Classification network

Zhang Shiyin

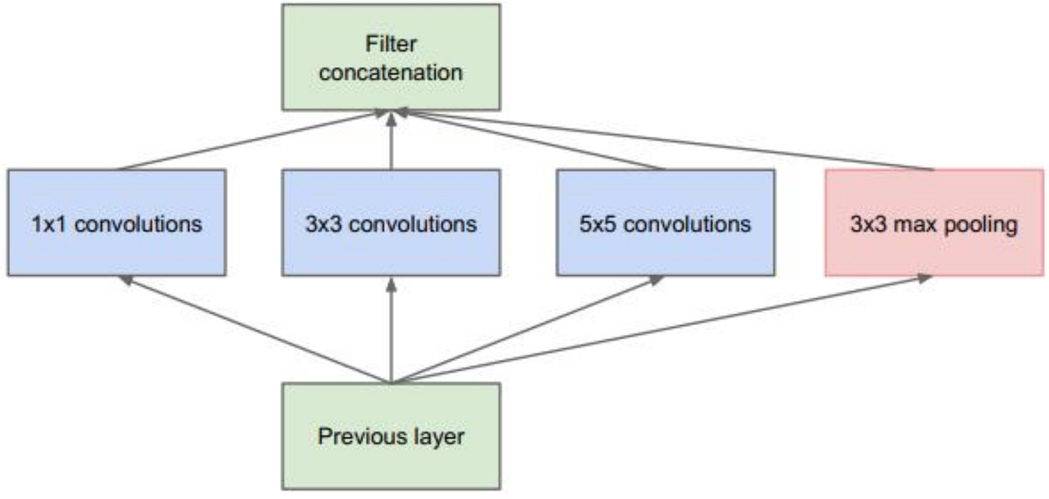
2020/12/12

- GoogLeNet
- Inception v3
- ResNet
- ResNeXt
- DenseNet
- Deformable Convolutional Networks

- SqueezeNet
- MobileNet V1 and V2

GoogLeNet： Going Deeper with Convolutions

（1）GoogLeNet相对于采用了模块化的结构；



(a) Inception module, naïve version

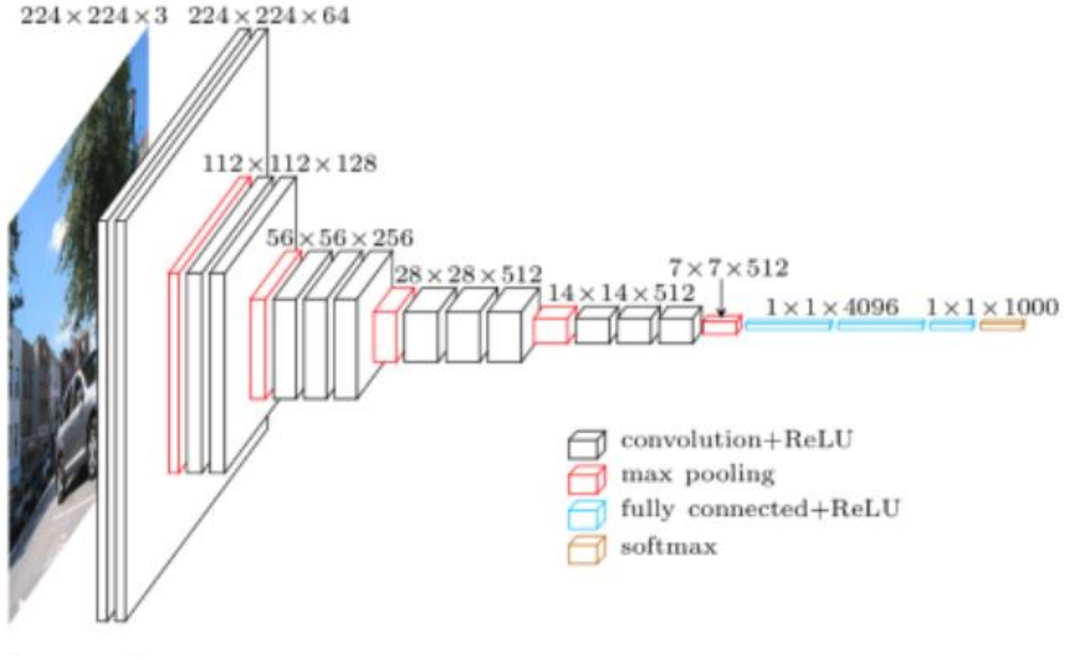
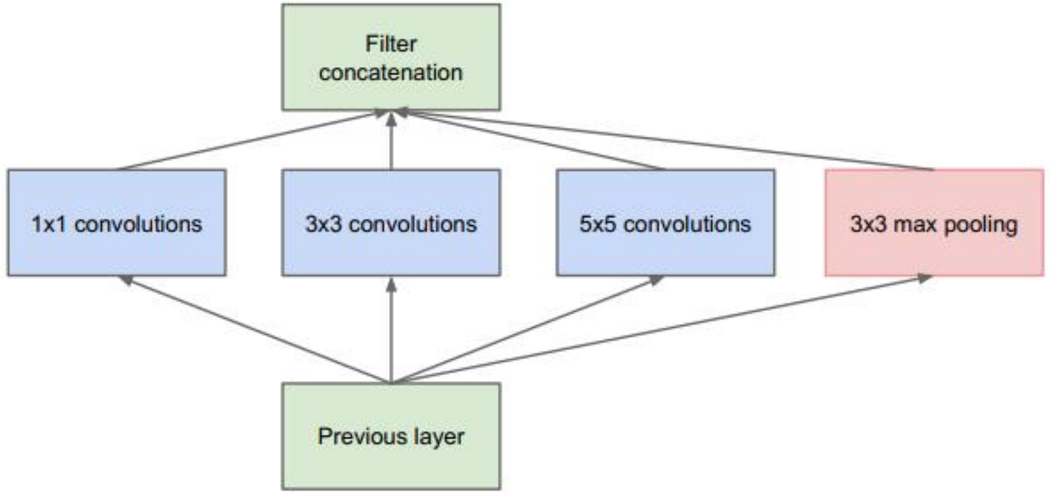


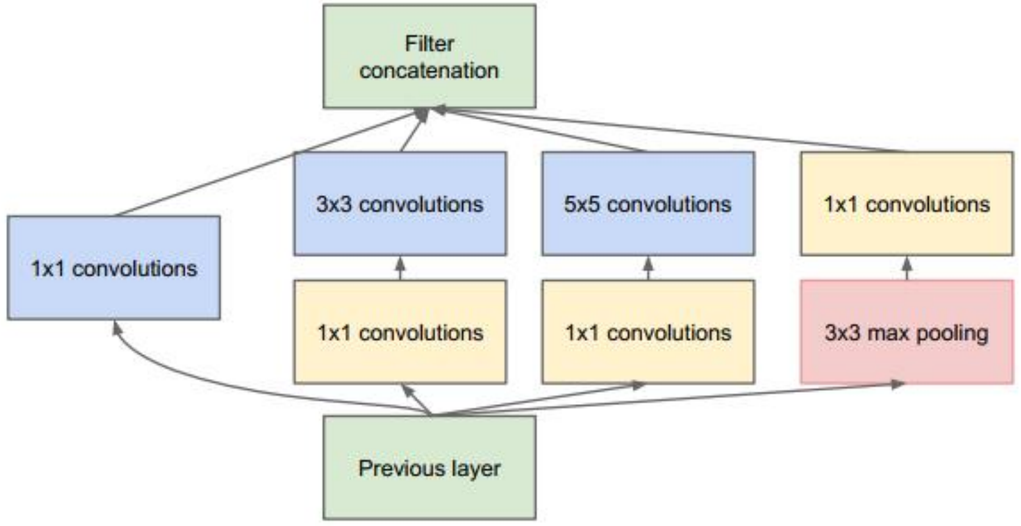
Figure 2: Inception module

GoogLeNet： Going Deeper with Convolutions

（1）GoogLeNet相对于采用了模块化的结构；



(a) Inception module, naïve version



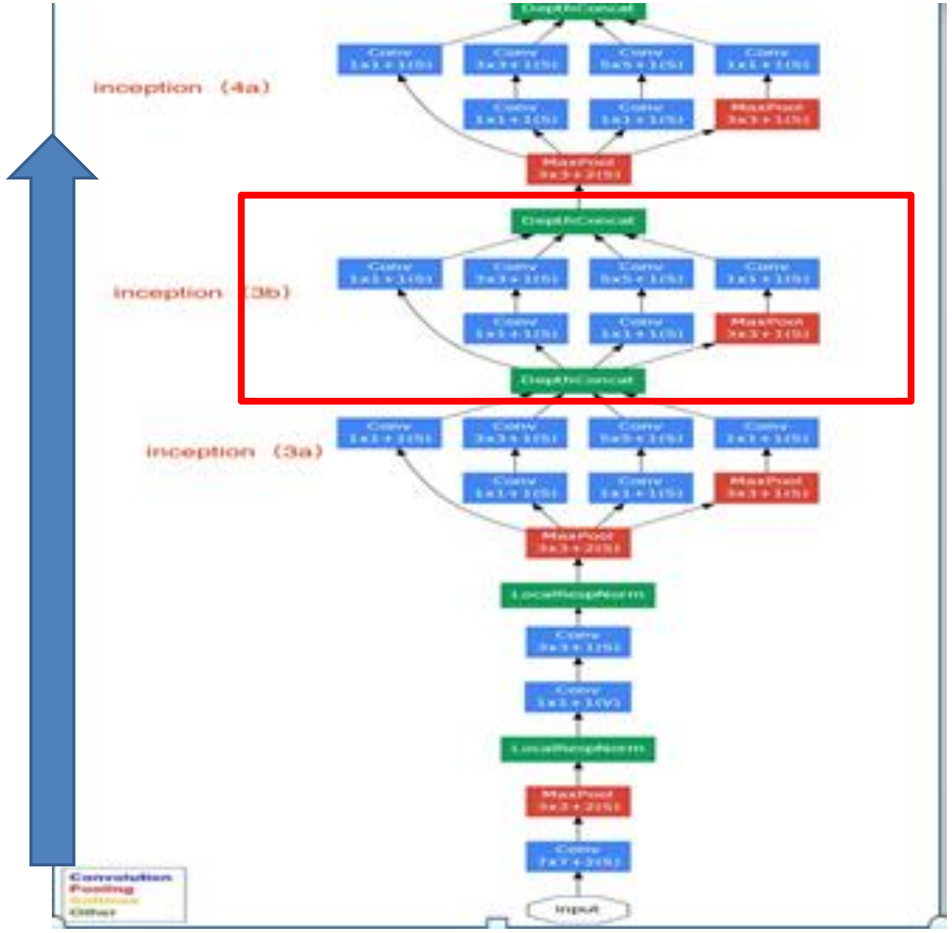
(b) Inception module with dimension reductions

