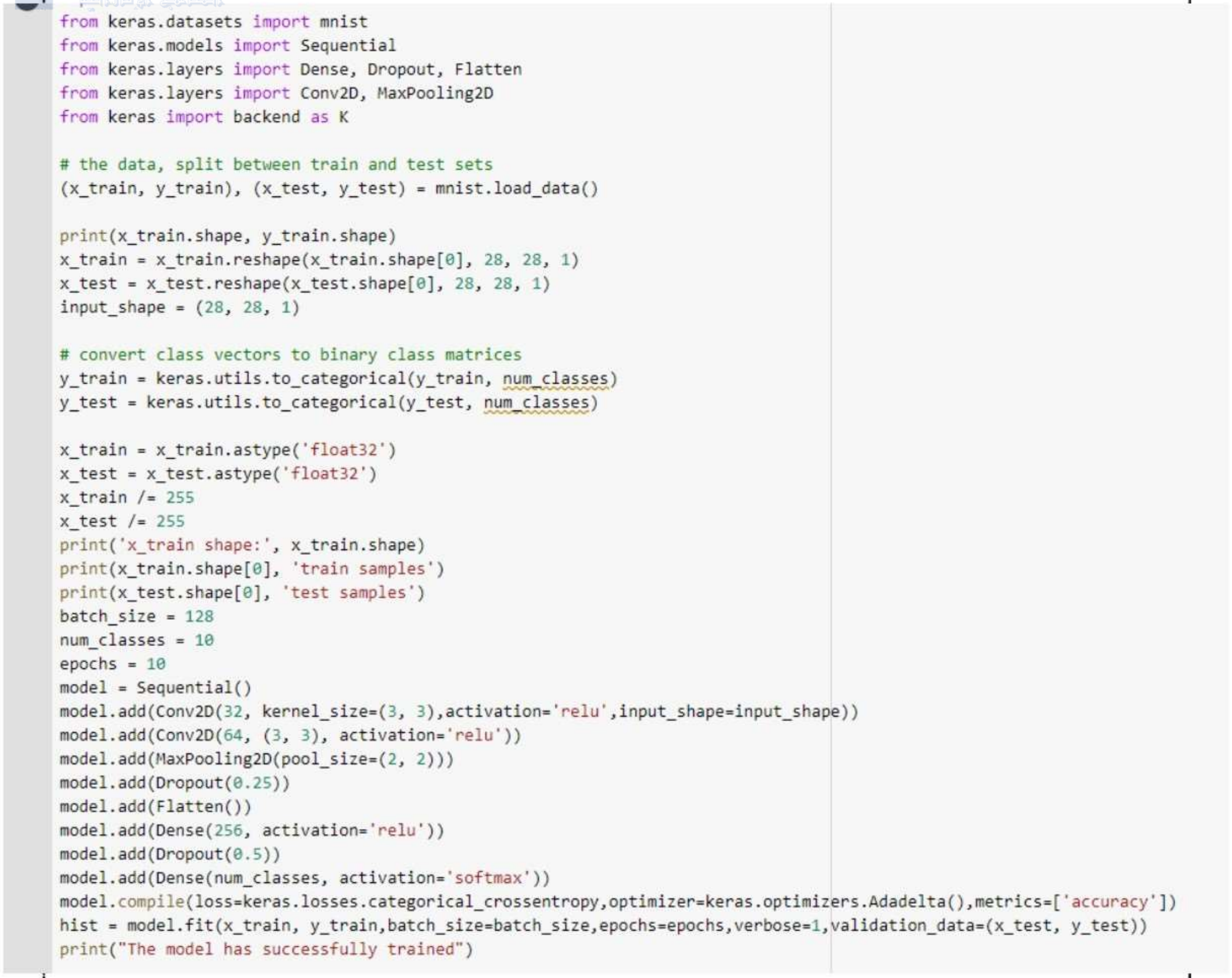
Note:

- When building your ANN model, you can use a variety of different combinations of layers. To begin with, you can use the combination you learned in the student book and then decide if you wish to change it.
- You can have more than two possible solutions to show that you have done a lot of tests and trials before finalising your design solution in Stage 5. Make sure to record all of your suggestions.

Activity 2.4.1

In the space below, write the code for creating a simple ANN model that will be able to classify the digits correctly.

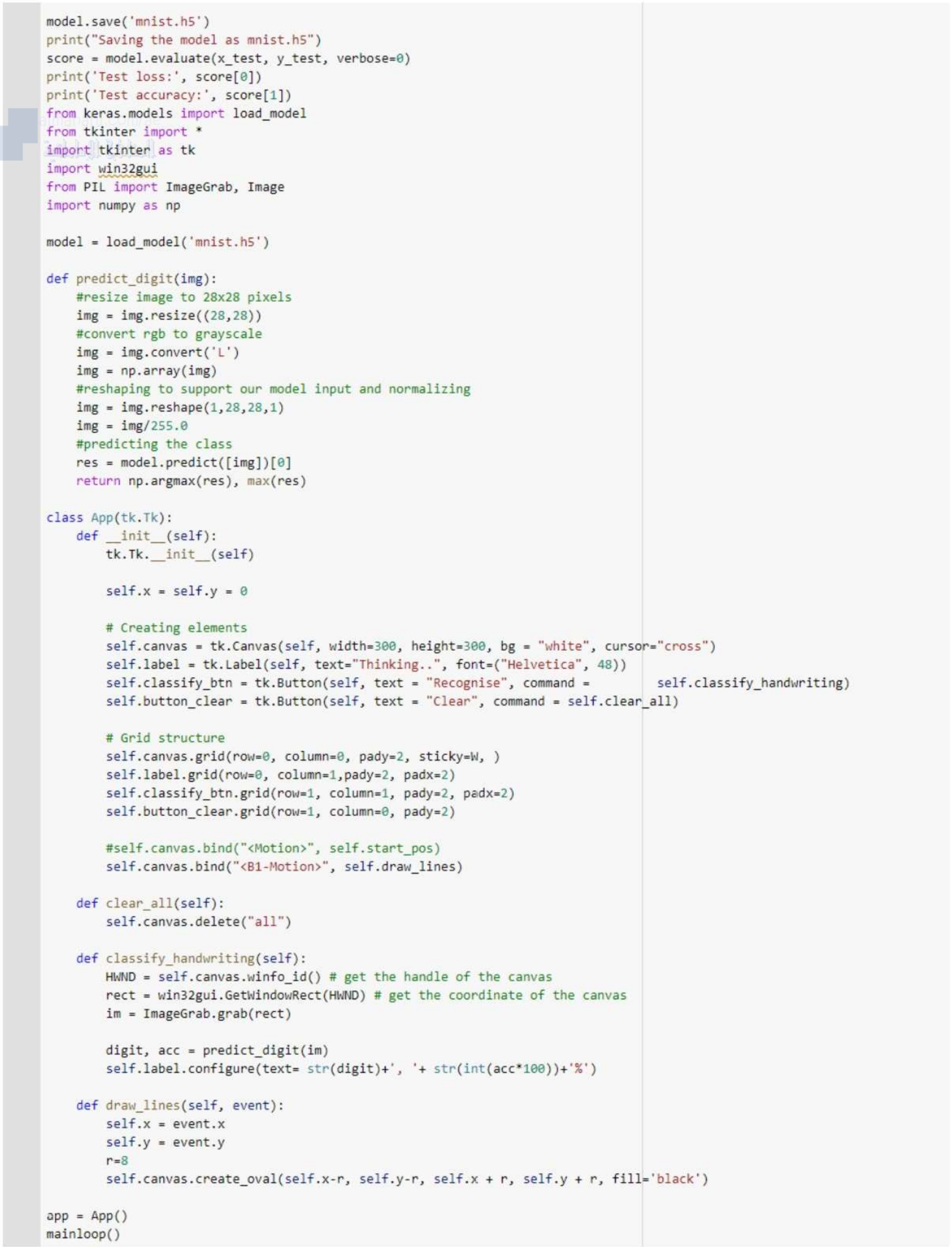
First code



```
from keras.datasets import mnist
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)
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.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

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')
batch_size = 128
num_classes = 10
epochs = 10
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])
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")
```

Advantages: Accuracy up to 99%. , medium speed to get results

Disadvantages : a lot of codes that can be skipped , needs a lot of training cont code 2

