```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True, reshape=False)

def fully_connected(prev_layer, num_units, batch_norm, is_training=False):
    layer = tf.layers.dense(prev_layer, num_units, use_bias=False,
activation=None)
    if batch_norm:
      layer = tf.layers.batch_normalization(layer, training=is_training)
    layer = tf.nn.relu(layer)
    return layer

def conv_layer(prev_layer, layer_depth, batch_norm, is_training=False):
    strides = 2 if layer_depth % 3 == 0 else 1
    conv_layer = tf.layers.conv2d(prev_layer, layer_depth*4, 3, strides, 'same',
use_bias=False, activation=None)
    if batch_norm:
      conv_layer = tf.layers.batch_normalization(conv_layer,
training=is_training)
    conv_layer = tf.nn.relu(conv_layer)
    return conv_layer


num_batches = 3000
batch_size = 110
learning_rate = 0.002

inputs = tf.placeholder(tf.float32, [None, 28, 28, 1])
labels = tf.placeholder(tf.float32, [None, 10])
is_training = tf.placeholder(tf.bool)

batch_norm = False

layer = inputs
for layer_i in range(1, 8):
    layer = conv_layer(layer, layer_i, batch_norm, is_training)


orig_shape = layer.get_shape().as_list()
layer = tf.reshape(layer, shape=[-1, orig_shape[1] * orig_shape[2] *
orig_shape[3]])

layer = fully_connected(layer, 100, batch_norm, is_training)

logits = tf.layers.dense(layer, 10)

model_loss =
tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=logits,
labels=labels))
tf.summary.scalar('conv_loss',model_loss)

if batch_norm:
    with tf.control_dependencies(tf.get_collection(tf.GraphKeys.UPDATE_OPS)):
```

```
        train_opt =
tf.train.GradientDescentOptimizer(learning_rate).minimize(model_loss)
else:
    train_opt =
tf.train.GradientDescentOptimizer(learning_rate).minimize(model_loss)

correct_prediction = tf.equal(tf.argmax(logits,1), tf.argmax(labels,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))


#with tf.Session() as sess:
sess = tf.Session()

merged = tf.summary.merge_all()
if batch_norm:
      logdir = "mnist/conv/SGD_batchnorm"
else:
      logdir = "mnist/conv/SGD_no_batchnorm"
writer = tf.summary.FileWriter(logdir, sess.graph)

sess.run(tf.global_variables_initializer())
for batch_i in range(num_batches):
    batch_xs, batch_ys = mnist.train.next_batch(batch_size)

    _,summary = sess.run([train_opt,merged], {inputs: batch_xs, labels:
batch_ys, is_training: True})

    writer.add_summary(summary, batch_i)

    if batch_i % 200 == 0:
        loss, acc = sess.run([model_loss, accuracy], {inputs:
mnist.validation.images,
            labels: mnist.validation.labels,
            is_training: False})
        print('Batch: {:>2}: Validation loss: {:>3.5f}, Validation accuracy:
{:>3.5f}'.format(batch_i, loss, acc))
    elif batch_i % 50 == 0:
        loss, acc = sess.run([model_loss, accuracy], {inputs: batch_xs, labels:
batch_ys, is_training: False})
        print('Batch: {:>2}: Training loss: {:>3.5f}, Training accuracy:
{:>3.5f}'.format(batch_i, loss, acc))

        # At the end, score the final accuracy for both the validation and test
sets
acc = sess.run(accuracy, {inputs: mnist.validation.images,
      labels: mnist.validation.labels,is_training: False})
print('Final validation accuracy: {:>3.5f}'.format(acc))
acc = sess.run(accuracy, {inputs: mnist.test.images,
      labels: mnist.test.labels,is_training: False})
print('Final test accuracy: {:>3.5f}'.format(acc))
```