Deep Learning for Computer Vision

Andrii Liubonko
Samsung R&D Institute Ukraine

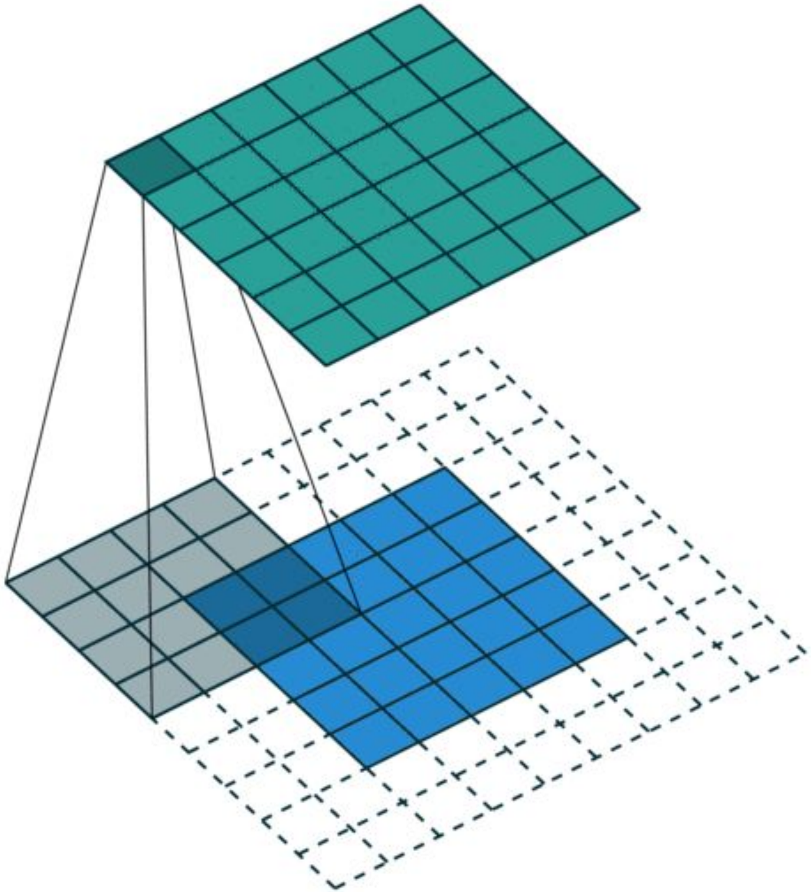
Contents

Layers [Convolution]

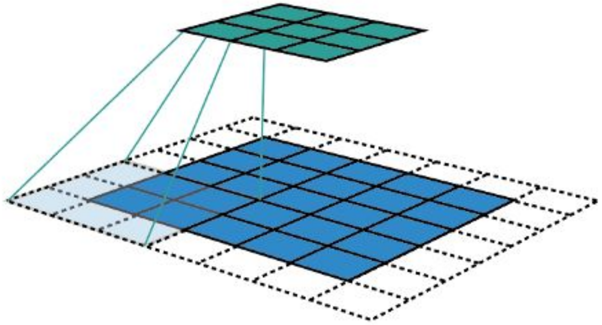
- Accepts a volume of size $W1 \times H1 \times D1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W2 \times H2 \times D2$ where:
  - $W2 = (W1 - F + 2P)/S + 1$,
  - $H2 = (H1 - F + 2P)/S + 1$
  - $D2 = K$
- With parameter sharing, it introduces $F \times F \times D1$ weights per filter, for a total of $(F \times F \times D1) \times K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W2 \times H2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Layers [Convolution]

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



Layers [Convolution] [Receptive field]



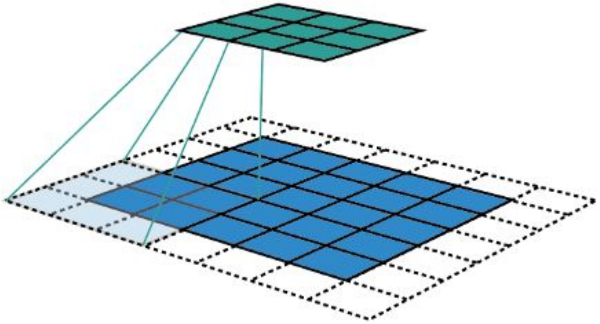
The **receptive field** is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by)

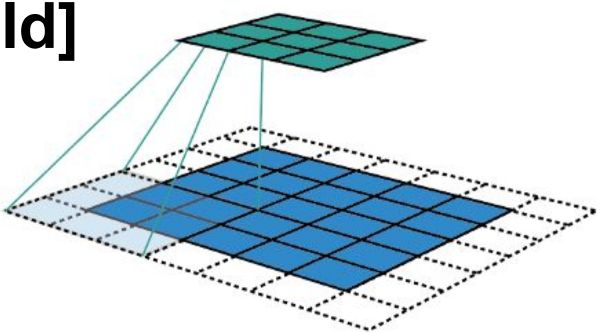
- kernel size, F = 3x3
- padding size, P = 1
- stride, S = 2

A receptive field of a feature can be described by
- its center location
- its size

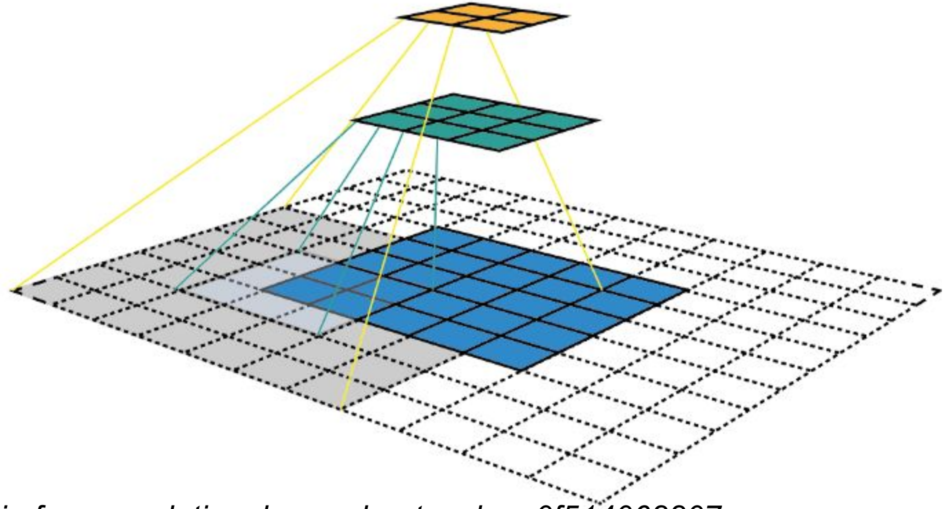
https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807

Layers [Convolution] [Receptive field]



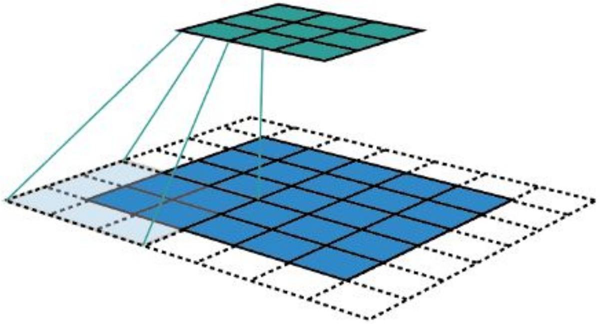


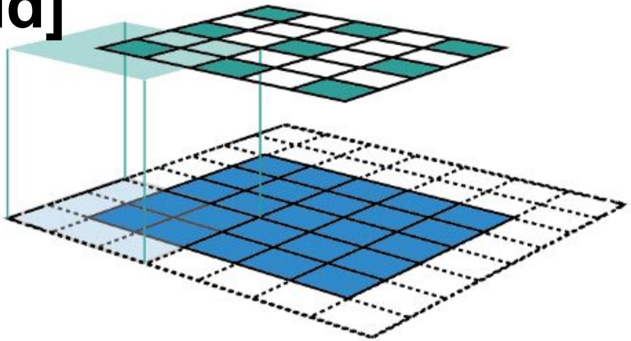
- kernel size F = 3x3,
- padding size P = 1,
- stride S = 2

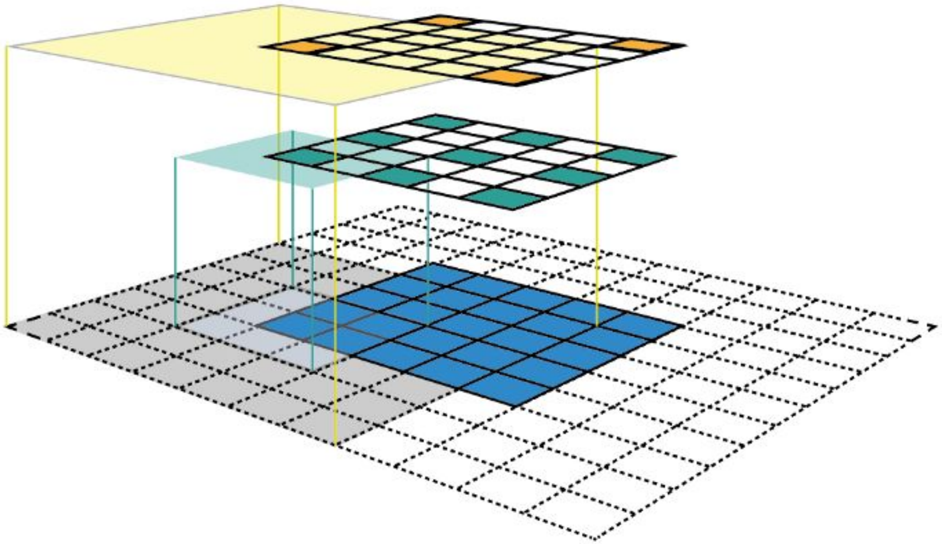


https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807

Layers [Convolution] [Receptive field]



