In [1]:

```
import numpy as np
import pandas as pd
import random
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Conv2D, Dense, MaxPooling2D
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import mnist
from keras.callbacks import TensorBoard, ReduceLROnPlateau
import sklearn
from sklearn.metrics import confusion_matrix
import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
# %load_ext tensorboard
%reload_ext tensorboard
```

In [2]:

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

In [3]:

```
X_train = (X_train - 0.0) / (255.0 - 0.0)
X_test = (X_test - 0.0) / (255.0 - 0.0)
```

In [4]:

```
np.unique(y_test, return_counts = True)
```

Out[4]:

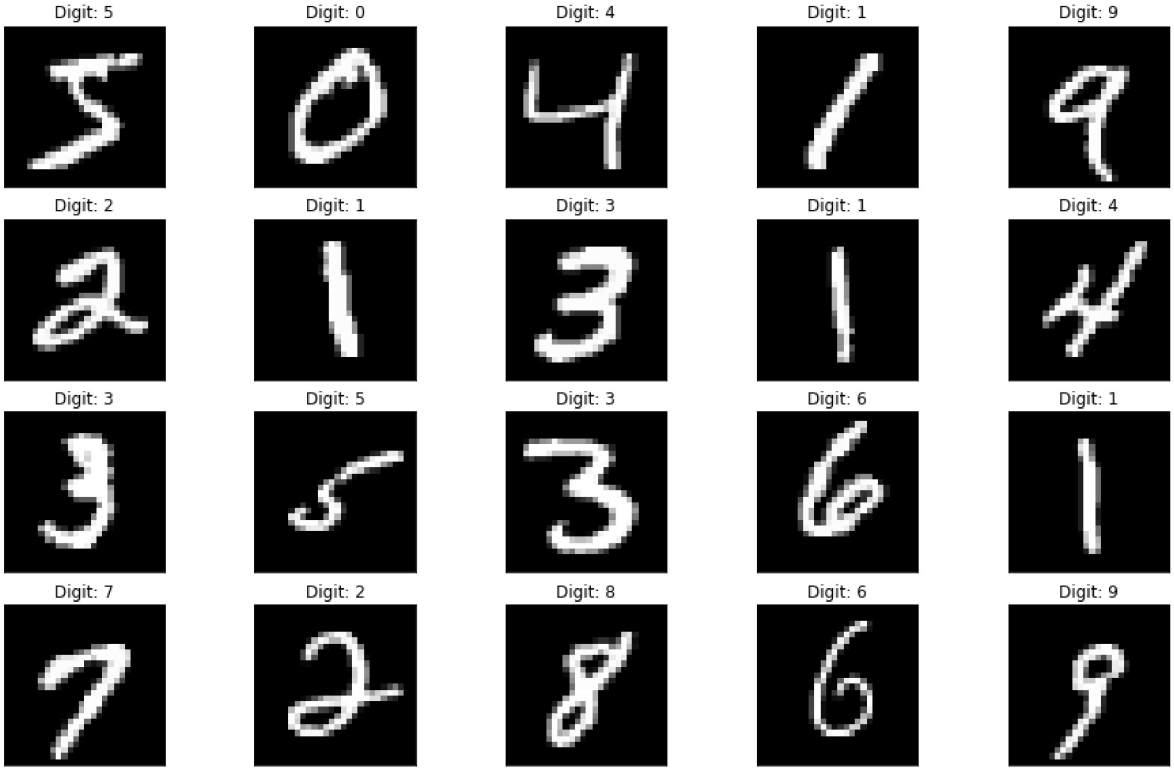
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=uint8),
 array([ 980, 1135, 1032, 1010,  982,  892,  958, 1028,  974, 1009],
       dtype=int64))
```

In [5]:

```
def plot_digit(image, digit, plt, i):
    plt.subplot(4, 5, i + 1)
    plt.imshow(image, cmap=plt.get_cmap('gray'))
    plt.title(f"Digit: {digit}")
    plt.xticks([])
    plt.yticks([])


plt.figure(figsize=(16, 10))
for i in range(20):
    plot_digit(X_train[i], y_train[i], plt, i)

plt.show()
```



In [6]:

```
X_train = X_train.reshape((len(X_train), 28, 28, 1))
X_test = X_test.reshape((len(X_test), 28, 28, 1))
```

In [7]:

```
model = Sequential()

model.add(Conv2D(64, (3, 3), activation="relu", input_shape=(28, 28, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(100, activation="relu"))
model.add(Dense(100, activation="relu"))
model.add(Dense(10, activation="softmax"))
```

In [8]:

```
optimizer = SGD(learning_rate=0.01, momentum=0.9)
model.compile(
    optimizer = optimizer,
    loss = "sparse_categorical_crossentropy",
    metrics = ["accuracy"]
)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 flatten (Flatten)           (None, 10816)             0

 dense (Dense)               (None, 100)               1081700

 dense_1 (Dense)             (None, 100)               10100

 dense_2 (Dense)             (None, 10)                1010

=================================================================
Total params: 1,093,450
Trainable params: 1,093,450
Non-trainable params: 0
_________________________________________________________________
```

In [9]:

```
TBcallback = TensorBoard(log_dir = './logs', histogram_freq=1)

reduceLR = ReduceLROnPlateau(monitor = "val_accuracy",
                             factor=0.1,
                             patience = 10,
                             verbose = 0,
                             mode = 'max')
```

In [10]:

```
history = model.fit(X_train,
                    y_train,
                    epochs = 20,
                    batch_size = 32,
                    validation_data=(X_test, y_test),
                    callbacks=[TBcallback, reduceLR])
```

```
Epoch 1/20
1875/1875 [==============================] - 16s 5ms/step - loss: 0.2435 - a
ccuracy: 0.9255 - val_loss: 0.1025 - val_accuracy: 0.9685 - lr: 0.0100
Epoch 2/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0726 - ac
curacy: 0.9774 - val_loss: 0.0618 - val_accuracy: 0.9801 - lr: 0.0100
Epoch 3/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0453 - ac
curacy: 0.9861 - val_loss: 0.0477 - val_accuracy: 0.9843 - lr: 0.0100
Epoch 4/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0321 - ac
curacy: 0.9899 - val_loss: 0.0507 - val_accuracy: 0.9841 - lr: 0.0100
Epoch 5/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0214 - ac
curacy: 0.9937 - val_loss: 0.0468 - val_accuracy: 0.9853 - lr: 0.0100
Epoch 6/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0162 - ac
curacy: 0.9948 - val_loss: 0.0397 - val_accuracy: 0.9880 - lr: 0.0100
Epoch 7/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0109 - ac
curacy: 0.9968 - val_loss: 0.0401 - val_accuracy: 0.9871 - lr: 0.0100
Epoch 8/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0089 - ac
curacy: 0.9974 - val_loss: 0.0382 - val_accuracy: 0.9883 - lr: 0.0100
Epoch 9/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0058 - ac
curacy: 0.9980 - val_loss: 0.0568 - val_accuracy: 0.9847 - lr: 0.0100
Epoch 10/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0039 - ac
curacy: 0.9991 - val_loss: 0.0506 - val_accuracy: 0.9878 - lr: 0.0100
Epoch 11/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0030 - ac
curacy: 0.9991 - val_loss: 0.0442 - val_accuracy: 0.9886 - lr: 0.0100
Epoch 12/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0025 - ac
curacy: 0.9994 - val_loss: 0.0443 - val_accuracy: 0.9889 - lr: 0.0100
Epoch 13/20
1875/1875 [==============================] - 8s 4ms/step - loss: 8.8137e-04
- accuracy: 0.9998 - val_loss: 0.0435 - val_accuracy: 0.9893 - lr: 0.0100
Epoch 14/20
1875/1875 [==============================] - 8s 4ms/step - loss: 3.2841e-04
- accuracy: 1.0000 - val_loss: 0.0467 - val_accuracy: 0.9891 - lr: 0.0100
Epoch 15/20
1875/1875 [==============================] - 8s 4ms/step - loss: 3.4957e-04
- accuracy: 1.0000 - val_loss: 0.0443 - val_accuracy: 0.9898 - lr: 0.0100
Epoch 16/20
1875/1875 [==============================] - 8s 4ms/step - loss: 2.3948e-04
- accuracy: 1.0000 - val_loss: 0.0451 - val_accuracy: 0.9894 - lr: 0.0100
Epoch 17/20
1875/1875 [==============================] - 8s 4ms/step - loss: 1.3075e-04
- accuracy: 1.0000 - val_loss: 0.0454 - val_accuracy: 0.9892 - lr: 0.0100
```

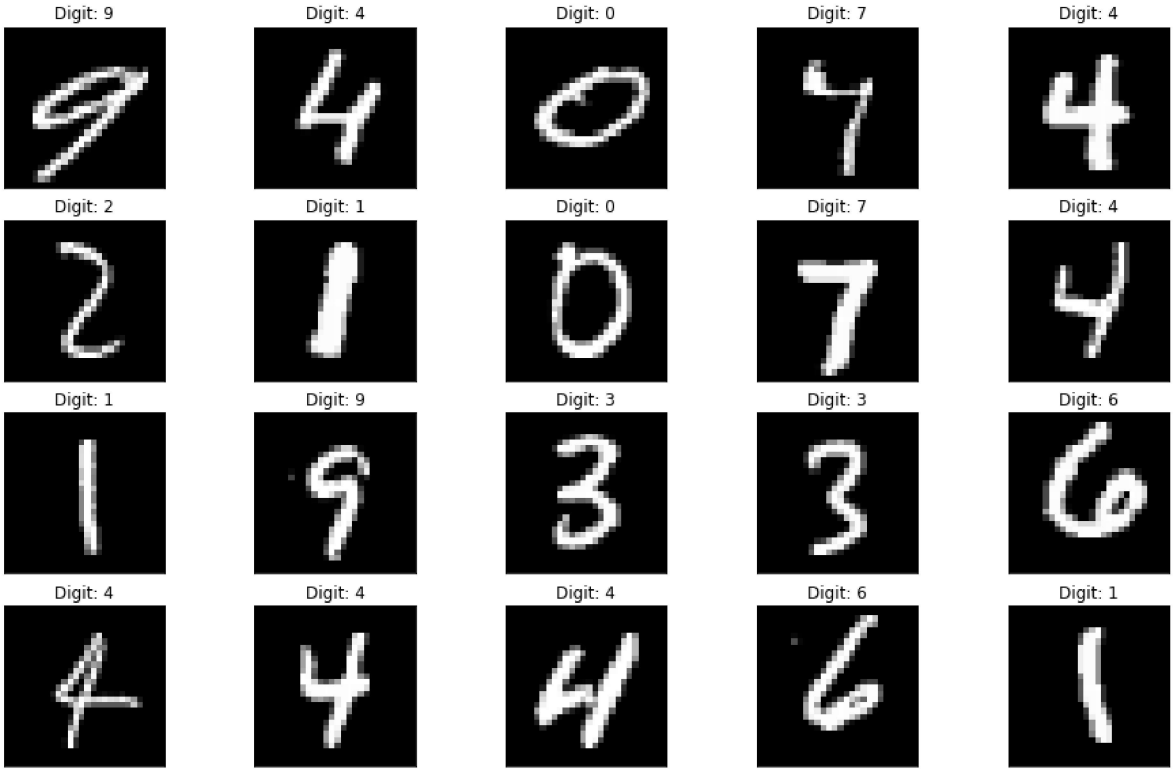
```
Epoch 18/20
1875/1875 [==============================] - 8s 4ms/step - loss: 1.1507e-04
- accuracy: 1.0000 - val_loss: 0.0467 - val_accuracy: 0.9898 - lr: 0.0100
Epoch 19/20
1875/1875 [==============================] - 8s 4ms/step - loss: 9.8021e-05
- accuracy: 1.0000 - val_loss: 0.0469 - val_accuracy: 0.9897 - lr: 0.0100
Epoch 20/20
1875/1875 [==============================] - 8s 4ms/step - loss: 8.4725e-05
- accuracy: 1.0000 - val_loss: 0.0474 - val_accuracy: 0.9894 - lr: 0.0100
```

In [11]:

