综述：轻量级CNN架构设计

极市平台 2022-02-04 22:00:00 手机阅读

以下文章来源于GiantPandaCV ，作者Ironboy

作者丨Ironboy

来源丨GiantPandaCV

编辑丨极市平台

极市导读

本文作者结合论文和项目比赛的经验，讲述了轻量级CNN的发展以及设计总结。内容包含基本概念、卷积计算类型、其他算子、常用激活函数、经典轻量化模型等。>>加入极市CV技术交流群，走在计算机视觉的最前沿

卷积神经网络架构设计，又指backbone设计，主要是根据具体任务的数据集特点以及相关的评价指标来确定一个网络结构的**输入图像分辨率，深度，每一层宽度，拓扑结构**等细节。

公开发表的论文大多都是基于ImageNet这种大型的公开数据集来进行的通用结构设计，早期只以其分类精度来证明设计的优劣，后来也慢慢开始对比参数量(Params)和计算量(FLOPs)，由于ImageNet的数据量十分巨大且丰富，所以通常在该数据集上获得很好精度的网络结构泛化到其他任务性能也都不会差。

但在很多特定任务中，这种通用的结构虽然效果还可以，却并不算最好，所以**一般在实际应用时通常是基于已公开发表的优秀网络结构再根据任务特点进行适当修改得到自己需要的模型结**

构。

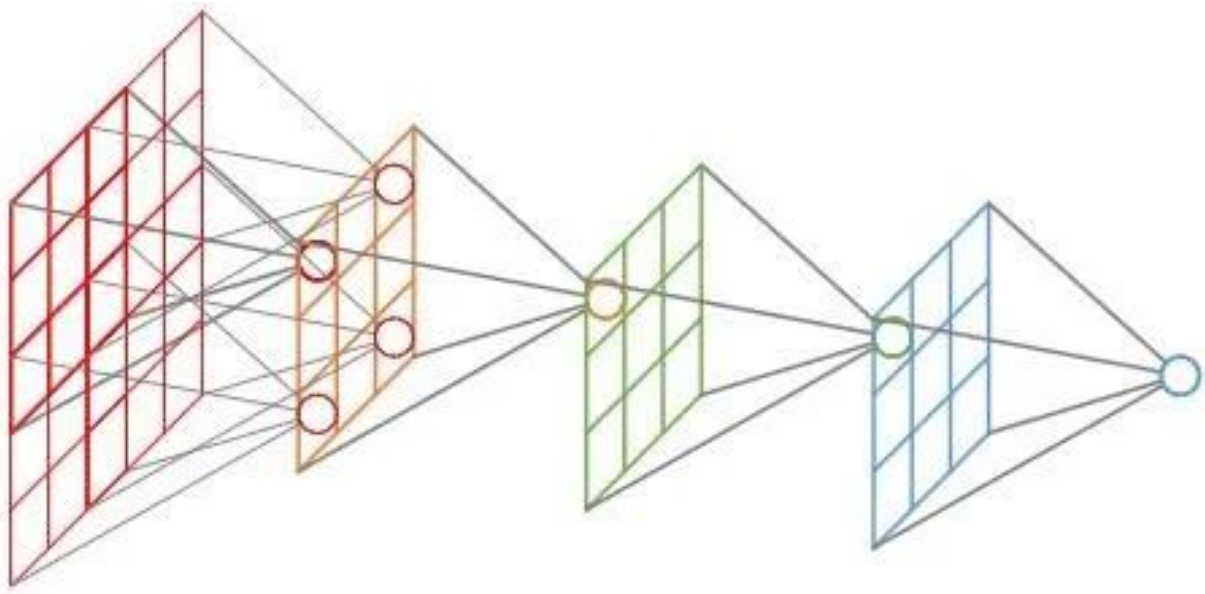
目前人工智能技术应用的一个趋势是在端侧平台上部署高性能的神经网络模型并能在真实场景中实时(大于30帧)运行，如移动端/嵌入式端设备。这些平台的特点是内存资源少，处理器性能不高，功耗受限，这使得目前精度最高的模型由于对内存和计算资源的超额要求使得根本无法在上面部署且达到实时性的要求。虽然可以通过知识蒸馏，通道剪枝，低比特量化等一系列手段来降低模型参数量和计算量，但仍然远远不够，且在精度和帧率之间各种trade-off也非常繁琐。所以直接设计轻量级的架构，然后结合剪枝量化是最有效的解决办法。

本文将结合自己看的论文和参加项目比赛的经验讲述轻量级CNN的发展以及一些设计总结，如有不对之处请不吝赐教。

基本概念

- **感受野(Receptive Field)**

感受野指的是卷积神经网络每一层输出的特征图(feature map)上每个像素点映射回输入图像上的区域大小，神经元感受野的范围越大表示其能接触到的原始图像范围就越大，也意味着它能学习更为全局，语义层次更高的特征信息，相反，范围越小则表示其所包含的特征越趋向局部和细节。因此感受野的范围可以用来大致判断每一层的抽象层次，并且我们可以很明显地知道网络越深，神经元的感受野越大。



感受野

- **分辨率(Resolution)**

分辨率指的是输入模型的图像尺寸，即长宽大小。通常情况会根据模型下采样次数n和最后一次下采样后feature map的分辨率k×k来决定输入分辨率的大小，即：

$$r = k \times 2^n$$

从输入r×r到最后一个卷积特征feature map的k×k，整个过程是一个信息逐渐抽象化的过程，即网络学习到的信息逐渐由低级的几何信息转变为高级的语义信息，这个feature map的大小可以是3×3，5×5，7×7，9×9等等，k太大会增加后续的计算量且信息抽象层次不够高，影响网络性能，k太小会造成非常严重的信息丢失，如原始分辨率映射到最后一层的feature map有效区域可能不到一个像素点，使得训练无法收敛。

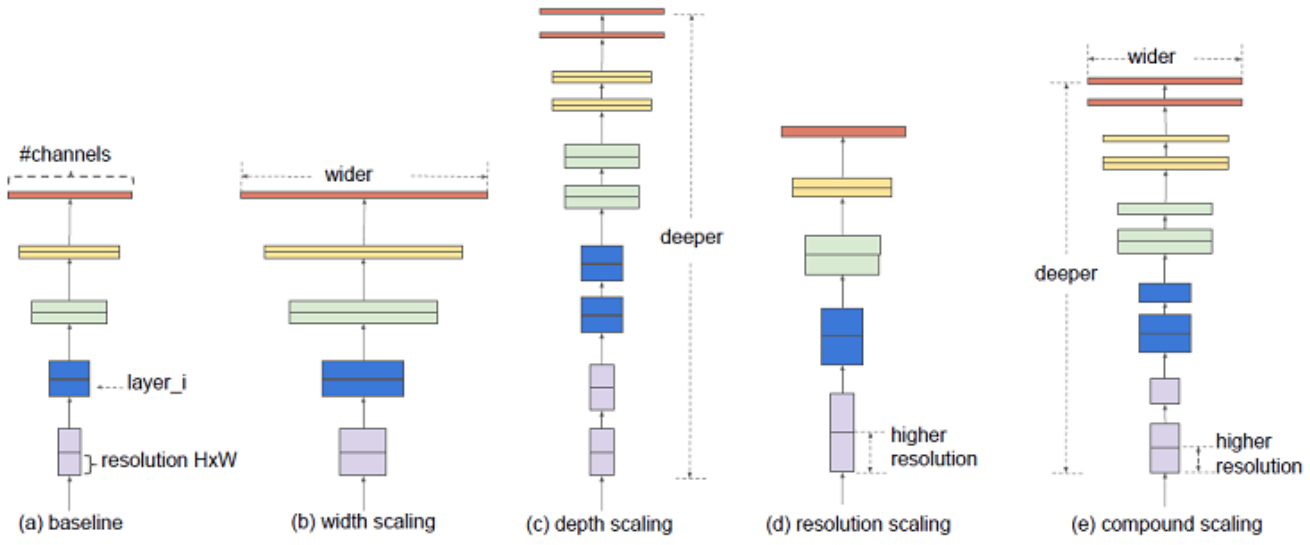
在ImageNet分类任务中，通常设置的5次下采样，并且考虑到其原始图像大多数在300分辨率左右，所以把最后一个卷积特征大小设定为7×7，将输入尺寸固定为224×224×3。在目标检测任务中，很多采用的是416×416×3的输入尺寸，当然由于很多目标检测模型是全卷积的结构，通常可以使用多尺寸训练的方式，即每次输入只需要保证是32×的图像尺寸大小就行，不固定具体数值。但这种多尺度训练的方式在图像分类当中是不通用的，因为分类模型最后一层是全连接结构，即矩阵乘法，需要固定输入数据的维度。

- **深度(Depth)**

神经网络的深度决定了网络的表达能力，它有两种计算方法，早期的backbone设计都是直接使用卷积层堆叠的方式，它的深度即神经网络的层数，后来的backbone设计采用了更高效的module(或block)堆叠的方式，每个module是由多个卷积层组成，它的深度也可以指module的个数，这种说法在神经架构搜索(NAS)中出现的更为频繁。通常而言网络越深表达能力越强，但深度大于某个值可能会带来相反的效果，所以它的具体设定需要不断调参得到。

- **宽度(Width)**

宽度决定了网络在某一层学到的信息量，但网络的宽度时指的是卷积神经网络中最大的通道数，由卷积核数量最多的层决定。通常的结构设计中卷积核的数量随着层数越来越多的，直到最后一层feature map达到最大，这是因为越到深层，feature map的分辨率越小，所包含的信息越高级，所以需要更多的卷积核来进行学习。通道越多效果越好，但带来的计算量也会大大增加，所以具体设定也是一个调参的过程，并且各层通道数会按照8×的倍数来确定，这样有利于GPU的并行计算。



width,depth and resolution

- **下采样(Down-Sample)**

下采样层有两个作用，一是减少计算量，防止过拟合，二是增大感受野，使得后面的卷积核能够学到更加全局的信息。下采样的设计有两种：

1.采用stride为2的池化层，如Max-pooling或Average-pooling，目前通常使用Max-pooling，因为它计算简单且最大响应能更好保留纹理特征；

