Learning for vision V architectures

Karel Zimmermann

http://cmp.felk.cvut.cz/~zimmerk/

Vision for Robotics and Autonomous Systems
https://cyber.felk.cvut.cz/vras/

Center for Machine Perception
https://cmp.felk.cvut.cz

Department for Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

Outline

- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Architectures of feature matching networks

IMAGENET

Classification results

http://image-net.org/challenges/LSVRC/2017/index

Label: Steel drum

IMAGENET

Classification results

http://image-net.org/challenges/LSVRC/2017/index

Label: Steel drum

Output:
Scale
T-shirt
<u>Steel drum</u>
Drumstick
Mud turtle

✓

IMAGENET

Classification results

http://image-net.org/challenges/LSVRC/2017/index

Label: Steel drum

Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

✓

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

IMAGENET

Classification results

http://image-net.org/challenges/LSVRC/2017/index

Label: Steel drum

Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

✓

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

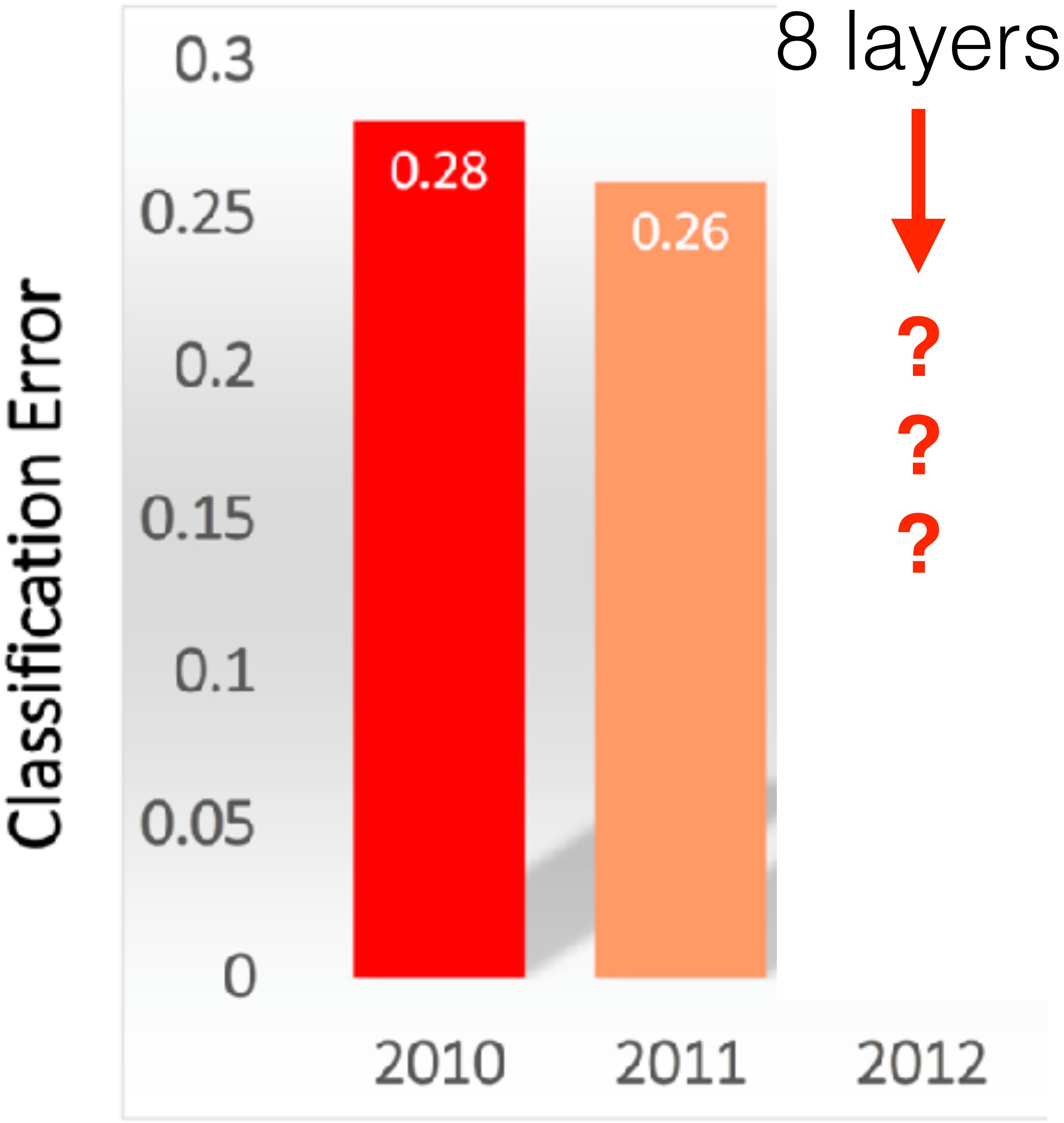
✗

$$\text{Error} = \frac{1}{100{,}000} \sum_{\substack{100{,}000 \\ \text{images}}} 1[\text{incorrect on image i}]$$

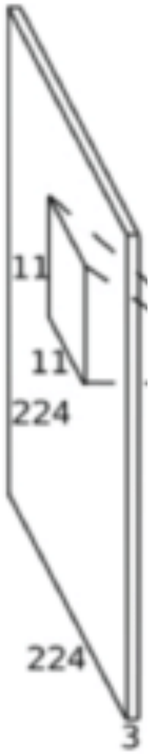
IMAGENET

Classification results

AlexNet
8 layers



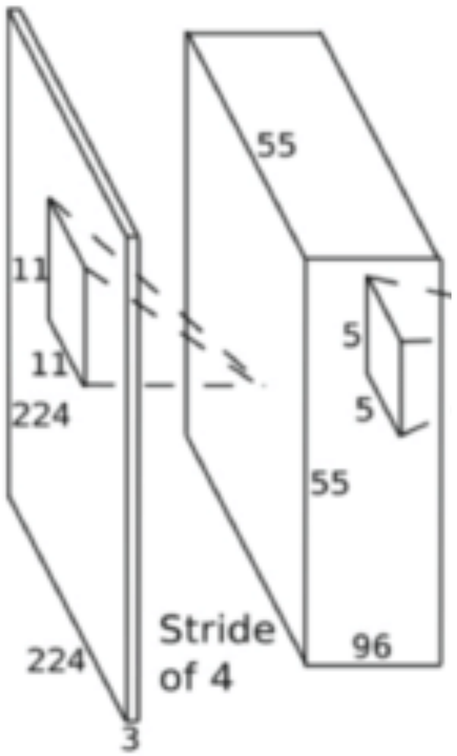
AlexNet on ImageNet 2012 (**over 27k citations !!!**)



- Param in layer1 (conv, 96 11x11 filters, stride=4, pad=0)?

Alex Krizhevsky et al, Imagenet classification with deep convolutional neural networks, NIPS, 2012 https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

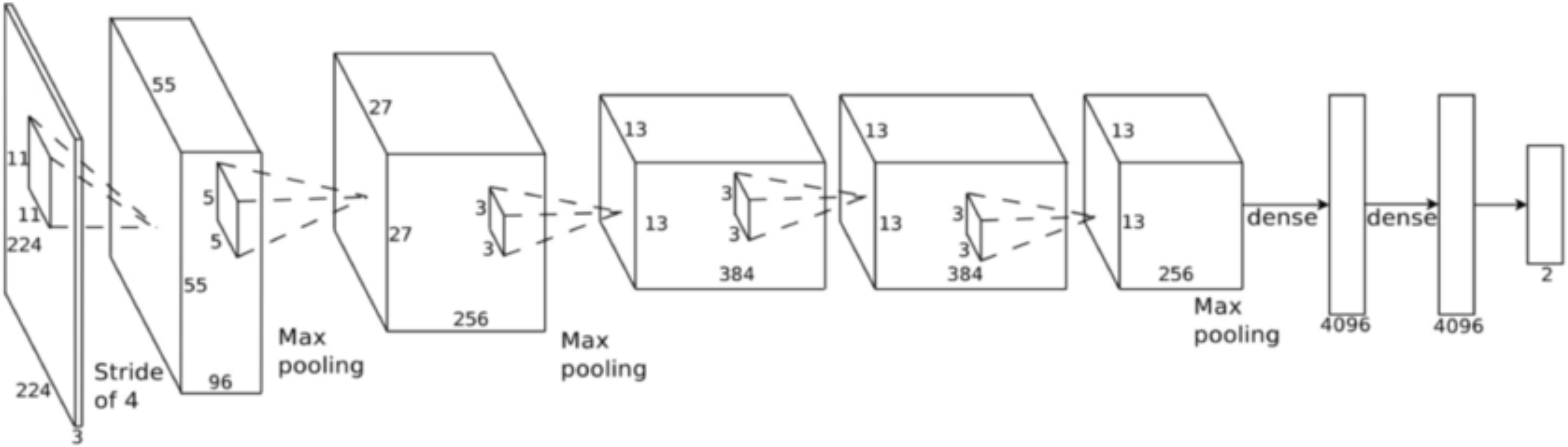
AlexNet on ImageNet 2012 (**over 27k citations !!!**)



- Param in layer1 (conv, 96 11x11 filters, stride=4, pad=0)?
- Param in layer2 (maxp,3x3 filters, stride=2, pad=0)?

Alex Krizhevsky et al, Imagenet classification with deep convolutional neural networks, NIPS, 2012 https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

AlexNet on ImageNet 2012 (**over 27k citations !!!**)

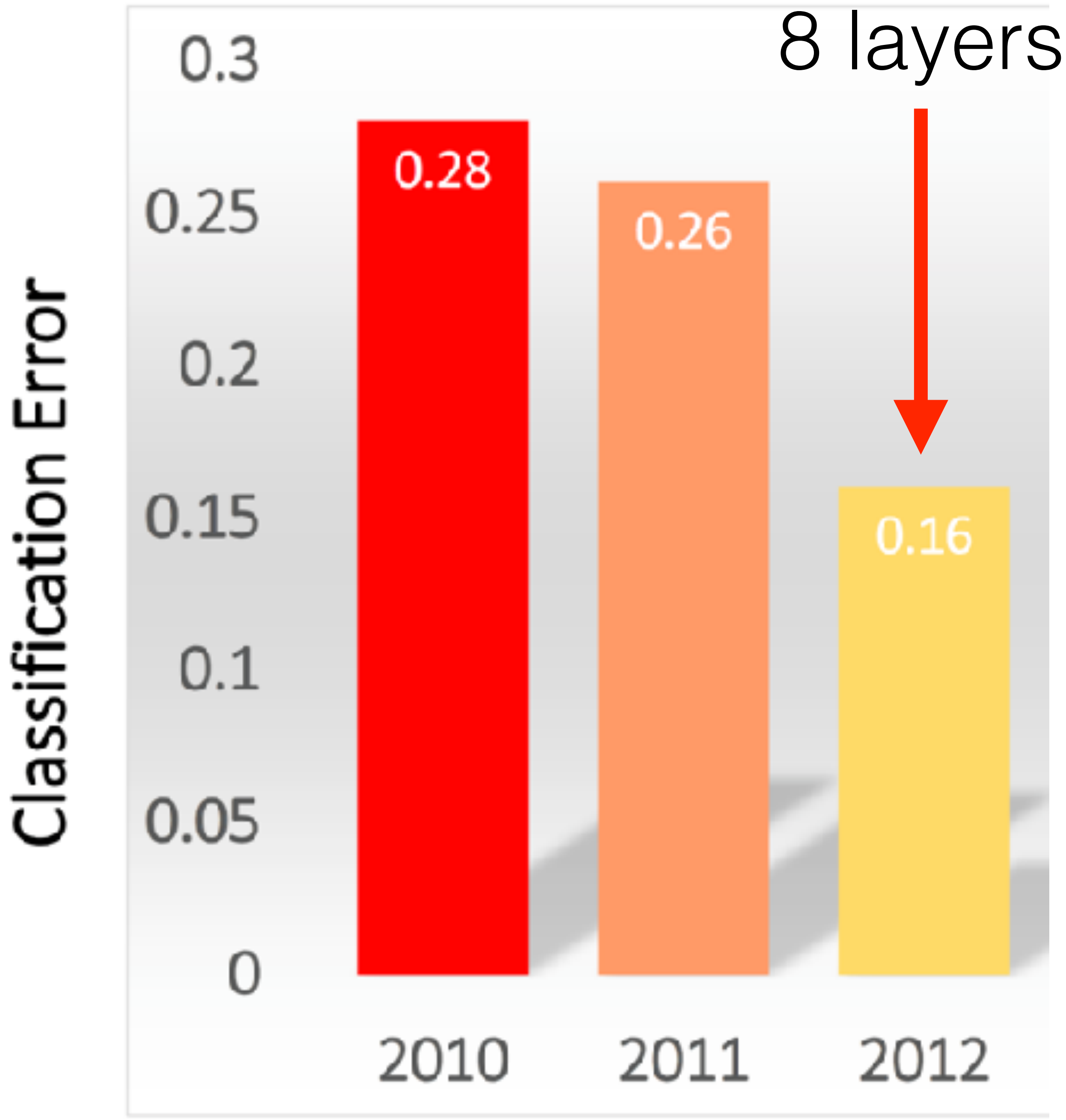


- Param in layer1 (conv, 96 11x11 filters, stride=4, pad=0)?
- Param in layer2 (maxp,3x3 filters, stride=2, pad=0)?
- Param in layer3 (conv, 256 5x5 filters, stride=1, pad=2?
- Parameters in total: 60M, Depth: 8 layers

Alex Krizhevsky et al, Imagenet classification with deep convolutional neural networks, NIPS, 2012 https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

