ResNet与ResNext

残差网络是针对深度网络退化问题的一种方法，经验来看，网络的深度对模型的性能至关重要，当增加网络层数后，网络可以进行更加复杂的特征模式的提取，所以当模型更深时理论上可以取得更好的结果，但实际上，当网络深度增加时，网络准确度出现饱和，甚至出现下降。

其**原因一在于随着网络层数的增加，其梯度在链式法则的放大或者缩小的作用下**，将会出现**梯度弥散/爆炸**等现象，当然，**这个问题很大程度上已经被标准初始化和中间层正规化方法有效控制了**，这些方法使得深度神经网络可以收敛。

其原因二在于**网络退化问题**，根据一般的理论：如果存在K层网络f是当前的最优网络，那么就应该可以构造出一个N层的更深的网络，其K层之后的网络是第K层的恒等映射，就可以获得与K层网络同样的效果，**总而言之，与浅层网络相比，更深的网络的表现不应该更差。**而这与实验结果相反，基于此，一个合理的猜测就是，**对神经网络来说，恒等映射并不容易拟合。**

基于此，可以对网络单元进行一定的改造，来改善退化问题。

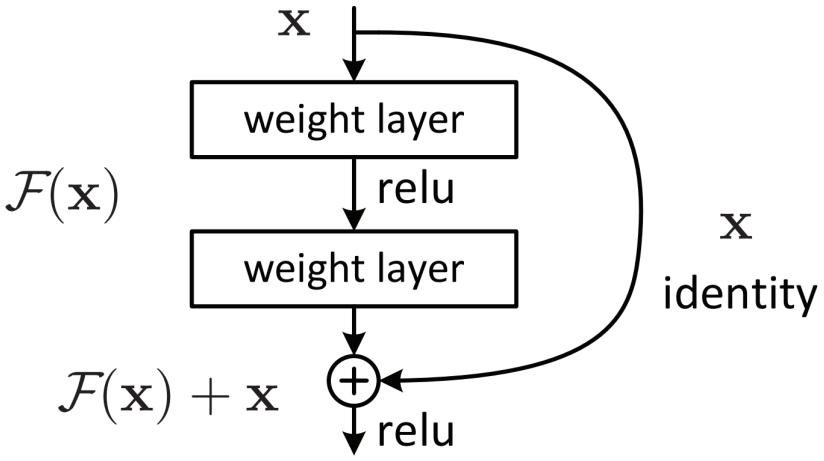
残差函数

为了解决神经网络不容易拟合一个恒等映射的问题，有一种解决思路即是构造一种天然的恒等映射，即：可以将神经网络单元希望拟合的函数$\mathbb{H}$分解为两个部分：即恒等映射与残差：

$$z^{(l)} = \mathbb{H}(a^{(l-1)}) = a^{l-1} + F(a^{l-1})$$

其中$F(\cdot)$是残差函数。而在网络高层，学习一个恒等映射$\mathbb{H}(a^{(l-1)}) \to a^{(l-1)}$，即令残差部分趋向于0，即$F(a^{l-1}) \to 0$

基于这一公式，可以选择将残差网络使用跳层的方式进行连接，将单元的输入直接与单元的输出进行连接：



基于这一构想，可以发现，这一网络可以在同等层数的前提下残差网络也**收敛得更快**，使得前馈神经网络可以采用更深的设计。除此之外，**去除个别神经网络层，残差网络的表现不会受到显著影响**，这与传统的前馈神经网络大相径庭。

对于为何可以实现这一功能，存在多种解释，即：

可以考虑将任意层数的网络展开（不考虑激活函数）：

$$
\begin{aligned}
&a^{l_2} = a^{(l_2-1)} + F(a^{(l_2-1)}) \\
&= (a^{(l_2-2)} + F(a^{(l_2-2)}) + F(a^{(l_2-1)}) \\
&\cdots \\
&\Longrightarrow a^{l_2} = a^{l_1} + \sum_{i=l_1}^{l_2-1} F(a^i)
\end{aligned}
$$

即：在前向传播时，**输入信号可以从任意低层直接传播到高层。**由于包含了一个天然的恒等映射，**一定程度上可以解决网络退化问题。**

而针对最终的损失σ的损失如下：

$$
\begin{aligned}
&\frac{\partial \sigma}{\partial a^{l_1}} = \frac{\partial \sigma}{\partial a^{l_2}} \frac{\partial a^{l_2}}{\partial a^{l_1}} \\
&= \frac{\partial \sigma}{\partial a^{l_2}} (1 + \frac{\partial}{\partial a^{l_1}} \sum_{i=l_1}^{l_2-1} F(a^i))
\end{aligned}
$$