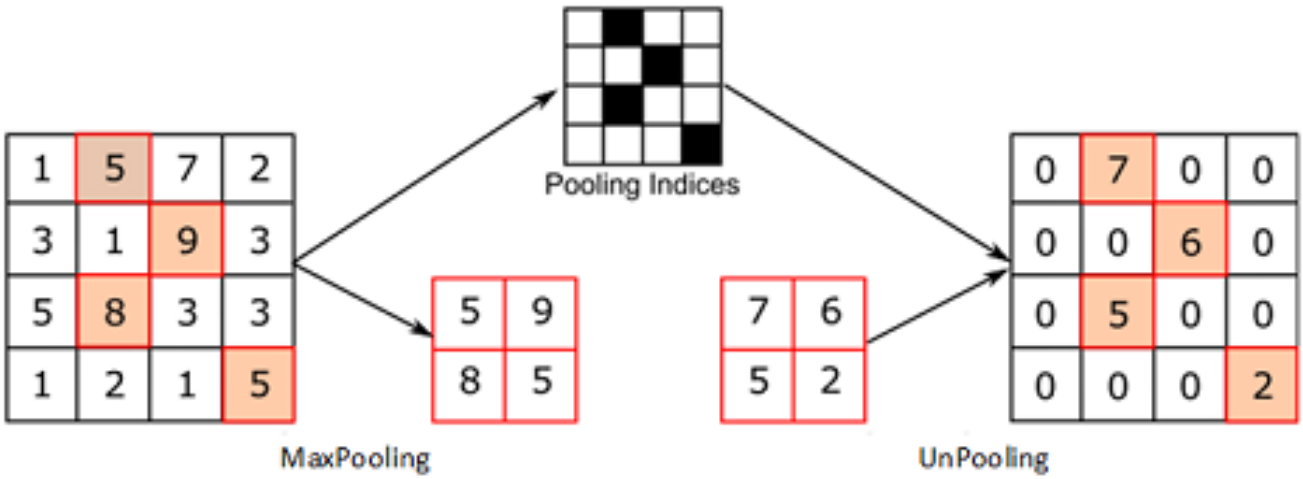
2.采用stride为2的卷积层，下采样的过程是一个信息损失的过程，而池化层是不可学习的，用stride为2的可学习卷积层来代替pooling可以得到更好的效果，当然同时也增加了一定的计算量。

(突然想到为啥不使用双线性插值，向下插值来代替Pooling，这个虽然比MaxPooling计算量更大，但是保留的信息应该更丰富才是)

- **上采样(Up-Sampling)**

在卷积神经网络中，由于输入图像通过卷积神经网络(CNN)提取特征后，输出的尺寸往往会变小，而有时我们需要将图像恢复到原来的尺寸以便进行进一步的计算(如图像的语义分割)，这个使图像由小分辨率映射到大分辨率的操作，叫做上采样，它的实现一般有三种方式：

- 插值，一般使用的是双线性插值，因为效果最好，虽然计算上比其他插值方式复杂，但是相对于卷积计算可以说不值一提；
- 转置卷积又或是说反卷积，通过对输入feature map间隔填充0，再进行标准的卷积计算，可以使得输出feature map的尺寸比输入更大；
- Max Unpooling，在对称的max pooling位置记录最大值的索引位置，然后在unpooling阶段时将对应的值放置到原先最大值位置，其余位置补0；



Max Unpooling

- **参数量(Params)**

参数量指的网络中可学习变量的数量，包括卷积核的权重weight，批归一化(BN)的缩放系数γ，偏移系数β，有些没有BN的层可能有偏置bias，这些都是可学习的参数 ，即在模型训练开

始前被赋予初值，在训练过程根据链式法则中不断迭代更新，整个模型的参数量主要由卷积核的权重weight的数量决定，参数量越大，则该结构对运行平台的内存要求越高，参数量的大小是轻量化网络设计的一个重要评价指标。

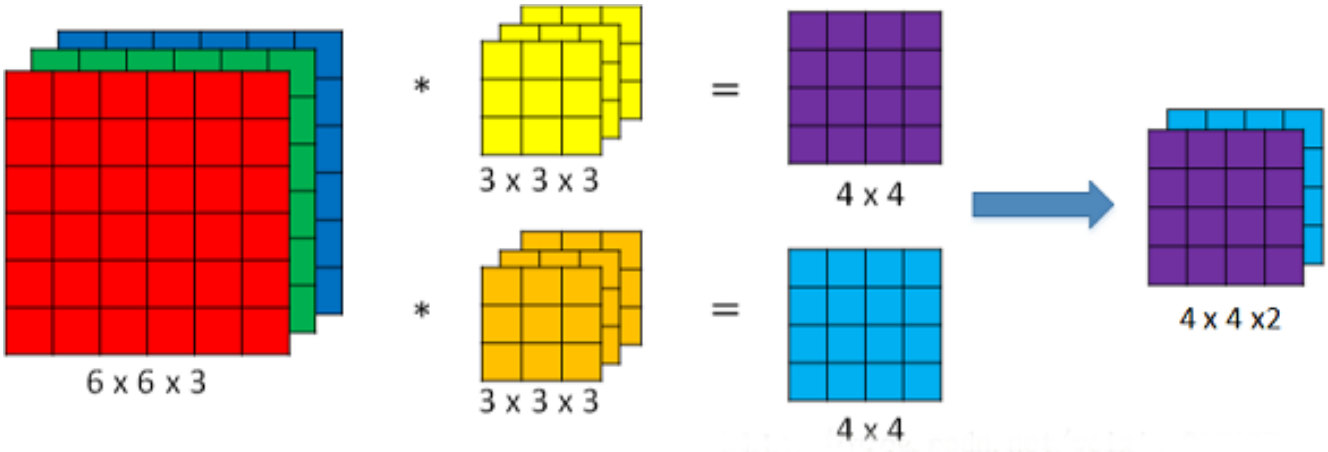
- **计算量(FLOPs)**

神经网络的前向推理过程基本上都是乘累加计算，所以它的计算量也是指的前向推理过程中乘加运算的次数，通常用FLOPs来表示，即floating point operations(浮点运算数)。计算量越大，在同一平台上模型运行延时越长，尤其是在移动端/嵌入式这种资源受限的平台上想要达到实时性的要求就必须要求模型的计算量尽可能地低，但这个不是严格成正比关系，也跟具体算子的计算密集程度(即计算时间与IO时间占比)和该算子底层优化的程度有关。

卷积计算类型

- **标准卷积 (Convolution)**

在神经网络架构设计中，标准卷积是最常见的结构，假设其输入feature map的维度是(1, iC, iH, iW)，每个卷积核的维度是(1, iC, k, k)，一次卷积滤波得到一层feature map的维度为(1,1, oH, oW)，一共有oC个卷积核，则输出feature map的维度是(1, oC, oH, oW)，计算量为iC×k×k×oC×oH×oW，计算过程如下图所示，由于太过基础，故不赘述。

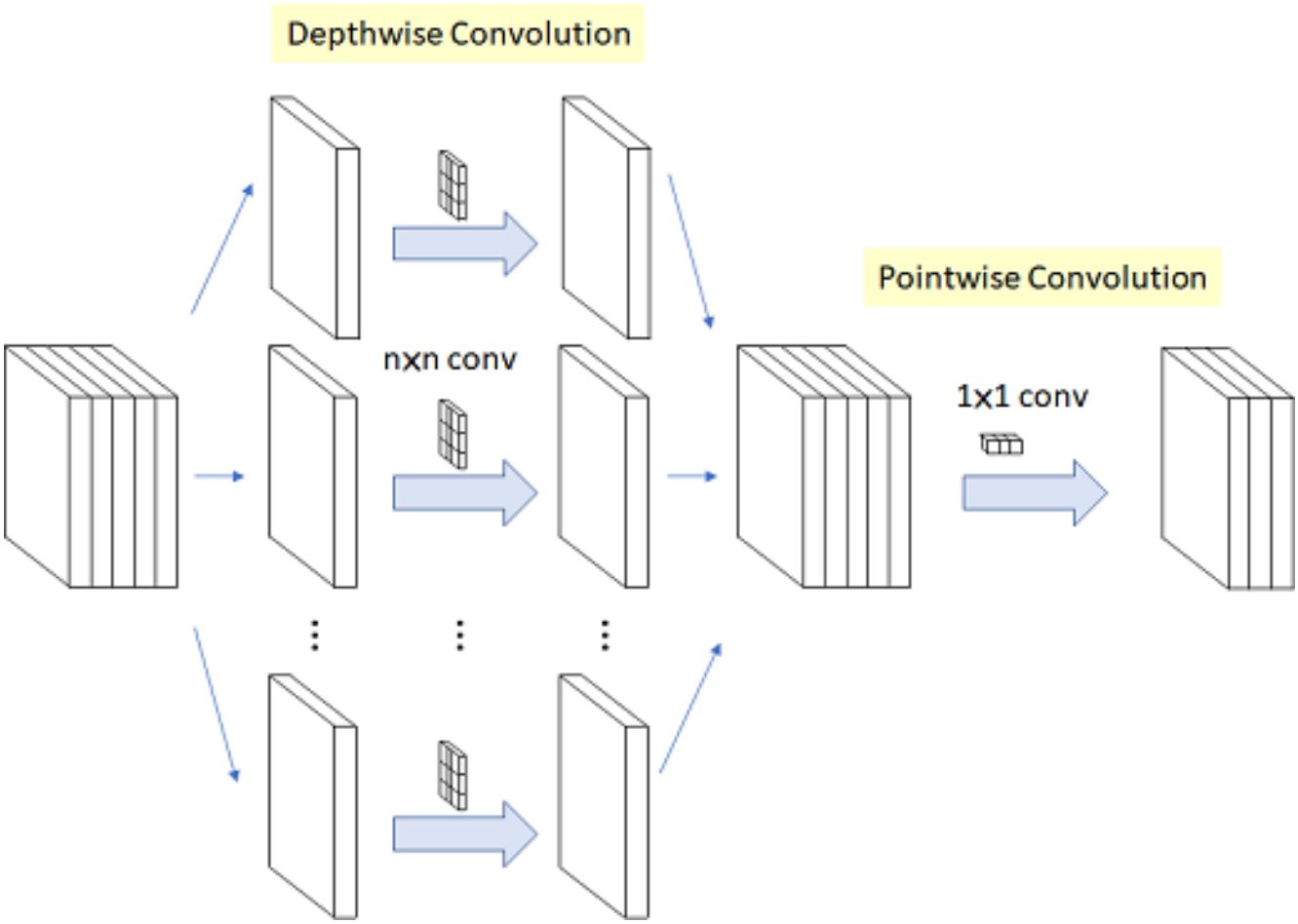


标准卷积

- **深度卷积 (Depthwise Convolution)**

深度卷积与标准卷积相比，顾名思义是在深度上做了文章，而这里的深度跟网络的深度无关，它指的通道，标准卷积中每个卷积核都需要与feature map的所有层进行计算，所以每个卷积核的通道数等于输入feature map的通道数，通过设定卷积核的数量可以控制输出feature map的通道数。而深度卷积每个卷积核都是单通道的，维度为(1,1,k,k) ，卷积核的个数为iC，即第i个卷积核与feature map第i个通道进行二维的卷积计算，最后输出维度为(1,iC,oH,oW)，它不能改变输出feature map的通道数，所以通常会在深度卷积后面接上一个(oC,iC,1,1)的标准卷积来代替3×3或更大尺寸的标准卷积，总的计算量为k×k×iC×oH×oW+iC×1×1×oH×oW×oC，是普通卷积的1/oC+1/(k×k)，大大减少了计算量和参数量，又可以达到相同的效果，这种结构被称

