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：　CNN		学号：201918130222
日期：2021/10/25	班级：　智能	姓名：魏江峰

Email：2257263015@qq.com

实验目的：

Understand the architecture of CNN

Get practice with training these on data

实验软件和硬件环境：

硬件环境：

处理器：Intel core i7 9750-H

电脑：神州 z7m-ct7nk

软件环境：

Pycharm 与 jupyter notebook

实验原理和方法：

Week1:CNN

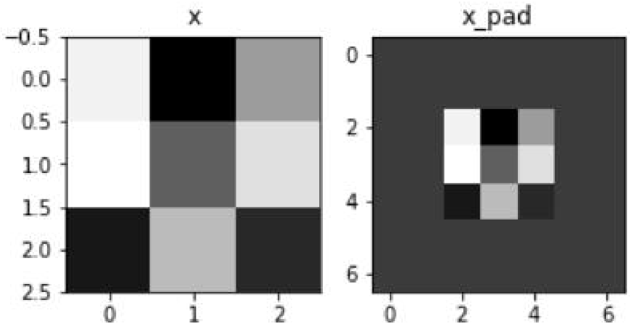
1，Convolutional model -Step by Step

（1）加载库

（2）卷积函数

1，zero_pad:

X_pad = np.pad(X,((0,0),(pad,pad),(pad,pad),(0,0)))



2,Single step of convolution

```
s = W*a_slice_prev

Z = np.sum(s)+b
```

3.3 - Convolutional Neural Networks - Forward pass

```
(m, n_H_prev, n_W_prev, n_C_prev) = A_prev.shape

(f, f, n_C_prev, n_C) = W.shape

 stride = hparameters['stride']

pad = hparameters['pad']

given above. Hint: use int() to floor. (≈2 lines)

n_H = int((n_H_prev - f +2*pad)/stride +1)

n_W = int((n_W_prev - f +2*pad)/stride +1)

Z = np.zeros((m,int(n_H),int(n_W),n_C))

A_prev_pad = zero_pad(A_prev,pad)

for i in range(m):

    a_prev_pad = A_prev_pad[i,:,:,:]

    for h in range(n_H):
```

```
        for w in range(n_W):
            for c in range(n_C):
                vert_start = h*stride
                vert_end   = h*stride +f
                horiz_start = w*stride
                horiz_end = w*stride +f
                a_slice_prev                                      =
            a_prev_pad[vert_start:vert_end,horiz_start:horiz_end,:]
             Z[i, h, w, c] = conv_single_step(a_slice_prev,W[:,:,:,c],b[:,:,:,c])
```

4 - Pooling layer

```
for i in range(m):
    for h in range(n_H):
        for w in range(n_W):
            for c in range (n_C):
                vert_start = h*stride
                vert_end = h*stride +f
                horiz_start = w*stride
                horiz_end = w*stride +f
                a_prev_slice        =        (A_prev[i,:,:,c])[vert_start        :
                vert_end,horiz_start:horiz_end]
                if mode == "max":
                    A[i, h, w, c] = np.max(a_prev_slice)
```

```
elif mode == "average":
    A[i, h, w, c] = np.mean(a_prev_slice)
```

2, Convolutional Neural Networks: Application

2.0 - 加载 TensorFlow model

2.1 - Create placeholders

在训练神经网络时需要每次提供一个批量的训练样本，如果每次迭代选取的数据要通过常量表示，那么 TensorFlow 的计算图会非常大。因为每增加一个常量，TensorFlow 都会在计算图中增加一个结点，所以说拥有几百万次迭代的神经网络会拥有极其庞大的计算图，而占位符却可以解决这一点，它只会拥有占位符这一个结点，Placeholder 机制的出现就是为了解决这个问题，我们在编程的时候只需要把数据通过 placeholder 传入 tensorflow 计算图即可。

```
X = tf.placeholder(tf.float32,shape=(None,n_H0,n_W0,n_C0))
Y = tf.placeholder(tf.float32,shape=(None,n_y))
```

1.2 - Initialize parameters

这里使用 xaiver 初始化

```
W1=tf.get_variable("W1",[4,4,3,8],initializer=tf.contrib.layers.xavier_initializer(seed = 0))

W2=tf.get_variable("W2",[2,2,8,16],initializer=tf.contrib.layers.xavier_initializer(seed = 0))
```

1.2 - Forward propagation

```
# CONV2D: stride of 1, padding 'SAME'
```

```
Z1 =tf.nn.conv2d(X,W1,strides = [1,1,1,1],padding = "SAME")
# RELU
A1 = tf.nn.relu(Z1)
# MAXPOOL: window 8x8, sride 8, padding 'SAME'
P1 = tf.nn.max_pool(A1,ksize = [1,8,8,1],strides = [1,8,8,1],padding = "SAME")
# CONV2D: filters W2, stride 1, padding 'SAME'
Z2 = tf.nn.conv2d(P1,W2,strides=[1,1,1,1],padding="SAME")
# RELU
A2 = tf.nn.relu(Z2)
# MAXPOOL: window 4x4, stride 4, padding 'SAME'
P2 = tf.nn.max_pool(A2,ksize= [1,4,4,1],strides=[1,4,4,1],padding="SAME")
# FLATTEN
P2 = tf.contrib.layers.flatten(P2)
# FULLY-CONNECTED without non-linear activation function (not not call softmax).
# 6 neurons in output layer. Hint: one of the arguments should be "activation_fn=None"
Z3 = tf.contrib.layers.fully_connected(P2,6,activation_fn= None)
```

