TF搭建卷积神经网络

目 录

- 一. Tensorflow卷积的简介
- 二. CNN的搭建

一. Tensorflow卷积的简介

使用TensorFlow来实现卷积

TensorFlow提供了一个tf.nn.conv2d函数，可以很容易的计算一个卷积层的前向传播过程。

在TF中一个完整的卷积层如下：

```
# Apply Convolution
conv_layer = tf.nn.conv2d(input, weight, strides=[1, 2, 2, 1], padding='SAME’)

# Add bias
conv_layer = tf.nn.bias_add(conv_layer, bias)

 # Apply activation function
conv_layer = tf.nn.relu(conv_layer)
```

一. Tensorflow卷积的简介

tf.nn.conv2d函数

tf.nn.conv2d(input, filter, strides, padding)

input：输入的节点矩阵，注意这个举证是一个四维矩阵，第一维代表的是输入的batch，如训练手写数字的时候，需要设置每次训练图片的数量，后面三个维度代表输入数据的节点矩阵，如第一张图片input[0,:,:,:]。

filter：表示的是卷积层的权重，权重的类型必须与数据的类型一致。

strides：一个长度为4的数据，可以是列表或者张量，第一维和最后一维的数字要求一定是1，因为卷积层的步长只对矩阵的长和宽有效。

padding：是否填充，选项为“SAME”或“VALID”。"SAME"表示添加全0填充，保证卷积前输入的矩阵和卷积后的输出矩阵大小一致，"VALID"表示不添加填充。

一. Tensorflow卷积的简介

池化函数

tf.nn.max_pool() 函数实现最大池化时， ksize参数是滤波器大小，strides参数是步长。2x2 的滤波器配合 2x2 的步长是常用设定。

ksize 和 strides 参数也被构建为四个元素的列表，每个元素对应 input tensor 的一个维度 ([batch, height, width, channels])，对 ksize 和 strides 来说，batch 和 channel 通常都设置成 1。

```
# Apply Max Pooling
conv_layer = tf.nn.max_pool( conv_layer,
                             ksize=[1, 2, 2, 1],
                             strides=[1, 2, 2, 1],
                             padding='VALID' )
```

二. CNN的搭建

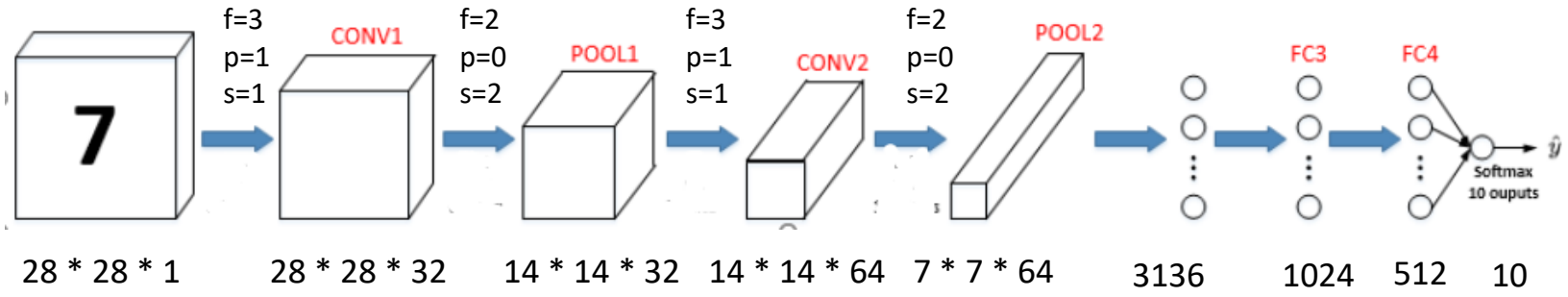
CNN的设计

同样对于mnist数字识别数据集的分类任务：
使用一个简单的CNN网络结构如下，括号里边表示tensor经过该层后的输出shape：

input （28 * 28 * 1）
conv1　　f=3 p=1 s=1 （28 * 28 * 32）
pooling1　f=2 p=0 s=2 （14 * 14 * 32）
conv2　　f=3 p=1 s=1 （14 * 14 * 64）
pooling2　f=2 p=0 s=2 （7 * 7 * 64）
FC3（1 * 1024）
FC4（1 * 512）
softmax（ 1 * 10）