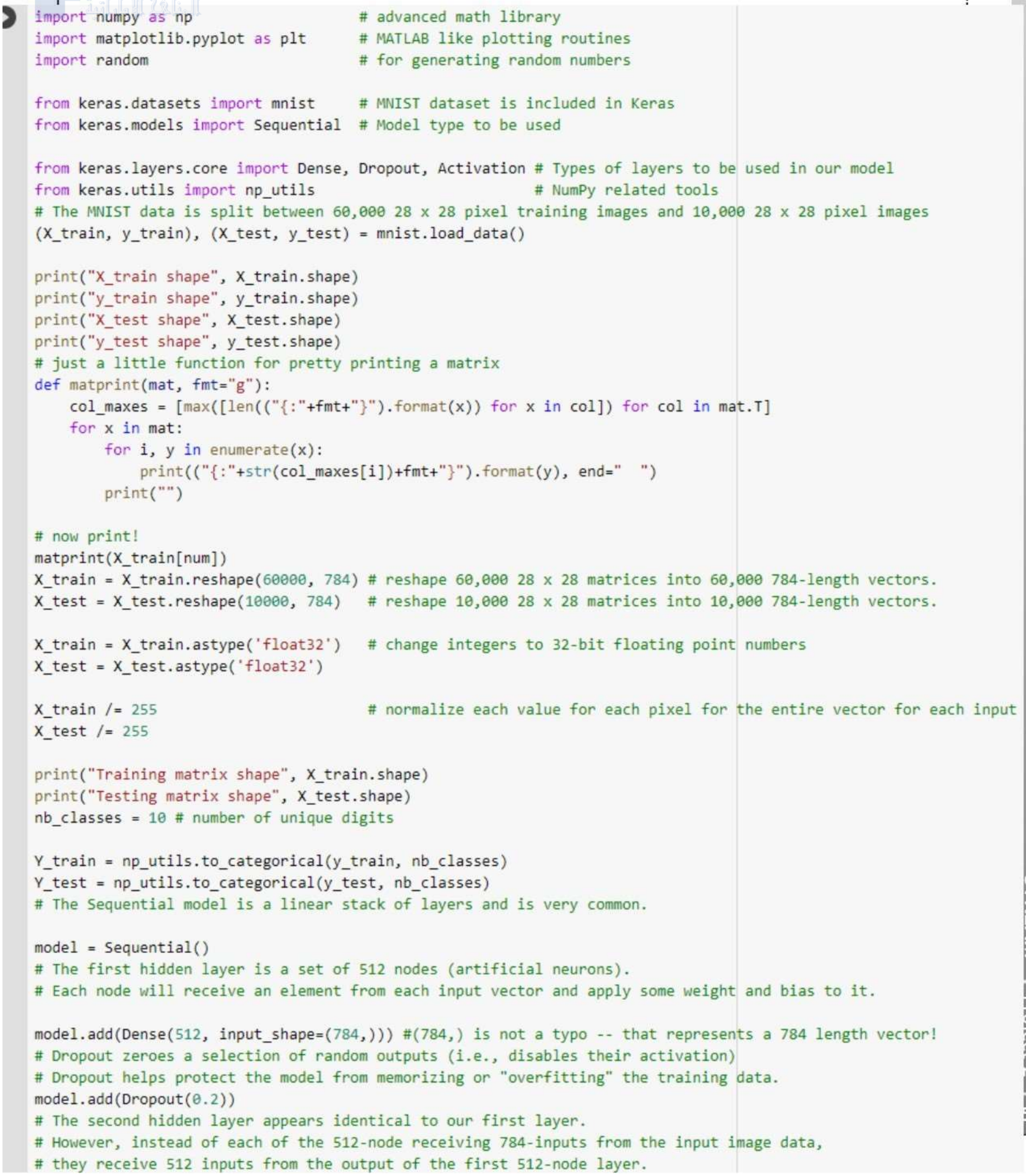


```
model.save('mnist.h5')
print("Saving the model as mnist.h5")
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
from keras.models import load_model
from tkinter import *
import tkinter as tk
import win32gui
from PIL import ImageGrab, Image
import numpy as np

model = load_model('mnist.h5')

def predict_digit(img):
    #resize image to 28x28 pixels
    img = img.resize((28,28))
    #convert rgb to grayscale
    img = img.convert('L')
    img = np.array(img)
    #reshaping to support our model input and normalizing
    img = img.reshape(1,28,28,1)
    img = img/255.0
    #predicting the class
    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

        # Creating elements
        self.canvas = tk.Canvas(self, width=300, height=300, bg = "white", cursor="cross")
        self.label = tk.Label(self, text="Thinking..", 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)

        # Grid structure
        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("<Motion>", self.start_pos)
        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() # get the handle of the canvas
        rect = win32gui.GetWindowRect(HWND) # get the coordinate of the canvas
        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()
```

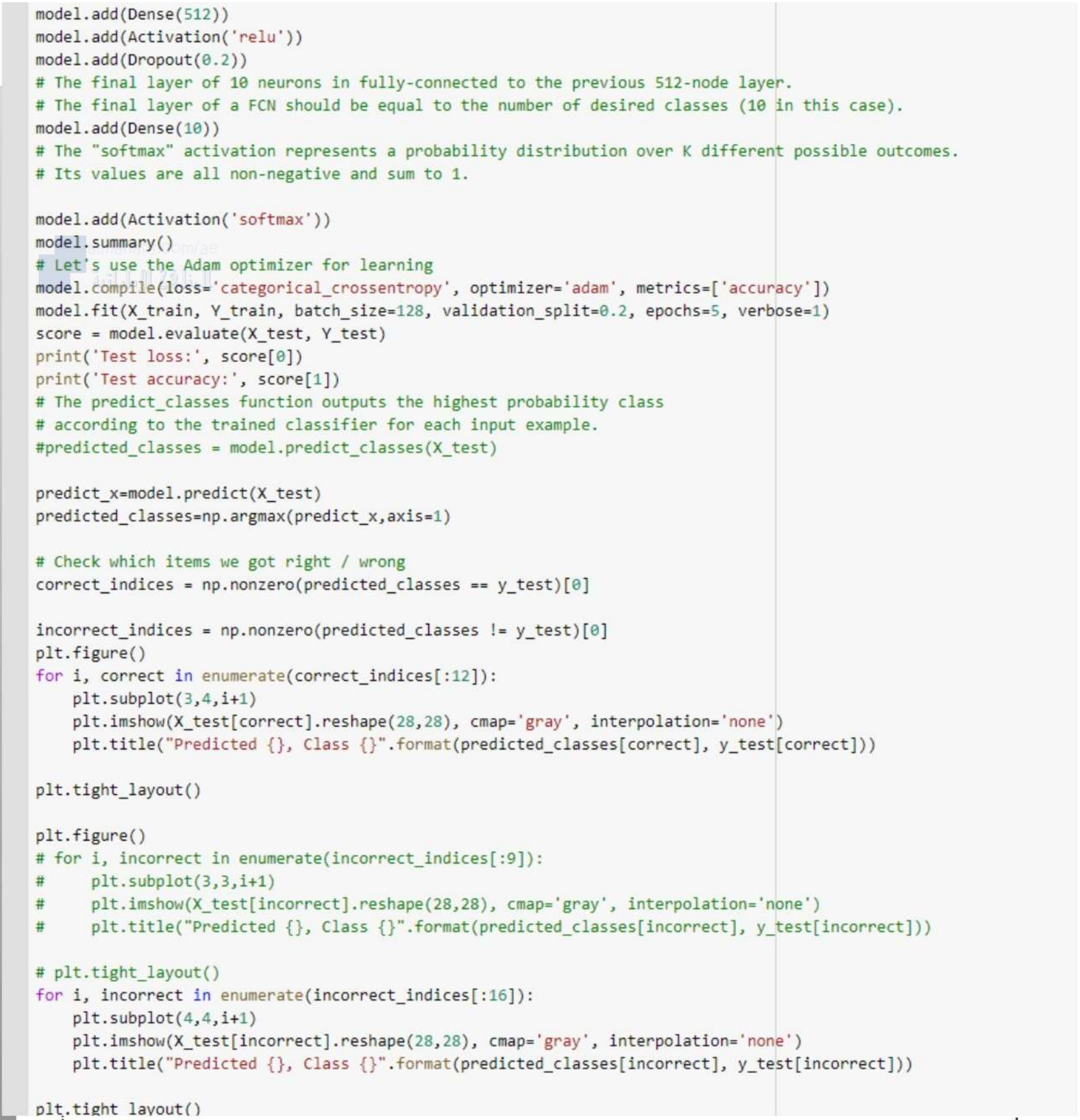
Activity 2.4.2

In the space below, write another possible code for creating an ANN model that will be able to classify the digits correctly.

