简介

ResNet 原始论文地址《Deep Residual Learning for Image Recognition》，该论文发表于2015年，作者均来自于 Microsoft Research。

越深的网络越难训练，因此论文提出了残差学习框架（ residual learning framework）来解决这个问题，使得较深的网络也很容易训练。而且残差学习框架更容易优化，通过增加深度已获得更好的准确率（accuracy）。ResNet 网络深度达到了 152 层，是 VGG 深度的 8 倍，但结构并没有变得更复杂。

论文主要用于解决网络深度的问题，同时又解决深度网络中梯度消失和爆炸的问题。【但从《CSDN - 对ResNet的理解》文章的分析来看，似乎并没有解决这个问题】

随着深度加深，精确度也达到了饱和。【精度达到了饱和，我的理解是随着网路的加深，精确度一直保持不变，不再有所提高 】。这种退化也不是由于过拟合（overfitting）造成的，添加更多的层导致更高的训练误差（training error）

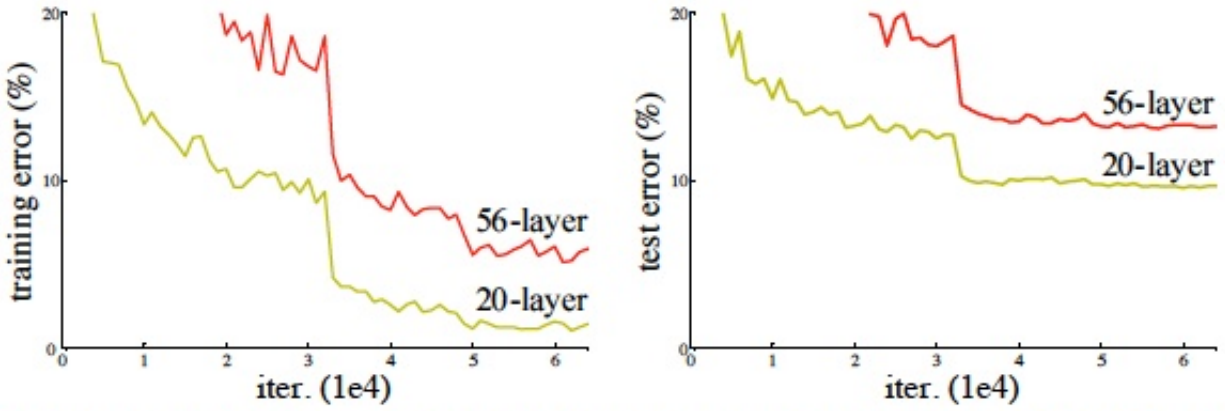


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.

论文提出了 deep residual learning framework 来解决上面的问题。残差学习的基本结构如下图：

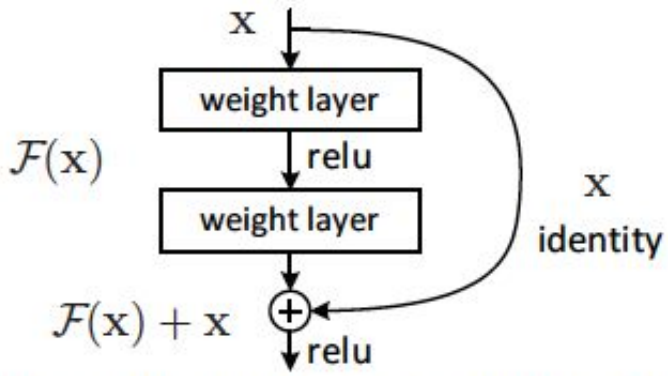
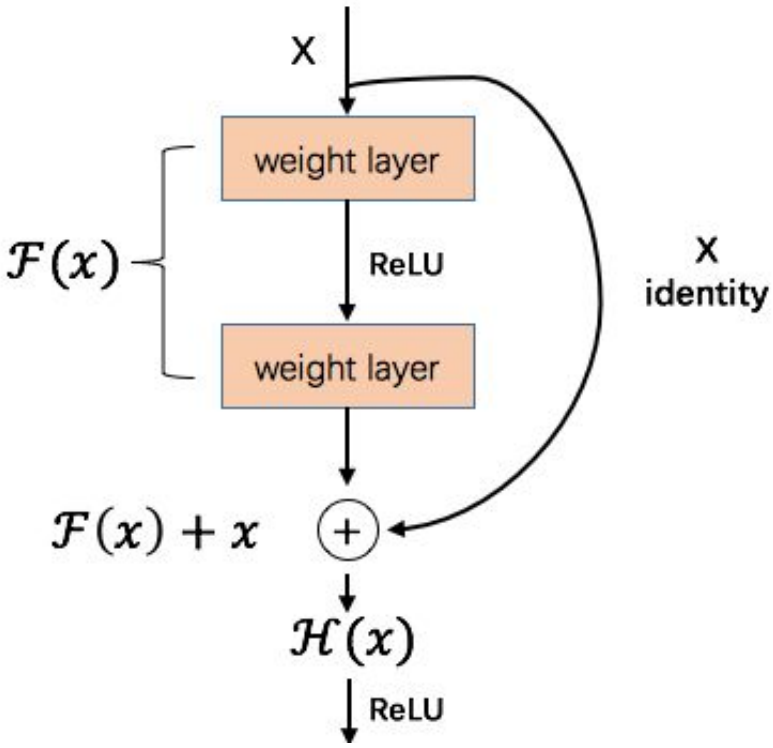


Figure 2. Residual learning: a building block.



