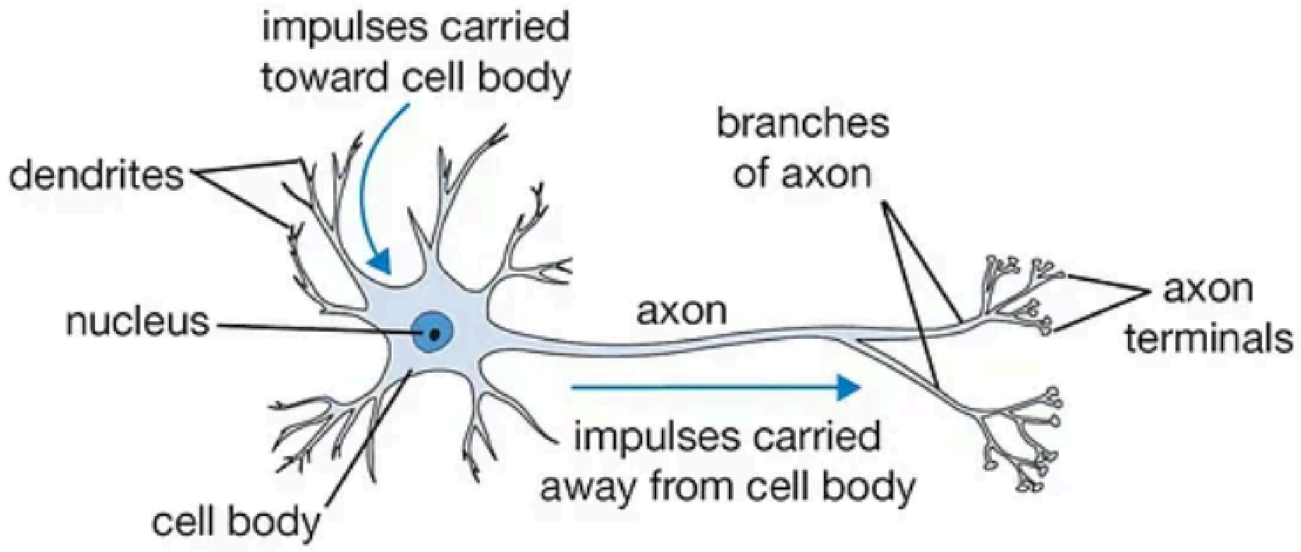
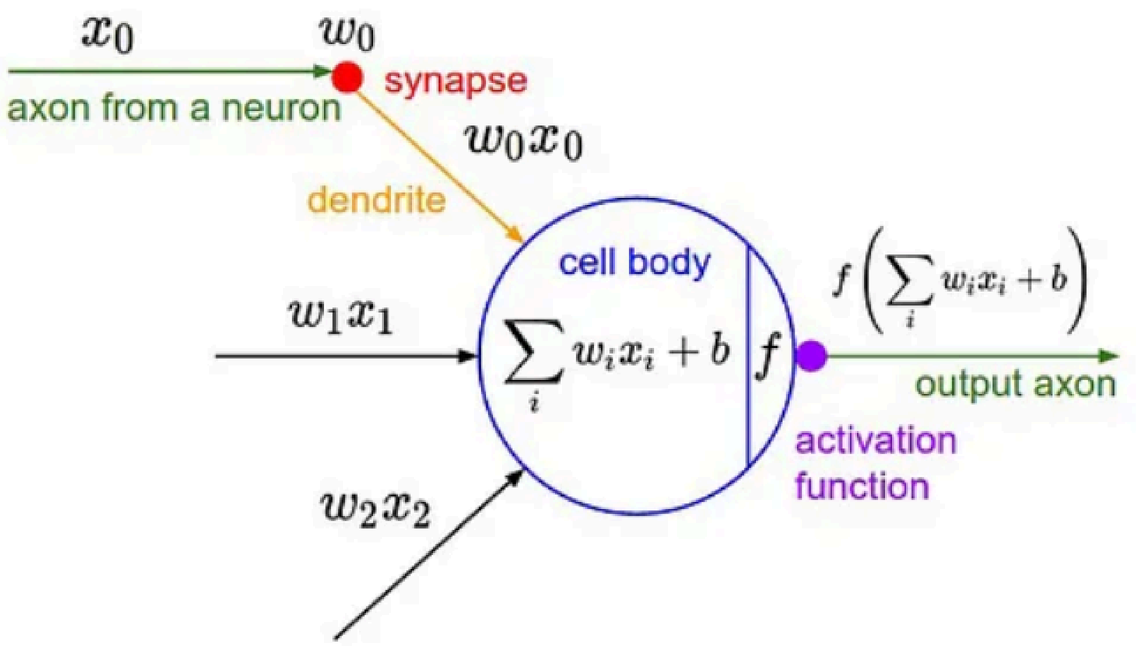
MNIST DIGITS CLASSIFICATION NEURAL NETWORK

