7.MobileNet V1 V2

2017年Google团队提出MobileNetV1，2018年Google团队提出MobileNetV2，论文地址MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, MobileNetV2: Inverted Residuals and Linear Bottlenecks

专注于移动端或嵌入式设备中的轻量级CNN网络，相比于传统CNN，在准确率小幅度降低的前提下大大减小模型参数与运算量。（相比VGG16准确率减少了0.9%，模型参数只有1/32）。

MobileNetV1:

网络的亮点：深度可分离卷积（Depthwise Separable Convolution）

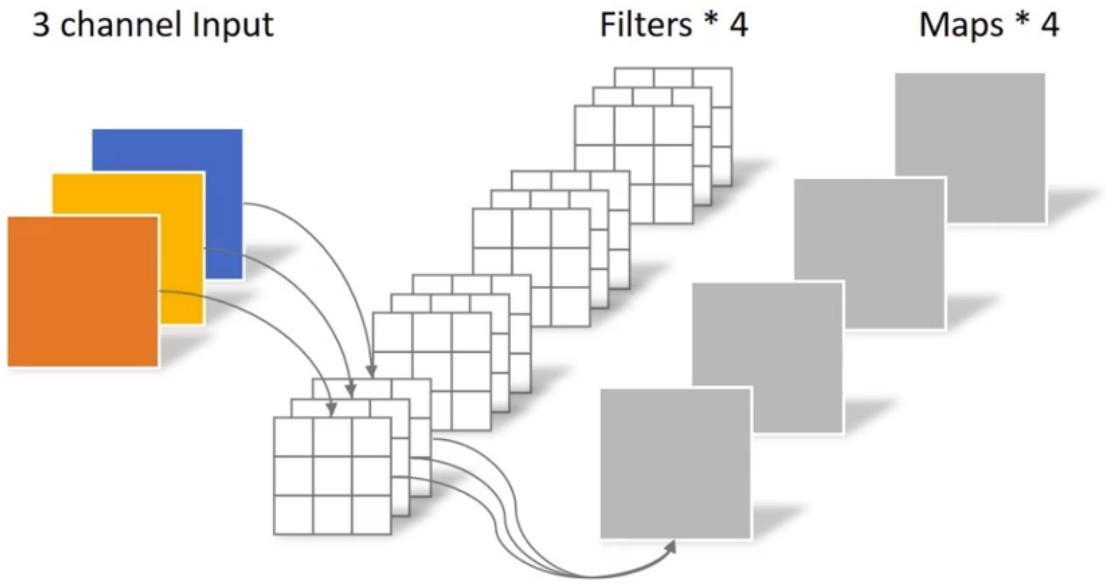


图24 传统卷积

卷积核channel=输入特征矩阵channel
输出特征矩阵channel=卷积核个数

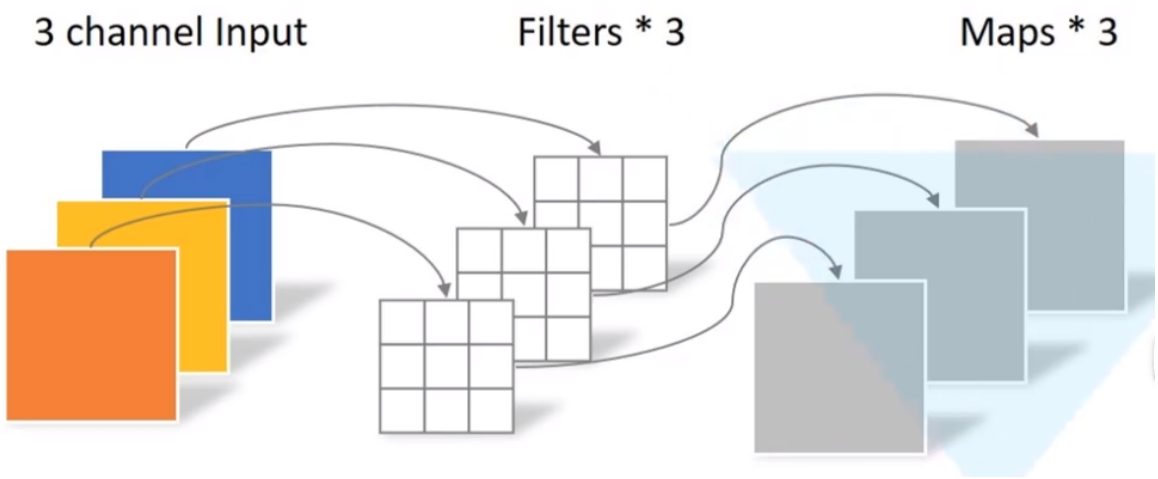


图25 DW卷积

卷积核channel=1
输入特征矩阵channel=卷积核个数=输出特征矩阵channel

每个卷积核只负责与输入中的一个channel进行运算

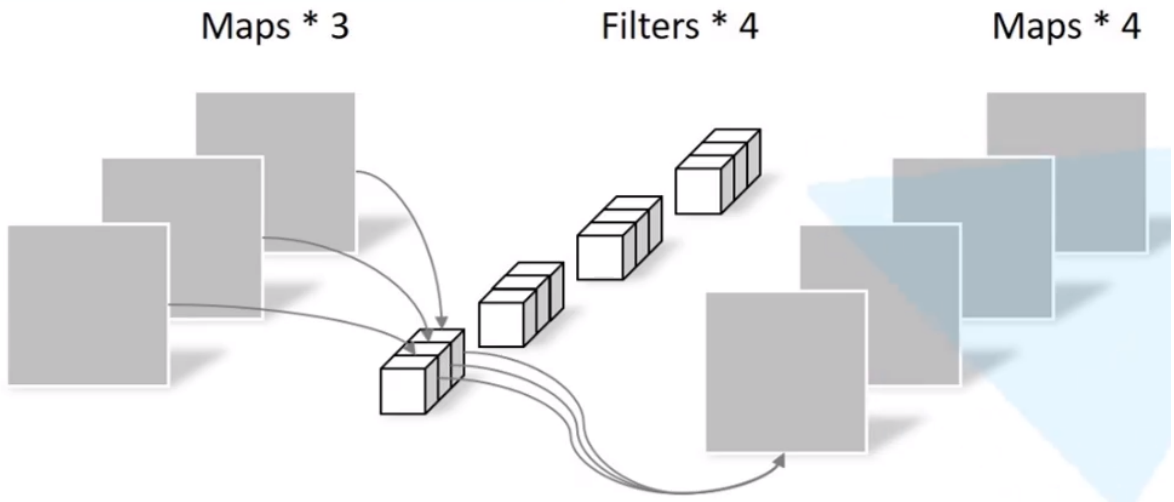
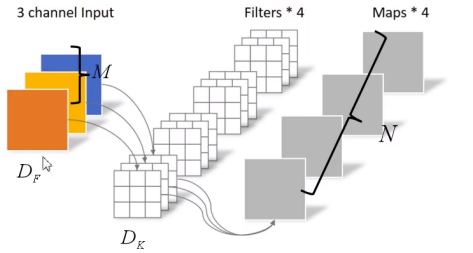


图26 PW卷积

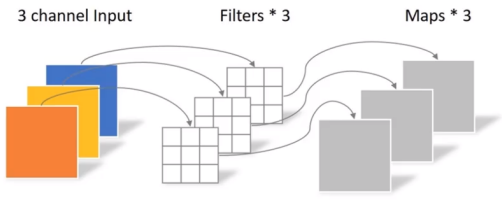


$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}$$ DW + PW

普通卷积