图中的曲线部分就是恒等映射（identity mapping）非曲线部分就是残差映射（residual mapping），$\mathcal{F}(x)$ 既是残差。

新的结构可以使用 SGD 和反向传播执行 end-to-end 的训练，使用现有的公共包（caffe）就可以实现。在 ImageNet 上实现，证明了：

- ResNet 更容易优化，相比于传统的扁平化网络结构
- ResNet 可以轻松的从网络深度的增加，获取更好的 accuracy
- 可以泛化到不同的任务中，如 ImageNet detection、 localization、COCO detection、segmentation 都获得了第一名。 证明了 残差学习原则的可泛化性是极强的，可以被用于视觉与非视觉的问题中。

深度残差学习（Deep Residual Learning）

残差模块定义为：

$$y = \mathcal{F}(x, \{W_i\}) + x \tag{1}$$

x 和 y 表示输入输出向量， $\mathcal{F}(x, \{W_i\})$ 表示需要学习的残差映射，即残差，以上图为例，一种有两层，那么残差学习为 $\mathcal{F} = W_2\sigma(W_1x)$ 其中 σ 表示 ReLU 函数， 偏差（bias）被省略了，为了计算的方便。公式（1）既没有引入额外的参数也没有复杂的计算。

$\mathcal{F} + x$ 通过捷径连接起来，执行逐元素相加，因此 $\mathcal{F}$ 和 x 的大小应该是相同的。若两者不相同，可以通过 W_s 线性投影来完成维度的匹配，W_s 其实就是卷积操作，通过它可以调整 x 通道的个数：

$$y = \mathcal{F}(x, \{W_i\}) + W_s x \tag{2}$$

在 ResNet 结构中， $\mathcal{F}$ 通常含有两到三层，也可以含有更多的层。但当其中只含有一层的时候，他就和线性层非常的相似：

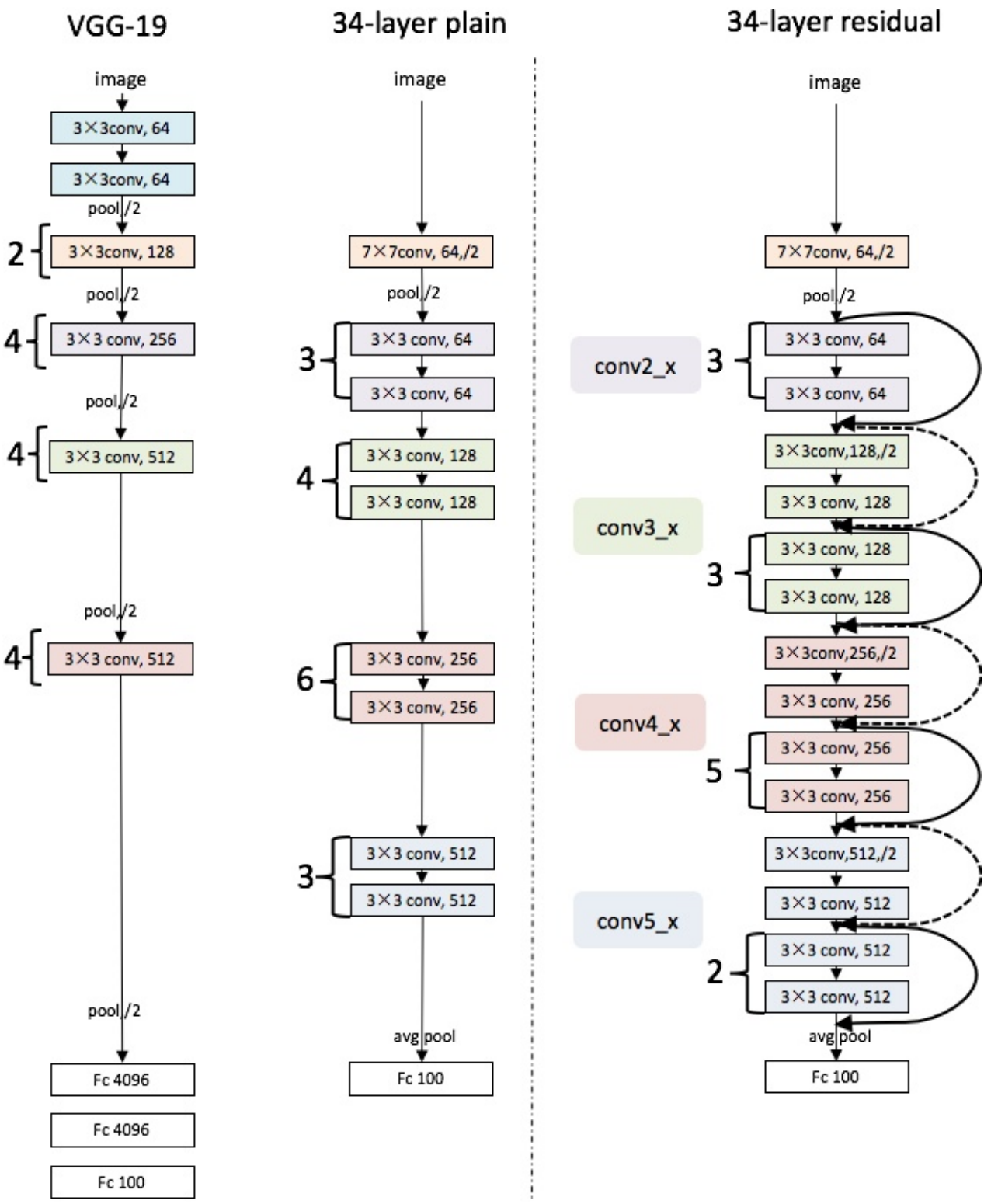
$$y = W_i x + x$$

使用一层的话，论文中提到并没有观察到优势

网络架构

ResNet 使用了更少的 filter 和更低的复杂度，相比于 VGG。ResNet 34 层的 baseline 有 36 亿的 FLOPs（multiply-adds），只是 VGG 的18%，VGG 含有196亿 FLOPs。

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



上图中的中括号表示括号中的结构重复的次数，其中的每一个结构都是公式（1）在输入与输出维度相等的情况下。当维度不同的时候就需要进行升维操作，上面的34-layer residual 虚线部分就是进行升维操作，对应着公式（2）。