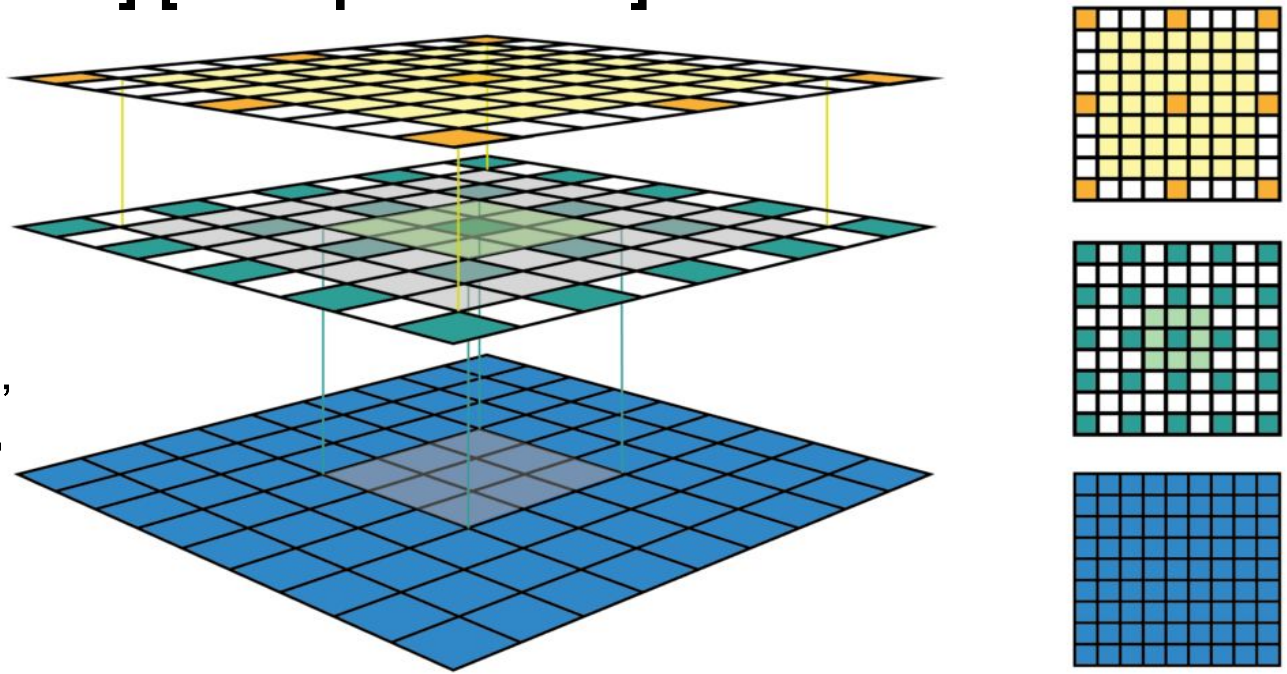




- kernel size F = 3x3,
- padding size P = 1,
- stride S = 2

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

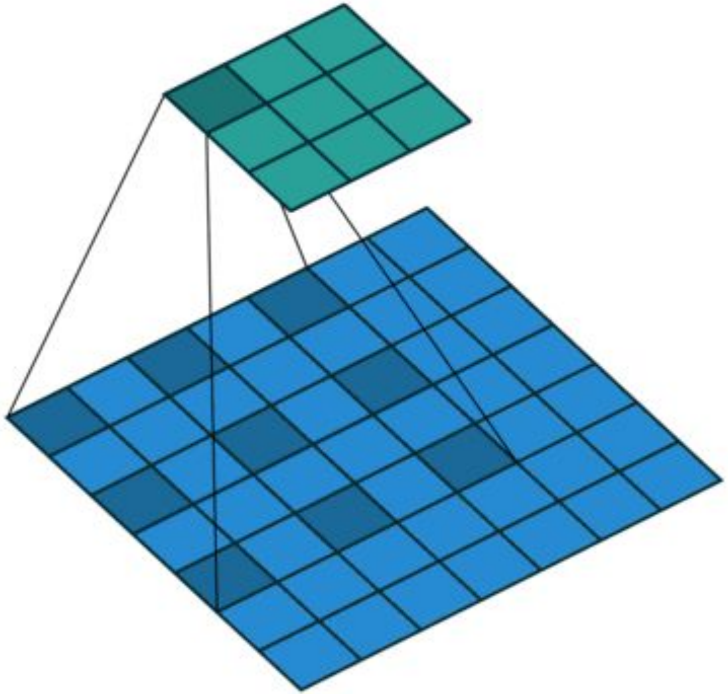
Contents

Layers [Dilated Convolution]

[Used for *Semantic Segmentation*, *Object Recognition,* especially to include context information]

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**.



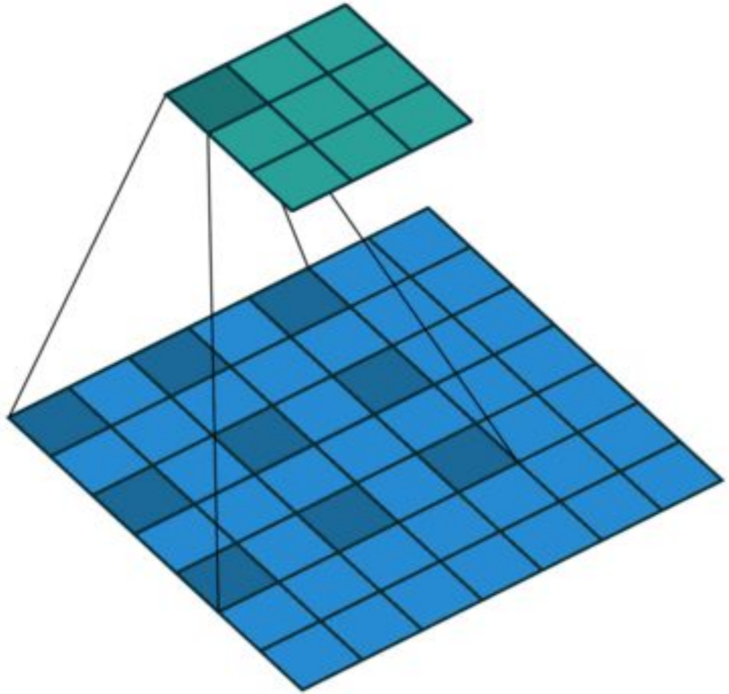
- input size I = 7x7
- kernel size F = 3x3,
- padding size P = 0,
- stride S = 1
- dilation rate **d** = 2

arXiv:1511.07122

Layers [Dilated Convolution]

[Used for *Semantic Segmentation*, *Object Recognition,* especially to include context information]

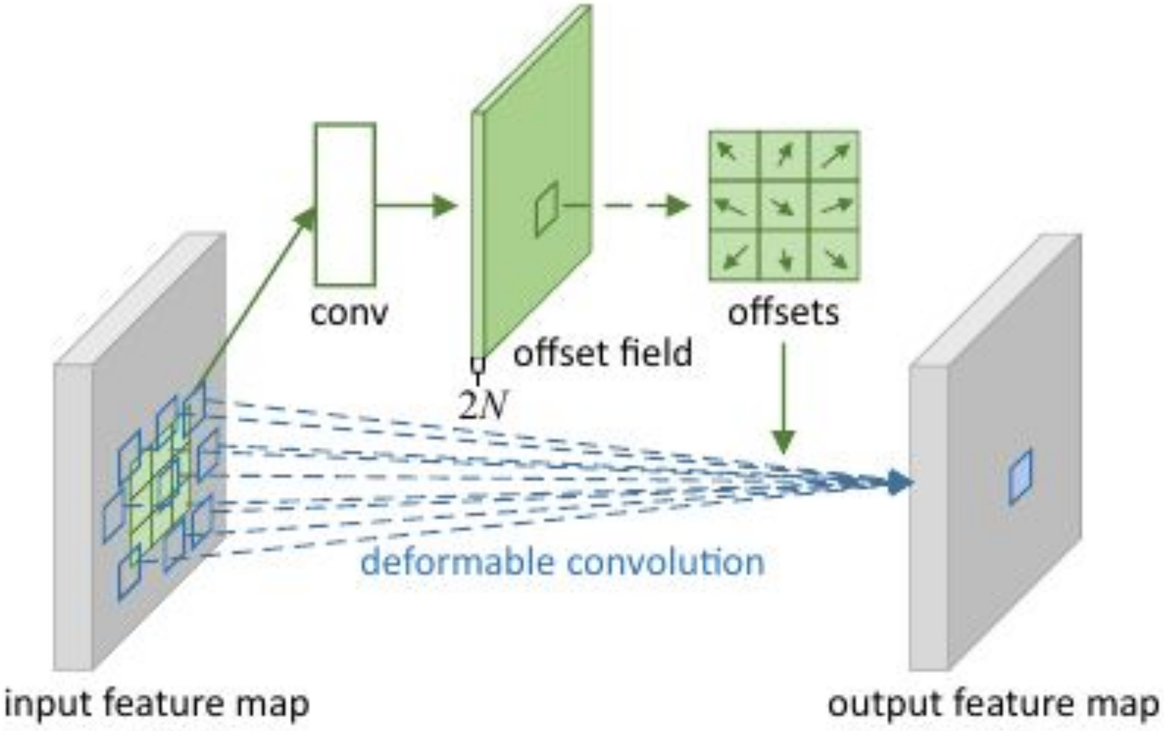
- Detection of fine-details by processing inputs in higher resolutions.
- Broader view of the input to capture more contextual information.
- Faster run-time with less parameters



arXiv:1511.07122

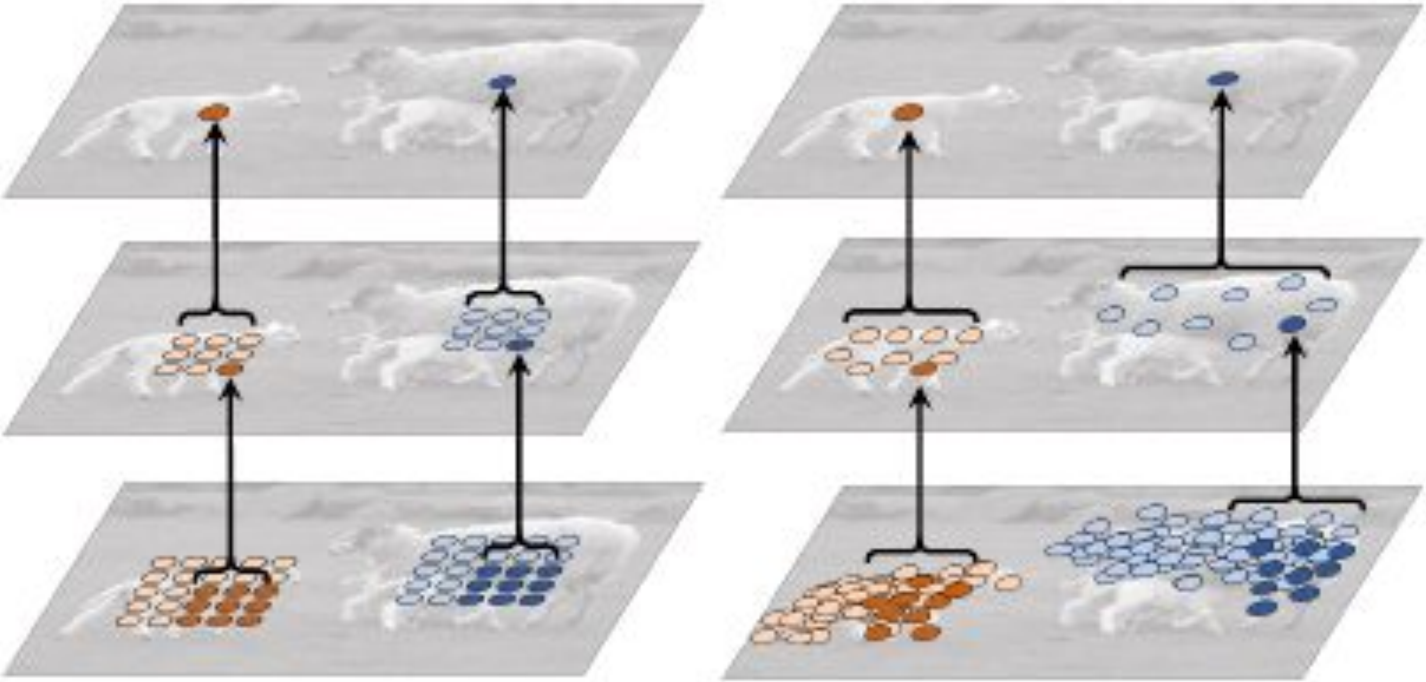
Layers [Deformable Convolution]

