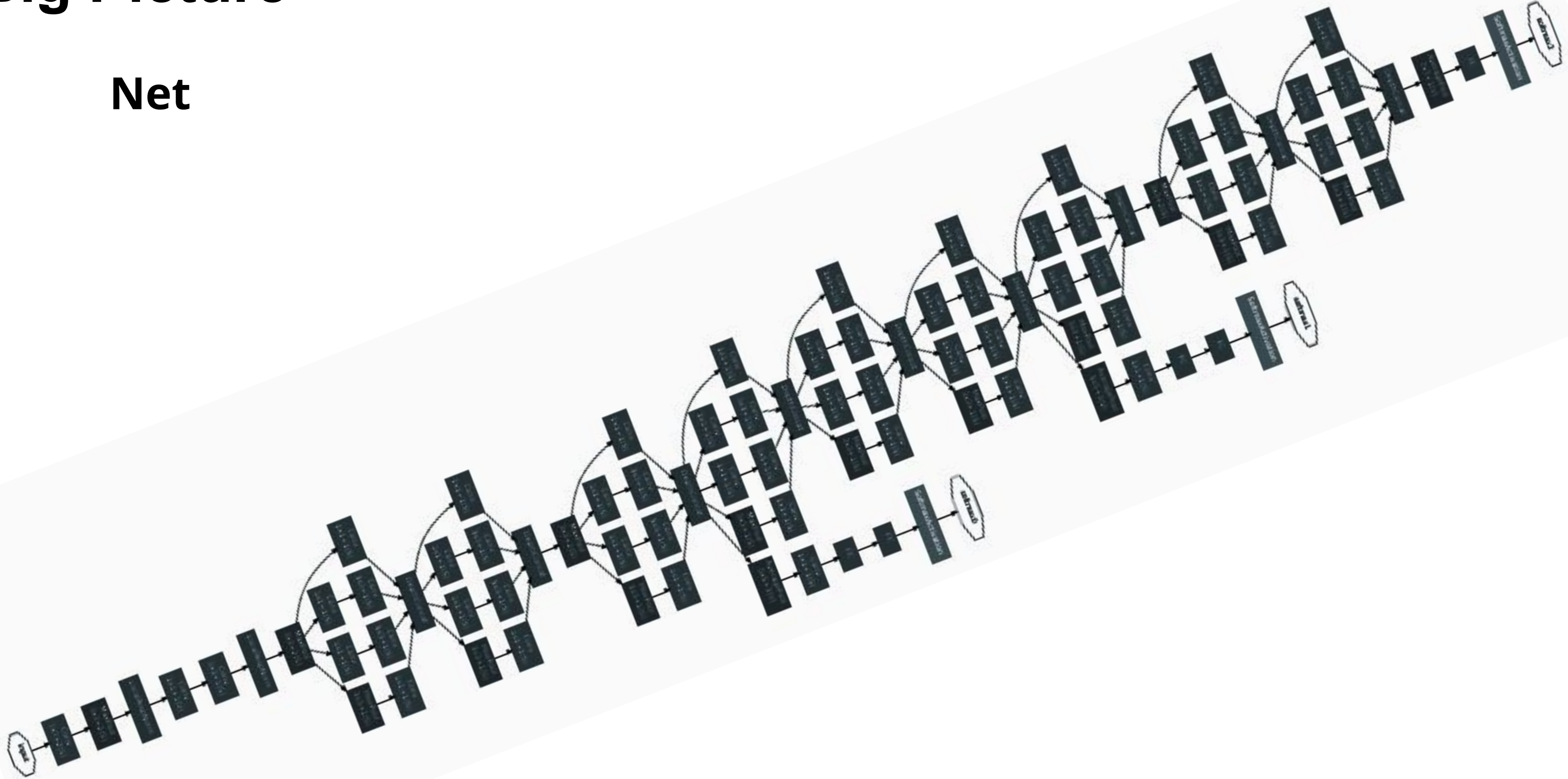
Contents

Big Picture

Net

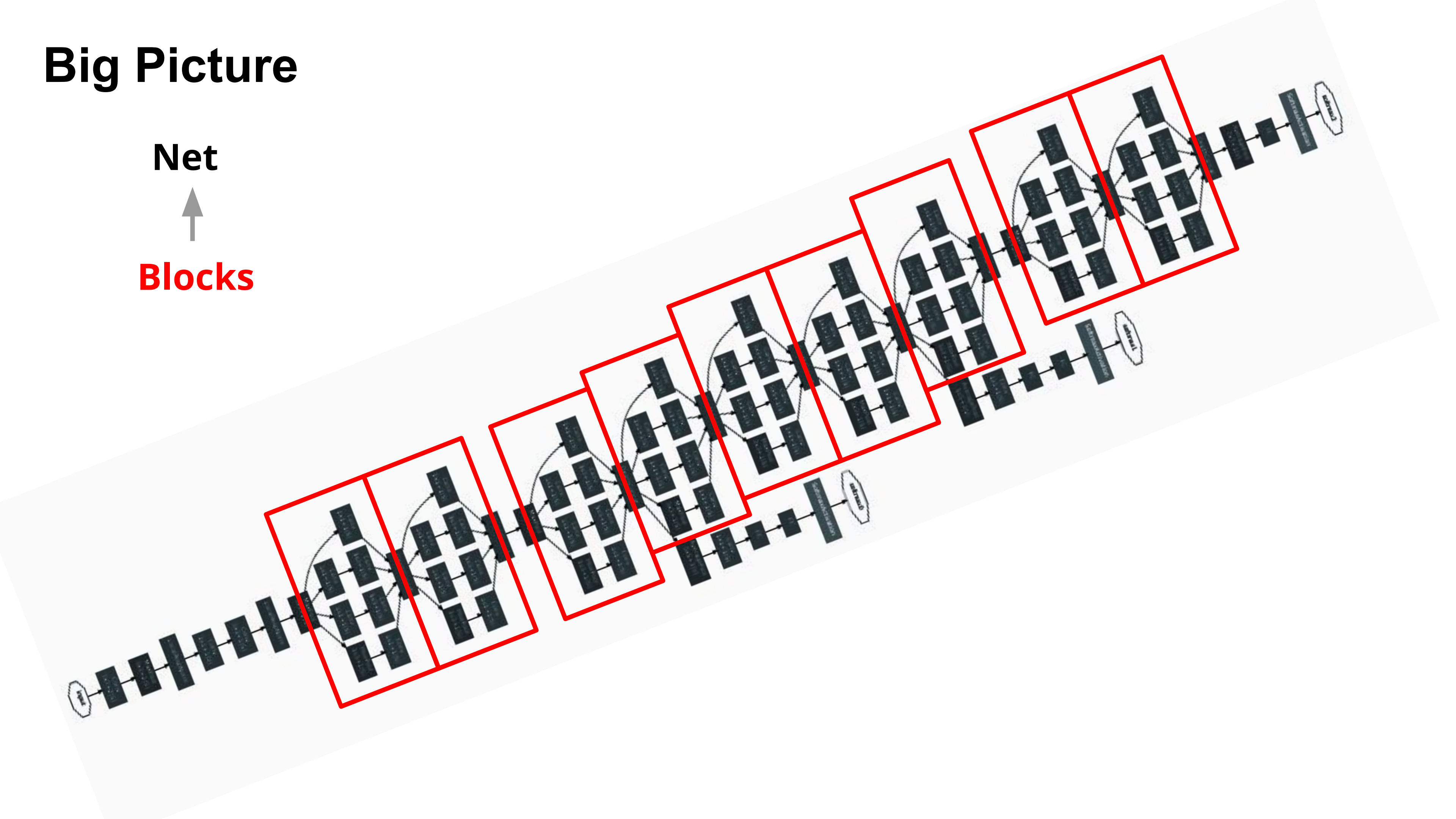


Big Picture

Net

↑

Blocks



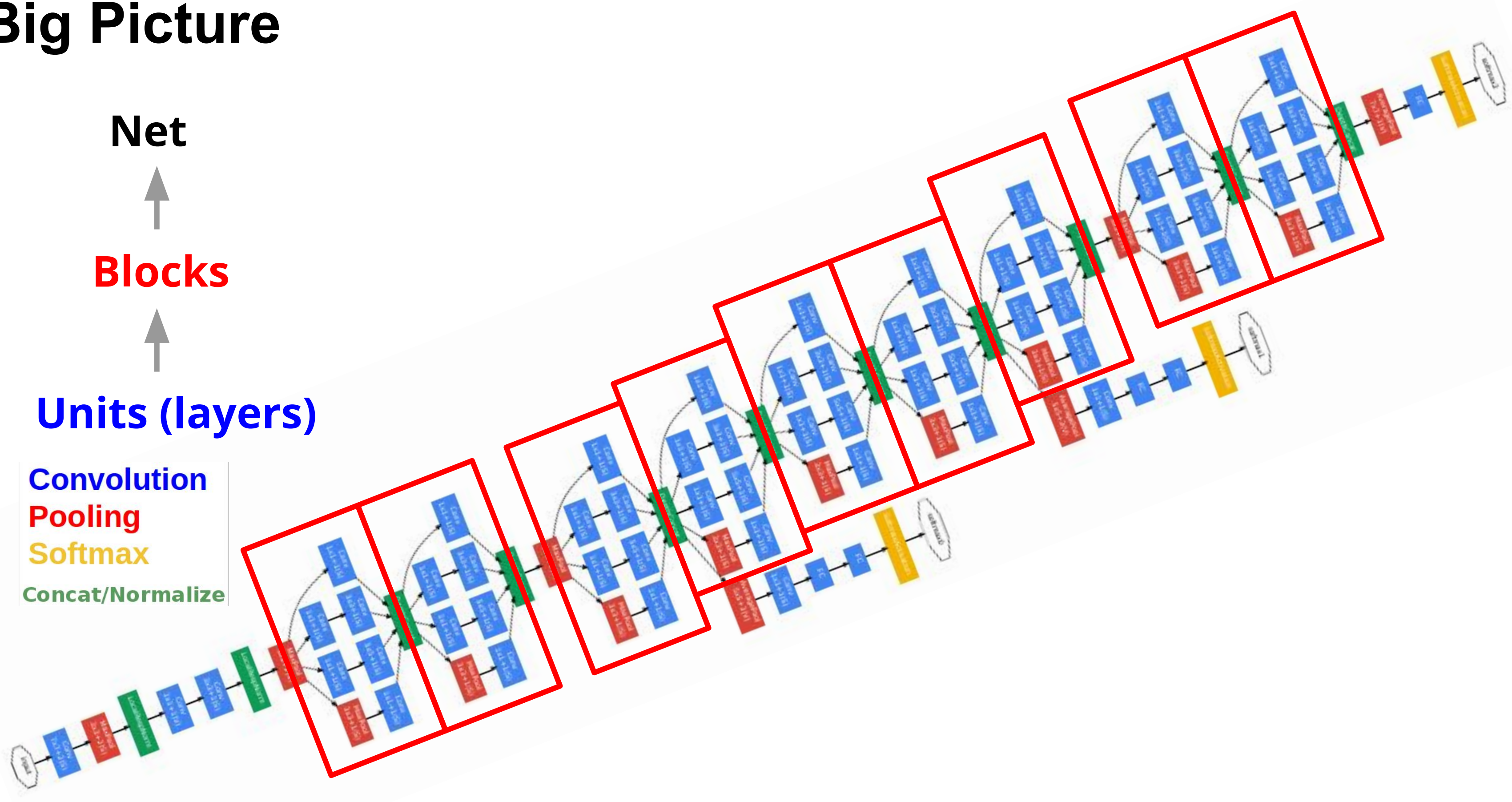
Big Picture

Net

Blocks

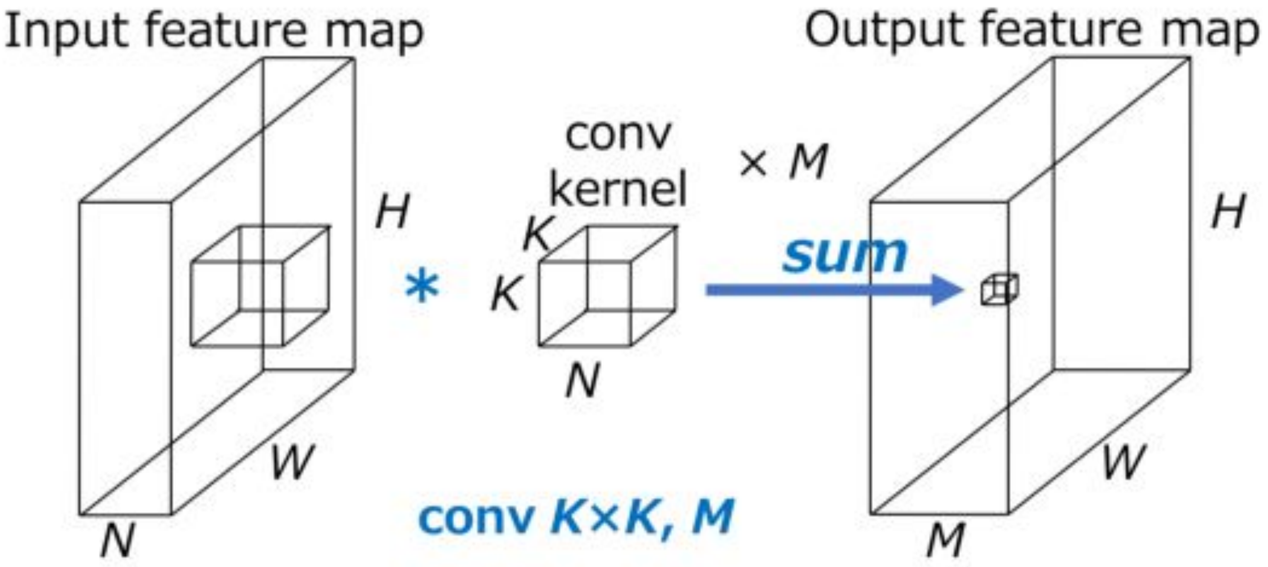
Units (layers)

Convolution
Pooling
Softmax
Concat/Normalize



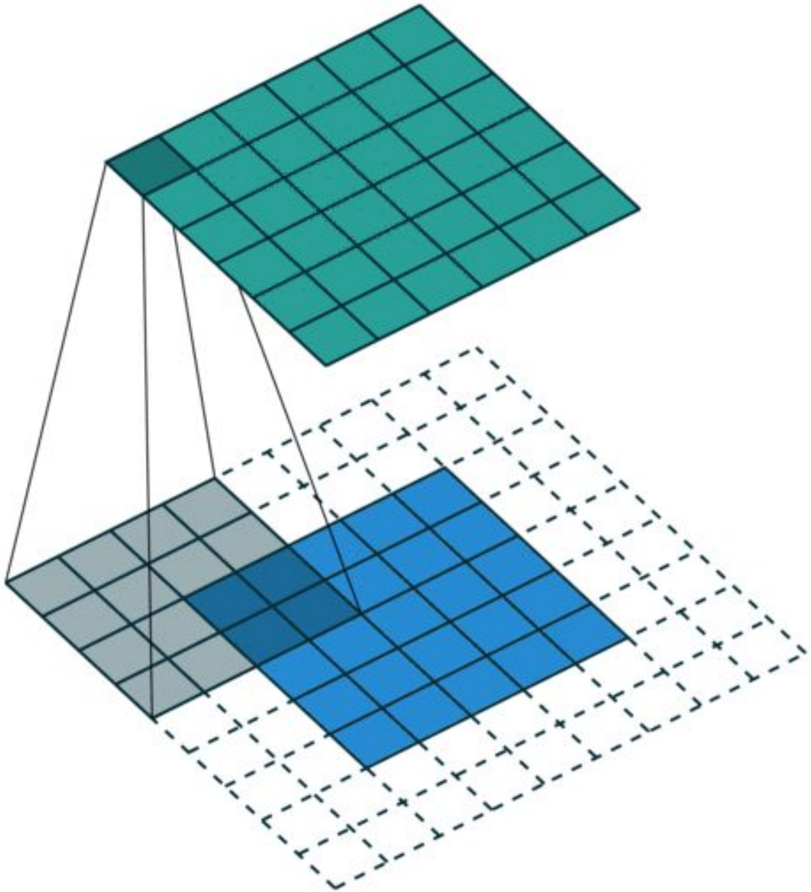
Contents

Layers [Convolution]



https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

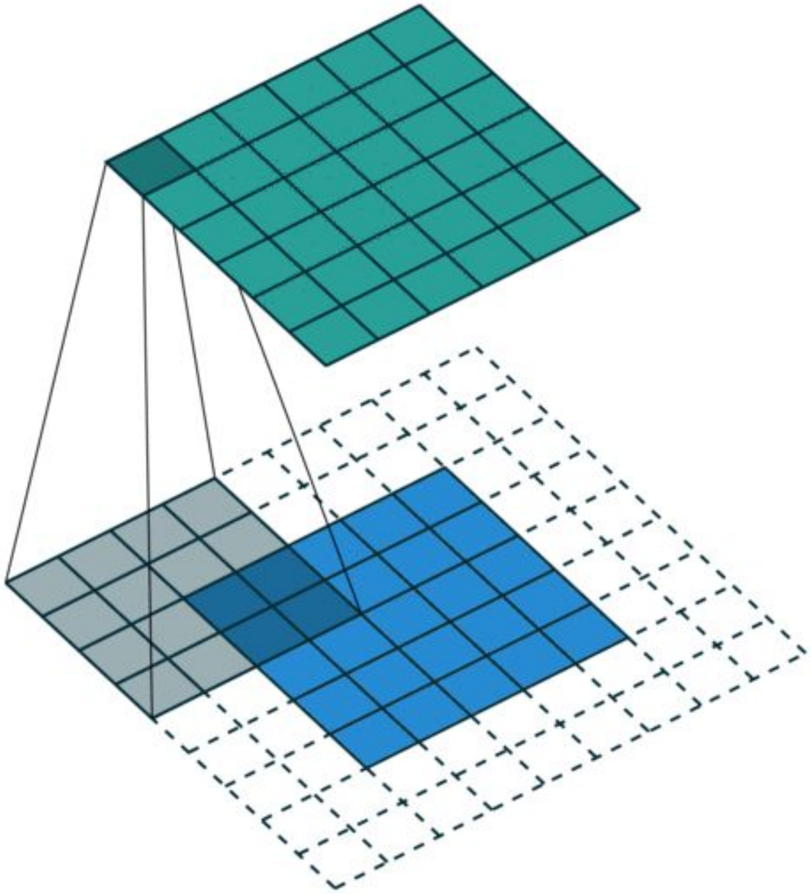
Layers [Convolution]