升维有两种操作方式，一种是通过补零的方式；还有一种是通过映射来匹配维度，即通过公式（2）来做，主要是通过 1×1 的卷积实现。

操作

实现

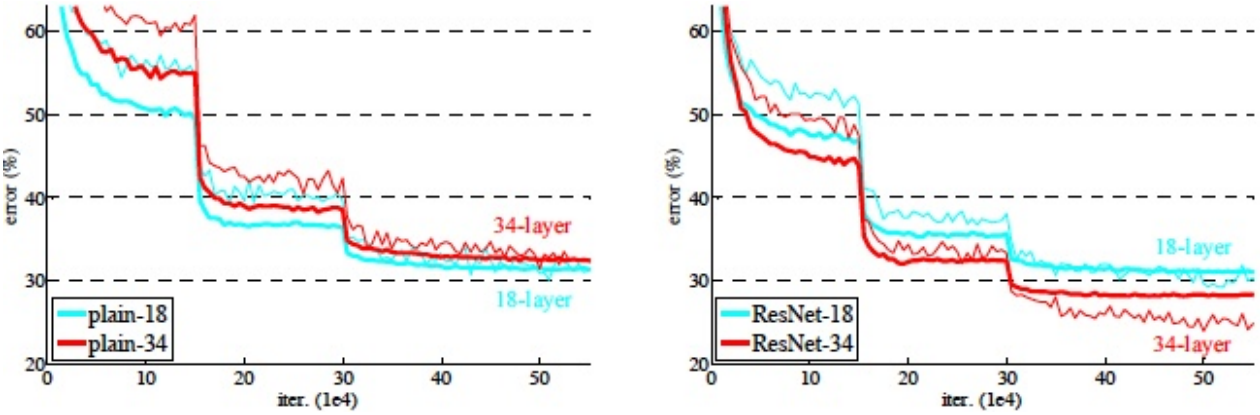
和 AlexNet与 VGG 类似，都是将图片的最小边缩放到一个尺寸，这里是从[256, 480] 范围进行采样。然后截取 224×224 的区域，让截取区域的每个像素减去均值。使用标准颜色增强。在卷积层与激活层中间加入 BN（Batch Normalization）。初始化权重，并从头开始训练。使用批次大小为256 的 SGD。学习速率从 0.1 开始，每次误差停止下降时就让学习速率下降10倍。模型迭代次数达到 60×10^4 次。动量为 0.9，权重衰减 0.0001。不使用 Dropout。

实验

首先评估18层与34层的扁平结构，34层的比18层验证误差要高。对比结果如下表：

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

训练过程如下图：



上图中细的线表示训练误差，粗的线表示居中剪切的验证误差。左图表示扁平的网络结构，右侧表示结构相同但是使用了残差连接的结构。在结构上残差结构并没有额外的增加参数与扁平结构相比。

左图的结构是很难优化的，它并不是由于梯度消失引起的。在训练时向前传播与向后传播的信号都没有消失。

而在 ResNet 中结果刚好与扁平结构相反，越深的网络得到的结果越好。而且34层的训练误差更低，结果也可以更好的泛化到验证集。而且可以看出，在18层结构中，扁平结构和 ResNet 有相似的精确度，但 ResNet 收敛的更快。当网络的深度不超过18层，当前的 SGD 方法可以找到很好的解在扁平结构中。

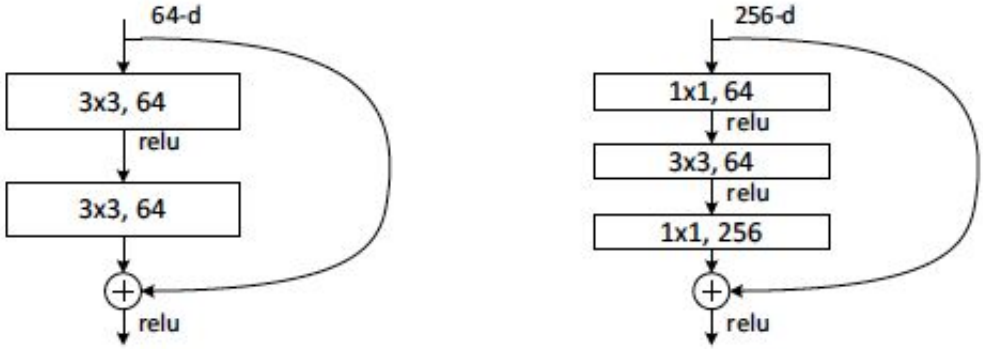
Identity vs. Projection Shortcuts

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

上图中探讨了ResNet-34 的三中 shortcut 的方式。A 使用 zero-padding 方式，B 使用投影连接方式来升维，其他使用 identity 方式 shortcut ，C 全部使用投影的方式 shortcut。

从上图可以看到三种方式均好于扁平的结构。B 方式比 A 方式好，可能是由于 zero-padding的维度没有残差学习，C 比 B 也好好一点，可能是由于引入了额外的参数。从 A/B/C 三种方式可以看到，projection shortcut 并不是解决退化问题的最好方式。因此论文中不在使用 C 的方式，而是使用 identity shortcut 的方式。

更深架构的瓶颈（ Deeper Bottleneck Architectures）



在 ResNet-50/101/152 的结构中使用了右侧的基础结构，使用 3 层堆叠来替代 2 层的堆叠，右侧的瓶装结构本质上就是为了降维。首先使用一个1×1的卷积核对channel 从256降到64，然后在使用1×1的卷积核恢复到256维。那么此时的总参数为：1x1x256x64 + 3x3x64x64 + 1x1x64x256 = 69632，如果不使用这样的结构，而是使用两个3×3×256的结构，那么参数将是3x3x256x256x2 = 1179648，参数差不多多了20倍。

在右侧的结构中，identity shortcut 非常重要，如果使用 projection shortcut 那么模型的时间复杂度和大小都将翻倍。因此 identity shortcut 将会产生更高效的模型。

