Deep Learning for Computer Vision

Andrii Liubonko

Grammarly*

* The opinions expressed in this presentation and on the following slides are solely those of the presenter and not necessarily those of Grammarly

Logistics

4 units

2 types of homework:

- notebooks [30 % of FINAL SCORE]
- paper review [20 % of FINAL SCORE]
- mini-project [50 % of FINAL SCORE]

important dates (preliminary):

15 January, 23:59 : SOFT

29 January, 23:59 : HARD, PENALTY 30%

course repo:

https://github.com/lyubonko/ucu2022cv

Overview of the course

Unit I

[T] Intro
[P] pytorch

Unit II

[T] CNNs in depth, Object Detection
[P] simple nets

Unit III

[T] Attention in CV, Transformers
[P] classification

Unit IV

[T] Generative models, Diffusion Models
[P] project structure, stable diffusion

Overview of the course

Unit I

[T] Intro
[P] pytorch

Unit II

[T] CNNs in depth, Object Detection
[P] simple nets

Unit III

[T] Attention in CV, Transformers
[P] classification

Unit IV

[T] Generative models, Diffusion Models
[P] project structure, stable diffusion

Content of today lecture

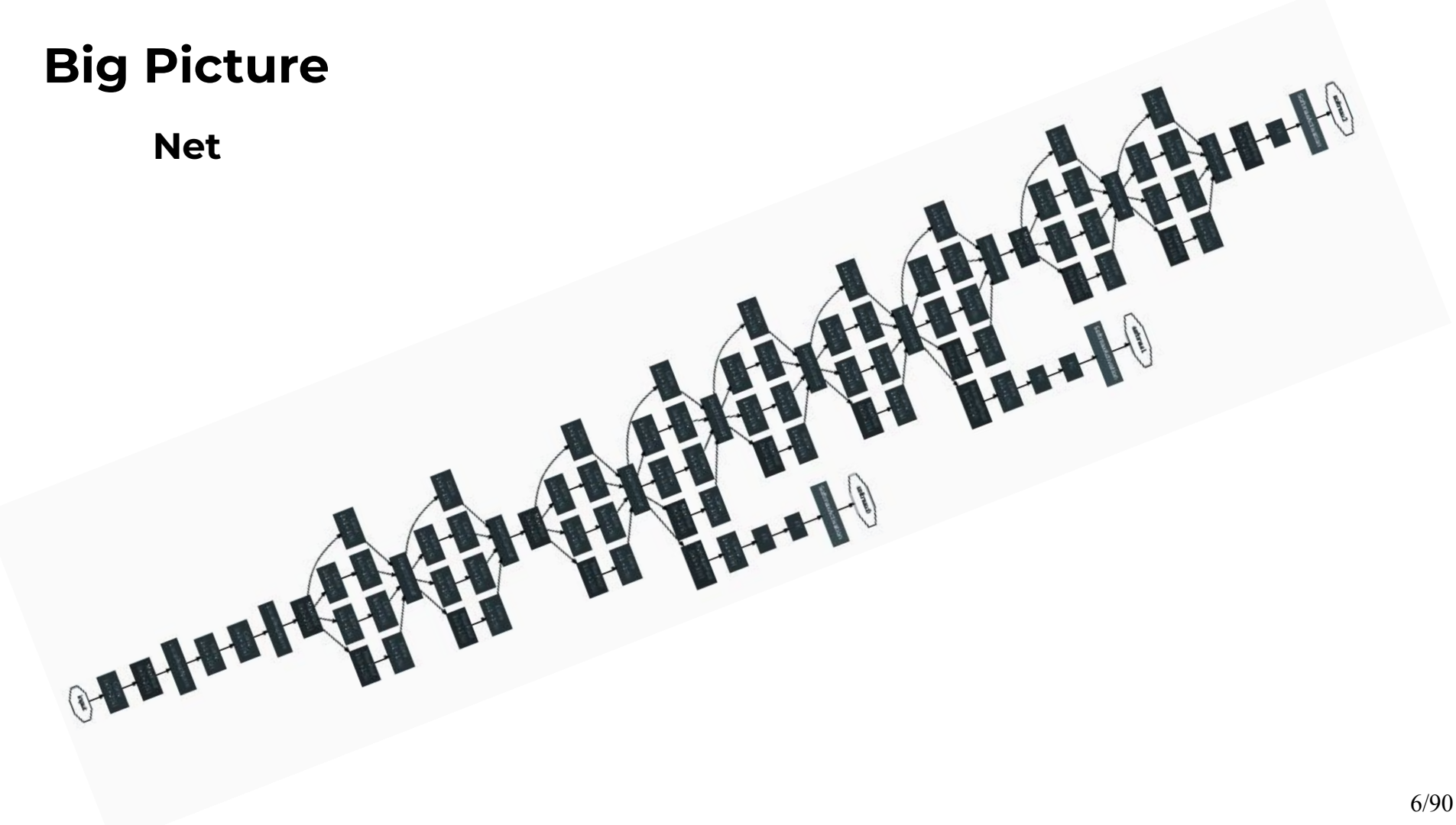
- **Layers**
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Group Convolutions and its variants]
  - Layers [Upsampling, Learnable Upsampling]
  - Layers [Normalization, Batch Norm, Dropout]

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

- Architectures
  [historical]: AlexNet, VGG, Inception,
  [modern]: ResNet, MobileNet, EfficientNet

Big Picture

Net



Big Picture

Net

↑

Blocks

Big Picture

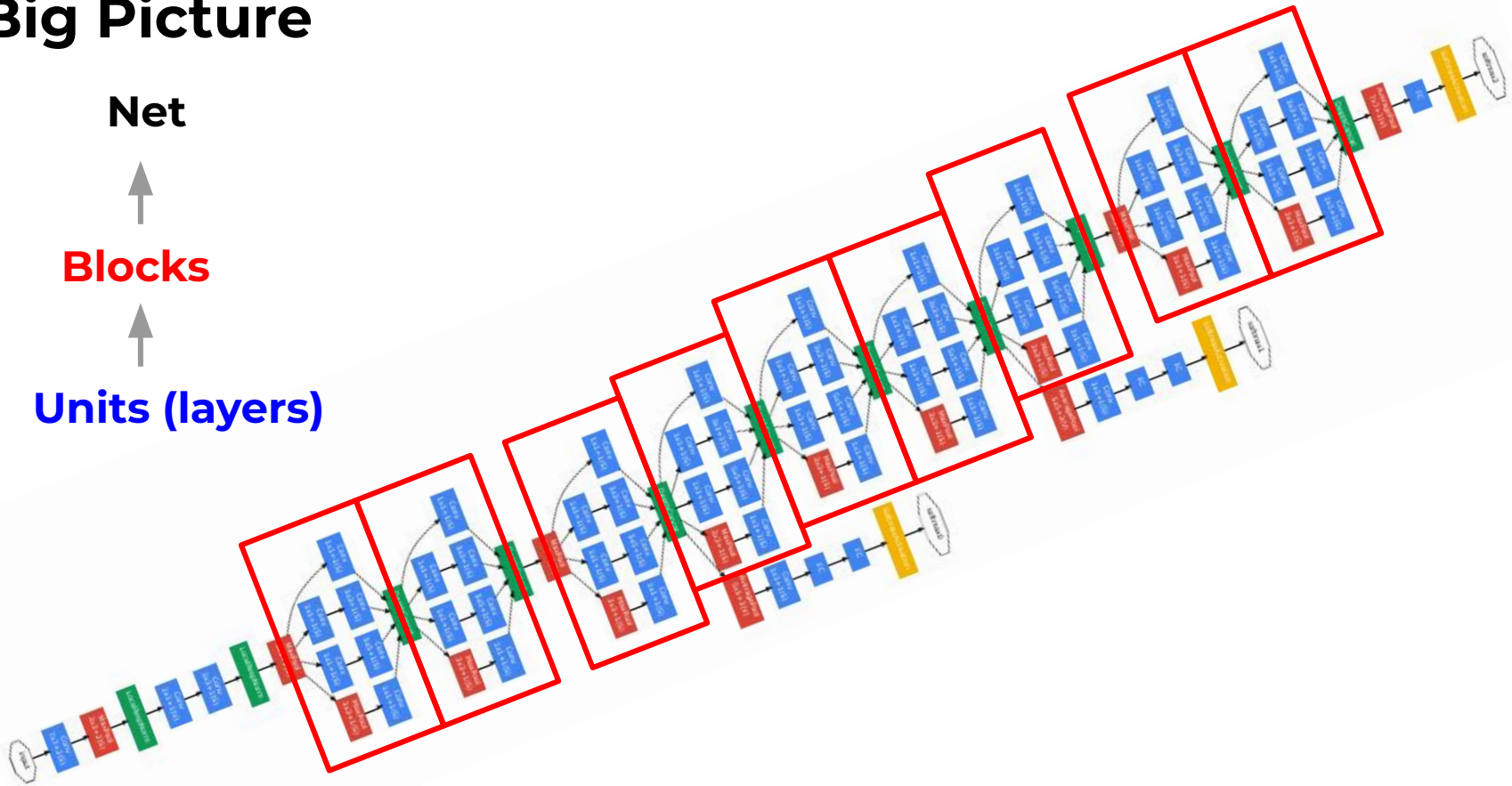
Net

↑

Blocks

↑

Units (layers)



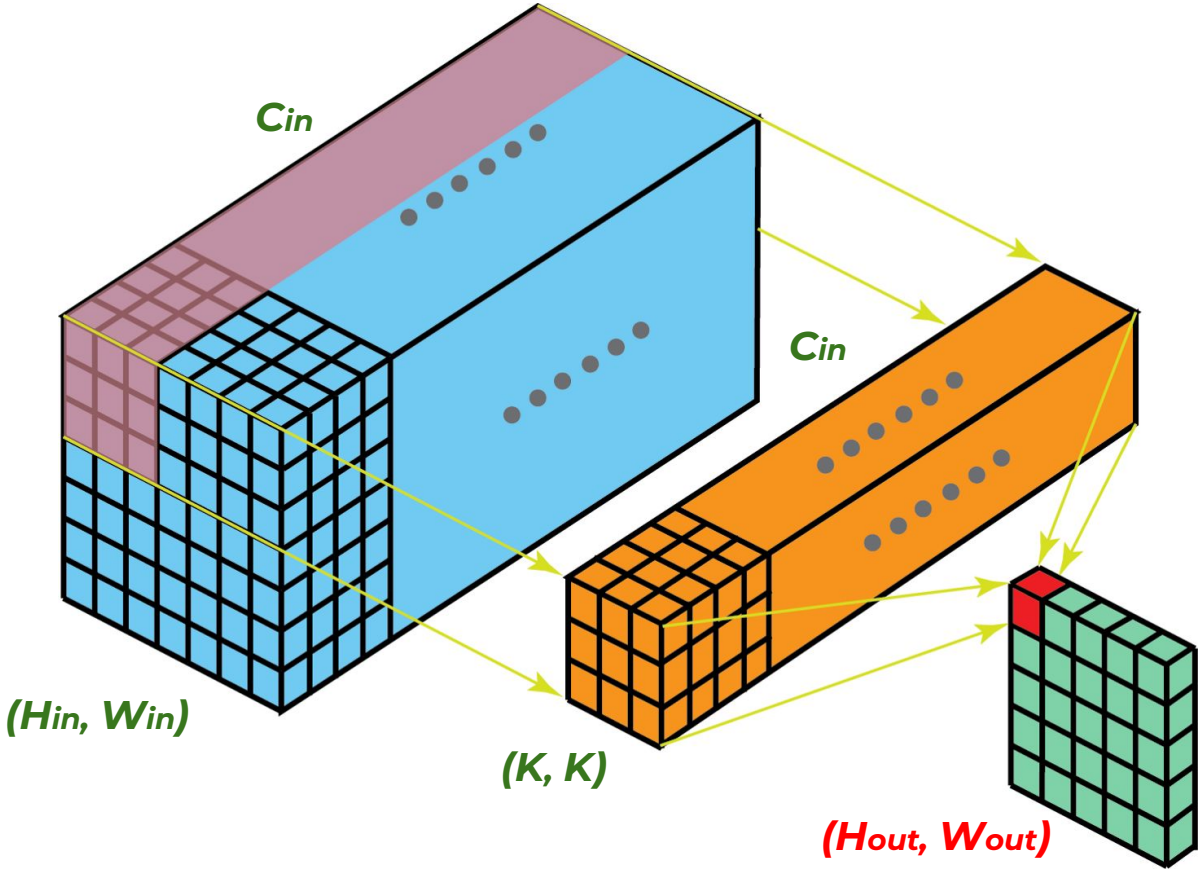
Content of today lecture

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

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

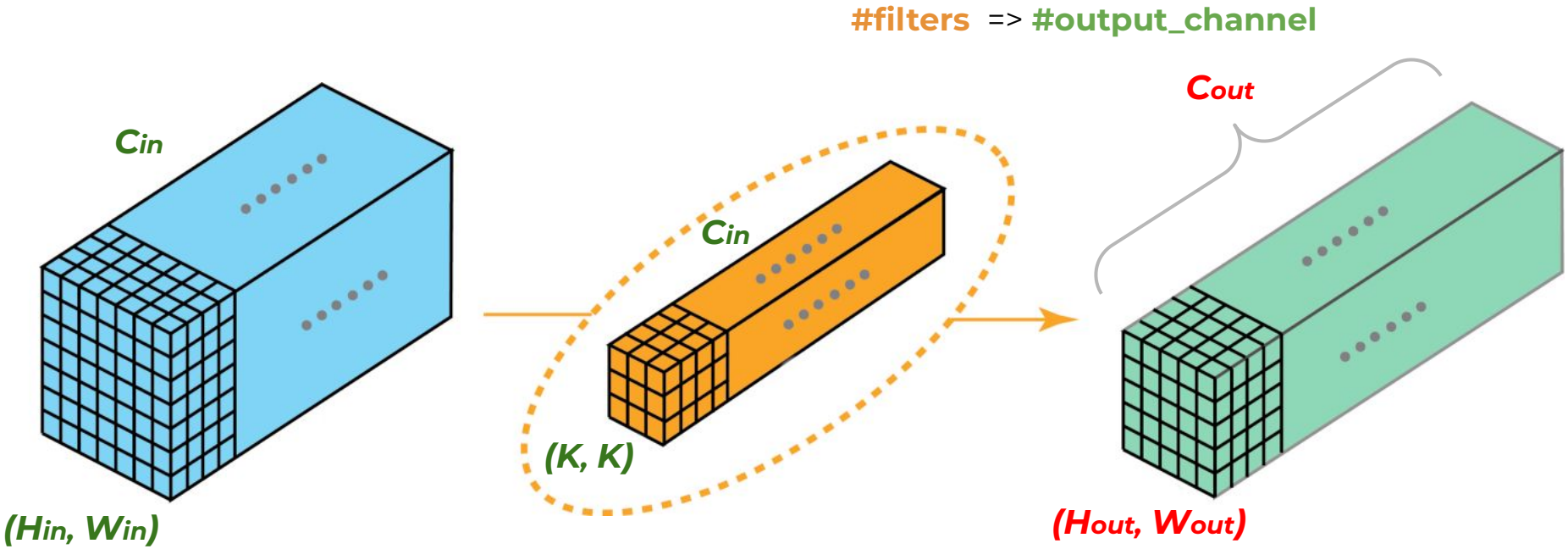
- Architectures
  [historical]: AlexNet, VGG, Inception,
  [modern]: ResNet, MobileNet, EfficientNet

Layers [Convolution]



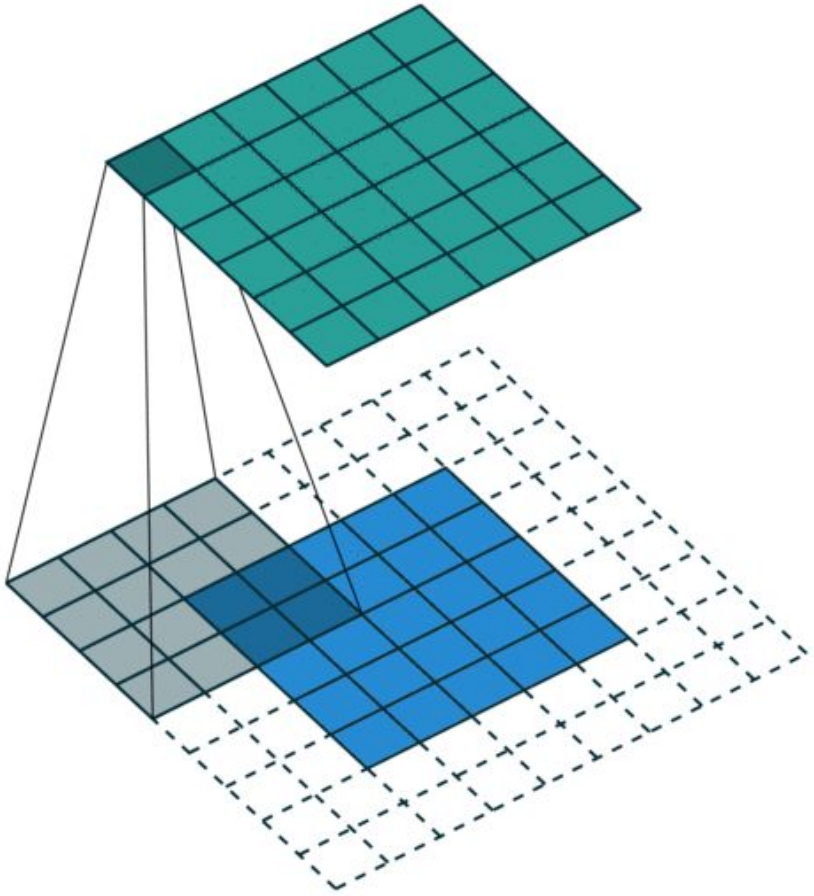
spatial extent is controlled by **stride**, **kernel_size**, **padding**

Layers [Convolution]



Layers [Convolution]

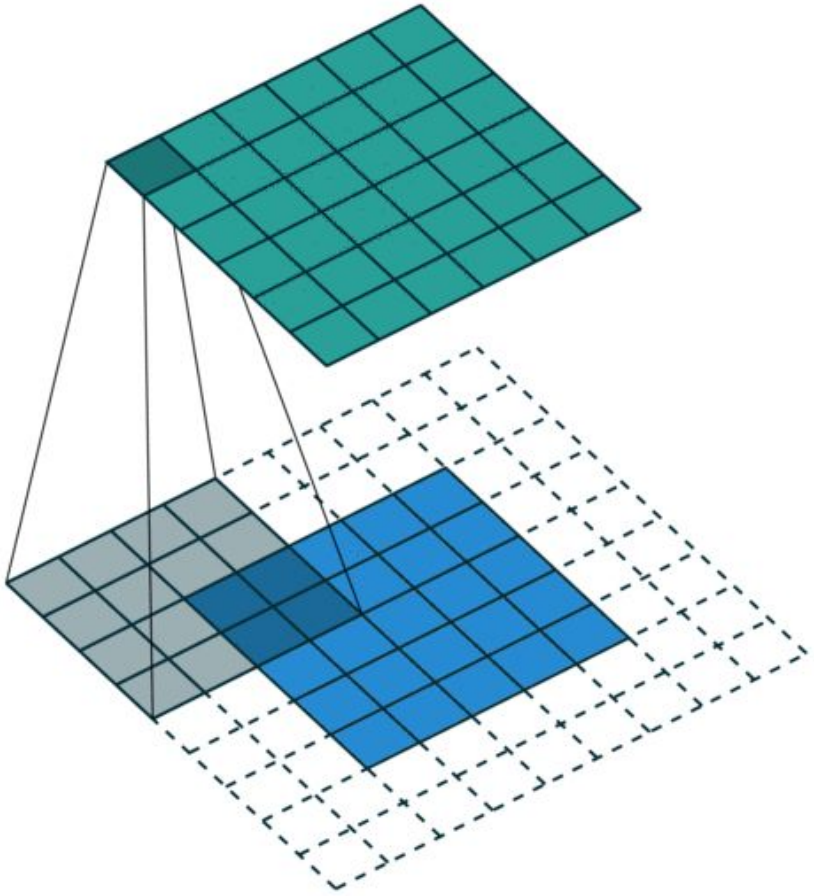
- kernel size F ?
- padding size P ?
- stride S ?



