专业：计算机科学与技术
姓名：卢佳盈
学号：3180103570
日期：2020/1/12

浙江大学实验报告

课程名称：计算机视觉 指导老师：宋明黎 成绩：

实验名称：光荣题：学习 CNN

一、实验目的和要求

框架：TensorFlow（已包含下面网络结构与数据集）

数据集：The Mnist Database of handwritten digits

网络结构：LeNet-5

具体任务：利用上述数据集、网络结构以及选定的 TensorFlow 框架实现手写数字的识别

二、实验内容和说明

2.1 开发环境

- Windows X64
- Python 3.7.4
- TensorFlow 1.14.0

2.2 运行方式

```
python main.py
```

三、 实现过程

3.1 加载 MNIST 数据库

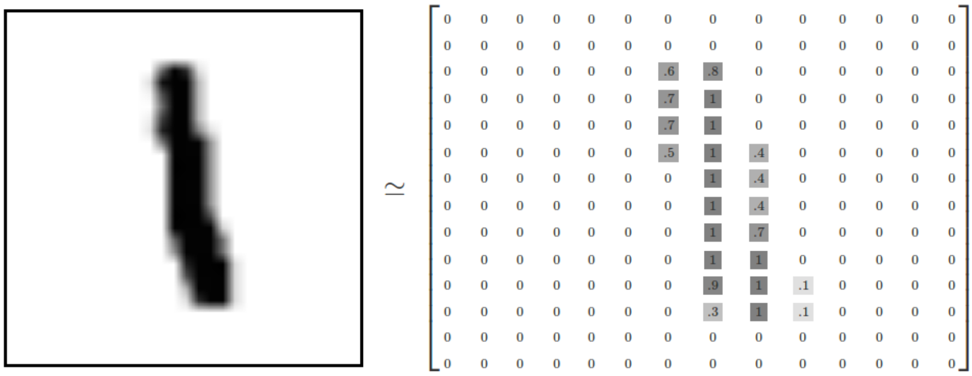
下载 MNIST 数据库放在代码所在目录下，该数据库中包含内容如下：

```
MNIST_data
 |   train-images-idx3-ubyte.gz
 |   train-labels-idx1-ubyte.gz
 |   t10k-images-idx3-ubyte.gz
 |   t10k-labels-idx1-ubyte.gz
```

代码如下：

```python
from tensorflow.examples.tutorials.mnist import input_data
data_path="./MNIST_data"
mnist=input_data.read_data_sets(data_path,reshape=False)
```

每个 MNIST 图片包含个像素，如下图表示：



3.2 数据集处理

【划分数据集】

```
x_train,y_train=mnist.train.images,mnist.train.labels
x_val, y_val = mnist.validation.images, mnist.validation.labels
x_test, y_test = mnist.test.images, mnist.test.labels
```

【Padding 预处理】

```
x_train = np.pad(x_train, [(0,0),(2,2),(2,2),(0,0)], "constant")
x_val = np.pad(x_val, [(0,0),(2,2),(2,2),(0,0)], "constant")
x_test = np.pad(x_test, [(0,0),(2,2),(2,2),(0,0)], "constant")
```

【随机打乱】

```
x_train,y_train=shuffle(x_train,y_train)
```

【标签 one-hot 处理】

```
x = tf.placeholder(tf.float32, shape=(None, 32, 32, 1))
y = tf.placeholder(tf.int32, shape=(None))
y_onehot=tf.one_hot(y,10)
```

