



Deep Learning

Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir



<https://www.aparat.com/mehran.safayani>



https://github.com/safayani/deep_learning_course

Department of Electrical and computer engineering, Isfahan university of technology, Isfahan, Iran

Computer Vision Problems

Image Classification



→ Cat? (0/1)

64x64
Object detection



Neural Style Transfer



Content Style



Generated image

Deep Learning on large images



Cat? (0/1)

$64 \times 64 \times 3 = 12288$

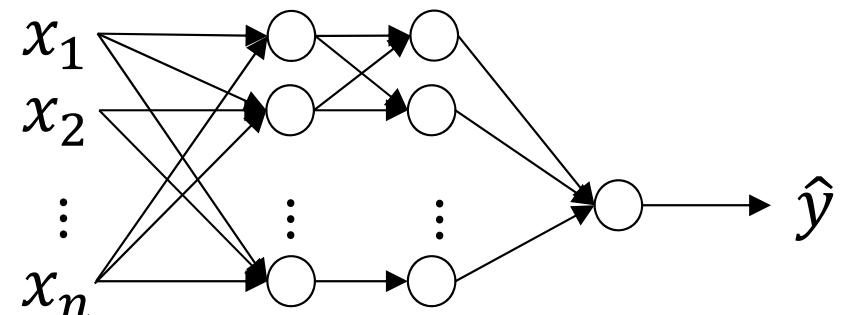


$1000 \times 1000 \times 3 = 3m$

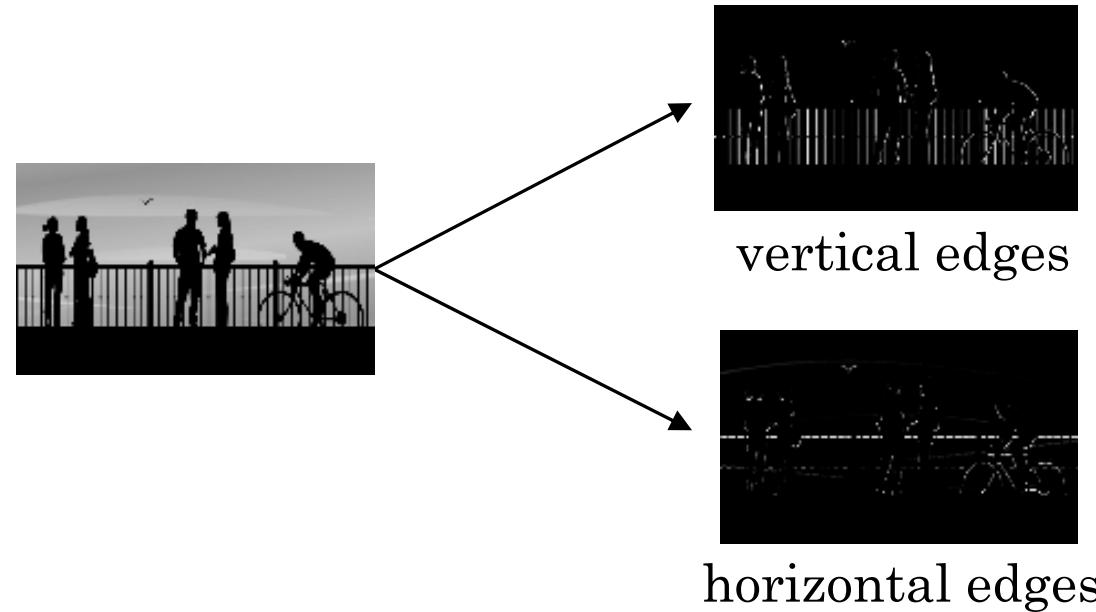
$n=3m$

$n^1 = 1000$

$W_{(1000,3m)}^1$



Edge detection



Vertical edge detection

3 ¹	0 ⁰	1 ⁻¹	2 ⁻¹	7 ⁰	4 ⁻¹
1 ¹	5 ⁰	8 ⁻¹	9 ⁻¹	3 ⁰	1 ⁻¹
2 ⁻¹	7 ⁰	2 ⁻¹	5 ⁻¹	1 ⁰	3 ⁻¹
0 ¹	1 ⁰	3 ⁻¹	1 ⁻¹	7 ⁰	8 ⁻¹
4	2	1	6	2	8
2	4	5	2	3	9

*

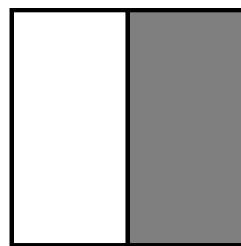
1	0	-1
1	0	-1
1	0	-1

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Vertical edge detection

10^1	10^0	10^{-1}	0	0	0
10^1	10^0	10^{-1}	0	0	0
10^1	10^0	10^{-1}	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

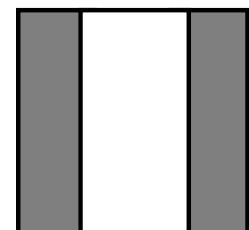


1	0	-1
1	0	-1
1	0	-1

*

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

=



*

Vertical edge detection examples

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

1	0	-1
1	0	-1
1	0	-1



=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

1	0	-1
1	0	-1
1	0	-1



=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0



Vertical and Horizontal Edge Detection

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

1	1	1
0	0	0
-1	-1	-1

$$= \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 30 & 10 & -10 & -30 \\ \hline 30 & 10 & -10 & -30 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Learning to detect edges

1	0	-1
1	0	-1
1	0	-1

1	0	-1
2	0	-2
1	0	-1

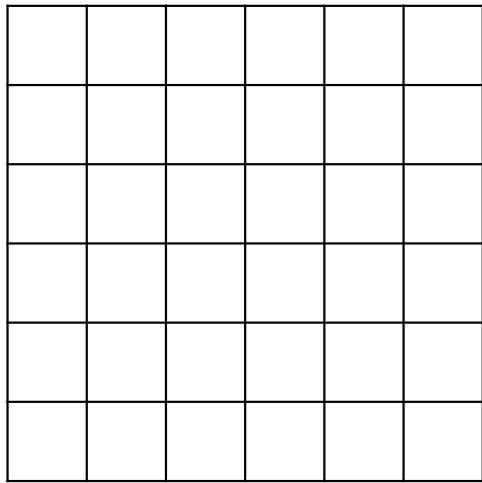
Sobel filter

3	0	-1
10	0	-10
3	0	-3

Schass filter

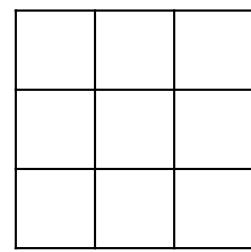
3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



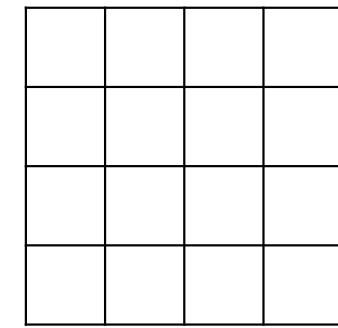
$n \times n$

*



$f \times f$

=



$n - f + 1 \times n - f + 1$

6×6

3×3

4×4

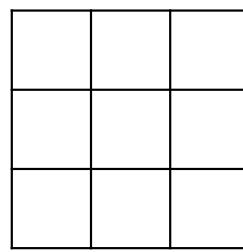
Padding

0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

$$n + 2p \times n + 2p$$

$$6 + 2 \times 6 + 2$$

*



$$f \times f$$

=

$$n + 2p - f + 1 \times n + 2p - f + 1$$

$$3 \times 3$$

$$6 \times 6$$

Valid and Same convolutions

“Valid”:

$$n \times n$$

$$f \times f$$

$$n - f + 1 \times n - f + 1$$

“Same”: Pad so that output size is the same as the input size.

$$n + 2p - f + 1 = n$$

$$p = \frac{f - 1}{2}$$

Strided convolution

2	3	4	7	3	4	4	6	3	2	4	9	4	
6	1	6	0	9	1	8	0	7	1	4	0	3	2
3	-3	4	4	8	3	3	4	8	3	9	4	7	4
7	1	8	0	3	1	6	0	6	1	3	0	4	2
4	-3	2	4	1	3	8	4	5	3	4	4	6	4
3	1	2	0	4	1	1	0	9	1	8	0	3	2
0	-1	1	0	3	-1	9	0	2	-1	1	0	4	3

*

3	4	4
1	0	2
-1	0	3

=

91	100	83
69	91	127
44	72	74

$$\left\lfloor \frac{n-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n-f}{s} + 1 \right\rfloor$$

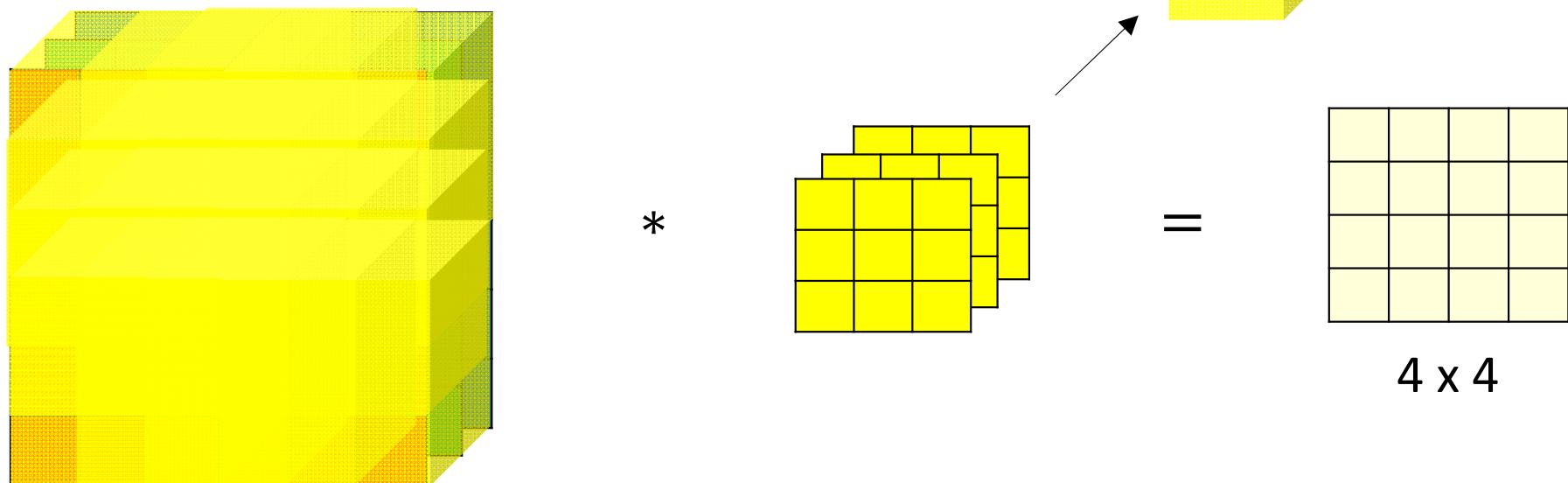
Summary of convolutions

$n \times n$ image $f \times f$ filter

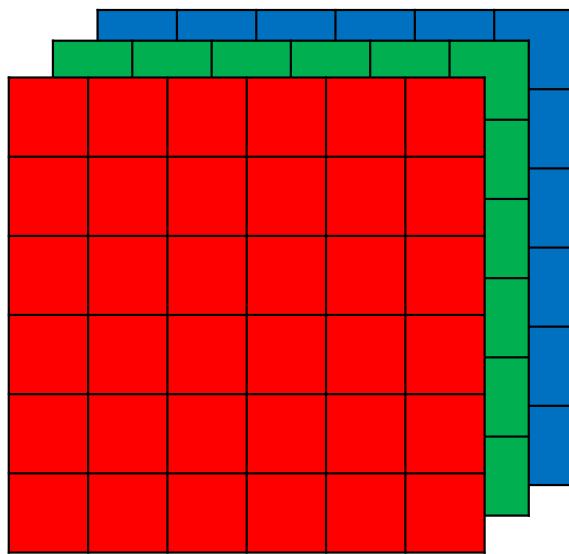
padding p stride s

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \quad \times \quad \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

Convolutions on RGB image



Multiple filters



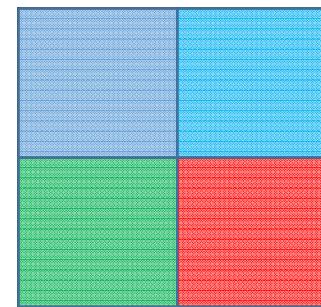
$6 \times 6 \times 3$

$$\begin{matrix} * & \begin{matrix} \text{3} \times \text{3} \times \text{3} \\ \text{3} \times \text{3} \times \text{3} \end{matrix} & = & \begin{matrix} \text{4} \times \text{4} \\ \text{4} \times \text{4} \end{matrix} \\ & \begin{matrix} \text{yellow} \\ \text{yellow} \end{matrix} & & \begin{matrix} \text{yellow} \\ \text{yellow} \end{matrix} \\ & \begin{matrix} \text{yellow} \\ \text{yellow} \end{matrix} & = & \begin{matrix} \text{4} \times \text{4} \times \text{2} \\ \text{4} \times \text{4} \end{matrix} \end{matrix}$$

The diagram illustrates the convolution process. On the left, two 3D kernel tensors are shown. The top one is labeled $3 \times 3 \times 3$ and is colored yellow. The bottom one is also labeled $3 \times 3 \times 3$ and is colored orange. Both kernels are shown in a 3x3x3 grid format. An asterisk (*) between the input and the first kernel indicates the multiplication operation. An equals sign (=) followed by a 4x4 grid of yellow blocks indicates the result of the first convolution step. A second equals sign (=) followed by another 4x4 grid of yellow blocks indicates the result of applying the second kernel to the previous result. This results in a final output tensor of size $4 \times 4 \times 2$, shown as a 4x4x2 grid of yellow blocks.

Pooling layer: Max pooling

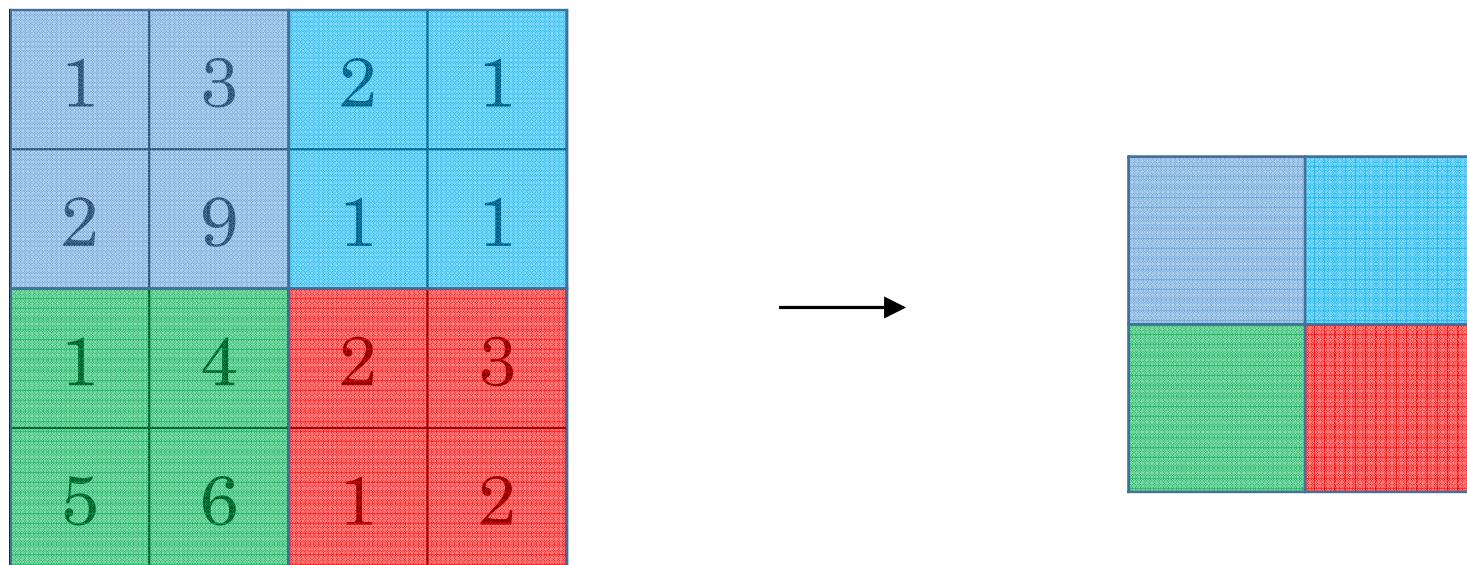
1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2



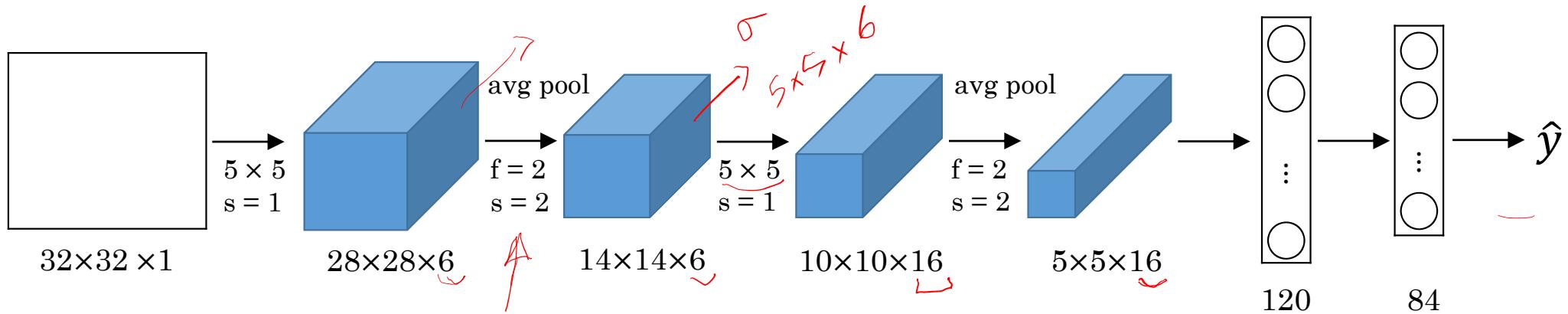
Pooling layer: Max pooling

1	3	2	1	3
2	9		1	5
1				2
8	3		1	0
5	6	1	2	9

Pooling layer: Average pooling



LeNet - 5



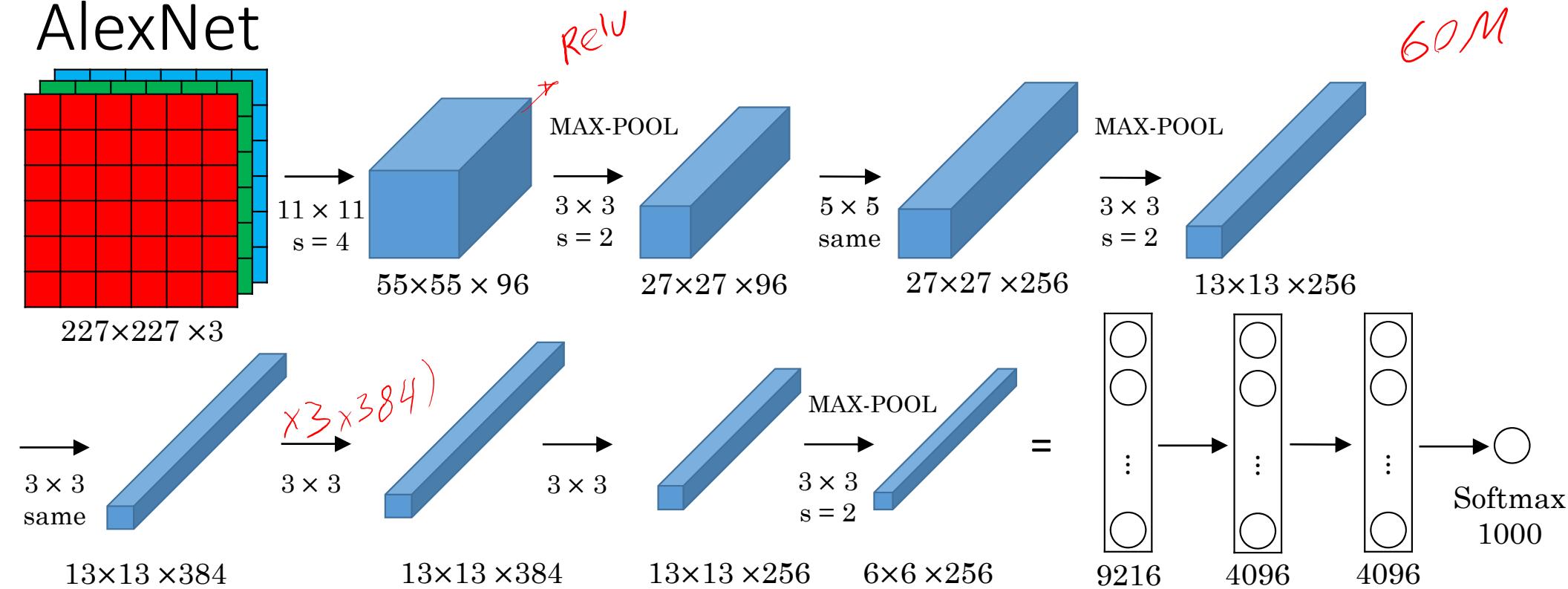
	0	1	2	3	4	...	15
1	x		x		x		x
2	x		x		x		x
3	x			x		x	x
4		x		x		x	x
5		x			x		x
6		x					x

[LeCun et al., 1998. Gradient-based learning applied to document recognition]

Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
			$(5 \times 5 \times 3 + 1) \times 8$
$x=2$ $y=2$	↙		
			$(5 \times 5 \times 8 + 1) \times 16$
$x=2$ $y=2$			
			$5 \times 5 \times 16 \times 120 + 120$
			$120 \times 84 + 84$
			$84 \times 10 + 10$

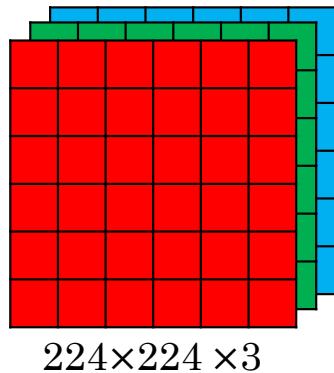
AlexNet



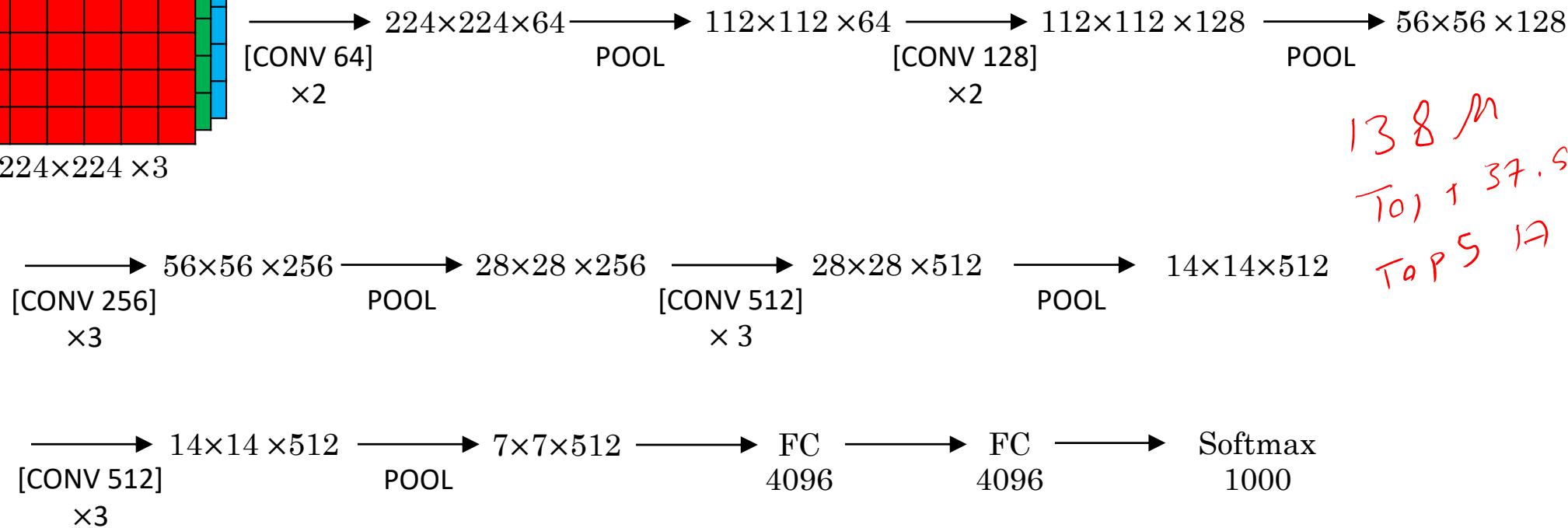
[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

VGG - 16 ^{VGG-19}

CONV = 3×3 filter, s = 1, same



MAX-POOL = 2×2 , s = 2



ResNet

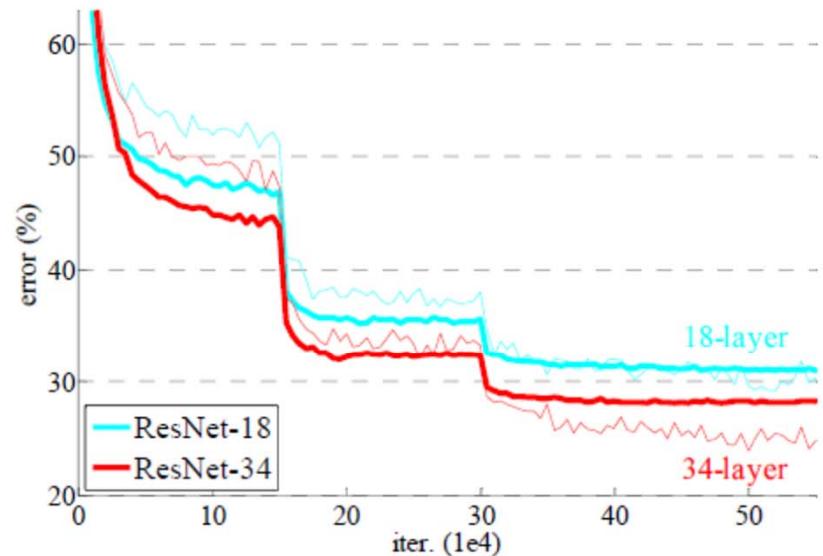
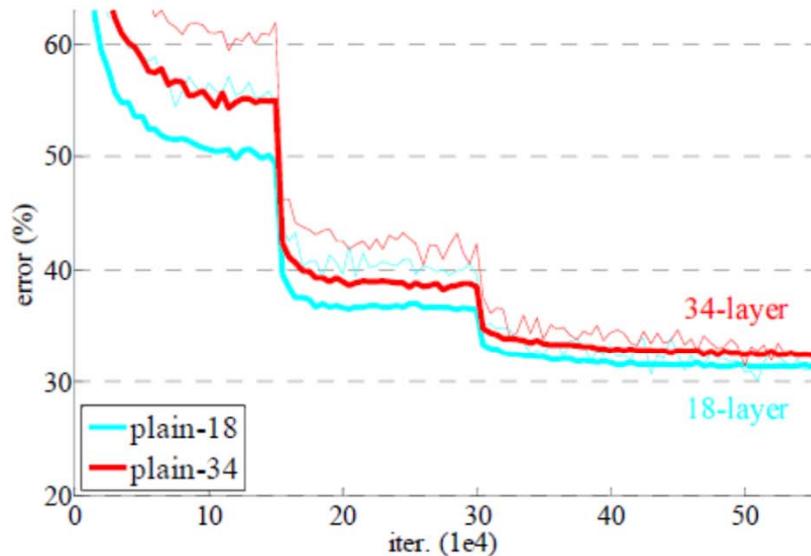
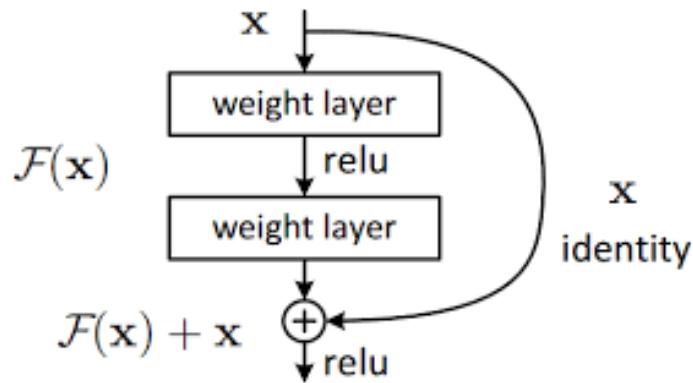