Second code



```
import numpy as np                    # advanced math library
import matplotlib.pyplot as plt       # MATLAB like plotting routines
import random                         # for generating random numbers

from keras.datasets import mnist      # MNIST dataset is included in Keras
from keras.models import Sequential   # Model type to be used

from keras.layers.core import Dense, Dropout, Activation # Types of layers to be used in our model
from keras.utils import np_utils                         # NumPy related tools
# The MNIST data is split between 60,000 28 x 28 pixel training images and 10,000 28 x 28 pixel images
(X_train, y_train), (X_test, y_test) = mnist.load_data()

print("X_train shape", X_train.shape)
print("y_train shape", y_train.shape)
print("X_test shape", X_test.shape)
print("y_test shape", y_test.shape)
# just a little function for pretty printing a matrix
def matprint(mat, fmt="g"):
    col_maxes = [max([len(("{:"+fmt+"}").format(x)) for x in col]) for col in mat.T]
    for x in mat:
        for i, y in enumerate(x):
            print(("{:"+str(col_maxes[i])+fmt+"}").format(y), end="  ")
        print("")

# now print!
matprint(X_train[num])
X_train = X_train.reshape(60000, 784) # reshape 60,000 28 x 28 matrices into 60,000 784-length vectors.
X_test = X_test.reshape(10000, 784)   # reshape 10,000 28 x 28 matrices into 10,000 784-length vectors.

X_train = X_train.astype('float32')   # change integers to 32-bit floating point numbers
X_test = X_test.astype('float32')

X_train /= 255                        # normalize each value for each pixel for the entire vector for each input
X_test /= 255

print("Training matrix shape", X_train.shape)
print("Testing matrix shape", X_test.shape)
nb_classes = 10 # number of unique digits

Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)
# The Sequential model is a linear stack of layers and is very common.

model = Sequential()
# The first hidden layer is a set of 512 nodes (artificial neurons).
# Each node will receive an element from each input vector and apply some weight and bias to it.

model.add(Dense(512, input_shape=(784,))) #(784,) is not a typo -- that represents a 784 length vector!
# Dropout zeroes a selection of random outputs (i.e., disables their activation)
# Dropout helps protect the model from memorizing or "overfitting" the training data.
model.add(Dropout(0.2))
# The second hidden layer appears identical to our first layer.
# However, instead of each of the 512-node receiving 784-inputs from the input image data,
# they receive 512 inputs from the output of the first 512-node layer.
```



```
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
# The final layer of 10 neurons in fully-connected to the previous 512-node layer.
# The final layer of a FCN should be equal to the number of desired classes (10 in this case).
model.add(Dense(10))
# The "softmax" activation represents a probability distribution over K different possible outcomes.
# Its values are all non-negative and sum to 1.

model.add(Activation('softmax'))
model.summary()
# Let's use the Adam optimizer for learning
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=128, validation_split=0.2, epochs=5, verbose=1)
score = model.evaluate(X_test, Y_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
# The predict_classes function outputs the highest probability class
# according to the trained classifier for each input example.
#predicted_classes = model.predict_classes(X_test)

predict_x=model.predict(X_test)
predicted_classes=np.argmax(predict_x,axis=1)

# Check which items we got right / wrong
correct_indices = np.nonzero(predicted_classes == y_test)[0]

incorrect_indices = np.nonzero(predicted_classes != y_test)[0]
plt.figure()
for i, correct in enumerate(correct_indices[:12]):
    plt.subplot(3,4,i+1)
    plt.imshow(X_test[correct].reshape(28,28), cmap='gray', interpolation='none')
    plt.title("Predicted {}, Class {}".format(predicted_classes[correct], y_test[correct]))

plt.tight_layout()

plt.figure()
# for i, incorrect in enumerate(incorrect_indices[:9]):
#     plt.subplot(3,3,i+1)
#     plt.imshow(X_test[incorrect].reshape(28,28), cmap='gray', interpolation='none')
#     plt.title("Predicted {}, Class {}".format(predicted_classes[incorrect], y_test[incorrect]))

# plt.tight_layout()
for i, incorrect in enumerate(incorrect_indices[:16]):
    plt.subplot(4,4,i+1)
    plt.imshow(X_test[incorrect].reshape(28,28), cmap='gray', interpolation='none')
    plt.title("Predicted {}, Class {}".format(predicted_classes[incorrect], y_test[incorrect]))

plt.tight_layout()
```

Advantages : shorter in terms of codes which means easier for maintenance and error check , faster in getting results

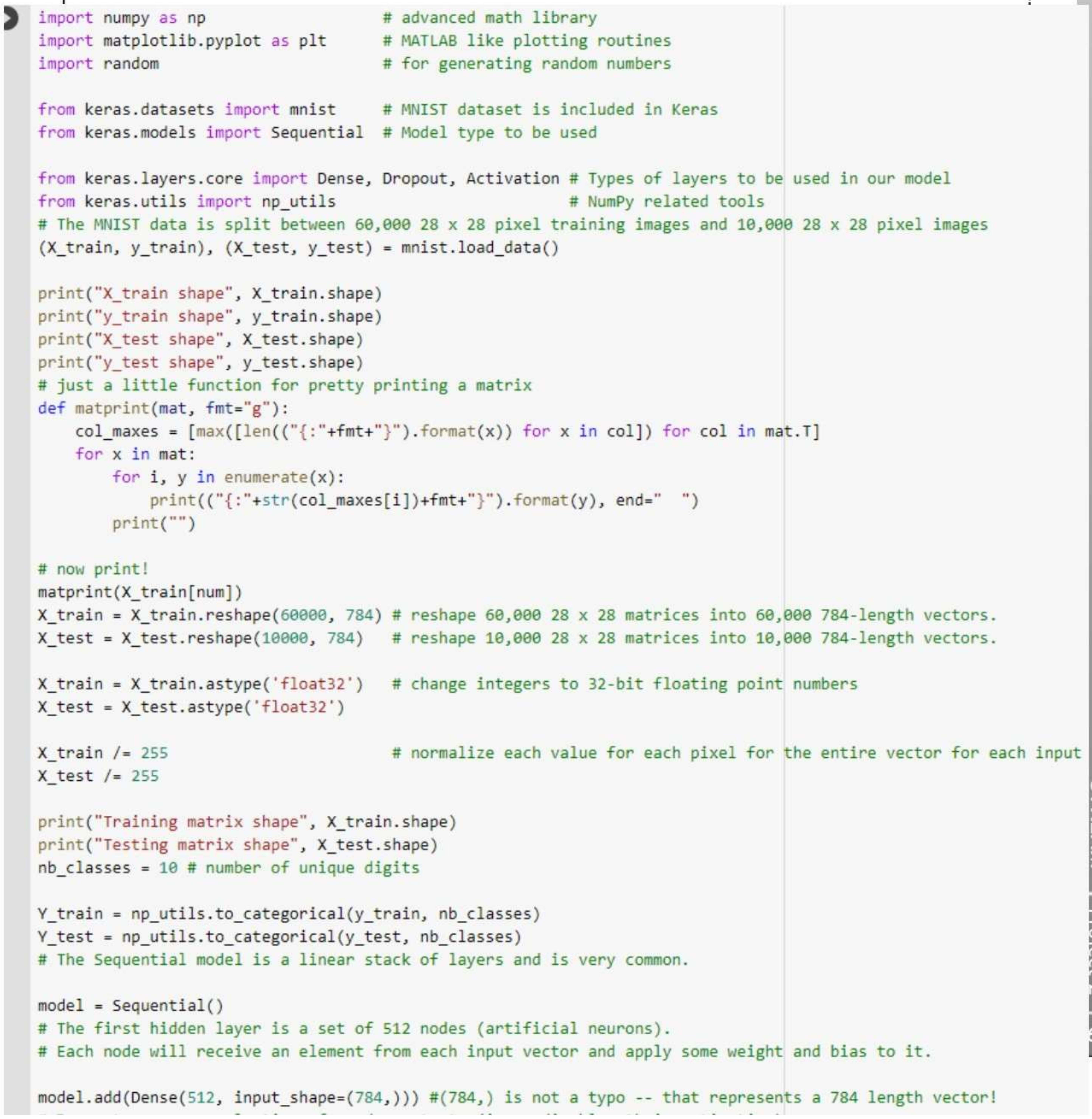
Disadvantages : needs time to write the full code , every code is connected directly to the code behind it .

