ResNet总结

1. ResNet背景

1. **ResNet网络解决了深度CNN模型训练难的问题**

VGG网络试着探寻了一下深度学习网络的深度究竟可以深到何种程度还可以持续提高分类的准确率。对于传统的深度学习网络，我们普遍认为网络深度越深（参数越多）非线性的表达能力越强，该网络所能学习到的东西就越多。

但是传统CNN网络结构随着测概述加深到一定程度后，越深的网络反而效果越差, 过深的网络使分类的准确率下降：

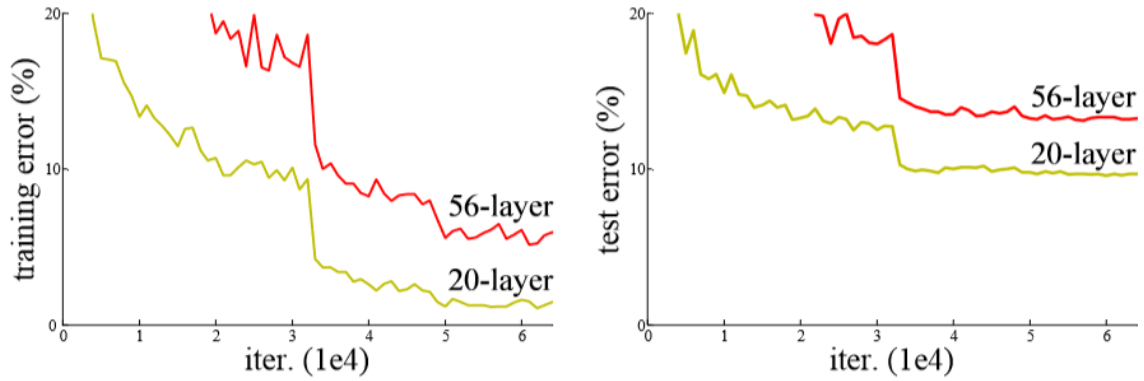


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

2. **网络退化问题**

上述问题并不是过拟合导致的，因为如上图所示，56层网络比20层网络在训练数据上的损失还要大，而过拟合是模型在训练数据上的损失不断减小，在测试数据上的损失先减小再增大，这种问题称之为**网络退化问题（Degradation problem）**。

> The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution by construction to the deeper model: the added layers are identity mapping, and the other layers are copied from the learned shallower model. The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart. But experiments show that our current solvers on hand are

> unable to find solutions that are comparably good or better than the constructed solution (or unable to do so in feasible time).
>
> 考虑一个训练好的网络结构，如果加深层数的时候，不是单纯的堆叠更多的层，而是堆上去一层使得堆叠后的输出和堆叠前的输出相同，也就是恒等映射/单位映射（identity mapping），然后再继续训练。这种情况下，按理说训练得到的结果不应该更差，因为在训练开始之前已经将加层之前的水平作为初始了，然而实验结果结果表明在网络层数达到一定的深度之后，结果会变差，这就是退化问题。这里至少说明传统的多层网络结构的非线性表达很难去表示恒等映射（identity mapping），或者说你不得不承认目前的训练方法或许有点问题，才使得深层网络很难去找到一个好的参数去表示恒等映射（identity mapping）。

2. 深度残差网络结构学习（Deep Residual Learning)

1. **残差单元**

对于一个堆积层结构（几个基本层堆积而成），当输入为x时，其学习到的特征记为$H(x)$，现在希望网络可以学习到残差$F(x)=H(x)-x$，这样其实原始的学习特征是$H(x)$。之所以是这样是因为残差学习相比原始特征直接学习更容易。当残差$F(x)=0$时，此时对基层仅做了恒等映射，至少网络性能不会下降，实际上残差不会为0，这也会是的对基层在输入特征基础上学习到新的特征，从而有更好的性能。残差单元如图所示：

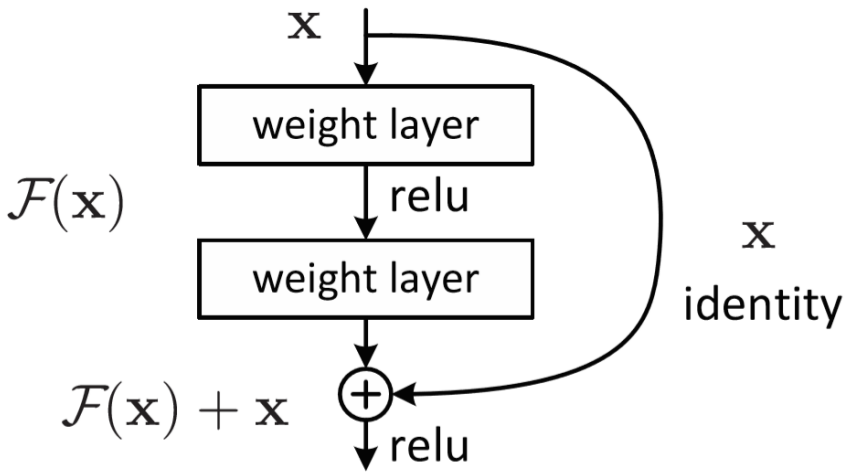


Figure 2. Residual learning: a building block.

