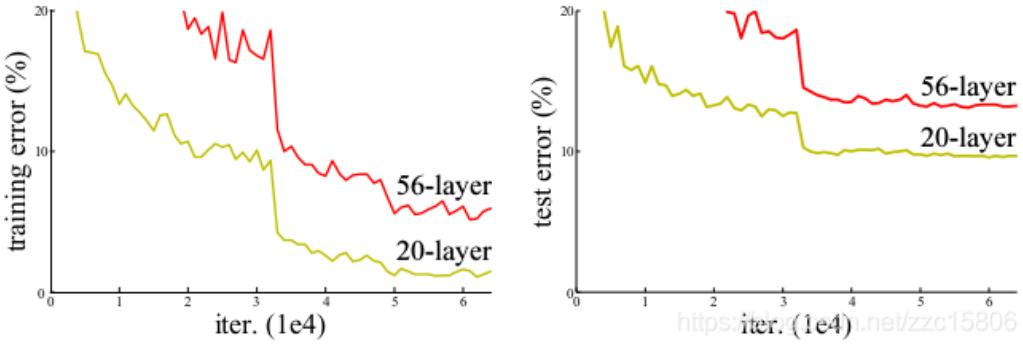
Resnet

一、问题产生

残差网络(ResNet)是微软亚洲研究院的何恺明、孙剑等人 2015 年提出的，它解决了深层网络训练困难的问题。利用这样的结构我们很容易训练出上百层甚至上千层的网络。

> 要理解 ResNet 首先要理解网络变深后会带来什么样的问题。增大网络深度 后带来的第一个问题就是**梯度消失、爆炸**，这个问题在 Szegedy 提出 BN(Batch Normalization)结构后被顺利解决，BN 层能对各层的输出做归一化，这样梯度在反向层层传递后仍能保持大小稳定，不会出现过小或过大的情况。加了 BN 后再加大深度是不是就很容易收敛了呢？答案仍是否定的，作者提到了第二个问题--准确率下降问题(degradation problem)：**层级大到一定程度时准确率就会饱和，然后迅速下降，这种下降即不是梯度消失引起的也不是 overfit 造成的，而是由于网络过于复杂，以至于光靠不加约束的放养式的训练很难达到理想的错误率。**
>
> degradation problem 不是网络结构本身的问题，而是现有的训练方式不够理想造成的。当前广泛使用的训练方法，无论是 SGD，还是 AdaGrad，还是 RMSProp，都无法在网络深度变大后达到理论上最优的收敛结果。我们还可以证明只要有理想的训练方式，更深的网络肯定会比较浅的网络效果要好。证明过程也很简单：假设在一种网络 A 的后面添加几层形成新的网络 B，如果增加的层级只是对 A 的输出做了个恒等映射(identity mapping)，即 A 的输出经过新增的层级变成 B 的输出后没有发生变化，这样网络 A 和网络 B 的错误率就是相等的，也就证明了加深后的网络不会比加深前的网络效果差。

随着网络的加深，梯度弥散问题会越来越严重，导致网络很难收敛甚至无法收敛。梯度弥散问题目前有很多的解决办法，包括网络初始标准化，数据标准化以及中间层的标准化（Batch Normalization）等。但是网络加深还会带来另外一个问题：随着网络加深，出现训练集准确率下降的现象，如下图，



二、残差网络

1.网络结构

何恺明提出了一种残差结构来实现上述恒等映射：**整个模块除了正常的卷积层输出外，还有一个分支把输入直接连到输出上，该输出和卷积的输出做算术相加得到最终的输出，用公式表达就是 H(x)=F(x)+x，x 是输入，F(x)是卷积分支的输出，H(x)是整个结构的输出。**可以证明如果 F(x)分支中所有参数都是 0，H(x)就是个恒等映射。

（直接让一些层去拟合一个潜在的恒等映射函数 H(x) = x，比较困难，这可能就是深层网络难以训练的原因。但是，如果把网络设计为 H(x) = F(x) + x，可以转换为学习一个残差函数 F(x) = H(x) - x. 只要 F(x)=0，就构成了一个恒等映射 H(x) = x。）之所以这样是因为残差学习相比原始特征直接学习更容易。当残差为 0 时，此时堆积层仅仅做了恒等映射，至少网络性能不会下降，实际上残差不会为 0，这也会使得堆积层在输入特征基础上学习到新的特征，从而拥有更好的性能。因此，可以说是在学习残差。简单来说，在进行恒等映射的回收，学习残差比学习原来的映射函数要简单。残差结构人为制造了恒等映射，就能让整个结构朝着恒等映射的方向去收敛，确保最终的错误率不会因为深度的变大而越来越差。如果一个网络通过简单的手工设置参数值就能达到想要的结果，那这种结构就很容易通过训练来收敛到该结果，这是一条设计复杂的网络时百试不爽的规则。回想一下 BN 中为了在 BN 处理后恢复原有的分布，使用了 y=rx+delta 公式， 当手动设置 r 为标准差，delta 为均值时，y 就是 BN 处理前的分布，这就是利用了这条规则。

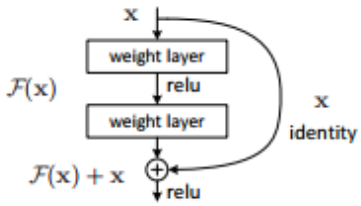


Figure 2. Residual learning: a building block.

