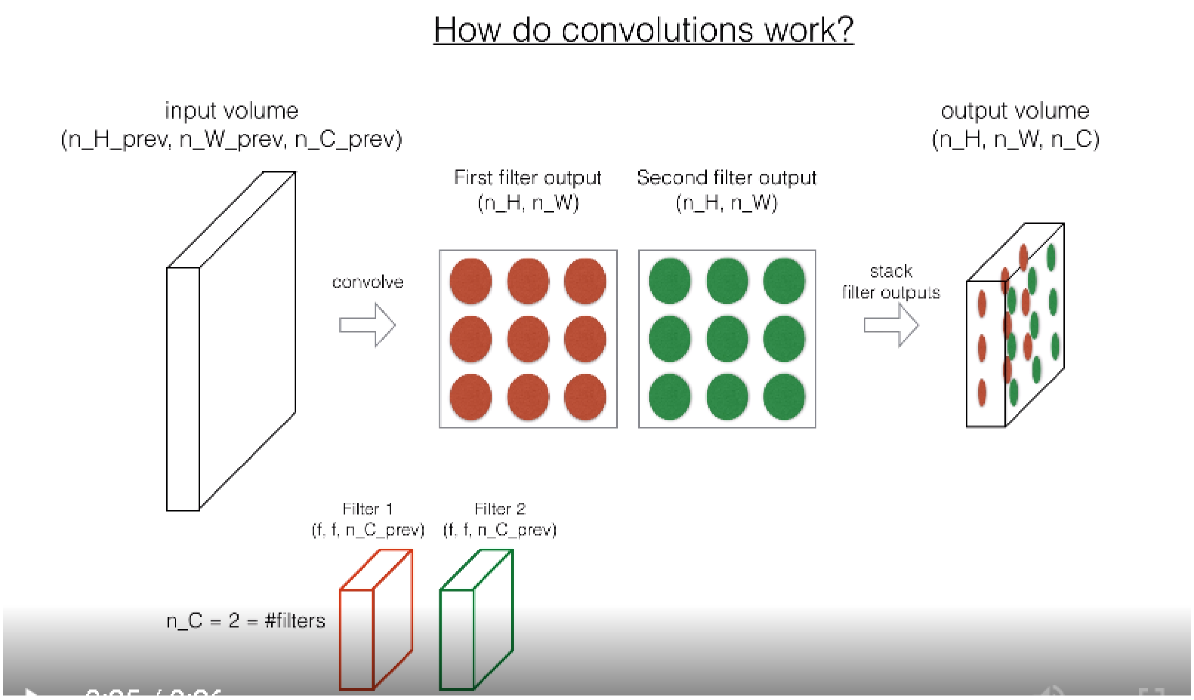
CNN&Tesorflow1&Resnet实验报告

201900161098 马田慧 智能19

CNN&Tesorflow1

每个卷积核和原来的有相同的深度channel，拿一个三维举例，每次卷积形成一个二维的平面，然后有几个卷积核就形成几个平面，最后这些平面叠加起来就是最后输出的深度：



卷积过程中的参数：

padding：边框；stride：步长；padding-type：边框填充的种类；

`np.pad()`：

`np.pad(X,((0,0),(pad,pad),(pad,pad),(0,0)),constant_values=((0,0),(0,0),(0,0),(0,0)))` 表示四个维度是否填充pad，后面表示四个维度的填充值；

计算每个卷积位置：

```
vert_start = h*stride
vert_end = h*stride+f
horiz_start = w*stride
horiz_end = w*stride+f
```