残差单元是一种**短路连接（shortcut connection）**。

从数学角度分析为何残差学习相对更容易：

首先残差单元可以表示为：

$$y_l = h(x_l) + F(x_l, W_l)$$
$$x_{l+1} = f(y_1)$$

其中x_l和x_{l+1}分别表示第l个残差单元的输入和输出。每一个残差单元一般包含多层结构，F是残差函数，表示学习到的残差，而$h(x_l) = x_l$表示恒等映射，f是$ReLu$激活函数。可求得从浅层l到深层L的学习特征：

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i, W_i)$$

对于传统的CNN，直接堆叠的网络相当于一层层地做一一仿射变换-非线性变换，而仿射变换这一步主要是矩阵乘法。所以总体来说直接堆叠的网络相当于是乘法性质的计算。

而在ResNet中，相对于直接堆叠的网络，因为shortcut的出现，计算的性质从乘法变成了加法。计算变的更加稳定。当然这些是从前向计算的角度，从后向传播的角度，如果代价函数用$Loss$表示，则有:

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

即：更高层的梯度成分$\frac{\partial loss}{\partial x_L}$可以直接传过去。小括号中的1表明短路机制（shortcut）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。

这样一来梯度的衰减得到进一步抑制，并且加法的计算让训练的稳定性和容易性也得到了提高。所以可训练的网络的层数也大大增加了。

2. **恒等映射/单位映射（identity mapping)**

残差单元通过identity mapping的引入在输入和输出之间建立了一条直接的关联通道，从而使得有参数层集中学习输入和输出之间的残差。

一般我们用$F(X, W_i)$来表示残差映射，则输出即为：$Y = F(X, W_i) + X$。当输入和输出通道数相同时，可以直接使用X相加。当输入和输出的通道数不同时，需要考虑建立一种有效的identity mapping从而可以使得处理后的输入X与输出Y的通道数目相同，即：

$$Y = F(X, W_i) + W_s X$$

当X与Y通道数目不同时，作者尝试了两种 identity mapping 的方式。一种即简单地将X相对Y缺失的通道直接补零从而使其能够相对齐的方式，另一种则是通过使用$1x1$的$conv$来表示W_s映射从而使得最终输入与输出的通道达到一致的方式。

3. **瓶颈（BottleNeck）模块**

如下图所示，左图是原始的常规模块（Residual block），实际使用的时候，残差模块和Inception模块一样希望可以降低计算开销。因此提出**瓶颈（BottleNeck）**模块，思路和Inception一样，通过使用$1 \times 1conv$来缩减或扩张feature map维度从而使得我们的$3 \times 3conv$的filter数目不受上一层输入的影响，同样其输出也不会影响到下一层module。