根据此式可以看出，损失对某底层的输出的梯度，被分解为了两项，其中前一项表明：反向传播时，**错误信号可以不经过任何中间权重矩阵变换直接传播到低层，这一定程度上可以缓解梯度弥散问题（即便中间层矩阵权重很小，梯度也基本不会消失）。**

ResNext基本信息

`ResNeXt`实际上是基于将`Resnet`与Inception 的`Split-Transfrom-Concat`思想结合而出的产物；其主要思路即是将**单路卷积变成多个支路的多路卷积，这多个支路之间结构一致，进行分组卷积。**

对于一个最简单的，没有激活函数的神经元来说，其最基本的公式与结构如下：

$$
\sum_{i=1}^{D} w_i x_i
$$

即有如下结构：

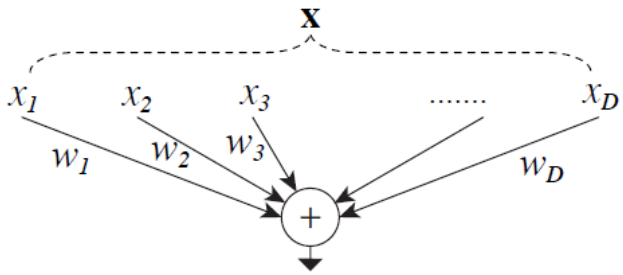


Figure 2. A simple neuron that performs inner product.

它实际上就是一个`Split-Transfrom-Concat`思想的实例：

- Split：将数据X分成D个特征
- Transform 对每一个特征进行一次线性变换
- Merge 通过单位加和得到最终输出

由此可以归纳出一个通用单元的公式：

$$
F(x) = \sum_{i=1}^{C} \mathrm{T}_i(x)
$$

若引入残差网络，则可以写成如下形式：

$$
y = x + \sum_{i=1}^{C} \mathrm{T}_i(x)
$$

而针对Inception网络，其具有`Split-Transfrom-Concat`这一结构的明显特征，而刻意的对其中的网络拓扑结构进行修改，可能会破坏其功能，且这一人为调整会产生众多超参数，不利于模型的改进。

基于此，作者希望每个结构使用相同的拓扑结构，那么这时候的Inception的结构如下：

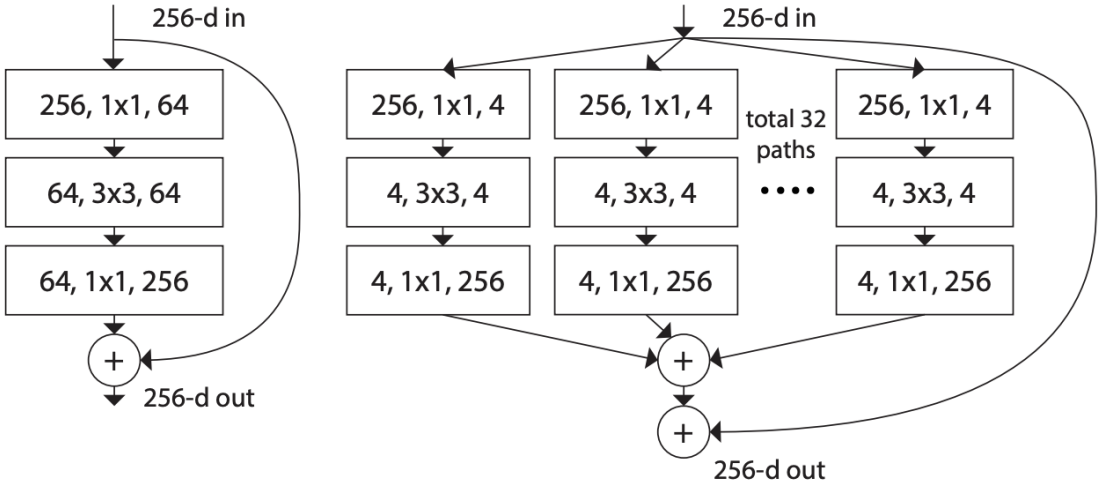


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

在这里，将会使用`ResNet`的基本架构，提出了 Aggregated transformations，用一种平行堆叠相同拓扑结构的blocks代替原来 `ResNet` 的三层卷积的block，将整个卷积网络分为32个path分别进行操作，提出了 Aggregated transformations，用一种平行堆叠相同拓扑结构的blocks代替原来 `ResNet` 的三层卷积的block，每一个Block中的每一条Path被称为一个`cardinality`,它是一个独立于网络的深度与宽度外的独立的维度，一般而言，增加`cardinality`是获得精度的一种有效方式。