[Used for *Semantic Segmentation*, *Object Recognition*]



arXiv:1703.06211

Layers [Deformable Convolution]



(a) standard convolution (b) deformable convolution

arXiv:1703.06211

Layers [Deformable Convolution]

arXiv: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

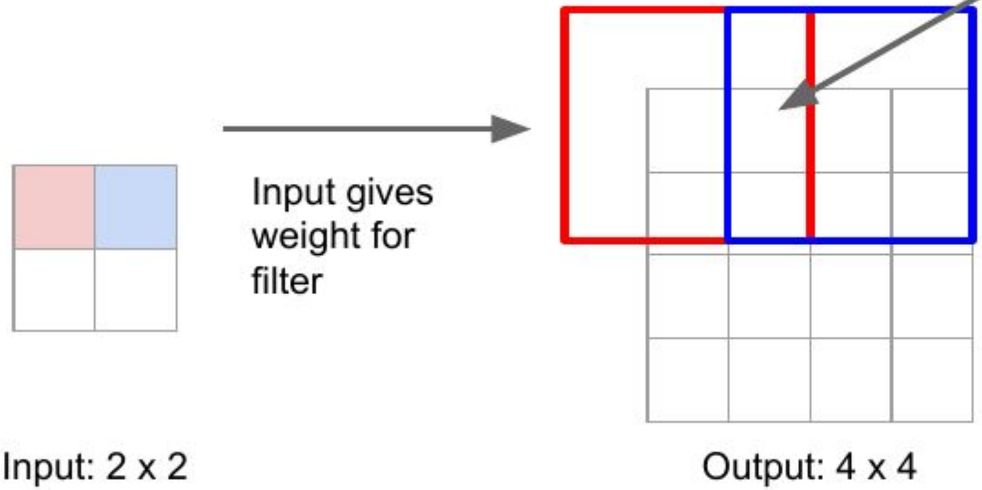
Output: [2 x 2]

Layers [Learnable Upsampling: Transpose Convolution]

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

Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

3 x 3 **transpose** convolution, stride 2 pad 1



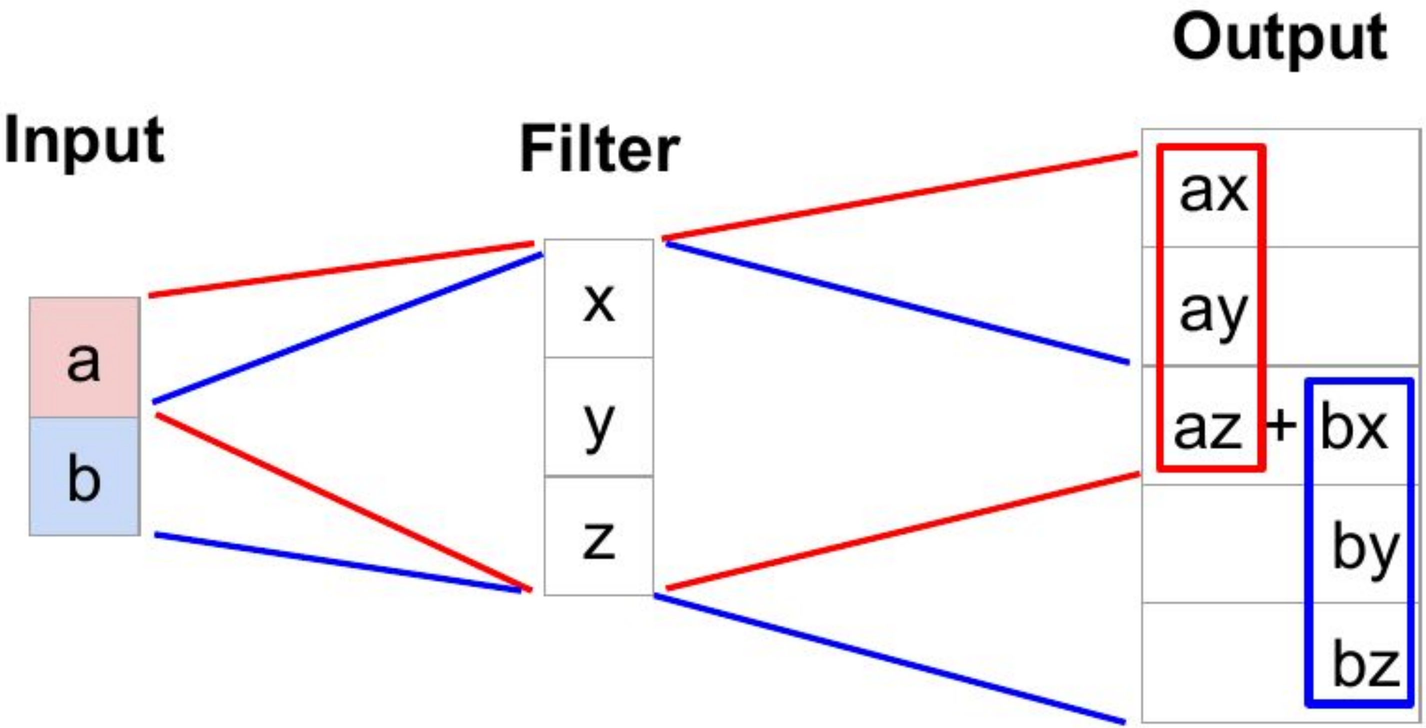
Sum where output overlaps

Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

Stride gives ratio between movement in output and input

http://cs231n.stanford.edu

Layers [Learnable Upsampling: Transpose Convolution]

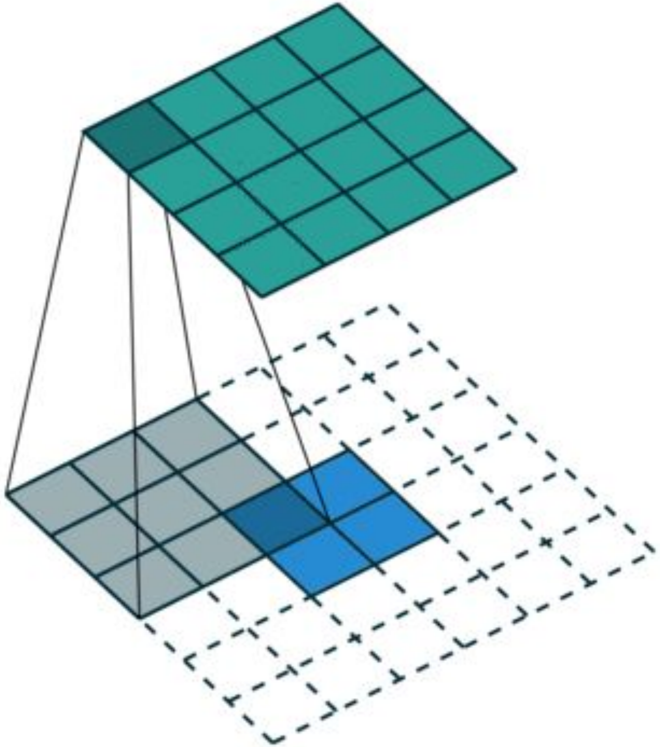


Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

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

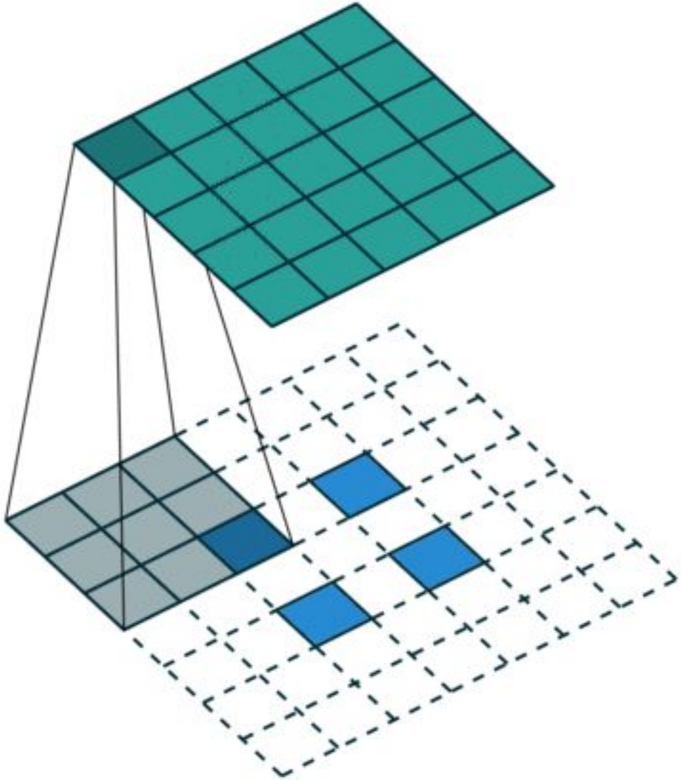
http://cs231n.stanford.edu

Layers [Learnable Upsampling: Transpose Convolution]



https://github.com/vdumoulin/conv_arithmetic & arXiv:1603.07285

Layers [Learnable Upsampling: Transpose Convolution]



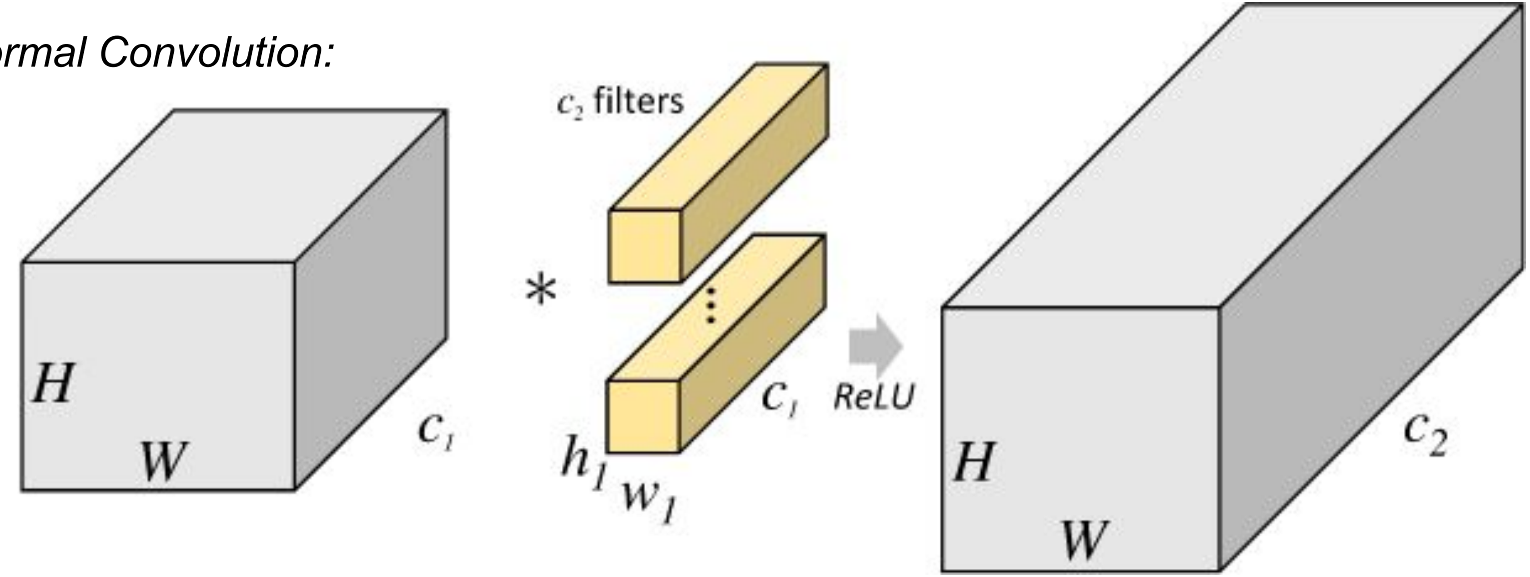
https://github.com/vdumoulin/conv_arithmetic & arXiv:1603.07285

