```
import keras
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
from keras.utils import np_utils
# 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)
#reshape the input image to one dimension
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)
# One Hot Encoder for the labels of the mnist dataset.
y_train = np_utils.to_categorical(y_train, 10)
y_test = np_utils.to_categorical(y_test, 10)
print(y_train)
#set all input image as a type float32
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
#Normalise the input data. The pixels values is change from the range
# of 0 - 255 to 0 - 1 for the better accuracy of the neural networks.
x_train /= 255
x_test /= 255
```

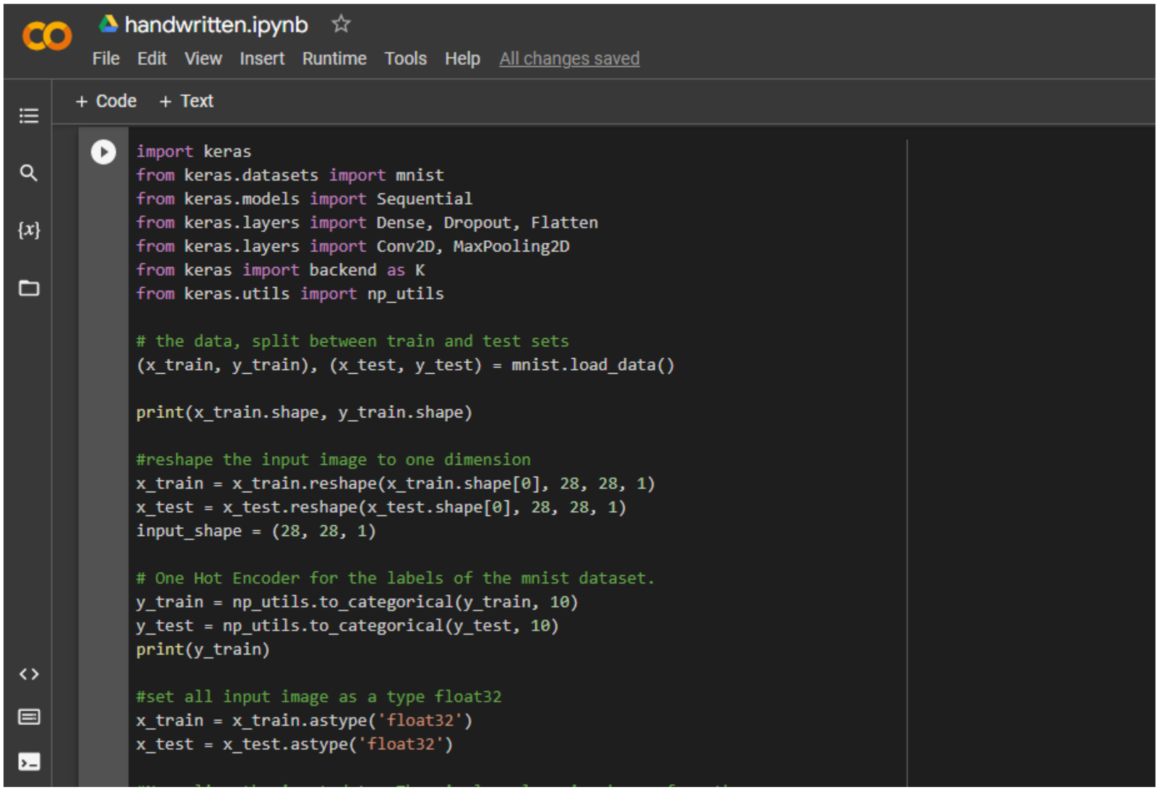
```
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
batch_size = 128
num_classes = 10
epochs = 10
# relu activation for the input layer and softmax activation for the output layers.
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 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'))
#The Adam optimizer is used for compile the neural network
model.compile(loss='categorical_crossentropy',optimizer="Adam",metrics=['accuracy'])
#Train the model with 10 epochs, i.e 10 iteration of the input data.
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")
#The accuracy of the Convolutional Neural Network model is 0.99
score = model.evaluate(x_test, y_test, verbose=0)
```

print('Test loss:', score[0])