STAGE 5

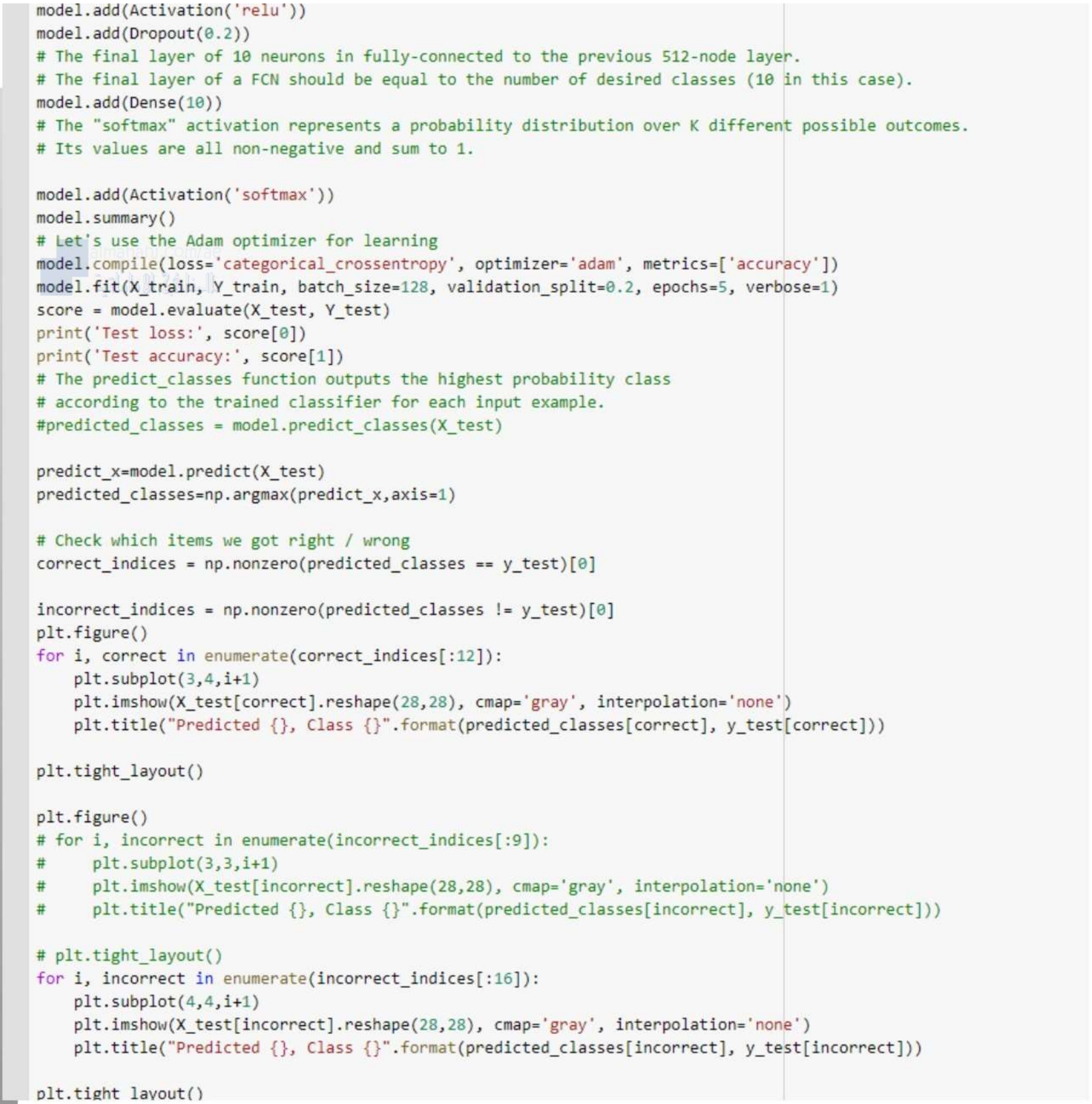
Final solution / Python coding

Activity 2.5.1

After you have suggested and tried different ANN network structures, you need to choose the code that showed the best accuracy to complete your project. (Final solution)



```
import numpy as np                    # advanced math library
import matplotlib.pyplot as plt      # MATLAB like plotting routines
import random                         # for generating random numbers

from keras.datasets import mnist     # MNIST dataset is included in Keras
from keras.models import Sequential  # Model type to be used

from keras.layers.core import Dense, Dropout, Activation # Types of layers to be used in our model
from keras.utils import np_utils                         # NumPy related tools
# The MNIST data is split between 60,000 28 x 28 pixel training images and 10,000 28 x 28 pixel images
(X_train, y_train), (X_test, y_test) = mnist.load_data()

print("X_train shape", X_train.shape)
print("y_train shape", y_train.shape)
print("X_test shape", X_test.shape)
print("y_test shape", y_test.shape)
# just a little function for pretty printing a matrix
def matprint(mat, fmt="g"):
    col_maxes = [max([len(("{:"+fmt+"}").format(x)) for x in col]) for col in mat.T]
    for x in mat:
        for i, y in enumerate(x):
            print(("{:"+str(col_maxes[i])+fmt+"}").format(y), end="  ")
        print("")

# now print!
matprint(X_train[num])
X_train = X_train.reshape(60000, 784) # reshape 60,000 28 x 28 matrices into 60,000 784-length vectors.
X_test = X_test.reshape(10000, 784)   # reshape 10,000 28 x 28 matrices into 10,000 784-length vectors.

X_train = X_train.astype('float32')   # change integers to 32-bit floating point numbers
X_test = X_test.astype('float32')

X_train /= 255                        # normalize each value for each pixel for the entire vector for each input
X_test /= 255

print("Training matrix shape", X_train.shape)
print("Testing matrix shape", X_test.shape)
nb_classes = 10 # number of unique digits

Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)
# The Sequential model is a linear stack of layers and is very common.

model = Sequential()
# The first hidden layer is a set of 512 nodes (artificial neurons).
# Each node will receive an element from each input vector and apply some weight and bias to it.

model.add(Dense(512, input_shape=(784,))) #(784,) is not a typo -- that represents a 784 length vector!
```



```
model.add(Activation('relu'))
model.add(Dropout(0.2))
# The final layer of 10 neurons in fully-connected to the previous 512-node layer.
# The final layer of a FCN should be equal to the number of desired classes (10 in this case).
model.add(Dense(10))
# The "softmax" activation represents a probability distribution over K different possible outcomes.
# Its values are all non-negative and sum to 1.

model.add(Activation('softmax'))
model.summary()
# Let's use the Adam optimizer for learning
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=128, validation_split=0.2, epochs=5, verbose=1)
score = model.evaluate(X_test, Y_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
# The predict_classes function outputs the highest probability class
# according to the trained classifier for each input example.
#predicted_classes = model.predict_classes(X_test)

predict_x=model.predict(X_test)
predicted_classes=np.argmax(predict_x,axis=1)

# Check which items we got right / wrong
correct_indices = np.nonzero(predicted_classes == y_test)[0]

incorrect_indices = np.nonzero(predicted_classes != y_test)[0]
plt.figure()
for i, correct in enumerate(correct_indices[:12]):
    plt.subplot(3,4,i+1)
    plt.imshow(X_test[correct].reshape(28,28), cmap='gray', interpolation='none')
    plt.title("Predicted {}, Class {}".format(predicted_classes[correct], y_test[correct]))

plt.tight_layout()

plt.figure()
# for i, incorrect in enumerate(incorrect_indices[:9]):
#     plt.subplot(3,3,i+1)
#     plt.imshow(X_test[incorrect].reshape(28,28), cmap='gray', interpolation='none')
#     plt.title("Predicted {}, Class {}".format(predicted_classes[incorrect], y_test[incorrect]))

# plt.tight_layout()
for i, incorrect in enumerate(incorrect_indices[:16]):
    plt.subplot(4,4,i+1)
    plt.imshow(X_test[incorrect].reshape(28,28), cmap='gray', interpolation='none')
    plt.title("Predicted {}, Class {}".format(predicted_classes[incorrect], y_test[incorrect]))

plt.tight_layout()
```

Advantages : shorter in terms of codes which means easier for maintenance and error check , faster in getting results