将每个 MNIST 标签转换为一个 one-hot 编码的向量

3.3 LeNet-5

手写数字识别实现使用的是卷积神经网络，主要网络结构为 LeNet-5，如下图所示：

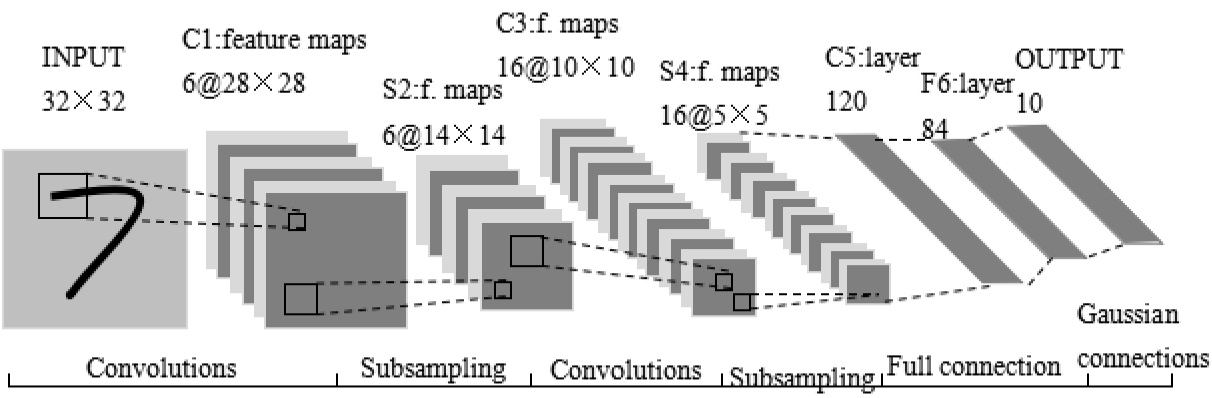


图 2.3 LeNet-5 结构

LeNet-5 不包括输入，一共 7 层，较低层由卷积层和最大池化层交替构成，更高层则是全连接和高斯连接。

第一个卷积层（C1 层）由 6 个特征映射构成，每个特征映射是一个 28×28 的神经元阵列，其中每个神经元负责从 5×5 的区域通过卷积滤波器提取局部特征。

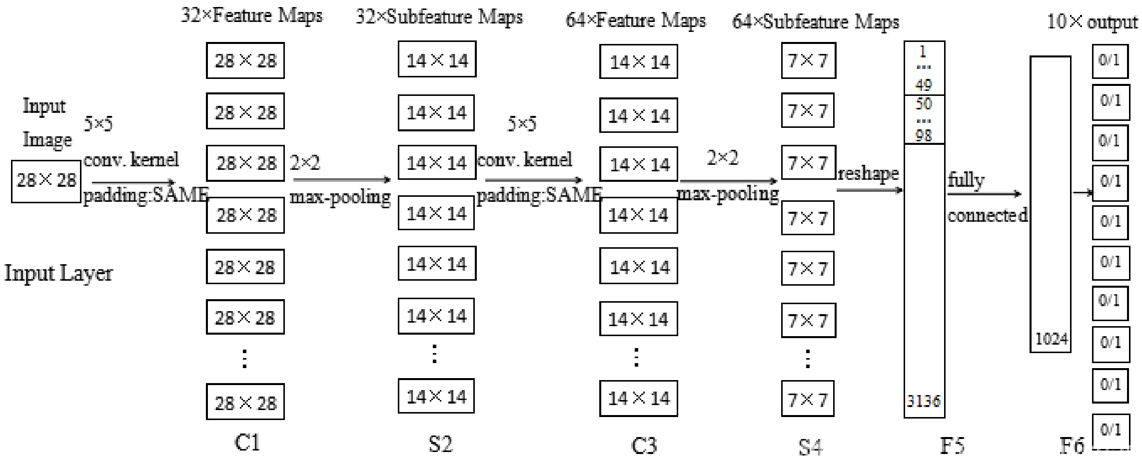
S2 层是对应上述 6 个特征映射的降采样层（pooling 层）。pooling 层的实现方法有两种，分别是 max-pooling 和 mean-pooling，LeNet-5 采用的是 mean-pooling，即取 n×n 区域内像素的均值。

S2 层和 C3 层的连接比较复杂。C3 卷积层是由 16 个大小为 10×10 的特征映射组成的，当中的每个特征映射与 S2 层的若干个特征映射的局部感受野（大小为 5×5）相连。

S4 层是对 C3 层进行的降采样，与 S2 同理，学习参数有 16×1+16=32 个，同时共有 16×(2×2+1)×5×5=2000 个连接。

C5 层是由 120 个大小为 1×1 的特征映射组成的卷积层，而且 S4 层与 C5 层是全连接的，因此学习参数总个数为 120×(16×25+1)=48120 个。