为深度可分离卷积(Depthwise Separable Convolution)，在MobileNet V1被提出，后来渐渐成为轻量化结构设计的标配。

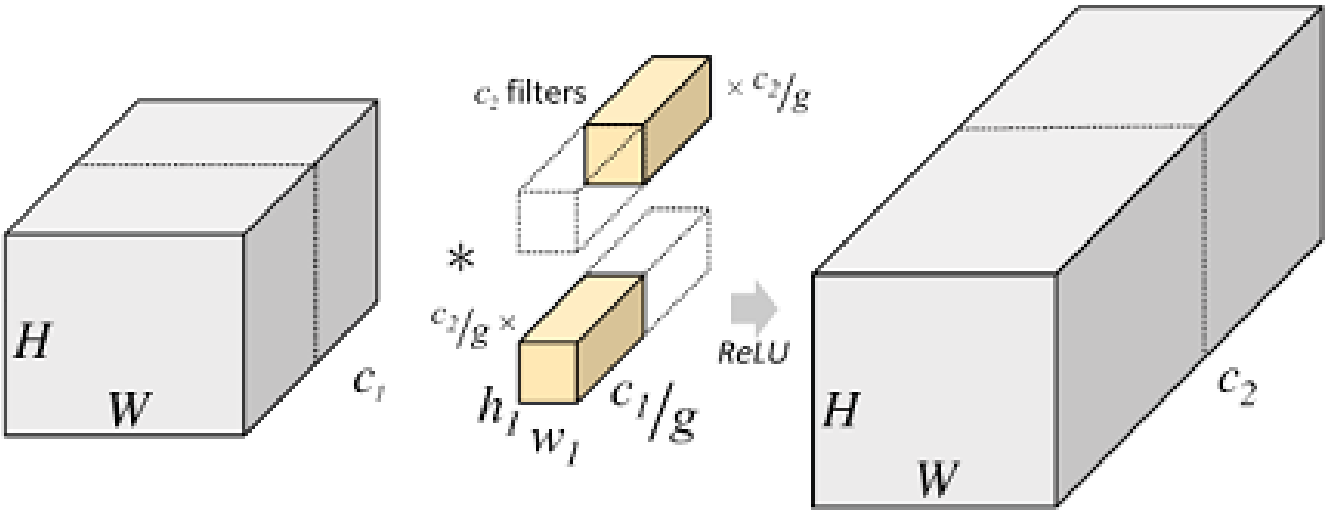


深度可分离卷积

深度卷积之前一直被吐槽在GPU上运行速度还不如一般的标准卷积，因为depthwise 的卷积核复用率比普通卷积要小很多，计算和内存访问的比值比普通卷积更小，因此会花更多时间在内存开销上，而且per-channel的矩阵计算很小不容易并行导致的更慢，但理论上计算量和参数量都是大大减少的，只是底层优化的问题。

- **分组卷积 (Group Convolution)**

分组卷积最早在AlexNet中出现，当时作者在训练模型时为了减少显存占用而将feature map分组然后给多个GPU进行处理，最后把多个输出进行融合。具体计算过程是，分组卷积首先将输入feature map分成g个组，每个组的大小为(1, iC/g, iH, iW)，对应每组中一个卷积核的大小是(1,iC/g,k,k)，每组有oC/g个卷积核，所以每组输出feature map的尺寸为(1,oC/g,oH,oW)，最终g组输出拼接得到一个(1,oC,oH,oW)的大feature map，总的计算量为iC/g×k×k×oC×oH×oW，是标准卷积的1/g，参数量也是标准卷积的1/g。

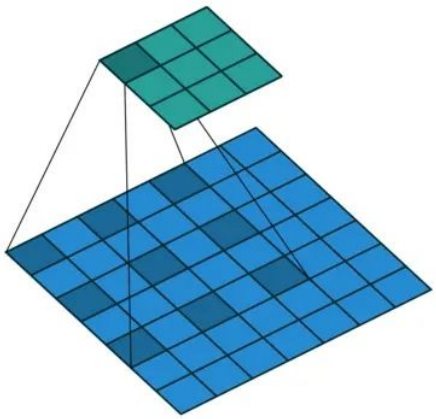


分组卷积

但由于feature map组与组之间相互独立，存在信息的阻隔，所以ShuffleNet提出对输出的feature map做一次channel shuffle的操作，即通道混洗，打乱原先的顺序，使得各个组之间的信息能够交互起来。

- **空洞卷积 (Dilated Convolution)**

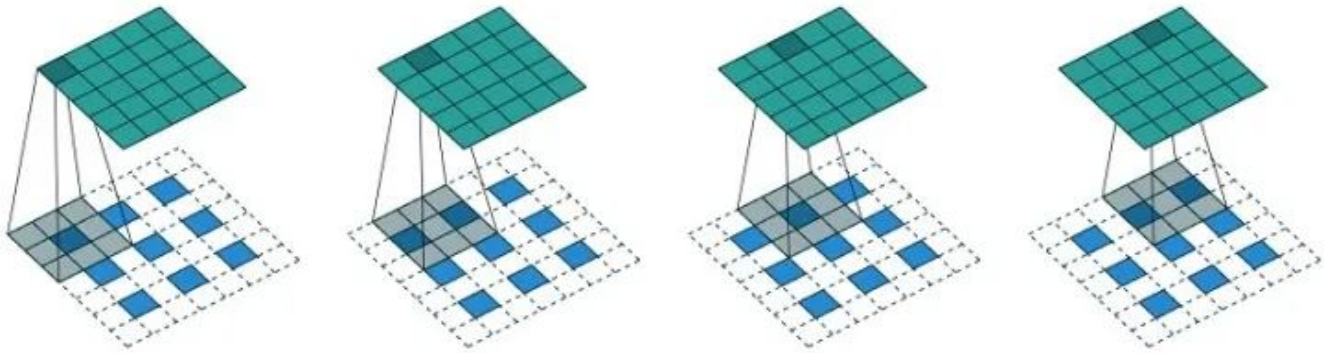
空洞卷积是针对图像语义分割问题中下采样会降低图像分辨率、丢失信息而提出的一种卷积思路。通过间隔取值扩大感受野，让原本3x3的卷积核，在相同参数量和计算量下拥有更大的感受野。这里面有个扩张率(dilation rate)的系数，这个系数定义了这个间隔的大小，标准卷积相当于dilation rate为1的空洞卷积，下图展示的是dilation rate为2的空洞卷积计算过程，可以看出3×3的卷积核可以感知标准的5×5卷积核的范围，还有一种理解思路就是先对3×3的卷积核间隔补0，使它变成5×5的卷积，然后再执行标准卷积的操作。



空洞卷积

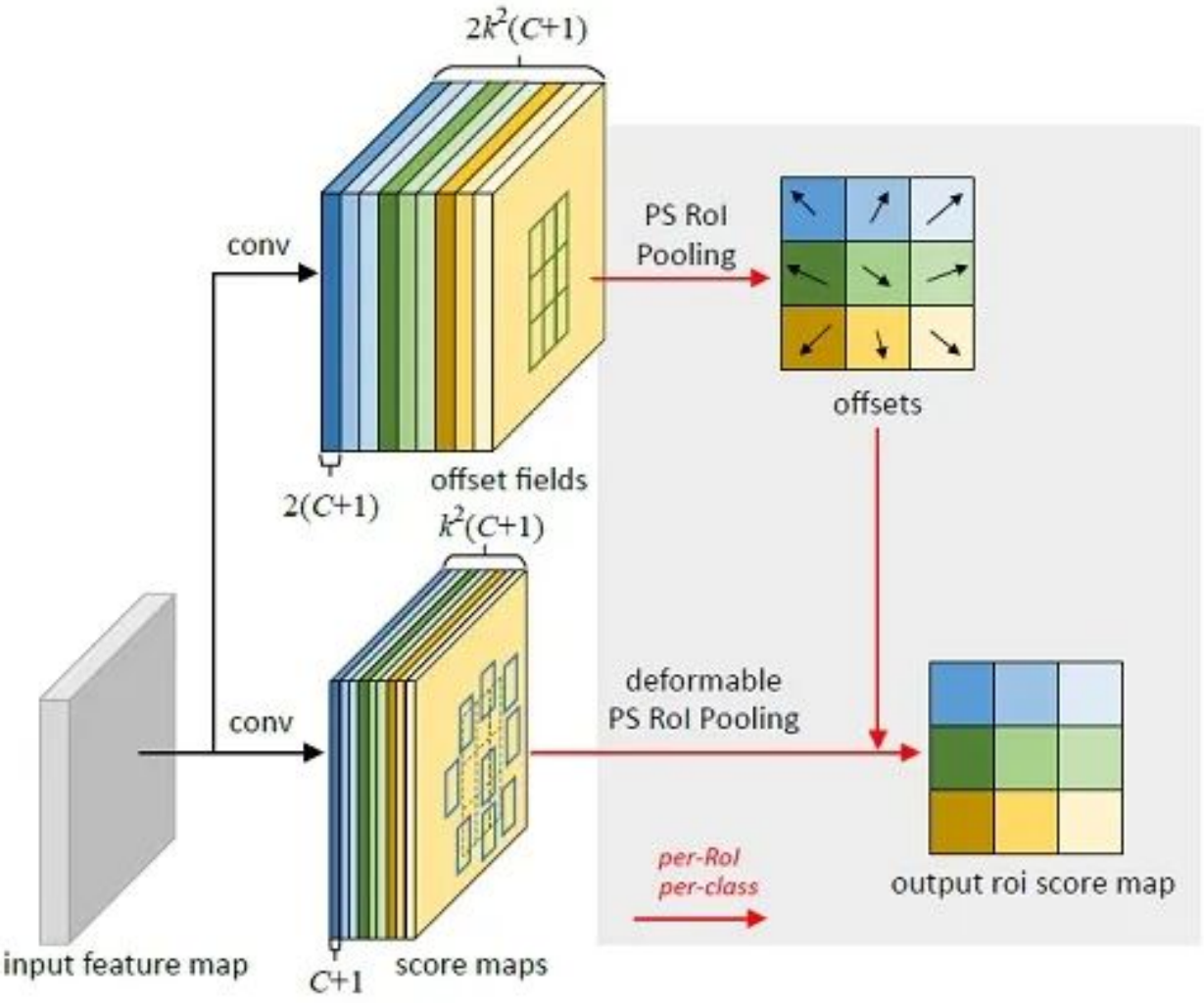
- **转置卷积 (Transposed Convolutions)**

转置卷积又称反卷积(Deconvolution)，它和空洞卷积的思路正好相反，是为上采样而生，也应用于语义分割当中，而且他的计算也和空洞卷积正好相反，先对输入的feature map间隔补0，卷积核不变，然后使用标准的卷积进行计算，得到更大尺寸的feature map。



- **可变形卷积 (deformable convolution)**

以上的卷积计算都是固定的，每次输入不同的图像数据，卷积计算的位置都是完全固定不变，即使是空洞卷积/转置卷积，0填充的位置也都是事先确定的。而可变性卷积是指卷积核上对每一个元素额外增加了一个h和w方向上偏移的参数，然后根据这个偏移在feature map上动态取点来进行卷积计算，这样卷积核就能在训练过程中扩展到很大的范围。而显而易见的是可变性卷积虽然比其他卷积方式更加灵活，可以根据每张输入图片感知不同位置的信息，类似于注意力，从而达到更好的效果，但是它比可行变卷积在增加了很多计算量和实现难度，目前感觉只在GPU上优化的很好，在其他平台上还没有见到部署。