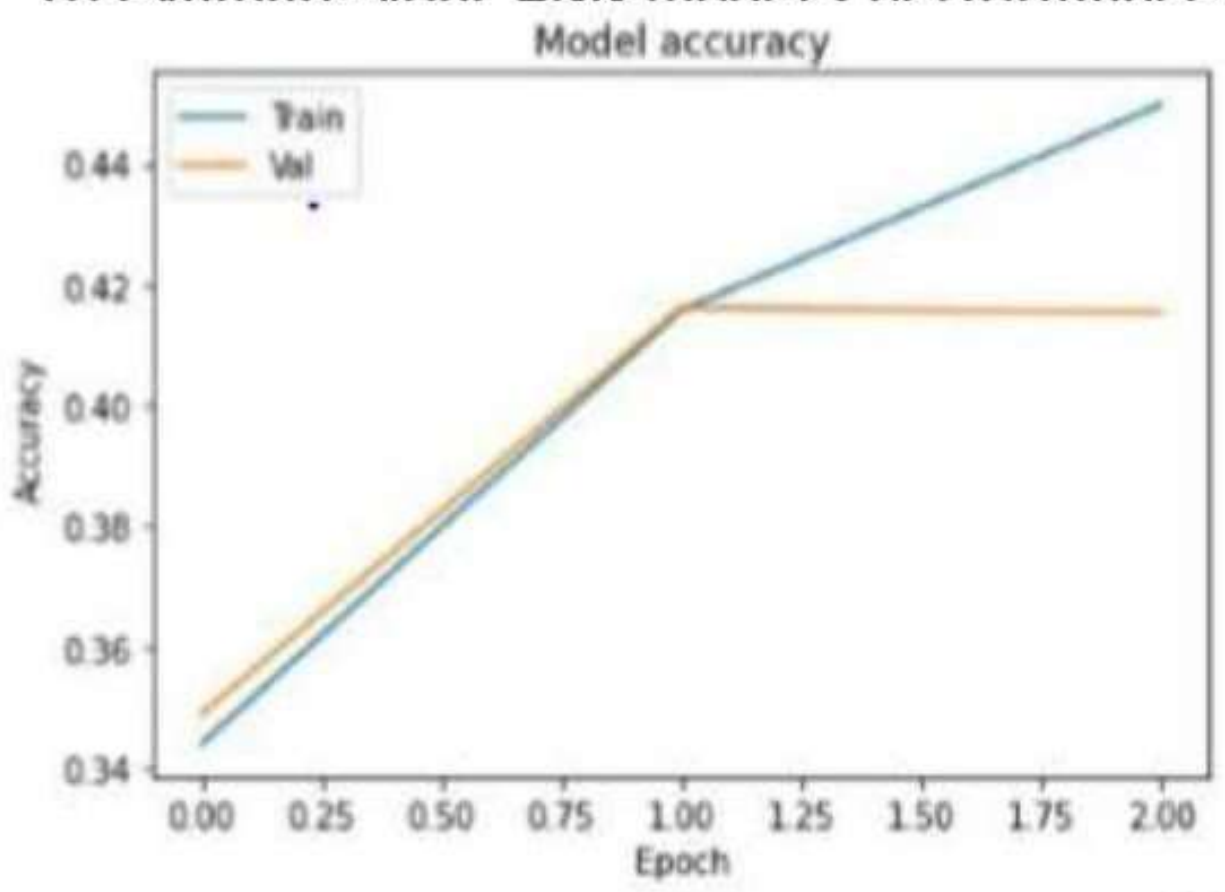
Disadvantages : needs time to write the full code , every code is connected directly to the code behind it