Contents

Layers [Group Convolution]

[Used in AlexNet, but also in modern architectures like ResNeXt, ShuffleNet]

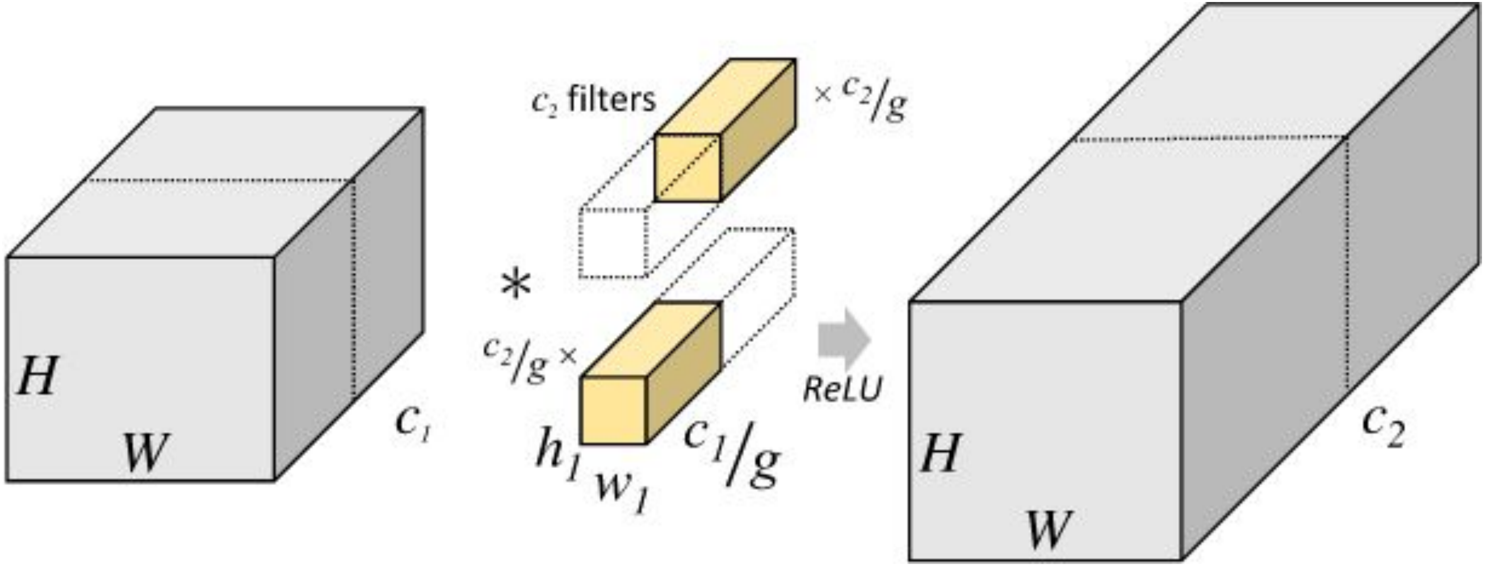
N*ormal Convolution:*



https://blog.yani.io/filter-group-tutorial/

Layers [Group Convolution]

[Used in AlexNet, but also in modern architectures like ResNeXt, ShuffleNet]

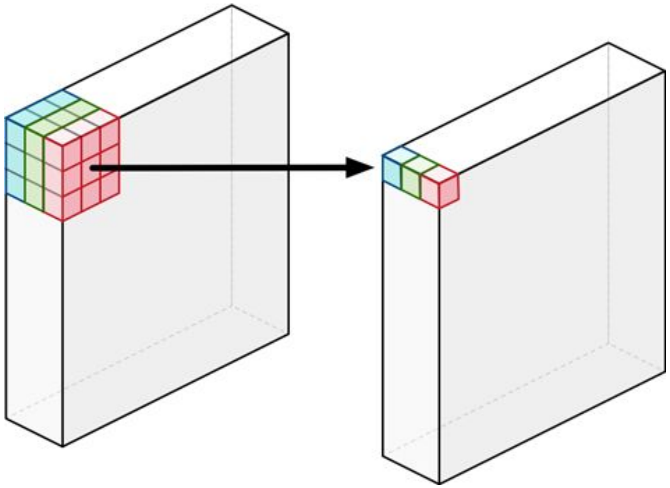


A special case of grouped convolutions is when *g* equals the number of input channels.
This is called **depth-wise convolutions** or **channel-wise convolutions**

https://blog.yani.io/filter-group-tutorial/

Layers [Group Convolution]

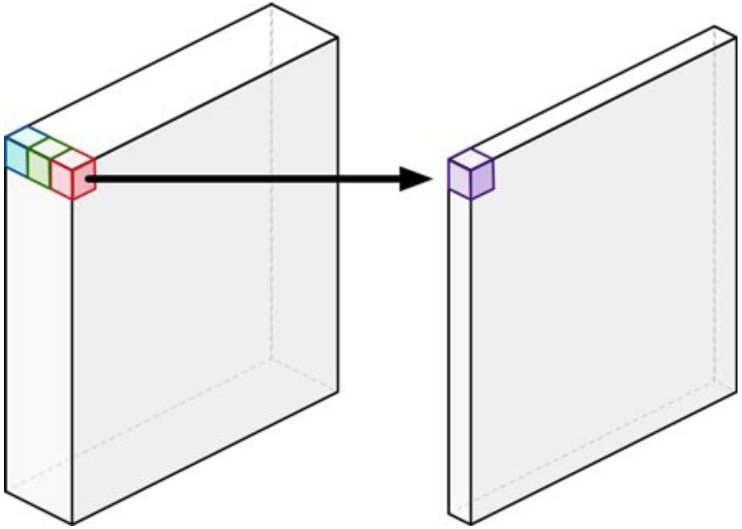
A special case of grouped convolutions is when *g* equals the number of input channels. This is called **depth-wise convolutions** or **channel-wise convolutions**



http://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/

Layers [Pointwise convolution]

[Used in modern architectures like MobileNet, ShuffleNet]



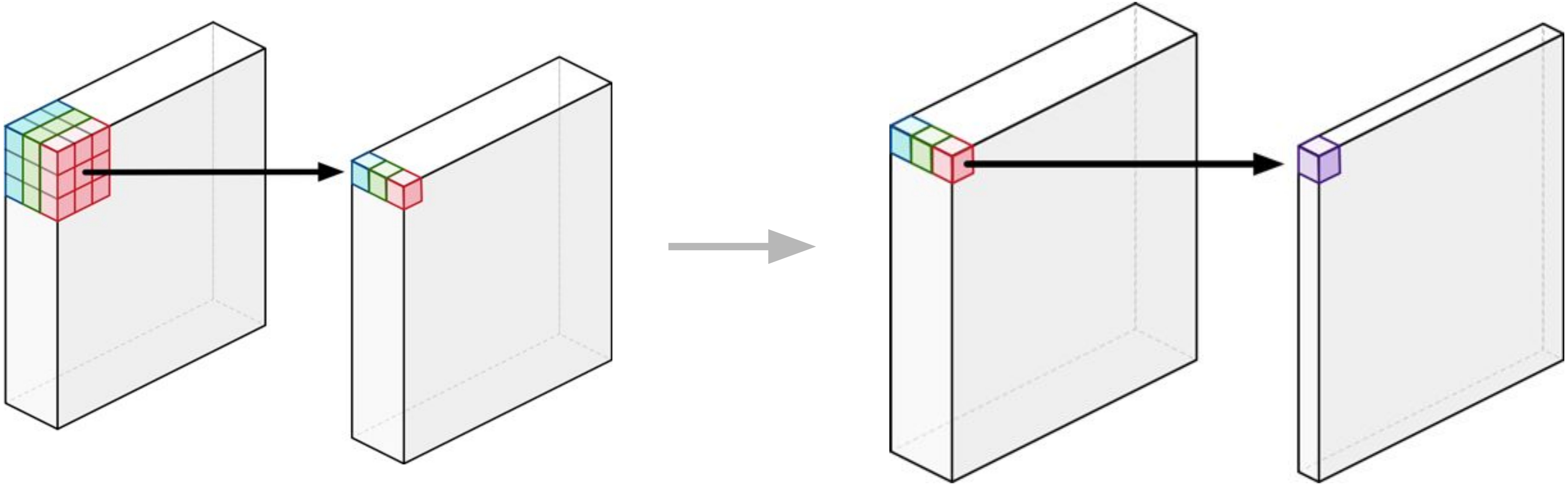
This really is the same as a regular convolution but with a 1×1 kernel

http://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/

Layers [Separable Convolution]

[Used in modern architectures like MobileNet, ShuffleNet]

Separable convolutions = depth-wise convolutions + point-wise convolutions

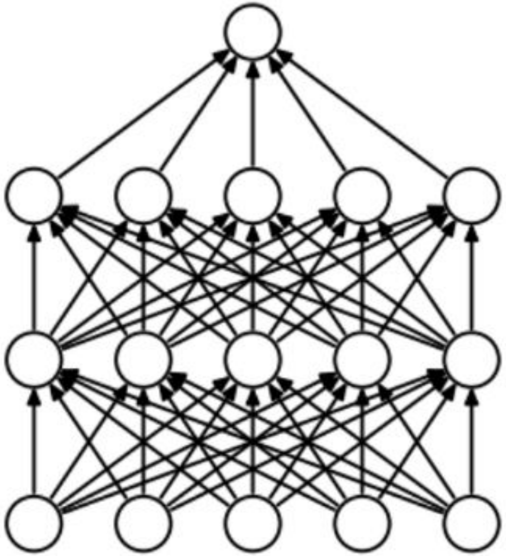


http://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/

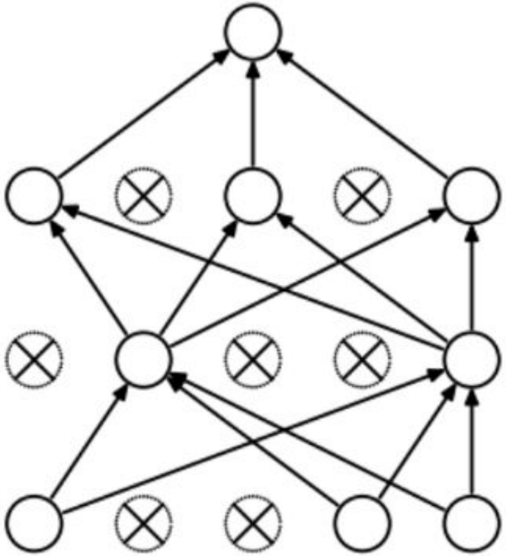
Contents

Layers [Dropout]