- kernel size F ?
- padding size P ?
- stride S ?

https://arxiv.org/abs/1603.07285
http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

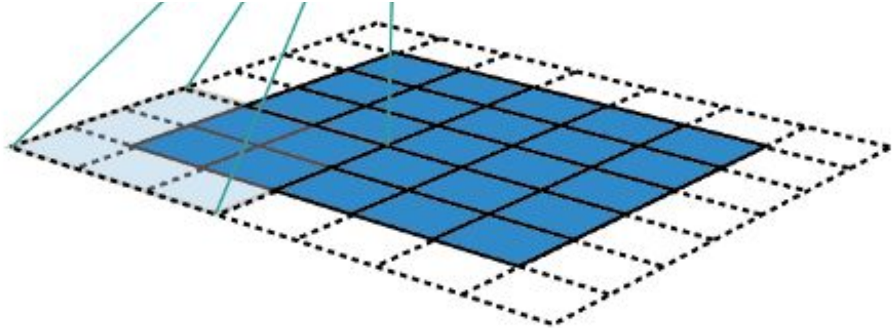
Layers [Convolution]



- kernel size F = 4x4
- padding size P = 2
- stride S = 1

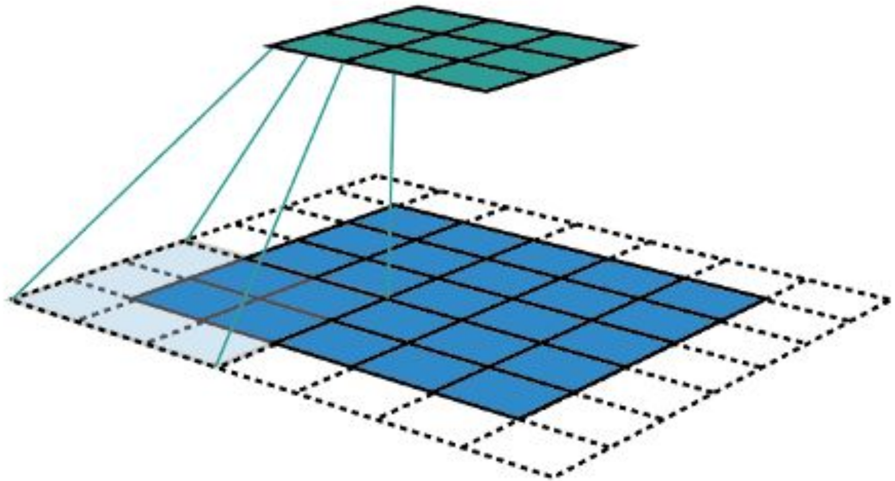
https://arxiv.org/abs/1603.07285
http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

Layers [Convolution]



- kernel size, F = 3x3
- padding size, P = 1
- stride, S = 2

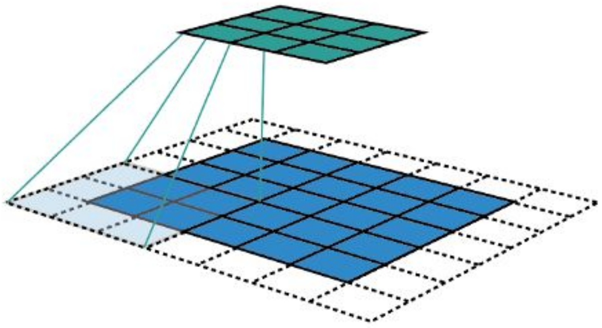
Layers [Convolution]



- kernel size, F = 3x3
- padding size, P = 1
- stride, S = 2

Contents

Layers [Convolution] [Receptive field]



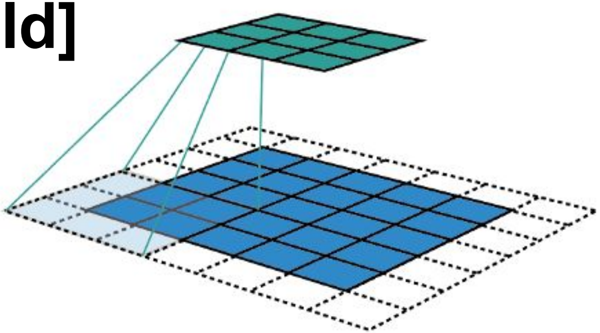
- kernel size, F = 3x3
- padding size, P = 1
- stride, S = 2

The **receptive field** is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by)

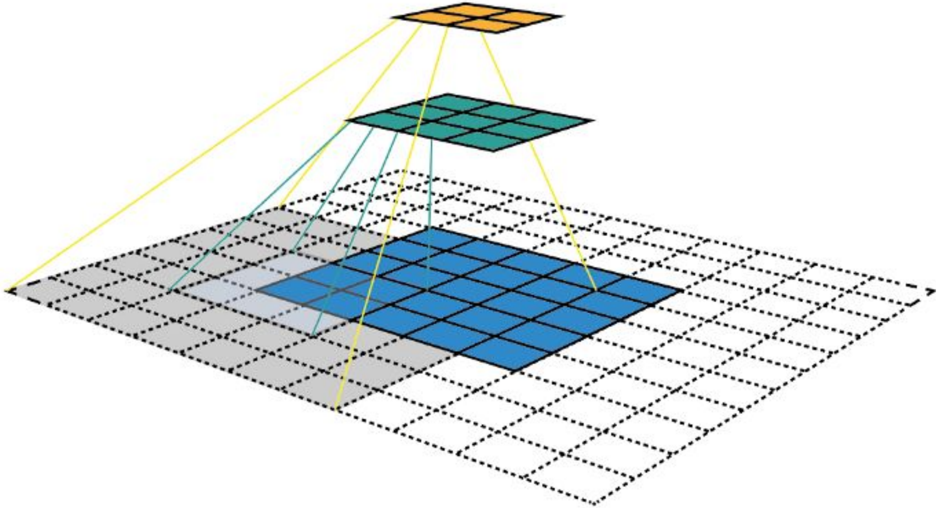
A receptive field of a feature can be described by

- its center location
- its size

Layers [Convolution] [Receptive field]

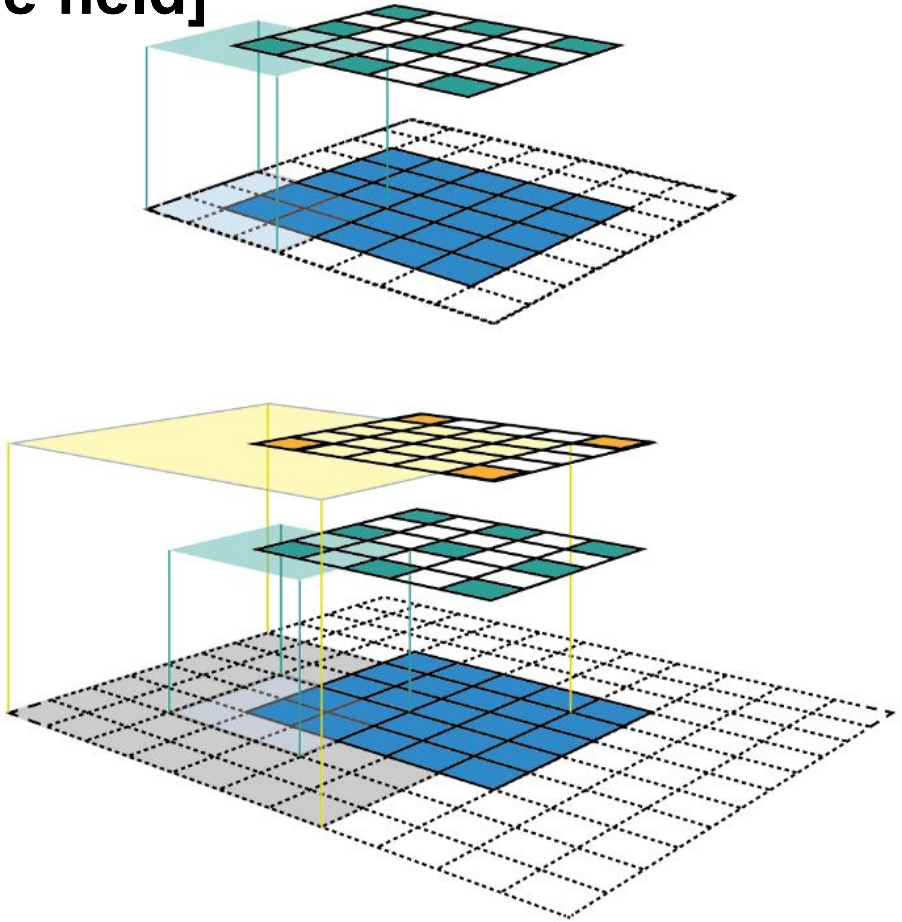


- kernel size F = 3x3,
- padding size P = 1,
- stride S = 2



https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807

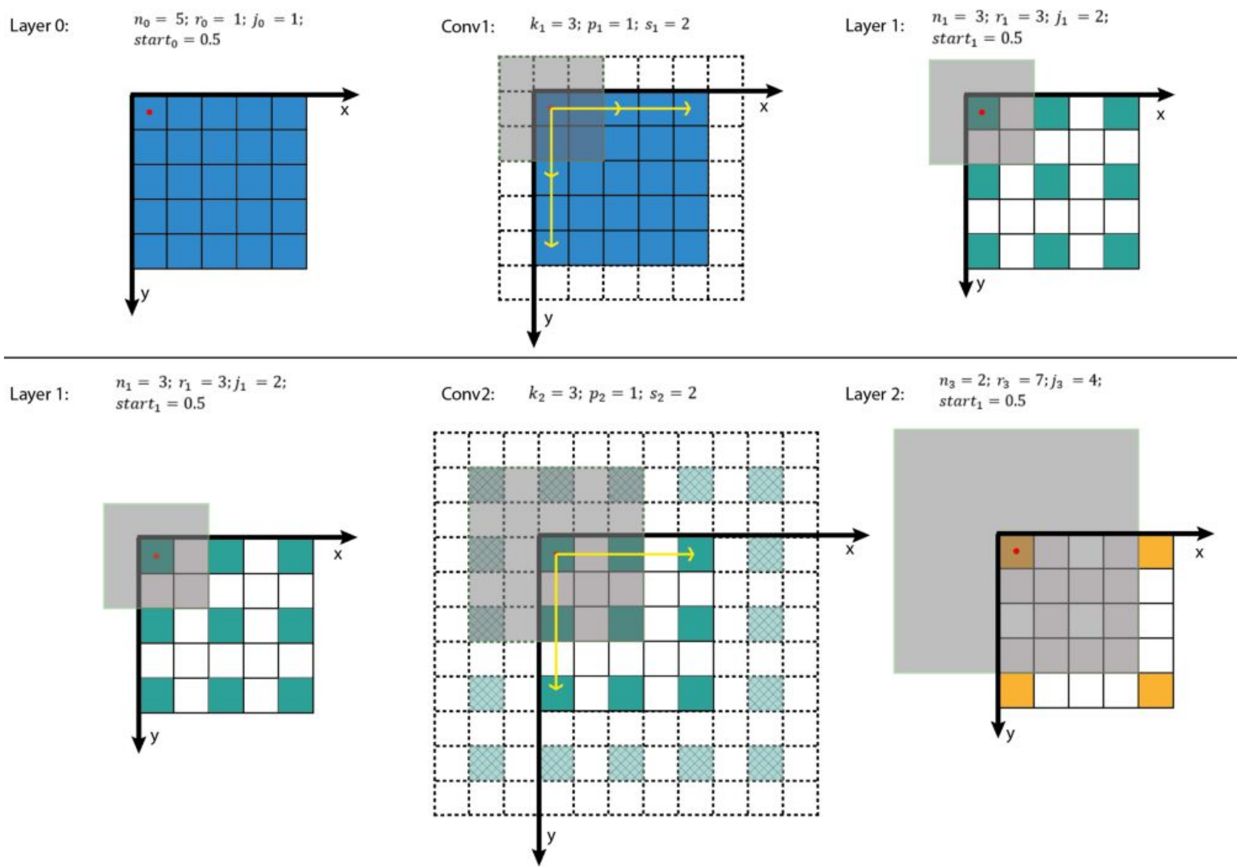
Layers [Convolution] [Receptive field]



- kernel size F = 3x3,
- padding size P = 1,
- stride S = 2

https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807

Layers [Convolution] [Receptive field]



$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

$$j_{out} = j_{in} * s$$

$$r_{out} = r_{in} + (k - 1) * j_{in}$$

$$start_{out} = start_{in} + \left(\frac{k-1}{2} - p\right) * j_{in}$$

https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807

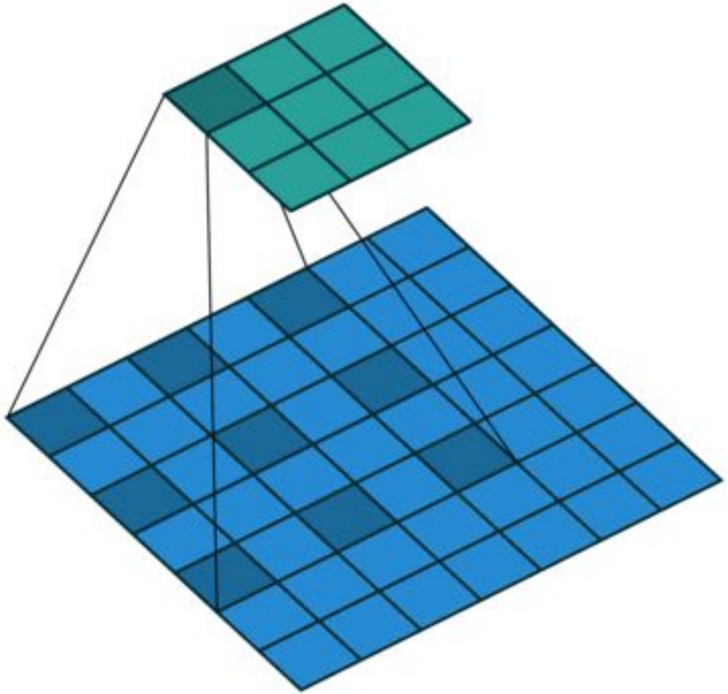
Contents

Layers [Dilated Convolution]

(used for *Semantic Segmentation*, *Object Recognition*)

Also known as “atrous convolutions”.

Dilated convolutions “inflate” the kernel by inserting spaces between the kernel elements. The dilation “rate” is controlled by an additional hyperparameter **d**.



- input size I = 7x7
- kernel size F = 3x3,
- padding size P = 0,
- stride S = 1
- dilation rate **d** = 2

https://arxiv.org/abs/1511.07122
http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

Layers [Dilated Convolution]

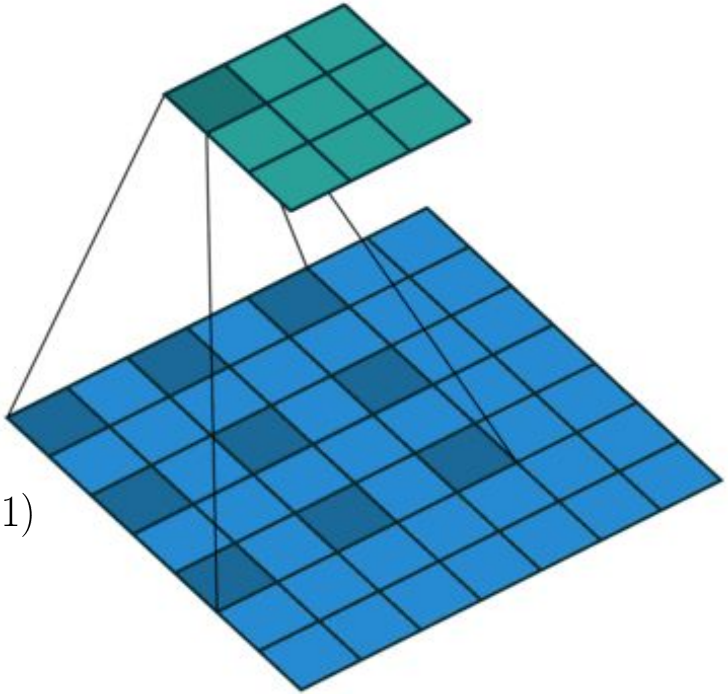
(used for *Semantic Segmentation*, *Object Recognition*)

- input size I = 7x7
- kernel size F = 3x3,
- padding size P = 0,
- stride S = 1
- dilation rate **d** = 2

For ‘normal’ convolution we have:

$$W2 = (W1 - F + 2P)/S + 1,$$
$$H2 = (H1 - F + 2P)/S + 1$$
$$D2 = K$$

$$F' = (F - 1) \cdot (d - 1)$$



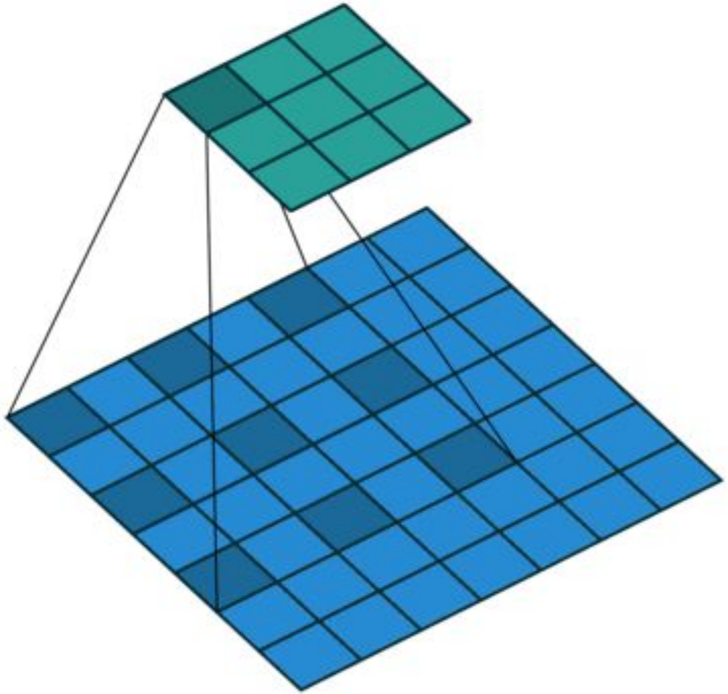
How this will change ?