IMAGENET

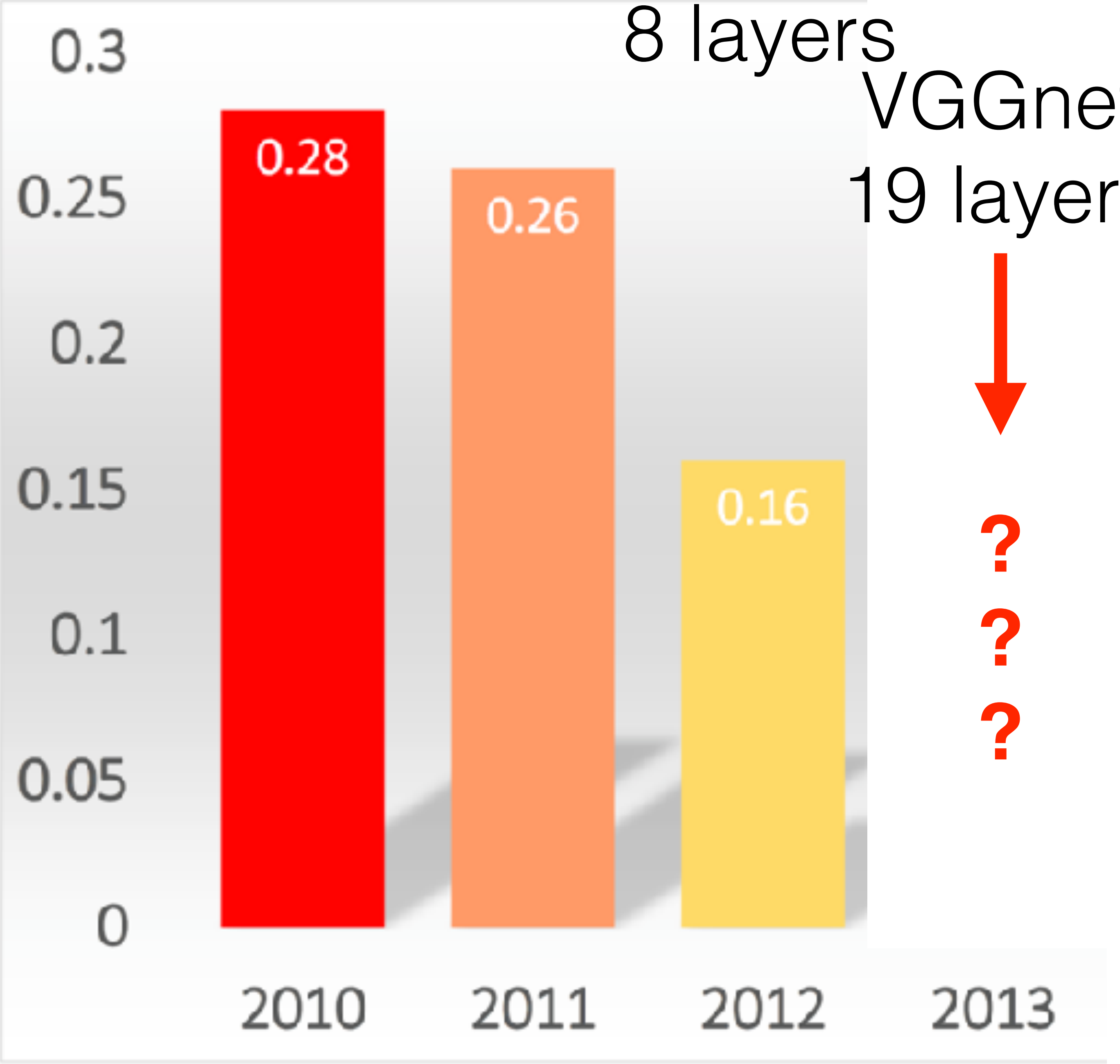
Classification results



AlexNet
8 layers

IMAGENET

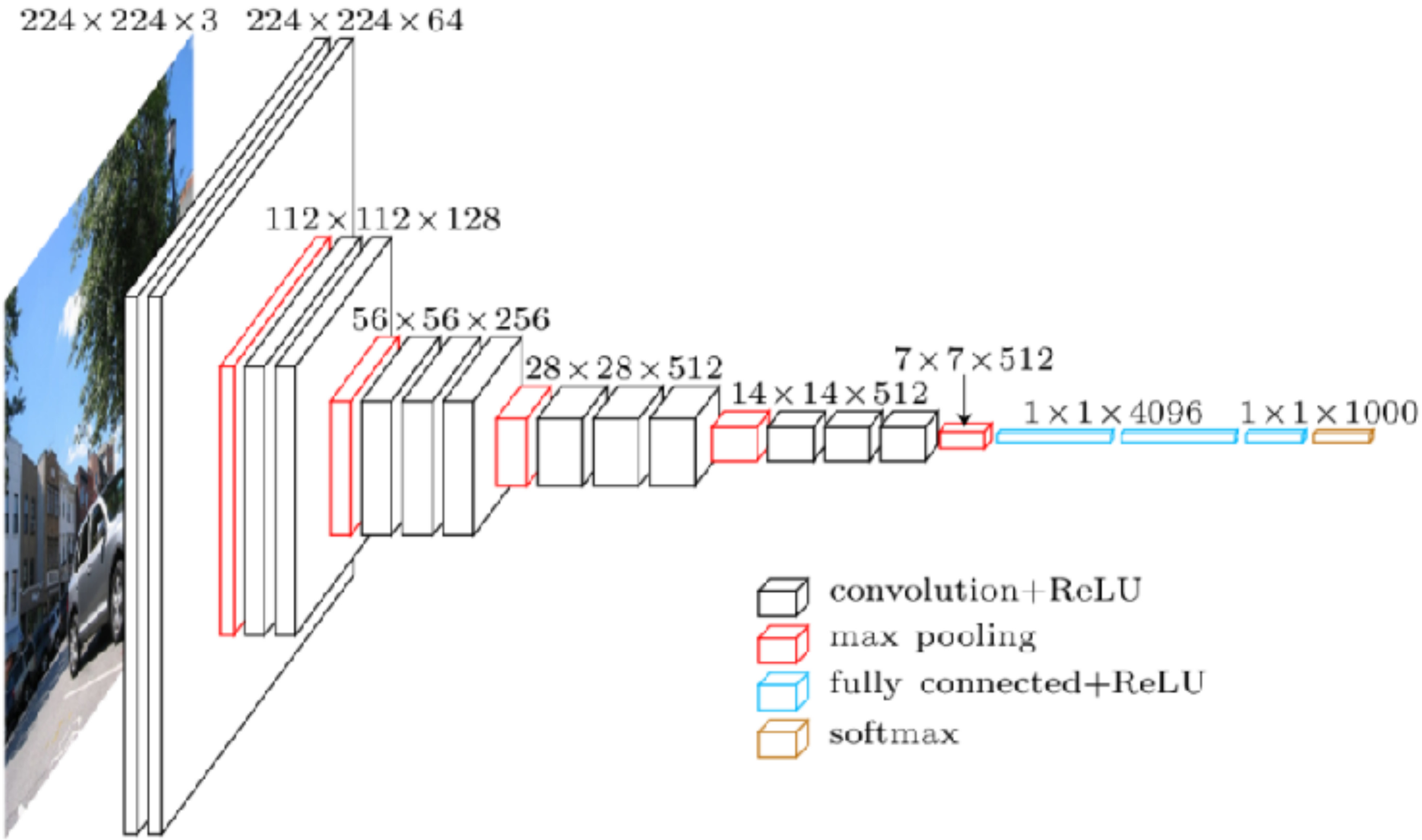
Classification results



AlexNet
8 layers

VGGnet
19 layers

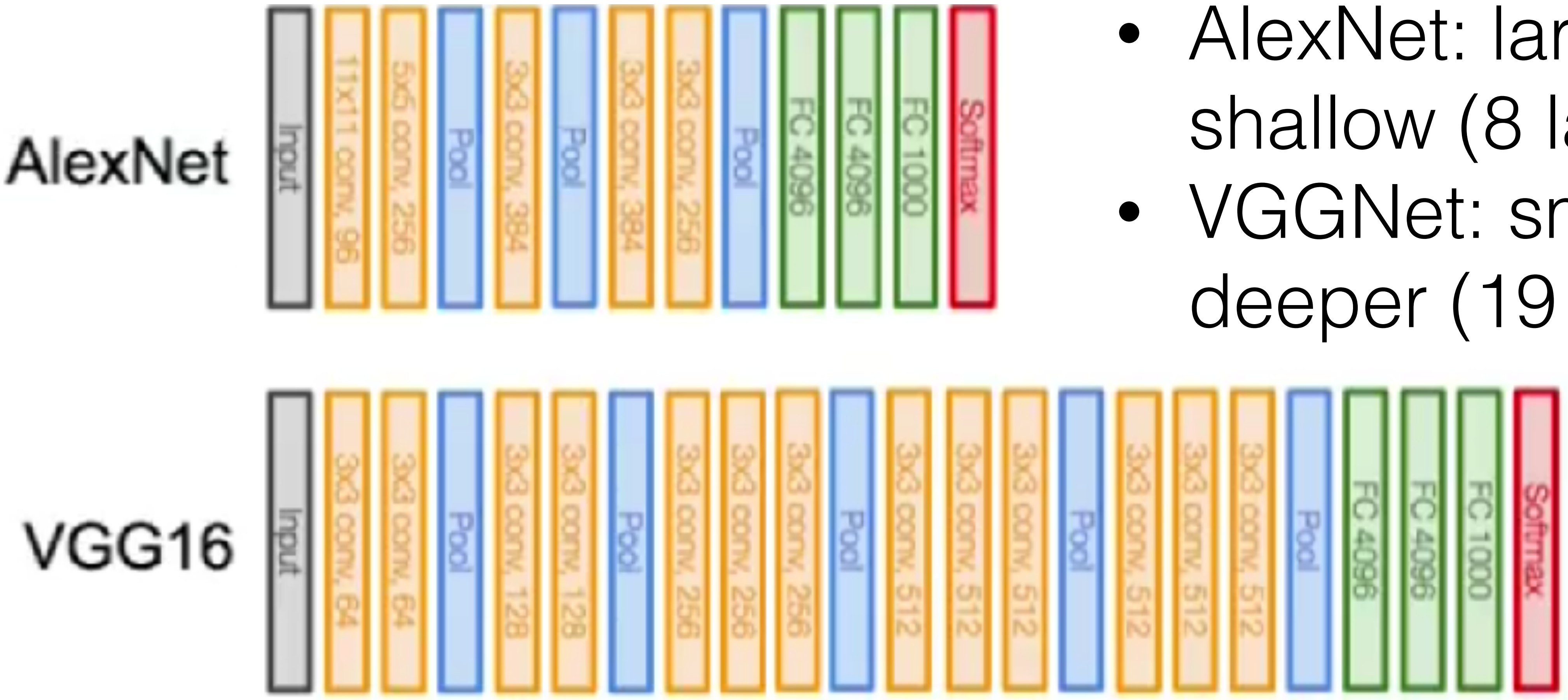
VGGNet



- Parameters in total: 138M, Depth: 19 layers

Simonyan and Zissermann,Very Deep Convolutional Networks for Large Scale Image Recognition, 2014 https://arxiv.org/abs/1409.1556

VGGNet vs AlexNet

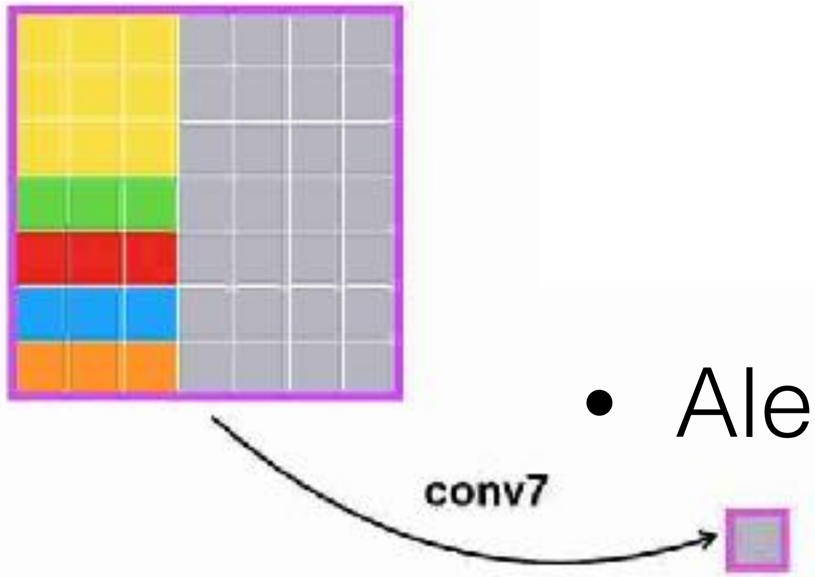


- AlexNet: large filters shallow (8 layers)
- VGGNet: small filters deeper (19 layers)

- Parameters in total: 138M, Depth: 19 layers

Simonyan and Zissermann,Very Deep Convolutional Networks for Large Scale Image Recognition, 2014 https://arxiv.org/abs/1409.1556

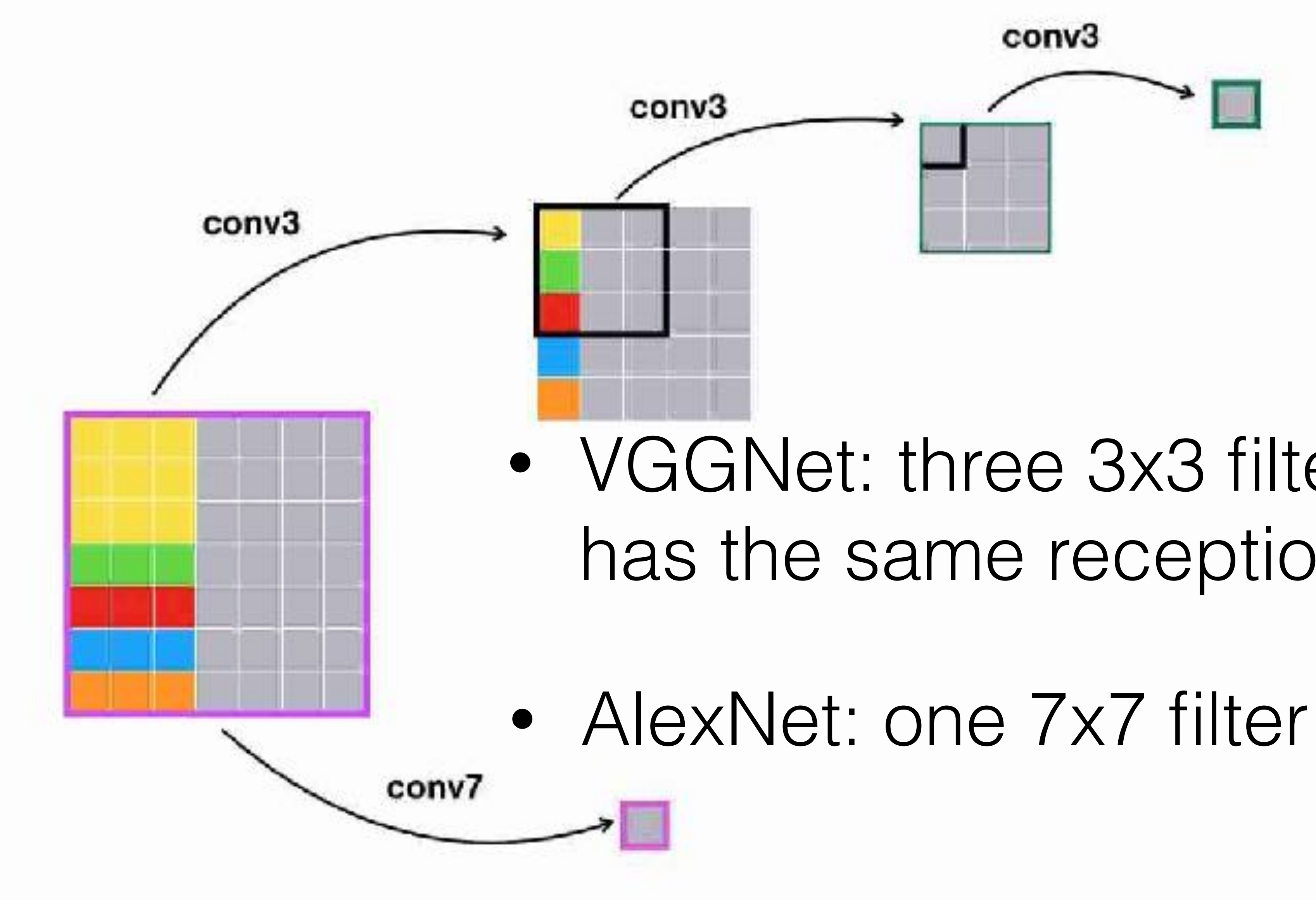
VGGNet vs AlexNet



- AlexNet: one 7x7 filter (49+1 params)