- In-network ensembling
- Reduce overfitting (might be instead done by BN)



(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

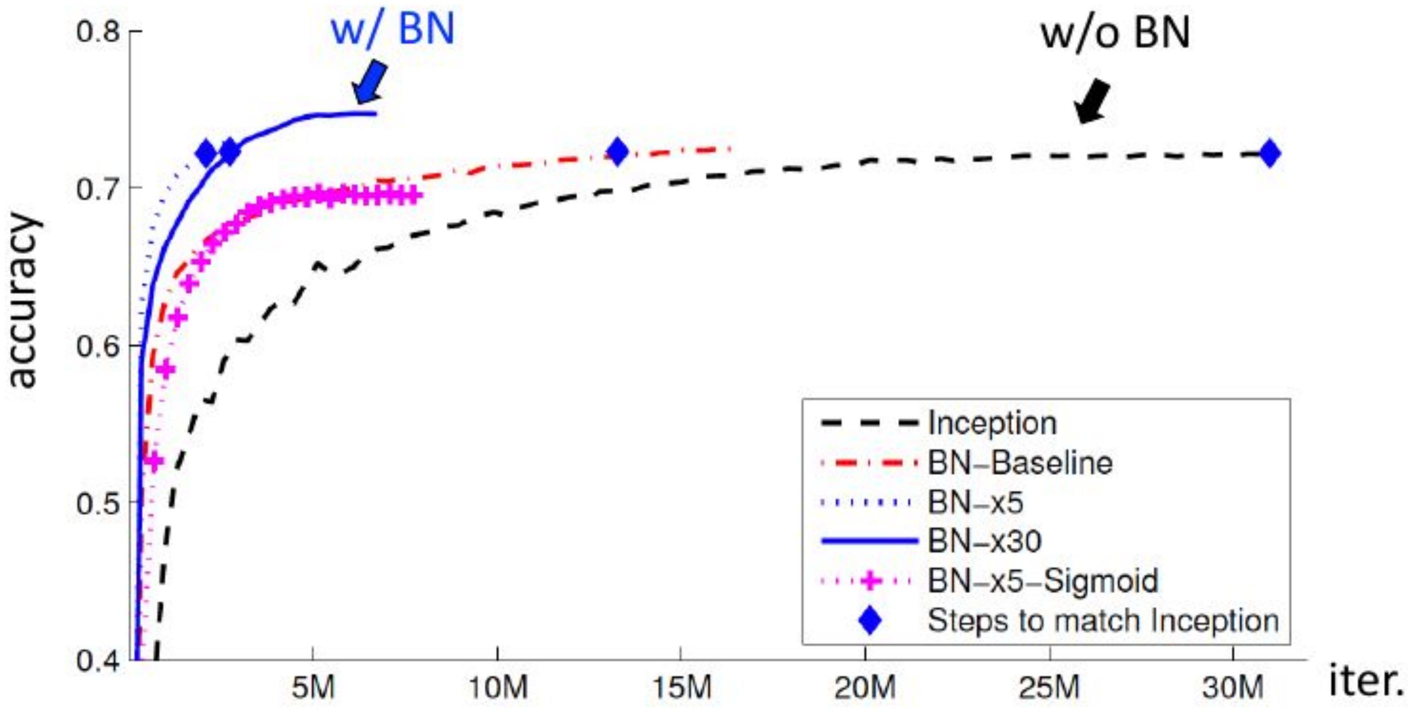
$$\rightarrow \boxed{\text{layer}} \rightarrow x \rightarrow \hat{x} = \frac{x-\mu}{\sigma} \rightarrow y = \gamma\hat{x} + \beta$$

- 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

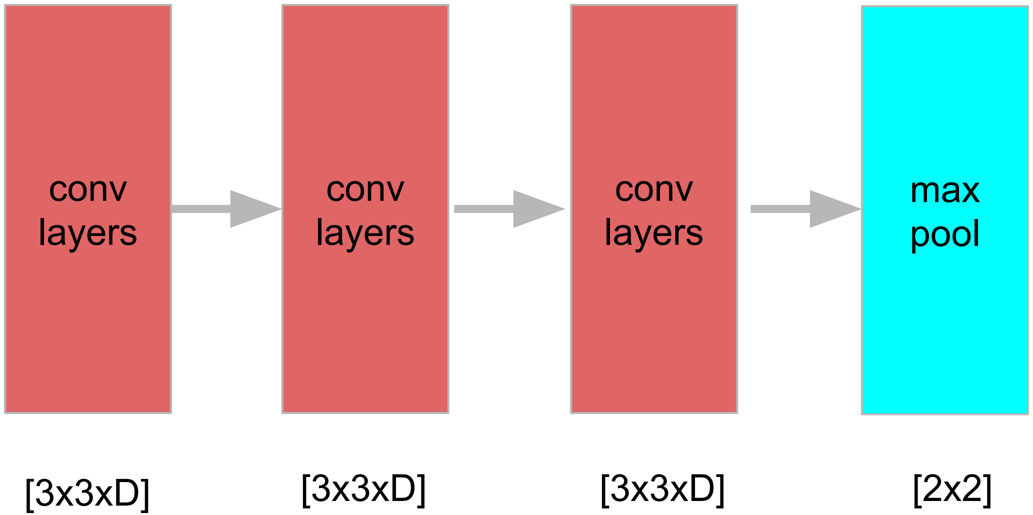
Layers [Batch Normalization]



arXiv:1502.03167

Contents

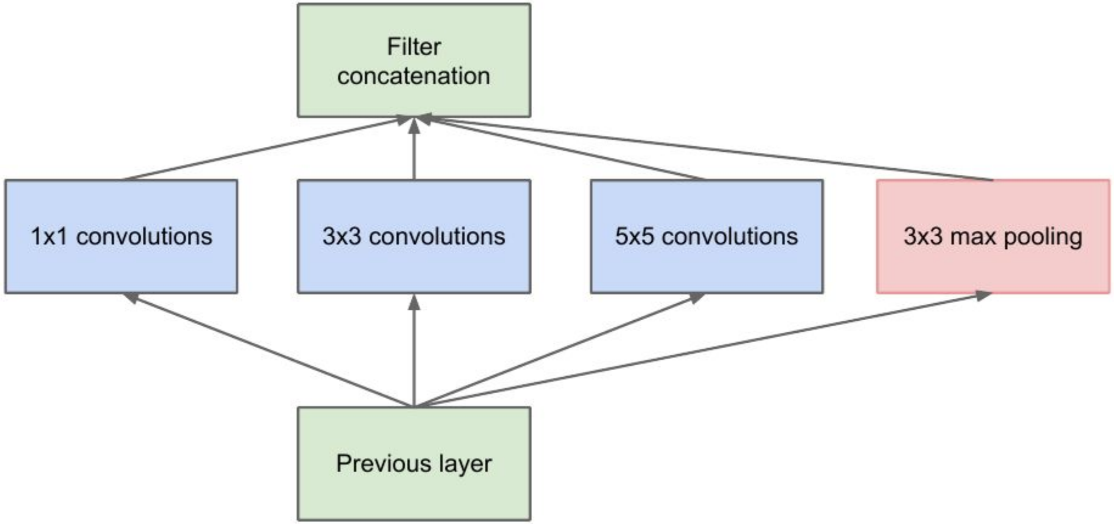
Blocks [VGG]



arXiv:1409.4842

Blocks [GoogleNet / Inception]

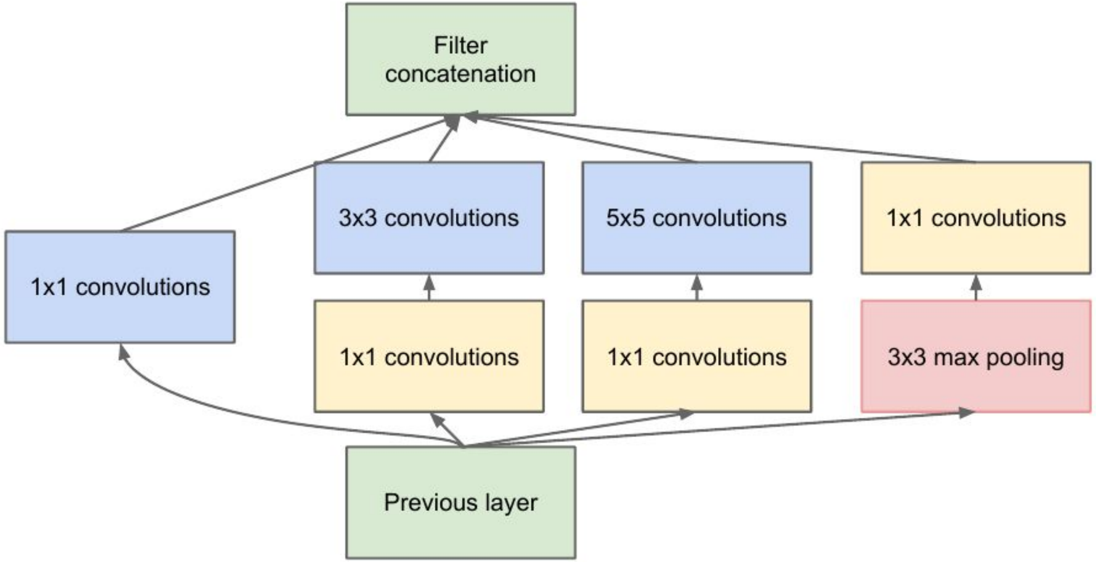
- to find out optimal local **sparse structure** and to repeat it spatially
- to split operations for cross-channel correlations and at spatial correlations into a series of independently operations.
- split-transform-merge strategy



arXiv:1409.4842

Blocks [GoogleNet / Inception]

