实 验 报 告

学生姓名	王科翔	学号	16061040	指导老师	李辉勇
实验地点	F332	实验时间	19.03.26	班级	160612

一、 实验名称

理解并掌握基于卷积神经网络的图像分类任务

二、 实验学时

4

三、 实验原理

卷积神经网络（Convolutional Neural Network，简称 CNN），是一种前馈神经网络，人工神经元可以响应周围单元，可以进行大型图像处理。卷积神经网络包括卷积层和池化层。

CNN 网络一共有 5 个层级结构：

1. 输入层
2. 卷积层
3. 激活层
4. 池化层
5. 全连接 FC 层

输入层

与传统神经网络/机器学习一样，模型需要输入的进行预处理操作，常见的 3 种预处理方式有：

1. 去均值
2. 归一化
3. PCA/SVD 降维等

卷积层

局部感知：人的大脑识别图片的过程中，并不是一下子整张图同时识别，而是对于图片中的每一个特征首先局部感知，然后更高层次对局部进行综

合操作，从而得到全局信息。

权值共享：卷积层使用“卷积核”进行局部感知。对于每一层来讲，所有神经元对应的权值是相等的。

激励层

所谓激励，实际上是对卷积层的输出结果做一次非线性映射。

如果不用激励函数（其实就相当于激励函数是 f(x)=x），这种情况下，每一层的输出都是上一层输入的线性函数。容易得出，无论有多少神经网络层，输出都是输入的线性组合，与没有隐层的效果是一样的，这就是最原始的感知机了。

常用的激励函数有：

1. Sigmoid 函数
2. Tanh 函数
3. ReLU
4. Leaky ReLU
5. ELU
6. Maxout

池化层

池化（Pooling）：也称为欠采样或下采样。主要用于特征降维，压缩数据和参数的数量，减小过拟合，同时提高模型的容错性。主要有：

1. Max Pooling：最大池化
2. Average Pooling：平均池化

输出层

经过前面若干次卷积+激励+池化后，终于来到了输出层，模型会将学到的一个高质量的特征图片全连接层。其实在全连接层之前，如果神经元数目过大，学习能力强，有可能出现过拟合。因此，可以引入 dropout 操作，来随机删除神经网络中的部分神经元，来解决此问题。还可以进行局部归一化（LRN）、数据增强等操作，来增加鲁棒性，这里不做介绍。

四、 实验目的

理解并掌握基于卷积神经网络的图像分类任务。

五、 实验内容

通过 PC 上位机连接服务器，登陆 SimpleAI 平台，利用 python 语言编写 CNN 代码。实验使用 Kaggle 竞赛平台上的 Flower Recognition 数据集，包含 5 种花卉种类，总计 4242 张标记过的图片。通过搭建多层卷积神经网络进行监督学习完成分类任务。

。

六、 实验步骤

任务 1：

修改代码，可视化查看经过 cnn 处理后的特征图。去掉 x = x[:, :2]之后就是取了所有统特征

```
conv1_img, pool1_img, conv2_img, pool2_img, conv3_img, pool3_img = sess.run(
    [conv1, pool1, conv2, pool2, conv3, pool3])
loc = 1
transpose = sess.run(tf.transpose(conv1_img, [3, 0, 1, 2]))
print(transpose.shape)
fig, ax = plt.subplots(nrows=8, ncols=8, figsize=(28, 28))
for i in range(8):
    for j in range(8):
        ax[i][j].imshow(transpose[i * 8 + j][loc])
plt.title('Conv1 64x28x28')
plt.show()

transpose = sess.run(tf.transpose(pool1_img, [3, 0, 1, 2]))
print(transpose.shape)
fig, ax = plt.subplots(nrows=8, ncols=8, figsize=(28, 28))
for i in range(8):
    for j in range(8):
        ax[i][j].imshow(transpose[i * 8 + j][loc])
plt.title('pool1 64x14x14')
plt.show()

transpose = sess.run(tf.transpose(conv2_img, [3, 0, 1, 2]))
print(transpose.shape)
fig, ax = plt.subplots(nrows=4, ncols=8, figsize=(28, 28))
for i in range(4):
    for j in range(8):
        ax[i][j].imshow(transpose[i * 4 + j][loc])
plt.title('Conv2 32x14x14')
plt.show()
```