Image from: https://mc.ai/cnn-architectures-vggnet/
Simonyan and Zissermann,Very Deep Convolutional Networks for Large Scale Image Recognition, 2014
https://arxiv.org/abs/1409.1556

VGGNet vs AlexNet



- VGGNet: three 3x3 filters (3x9+3 params) has the same reception filed

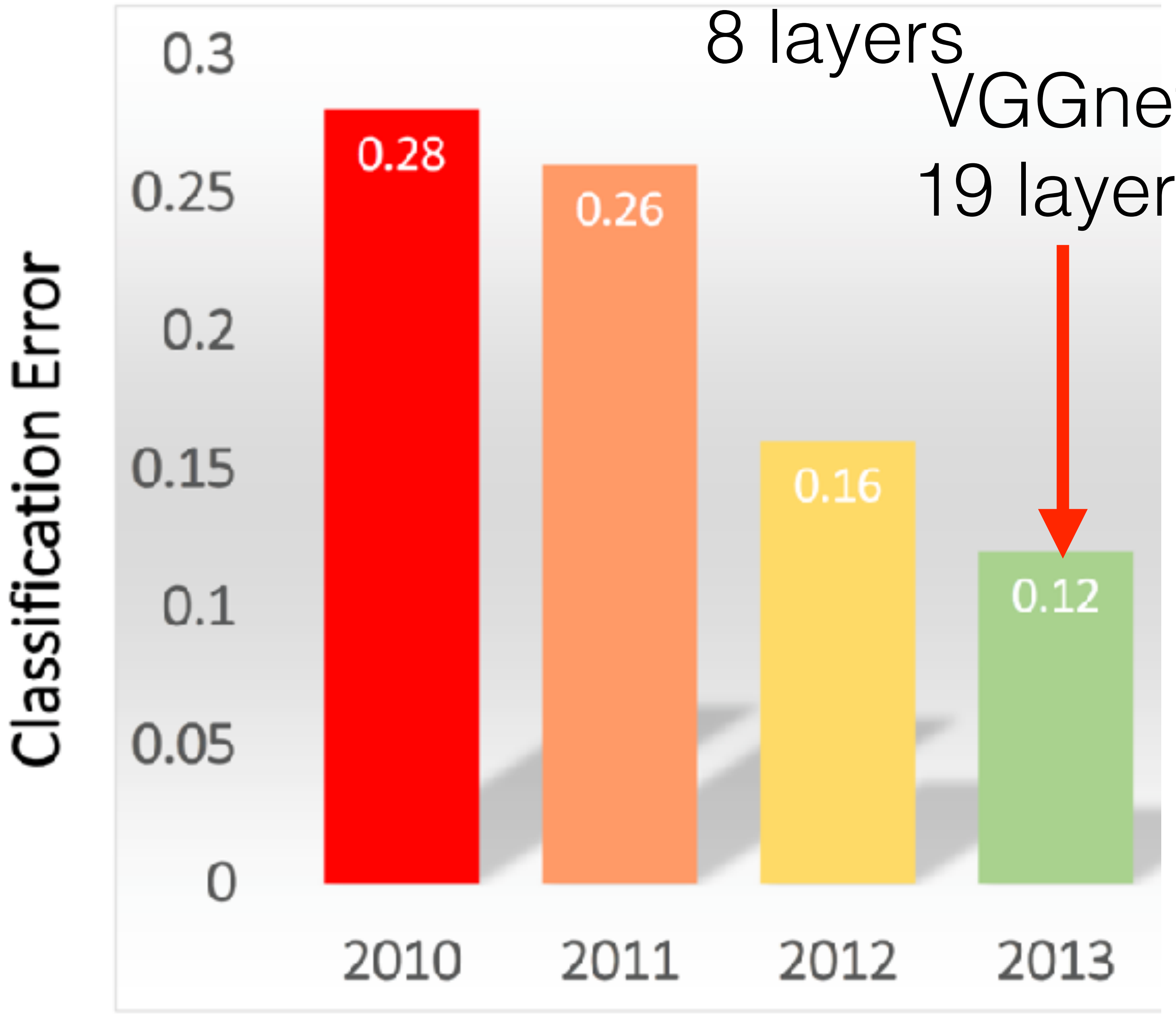
- AlexNet: one 7x7 filter (49+1 params)

Image from: https://mc.ai/cnn-architectures-vggnet/

Simonyan and Zissermann,Very Deep Convolutional Networks for Large Scale Image Recognition, 2014 https://arxiv.org/abs/1409.1556

IMAGENET

Classification results

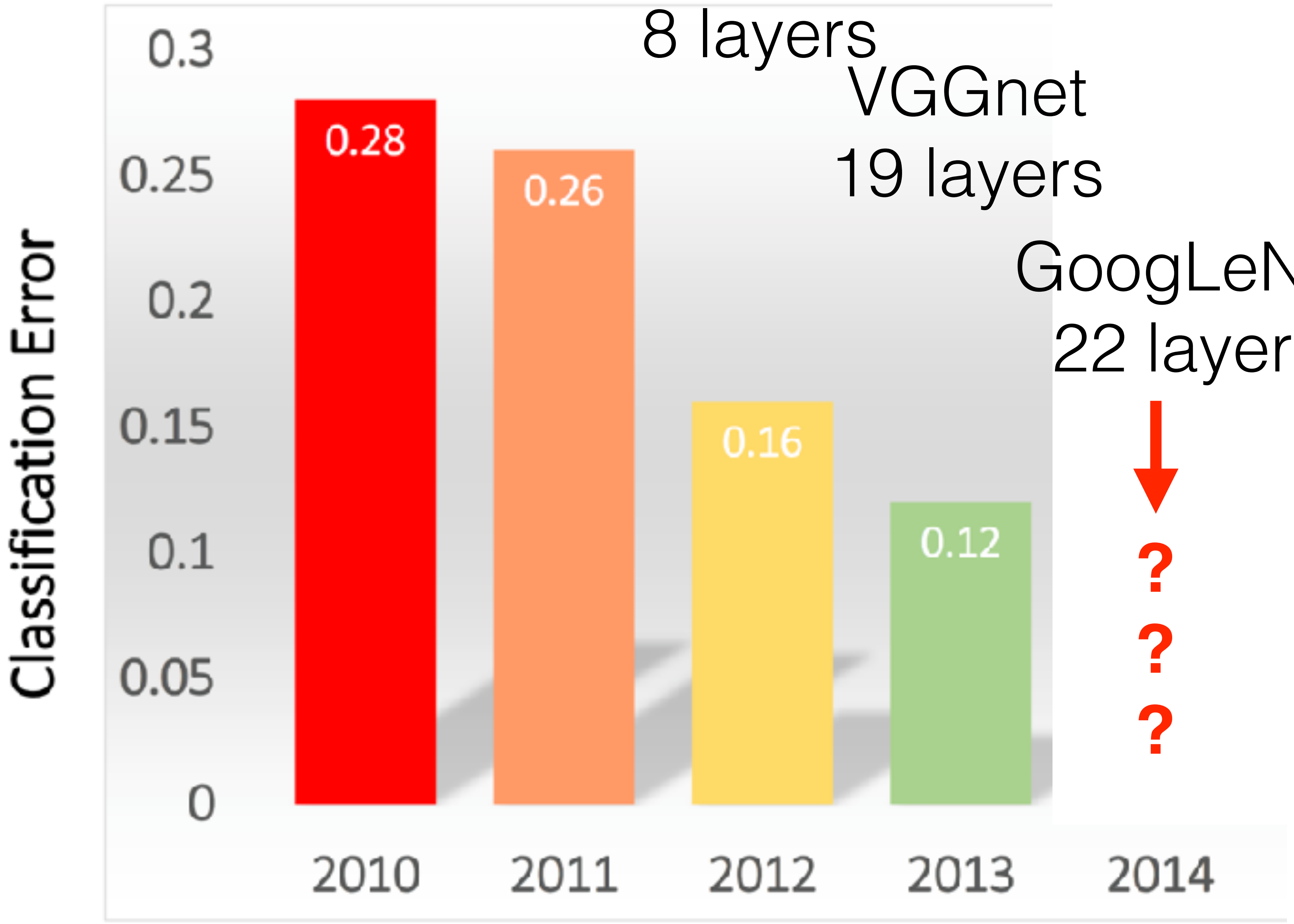


AlexNet
8 layers

VGGnet
19 layers

IMAGENET

Classification results

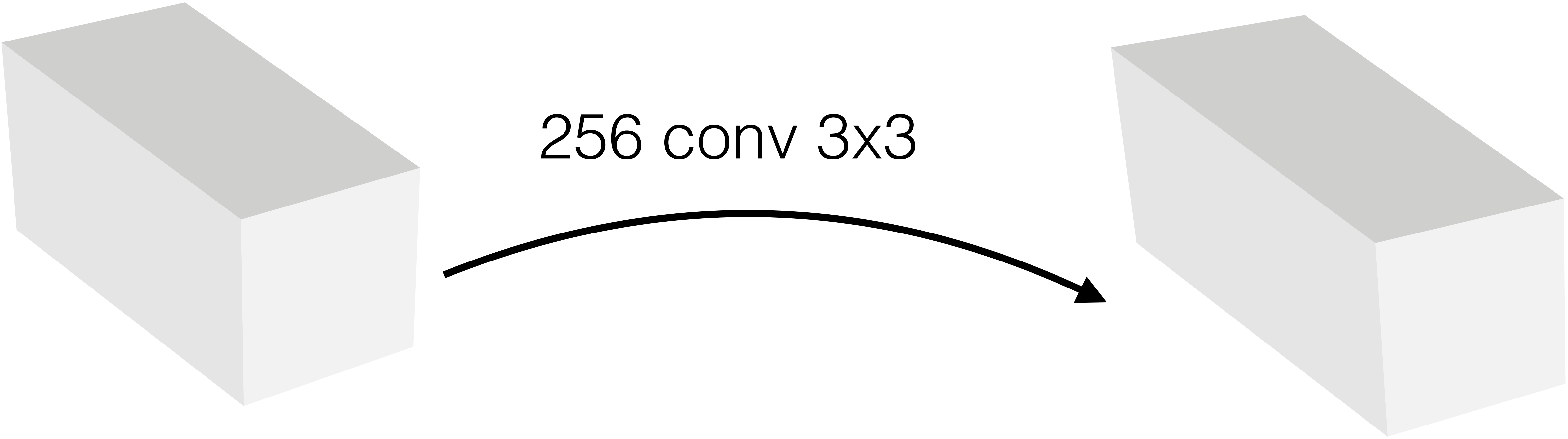


AlexNet
8 layers

VGGnet
19 layers

GoogLeNet
22 layers

GoogLeNet: concatenation of inception modules:

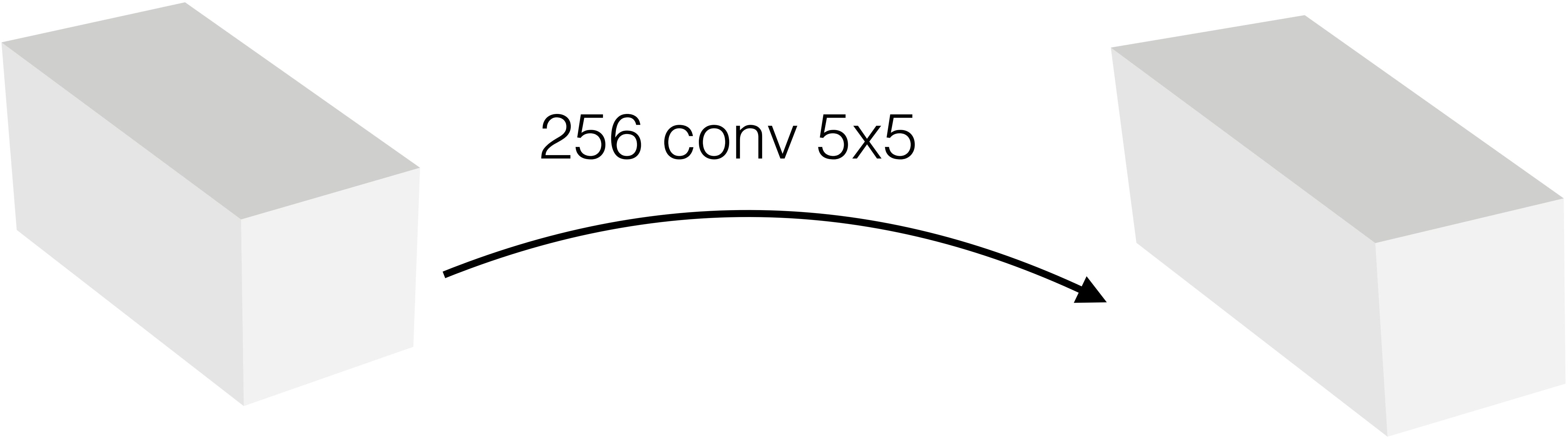


28x28x192

28x28x256

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

GoogLeNet: concatenation of inception modules:

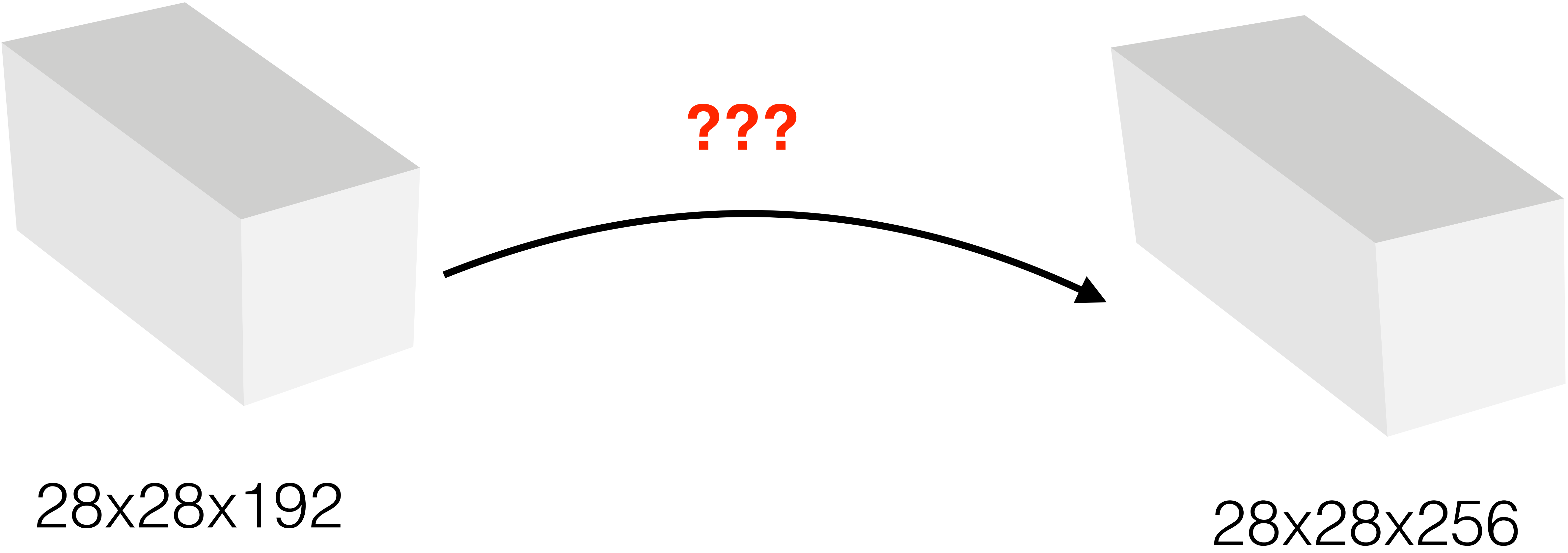


28x28x192

28x28x256

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

GoogLeNet: concatenation of inception modules:



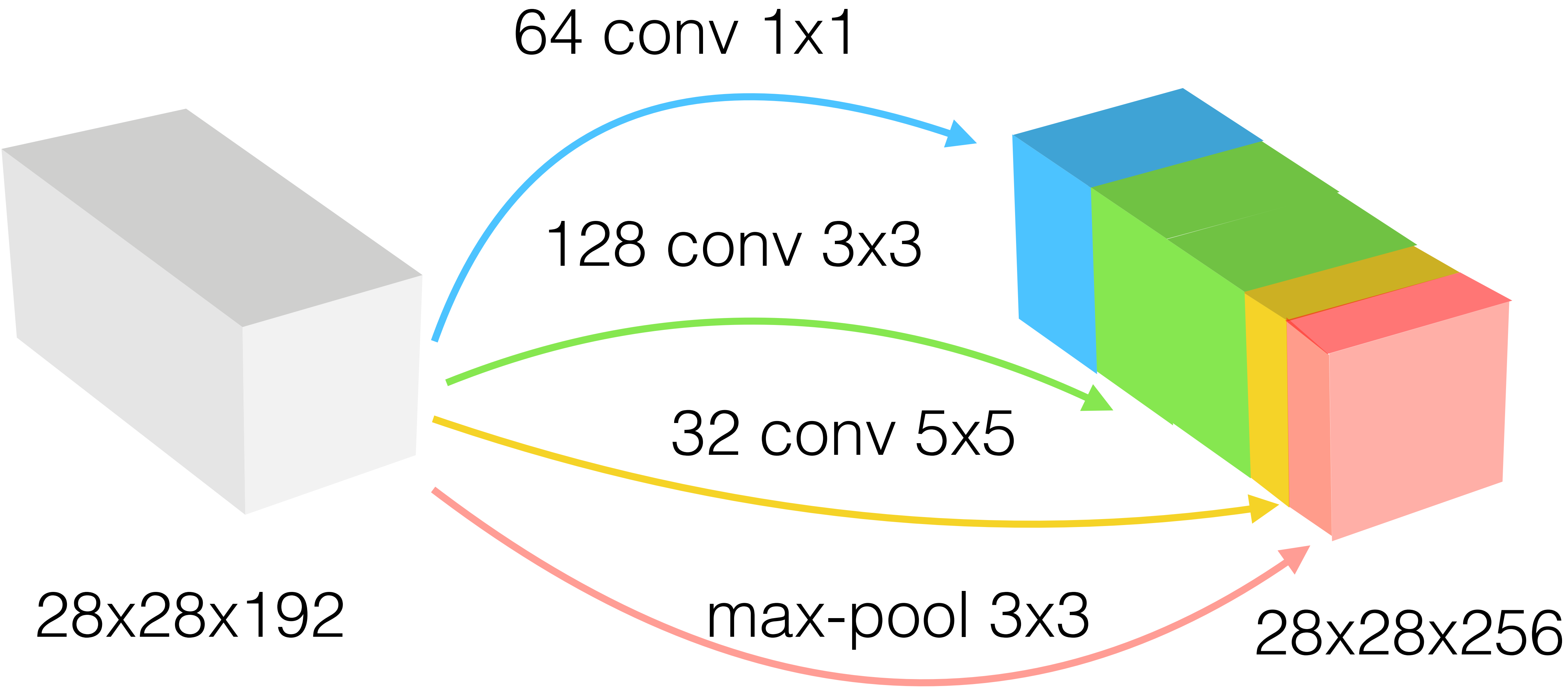
???

28x28x192

28x28x256

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

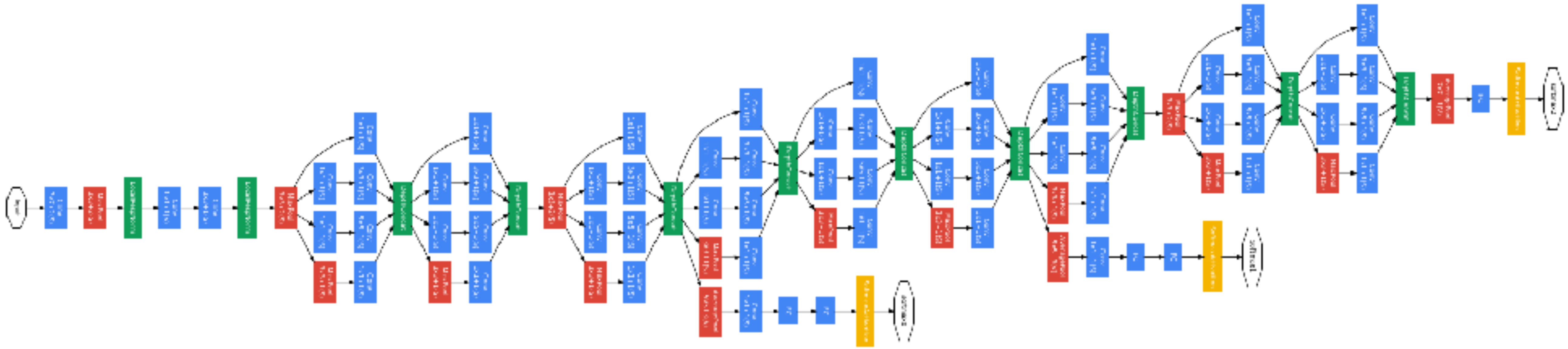
GoogLeNet: concatenation of inception modules:



Too many operations! => simplification using 1x1 conv

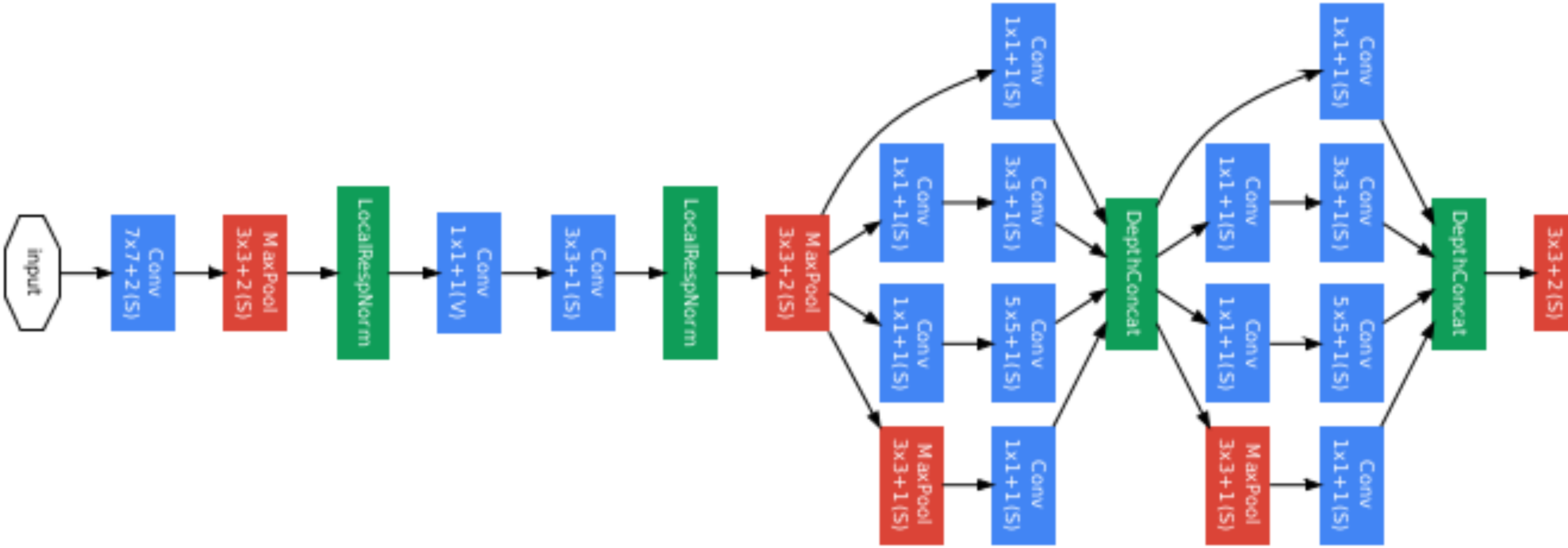
Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

GoogLeNet



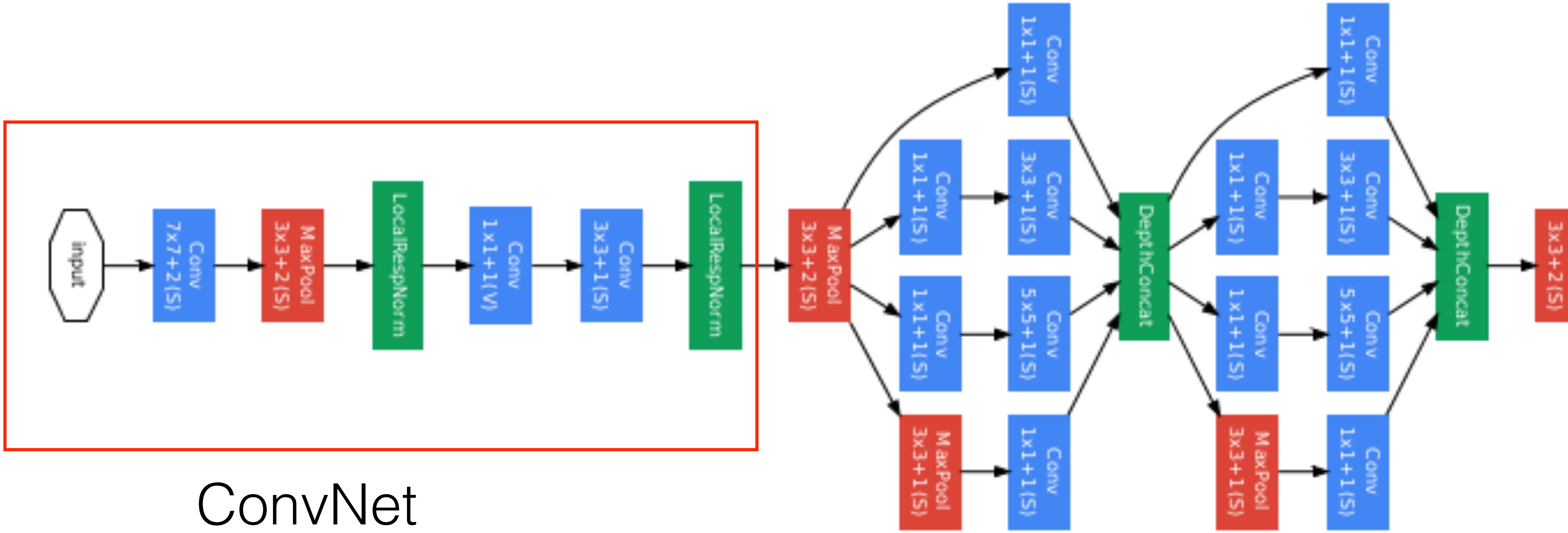
Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

