# Backbones

VGG：

1. 首次采用 3*3 的卷积核。
2. 结构简洁，卷积层+池化层。
3. 提高了模型的深度。

Resnet：

1. 采用 short-cut 来缓解由于网络过深造成的梯度消失。
2. 对于 50，101，152 的深度，用 1*1 卷积核降维。
3. 大量使用了 BN 层。
4. 模块化。

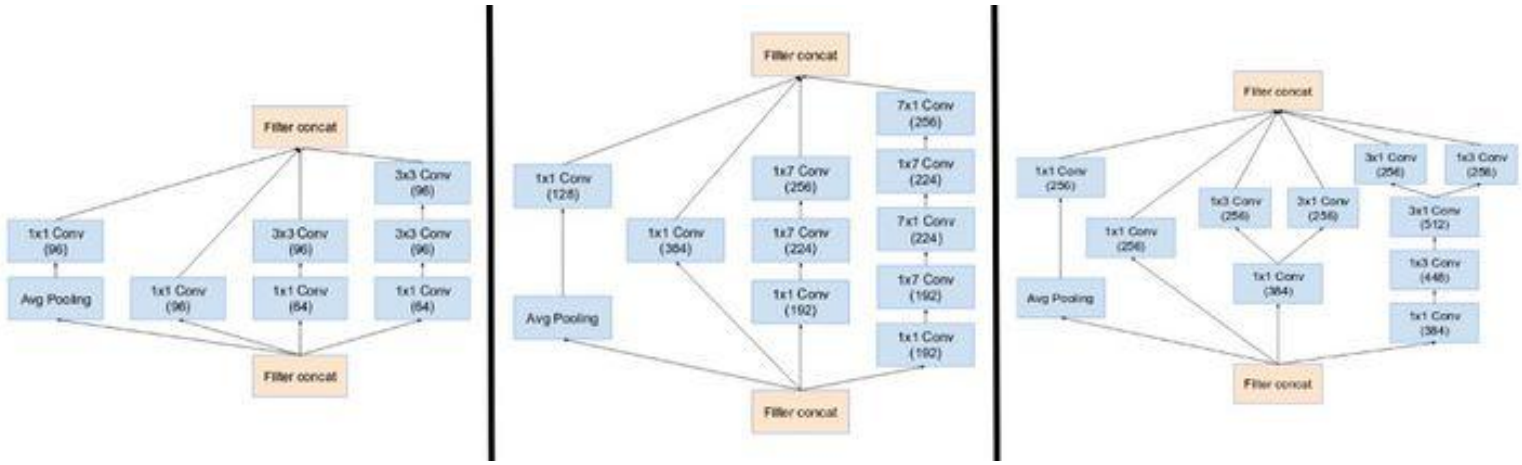
InceptionNet：

1. InceptionNet1：
    1. 用 1*1，3*3， 5*5 的卷积核来捕捉不同尺度的信息。
    2. 为了节省计算量，用 1*1 卷积来降维。
    3. 使用辅助分类分支来降低梯度消失的影响。
2. InceptionNet2：
    1. 用两个 3*3 取代 5*5 卷积。
    2. 将 3*3 分解为 1*3 和 3*1。
3. InceptionNet3：
    1. 使用 RMSProp 优化器。
    2. 在辅组分类分支中使用 BN。
    3. 使用了 Lable Smoothing。

4. InceptionNet4：

    1. 修改了前面几层网络的结构。

    2. 使用了三种不同的模块，A，B 和 C：




    3. 引入了“Reduction Blocks”，用于改变 H*W 的大小。

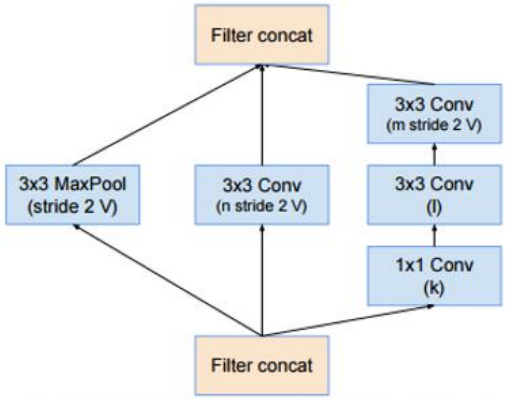



Figure 7. The schema for $35 \times 35$ to $17 \times 17$ reduction module. Different variants of this blocks (with various number of filters) are used in Figure 9, and 15 in each of the new Inception(-v4, -ResNet-v1, -ResNet-v2) variants presented in this paper. The $k$, $l$, $m$, $n$ numbers represent filter bank sizes which can be looked up in Table 1.

