```
# baseline cnn model for mnist
from numpy import mean
from numpy import std
from matplotlib import pyplot as plt
from sklearn.model_selection import KFold
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.optimizers import SGD
```

In [2]:

```
def load_dataset():
        # load dataset
        (trainX, trainY), (testX, testY) = mnist.load_data()
        # reshape dataset to have a single channel
        trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
        testX = testX.reshape((testX.shape[0], 28, 28, 1))
        # one hot encode target values
        trainY = to_categorical(trainY)
        testY = to_categorical(testY)
        return trainX, trainY, testX, testY
```

In [3]:

```
def prep_pixels(train, test):
        # convert from integers to floats
        train_norm = train.astype('float32')
        test_norm = test.astype('float32')
        # normalize to range 0-1
        train_norm = train_norm / 255.0
        test_norm = test_norm / 255.0
        # return normalized images
        return train_norm, test_norm
```

In [4]:

```
def define_model():
        model = Sequential()
        model.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', input_shape=(28, 28, 1)))
        model.add(MaxPooling2D((2, 2)))
        model.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform'))
        model.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform'))
        model.add(MaxPooling2D((2, 2)))
        model.add(Flatten())
        model.add(Dense(100, activation='relu',
kernel_initializer='he_uniform'))
        model.add(Dense(10, activation='softmax'))
        # compile model
        opt = SGD(learning_rate=0.01, momentum=0.9)
        model.compile(optimizer=opt, loss='categorical_crossentropy',
metrics=['accuracy'])
        return model
```

In [5]:

```
def run_test_harness():
  # load dataset
  trainX, trainY, testX, testY = load_dataset()
  # prepare pixel data
  trainX, testX = prep_pixels(trainX, testX)
  # define model
  model = define_model()
  # fit model
  model.fit(trainX, trainY, epochs=10, batch_size=32, verbose=0)
  _, acc = model.evaluate(testX, testY, verbose=0)
  print('> %.3f' % (acc * 100.0))
  # save model
  model.save('final_model.h5')
```

In [6]:

```
run_test_harness()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-da
tasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step
> 98.820
```