$$= \frac{1}{N} + \frac{1}{D_K^2} = \frac{1}{N} + \frac{1}{9}$$

理论上普通卷积计算量是DW+PW的8到9倍



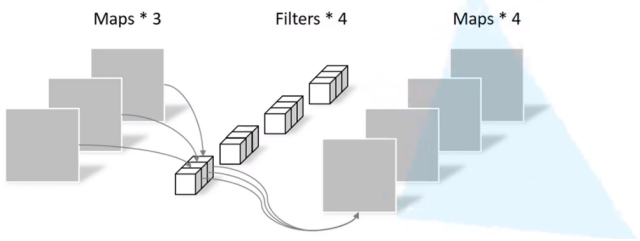


图27 卷积参数对比

在训练过程中，Depthwise部分的卷积核容易废掉，即卷积核参数大部分为0，在MobileNetV2中会进行改善。

MobileNetV2:

网络的亮点:

Inverted Residuals倒残差结构
Linear Bottleneck

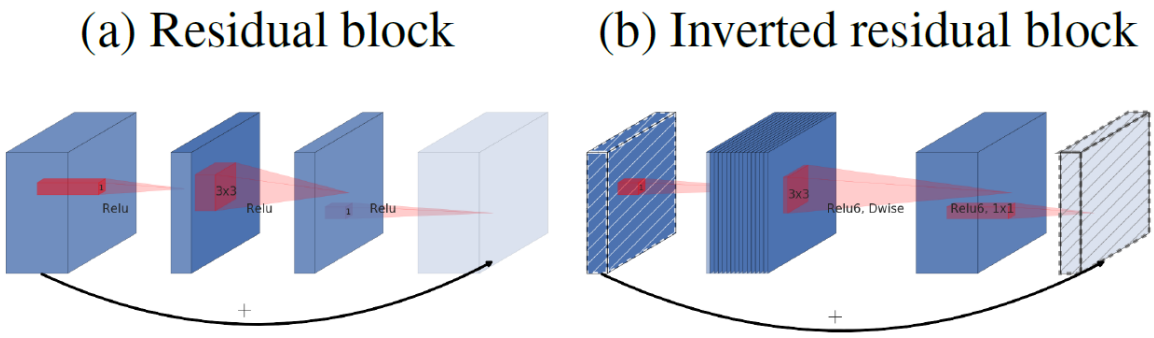


图28 倒残差结构

普通残差结构：1×1conv降维，3×3conv，1×1conv升维
倒残差结构：1×1conv升维，3×3DW，1×1PW降维

针对倒残差结构的最后一个1×1卷积层，使用线性激活函数而不是ReLU，因为ReLU激活函数对低维特征信息造成大量损失，而倒残差结构中间大两边小，最后一个1×1卷积后是低维的，则使用线性激活函数来替代ReLU，来避免信息损失。

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

图29 Bottleneck residual block

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

图30 MobileNetV2网络结构

其中，t是图29中的扩展因子，c是输出特征矩阵深度，n为bottleneck重复次数，s为步距（针对第一层，其他为1）

pytorch实现

在pytorch实现时，在bottleneck的conv运算中，加入了groups，分解卷积。