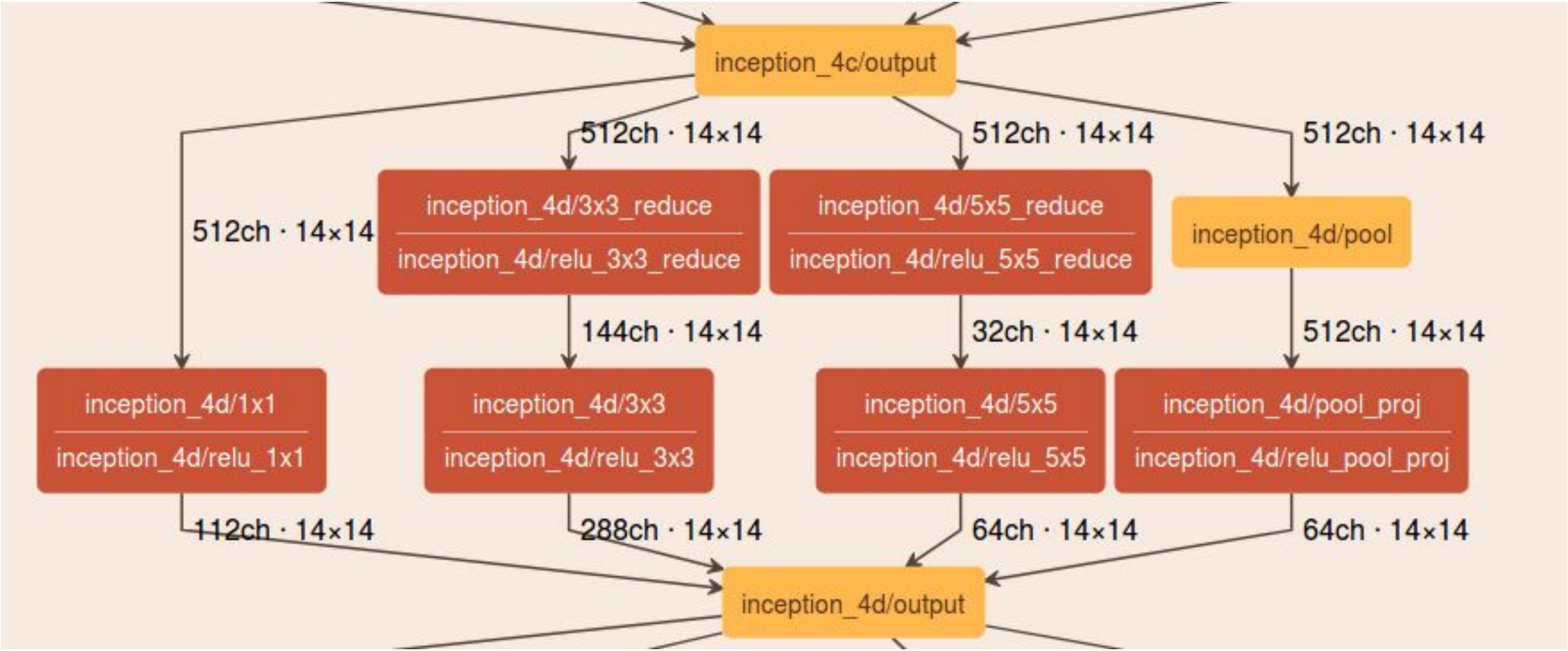
- to find out optimal local sparse structure and to repeat it spatially
- to split operations for cross-channel correlations and at spatial correlations into a series of independently operations.
- split-transform-merge strategy



Bottleneck

arXiv:1409.4842

Blocks [GoogleNet / Inception]

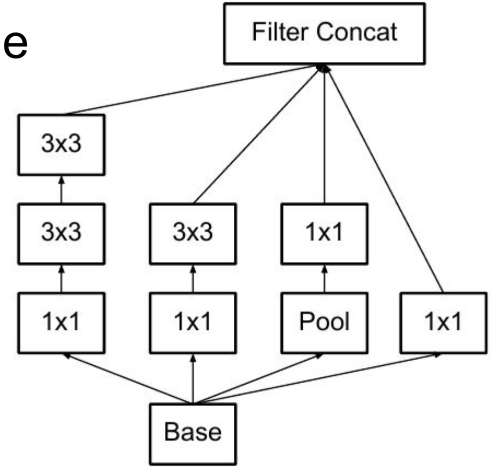


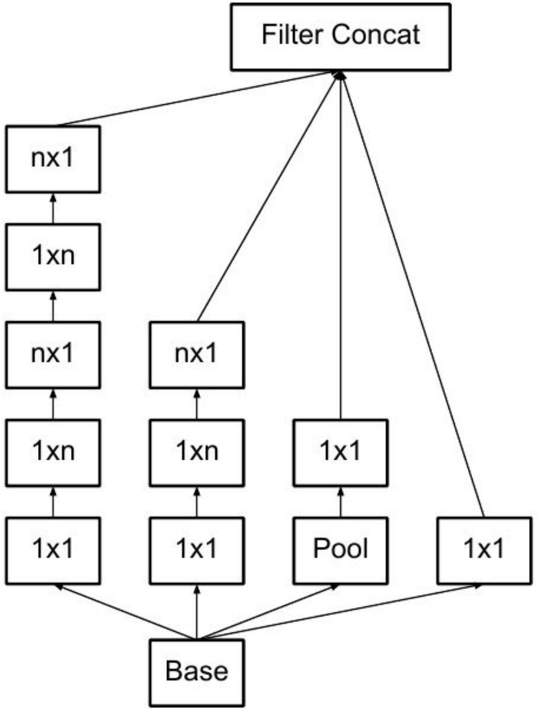
arXiv:1409.4842

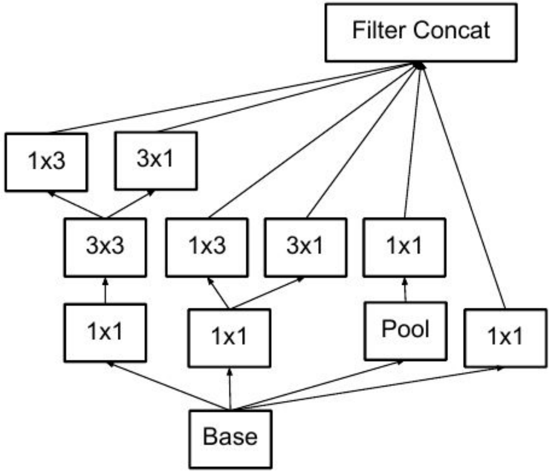
Blocks [GoogleNet / Inception v1-v3]

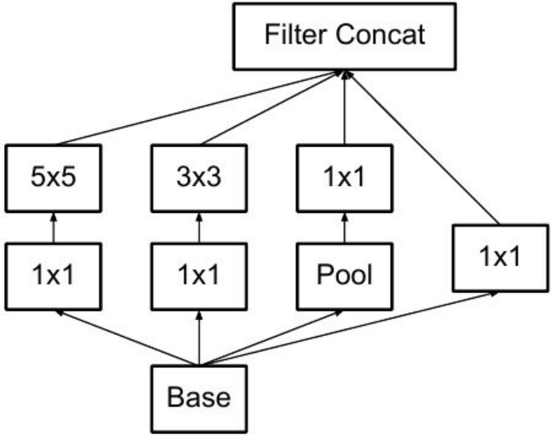
More templates, but the same 3 main properties are kept:

- Multiple branches
- Shortcuts (1x1, concate.)
- Bottleneck







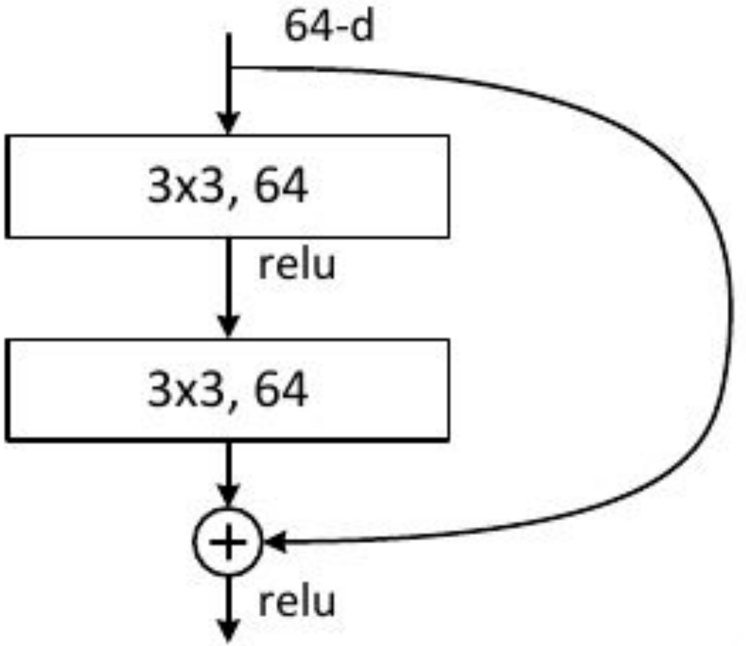


arXiv:1512.00567

Blocks [ResNet]

$$G(x) = x + F(x)$$

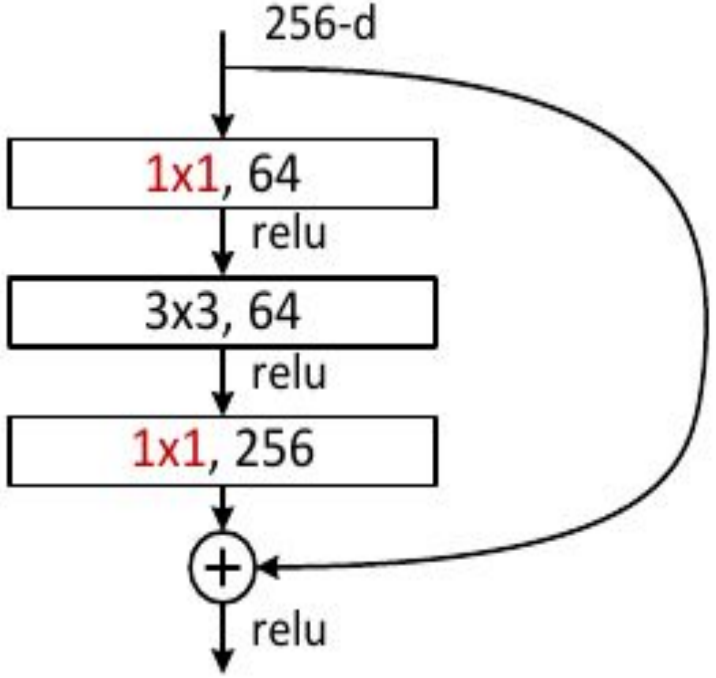
In the basic design, F(x) contains two 3×3 convolution layers along with a batch normalization and/or a rectied linear unit activation function.



arXiv:1512.03385

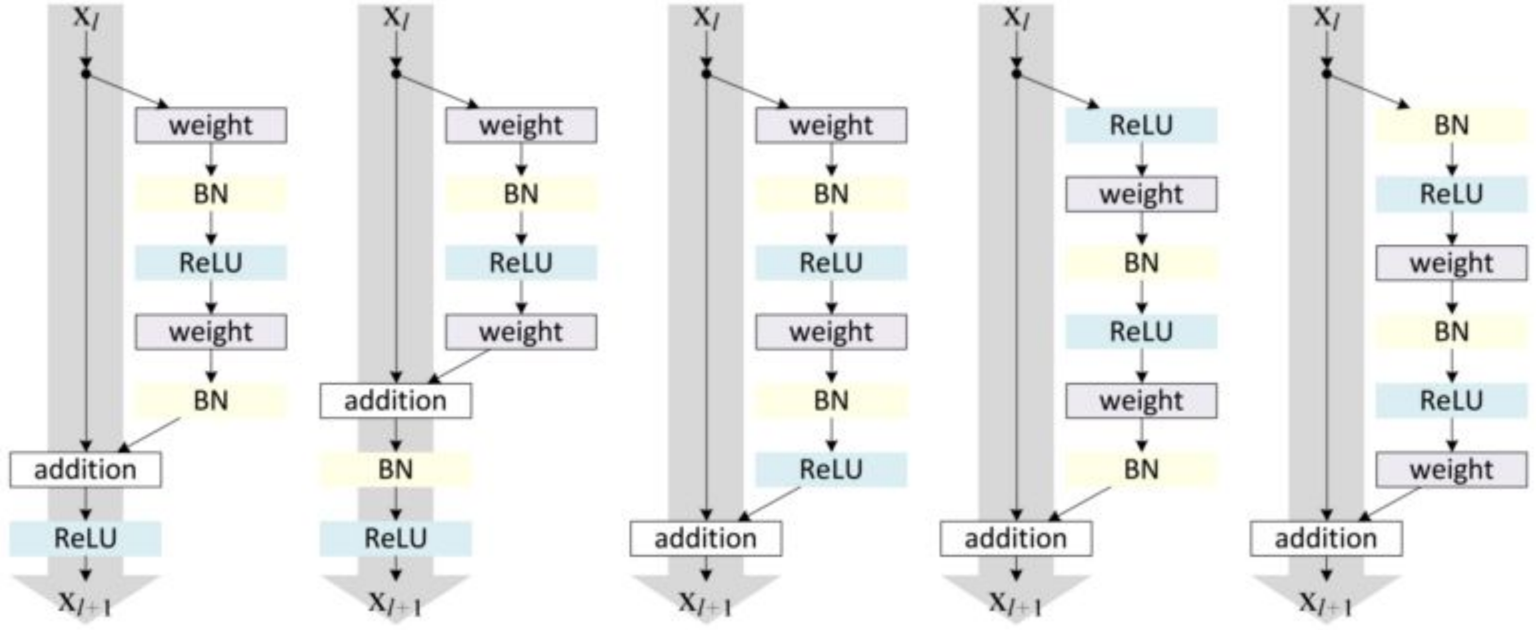
Blocks [ResNet, bottleneck]