https://arxiv.org/abs/1511.07122
http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

Layers [Dilated Convolution]

(used for *Semantic Segmentation*, *Object Recognition*)

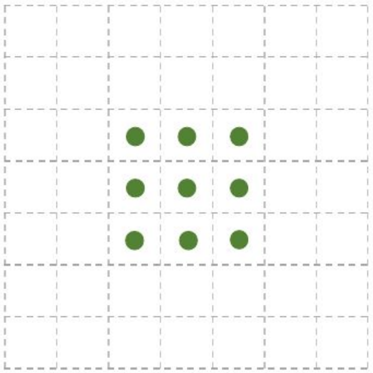
- Detection of fine-details by processing inputs in higher resolutions.
- Broader view of the input to capture more contextual information.
- Faster run-time with less parameters



https://arxiv.org/abs/1511.07122
http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

Layers [Deformable Convolution]

(used for *Semantic Segmentation*, *Object Recognition*)



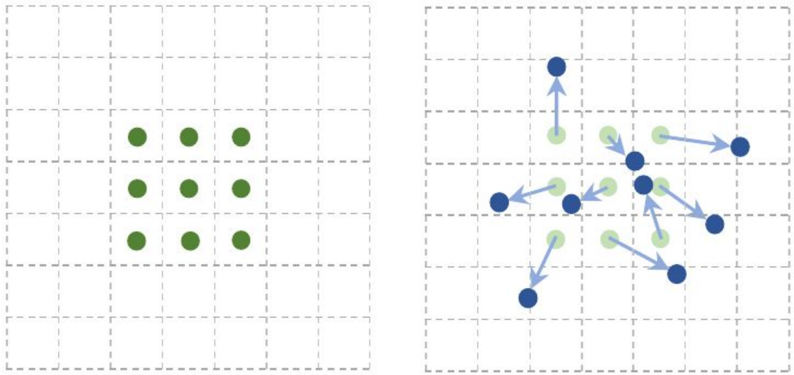
$$\mathcal{R} = \{(-1,-1),(-1,0),\ldots,(0,1),(1,1)\}$$

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

https://arxiv.org/abs/1703.06211

Layers [Deformable Convolution]

(used for *Semantic Segmentation*, *Object Recognition*)



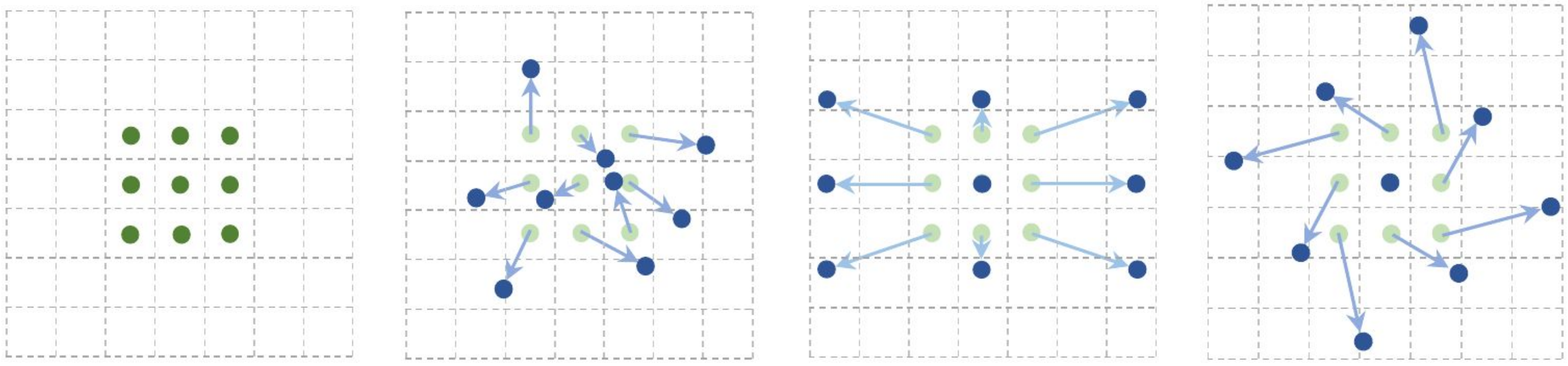
$$\mathcal{R} = \{(-1,-1), (-1,0), \ldots, (0,1), (1,1)\}$$

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

https://arxiv.org/abs/1703.06211

Layers [Deformable Convolution]

(used for *Semantic Segmentation*, *Object Recognition*)



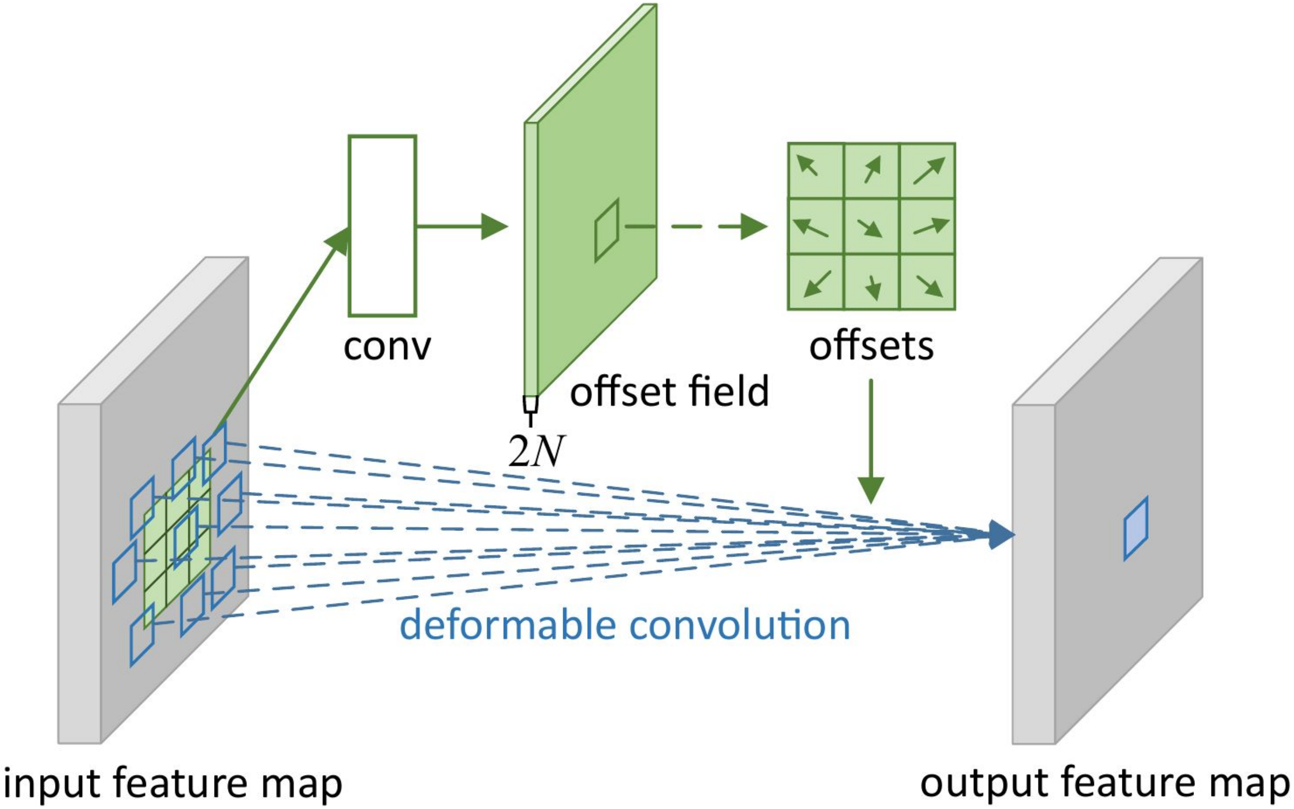
$$\mathcal{R} = \{(-1,-1), (-1,0), \ldots, (0,1), (1,1)\}$$

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

https://arxiv.org/abs/1703.06211

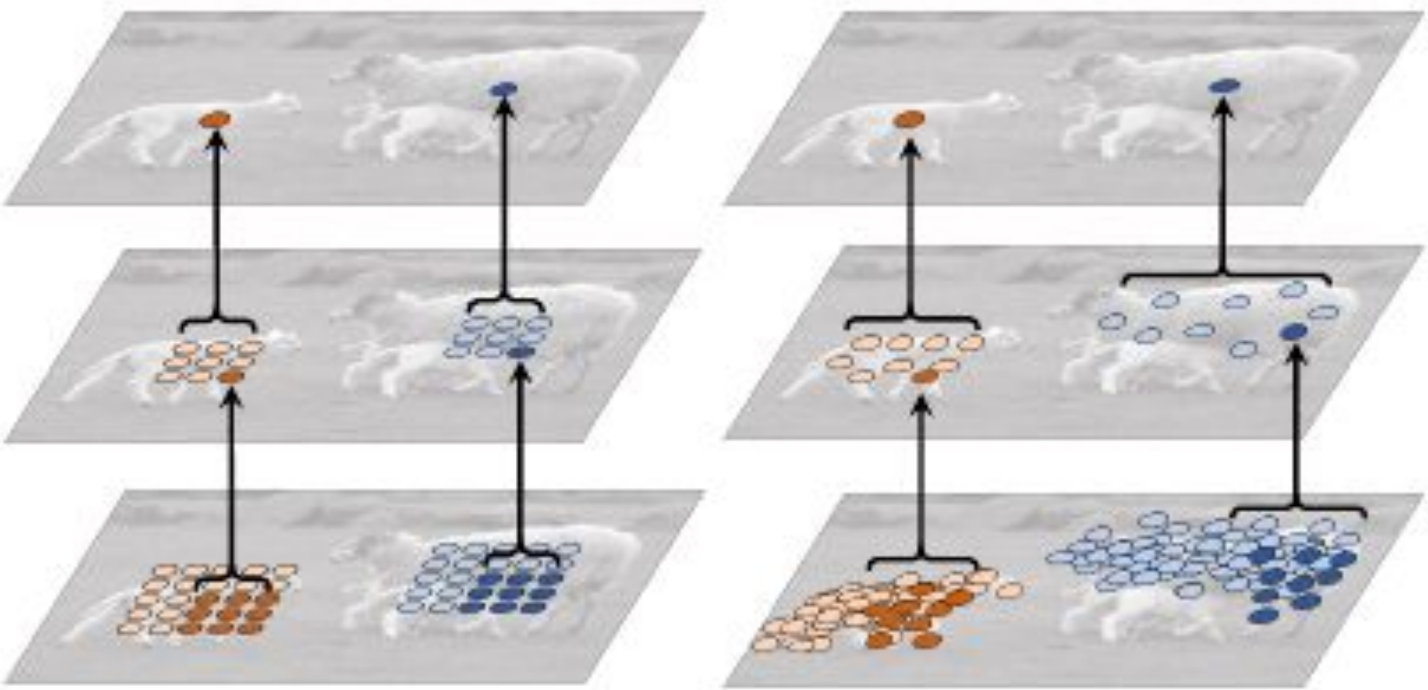
Layers [Deformable Convolution]

(used for *Semantic Segmentation*, *Object Recognition*)



https://arxiv.org/abs/1703.06211

Layers [Deformable Convolution]



(a) standard convolution (b) deformable convolution

https://arxiv.org/abs/1703.06211

Layers [Deformable Convolution]

https://arxiv.org/abs/1703.06211

Contents

Layers [Upsampling]

- **Nearest Neighbor**

1	2
3	4

Input: [2 x 2]

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output: [2 x 2]

- **Bilinear**

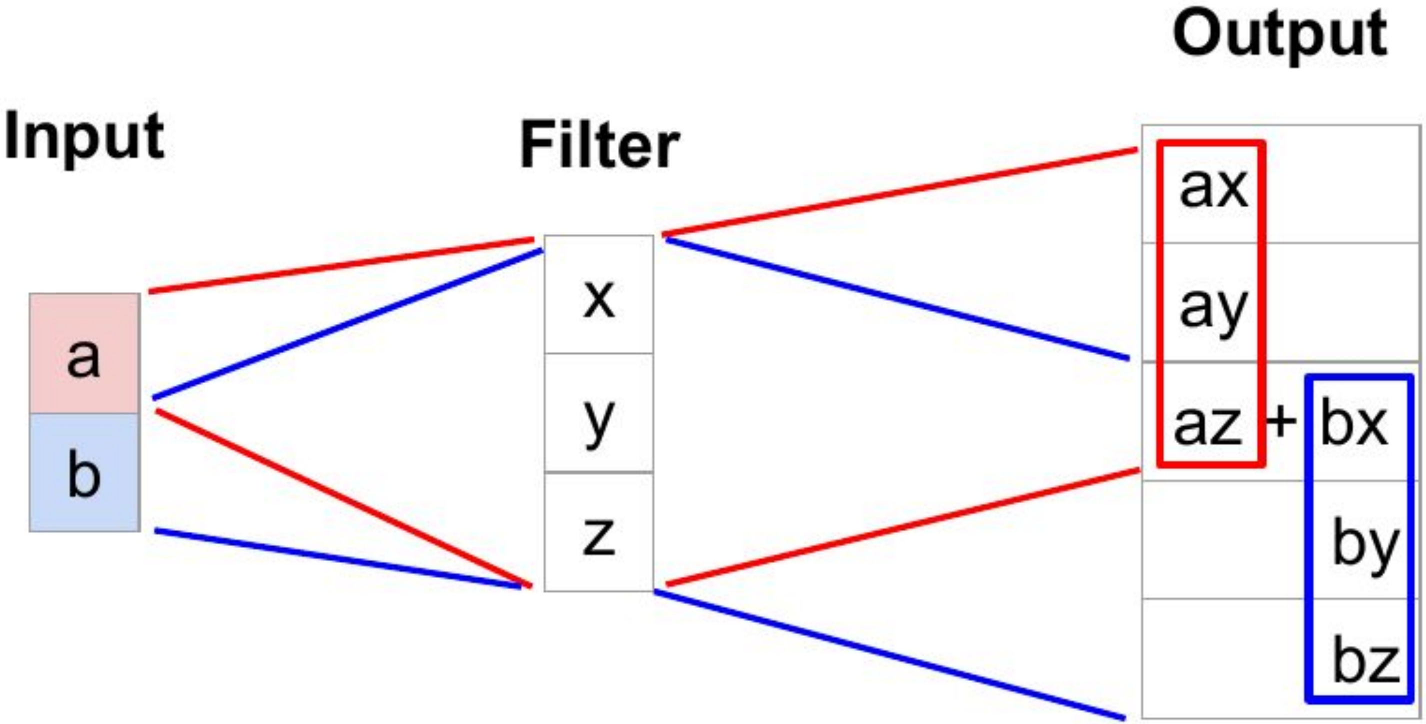
1	2
3	4

Input: [2 x 2]

1.00	1.25	1.75	2.00
1.50	1.75	2.00	2.50
2.50	2.75	3.25	3.50
3.00	3.25	3.75	4.00

Output: [2 x 2]

Layers [Learnable Upsampling: Transpose Convolution]



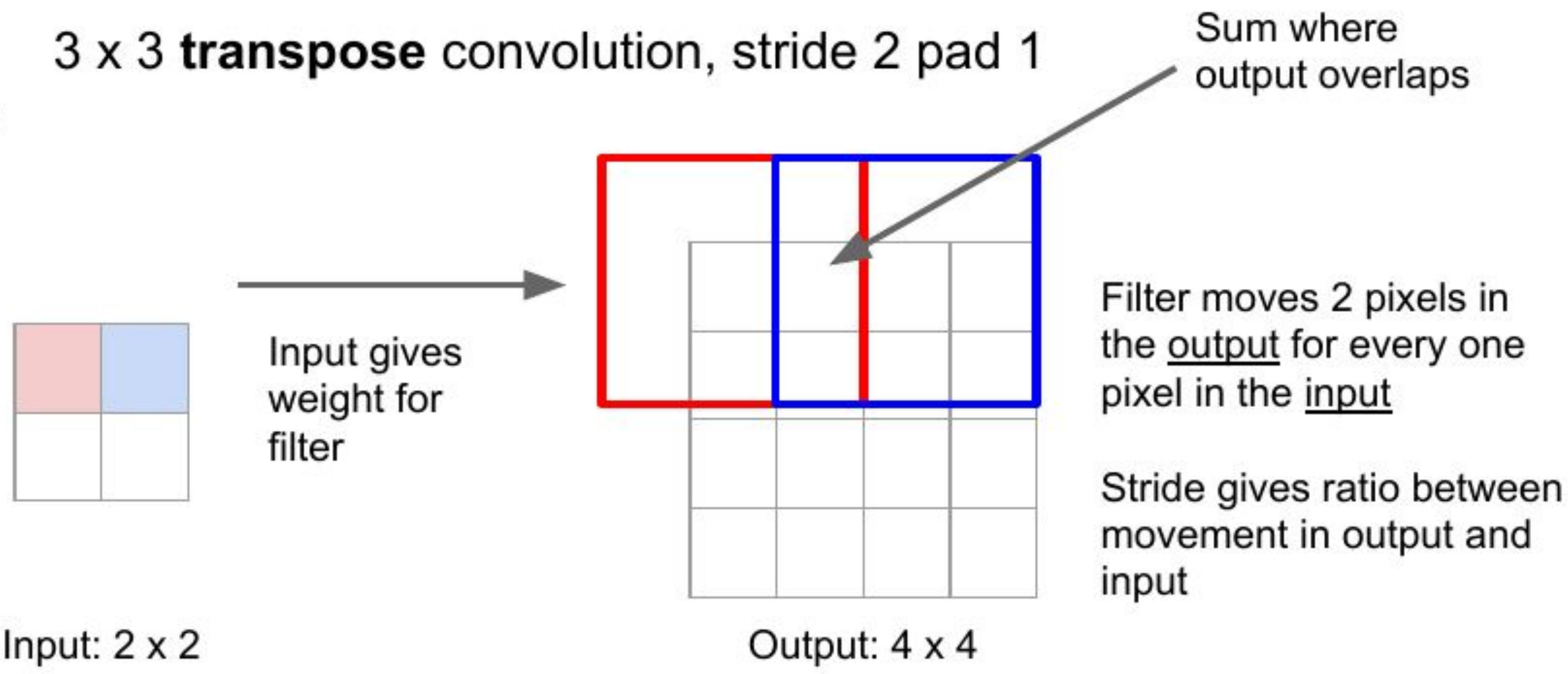
Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

http://cs231n.stanford.edu

Layers [Learnable Upsampling: Transpose Convolution]

[Heavily used for *Semantic Segmentation*, *GANs, Autoencoders*]

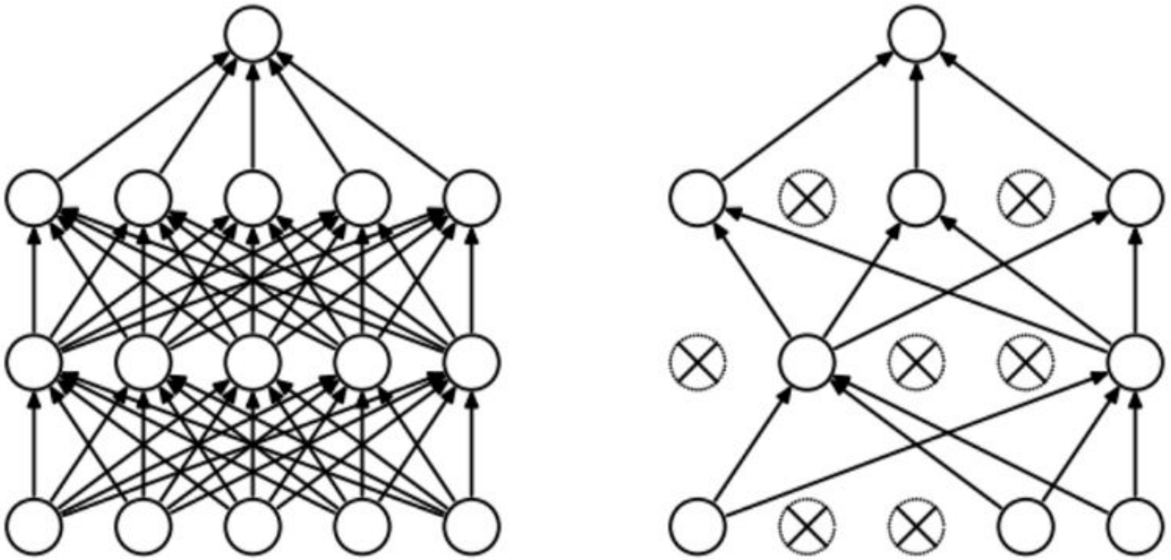


http://cs231n.stanford.edu

Contents

Layers [Dropout]

- In-network ensembling
- Reduce overfitting



(a) Standard Neural Net

(b) After applying dropout.

