本篇博文来介绍一个深度学习模型加速的算法：MobileNets，可以在基本不影响准确率的前提下大大减少计算时间和参数数量。

论文：MobileNets Efficient Convolutional Neural Networks for Mobile Vision Applications

论文链接：https://arxiv.org/abs/1704.04861

MXNet框架代码：https://github.com/miraclewkf/mobilenet-MXNet

算法概述：

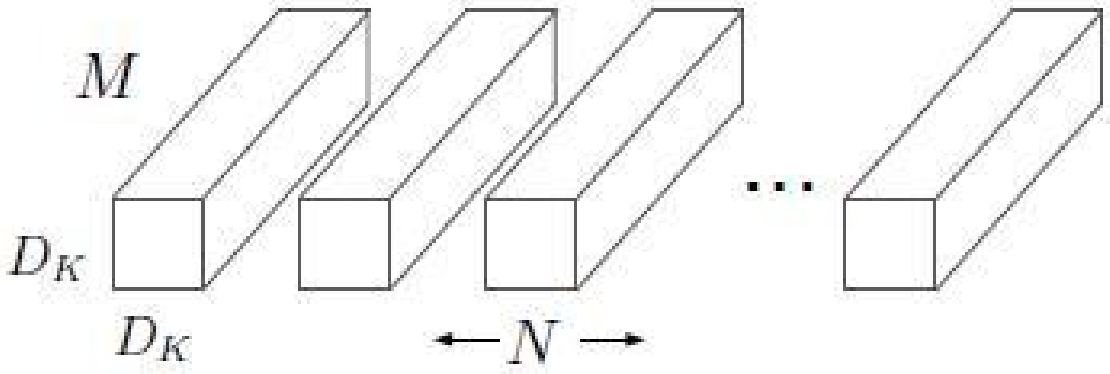
深度学习网络广泛应用在图像分类，检测中，但是网络结构复杂，参数过多，计算时间过长使其不容易在移动端应用。因此像模型压缩、模型加速应该会是未来深度学习比较活跃的一个领域。本文提出一种将传统的卷积结构改造成两层卷积结构的网络：MobileNets，采用类似ResNext里面的group操作来实现。这种新的结构可以在基本不影响准确率的前提下大大减少计算时间（约为原来的1/9）和参数数量（约为原来的1/7）。

这篇博客中不特殊强调的话，卷积核默认都是三维，这三维分别对应长、宽和输入通道数，因为不同人对卷积核维度的理解不同。对于常规卷积而言，假设输入特征通道数是M，卷积核的长宽分别是DK和DK，卷积核的数量是N，那么可以说是有N个M*DK*DK卷积核，也可以说是有N组卷积核，每组有M个DK*DK的卷积核。不管是哪种理解，都不影响卷积层的本质：该层有N*M*DK*DK个参数。

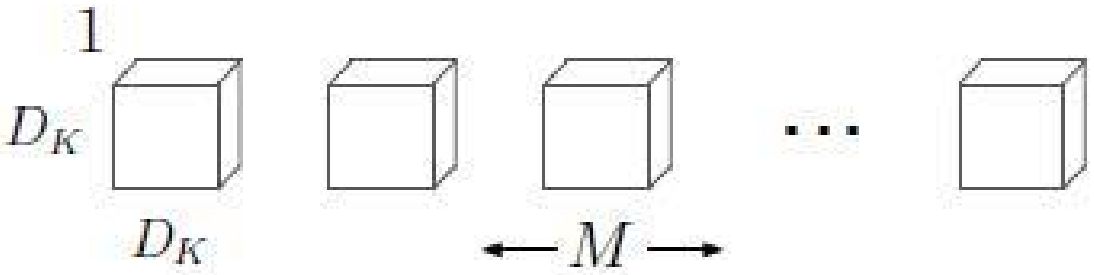
算法详解：

MobileNet模型的核心就是将原本标准的卷积操作因式分解成一个depthwise convolution和一个1*1的卷积（文中叫pointwise convolution）操作。简单讲就是将原来一个卷积层分成两个卷积层，其中前面一个卷积层的每个filter都只跟input的每个channel进行卷积，然后后面一个卷积层则负责combining，即将上一层卷积的结果进行合并。

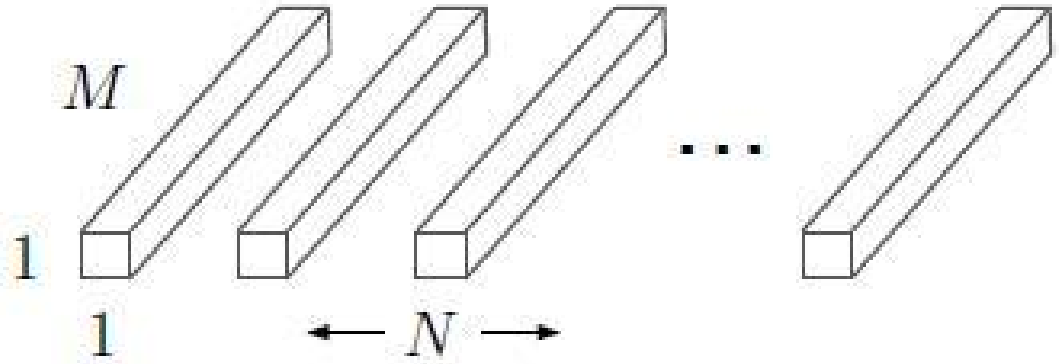
如下图：M表示输入特征的通道数，N表示输出特征的通道数（也是本层的卷积核个数）。因此如果假设卷积核大小是DK*DK*M*N，输出是DF*DF*N，那么标准卷积的计算量是DK*DK*M*N*DF*DF。这个式子可以这么理解，先去掉M*N，那么就变成一个二维卷积核去卷积一个二维输入feature map；那么如果输出feature map的尺寸是DF*DF，由于输出feature map的每个点都是由卷积操作生成的，而每卷积一次就会有DK*DK个计算量，因此一个二维卷积核去卷积一个二维输入feature map就有DF*DF*DK*DK个计算量；如果有M个输入feature map和N个卷积核，那么就会有DF*DF*DK*DK*M*N计算量。（另外博主认为文中DF和DG使用比较混乱，且有些错误，所以希望能认同我这段分析）



