```
#Instalamos la libreria tflearn
!pip install tflearn
```

```
#Importamos todo lo que necesitamos de tflearn
import tflearn
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.normalization import local_response_normalization
from tflearn.layers.estimator import regression
```

```
#Importamos el dataset de MNIST
import tflearn.datasets.mnist as mnist
#Cargamos el dataset en las variables de entrenamiento y prueba
X, Y, testX, testY = mnist.load_data(one_hot=True)
X = X.reshape([-1, 28, 28, 1])
testX = testX.reshape([-1, 28, 28, 1])
```

```
#Definimos la estructura de la red neuronal
net = input_data(shape=[None, 28, 28, 1])
net = conv_2d(net, 32, 3, activation='relu', regularizer="L2")
net = max_pool_2d(net, 2)
net = local_response_normalization(net)
net = conv_2d(net, 64, 3, activation='relu', regularizer="L2")
net = max_pool_2d(net, 2)
net = local_response_normalization(net)
net = conv_2d(net, 128, 3, activation='relu', regularizer="L2")
net = max_pool_2d(net, 2)
net = fully_connected(net, 625, activation='relu')
net = dropout(net, 0.8)
net = fully_connected(net, 10, activation='softmax')
net = regression(net, optimizer='adam', learning_rate=0.001, loss='categorical_crossentropy')

model = tflearn.DNN(net)
```

```
#Entrenamos el modelo y lo validamos
model.fit(X, Y, validation_set=(testX, testY),snapshot_step=1000, show_metric=True, n_epoch=1
```