```
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import utils
import pandas as pd
from sklearn.metrics import classification_report,confusion_matrix
from tensorflow.keras.preprocessing import image
```

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
X_train.shape
```

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

```
X_test.shape
```

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

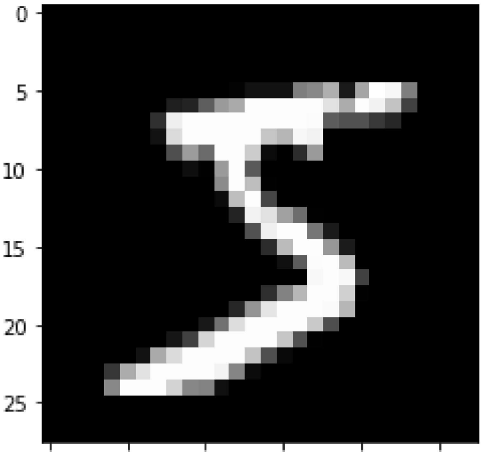
```
single_image= X_train[0]
```

```
single_image.shape
```

```
(28, 28)
```

```
plt.imshow(single_image,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f3ad22d1b90>
```



```
y_train.shape
```

```
     (60000,)
```

```
X_train.min()
```

```
     0
```

```
X_train.max()
```

```
     255
```

```
X_train_scaled = X_train/255.0
X_test_scaled = X_test/255.0
```

```
X_train_scaled.min()
```

```
     0.0
```

```
X_train_scaled.max()
```

```
     1.0
```

```
y_train[0]
```

```
     5
```

```
y_train_onehot = utils.to_categorical(y_train,10)
y_test_onehot = utils.to_categorical(y_test,10)
```

```
type(y_train_onehot)
```

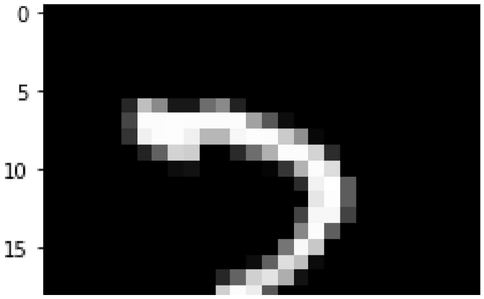
```
     numpy.ndarray
```

```
y_train_onehot.shape
```

```
     (60000, 10)
```

```
single_image = X_train[500]
plt.imshow(single_image,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f3ad25528d0>
```



```
y_train_onehot[500]
```

```
array([0., 0., 0., 1., 0., 0., 0., 0., 0., 0.], dtype=float32)
```

```
X_train_scaled = X_train_scaled.reshape(-1,28,28,1)
X_test_scaled = X_test_scaled.reshape(-1,28,28,1)
```

```
model = keras.Sequential()
model = keras.Sequential()
model.add (layers. Input (shape=(28,28,1)))
model.add (layers.Conv2D (filters=32, kernel_size=(3,3), activation='relu'))
model.add (layers.MaxPool2D (pool_size=(2,2)))
model.add (layers. Flatten())
model.add (layers.Dense (32, activation='relu'))
model.add (layers.Dense (10, activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_1 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 flatten_1 (Flatten)         (None, 5408)              0

 dense_2 (Dense)             (None, 32)                173088

 dense_3 (Dense)             (None, 10)                330

=================================================================
Total params: 173,738
Trainable params: 173,738
Non-trainable params: 0
_________________________________________________________________
```

```
# Choose the appropriate parameters
```

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

```
model.fit(X_train_scaled ,y_train_onehot, epochs=5,
          batch_size=64,
          validation_data=(X_test_scaled,y_test_onehot))
```

```
Epoch 1/5
938/938 [==============================] - 26s 27ms/step - loss: 0.2634 - accuracy: 0.9
Epoch 2/5
938/938 [==============================] - 23s 25ms/step - loss: 0.0913 - accuracy: 0.9
Epoch 3/5
938/938 [==============================] - 23s 25ms/step - loss: 0.0622 - accuracy: 0.9
Epoch 4/5
938/938 [==============================] - 23s 25ms/step - loss: 0.0472 - accuracy: 0.9
Epoch 5/5
938/938 [==============================] - 23s 24ms/step - loss: 0.0391 - accuracy: 0.9
<keras.callbacks.History at 0x7f3ad21f7590>
```

