```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD


# Load the training data from CSV file
train_data = pd.read_csv('mnist_train.csv')
x_train = train_data.drop('label', axis=1).values
y_train = train_data['label'].values

# Load the testing data from CSV file
test_data = pd.read_csv('mnist_test.csv')
x_test = test_data.drop('label', axis=1).values
y_test = test_data['label'].values

num_classes = 10

# Normalize pixel values to the range [0, 1]
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# Flatten the images to a 1D array (for MNIST)
x_train = x_train.reshape((-1, 28*28))
x_test = x_test.reshape((-1, 28*28))

# Convert labels to one-hot encoding
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)


model = Sequential()
model.add(Dense(512, input_shape=(28*28,), activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 512)               401920

 dense_7 (Dense)             (None, 256)               131328

 dense_8 (Dense)             (None, 10)                2570

=================================================================
Total params: 535818 (2.04 MB)
Trainable params: 535818 (2.04 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
sgd = SGD(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=11, batch_size=128, validation_data=(x_test, y_test))
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizer
Epoch 1/11
469/469 [==============================] - 5s 11ms/step - loss: 1.0581 - accuracy: 0.7662 - val_loss: 0.4972 - val_accuracy: 0.8801
Epoch 2/11
469/469 [==============================] - 6s 12ms/step - loss: 0.4300 - accuracy: 0.8886 - val_loss: 0.3572 - val_accuracy: 0.9052
Epoch 3/11
469/469 [==============================] - 5s 10ms/step - loss: 0.3462 - accuracy: 0.9047 - val_loss: 0.3084 - val_accuracy: 0.9138
Epoch 4/11
469/469 [==============================] - 5s 11ms/step - loss: 0.3079 - accuracy: 0.9136 - val_loss: 0.2810 - val_accuracy: 0.9217
Epoch 5/11
469/469 [==============================] - 5s 10ms/step - loss: 0.2830 - accuracy: 0.9202 - val_loss: 0.2620 - val_accuracy: 0.9267
Epoch 6/11
469/469 [==============================] - 4s 9ms/step - loss: 0.2642 - accuracy: 0.9258 - val_loss: 0.2462 - val_accuracy: 0.9327
Epoch 7/11
469/469 [==============================] - 5s 12ms/step - loss: 0.2487 - accuracy: 0.9299 - val_loss: 0.2335 - val_accuracy: 0.9341
Epoch 8/11
469/469 [==============================] - 5s 10ms/step - loss: 0.2353 - accuracy: 0.9333 - val_loss: 0.2237 - val_accuracy: 0.9364
Epoch 9/11
469/469 [==============================] - 5s 10ms/step - loss: 0.2237 - accuracy: 0.9364 - val_loss: 0.2142 - val_accuracy: 0.9393
Epoch 10/11
469/469 [==============================] - 5s 11ms/step - loss: 0.2132 - accuracy: 0.9397 - val_loss: 0.2052 - val_accuracy: 0.9397
Epoch 11/11
469/469 [==============================] - 4s 9ms/step - loss: 0.2039 - accuracy: 0.9425 - val_loss: 0.1960 - val_accuracy: 0.9427
```

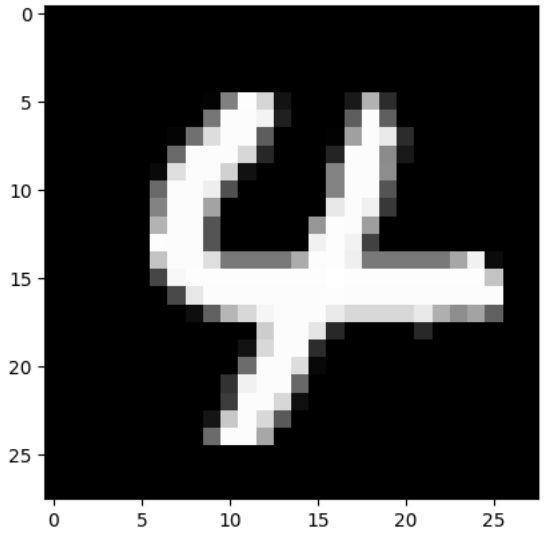
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss: ', score[0])
print('Test accuracy: ', score[1])
```

```
Test loss:  0.19603745639324188
Test accuracy:  0.9427000284194946
```

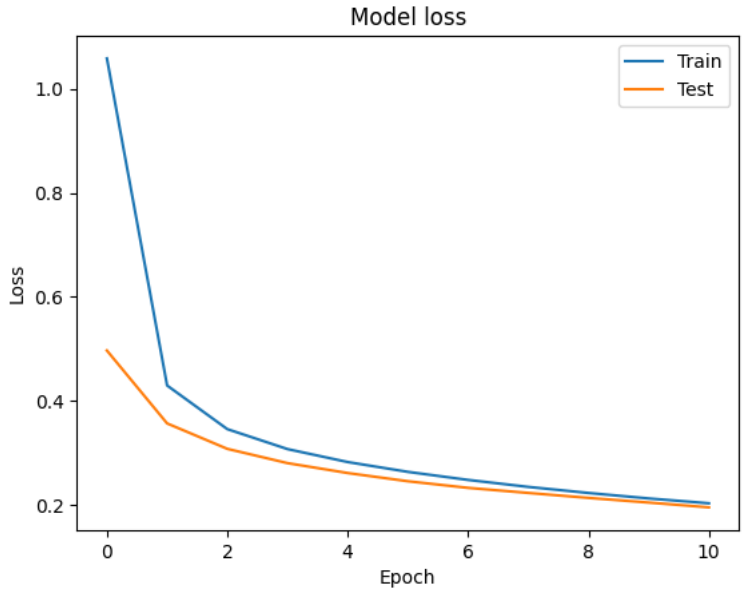
```
import random

n = random.randint(0,9999)
plt.imshow(x_test[n].reshape(28, 28), cmap='gray')
predicted_value = model.predict(x_test)
print("Actual Number: ",np.argmax(y_test[n]))
print("Predicted Number: ", np.argmax(predicted_value[n]))
```

```
313/313 [==============================] - 1s 3ms/step
Actual Number:  4
Predicted Number:  4
```




```
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper right')
plt.show()
```




```
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
```

```
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```