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**

ResNet 的本质

为什么 ResNet 会这么有效，仅仅是因为深度增加的原因吗？其实真正起作用的是 shortcut 连接，这个才是关键。

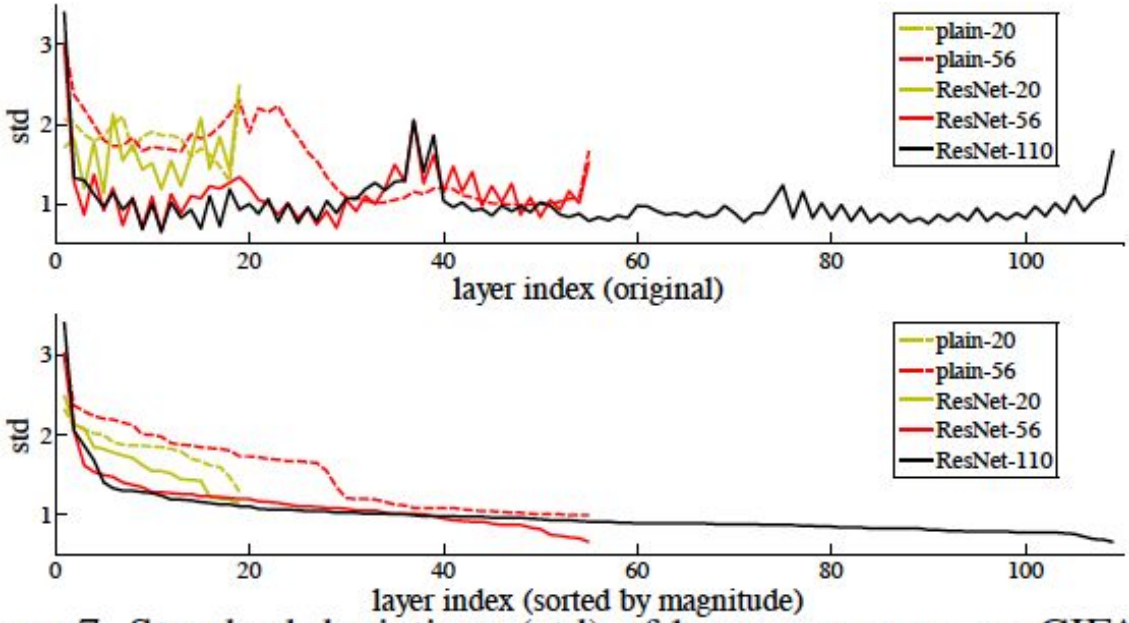
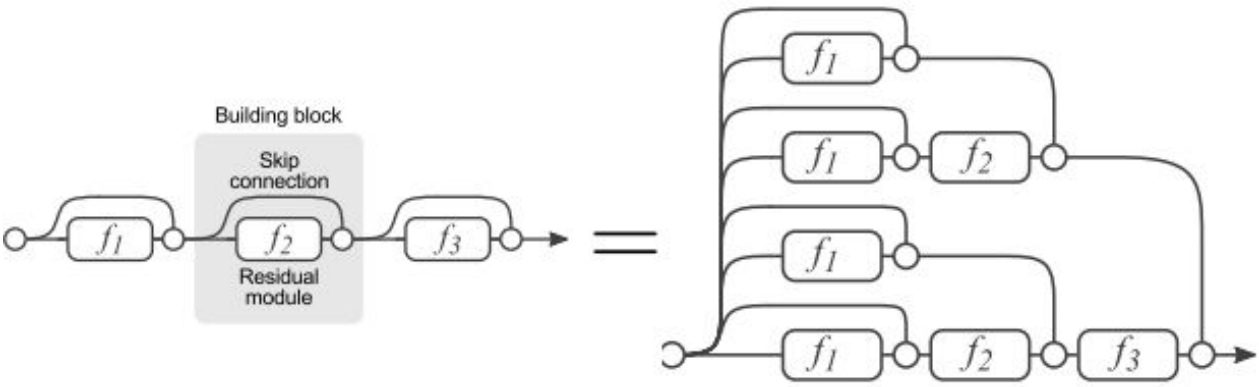


Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

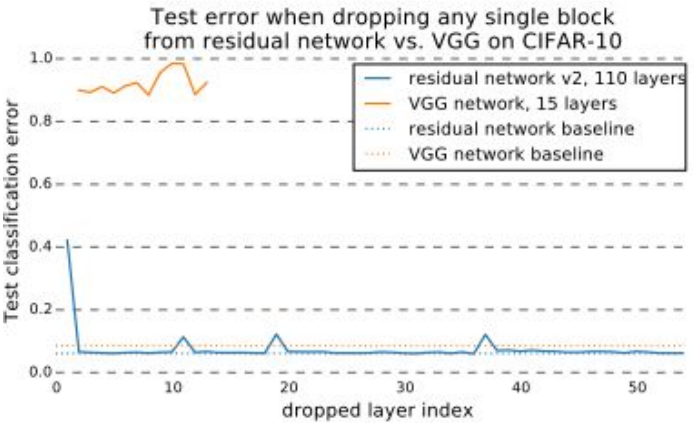
论文中对于网络响应标准差（std） 进行了分析，如上图可以看出，残差网络中大部分层的响应方差都处在较低水平。也就是说，响应方差较低的层响应较为固定，很有可能权重近似于零，这也就是说其所对应的残差结构可能近似于单位映射,网络的实际有效层数是要比全部层数要少一些的，产生了跳过连接(Skip-connection)的作用。这样也就是网络为什么在较深的深度下仍可以保持并提升性能，且没有过多增加训练难度的原因。