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

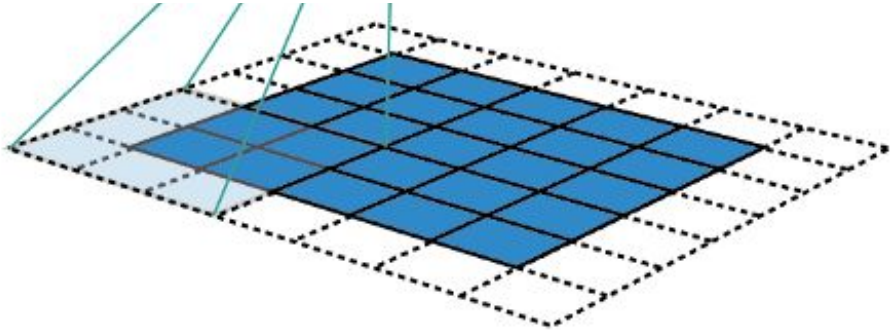
Layers [Convolution]

- 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

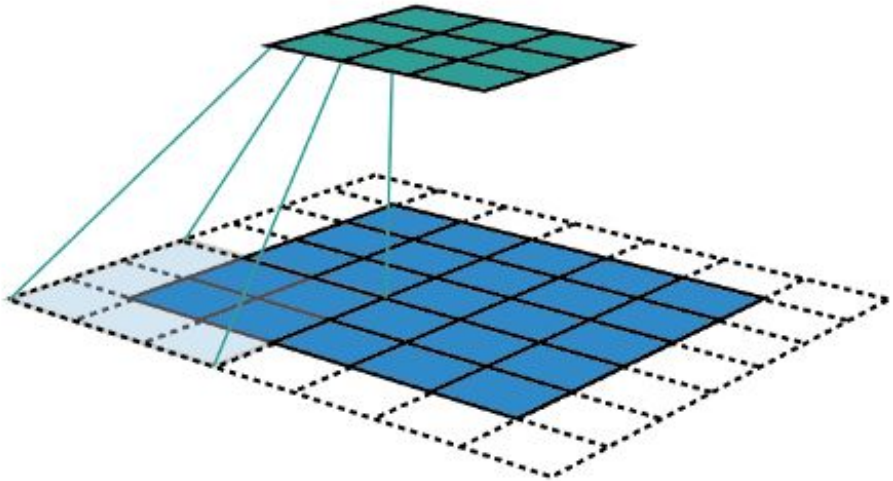
Layers [Convolution]



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

output size ?

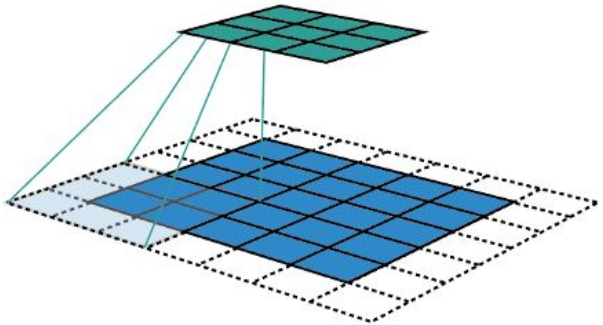
Layers [Convolution]



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

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

Layers [Convolution][Receptive field]



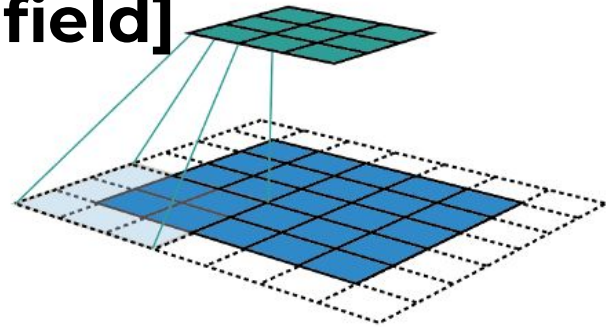
The **receptive field** is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by)

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

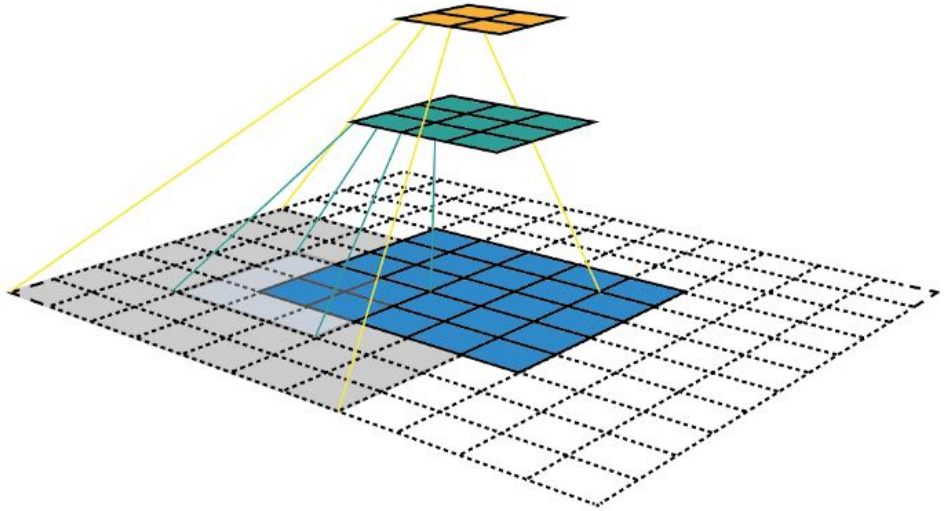
A receptive field of a feature can be described by

- its center location
- its size

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

Content of today lecture

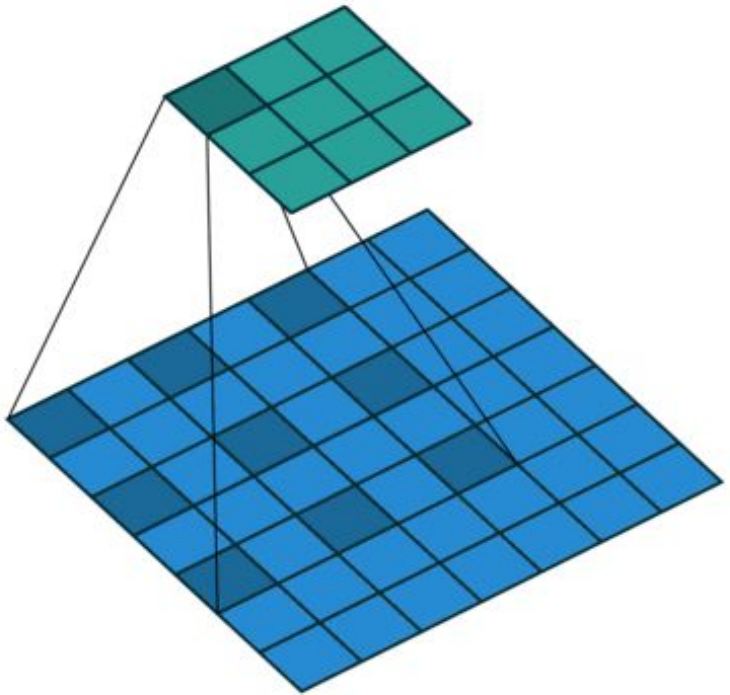
- Layers
  - Layers [Convolution] [Receptive field]
  - **Layers [Dilated Convolution, Deformable Convolution]**
  - Layers [Group Convolutions and its variants]
  - Layers [Upsampling, Learnable Upsampling]
  - Layers [Batch Norm, Dropout]
- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*
  - EfficientNet
- Architectures
  VGG, Inception, ResNet*, MobileNet*, EfficientNet
- Neural Architecture Search (NAS)
- Summary

Layers [Dilated Convolution]

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**.

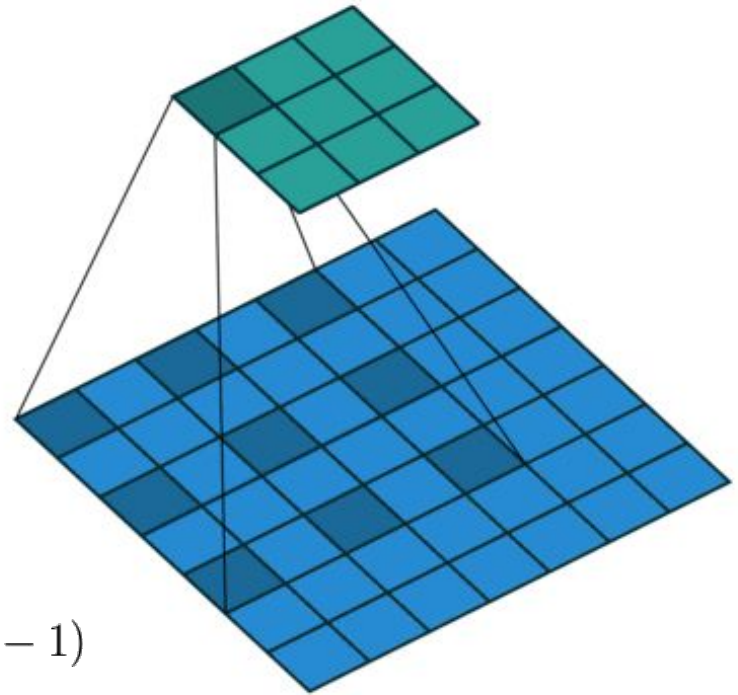
- input size I = 7x7
- kernel size F = 3x3,
- padding size P = 0,
- stride S = 1
- dilation rate **d** = 2



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

Layers [Dilated Convolution]

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



For ‘normal’ convolution we have:

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

How this will change ?

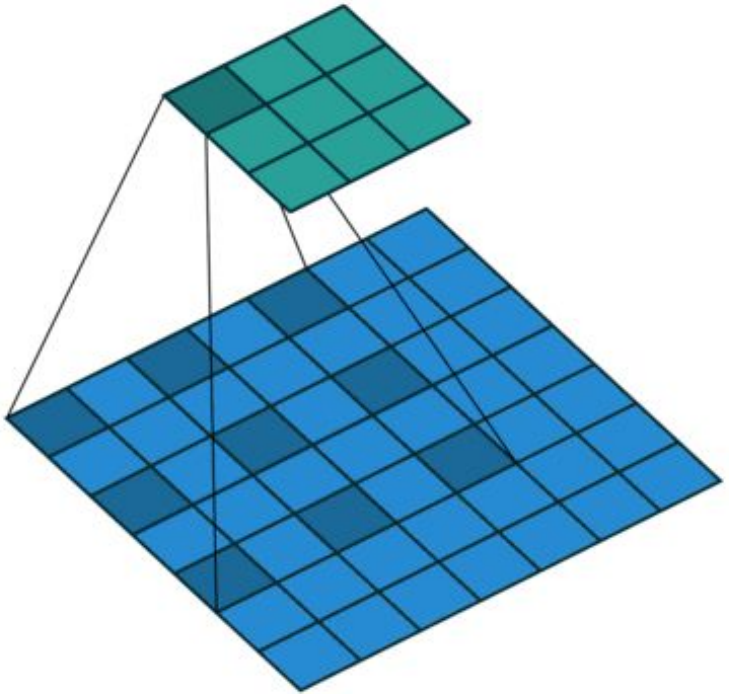
$$F' = F + (F - 1) \cdot (d - 1)$$

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

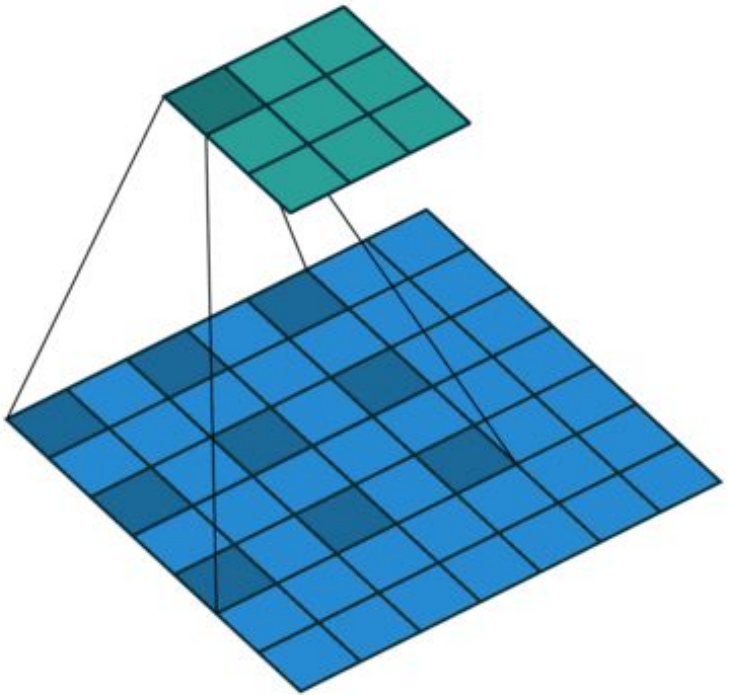


https://arxiv.org/abs/1511.07122
http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

Layers [Dilated Convolution]

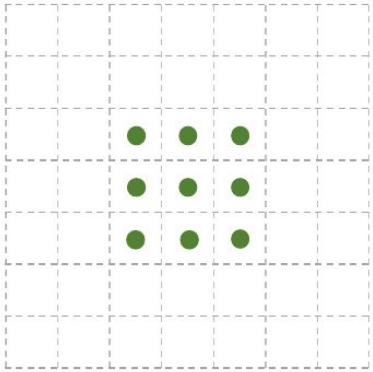
(used for *Semantic Segmentation*, *Object Recognition*)

- 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]



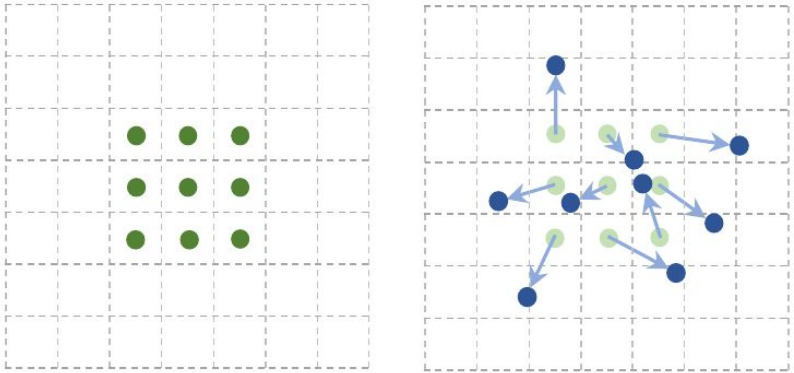
$$\mathcal{R} = \{(-1,-1),(-1,0),\ldots,(0,1),(1,1)\}$$

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

https://arxiv.org/abs/1703.06211

Layers [Deformable Convolution]

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



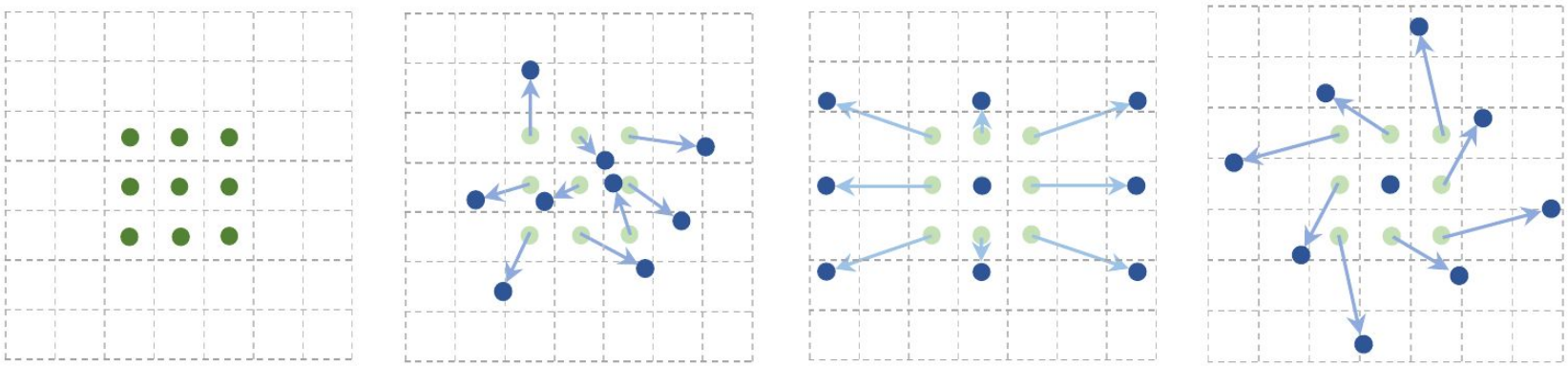
$$\mathcal{R} = \{(-1,-1),(-1,0),\ldots,(0,1),(1,1)\}$$

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

https://arxiv.org/abs/1703.06211

Layers [Deformable Convolution]

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

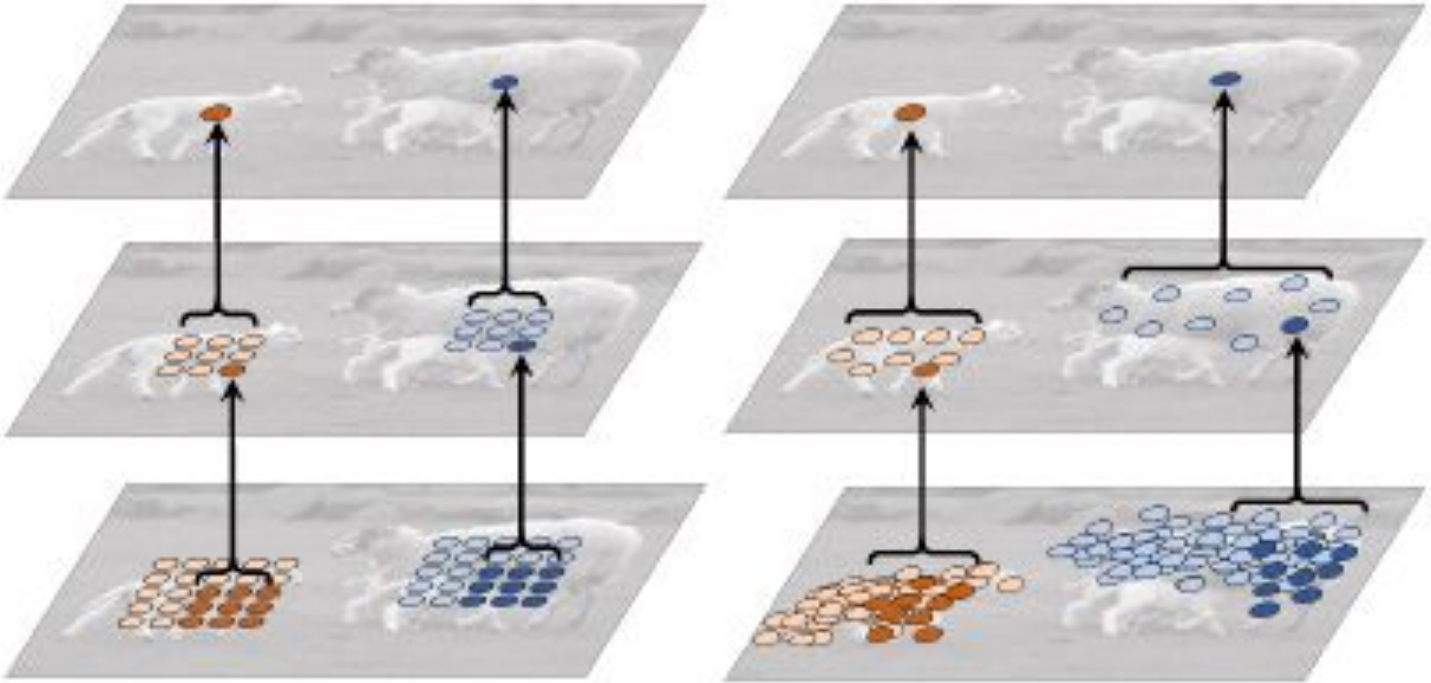


$$\mathcal{R} = \{(-1,-1),(-1,0),\ldots,(0,1),(1,1)\}$$

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

https://arxiv.org/abs/1703.06211

Layers [Deformable Convolution]