Layers [Batch Normalization]

BN: data-driven normalizing each layer, for each batch

- Greatly accelerate training
- Less sensitive to initialization
- Improve regularization

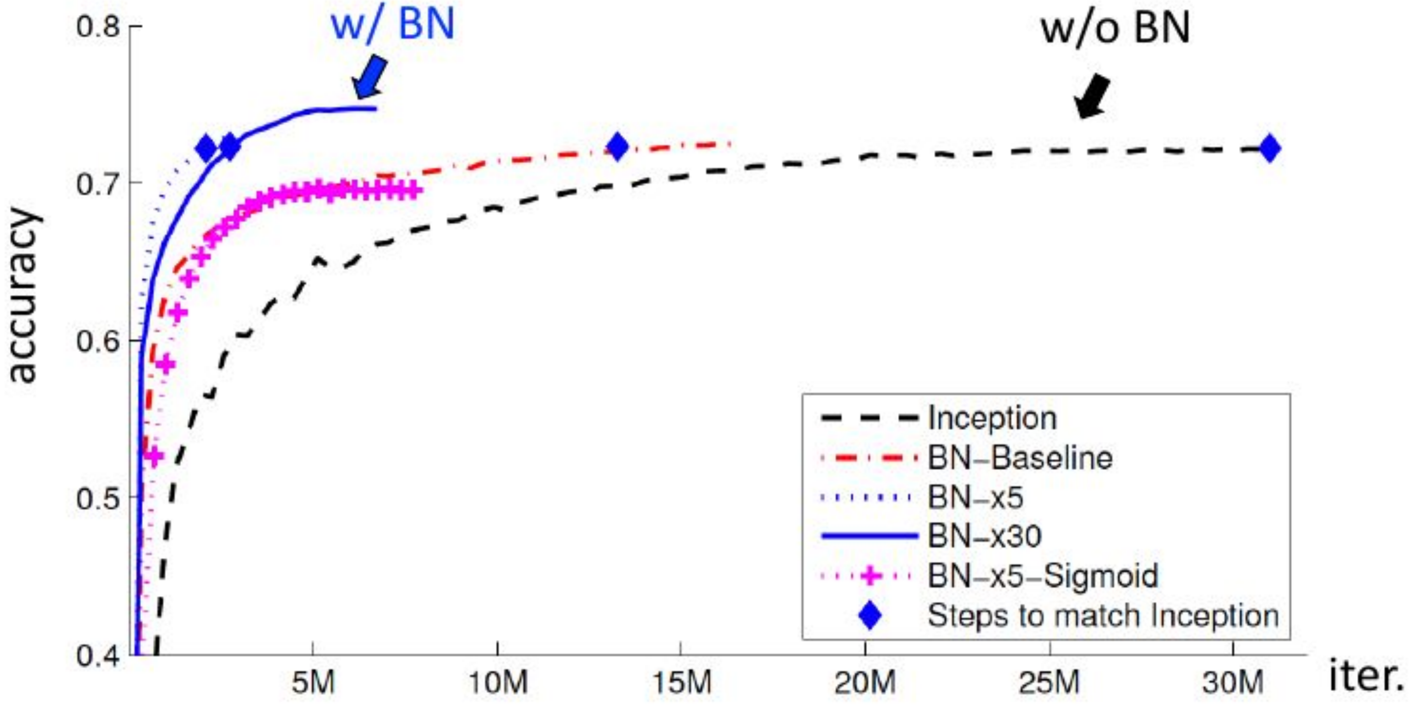
$$\rightarrow \boxed{\text{layer}} \rightarrow x \rightarrow \quad \hat{x} = \frac{x - \mu}{\sigma} \quad \rightarrow \quad y = \gamma \hat{x} + \beta$$

- μ: mean of x in mini-batch
- σ: std of x in mini-batch
- γ: scale
- β: shift

- μ, σ: functions of x, analogous to responses
- γ, β: parameters to be learned, analogous to weights

arXiv:1502.03167

Layers [Batch Normalization]



arXiv:1502.03167

Contents

Units

- Layers [Convolution]
- Layers [Convolution] [Receptive field]
- Layers [Dilated Convolution, Deformable Convolution]
- Layers [Upsampling, Learnable Upsampling]
- Layers [Batch Norm, Dropout]
- Layers [Group Convolutions and its variants]

Blocks

- VGG
- Inception
- ResNet*
- MobileNet*

Architectures

- VGG, Inception, ResNet*, MobileNet*

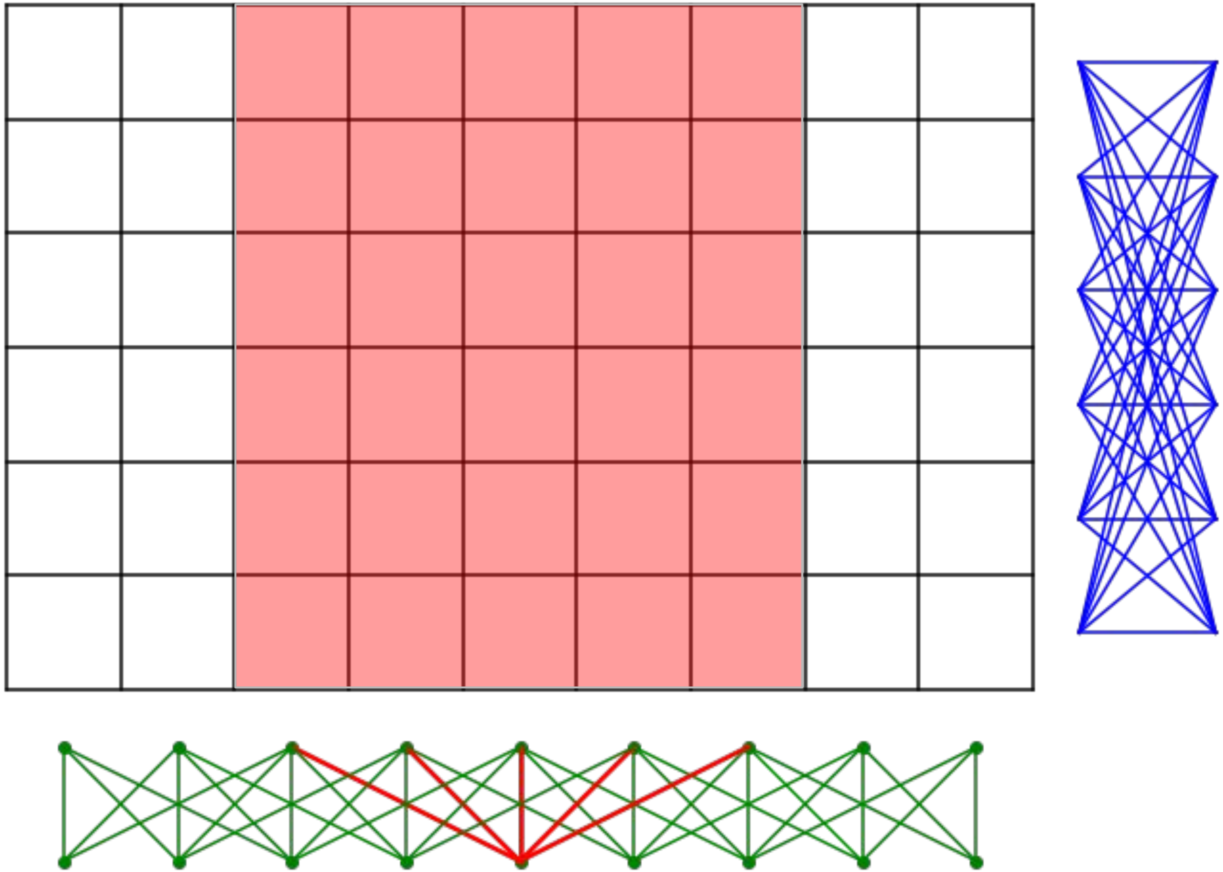
Neural Architecture Search (NAS)

Summary

Layers [Group Convolution]

Normal Convolution:

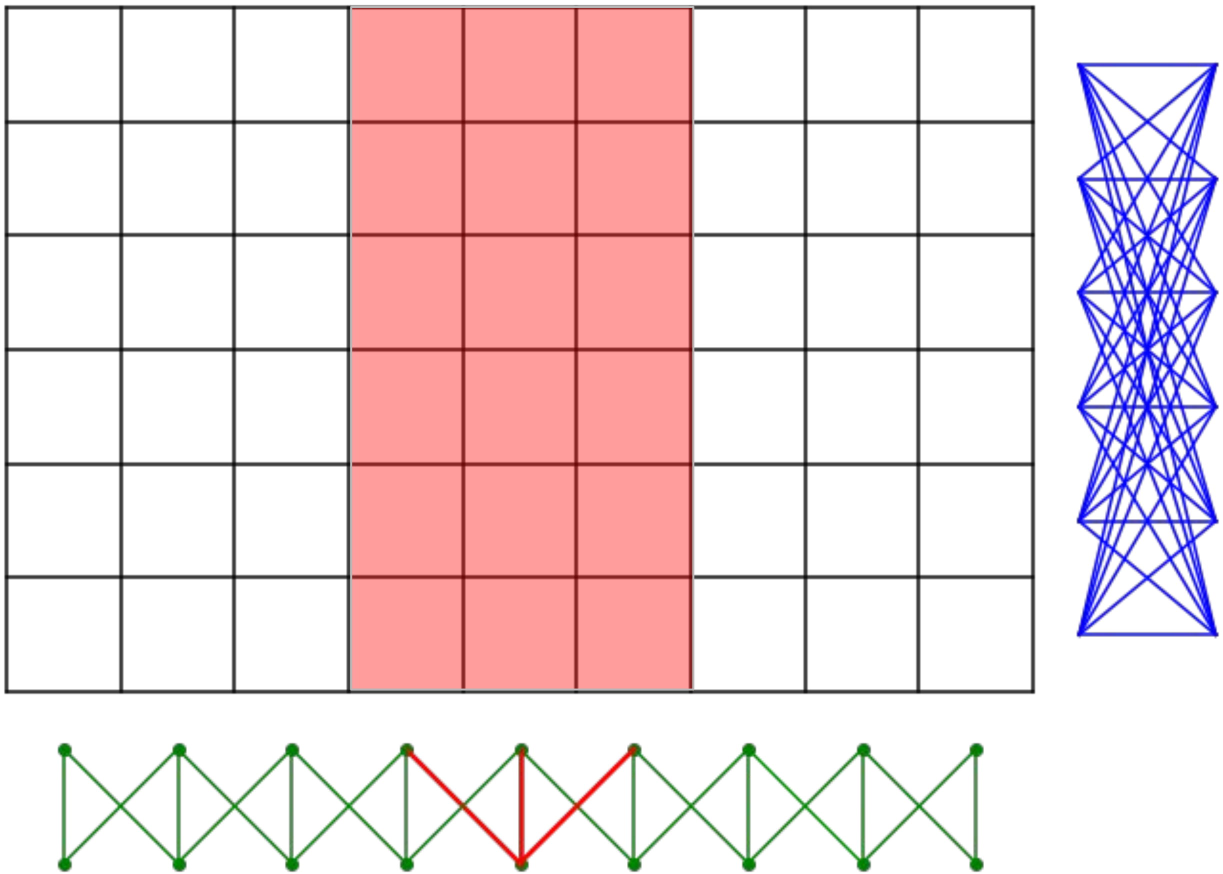
conv 5x5



Layers [Group Convolution]

Normal Convolution:

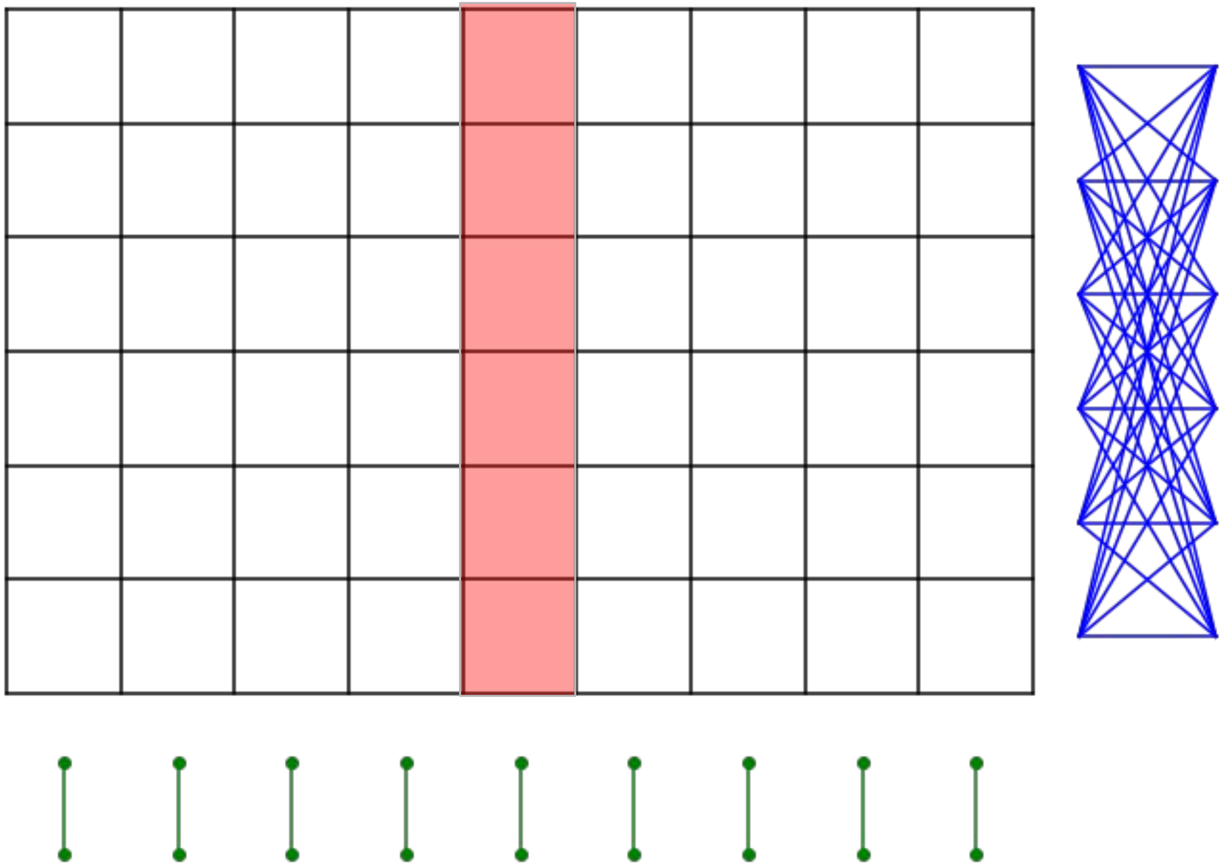
conv 3x3



Layers [Group Convolution]

Normal Convolution:

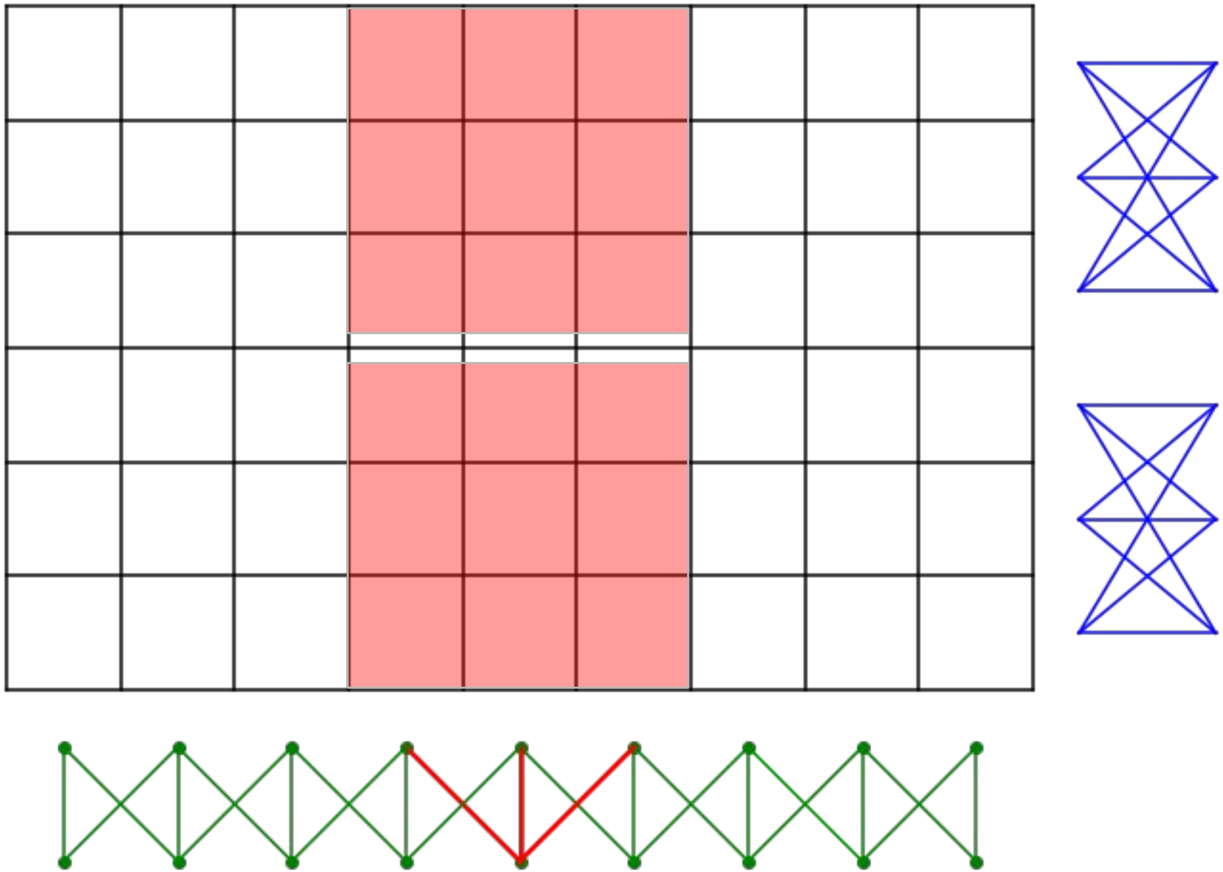
conv 1x1



A special name:
point-wise convolutions

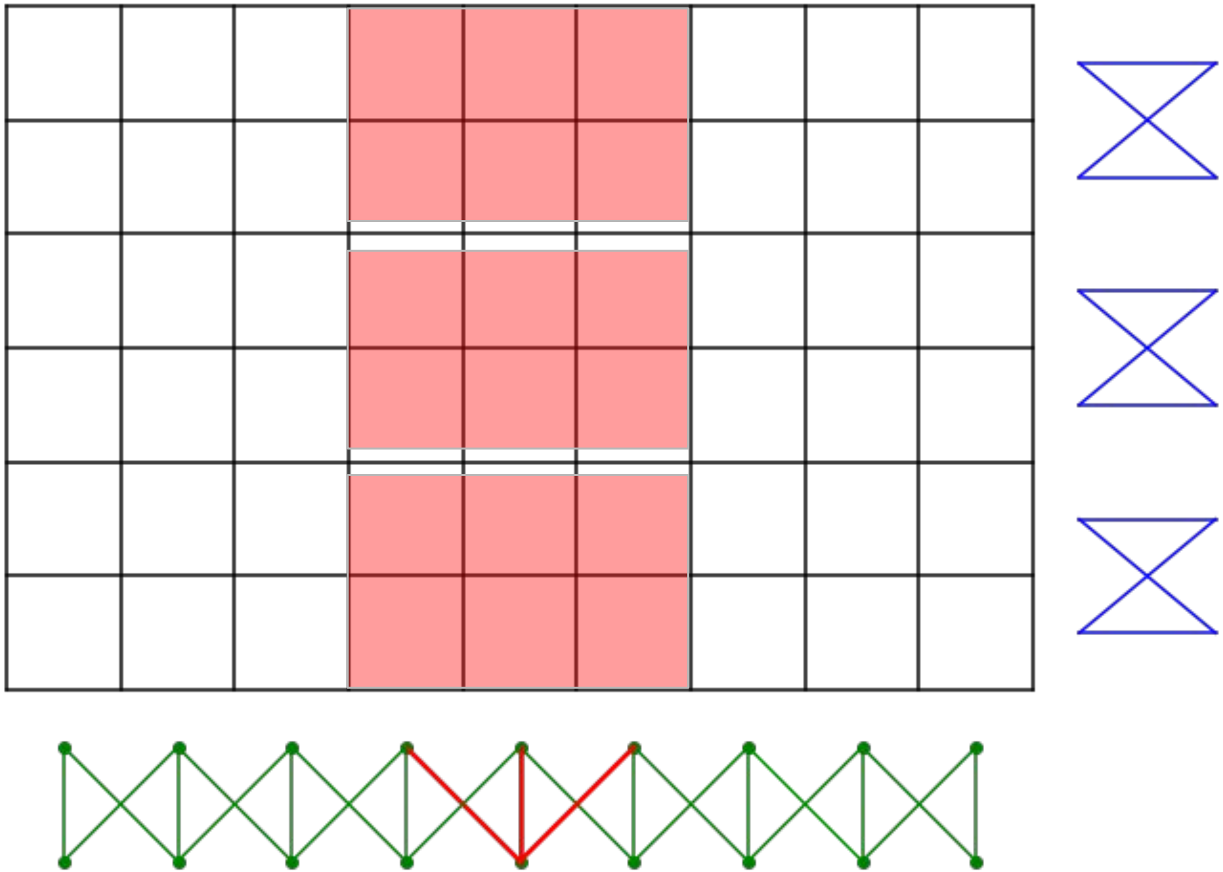
Layers [Group Convolution]

gconv 3x3 (g=2)



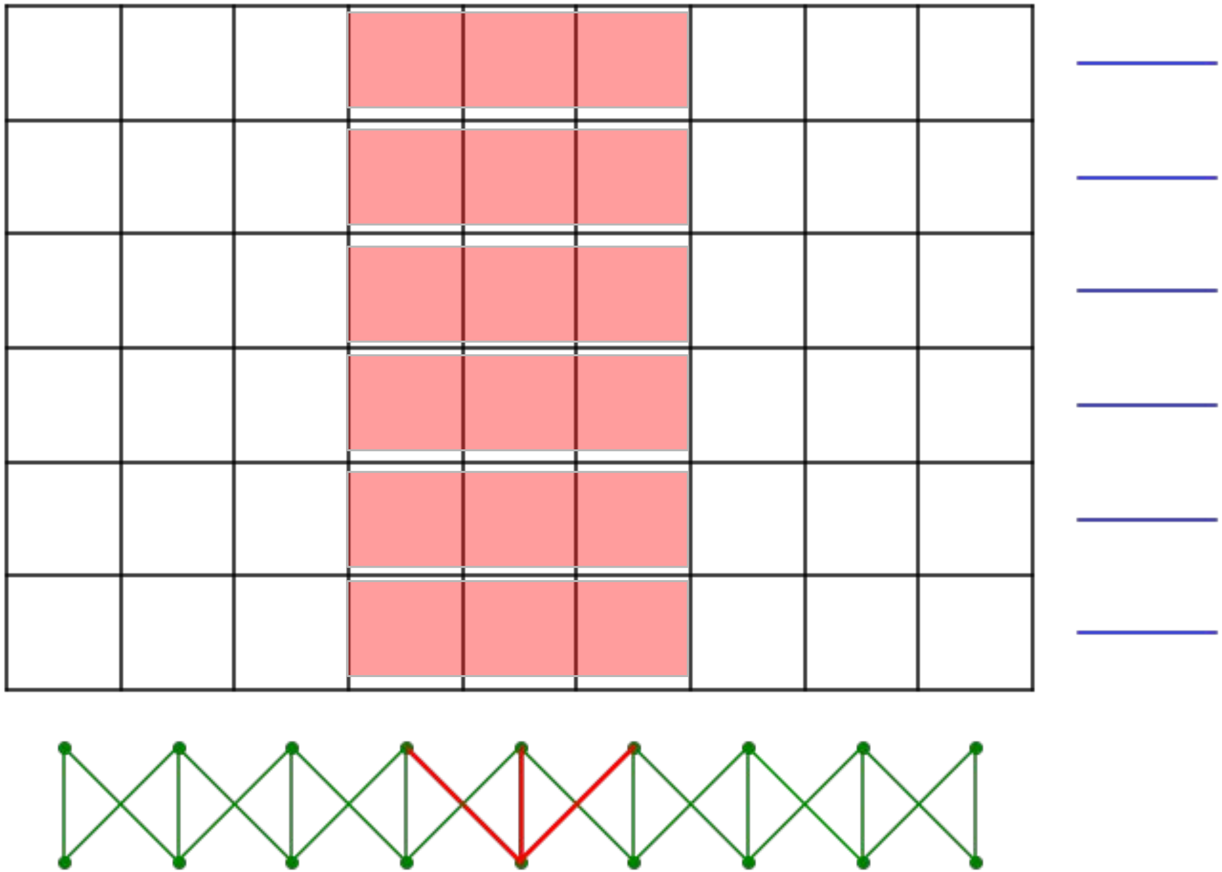
Layers [Group Convolution]

gconv 3x3 (g=3)



Layers [Group Convolution]

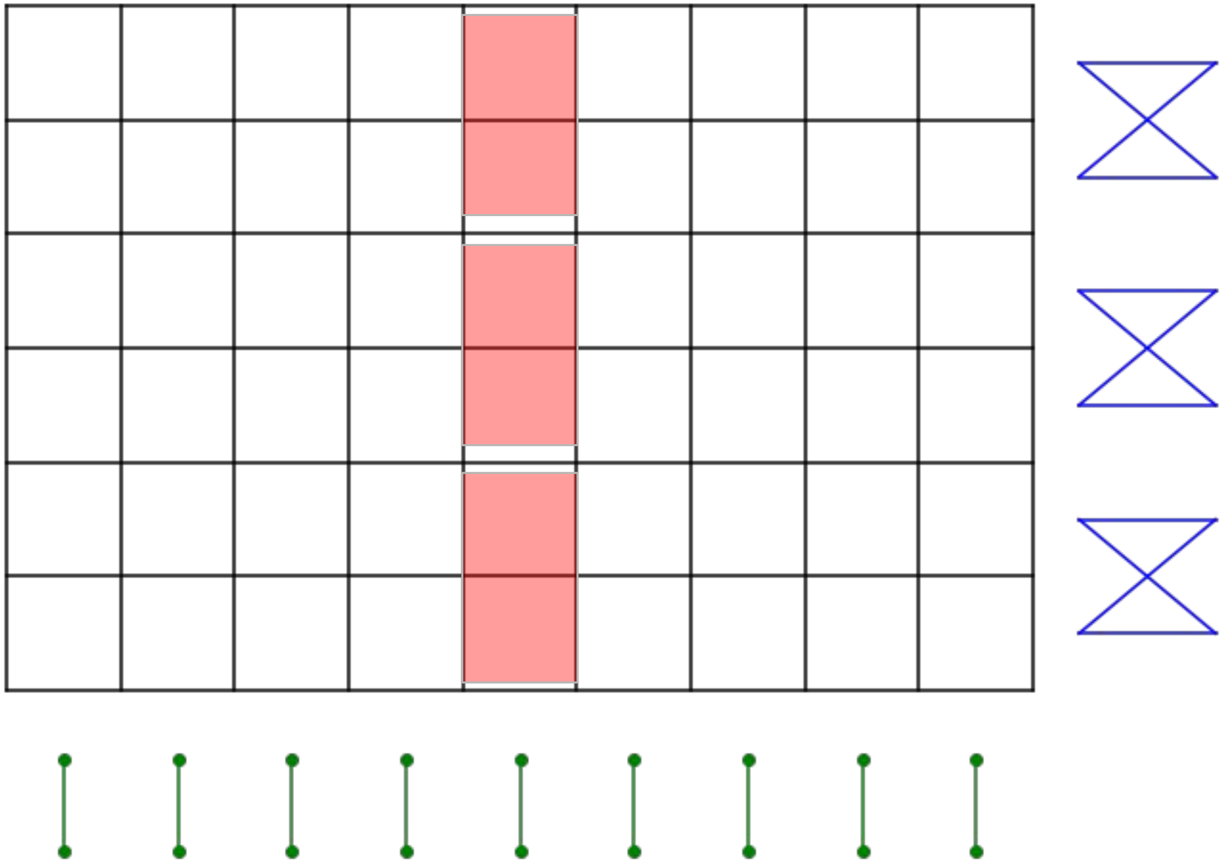
gconv 3x3 (g=3)



special name - **depth-wise convolutions** or **channel-wise convolutions**

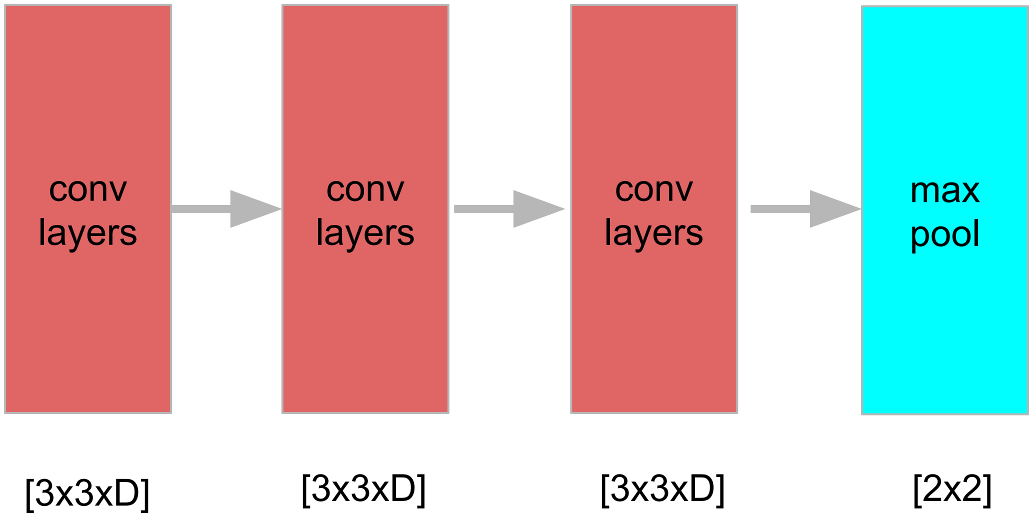
Layers [Group Convolution]

gconv 1x1 (g=3)



Contents

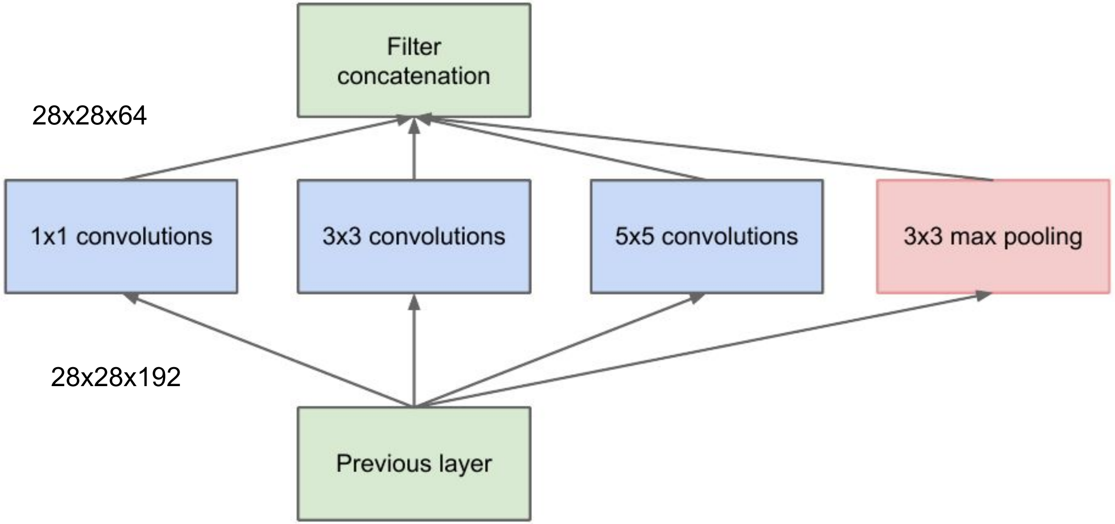
Blocks [VGG]



arXiv:1409.4842

Blocks [GoogleNet / Inception]

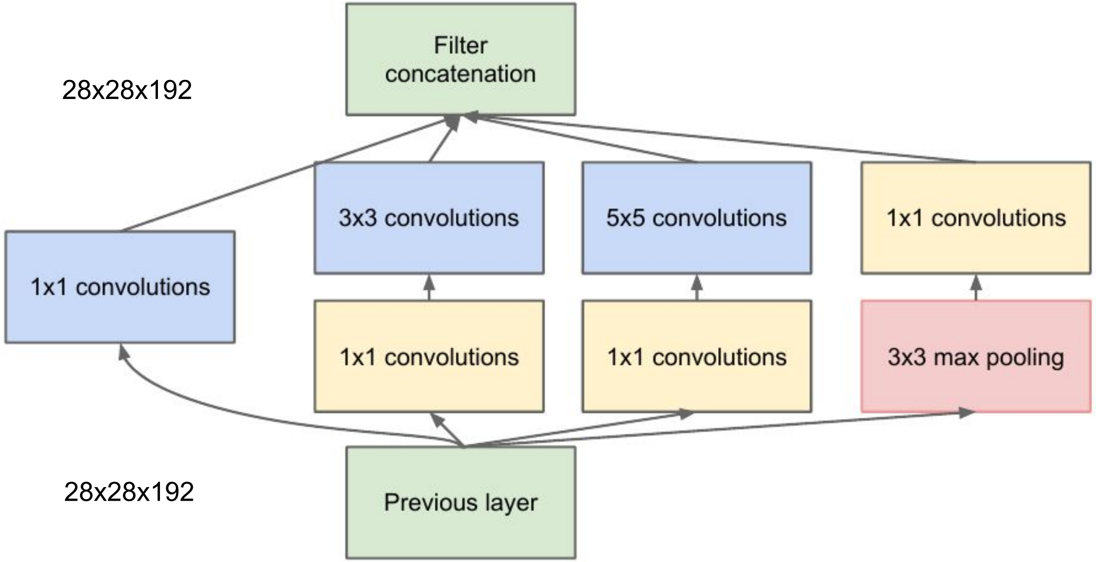
- to find out optimal local **sparse structure** and to repeat it spatially
- to split operations for cross-channel correlations and at spatial correlations into a series of independently operations.
- split-transform-merge strategy



arXiv:1409.4842

Blocks [GoogleNet / Inception]