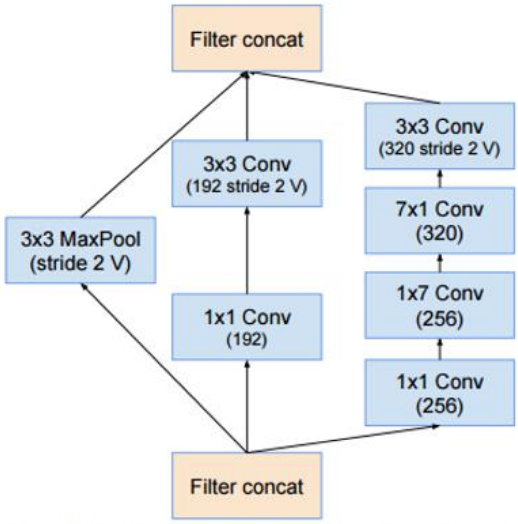



Figure 8. The schema for $17 \times 17$ to $8 \times 8$ grid-reduction module. This is the reduction module used by the pure Inception-v4 network in Figure 9.



A（从 35x35 到 17x17 的缩减）和 B（从 17x17 到 8x8 的缩减）

5. InceptionNet-ResNet：

    1. 引入 short-cut 路径

    2. 为了匹配输入和输出的尺寸，在 inception 卷积后加上 1*1 卷积。

MobileNets：

1. 使用 Deepwise Conv + Pointwise Conv 代替原始的卷积操作，来轻量化网络。计算量下降 1/9 到 1/8.

2. 适合部署到移动端或者嵌入式系统中。

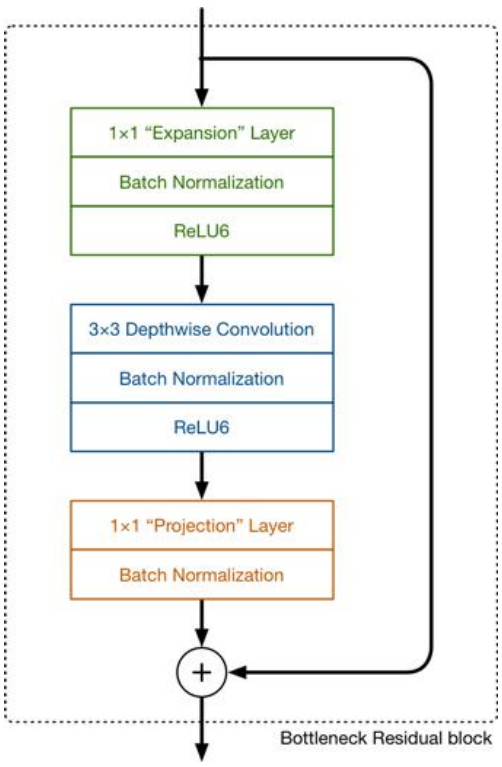
3. 引入宽度 α 和分辨率 ρ 缩放因子，他们的取值都是 0 到 1 之间，进一步缩小模型。具体来说 α对输入和输出通道数进行压缩，而 ρ 对 feature map size 进行压缩。

4. V1 中没有 pooling

5. 用 Relu6 代替 Relu

6. V2 中用了 short-cut，但是先扩展 6 倍再压缩。"Expansion" Layer。

7. 去掉每个模块最后一个 1*1 卷积后面的 relu。这是因为作者发现 ReLU 会对 channel 数较低的张量造成较大的信息损耗，因此执行降维的卷积层后面不会接类似于 ReLU 这样的非线性激活层。