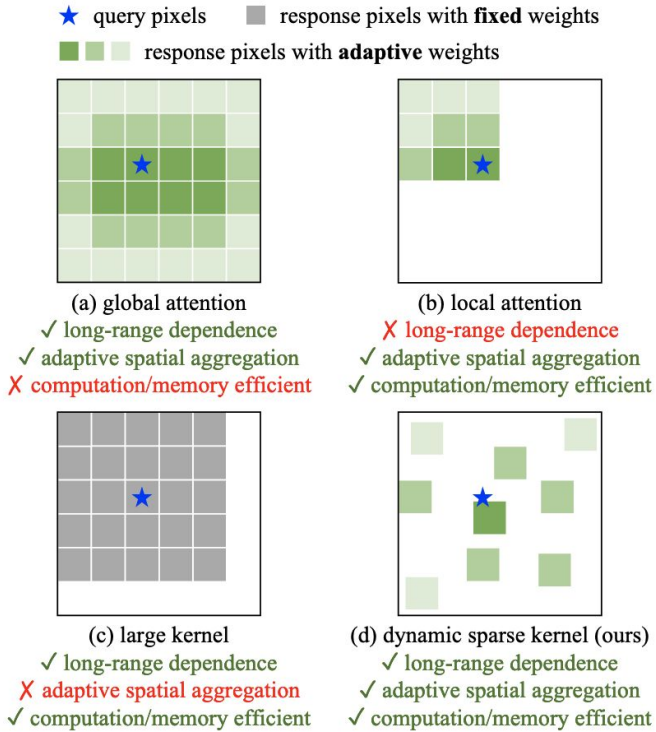


https://arxiv.org/abs/1703.06211

Layers [Deformable Convolution]

https://arxiv.org/abs/1703.06211

Layers [Deformable Convolution]



current SOTA at COCO

[2211.05778] InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions

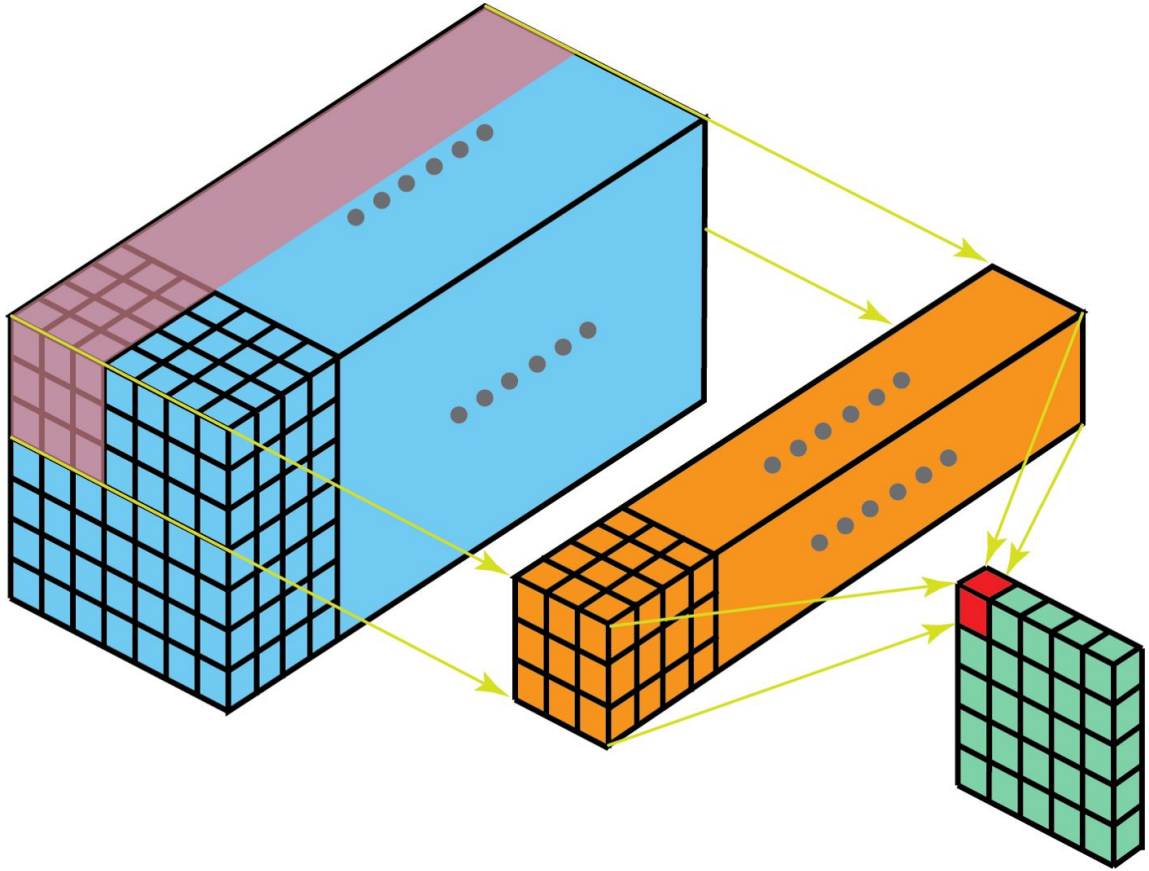
Content of today lecture

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

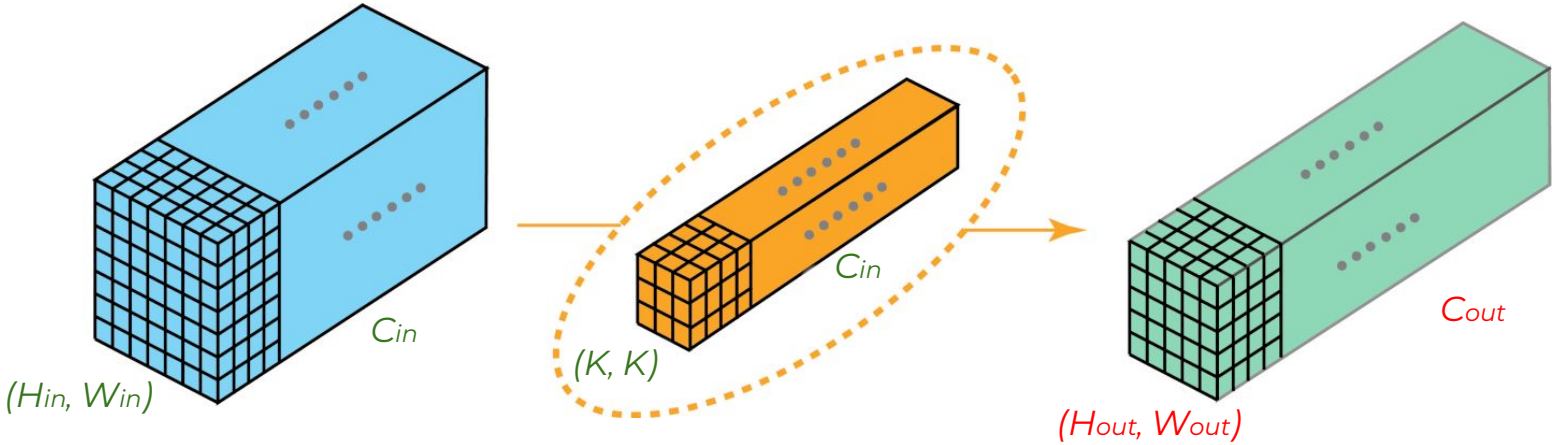
- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

- Architectures
  [historical]: AlexNet, VGG, Inception,
  [modern]: ResNet, MobileNet, EfficientNet

Layers [Convolution]



Layers [Convolution]



The number of parameters: of standard convolution is $C_{in}KK \cdot C_{out}$ when 'bias' is False and $C_{in}(KK+1) \cdot C_{out}$ otherwise.

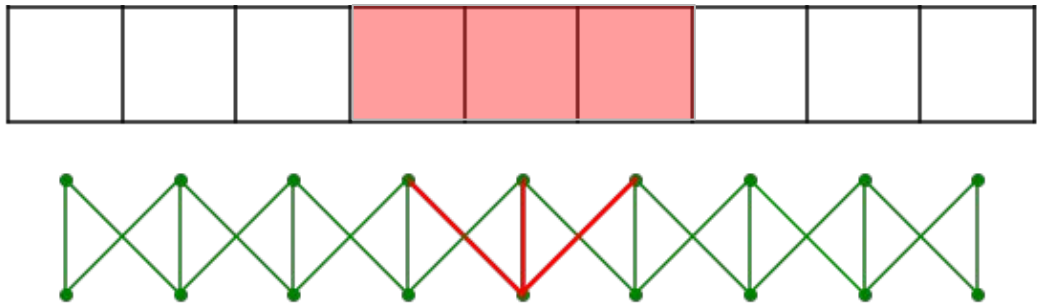
The computational cost: of standard convolution is $C_{in}KK \cdot H_{out}W_{out}C_{out}$

1. the spatial size of the output feature map $(H_{out}W_{out})$
2. the size of conv. kernel K^2
3. the number of input and output channels $C_{in} \cdot C_{out}$

Layers [Convolution]

Normal
Convolution:

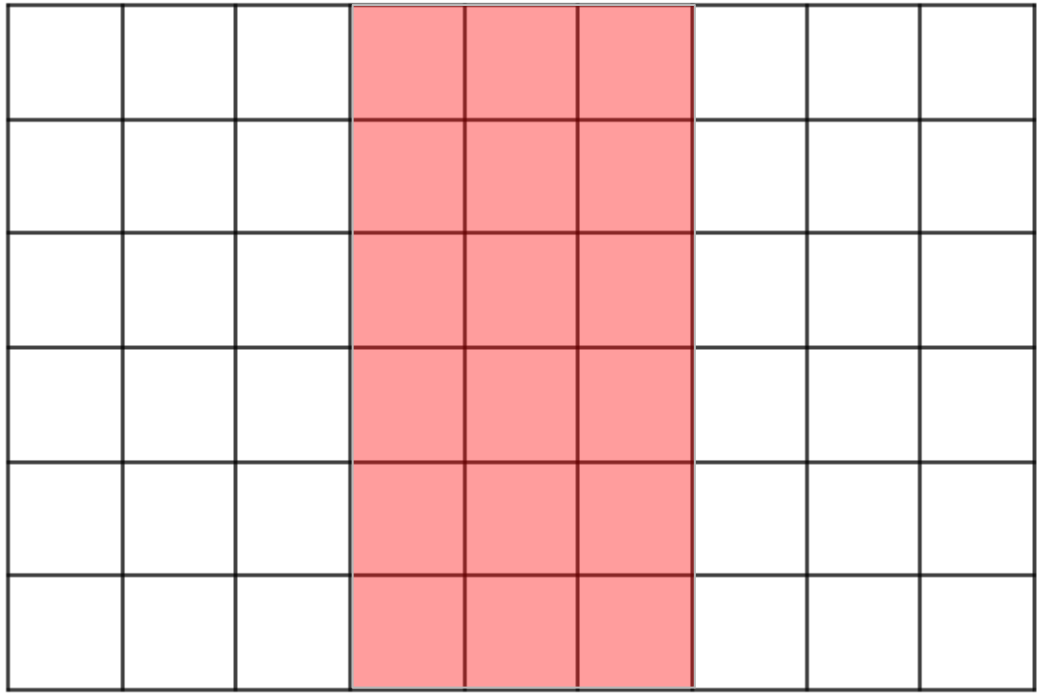
conv 3x3

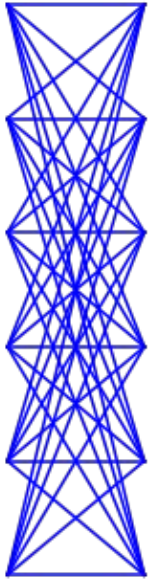


Layers [Convolution]

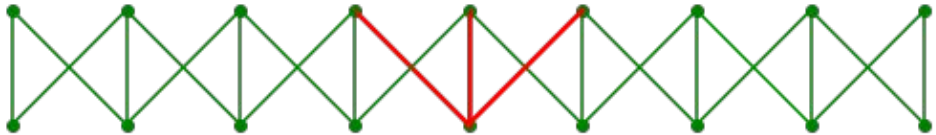
Normal Convolution:

conv 3x3



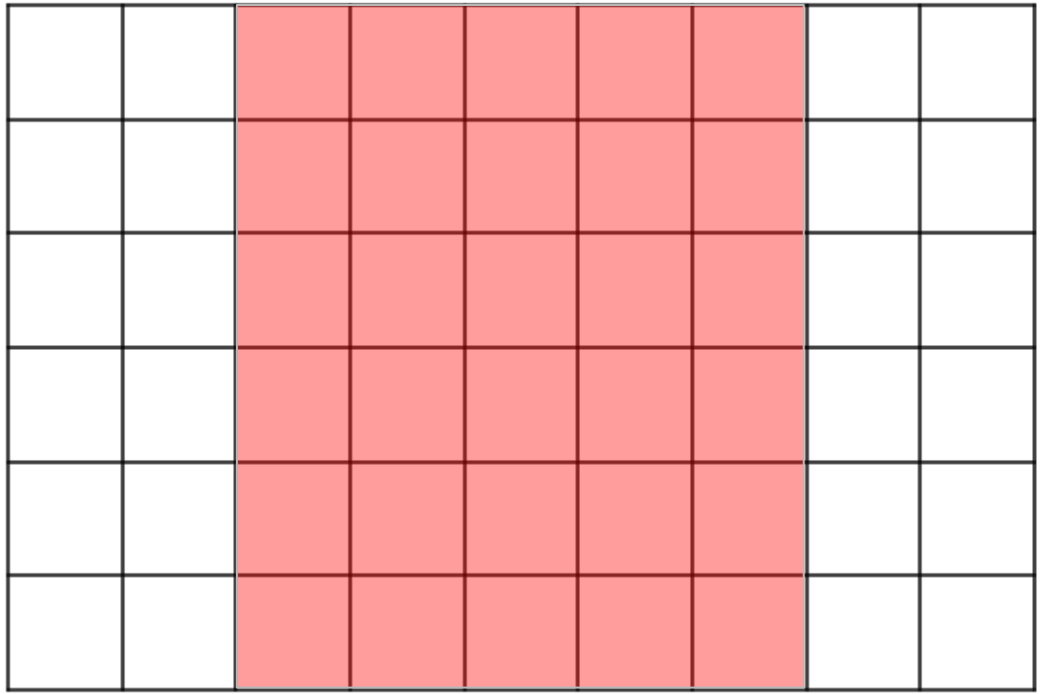


channels

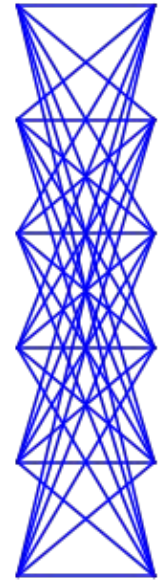


Layers [Convolution]

Normal Convolution:



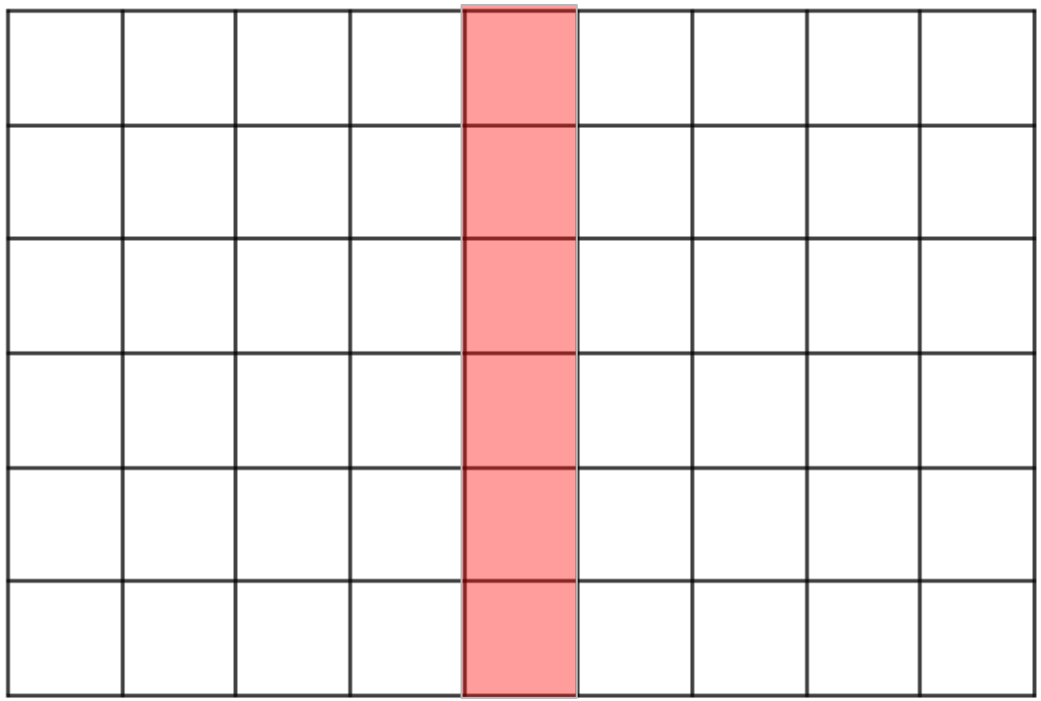
conv 5x5

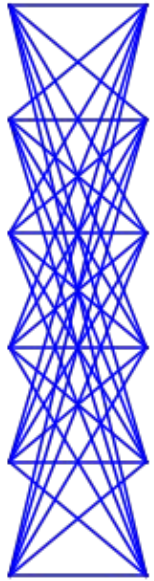


channels

Layers [1x1 Convolution, Pointwise Convolution]

conv 1x1

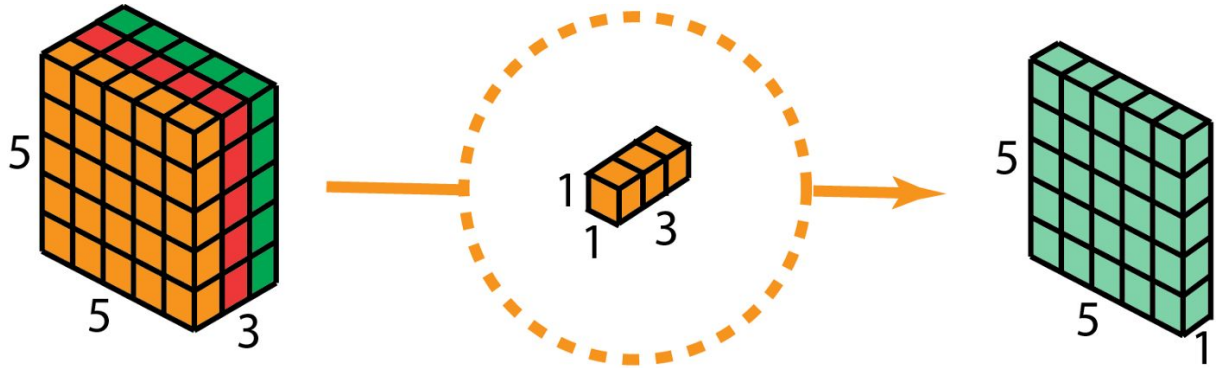




channels

A special name:
point-wise convolutions

Layers [1x1 Convolution, Pointwise Convolution]

