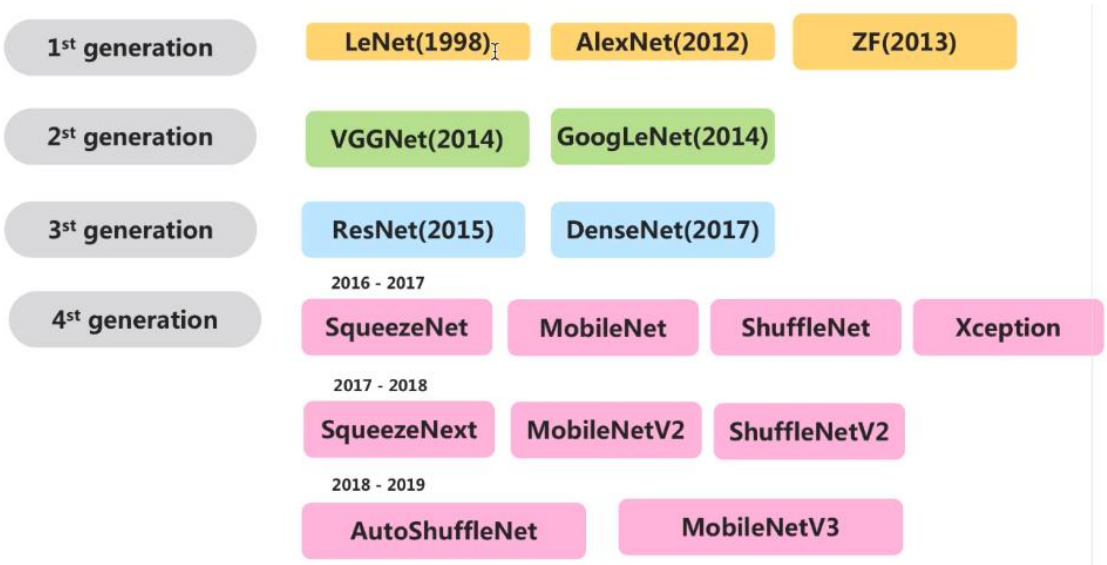
分类模型

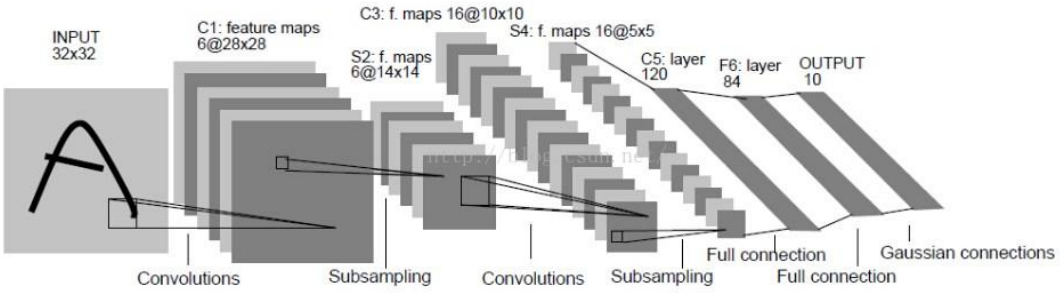


加深 VGGNet

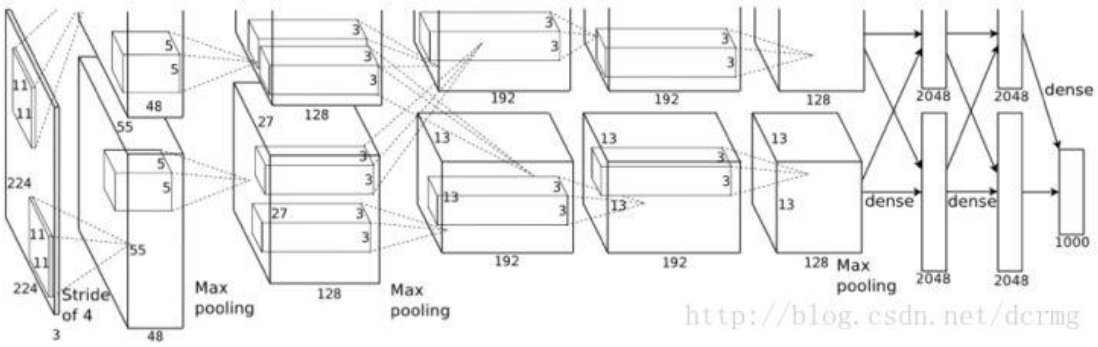
加宽，模块化 GoogLeNet，ResNet，DenseNet

加快 MobileNet

LeNet 引入局部感知，权值共享，多卷积核



AlexNet 更深的网络，数据增广，ReLU，Dropout



输入数据：227×227×3；卷积核：11×11×3×96；步长：4；卷积后数据：55×55×96；

Max pool1 的核：3×3，步长：2；Max pool1 后的数据：27×27×96

输入数据：27×27×96；卷积核：5×5×96×256；步长：1；卷积后数据：27×27×256（Same padding）；Max pool2 的核：3×3，步长：2；Max pool2 后的数据：13×13×256