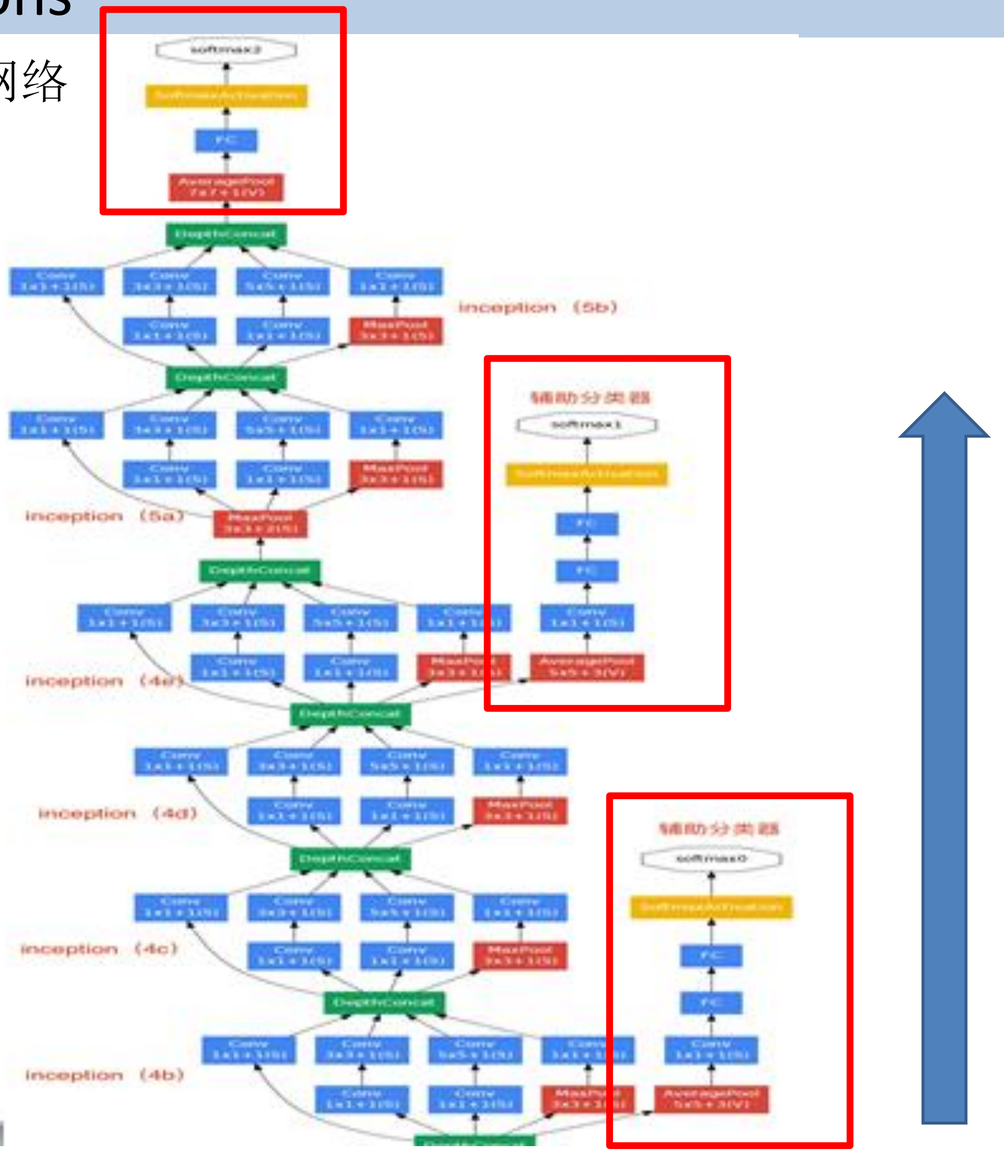
1x1conv降低featuremap的维度

Figure 2: Inception module

GoogLeNet： Going Deeper with Convolutions

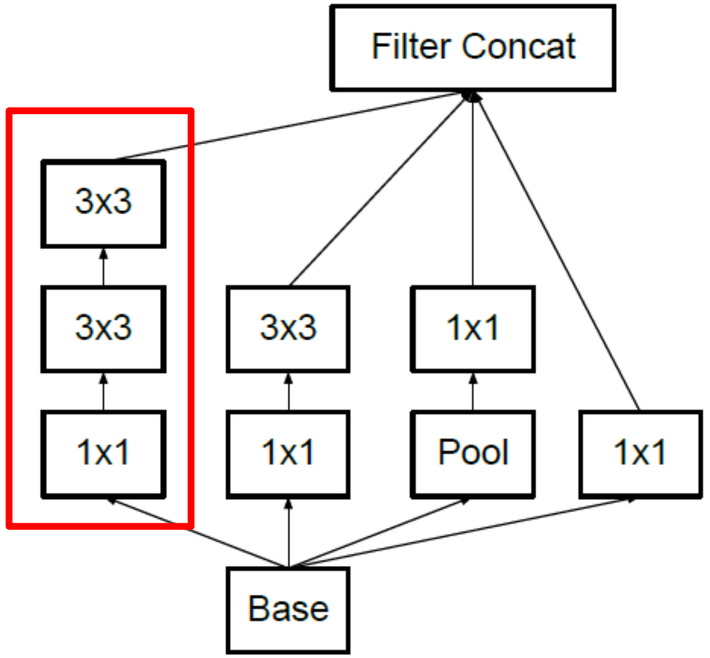
（2）网络额外增加了2个辅助分类器（0.3），这样给网络增加了反向传播的梯度信号。

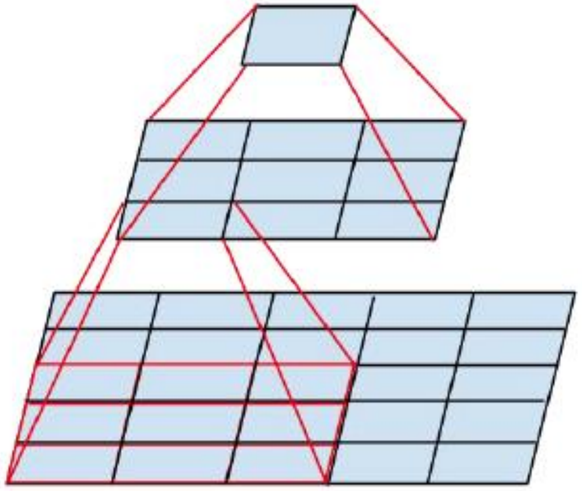




Inception v3：Rethinking the Inception Architecture for Computer Vision

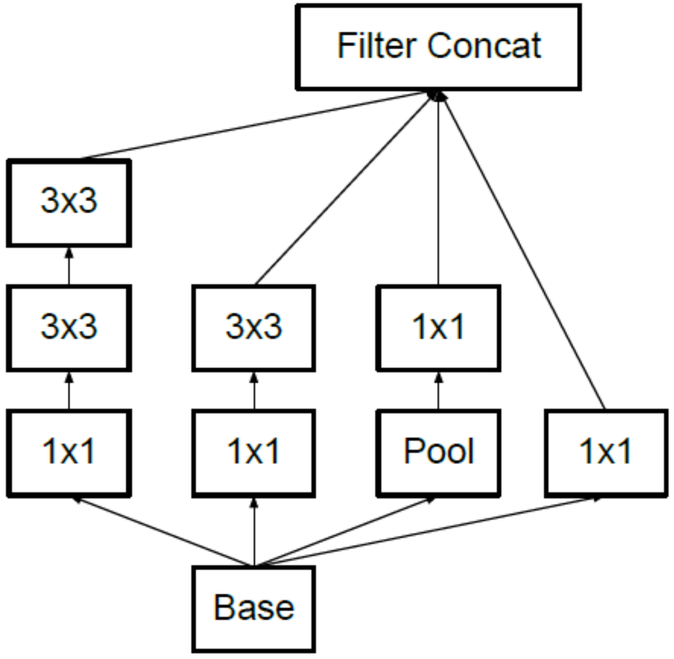
使用两个3x3卷积核代替5x5卷积核，减少参数量，加快计算。

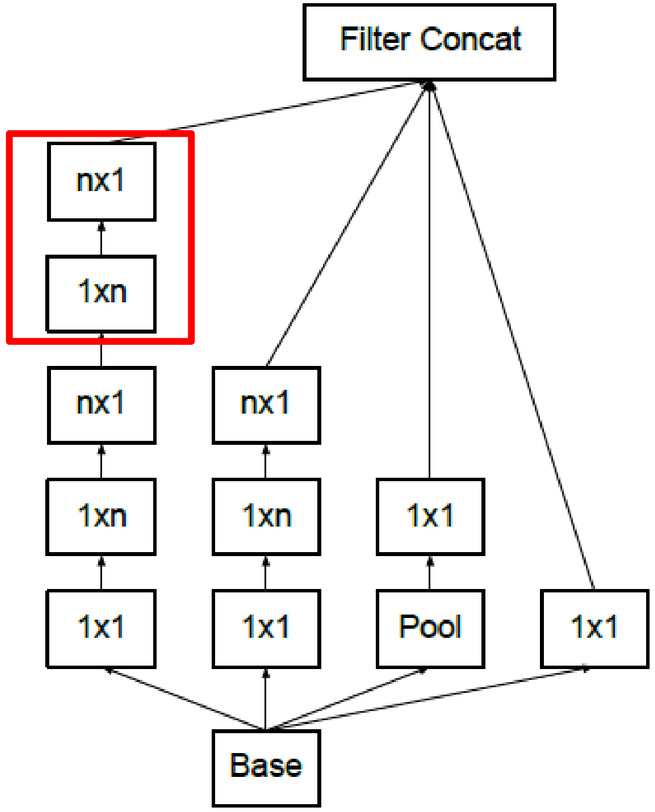


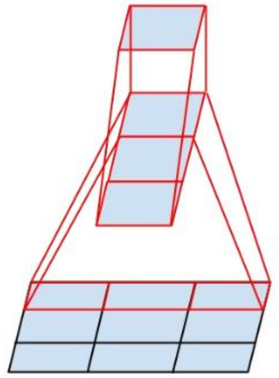


Inception v3： Rethinking the Inception Architecture for Computer Vision

进一步将nxn卷积核分解为1xn和nx1卷积核







Resnet： Deep Residual Learning for Image Recognition

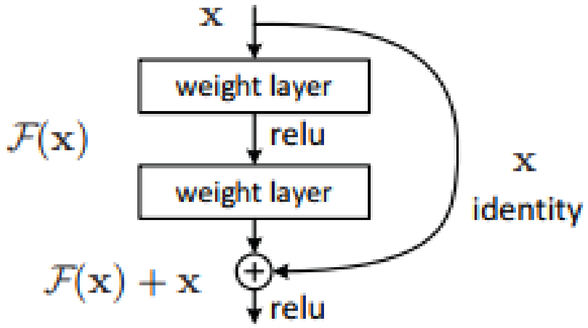


Figure 2. Residual learning: a building block.