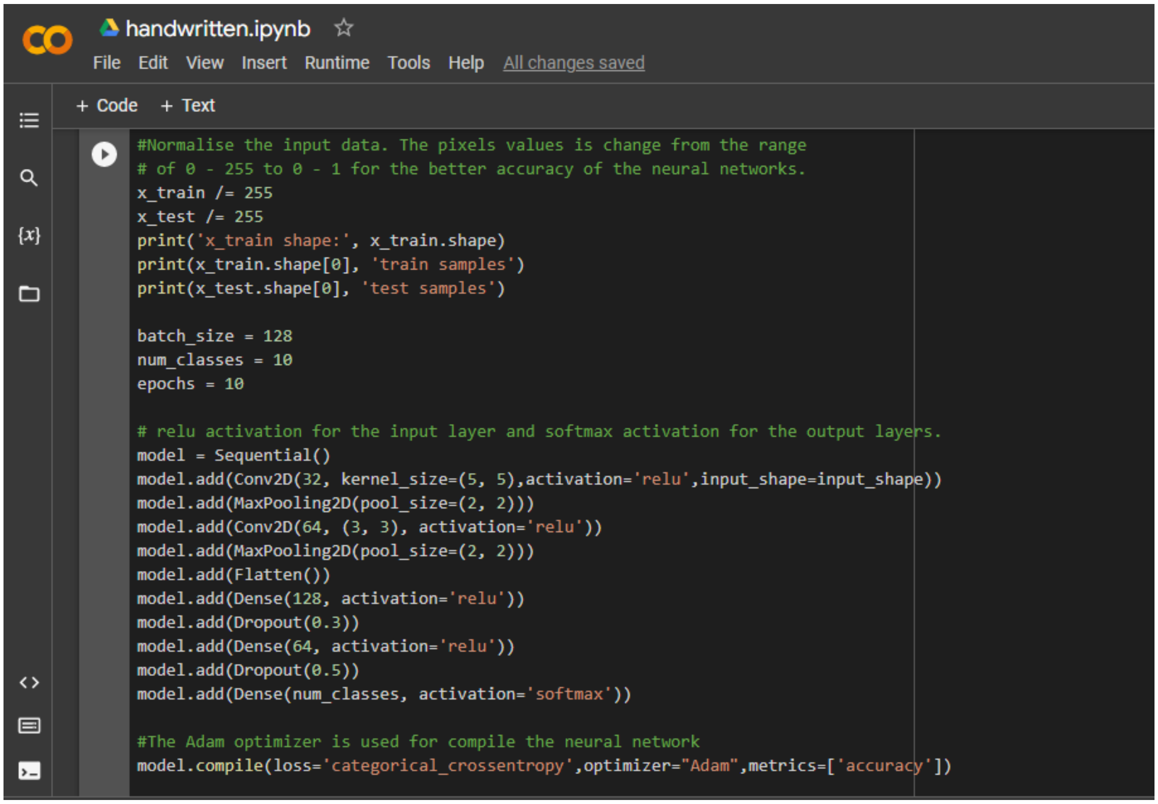
print('Test accuracy:', score[1])

model.save('mnist.h5')