（部分代码）

任务 2：

调整超参数或者网络结构（比如增减层），在迭代一定的情况下，尽量提高精度，并对结果进行分析说明。

具体操作，先调整 learning_rate，再调整 batch_size 和 capacity，增加网络中卷积核的数量，扩大输入图片的大小。

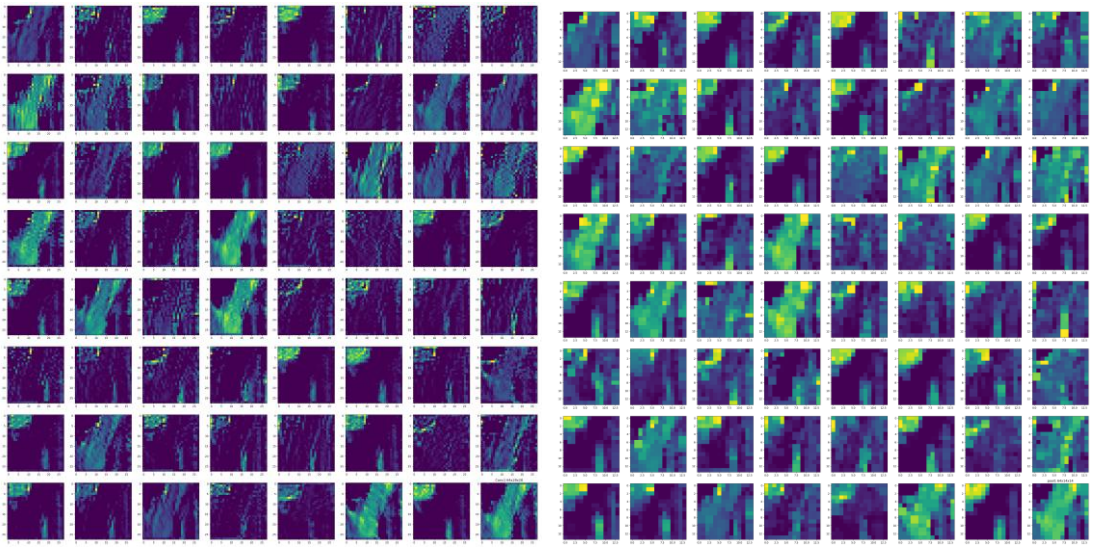
```
import tensorflow as tf

N_CLASSES = 5
IMG_W = 56
IMG_H = 56
BATCH_SIZE = 40
CAPACITY = 400
MAX_STEP = 10000
learning_rate = 0.0001
```

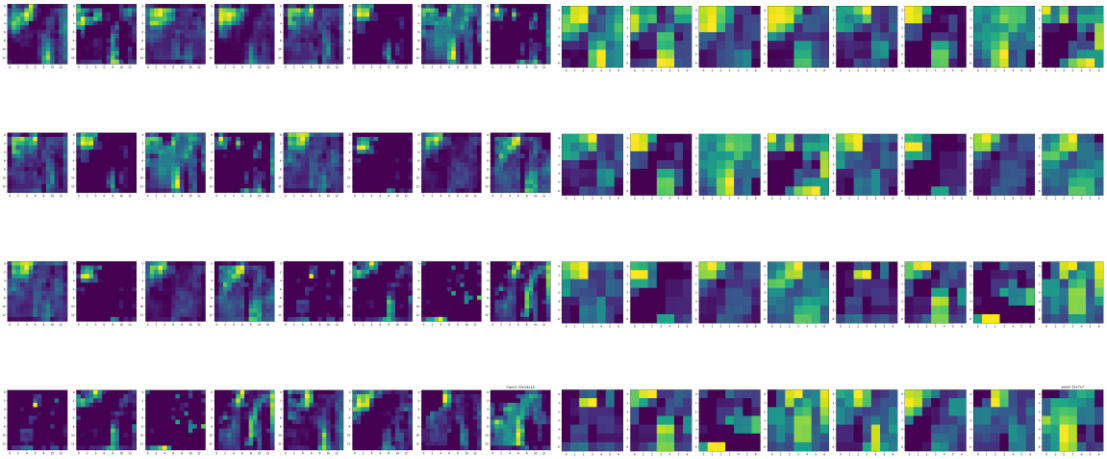
```
with tf.variable_scope('conv1') as scope:
    w_conv1 = tf.Variable(weight_variable([3, 3, 3, 64], 1.0), name='weights',
e=tf.float32)
    b_conv1 = tf.Variable(bias_variable([64]), name='biases', dtype=tf.float32)
    h_conv1 = tf.nn.relu(conv2d(images, w_conv1) + b_conv1, name='conv1')

with tf.variable_scope('pooling1_lrn') as scope:
    pool1 = max_pooling_2x2(h_conv1, 'pooling1')
    norm1 = tf.nn.lrn(pool1, depth_radius=4, bias=1.0, alpha=0.001 / 9.0,
=0.75, name='norm1')

with tf.variable_scope('conv2') as scope:
    w_conv2 = tf.Variable(weight_variable([3, 3, 64, 128], 0.1), name='weights',
e=tf.float32)
    b_conv2 = tf.Variable(bias_variable([128]), name='biases', dtype=tf.float32)
    h_conv2 = tf.nn.relu(conv2d(norm1, w_conv2) + b_conv2, name='conv2')

with tf.variable_scope('pooling2_lrn') as scope:
    pool2 = max_pooling_2x2(h_conv2, 'pooling2')
    norm2 = tf.nn.lrn(pool2, depth_radius=4, bias=1.0, alpha=0.001 / 9.0,
=0.75, name='norm2')

with tf.variable_scope('conv3') as scope:
    w_conv3 = tf.Variable(weight_variable([3, 3, 128, 256], 0.1), name='weights',
e=tf.float32)
    b_conv3 = tf.Variable(bias_variable([256]), name='biases', dtype=tf.float32)
```