```
plt.figure(figsize=(16, 10))

for i in range(20):
    image = random.choice(X_test).squeeze()
    digit = np.argmax(model.predict(image.reshape((1, 28, 28, 1)), verbose = 0)[0], axis=-1
    plot_digit(image, digit, plt, i)

plt.show()
```



In [12]:

```
predictions = np.argmax(model.predict(X_test, verbose = 0), axis=-1)
accuracy_score(y_test, predictions)
```

Out[12]:

```
0.9894
```

In [13]:

```
matrixConfusion = confusion_matrix(y_test, predictions)
```

In [14]:

```
matrixConfusion = pd.DataFrame(matrixConfusion,
                               columns = ['0','1','2','3','4','5','6','7','8','9'],
                               index = ['previsto 0', 'previsto 1', 'previsto 2', 'previsto
                                        'previsto 5', 'previsto 6', 'previsto 7', 'previsto
```

In [15]:

```
matrixConfusion
```

Out[15]:

	0	1	2	3	4	5	6	7	8	9
previsto 0	974	0	0	0	0	2	2	1	0	1
previsto 1	0	1130	1	0	0	0	2	1	1	0
previsto 2	1	3	1019	1	1	0	1	4	2	0
previsto 3	0	0	1	1003	0	3	0	1	2	0
previsto 4	0	1	1	0	971	0	1	0	1	7
previsto 5	2	0	0	6	0	882	1	0	1	0
previsto 6	3	3	0	0	2	1	948	0	1	0
previsto 7	1	1	5	0	0	0	0	1018	0	3
previsto 8	5	0	2	0	0	0	1	1	961	4
previsto 9	1	2	0	2	9	1	0	4	2	988