In [1]:

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils


(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape((X_train.shape[0], 28, 28, 1)).astype('float32')
X_test = X_test.reshape((X_test.shape[0], 28, 28, 1)).astype('float32')


X_train = X_train / 255
X_test = X_test / 255
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]


def baseline_model():
    model = Sequential()
    model.add(Conv2D(32, (5, 5), input_shape=(28, 28, 1), activation='relu'))
    model.add(MaxPooling2D())
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accurac
    return model


model = baseline_model()
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=20
scores = model.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
```

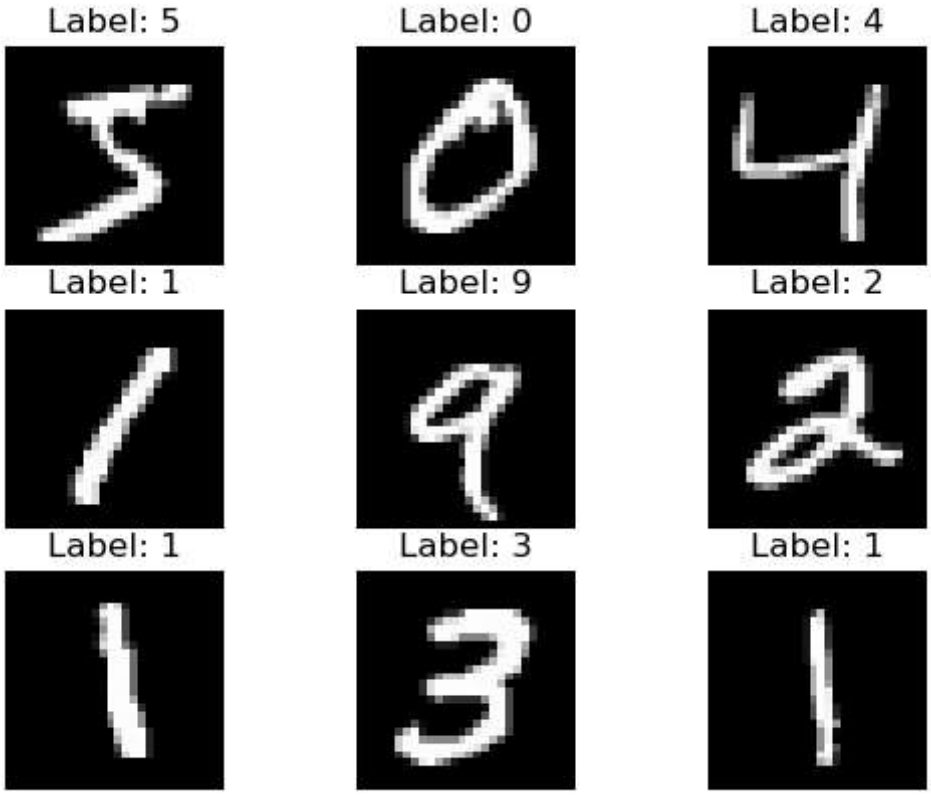
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mni
st.npz
11490434/11490434 [==============================] - 13s 1us/step
Epoch 1/10
300/300 [==============================] - 24s 75ms/step - loss: 0.2322 - accuracy:
0.9348 - val_loss: 0.0755 - val_accuracy: 0.9772
Epoch 2/10
300/300 [==============================] - 22s 73ms/step - loss: 0.0721 - accuracy:
0.9783 - val_loss: 0.0467 - val_accuracy: 0.9843
Epoch 3/10
300/300 [==============================] - 22s 75ms/step - loss: 0.0522 - accuracy:
0.9839 - val_loss: 0.0408 - val_accuracy: 0.9868
Epoch 4/10
300/300 [==============================] - 22s 75ms/step - loss: 0.0417 - accuracy:
0.9874 - val_loss: 0.0428 - val_accuracy: 0.9851
Epoch 5/10
300/300 [==============================] - 23s 75ms/step - loss: 0.0343 - accuracy:
0.9892 - val_loss: 0.0324 - val_accuracy: 0.9887
Epoch 6/10
300/300 [==============================] - 23s 75ms/step - loss: 0.0270 - accuracy:
0.9916 - val_loss: 0.0327 - val_accuracy: 0.9888
Epoch 7/10
300/300 [==============================] - 22s 74ms/step - loss: 0.0239 - accuracy:
0.9923 - val_loss: 0.0331 - val_accuracy: 0.9892
Epoch 8/10
300/300 [==============================] - 23s 76ms/step - loss: 0.0211 - accuracy:
0.9929 - val_loss: 0.0313 - val_accuracy: 0.9898
Epoch 9/10
300/300 [==============================] - 22s 75ms/step - loss: 0.0170 - accuracy:
0.9946 - val_loss: 0.0332 - val_accuracy: 0.9884
Epoch 10/10
300/300 [==============================] - 23s 75ms/step - loss: 0.0149 - accuracy:
0.9953 - val_loss: 0.0308 - val_accuracy: 0.9906
CNN Error: 0.94%
```

In [ ]:
```
#Use Cases of MNIST Dataset
#MNIST dataset is used widely for handwrittern digit classifier.
#It is the supporting base for handwritting, signature recognisation.
#MNIST dataset is also used for image classifiers dataset analysis.
#MNIST Dataset is an intergal part of Date predictions from pieces of texts in coorpor
#MNIST dataset is also used for predicting the students percentages from their resumes
#check their qualifying level.
```

In [2]:
```
import matplotlib.pyplot as plt

# Display the first 9 images from the training dataset
for i in range(9):
    plt.subplot(3, 3, i + 1)
    plt.imshow(X_train[i].reshape(28, 28), cmap='gray')
    plt.title(f"Label: {y_train[i].argmax()}")  # Display the label
    plt.axis('off')

plt.show()
```


Label: 5
Label: 0
Label: 4
Label: 1
Label: 9
Label: 2
Label: 1
Label: 3
Label: 1