Figure 4. Training on ImageNet. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Residual Networks(ResNets)

- Residual Block



$$z^{[1]} = w^{[1]}x + b^{[1]}$$

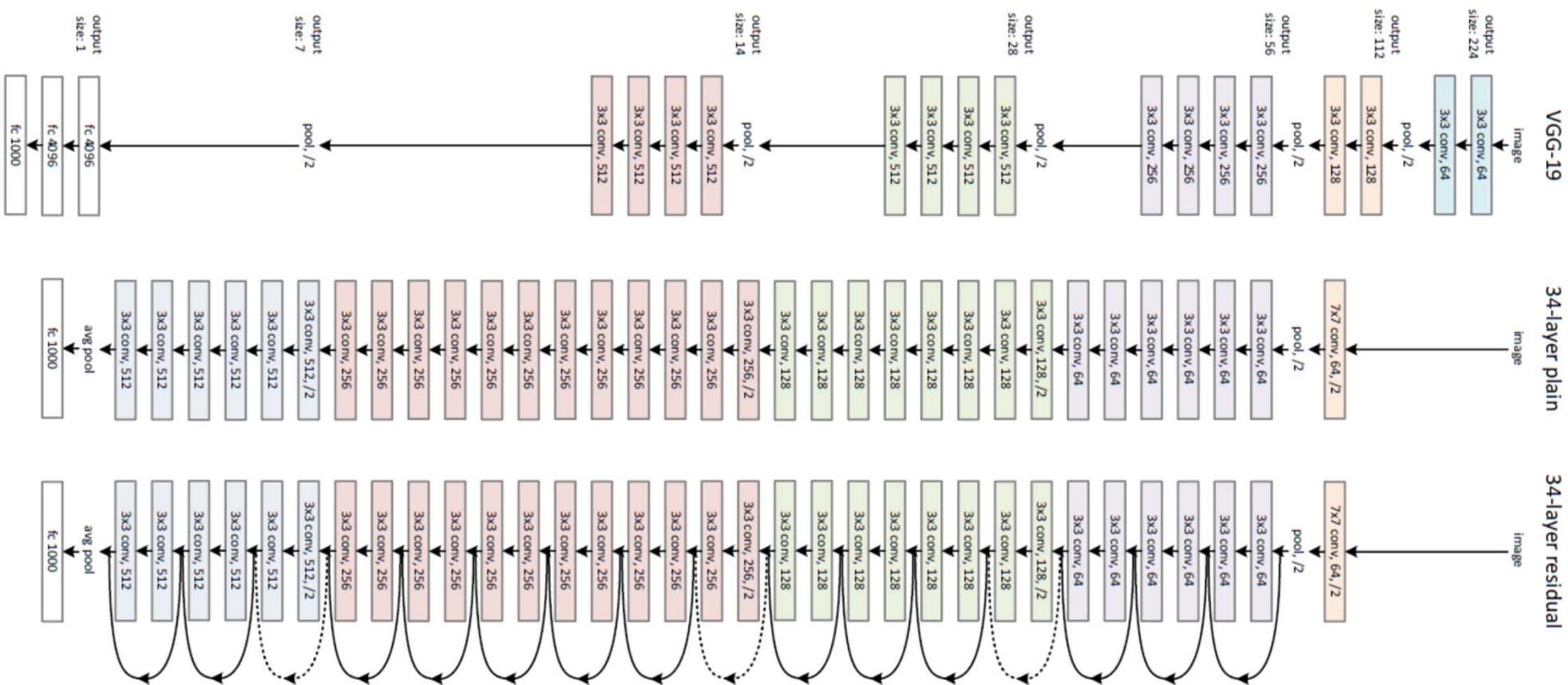
$$a^{[1]} = g(z^{[1]})$$

$$z^{[2]} = w^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = g(z^{[2]} + x)$$

$$a^{[2]} = g(z^{[2]} + w_s x)$$

ResNet



ResNet

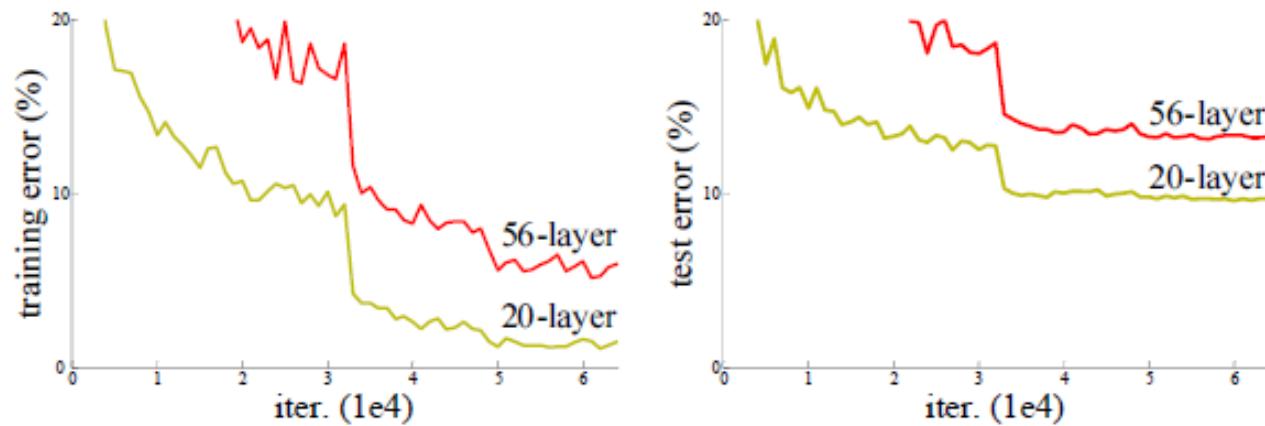
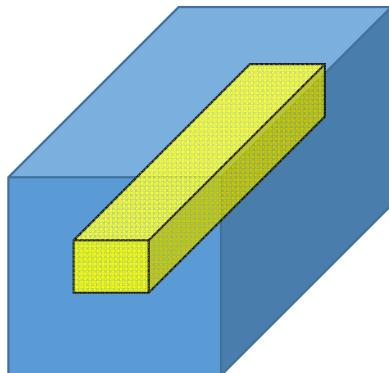


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Why does a 1×1 convolution do?

1	2	3	6	5	8
3	5	5	1	3	4
2	1	3	4	9	3
4	7	8	5	7	9
1	5	3	7	4	8
5	4	9	8	3	5

6×6



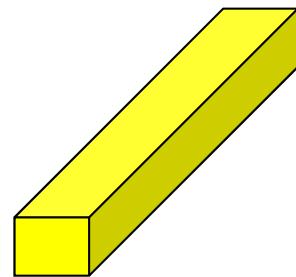
$6 \times 6 \times 192$

*

2

=

*

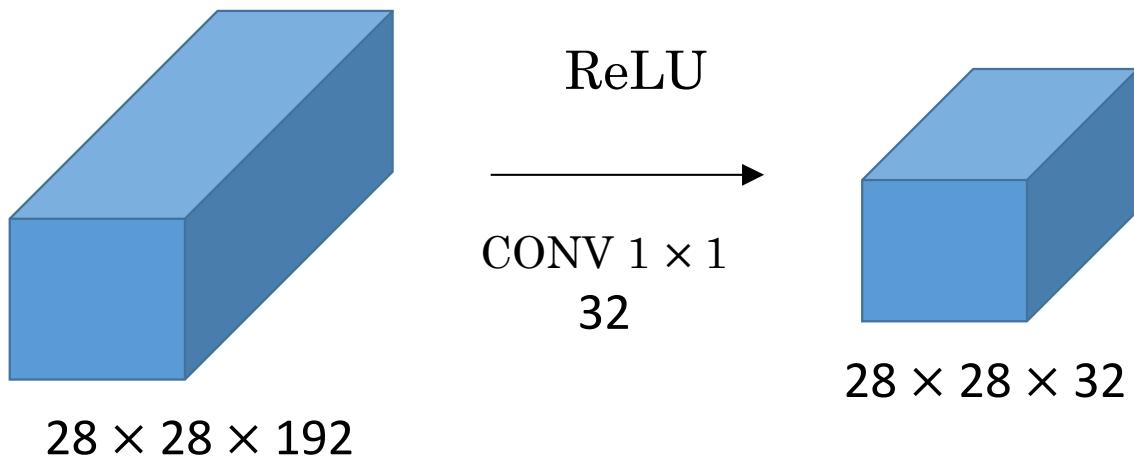


=

$6 \times 6 \times \# \text{ filters}$

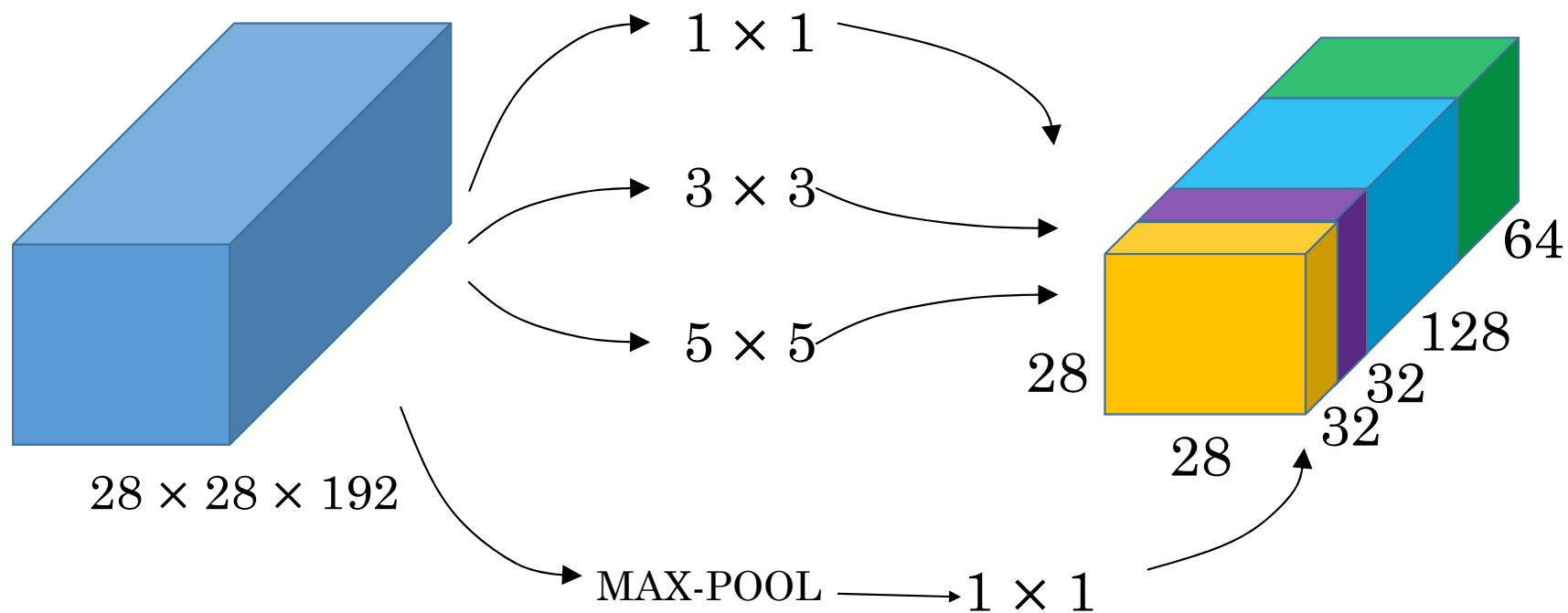
[Lin et al., 2013. Network in network]

Using 1×1 convolutions



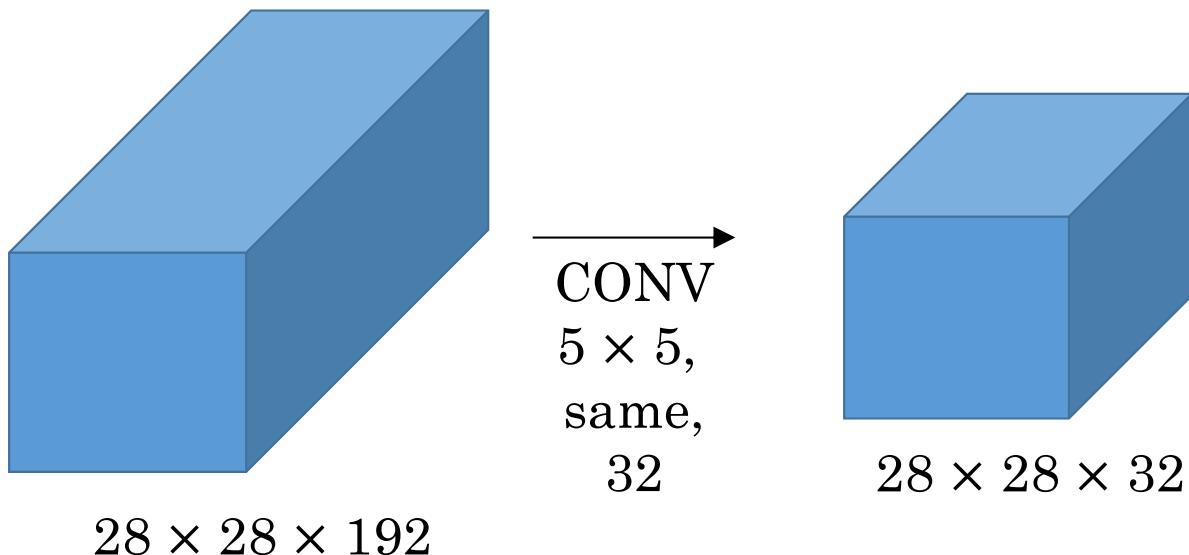
[Lin et al., 2013. Network in network]

Motivation for inception network



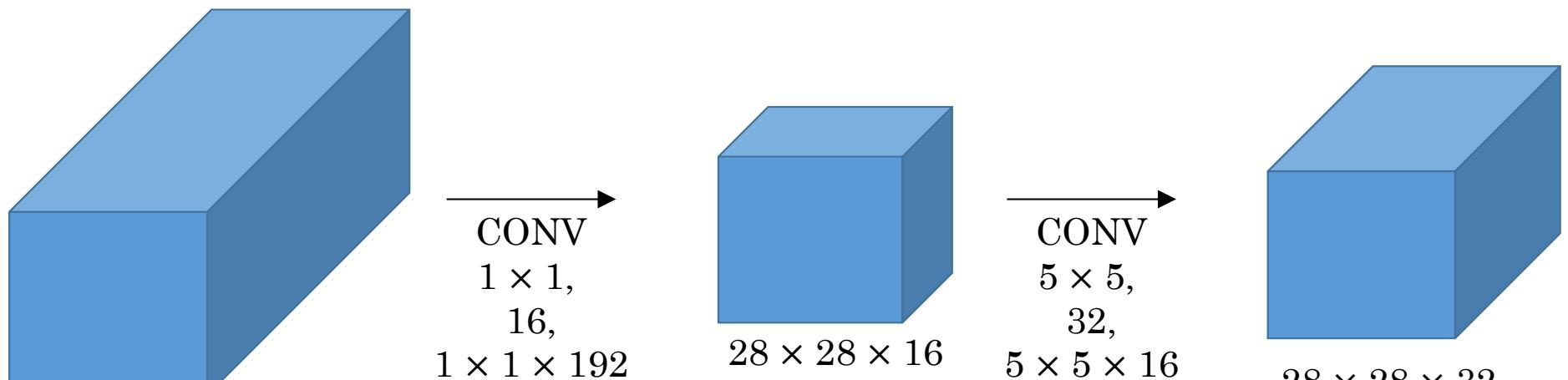
[Szegedy et al. 2014. Going deeper with convolutions]

The problem of computational cost



$$28 \times 28 \times 32 \times 5 \times 5 \times 192 = 120m$$

Using 1×1 convolution



$28 \times 28 \times 192$

$28 \times 28 \times 16 \times 192 = 2.4m$

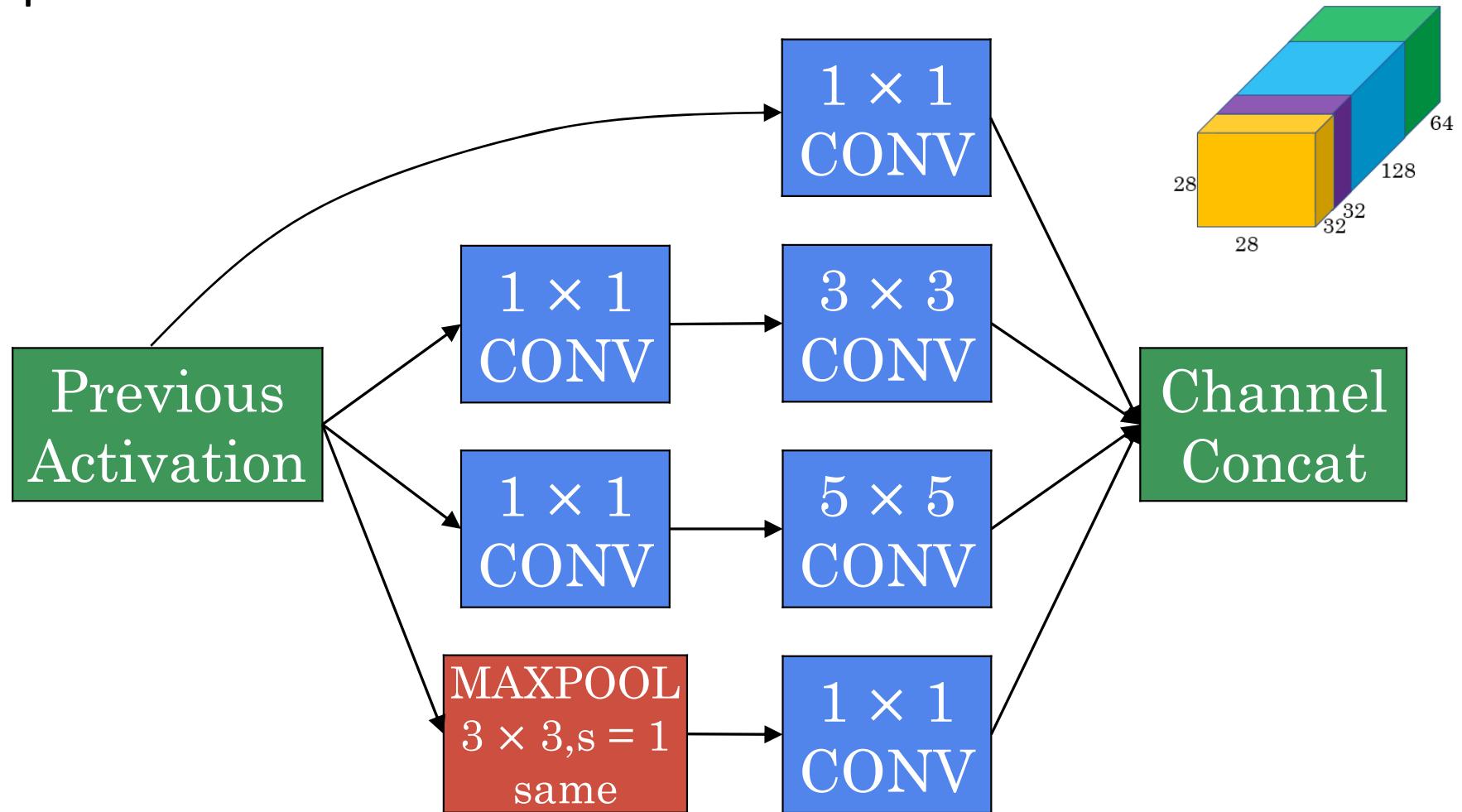
$\xrightarrow{\text{CONV}}$
 $5 \times 5, 32, 5 \times 5 \times 16$

$28 \times 28 \times 32$

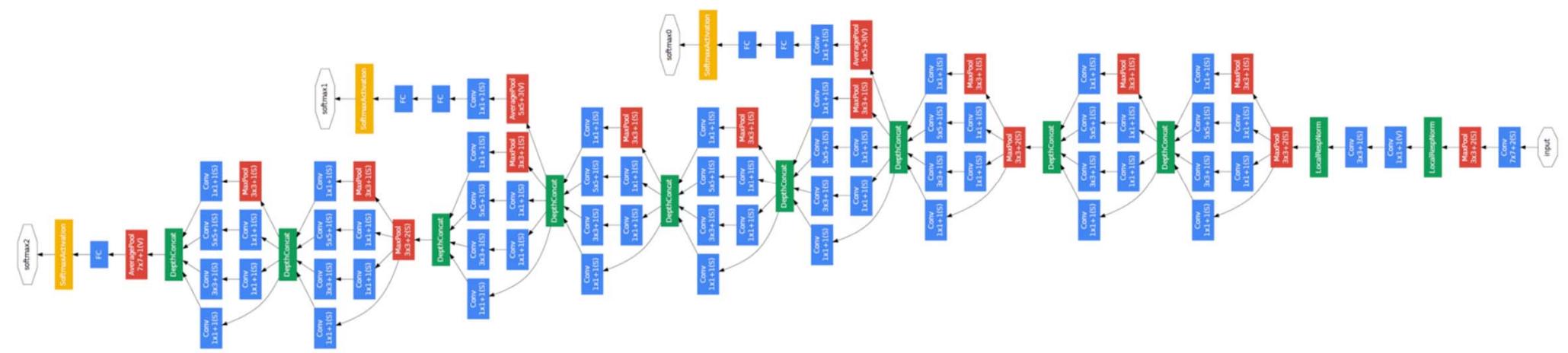
$28 \times 28 \times 32 \times 5 \times 5 \times 16 = 10m$

Total computational cost = $10m + 2.4m = 12.4m$

Inception module



GoogLeNet



[Szegedy et al. 2014. Going deeper with convolutions]

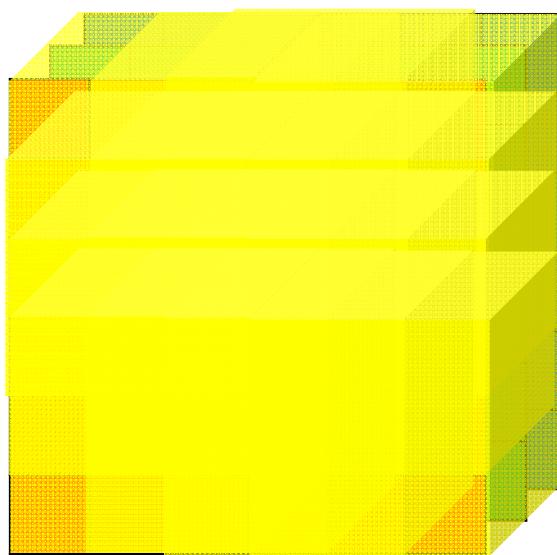
Motivation for MobileNets

- Low computational cost at deployment
- Useful for mobile and embedded vision applications
- Key idea: Normal vs. depthwise-separable convolutions

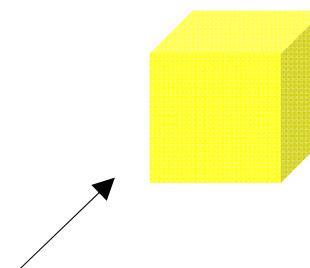
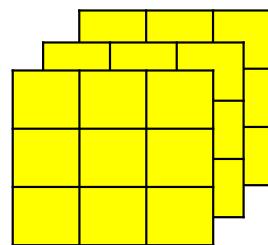


[Howard et al. 2017, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications]