Presented By : M Tsani Faishal Azhar

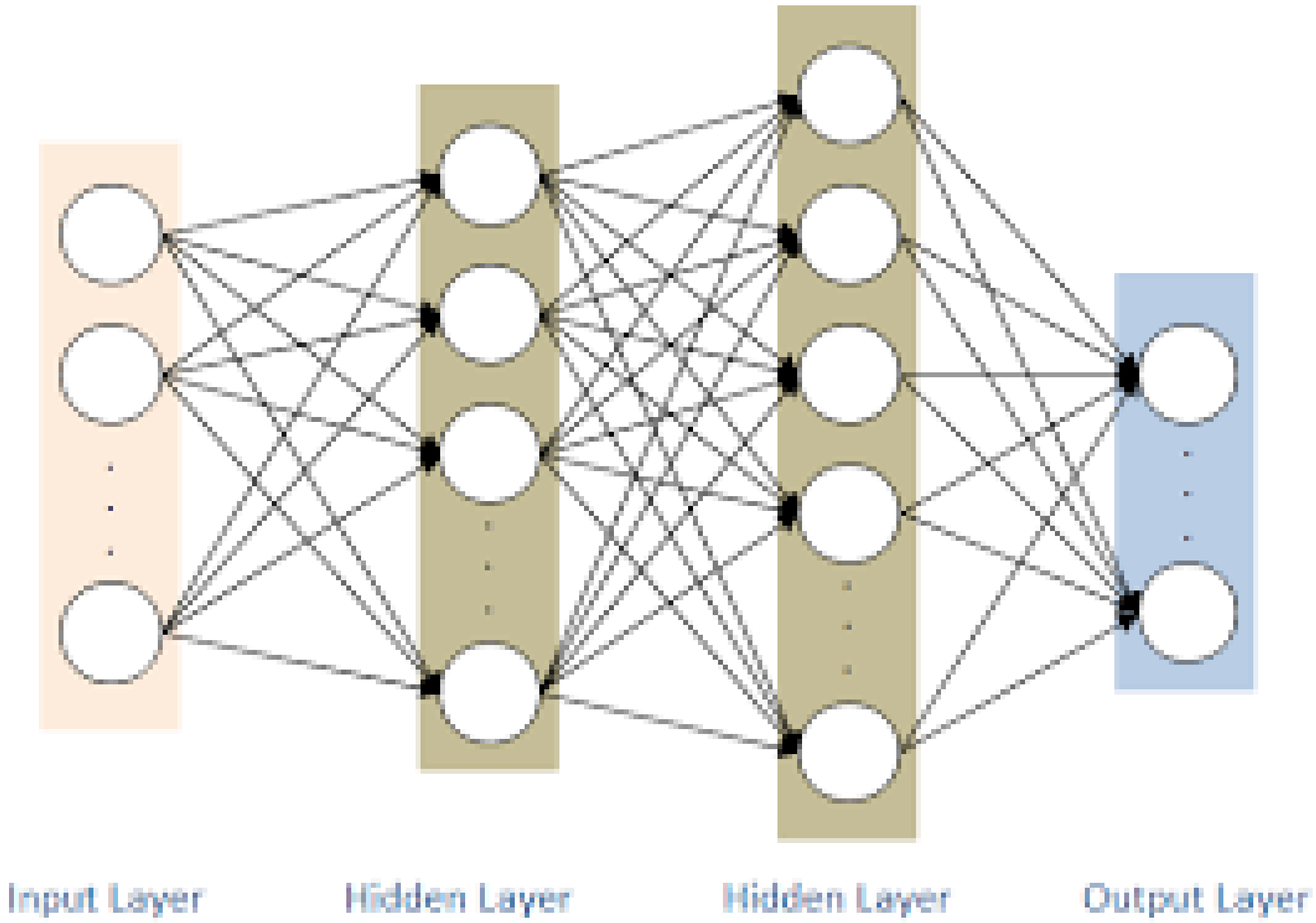
Telkom University | 2024

NEURAL NETWORK

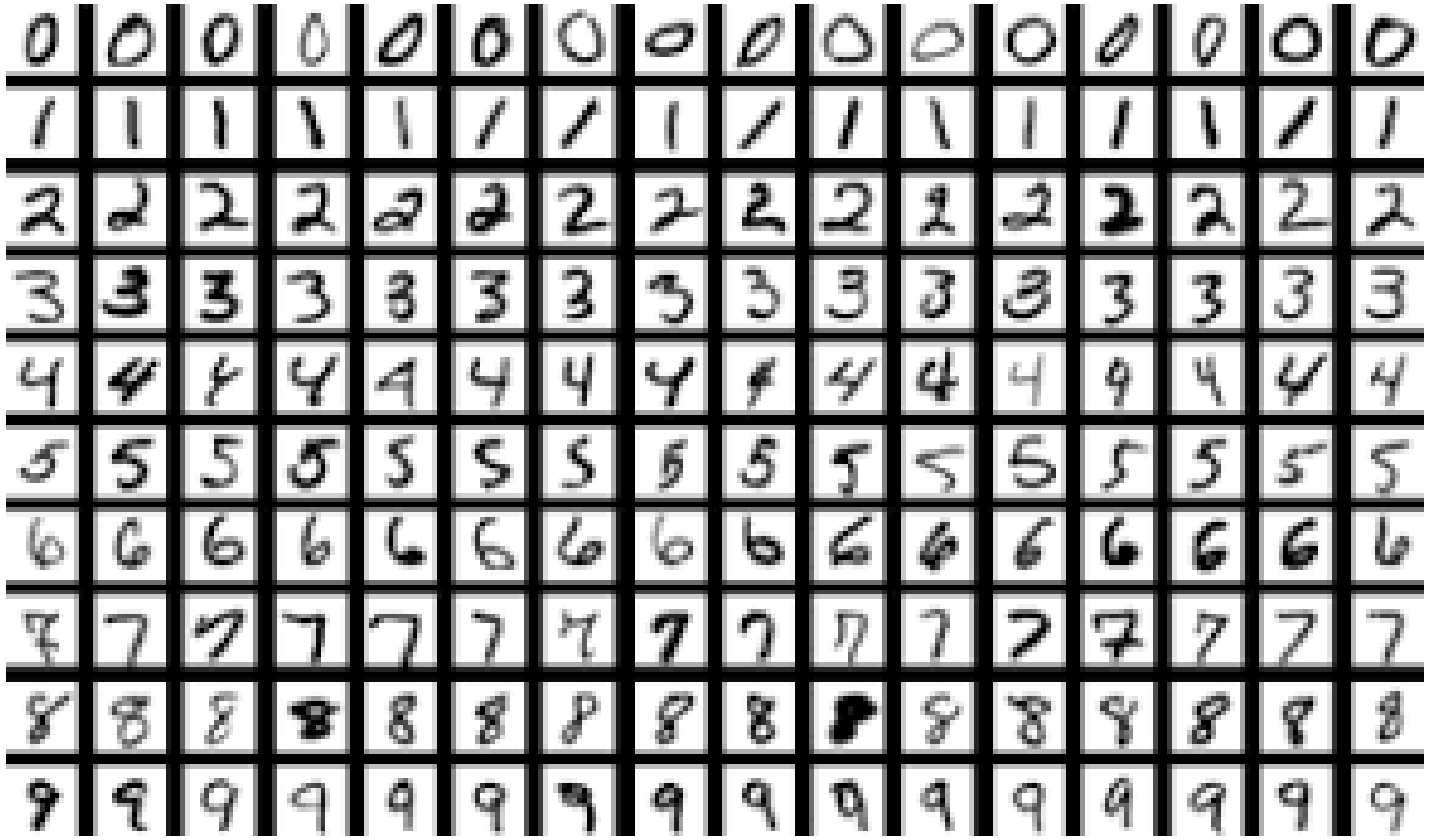

impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body



FCNN