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



arXiv:1512.03385

Blocks [ResNet, bottleneck]



(a) original

(b) BN after addition

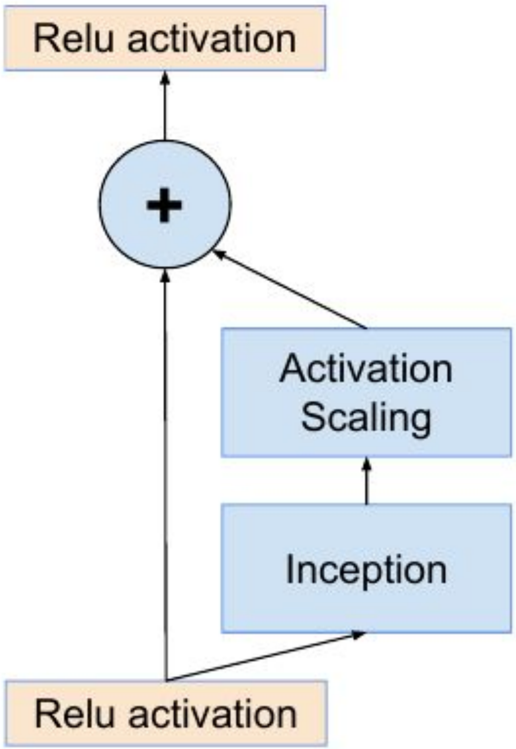
(c) ReLU before addition

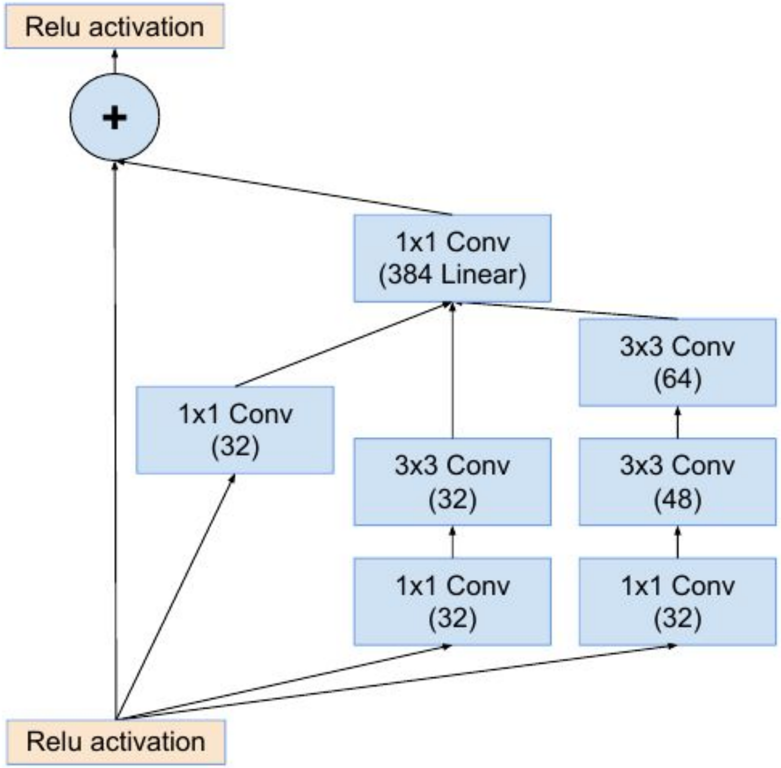
(d) ReLU-only pre-activation

(e) **full pre-activation**

arXiv:1512.03385

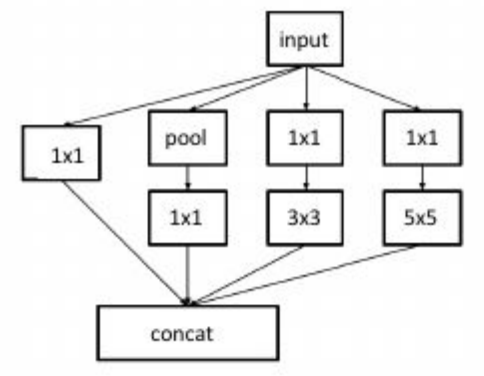
Blocks [Inception-ResNet]



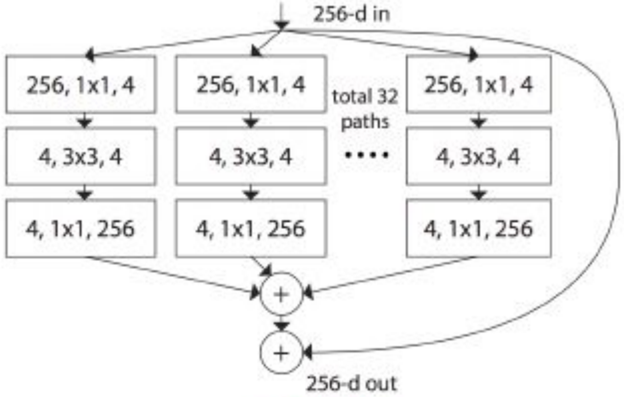


arXiv:1602.07261

Blocks [ResNeXt]



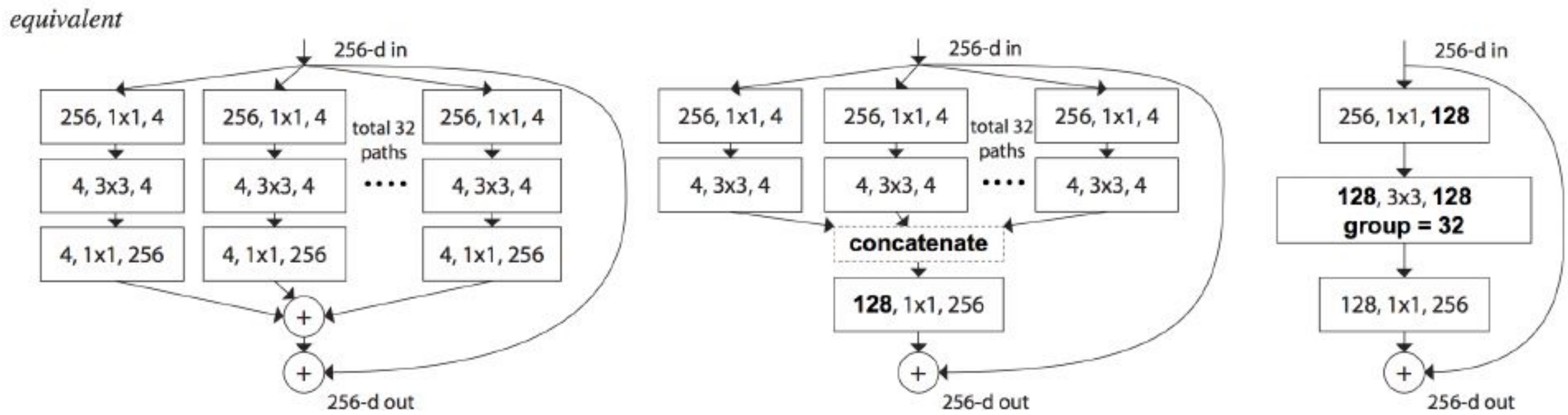
Inception:
heterogeneous multi-branch



ResNeXt:
uniform multi-branch

arXiv:1611.05431

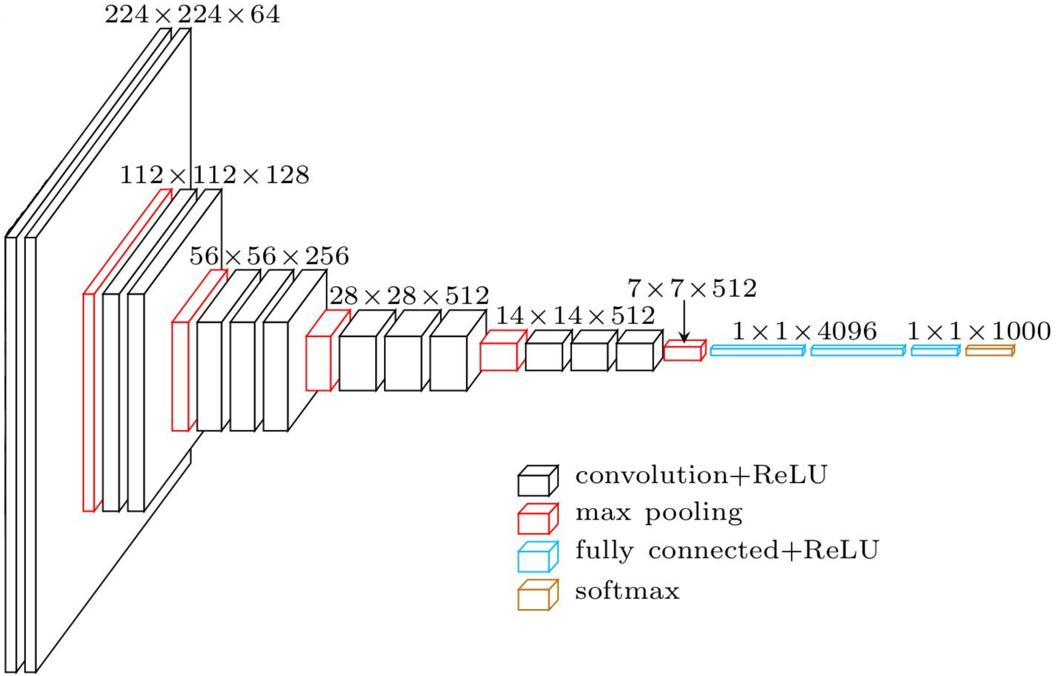
Blocks [ResNeXt]



arXiv:1611.05431

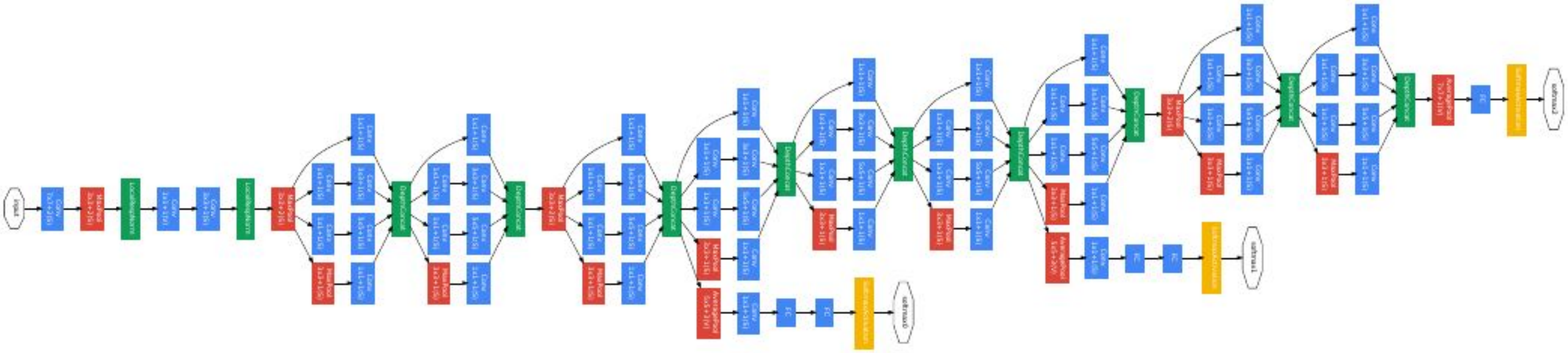
Contents

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]