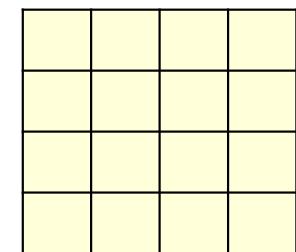
Normal Convolution



*

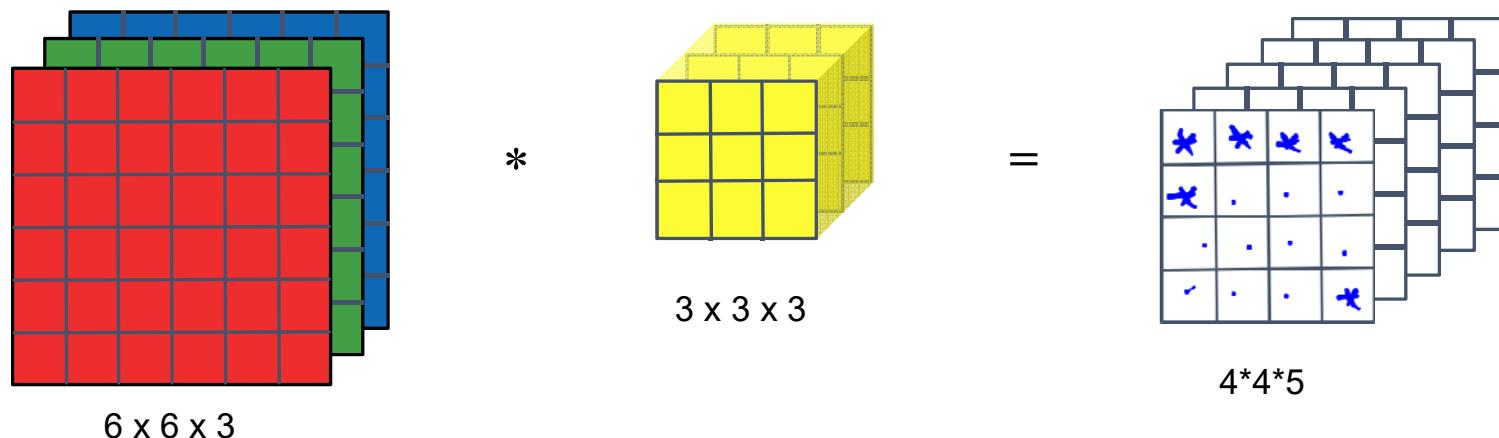


=



4×4

Normal Convolution

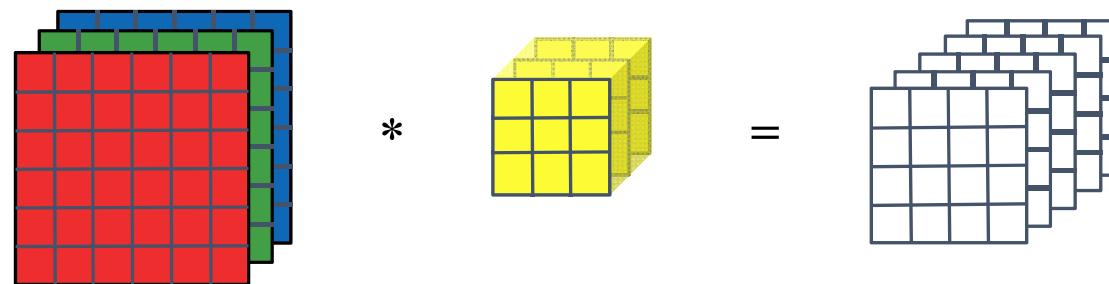


Computational cost = #filter params x # filter positions x # of filters

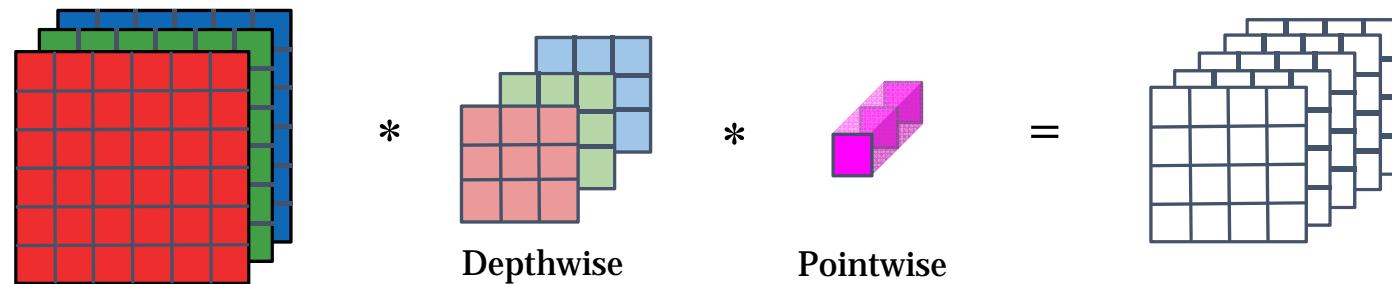
$$2160 = 4 * 4 * 5 * 3 * 3 * 3$$

Depthwise Separable Convolution

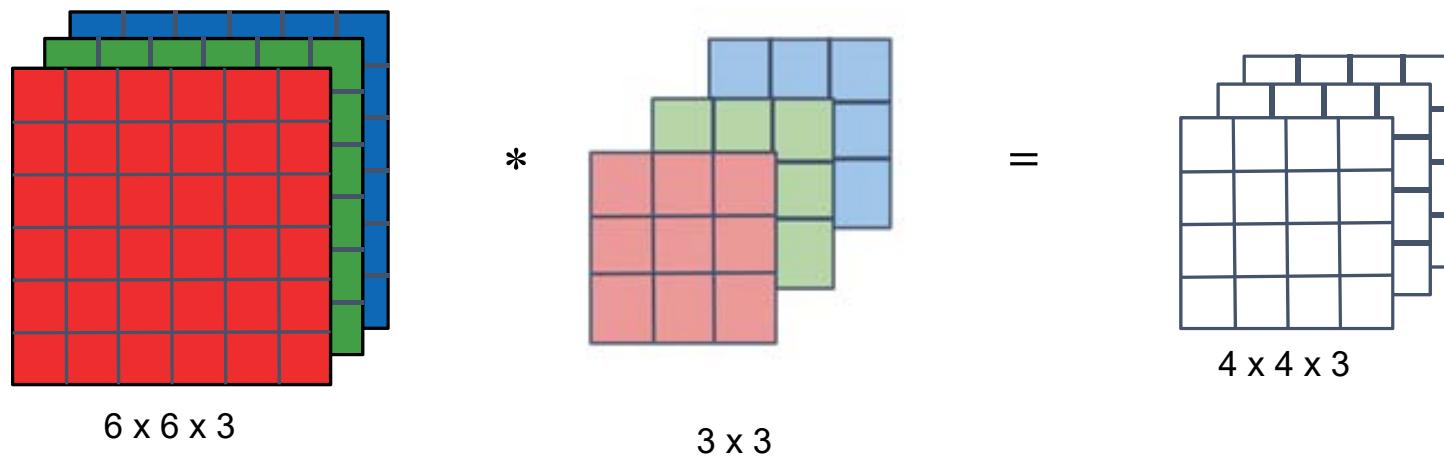
Normal Convolution



Depthwise Separable Convolution



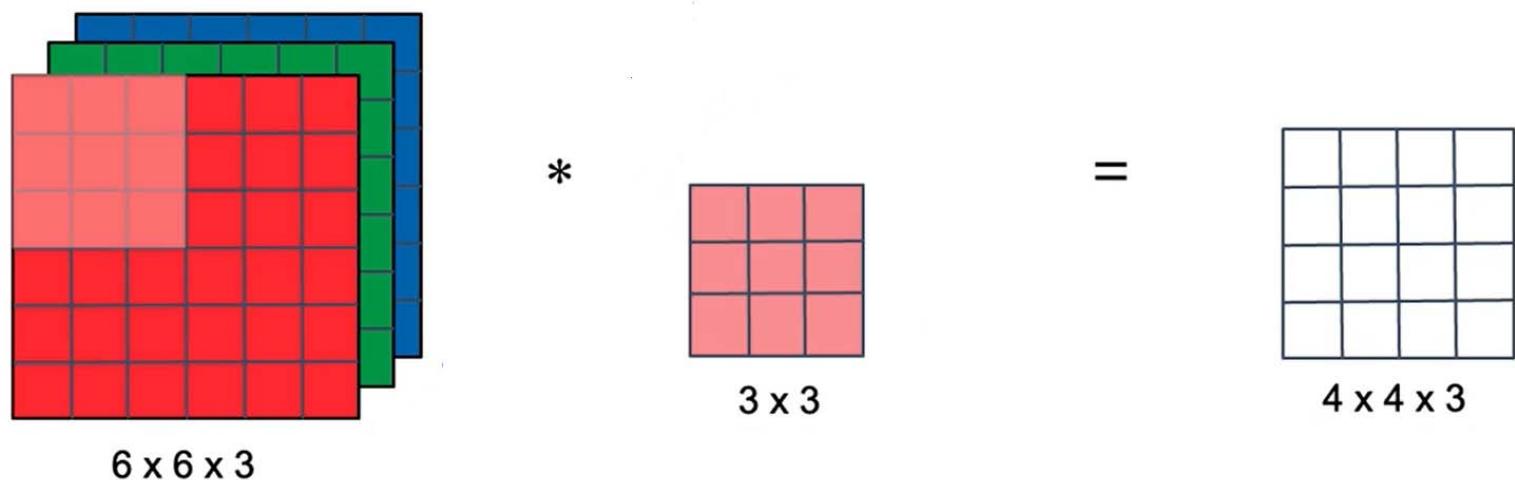
Depthwise Convolution



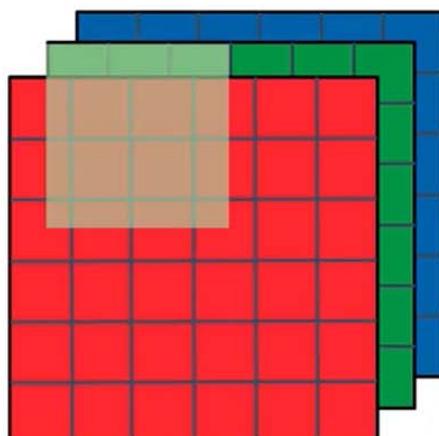
$$\text{Computational cost} = \# \text{filter params} \times \# \text{filter positions} \times \# \text{of filters}$$

$$432 = 4 * 4 * 3 * 3 * 3$$

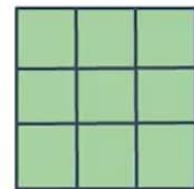
Depthwise Convolution



Depthwise Convolution

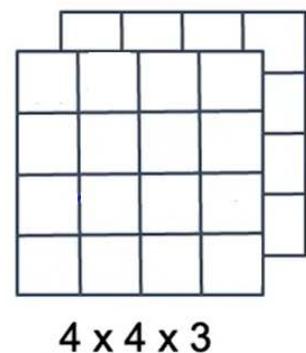


*



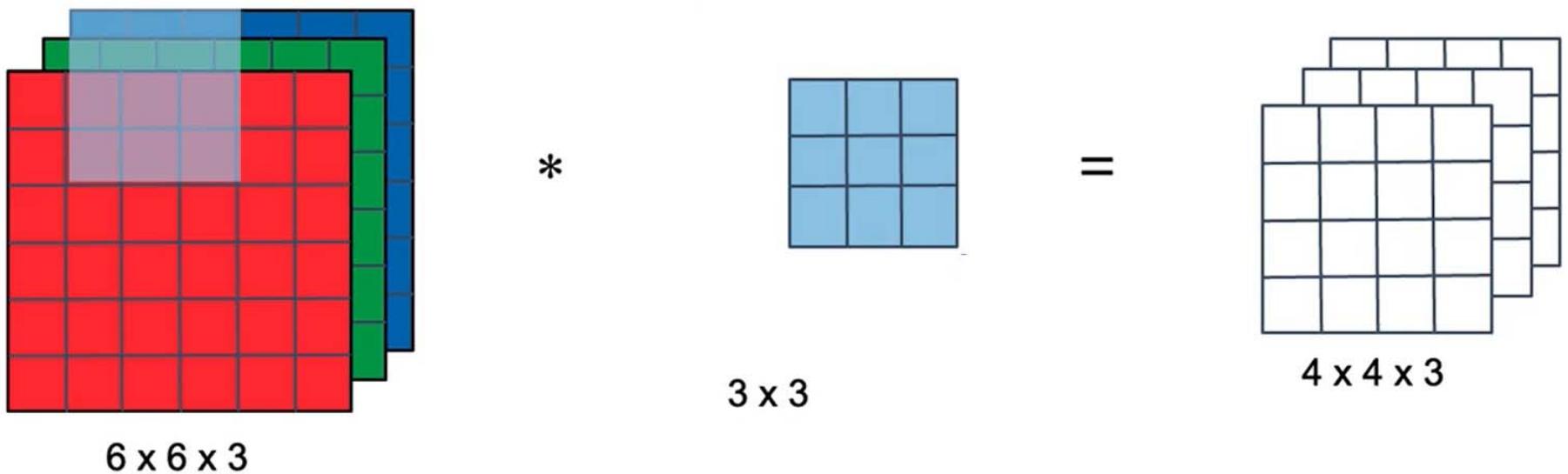
3×3

=



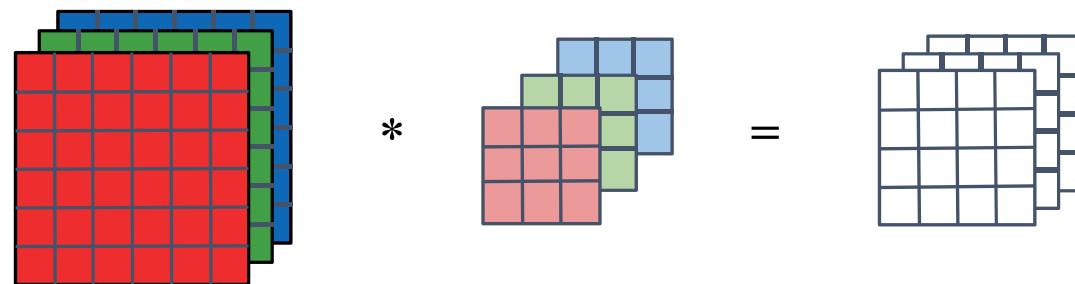
$4 \times 4 \times 3$

Depthwise Convolution

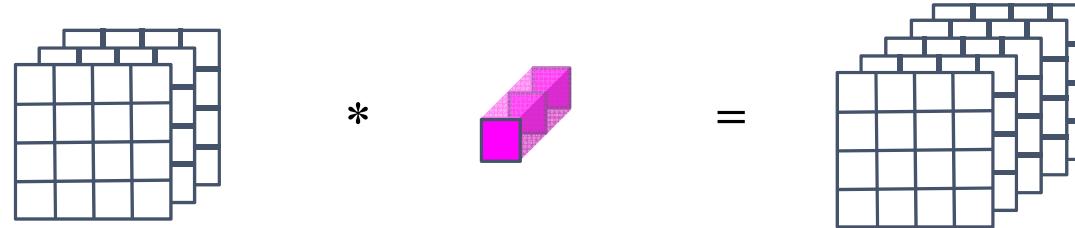


Depthwise Separable Convolution

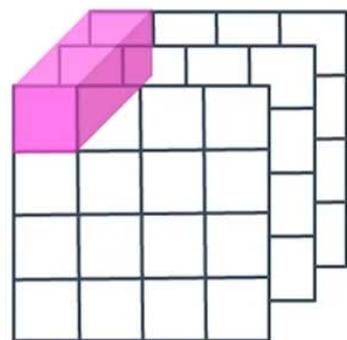
Depthwise Convolution



Pointwise Convolution



Pointwise Convolution



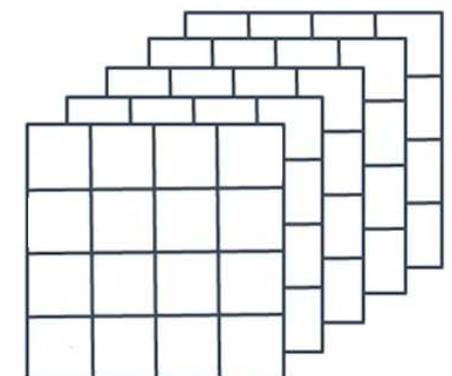
$4 \times 4 \times 3$

*



$1 \times 1 \times 3$

=



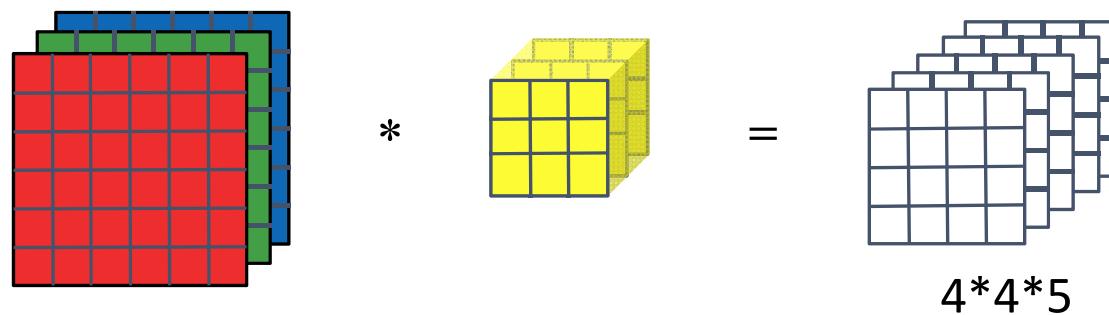
$4 \times 4 \times 5$

Computational cost = #filter params \times # filter positions \times # of filters

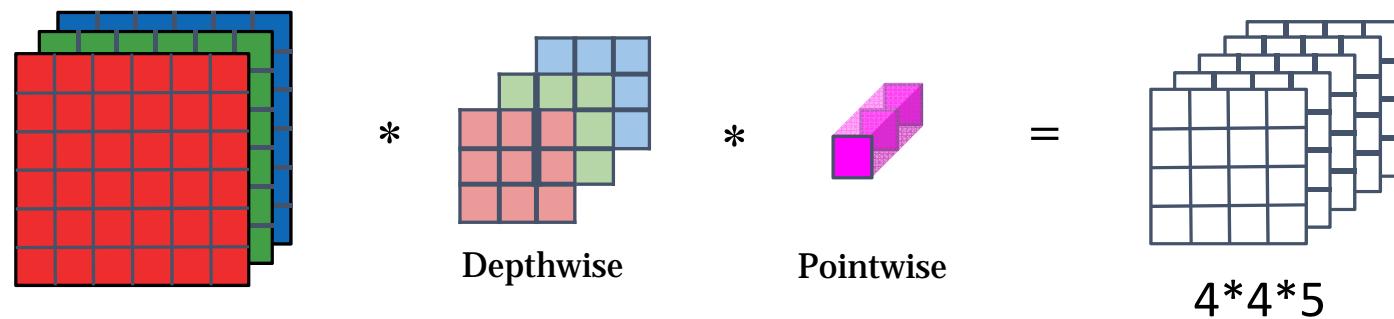
$$240 = 4 * 4 * 5 * 1 * 1 * 3$$

Depthwise Separable Convolution

Normal Convolution



Depthwise Separable Convolution



Cost Summary

Cost of normal convolution 2160

Cost of depthwise separable convolution

$$672 = 432 + 240$$

$$\frac{672}{2160} = 0.31$$

$$filter_{size}=f^2$$

$$\frac{output_{size} * f^2 + output_{size} * n_c}{output_{size} * f^2 * n_c} = \frac{1}{n_c} + \frac{1}{f^2} \sim \frac{1}{512} + \frac{1}{9} \sim \frac{1}{9}$$

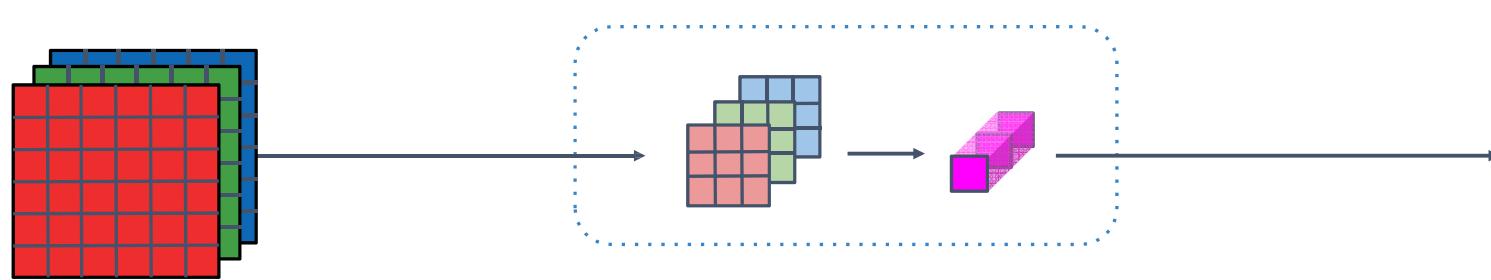
[Howard et al. 2017, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications]

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

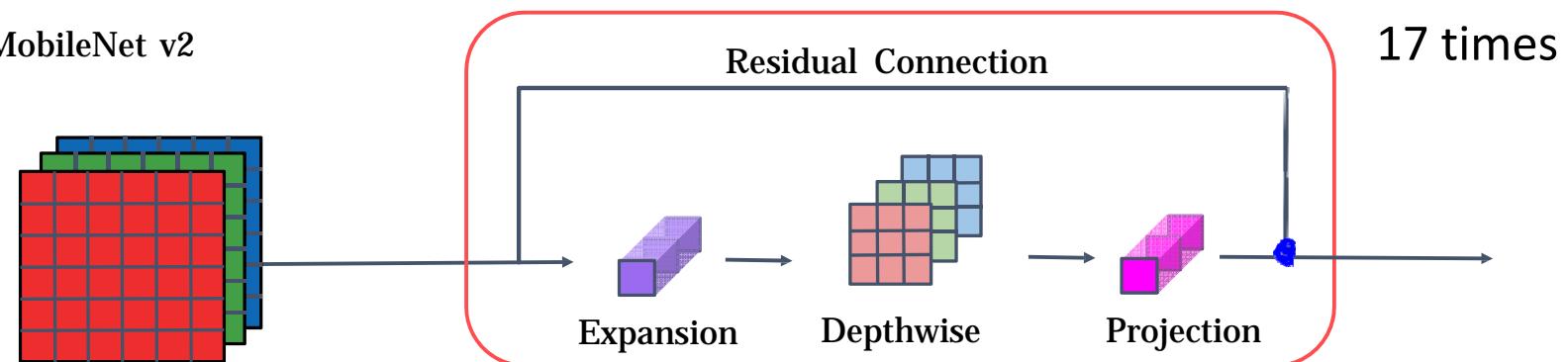
MobileNet-V2

MobileNet v1



13 times

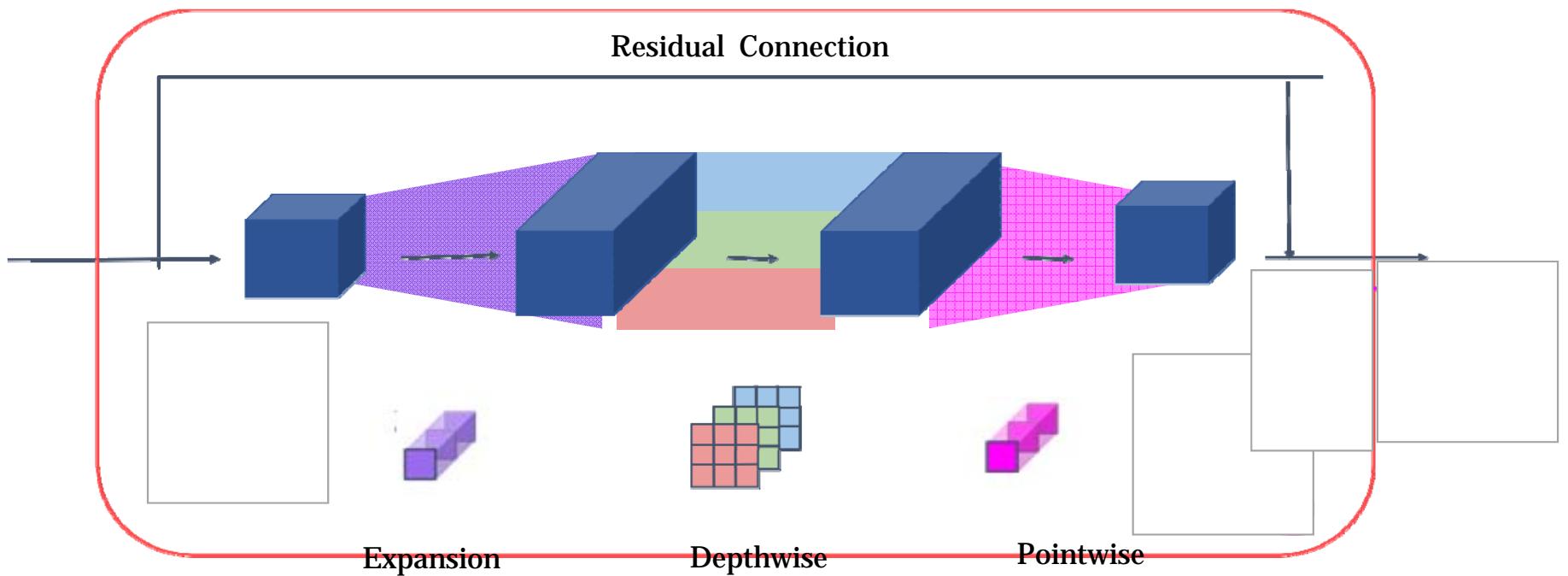
MobileNet v2



17 times

[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

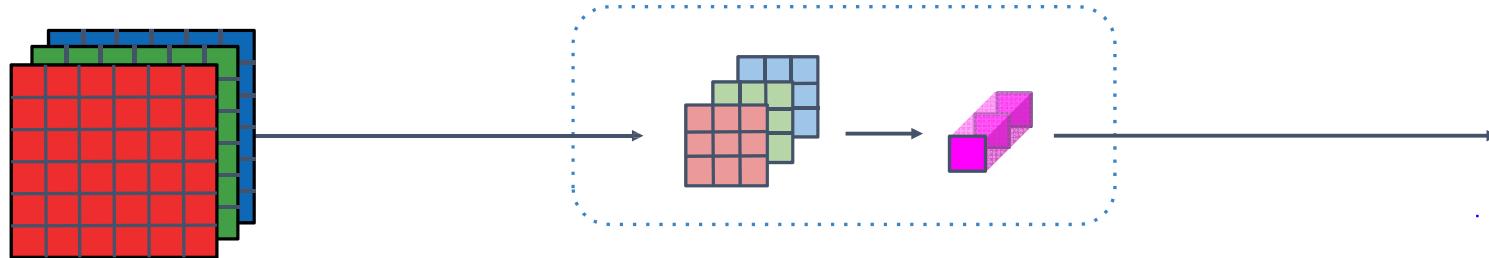
MobileNet v2 Bottleneck



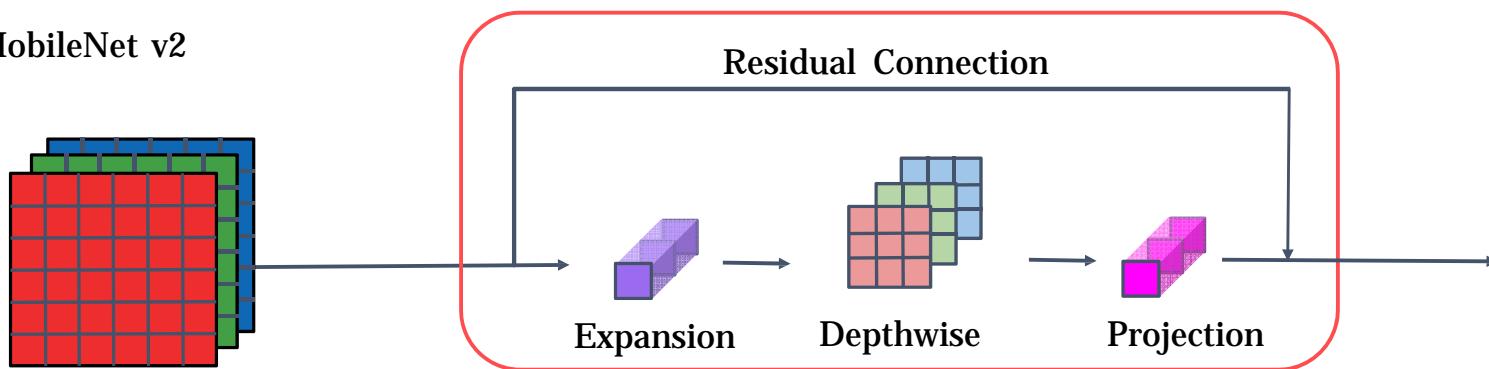
[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

MobileNet

MobileNet v1



MobileNet v2



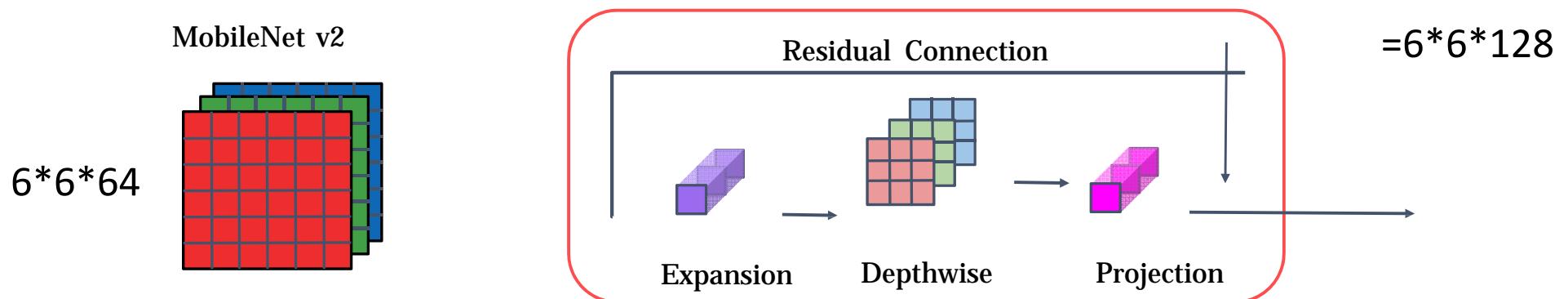
[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