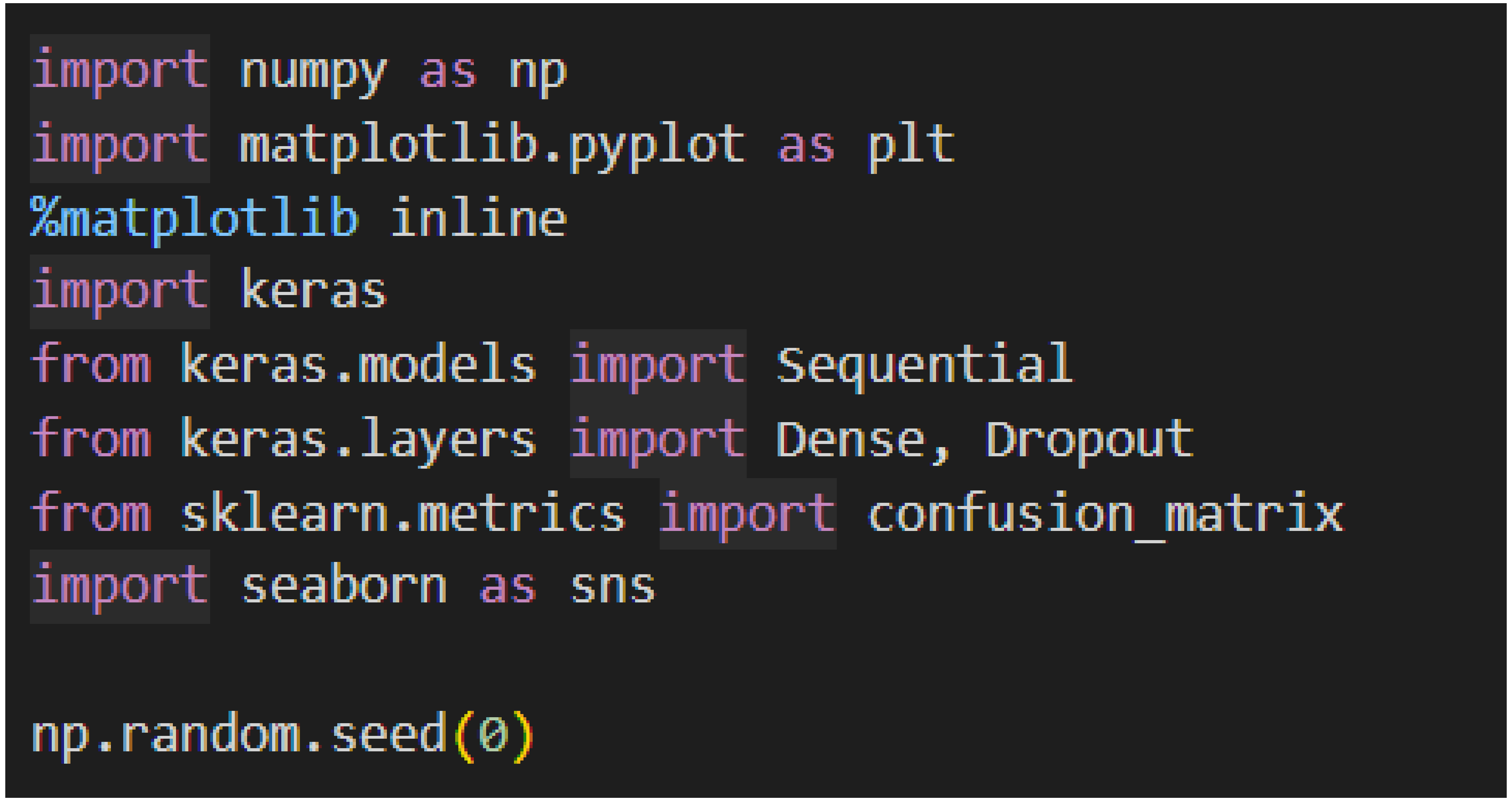


MNIST



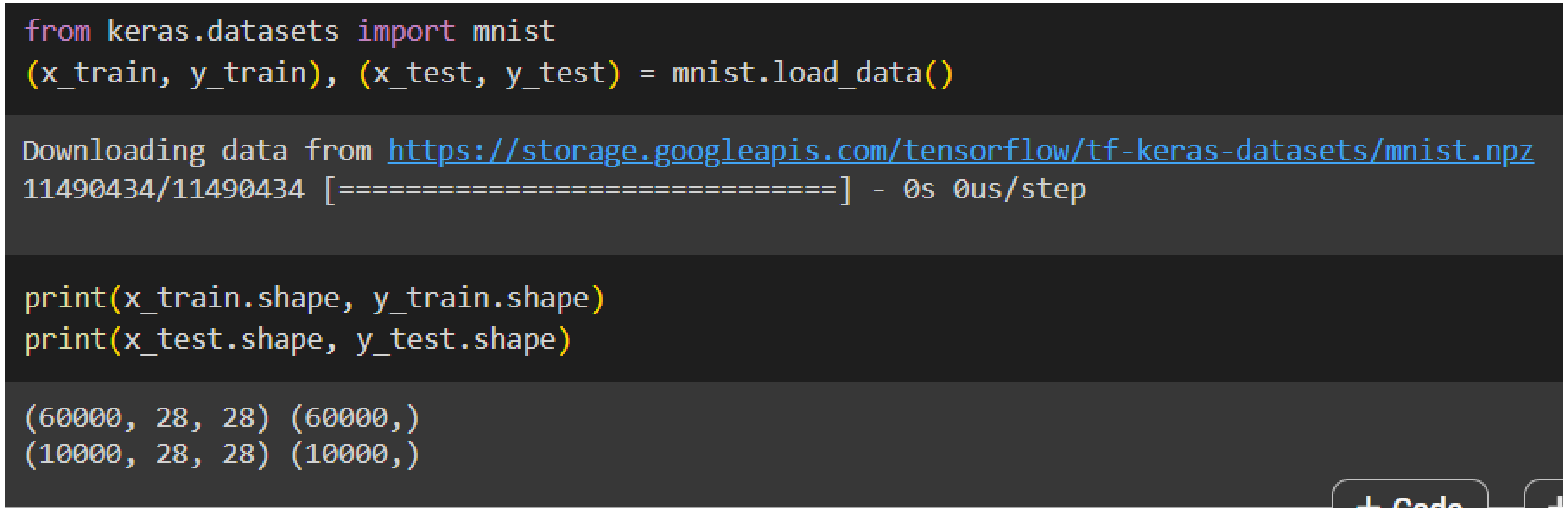
IMPLEMENTATION

LIBRARY



```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import confusion_matrix
import seaborn as sns

np.random.seed(0)
```

DATA



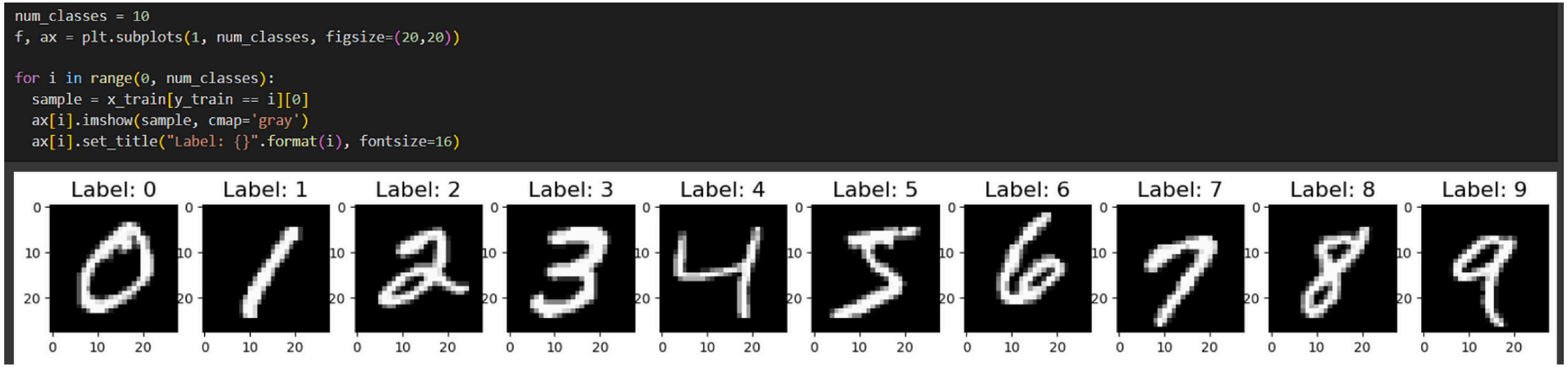