```
metrics = pd.DataFrame(model.history.history)
```

```
metrics.head()
```

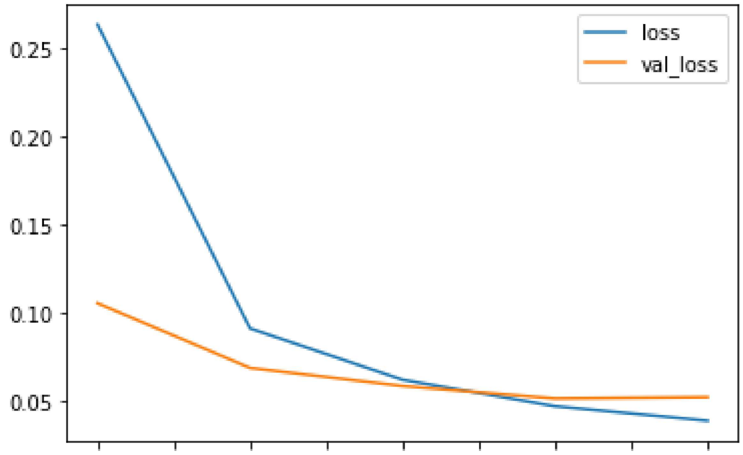
	loss	accuracy	val_loss	val_accuracy
0	0.263423	0.923300	0.105565	0.9684
1	0.091317	0.973617	0.068945	0.9766
2	0.062245	0.981667	0.058717	0.9821
3	0.047243	0.985833	0.051613	0.9817
4	0.039052	0.988117	0.052372	0.9830

```
metrics[['accuracy','val_accuracy']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3ace916e90>
```

```
metrics[['loss','val_loss']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3ad22fa290>
```



```
x_test_predictions = np.argmax(model.predict(X_test_scaled), axis=1)
```

```
print(confusion_matrix(y_test,x_test_predictions))
```

```
[[ 973    0    1    1    0    2    0    1    2    0]
 [   0 1129    2    1    0    0    2    0    1    0]
 [   2    6 1008    4    1    0    1    5    4    1]
 [   0    0    0  997    0    5    0    4    3    1]
 [   0    0    2    0  957    0    4    0    1   18]
 [   2    0    0    4    0  883    1    0    2    0]
 [   9    2    0    1    1    6  937    0    2    0]
 [   1    3    6    1    0    0    0 1009    1    7]
 [   6    0    2    3    0    3    1    5  943   11]
 [   2    2    0    4    1    3    0    2    1  994]]
```

```
print(classification_report(y_test,x_test_predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.99      0.98      0.98      1032
           3       0.98      0.99      0.98      1010
           4       1.00      0.97      0.99       982
           5       0.98      0.99      0.98       892
           6       0.99      0.98      0.98       958
           7       0.98      0.98      0.98      1028
           8       0.98      0.97      0.98       974
           9       0.96      0.99      0.97      1009
```

```
    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

Prediction for a single input

```
img = image.load_img('/content/PIC-03.png')
```

```
type(img)
```

```
PIL.Image.Image
```

```
img = image.load_img('/content/PIC-03.png')
img_tensor = tf.convert_to_tensor(np.asarray(img))
img_28 = tf.image.resize(img_tensor,(28,28))
img_28_gray = tf.image.rgb_to_grayscale(img_28)
img_28_gray_scaled = img_28_gray.numpy()/255.0
```

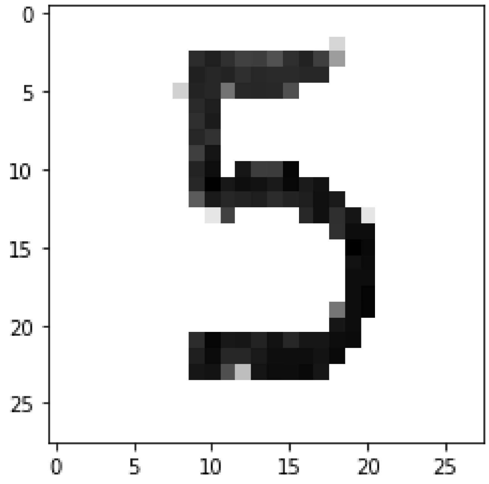
```
x_single_prediction = np.argmax(
    model.predict(img_28_gray_scaled.reshape(1,28,28,1)),
     axis=1)
```

```
print(x_single_prediction)
```

```
[8]
```

```
plt.imshow(img_28_gray_scaled.reshape(28,28),cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f3ace76cf50>
```



```
img_28_gray_inverted = 255.0-img_28_gray
img_28_gray_inverted_scaled = img_28_gray_inverted.numpy()/255.0
```

```
img_28_gray_inverted_scaled = img_28_gray_inverted.numpy()/255.0



x_single_prediction = np.argmax(
    model.predict(img_28_gray_inverted_scaled.reshape(1,28,28,1)),
     axis=1)


print(x_single_prediction)

     [5]
```

✓ 0s completed at 18:39