DenseNet：

1. 将每一层都和其它的层连起来。

2. Feature map 数量很小(小于 100).

3. 每一层的 feature map 之间是 concatenate 的，不是像 resnet 那样是相加的。

4. DenseNet 分成多个 dense block，原因是希望各个 dense block 内的 feature map 的 size 统一，这样在做 concatenation 就不会有 size 的问题。

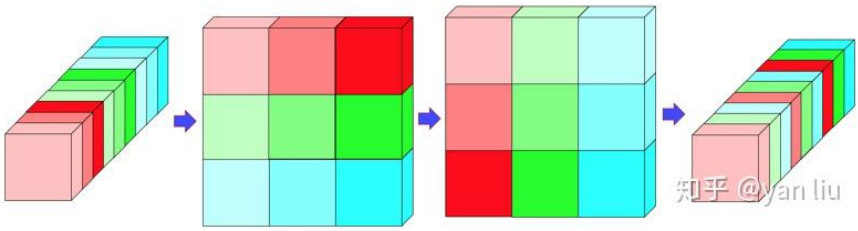
5. 每个 dense block 的 3*3 卷积前面都包含了一个 1*1 的卷积操作，即 bottleneck layer，目的是减少输入的 feature map 数量，这样既能降维减少计算量，又能融合各个通道的特征。

6. 两个 dense block 之间也加了一个 1*1 卷积。进一步压缩了参数。即 Translation layer。

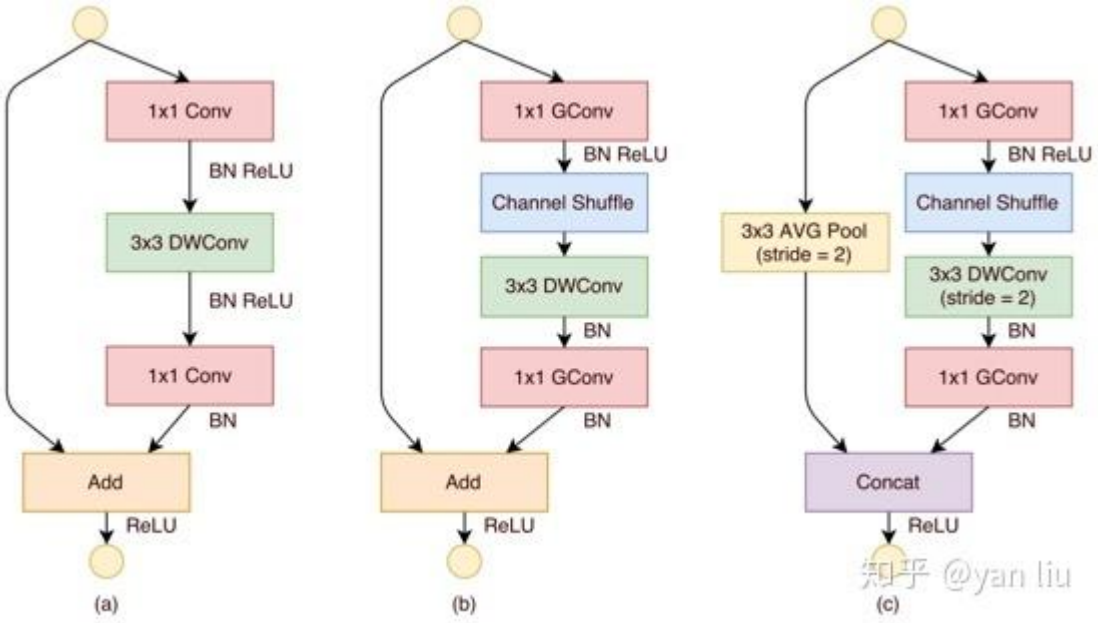
ShuffleNet：

1. MobileNet 的性能瓶颈在 Pointwise Conv 上。ShuffleNet v1 使用了分组卷积，将 feature map 和 filter 在 channels 上分成几组，每组 feature map 和本组的 filter 卷积。这样可以大大减少计算量。

2. 分组卷积会造成 feature map 的 channels 之间信息无法互通。因此为了解决这个问题，提出了 Channel Shuffle 思想。假设分组 Feature Map 的尺寸为 W*H*C1，C1=g*n，其中 g 表示分组的组数。将 Feature Map 展开成 g*n*w*h 的四维矩阵。再沿着尺寸为 g*n*w*h 的矩阵的 g 轴和 n 轴进行转置。将 g 轴和 n 轴进行平铺后得到洗牌之后的 Feature Map。再进行组内的 1*1 卷积。