```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

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

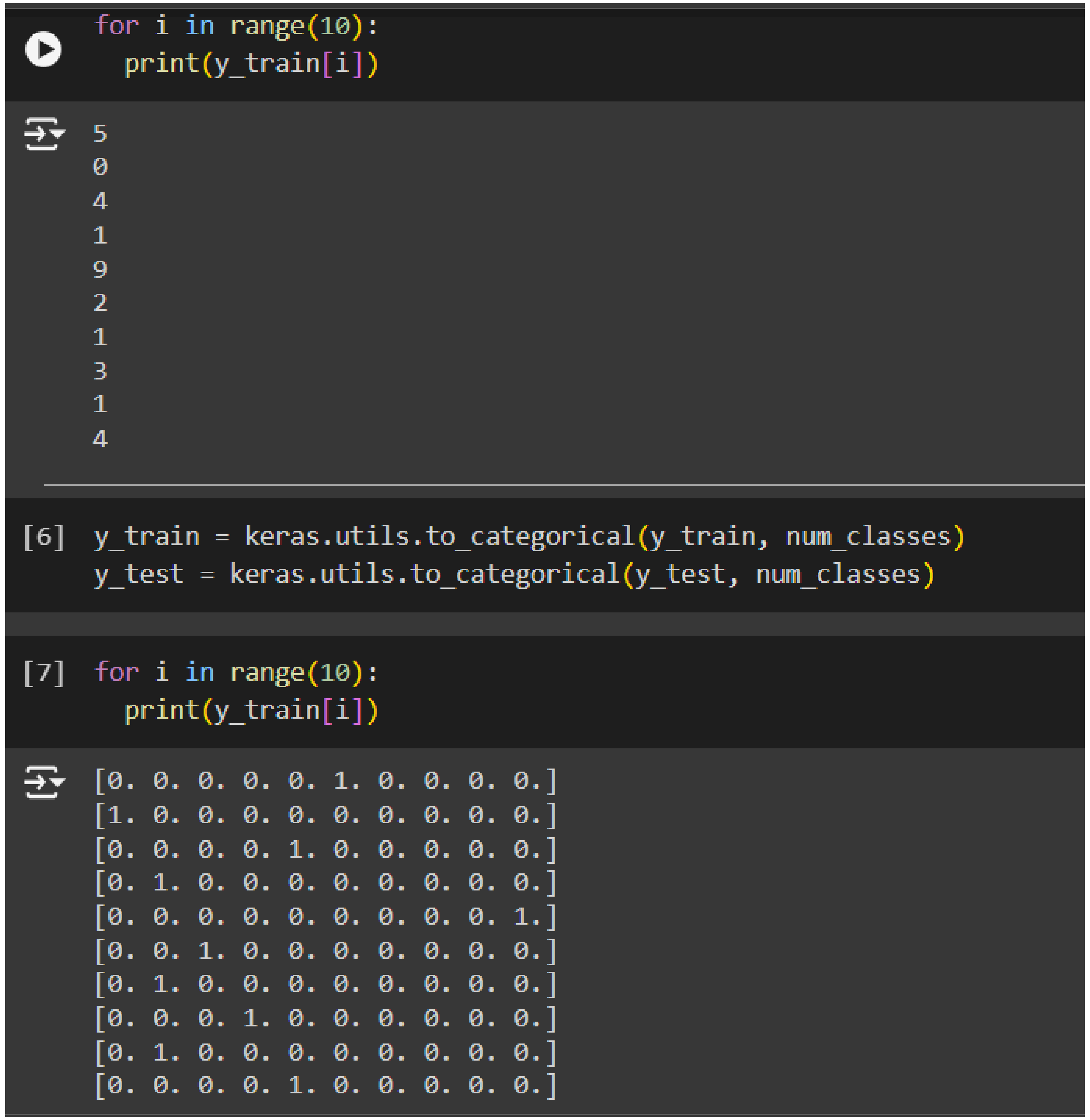
```
(60000, 28, 28) (60000,)
(10000, 28, 28) (10000,)
```

VISUALIZE EXAMPLES