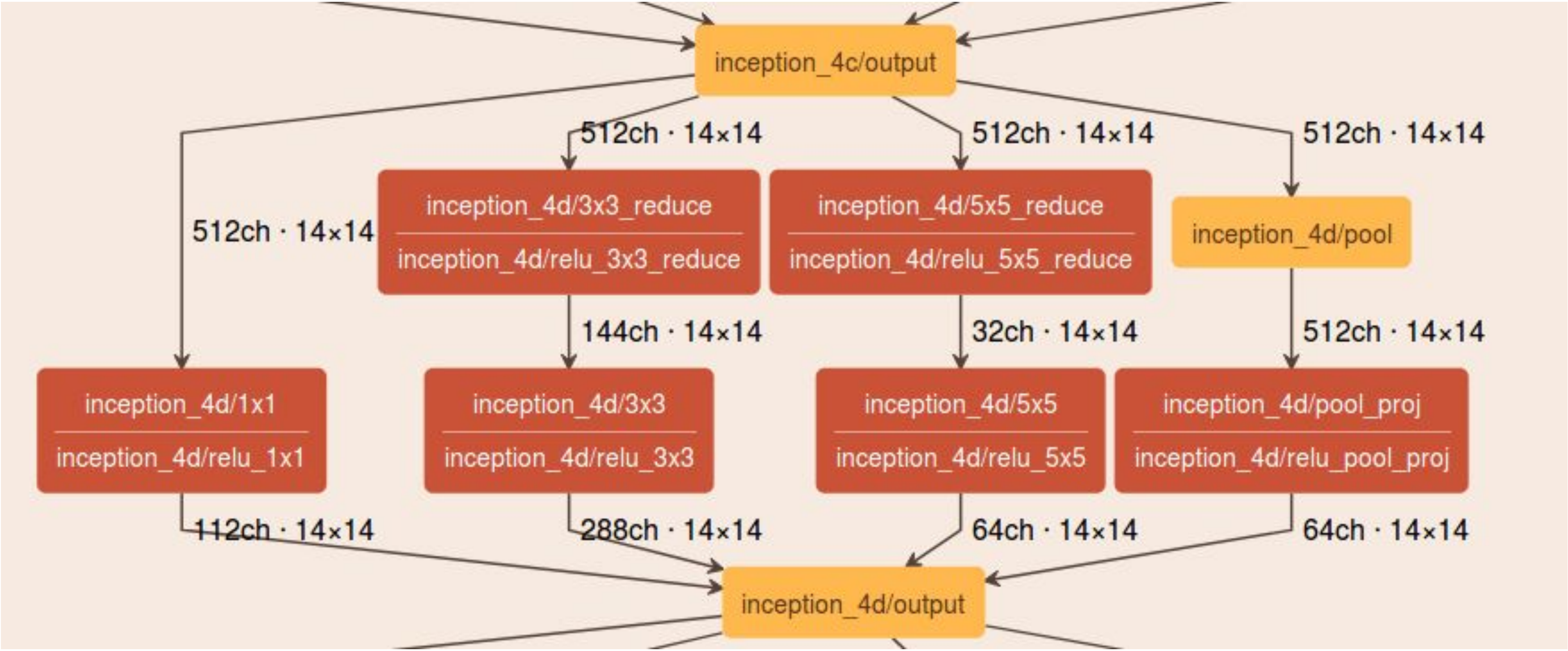
- to find out optimal local sparse structure and to repeat it spatially
- to split operations for cross-channel correlations and at spatial correlations into a series of independently operations.
- split-transform-merge strategy



Bottleneck

arXiv:1409.4842

Blocks [GoogleNet / Inception]

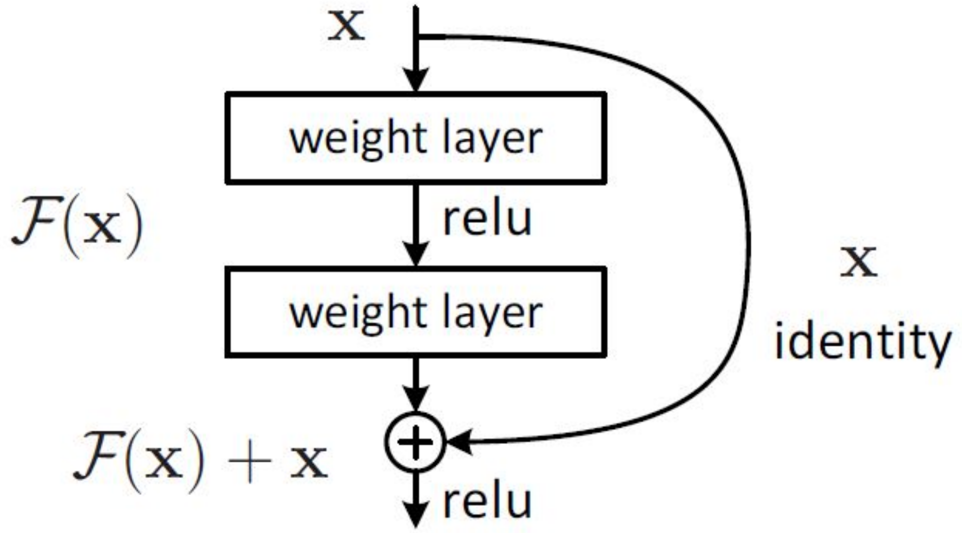


arXiv:1409.4842

Blocks [ResNet]

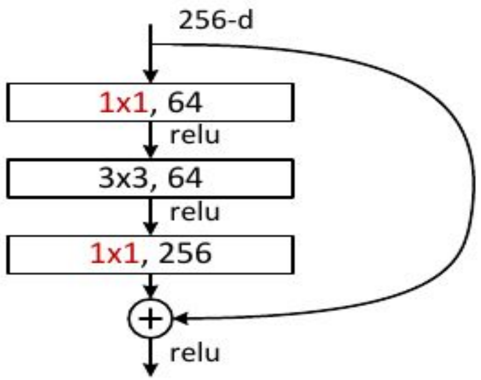
G(x) = x + F(x)

In the basic design, F(x) contains two 3×3 convolution layers along with a batch normalization and/or a rectied linear unit activation function.



arXiv:1512.03385

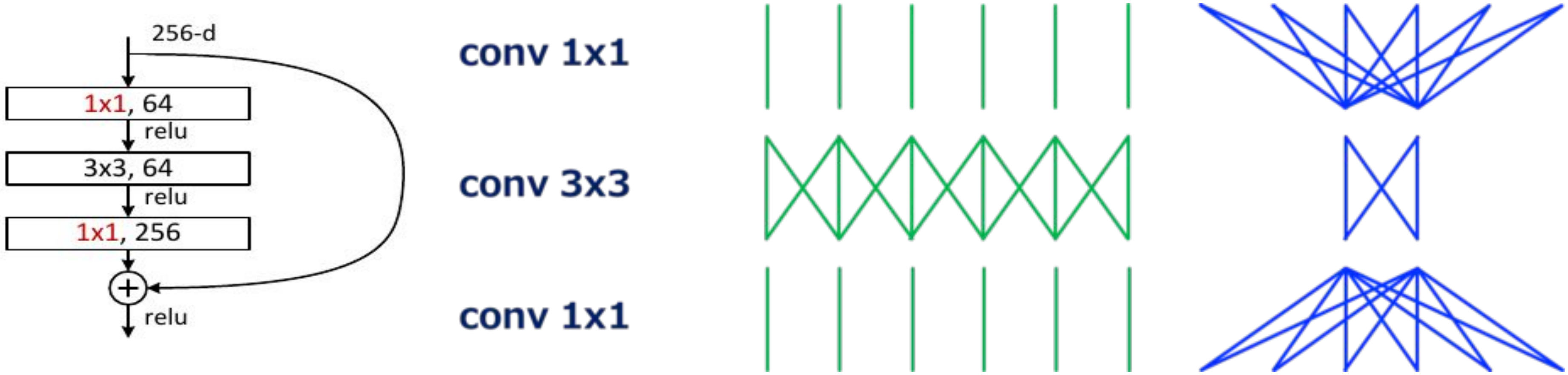
Blocks [ResNet, bottleneck]



For deeper networks (ResNet-50+), use “bottleneck” layer to improve efficiency (similar to Inception)

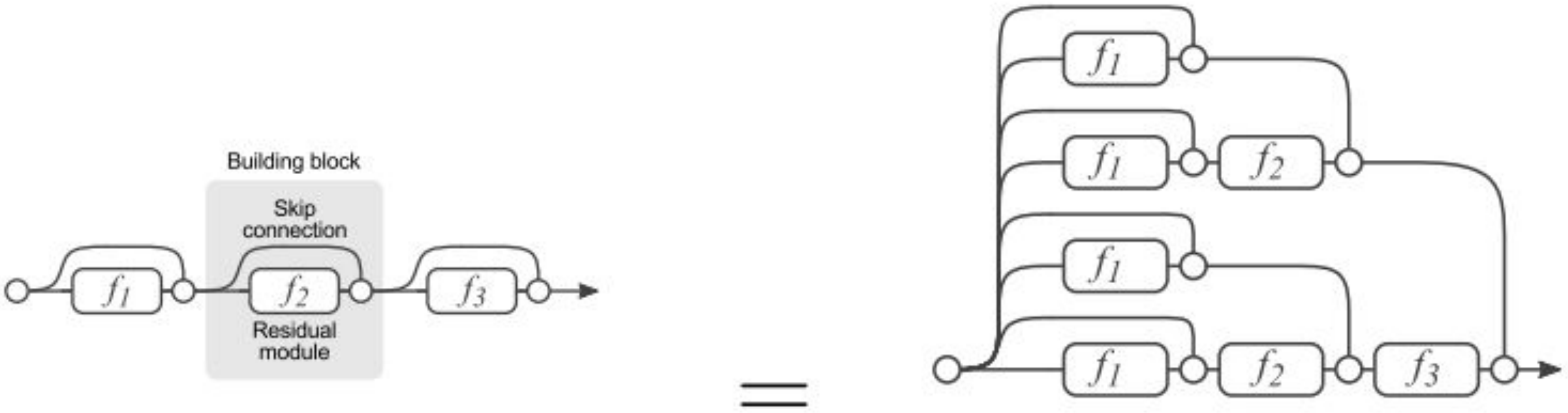
arXiv:1512.03385

Blocks [ResNet, bottleneck]

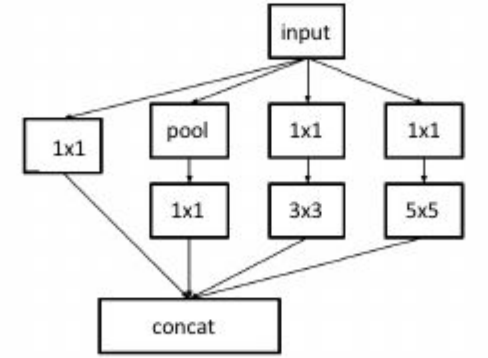


arXiv:1512.03385
https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

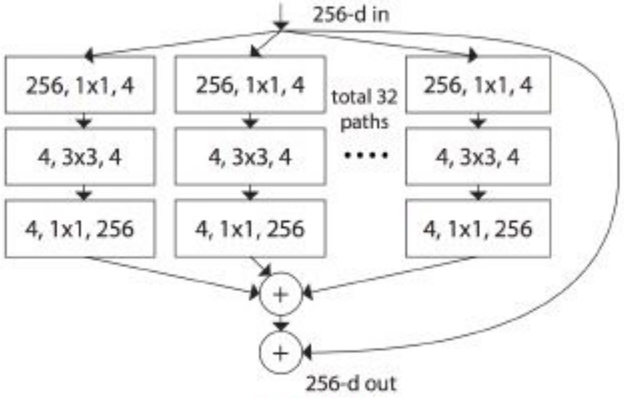
Blocks [ResNet, bottleneck]



Blocks [ResNeXt]



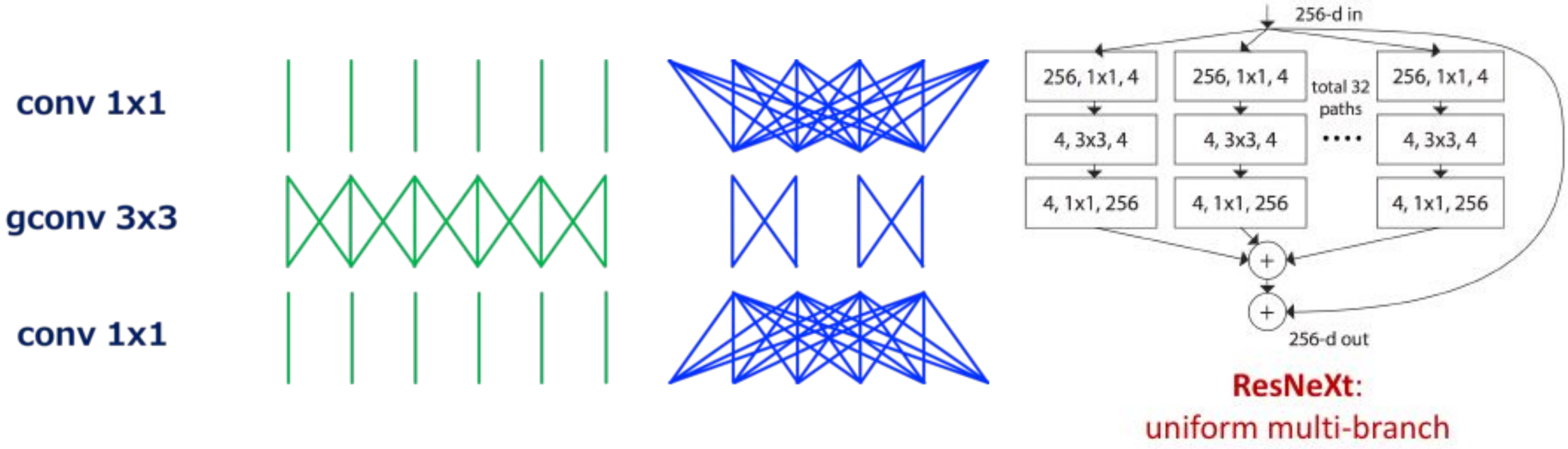
Inception:
heterogeneous multi-branch



ResNeXt:
uniform multi-branch

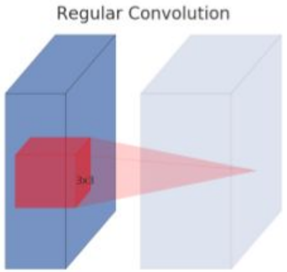
arXiv:1611.05431
https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

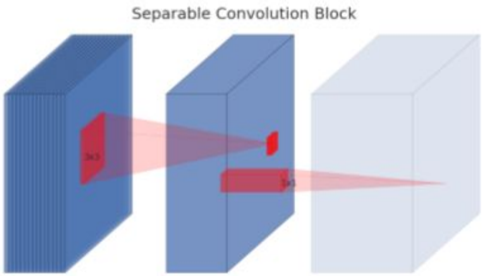
Blocks [ResNeXt]

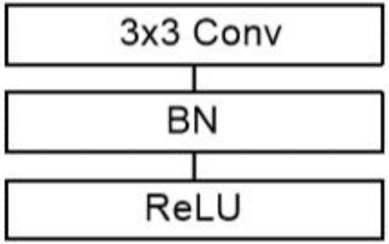


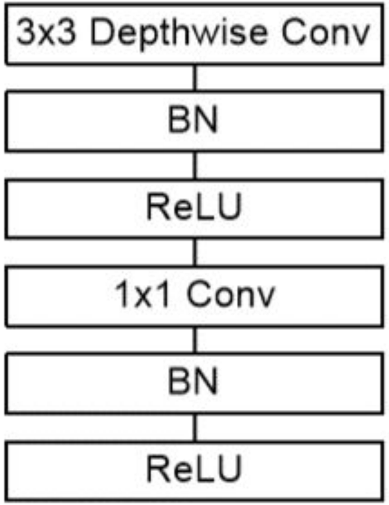
arXiv:1611.05431
https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

Blocks [MobileNet V1]



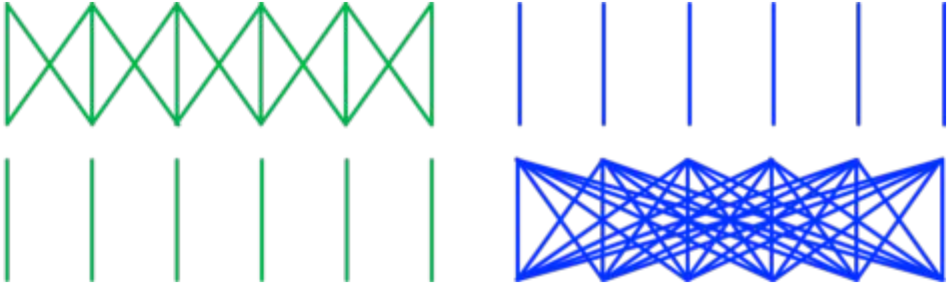






depthwise conv

conv 1x1

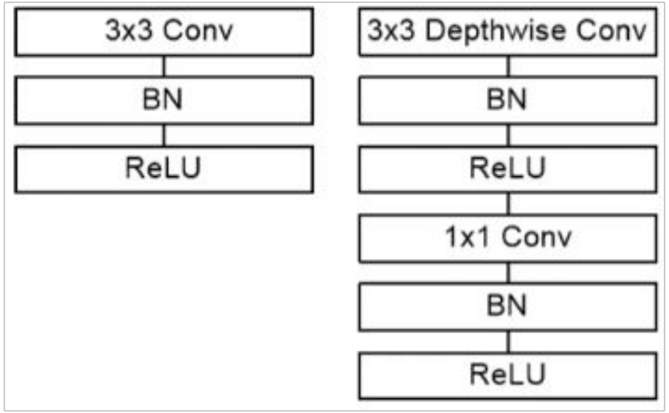


arXiv:1704.04861v3

https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

MobileNet v1

- M – number of input channels
- N – the number of output channels
- $D_K * D_K$ the kernel size
- $D_F * D_F$ the feature map size