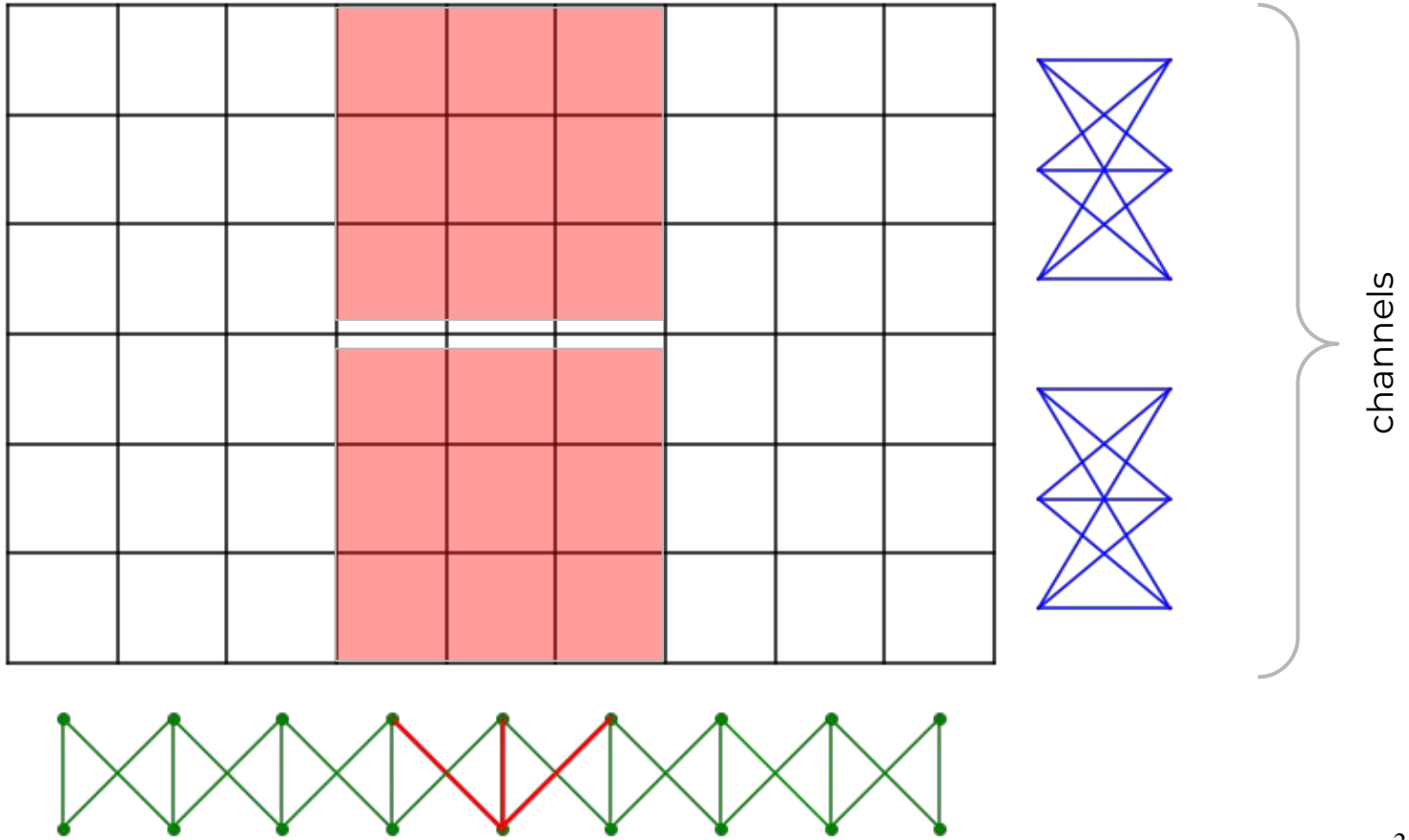


The number of parameters: $C_{in} \cdot C_{out}$ (no bias)

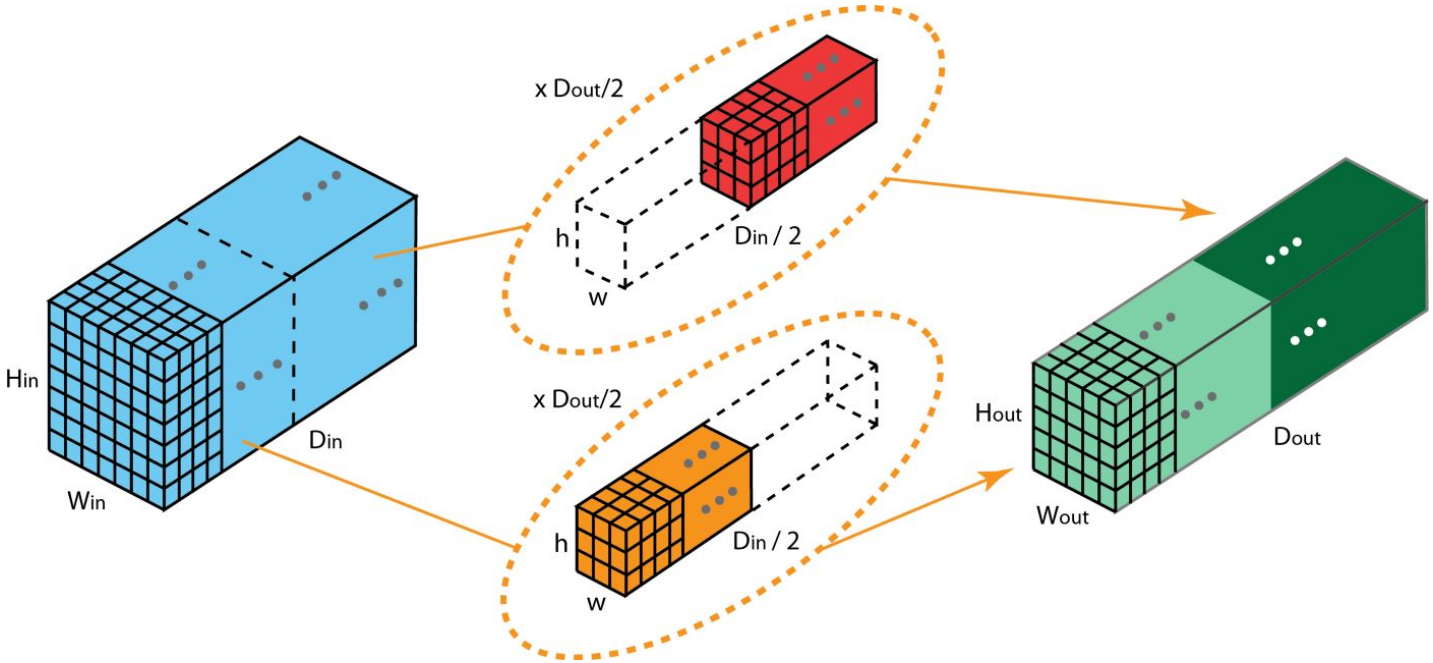
The computational cost: $C_{in} \cdot H_{out} W_{out} C_{out}$ (1/9 reduction in comparison to 3x3 kernel)

Layers [Group Convolution]

gconv 3x3 (g=2)



Layers [Group Convolution]

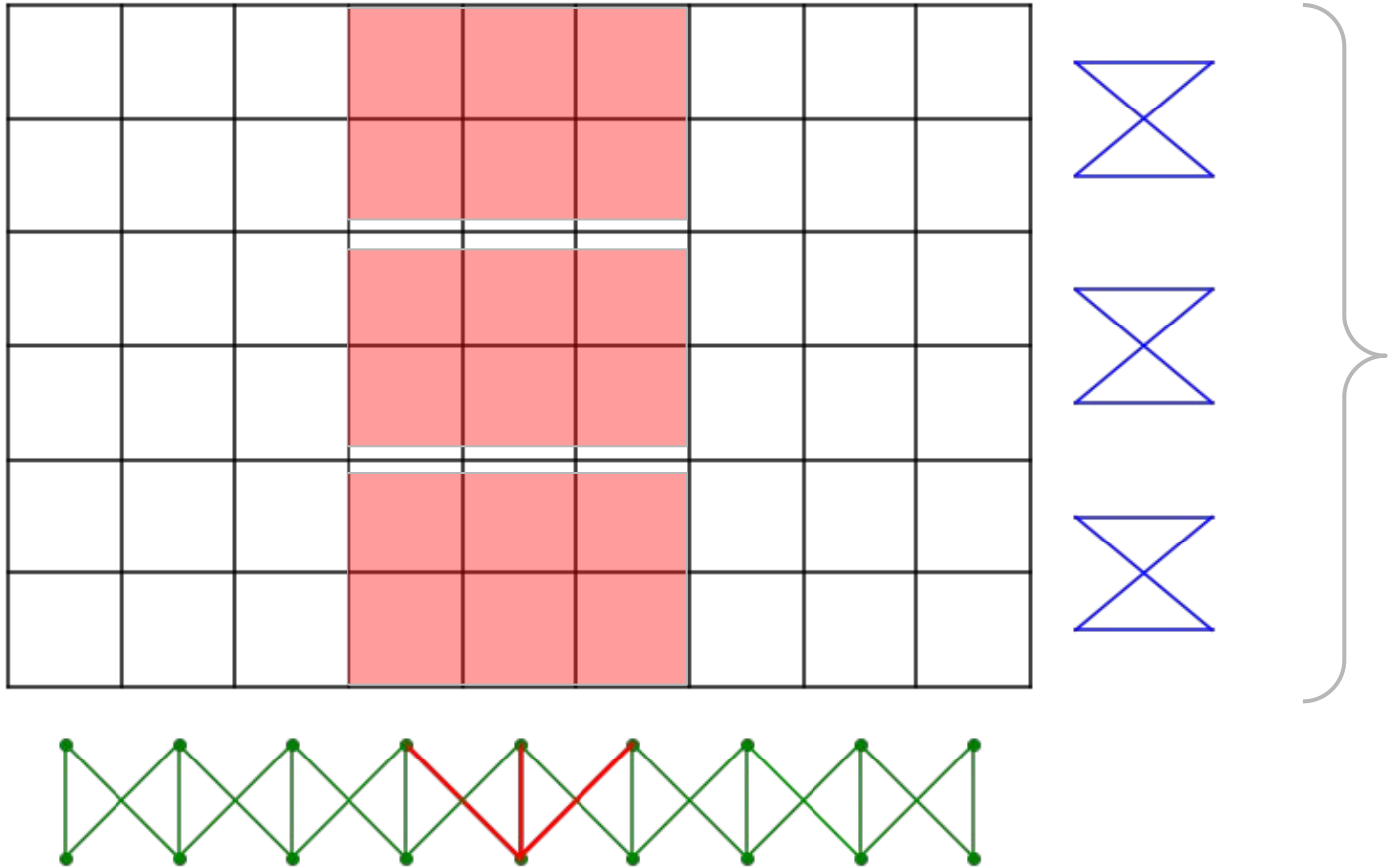


The number of parameters: $C_{in}/G\ KK \cdot C_{out}$ (no bias), the tensor (of kernel parameters) has the following sizes: $[C_{out}, C_{in}/G, K, K]$

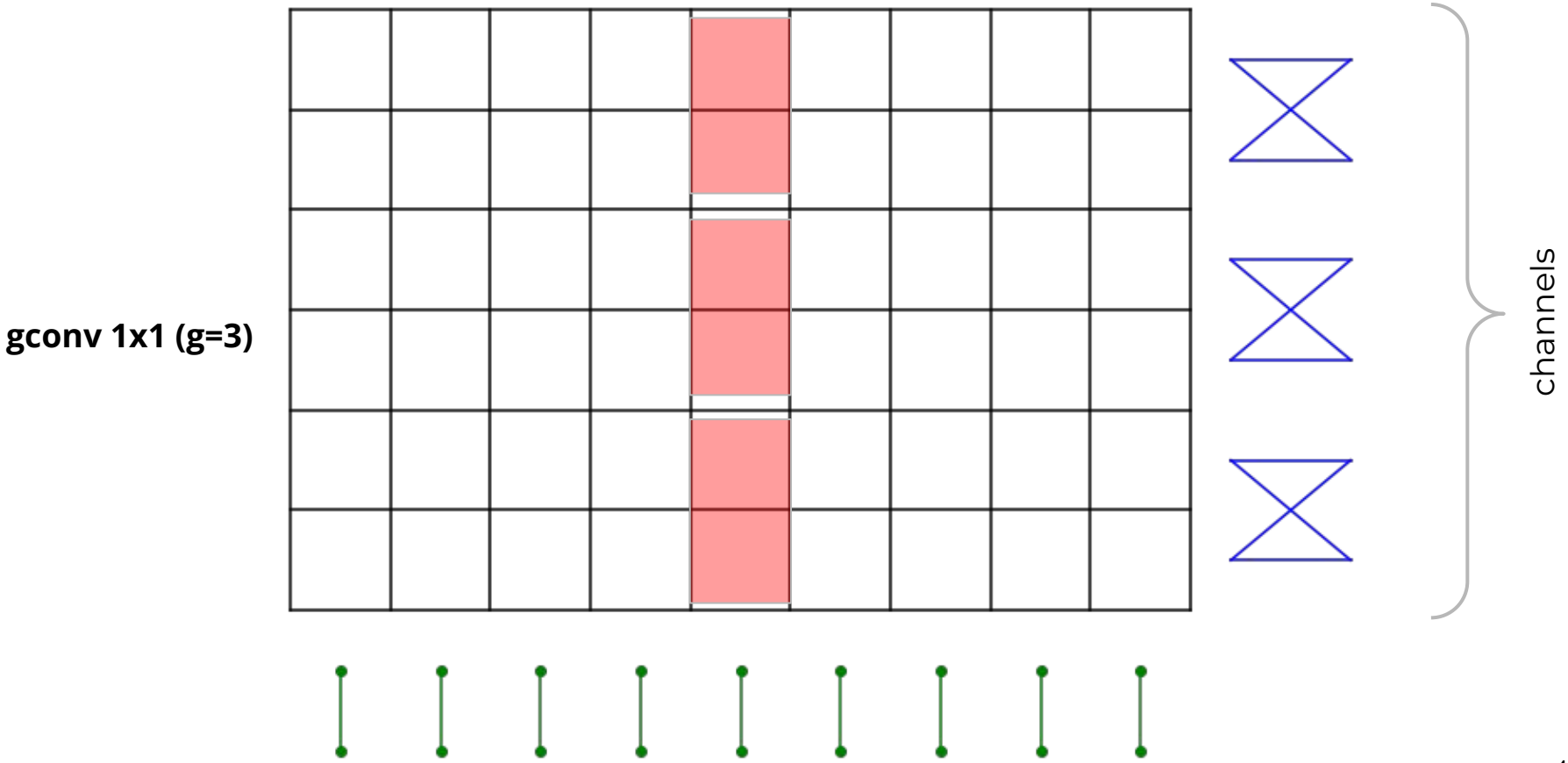
The computational cost: $C_{in}/G\ KK \cdot H_{out} W_{out} C_{out}$

Layers [Group Convolution]

gconv 3x3 (g=3)

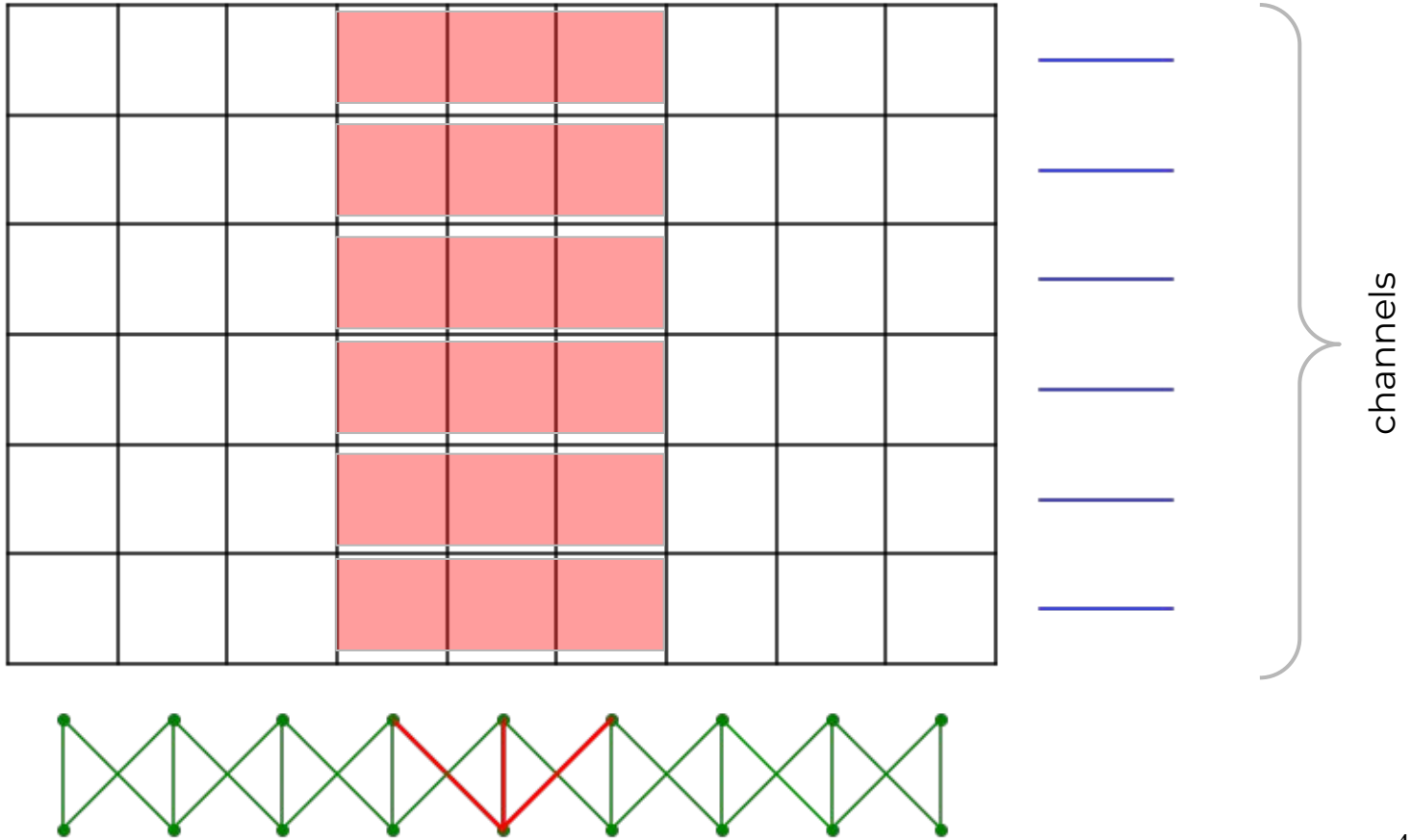


Layers [Group Convolution]



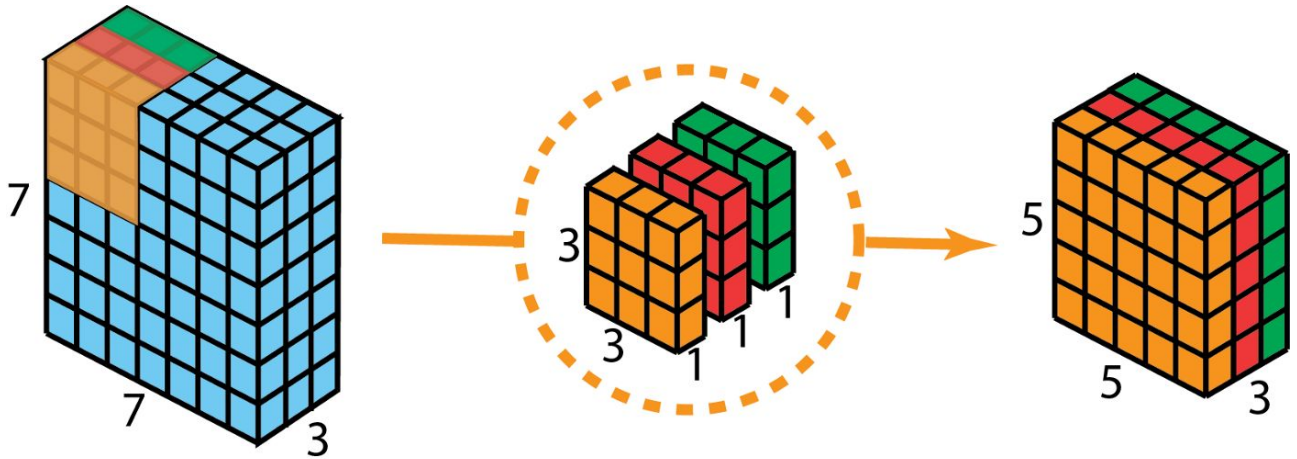
Layers [Group Convolution, depth-wise convolutions]

gconv 3x3 (g=6)



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

Layers [Group Convolution]



The number of parameters: $KK \cdot C_{out}$ (no bias), the tensor has the following sizes: $[C_{out}, 1, K, K]$

The computational cost: $KK \cdot H_{out} W_{out} C_{out}$

Content of today lecture

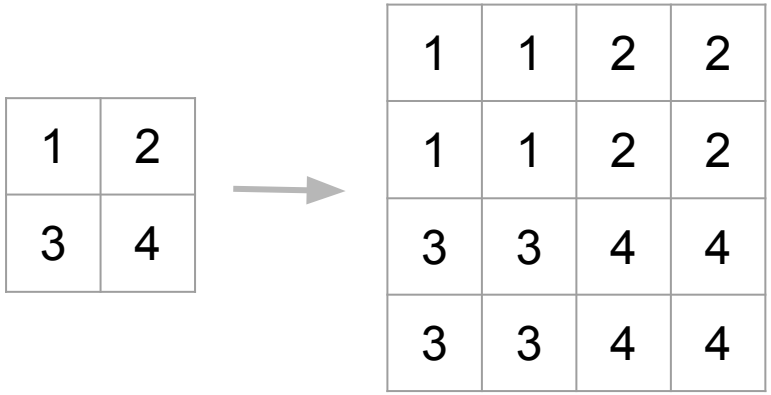
- **Layers**
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Group Convolutions and its variants]
  - **Layers [Upsampling, Learnable Upsampling]**
  - Layers [Normalization, Batch Norm, Dropout]

