# 背景

因为之前的一些SOTA网络中运用了大量的1×1卷积操作，例如图1所示的在MobileNetV1中，1×1卷积占总计算量的94.86%，占总参数量的74.59%，可见1×1卷积导致计算量很大，这样的网络效率不高，本文提出了逐点分组卷积来减少1×1卷积带来的计算量。

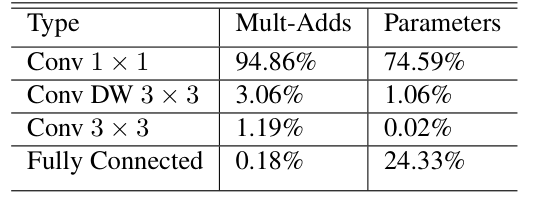


图 1

传统的分组卷积（如图2所示）存在的问题是，每一组卷积仅仅对输入通道的一部分进行特征映射，这就导致了每一组的输出仅与其输入有关，不同组之间没有任何的信息传递，导致最后学习的特征非常局限，因此提出了通道重排操作以此来增加特征表达能力。

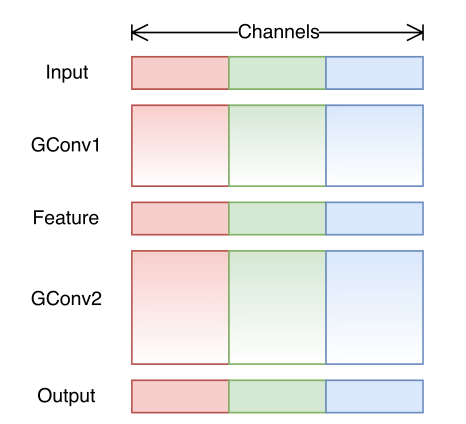


图 2

# 方法

## ①逐点分组卷积

分组卷积，就是将原始的特征图分成几组后再分别对每一组进行卷积，逐点分组卷积就是将分组卷积核逐点卷积结合起来。

### 为什么要采用分组卷积？

假设输入特征图大小为H×W×C，卷积核尺寸为K×K，c个卷积核，输出特征图尺寸为h×w×c。

那么使用普通卷积参数量为：

 (1)

计算量为：

 (2)

使用分组卷积，假设分4组，参数量为：

 (3)

计算量为：

 (4)

