CNNs

1. 处理数据
2. 读取数据
3. 查看数据是否需要处理，比如空格标点等
4. 划分数据特征和目标
5. 数据占位符 tf.placeholder(float32, shape, name=’’)
6. 把数据集分batch

1)batch的size太小，使得训练太慢，但如果size太大，准确率会降低

1. 初始化model
2. 定义weights和bias,需要转为tf.Variable()

Weights = tf.Variable(tf.truntated\_normal([shape], stddev=0.1))

Bias = tf.Variable(tf.zeros(shape[1]))

1. 定义卷积层并forward pass

卷积层中传入input, weight, strides, padding

一层卷积之后一层池化层，卷积池化完了之后需要将多维数据转化为一维的数据传入到全连接层，多分类问题最后经过softmax激活函数得出目标

1. 定义损失函数和优化器，以及一些超参数的初始化，例如epochs
2. 训练
3. 验证

1)通过valid数据进行训练完模型的验证

1. 测试

1)用测试集数据测试数据

**TensorFlow的使用**

**在TensorFlow中，所有的操作只有当你执行，或者另一操作依赖与这一操作的结果，这一操作才会被执行**

1. 在创建weights和bias或其他变量是，要将初始值传入构造函数Variable(),因为训练需要update weights和bias的值。

需要指明变量的shape, 最好把name值也指定，因为tf会自动给变量命名，之后如果有重复赋值可能会报错，每个变量指定唯一name

weights = tf.Variable(tf.random\_normal([784, 200], stddev=0.35),

name="weights")

biases = tf.Variable(tf.zeros([200]), name="biases")

1. 构建完model后需要初始化全部变量，这样变量才能进行操作在Session中初始化这个操作

# Add an op to initialize the variables.

init\_op = tf.initialize\_all\_variables()

with tf.Session() as sess:

# Run the init operation.

sess.run(init\_op)

1. 关于变量由另一个变量初始化，使用变量的initialized\_value()来初始化

weights = tf.Variable(tf.random\_normal([784, 200], stddev=0.35),

name="weights")

w2 = tf.Variable(weights.initialized\_value(), name="w2")

w\_twice = tf.Variable(weights.initialized\_value() \* 0.2, name="w\_twice")

1. 变量的保存与加载(model也一样)

Saver = tf.train.Saver()

With tf.Session() as sess:

Sess.run(init\_op)

Saver.save(sess, “/tem/model.ckpt”)

# 加载

Saver.restore(sess, “/tem/model.ckpt”)

**TensorBoard的使用**

对模型进行可视化操作，还可以记录一些比如learning rate或者loss的变化等

Merge = tf.merge\_all\_summaries()

Summary\_writer = tf.train.SummaryWriter(“/tmp/mnist\_logs”, sess.graph)

Total\_step=0

While training:

Total\_step += 1

Session.run(training\_op)

If total\_step % 100 == 0:

Summary\_str = session.run(Merge)

Summary\_writer.add\_summary(summary\_str, total\_step)