MobileNet

MobileNet v1

$$2.6M = 6 \times 6 \times 128 \times 9 \times 64$$

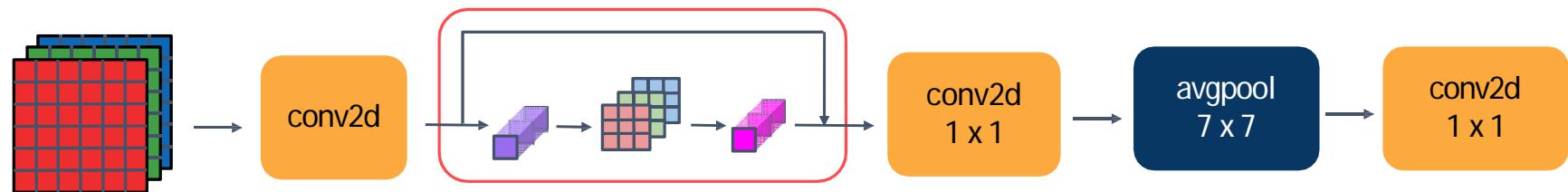
$$2.7M = 6 \times 6 \times 384 \times 64 + 6 \times 6 \times 384 \times 9 + 6 \times 6 \times 128 \times 384$$



[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

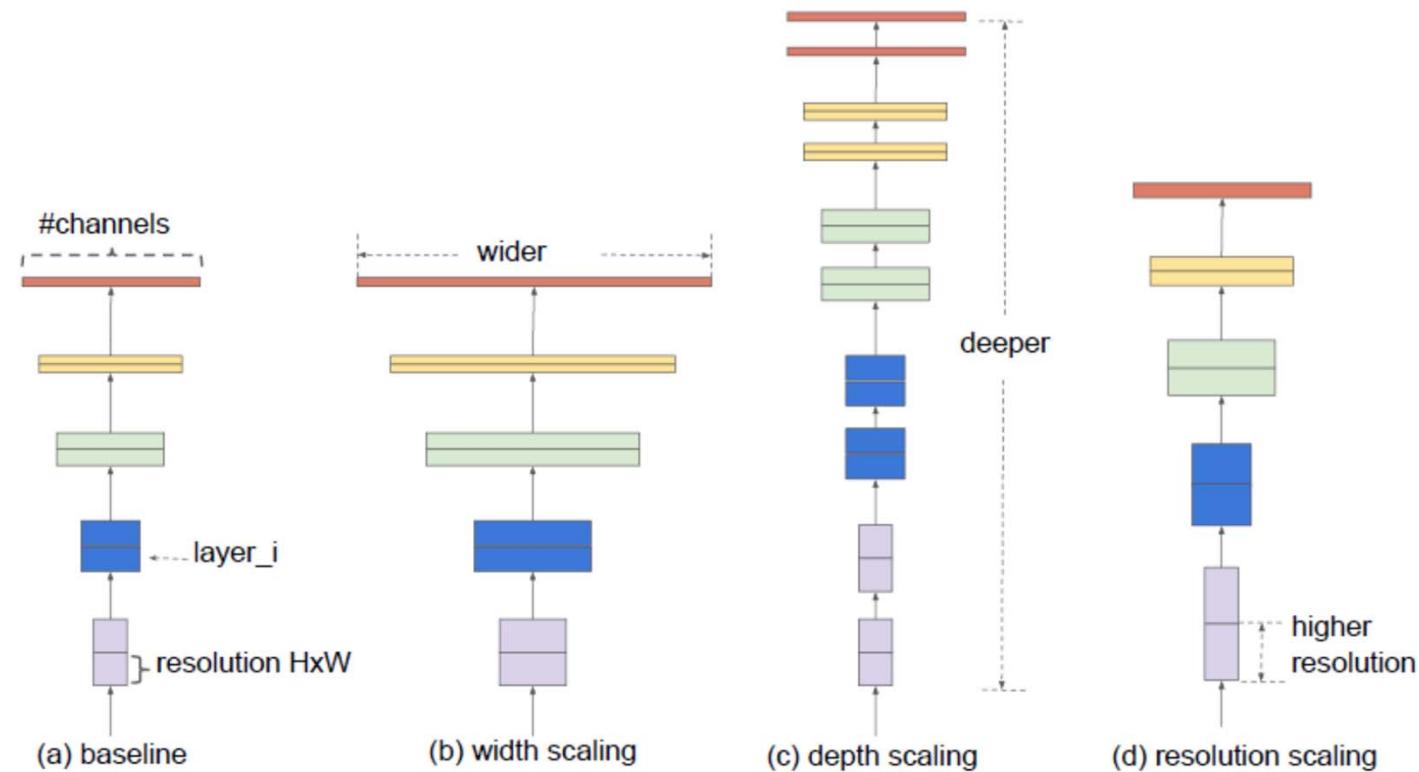
$$6 \times 6 \times (6 \times 64)$$

MobileNet v2 Full Architecture



[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

EfficientNet



[Tan and Le, 2019, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks]

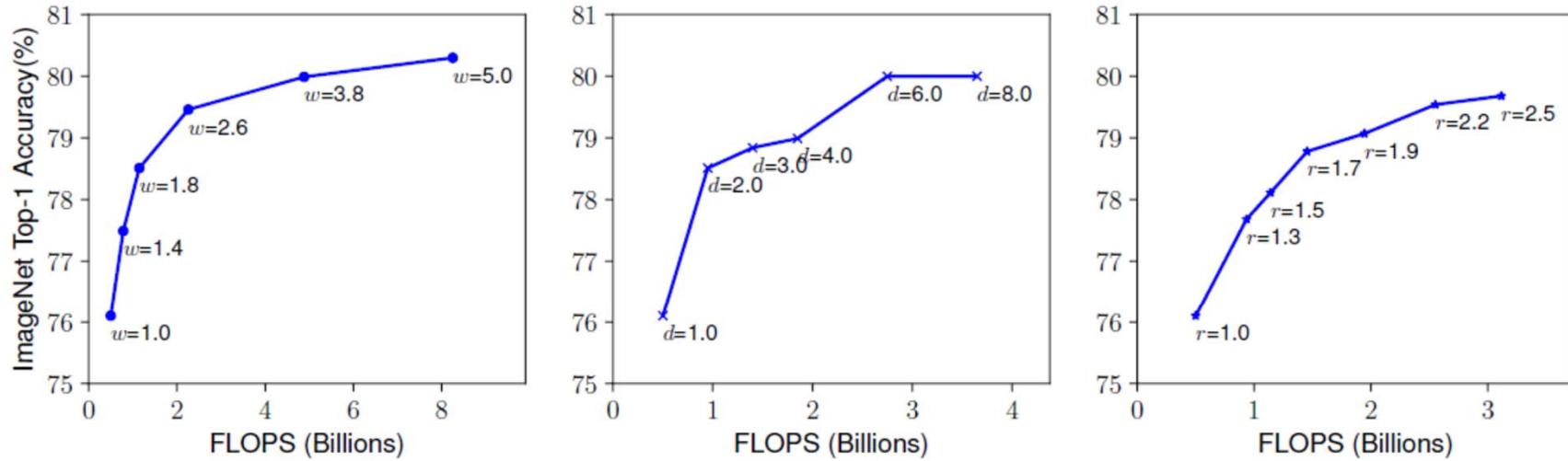


Figure 3. Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients. Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturates after reaching 80%, demonstrating the limitation of single dimension scaling. Baseline network is described in Table 1.

- **Observation 1 –** Scaling up any dimension of network width, depth, or resolution improves accuracy, but the accuracy gain diminishes for bigger models.

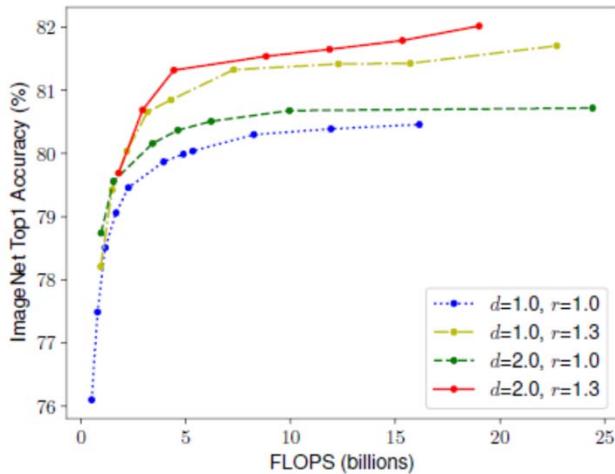


Figure 4. Scaling Network Width for Different Baseline Networks. Each dot in a line denotes a model with different width coefficient (w). All baseline networks are from Table 1. The first baseline network ($d=1.0, r=1.0$) has 18 convolutional layers with resolution 224x224, while the last baseline ($d=2.0, r=1.3$) has 36 layers with resolution 299x299.

- If we only scale network width w without changing depth ($d=1.0$) and resolution ($r=1.0$), the accuracy saturates quickly. With deeper ($d=2.0$) and higher resolution ($r=1.3$), width scaling achieves much better accuracy under the same FLOPS cost.
- **Observation 2** – In order to pursue better accuracy and efficiency, it is critical to balance all dimensions of network width, depth, and resolution during ConvNet scaling.

EfficientNet

Starting from the baseline EfficientNet-B0, we apply our compound scaling method to scale it up with two steps:

- STEP 1: we first fix $\phi = 1$, assuming twice more resources available, and do a small grid search of α, β, γ based on Equation 2 and 3. In particular, we find the best values for EfficientNet-B0 are $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$, under constraint of $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$.
- STEP 2: we then fix α, β, γ as constants and scale up baseline network with different ϕ using Equation 3, to obtain EfficientNet-B1 to B7 (Details in Table 2).

$$\begin{aligned} & \max_{d,w,r} \text{Accuracy}(\mathcal{N}(d, w, r)) \\ \text{s.t. } & \mathcal{N}(d, w, r) = \bigodot_{i=1 \dots s} \hat{\mathcal{F}}_i^{d \cdot \hat{L}_i}(X_{\langle r \cdot \hat{H}_i, r \cdot \hat{W}_i, w \cdot \hat{C}_i \rangle}) \\ & \text{Memory}(\mathcal{N}) \leq \text{target_memory} \\ & \text{FLOPS}(\mathcal{N}) \leq \text{target_flops} \end{aligned} \tag{2}$$

$$\begin{aligned} & \text{depth: } d = \alpha^\phi \\ & \text{width: } w = \beta^\phi \\ & \text{resolution: } r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{3}$$

[Tan and Le, 2019, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks]

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	77.1%	93.3%	5.3M	1x	0.39B	1x
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	79.1%	94.4%	7.8M	1x	0.70B	1x
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	80.1%	94.9%	9.2M	1x	1.0B	1x
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	81.6%	95.7%	12M	1x	1.8B	1x
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	82.9%	96.4%	19M	1x	4.2B	1x
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	83.6%	96.7%	30M	1x	9.9B	1x
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	84.0%	96.8%	43M	1x	19B	1x
EfficientNet-B7	84.3%	97.0%	66M	1x	37B	1x
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

Common augmentation method

Mirroring



Random Cropping



Rotation

Shearing

Local warping

...

Color shifting



+20,-20,+20



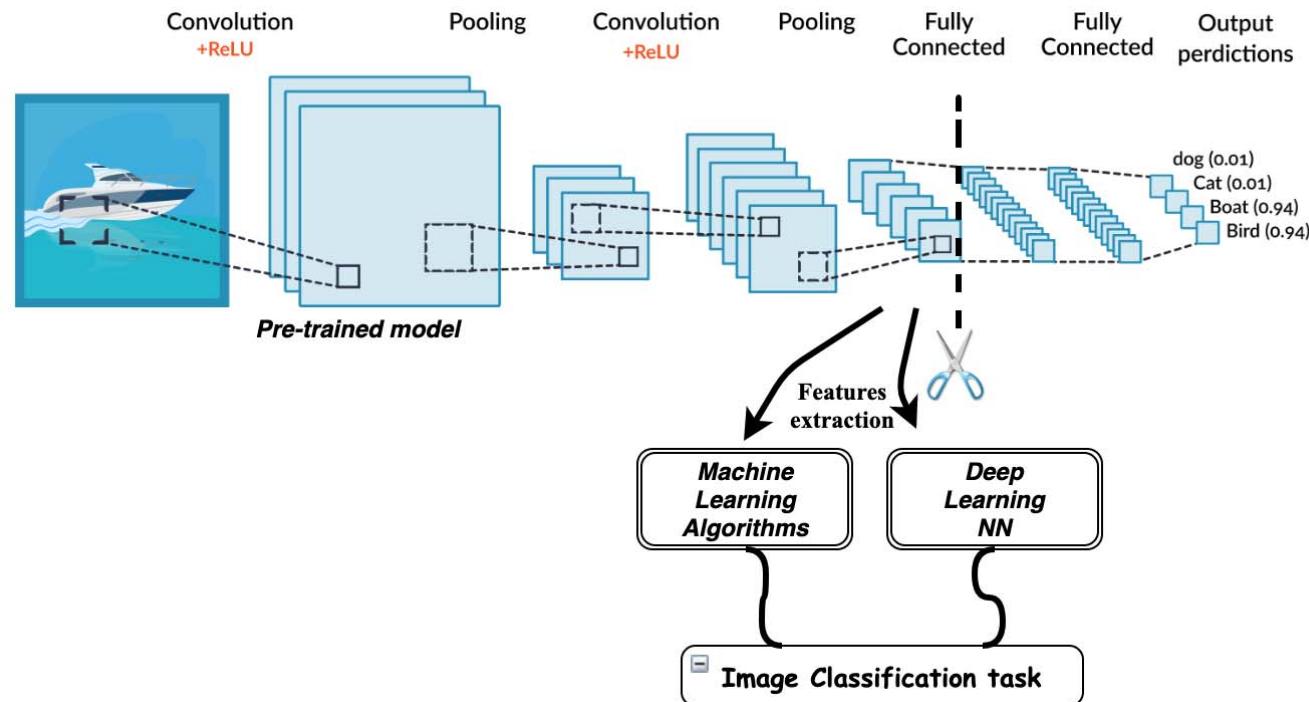
-20,+20,+20



+5.0.+50



Transfer Learning

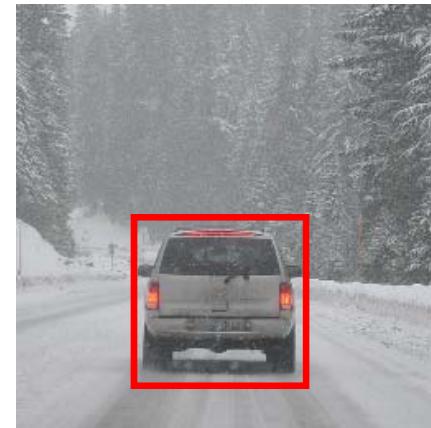


What are localization and detection?

Image classification



Classification with
localization

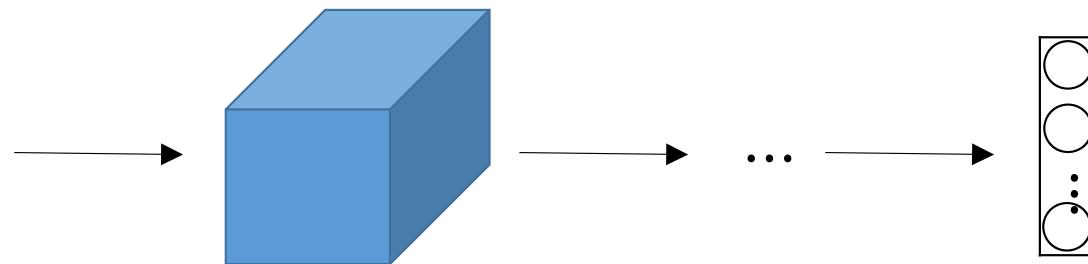
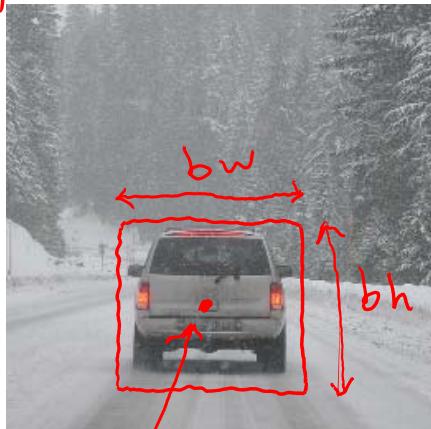


Detection



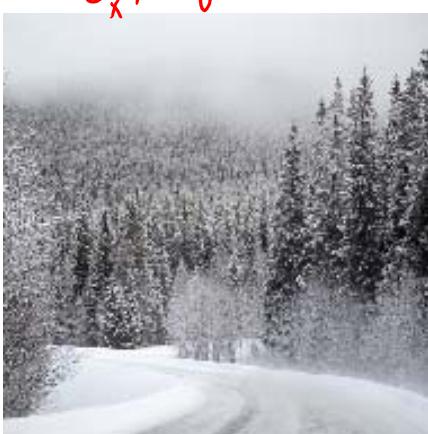
Classification with localization

(..10)



$$\begin{array}{ll} b_x = 0.5 & b_h = 0.3 \\ b_y = 0.7 & b_w = 0.4 \end{array}$$

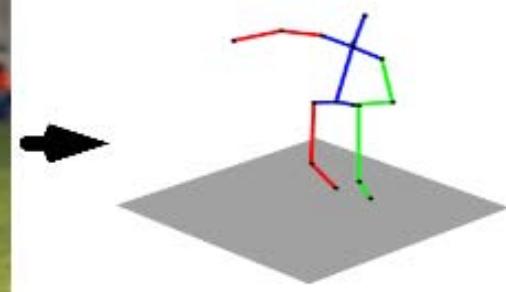
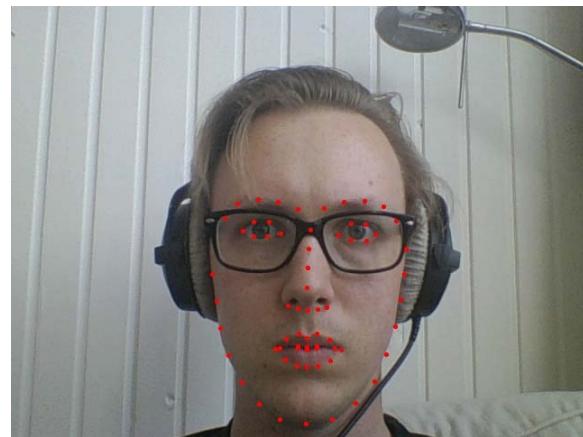
(1,1)



- 1 - pedestrian
- 2 - car
- 3 - motorcycle
- 4 - background

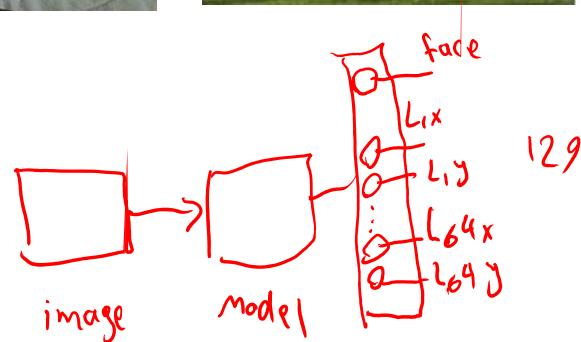


Landmark detection



b_x, b_y, b_h, b_w

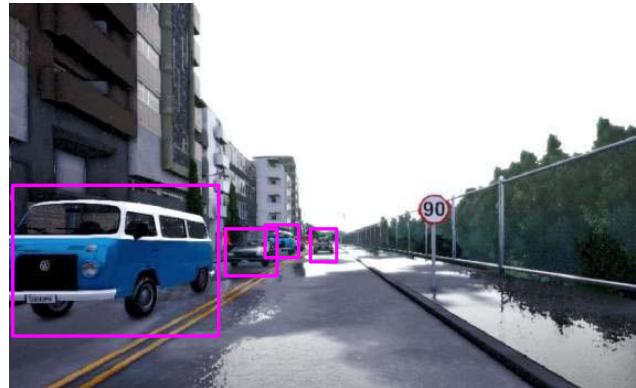
L_1x, L_1y
 L_2x, L_2y
 \vdots
 $L_{64}x, L_{64}y$



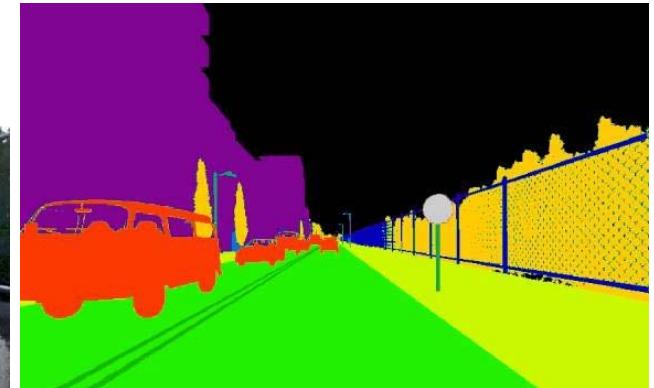
Object Detection vs. Semantic Segmentation



Input image

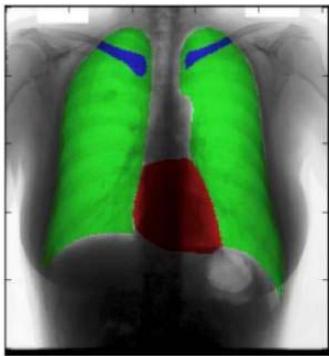


Object Detection

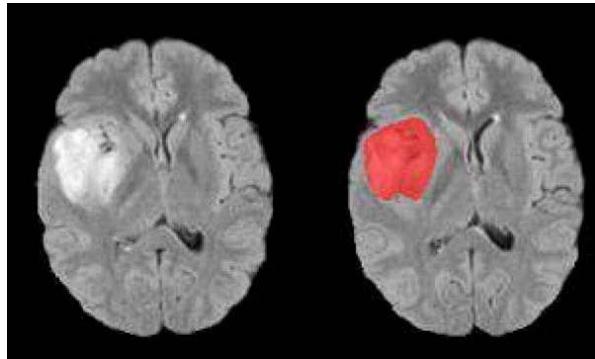


Semantic Segmentation

Motivation for U-Net



Chest X-Ray

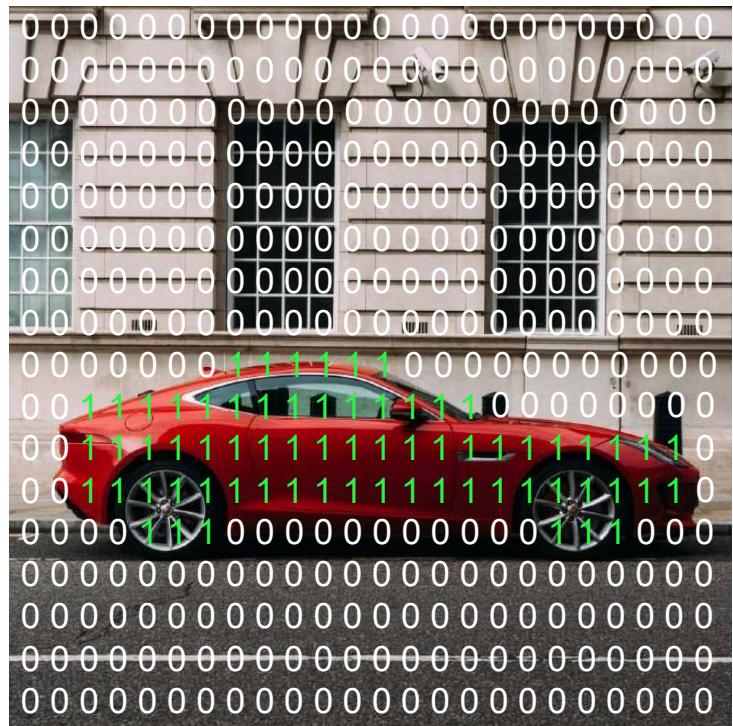


Brain MRI

[Novikov et al., 2017, Fully Convolutional Architectures for Multi-Class Segmentation in Chest Radiographs]

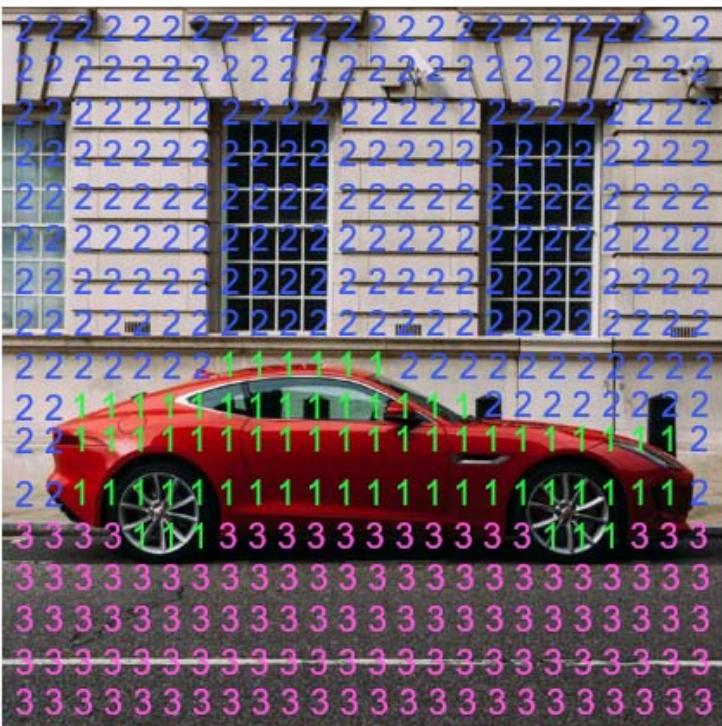
[Dong et al., 2017, Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks]