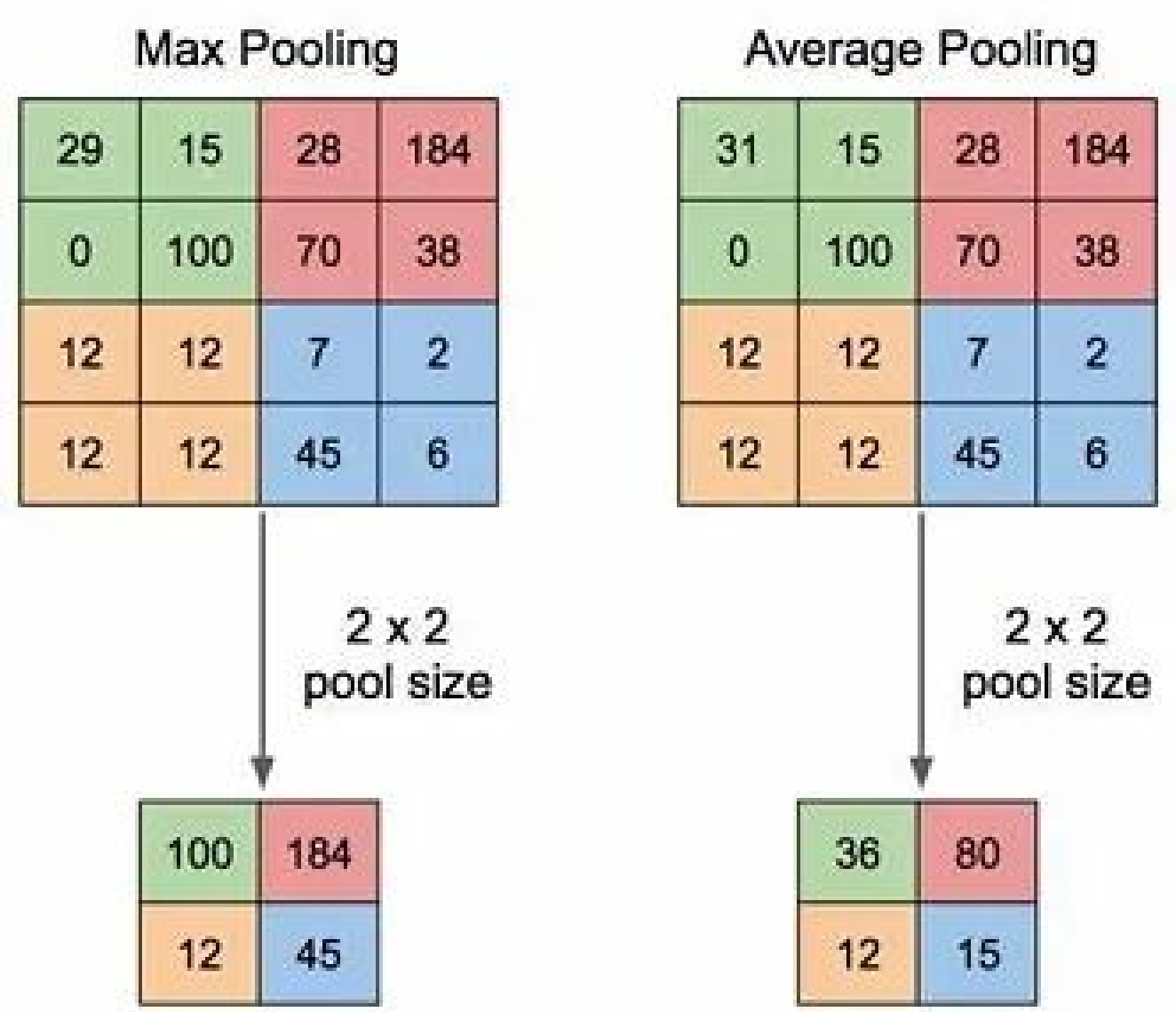


其他算子

- **池化(pooling)**

池化这个操作比较简单，一般在上采样和下采样的时候用到，没有参数，不可学习，但操作极为简单，和depthwise卷积类似，只是把乘累加操作替换成取最大/取平均操作。

- **最大池化和平均池化**



最大池化和平均池化

- **全局平均池化**

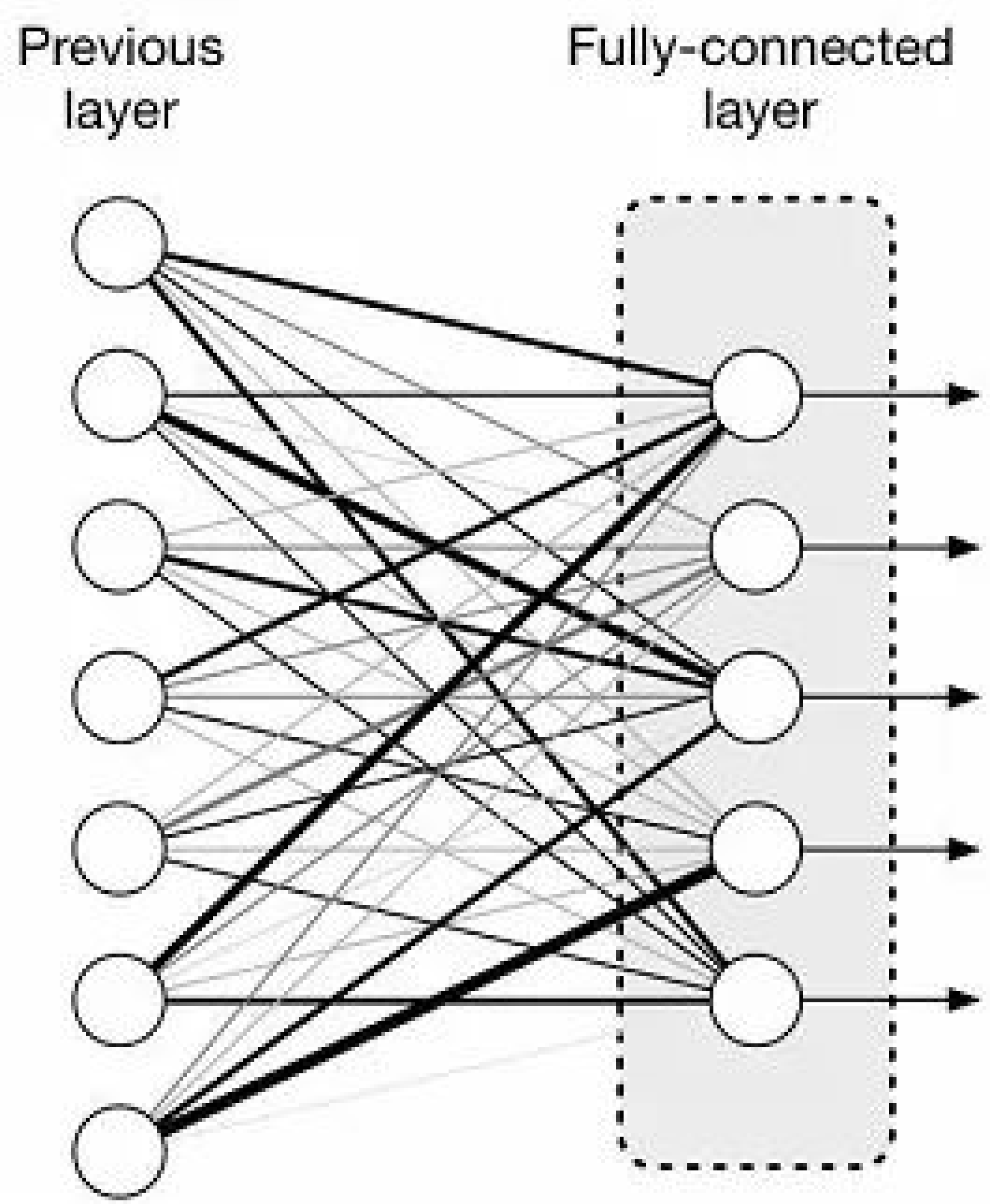
全局平均池化的操作是对一个维度为(C,H,W)的feature map，在HW方向整个取平均，然后输出一个长度为C的向量，这个操作一般在分类模型的最后一个feature map之后出现，然后接一个全连接层就可以完成分类结果的输出了。早期的分类模型都是把最后一个feature map直接拉平成C×H×W的向量，然后再接全连接层，但是显然可以看出来这个计算量极大，甚至有的模型最后一个全连接层占了整个模型计算量的50%以上，之后由研究人员发现对这个feature map做一个全局平均池化，然后再加全连接层可以达到相似的效果，且计算量降低到了原来的1/HW。

- **最大向上池化**

这个操作在前面基本概念一节上采样段落中有描述，故不赘述。

- **全连接计算(Full Connected)**

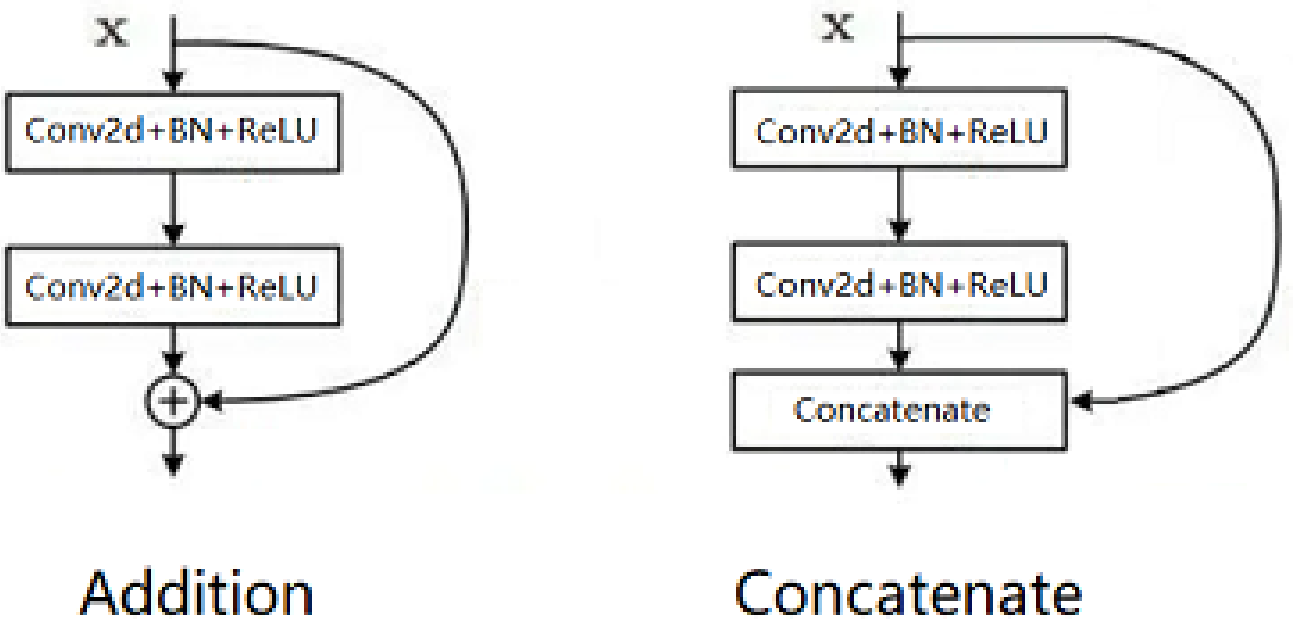
这个本质其实就是矩阵乘法，输入一个(B, iC)的数据，权重为(iC, oC)，那么输出为(B, oC)，在多层感知机和分类模型最后一层常常见到。