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNeXt: Aggregated Residual Transformations for Deep Neural Networks

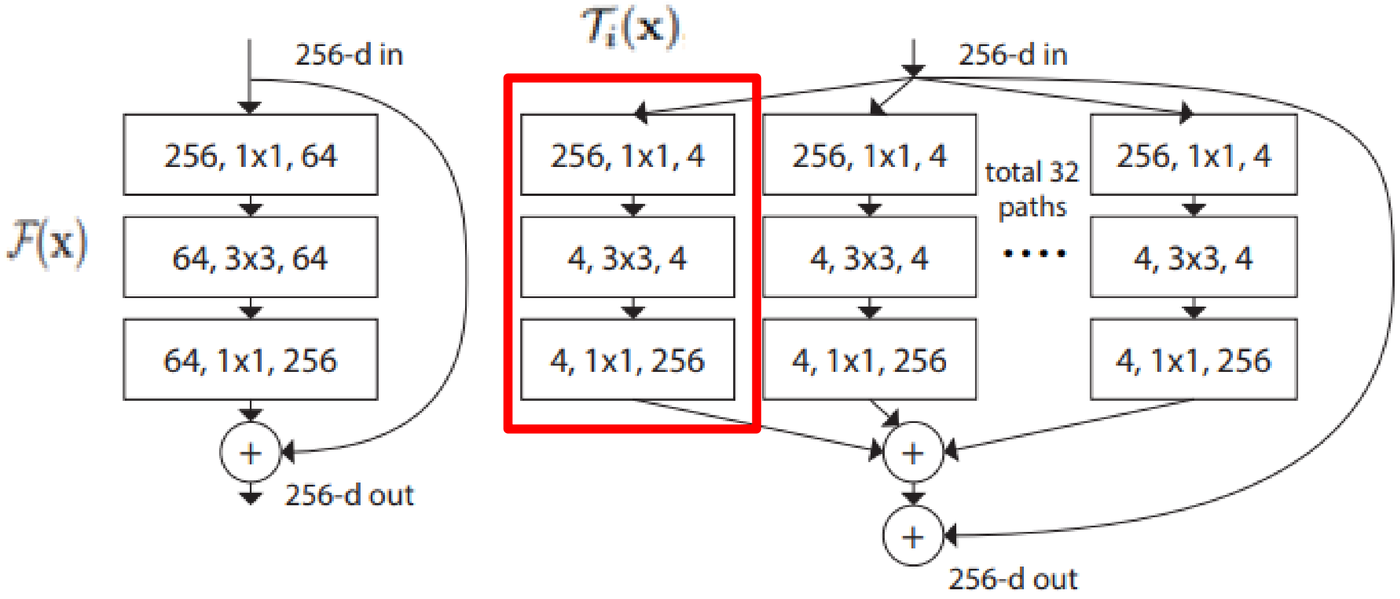
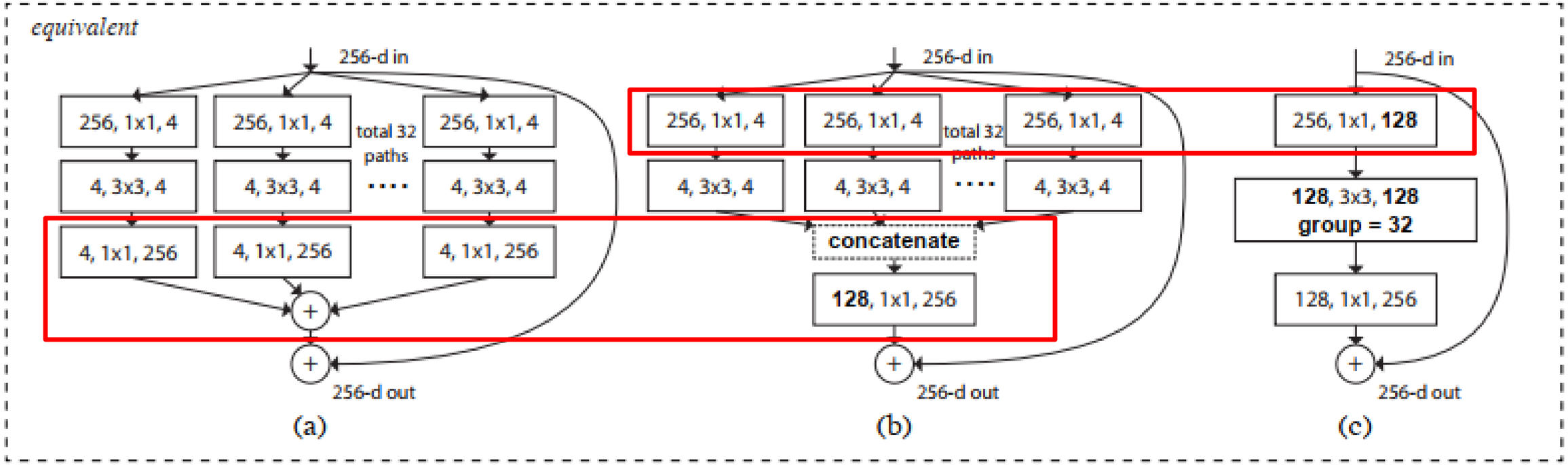


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x}),$$

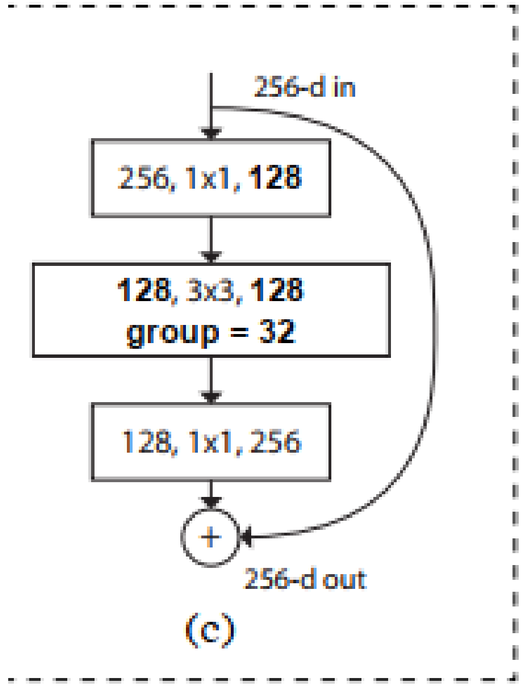
$$\mathbf{y} = \mathbf{x} + \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x}),$$

ResNeXt: Aggregated Residual Transformations for Deep Neural Networks

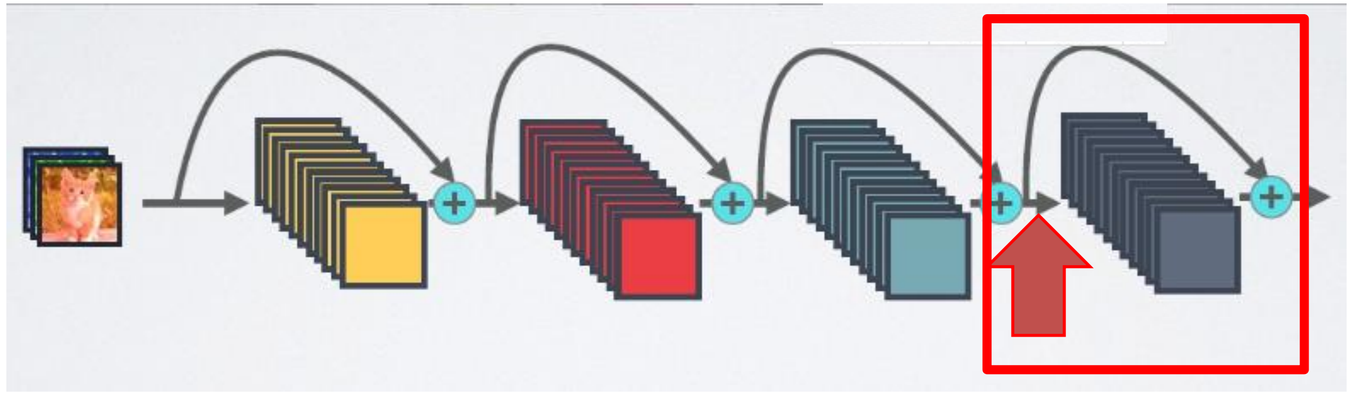


ResNeXt: Aggregated Residual Transformations for Deep Neural Networks

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.5}\times10^6$	$\mathbf{25.0}\times10^6$
FLOPs		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$