残差学习的思想就是上面这张图，可以把它理解为一个 block，定义如下：

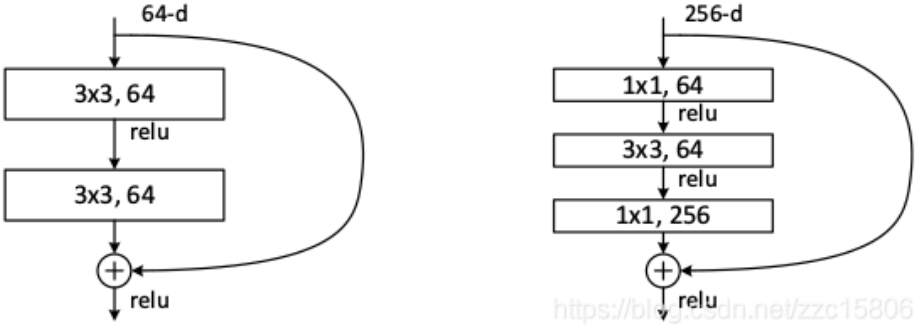
$$y = F(x, \{W_i\}) + x$$

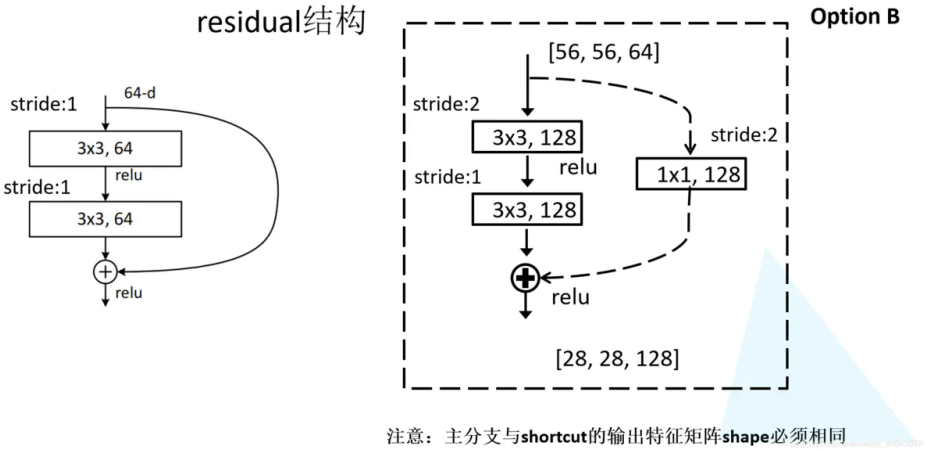
残差学习的 block 一共包含两个分支或者两种映射（mapping）：

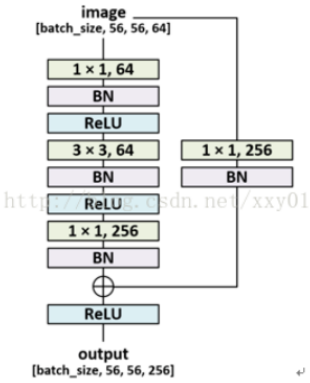
1. identity mapping，指的是上图右边那条弯的曲线。顾名思义，identity mapping 指的就是本身的映射，也就是$_x$ 自身；

2. residual mapping，指的是另一条分支，也就是$F(x)$ 部分，这部分称为残差映射，也就是$y - x$ 。

文中提到了一个名词叫“Shortcut Connection”，实际上它指的就是 identity mapping，这里先解释一下，免的大家后面会 confuse。针对不同深度的 ResNet，作者提出了两种 Residual Block：







对上图做如下说明：

1. 左图为基本的 residual block，residual mapping 为两个 64 通道的 3x3 卷积，输入输出均为 64 通道，可直接相加。该 block 主要使用在相对浅层网络，比如 ResNet-34；

2. 右图为针对深层网络提出的 block，称为“bottleneck” block，主要目的就是为了**降维**。首先通过一个 1x1 卷积将 256 维通道（channel）降到 64 通道，最后通过一个 256 通道的 1x1 卷积恢复。

通过上面的介绍我们知道，residual mapping 和 identity mapping 是沿通道维度相加的，那么如果**通道维度不相同怎么办?**

作者提出在 identity mapping 部分使用 1x1 卷积进行处理，表示如下：

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

其中，W_s 指的是 1x1 卷积操作。

$$a^{[1+2]} = \text{Relu}\left(W^{[1+2]}\left(\text{Relu}\left(\text{W}^{[1+1]}a^{[l]} + \text{b}^{[1+1]}\right) + \text{b}^{[1+2]} + a^{[l]}\right)\right)$$

2.公式推导

残差结构的公式表达：　$x_{l+1} = x_l + F(x_l, W_l)$

通过递归，可以得到任意深层单元 L 特征的表达：　$x_L = x_l + \sum_{i=l}^{L-1} F(x_i, W_i)$

对于任意深的单元 L 的特征 x_L 可以表达为浅层单元 l 的特征 x_l 加上一个形如 $\sum_{i=1}^{L-1} F$ 的残差函数，**表明了任何单元 L 和 l 之间都具有残差特性。**

同样的，对于任意深的单元 L ，它的特征 $x_L = x_0 + \sum_{i=0}^{L-1} F(x_i, W_i)$ ，即为之前所有残差函数输出的总和再加上 x_0 。