全连接结构

- **Addition / Concatenate分支**

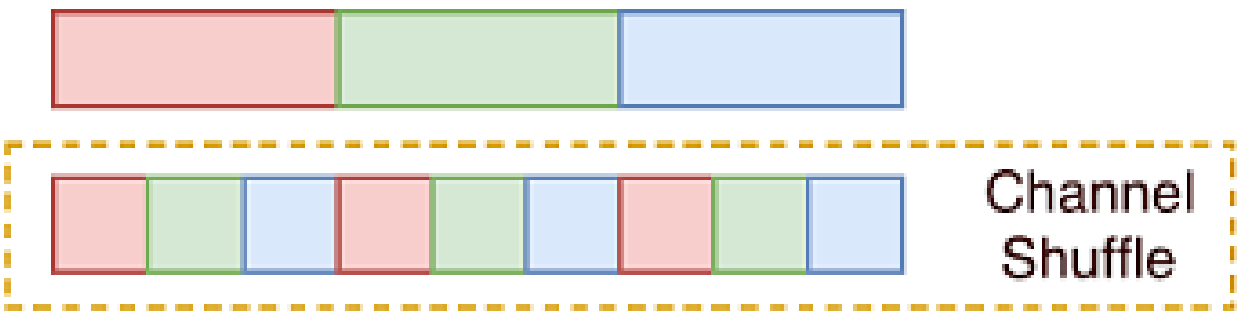
Addition和Concatenate分支操作统称为shortcut，如下图所示，操作极为简单。Addition是在ResNet中提出，两个相同维度的feature map相同位置点的值直接相加，得到新的相同维度feature map，这个操作可以融合之前的特征，增加信息的表达，Concatenate操作是在Inception中首次使用，被DenseNet发扬光大，和addition不同的是，它只要求两个feature map的HW相同，通道数可以不同，然后两个feature map在通道上直接拼接，得到一个更大的feature map，它保留了一些原始的特征，增加了特征的数量，使得有效的信息流继续向后传递。



Add & Concat

- **Channel shuffle**

channel shuffle是ShuffleNet中首次提出，主要是针对分组卷积中不同组之间信息不流通，对不同组的feature map进行混洗的一个操作，如下图所示，假设原始的feature map维度为(1,9,H,W)，被分成了3个组，每个组有三个通道，那么首先将这个feature map进行reshape操作，得到(1,3,3,H,W)，然后对中间的两个大小为3的维度进行转置，依然是(1,3,3,H,W)，最后将通道拉平，变回(1,9,H,W)，就完成了通道混洗，使得不同组的feature map间隔保存，增强了信息的交互。



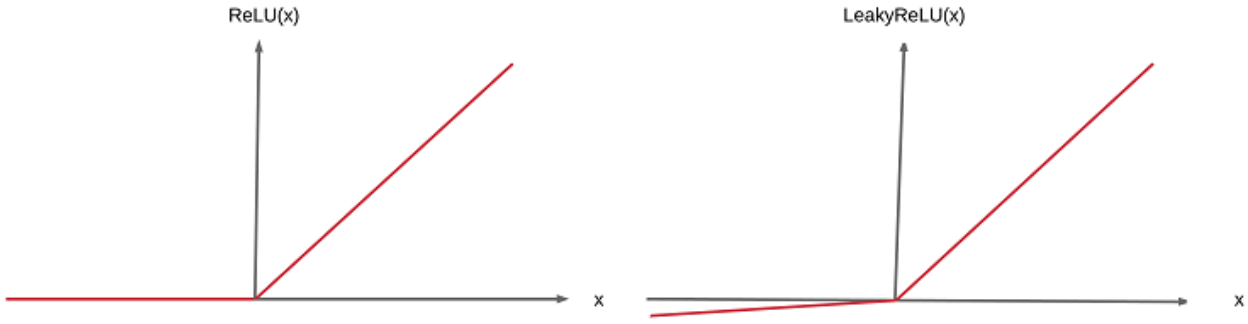
channel shuffle

常用激活函数

激活函数的非线性是神经网络发挥作用最重要的因素之一，而对于实际部署，激活函数的实现也是很重要的一个方面，实现的不好对加速效果影响很大，这里主要讲几个部署当中常见的激活函数。

- **ReLU系列**

这里主要指常用的ReLU，ReLU6和leaky ReLU。ReLU比较好部署，小于0的部分为0，大于0的部分为原始值，只需要判断一下符号位就行；ReLU6与ReLU相比也只是在正向部分多了个阈值，大于6的值等于6，在实现时多了个比较也不算麻烦；而leaky ReLU和ReLU正向部分一样，都是大于0等于原始值，但负向部分却是等于原始值的1/10，浮点运算的话乘个0.1就好了，如果因为量化要实现整数运算，这块可以做个近似，如0.1用13>>7来代替，具体实现方法多种多样 ，还算简单。



ReLU & LeakyReLU

- **Sigmoid系列**

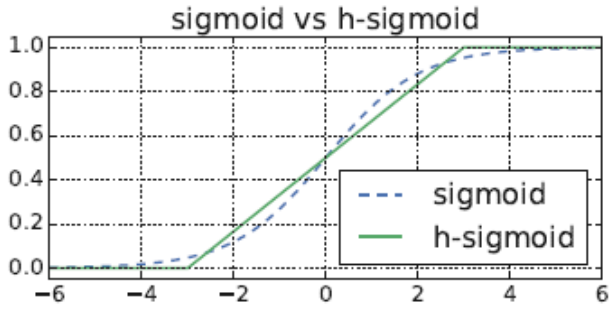
这里主要指sigmoid，还有和他相关的swish:

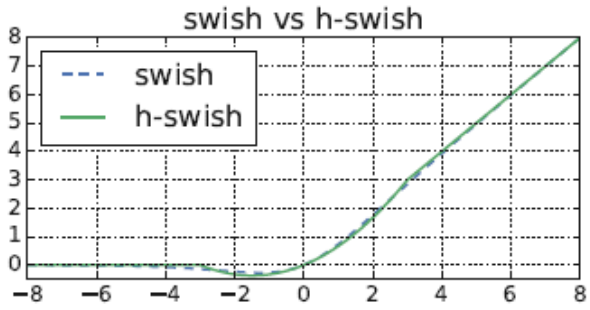
$$\text{sigmoid}(x) = \frac{1}{1+\exp -x} \text{swish} = x \times \text{sigmoid}(x)$$

可以看出，如果按照公式来实现sigmoid对低性能的硬件来说非常不友好，因为涉及到大量的exp指数运算和除法运算，于是有研究人员针对此专门设计了近似的硬件友好的函数h-sigmoid和h-swish函数，这里的h指的就是hardware的意思：

$$\text{Hsigmoid}(x) = \max(0, \min(1, \frac{x+1}{2})) \text{Hswish} = x \frac{\text{ReLU6}(x+3)}{6}$$

可视化的对比如下图所示，可以看出在保证精度的同时又能大大方便硬件的实现，当然要直接实现sigmoid也是可以的，毕竟sigmoid是有限输出，当输入小于-8或大于8的时候，输出基本上接近于-1和1，可以根据这个特点设计一个查找表，速度也超快，且我们实测对精度没啥影响。





经典轻量化模型

早期比较经典的卷积神经网络，如AlexNet，VGG，GoogleNet(或Inception)，ResNet，DenseNet都是以提升模型在ImageNet数据集上的分类精度为主了，很少考虑参数量和计算量的问题，他们的主要结构解析起来也比较简单，基本都是由标准卷积(7×7，5×5，3×3和1×1)，Pooling和shortcut操作(Addition / Concatenate)构成，而且以3×3及其以上的卷积核为主，通道数也是动辄上千，所以参数量和计算量巨大。后续研究人员慢慢发现两个3×3卷积可以代替一个5×5卷积的效果，三个3×3卷积可以代替一个7×7卷积的效果，大量使用1×1卷积，使用3×3 depthwise conv + pointwise conv(1×1标准卷积)可以代替3×3普通卷积.....一系列操作可以减少参数量和计算量，所以下面讲述一下一些轻量级神经网络发展的历史，因为这块很多人都讲过，所以我会简单一些，挑重点说说。

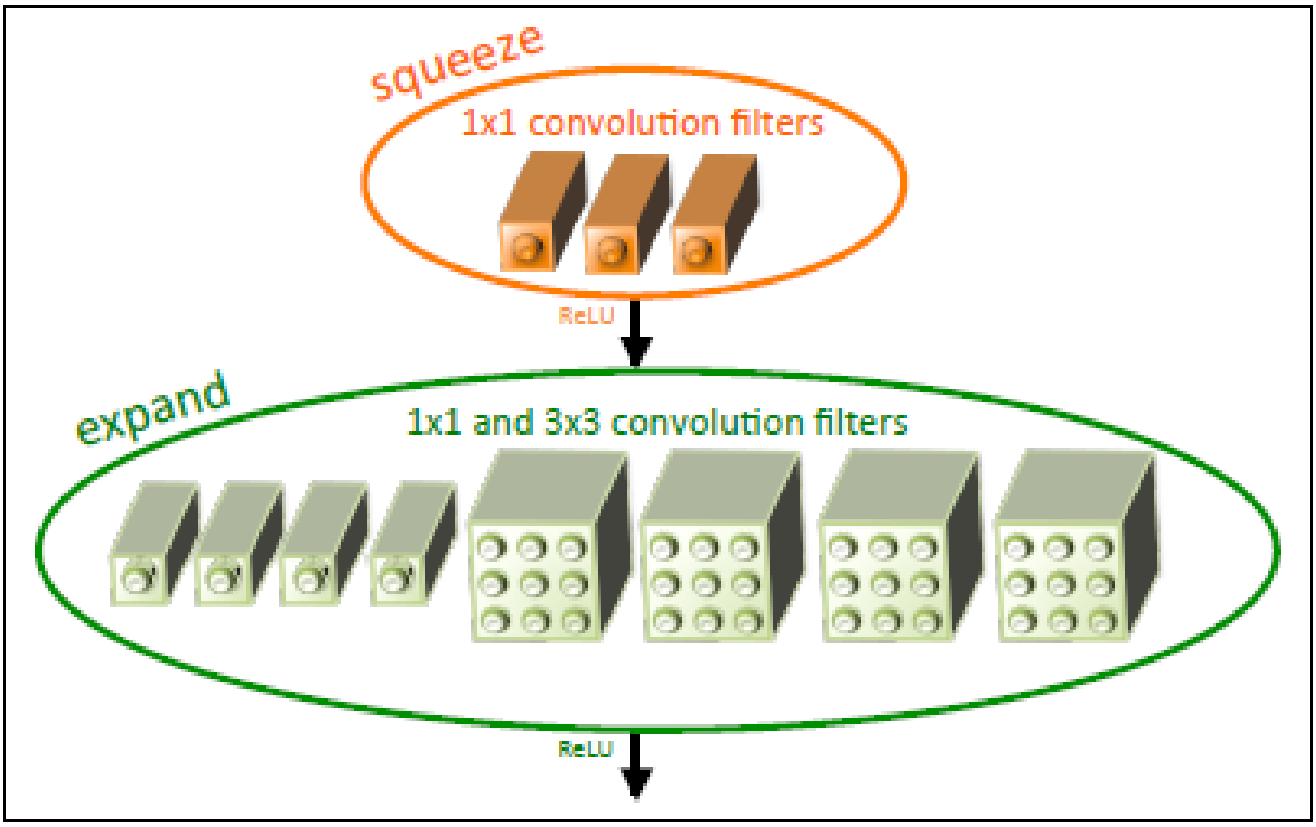
- **SqueezeNet**

SqueezeNet是公认的轻量级模型设计最早期的工作之一，作者提出了三种策略来实现在保持精度的情况下大大减少当时主流模型(以AlexNet为例)的计算量和参数量：

