浙江大学实验报告

专业：计算机科学与技术
姓名：张雯琪
学号：3180103770
日期：2021/01/07

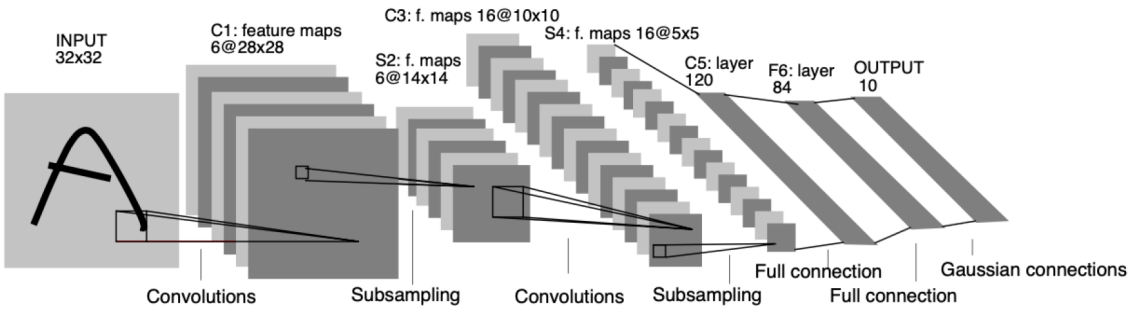
课程名称：计算机视觉 指导老师：宋明黎 成绩：xx
实验名称： 利用CNN进行手写数字识别

1 实验目的和要求

利用Mnist数据集、LeNet-5网络结构以及选定的TensorFlow框架实现手写数字的识别

2 实验内容和原理

2.1 LeNet-5网络结构



- 输入层的原始图像大小为32*32。
- C1层为卷积层，使用6个卷积核，每个核大小为5*5，得到6个特征图，每个特征图的大小为28*28。
- S2为池化层，池化单元大小为2*2，6个特征图经池化后大小变为14*14。
- C3层为卷积层，使用16个卷积核，每个核大小为5*5，得到16个特征图，每个特征图的大小为10*10。
- S4层为池化层，池化单元大小为2*2，16个特征图经池化后大小变为5*5。
- C5层为卷积层，使用120个卷积核，每个核大小为5*5，得到120个特征图，每个特征图的大小为1*1。
- F6层为全连接层，使用84个单元，得到84个特征图，每个特征图的大小为1*1，与C5层全连接。

- 第7层为输出层，也是全连接层，共有10个节点，分别代表数字0到9，如果第i个节点的值为0，则表示网络识别的结果是数字i。

3 实验步骤与分析

3.1 模型搭建

在搭建卷积层时，调用tf.Variable()函数分配weight和bias变量，并调用tf.nn.conv2d搭建卷积层，激活函数使用relu。在搭建池化层时，调用tf.nn.max_pool()函数。在全连接层中，用矩阵的乘法进行实现。

```
def lenet(x):
    # C1
    conv1_W = tf.Variable(tf.truncated_normal(
        shape=(5, 5, 1, 6), mean=config.mu, stddev=config.sigma))
    conv1_b = tf.Variable(tf.zeros(6))
    conv1 = tf.nn.conv2d(x, conv1_W, strides=[
                         1, 1, 1, 1], padding='VALID') + conv1_b
    conv1 = tf.nn.relu(conv1)

    # S2
    conv1 = tf.nn.max_pool(conv1, ksize=[1, 2, 2, 1], strides=[
                           1, 2, 2, 1], padding='VALID')

    # C3
    conv2_W = tf.Variable(tf.truncated_normal(
        shape=(5, 5, 6, 16), mean=config.mu, stddev=config.sigma))
    conv2_b = tf.Variable(tf.zeros(16))
    conv2 = tf.nn.conv2d(conv1, conv2_W, strides=[
                         1, 1, 1, 1], padding='VALID') + conv2_b
    conv2 = tf.nn.relu(conv2)

    # S4
    conv2 = tf.nn.max_pool(conv2, ksize=[1, 2, 2, 1], strides=[
                           1, 2, 2, 1], padding='VALID')

    # C5
    fc0 = flatten(conv2)
    fc1_W = tf.Variable(tf.truncated_normal(
        shape=(400, 120), mean=config.mu, stddev=config.sigma))
    fc1_b = tf.Variable(tf.zeros(120))
    fc1 = tf.matmul(fc0, fc1_W) + fc1_b
    fc1 = tf.nn.relu(fc1)

    # F6
    fc2_W = tf.Variable(tf.truncated_normal(
        shape=(120, 84), mean=config.mu, stddev=config.sigma))
    fc2_b = tf.Variable(tf.zeros(84))
    fc2 = tf.matmul(fc1, fc2_W) + fc2_b
    fc2 = tf.nn.relu(fc2)

    # output
    fc3_W = tf.Variable(tf.truncated_normal(
```

```
        shape=(84, 10), mean=config.mu, stddev=config.sigma))
    fc3_b = tf.Variable(tf.zeros(10))
    logits = tf.matmul(fc2, fc3_W) + fc3_b

    return logits
```

3.2 模型训练

TensorFlow中自带从MNIST数据集读入数据的函数。读入数据后，将每张28*28图片扩展为32*32，即上下左右的边界各加入两行/列空白。调用tf.placeholder()定义网络的输入输出，定义softmax交叉熵，定义Adam优化算法。

接下来，进行模型的训练。在每次迭代前，对数据进行打乱，以防过拟合。每次迭代后，计算验证集上的准确率。

```
    with tf.Session() as sess:
        print("Start training", file=sys.stderr)
        sess.run(tf.global_variables_initializer())
        num_examples = len(x_train)

        for i in range(config.epochs):
            x_train, y_train = shuffle(x_train, y_train)
            for offset in range(0, num_examples, config.batchSize):
                end = offset + config.batchSize
                batch_x, batch_y = x_train[offset:end], y_train[offset:end]
                sess.run(training_operation, feed_dict={
                         x: batch_x, y: batch_y})

            validation_accuracy = evaluate(x_validation, y_validation)
            print(i, validation_accuracy)
            print(i, validation_accuracy, file=sys.stderr)

        saver.save(sess, config.model)
        print("Finish training", file=sys.stderr)
        print()
```

3.3 模型测试

训练完成后，得到模型文件。使用测试集的数据进行测试。

```
    with tf.Session() as sess:
        saver.restore(sess, tf.train.latest_checkpoint(config.modelDir))
        test_accuracy = evaluate(x_test, y_test)
        print("Accuracy with {} test data = {:.3f}".format(
            len(x_test), test_accuracy))
        print("Accuracy with {} test data = {:.3f}".format(
            len(x_test), test_accuracy), file=sys.stderr)
```

4 实验结果

4.1 程序运行

测试数据集及网络结构路径

```
datasetDir = '../dataset/'
model = '../model/lenet'
modelDir = '../model/'
```

```
python test.py
```

4.2 程序输出

```
PS D:\Document\ZJU\Courses\20\20FW\计算机视觉\作业\hw_bonus> python test.py
Image shape: (28, 28, 1)
Training set length: 55000
Validation set length: 5000
Test set length: 10000

Image shape padded: (32, 32, 1)
Start training
0 0.9658
1 0.9794
2 0.982
3 0.9832
4 0.986
5 0.9872
6 0.9886
7 0.9872
8 0.9882
9 0.9896
10 0.9858
11 0.9856
12 0.9912
13 0.9888
14 0.9874
15 0.9862
16 0.9876
17 0.9884
18 0.9888
19 0.9894
Finish training

Accuracy with 10000 test data = 0.991
```