输入数据:13×13×256;卷积核:3×3×256×384;步长:1;卷积后数据:13×13×384 （Same padding）

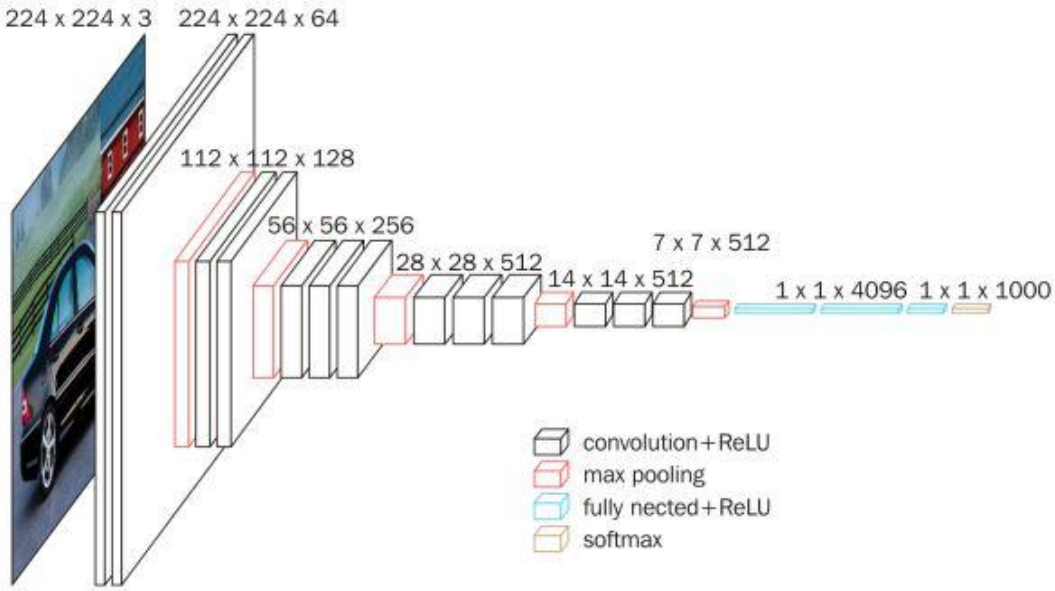
输入数据：13×13×384；卷积核：3×3×384×256；步长：1；卷积后数据：13×13×256（Same padding）；Max pool5 的核：3×3，步长：2；Max pool2 后的数据：6×6×256

输入数据：6×6×256；全连接输出：4096×1

输入数据：4096×1；全连接输出：4096×1

输入数据：4096×1；全连接输出：1000

VGGNet 11,13,16,19 层，更深的网络有助于性能的提升，但是不好训练，更容易过拟合



共 16 层（不包括 Max pooling 层和 softmax 层）；所有的卷积核都使用 3*3 的大小，池化核都使用大小为 2*2；采用步长 stride=1，padding=0 的 Max pooling；卷积层深度依次为 64 -> 128 -> 256 -> 512 ->512。