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

本文的算法是用上图中的（b）+（c）代替（a），接下来详细讲解下什么怎么卷积的。依然假设有N个卷积核，每个卷积核维度是DK*DK*M，输入feature map的通道数是M，输出feature map为DF*DF*N。那么（b）表示用M个维度为DK*DK*1的卷积核去卷积对应输入的M个feature map，然后得到M个结果，而且这M个结果相互之间不累加（传统的卷积是用N个卷积核卷积输入的所有（也就

是M个）feature map，然后累加这M个结果，最终得到N个累加后的结果），注意这里是用M个卷积核而不是N个卷积核，所以（b）中没有N，只有M。因此计算量是DF*DF*DK*DK*M。（b）生成的结果应该是DF*DF*M，图中的（b）表示的是卷积核的维度。

（c）表示用N个维度为1*1*M的卷积核卷积（b）的结果，即输入是DF*DF*M，最终得到DF*DF*N的feature map。这个就可以当做是普通的一个卷积过程了，所以计算量是DF*DF*1*1*M*N（联系下前面讲的标准卷积是DF*DF*DK*DK*M*N，就可以看出这个（c）其实就是卷积核为1*1的标准卷积）。

所以最重要的来了，采用这种算法的计算量变成了DF*DF*DK*DK*M+DF*DF*M*N。具体和原来相比减少了多少计算量？可以看下面这个式子：

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

也就是说如果卷积核大小为3*3，那么差不多卷积操作的时间能降到原来的1/9左右！

所以看看Fig3表达的标准卷积（左边）和因式分解后的卷积（右边）的差别。注意到卷积操作后都会跟一个Batchnorm和ReLU操作。

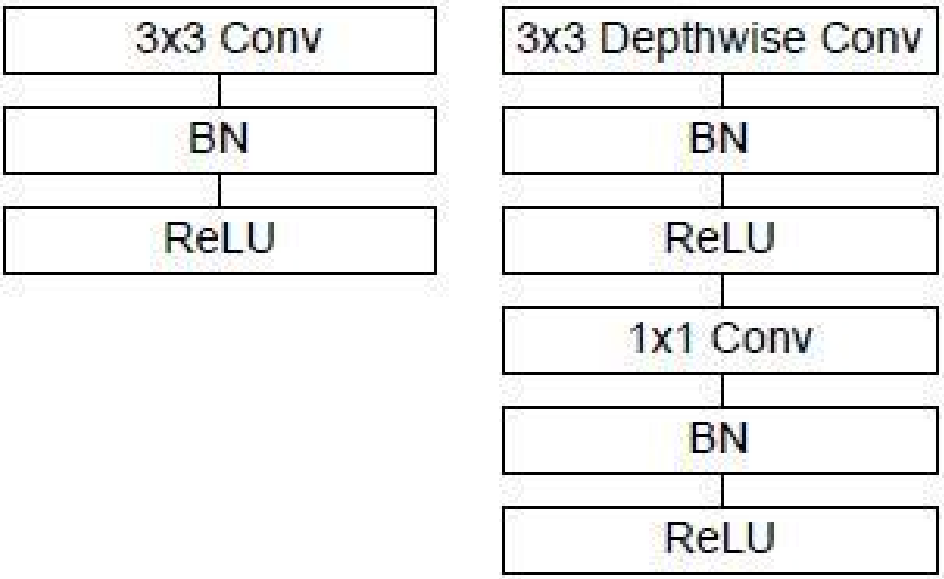


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

然后看看这个网络结构：如果把depthwise和pointwise看做不同层的话，MobileNet一共包含28层。第一个卷积层不做分解，另外最后有个均值pooling层，全连接层和softmax层。这里dw就表示depthwise。

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

这个Table2介绍了网络中不同类型层的计算时间在总时间中的比例以及参数的数量在总的参数数量中的比例。

Table 2. Resource Per Layer Type

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

前面讲的都是在计算时间和参数的减少，现在看看计算准确率的对比：Table4，表示本文卷积和标准卷积的对比。可以看出Accuracy减少得不是很明显，但是却大大减少了计算时间和参数数量。

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

为了得到更小和更快的模型，作者介绍了两个概念：Width Multiplier 和 Resolution Multiplier。前者表示输入channel变成baseline的多少倍，如Table6；后者表示对输入图像做缩放，如Table7。

Table 6. MobileNet Width Multiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Table 7. MobileNet Resolution

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

其他更多实验对比可以参看论文。

一个细节：实际中L2正则项的系数要比较小，因为本身参数已经减少许多了。

总结

作者主要提出一种将标准卷积层拆分成两个卷积层的MolileNet网络，可以在基本保证准确率的前提下大大减少计算时间和参数数量。个人认为应该对于全卷积网络（ResNet等）的提升效果比较明显，可以在最近的object detection算法中借鉴。另外模型加速和压缩应该会是最近几年比较活跃和值得关注的领域。