1.将模型中一部分的3×3卷积用1×1来代替，1×1卷积是3×3参数量和计算量的1/9，所以可以大大减少参数量和计算量；

2.减少3×3卷积的输入通道数，这个可以通过在进入3×3卷积之前加一个1×1卷积来实现通道数量的减少；

3.将下采样层的位置往后推，使得模型可以在更大的feature map上进行更多的学习，这一步虽然会在增加计算量，但是和上面两个策略结合可以在维持模型精度的情况下仍大大减少参数量和计算量；



fire module

根据上面的策略，作者提出了fire module的子结构，如下图所示，然后整个模型由这样的子结构堆叠而成。这个fire module由squeeze部分和expand部分构成，squeeze部分是1×1的卷积层，而expand部分是1×1的卷积和3×3的卷积拼接起来的，每次feature map输入这个fire module会在squeeze层降低通道数，然后在expand通道增加通道数，从而在参数量更少的情况下仍然可以得到充分的学习。最后结合一些模型压缩的方法可以使得SqueezeNet在达到AlexNet同等精度的情况下，参数量减少到后者的1/50，计算量减少到后者的1/510。

这篇论文使用大量1×1的卷积核代替3×3卷积，并且利用1×1卷积改变大尺度卷积层输入feature map的通道数从而减少计算量的思想是非常有意义的，后续的很多轻量级网路的论文都沿用了这种套路。

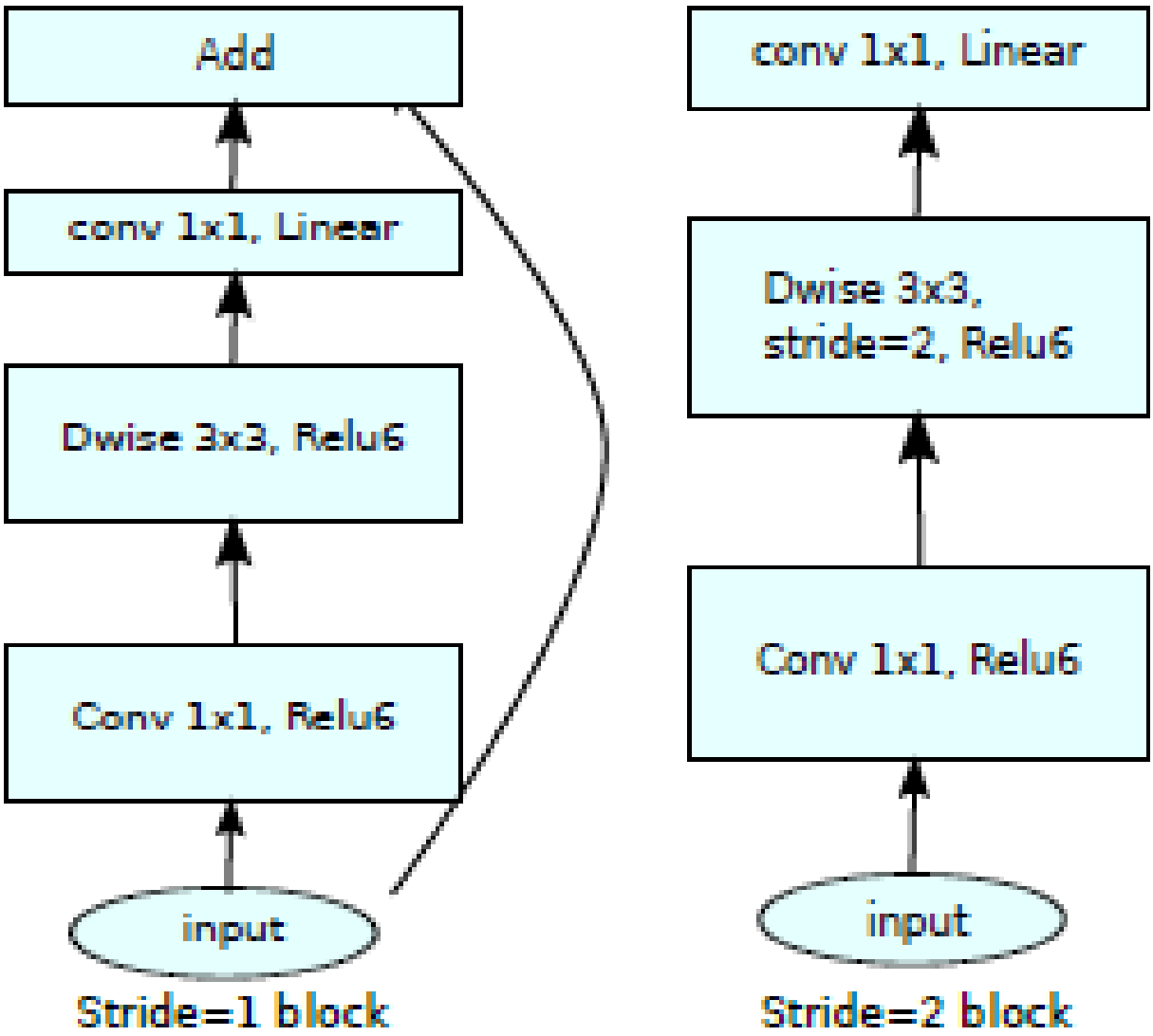
- **MobileNet系列**

MobileNet系列一共有V1,V2和V3三篇论文，简要的讲：

1.MobileNet V1主要思想是提出了一种新的结构—深度可分离卷积(Depthwise Separable Convolution)来代替标准3×3卷积，从而大大减少模型的参数量和计算量；

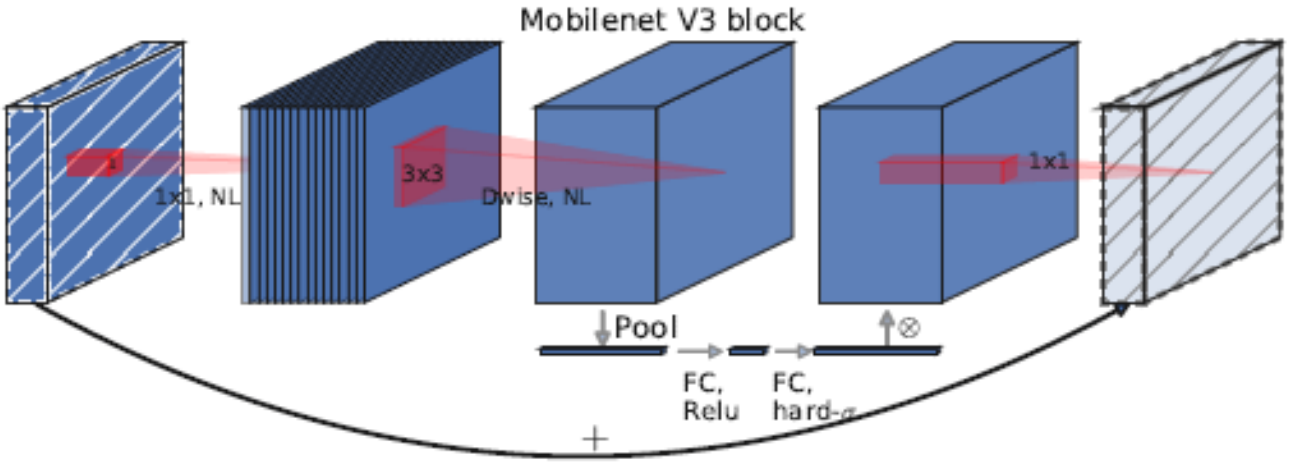
2.MobileNet V2在V1的基础上提出了一种倒置残差的模块，这个模块有三个卷积，第一个部分是一个1×1标准卷积，用来升维，第二个部分是由3×3深度卷积+1×1标准卷积构成的深度分离卷积，用来学习特征和降维，模块的输出和输入再进行一个Addition的操作，由于和ResNet中维度升降方式相反，所以称为倒置残差。中间升维的作用是让深度可分离卷积得到更充分的学习，计算量相对于标准卷积来说也不大，而且这种升降维的方式非常灵活，可以大大减少计算

量。本文还从流形学的角度探究了输入深度可分离卷积上一层的ReLU6对信息传递的影响，理论证明去掉上一个1×1标准卷积的ReLU激活函数能更有利于后面的深度可分离卷积对特征的学习。



MobileNet V2

3. MobileNet V3感觉相对于前两篇没有那么大的结构创新了，主要思想是神经架构搜索(NAS)和硬件友好结构，总的来看V3的结构是在V2的基础上进行了一些修改，如增加了SE block这种已被提出的注意力机制，激活函数换成了H-swish，last stage减少了几层计算，针对语义分割提出了Lite R-ASPP的head(不在讨论之列)，整个论文看着像是堆tricks，重点不是很突出，有点年底冲业绩的嫌疑。