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

- Architectures
  [historical]: AlexNet, VGG, Inception,
  [modern]: ResNet, MobileNet, EfficientNet

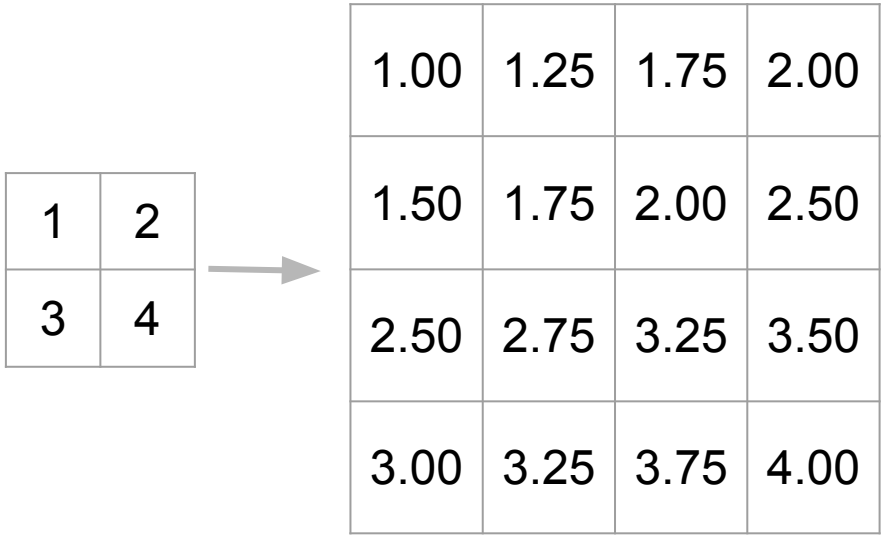
Layers [Upsampling]

- **Nearest Neighbor**



Input: [2 x 2] Output: [4 x 4]

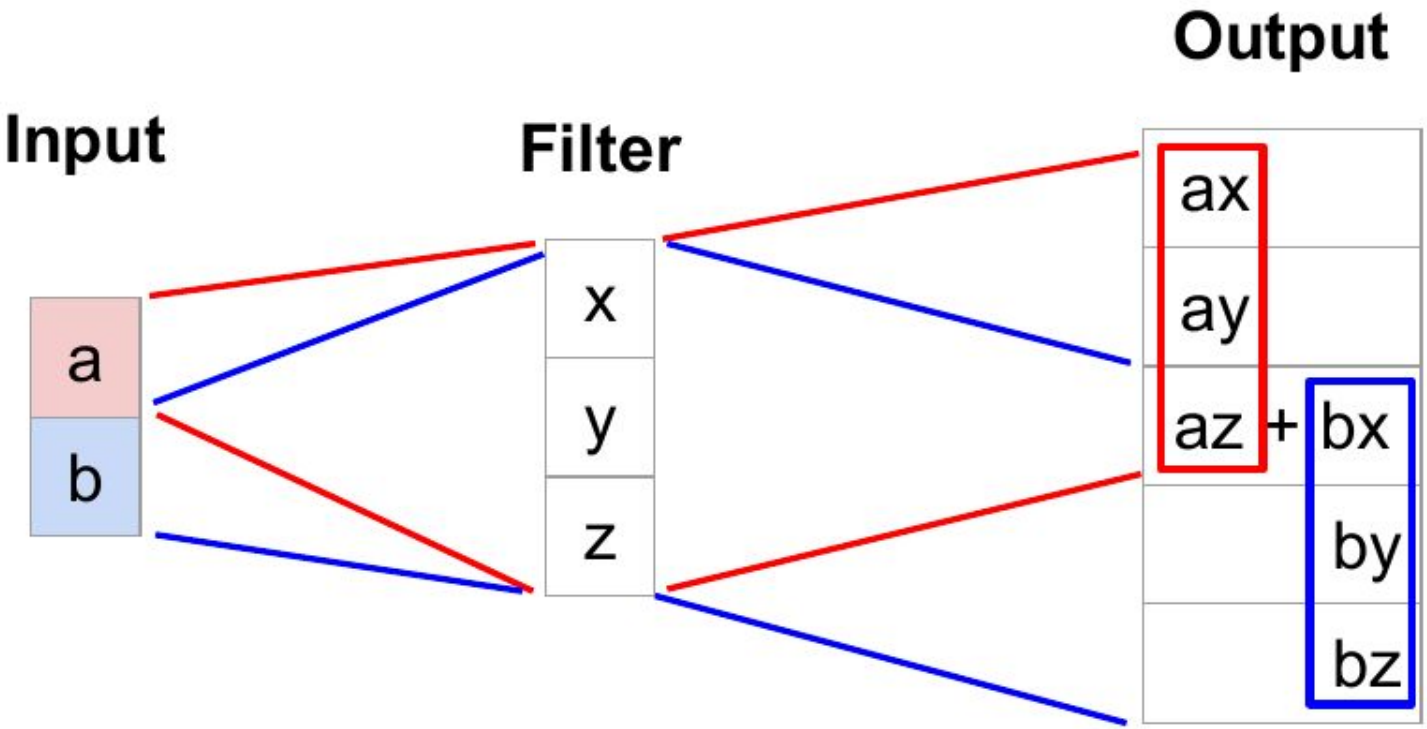
- **Bilinear**



Input: [2 x 2] Output: [4 x 4]

Layers [Learnable Upsampling: Transpose Convolution]

[used in *Semantic Segmentation*, *GANs*, *Autoencoders*]



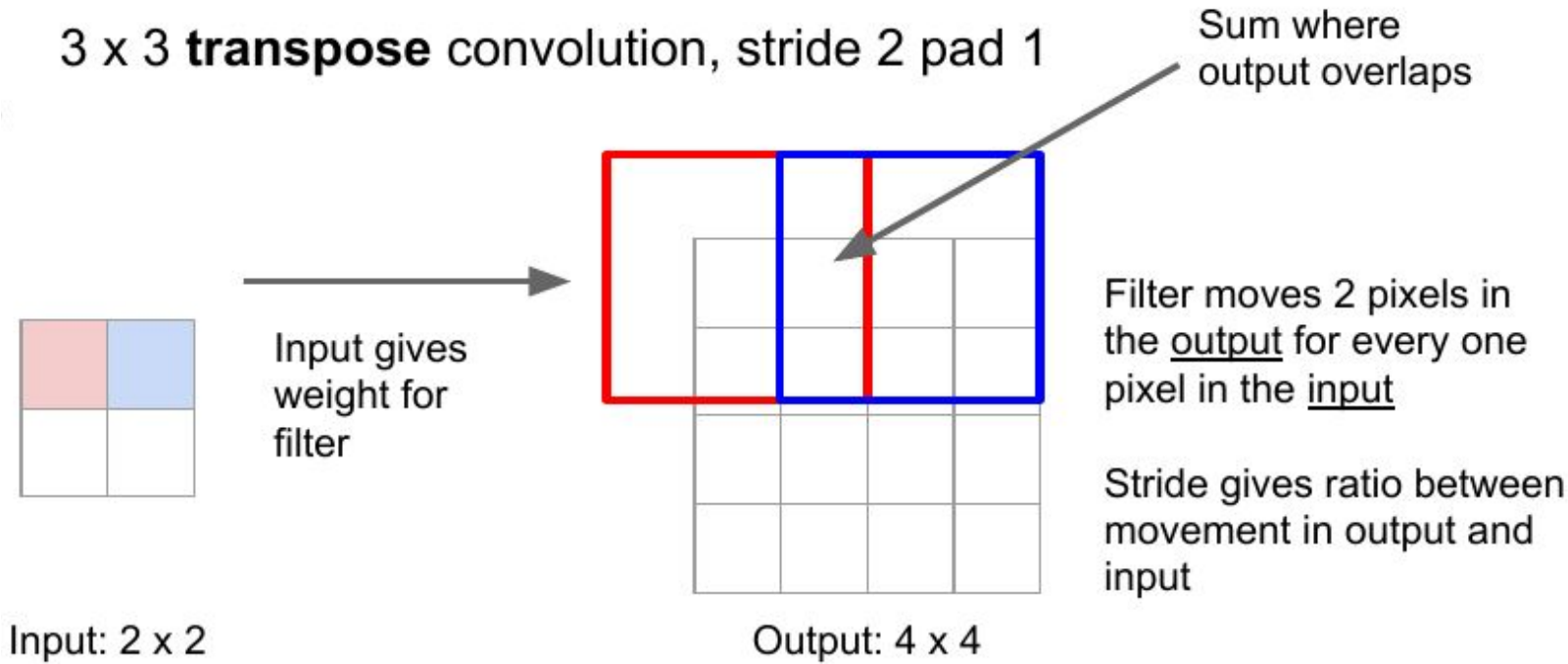
Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

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

http://cs231n.stanford.edu

Layers [Learnable Upsampling: Transpose Convolution]

[used in *Semantic Segmentation*, *GANs*, *Autoencoders*]



http://cs231n.stanford.edu

Convolutions

Computer Vision · **Image Feature Extractors** · 57 methods

Edit

Convolutions are a type of operation that can be used to learn representations from images. They involve a learnable kernel sliding over the image and performing element-wise multiplication with the input. The specification allows for parameter sharing and translation invariance. Below you can find a continuously updating list of convolutions.

Methods

Add a Method

Method	Year	Papers
Convolution	1980	13119
1x1 Convolution Network In Network	2013	4047
Depthwise Convolution	2016	795
Pointwise Convolution	2016	785
Depthwise Separable Convolution Xception: Deep Learning With Depthwise Separable Convolutions	2017	696
Grouped Convolution ImageNet Classification with Deep Convolutional Neural Networks	2012	514
Dilated Convolution Multi-Scale Context Aggregation by Dilated Convolutions	2015	459
3D Convolution	2015	156
Deformable Convolution Deformable Convolutional Networks	2017	97
SAC DetectoRS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution	2020	84
Invertible 1x1 Convolution Glow: Generative Flow with Invertible 1x1 Convolutions	2018	50
1D CNN Convolutional Neural Network and Rule-Based Algorithms for Classifying 12-lead ECGs	2020	43
Groupwise Point Convolution ESPNetv2: A Light-weight, Power Efficient, and General Purpose Convolutional Neural Network	2018	42
Transposed convolution Fully Convolutional Networks for Semantic Segmentation	2016	25

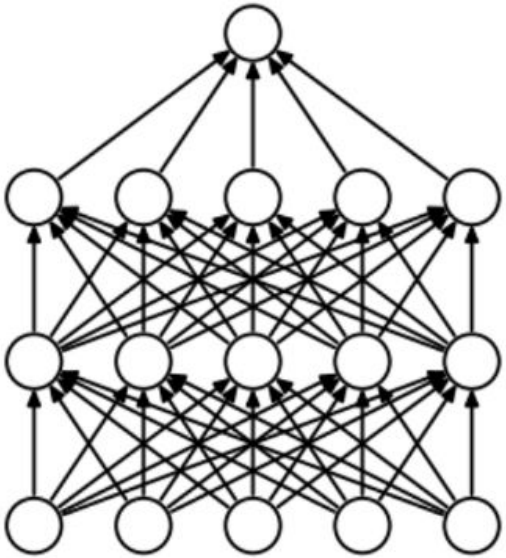
link

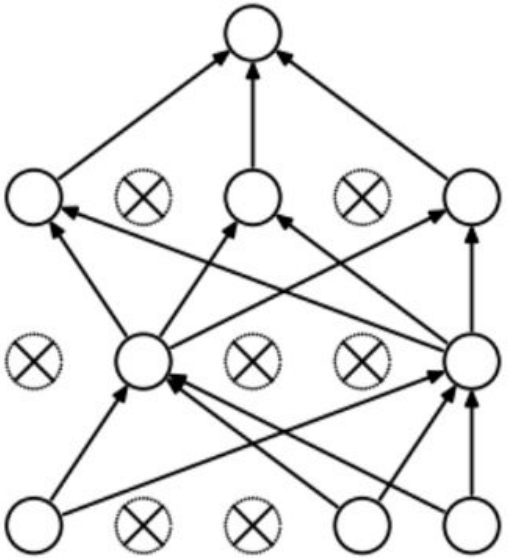
Content of today lecture

- Layers
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Upsampling, Learnable Upsampling]
  - **Layers [Batch Norm, Dropout]**
- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*
  - EfficientNet
- Architectures
  VGG, Inception, ResNet*, MobileNet*, EfficientNet
- Neural Architecture Search (NAS)
- Summary

Layers [Dropout]

- In-network ensembling
- Reduce overfitting

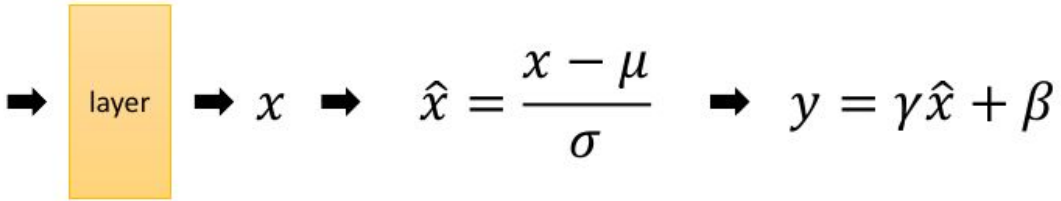




Layers [Batch Normalization]

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

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



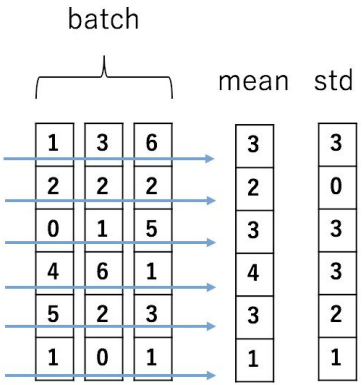
- μ: mean of x in mini-batch
- σ: std of x in mini-batch
- γ: scale
- β: shift

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

arXiv:1502.03167

Layers [Batch Normalization]

Batch normalization:

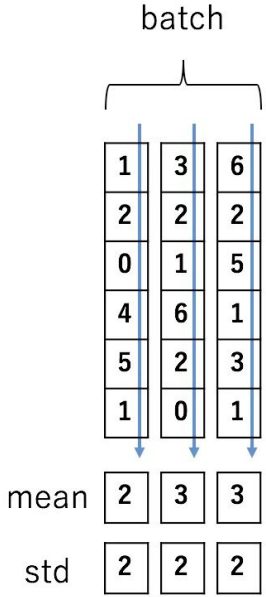


$$\mu_j = \frac{1}{N_b}\sum_{i=1}^{N_b} x_{ij}$$

$$\sigma_j^2 = \frac{1}{N_b}\sum_{i=1}^{N_b}(x_{ij} - \mu_j)^2$$

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

Layer normalization:

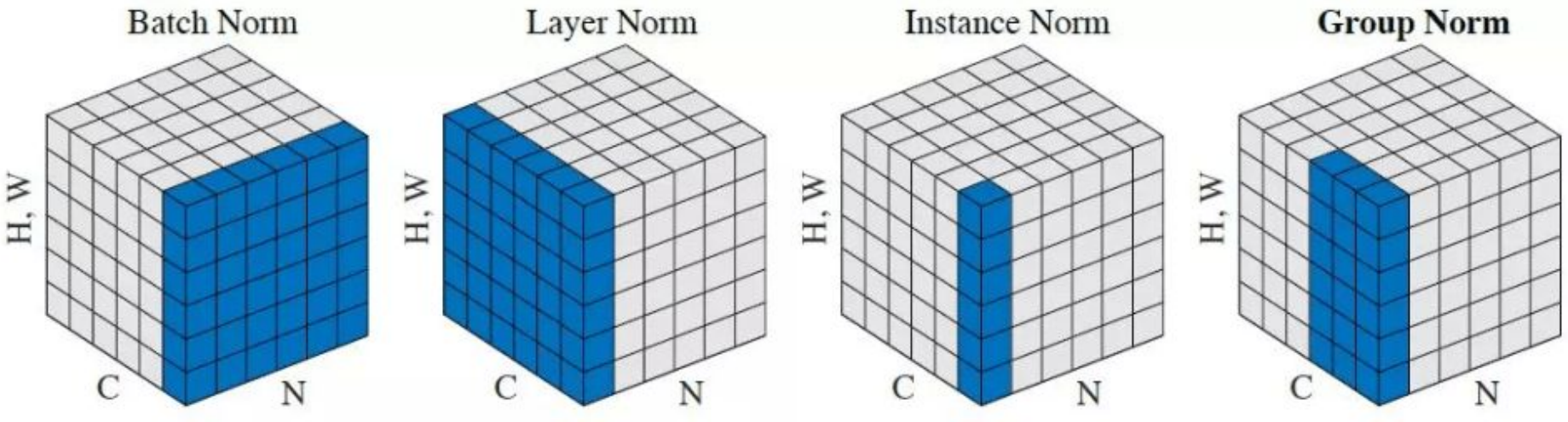


$$\mu_i = \frac{1}{N_f}\sum_{j=1}^{N_f} x_{ij}$$

$$\sigma_i^2 = \frac{1}{N_f}\sum_{j=1}^{N_b}(x_{ij} - \mu_i)^2$$

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

Layers [Other Normalization]



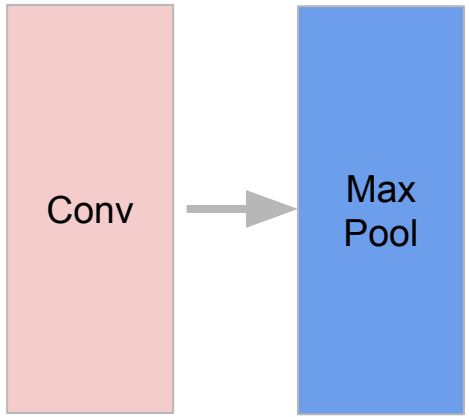
Content of today lecture

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

- **Blocks**
  - VGG, Inception
  - ResNet*
  - MobileNet*

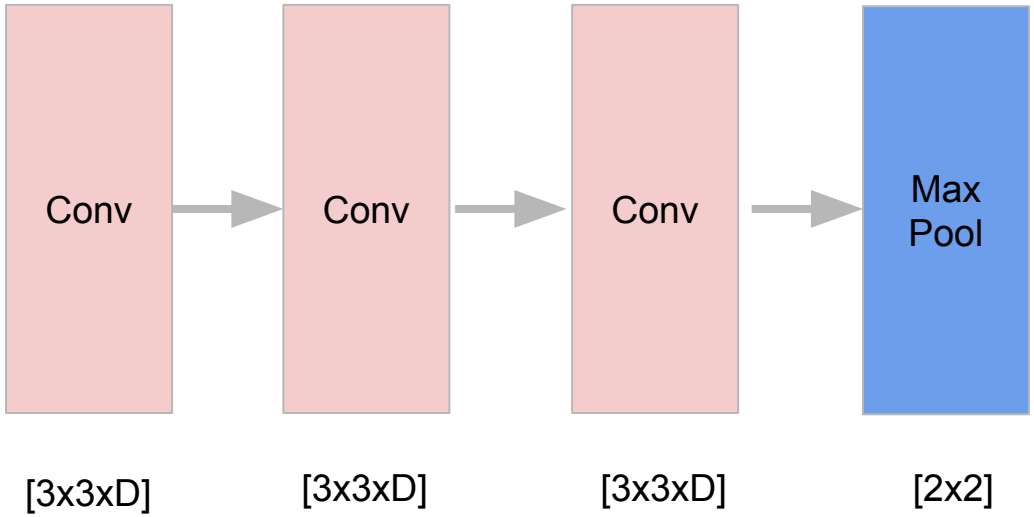
- Architectures
  [historical]: AlexNet, VGG, Inception,
  [modern]: ResNet, MobileNet, EfficientNet

