计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目：Convolutional neural networks		学号：201900130143
日期：10/28	班级： 智能班	姓名：吴家麒
Email：wjq_777@126.com		

实验目的：

In this assignment you will understand the architecture of Convolutional Neural Networks and get practice with training these models on data.

- you will be given two subtasks: Building a Convolutional Neural Network Model and Application, using Keras to build a residual network (Optional).

实验软件和硬件环境：

Anaconda3 + Jupyter notebook

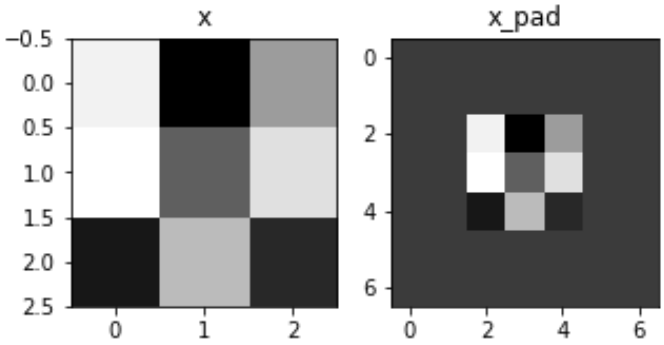
实验原理和方法：

Convolutional neural networks 的基本结构包括卷积层、池化层

实验步骤：（不要求罗列完整源代码）

1、 一步步的构建 CNN

首先对图像的边缘进行 zero-padding：



尝试对数据进行单步卷积，得到卷积后的和：

```
Z = -6.999089450680221
```

Expected Output:

Z	-6.99908945068

下一步进行 CNN 的前向传播卷积计算，

```
Z's mean = 0.048995203528855794
Z[3,2,1] = [-0.61490741 -6.7439236  -2.55153897  1.75698377  3.56208902  0.53036437
  5.18531798  8.75898442]
cache_conv[0][1][2][3] = [-0.20075807  0.18656139  0.41005165]
```

Expected Output:

Z's mean	0.0489952035289
Z[3,2,1]	[-0.61490741 -6.7439236 -2.55153897 1.75698377 3.56208902 0.53036437 5.18531798 8.75898442]
cache_conv[0][1][2][3]	[-0.20075807 0.18656139 0.41005165]

然后需要构建 Pooling Layer（池化层），
计算 max pooling 和 average pooling 后得到的值：

```
mode = max
A = [[[[1.74481176 0.86540763 1.13376944]]]


 [[[1.13162939 1.51981682 2.18557541]]]]

mode = average
A = [[[[ 0.02105773 -0.20328806 -0.40389855]]]


 [[[-0.22154621  0.51716526  0.48155844]]]]
```

Expected Output:

A = [[[[1.74481176 0.86540763 1.13376944]]]
[[[1.13162939 1.51981682 2.18557541]]]]

A = [[[[0.02105773 -0.20328806 -0.40389855]]]
[[[-0.22154621 0.51716526 0.48155844]]]]

Backpropagation:
对卷积层进行反向传播的卷积计算：

```
dA_mean = 1.4524377775388075
dW_mean = 1.7269914583139097
db_mean = 7.839232564616838
```

** Expected Output: **

dA_mean	1.45243777754
dW_mean	1.72699145831
db_mean	7.83923256462

对池化层进行反向传播的卷积计算：
构建一个 max pooling 的 mask，

```
x =  [[ 1.62434536 -0.61175641 -0.52817175]
 [-1.07296862  0.86540763 -2.3015387 ]]
mask =  [[ True False False]
 [False False False]]
```

构建 average pooling，

```
a = distribute_value(2, (2,2))
print('distributed value =', a)
```

```
distributed value = [[0.5 0.5]
 [0.5 0.5]]
```

将两种 pooling 结合，实现完整的 pooling layer：

```
mode = max
mean of dA =  0.14571390272918056
dA_prev[1,1] =  [[0.  0.]
 [0.  0.]
 [0.  0.]]

mode = average
mean of dA =  0.14571390272918056
dA_prev[1,1] =  [[-0.07752919 -0.60870944]
 [ 0.18217696 -0.06196453]
 [ 0.25970615  0.54674491]]
```