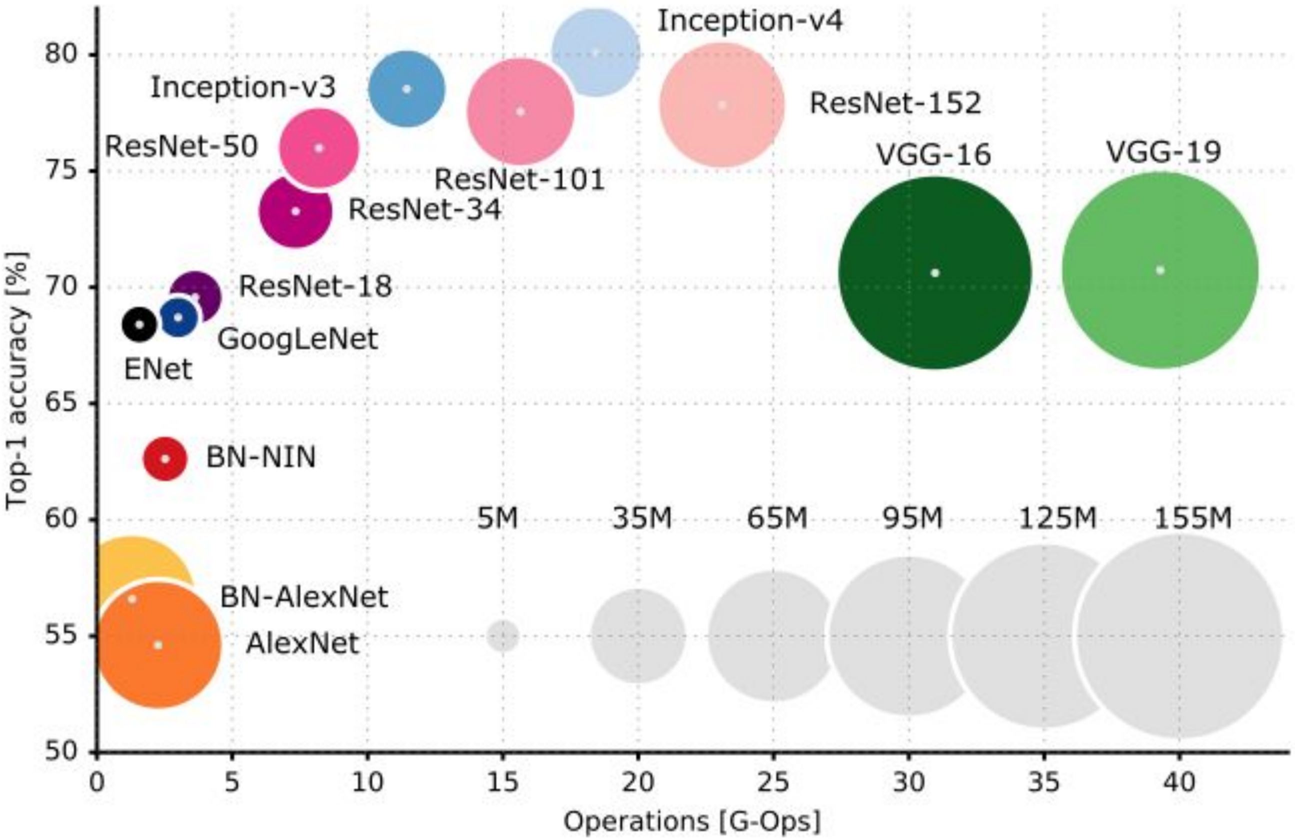
Net [ResNet & ResNetX]

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 $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times 3$
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		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$

arXiv:1611.05431

Net [Comparison]

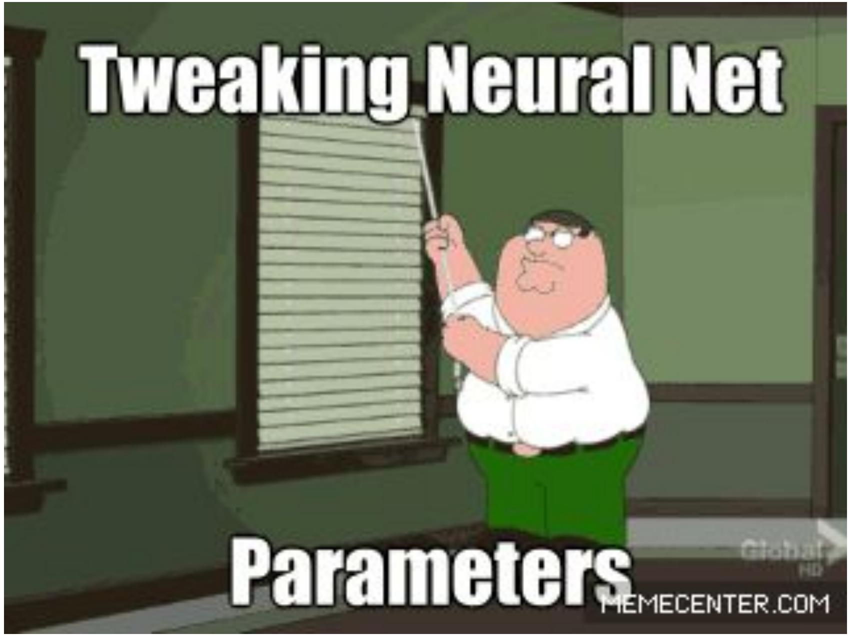


Update is needed !

arXiv:1605.07678

Contents

AutoML



AutoML

Recently, there has been some research in complexity issue by automating the architecture discovery process. We can consider these methods as falling into one of two categories.

- The first set of methods focus on discovering the entire architecture from primary building blocks i.e., convolution layers, pooling layers, fully connected layers etc.
- The other set of methods focus on building these architectures from the afore-mentioned more complex blocks involving branching and skip connections. The goal with this second set of methods is nding one particular building block which is then repeated many times to create the deep architecture.

AutoML

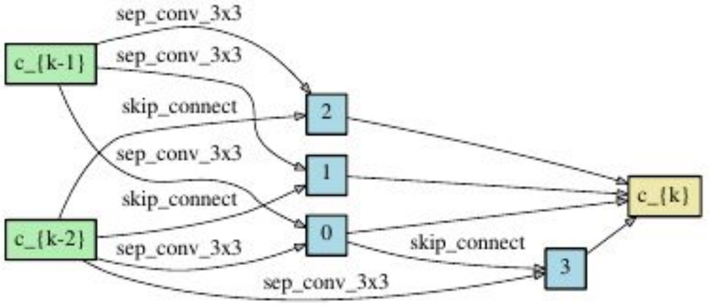


Figure 4: Normal cell learned on CIFAR-10.

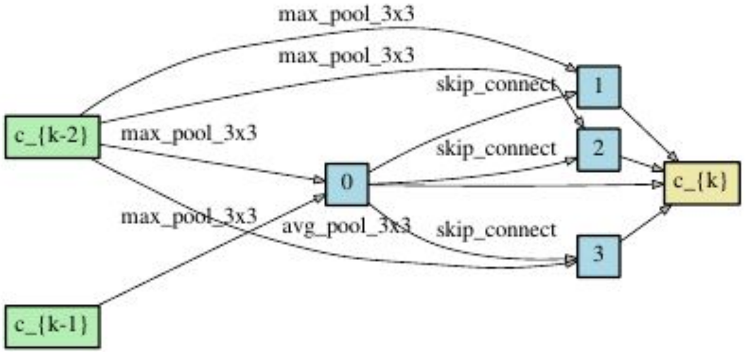


Figure 5: Reduction cell learned on CIFAR-10.

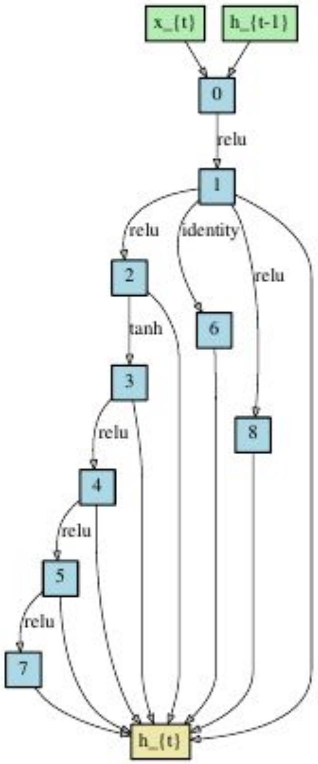


Figure 6: Recurrent cell learned on PTB.

arXiv:1806.09055

AutoML

Table 1: Comparison with state-of-the-art image classifiers on CIFAR-10. Results marked with † were obtained by training the corresponding architectures using our setup.

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	Search Method
DenseNet-BC (Huang et al., 2017)	3.46	25.6	–	manual
NASNet-A + cutout (Zoph et al., 2017)	2.65	3.3	1800	RL
NASNet-A + cutout (Zoph et al., 2017)†	2.83	3.1	3150	RL
AmoebaNet-A + cutout (Real et al., 2018)	3.34 ± 0.06	3.2	3150	evolution
AmoebaNet-A + cutout (Real et al., 2018)†	3.12	3.1	3150	evolution
AmoebaNet-B + cutout (Real et al., 2018)	2.55 ± 0.05	2.8	3150	evolution
Hierarchical Evo (Liu et al., 2017b)	3.75 ± 0.12	15.7	300	evolution
PNAS (Liu et al., 2017a)	3.41 ± 0.09	3.2	225	SMBO
ENAS + cutout (Pham et al., 2018b)	2.89	4.6	0.5	RL
Random + cutout	3.49	3.1	–	–
DARTS (first order) + cutout	2.94	2.9	1.5	gradient-based
DARTS (second order) + cutout	2.83 ± 0.06	3.4	4	gradient-based

arXiv:1806.09055

Contents

Summary

- When you see huge DN, don’t be scare. Usually it can be decomposed.
- Automatic topology learning (AutoML)
- Classification problem is solved, but features matter