Per-pixel class labels



1. Car
0. Not Car

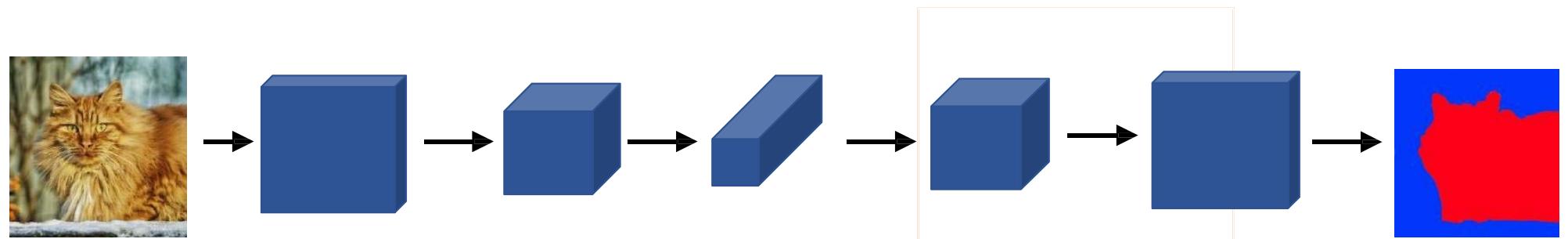
Per pixel class label



- 1. Car
 - 2. Building
 - 3. Road

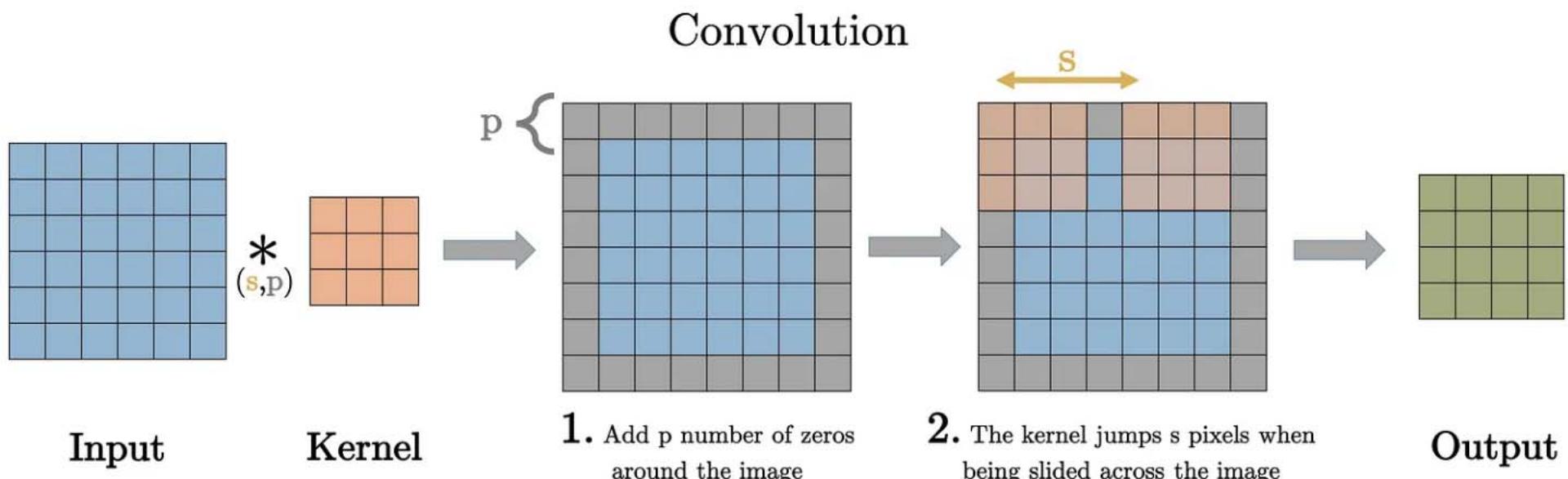
Segmentation Map

Deep Learning for Semantic Segmentation

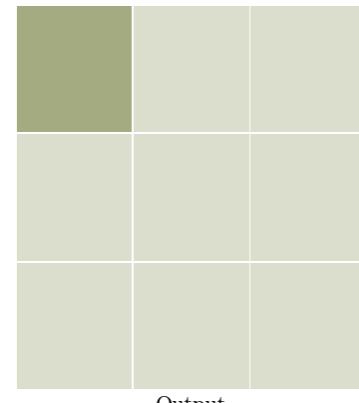
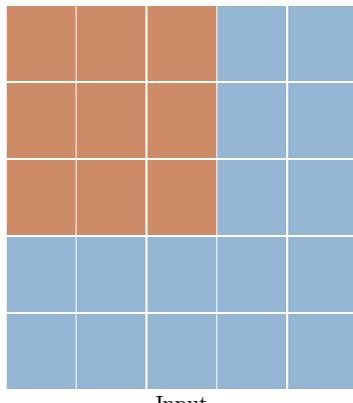


Transpose Convolution

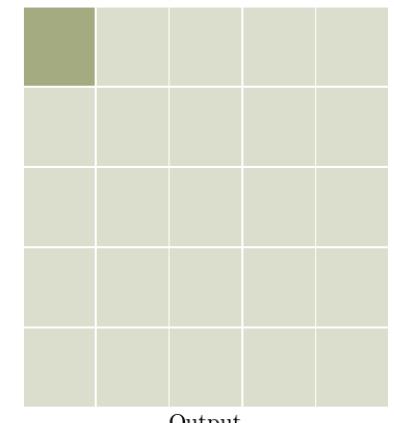
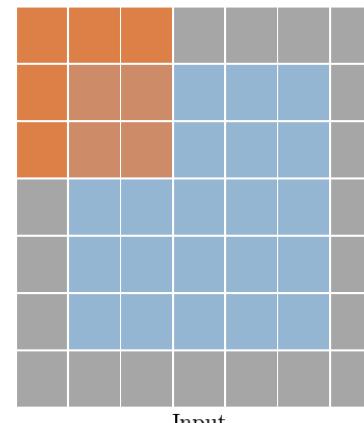
Standard Convolution:



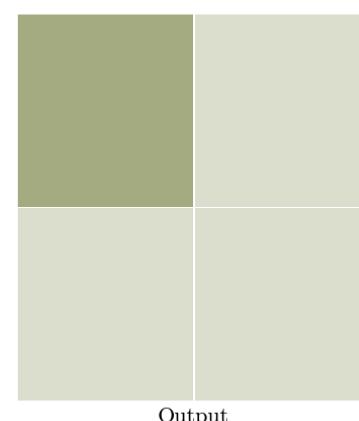
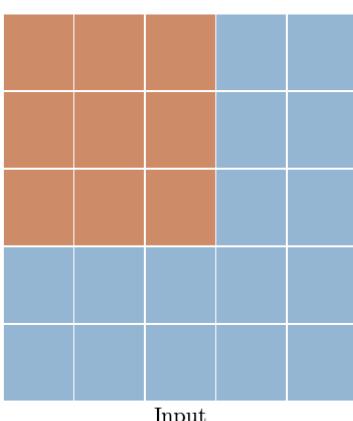
Type: conv - Stride: 1 Padding: 0



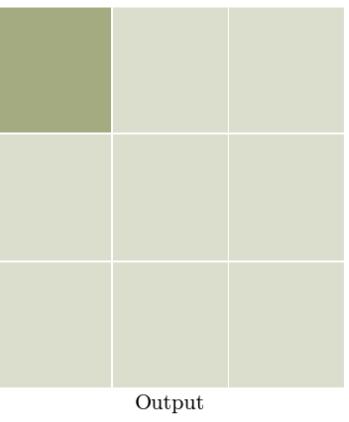
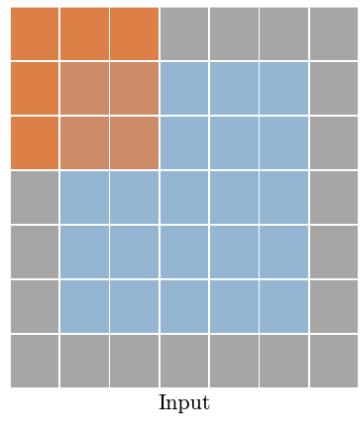
Type: conv - Stride: 1 Padding: 1



Type: conv - Stride: 2 Padding: 0



Type: conv - Stride: 2 Padding: 1

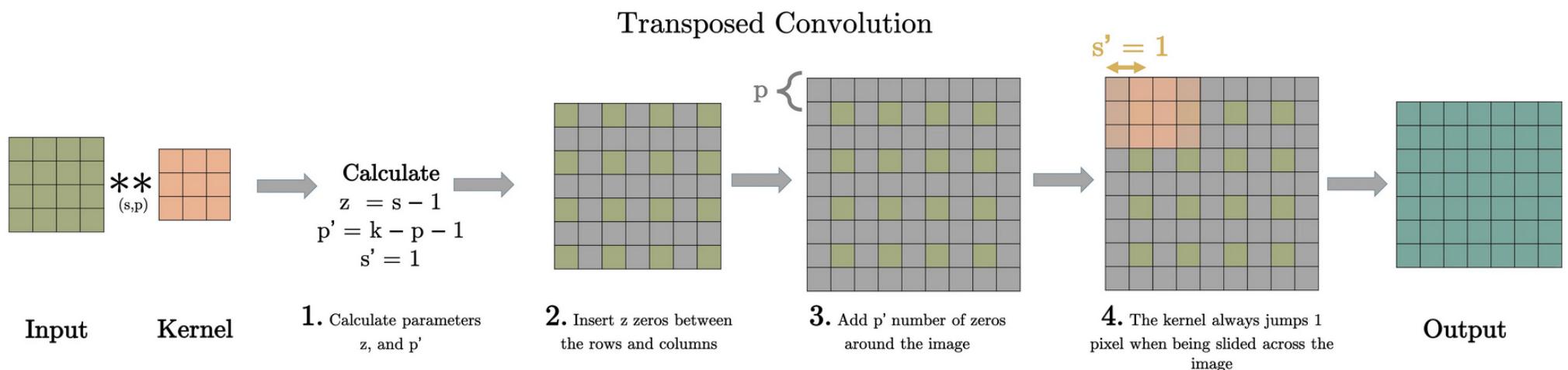


<https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11#:~:text=Transposed%20convolutions%20are%20standard%20convolutions,in%20a%20standard%20convolution%20operation.>

Transpose Convolution

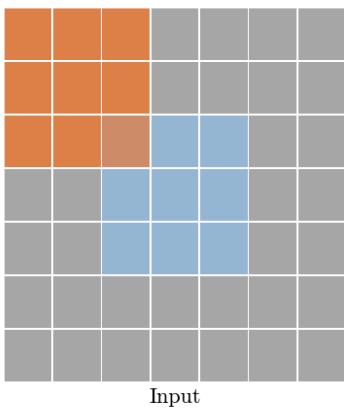
- **Step 1:** Calculate new parameters z and p'
- **Step 2:** Between each row and columns of the input, insert z number of zeros. This increases the size of the input to $(2*i-1) \times (2*i-1)$
- **Step 3:** Pad the modified input image with p' number of zeros
- **Step 4:** Carry out standard convolution on the image generated from step 3 with a stride length of 1

<https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11#:~:text=Transposed%20convolutions%20are%20standard%20convolutions,in%20a%20standard%20convolution%20operation.>

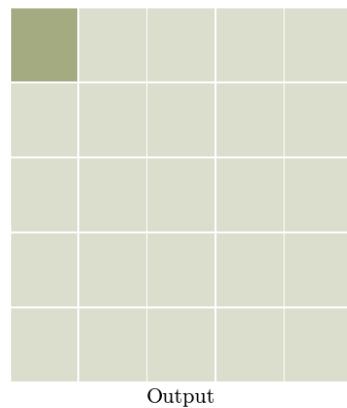


<https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11#:~:text=Transposed%20convolutions%20are%20standard%20convolutions,in%20a%20standard%20convolution%20operation.>

Type: transposed`conv - Stride: 1 Padding: 0

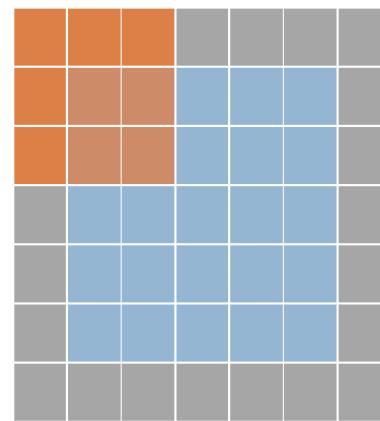


Input

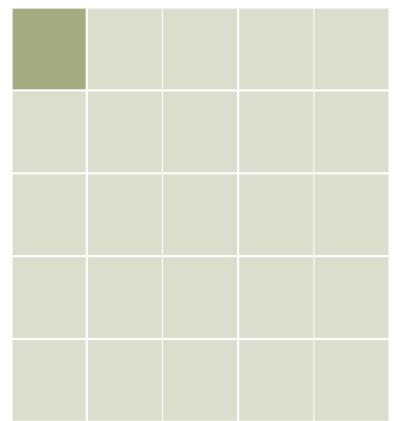


Output

Type: transposed`conv - Stride: 1 Padding: 1

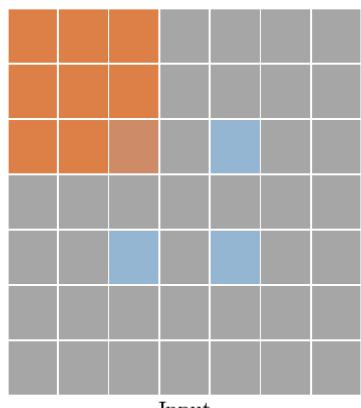


Input

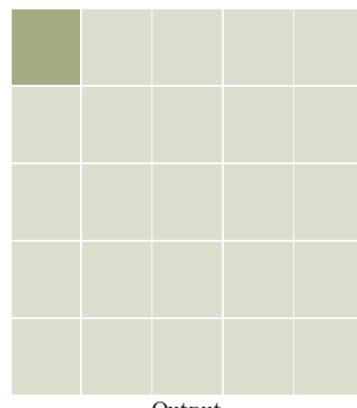


Output

Type: transposed`conv - Stride: 2 Padding: 0

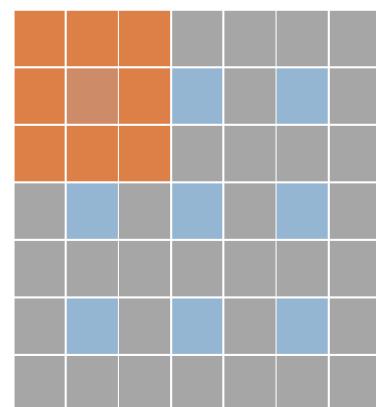


Input

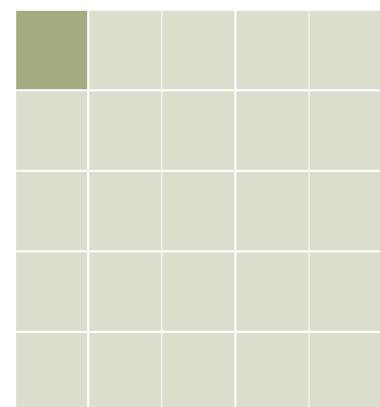


Output

Type: transposed`conv - Stride: 2 Padding: 1



Input



Output

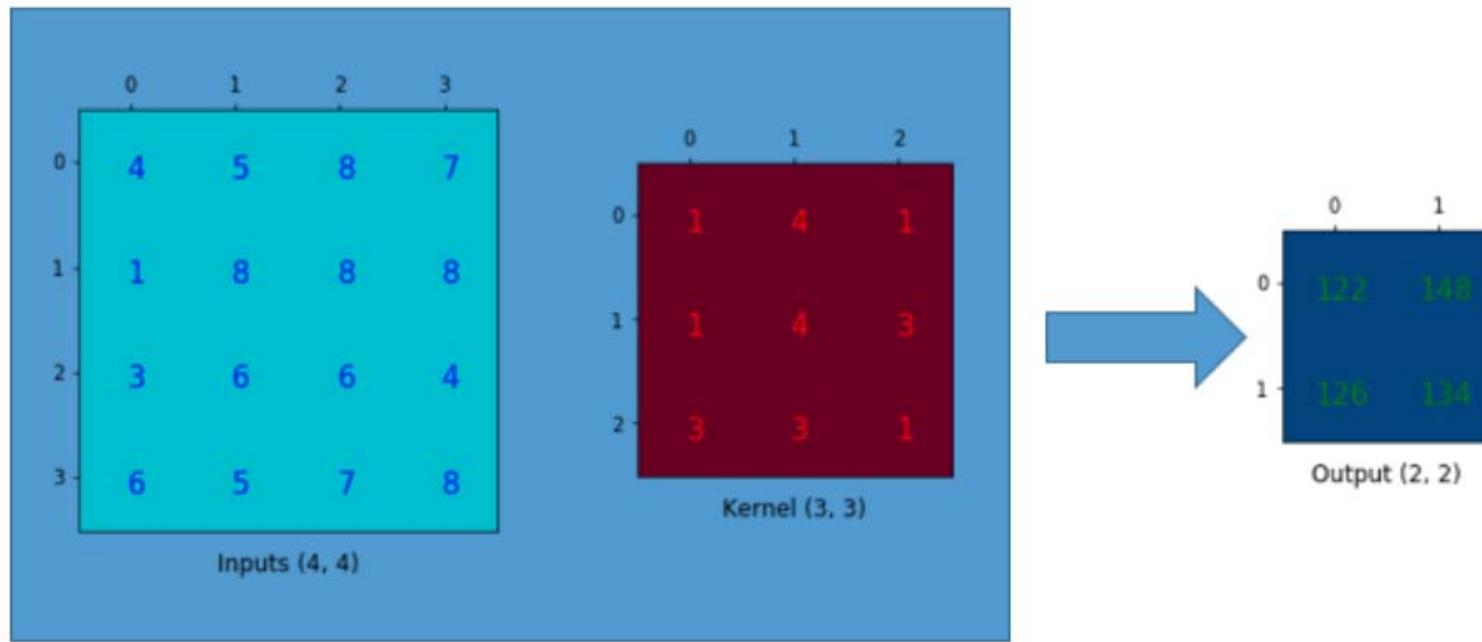
<https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11#:~:text=Transposed%20convolutions%20are%20standard%20convolutions,in%20a%20standard%20convolution%20operation.>

Convolution vs Transpose Convolution

Comparison					
Conv Type	Operation	Zero Insertions	Padding	Stride	Output Size
Standard	Downsampling	0	p	s	$(i+2p-k)/s + 1$
Transposed	Upsampling	$(s - 1)$	$(k-p-1)$	1	$(i-1)*s+k-2p$

<https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11#:~:text=Transposed%20convolutions%20are%20standard%20convolutions,in%20a%20standard%20convolution%20operation.>

Why transpose convolution?



<https://towardsdatascience.com/up-sampling-with-transposed-convolution-9ae4f2df52d0>

1	4	1	
1	4	3	
3	3	1	
122 148			
126 134			
4	5	8	7
1	8	8	8
3	6	6	4
6	5	7	8

1	4	1	
1	4	3	
3	3	1	
122 148			
126 134			
4	5	8	7
1	8	8	8
3	6	6	4
6	5	7	8

1	4	1	
1	4	3	
3	3	1	
122 148			
126 134			
4	5	8	7
1	8	8	8
3	6	6	4
6	5	7	8

1	4	1	
1	4	3	
3	3	1	
122 148			
126 134			
4	5	8	7
1	8	8	8
3	6	6	4
6	5	7	8

	0	1	2
0	1	4	1
1	1	4	3
2	3	3	1

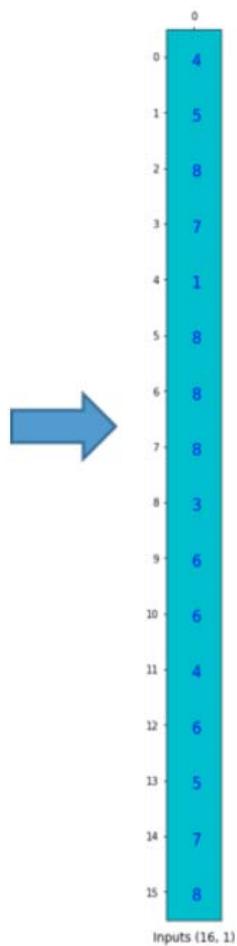
Kernal (3, 3)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	4	1	0	1	4	3	0	3	3	1	0	0	0	0	0
1	0	1	4	1	0	1	4	3	0	3	3	1	0	0	0	0
2	0	0	0	0	1	4	1	0	1	4	3	0	3	3	1	0
3	0	0	0	0	0	1	4	1	0	1	4	3	0	3	3	1

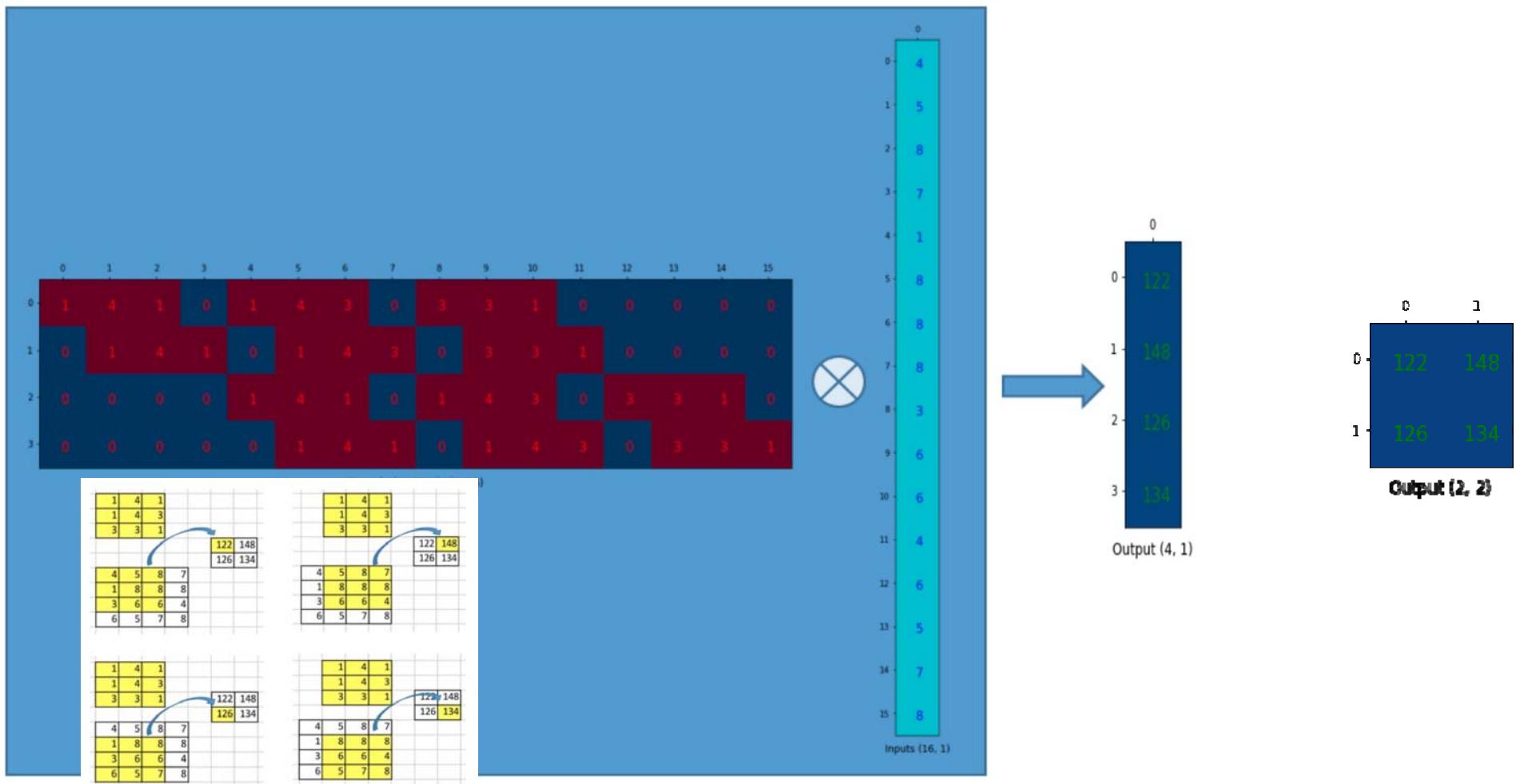
Convolution Matrix (4, 16)

	0	1	2	3
0	4	5	8	7
1	1	8	8	8
2	3	6	6	4
3	6	5	7	8

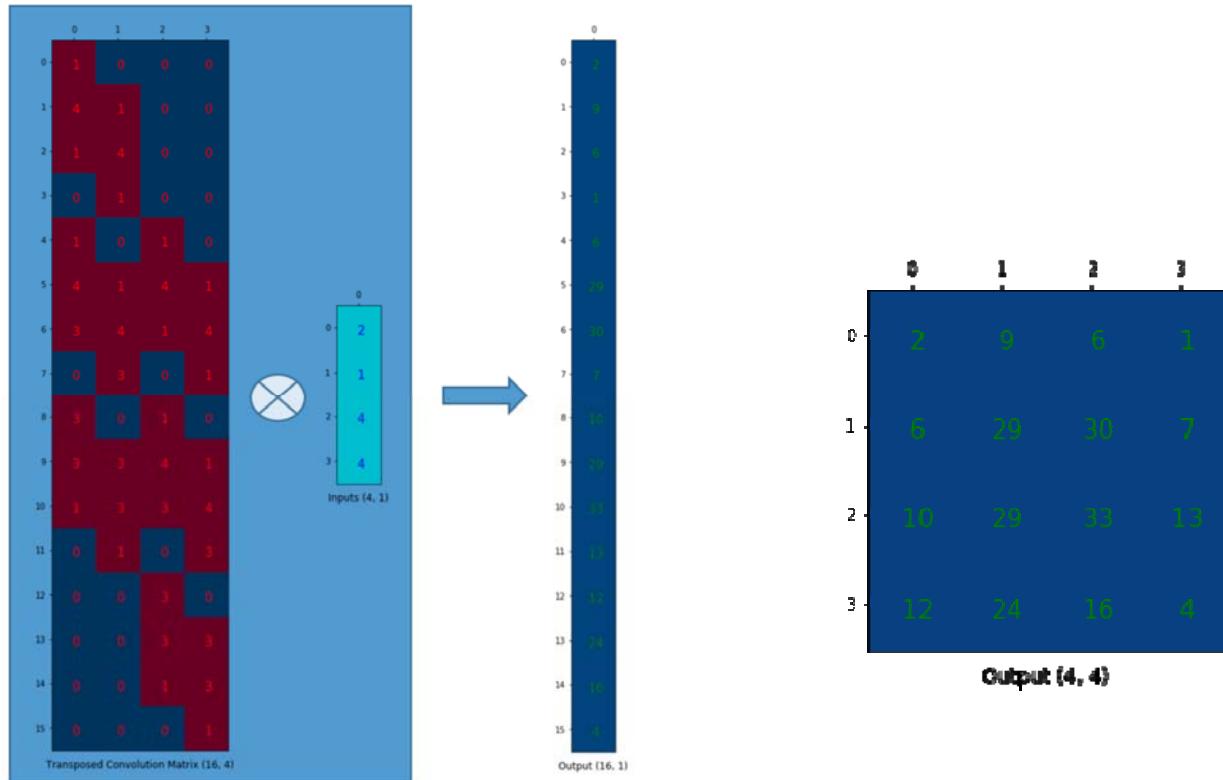
Inputs (4, 4)



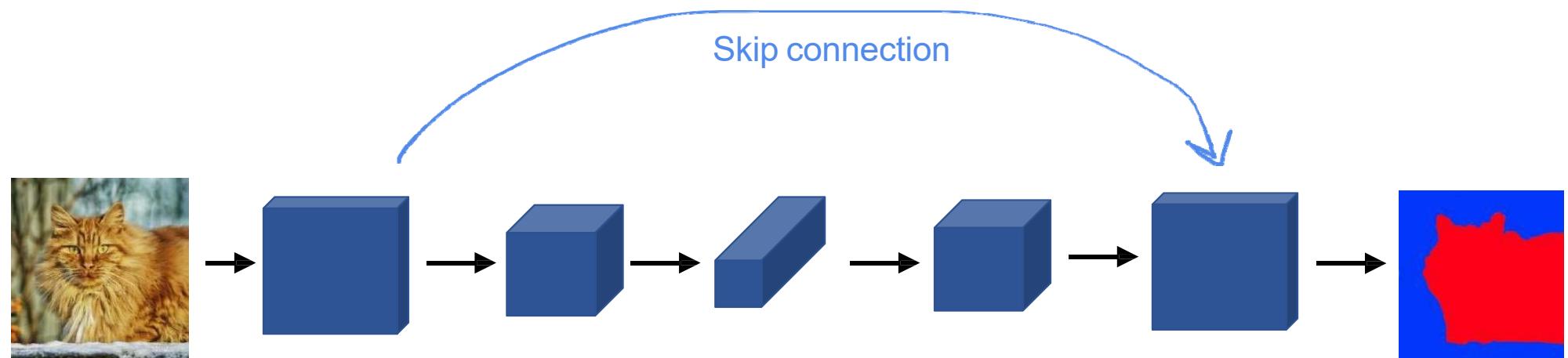
Inputs (16, 1)