F6 是与 C5 全连接的 84 个神经元。网络模型的总体结构如下：



代码如下：

```
def LeNet(x):
    mu = 0
    sigma = 0.1
    conv1_w = tf.Variable(tf.truncated_normal(shape=[5,5,1,6], mean=mu, stddev=sigm
a))
    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)
    pool_1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1],strides=[1,2,2,1], padding="VALI
D")

    conv2_w = tf.Variable(tf.truncated_normal(shape=[5,5,6,16], mean=mu, stddev=sig
ma))
    conv2_b = tf.Variable(tf.zeros(16))
    conv2 = tf.nn.conv2d(pool_1, conv2_w, strides=[1,1,1,1], padding="VALID") + con
v2_b

    conv2 = tf.nn.relu(conv2)
```

```
    pool_2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1],strides=[1,2,2,1], padding="VALI
D")

    fc1 = flatten(pool_2)
    fc1_w = tf.Variable(tf.truncated_normal(shape=(400,120), mean=mu, stddev=sigma)
)
    fc1_b = tf.Variable(tf.zeros(120))
    fc1 = tf.matmul(fc1, fc1_w)+fc1_b
    fc1 = tf.nn.relu(fc1)

    fc2_w = tf.Variable(tf.truncated_normal(shape=(120,84), mean=mu, stddev=sigma))
    fc2_b = tf.Variable(tf.zeros(84))
    fc2 = tf.matmul(fc1, fc2_w) + fc2_b
    fc2 = tf.nn.relu(fc2)

    fc3_w = tf.Variable(tf.truncated_normal(shape=(84,10), mean=mu, stddev=sigma))
    fc3_b = tf.Variable(tf.zeros(10))
    logits = tf.matmul(fc2, fc3_w) + fc3_b

    return logits
```

3.4 获取损失函数

使用交叉熵来就算损失函数，loss 越小说明模型越精确

$$H_{y}(y) = -\sum_{i} y_i' \log(y_i)$$

```
logits = LeNet(x)
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(labels=y_onehot, logits=log
its)
loss_operation = tf.reduce_mean(cross_entropy)
```

3.5 Adam 优化器

使用 Adam 优化器来进行训练

```
learning_rate=0.001
optimizer = tf.train.AdamOptimizer(learning_rate)
```

3.6 模型训练

使用了 batch 来训练，其中 batch_size 的默认大小为 128，默认 Epochs 数为 10，具体代码如下：

```python
with tf.Session() as sess:
    total_batch = int(mnist.train.num_examples/Batch_size)
    writer = tf.summary.FileWriter(log_path)
    writer.add_graph(sess.graph)

    sess.run(tf.global_variables_initializer())
    num_examples = len(x_train)
    print("-------Start Training-------")
    for i in range(Epochs):
        x_train, y_train = shuffle(x_train, y_train)
        for j in range(total_batch):
            begin = j * Batch_size
            end = begin + Batch_size
            # batch_x, batch_y = mnist.train.next_batch(Batch_size)
            batch_x, batch_y = x_train[begin:end], y_train[begin:end]
            _, summ1 = sess.run([training_operation,summ], feed_dict={x:batch_x, y:
batch_y})
            # writer.add_summary(summ1, i)
            writer.add_summary(summ1, i * total_batch + j)
        val_acc = evaluate(x, y, x_val, y_val, Batch_size, acc)
        print("Epochs {}".format(i+1))
        print("Validation Accuracy = {:.4f}".format(val_acc))
        saver.save(sess, model_path+"model.ckpt")
    print("model saved")

    writer.close()
```