到此，CNN 的一般结构建立完成。

2、 CNN 的应用

加载手形数字数据库，查看数据集大小：

```
number of training examples = 1080
number of test examples = 120
X_train shape: (1080, 64, 64, 3)
Y_train shape: (1080, 6)
X_test shape: (120, 64, 64, 3)
Y_test shape: (120, 6)
```

创建 tensorflow 需要的 placeholder，

```
### START CODE HERE ### (≈2 lines)
X = tf.placeholder(tf.float32, shape=(None, n_H0, n_W0, n_C0))
Y = tf.placeholder(tf.float32, shape=(None, n_y))
### END CODE HERE ###
```

```
X, Y = create_placeholders(64, 64, 3, 6)
print ("X = " + str(X))
print ("Y = " + str(Y))
```

```
X = Tensor("Placeholder:0", shape=(?, 64, 64, 3), dtype=float32)
Y = Tensor("Placeholder_1:0", shape=(?, 6), dtype=float32)
```

初始化训练参数：

```
W1 = [ 0.00131723  0.1417614  -0.04434952  0.09197326  0.14984085 -0.03514394
 -0.06847463  0.05245192]
W2 = [-0.08566415  0.17750949  0.11974221  0.16773748 -0.0830943  -0.08058
 -0.00577033 -0.14643836  0.24162132 -0.05857408 -0.19055021  0.1345228
 -0.22779644 -0.1601823  -0.16117483 -0.10286498]
```

前向传播的计算，步骤如下：

```
- Conv2D: stride 1, padding is "SAME"
- ReLU
- Max pool: Use an 8 by 8 filter size and an 8 by 8 stride, padding is "SAME"
- Conv2D: stride 1, padding is "SAME"
- ReLU
- Max pool: Use a 4 by 4 filter size and a 4 by 4 stride, padding is "SAME"
- Flatten the previous output.
```

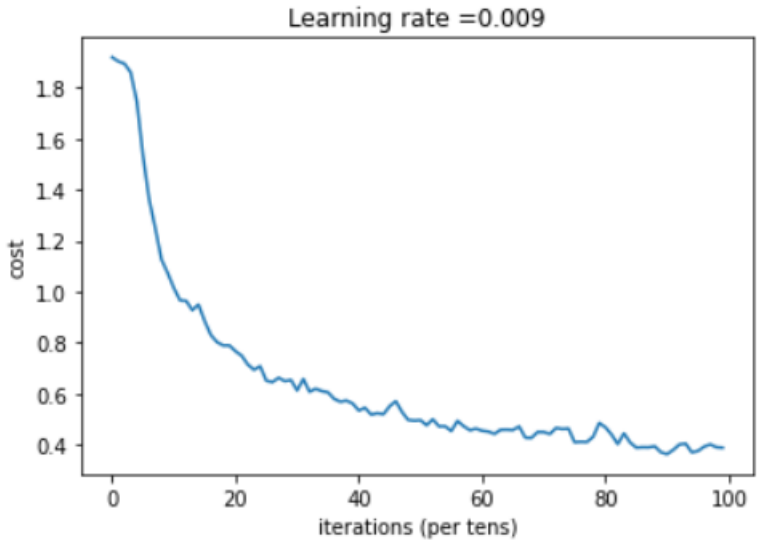
得到输出 Z3：

```
Z3 = [[-0.44670227 -1.5720876  -1.5304923  -2.3101304  -1.2910438   0.46852064]
 [-0.17601591 -1.5797201  -1.4737016  -2.616721   -1.0081065   0.5747785 ]]
```

采用 tensor 各个维度上的均值计算损失：

```
cost = 2.9103398
```

进行模型训练，结果如下：




```
Tensor("Mean_1:0", shape=(), dtype=float32)
Train Accuracy: 0.86851853
Test Accuracy: 0.73333335
```