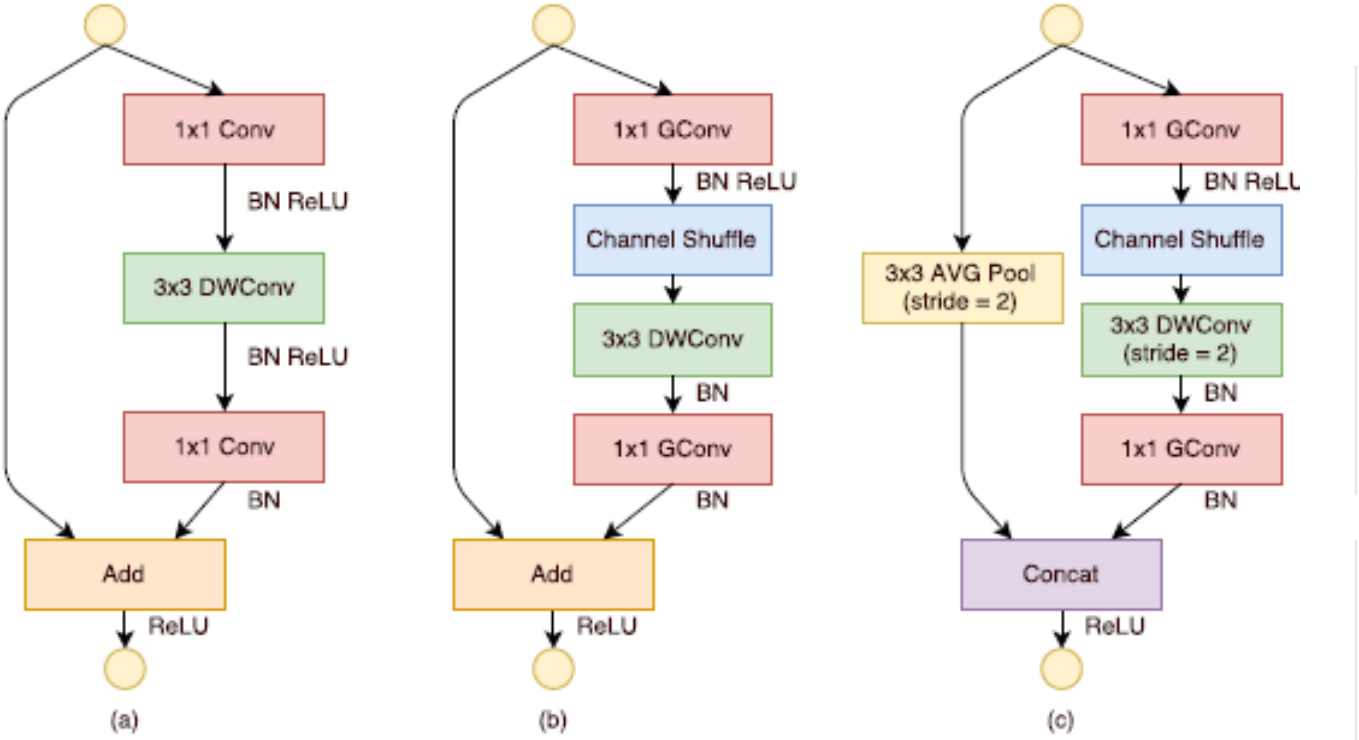
MobileNet V3

根据我自己的比赛和项目经验来看，还是MobileNet V1和V2的结构比较实用，参数量和计算量小，可拓展性好，SE block这种模块对延时影响还是不小，而且我们发现其他各种花里胡哨的激活函数跟ReLU/ReLU6相比都差不多，对精度没有很大的影响，还不如直接部署ReLU/ReLU6来的方便。

- **ShuffleNet系列**

旷视出品的ShuffleNet系列有两篇论文，且后一篇在打前一篇的脸，很有意思。

1.ShuffleNet V1是在MobileNet V1后MobileNet V2前提出的，说实话结构上和MobileNet V2还挺像，大家可以上下两张图片对比一下。两者都想到了学习ResNet的残差结构，区别在于ShuffleNet V1觉得block当中的1×1标准卷积也非常耗时，于是用1×1的分组卷积外加channel shuffle的操作给替换了，然后MobileNet V2会先升维让深度可分离卷积得到充分的学习再降维回来，ShuffleNet V1中stride为2的模块也有自己的特色，虽然看着MobileNet V2的结构更简洁一些，但ShuffleNet V1创新也是不少，尤其那个用channel shuffle增强不同组之间信息交互的操作。

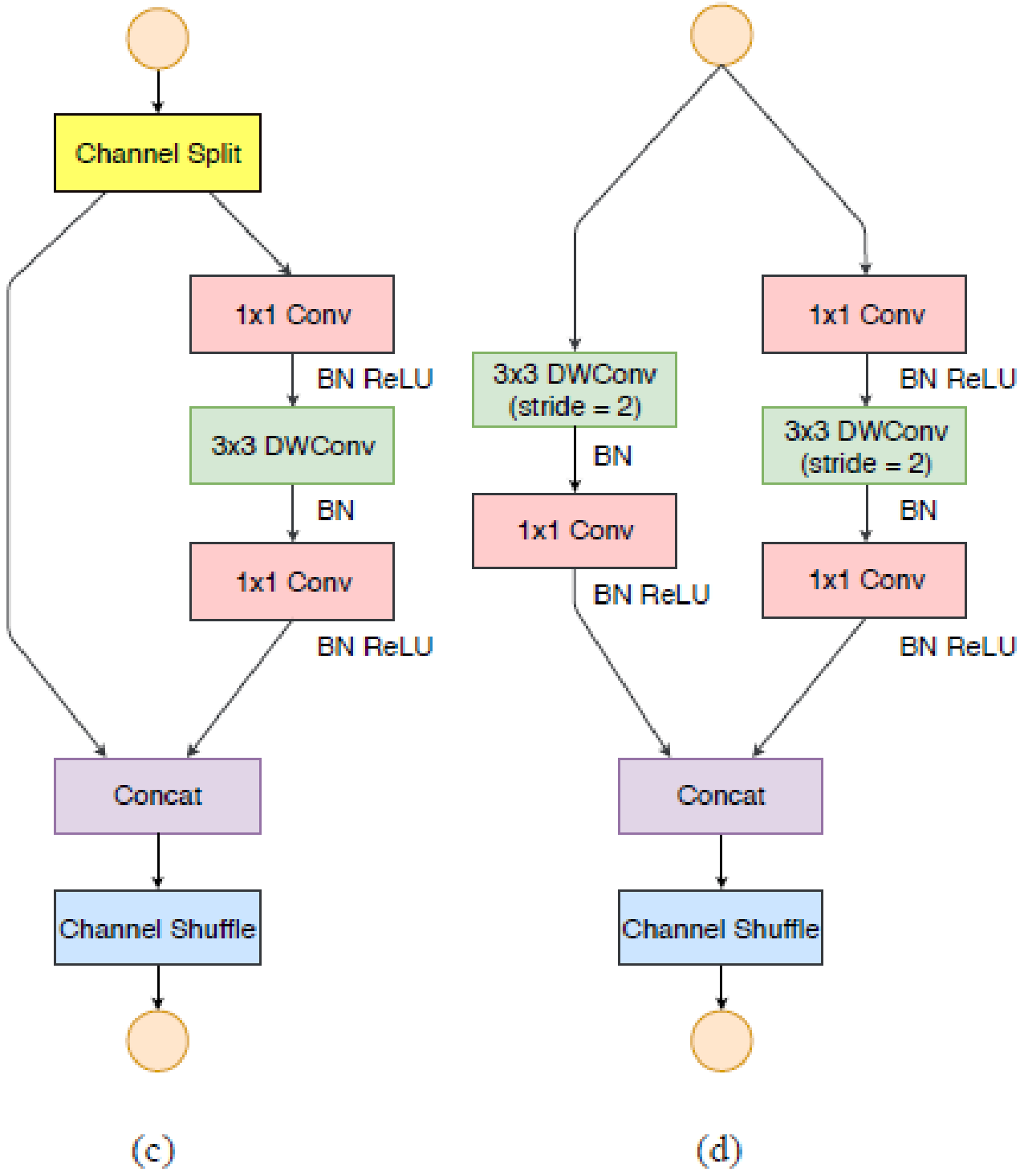


ShuffleNet V1

2. ShuffleNet V2论文是一篇诚意满满之作，作者通过分析ShuffleNet v1与MobileNet v2这两个移动端网络在GPU/ARM两种平台下的时间消耗分布，看出Conv等计算密集型操作占了绝大多数时间，但其它像Elemwise和IO等内存读写密集型操作也占了相当比例的时间，因此像以往那样仅以FLOPs来作为指导准则来设计CNN网络是不完备的，虽然它可以反映出占大比例时间的Conv操作，但不够准确。于是作者提出了高效网络设计的四个指导原则：

> 1. 当输入和输出的通道数相同时，conv计算所需的MAC(memory access cost)最小；
> 2. 大量的分组卷积会增加MAC开销；
> 3. 网络结构的碎片化会减少其可并行优化的程度，GoogleNet系列和NASNet中很多分支进行不同的卷积/pool计算非常碎片，对硬件运行很不友好；
> 4. Element-wise操作不可忽视，对延时影响很大，包括ReLU，Addition，AddBias等，主要是因为这些操作计算与内存访问的占比太小；

基于此，作者提出了ShuffleNet V2的blocks，如下所示，与V1相比，去掉了分组卷积的操作，去掉了Add操作，换成了Concat，stride为2的block的旁路把平均池化换成了深度可分离卷积，为了继续延续channel shuffle的操作，作者在block进去的地方做了个split的操作，最后再concat+channel shuffle，这里是为了替换掉之前的Add，同时也可以减少计算量。

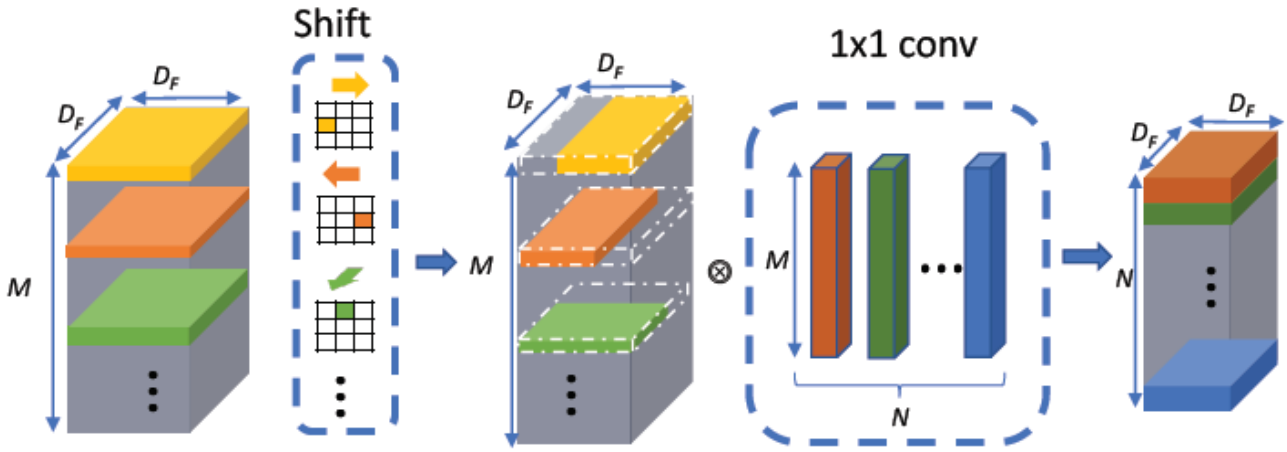


ShuffleNet V2

- **Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions**

这是一篇很有意思的论文，主要是提出了一种无参数，无计算的移位算子来代替ResNet中计算量占比很高的3×3卷积，这种算子称为shift kernel，如下图所示，只需要根据kernel上的shift信息对feature map进行位置上的上下左右偏移即可得到新的feature map，没有计算，仅仅是访存操作。详细一点说就是如下面Shift框里面的第一个卷积核，它只在黄色的区域为1，其他白色的区域为0，在做卷积计算的时候其实就相当于把输入feature map中间偏左边的那个点的值平移到输出feature map中间的地方，正如作者标注的向右的箭头所示。而且这个操作都是per-channel的，所以每个卷积核只有k×k种可能性，当通道数大于k×k时，就需要将所有通道分成i C/(k×k)组，剩下的channel设置为center，即中间为1，其余为0，然后怎么去选取每个卷积核