对于反向传播，假设损失函数为 E ，根据反向传播的链式法则可以得到：

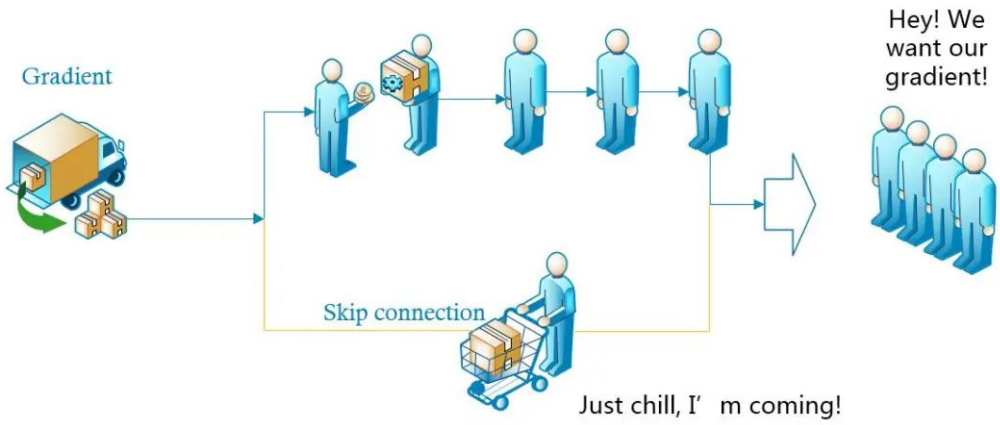
$$\frac{\partial \varepsilon}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i, w_i)\right)$$

式子被分为两个部分：

- 不通过权重层的传递： $\frac{\partial \varepsilon}{\partial x_L}$
- 通过权重层的传递： $\frac{\partial \varepsilon}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i, w_i)\right)$

前者保证了信号能够直接传回到任意的浅层 x_l ，同时这个公式也保证了不会出现梯度消失的现象，因为 $\frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} F(x_i, w_i)$ 不可能为-1。

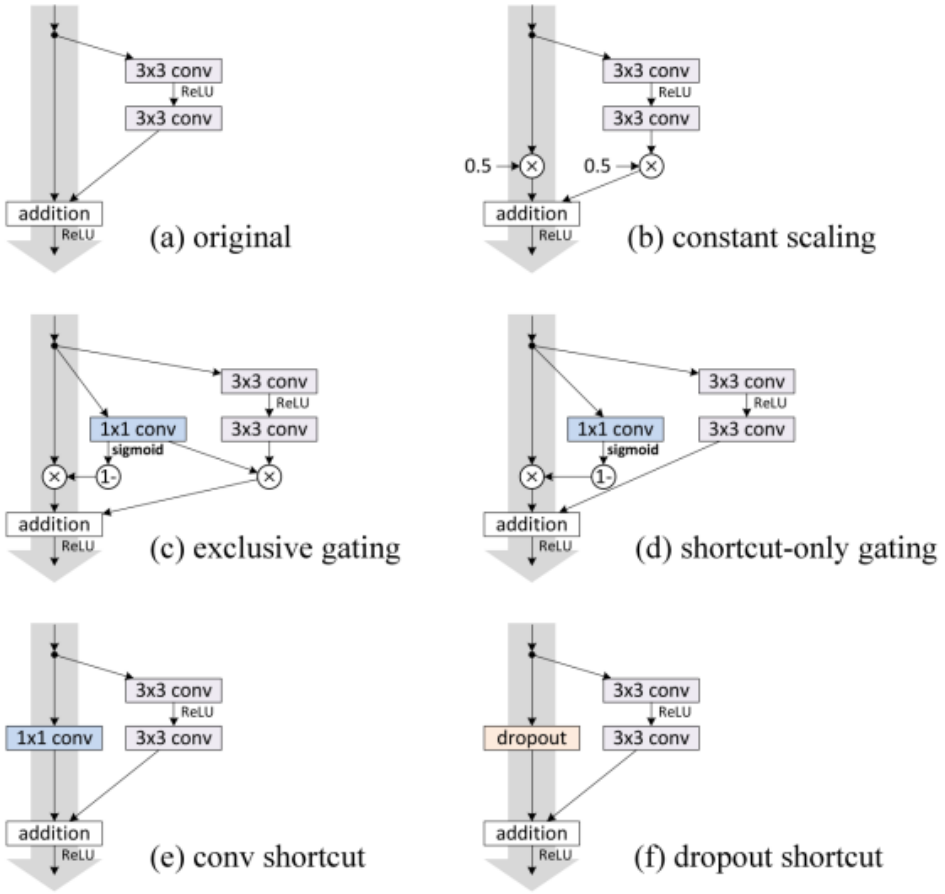
一个直观理解图：



如上图所示，左边来了一辆装满了“梯度”商品的货车，来领商品的客人一般都要排队一个个拿才可以，如果排队的人太多，后面的人就没有了。于是这时候派了一个人走了“快捷通道”，到货车上领了一部分“梯度”，直接送给后面的人，这样后面排队的客人就能拿到更多的“梯度”。