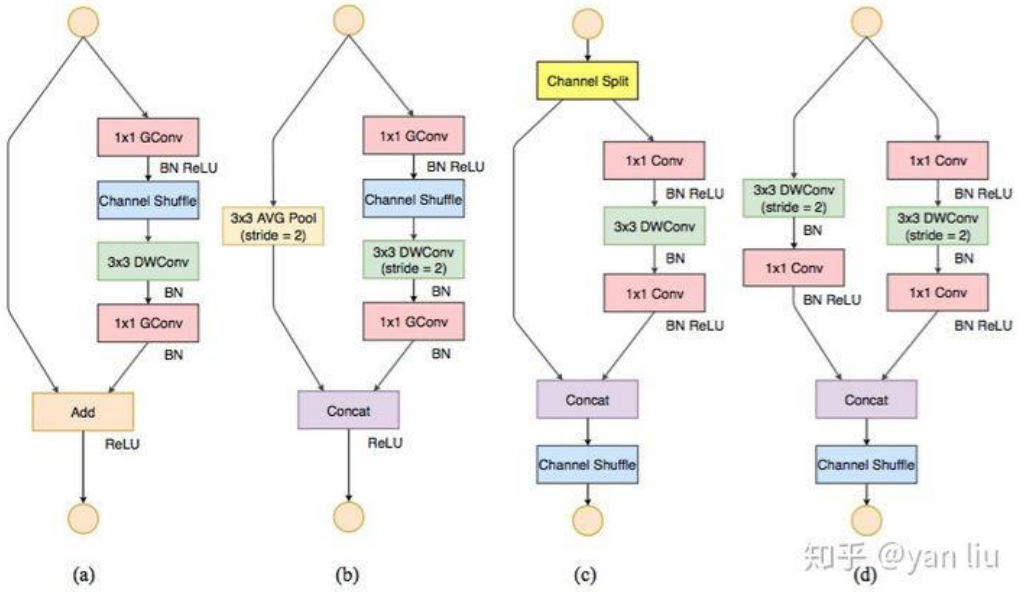

3. ShuffleNet V1 单元如下：




上下两个红色部分的 1*1 卷积替换为 1*1 的分组卷积，分组 g 一般不会很大，论文中的几个值分别是 1，2，3，4，8。g 的值确保能够被通道数整除，保证 reshape 操作的有效执行。在第一个 1*1 卷积之后添加 Channel Shuffle 操作。如图 3.(c)中需要降采样的情况，左侧 shortcut 部分使用的是步长为 2 的 3*3 平均池化，右侧使用的是步长为 2 的 3*3 的 Depthwise 卷积。去掉了 3*3 卷积之后的 ReLU 激活，目的是为了减少 ReLU 激活造成的信息损耗. 如果进行了降采样，为了保证参数数量不骤减，往往需要加倍通道数量。所以在 3.(c)中使用的是 Concat，而 3.(b)中则是相加。

4. ShuffleNet V2 中，作者提出了设计高性能网络的 4 点要求：G1). 使用输入通道和输出通道相同的卷积操作；G2). 谨慎使用分组卷积；G3). 减少网络分支数；G4). 减少 element-wise 操作。按照这个要求，在 ShuffleNet v1

中使用的分组卷积是违背 G2 的，而每个 ShuffleNet v1 单元使用了 bottleneck 结构是违背 G1 的。MobileNet v2 中的大量分支是违背 G3 的，在 Depthwise 处使用 ReLU6 激活是违背 G4 的。



