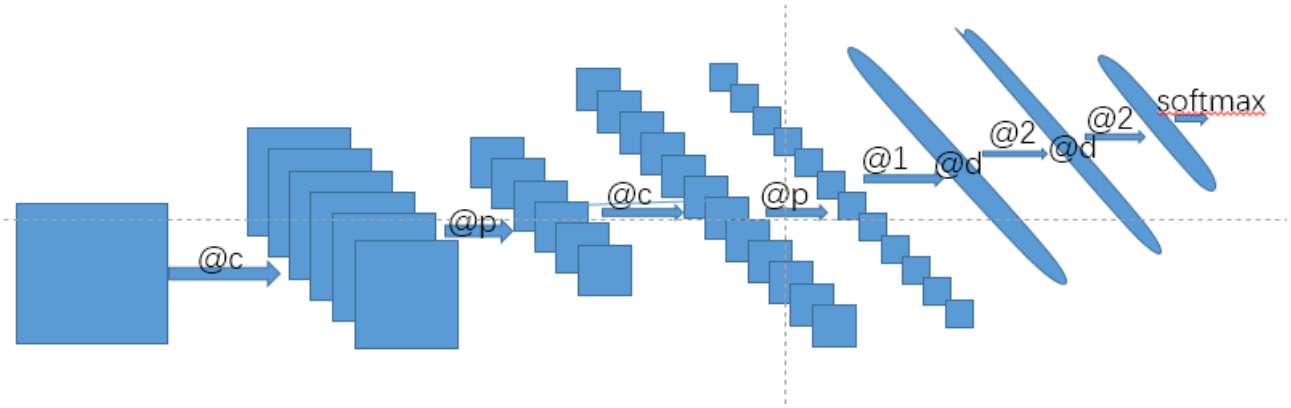
CNN 说明文档

一、网络结构



1. @c 表示卷积操作，@p 表示 pooling 操作，@1 表示将所有的 feature map 展开成一维的向量，@2 就是和 BP 一样的矩阵乘操作，@p 表示这一层的数据要经过 dropout 操作，然后得到计算下一层数据的输入。除了输入层和最后一层中间每层都是用 relu 作为激活函数，最后一层的结果用 softmax 处理。
2. 总的来说这是一个两层卷积，两层 pooling，最后全连接层做了两次 dropout 防止过拟合的网络。

二、参数调整算法

1. 参数调整总的来说还是反向传播方法，但是该实现基于 tensorflow 库实现，内部调整算法并没有细致了解，所以不会像 BP 一样细致阐述。
2. **Loss function**

Loss function 同样是交叉熵，最后的输出实用 softmax 处理，这两步合在一起可以用 tensorflow 提供的方法实现，如下：

```
tf.nn.softmax_cross_entropy_with_logits(logits, labels, dim=-1, name=None)
```

Computes softmax cross entropy between `logits` and `labels`.

- `logits`: Unscaled log probabilities.
- `labels`: Each row `labels[i]` must be a valid probability distribution.
- `dim`: The class dimension. Defaulted to -1 which is the last dimension.
- `name`: A name for the operation (optional).

后面两个使用默认值的可以不用设置。

3. 参数调整

基于 tensorflow，这一点也非常容易实现。Tensorflow 实现了许多版本的优化器，本次实验选用的是实现了 Adam algorithm 的优化器，如下：

```
tf.train.AdamOptimizer.__init__(learning_rate=0.001, beta1=0.9,
beta2=0.999, epsilon=1e-08, use_locking=False, name='Adam')
```

Construct a new Adam optimizer.

同样，可以使用所有默认的参数。

然后所有的优化器都实现了减小 loss function 的接口，如下：

```
tf.train.Optimizer.minimize(loss, global_step=None, var_list=None,
gate_gradients=1, aggregation_method=None,
colocate_gradients_with_ops=False, name=None, grad_loss=None)
```

我们只要传入前面的 loss function，就可以使用此方法更新参数了。

三、利用 CNN 网络训练图片识别

1. 基本说明

由于 CNN 网络超参数是在太多了，所以不会都去调，也不会一个个参数去讲，下面大概是会调整的参数清单：

```
# convolution and pooling 1
patch_size_conv1 = 5;
feature_num_conv1 = 32
stride_conv = 1
padding_conv = 'VALID'
size_pool = 2
stride_pool = 2
padding_pool = 'VALID'

# convolution and pooling 2
feature_num_conv2 = 64
patch_size_conv2 = 5
# stride_conv2 = 1
# padding_conv2 = 'VALID'
# stride_pool2 = 2
# padding_pool2 = 'VALID'

# full connection 1
size_fc1 = 1024 # similar like hidden layer size

# dropout
dropout_p = 0.38
# dropout2_p = 0.4
```

2. 输入处理

Cnn 训练图片保留 28*28 的二维数组。根据经验，我觉得输入在 0-1 之间会有较好的结果。所以会将 0-255 的输入值除以 255.0 再作为网络的输入。再有就是需要将输入的图片打

乱顺序后再作为输入。还有就是采用 mini-batch 方法每次训练 50 个，因为这个训练还是十分的慢。

3. 参数调整

最开始的时候，只有前面结构中的后一个 dropout 层，p=0.5。到底是 tensorflow，跑一下结果就非常好了。

step 9600, training accuracy 1
step 9700, training accuracy 1
step 9800, training accuracy 1
step 9900, training accuracy 1
step 10000, training accuracy 1
test accuracy 0.946562

训练过程前面还有很多个 1，虽然每次只有 50 个，但 200 次就足以包含所有数据了，所以基本上训练集就全对了。而验证集却有待提高，为了防止过拟合，我觉得模型太复杂了，简化一点。首先 zero-padding 设成 0，tensorflow 是通过 valid 来标明的，然后卷积 feature 数目减下去，可是效果不好，下面是一个两层卷积 feature 数分别为 16,32 的例子：

step 9600, training accuracy 1
step 9700, training accuracy 1
step 9800, training accuracy 0.98
step 9900, training accuracy 1
step 10000, training accuracy 1
test accuracy 0.942813

也调了其它参数简化模型，效果不怎么样，后来想想思路有点问题，觉得应该从 dropout 下手，就加了一层 dropout，同时也在训练过程中输出正确率，以供参考。

p 都是 0.5,

step 29900, training accuracy 1
test accuracy 0.945625
step 30000, training accuracy 0.98
test accuracy 0.944063
test accuracy 0.944063

p 都是 0.3，不行了。

step 29800, training accuracy 0.9
test accuracy 0.936875
step 29900, training accuracy 0.94
test accuracy 0.936562
step 30000, training accuracy 0.92
test accuracy 0.933438
test accuracy 0.933438

p 都是 0.45

step 39800, training accuracy 1
test accuracy 0.949687
step 39900, training accuracy 1
test accuracy 0.95125
step 40000, training accuracy 1
test accuracy 0.950938

test accuracy 0.950938

p 都是 0.4

step 39600, training accuracy 1

test accuracy 0.951562

step 39700, training accuracy 1

test accuracy 0.952188

step 39800, training accuracy 1

test accuracy 0.953438

step 39900, training accuracy 1

test accuracy 0.952812

step 40000, training accuracy 1

test accuracy 0.954062

test accuracy 0.954062

使用 dropout 解决过拟合问题还是有效的，后面就不想调了。