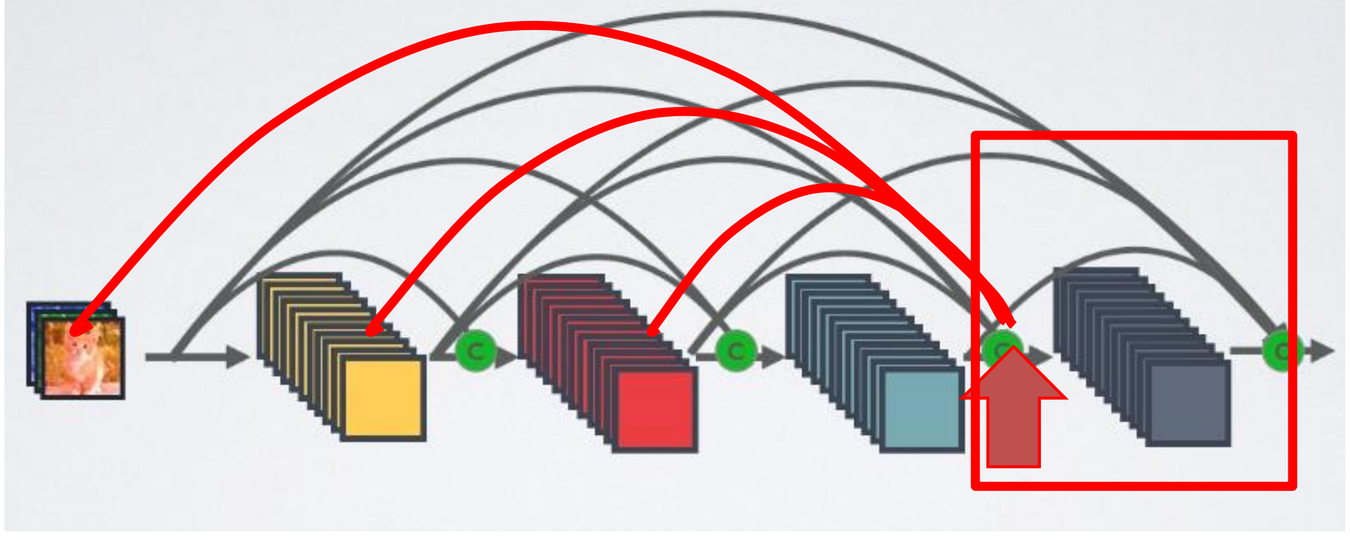
DenseNet： **Densely Connected Convolutional Networks**



ResNets.

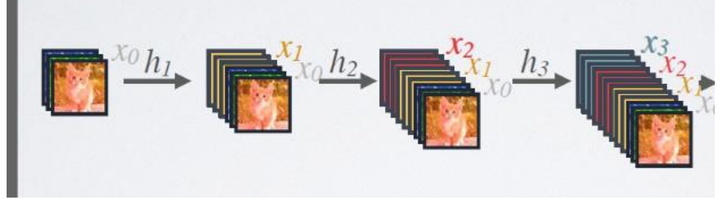
$$\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}.$$

Dense block：每个层都会接受其前面所有层作为其额外的输入



Dense connectivity

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]),$$



DenseNet： **Densely Connected Convolutional Networks**

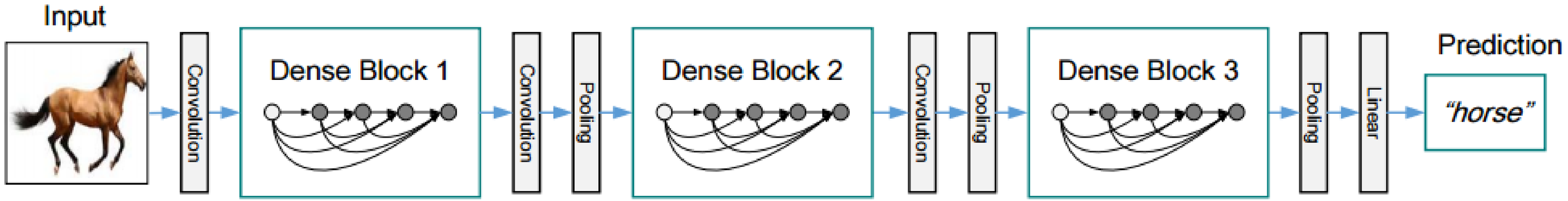


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

DenseNet网络中使用DenseBlock+Transition的结构，Transition模块是连接两个相邻的DenseBlock，并且通过Pooling使特征图大小降低，同时降维。

DenseNet： **Densely Connected Convolutional Networks**

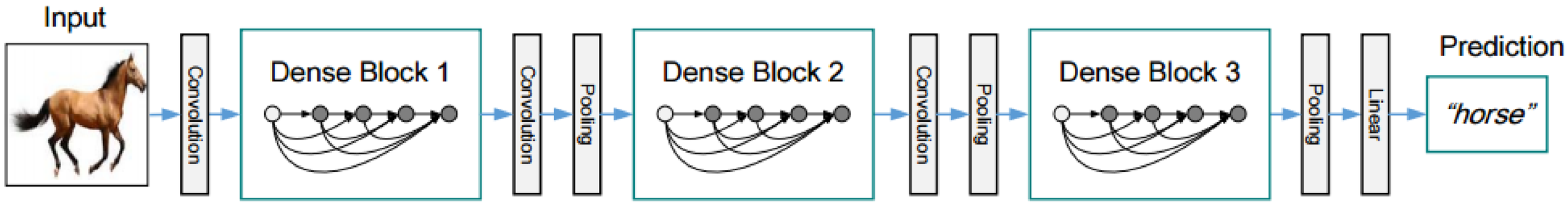
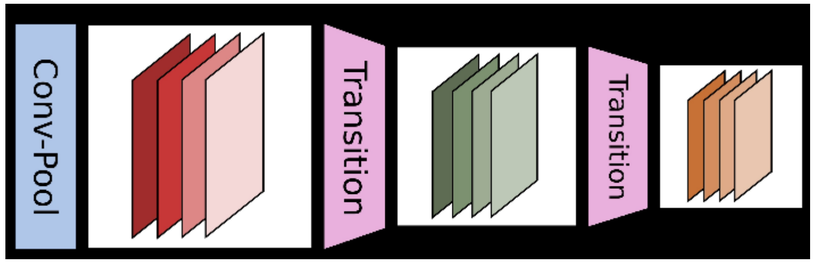


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

DenseNet网络中使用DenseBlock+Transition的结构，Transition模块是连接两个相邻的DenseBlock，并且通过Pooling使特征图大小降低，同时降维。



Transition包括一个1x1的卷积和2x2的AvgPooling

Transition层可以起到压缩模型的作用

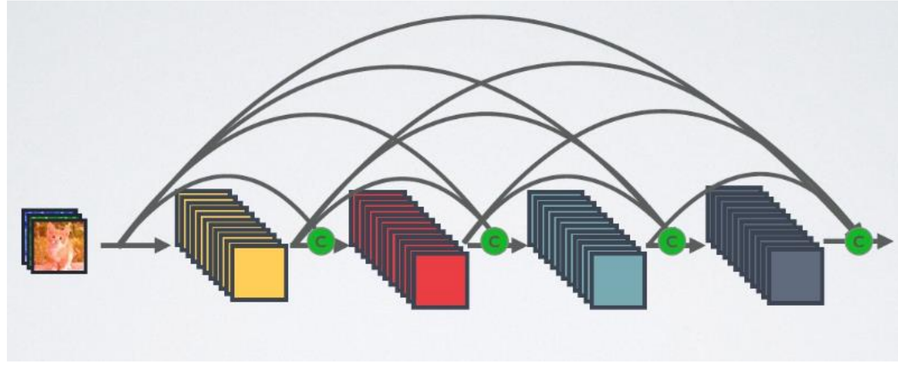
DenseNet： **Densely Connected Convolutional Networks**

```
class Transition(nn.Module):
    def __init__(self, nChannels, nOutChannels):
        super(Transition, self).__init__()
        self.bn1 = nn.BatchNorm2d(nChannels)
        self.conv1 = nn.Conv2d(nChannels, nOutChannels, kernel_size=1,
                               bias=False)

    def forward(self, x):
        out = self.conv1(F.relu(self.bn1(x)))
        out = F.avg_pool2d(out, 2)
        return out
```

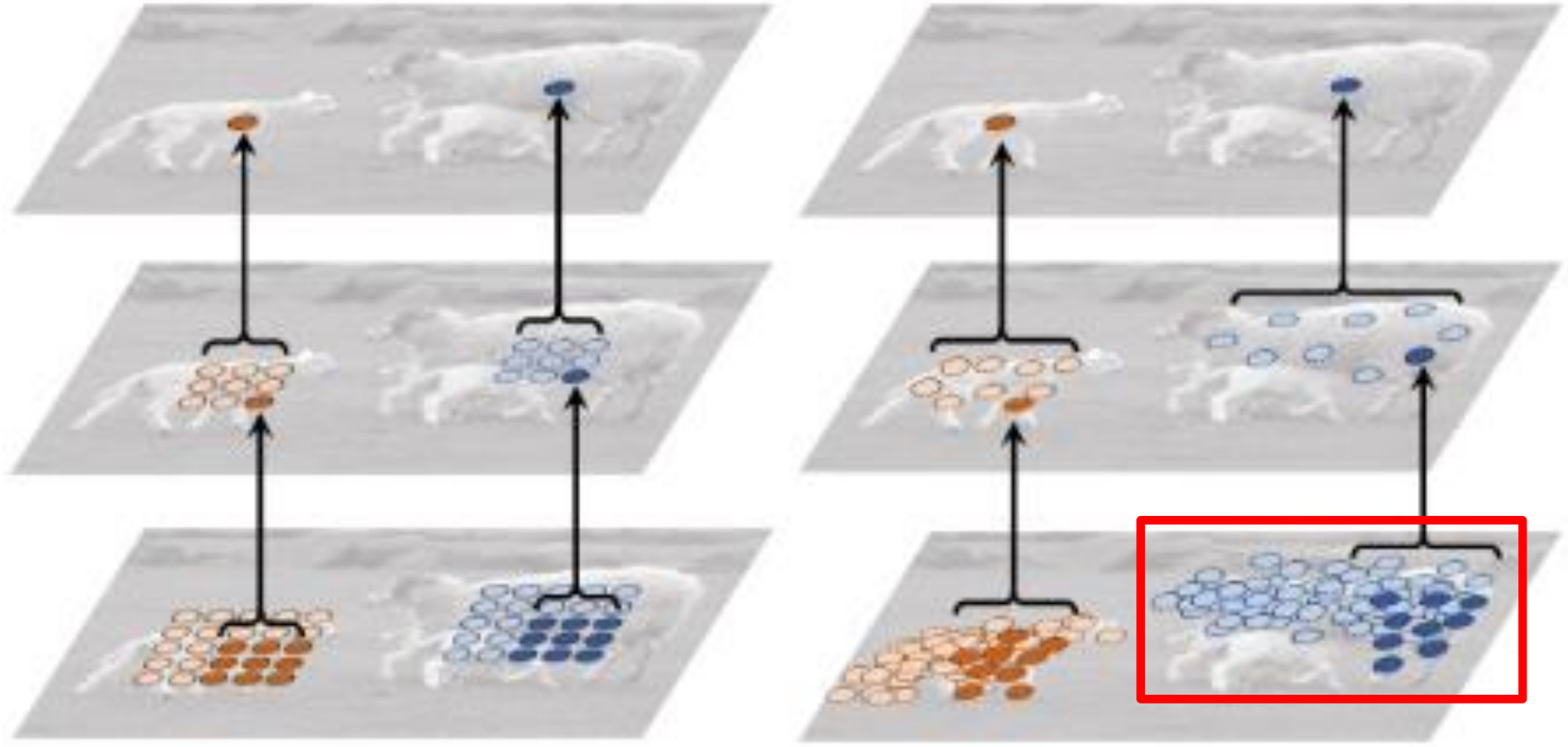
```
nOutChannels = int(math.floor(nChannels*reduction))
```

```
def _make_dense(self, nChannels, growthRate, nDenseBlocks, bottleneck):
    layers = []
    for i in range(int(nDenseBlocks)):
        if bottleneck:
            layers.append(Bottleneck(nChannels, growthRate))
        else:
            layers.append(SingleLayer(nChannels, growthRate))
        nChannels += growthRate
    return nn.Sequential(*layers)
```