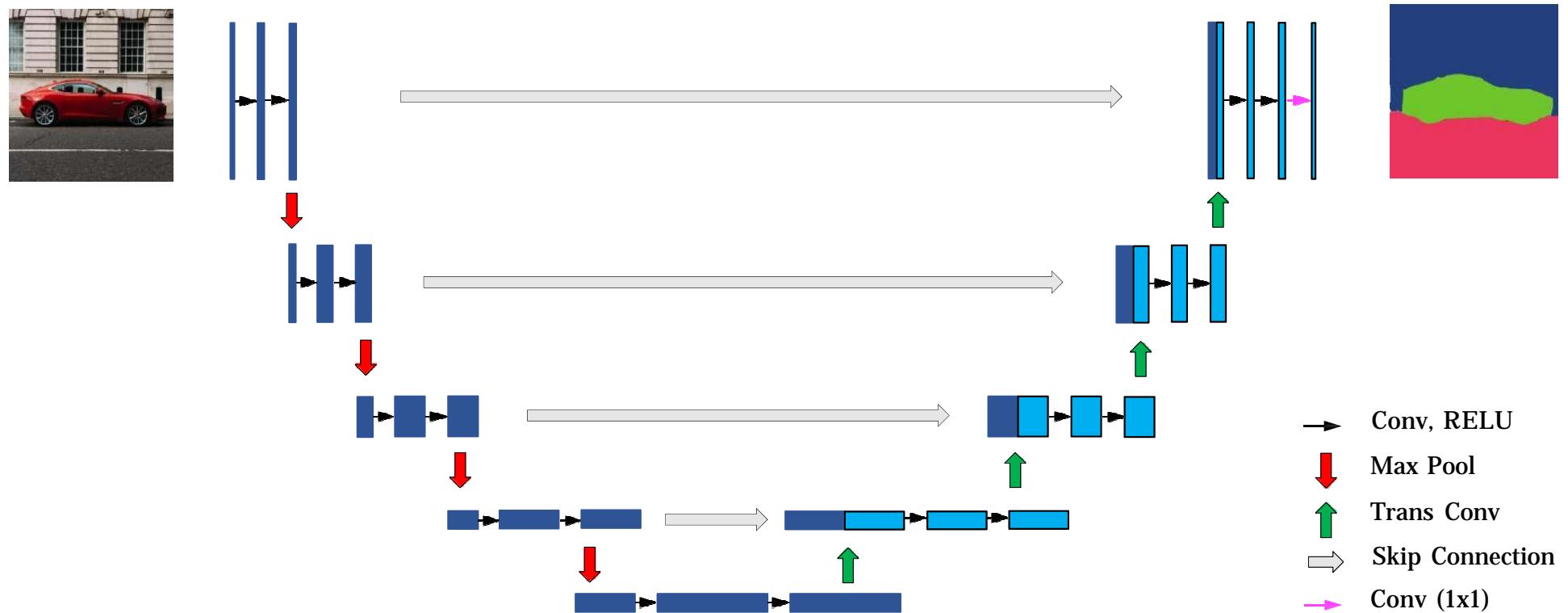
Transposed Convolution Matrix



UNET Motivation

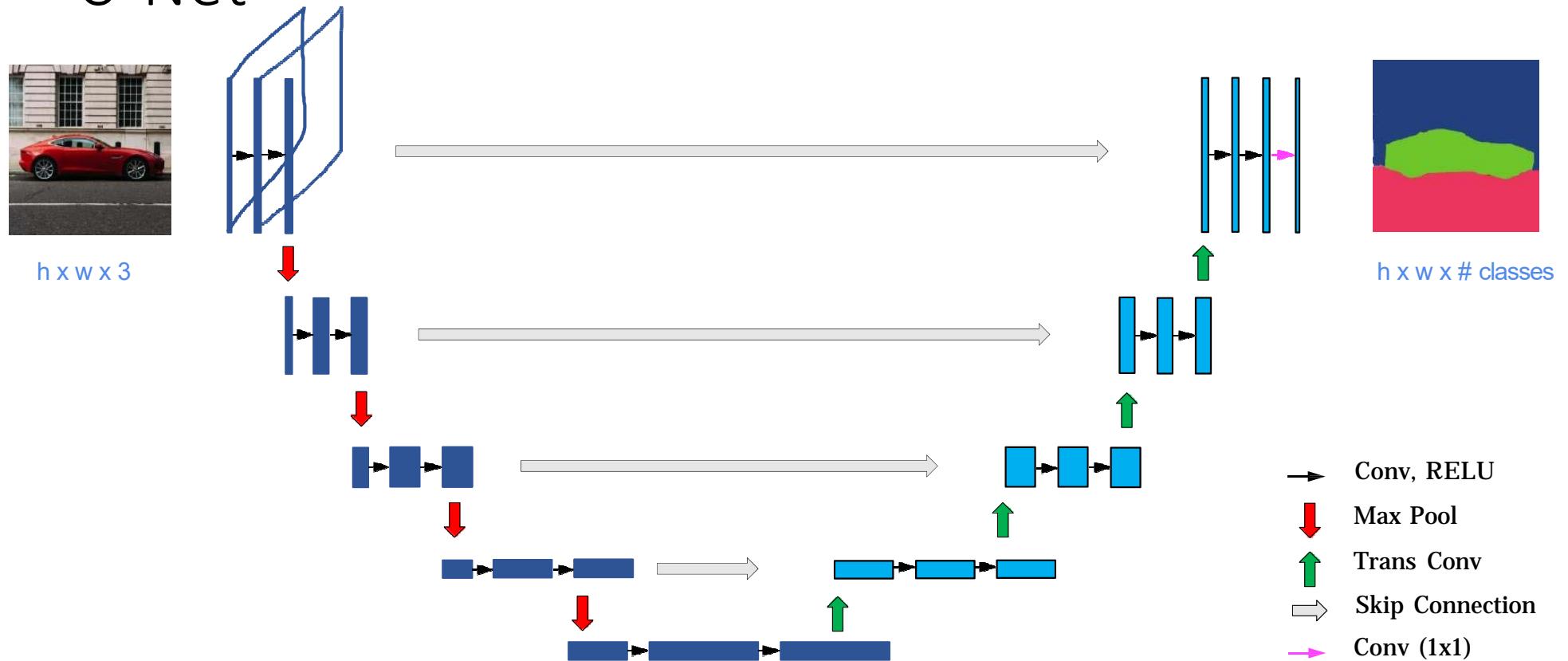


U-Net



[Ronneberger et al., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation]

U-Net



[Ronneberger et al., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation]

Car detection example

Training set:



1

1

1

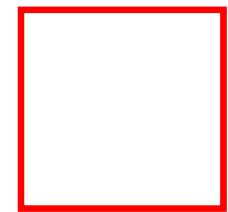
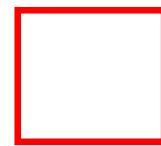
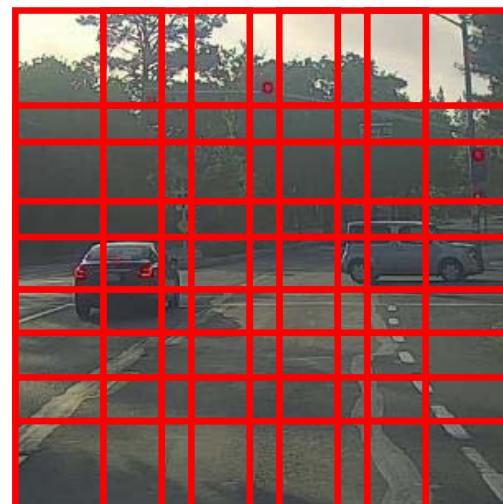
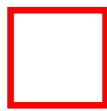
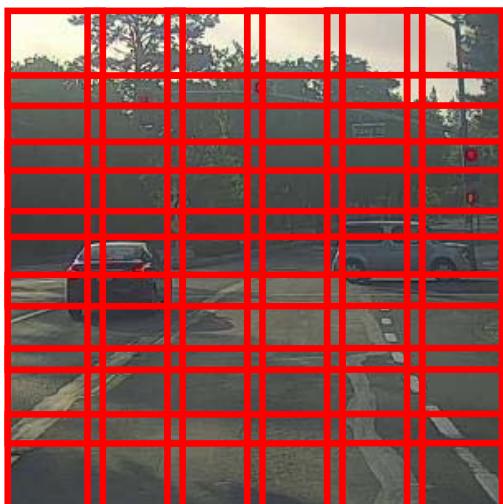
0

0

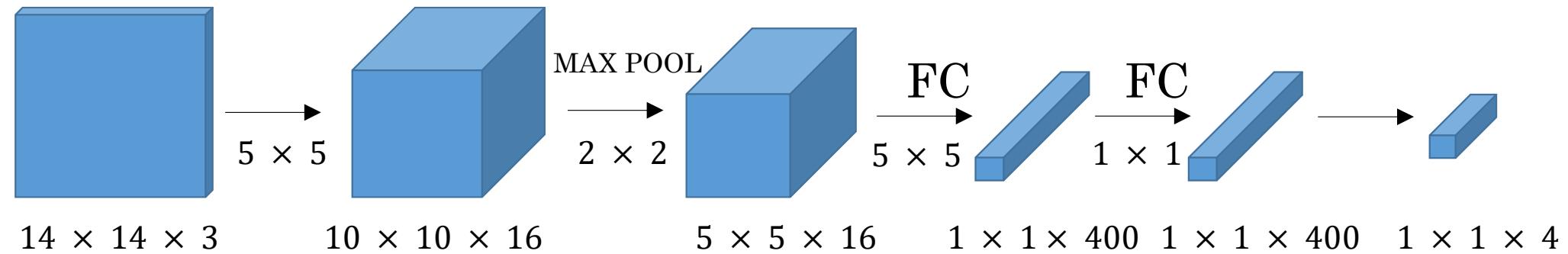
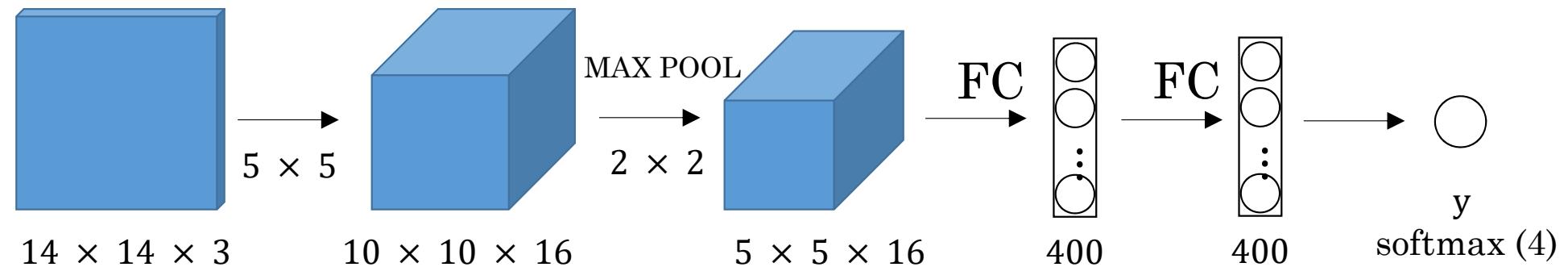


→ ConvNet → ↗

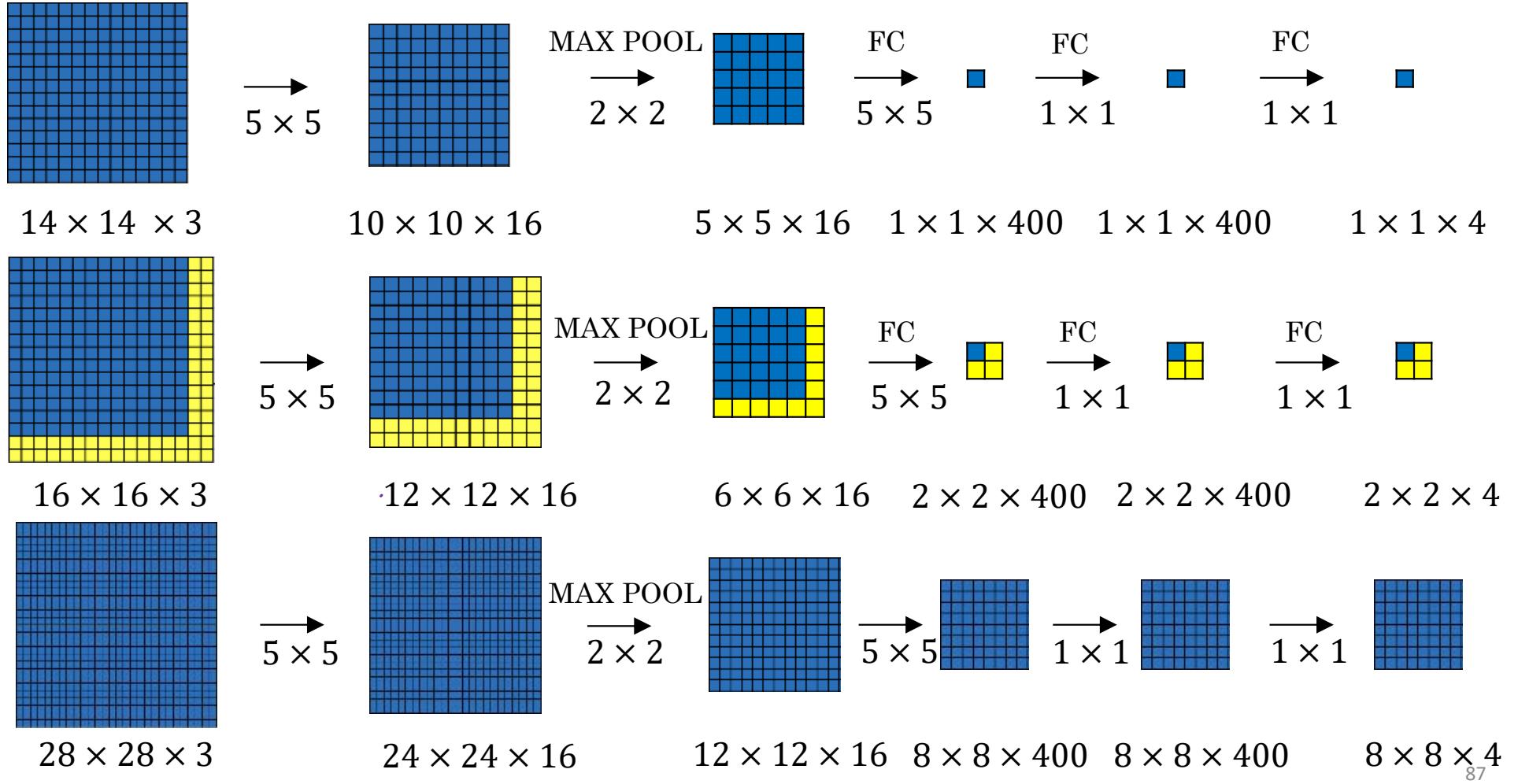
Sliding windows detection



Turning FC layer into convolutional layers

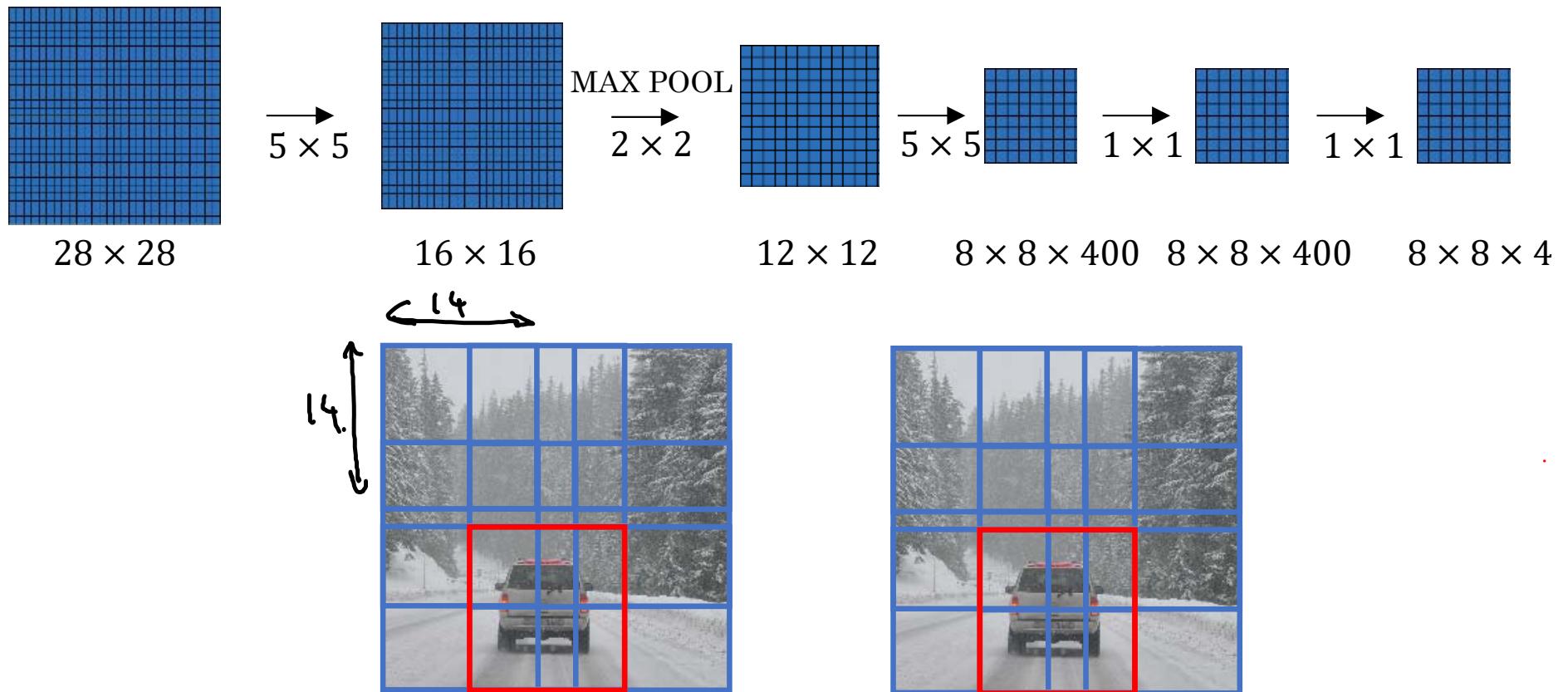


Convolution implementation of sliding windows

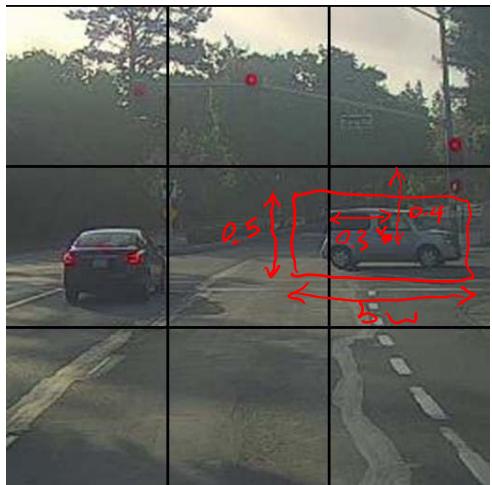


[Sermanet et al., 2014, OverFeat: Integrated recognition, localization and detection using convolutional networks]

Convolution implementation of sliding windows



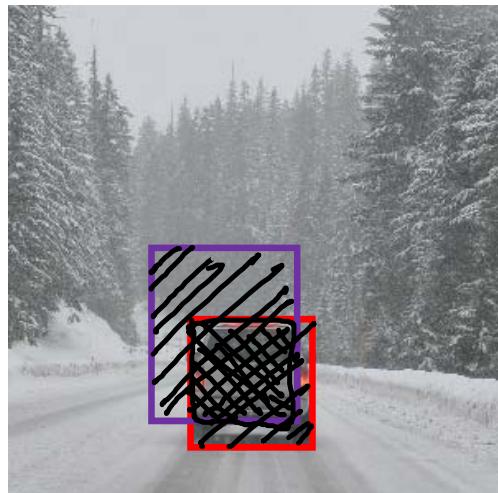
Specify the bounding boxes



$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{aligned} b_x &= 0.3 \\ b_y &= 0.4 \\ b_h &= 0.5 \\ b_w &= 1.1 \end{aligned}$$

Redmon, et al., 2015, You Only Look Once, unified real-time object detection

Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{Size of intersection}}{\text{Size of union}}$$

“Correct” if $\text{IoU} \geq 0.5$

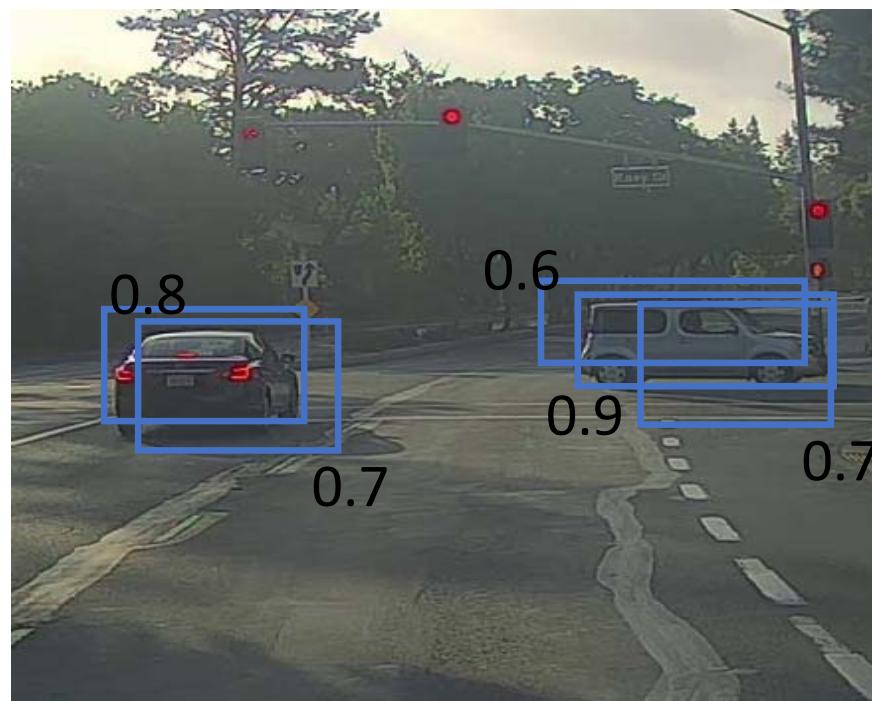
More generally, IoU is a measure of the overlap between two bounding boxes.

Non-max suppression example

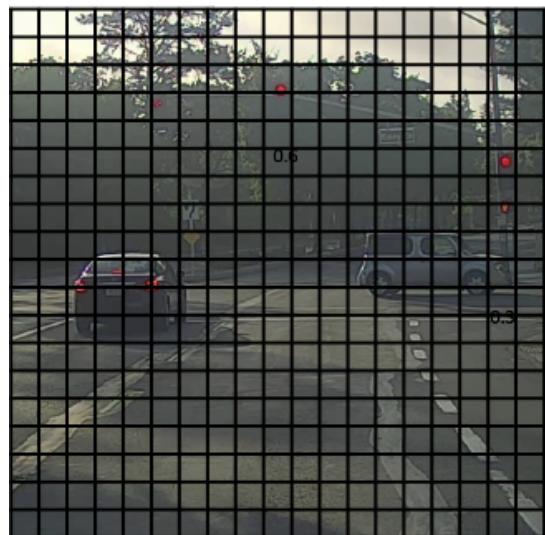


19x19

Non-max suppression example



Non-max suppression algorithm



19×19

Each output prediction is:

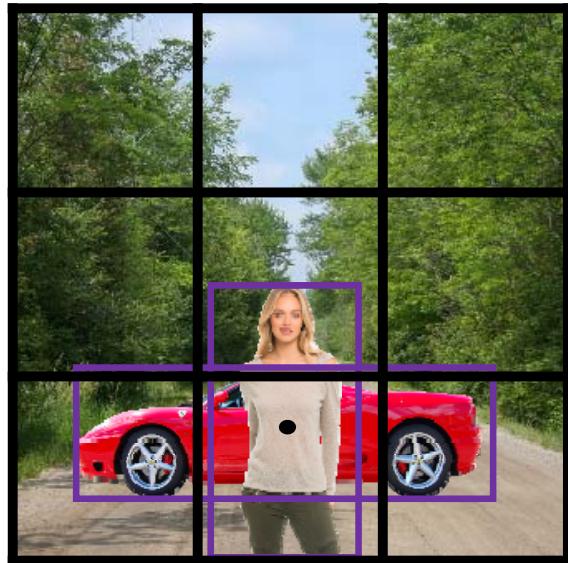
$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$$

Discard all boxes with $p_c \leq 0.6$

While there are any remaining boxes:

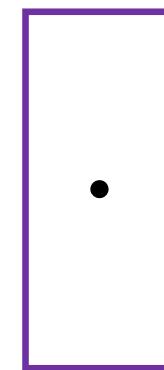
- Pick the box with the largest p_c
Output that as a prediction.
- Discard any remaining box with
 $\text{IoU} \geq 0.5$ with the box output
in the previous step

Overlapping objects:

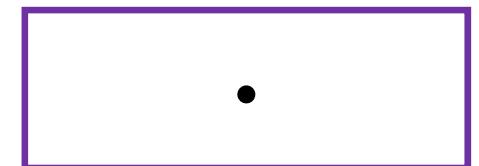


$$\mathbf{y} = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1:



Anchor box 2:



Redmon et al., 2015, You Only Look Once: Unified

Anchor box algorithm

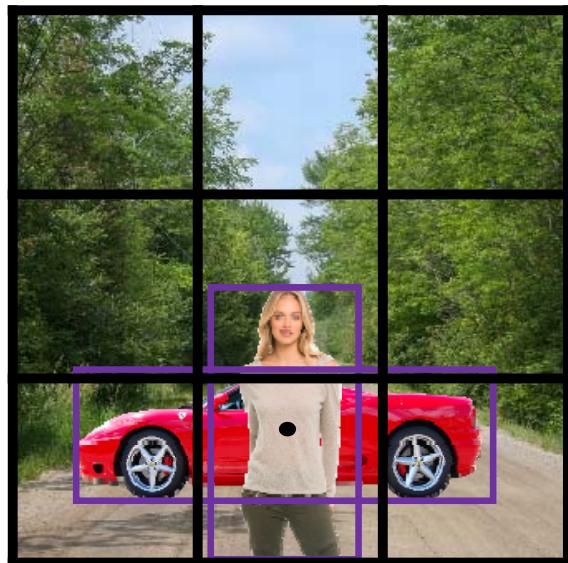
Previously:

Each object in training image is assigned to grid cell that contains that object's midpoint.

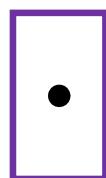
With two anchor boxes:

Each object in training image is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IoU.