```
num_classes = 10
f, ax = plt.subplots(1, num_classes, figsize=(20,20))

for i in range(0, num_classes):
  sample = x_train[y_train == i][0]
  ax[i].imshow(sample, cmap='gray')
  ax[i].set_title("Label: {}".format(i), fontsize=16)
```

VISUALIZE EXAMPLES



```
for i in range(10):
  print(y_train[i])
```

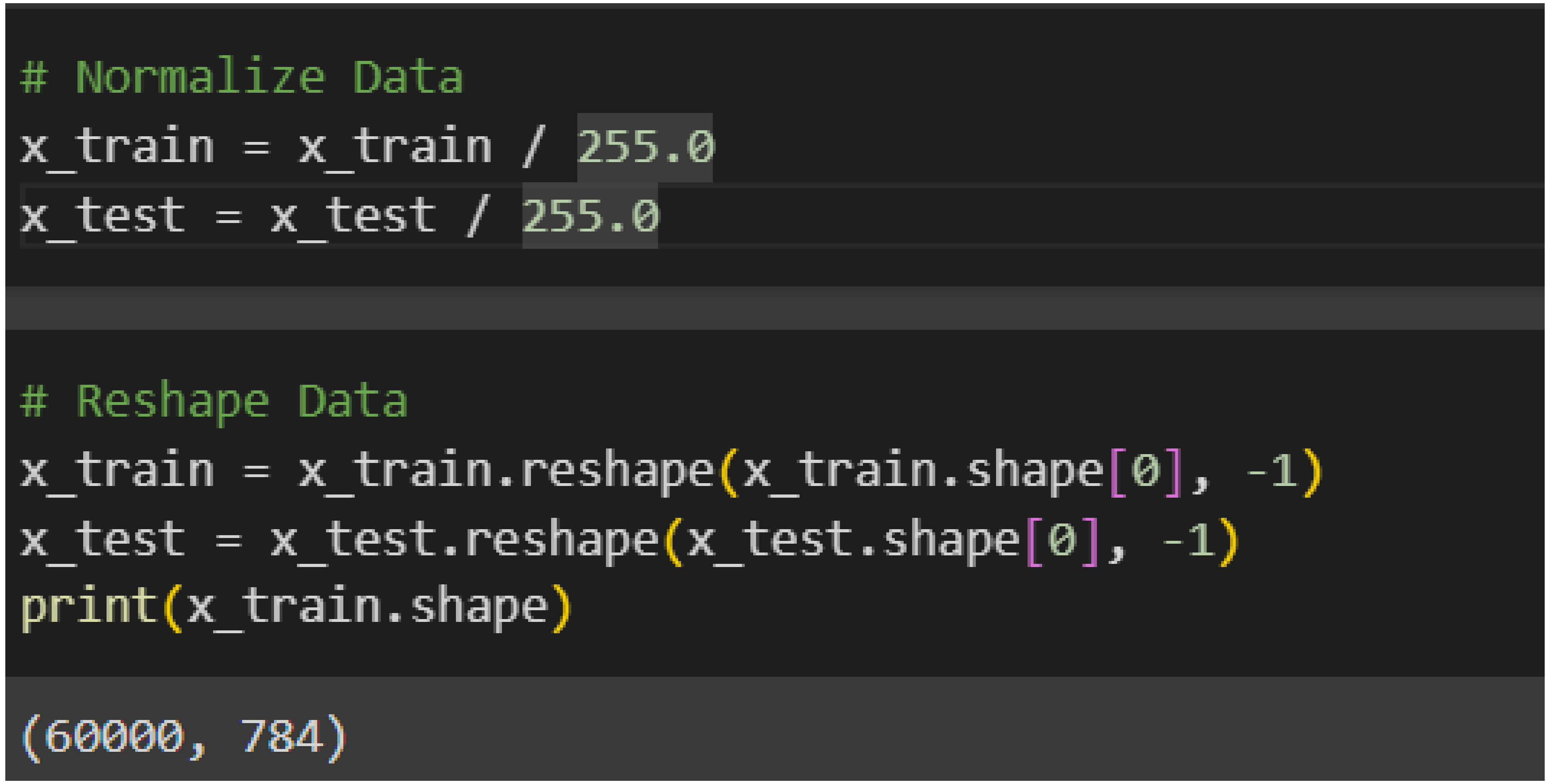
```
5
0
4
1
9
2
1
3
1
4
```

```
[6] y_train = keras.utils.to_categorical(y_train, num_classes)
    y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
[7] for i in range(10):
      print(y_train[i])
```