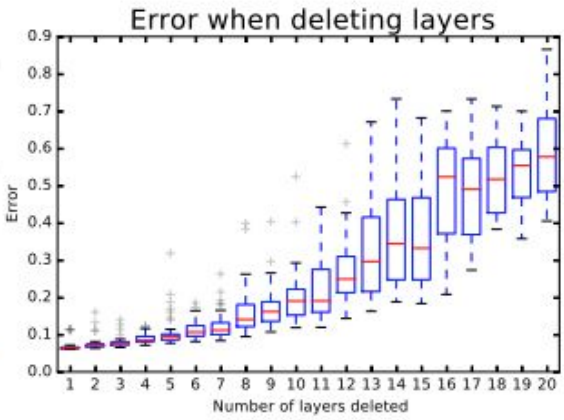
从连接的角度来理解 ResNet 会更方便，下面是摘录《CSDN - 对ResNet的理解》的部分内容，对此解释的较好：

> 基本的残差网络其实可以从另一个角度来理解，这是从另一篇论文里看到的，如下图所示：



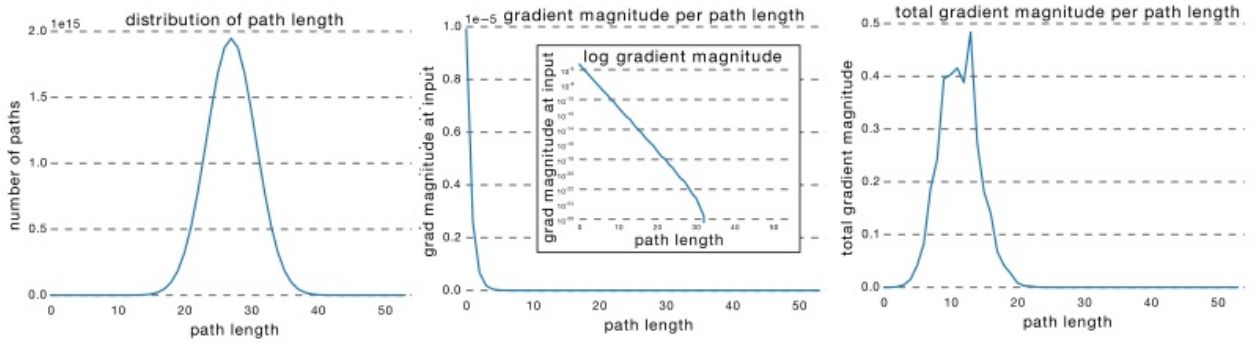
> 残差网络单元其中可以分解成右图的形式，从图中可以看出，残差网络其实是由多种路径组合的一个网络，直白了说，残差网络其实是很多并行子网络的组合，整个残差网络其实相当于一个多人投票系统（Ensembling）。
>
> 删除网络的一部分
> 如果把残差网络理解成一个Ensambling系统，那么网络的一部分就相当于少一些投票的人，如果只是删除一个基本的残差单元，对最后的分类结果应该影响很小；而最后的分类错误率应该是和删除的残差单元的个数成正比的，论文里的结论也印证了这个猜测。
> 下图是比较VGG和ResNet分别删除一层网络的分类错误率变化（左），ResNet分类错误率和删除的基本残差网络单元个数的关系（右）：





ResNet 真面目

ResNet的确可以做到很深，但是从上面的介绍可以看出，网络很深的路径其实很少，大部分的网络路径其实都集中在中间的路径长度上，如下图（左）：



从这可以看出其实ResNet是由大多数中度网络和一小部分浅度网络和深度网络组成的，说明虽然表面上ResNet网络很深，但是其实起实际作用的网络层数并没有很深，我们能来进一步阐述这个问题，我们知道网络越深，梯度就越小，如上图（中）所示。
而通过各个路径长度上包含的网络数乘以每个路径的梯度值，我们可以得到ResNet真正起作用的网络是什么样的，如上图（右）所示。我们可以看出大多数的梯度其实都集中在中间的路径上，论文里称为effective path。
从这可以看出其实ResNet只是表面上看起来很深，事实上网络却很浅。所示ResNet真的解决了深度网络的梯度消失的问题了吗？似乎没有，ResNet其实就是一个多人投票系统。

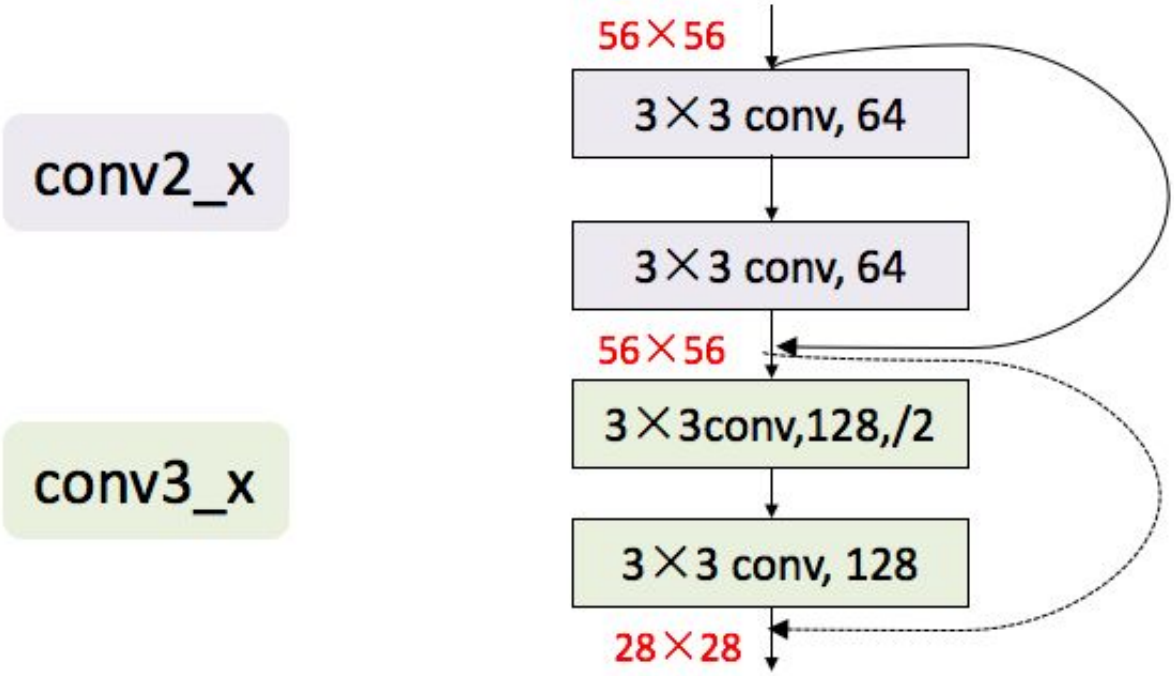
原博客作者没有给出论文出处，找了下，是这篇论文[《Residual Networks Behave Like Ensembles of Relatively Shallow Networks》]