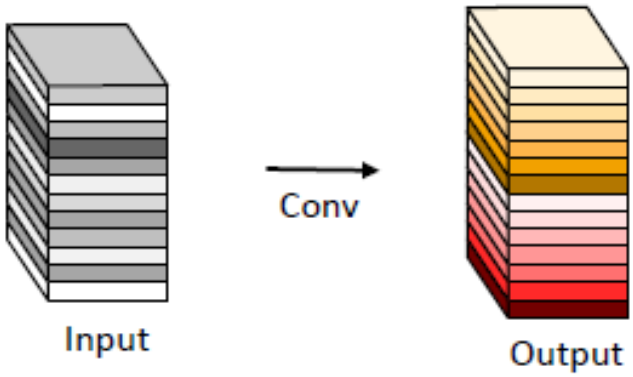
的类型呢，论文在两个shift kernel中间夹了一个1×1的标准卷积，要保证第二次shift操作之后数据与第一次输入shift kernel之前顺序一致，并且这个shift操作可以通过SGD的方式进行端到端训练，再具体的细节论文其实也没有阐述的很清楚，而且我目前也没有看到作者公布源代码，不过这篇论文看起来还是很有意思的，论文中还分析了depthwise计算不高效的原因在于计算/IO时间导致运行更慢的问题，其实这个shift kernel的作用并没有产生新的信息或者去进行特征的学习(毕竟连参数都没有)，而是对feature map空域的信息做了个混洗，使得与他相接的1×1卷积可以学到更丰富的特征，在我看来有点类似于在网络内部做数据增强的感觉。



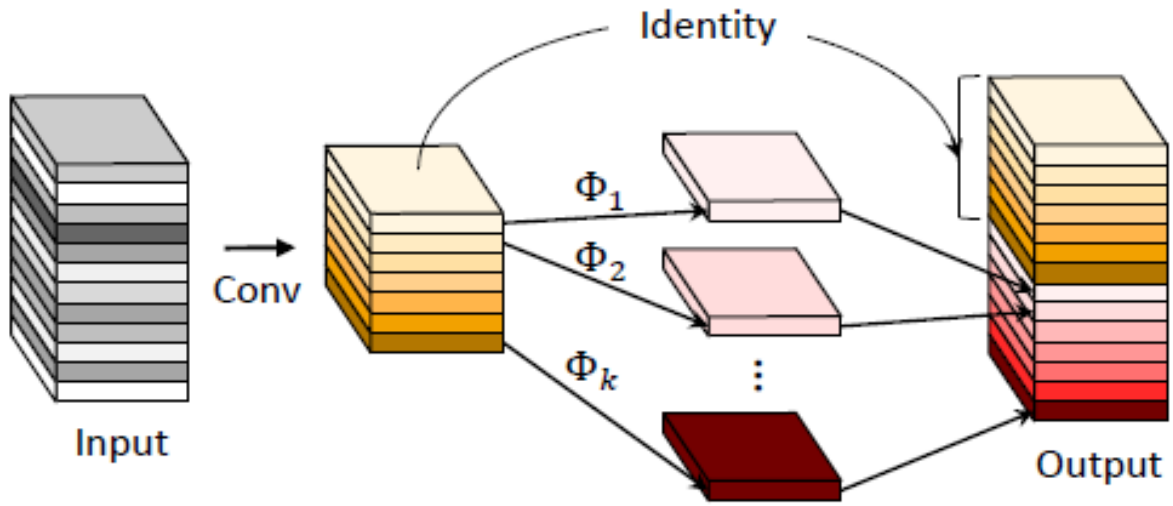
Shift

- **GhostNet**

GhostNet也是一篇很有意思且简洁的架构设计的论文，作者在可视化一些训练好的神经网络中间feature map时发现它们通常会包含一些相似且冗余的特征图，使得神经网络能得到更充分的学习。基于这个想法，作者通过设定一系列廉价的线性运算操作来代替部分卷积计算，以此来产生更多的特征图，仅仅这么一个简单的操作就可以减少模型的参数量和计算量，而且在几个视觉公开数据集上取得了很不错的效果，甚至超越了MobileNet V3，感觉非常的大道至简，这也是我比较喜欢的原因。



(a) The convolutional layer.



(b) The Ghost module.

GhostNet

基于特定硬件的神经架构搜索

基于特定硬件的神经架构搜索(MIT HAN Lab论文总结 https://zhuanlan.zhihu.com/p/320290820)

设计方法总结

接下来我将结合自己看过的论文，还有这一年多的项目比赛经历谈一谈我所理解的**图像分类和目标检测**相关轻量级模型设计，本文思想还比较浅薄，主要是给自己的工作做个总结，有不正确的地方希望大家能共同讨论。

设计之前

通常我们都是基于已有的硬件架构去进行模型的部署，这个时候就需要确定这个架构下能部署什么算子，可以通过已有的接口自己拓展哪些算子，并且哪些算子不是很高效。就拿我去年参加的某视觉加速比赛来说，当时初出茅庐，不太懂硬件底层，头铁要在Xilinx ultra96 V1板子 +

我们组自研的硬件架构上部署当时最新，准确率最高的EfficientNet，因为准确率确实高，老板就钦定必须使用这个模型，并且选择了比较合适的EfficientNet-B4，输入分辨率由384改成256。后来一顿开搞发现有很多swish操作，这个虽然之前还没有经验，但是还好很快想到用查找表去实现了，并且还发现我们的架构尚且不能实现全连接层(勿喷，之前都是在搞全卷积网络)，所以里面的SE block还有最后一层都无法用我们的架构部署，然后博士师兄就想到了利用ultra96板子上的ARM+NEON加速技术去实现这一部分，每次PS和PL交互数据，当时只有一个月的开发时间，因为这个模型比较大且当时我们开发模式的问题，连续熬夜差点没把整个组人的命搭进去，最后上板跑帧率也只有6帧(师兄最开始预估能达到80帧，所以大家一直不停的往下做hhhhh)，在ImageNet验证集上纯浮点准确率是0.805，INT8准确率是0.793，帧率低的原因有两点，一个是模型确实很大，另一个是因为SE block在每一个stride为1的module中都出现了，整个模型PS和PL交互十分频繁，而我们当时间很紧刚刚完成一版就必须提交了，所以这块也根本没有优化，导致了延时很高，并且当时我们的架构只跑了100M的频率，更高频率会有一些bug(貌似是板子的问题)，所以帧率非常低，这个真的是血的教训：一定要量力而行，仔细研究评价指标，在硬件友好程度和精度上做一个trade-off，一味片面地追求精度真的要命呀。

轻量级CNN架构设计

总的思路： 选定合适结构 + 通道剪枝 + 量化

训练 ： ImageNet pretrain model + Data Normalization(统计自己数据集的均值和方差) + Batch Normlization + 大batch size + 一堆数据增强tricks + 尝试各种花里胡哨的loss function和optimizer

(再次说明这部分只讨论图像分类和目标检测两种任务，目前的视觉加速比赛基本都是基于这两个任务做的，按照计算资源和内存从小到大排列，不用问，没有划分原则)

1.绝对贫穷人口

- 输入分辨率要小，如128,160,192或256；
- 下采样使用MaxPooling；
- 特征学习层使用Depthwise Separable Convolution，即一层3×3的Depthwise + 一层pointwise(1×1标准卷积)堆叠；
- 激活函数用ReLU；

另外这里推荐看看MCUNet，感觉非常呦西，MIT韩松团队yyds！

2. 相对贫穷人口

- 输入分辨率依旧要小，记住分辨率对计算量的影响都是翻倍的；
- 下采样可以使用MaxPooling或者stride为2的Depthwise Separable Convolution；
- 特征学习层使用Depthwise Separable Convolution，或者MobileNet V2的倒置残差结构，又或是ShuffleNet V2的unit，1×1的Group convolution能处理的比较好的话其实也推

荐使用(主要是计算/访存)；

- 激活函数用ReLU或者ReLU6；

3. 低收入人口

- 输入数据选用小分辨率，如果对精度要求高一些可以适当增大；
- 下采样可以使用MaxPooling或者stride为2的Depthwise Separable Convolution；
- 特征学习层可以使用MobileNet V2的倒置残差结构，又或是ShuffleNet V2的unit，也可以使用通道数小一点的3×3标准卷积；
- 激活函数用ReLU，ReLU6或者leaky ReLU，看效果了；

4.一般收入人口

- 输入数据可以稍微大一些，288,320,384啥的可以考虑上了；
- 下采样使用stride为2的Depthwise Separable Convolution或者stride为2的3×3卷积；
- 特征学习层既可以使用上述的，也可以使用3×3标准卷积 + 1×1标准卷积的堆叠形式，SE block这种硬件支持的还不错的可以尝试加上；
- 激活函数除了上述的可以试试H-sigmoid和H-swish，不过据我经验效果基本和ReLU差不多；

5.高收入人口

- 输入数据的分辨率可以往500-600靠拢了；
- 下采样和上述一样；
- 特征学习层可以上ResNet + SE block的配置，ResNet是真的牛逼，5×5的卷积啥的也可以整上，第一层直接上7×7的标准卷积也不是不可以，资源再多了可以增加通道数和深度 或者上多核并行计算；
- 激活函数可以用H-swish；

番外

目标检测任务中感觉Tiny YOLO V3非常受欢迎，建议尝试！计算量太大可以换更轻量的backbone或者改输入分辨率，轻量级的backbone+FPN的结构也很棒，且推荐使用商汤开源的mmdetection，训练调参当场起飞。

另外之前阅读MIT HAN lab基于特定硬件的神经架构搜索相关文章时发现他们设计模型常用的一个子结构：MobileNetV2的倒置残差模块 + SE block + H-swish，看他们在很多算法加速比赛上拿了冠军，感觉百试不爽呀，且根据硬件资源进行拓展的灵活度也很高，具体可以参见他们今年发表的OnceForAll论文中的模型，源代码都在Github上能找到，MIT HAN lab真学术界良心，再喊一句MIT韩松团队yyds！

最后总结

过去的一年被导师安排着参加各种比赛，去年啥也不懂的时候还能拿几个不错的奖，今年感觉学了很多懂了很多，使用的模型也都对硬件比较友好，量化后几乎无损(一个点以内)，反倒连连受挫，心情非常沮丧，而且总被奇奇怪怪的模型打败，总有一种自己学了一身正派功夫，最后反倒被野路子出招一击即溃的感觉，然后论文也被拒了，没心思改投，回想一下碰上疫情的这一年真的好失落，可能还是我太菜了吧。

马上也要开始投入找实习(希望老板能放我)刷题找工作的阶段了，最近也在培养下一届，把工作慢慢移交给他们，一想到找工作，整个人的心态都不一样了，无心科研，这篇文章就算是对我过去一年多的工作做个总结吧，同时也希望我们课题组能够发展的越来越好，多多拿比赛大奖，多多发论文。

阅读原文

极市平台