```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```

PREPROCESSING

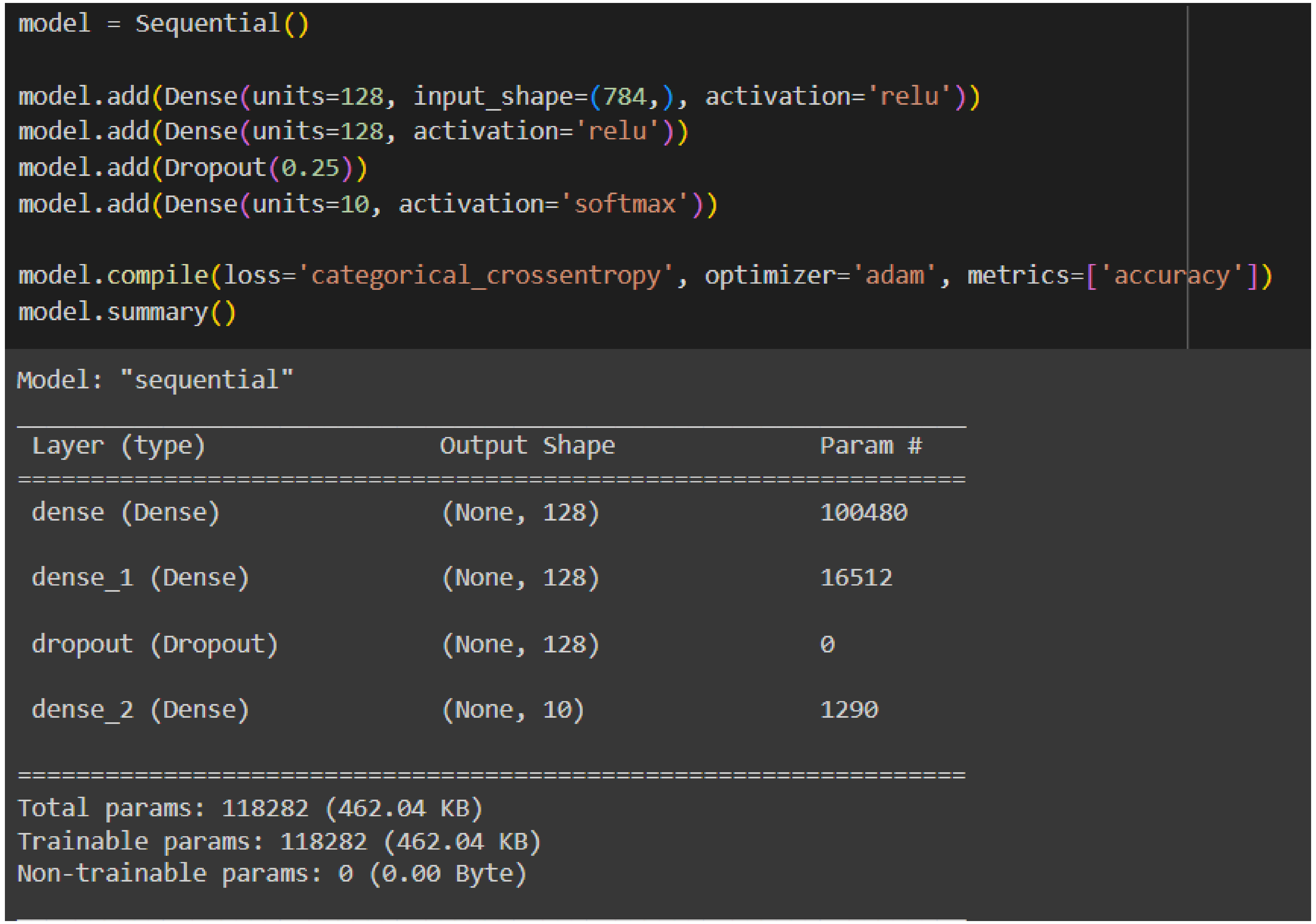


```
# Normalize Data
x_train = x_train / 255.0
x_test = x_test / 255.0
```

```
# Reshape Data
x_train = x_train.reshape(x_train.shape[0], -1)
x_test = x_test.reshape(x_test.shape[0], -1)
print(x_train.shape)
```

```
(60000, 784)
```