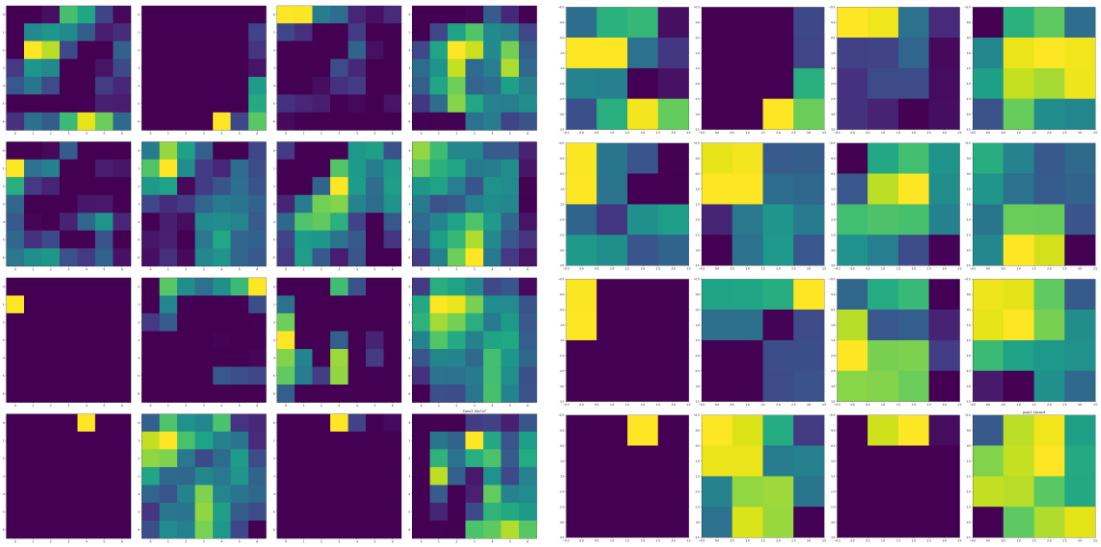
七、 实验结果及分析：

任务 1：

以下分别为一张图经过 conv1, pool1, conv2, pool2, conv3, pool3 层后的结果

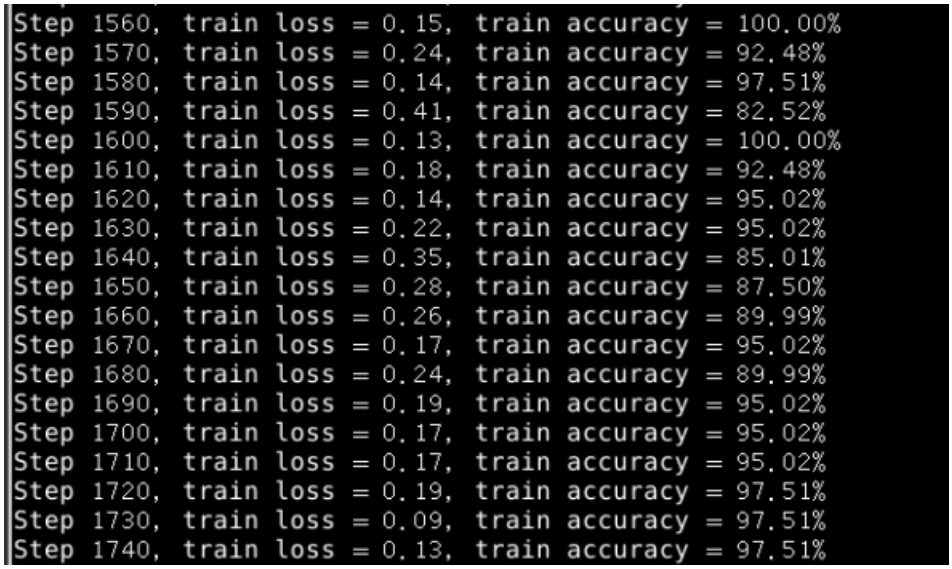


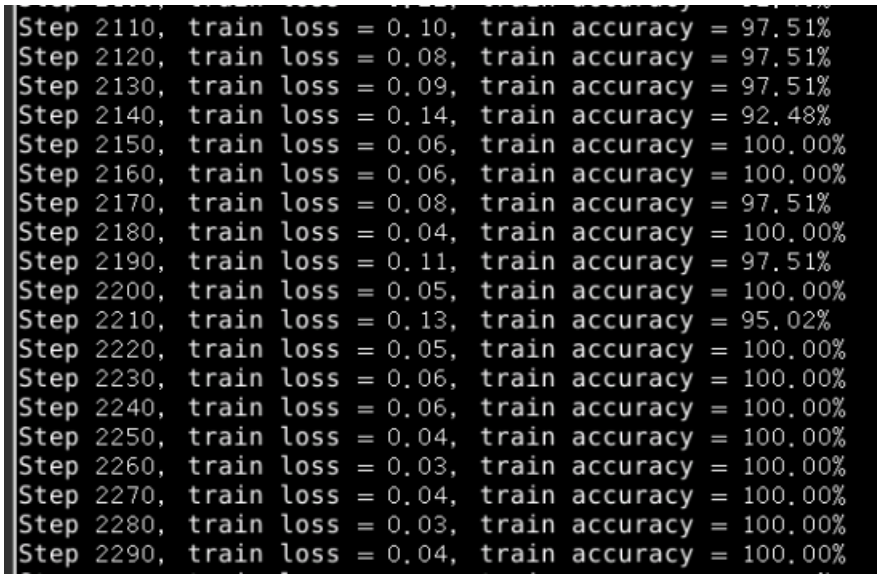




任务 2：

```
Step 1230, train loss = 0.28, train accuracy = 92.48%
Step 1240, train loss = 0.63, train accuracy = 75.00%
Step 1250, train loss = 0.80, train accuracy = 70.02%
Step 1260, train loss = 0.60, train accuracy = 79.98%
Step 1270, train loss = 0.57, train accuracy = 75.00%
Step 1280, train loss = 0.53, train accuracy = 79.98%
Step 1290, train loss = 0.43, train accuracy = 79.98%
Step 1300, train loss = 0.62, train accuracy = 82.52%
Step 1310, train loss = 0.28, train accuracy = 87.50%
Step 1320, train loss = 0.44, train accuracy = 87.50%
Step 1330, train loss = 0.34, train accuracy = 87.50%
Step 1340, train loss = 0.42, train accuracy = 79.98%