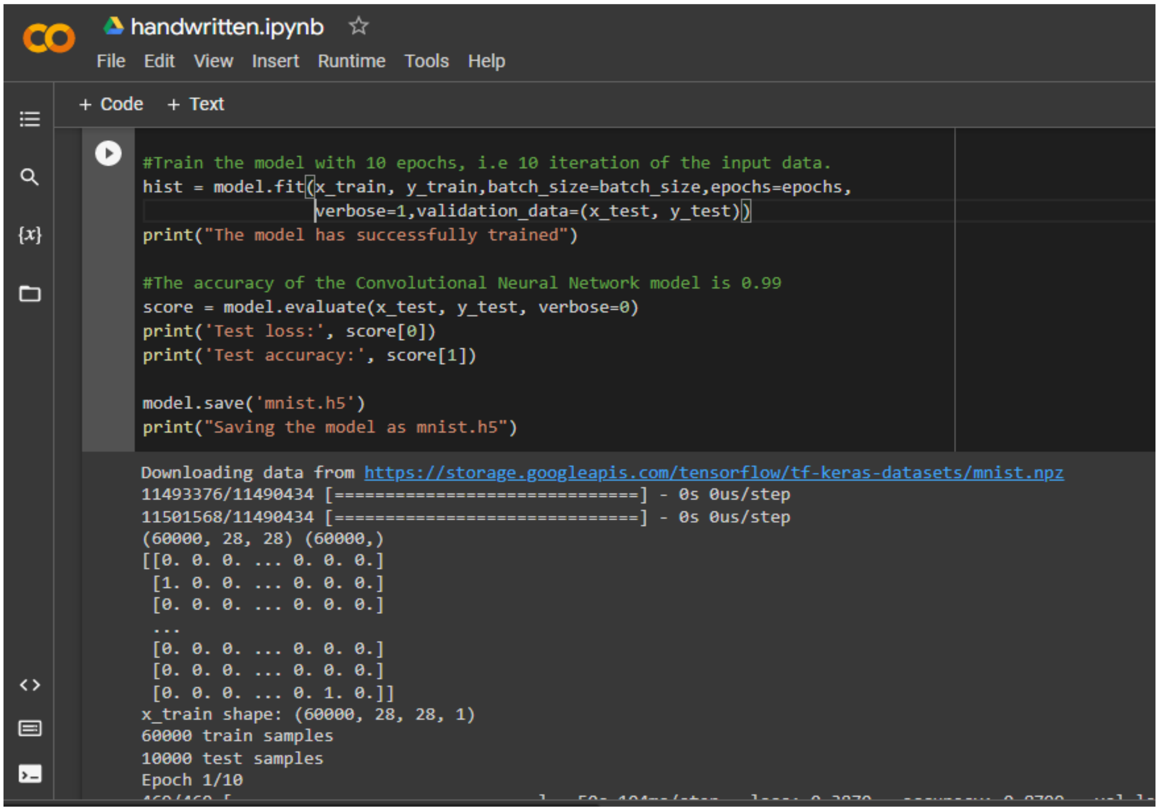
print("Saving the model as mnist.h5")



```
import keras
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
from keras.utils import np_utils

# 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)

#reshape the input image to one dimension
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)

# One Hot Encoder for the labels of the mnist dataset.
y_train = np_utils.to_categorical(y_train, 10)
y_test = np_utils.to_categorical(y_test, 10)
print(y_train)

#set all input image as a type float32
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```

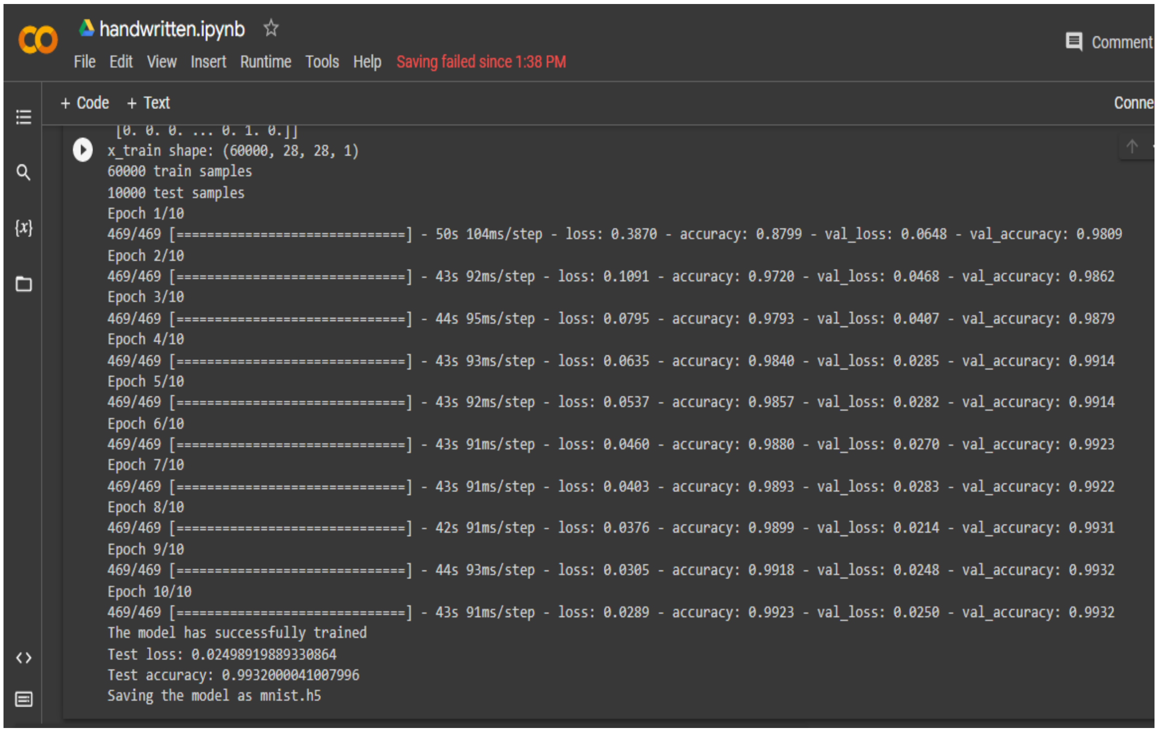