Anchor box example



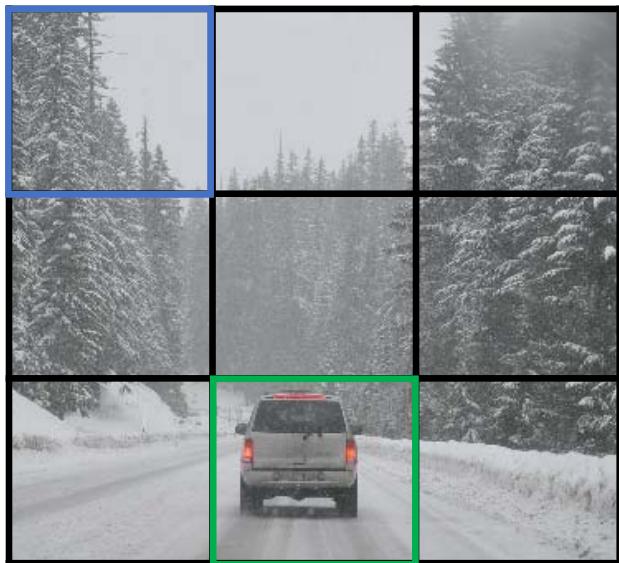
Anchor box 1: Anchor box 2:



$$\mathbf{y} = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

→

Training



- 1 - pedestrian
- 2 - car
- 3 - motorcycle

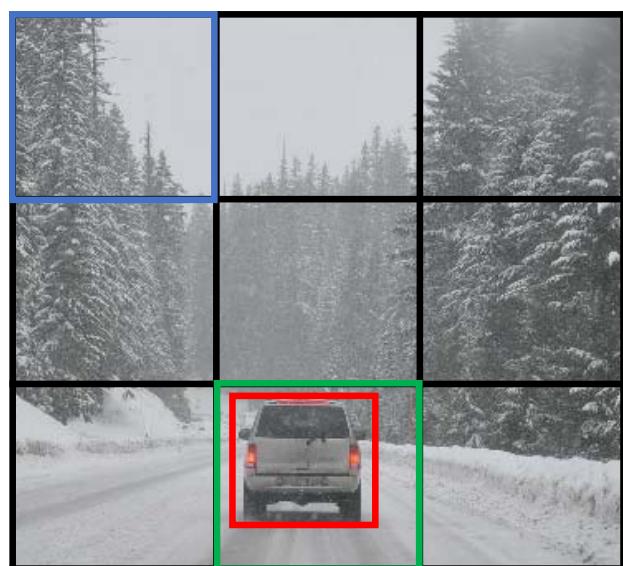
$$\mathbf{y} =$$

\mathbf{y} is $3 \times 3 \times 2 \times 8$

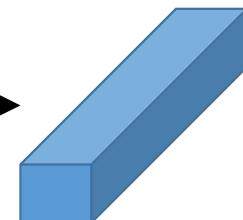
$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Making predictions



...

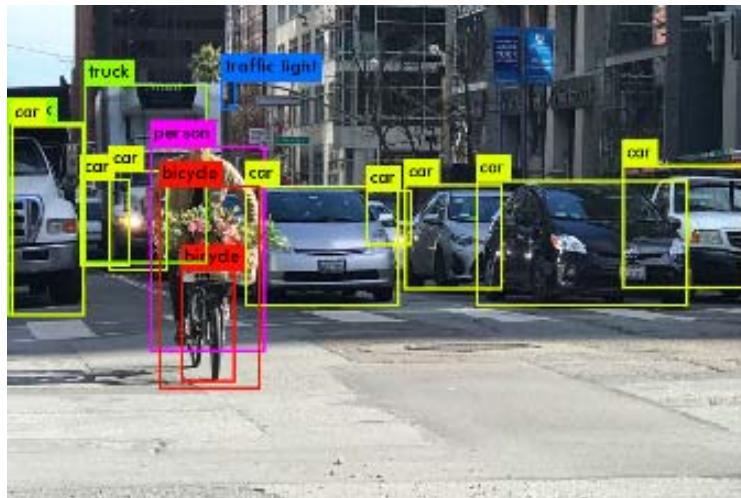


$$y = \begin{matrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{matrix}$$

$3 \times 3 \times 2 \times 8$

$$\begin{matrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{matrix}$$

Outputting the non-max suppressed outputs



- For each grid cell, get 2 predicted bounding boxes.
- Get rid of low probability predictions.
- For each class (pedestrian, car, motorcycle) use non-max suppression to generate final predictions.

Faster algorithms

R-CNN: Propose regions. Classify proposed regions one at a time. Output label + bounding box.

Fast R-CNN: Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions.

Faster R-CNN: Use convolutional network to propose regions.

[Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]

[Girshik, 2015. Fast R-CNN]

[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks]¹⁰⁰

Face verification vs. face recognition

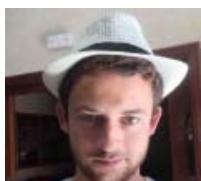
Verification

- Input image, name/ID
- Output whether the input image is that of the claimed person

Recognition

- Has a database of K persons
- Get an input image
- Output ID if the image is any of the K persons (or “not recognized”)

One-shot learning



Learning from one example to recognize the person again

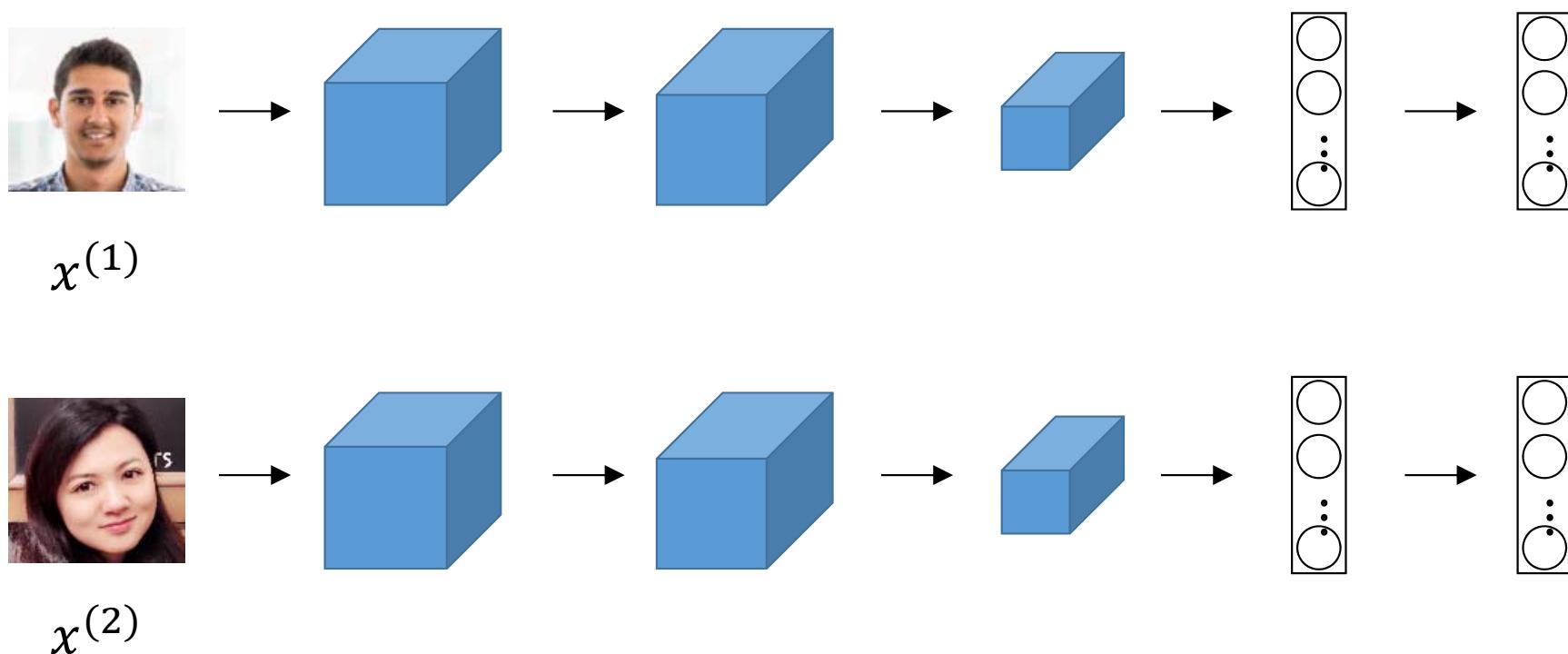
Learning a “similarity” function

$d(\text{img1}, \text{img2}) = \text{degree of difference between images}$

If $d(\text{img1}, \text{img2}) \leq \tau$ same
 $> \tau$ different

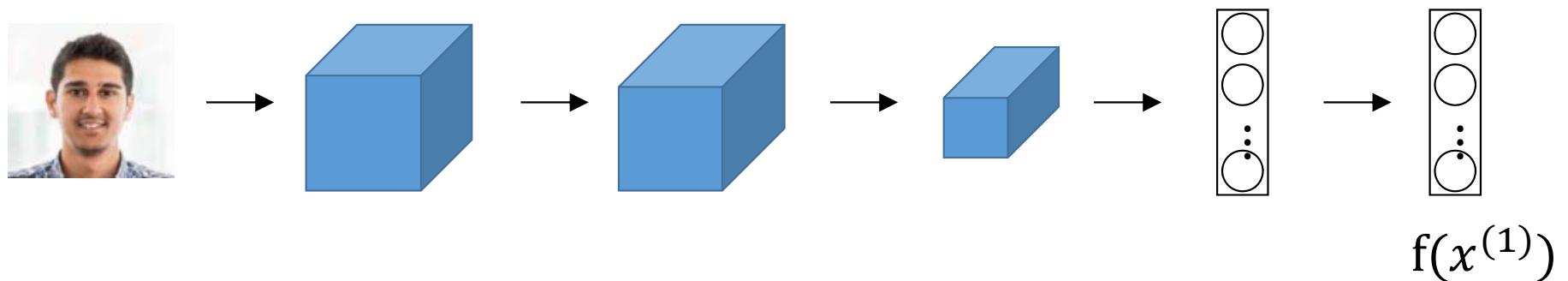


Siamese network



[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Goal of learning



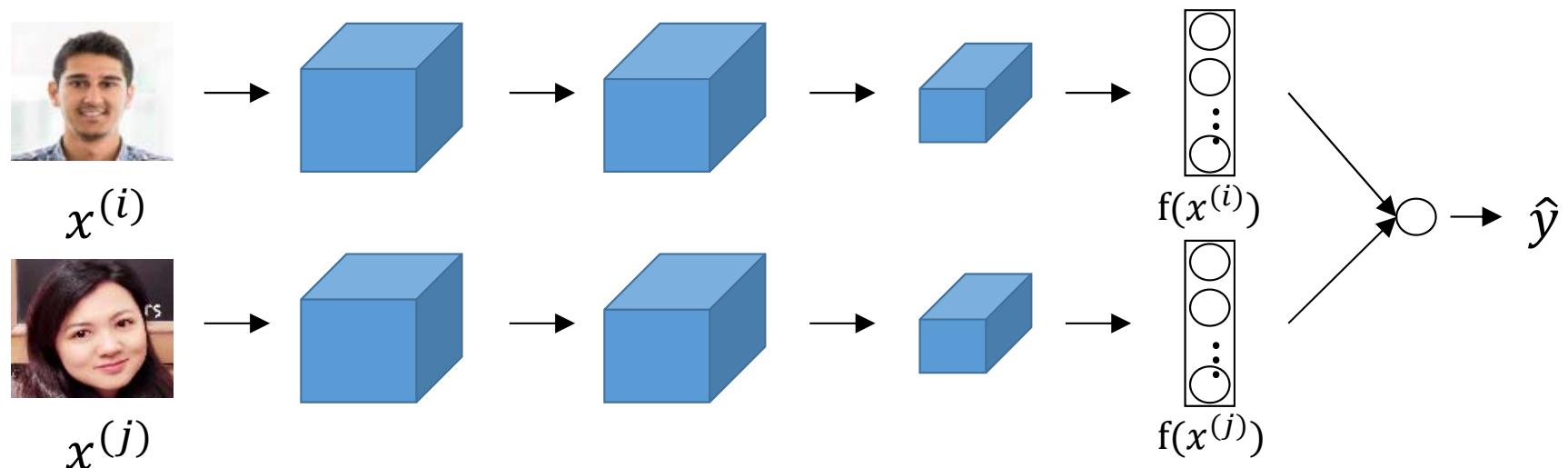
Parameters of NN define an encoding $f(x^{(i)})$

Learn parameters so that:

If $x^{(i)}, x^{(j)}$ are the same person, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is small.

If $x^{(i)}, x^{(j)}$ are different persons, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is large.

Learning the similarity function



$$d(f_1, f_2) = \sum_i \alpha_i |f_1[i] - f_2[i]|$$

$$\chi^2(f_1, f_2) = \sum_i w_i (f_1[i] - f_2[i])^2 / (f_1[i] + f_2[i])$$

[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Face verification supervised learning

x	y
	 1
	 0
	 0
	 1

[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

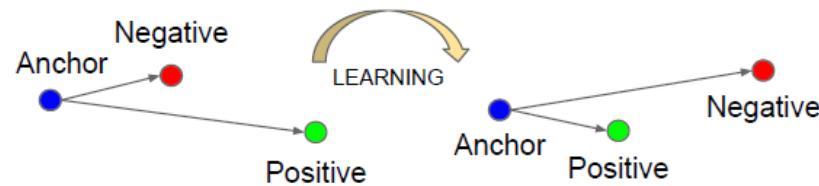
Triplet loss



Anchor Positive



Anchor Negative



$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2$$

$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

Choosing the triplets A,P,N

During training, if A,P,N are chosen randomly,
 $d(A, P) + \alpha \leq d(A, N)$ is easily satisfied.

Choose triplets that're “hard” to train on.

Training set using triplet loss

Anchor



Positive



Negative



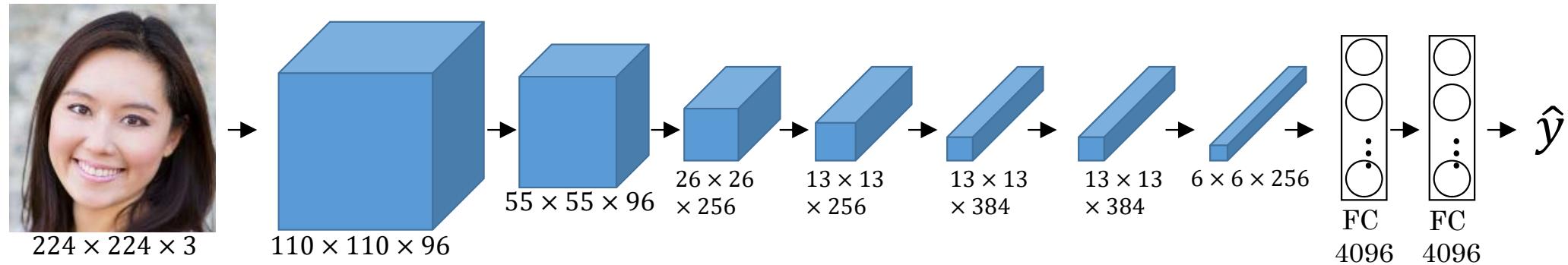
:

:

:



Visualizing what a deep network is learning



Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

Repeat for other units.

Visualizing deep layers: Layer 1



Layer 1



Layer 2



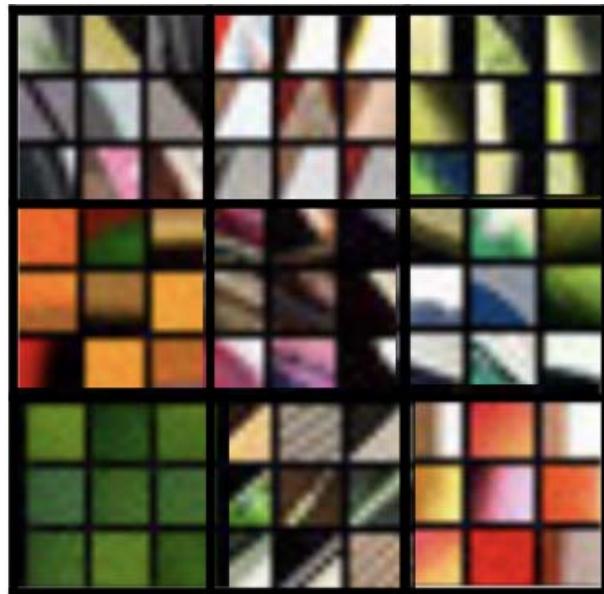
Layer 3



Layer 4



Layer 5



Visualizing deep layers: Layer 2



Layer 1



Layer 2



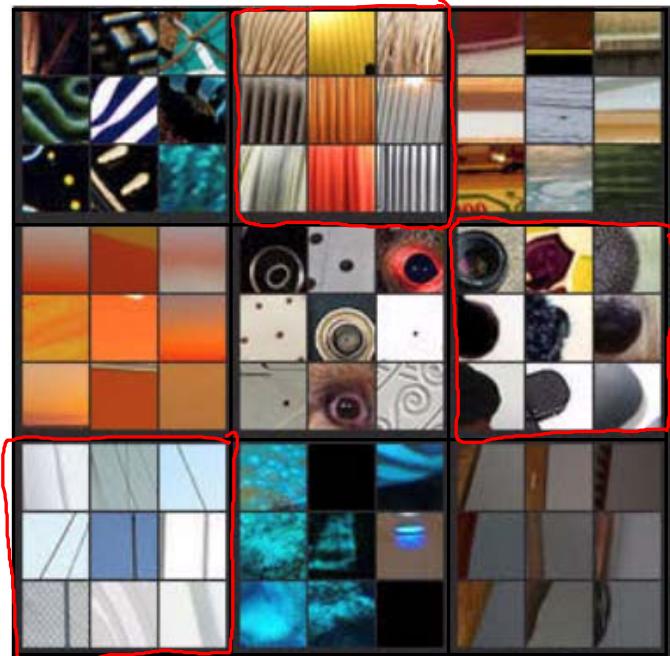
Layer 3



Layer 4



Layer 5



Visualizing deep layers: Layer 3



Layer 1



Layer 2



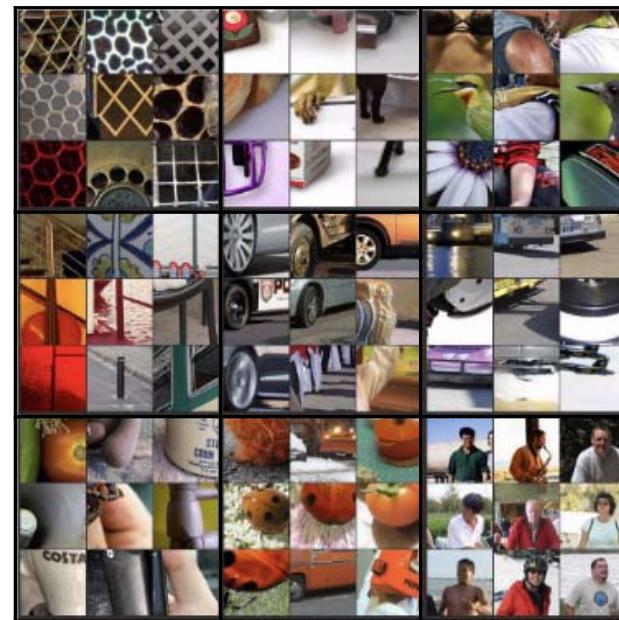
Layer 3



Layer 4



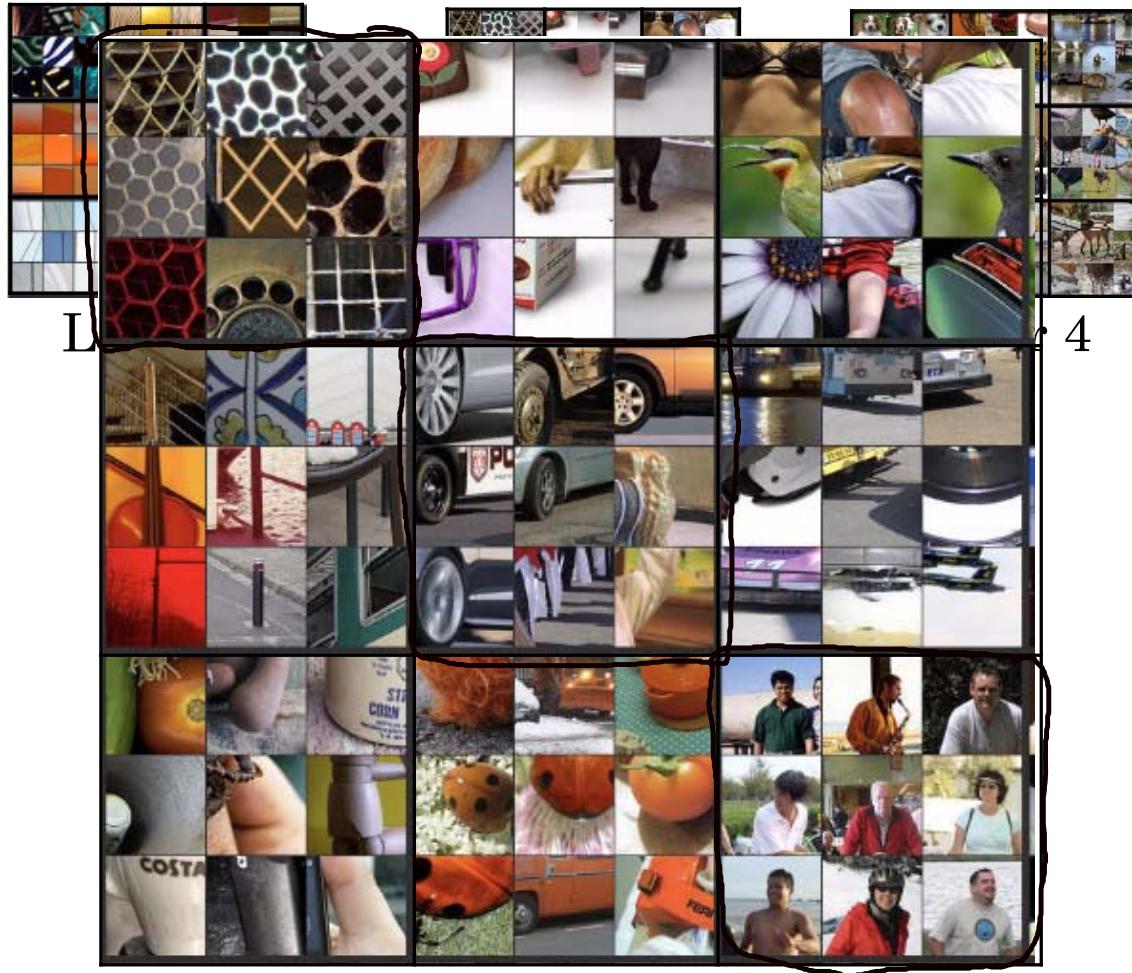
Layer 5



Visualizing deep layers: Layer 3



Layer 1



Layer 3

4



Layer 5

Visualizing deep layers: Layer 4



Layer 4



Layer 4

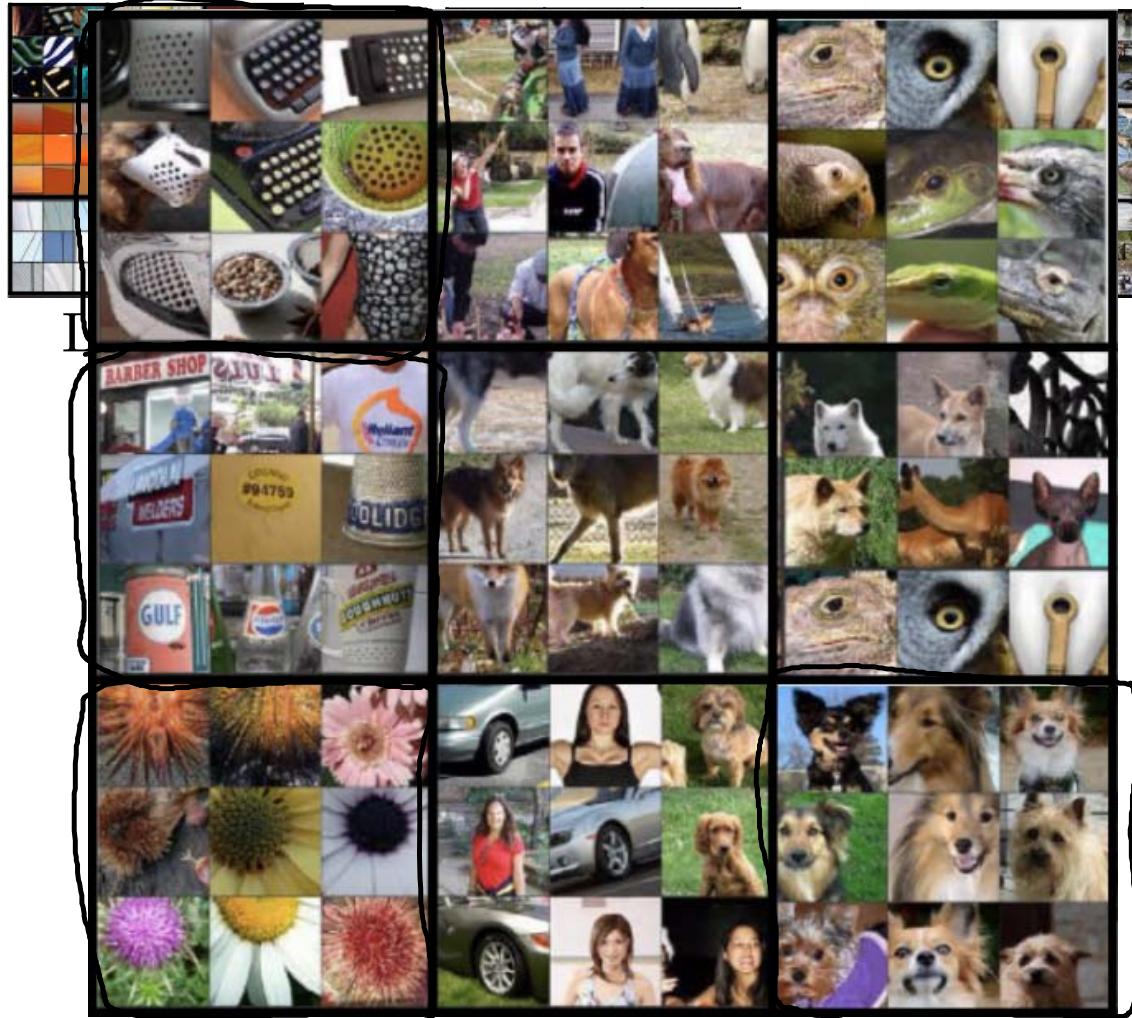


Layer 5

Visualizing deep layers: Layer 5

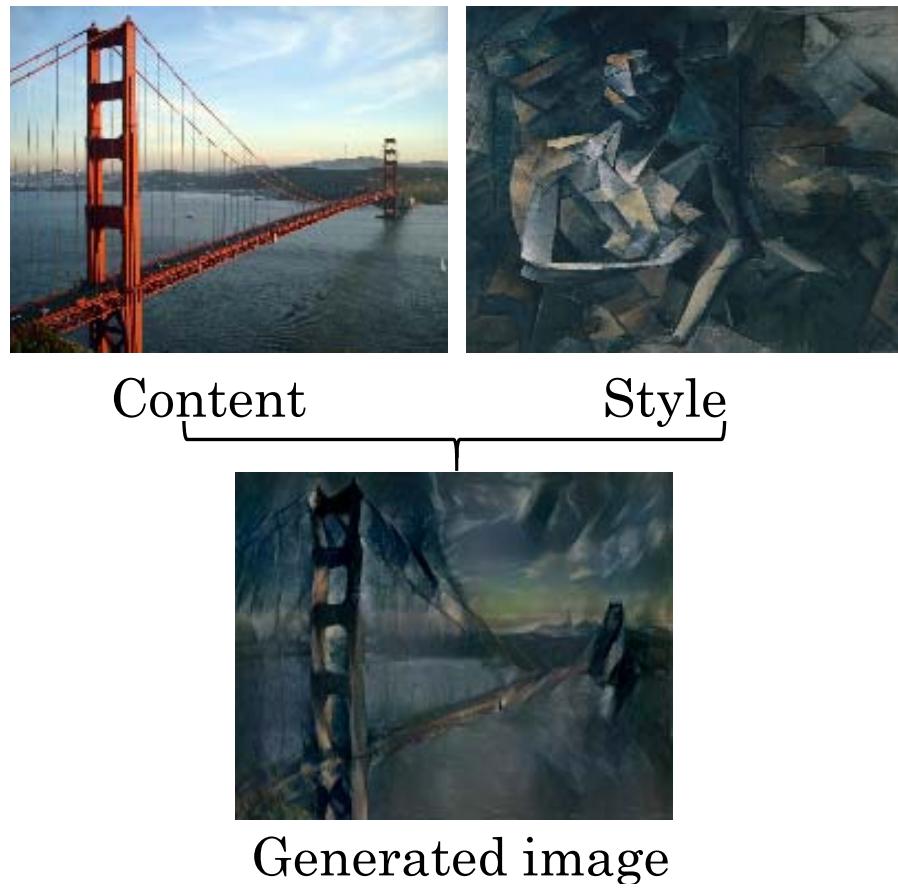


Layer 1



Layer 5

Neural Style Transfer



Neural style transfer cost function



Content C

Style S



Generated image G

[Gatys et al., 2015. A neural algorithm of artistic style. Images on slide generated by Justin Johnson]¹¹⁹

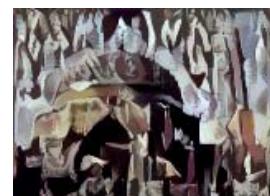
Find the generated image G

1. Initiate G randomly

$G: 100 \times 100 \times 3$



2. Use gradient descent to minimize $J(G)$



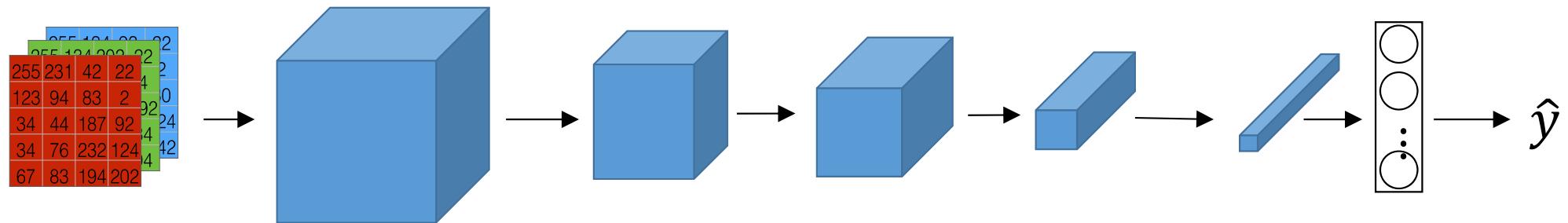
[Gatys et al., 2015. A neural algorithm of artistic style]

Content cost function

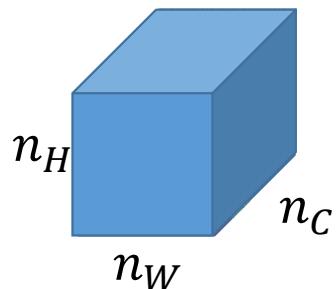
$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$

- Say you use hidden layer l to compute content cost.
- Use pre-trained ConvNet. (E.g., VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

Meaning of the “style” of an image



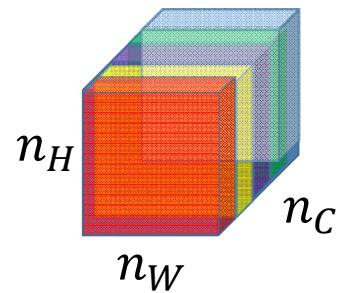
Say you are using layer l 's activation to measure “style.”
Define style as correlation between activations across channels.