卷积也有对应的池化层；

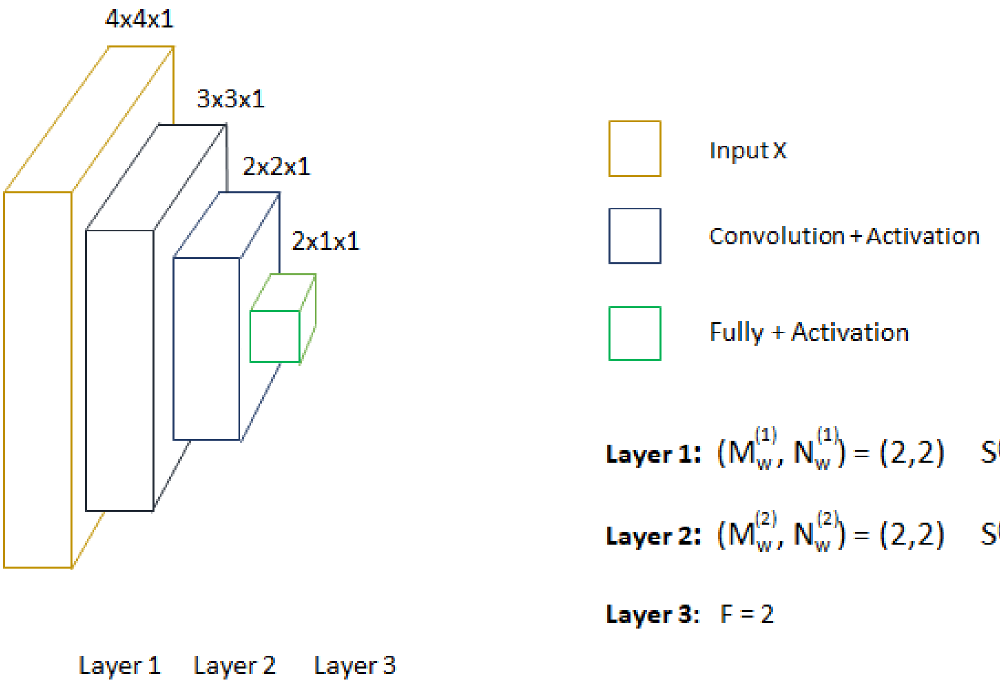
```
    for i in range(m):                          # loop over the training examples
        for h in range(n_H):                     # loop on the vertical axis of
the output volume
            for w in range(n_W):                 # loop on the horizontal axis of
the output volume
                for c in range (n_C):            # loop over the channels of the
output volume
```

```
                    # Find the corners of the current "slice" (≈4 lines)
                    vert_start = h*stride
                    vert_end = h*stride+f
                    horiz_start = w*stride
                    horiz_end = w*stride+f

                    # Use the corners to define the current slice on the ith
training example of A_prev, channel c. (≈1 line)
                    a_prev_slice =
A_prev[i,vert_start:vert_end,horiz_start:horiz_end,c]#i --all

                    # Compute the pooling operation on the slice. Use an if
statment to differentiate the modes. Use np.max/np.mean.
                    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)#average
```

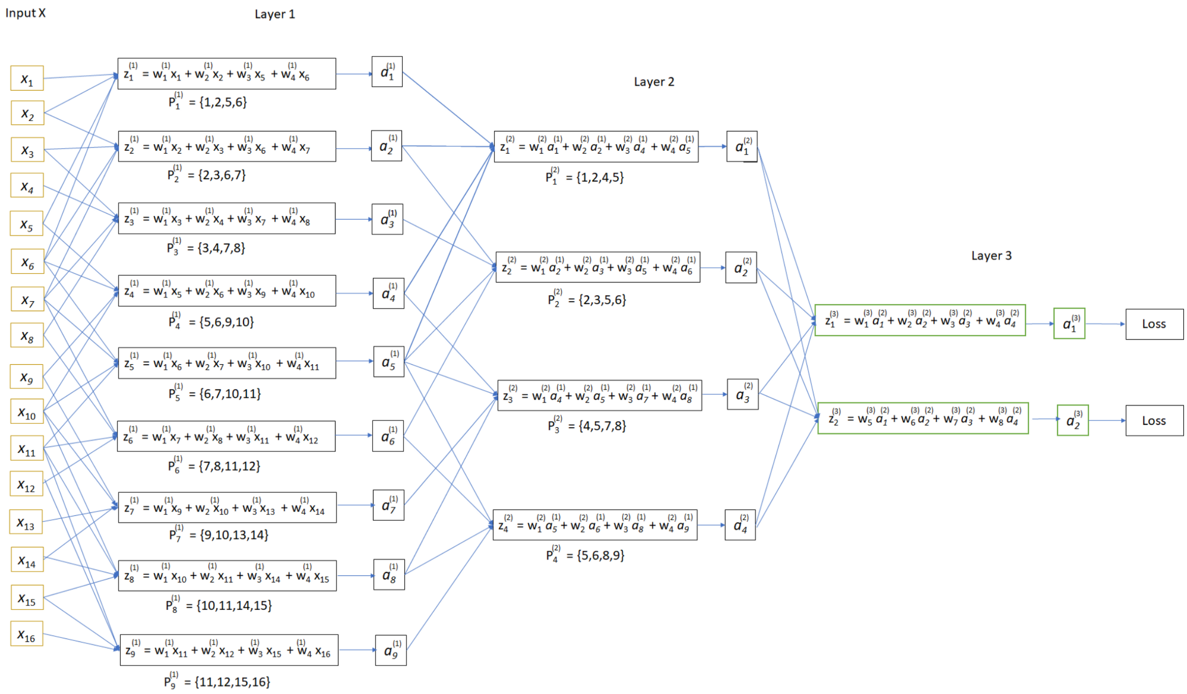
详解CNN反向传播：https://towardsdatascience.com/backpropagation-in-fully-convolutional-networks-fcns-1a13b75fb56a