二. CNN的搭建

CNN的设计结构图



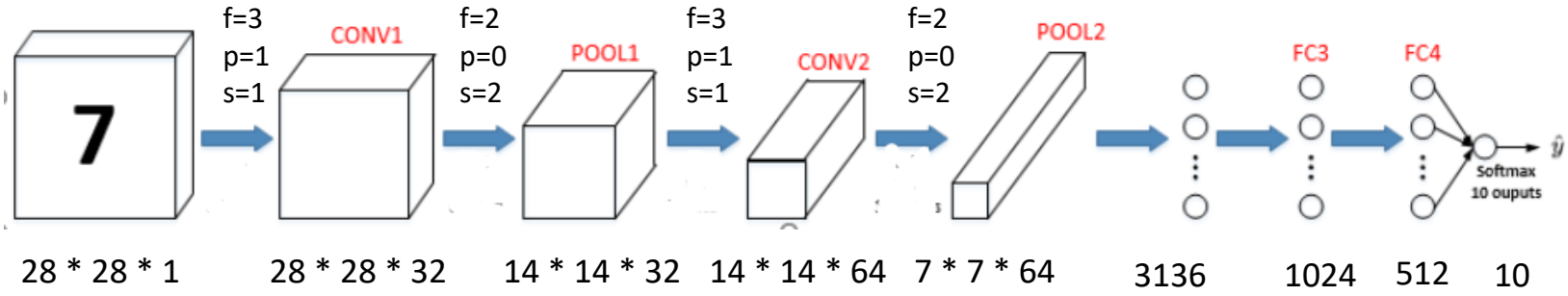
input （28 * 28 * 1）
conv1　　f=3 p=1 s=1 （28 * 28 * 32）
pooling1　f=2 p=0 s=2 （14 * 14 * 32）
conv2　　f=3 p=1 s=1 （14 * 14 * 64）
pooling2　f=2 p=0 s=2 （7 * 7 * 64）
FC3（1 * 1024）
FC4（1 * 512）
softmax（ 1 * 10）

二. CNN的搭建

CNN与神经网络的设计不同点

与之前的数字识别网络的搭建例子程序不同的是：
（1）网络结构的不同；
（2）数据输入时的维度不一样。

其他的是相同的，例如：
（1）损失、优化函数；
（2）训练的流程；
（3）结果的展示。



二. CNN的搭建

定义数据格式

定义数据格式相同：

```
x = tf.placeholder(tf.float32, [None, size_input*size_input])  #数据
y = tf.placeholder(tf.float32, [None, size_output]) #标签
```

接下来转化为图像格式：

```
x_image = tf.reshape(x, [-1, size_input, size_input, c_in]) # 转换为图像的格式
```

二. CNN的搭建

定义卷积层

定义卷积层如下：

```
def conv2d(x, c_in, c_out, k_size, stride, padding='VALID', activation_function=tf.nn.relu):
    Weight = tf.Variable(tf.random_normal([k_size, k_size, c_in, c_out], stddev=0.02))
    biases = tf.Variable(tf.zeros([c_out]) + 0.1)
    conv_1 = tf.nn.conv2d(x, Weight, strides=[1, stride, stride, 1], padding=padding)
    conv_1_b = tf.nn.bias_add(conv_1, biases)
    if activation_function == None:
        output = conv_1_b
    else:
        output = activation_function(conv_1_b)
    return output
```

里面包括：卷积操作、偏置相加、激活函数

二. CNN的搭建

定义网络结构

定义网络结构如下：

```
def network_mnist(input, c_in, c_out):
    conv_1 = conv2d(input, c_in, 32, 3, 1, padding='SAME', activation_function=tf.nn.relu)
    pool_1 = tf.nn.max_pool(conv_1, [1, 2, 2, 1], [1, 2, 2, 1], padding='VALID')
    conv_2 = conv2d(pool_1, 32, 64, 3, 1, padding='SAME', activation_function=tf.nn.relu)
    pool_2 = tf.nn.max_pool(conv_2, [1, 2, 2, 1], [1, 2, 2, 1], padding='VALID')

    pool_2_flat = tf.reshape(pool_2, [-1, 7 * 7 * 64])
    fc_3 = add_layer(pool_2_flat, 7*7*64, 1024, activation_function=tf.nn.relu)
    fc_4 = add_layer(fc_3, 1024, 512, activation_function=tf.nn.relu)
    prediction = add_layer(fc_4, 512, c_out, activation_function=tf.nn.softmax)
    return prediction
```