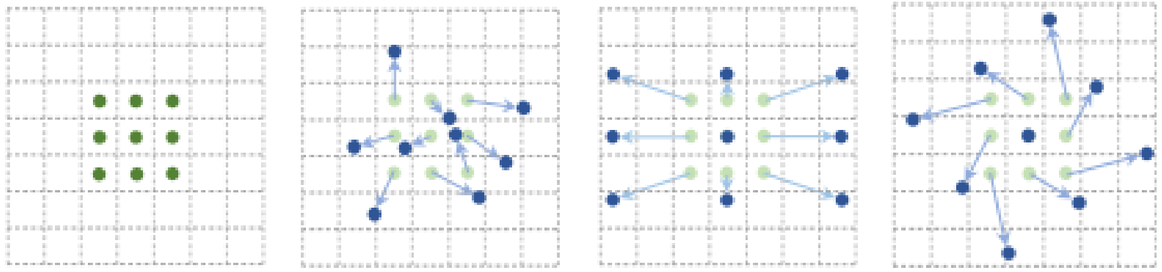


Deformable Convolutional Networks

传统卷积使用固定形状的卷积核，感受野是规则的。而可变形卷积考虑到不同目标的外形、大小，映射到前面层的采样点大多会覆盖在目标上面，采样到更多我们感兴趣的信息。



Deformable Convolutional Networks



(a) (b) (c) (d)

Figure 1: Illustration of the sampling locations in 3×3 standard and deformable convolutions. (a) regular sampling grid (green points) of standard convolution. (b) deformed sampling locations (dark blue points) with augmented offsets (light blue arrows) in deformable convolution. (c)(d) are special cases of (b), showing that the deformable convolution generalizes various transformations for scale, (anisotropic) aspect ratio and rotation.

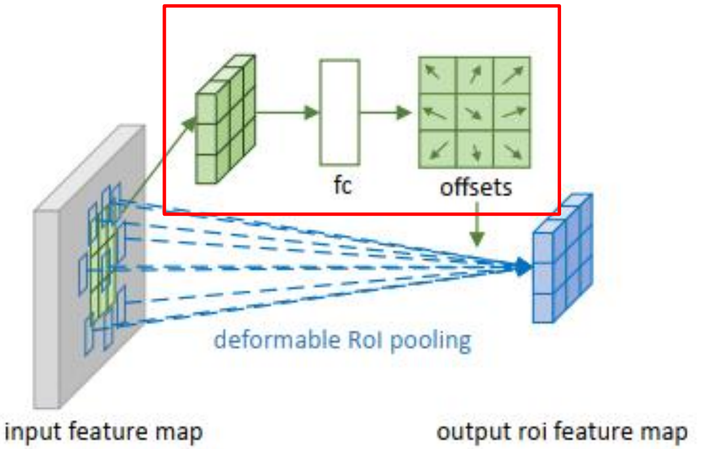


Figure 3: Illustration of 3×3 deformable RoI pooling.

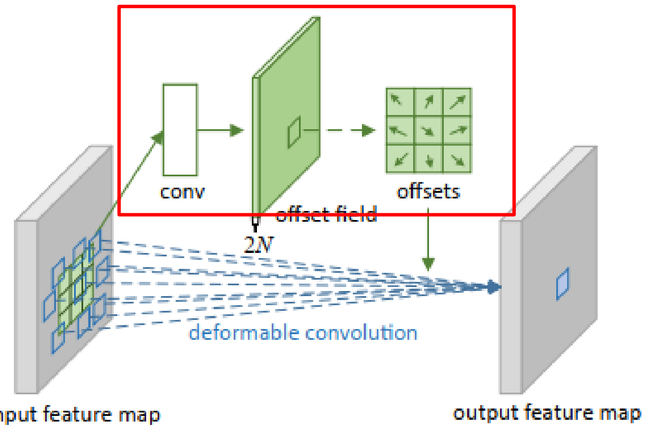


Figure 2: Illustration of 3×3 deformable convolution.

Deformable Convolutional Networks

Deformable convolution

For each location $\mathbf{p}_0$ on the output feature map $\mathbf{y}$,

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n), \qquad (1)$$

$$\mathcal{R} = \{(-1,-1), (-1,0), \ldots, (0,1), (1,1)\}$$

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n). \qquad (2)$$

Now, the sampling is on the irregular and offset locations $\mathbf{p}_n + \Delta\mathbf{p}_n$. As the offset $\Delta\mathbf{p}_n$ is typically fractional, Eq. (2) is implemented via bilinear interpolation as

$$\mathbf{x}(\mathbf{p}) = \sum_{\mathbf{q}} G(\mathbf{q}, \mathbf{p}) \cdot \mathbf{x}(\mathbf{q}), \qquad (3)$$

where $\mathbf{p}$ denotes an arbitrary (fractional) location ($\mathbf{p} = \mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n$ for Eq. (2)), $\mathbf{q}$ enumerates all integral spatial locations in the feature map $\mathbf{x}$, and $G(\cdot,\cdot)$ is the bilinear interpolation kernel. Note that G is two dimensional. It is separated into two one dimensional kernels as

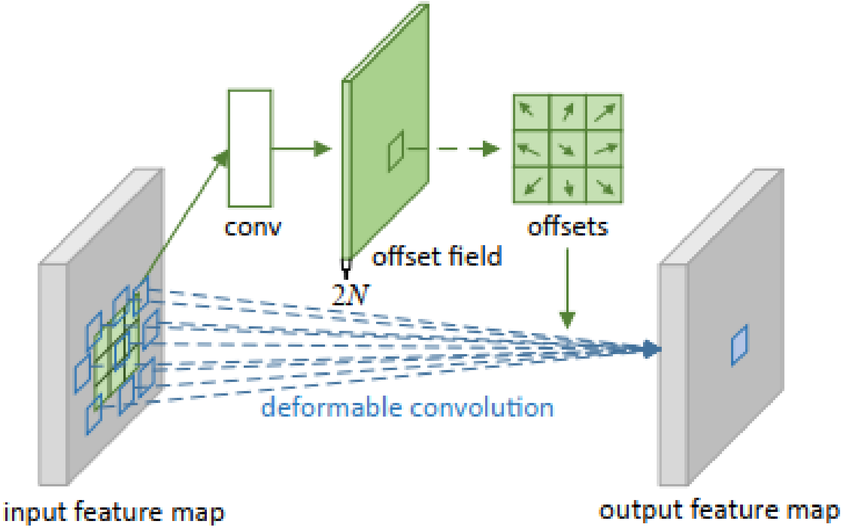


Figure 2: Illustration of 3×3 deformable convolution.

Deformable Convolutional Networks

Deformable convolution

额外的channel为72：对应着3X3的kernel size， 偏移（x,y），预测4个group。因此最后的输出channel 是72。4个group: 由于不同的通道负责检测不同的部分，可能会有不同的几何变化，因此在**deformable_group**较大的情况下具有更大的灵活性.

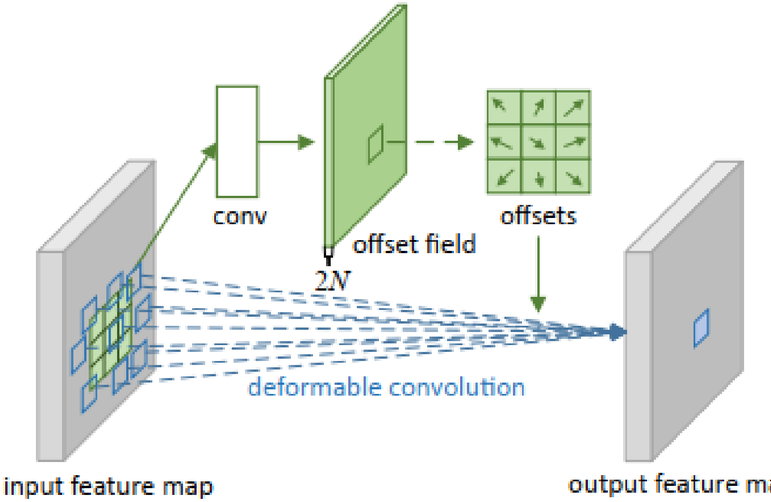


Figure 2: Illustration of 3 × 3 deformable convolution.