这实际上也是对所谓"分组卷积"的一种应用，将输入的高维度图像分解为若干组较小维度的输入，分别进行卷积运算后进行堆叠。

基于此，可以得到其网络架构的一些参数：

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.5}\times10^6$	$\mathbf{25.0}\times10^6$
FLOPs		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$

Table 1. (**Left**) ResNet-50. (**Right**) ResNeXt-50 with a 32×4d template (using the reformulation in Fig. 3(c)). Inside the brackets are the shape of a residual block, and outside the brackets is the number of stacked blocks on a stage. "C=32" suggests grouped convolutions [24] with 32 groups. *The numbers of parameters and FLOPs are similar between these two models.*

可以看出，在进行分组后，其参数数量并没有太大的改变，不会真正地影响训练难度。

而在引入不同的 cardinality后，其可以取得如下的效果,可以看出，其在cardinality为32的时候取得了较好的效果：

	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

Table 3. Ablation experiments on ImageNet-1K. (**Top**): ResNet-50 with preserved complexity (~4.1 billion FLOPs); (**Bottom**): ResNet-101 with preserved complexity (~7.8 billion FLOPs). The error rate is evaluated on the single crop of 224×224 pixels.

卷积方式

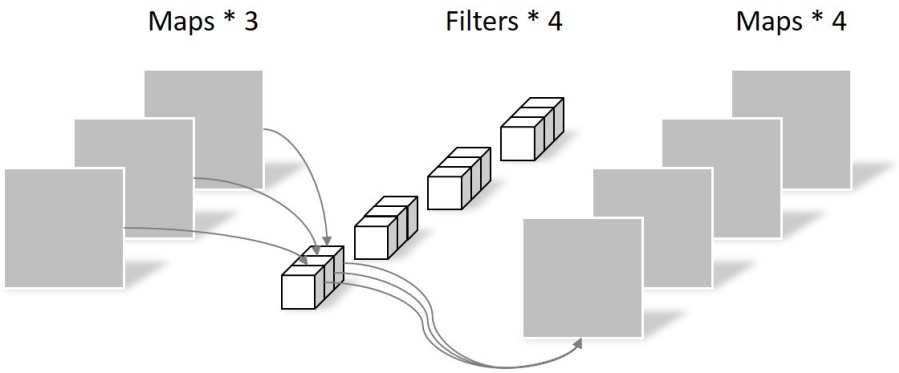
所谓卷积方式，指的主要就是在一个卷积层中的卷积运算步骤时，以哪种方式进行卷积运算，或者说，对输入层中的哪些层进行卷积。
即，针对某一个卷积层，其接受的输入为$28*28*N$，即其有N个通道，每一个通道的通道数为$28*28$，则对于常规的卷积方式，其应当接受K个$X*X*N$的卷积核$Conv_i$进行卷积操作，在这一情况下，其卷积是：

1. 每一个卷积层与其卷积核中对应的层进行卷积
2. 各层卷积进行加和

即，单个卷积核得到的结果应当只是特征图中的一层，而对于K个卷积层，其结果应当为一个有K个通道的Feature Map。

以上是常规的卷积方式，而对于`Depthwise separable convolutions`，他会针对N个层分别进行卷积但是对各层的卷积结果并不加和，一个$28*28*N$的卷积输入得到的结果应当是$X*X*N$的卷积输出，某一层的卷积结果即为输入层与卷积核在对应层的一次卷积运算的结果。

因此可知在`Depthwise Convolution`中，**卷积核的个数固定为输入的图像的层数，卷积核的channel固定为1；**即：



在这一情况下：`Depthwise Convolution`完成后的**Feature map数量与输入层的depth相同**，但是这种运算对输入层的每个channel独立进行卷积运算后就结束了。

Group Convolution则是对两者的折衷，即，它会对输入的通道进行分组后进行卷积。譬如针对$28*28*32$的输入层，将其分为8组进行卷积运算，则此时每一个卷积核应当由32个通道组成（也可以理解为由8个4通道的卷积核组成），在进行计算时，只针对某个组内的卷积结果进行加和，即在这一个实例中，每四个卷积输出层进行加和，因此总共可以得到8个单通道的卷积输出。

总结

与`Xception`类似，`ResNext`是对已有模型在"如何卷积"问题上的一种拓展，其希望使用一种比较自然的方式引入新的维度以及超参数而非在已有超参数上进行调整；实现性能的提升，而其各个`cardinality`的一致型使得其在计算并行性上优于Inception。

ResNext 的一些理解

残差网络的理解

源码