Blocks [LeNet, AlexNet]



arXiv:1409.4842

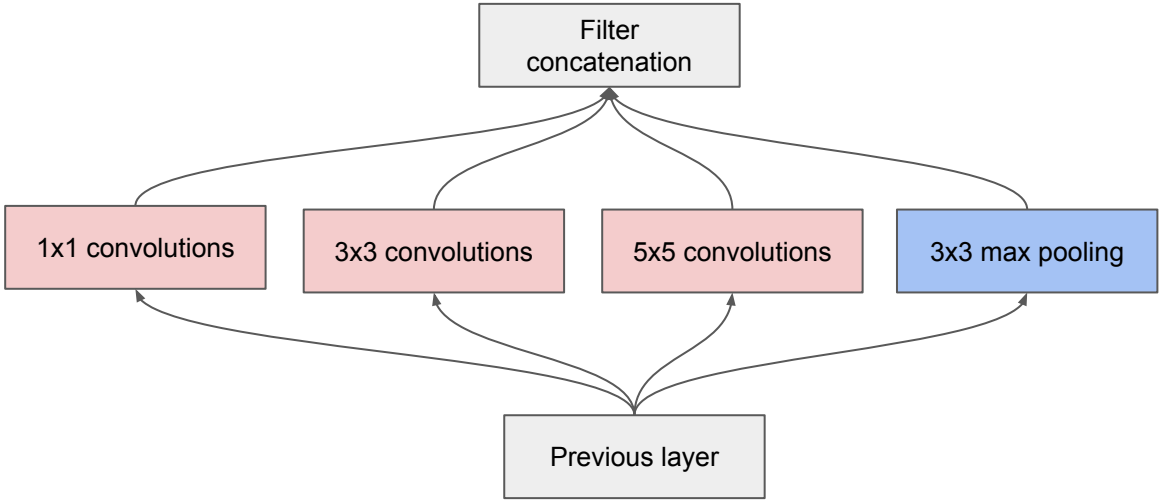
Blocks [VGG]



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

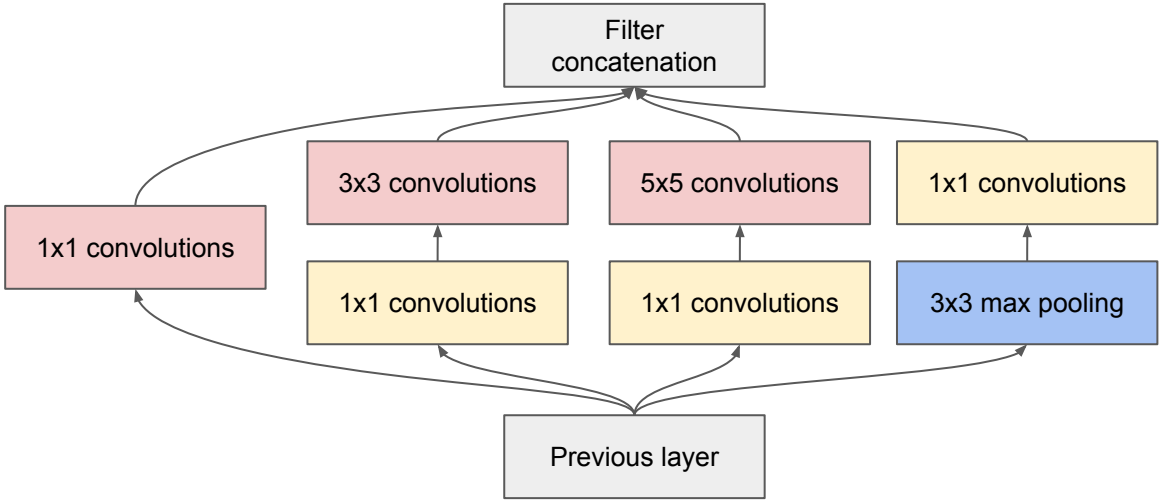


arXiv:1409.4842

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

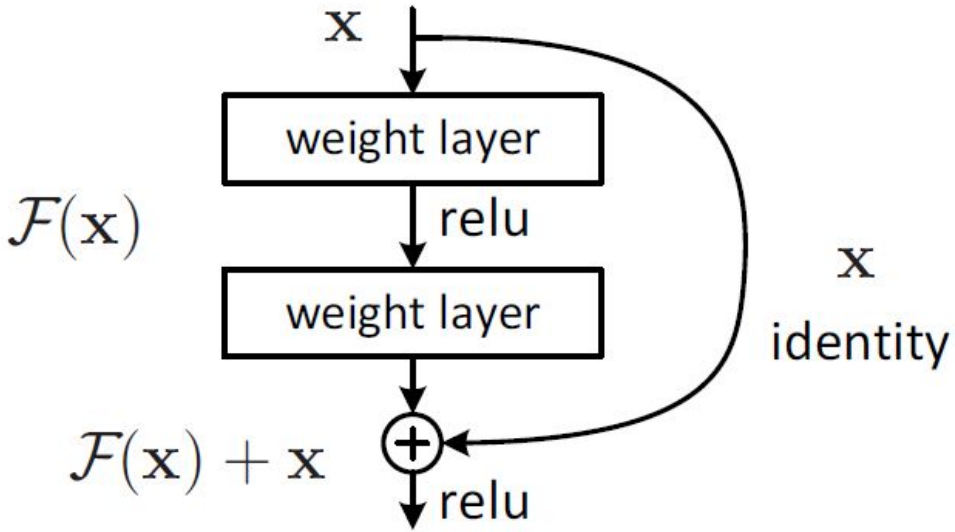


arXiv:1409.4842

Blocks [ResNet]

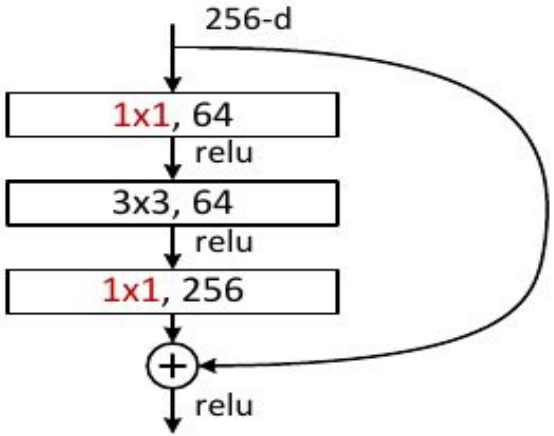
G(x) = x + F(x)

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



arXiv:1512.03385

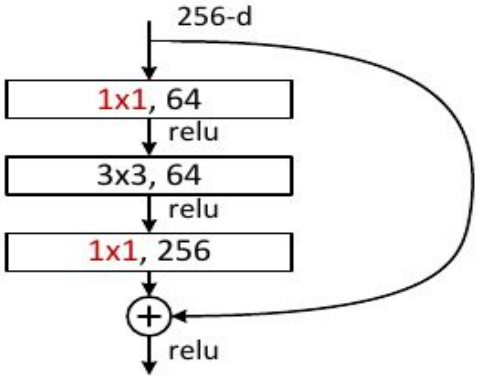
Blocks [ResNet, bottleneck]

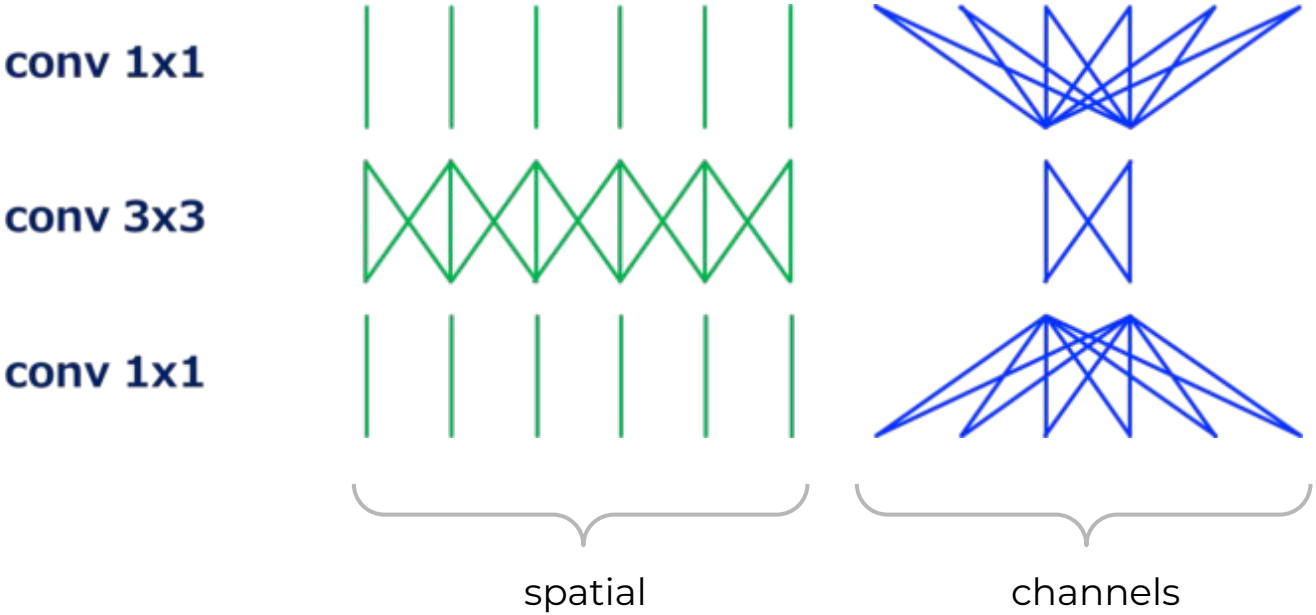


For deeper networks (ResNet-50+), use “bottleneck” layer to improve efficiency (similar to Inception)

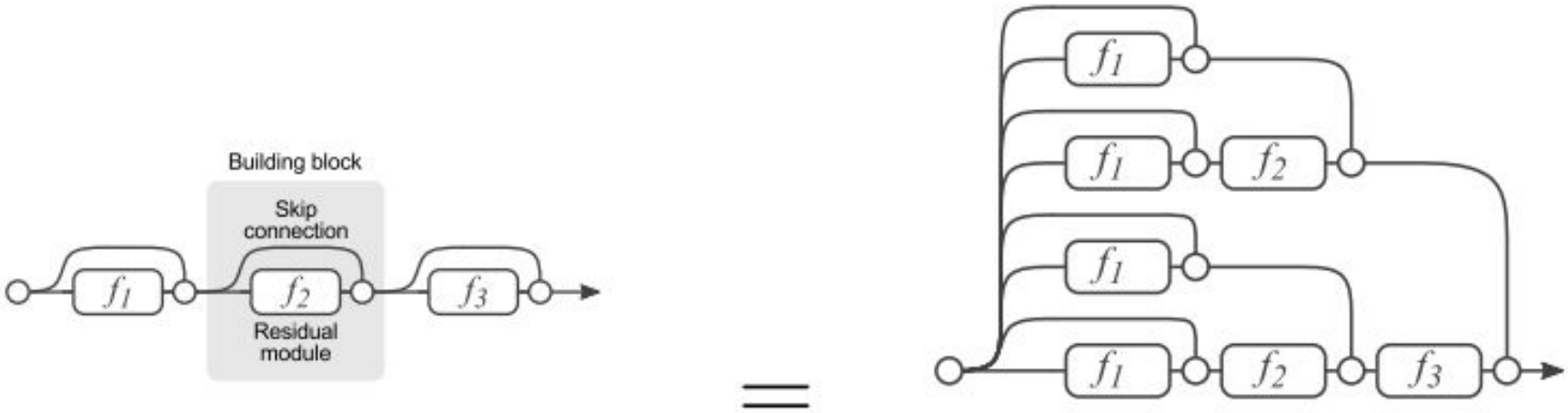
arXiv:1512.03385

Blocks [ResNet, bottleneck]

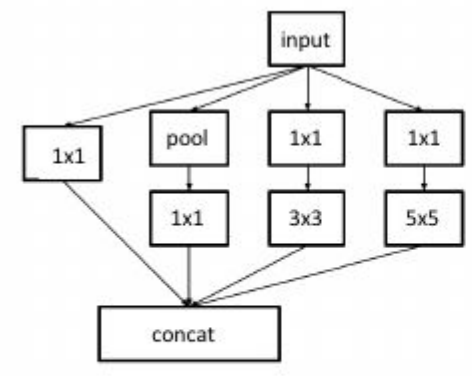




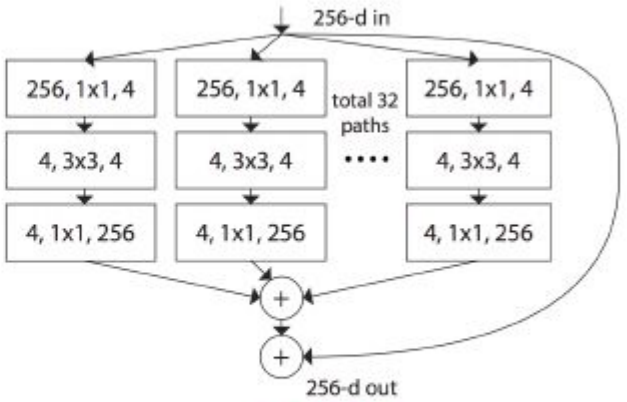
Blocks [ResNet, bottleneck]



Blocks [ResNeXt]

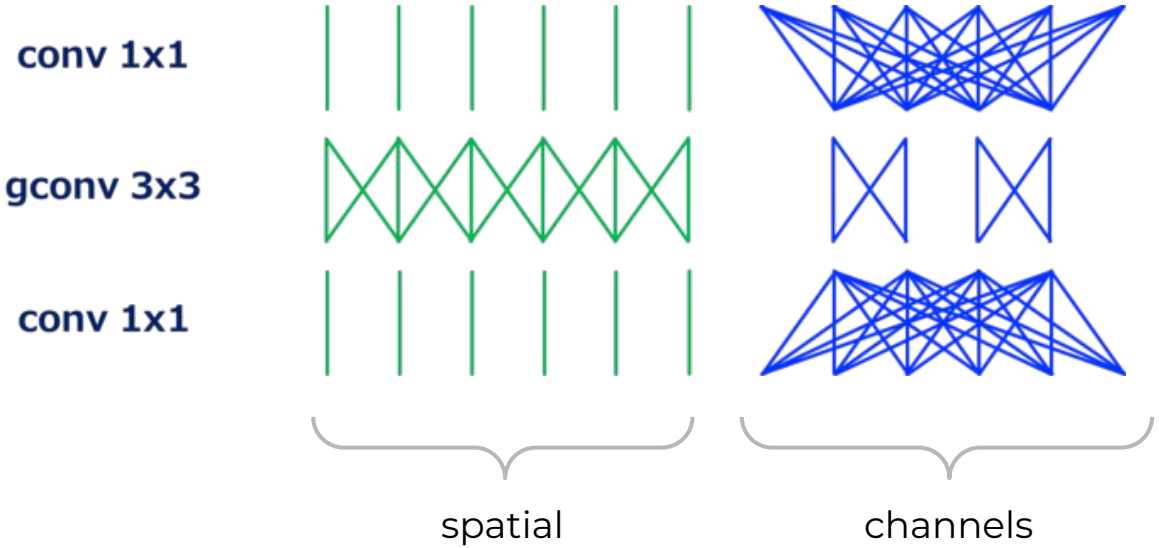


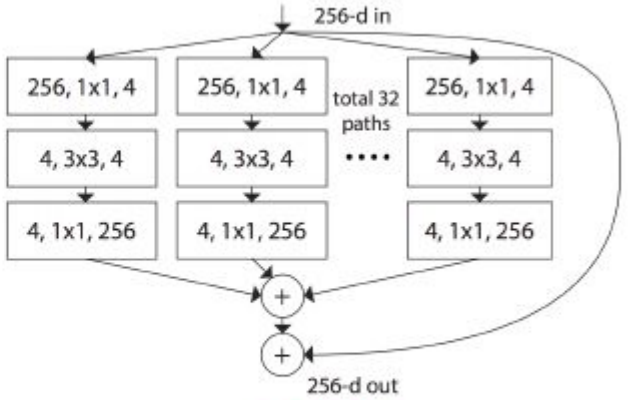
Inception:
heterogeneous multi-branch



ResNeXt:
uniform multi-branch

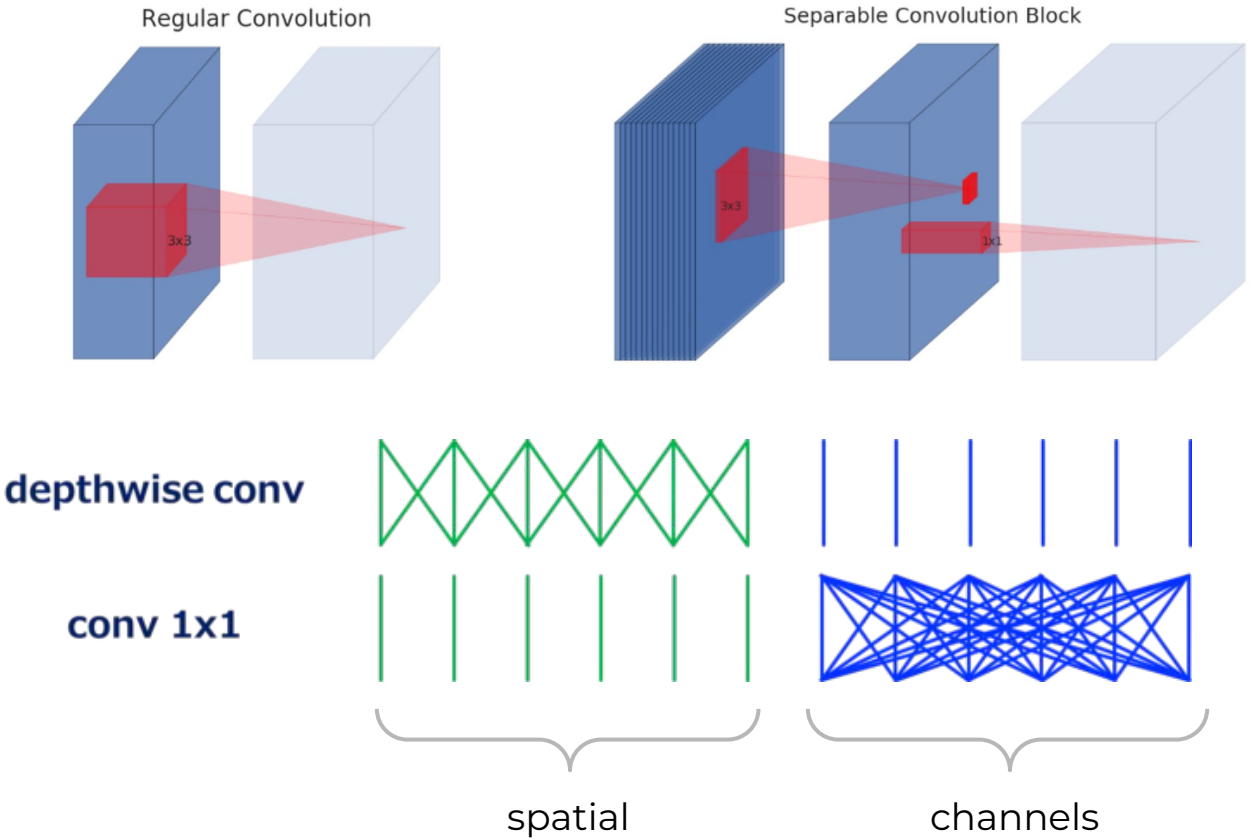
Blocks [ResNeXt]