Deformable Convolutional Networks

Deformable RoI pooling

RoI被分为3*3个bin，被输入到一个额外的fc层来学习offset，然后通过一个deformable RoI pooling层来操作使每个bin发生偏移。

$$\mathbf{y}(i,j) = \sum_{\mathbf{p}\in bin(i,j)} \mathbf{x}(\mathbf{p}_0 + \mathbf{p})/n_{ij},$$

$$\mathbf{y}(i,j) = \sum_{\mathbf{p}\in bin(i,j)} \mathbf{x}(\mathbf{p}_0 + \mathbf{p} + \Delta\mathbf{p}_{ij})/n_{ij}.$$

首先将输入的featrue map进行roi pooling为3*3大小的feature，然后通过全连接

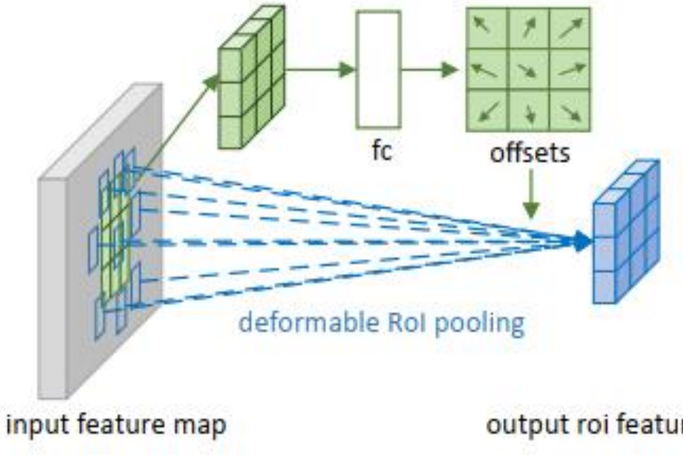
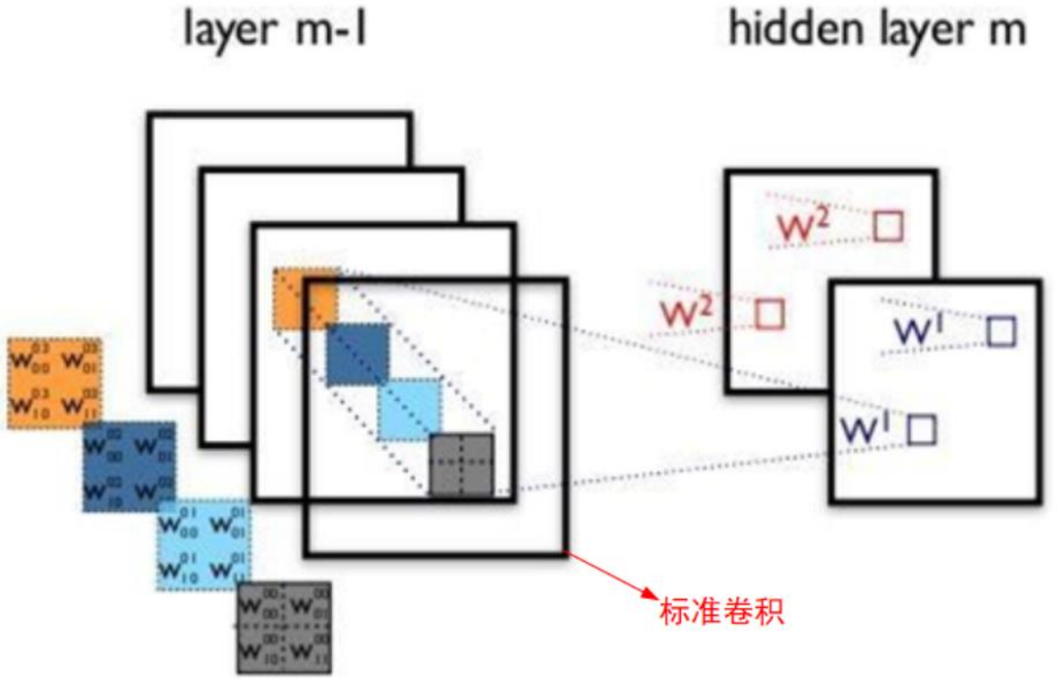
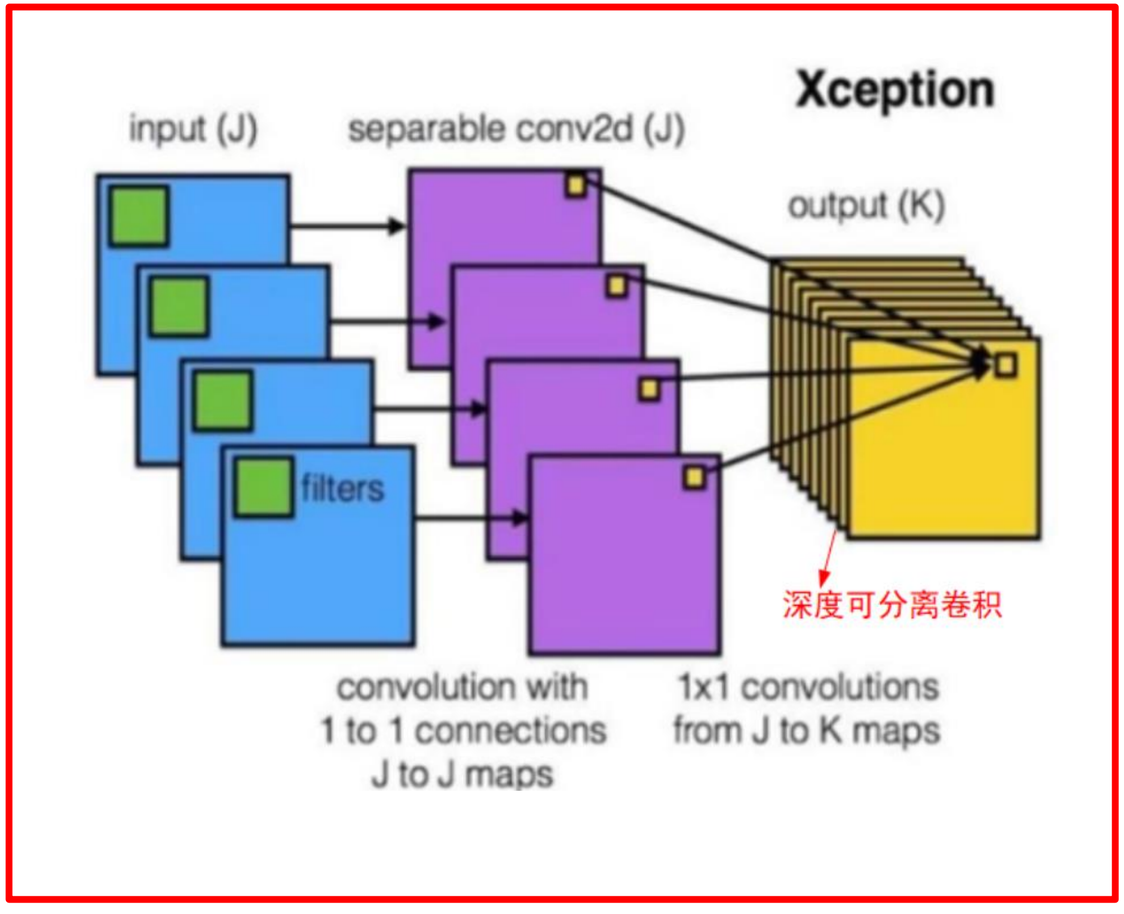


Figure 3: Illustration of 3 × 3 deformable RoI pooling.

MobileNetV1: Efficient Convolutional Neural Networks for Mobile Vision Application

MobileNetv1 使用Depthwise Separable Convolution，它将跨通道的3x3卷积换成单通道的3x3卷积+跨通道的1x1卷积来达到减少参数数量和计算量目的。





MobileNetV1: Efficient Convolutional Neural Networks for Mobile Vision Application

MobileNetv1 使用Depthwise Separable Convolution，它将跨通道的3x3卷积换成单通道的3x3卷积+跨通道的1x1卷积来达到减少参数数量和计算量目的。

计算量（ **FLOPs** ）：

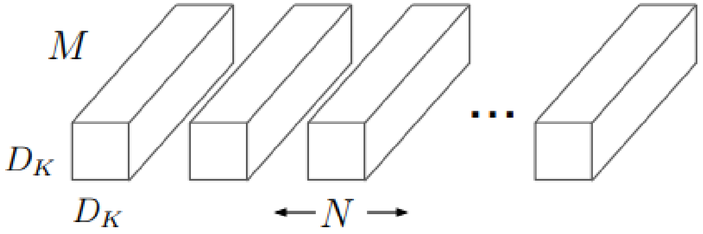
Standard convolutions have the computational cost of:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F \qquad (2)$$

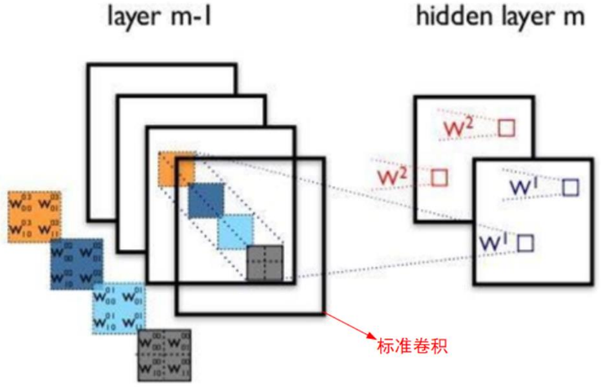
Params

Depthwise separable convolutions cost:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$



(a) Standard Convolution Filters



MobileNetV1: Efficient Convolutional Neural Networks for Mobile Vision Application

MobileNetv1 使用Depthwise Separable Convolution，它将跨通道的3x3卷积换成单通道的3x3卷积+跨通道的1x1卷积来达到减少参数数量和计算量目的。

计算量（ **FLOPs** ）：

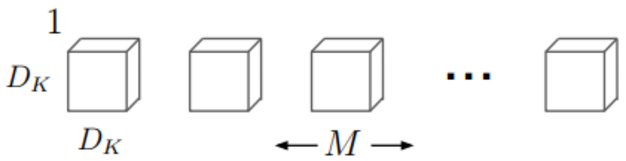
Standard convolutions have the computational cost of:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F \qquad (2)$$

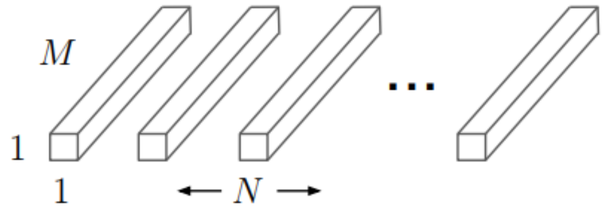
Depthwise separable convolutions cost:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