1.4 - Compute cost

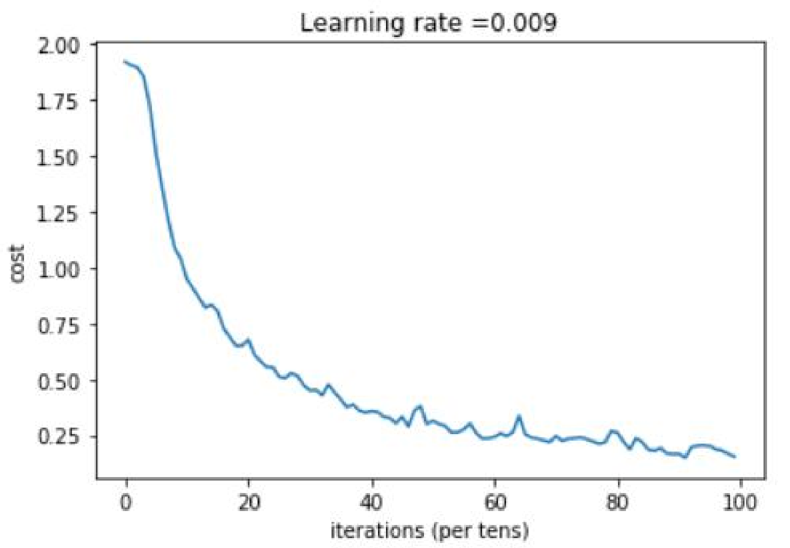
使用交叉熵损失函数

cost=tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits= Z3,labels= Y))

1.4 Model

代码太多略过…………

优化过程，结果：




```
Tensor("Mean_1:0", shape=(), dtype=float32)
Train Accuracy: 0.92314816
Test Accuracy: 0.7916667
```

Week2:Residual Networks

1 - The problem of very deep neural networks

收敛满，overfitting

2 - Building a Residual Network

实现 identity_block 函数

```
  # Second component of main path (≈3 lines)
    X = Conv2D(filters = F2, kernel_size = (f, f), strides= (1,1),padding='same',name = conv_name_base+'2b',kernel_initializer= glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
    X = Activation('relu')(X)
```

```
    # Third component of main path (≈2 lines)
    X = Conv2D(filters = F3, kernel_size = (1, 1), strides = (1,1), padding = 'valid', name = conv_name_base
+ '2c', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)

    # Final step: Add shortcut value to main path, and pass it through a RELU activation (≈2 lines)
    X = Add()([X_shortcut,X])
    X = Activation('relu')(X)
```

2.2 - The convolutional block

```
 # Second component of main path (≈3 lines)

    X = Conv2D(F2,(f,f),strides = (1,1),padding = 'same',name =
conv_name_base+'2b',kernel_initializer = glorot_uniform(seed = 0))(X)

    X = BatchNormalization(axis = 3,name = bn_name_base +'2b')(X)

    X = Activation('relu')(X)



    # Third component of main path (≈2 lines)

    X = Conv2D(F3,(1,1),strides = (1,1),padding = 'valid',name =
conv_name_base+'2c',kernel_initializer = glorot_uniform(seed = 0))(X)

    X = BatchNormalization(axis = 3,name = bn_name_base +'2c')(X)



    ##### SHORTCUT PATH #### (≈2 lines)

    X_shortcut = Conv2D(F3,(1,1),strides = (s,s),padding = 'valid',name =
conv_name_base+'1',kernel_initializer = glorot_uniform(seed = 0))(X_shortcut)

    X_shortcut = BatchNormalization(axis = 3,name = bn_name_base +'1')(X_shortcut)
```

```
    # Final step: Add shortcut value to main path, and pass it through a RELU activation (≈2 lines)
    X = Add()([X_shortcut,X])
    X = Activation('relu')(X)
```

3 - Building your first ResNet model (50 layers)

```
 # Stage 3 (≈4 lines)
    X = convolutional_block(X,f = 3,filters=[128,128,512],stage= 3,block= 'a',s= 2)
    X = identity_block(X,f = 3,filters=[128,128,512],stage = 3,block= 'b')
    X = identity_block(X,f = 3,filters=[128,128,512],stage =3,block= 'c')
    X = identity_block(X,f = 3,filters=[128,128,512],stage = 3,block= 'd')

    # Stage 4 (≈6 lines)
    X = convolutional_block(X,f = 3,filters=[256,256,1024],stage= 4,block= 'a',s= 2)
    X = identity_block(X,f = 3,filters=[256,256,1024],stage = 4,block= 'b')
    X = identity_block(X,f = 3,filters=[256,256,1024],stage = 4,block= 'c')
    X = identity_block(X,f = 3,filters=[256,256,1024],stage = 4,block= 'd')
    X = identity_block(X,f = 3,filters=[256,256,1024],stage = 4,block= 'e')
    X = identity_block(X,f = 3,filters=[256,256,1024],stage = 4,block= 'f')

    # Stage 5 (≈3 lines)
    X = convolutional_block(X,f = 3,filters=[512,512,2048],stage= 5,block= 'a',s= 2)
    X = identity_block(X,f = 3,filters=[512,512,2048],stage = 5,block= 'b')
```

X = identity_block(X,f = 3,filters=[512,512,2048],stage = 5,block= 'c')

AVGPOOL (≈1 line). Use "X = AveragePooling2D(...)(X)"

X = AveragePooling2D(pool_size=(2,2),padding= 'same')(X)

```
120/120 [==============================] - 2s 16ms/step
Loss = 1.977328856786092
Test Accuracy = 0.16666666666666666
```

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1 - 3 道问答题：

在卷积向前传播的过程中，刚开始的时候没有考虑步长，即默认了 stride 是 1，导致测试的时候一直出错。原因是有很多中可以改变步长的写法，但是与框架相符合的写法只有一中。