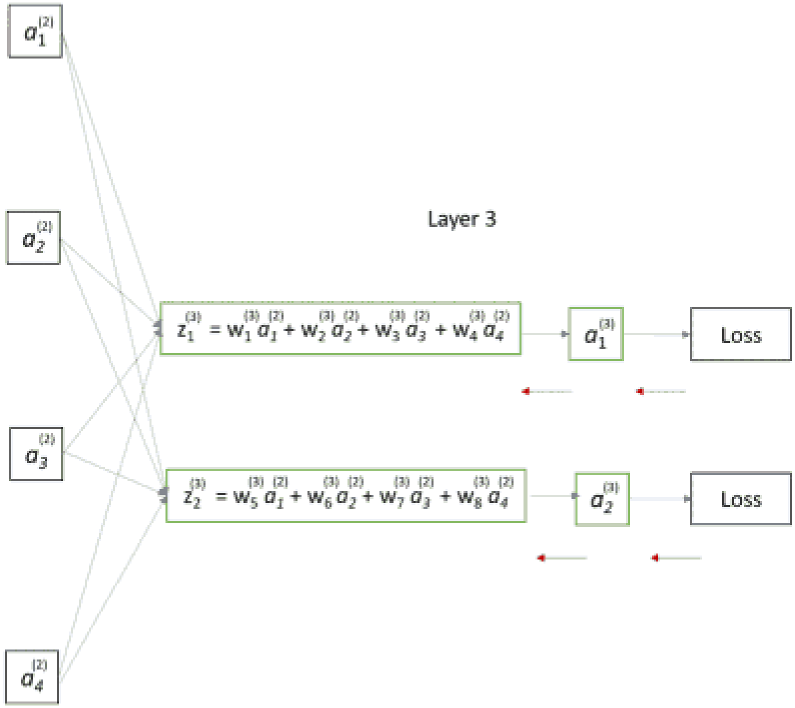
Layer 1: $(M_w^{(1)}, N_w^{(1)}) = (2,2)$ $S^{(1)} = [1,1]$

