```
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=['accuracy'])
        return model
model = baseline_model()
model.fit(x_train, y_train, validation_data=(x_test, y_test),
epochs=10, batch_size=100)
scores = model.evaluate(x_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
print("CNN EFFICIENCY: %.2f%%" % (scores[1]*100))
```