几种连接类型

在 ResNet 基本的结构中，有两种映射方式恒等映射（identity 即公式1）和线性投影映射（linear projection 即公式2）。当输入与输出的维度相同时，可以逐channel 的将 feature map 相加，即按照公式 1 恒等方式。对于输入与输出维度不同的情况，$\mathcal{F}(x)$ 就不能按照channel 与 x 进行相加操作，这时候就需要按照公式 2 的线性映射方式操作。在维度不同时也有两种操作方式：

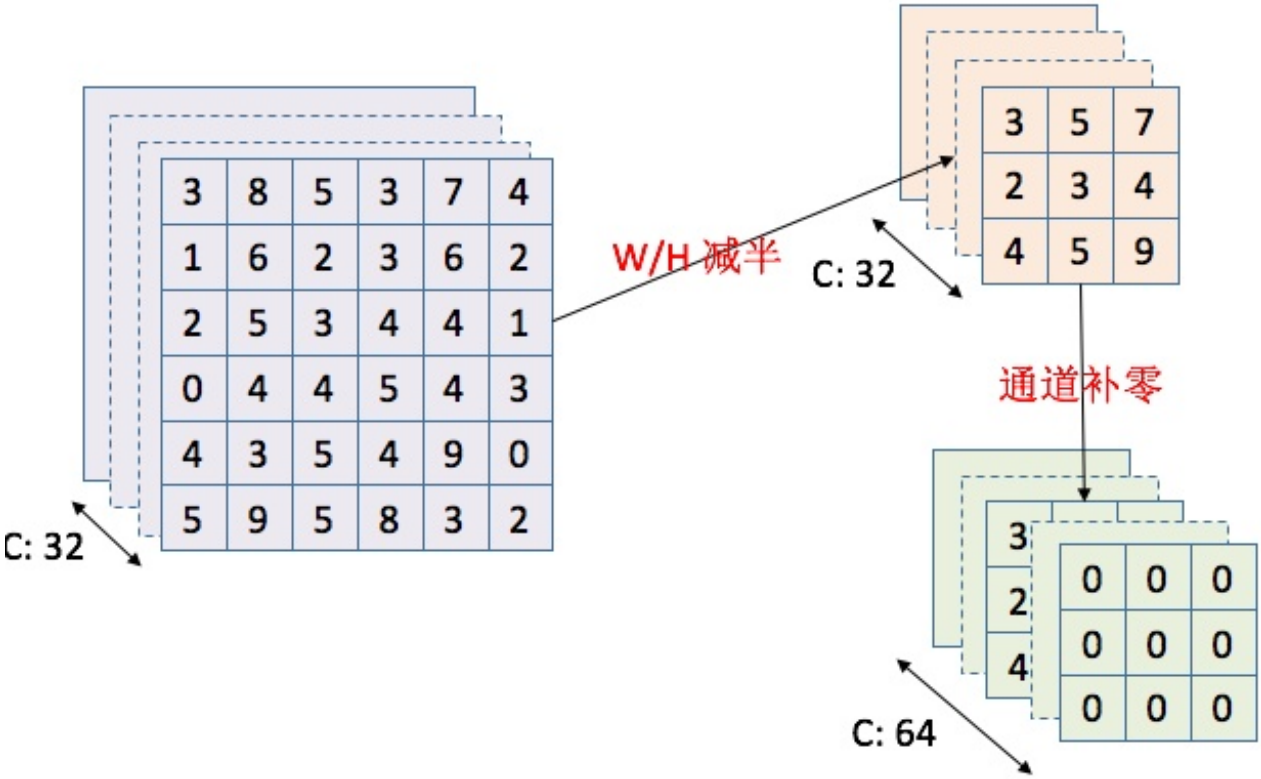
- zero-padding
- 使用公式 2 进行linear projection

以 conv2_x 与 conv3_x 之间的升维操作为例，如下图：



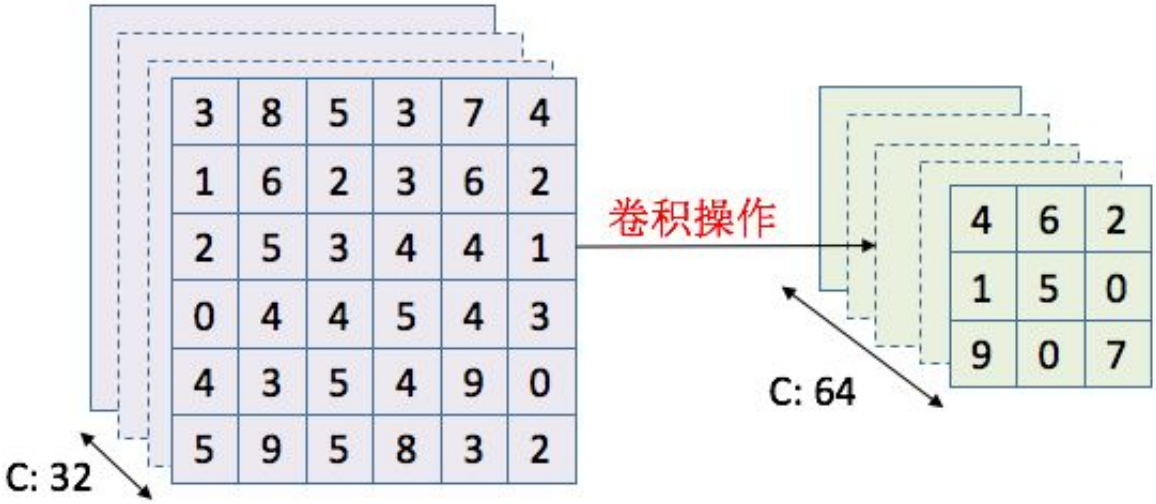
conv2_x 的输出是56×56×64，而conv3_x 第一个残差结构输出的是 28×28×256，维度不同，因此不能使用恒等映射，需要使用其他的方式。（对于 $\mathcal{F}(x)$ 里面为什么经过卷积之后输出的大小没有改变，这是因为里面的操作都进行了补零操作）