其他的参考解释

- F 是求和前网络映射，H 是从输入到求和后的网络映射。比如把 5 映射到 5.1，那么引入残差前是 F’(5)=5.1，引入残差后是 H(5)=5.1, H(5)=F(5)+5, F(5)=0.1。这里的 F’和 F 都表示网络参数映射，引入残差后的映射对输出的变化更敏感。比如 s 输出从 5.1 变到 5.2，映射 F’的输出增加了 1/51=2%，而对于残差结构输出从 5.1 到 5.2，映射 F 是从 0.1 到 0.2，增加了 100%。**明显后者输出变化对权重的调整作用更大，所以效果更好。**残差的思想都是去掉相同的主体部分，从而突出微小的变化，看到残差网络我第一反应就是差分放大器
- 至于为何 shortcut 的输入时 X，而不是 X/2 或是其他形式。kaiming 大神的另一篇文章[2]中探讨了这个问题，对以下 6 种结构的残差结构进行实验比较，shortcut 是 X/2 的就是第二种，结果发现还是第一种效果好啊（摊手）。



这种残差学习结构可以通过前向神经网络+shortcut 连接实现，如结构图所示。而且 shortcut 连接相当于简单执行了同等映射，不会产生额外的参数，也不会增加计算复杂度。 而且，整个网络可以依旧通过端到端的反向传播训练。

3.problem

- **为什么想到使用跨层连接？**

1. 在多层感知机（multi-layer perceptrons，MLP）中加一层从输入到输出的线性层
   GoogLeNet 中使用辅助分类器防止梯度爆炸 / 消失
2. 在此之前已有研究者使用跨层连接对响应和梯度中心化（center）处理
3. inception 结构本质也是跨层连接
4. highway 网络也使用到了跨层连接

- **为什么残差学习可以解决“网络加深准确率下降”的问题?**

对于一个神经网络模型，如果该模型是最优的，那么训练就很容易将 residual mapping 优化到 0，此时只剩下 identity mapping，那么无论怎么增加深度，理论上网络会一直处于最优状态。因为相当于后面所有增加的网络都会沿着 identity mapping（自身）进行信息传输，可以理解为最优网络后面的层数都是废掉的（不具备特征提取的能力），实际上没起什么作用。这样，网络的性能也就不会随着深度的增加而降低了。

1. **自适应深度：**网络退化问题就**体现了多层网络难以拟合恒等映射这种情况**，也就是说 $H(x)$ 难以拟合 x ，但使用了残差结构之后，拟合恒等映射变得很容易，直接把网络参数全学习到为 0，只留下那个恒等映射的跨层连接即可。于是当网络不需

要这么深时，中间的恒等映射就可以多一点，反之就可以少一点。（当然网络中出现某些层仅仅拟合恒等映射的可能性很小，但根据下面的第二点也有其用武之地；另外关于为什么多层网络难以拟合恒等映射，这涉 及到信号与系统的知识

2. **差分放大器”** ：假设最优 $H(x)$ 更接近恒等映射，那么网络更容易发现除恒等映射之外微小的波动.
3. **模型集成**：整个 ResNet 类似于多个网络的集成，原因是删除 ResNet 的部分网络结点不影响整个网络的性能，但 VGGNet 会崩溃，具体可以看 这篇NIPS论文 .
4. **缓解梯度消失**：针对一个残差结构对输入 x 求导就可以知道，由于跨层连接的存在，总梯度在 $F(x)$ 对 x 的导数基础上还会加 1.

4.相关的工作

- **残差表示**

  VALD，Fisher Vector 都是是对残差向量编码来表示图像，在图像分类，检索表现出优于编码原始向量的性能。

  在 low-level 的视觉和计算机图形学中，为了求解偏微分方程，广泛使用的 **Multigrid** 方法将系统看成是不同尺度上的子问题。每个子问题负责一种更粗糙与更精细尺度的残差分辨率。**Multigrid** 的一种替换方法是层次化的预处理，层次化的预处理依赖于两种尺度的残差向量表示。实验表明，这些求解器要比对残差不敏感的求解器收敛更快。
- **shortcut 连接**

  shortcut 连接被实验和研究了很久。**Highway networks** 也使用了带有门函数的 shortcut。但是这些门函数需要参数，而 ResNet 的 shortcut 不需要参数。而且当 **Highway networks** 的门函数的 shortcut 关闭时，相当于没有了残差函数，但是 ResNet 的 shortcut 一直保证学习残差函数。而且，当 **Highway networks** 的层数急剧增加时，没有表现出准确率的上升了。总之，ResNet 可以看成是 **Highway networks** 的特例，但是从效果上来看，要比 **Highway networks** 好。

作者使用了 ImageNet 和 CIFAR 两种数据来证明 ResNet 的有效性：

首先是 ImageNet，作者比较了相同层数的 ResNet 结构和传统结构的训练效果。左侧是一个传统结构的 VGG-19 网络(每个卷积后都跟了 BN)，中间是传统结构的 34 层网络(每个卷积后都跟了 BN)，右侧是 34 层的 ResNet(实线表示直连，虚线表示用 1x1 卷积进行了维度变化，匹配输入输出的特征数)。下图是这几种网络训练后的结果，左侧的数据看出传统结构的 **34 层网络(红线)要比 VGG-19(蓝绿色线)的错误率高，由于每层都加了 BN 结构，所以错误高并不是由于层级增大后梯度消失引起的，而是 degradation problem 造成** ；下图右侧的 ResNet 结构可以看到 34 层网络(红线)要比 18 层网络(蓝绿色线)错误率低，这是因为 ResNet 结构已经克服了 degradation problem。此外右侧 ResNet 18 层网络最后的错误率和左侧传统 18 层网络的错误率相近，这是因为 18 层网络较为简单，即使不用 ResNet 结构也可以收敛到比较理想的结果。