3. ResNets 的构建

首先实现如下图所示的 identity block:

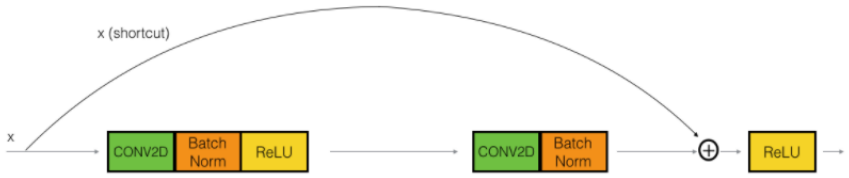


Figure 3 : **Identity block.** Skip connection "skips over" 2 layers.

当输入维度与输出维度不匹配时，可以采用另一种 convolutional block：

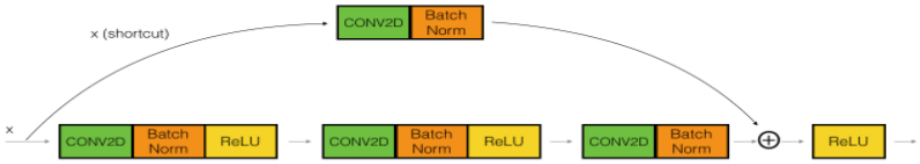


Figure 4 : **Convolutional block**

构建 ResNets 模型，
结构如下：

- Zero-padding pads the input with a pad of (3,3)
- Stage 1:
  - The 2D Convolution has 64 filters of shape (7,7) and uses a stride of (2,2). Its name is "conv1".
  - BatchNorm is applied to the channels axis of the input.
  - MaxPooling uses a (3,3) window and a (2,2) stride.
- Stage 2:
  - The convolutional block uses three set of filters of size [64,64,256], "f" is 3, "s" is 1 and the block is "a".
  - The 2 identity blocks use three set of filters of size [64,64,256], "f" is 3 and the blocks are "b" and "c".
- Stage 3:
  - The convolutional block uses three set of filters of size [128,128,512], "f" is 3, "s" is 2 and the block is "a".
  - The 3 identity blocks use three set of filters of size [128,128,512], "f" is 3 and the blocks are "b", "c" and "d".
- Stage 4:
  - The convolutional block uses three set of filters of size [256, 256, 1024], "f" is 3, "s" is 2 and the block is "a".
  - The 5 identity blocks use three set of filters of size [256, 256, 1024], "f" is 3 and the blocks are "b", "c", "d", "e" and "f".
- Stage 5:
  - The convolutional block uses three set of filters of size [512, 512, 2048], "f" is 3, "s" is 2 and the block is "a".
  - The 2 identity blocks use three set of filters of size [256, 256, 2048], "f" is 3 and the blocks are "b" and "c".
- The 2D Average Pooling uses a window of shape (2,2) and its name is "avg_pool".
- The flatten doesn't have any hyperparameters or name.

导入手形数字数据集，查看训练集结果：

```
model.fit(X_train, Y_train, epochs = 2, batch_size = 32)
```

```
Epoch 1/2
1080/1080 [==============================] - 68s 63ms/step - loss: 3.0458 - acc: 0.2537
Epoch 2/2
1080/1080 [==============================] - 70s 65ms/step - loss: 2.0860 - acc: 0.3944
```

测试集：

```
120/120 [==============================] - 3s 21ms/step
Loss = 3.0729623953501384
Test Accuracy = 0.16666666666666666
```

结论分析与体会：

1、 通过一步步的构建卷积神经网络，对于 CNN 的具体实现过程与实际结构有了深入的理解与学习。

2、 通过 tensorflow 和手形数字数据集实现了 CNN 的具体应用，让我对 tensorflow 的使用与 CNN 的应用方式进行了学习与理解。

3、 还学习了基于 shortcut connection 建立的 ResNet 模型，解决了深度 CNN 面临的问题，对于 ResNets 模型的建立过程也有了深入了学习与理解。