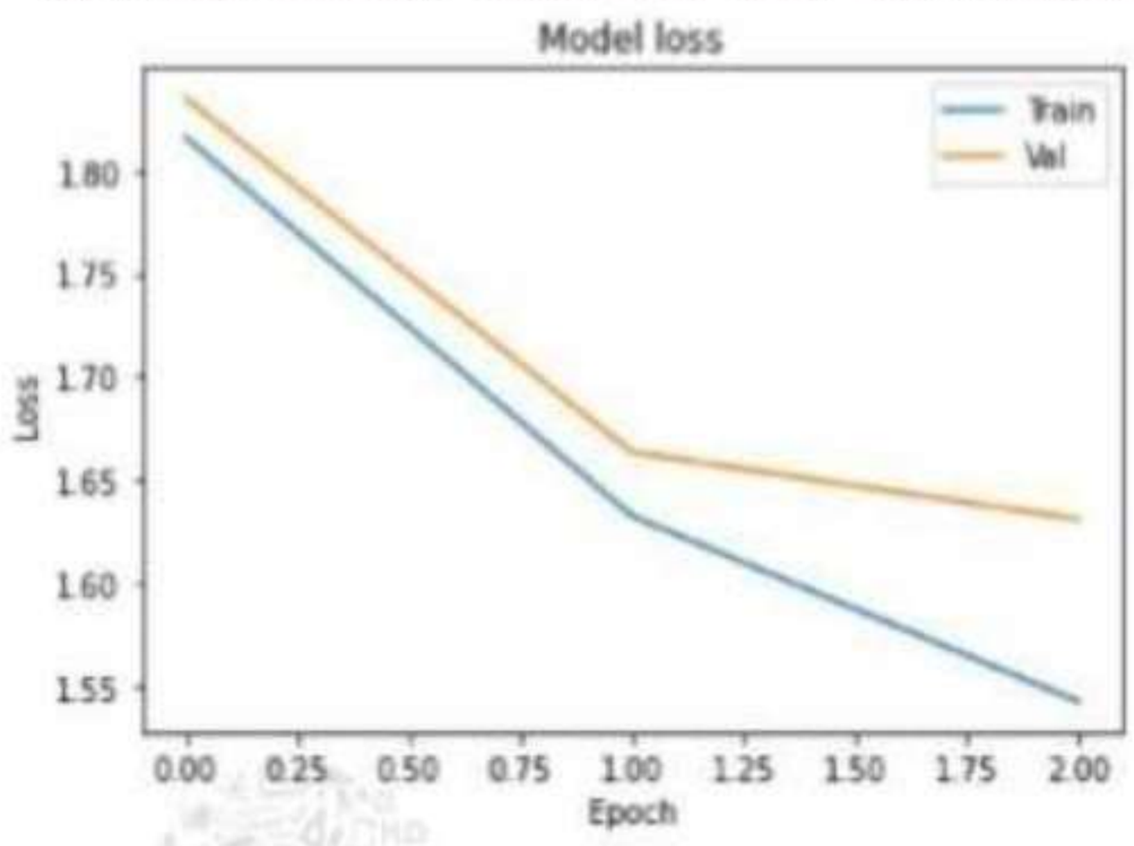
STAGE 6

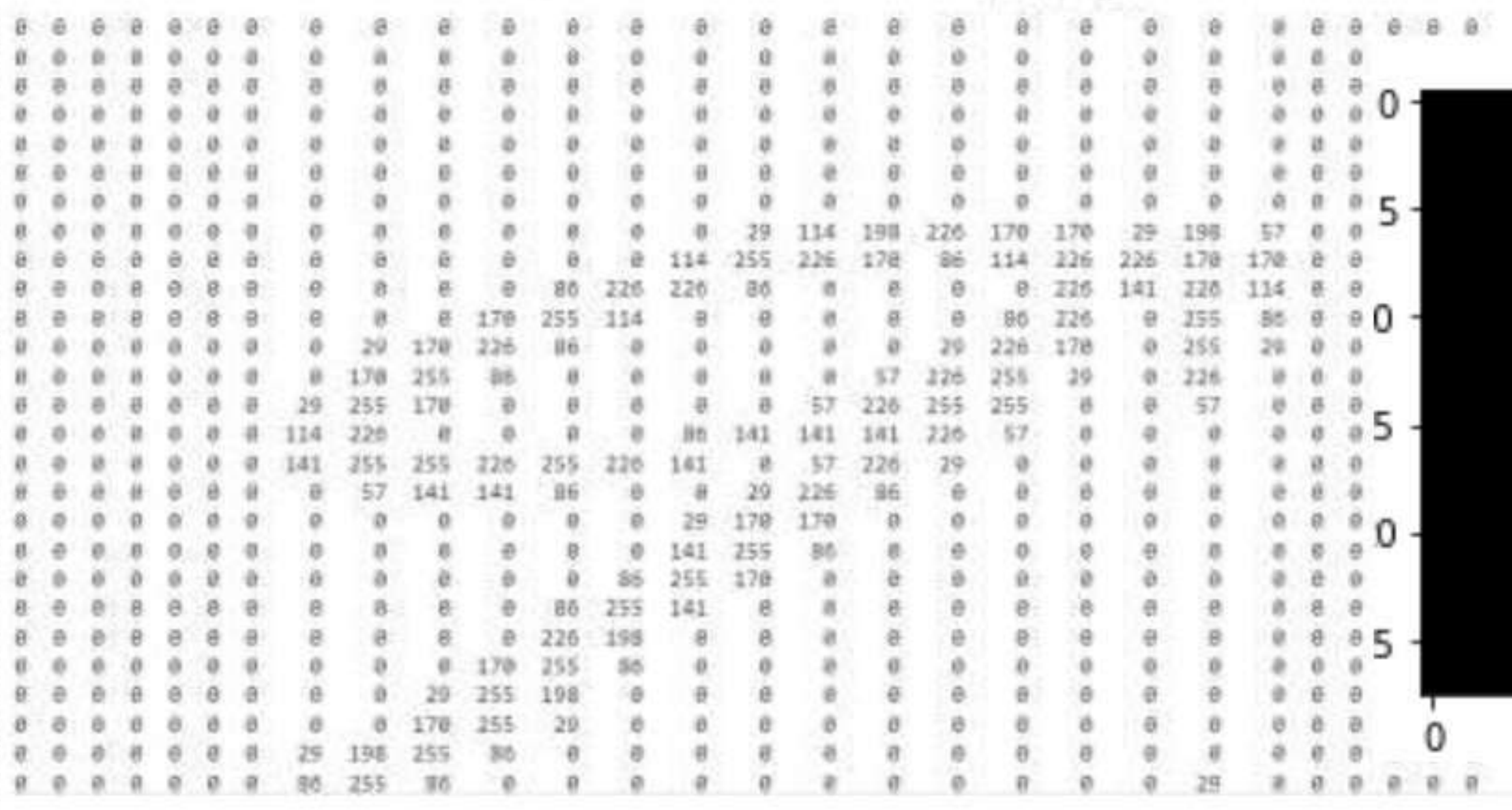
Design realisation

Activity 2.6.1

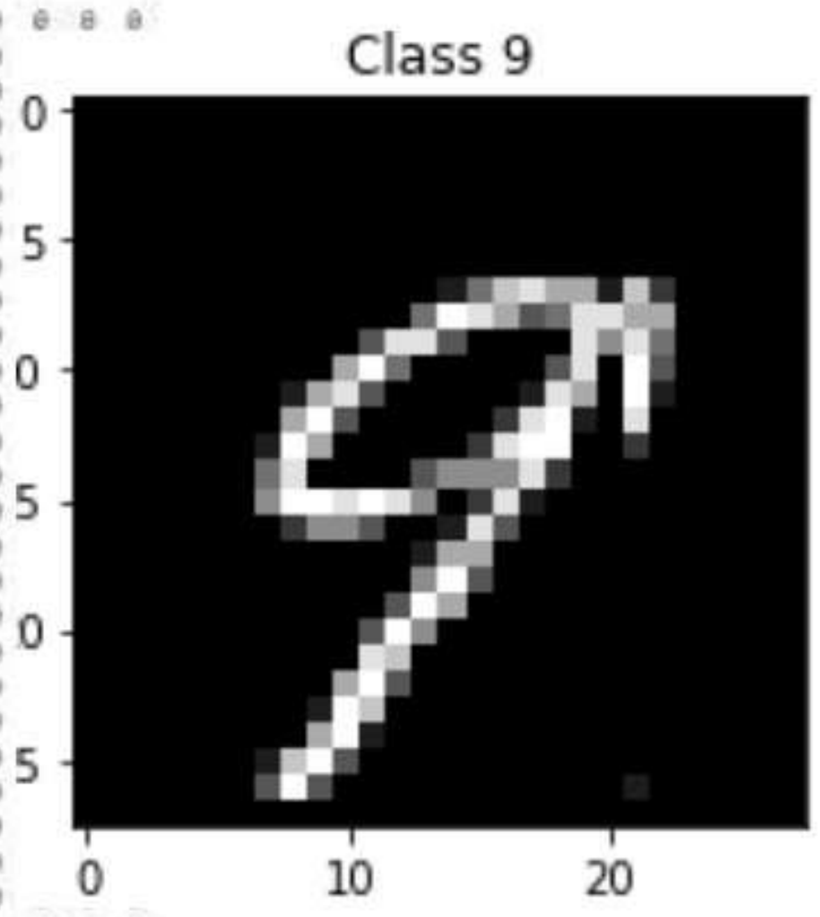
To evaluate your ANN model's performance, plot the model's accuracy and loss figures.