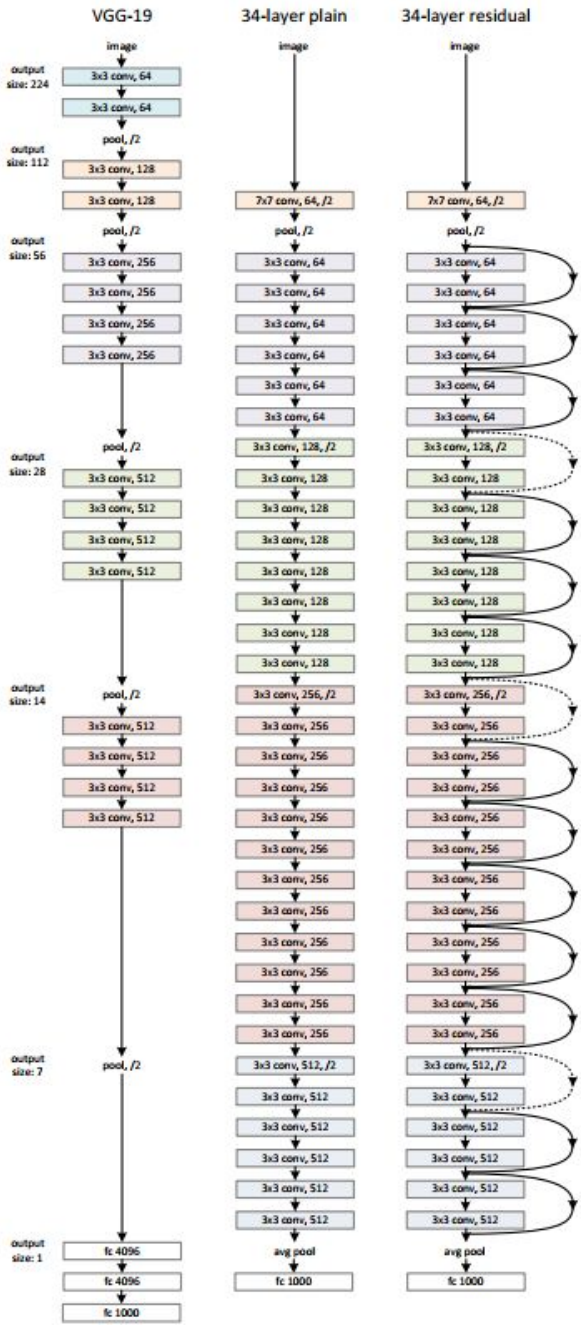
对下图进行如下说明：

1. 相比于 VGG-19，堆叠了 34 层的网络，记为 plain-34，虽然更深了，但 FLOPs（代表计算复杂度，multiply-adds）仅为 VGG-19 的 18%ResNet 没有使用全连接层，而使用了全局平均池化层，可以减少大量参数。VGG-19 大量参数集中在全连接层；

2. ResNet-34 中跳跃连接**“实线”**为 identity mapping 和 residual mapping 通道数相同，**“虚线”**部分指的是两者通道数不同，需要使用 1x1 卷积调整通道维度，使其可以相加。

3.升维有两种方式：**第一种是直接全补 0，这样做优势是不会增加网络的参数；第二种是 1 x 1 卷积升维，后面实验部分会进行比较。**

注意这里除第一个 stage 之外都会在 stage 的第一层使用步长为 2 的卷积来进行下采样，倒数第二层输出的 feature map 后面是全局平均池化（global average pooling，VGGNet 预测时转化成全卷积网络的时候也用到了），也就是每个 feature map 求平均值，因为 ImageNet 输出是 1000 个类别，所以再连接一层 1000 个神经元的全连接层，最后再接上一个 Softmax。



三种结构来比较测试，VGG-19、34 层传统网络和 34 层 ResNet

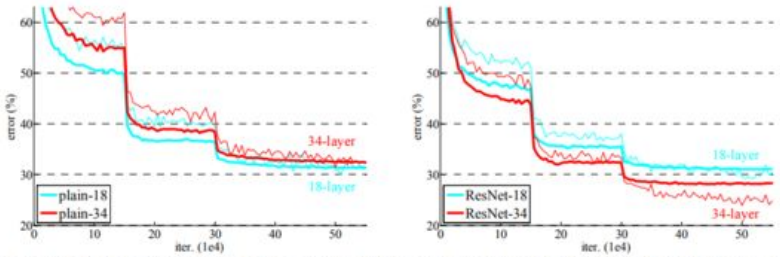
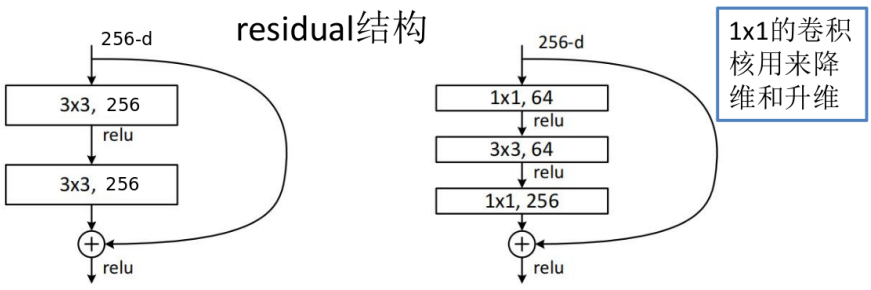