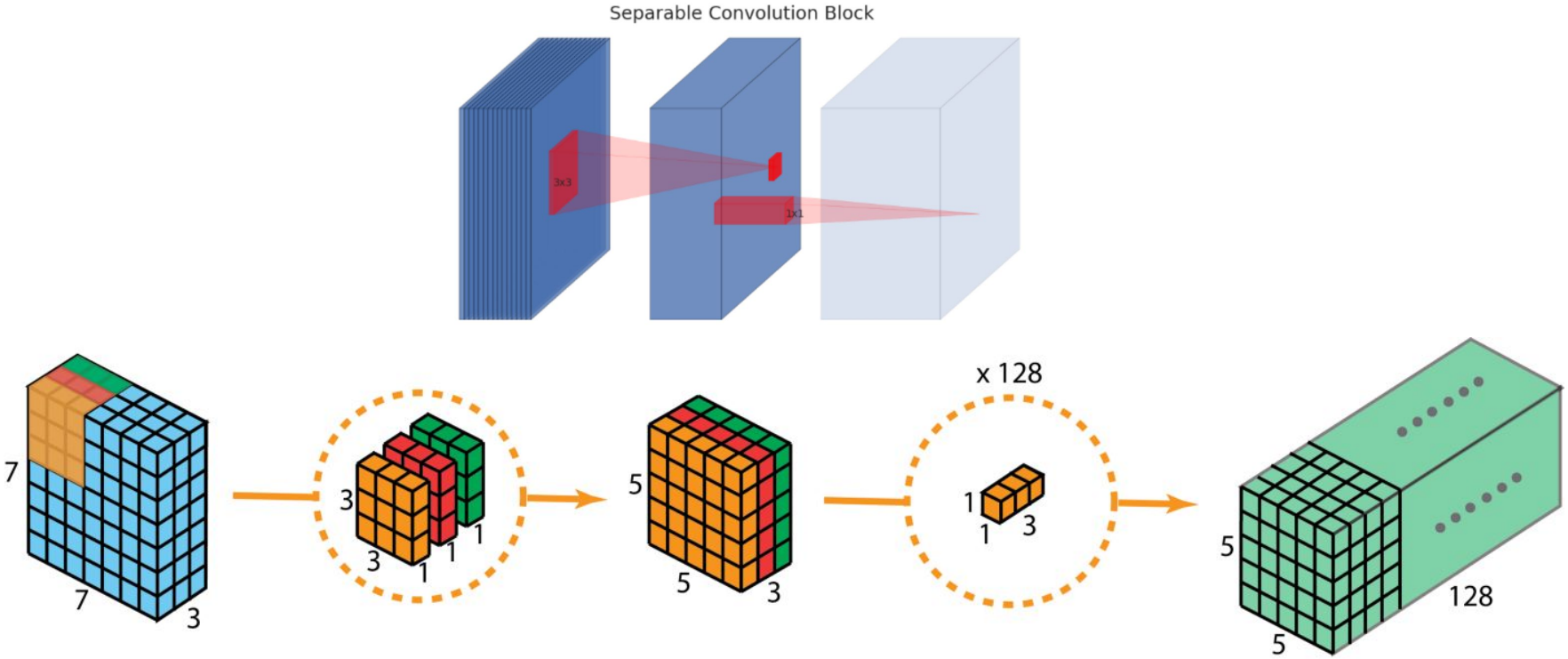


ResNeXt:
uniform multi-branch

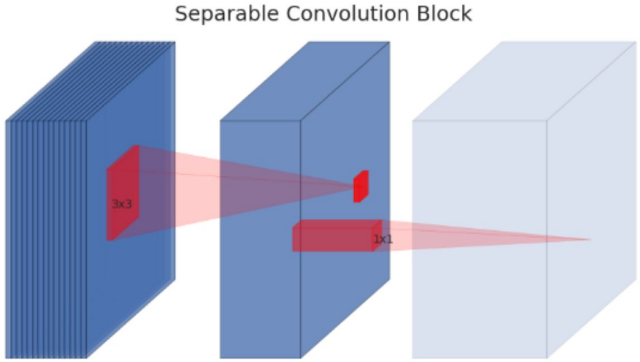
Blocks [depth-wise separable][MobileNet V1]

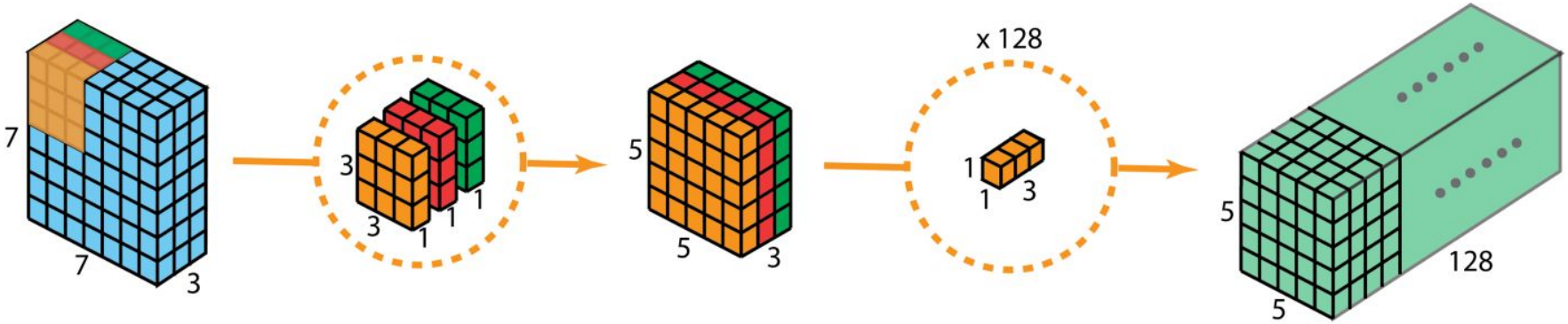


Blocks [depth-wise separable]



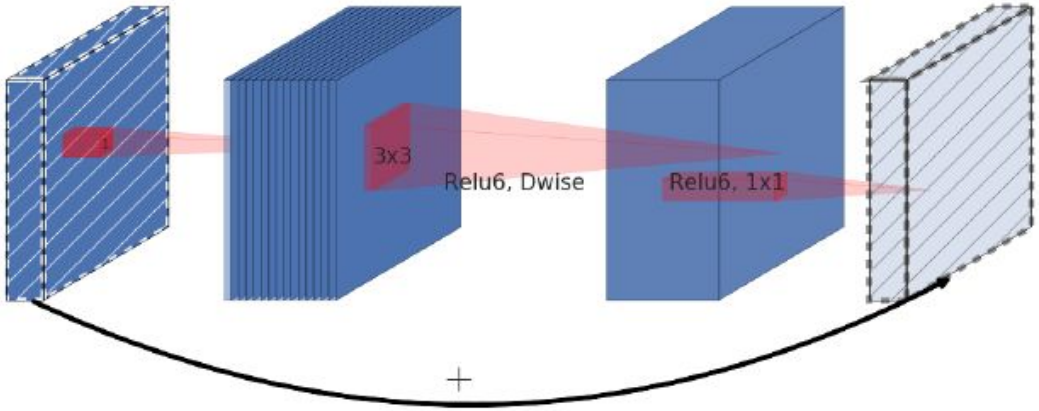
Blocks [depth-wise separable]

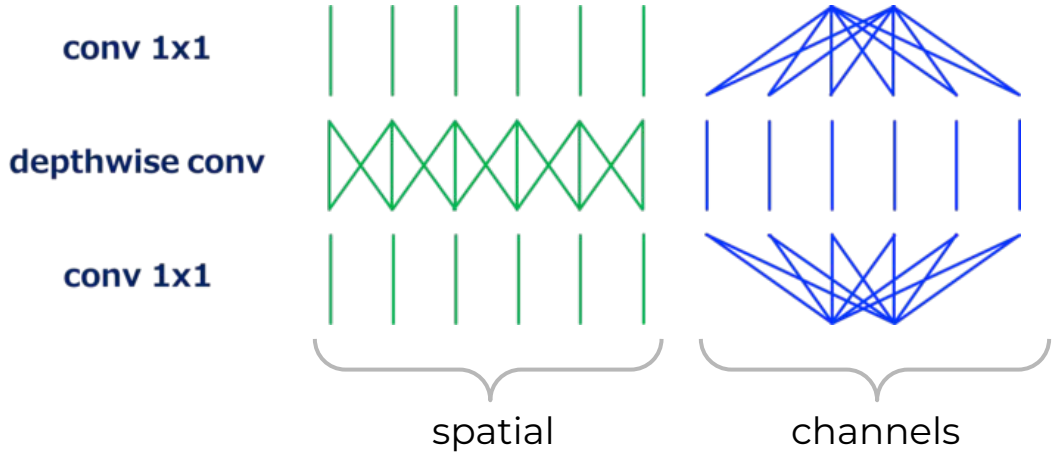




$$\frac{C_{in} \cdot H_{out} W_{out} C_{out} + KK \cdot H_{out} W_{out} C_{out}}{C_{in} KK \cdot H_{out} W_{out} C_{out}} = \frac{1}{K^2} + \frac{1}{C_{in}}$$

Blocks [mobile inverted bottleneck convolution (MBConv)]





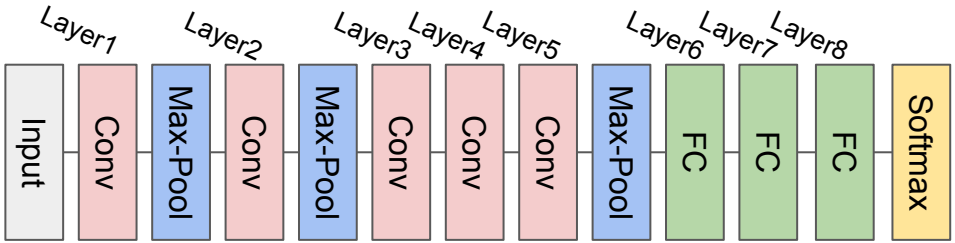
Content of today lecture

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

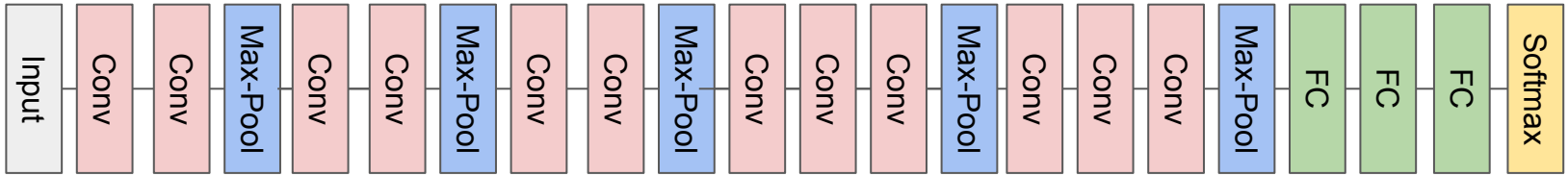
- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

- **Architectures**
  [historical]: AlexNet, VGG, Inception,
  [modern]: ResNet, MobileNet, EfficientNet

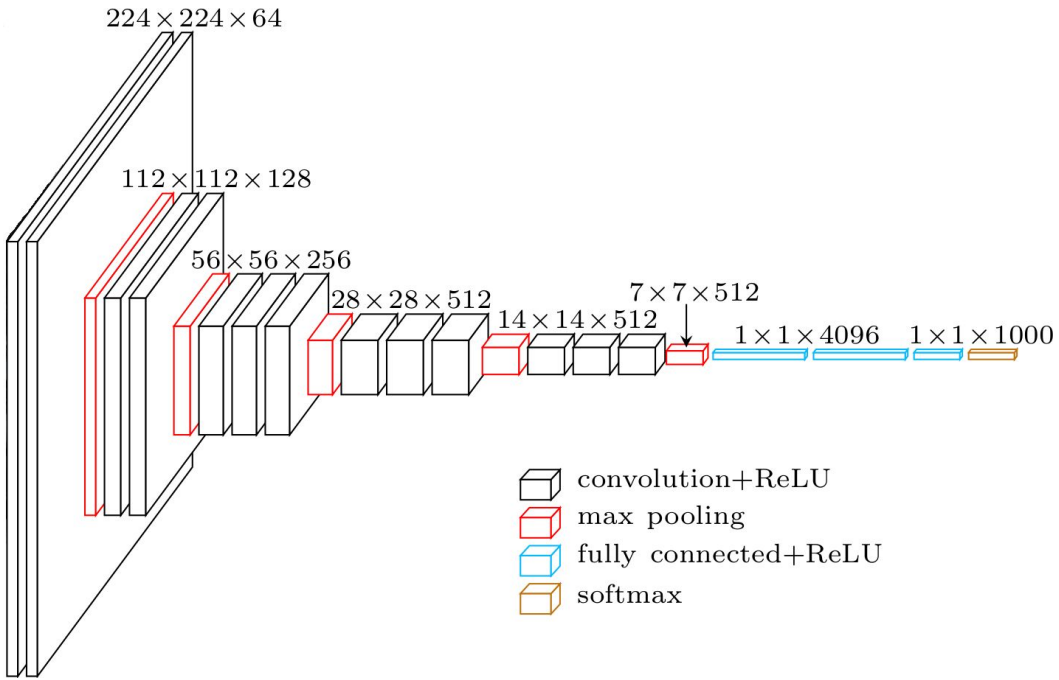
Net [AlexNet]


Layer1
Layer2
Layer3
Layer4
Layer5
Layer6
Layer7
Layer8
Input
Conv
Max-Pool
Conv
Max-Pool
Conv
Conv
Conv
Max-Pool
FC
FC
FC
Softmax

Net [VGG16]

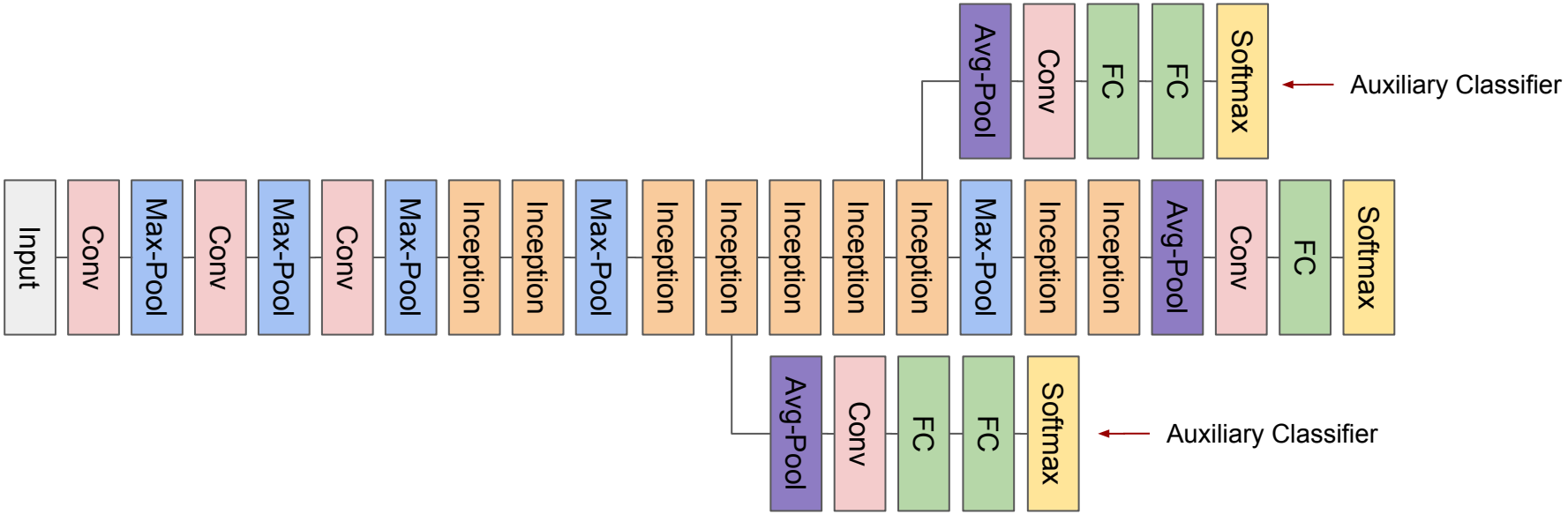

Input
Conv
Conv
Max-Pool
Conv
Conv
Max-Pool
Conv
Conv
Max-Pool
Conv
Conv
Conv
Max-Pool
Conv
Conv
Conv
Max-Pool
FC
FC
FC
Softmax

Net [VGG16]

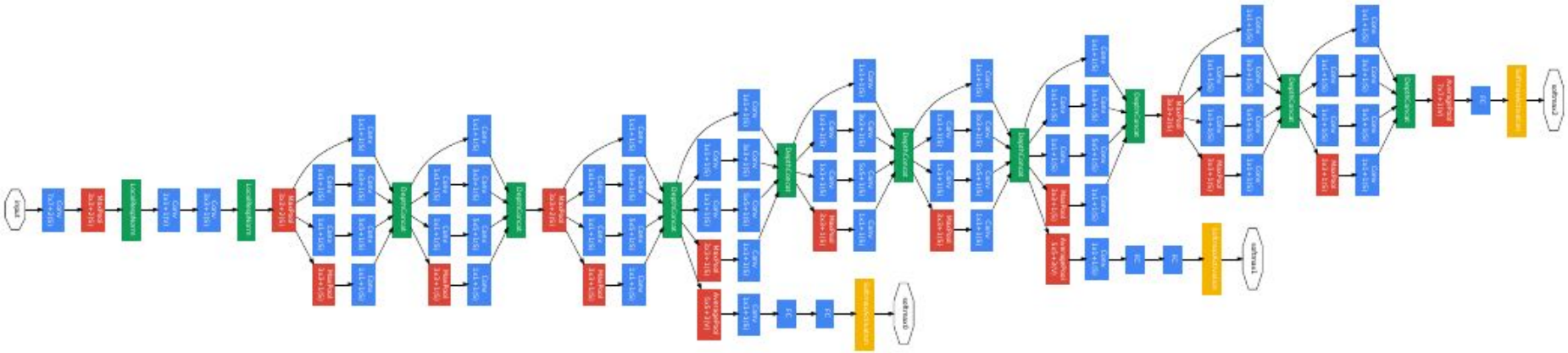


- 3 3x3 Conv as the module
- Stack the same module
- Same computation for each module
(1/2 spatial size => 2x filters)

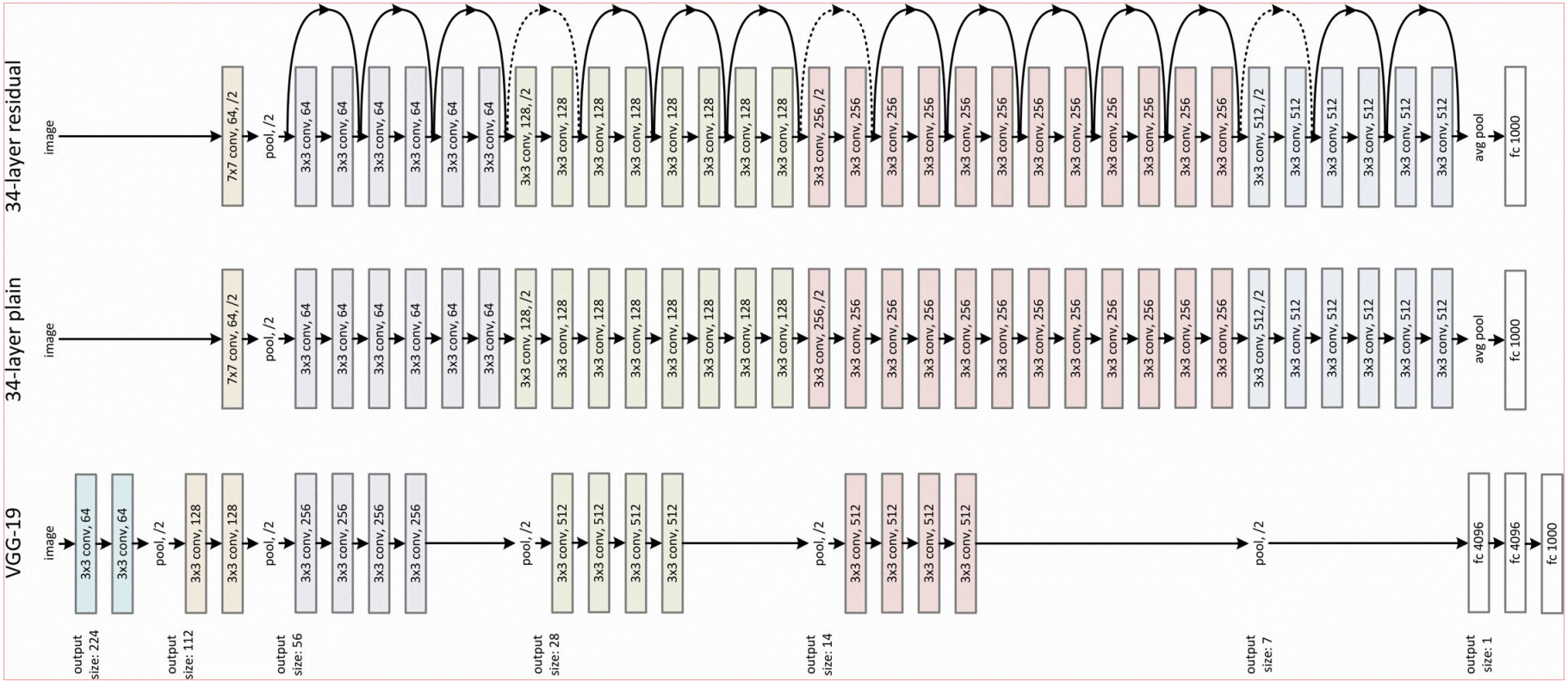
Net [GoogLeNet]



Net [GoogLeNet]



Net [ResNet]



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]

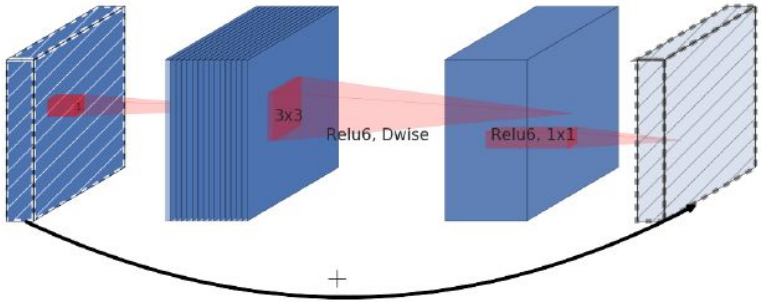
ResNet strikes back!