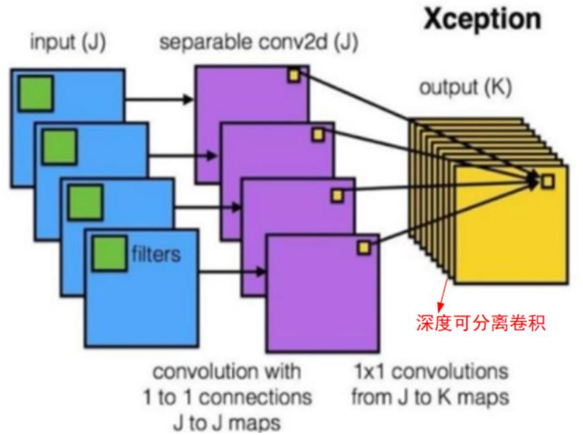
Params



(b) Depthwise Convolutional Filters



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



MobileNetV1: Efficient Convolutional Neural Networks for Mobile Vision Application

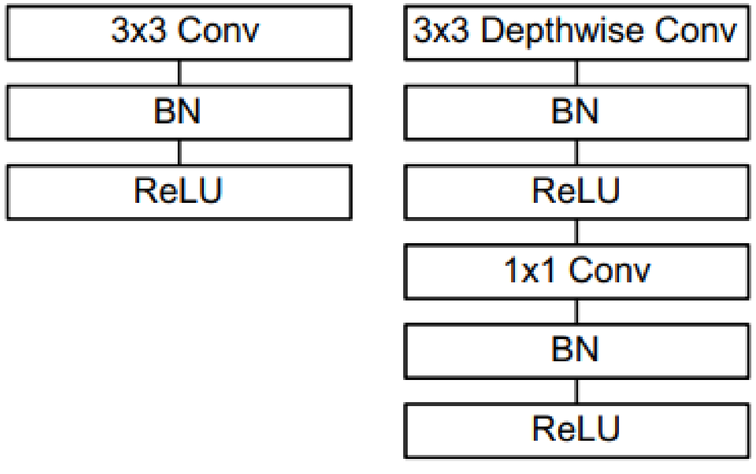


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

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	$1 \times 1 \times 1000$

MobileNetV1: Efficient Convolutional Neural Networks for Mobile Vision Application

两个超参数

- Width Multiplier(α): Thinner Models
  - 所有层的 **通道数（channel）** 乘以 α 参数(四舍五入)，模型大小近似下降到原来的 α^2 倍，计算量下降到原来的 α^2 倍
  - $\alpha \in (0, 1]$ with typical settings of 1, 0.75, 0.5 and 0.25，降低模型的**宽度**
- Resolution Multiplier(ρ): Reduced Representation
  - 输入层的 **分辨率（resolution）** 乘以 ρ 参数 (四舍五入)，等价于所有层的分辨率乘 ρ，模型大小不变，计算量下降到原来的 ρ^2 倍
  - $\rho \in (0, 1]$，降低输入图像的**分辨率**

Depthwise separable convolutions cost:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

MobileNetV1: Efficient Convolutional Neural Networks for Mobile Vision Application

两个超参数

- Width Multiplier(α): Thinner Models
  - 所有层的 **通道数（channel）** 乘以 α 参数(四舍五入)，模型大小近似下降到原来的 α^2 倍，计算量下降到原来的 α^2 倍
  - $\alpha \in (0, 1]$ with typical settings of 1, 0.75, 0.5 and 0.25，降低模型的**宽度**
- Resolution Multiplier(ρ): Reduced Representation
  - 输入层的 **分辨率（resolution）** 乘以 ρ 参数 (四舍五入)，等价于所有层的分辨率乘 ρ，模型大小不变，计算量下降到原来的 ρ^2 倍
  - $\rho \in (0, 1]$，降低输入图像的**分辨率**

Depthwise separable convolutions cost:

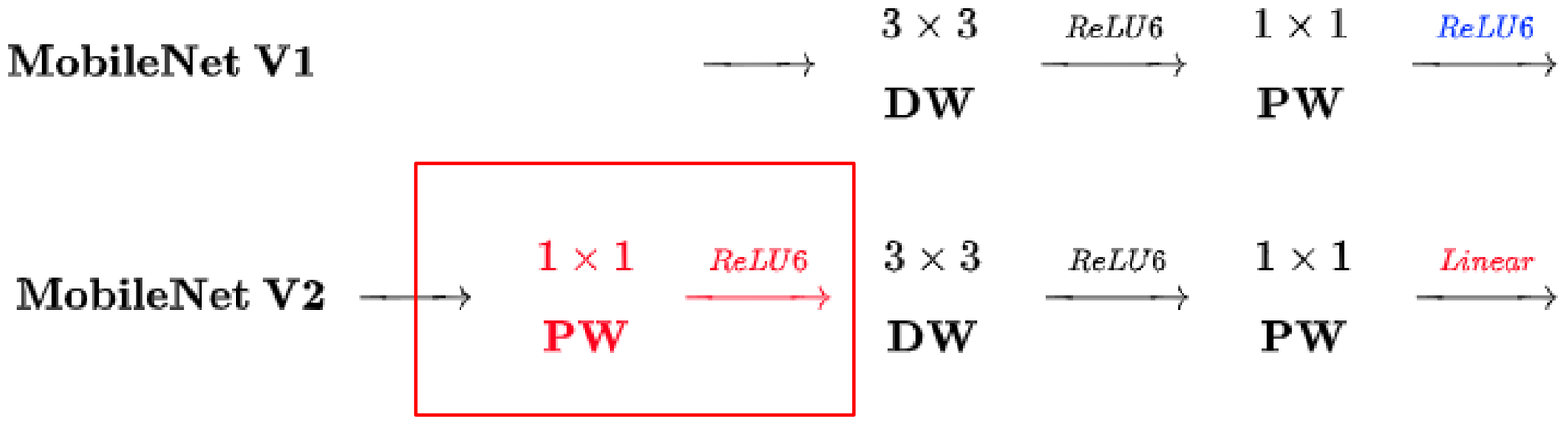
$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

MobileNetV2： Inverted Residuals and Linear Bottlenecks

MobileNet v2 是在v1的Depthwise Separable的基础上引入了残差结构。并发现了ReLU的在通道数较少的Feature Map上有非常严重信息损失问题，由此引入了Linear Bottlenecks和Inverted Residual。

1. Inverted residuals：

V2 在 DW 卷积之前新加了一个 PW 卷积：目的是为了提升通道数，获得更多特征。



MobileNetV2： Inverted Residuals and Linear Bottlenecks

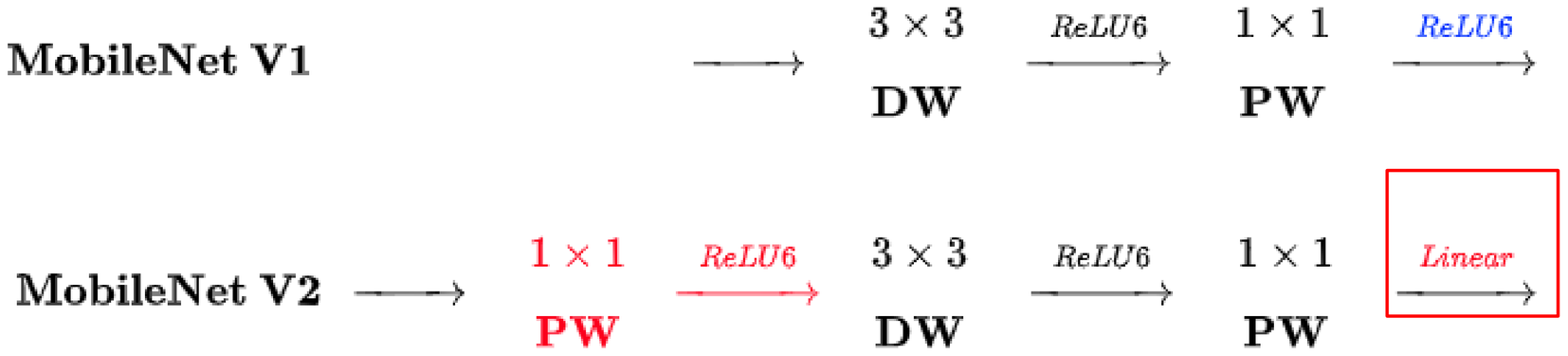
MobileNet v2 是在v1的Depthwise Separable的基础上引入了残差结构。并发现了ReLU的在通道数较少的Feature Map上有非常严重信息损失问题，由此引入了Linear Bottlenecks和Inverted Residual。

1. Inverted residuals：

V2 在 DW 卷积之前新加了一个 PW 卷积：目的是为了提升通道数，获得更多特征。

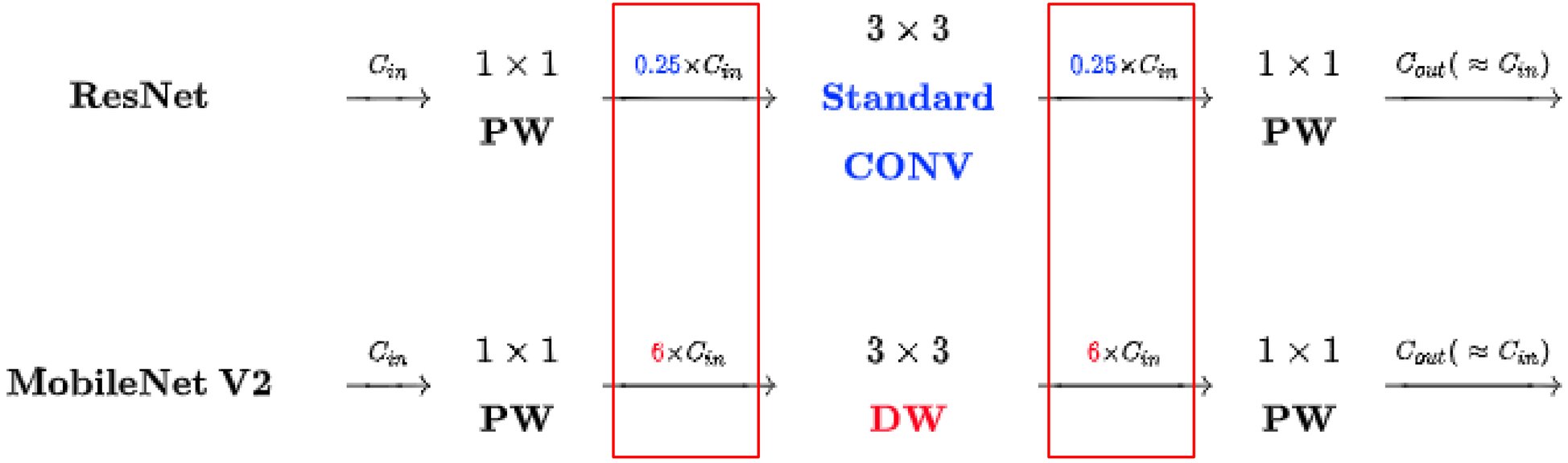
2. Linear Bottleneck：

V2 去掉了第二个 PW 的激活函数：激活函数在高维空间能够有效的增加非线性，而在低维空间时则会破坏特征。由于第二个 PW 的主要功能就是降维，因此降维之后使用线性的效果好。