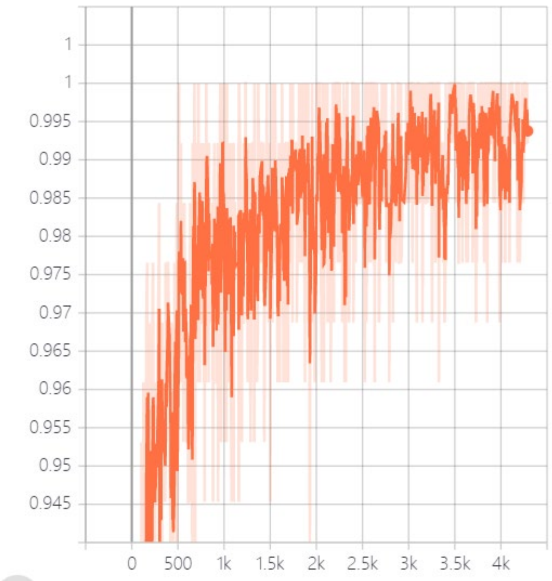
3.7 模型测试

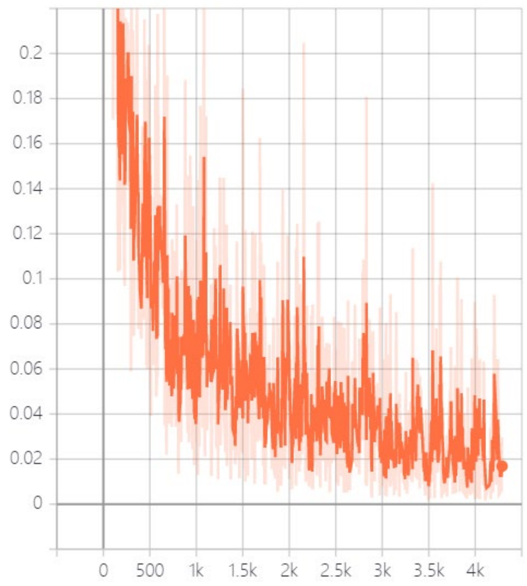
```python
with tf.Session() as sess:
    total_batch = int(mnist.train.num_examples/Batch_size)
    writer = tf.summary.FileWriter(log_path)
    writer.add_graph(sess.graph)

    sess.run(tf.global_variables_initializer())
    num_examples = len(x_train)
    print("-------Start Testing-------")
    model_file = tf.train.latest_checkpoint(model_path)
    saver.restore(sess,model_file)
    test_acc = sess.run(acc, feed_dict={x: x_test, y: y_test})
    print('test_acc: {:.4f}'.format(test_acc))
```

四、实验结果

【训练过程】

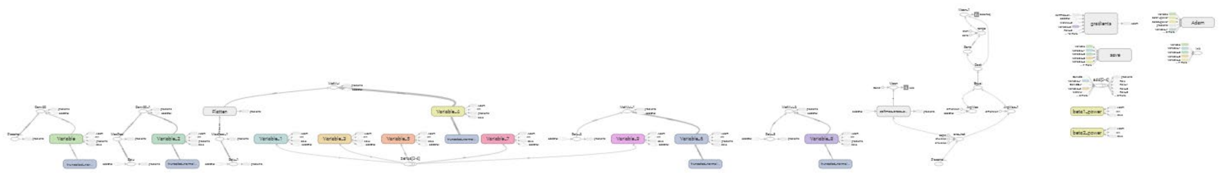
将训练过程使用 tensorboard 进行可视化描绘：





accuracy **loss**

神经网络结构图：



【训练结果】

```
-------Start Training-------
Epochs 1
Validation Accuracy = 0.9694
Epochs 2
Validation Accuracy = 0.9792
Epochs 3
Validation Accuracy = 0.9818
Epochs 4
Validation Accuracy = 0.9874
Epochs 5
Validation Accuracy = 0.9876
Epochs 6
Validation Accuracy = 0.9854
Epochs 7
Validation Accuracy = 0.9874
Epochs 8
Validation Accuracy = 0.9892
Epochs 9
Validation Accuracy = 0.9872
Epochs 10
Validation Accuracy = 0.9880
model saved
```

【测试结果】

```
-------Start Testing-------
WARNING:tensorflow:From D:\anaconda\lib\site-packages\tensorflow\python\trai
t_management) is deprecated and will be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
INFO:tensorflow:Restoring parameters from ./model/model.ckpt
test_acc: 0.9883
```

可以看出对于手写数字的识别率较高