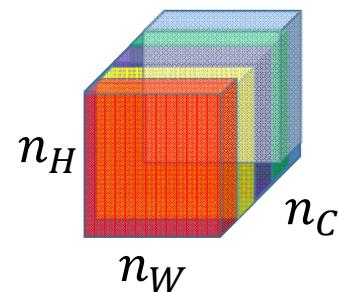
How correlated are the activations
across different channels?

Intuition about style of an image

Style image

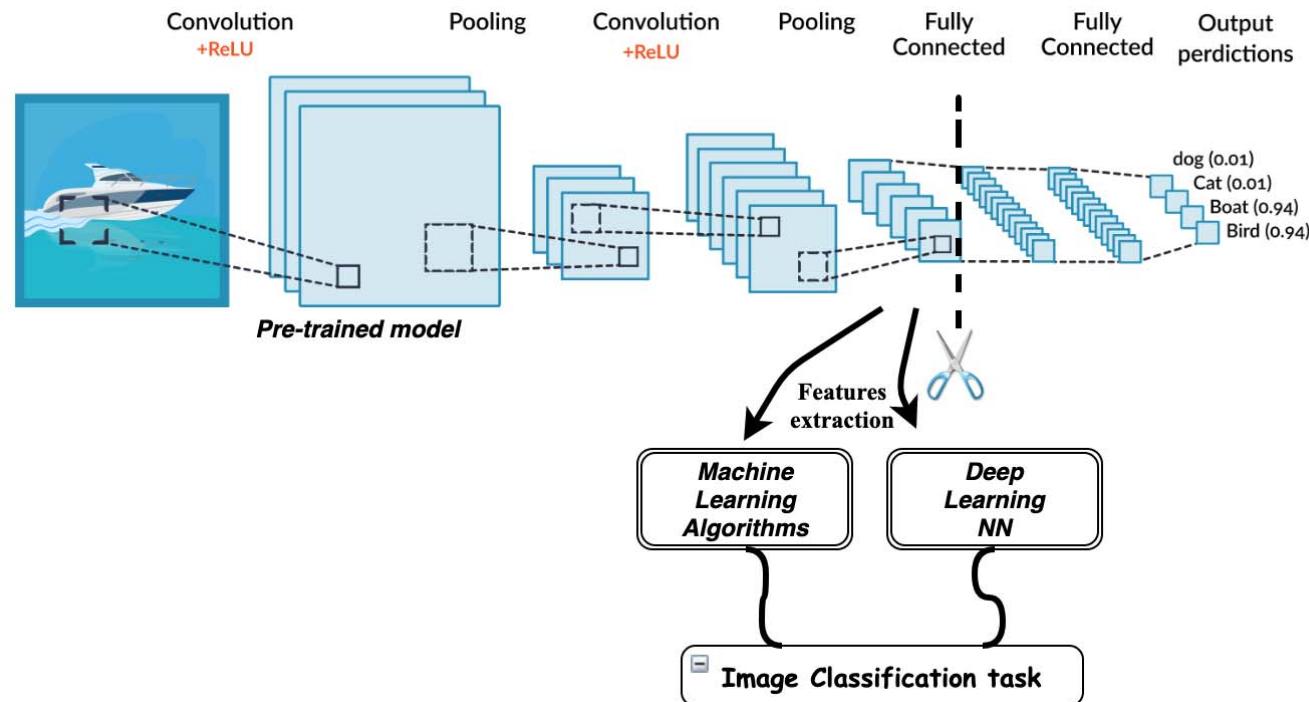


Generated Image

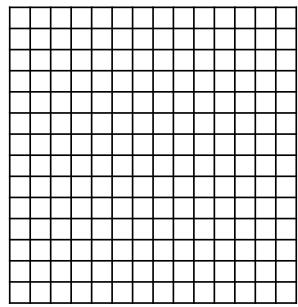


[Gatys et al., 2015. A neural algorithm of artistic style]

Transfer Learning



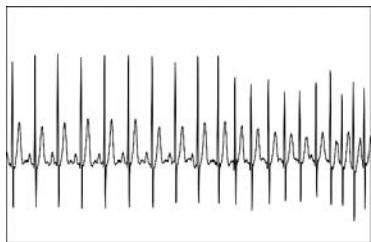
Convolutions in 2D and 1D



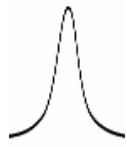
2D input image
 14×14

*

2D filter
 5×5



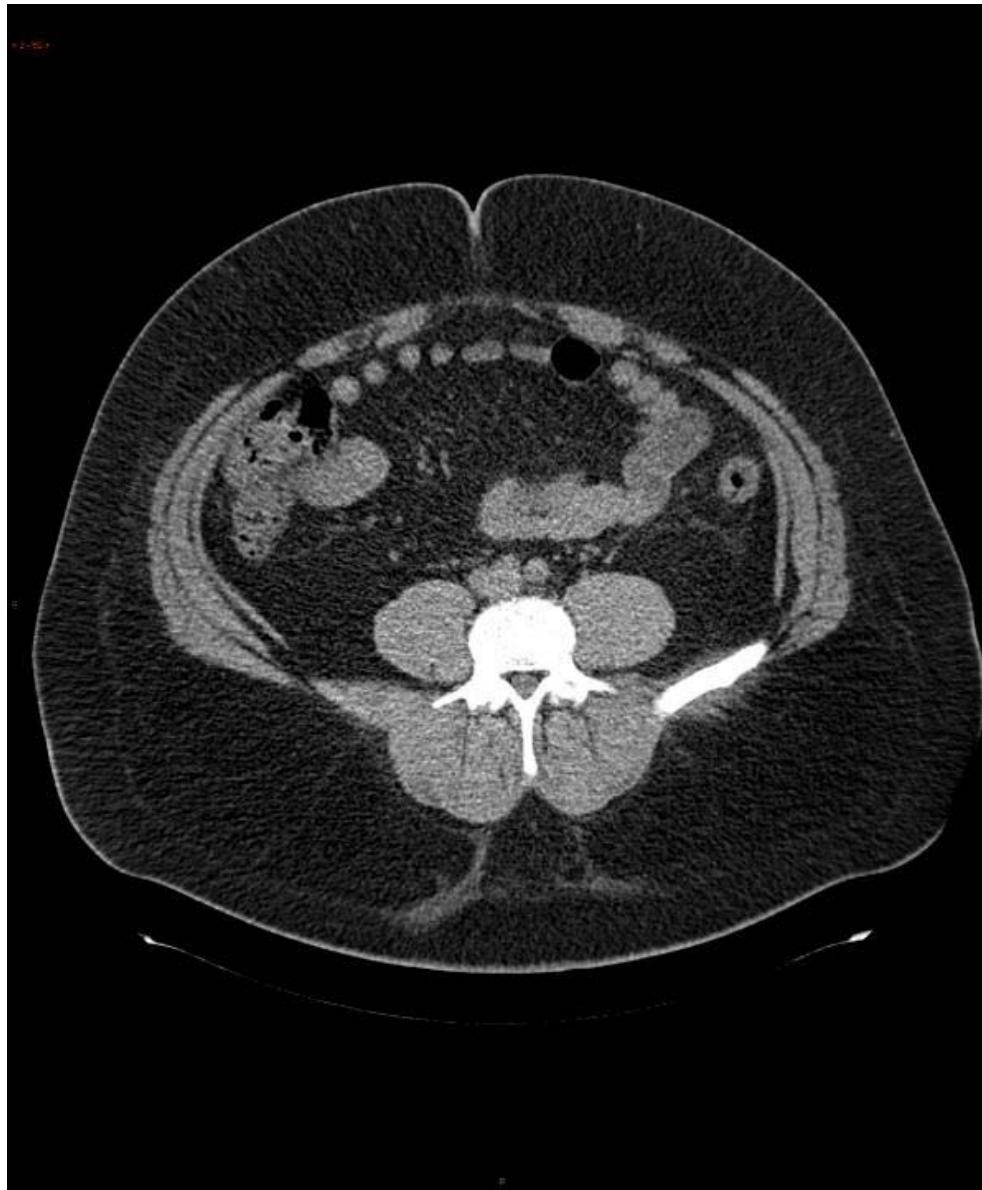
*



1	20	15	3	18	12	4	17
---	----	----	---	----	----	---	----

1	3	10	3	1
---	---	----	---	---

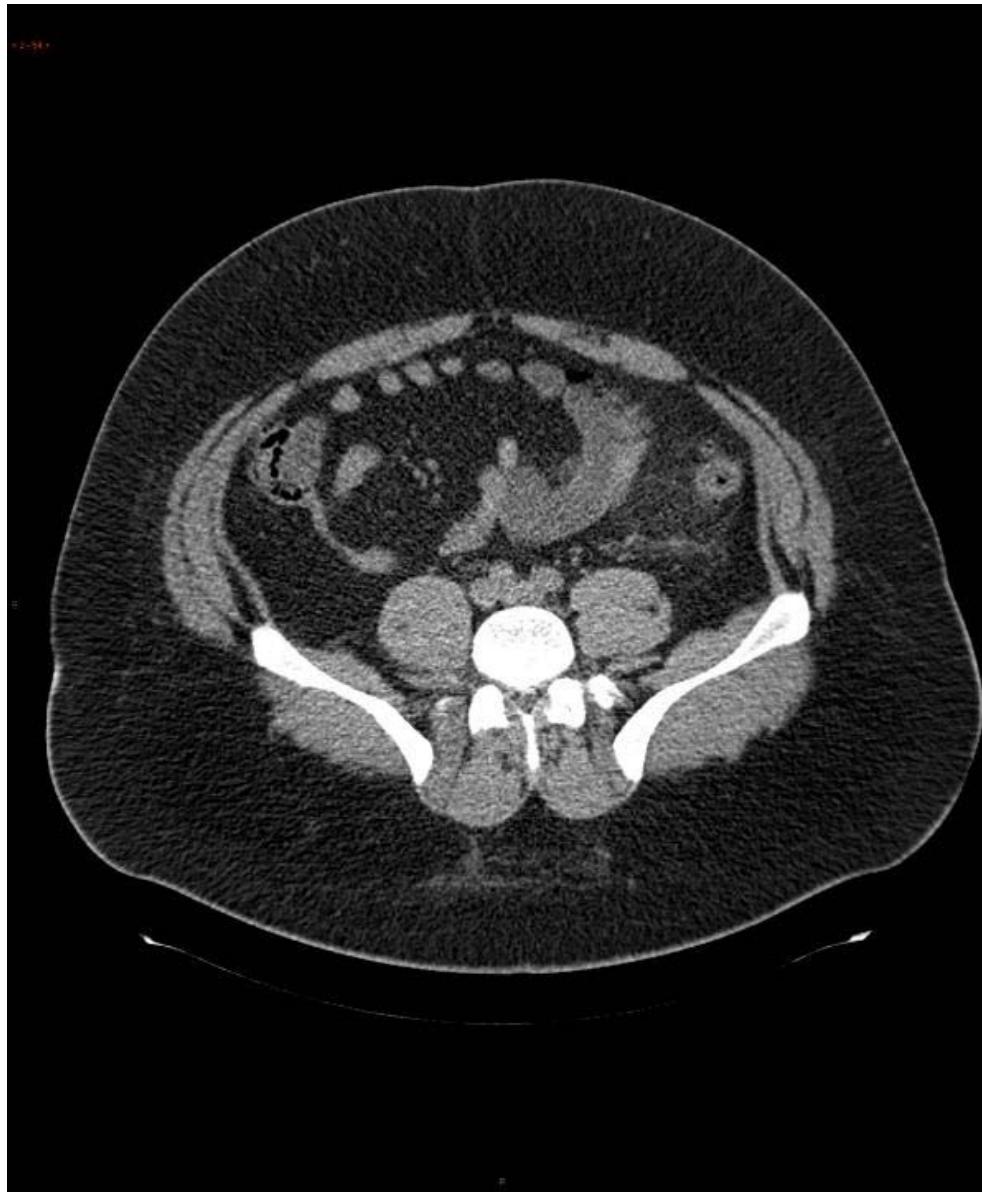
3D data



3D data

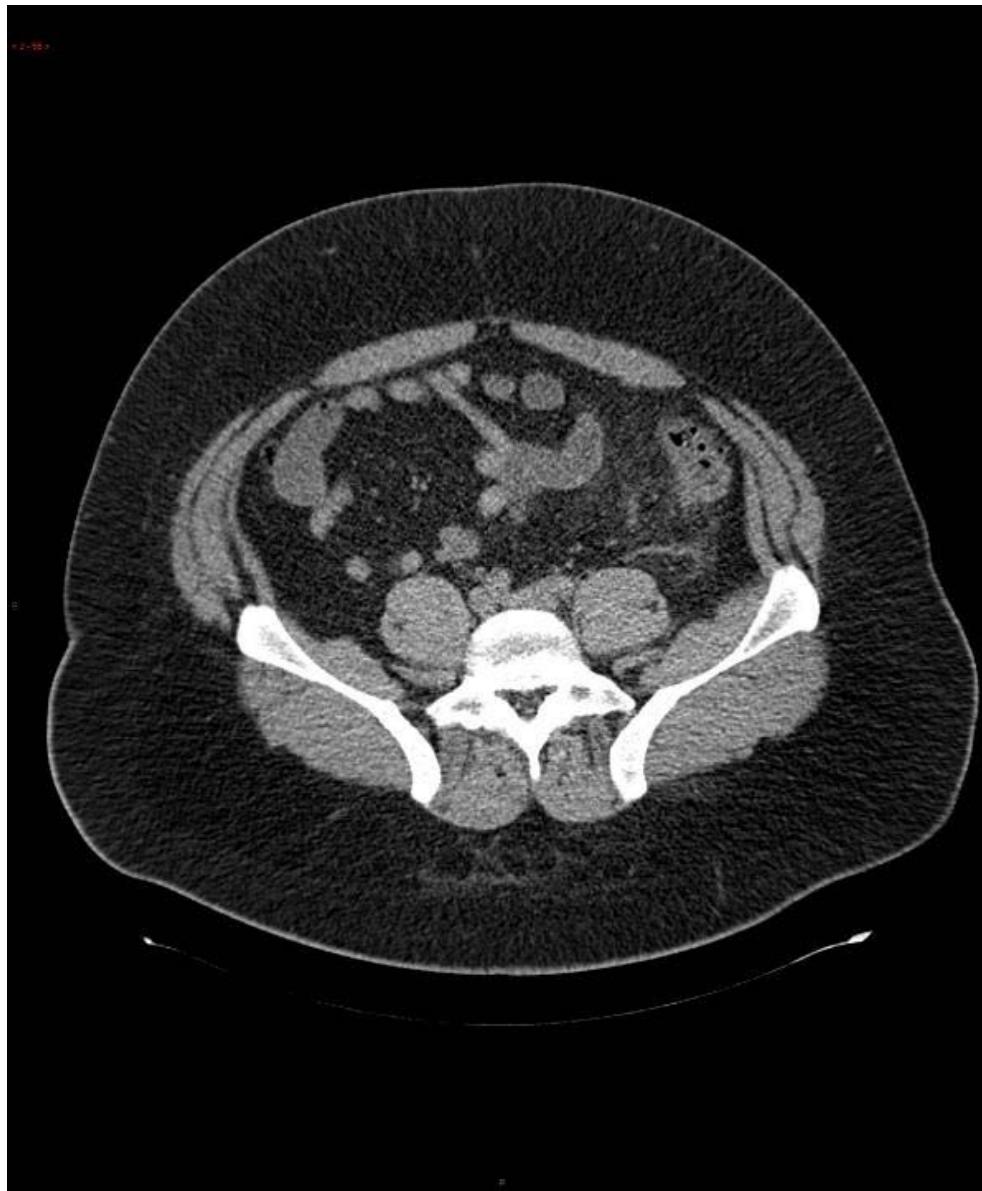


3D data

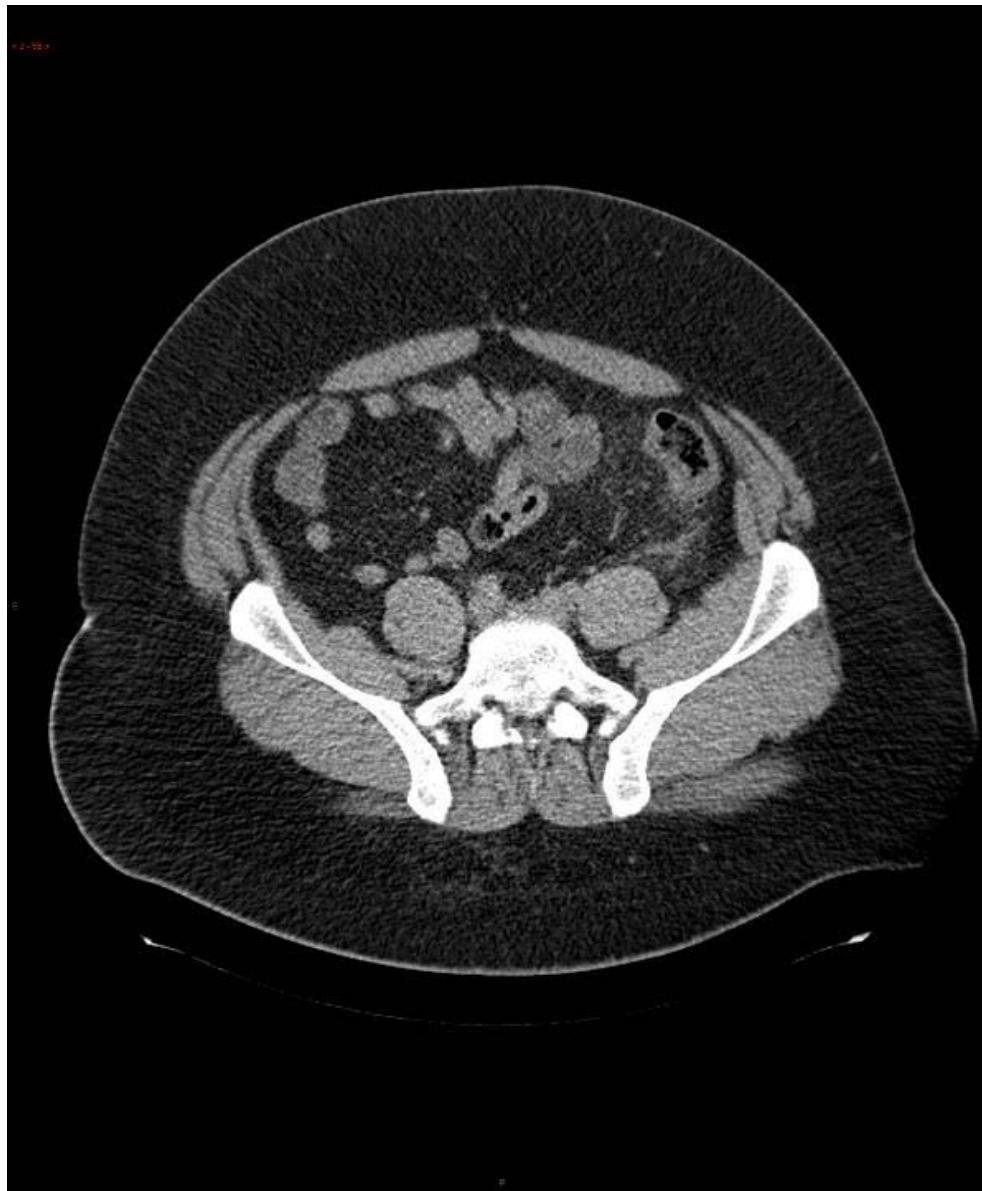


128

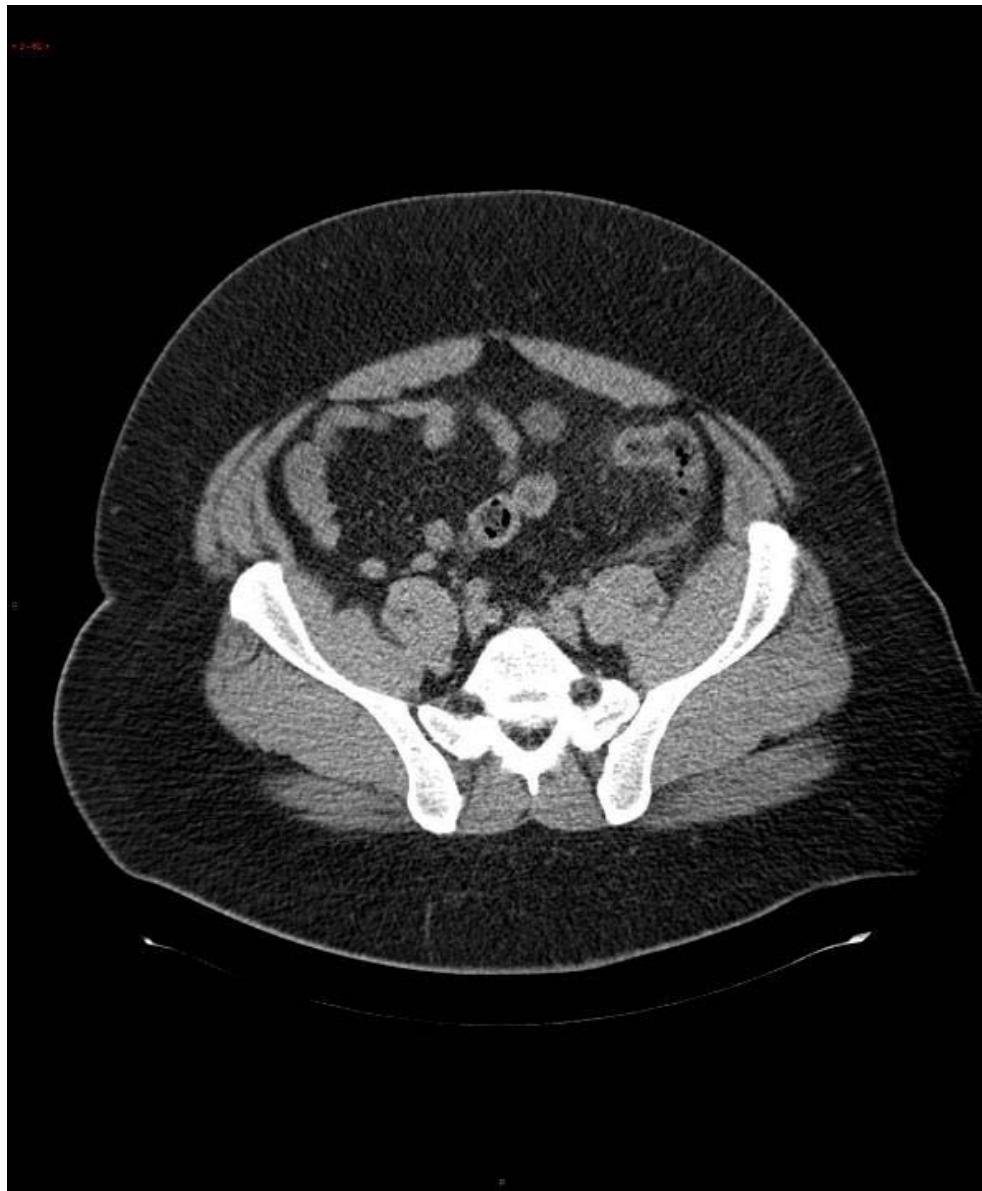
3D data



3D data



3D data



3D data



3D data



3D data



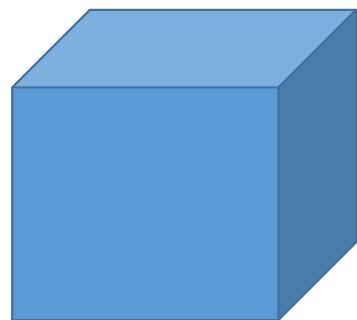
3D data



3D data



3D convolution



*



3D filter

3D volume