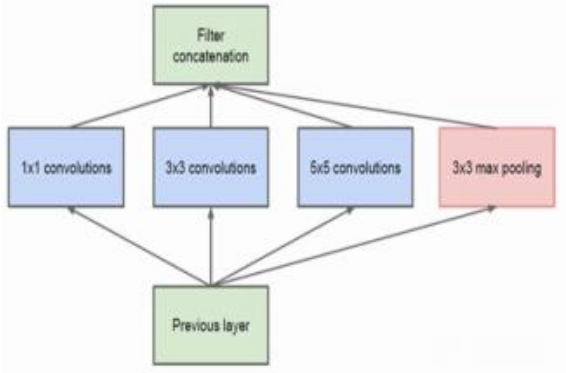
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

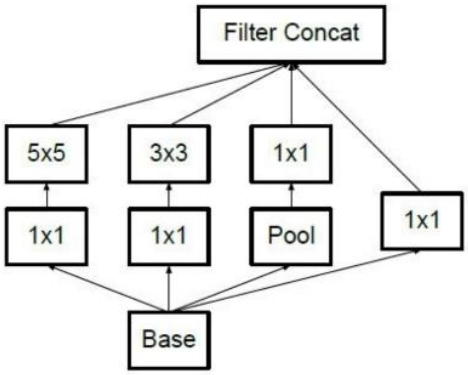
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

GoogLeNet 引入 Inception 结构，如下，

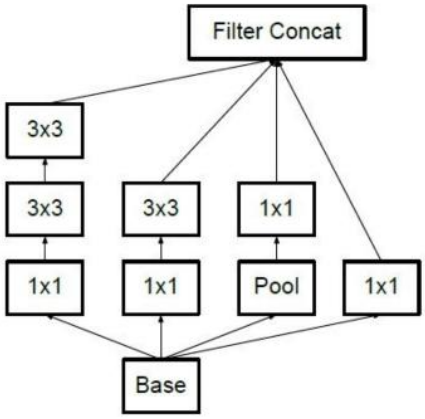
V0：1*1,3*3,5*5,3*3max pooling 的核分别卷积，然后 concat 到一起。



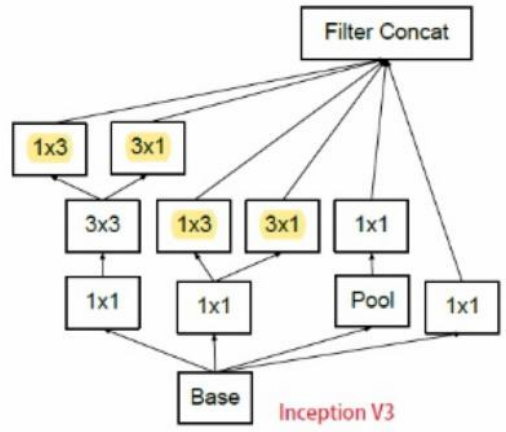
V1：在 3*3，5*5 卷积前加上一层 1*1 卷积，用于降维，进一步减少参数量。