Standard convolution computation cost

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

Depthwise separable convolution computational cost

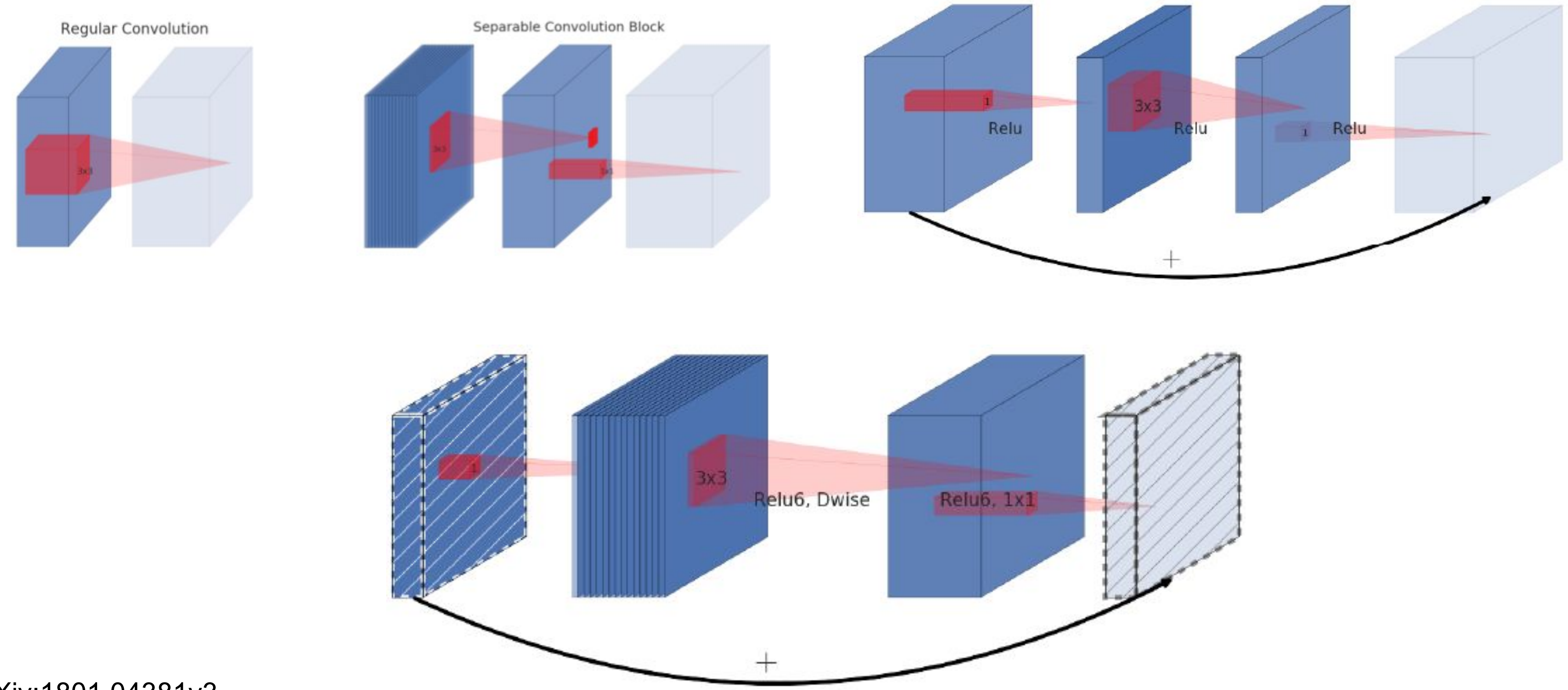
$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

Reduction in computation:

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

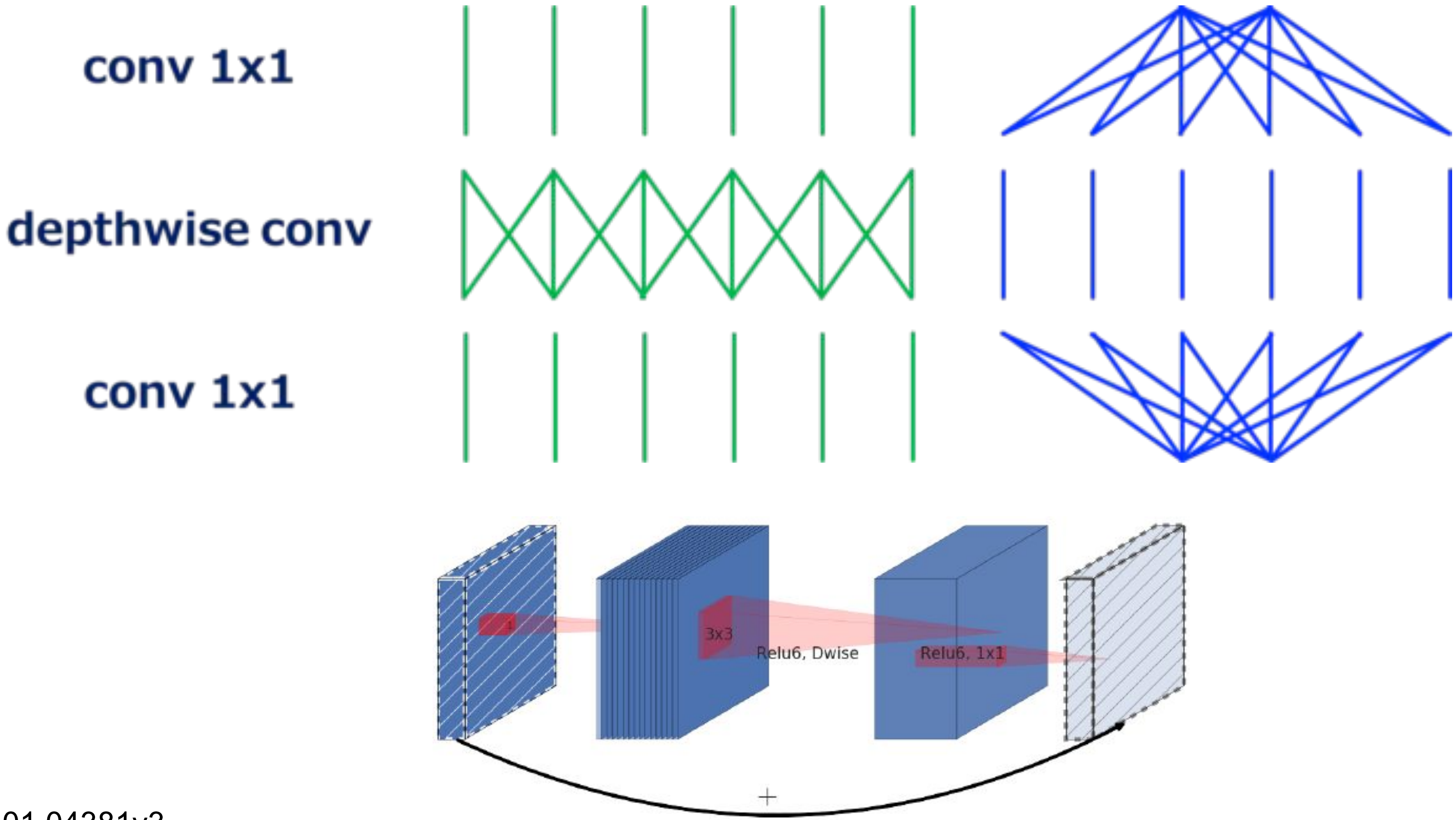
https://towardsdatascience.com/review-mobilenetv1-depthwise-separable-convolution-light-weight-model-a382df364b69

Blocks [MobileNet V2]



arXiv:1801.04381v3
https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

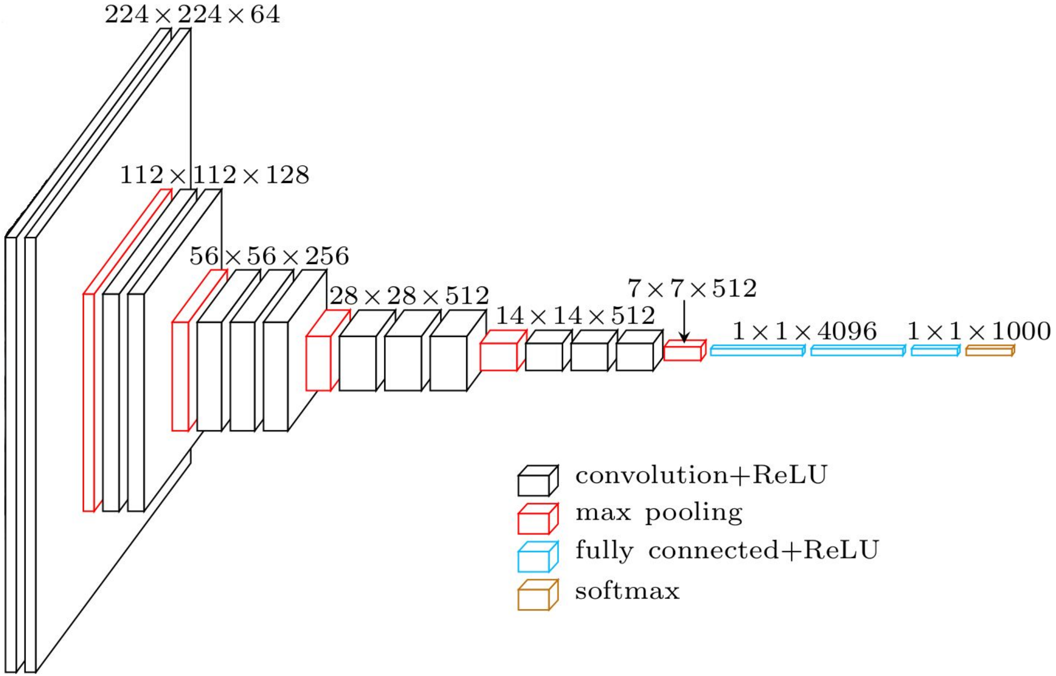
Blocks [MobileNet V2]



arXiv:1801.04381v3
https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

Contents

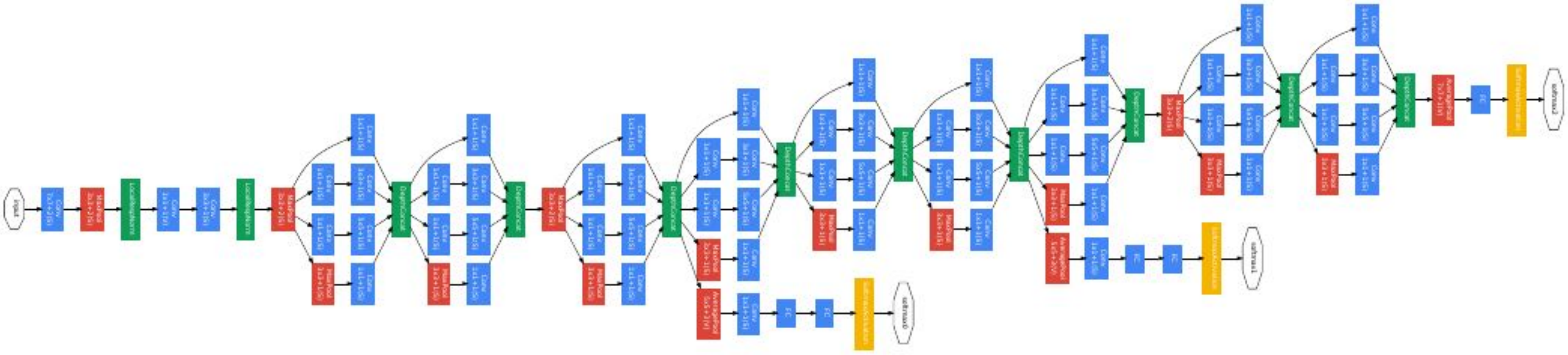
Net [VGG16]



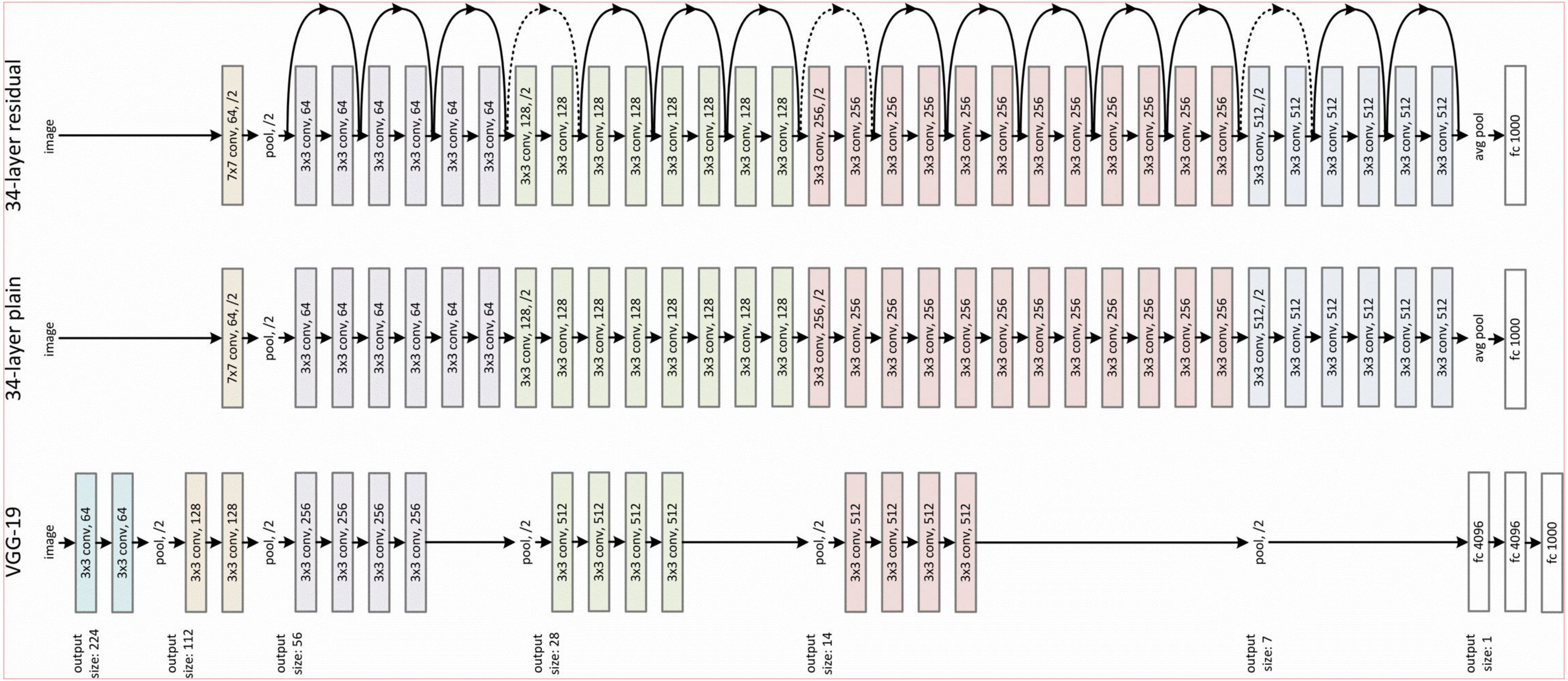
- 3 3x3 Conv as the module
- Stack the same module
- Same computation for each module

(1/2 spatial size => 2x filters)

Net [GoogLeNet]



Net [ResNet & ResNetX]



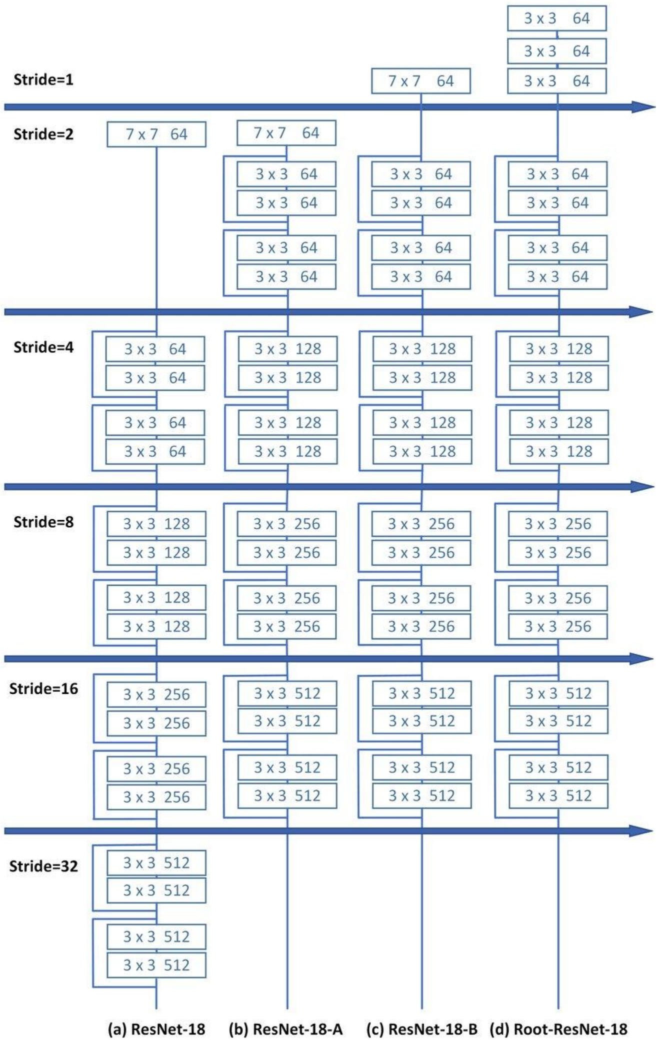
arXiv:1611.05431

Net [ResNet]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	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

Net [ResNet]

(a) ResNet18: original structure.
(b) ResNet-18-A: removing the first maxpooling layer.
(c) ResNet-18-B: changing the stride size in the first conv layer from 2 to 1.
(d) Root-ResNet-18: replacing the 7 × 7 conv layer with three stacked 3 × 3 conv layers in ResNet18-B.
The corresponding mAPs on PASCAL 2007 test (training on "07+12" from scratch) are 73.1%, 75.3%, 77.6% and 78.5%, respectively.



https://arxiv.org/abs/1810.08425

Net [ResNetX]

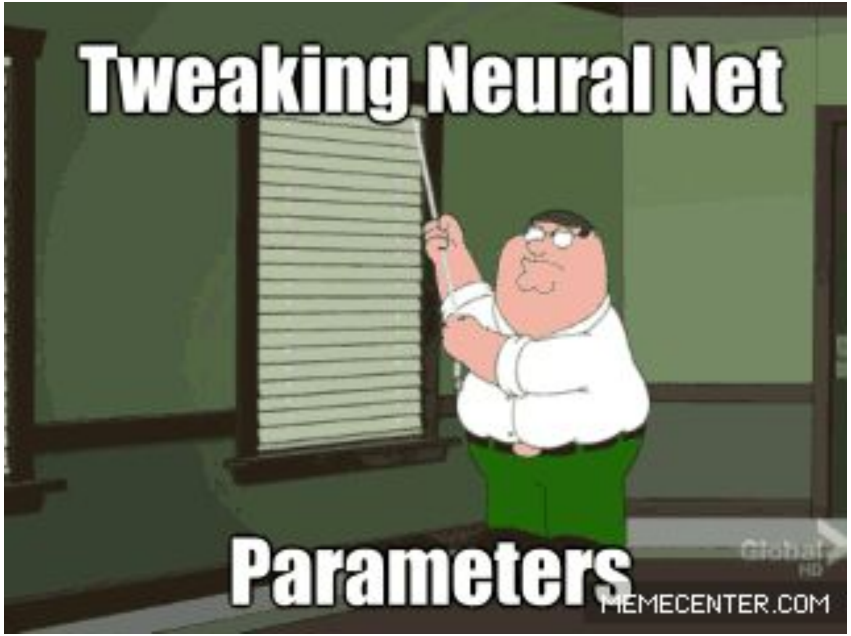
Table 1. (**Left**) ResNet-50. (**Right**) ResNeXt-50 with a 32×4d template (using the reformulation in Fig. 3(c)). Inside the brackets are the shape of a residual block, and outside the brackets is the number of stacked blocks on a stage. "C=32" suggests grouped convolutions [24] with 32 groups. *The numbers of parameters and FLOPs are similar between these two models.*

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.		**25.5**$\times10^6$	**25.0**$\times10^6$
FLOPs		**4.1**$\times10^9$	**4.2**$\times10^9$

https://arxiv.org/abs/1611.05431

Contents

Neural Architecture Search (NAS)



Neural Architecture Search (NAS)

- **searching for best blocks**
  The is finding one particular building block which is then repeated many times to create the deep architecture.

