深度学习方法与实践第五次作业

姓名：杨玉雷
学号：18023040

1.基础作业：Slim Lenet

用 slim 定义 Lenet 网络，并训练测试。要求：
1. 将 Lenet 单独定义到 Lenet.py 文件
可以定义为一个函数，例如：
def lenet(images):

2. 用 with slim.arg_scope .....: 去管理 lenet 中所有操作的默认参数，例如 activation_fn, weights_initializer 等

3. 编写 mnist_train.py 脚本，训练 slim 定义的 lenet 做 MNIST 字符分类。

这里可以不要求用 slim 中的 slim.learning.train，因为这个涉及转换数据为 TFRecord 以及用队列读取等复杂操作去自动取数据。
大家可以还用以前的 sess.run 去训练模型。

提交：
文档、源码。文档包括训练截屏、结果图片等，能帮助老师快速判断结果是否正确。

2.基础作业实验过程和关键代码

根据实验要求，实验过程如下：

（1）在 Lenet.py 文件中用 slim 实现 Lenet 网络结构，定义为 lenet 函数，并在此函数用 with slim.arg_scope: 去管理 lenet 中 activation_fn、weights_initializer、weights_regularizer 等默认参数：

```
# 通过 TensorFlow-Slim 来定义 LeNet-5 的网络结构。
def lenet(images):
    with slim.arg_scope([slim.conv2d, slim.fully_connected],activation_fn=tf.nn.relu, weights_initializer=tf.truncated_normal_initializer(0.0, 0.1),weights_regularizer=slim.l2_regularizer(0.0005)):
        inputs = tf.reshape(images, [-1, 28, 28, 1])
        net = slim.conv2d(inputs, 32, [5, 5], padding='SAME', scope='layer1-conv')
        net = slim.max_pool2d(net, 2, stride=2, scope='layer2-max-pool')
        net = slim.conv2d(net, 64, [5, 5], padding='SAME', scope='layer3-conv')
```

```
        net = slim.max_pool2d(net, 2, stride=2, scope='layer4-max-pool')
        net = slim.flatten(net, scope='flatten')
        net = slim.fully_connected(net, 500, scope='layer5')
        net = slim.fully_connected(net, 10, scope='output')
    return net
```

（2）编写 mnist_train.py 脚本，训练 slim 定义的 lenet 做 MNIST 字符分类。

```
from tensorflow.examples.tutorials.mnist import input_data
import tensorflow as tf
from Task05.Lenet import lenet

def train(mnist):
    # 训练数据及标签
    x = tf.placeholder(tf.float32, [None, 784], name='x-input')
    y_ = tf.placeholder(tf.float32, [None, 10], name='y-input')
    # 对数据进行训练
    y = lenet(x)

    # 交叉熵
    cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits(logits=
y, labels=tf.argmax(y_, 1))
    # 计算损失
    loss = tf.reduce_mean(cross_entropy)
    # 优化
    train_op = tf.train.GradientDescentOptimizer(0.01).minimize(loss)

    #计算准确率
    correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

    with tf.Session() as sess:
        tf.global_variables_initializer().run()
        for i in range(5001):
            xs, ys = mnist.train.next_batch(100)
            _, loss_value,acc = sess.run([train_op, loss,accuracy], feed_di
ct={x: xs, y_: ys})
            if i % 100 == 0:
                print("Step:",i," training batch loss:",loss_value," accura
cy:",acc)

mnist = input_data.read_data_sets(r'MNIST_data', one_hot=True)
train(mnist)
```

可通过调节学习率等参数调整 Lenet 对 Mnist 进行训练，当学习率为 0.01 时，其

中一个训练过程和实验结果如下：

```
Step: 0  training batch loss: 2.93418  accuracy: 0.15
Step: 100  training batch loss: 2.07251  accuracy: 0.34
Step: 200  training batch loss: 1.8043  accuracy: 0.45
Step: 300  training batch loss: 1.51521  accuracy: 0.53
Step: 400  training batch loss: 1.21636  accuracy: 0.69
Step: 500  training batch loss: 1.39962  accuracy: 0.58
Step: 600  training batch loss: 1.15166  accuracy: 0.68
Step: 700  training batch loss: 0.816832  accuracy: 0.75
Step: 800  training batch loss: 1.07062  accuracy: 0.64
Step: 900  training batch loss: 0.910688  accuracy: 0.69
Step: 1000  training batch loss: 0.96184  accuracy: 0.72
Step: 1100  training batch loss: 0.787549  accuracy: 0.78
Step: 1200  training batch loss: 0.976771  accuracy: 0.68
Step: 1300  training batch loss: 0.706286  accuracy: 0.79
Step: 1400  training batch loss: 0.887198  accuracy: 0.71
Step: 1500  training batch loss: 0.83228  accuracy: 0.73
Step: 1600  training batch loss: 0.848814  accuracy: 0.73
                    ......................
Step: 3200  training batch loss: 0.52576  accuracy: 0.83
Step: 3300  training batch loss: 0.472019  accuracy: 0.83
Step: 3400  training batch loss: 0.334657  accuracy: 0.87
Step: 3500  training batch loss: 0.417486  accuracy: 0.86
Step: 3600  training batch loss: 0.458623  accuracy: 0.84
Step: 3700  training batch loss: 0.501434  accuracy: 0.82
Step: 3800  training batch loss: 0.530328  accuracy: 0.82
Step: 3900  training batch loss: 0.458526  accuracy: 0.84
Step: 4000  training batch loss: 0.319755  accuracy: 0.89
Step: 4100  training batch loss: 0.35298  accuracy: 0.88
Step: 4200  training batch loss: 0.5386  accuracy: 0.82
Step: 4300  training batch loss: 0.445956  accuracy: 0.82
Step: 4400  training batch loss: 0.428818  accuracy: 0.86
Step: 4500  training batch loss: 0.296021  accuracy: 0.91
Step: 4600  training batch loss: 0.420627  accuracy: 0.86
Step: 4700  training batch loss: 0.350656  accuracy: 0.86
Step: 4800  training batch loss: 0.491636  accuracy: 0.84
Step: 4900  training batch loss: 0.213601  accuracy: 0.92
Step: 5000  training batch loss: 0.272351  accuracy: 0.92
```