这里c_in=1，c_out=10

二. CNN的搭建

损失、会话、训练等

损失函数、优化方法、创建会话、训练等操作，与之前的神经网络一致。
因此在这里略过。

```
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init)
acc_max = -1
for iepoch in range(epoch_max):
    is_save = 0
    acc_train_all = []
    for step in range(step_max):
        batch_x, batch_y = mnist.train.next_batch(batch_size)
        _, acc = sess.run([train, accuracy], feed_dict={x:batch_x, y:batch_y})
        acc_train_all.append(acc)
    acc_train = sum(acc_train_all)/len(acc_train_all)
    acc_val = sess.run(accuracy, feed_dict={x: mnist.validation.images, y: mnist.validation.labels})
    if acc_max < acc_val:
        saver.save(sess, model_path)
        is_save = 1
        acc_max = acc_val
    if acc_max >= acc_stop:
        print('End training early...')
        break
    print('epoch:%02d is_save:%d acc_max:%.4f%% acc_val:%.4f%% acc_train:%.4f%%'%(iepoch, is_save,
                                                    acc_max*100, acc_val*100, acc_train*100))

acc_test = sess.run(accuracy, feed_dict={x: mnist.test.images, y: mnist.test.labels})
print('acc_test: %.4f%%'%(acc_test*100))
print('**********DONE**********')
```