- **starting from units**
  Discover new connections of the whole net

Neural Architecture Search (NAS)

start from units/simple blocks

- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise-separable conv
- 7x7 depthwise-separable conv
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise-seperable conv

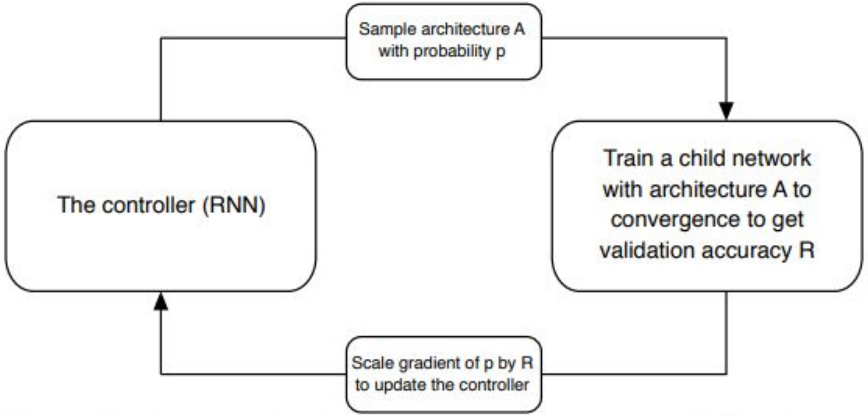
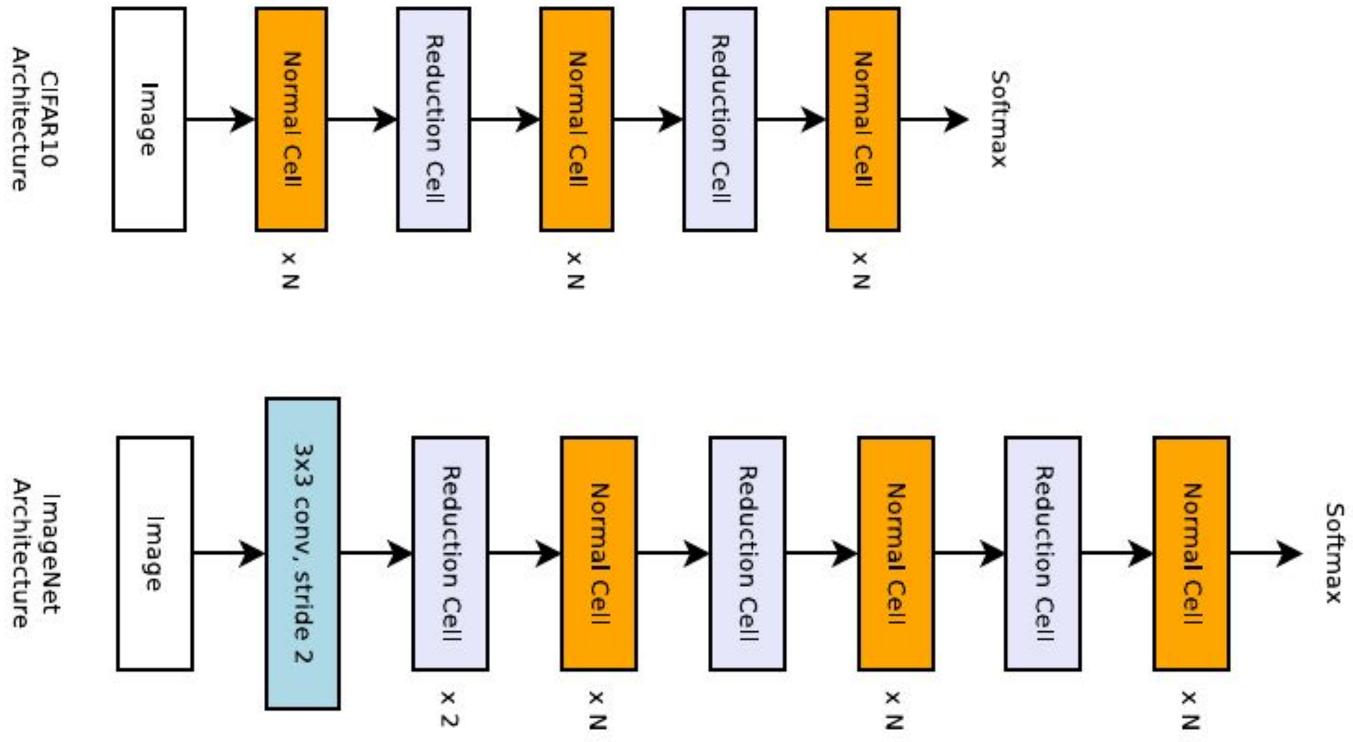


Figure 1. Overview of Neural Architecture Search [71]. A controller RNN predicts architecture A from a search space with probability p. A child network with architecture A is trained to convergence achieving accuracy R. Scale the gradients of p by R to update the RNN controller.

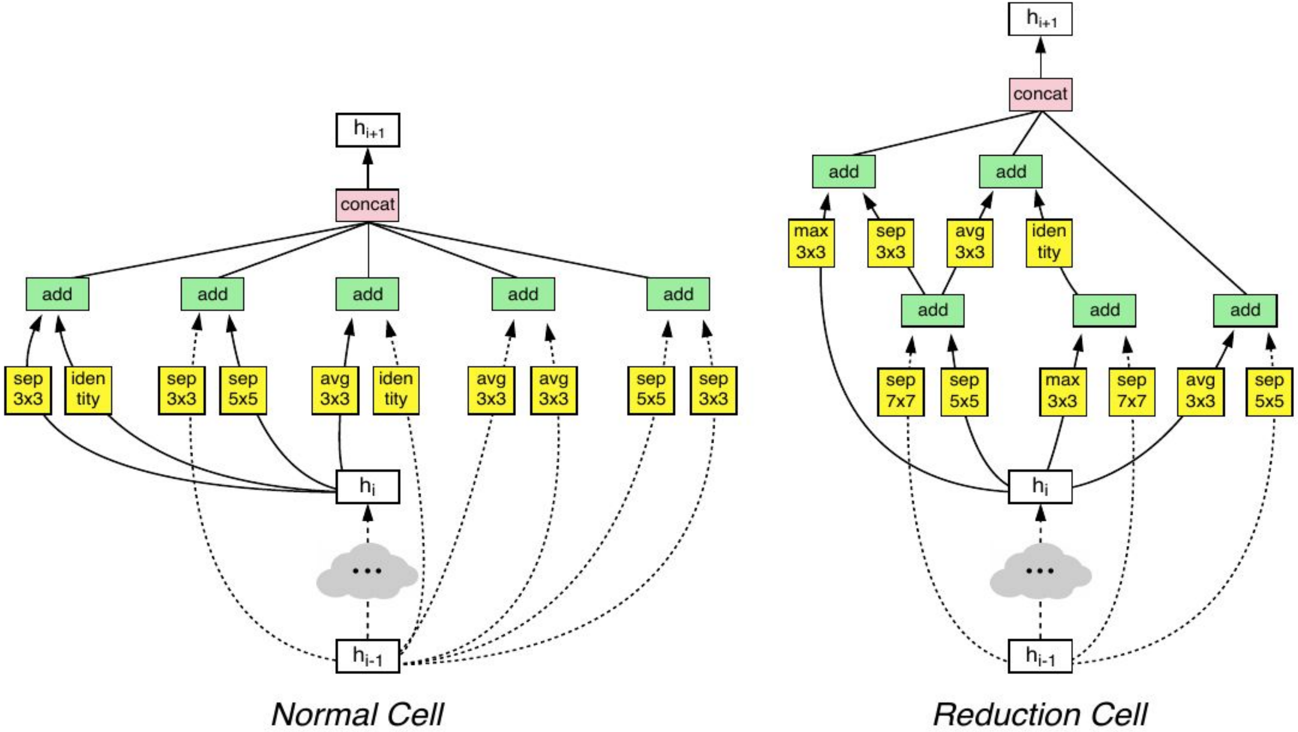
arXiv:1707.07012

Neural Architecture Search (NAS)



arXiv:1707.07012

Neural Architecture Search (NAS)

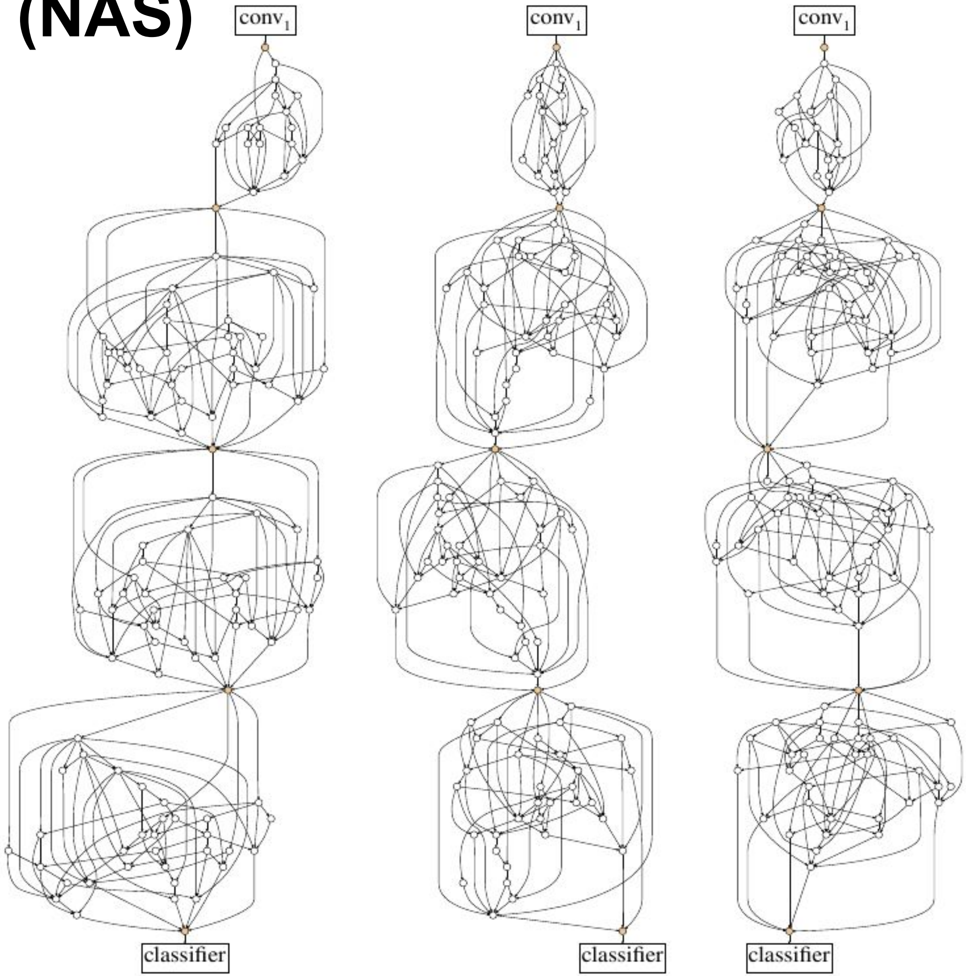


arXiv:1707.07012

Neural Architecture Search (NAS)

Exploring Randomly Wired Neural Networks for Image Recognition [*arXiv:1904.01569*]

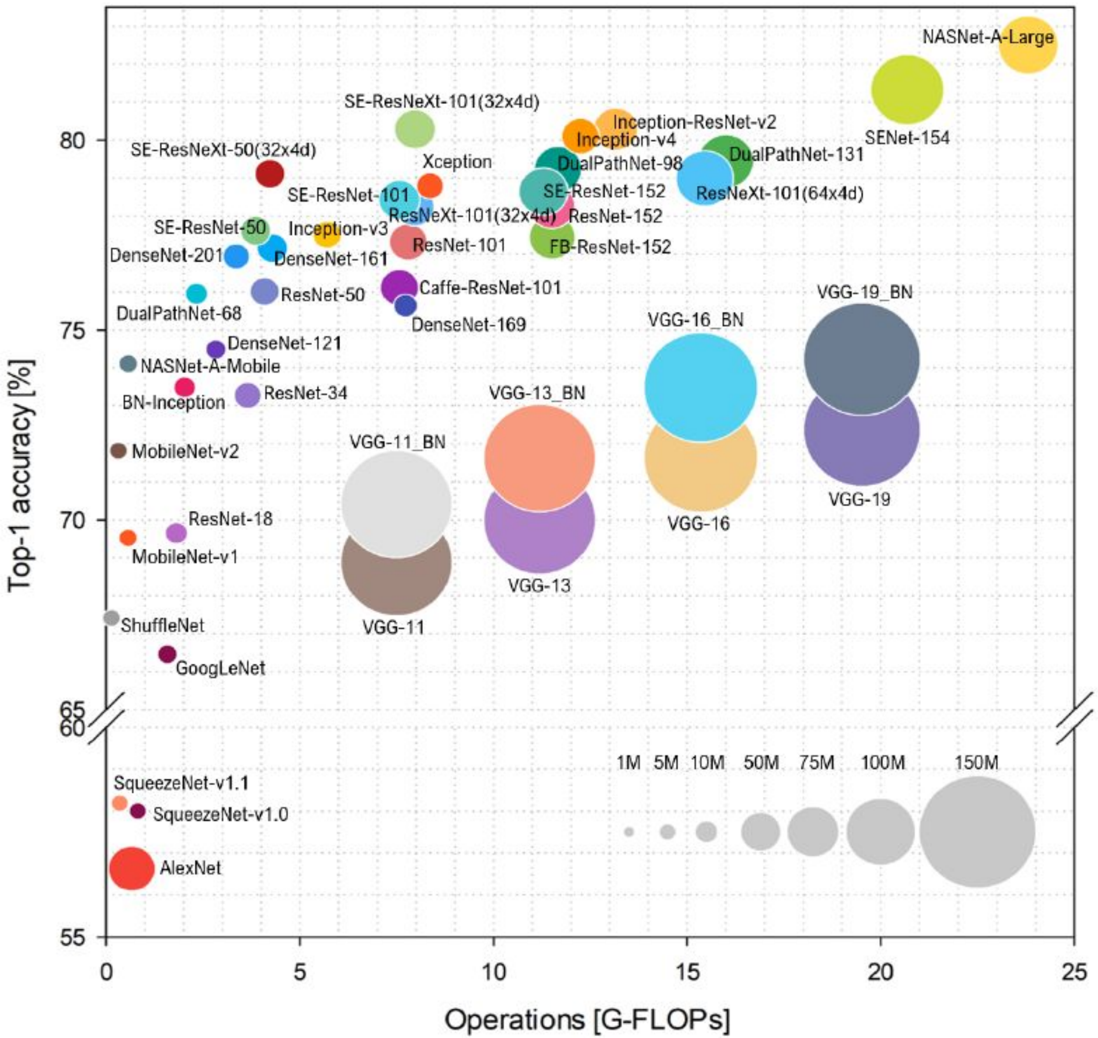
use three classical random graph models to generate randomly wired graphs for networks



arXiv:1904.01569

Contents

Net [Comparison]



https://arxiv.org/pdf/1810.00736.pdf

Contents

Units

Layers [Convolution]
Layers [Convolution] [Receptive field]
Layers [Dilated Convolution, Deformable Convolution]
Layers [Upsampling, Learnable Upsampling]
Layers [Batch Norm, Dropout]
Layers [Group Convolutions and its variants]

Blocks

VGG
Inception
ResNet*
MobileNet*

Architectures

VGG, Inception, ResNet*, MobileNet*

Neural Architecture Search (NAS)

Summary

Summary

- When you see huge DN, don’t be scare. Usually it can be decomposed.
- Automatic topology learning (NAS) is rising, but it is still important to understand basic blocks/units
- Classification problem is solved, but features matter