ResNet Notes

1. Motivation

- 随着网络的加深，出现了训练准确率下降的现象，我们可以确定这不是由于过拟合造成的(过拟合的情况训练集应该准确率很高)；所以作者针对这个问题提出了一种全新的网络，叫深度残差网络，它允许网络尽可能的加深，其中引入了全新的结构，如图所示：

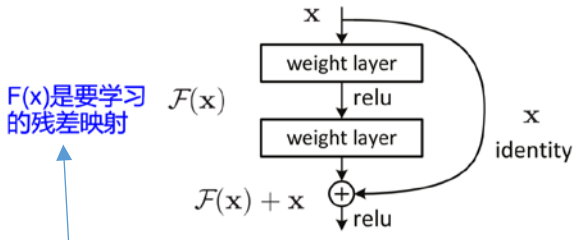

Figure 2. Residual learning: a building block.

- **identity mapping**

  恒等映射，指的就是图中“弯弯的曲线”（x）

- **residual mapping**

  指的就是除了“弯弯的曲线”那部分（F(x)），residual mapping 指的是“差”

2. 为什么 ResNet 可以解决神经网络的退化问题

- **实验证明**
- **理论证明**

  首先对于一个堆积层结构（几层堆积而成），当输入为 x 时其学习到的特征记为 H(x)，现在我们希望其可以学习到残差 F(x)=H(x)-x，这样其实原始的学习特征是 F(x)+x。

  为什么残差学习相对更容易，从直观上看残差学习需要学习的内容少，因为残差一般会比较小，学习难度小。

$$\frac{\partial loss}{\partial x_l} = \frac{\partial loss}{\partial x_L} \cdot \frac{\partial x_L}{\partial x_l} = \frac{\partial loss}{\partial x_L} \cdot \left(1 + \frac{\partial}{\partial x_L} \sum_{i=l}^{L-1} F(x_i, W_i)\right)$$

  小括号中的 1 表明短路机制可以无损地传播梯度，而另外一项残差梯度则需要经过带有 weights 的层，梯度不是直接传递过来的。残差梯度不会那么巧全为-1，而且就算其比较小，有 1 的存在也不会导致梯度消失。所以残差学习会更容易。要注意上面的推导并不是严格的证明。

3. ResNet 网络结构

- **Details**

  ResNet 网络是参考了 VGG19 网络，在其基础上进行了修改，并通过短路机制加入了残差单元，变化主要体现在 ResNet 直接使用 stride=2 的卷积做下采样，并且用 global average pool 层替换了全连接层。ResNet 的

一个重要设计原则是：当 feature map 大小降低一半时，feature map 的数量增加一倍，这保持了网络层的复杂度。

- **两种残差结构**

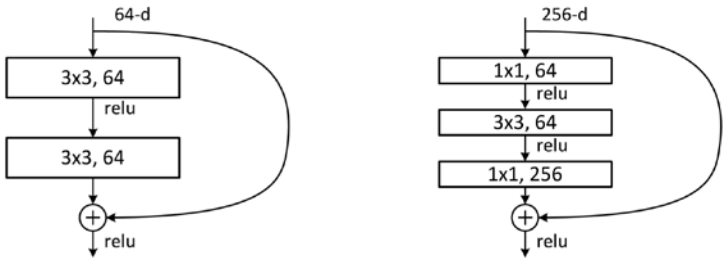

Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

这两种结构分别针对 ResNet34（左图）和 ResNet50/101/152（右图），一般称整个结构为一个"building block"。其中右图又称为"bottleneck design"，目的一目了然，就是为了降低参数，第一个 1x1 的卷积把 256 维 channel 降到 64 维，然后在最后通过 1x1 卷积恢复。

- **F(x) + x，两个 map 相加的问题**

对于短路连接，当输入和输出维度一致时，可以直接将输入加到输出上。当维度不一致时（channel 维度增加一倍，width、height 维度减小一倍），这就不能直接相加。有两种策略：（1）采用 zero-padding 增加维度，此时一般要先做一个降采样，可以采用 stride=2 的 avg pooling，这样不会增加参数；（2）采用新的映射（projection shortcut），一般采用 1x1 的卷积，这样会增加参数，也会增加计算量。

```
with tf.variable_scope('sub_add'):
    if in_filter != out_filter:
        # 使用平均池化给orig_x降采样，如果stride=1，origin_x没有变化
        orig_x = tf.nn.avg_pool(orig_x, stride, stride, 'VALID')

        # // 取整除法
        # tf.pad第一个参数是要填充的tensor，第二个参数规定这个tensor每个维度如何填充
        # 举例来说，tensor的维度为4,[batch, height, width, channel]
        # 如果要在channel维度上进行填充，就把其他三个维度的参数都设置成[0,0]，表示不填充
        # 填充参数的[0,0]第一个0表示在前面填充0行，第二个0表示在后面填充0行
        orig_x = tf.pad(
            orig_x, [[0, 0], [0, 0], [0, 0],
                     [(out_filter - in_filter) // 2, (out_filter - in_filter) // 2]])
    x += orig_x
```