V2：5*5 改成两个 3*3，进一步减少参数量。



V3：n*n 改成两层，1*n 和 n*1。

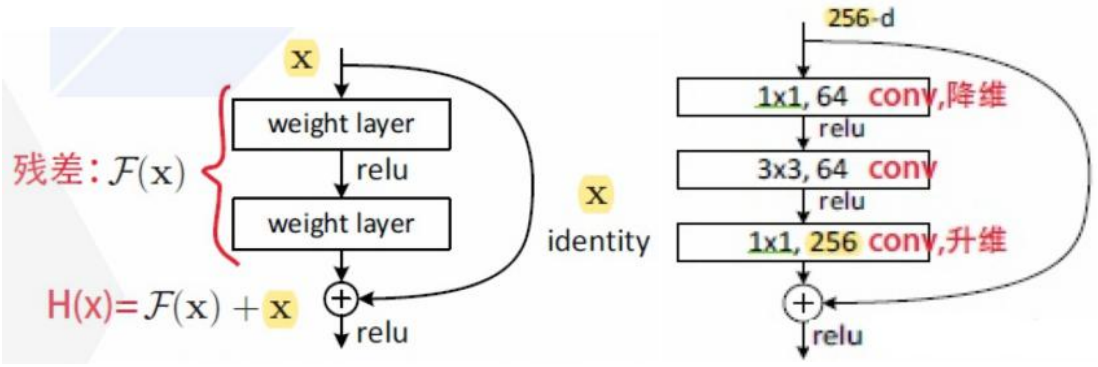


引入中间层辅助 loss 单元

最后的全连接替换为 average pooling

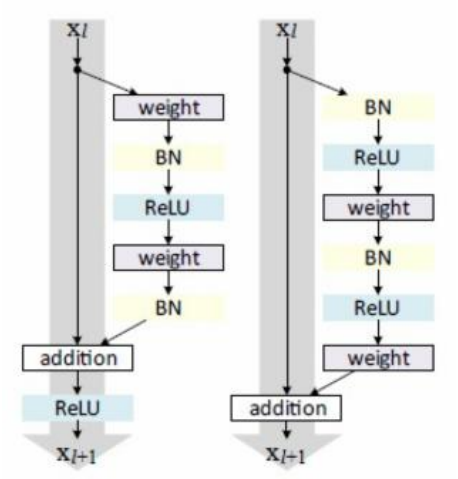
引入 BN 层

ResNet 残差设计

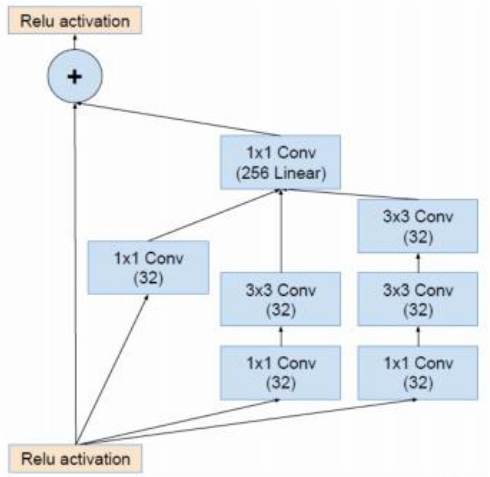


降维是为了减少计算过程中的参数量，升维是为了能和 x 维度一致

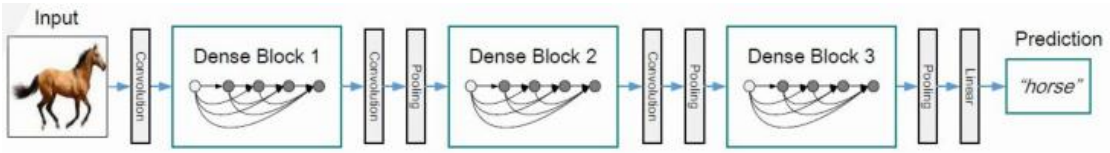
ResNet V2：



Inception-Resnet:

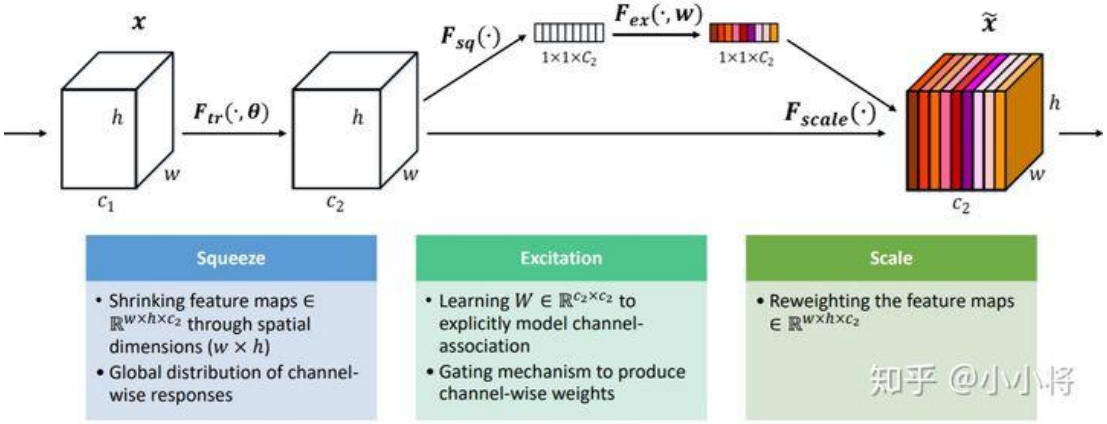


DenseNet



SENet

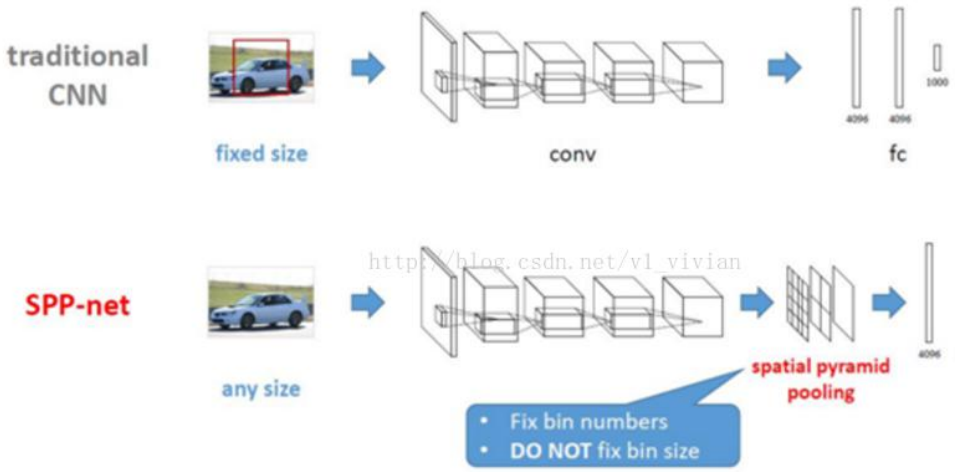
对于卷积操作，很大一部分工作是提高感受野，即空间上融合更多特征融合，或者是提取多尺度空间信息，如 Inception 网络的多分支结构。对于 channel 维度的特征融合，卷积操作基本上默认对输入特征图的所有 channel 进行融合。而 MobileNet 网络中的组卷积（Group Convolution）和深度可分离卷积（Depthwise Separable Convolution）对 channel 进行分组也主要是为了使模型更加轻量级，减少计算量。而 SENet 网络的创新点在于关注 channel 之间的关系，希望模型可以自动学习到不同 channel 特征的重要程度。为此，SENet 提出了 Squeeze-and-Excitation (SE)模块，如下图所示：

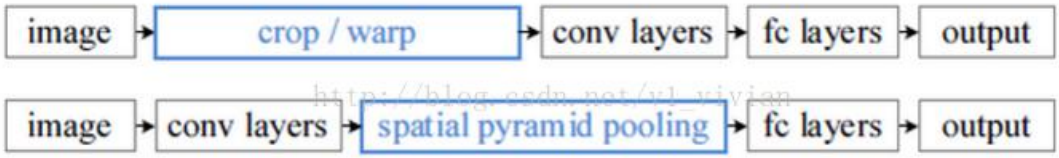


SE 模块首先对卷积得到的特征图进行 Squeeze 操作，得到 channel 级的全局特征，然后对全局特征进行 Excitation 操作，学习各个 channel 间的关系，也得到不同 channel 的权重，最后乘以原来的特征图得到最终特征。本质上，SE 模块是在 channel 维度上做 attention 或者 gating 操作，这种注意力机制让模型可以更加关注信息量最大的 channel 特征，而抑制那些不重要的 channel 特征。另外一点是 SE 模块是通用的，这意味着其可以嵌入到现有的网络架构中。