GoogLeNet



Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

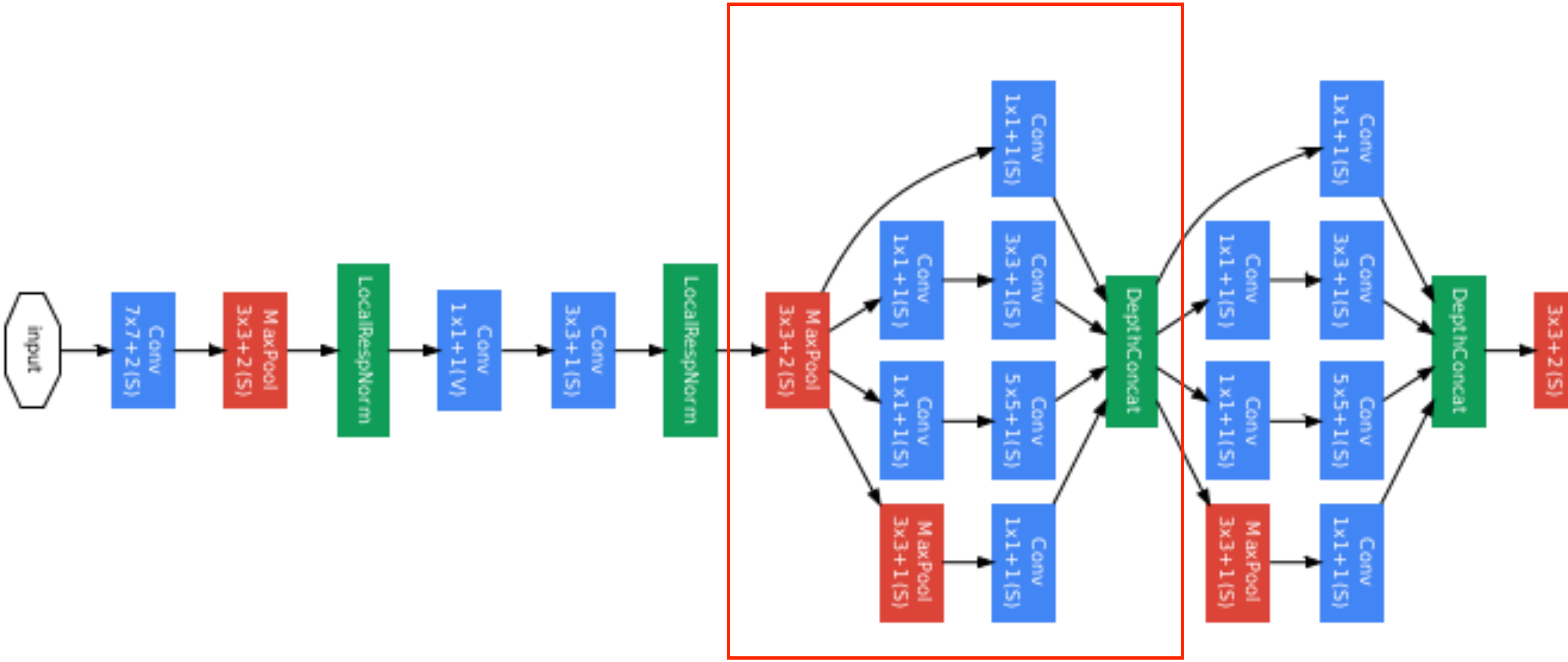
GoogLeNet



ConvNet

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

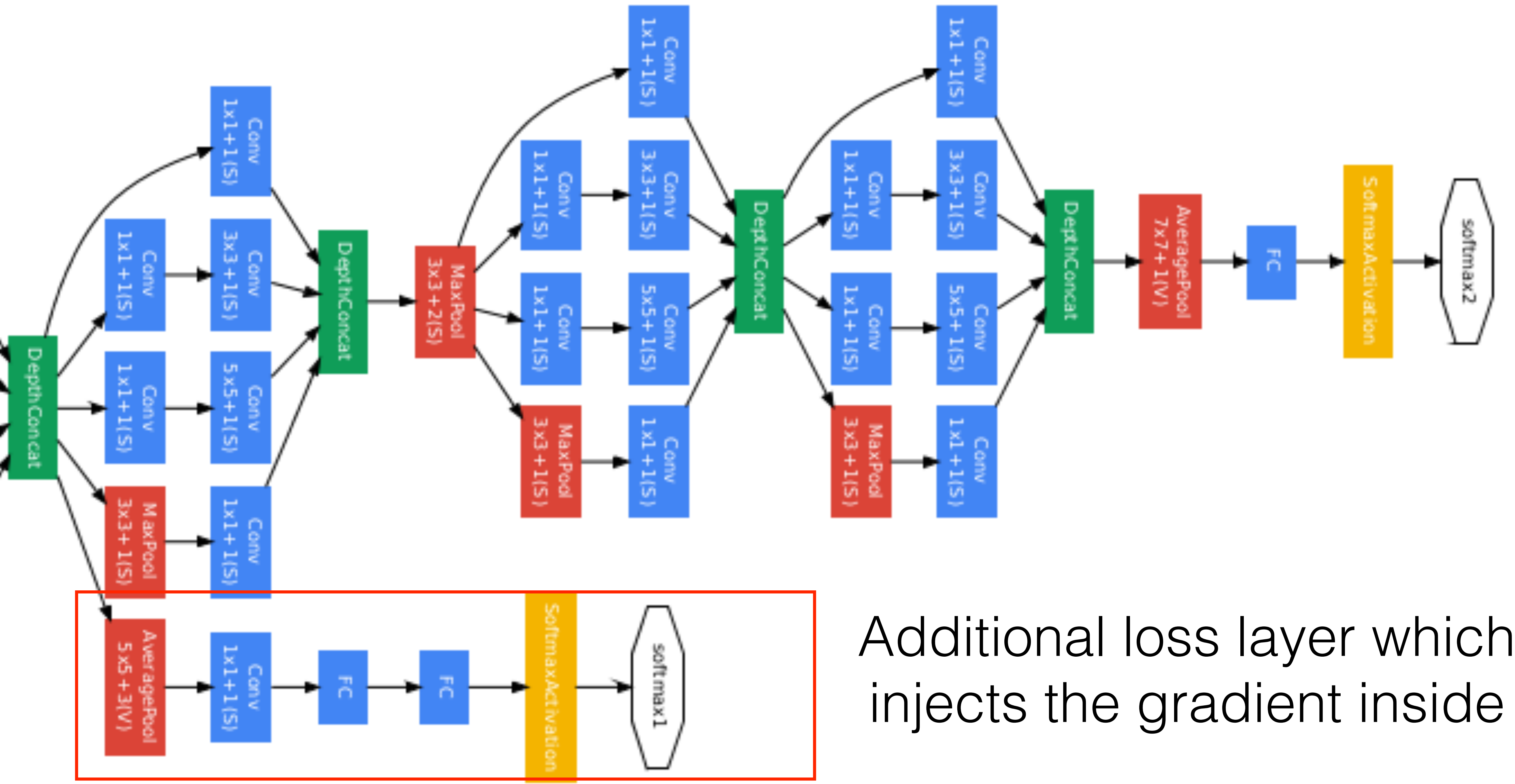
GoogLeNet



Inception module

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

GoogLeNet

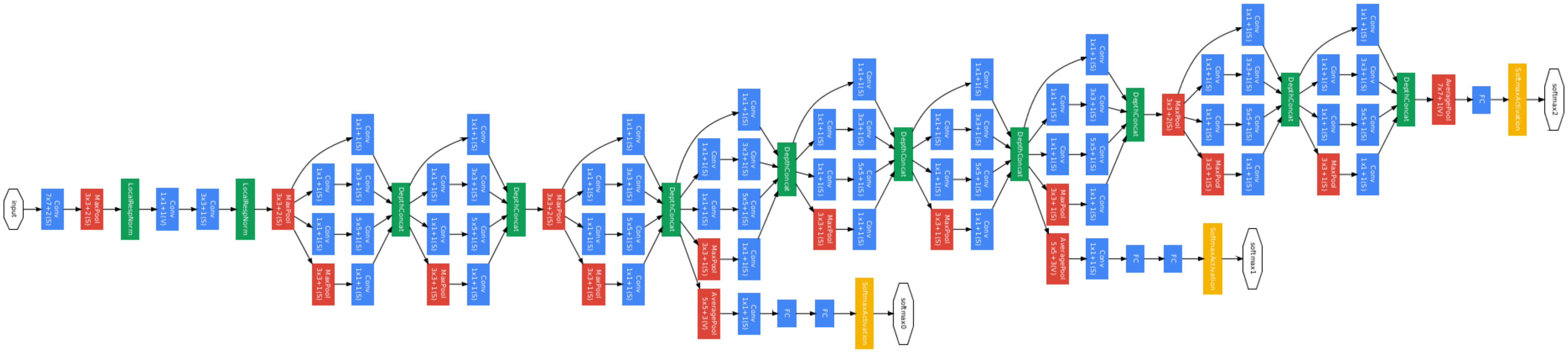


Additional loss layer which injects the gradient inside

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

image source: http://joelouismarino.github.io/images/blog_images/blog_googlenet_keras/googlenet_diagram.png

GoogLeNet



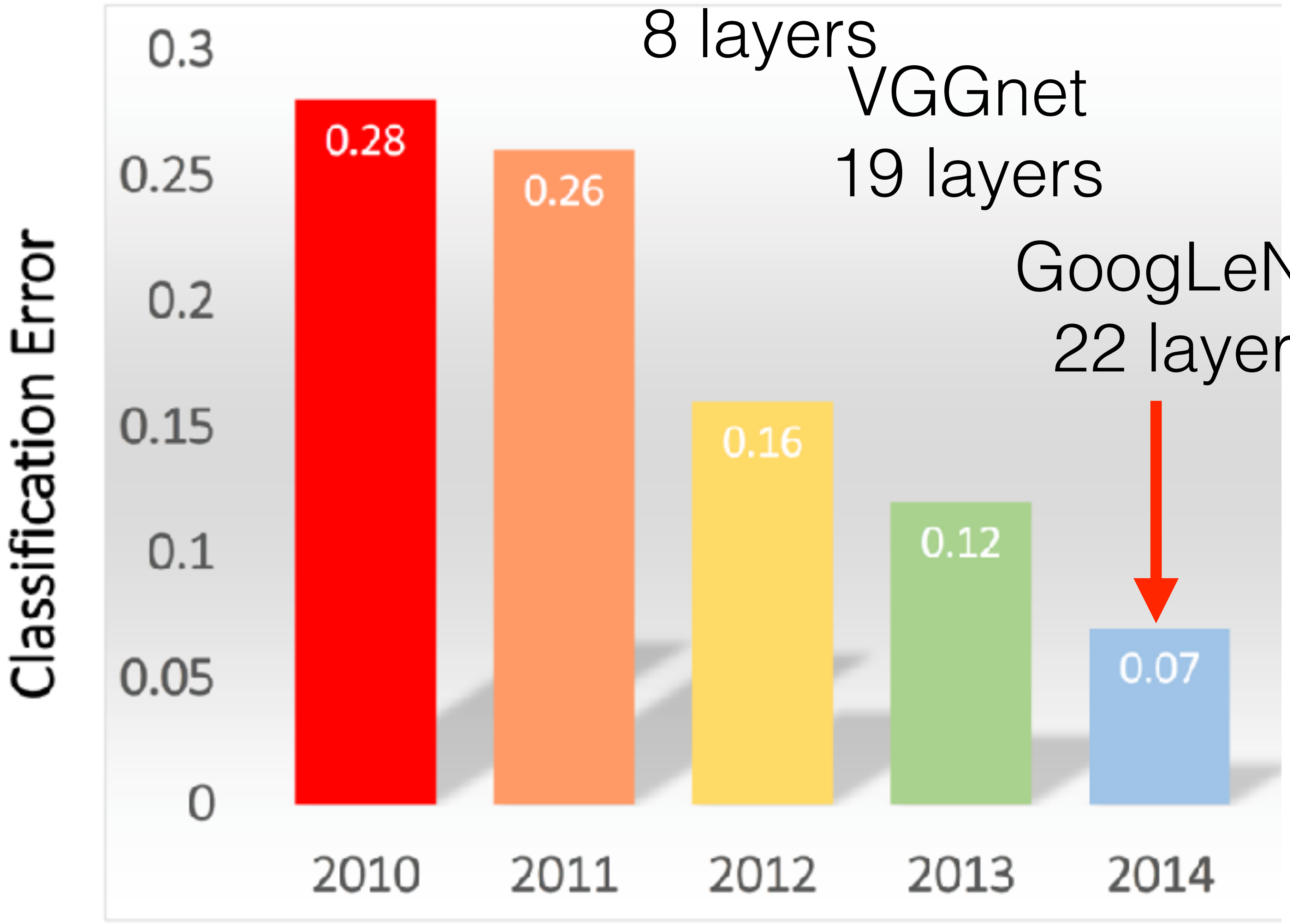
- 12x fewer parameters than AlexNet
- depth 22 layers
- training: few high-end GPU about a week

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

image source: http://joelouismarino.github.io/images/blog_images/blog_googlenet_keras/googlenet_diagram.png