Layer 2: $(M_w^{(2)}, N_w^{(2)}) = (2,2)$ $S^{(2)} = [1,1]$

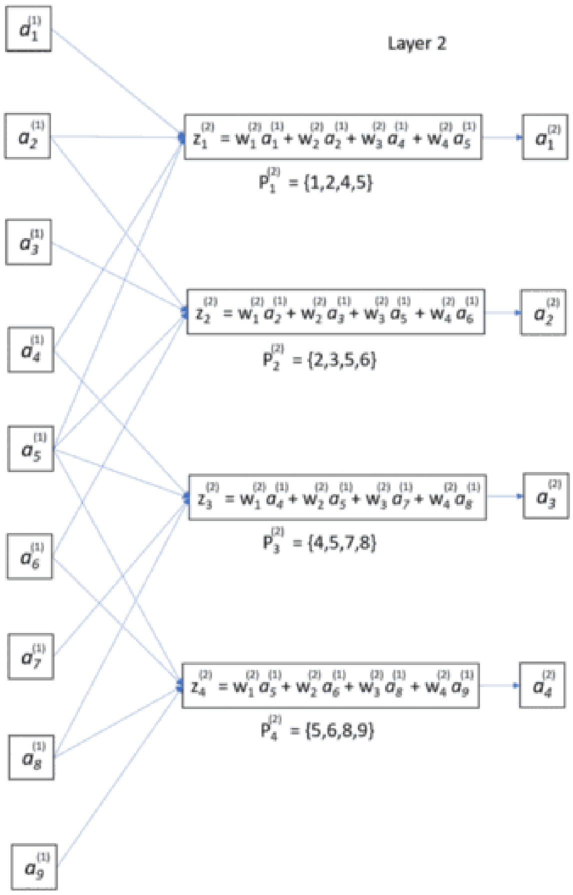
Layer 3: $F = 2$



Input X
Layer 1
Layer 2
Layer 3
P1(1) = {1,2,5,6}
P2(1) = {2,3,6,7}
P3(1) = {3,4,7,8}
P4(1) = {5,6,9,10}
P5(1) = {6,7,10,11}
P6(1) = {7,8,11,12}
P7(1) = {9,10,13,14}
P8(1) = {10,11,14,15}
P9(1) = {11,12,15,16}
P1(2) = {1,2,4,5}
P2(2) = {2,3,5,6}
P3(2) = {4,5,7,8}
P4(2) = {5,6,8,9}
Loss
Loss



Layer 3
Loss
Loss



具体公式cnn反向传播：

$$\frac{\partial L}{\partial a_1^{(1)}} = \frac{\partial L}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}}$$

$$\frac{\partial L}{\partial a_2^{(1)}} = \frac{\partial L}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_2^{(1)}} + \frac{\partial L}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial a_2^{(1)}}$$

$$\frac{\partial L}{\partial a_3^{(1)}} = \frac{\partial L}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial a_3^{(1)}}$$

$$\frac{\partial L}{\partial a_4^{(1)}} = \frac{\partial L}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_4^{(1)}} + \frac{\partial L}{\partial z_3^{(2)}} \frac{\partial z_3^{(2)}}{\partial a_4^{(1)}}$$

$$\frac{\partial L}{\partial a_5^{(1)}} = \frac{\partial L}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_2^{(1)}} + \frac{\partial L}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial a_5^{(1)}} + \frac{\partial L}{\partial z_3^{(2)}} \frac{\partial z_3^{(2)}}{\partial a_5^{(1)}} + \frac{\partial L}{\partial z_3^{(2)}} \frac{\partial z_4^{(2)}}{\partial a_4^{(1)}}$$

$$\frac{\partial L}{\partial a_6^{(1)}} = \frac{\partial L}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial a_6^{(1)}} + \frac{\partial L}{\partial z_4^{(2)}} \frac{\partial z_4^{(2)}}{\partial a_6^{(1)}}$$

$$\frac{\partial L}{\partial a_8^{(1)}} = \frac{\partial L}{\partial z_3^{(2)}} \frac{\partial z_3^{(2)}}{\partial a_8^{(1)}} + \frac{\partial L}{\partial z_4^{(2)}} \frac{\partial z_4^{(2)}}{\partial a_8^{(1)}}$$

$$\frac{\partial L}{\partial a_9^{(1)}} = \frac{\partial L}{\partial z_4^{(2)}} \frac{\partial z_4^{(2)}}{\partial a_9^{(1)}}$$