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

传统网络结构的训练结果 (左图) 和 ResNet 结构的训练结果 (右图)

shortcut 路径大致也可以分成 2 种，取决于残差路径是否改变了 feature map 数量和尺寸，一种是将输入 x 原封不动地输出，**另一种则需要经过 1× 1 卷积来升维 or/and 降采样**，主要作用是**将输出与 $F(x)$ 路径的输出保持 shape 一致**，对网络性能的提升并不明显，两种结构如下图所示，

左侧那样的 ResNet 结构只是用于较浅的 ResNet 网络，如果网络层数较多，靠近网络输出端的维度就会很大，仍使用图左侧的结构计算量就会极大， 对较深的网络我们都使用图 右侧的 bottleneck 结构，先用一个 1x1 卷积进行降维，然后 3x3 卷积，最后用 1x1 升维恢复原有的维度。为什么深层网络要使用右侧的残差结构呢。因为，右侧的残差结构能够减少网络参数与运算量。同样输入、输出一个 channel 为 256 的特征矩阵，如果使用左侧的残差结构需要大约 1170648 个参数，但如果使用右侧的残差结构只需要 69632 个参数。明显搭建深层网络时，使用右侧的残差结构更合适。



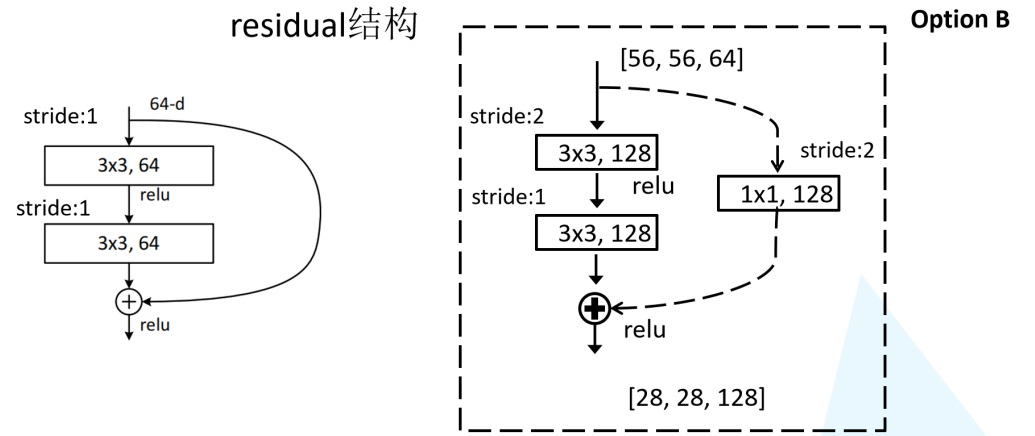
$$3\times3\times256\times256+3\times3\times256\times256=1{,}179{,}648$$
$$1\times1\times256\times64+3\times3\times64\times64+1\times1\times64\times256=69{,}632$$

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.

注意：主分支与shortcut的输出特征矩阵shape必须相同

较浅的网络的 ResNet 结构 (左)和较深网络的 ResNet 结构 (右)

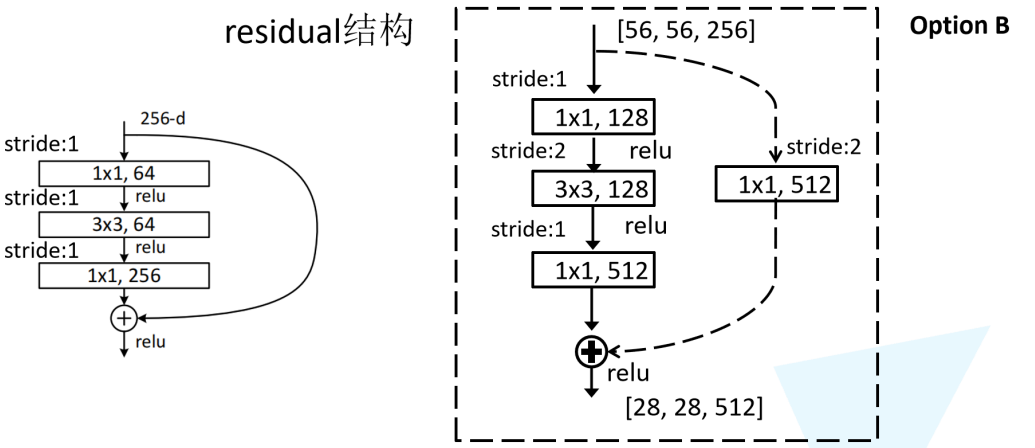
我们先对左侧的残差结构（针对 ResNet18/34）进行一个分析。如下图所示，该残差结构的主分支是由**两层 3x3 的卷积层组成，**而残差结构右侧的连接线是 shortcut 分支也称捷径分支（注意为了让主分支上的输出矩阵能够与我们捷径分支上的输出矩阵进行相加，必须保证这两个输出特征矩阵有相同的 shape）。如果刚刚仔细观察了 ResNet34 网络结构图，应该能够发现图中会有一些虚线的残差结构。在原论文中作者只是简单说了这些虚线残差结构有降维的作用，并在捷径分支上通过 1x1 的卷积核进行降维处理。而下图右侧给出了详细的虚线残差结构，**注意下每个卷积层的步距 stride，以及捷径分支上的卷积核的个数（与主分支上的卷积核个数相同）**。