IMAGENET

Classification results



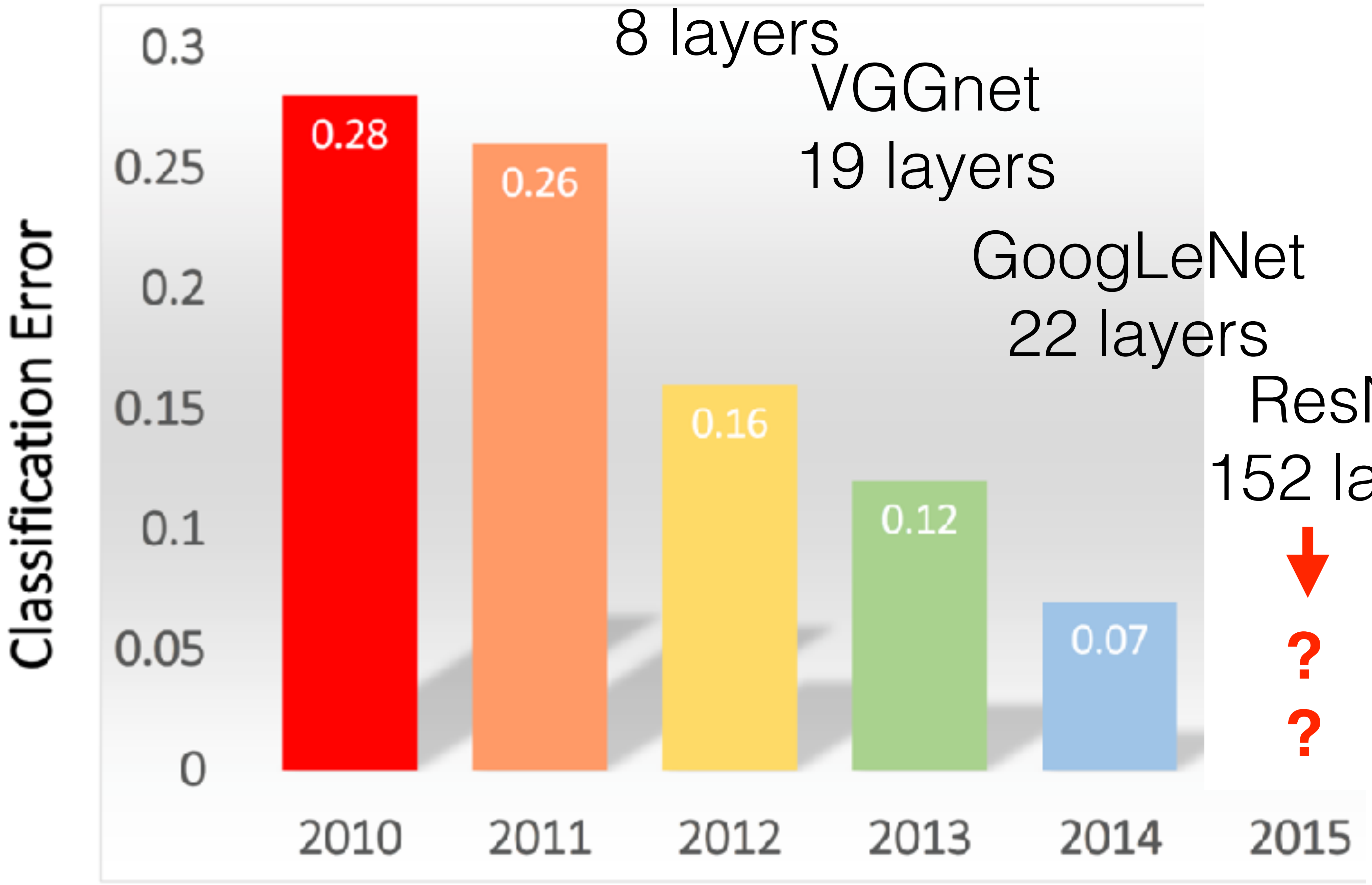
AlexNet
8 layers

VGGnet
19 layers

GoogLeNet
22 layers

IMAGENET

Classification results



AlexNet
8 layers

VGGnet
19 layers

GoogLeNet
22 layers

ResNet
152 layers

↓

?

?

ResNet

The main idea is as follows:

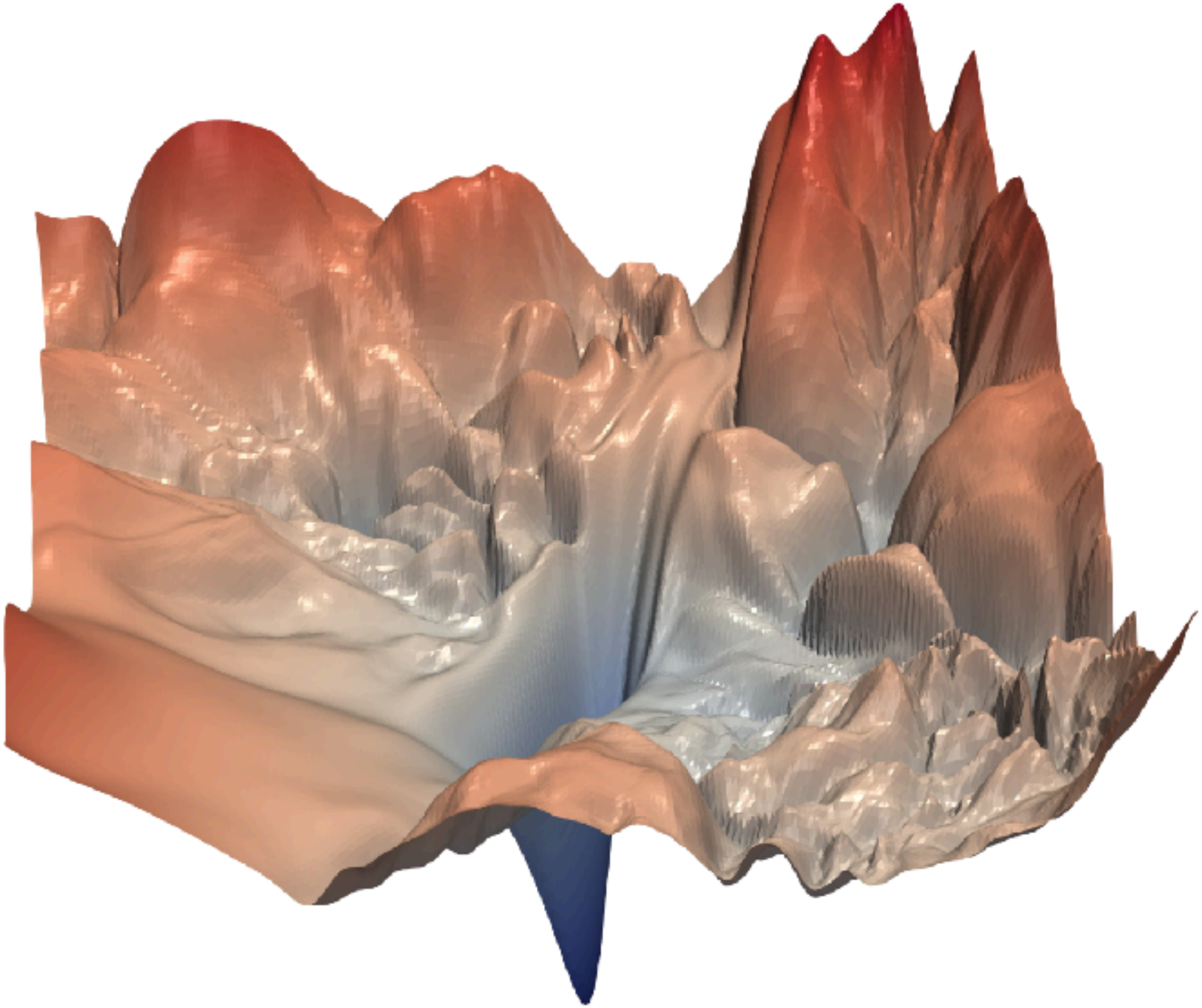
Well said Leo, well said

- deeper ConvNet architectures yielded higher errors.
- error was higher even in training => no overfitting
- problem stems from the optimization (vanishing gradient)

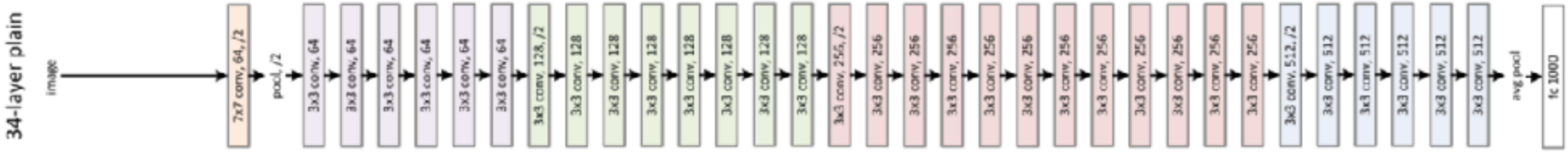
He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385

Visualizing Loss Landscape of Neural Nets

https://arxiv.org/pdf/1712.09913.pdf

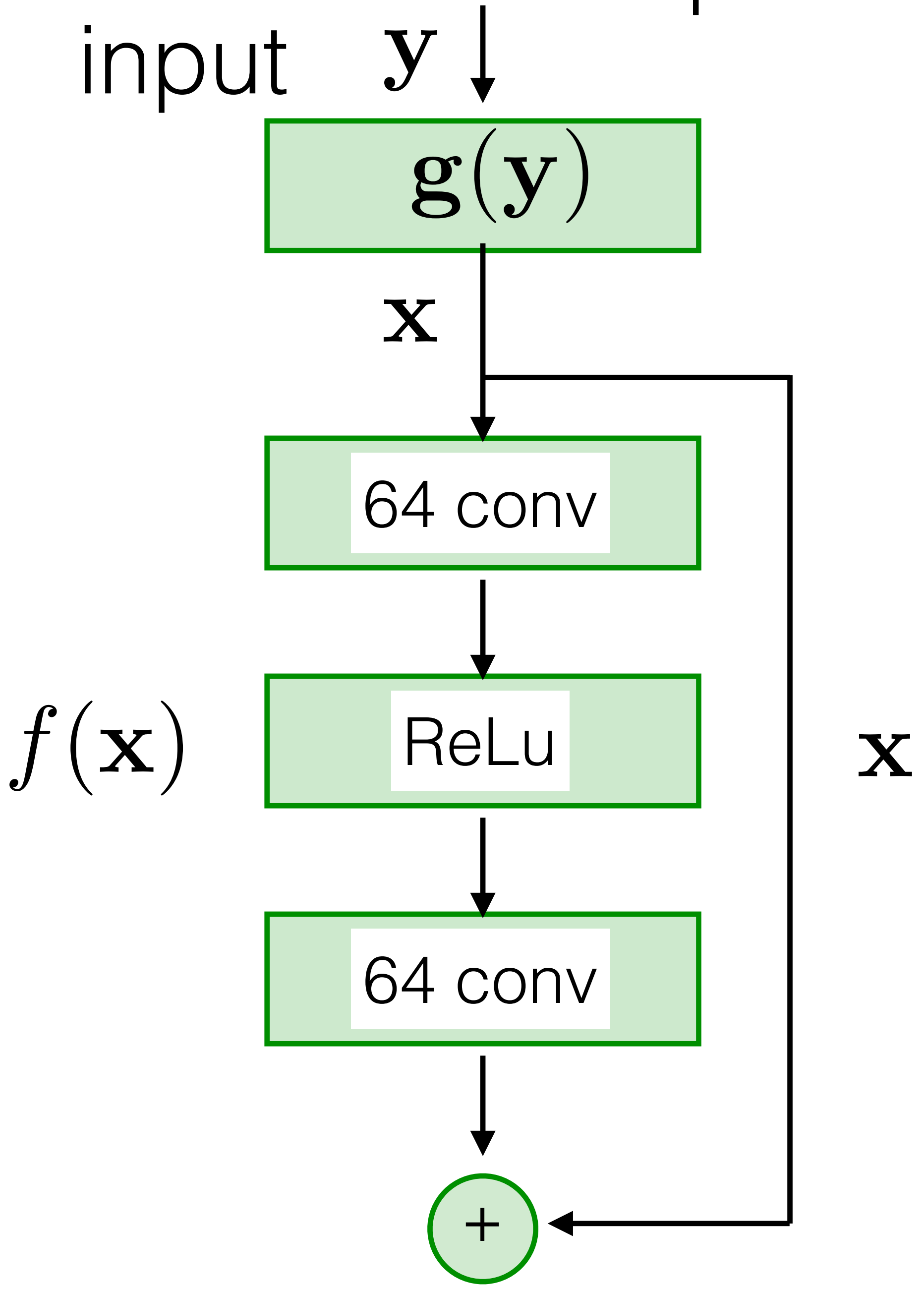


ResNet



- Gradient in deep nets vanishes quickly
- In straightforward conv architecture the weights from the beginning of the net has minor influence on the output !!!
- In backward-pass the gradient of weights in the first layer is computed by multiplication of the all following gradients => prone to diminish!

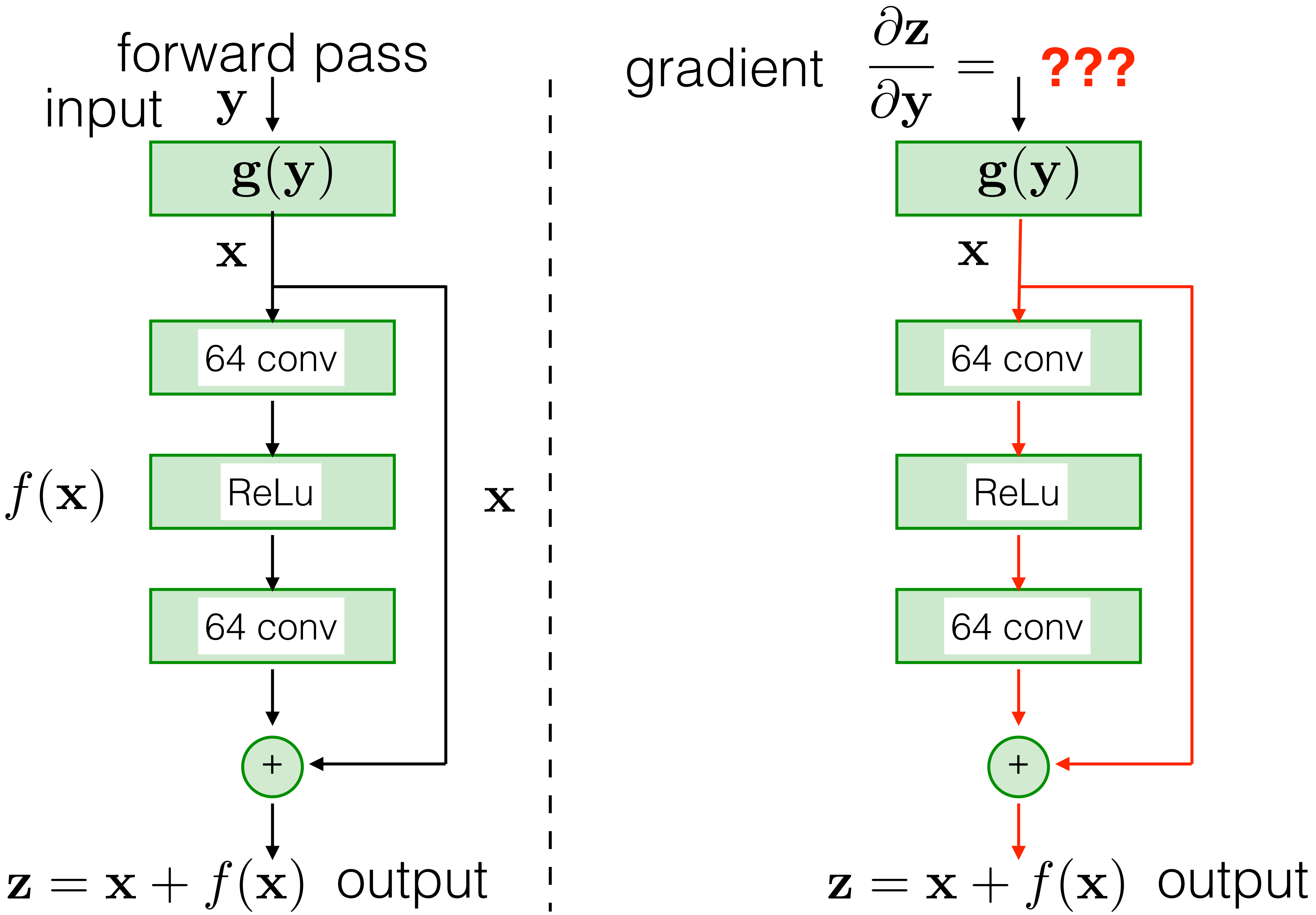
ResNet: skip connections layer preserve gradient