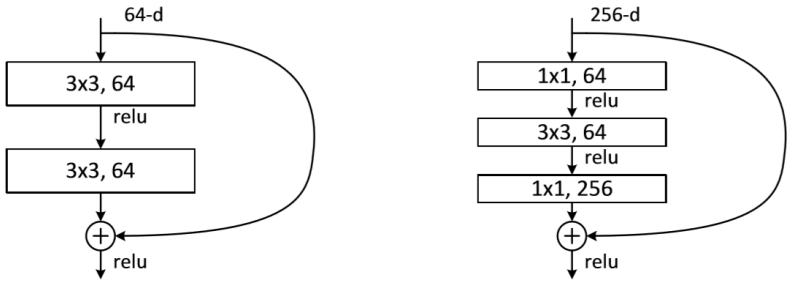


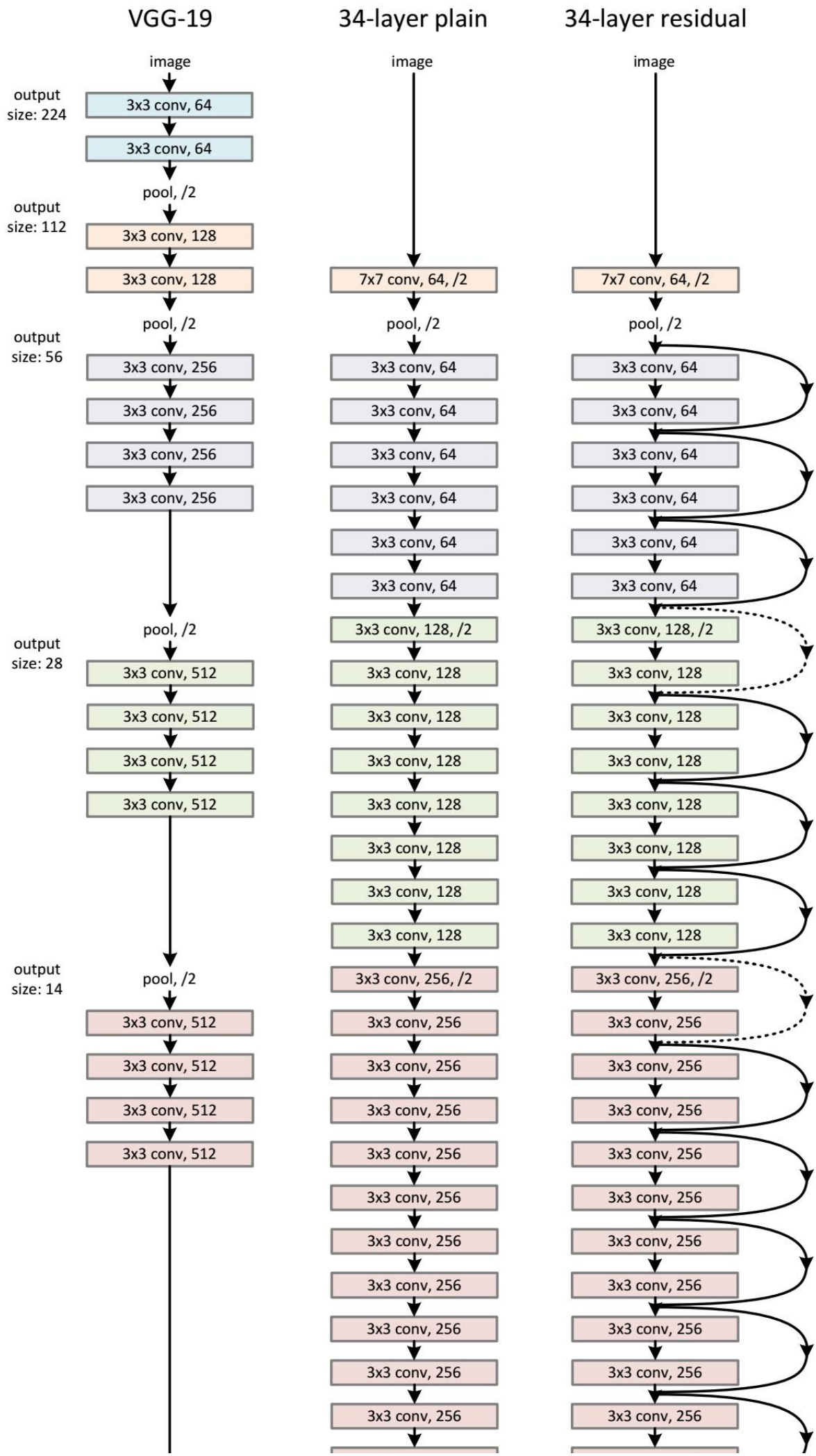
Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

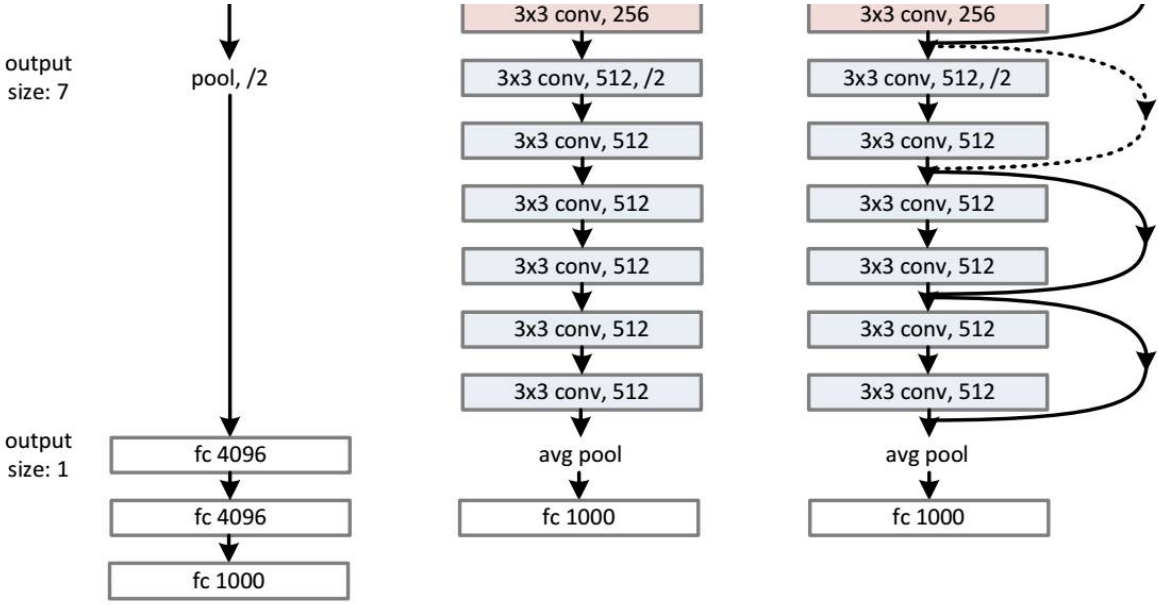
4. **RestNet结构**

ResNet网络参考了VGG19的网络结构，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直接使用了`stride=2`的卷积做下采样，并用`global average pool`层替换了全连接层。