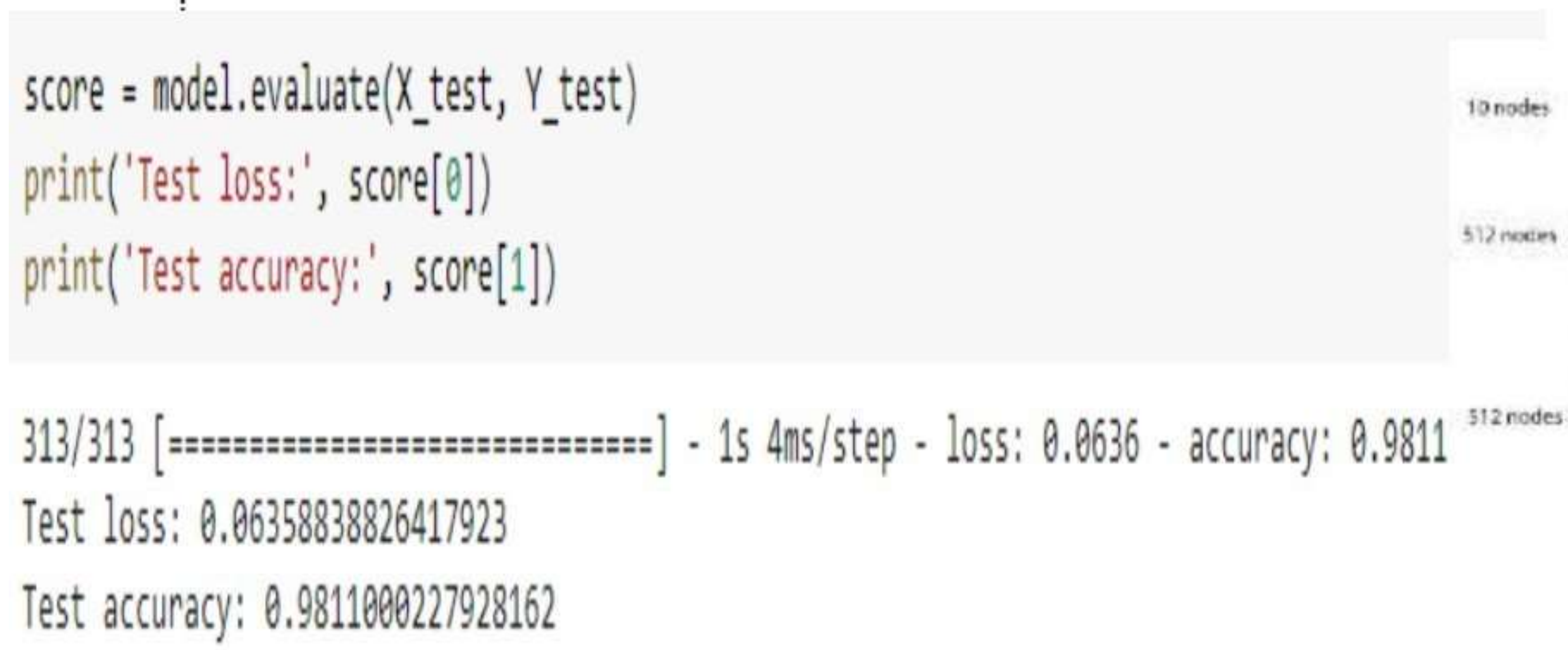






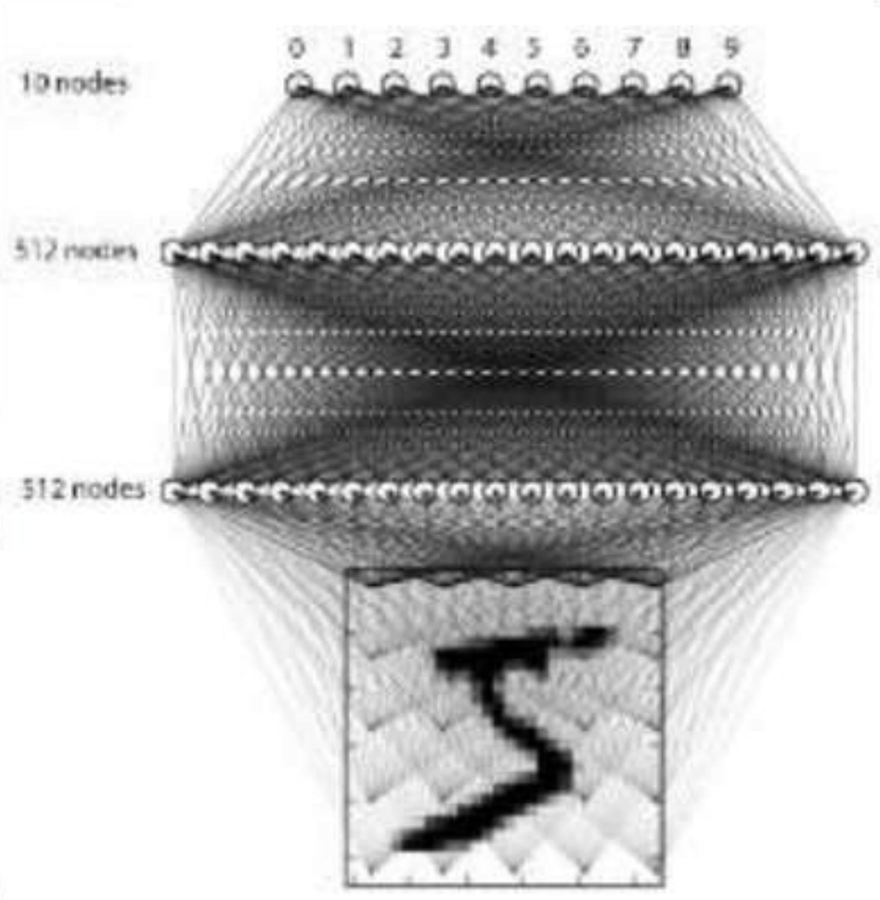
```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 114 198 226 170 170 29 198 57 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 114 255 226 170 86 114 226 226 170 170 0 0
0 0 0 0 0 0 0 0 0 0 0 86 226 226 86 0 0 0 0 226 141 226 114 0 0
0 0 0 0 0 0 0 0 0 0 170 255 114 0 0 0 0 0 86 226 0 255 86 0 0
0 0 0 0 0 0 0 0 29 170 226 86 0 0 0 0 0 29 226 170 0 255 29 0 0
0 0 0 0 0 0 0 0 170 255 86 0 0 0 0 0 57 226 255 29 0 226 0 0 0
0 0 0 0 0 0 0 29 255 170 0 0 0 0 0 57 226 255 255 0 0 57 0 0 0
0 0 0 0 0 0 0 114 226 0 0 0 0 86 141 141 141 226 57 0 0 0 0 0 0
0 0 0 0 0 0 0 141 255 255 226 255 226 141 0 57 226 29 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 57 141 141 86 0 0 29 226 86 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 29 170 170 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 141 255 86 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 86 255 170 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 86 255 141 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 226 198 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 170 255 86 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 29 255 198 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 170 255 29 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 29 198 255 86 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 86 255 86 0 0 0 0 0 0 0 0 0 0 0 29 0 0 0 0 0 0
```





```
score = model.evaluate(X_test, Y_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0636 - accuracy: 0.9811
Test loss: 0.06358838826417923
Test accuracy: 0.9811000227928162
```