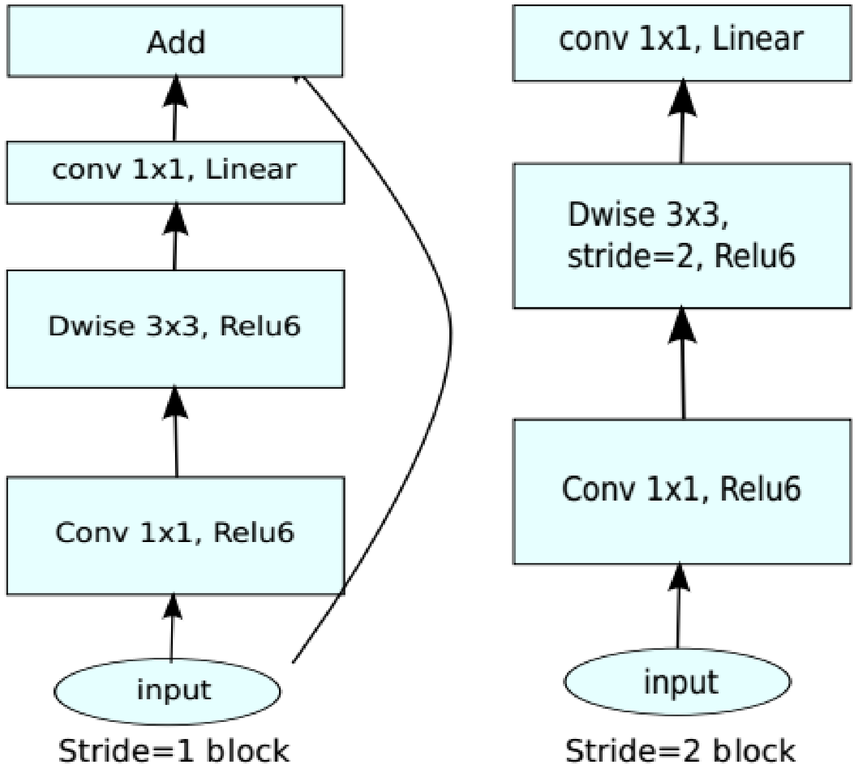
MobileNetV2： Inverted Residuals and Linear Bottlenecks

MobileNetV2的block与ResNet block



MobileNetV2： Inverted Residuals and Linear Bottlenecks

MobileNetV2的block与MobileNetV1 block



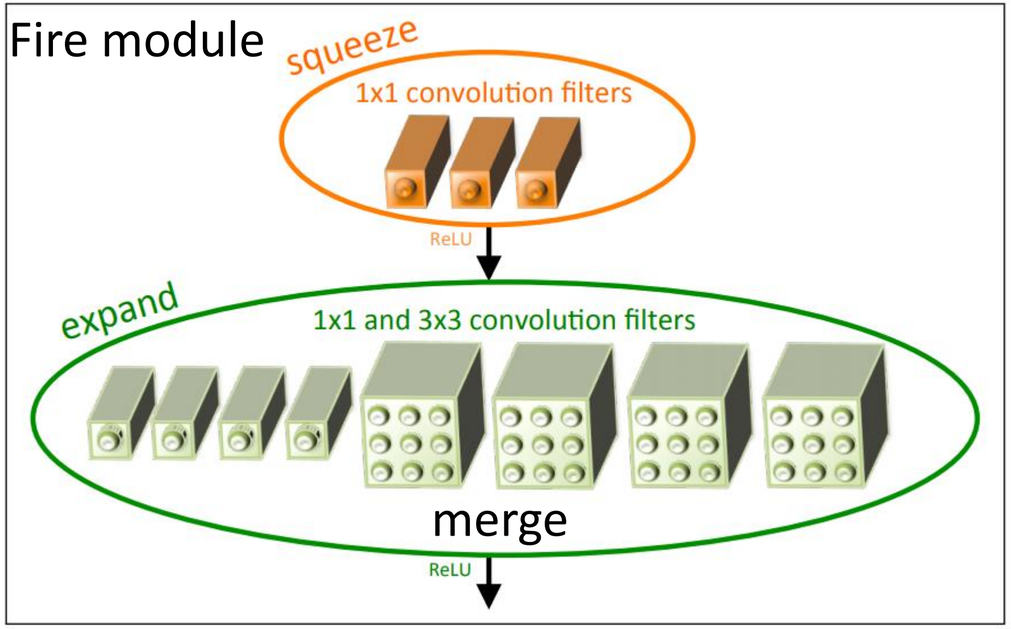
Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$28^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck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

Network	Top 1	Params
MobileNetV1	70.6	4.2M
ShuffleNet (1.5)	71.5	**3.4M**
ShuffleNet (x2)	73.7	5.4M
NasNet-A	74.0	5.3M
MobileNetV2	**72.0**	**3.4M**
MobileNetV2 (1.4)	**74.7**	6.9M

SqueezeNet

Fire module

使用以下三个策略来减少SqueezeNet设计参数
（1）使用1*1卷积代替3*3 卷积：参数减少为原来的1/9
（2）减少输入通道数量：这一部分使用squeeze layers来实现
（3）将降采样操作延后，可以给卷积层提供更大的激活图：更大的激活图保留了更多的信息，可以提供更高的分类准确率



squeeze convolution layer： 只使用1*1 卷积减少通道数。
Filter expand layer： 使用1*1 和3*3 卷积 filter的组合；

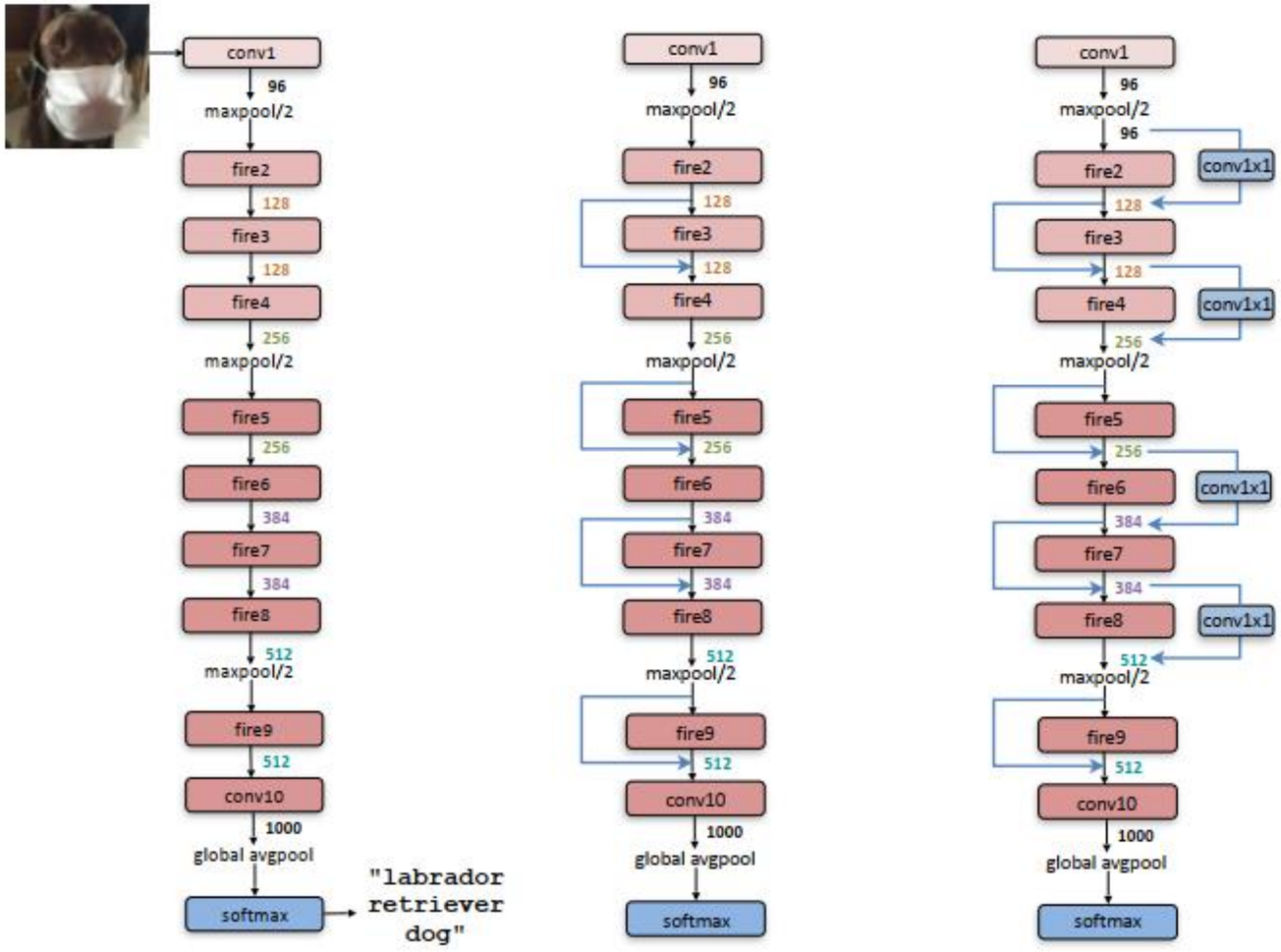


Figure 2: Macroarchitectural view of our SqueezeNet architecture. Left: SqueezeNet (Section 3.3); Middle: SqueezeNet with simple bypass (Section 6); Right: SqueezeNet with complex bypass (Section 6).

Thanks !