```
#Normalise the input data. The pixels values is change from the range
# of 0 - 255 to 0 - 1 for the better accuracy of the neural networks.
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

batch_size = 128
num_classes = 10
epochs = 10

# relu activation for the input layer and softmax activation for the output layers.
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'))

#The Adam optimizer is used for compile the neural network
model.compile(loss='categorical_crossentropy',optimizer="Adam",metrics=['accuracy'])
```



```
#Train the model with 10 epochs, i.e 10 iteration of the input data.
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")

#The accuracy of the Convolutional Neural Network model is 0.99
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")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
(60000, 28, 28) (60000,)
[[0. 0. 0. ... 0. 0. 0.]
 [1. 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. 1. 0.]]
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Epoch 1/10
```



```
 [0. 0. 0. ... 0. 1. 0.]]
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Epoch 1/10
469/469 [==============================] - 50s 104ms/step - loss: 0.3870 - accuracy: 0.8799 - val_loss: 0.0648 - val_accuracy: 0.9809
Epoch 2/10
469/469 [==============================] - 43s 92ms/step - loss: 0.1091 - accuracy: 0.9720 - val_loss: 0.0468 - val_accuracy: 0.9862
Epoch 3/10
469/469 [==============================] - 44s 95ms/step - loss: 0.0795 - accuracy: 0.9793 - val_loss: 0.0407 - val_accuracy: 0.9879
Epoch 4/10
469/469 [==============================] - 43s 93ms/step - loss: 0.0635 - accuracy: 0.9840 - val_loss: 0.0285 - val_accuracy: 0.9914
Epoch 5/10
469/469 [==============================] - 43s 92ms/step - loss: 0.0537 - accuracy: 0.9857 - val_loss: 0.0282 - val_accuracy: 0.9914
Epoch 6/10
469/469 [==============================] - 43s 91ms/step - loss: 0.0460 - accuracy: 0.9880 - val_loss: 0.0270 - val_accuracy: 0.9923
Epoch 7/10
469/469 [==============================] - 43s 91ms/step - loss: 0.0403 - accuracy: 0.9893 - val_loss: 0.0283 - val_accuracy: 0.9922
Epoch 8/10
469/469 [==============================] - 42s 91ms/step - loss: 0.0376 - accuracy: 0.9899 - val_loss: 0.0214 - val_accuracy: 0.9931
Epoch 9/10
469/469 [==============================] - 44s 93ms/step - loss: 0.0305 - accuracy: 0.9918 - val_loss: 0.0248 - val_accuracy: 0.9932
Epoch 10/10
469/469 [==============================] - 43s 91ms/step - loss: 0.0289 - accuracy: 0.9923 - val_loss: 0.0250 - val_accuracy: 0.9932
The model has successfully trained
Test loss: 0.02498919889330864
Test accuracy: 0.9932000041007996
Saving the model as mnist.h5
```