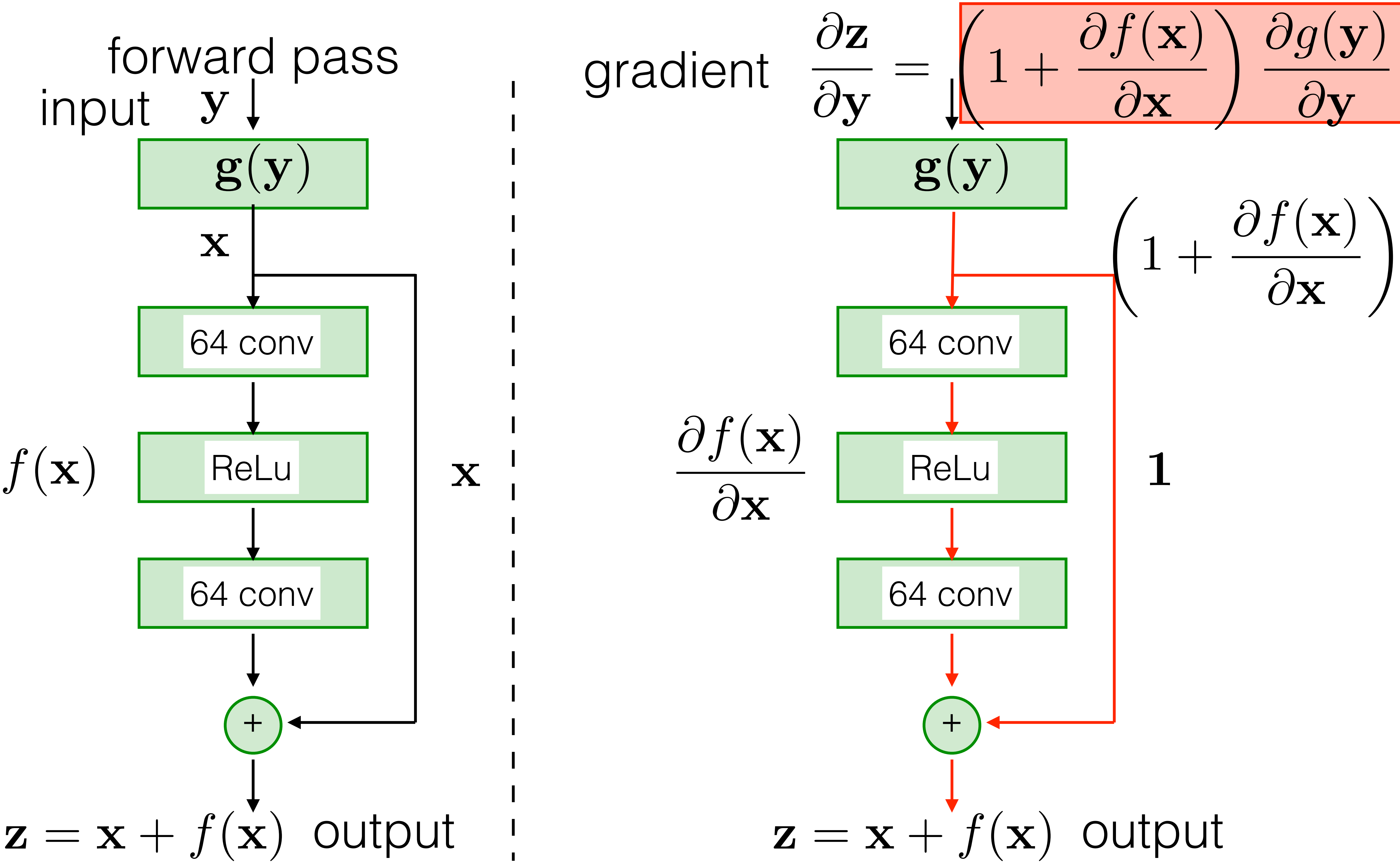
$\mathbf{y} = \mathbf{x} + f(\mathbf{x})$ output

$$\frac{\partial g(\mathbf{y})}{\partial \mathbf{y}}$$

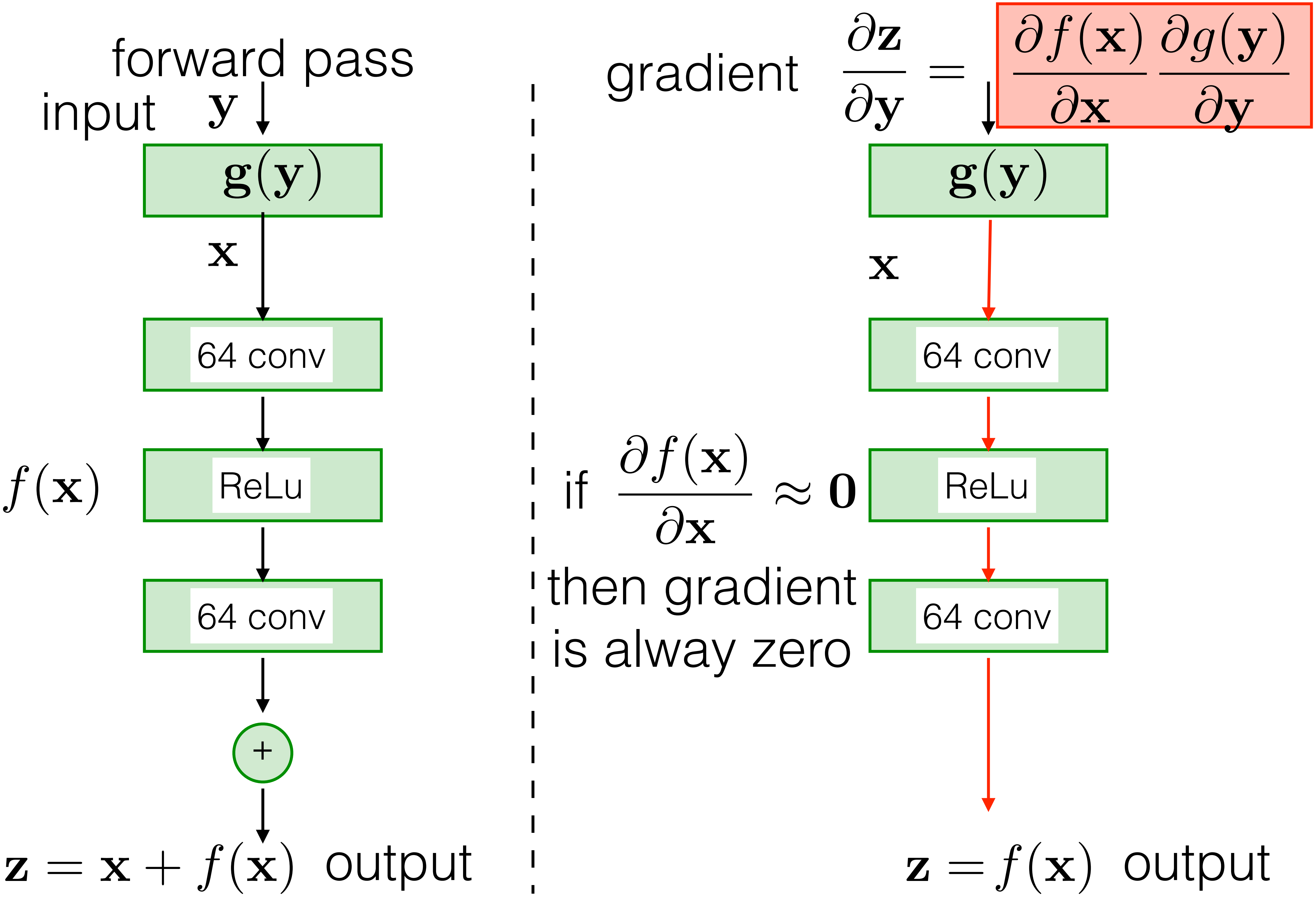
He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385



He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385


forward pass
input $\mathbf{y}$
$\mathbf{g}(\mathbf{y})$
$\mathbf{x}$
64 conv
$f(\mathbf{x})$
ReLu
$\mathbf{x}$
64 conv
+
$\mathbf{z} = \mathbf{x} + f(\mathbf{x})$ output
gradient $\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = \left(1 + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right) \frac{\partial g(\mathbf{y})}{\partial \mathbf{y}}$
$\mathbf{g}(\mathbf{y})$
$\left(1 + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right)$
64 conv
$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$
ReLu
$\mathbf{1}$
64 conv
+
$\mathbf{z} = \mathbf{x} + f(\mathbf{x})$ output

He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385



forward pass

input $\mathbf{y}$

$\mathbf{g}(\mathbf{y})$

$\mathbf{x}$

64 conv

$f(\mathbf{x})$ ReLu

64 conv

+

$\mathbf{z} = \mathbf{x} + f(\mathbf{x})$ output

gradient $\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = \boxed{\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial g(\mathbf{y})}{\partial \mathbf{y}}}$

$\mathbf{g}(\mathbf{y})$

$\mathbf{x}$

64 conv

ReLu

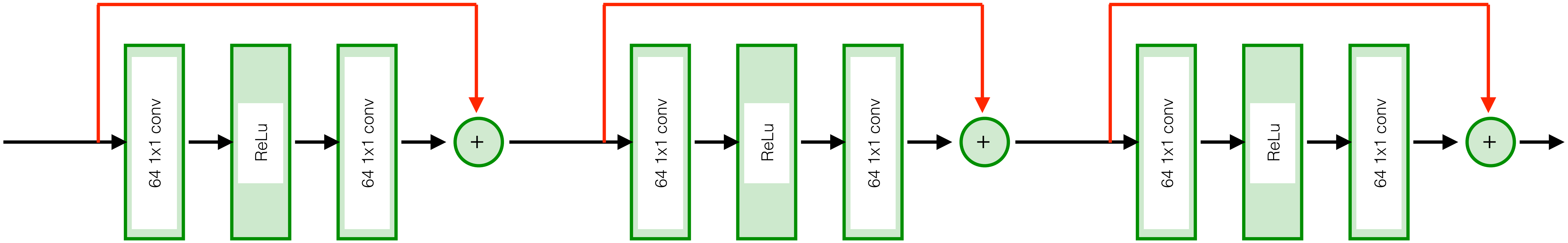
64 conv

if $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \approx \mathbf{0}$ then gradient is alway zero

$\mathbf{z} = f(\mathbf{x})$ output

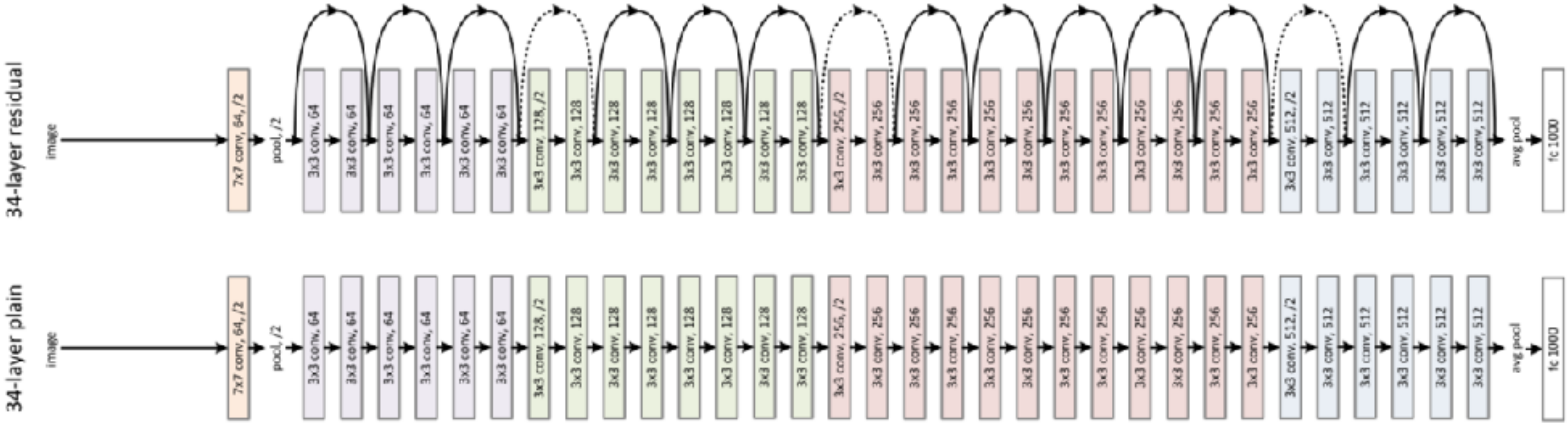
Compare with gradient without skip connection !!!!