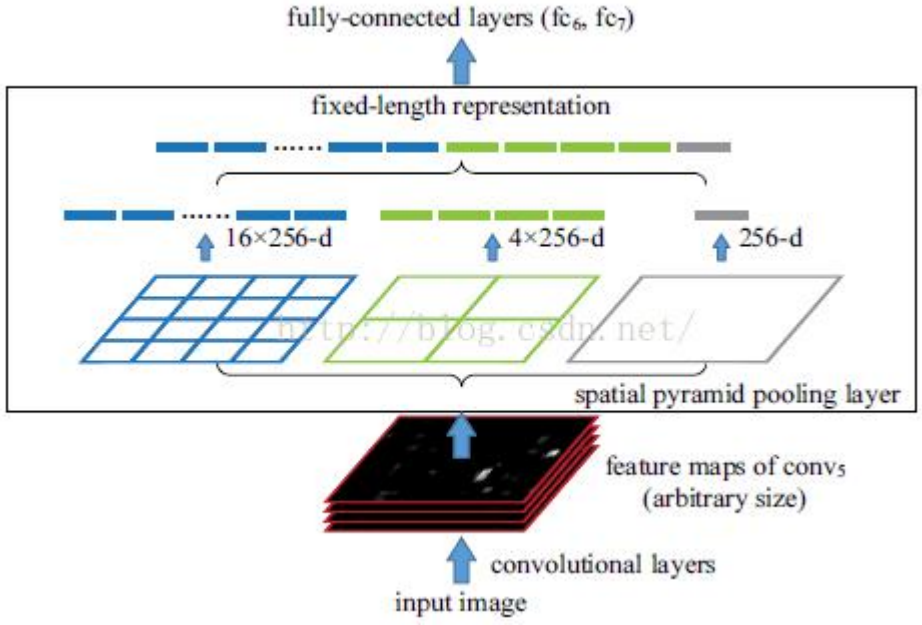
EfficientNet

SPP-Net

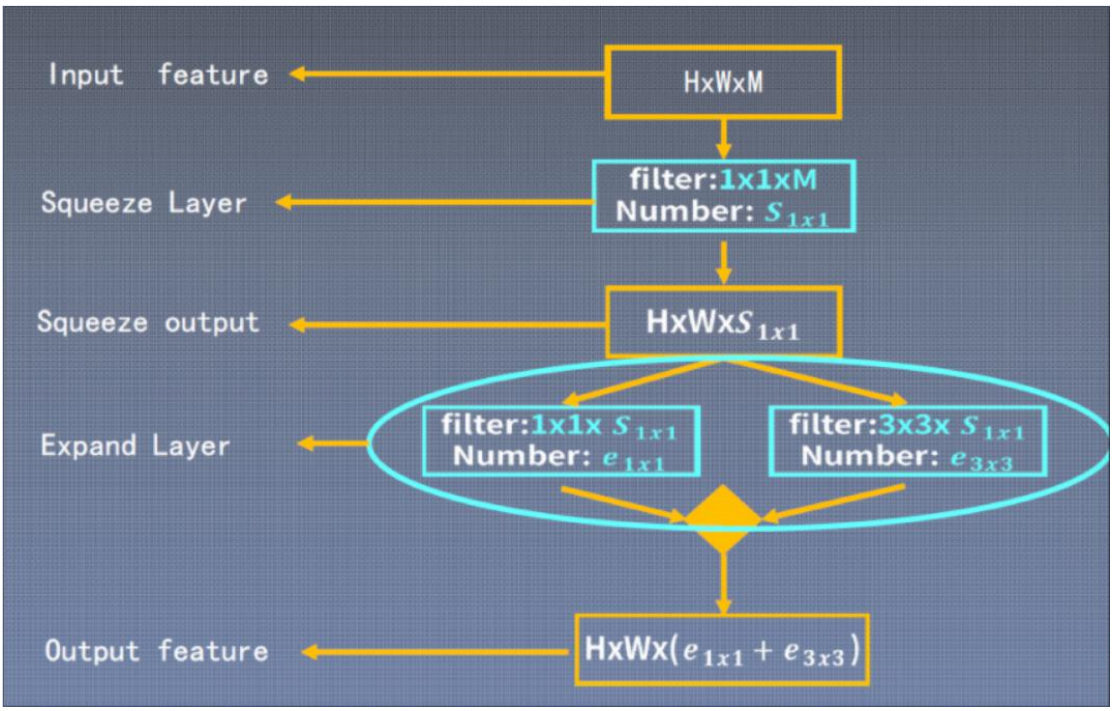




当我们输入一张图片的时候，我们利用不同大小的刻度，对一张图片进行了划分。下图中，利用了三种不同大小的刻度，对一张输入的图片进行了划分，最后总共可以得到 16+4+1=21 个块，我们即将从这 21 个块中，每个块提取出一个特征，这样刚好就是我们要提取的 21 维特征向量。

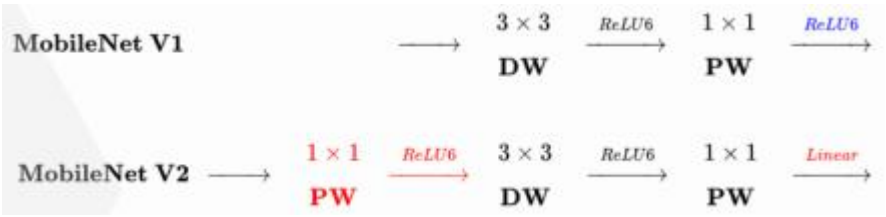


SqueezeNet 1*1 卷积替换 3*3 卷积，3*3 卷积采用更少的 channel． e1=e3=4s1

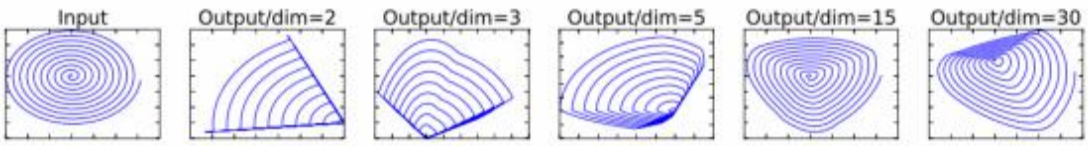


MobileNet 采用 Depthwise Convolution 和 Pointwise Convolution