注意：主分支与**shortcut**的输出特征矩阵**shape**必须相同

至于 Residual Block 之间的衔接，在原论文中，F(x)+x 经过 ReLU 后直接作为下一个 block 的输入 x。

接着我们再来分析下针对 ResNet50/101/152 的残差结构，如下图所示。在该残差结构当中，主分支使用了三个卷积层，第一个是 1x1 的卷积层用来压缩 channel 维度，第二个是 3x3 的卷积层，第三个是 1x1 的卷积层用来还原 channel 维度（注意主分支上第一层卷积层和第二层卷积层所使用的卷积核个数是相同的，第三层是第一层的 4 倍）。该残差结构所对应的虚线残差结构如下图右侧所示，同样在捷径分支上有一层 1x1 的卷积层，它的卷积核个数与主分支上的第三层卷积层卷积核个数相同，注意每个卷积层的步距。**（注意：原论文中，在下图右侧虚线残差结构的主分支中，第一个 1x1 卷积层的步距是 2，第二个 3x3 卷积层步距是 1。但在 pytorch 官方实现过程中是第一个 1x1 卷积层的步距是 1，第二个 3x3 卷积层步距是 2，这么做的好处是能够在 top1 上提升大概 0.5%的准确率。可参考** Resnet v1.5



注意：主分支与**shortcut**的输出特征矩阵**shape**必须相同

三、残差学习

根据多层的神经网络理论上可以拟合任意函数，那么可以利用一些层来拟合函数。问题是直接拟合 H(x)还是残差函数，由前文，拟合残差函数 F(x) = H(x) - x 更简单。虽然理论上两者都能得到近似拟合，但是后者学习起来显然更容易。

作者说，这种残差形式是由**退化问题**激发的。根据前文，如果增加的层被构建为同等函数，那么理论上，更深的模型的训练误差不应当大于浅层模型，但是出现的**退化问题**表面，求解器很难去利用多层网络拟合同等函数。但是，残差的表示形式使得多层网络近似起来要容易的多，如果同

等函数可被优化近似，那么多层网络的权重就会简单地逼近 0 来实现同等映射，即 F(x) = 0。实际情况中，同等映射函数可能不会那么好优化，但是对于残差学习，求解器根据输入的同等映射，也会更容易发现扰动，总之比直接学习一个同等映射函数要容易的多。根据实验，可以发现学习到的残差函数通常响应值比较小，**同等映射（shortcut）**提供了合理的前提条件。

整个残差结构可以形式化定义为 $y = F(x, \{W_i\}) + x$ ，这里的 $F(x, \{W_i\})$ 指拟合的残差映射，如上图中有两层全连接层，即 $F = W_2\sigma(W_1x)$ 其中 σ 指 ReLU,注意这里为了简洁没有写上 bias。当 F 和 x 维度相同时，可以直接逐元素相加；但如果不同，就必须给 x 再加一个线性映射，将其映射到一个与 F 维度相同的向量，此时整个残差结构为 $y = F(x, \{W_i\}) + W_sx$ ， W_s 就是一个用于维度匹配的矩阵。其实当 F 与 x 维度相同时也可以用一个方阵 W_s ，但经过后面的实验发现恒等编号更好。

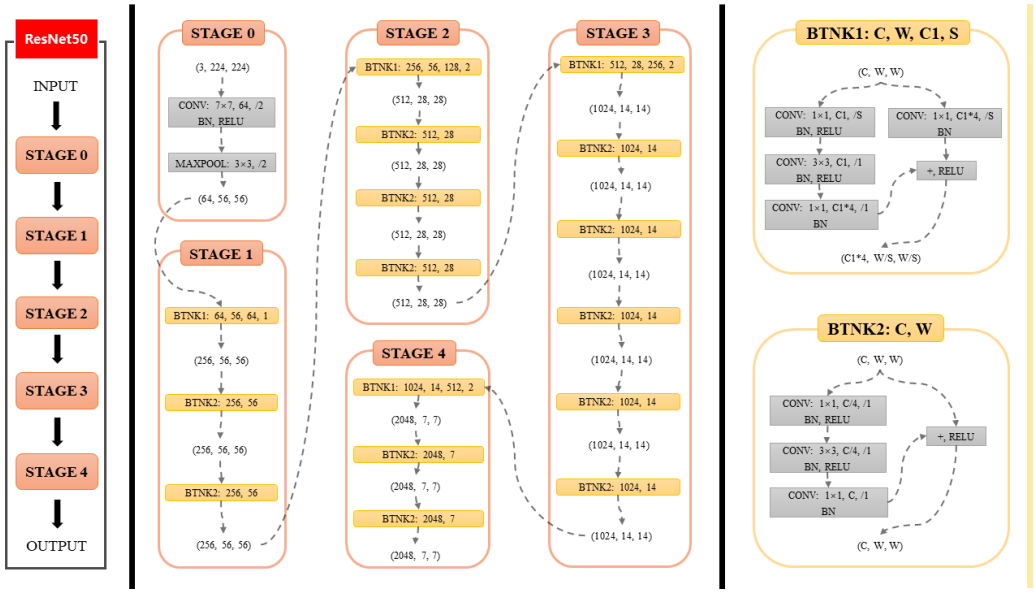
首先，ResNet 在 PyTorch 的官方代码中共有 5 种不同深度的结构，深度分别为 18、34、50、101、152（各种网络的深度指的是“需要通过训练更新参数”的层数，如卷积层，全连接层等），和论文完全一致。图 1 是论文里给出每种 ResNet 的具体结构：