	A1-A2-org.		A3		Cost							ImageNet-1k-val			
	train	test	train	test	A1	A2	A1-A2		A3			A1	A2	A3	org.
↓ Architecture	res.	res.	res.	res.	time (hour)		# GPU	Pmem	time	# GPU	Pmem	Accuracy(%)			
ResNet-18 [13]†	224	224	160	224	186	93	2	12.5	28	2	6.5	71.5	70.6	68.2	69.8
ResNet-34 [13]†	224	224	160	224	186	93	2	17.5	27	2	9.0	76.4	75.5	73.0	73.3
ResNet-50 [13]†	224	224	160	224	110	55	4	22.0	15	4	11.4	80.4	79.8	78.1	76.1
ResNet-101 [13]†	224	224	160	224	74	37	8	16.3	8	8	8.5	81.5	81.3	79.8	77.4
ResNet-152 [13]†	224	224	160	224	92	46	8	22.5	9	8	11.8	82.0	81.8	80.6	78.3

2110.00476

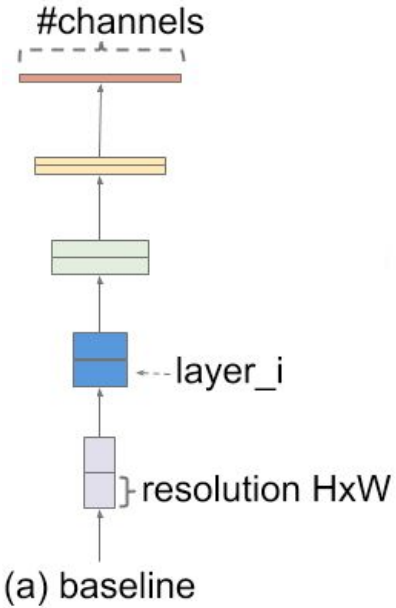
Net [MobileNet V2]

MBConv block



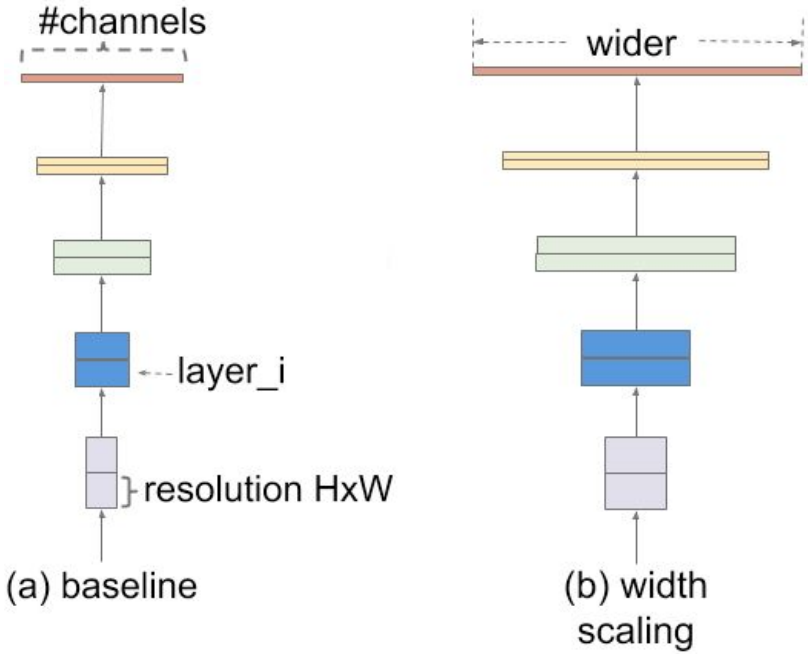
Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck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

Net [EfficientNet]



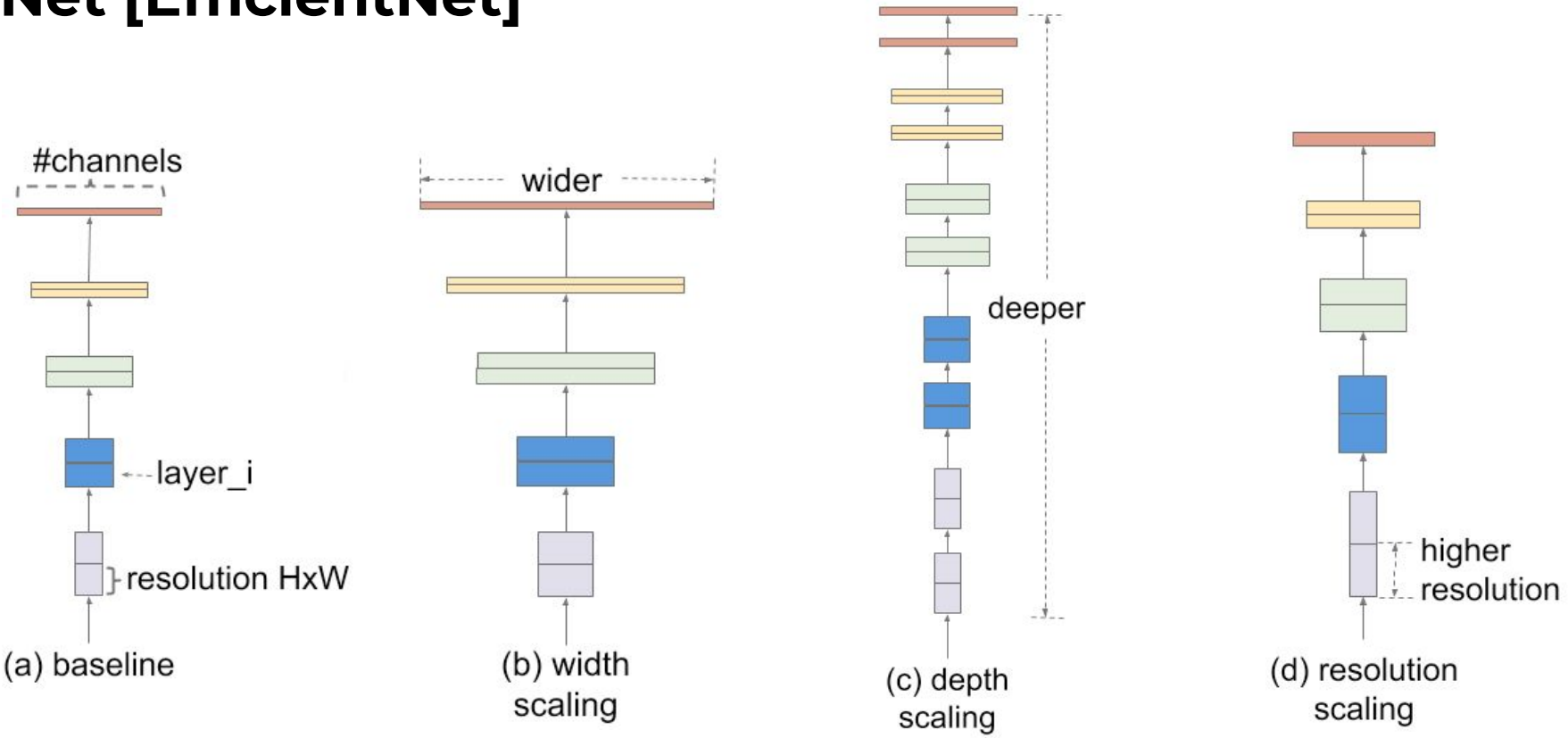
https://arxiv.org/abs/1905.11946

Net [EfficientNet]



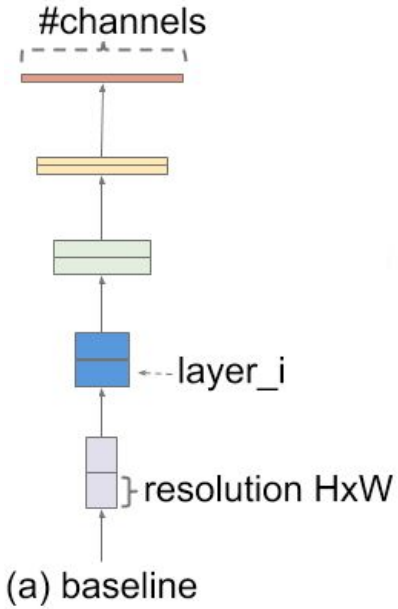
https://arxiv.org/abs/1905.11946

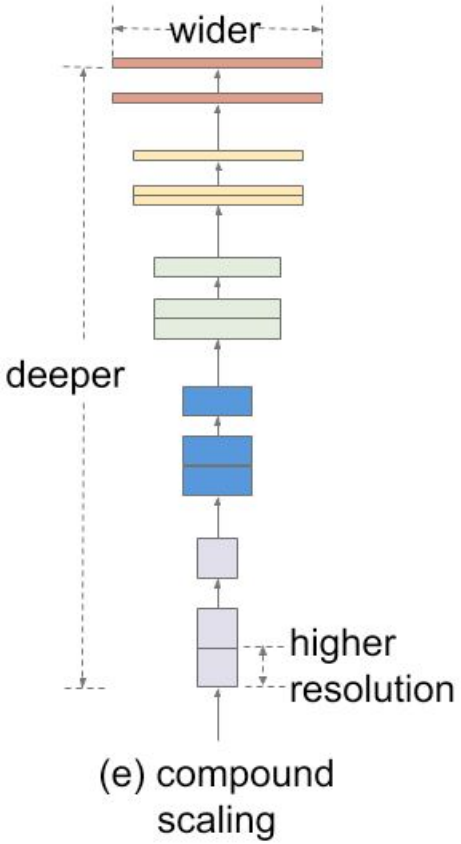
Net [EfficientNet]



https://arxiv.org/abs/1905.11946

Net [EfficientNet]





depth: $d = \alpha^\phi$

width: $w = \beta^\phi$

resolution: $r = \gamma^\phi$

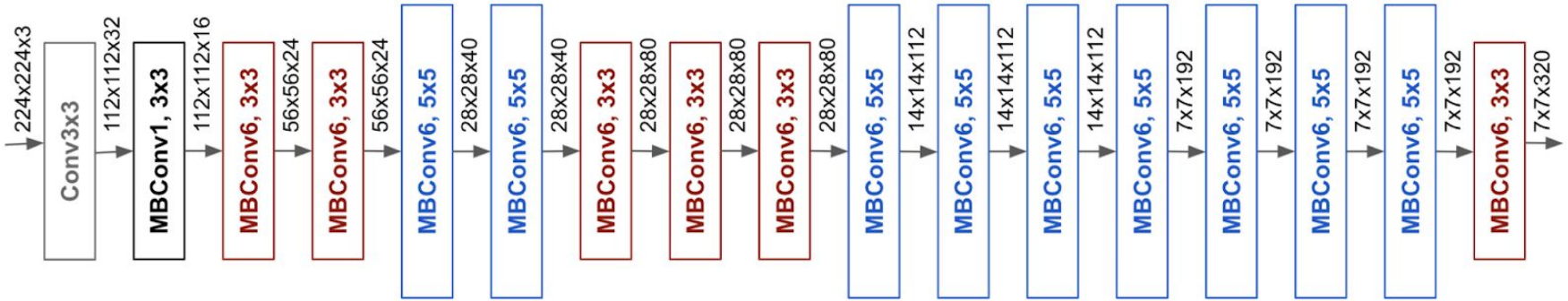
s.t. $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$

$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$

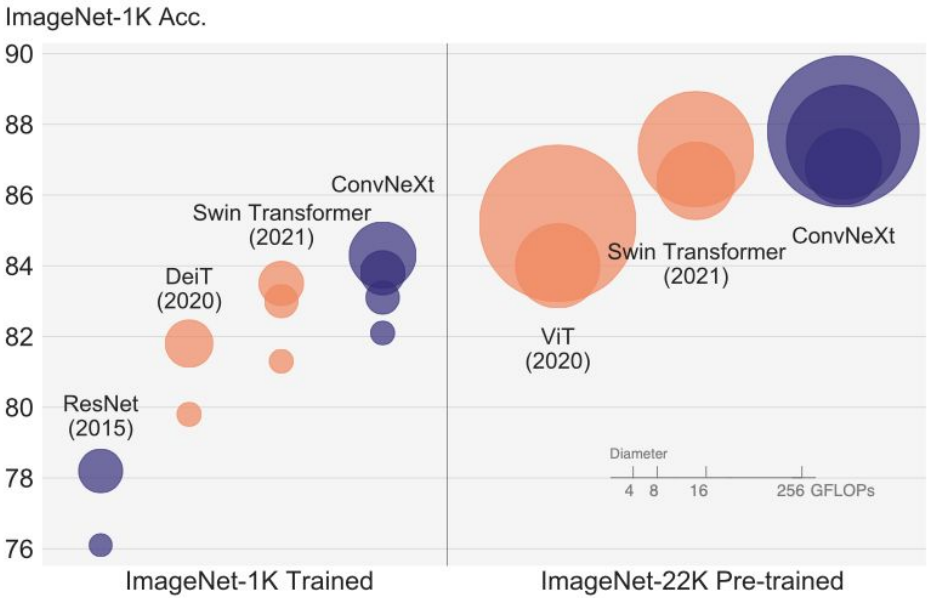
https://arxiv.org/abs/1905.11946

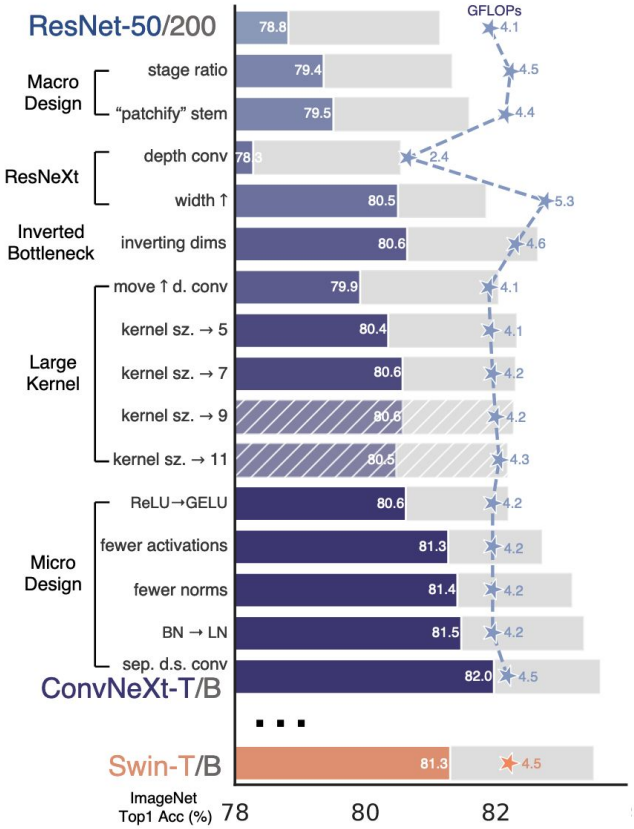
Net [EfficientNet]

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1



https://arxiv.org/abs/1905.11946





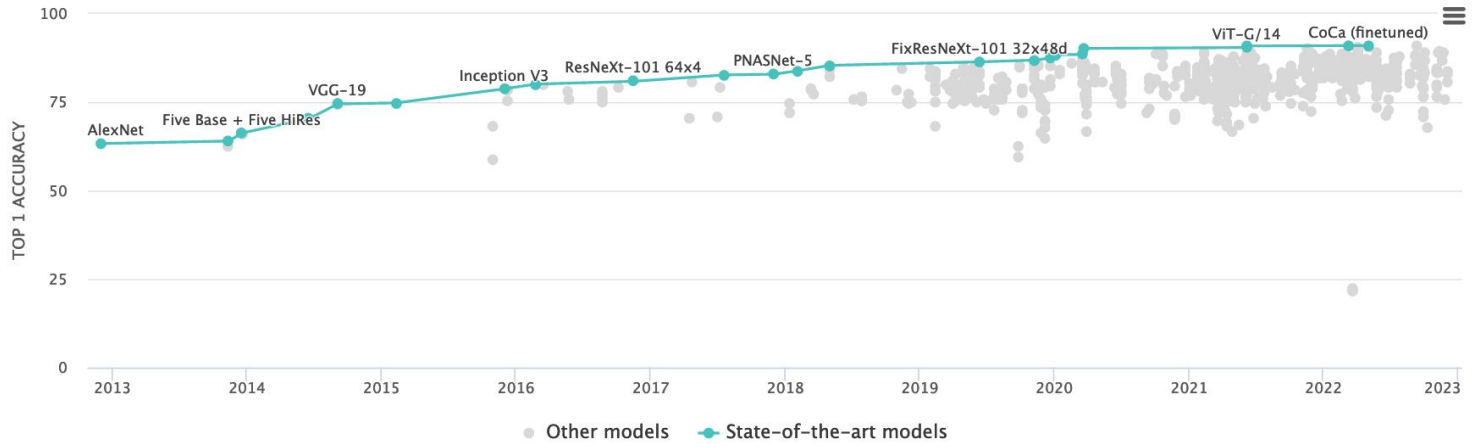
[2201.03545] A ConvNet for the 2020s

SOTA

Image Classification on ImageNet

Leaderboard Dataset

View Top 1 Accuracy by Date



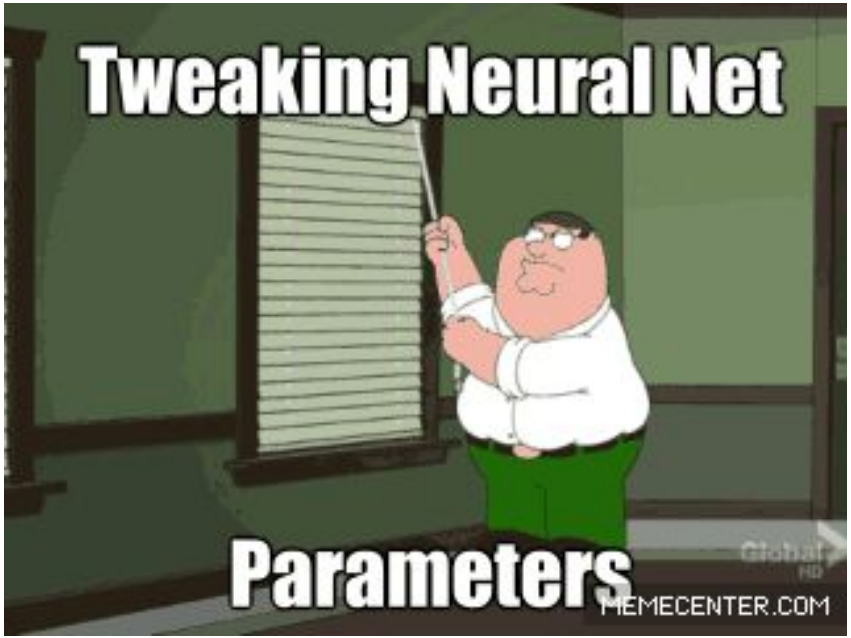
https://paperswithcode.com/sota/image-classification-on-imagenet

Content of today lecture

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

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

- Architectures
  [historical]: AlexNet, VGG, Inception,
  [modern]: ResNet, MobileNet, EfficientNet


Tweaking Neural Net
Parameters
MEMECENTER.COM