二. CNN的搭建

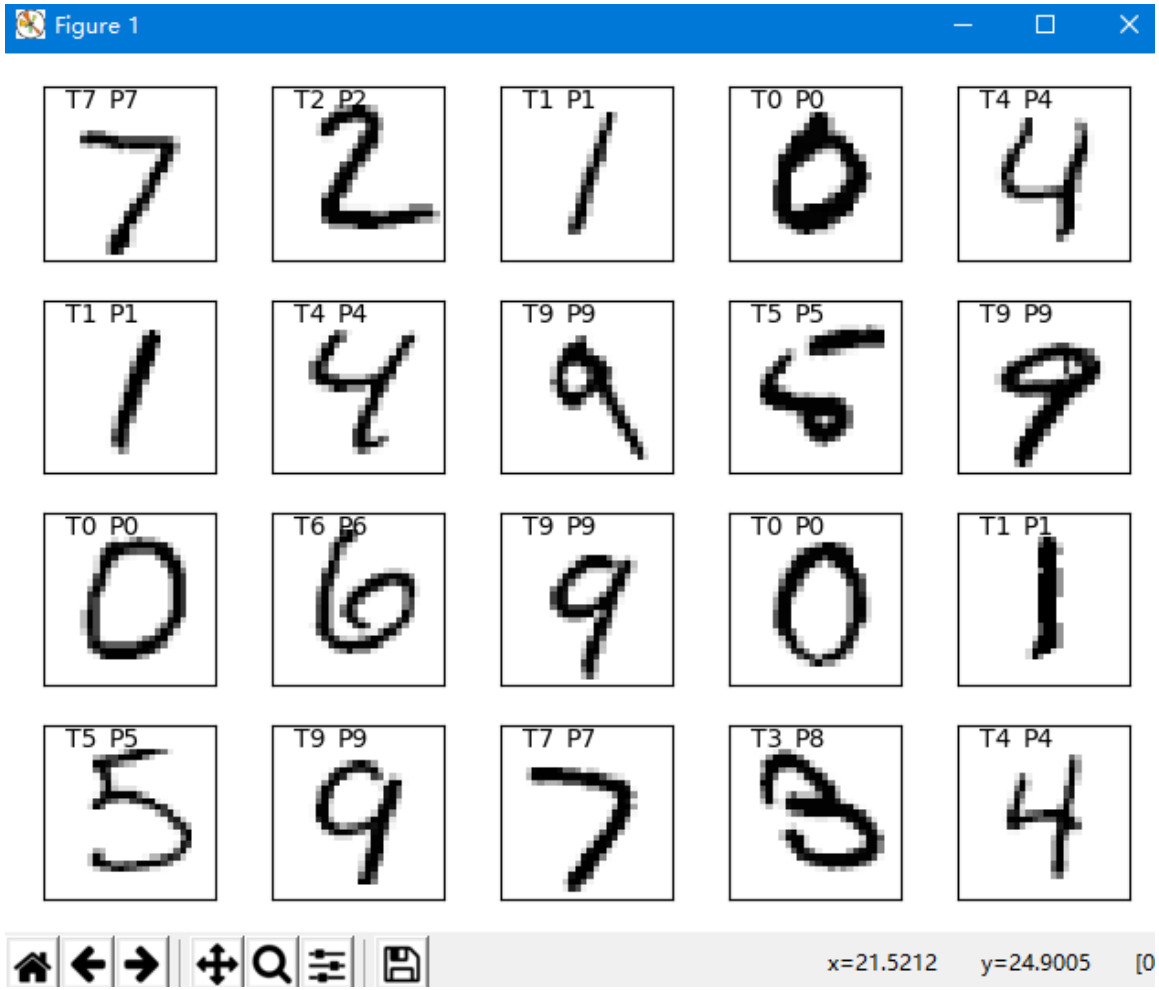
训练过程的输出：

从这里可以发现，利用CNN处理图像的任务时，能够更快得到的更高准确度的结构。

```
epoch:01 is_save:1 acc_max:91.1200% acc_val:91.1200% acc_train:72.1853%
epoch:02 is_save:1 acc_max:92.9000% acc_val:92.9000% acc_train:91.6709%
epoch:03 is_save:1 acc_max:96.6200% acc_val:96.6200% acc_train:95.0204%
epoch:04 is_save:1 acc_max:97.2600% acc_val:97.2600% acc_train:96.0335%
epoch:05 is_save:1 acc_max:97.6800% acc_val:97.6800% acc_train:97.1944%
epoch:06 is_save:1 acc_max:97.9000% acc_val:97.9000% acc_train:97.0083%
epoch:07 is_save:1 acc_max:98.2200% acc_val:98.2200% acc_train:97.9045%
epoch:08 is_save:0 acc_max:98.2200% acc_val:95.9000% acc_train:96.7089%
epoch:09 is_save:1 acc_max:98.3000% acc_val:98.3000% acc_train:97.6453%
epoch:10 is_save:0 acc_max:98.3000% acc_val:98.0800% acc_train:98.2531%
epoch:11 is_save:1 acc_max:98.5200% acc_val:98.5200% acc_train:98.5616%
epoch:12 is_save:0 acc_max:98.5200% acc_val:98.4200% acc_train:98.7058%
epoch:13 is_save:1 acc_max:98.6000% acc_val:98.6000% acc_train:98.8610%
epoch:14 is_save:0 acc_max:98.6000% acc_val:98.5000% acc_train:98.9431%
epoch:15 is_save:0 acc_max:98.6000% acc_val:98.5200% acc_train:98.5890%
epoch:16 is_save:1 acc_max:98.6600% acc_val:98.6600% acc_train:99.0344%
epoch:17 is_save:1 acc_max:98.8200% acc_val:98.8200% acc_train:99.1768%
epoch:18 is_save:0 acc_max:98.8200% acc_val:98.8000% acc_train:99.2699%
epoch:19 is_save:0 acc_max:98.8200% acc_val:98.5400% acc_train:99.3082%
epoch:20 is_save:1 acc_max:98.8600% acc_val:98.8600% acc_train:99.4104%
epoch:21 is_save:1 acc_max:98.9600% acc_val:98.9600% acc_train:99.4414%
epoch:22 is_save:0 acc_max:98.9600% acc_val:98.9600% acc_train:99.5510%
epoch:23 is_save:0 acc_max:98.9600% acc_val:98.8400% acc_train:99.4999%
epoch:24 is_save:1 acc_max:99.0400% acc_val:99.0400% acc_train:99.6313%
End training early...
acc_test: 98.8800%
**********DONE**********
```

二. CNN的搭建

测试集推理结果的可视化：
T-真实的数字
P-预测的数字



测试集推理的准确率：

```
acc_test: 98.8800%
```

神经网络测试集推理的准确率：

```
acc_test: 98.0900%
```

二. CNN的搭建

完成的程序在‘project\tf_build_CNN’目录下：
‘tf_mnist_show.py’：展示mnist与可视化数据；
‘tf_mnist_train.py’：训练并得到模型；
‘tf_mnist_test.py’：在测试集上测试，获得准确率与可视化结果。

总结

一. 简述了TF的卷积操作函数、池化函数等；
二. 介绍数字识别CNN网络的搭建流程，讲述CNN网络的基本训练流程；
三、从实验结果可以看出，CNN更适合于处于图像的任务。

谢谢聆听

参考网页：

tf.nn.conv2d：https://blog.csdn.net/sinat_29957455/article/details/78535621
tf.nn.max_pool() 函数：https://www.jianshu.com/p/9654dc57a059