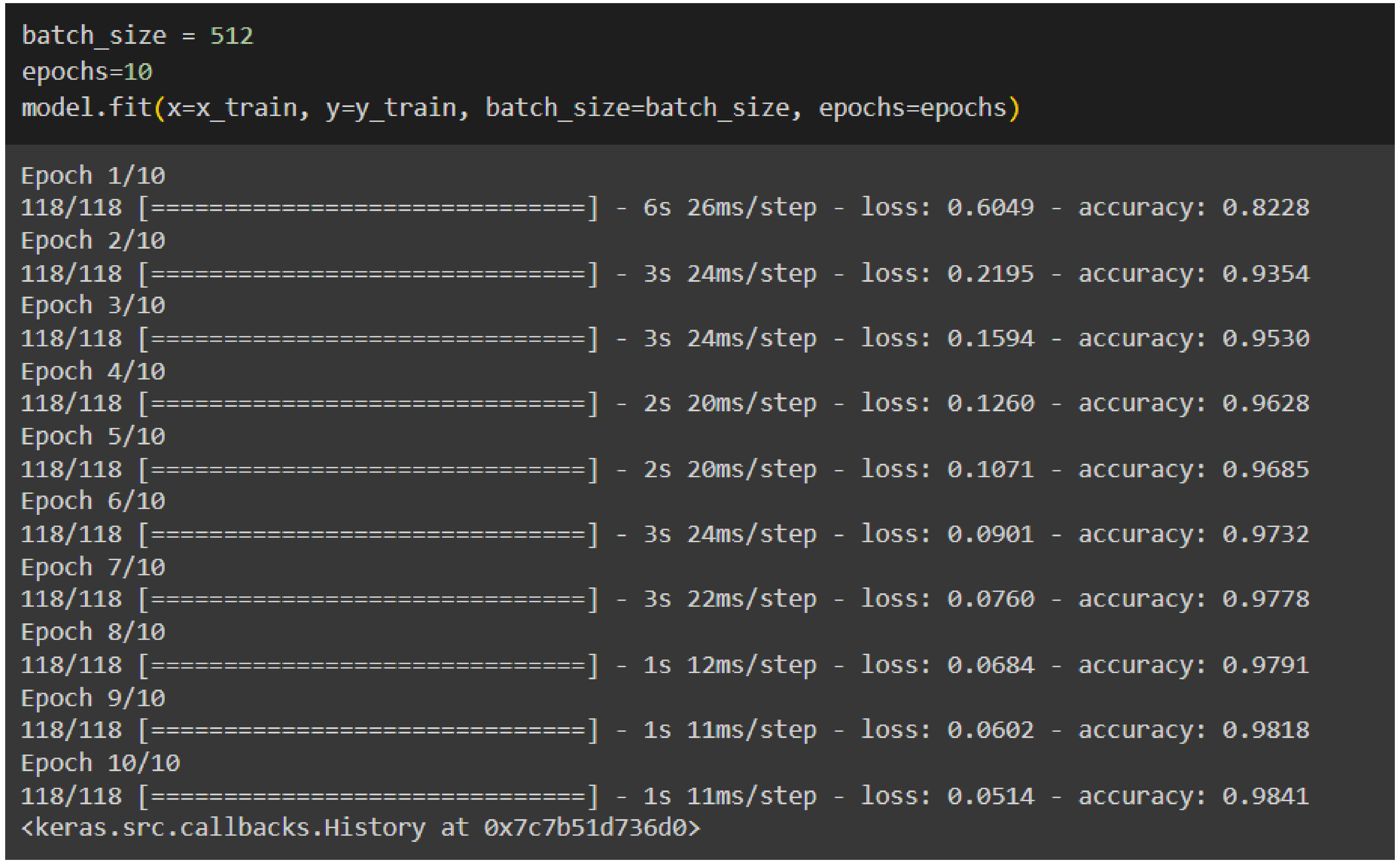
CREATE MODEL



```
model = Sequential()

model.add(Dense(units=128, input_shape=(784,), activation='relu'))
model.add(Dense(units=128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(units=10, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 128)               16512

 dropout (Dropout)           (None, 128)               0

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 118282 (462.04 KB)
Trainable params: 118282 (462.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

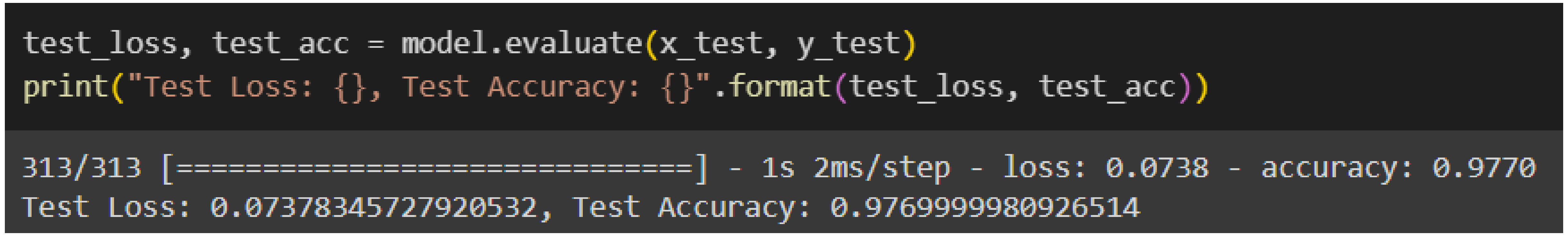
TRAIN



```
batch_size = 512
epochs=10
model.fit(x=x_train, y=y_train, batch_size=batch_size, epochs=epochs)
```