ResNet的其中一个重要的设计原则是：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。ResNet相比普通网络每两层间增加了短路机制，这就形成了残差学习。

ResNet网络结构：


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
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 256, /2

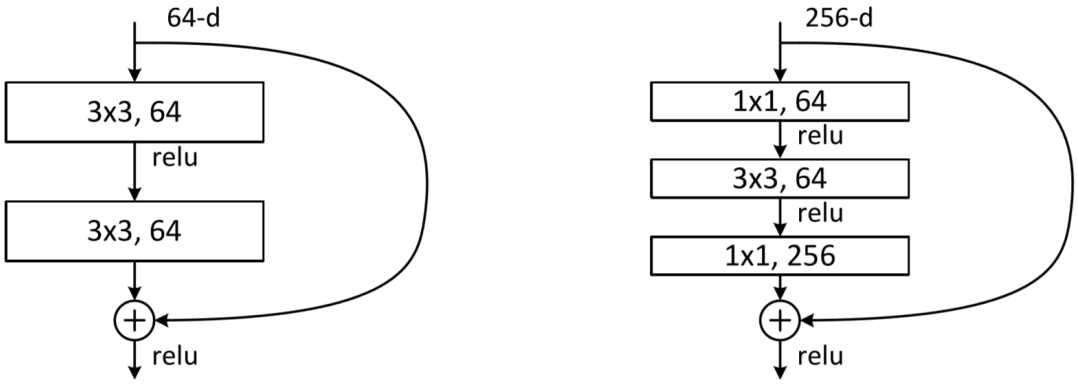


不同深度的ResNet：

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

从表中可以看到，对于18-layer和34-layer的ResNet，其进行的两层间的残差学习，当网络更深时，其进行的是三层间的残差学习，三层卷积核分别是1x1，3x3和1x1，一个值得注意的是隐含层的feature map数量是比较小的，并且是输出feature map数量的1/4。

ResNet使用两种残差单元：



左图是浅层网络，右图是深层网络。对于shortcut connection，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，可以采用两种策略：

（1）采用zero-padding增加维度

此时一般先做一个down swap，可以采用stride=2的pooling

（2）采用新的映射（projection shortcut）

一般采用1 × 1的卷积，这样会增加参数，也会增加计算量。短路连接除了直接使用恒等映　　射，同样都可以采用projection shortcut。

论文中18-layer和34-layer的网络效果：

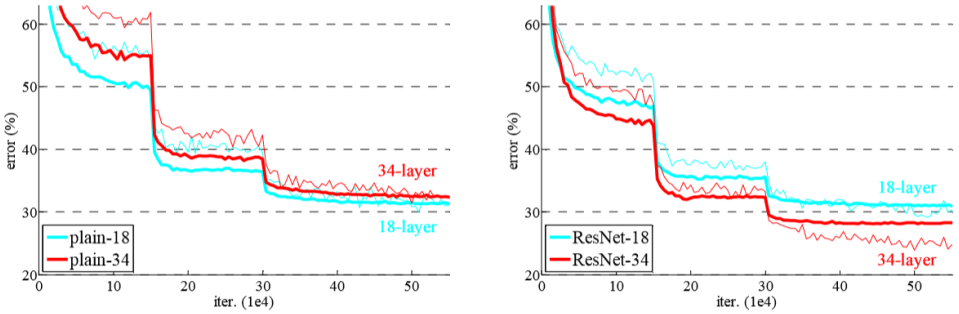


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

论文中ResNet与其他网络对比结果：

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

3. ResNet进一步改进

在2015年的ILSVRC比赛获得第一之后，何恺明对残差网络进行了改进，主要是把ReLu给移动到了conv之前，相应的shortcut不在经过ReLu，相当于输入输出直连。并且论文中对ReLu，BN和卷积层的顺序做了实验，最后确定了改进后的残差网络基本构造模块，如下图8所示，因为对于每个单元，激活函数放在了仿射变换之前，所以论文叫做预激活残差单元（pre-activation residual unit）。作者推荐在短路连接（shortcut）采用恒等映射（identity mapping）。

改进后的残差单元及效果：