首先看第一种方式，zero-padding 方式，这里 zero-padding 值的是对通道进行补零操作，因为输入的 channel 数是64，输出的是128。因此首先对56×56×64的输入尺寸进行下采样（可以使用1×1+2 的池化层完成此操作），使得输出为28×28×64，然后对通道补零，使得通道总数变为128，如下图：



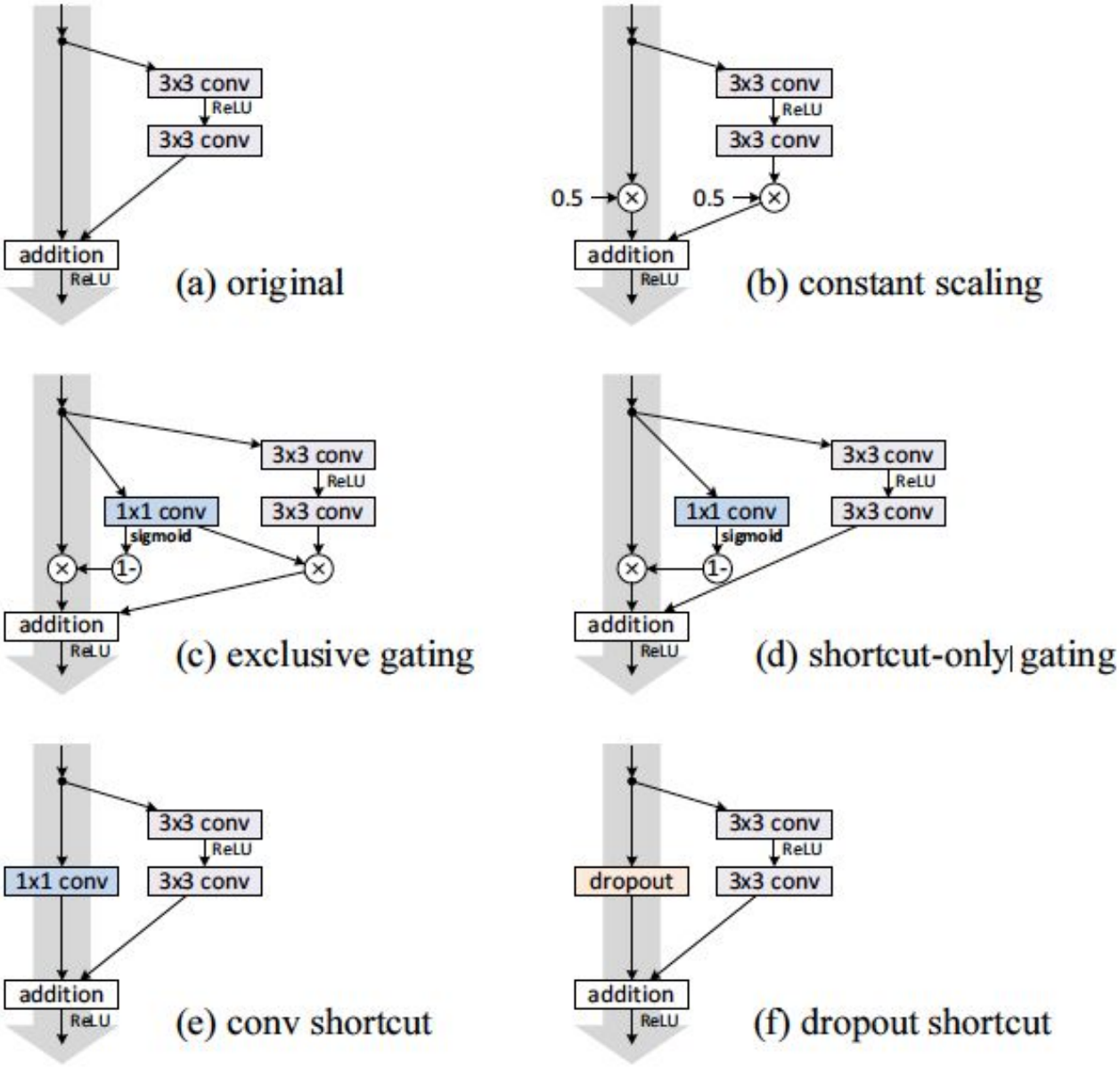
通过补零的方式，没有增加参数。

第二种方式是使用 linear projection，其实就是使用卷积来操作，只不过使用的卷积核为 1×1+2，通道的个数为需要的输出个数，以 conv3_x 为例选择的filter 个数为128个。操作示意图如下（size 是随意举的）



