```
from keras.datasets import mnist
import matplotlib.pyplot as plt
data = mnist.load_data()
```

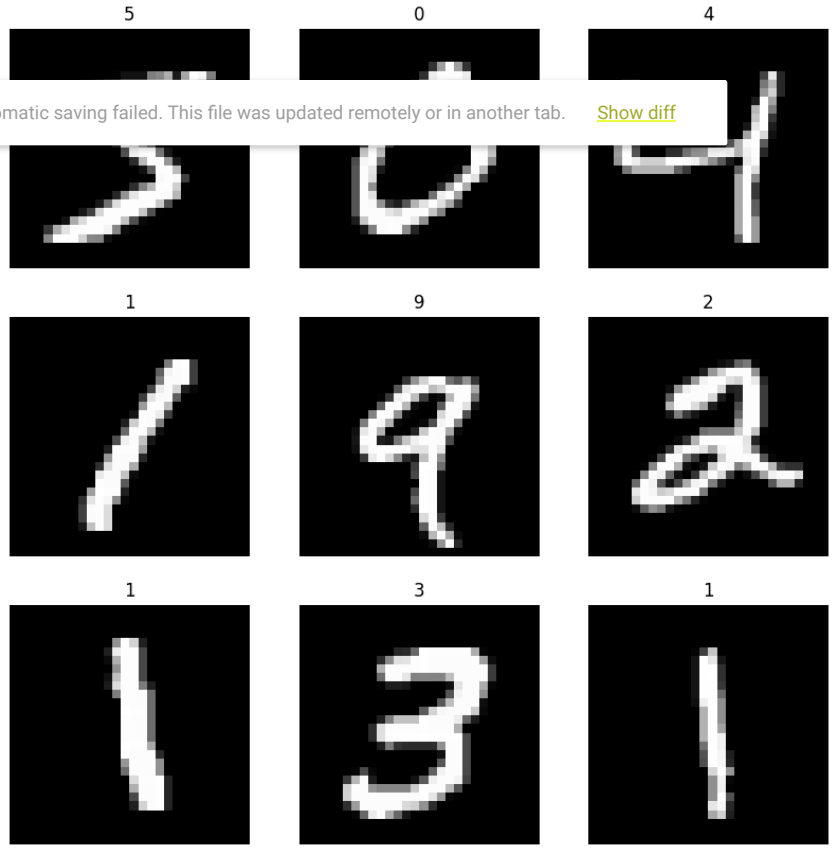
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

```
(X_train, y_train), (X_test, y_test) = data
```

```
X_train.shape
```

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

```
plt.figure(figsize=(10, 10))
for i in range(9):
 # Display the image
 ax = plt.subplot(3, 3, i + 1)
 plt.imshow(X_train[i], cmap='gray')
 # Print the label
 plt.title(y_train[i])
 plt.axis('off')
# Show the subplot
plt.show()
```



Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
X_train = X_train.reshape((X_train.shape[0], 28*28)).astype('float32')
X_test = X_test.reshape((X_test.shape[0], 28*28)).astype('float32')
```

```
X_train = X_train / 255
X_test = X_test / 255
```

```
from keras import utils
print(y_test.shape)
y_train = utils.to_categorical(y_train)
y_test = utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

```
(10000,)
```

```
print(y_test.shape)
```

```
(10000, 10)
```

```
from keras.models import Sequential
from keras.layers import Dense
```

```
model = Sequential()
model.add(Dense(32, input_dim = 28 * 28, activation= 'relu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(10, activation = 'softmax'))
```

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

```
model.summary()
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                25120

 dense_1 (Dense)             (None, 64)                2112

 dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 27,882
Trainable params: 27,882
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(X_train, y_train, epochs= 10, batch_size = 100)
```

```
Epoch 1/10
600/600 [==============================] - 5s 5ms/step - loss: 0.4255 - accuracy: 0.8797
Epoch 2/10
600/600 [==============================] - 1s 2ms/step - loss: 0.1952 - accuracy: 0.9425
Epoch 3/10
600/600 [==============================] - 1s 2ms/step - loss: 0.1544 - accuracy: 0.9534
Epoch 4/10
600/600 [==============================] - 1s 2ms/step - loss: 0.1309 - accuracy: 0.9599
Epoch 5/10
600/600 [==============================] - 1s 2ms/step - loss: 0.1134 - accuracy: 0.9657
Epoch 6/10
600/600 [==============================] - 1s 2ms/step - loss: 0.0999 - accuracy: 0.9701
Epoch 7/10
600/600 [==============================] - 1s 2ms/step - loss: 0.0907 - accuracy: 0.9719
Epoch 8/10
600/600 [==============================] - 1s 2ms/step - loss: 0.0826 - accuracy: 0.9750
Epoch 9/10
600/600 [==============================] - 2s 3ms/step - loss: 0.0754 - accuracy: 0.9770
Epoch 10/10
600/600 [==============================] - 2s 3ms/step - loss: 0.0684 - accuracy: 0.9783
<keras.callbacks.History at 0x7f7509cd54b0>
```

```
scores = model.evaluate(X_test, y_test)
print('Accuracy: ',scores[1] * 1000)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 2.3268 - accuracy: 0.0960
Accuracy:  96.00000083446503
```

✓ 0s completed at 10:20 PM

Automatic saving failed. This file was updated remotely or in another tab. Show diff