MobileNet V2 把 relu 换成了 linear。因为 relu 在维数较低的时候会丢失部分分布。

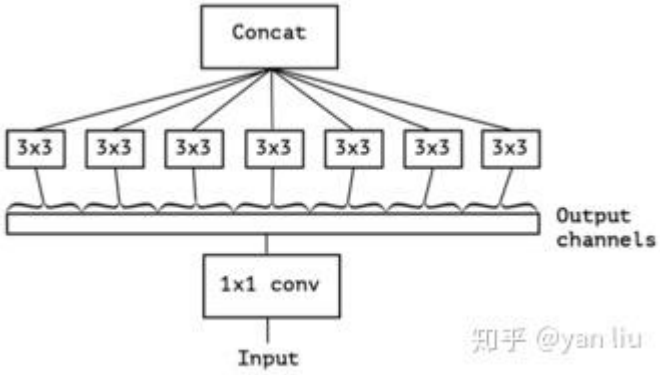


如图所示，relu 在维度为 2，3，5 的时候都有丢失分布的情况出现。



Xception

Xception 的结构和 MobileNet 非常像，两个算法的提出时间近似，不存在谁抄袭谁的问题。他们从不同的角度揭示了深度可分离卷积的强大作用，MobileNet 的思路是通过将层拆三份来减少参数数量，而 Xception 是每一层都拆出来来完成的。



ShuffleNet 首先分组卷积，将 M 个核分为 g 组，将上一层传入的 n 层输入也分为 g 组，分别卷积。得到 g 组大小为 h*w*m/g 的特征矩阵。然后将 g 组的 m/g 个 channel 混合 shuffle。