Step 1350, train loss = 0.47, train accuracy = 82.52%
Step 1360, train loss = 0.42, train accuracy = 85.01%
Step 1370, train loss = 0.47, train accuracy = 79.98%
Step 1380, train loss = 0.24, train accuracy = 89.99%
Step 1390, train loss = 0.35, train accuracy = 87.50%
Step 1400, train loss = 0.37, train accuracy = 82.52%
```





八、 实验结论：

任务 1：

可以看到，提取到的特征越来越抽象，课程给出的代码，卷基础层数越来越低这是不合常理的，这样会使可利用的信息越来越少。任务二中予以修正。

任务 2：

在过随着时间增长，在训练集上的准确率越来越高。
实验表明：
batch 的大小可以快速提高准确率。
learning_rate 过大会导致，准确率经常发生震荡，特别难稳定下来。过小会导致，准确率提升特别缓慢。
增加了卷积层数后，可以表达更多更加抽象的图片特征，故而准确率提高。
增加了图片尺寸后，表达信息也得到了较大提升。
但是显然，这样的网络已经发生了过拟合。

九、 总结及心得体会

实验总结

学到了关于 CNN 的理论知识和实践能力，提高了对于机器学习的实践能力。

存在问题

对于现有框架的理解不够深刻。不能非常自由的修改代码，不太会调参。给出的代码似乎有点问题。一个是卷积层数越来越低，一个是似乎有个地方写错了，导致网络层之间的连接有点错乱。

对本实验意见或建议

提醒学生那些参数是训练的重点，让学生的训练有所目的，同时给出一些关于画图的提示，否则对于新手，画图就很麻烦了。
给出的代码应该对泛化性能进行测试。