注意其中的**步长；**

Tesorflow1

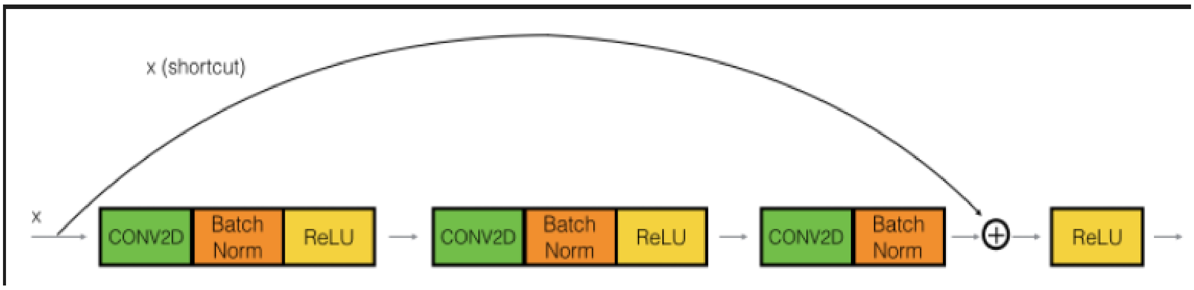
github 的 tensorflow example；

raw创建过程：正向传播；session启动；计算损失；建立模型；常规通过examplr了解tensorflow；

Resnet

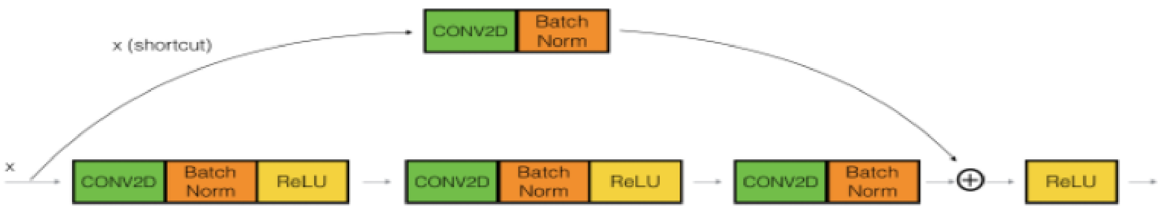
基本结构为通过skipconnection防止梯度小数或爆炸：

使用block概念：

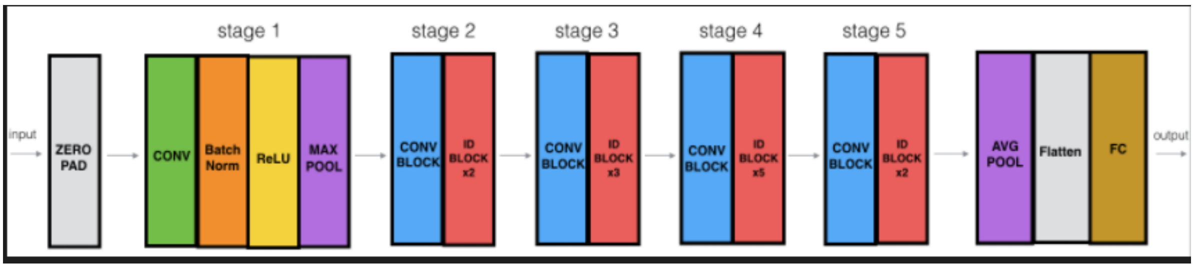


运算时使用keras框架计算：

对其中每一层进行命名；多种结构的block；

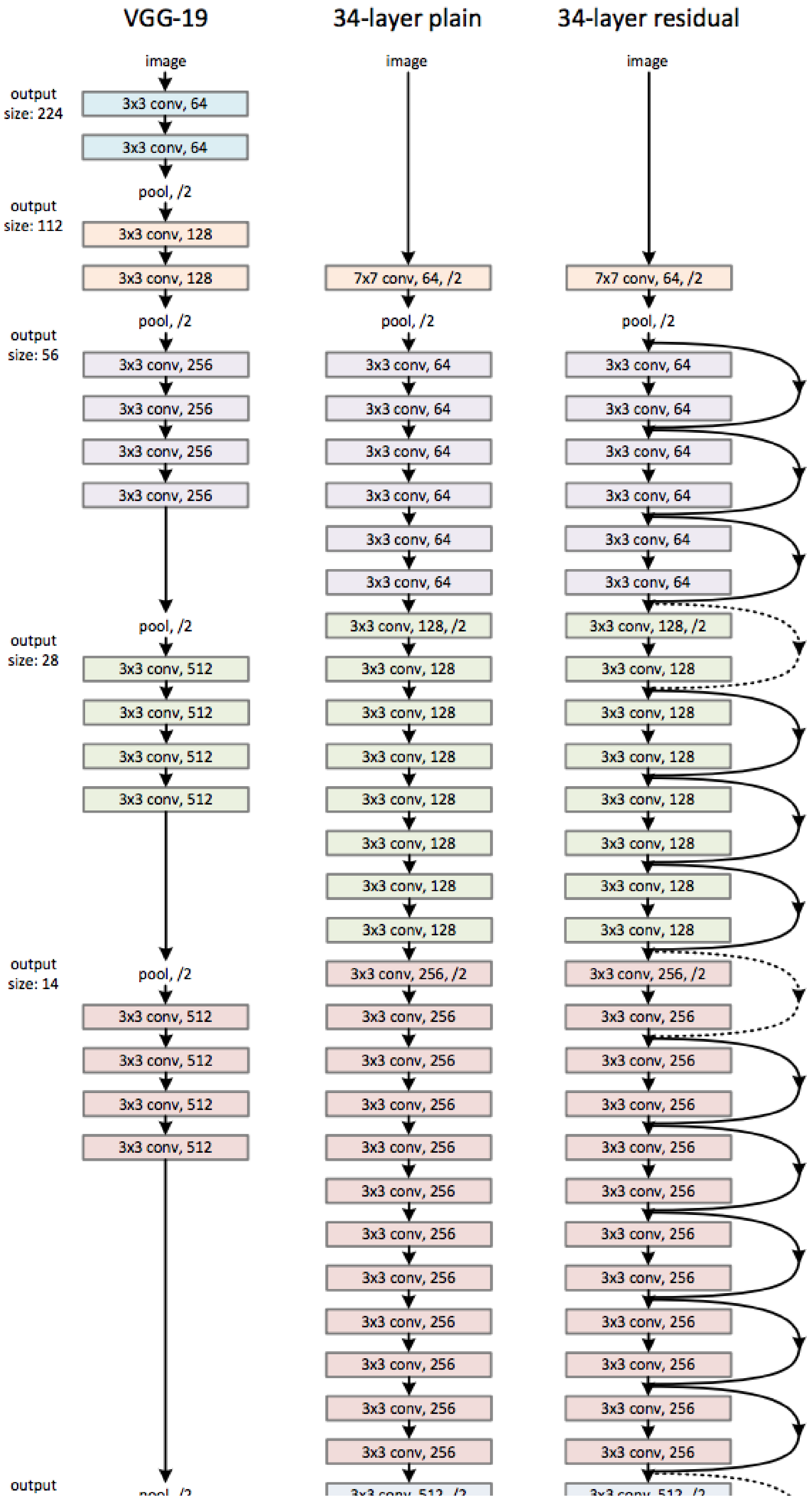


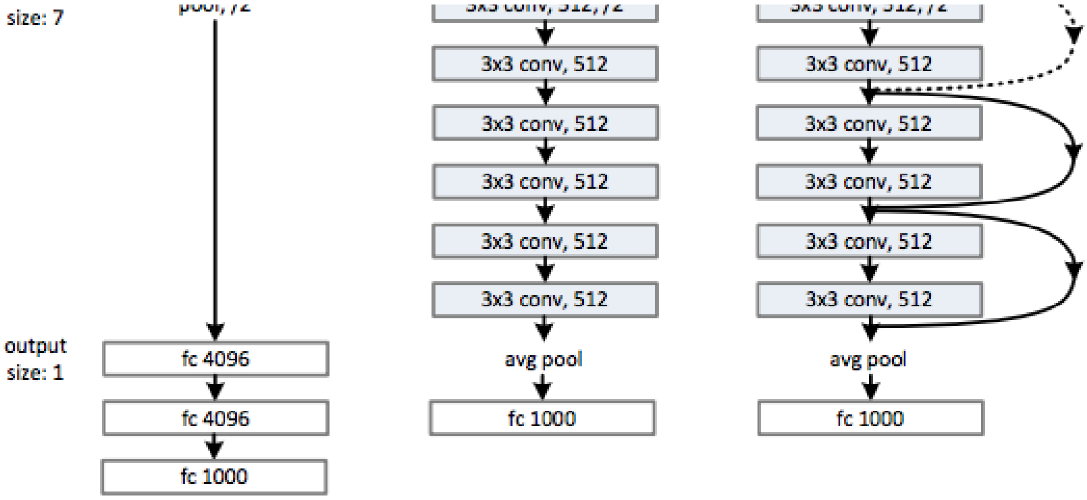
最后进行叠加，加上常规pad，bn层，avgpool以及展开操作的flatten，fc



实验细节包括每一层的维度的确定是个难点；以及tf适用 `float-->tf.float32`，`X+X_short-->Add()[X,X_short]`；

最后输出Restnet图像‘；图存维度；


VGG-19
34-layer plain
34-layer residual
image
output
size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output
size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output
size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output
size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
output
size: 14
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
output
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256



实验结论

- 进一步了解cnn细节
- 通过example学习tensorflow
- 配置anaconda环境
- 了解残差网络原理