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

其中，根据 Block 类型，可以将这五种 ResNet 分为两类：(1) 一种基于 BasicBlock，浅层网络 ResNet18, 34 都由 BasicBlock 搭成；(2) 另一种基于 Bottleneck，深层网络 ResNet50, 101, 152 乃至更深的网络，都由 Bottleneck 搭成。Block 相当于积木，每个 layer 都由若干 Block 搭建而成，再由 layer 组成整个网络。每种 ResNet 都是 4 个 layer（不算一开始的 7×7 7\times77×7 卷积层和 3×3 3\times33×3maxpooling 层），如图 1，conv2_x 对应 layer1，conv3_x 对应 layer2，conv4_x 对应 layer3，conv5_x 对应 layer4。方框中的“×2 \times2×2”、“×3 \times3×3”等指的是该 layer 由几个相同的结构组成。）

中间卷积部分主要是下图中的蓝框部分，通过 3*3 卷积的堆叠来实现信息的提取。红框中的[2, 2, 2, 2]和[3, 4, 6, 3]等则代表了 bolck 的重复堆叠次数。

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



resnet18( )函数中有一句 ResNet(BasicBlock, [2, 2, 2, 2], **kwargs)，这里的[2, 2, 2, 2]与图中红框是一致的，如果你将这行代码改为 ResNet(BasicBlock, [3, 4, 6, 3], **kwargs)， 那你就会得到一个 res34 网络。

四、ResNet50 整体结构

首先需要声明，这张图的内容是 ResNet 的 Backbone 部分（即图中没有 ResNet 中的全局平均池化层和全连接层）。

如本图所示，输入 `INPUT` 经过 ResNet50 的 5 个阶段（*Stage 0*、*Stage 1*、……）得到输出 `OUTPUT` 。

下面附上 ResNet 原文展示的 ResNet 结构

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
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

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

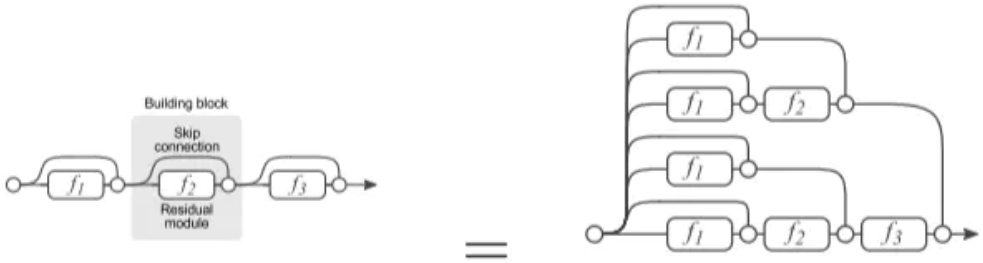
这五种网络从大的层面上来讲，都具有五个组，作者对它们进行了命名：conv1，conv2_x, conv3_x, conv4_x, conv5_x；五组的输出尺寸都是一样的，分别是 112x112, 56x56, 28x28, 14x14, 7x7。我们可以看到，在经过每组卷积层组之后，尺寸降低了一半。**原始输入为 224x224，用 7x7，stride=2 卷积，得到 112x112 的输出，我们可以知道 conv1 的 padding=3；conv2_x 的输出和输入相同，卷积核尺寸为 3x3，我们可以知道做了 same padding，stride=1；conv3_x, conv4_x, conv5_x 的输出均减半，没有使用池化层，所以这几组也是做了 same padding，第一个 block 第一层的 stride=2，其它层 stride=1。**

上图描述了 ResNet 多个版本的具体结构，本文描述的“ResNet50”中的 50 指有 50 个层。和上图一样，本图描述的 ResNet也分为5个阶段。

例如：101–layer 那列，101–layer 指的是 101 层网络，首先有个输入 7x7x64 的卷积，然后经过 3 + 4 + 23 + 3 = 33 个 building block，每个 block 为 3 层，所以有

33 x 3 = 99 层，最后有个 fc 层(用于分类)，所以 1 + 99 + 1 = 101 层，确实有 101 层网络； 注：101 层网络仅仅指卷积或者全连接层，而激活层或者 Pooling 层并没有计算在内；我们关注 50-layer 和 101-layer 这两列，可以发现，它们唯一的不同在于 conv4_x，ResNet50 有 6 个 block，而 ResNet101 有 23 个 block，两者之间差了 17 个 block，也就是 17 x 3 = 51 层。

五、对 Resnet 的讨论



(a) Conventional 3-block residual network　　(b) Unraveled view of (a)

我们可以把残差网络单元转换成右图的形式，从右图中可以看出，残差网络其实是由多种路径组合的一个网络，从输入到输出之间我们通过选择是否跳过残差块（F(x)=0）来选择不同的映射。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）。

六、好文

- https://lonepatient.top/2018/06/25/Deep_Learning_For_Computer_Vision_With_Python_PB_12
- https://zhuanlan.zhihu.com/p/79378841