使用此种方式，引入了额外的参数。

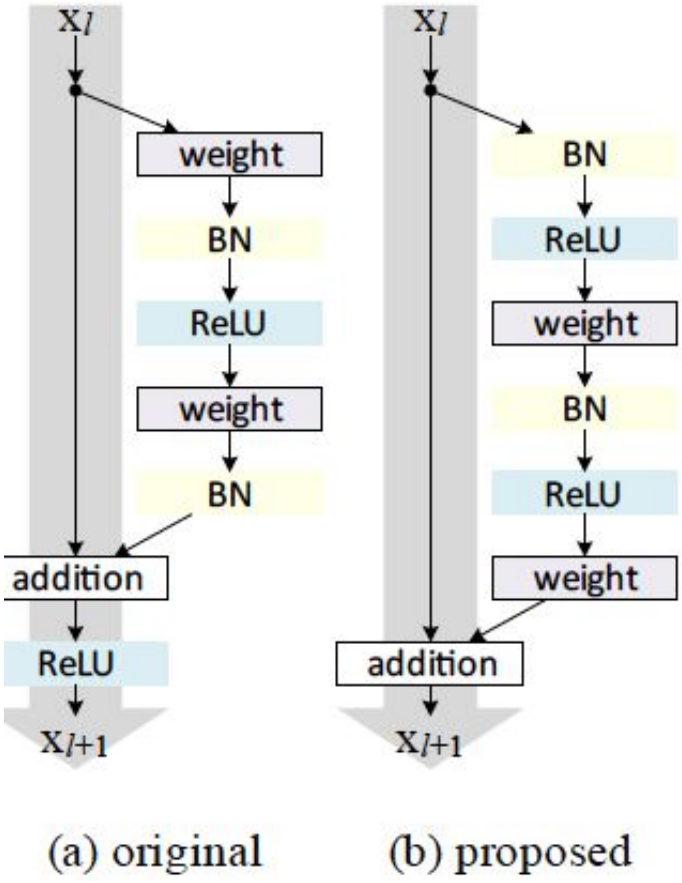
另外还有其他的多种连接方式在《Identity Mappings in Deep Residual Networks》论文中有探讨，引用一张图：



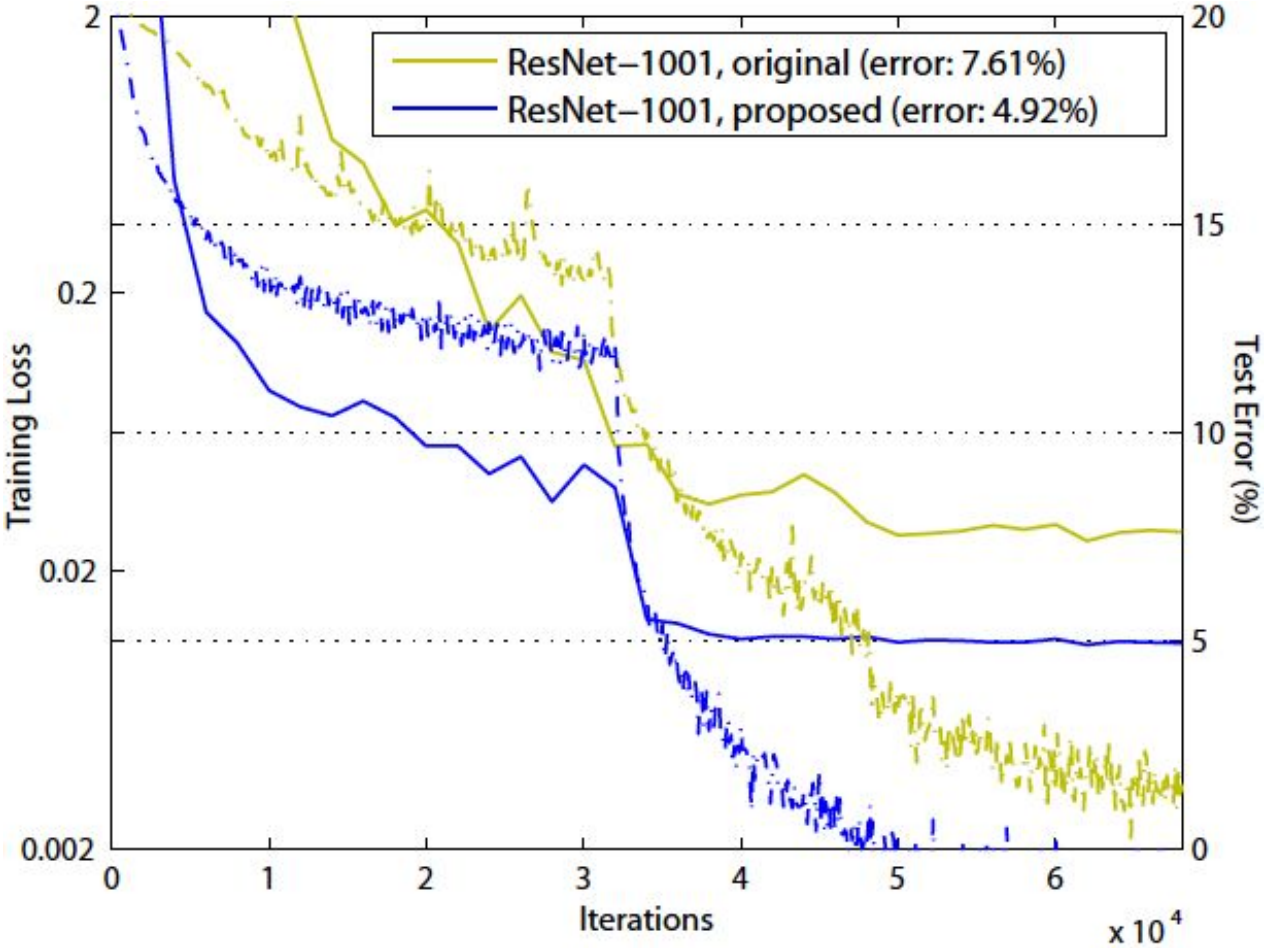
问题

BN 加在了什么地方?

论文中提到了在残差结构中添加 BN 层，但并没有提到具体怎么添加，在原作者的另一篇论文中有提及，如下图：



两种添加方式的训练和测试误差对比如下图：


ResNet−1001, original (error: 7.61%)
ResNet−1001, proposed (error: 4.92%)
Training Loss
Test Error (%)
Iterations
x 10^4
2
0.2
0.02
0.002
20
15
10
5
0
0
1
2
3
4
5
6