通过对比(1)、(3)以及(2)、(4)可以发现分组卷积的参数量和计算量都比普通卷积要小。

## ②通道重排

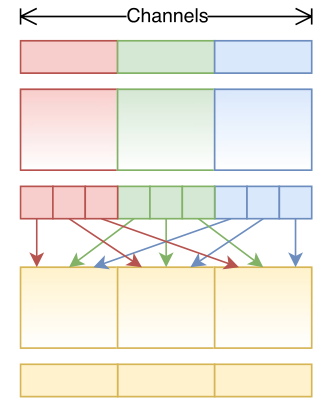


图 3

如图3所示，通道重排就是从分组卷积后得到的特征图中提取不同组的特征通道，然后将它们重新组合，其过程如下图所示（图片来源于博客）：

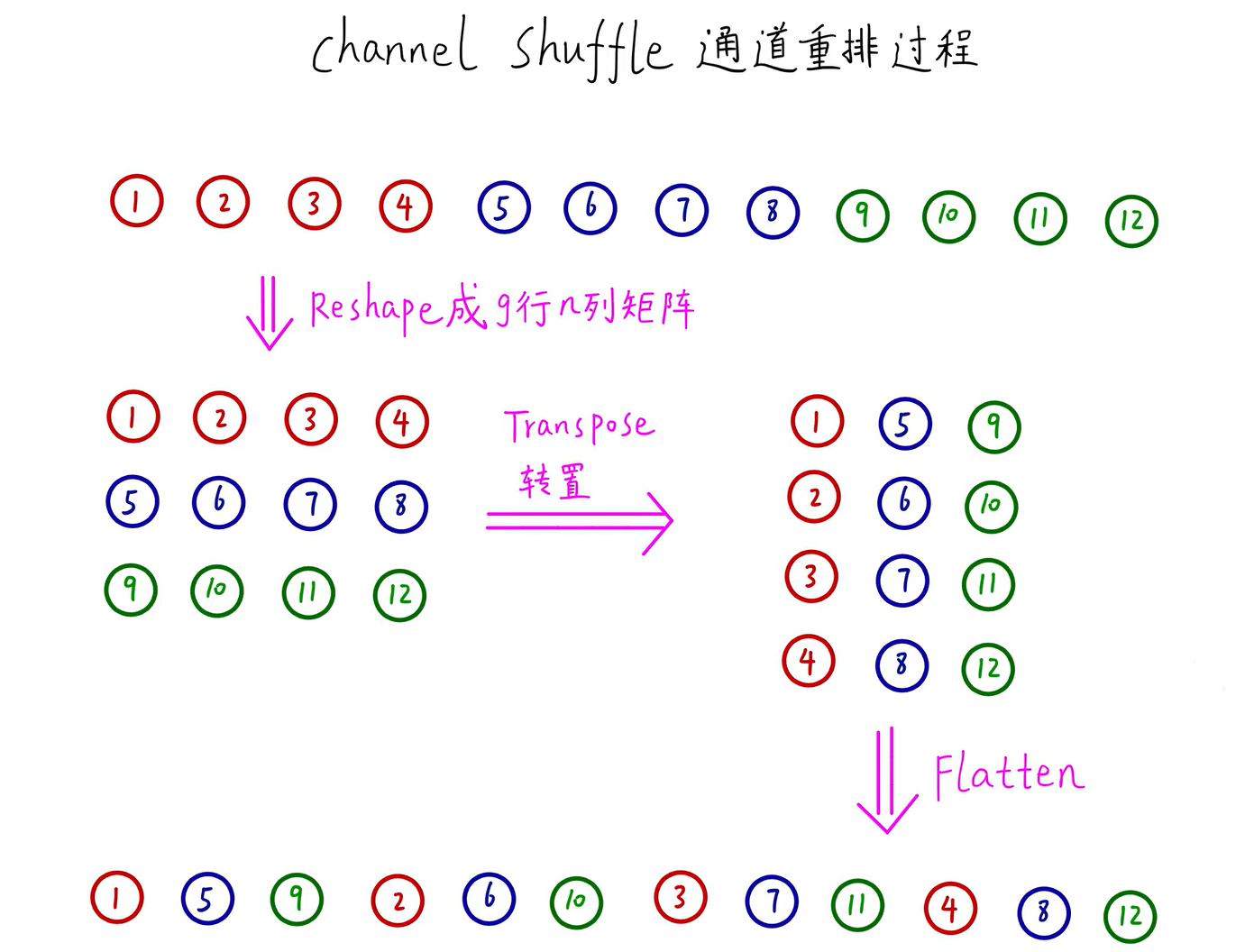


图 4

## ③单元结构

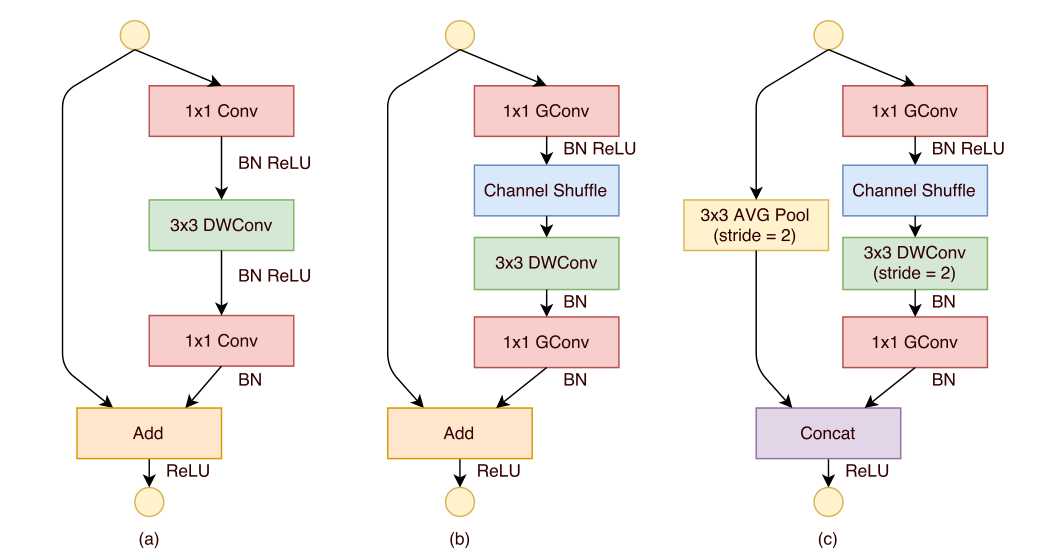


图 5

从图5可以看出ShuffleNet的单元结构的演变过程。图5(a)是ResNet中的瓶颈结构，然后使用逐点分组卷积将瓶颈结构中第一层的1×1卷积层代替掉，然后接了一层通道重排操作，中间的3×3深度方向的卷积操作保留，但是，注意，这里深度方向卷积后并没有使用ReLU激活函数，猜测可能的原因正如MobileNetV2中提到（图6所示），低维时使用ReLU会造成信息丢失，所以MobileNetV2采取的措施是倒残差结构，先将维度升高，避免信息丢失，而在这里是直接取消ReLU激活函数。

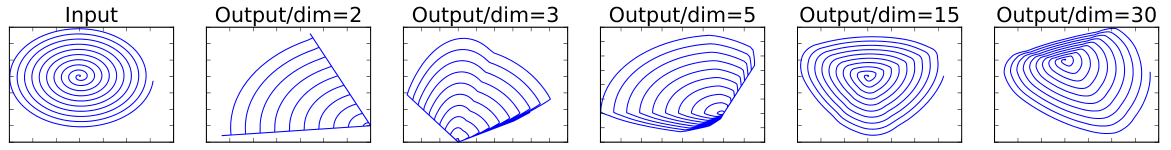


图 6

随后又将最后一层的1×1卷积替换为逐点分组卷积，注意，在这一层逐点分组卷积后没有使用通道重排操作是因为此时的效果已经很可观了。

那么图5(c)的结构和(b)的主要区别就是在跳跃连接分支中采用了3×3，步长为2的平均池化、主分支中深度方向的卷积步长变成了2，并且最后使用的是concat拼接而不是逐像素相加，这样的结构会使输入图像尺寸减半，通道数翻倍。

# 疑问

对单元结构中最后将add操作换为concat拼接有一些疑问。

自己在查资料的过程中发现