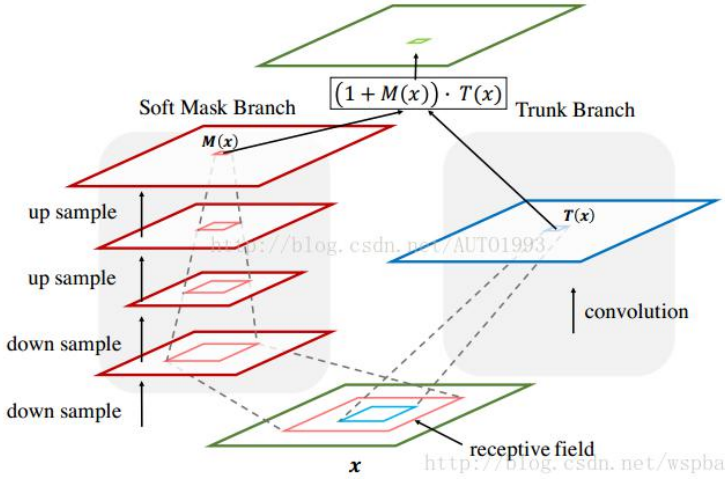

5. ShuffleNet v2 的结构，如图 c 和 d。在 c 中使用了一个 Channel Split 操作，这个操作将 c 个输入 Feature 分成 c-c'和 c'，一般来说 c'=c/2.这个设计是为了尽量控制分支数，满足 G3. 分割后，一个分支直接映射，一个是输入通道和输出通道数均相同的深度可分离卷积，满足了 G1. 在 c 和 d 中，右侧的通道没有使用 1*1 分组卷积，满足了 G2. 最后合并的时候用了 concat，满足了 G4. 在最后，通道拼接，通道洗牌和通道分割合并成 1 个 element-wise 操作，满足了 G4。

Residual Attention Net：

1. 由多层注意力模块堆叠而成，每个注意力模块包含了两个分支：掩膜分支（mask branch）和主干分支（trunk branch）。其中主干分支可以是当前的任何一种 SOTA 卷积神经网络模型，掩膜分支通过对特征图的处理输出维度一致的注意力特征图（Attention Feature Map），然后使用点乘操作将两

个分支的特征图组合在一起，得到最终的输出特征图。假如主干分支输出特征图为 T_{i,c}(x)，掩膜分支的输出特征图为 M_{i,c}(x)，那么最终该注意力模块的输出特征图为：$H_{i,c}(x) = T_{i,c}(x) * M_{i,c}(x)$

2. mask branch 与主干分支点乘，会使特征图的输出响应变弱，多层叠加这样的结构会使特征图每一个点的值变得很小。同时，会使得梯度不好回传。因此，使用公示$H_{i,c}(x) = (1 + M_{i,c}(x)) * F_{i,c}(x)$替代。



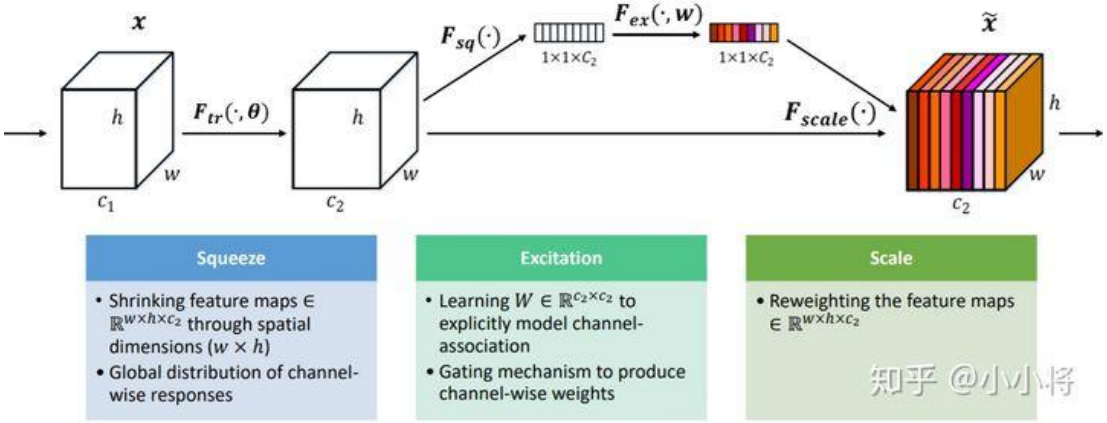

3. mask branch 是先提取特征，再上采样，过程中与之前未降采样的特征组合在一起。然后使用 2 个 1x1 的卷积层对通道做整合计算输出一个与 input 宽高维度相等，但是通道数为 1 的特征图，最后接一个 Sigmoid 激活函数层将特征图归一化到 0~1 之间。

4. Attention 由三种方式，空间 Attention，Channel Attention 和混合 Attention。

SENet：

1. 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。

2. 提出了 Squeeze-and-Excitation (SE)模块，如下图所示：




3. 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。

4. SE 模块是在 channel 维度上做 attention 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。

5. SE 模块是一个子结构,可以嵌到其他分类或检测模型中。