image classification

```
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import·random
from·keras.datasets·import·mnist
from matplotlib import pyplot
```

```
# loading
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

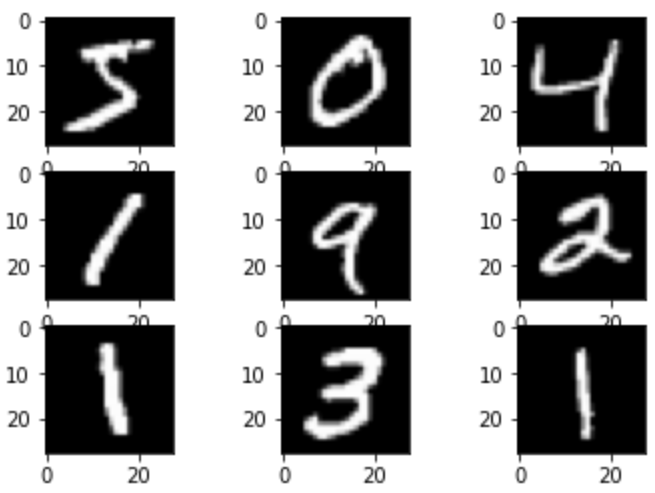
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

```
# shape of dataset
print('X_train: ' + str(x_train.shape))
print('Y_train: ' + str(y_train.shape))
print('X_test: ' + str(x_test.shape))
print('Y_test: ' + str(y_test.shape))
```

```
X_train: (60000, 28, 28)
Y_train: (60000,)
X_test: (10000, 28, 28)
Y_test: (10000,)
```

```
# plotting
from matplotlib import pyplot

for i in range(9):
  pyplot.subplot(330 + 1 + i)
  pyplot.imshow(x_train[i], cmap=pyplot.get_cmap('gray'))
pyplot.show()
```



```
#Define the network architecture using Keras
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')])
```

```
# Compile the model
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
# Train the model
history=model.fit(x_train, y_train,validation_data=(x_test,y_test),epochs=10)
```

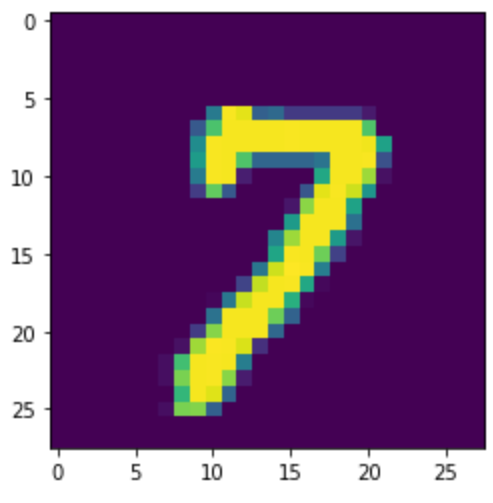
```
Epoch 1/10
1875/1875 [==============================] - 6s 3ms/step - loss: 21.9892 - accuracy: 0.1969 - val_loss: 2.1081 - val_accuracy: 0.
Epoch 2/10
1875/1875 [==============================] - 6s 3ms/step - loss: 2.1448 - accuracy: 0.1919 - val_loss: 2.1117 - val_accuracy: 0.1
Epoch 3/10
1875/1875 [==============================] - 7s 4ms/step - loss: 2.0875 - accuracy: 0.1961 - val_loss: 2.0307 - val_accuracy: 0.2
Epoch 4/10
1875/1875 [==============================] - 8s 4ms/step - loss: 2.0309 - accuracy: 0.2065 - val_loss: 2.0026 - val_accuracy: 0.2
Epoch 5/10
1875/1875 [==============================] - 12s 6ms/step - loss: 2.0436 - accuracy: 0.2181 - val_loss: 2.0353 - val_accuracy: 0.
Epoch 6/10
1875/1875 [==============================] - 7s 4ms/step - loss: 2.0242 - accuracy: 0.2079 - val_loss: 2.0155 - val_accuracy: 0.2
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 1.9734 - accuracy: 0.2363 - val_loss: 1.8774 - val_accuracy: 0.2
Epoch 8/10
1875/1875 [==============================] - 6s 3ms/step - loss: 1.9440 - accuracy: 0.2532 - val_loss: 1.9382 - val_accuracy: 0.2
```

```
Epoch 9/10
1875/1875 [==============================] - 6s 3ms/step - loss: 2.1101 - accuracy: 0.2213 - val_loss: 2.0435 - val_accuracy: 0.2
Epoch 10/10
1875/1875 [==============================] - 7s 4ms/step - loss: 2.0768 - accuracy: 0.1933 - val_loss: 2.0719 - val_accuracy: 0.1
```

```
# Evaluate the model
test_loss,test_acc=model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 2.0719 - accuracy: 0.1930
```

```
# Making Prediction on New Data
n=random.randint(0,9999)
plt.imshow(x_test[n])
plt.show()
```




```
# graph represents the model’s loss
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Training Loss and accuracy')
plt.ylabel('accuracy/Loss')
plt.xlabel('epoch')
plt.legend(['accuracy', 'val_accuracy','loss','val_loss'])
plt.show()
```