add是描述图像的特征下的信息量增多了，但是描述图像的维度本身并没有增加，只是每一维下的信息量在增加，这显然是对最终的图像的分类是有益的。而concatenate是通道数的合并，也就是说描述图像本身的特征数（通道数）增加了，而每一特征下的信息是没有增加，并且add的计算量要比concat的计算量小得多（[内容来源](https://blog.csdn.net/baidu_38172402/article/details/106157861?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522167229729016800184160742%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=167229729016800184160742&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-3-106157861-null-null.142%5ev68%5econtrol,201%5ev4%5eadd_ask,213%5ev2%5et3_control2&utm_term=element-wise%20%E7%9B%B8%E5%8A%A0%E5%92%8Cconcatenation&spm=1018.2226.3001.4187)）

那么为什么偏偏要增加通道数呢？每个特征下的信息又没有增加，并且相比add来说计算量还增加了，这样做是为什么呢？

# 补充

对“疑问部分”的补充：

Add是融合，concate是并存，concate后再加一层卷积就是融合了，所以它们的本质区别就是add是直接相加融合，concate是加权融合（卷积的计算就是在做可学习地加权融合），那么到底该选择哪种呢？其实没有绝对的定论，都是尝试两种，然后选择效果最好的那一种。

自己在选择的时候根据它们的区别和任务的特点来进行选择即可。