```
Epoch 1/10
118/118 [==============================] - 6s 26ms/step - loss: 0.6049 - accuracy: 0.8228
Epoch 2/10
118/118 [==============================] - 3s 24ms/step - loss: 0.2195 - accuracy: 0.9354
Epoch 3/10
118/118 [==============================] - 3s 24ms/step - loss: 0.1594 - accuracy: 0.9530
Epoch 4/10
118/118 [==============================] - 2s 20ms/step - loss: 0.1260 - accuracy: 0.9628
Epoch 5/10
118/118 [==============================] - 2s 20ms/step - loss: 0.1071 - accuracy: 0.9685
Epoch 6/10
118/118 [==============================] - 3s 24ms/step - loss: 0.0901 - accuracy: 0.9732
Epoch 7/10
118/118 [==============================] - 3s 22ms/step - loss: 0.0760 - accuracy: 0.9778
Epoch 8/10
118/118 [==============================] - 1s 12ms/step - loss: 0.0684 - accuracy: 0.9791
Epoch 9/10
118/118 [==============================] - 1s 11ms/step - loss: 0.0602 - accuracy: 0.9818
Epoch 10/10
118/118 [==============================] - 1s 11ms/step - loss: 0.0514 - accuracy: 0.9841
<keras.src.callbacks.History at 0x7c7b51d736d0>
```

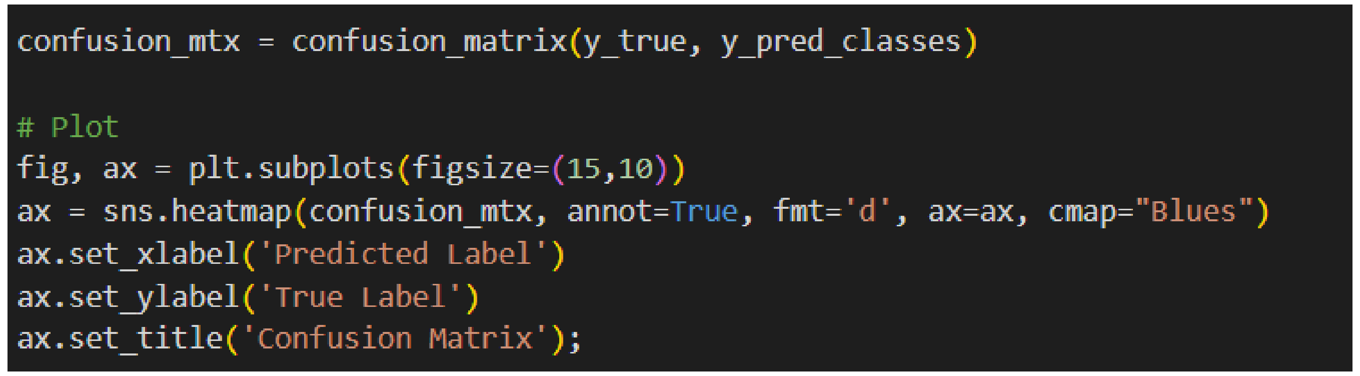
EVALUATE



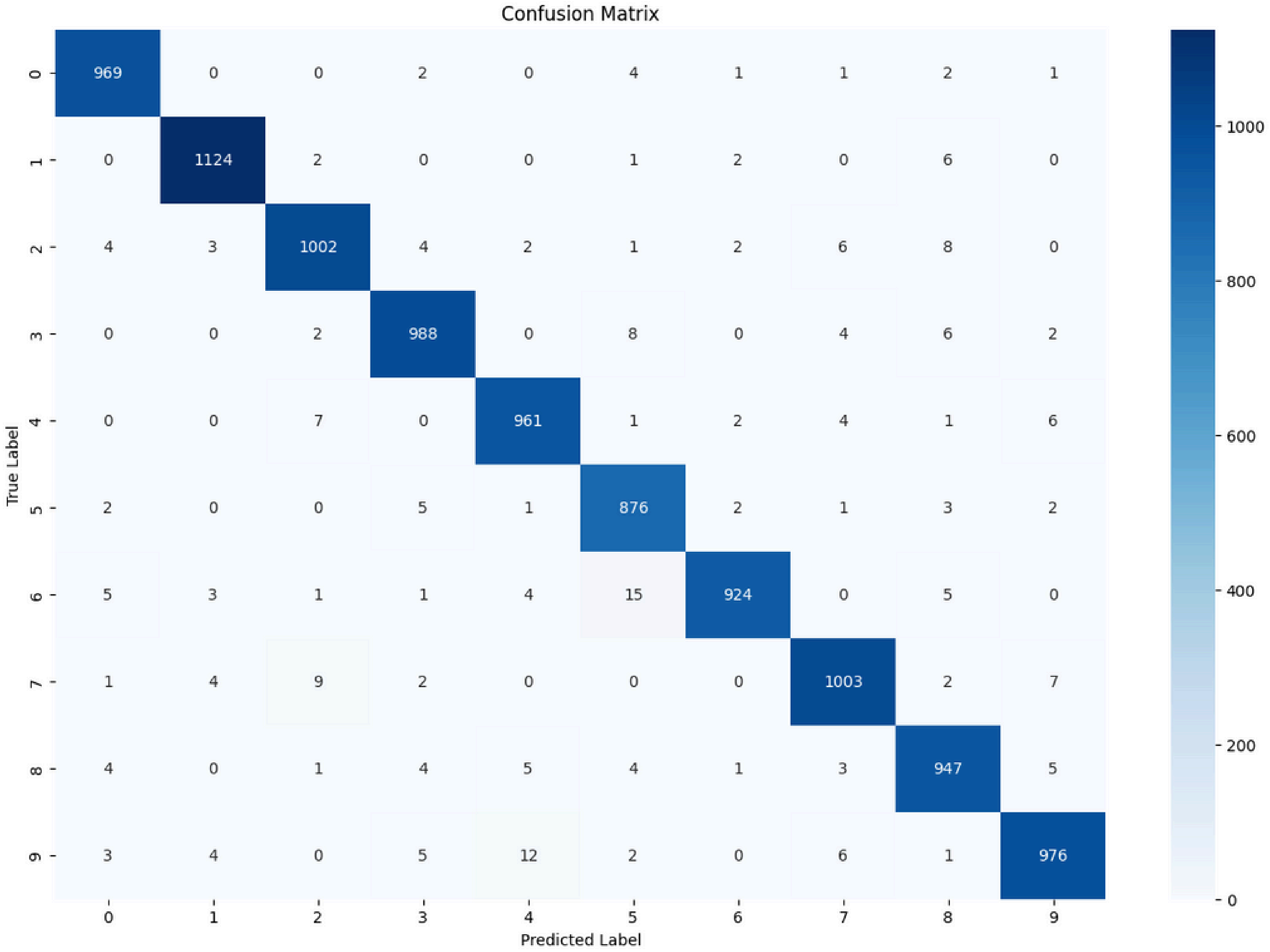
```
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test Loss: {}, Test Accuracy: {}".format(test_loss, test_acc))
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.0738 - accuracy: 0.9770
Test Loss: 0.07378345727920532, Test Accuracy: 0.9769999980926514
```

CONFUSION MATRIX



```
confusion_mtx = confusion_matrix(y_true, y_pred_classes)

# Plot
fig, ax = plt.subplots(figsize=(15,10))
ax = sns.heatmap(confusion_mtx, annot=True, fmt='d', ax=ax, cmap="Blues")
ax.set_xlabel('Predicted Label')
ax.set_ylabel('True Label')
ax.set_title('Confusion Matrix');
```



	0	1	2	3	4	5	6	7	8	9
0	969	0	0	2	0	4	1	1	2	1
1	0	1124	2	0	0	1	2	0	6	0
2	4	3	1002	4	2	1	2	6	8	0
3	0	0	2	988	0	8	0	4	6	2
4	0	0	7	0	961	1	2	4	1	6
5	2	0	0	5	1	876	2	1	3	2
6	5	3	1	1	4	15	924	0	5	0
7	1	4	9	2	0	0	0	1003	2	7
8	4	0	1	4	5	4	1	3	947	5
9	3	4	0	5	12	2	0	6	1	976

Telkom University | 2024

THANK YOU

Presented By : M Tsani Faishal Azhar