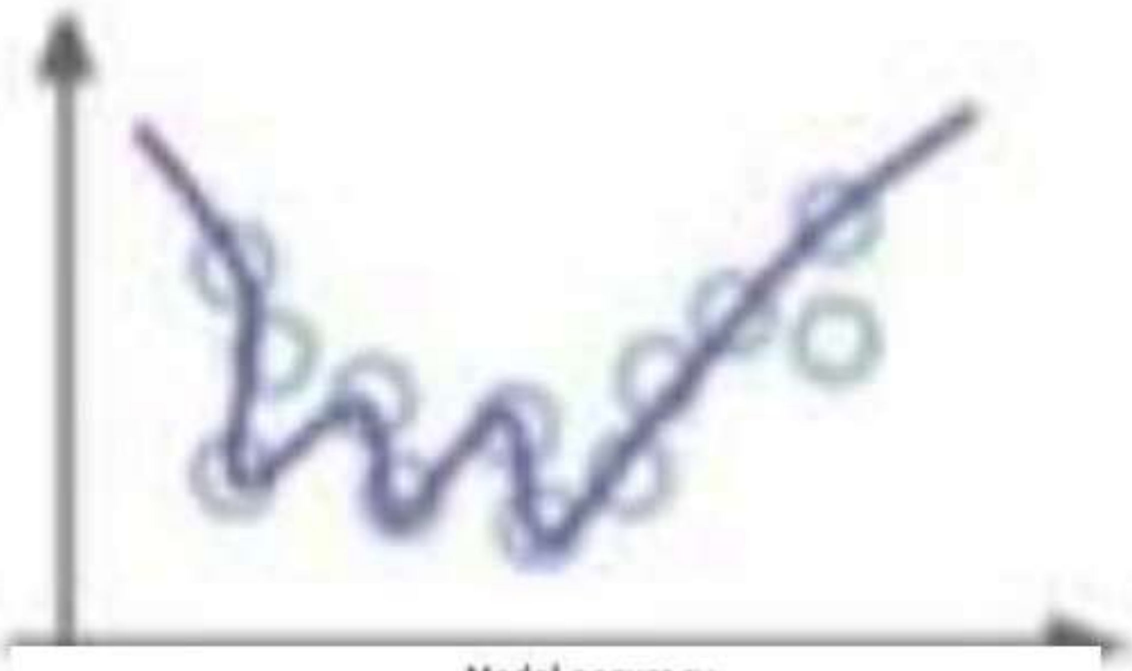
STAGE 7

Evaluation

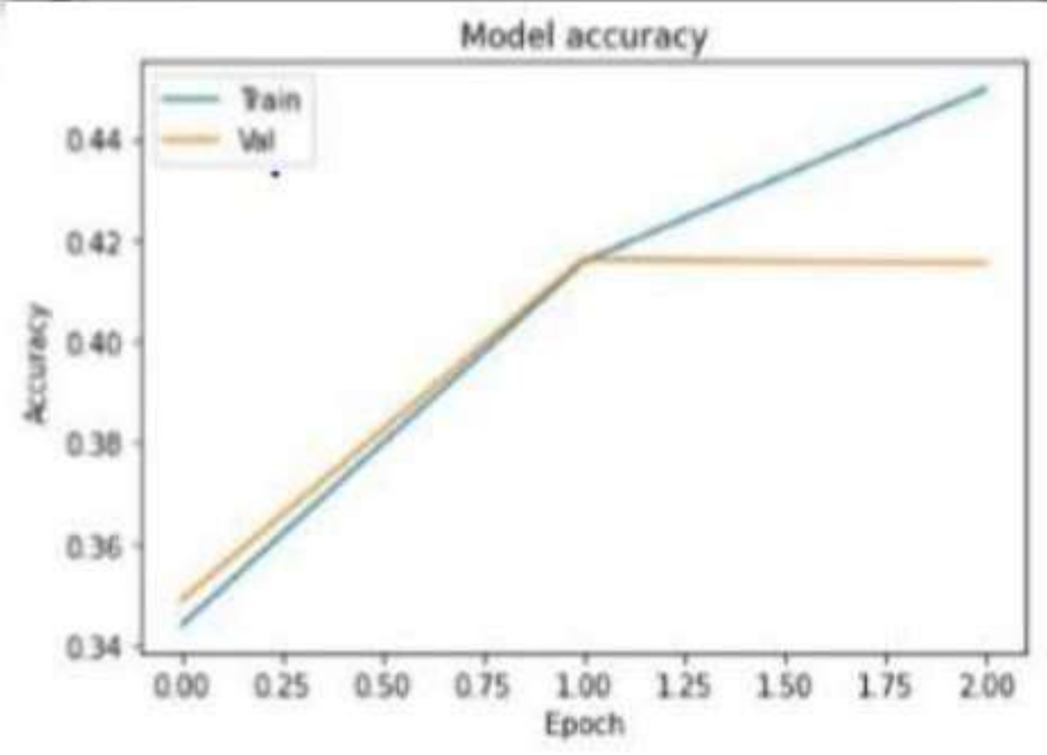
Activity 2.7.1

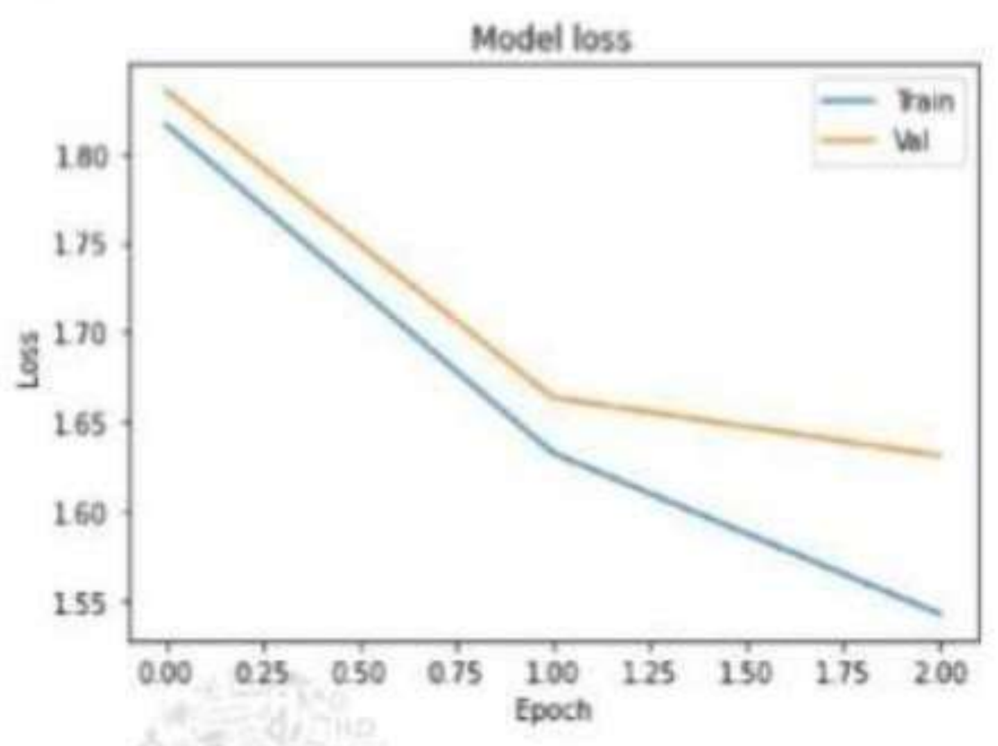
• Evaluate your ANN model's fit to the training data. Conclude whether your model is – underfit, overfit,or a good fit.

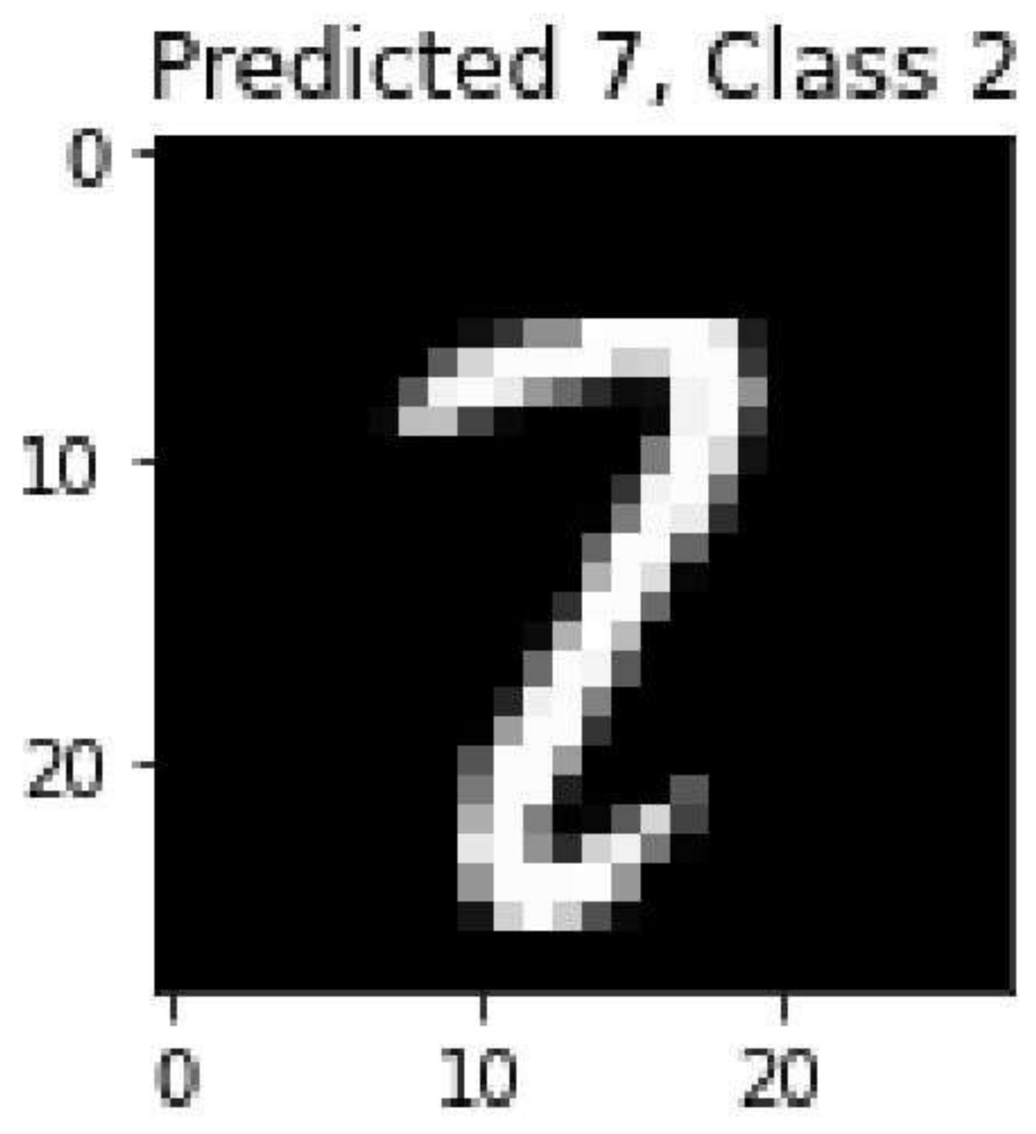
• Test your model by using a random digit image to see if it is capable of correctly identifying new images (untrained data).



these graphs show the average data tested which includes test dataset, validation dataset and training data set. We can conclude that the ANN model is **overfit model**.







As the model is not trained for this new image, we can notice that there's uncertainty in the result.

The CNN predicted that this image refers to number 7 ,But actually it is 2 .

With time and more practice the CNN model will gain experience and be able to distinguish easily between numbers embedded in images

Activity 2.7.2

Use the following questions to guide your evaluation:

How well does your program satisfy the brief?

Our program will satisfy the brief perfectly, based on the testing and training data reported various times .

How well does the overall model perform?

According to what we have got from results we can say that the project will perform just as wanted and maybe better .

How could it be improved?

We have putted three future plans for improving the ANN program :
1. Updating the code regularly
2. Publish it to the Audience so that we can gather more information from them (practice and repeat)
3. Using newest methods of technology to adapt Our ANN with it .

State two things that went well.

1. Programing the ANN to fulfill the brief.
2. Testing and creating the data plot .

State two things that could be improved.

1. The operating speed of the ANN program.
2. The ease of access using the program of the ANN

Business plan

As an entrepreneur and an aspiring engineer, you need to demonstrate how your model can be helpful for various applications that usedigit recognition.

Your business plan should prepare you to present your business idea to potential investors, such as banks. Highlight the advantages of your model, that will encourage the investor to accept your proposal.

Student reflection

List three things you have learned and two things you have enjoyed.

Three things I have learned:

1- How to program a neural network.

2- How does ANN operate.

3- different types of codes can be written with the same function.

Two things I have enjoyed:

1-programing and comparing codes

2-testing and training the ANN .

Learning outcome	Key skills (Please tick the box to show your understanding of the skills below).	I don't understand.	I understand.	I'm an expert.
Develop a simple neural network algorithm to solve a problem.	I can develop a neural network algorithm to make a prediction.			
Use training and testing datasets to build a model.	I can differentiate between training, validation, and test sets.			
Use learning curves to diagnose the performance of a machine learning model.	I can use learning curves on the train and validation datasets to diagnose an underfit, overfit, or well-fit model.			
Analyse the main sections of an advanced design brief.	I can identify key words from the brief.			
	I can identify key areas of the brief			
Address the constraints and requirements of an advanced design problem.	I can create a mindmap to show the constraints and requirments of the brief.			
Apply different methods of research to solve an advanced design problem.	I can complete primary and secondary research on the given project.			
	I can create a mood board to display information gathered.			
Transform research ideas into possible solutions for an advanced design problem.	I can create at least two possible solutions for the design problem.			

Construct a prototype to solve an advanced design problem through various prototyping means.	I can use creativity and imagination when applying iterative design processes to develop and modify designs.			
Test a design prototype for operational expectations solving an advanced design problem.	I can check prototype against design specifications and analyse results.			
Evaluate the success of implementing the proposed design idea solving an advanced design problem.	I can determine if the final prototype now fully satisfies the brief.			
Improve a design prototype to overcome identified design faults.	I can modify the design if needed.			
Create a business plan and work within a team to achieve it.	I can form a team, create a brand, conduct market research, and develop a business plan accordingly.			
Teacher's comment:				