ResNet - gradient flow



- Skip connections partially avoids diminishing gradient
- The weights from the beginning of the net has strong influence on the output!
-

ResNet: deep ConvNet with skip connections



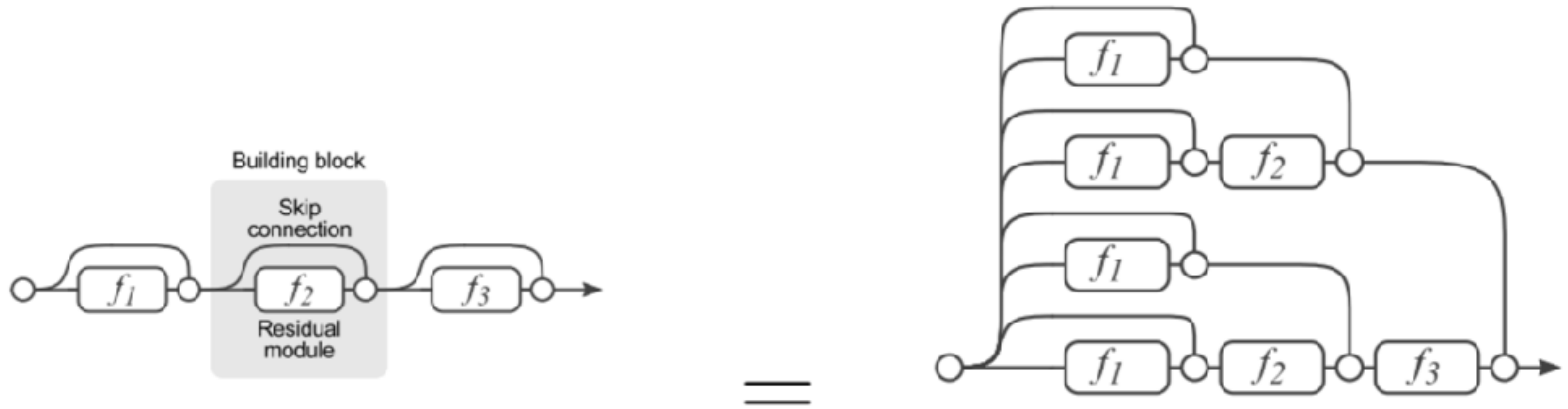
- Competition time about 152 layers ResNet,
- Recently they are able to train 1k layers ResNet
- Initialization with zero weights is meaningful
- Better gradient flow

https://www.kaggle.com/keras/resnet50/home

He et al. Going Deeper with Convolutions, CVPR, 2015

https://arxiv.org/abs/1512.03385

Unraveled view of ResNet



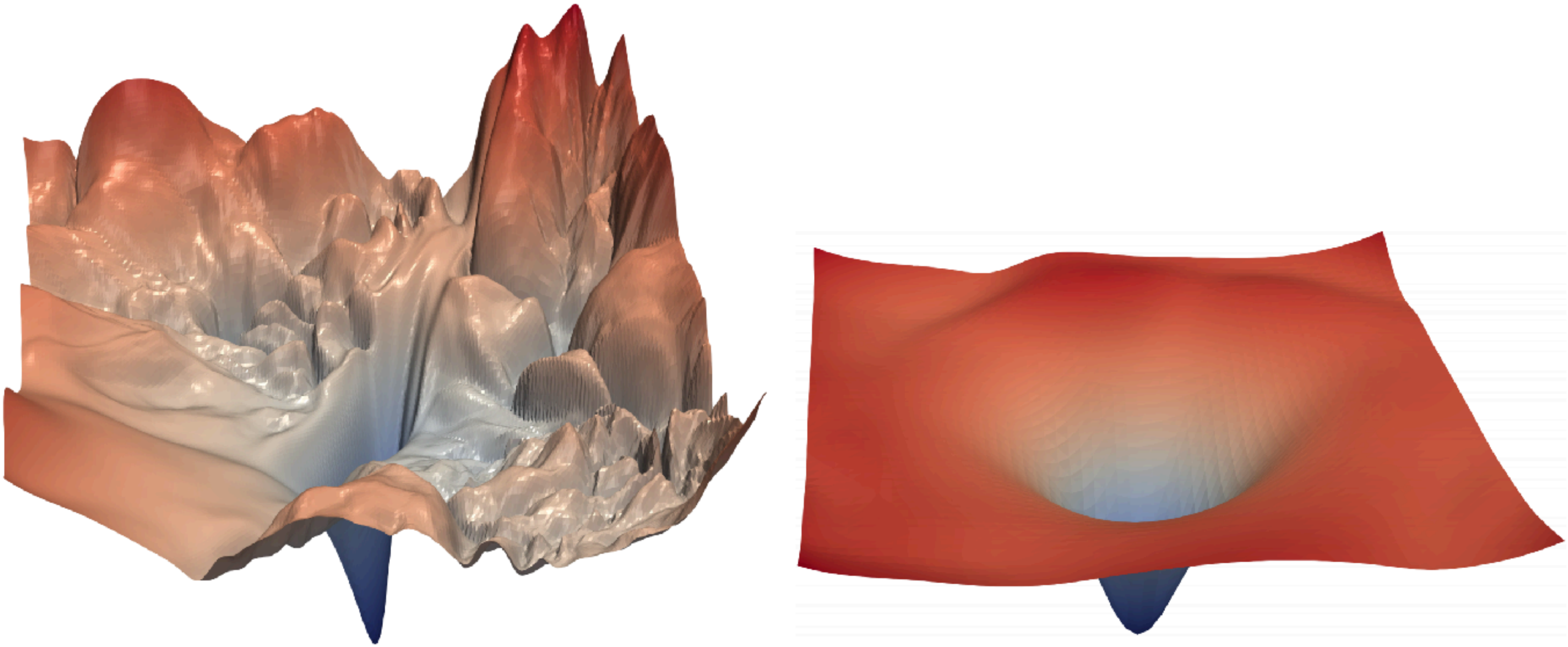
(a) Conventional 3-block residual network

(b) Unraveled view of (a)

- There exists many “almost independent” paths
- Unravelling of ResNet architecture allows to understand robustness wrt noise and layer removal

Visualizing Loss Landscape of Neural Nets

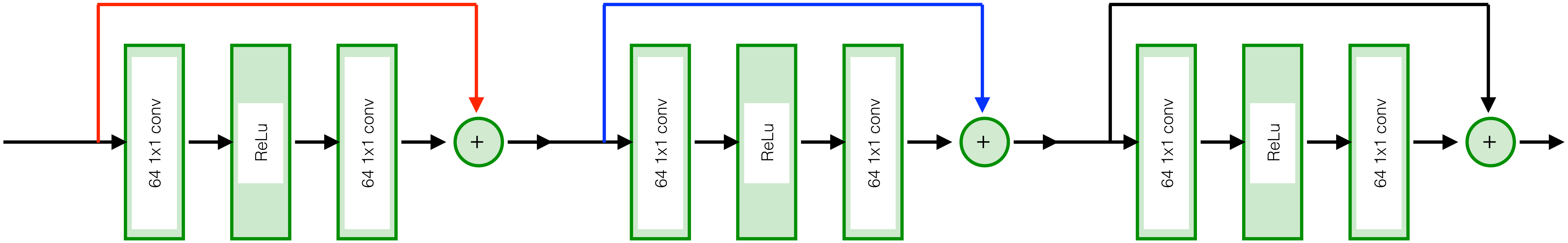
https://arxiv.org/pdf/1712.09913.pdf



(a) without skip connections

(b) with skip connections

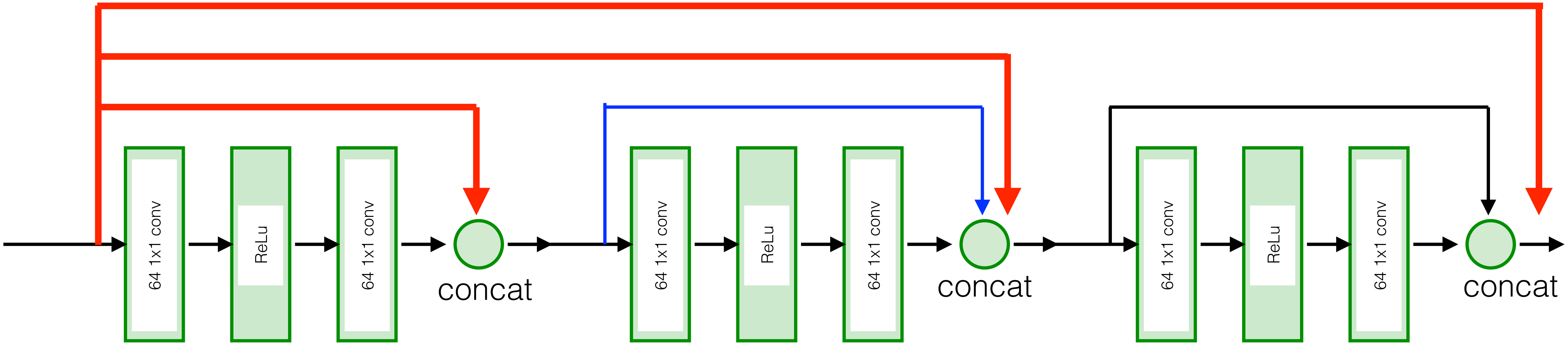
ResNet =>DenseNet



Start with multilayer ResNet architecture

Huang, Densely Connected Convolutional Networks, CVPR 2017. https://arxiv.org/abs/1608.06993

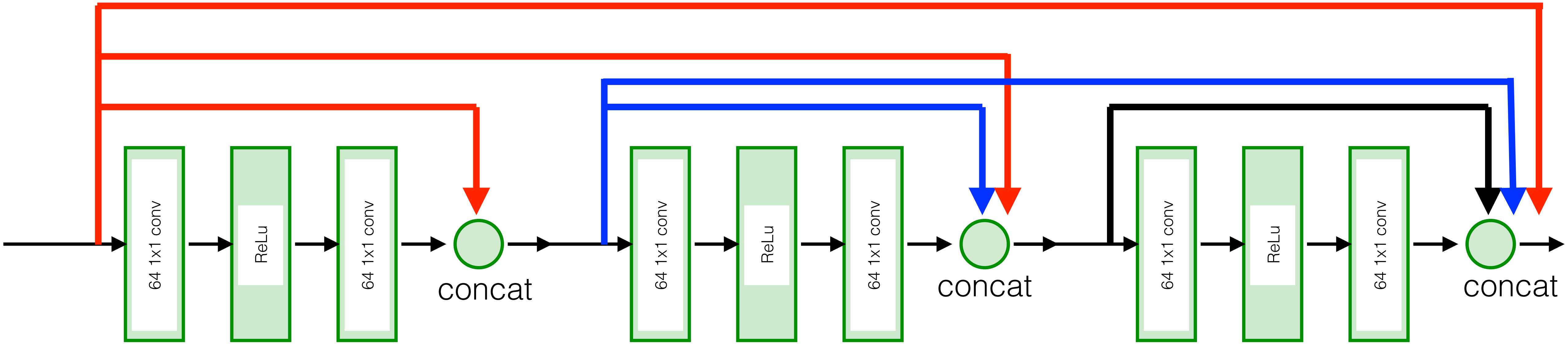
DenseNet



- Directly propagate each feature map to all following layers

Huang, Densely Connected Convolutional Networks, CVPR 2017. https://arxiv.org/abs/1608.06993

DenseNet

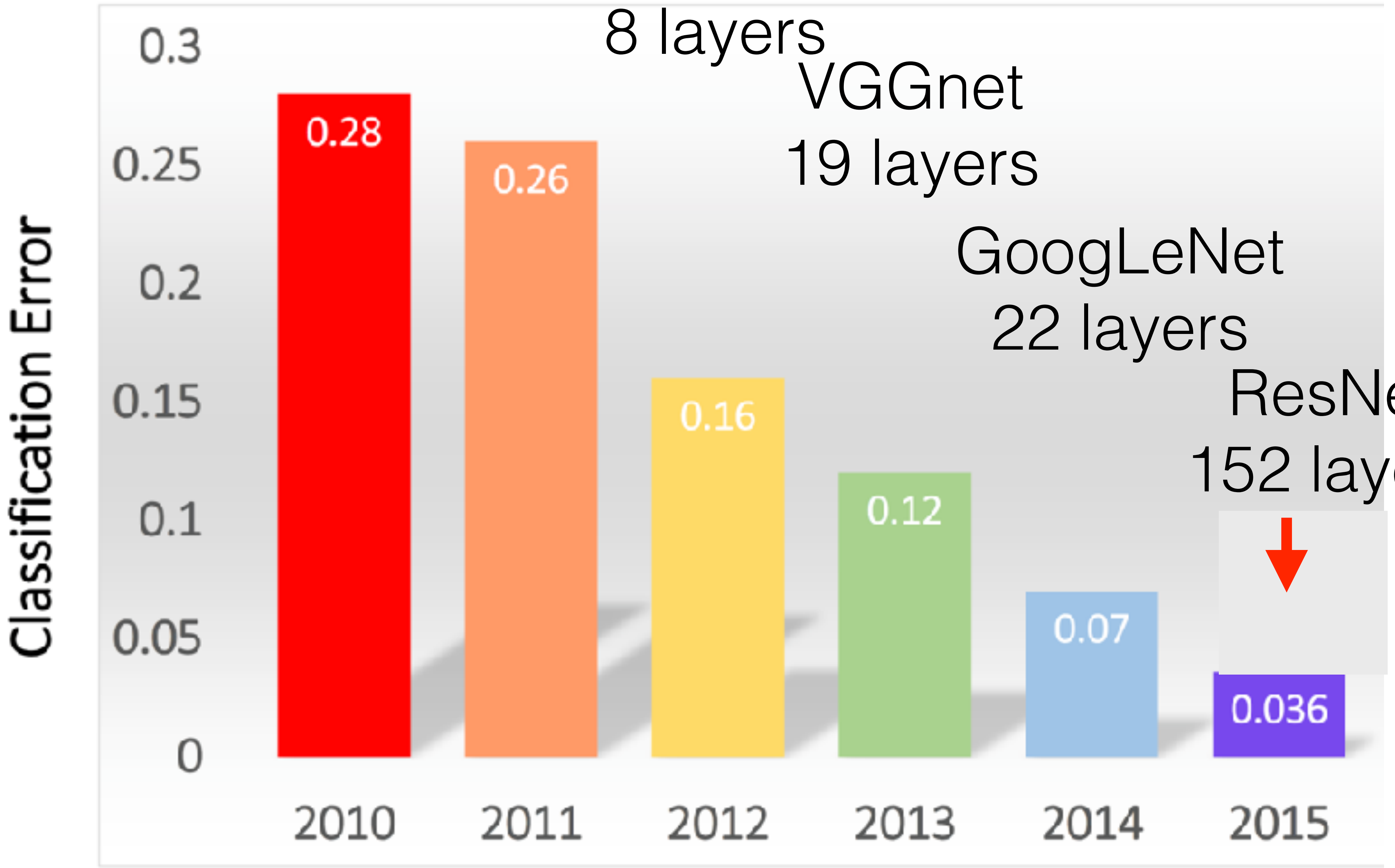


- Directly propagate each feature map to all following layers
- Improves gradient flow in backward pass

Huang, Densely Connected Convolutional Networks, CVPR 2017. https://arxiv.org/abs/1608.06993

IMAGENET

Classification results



AlexNet
8 layers

VGGnet
19 layers

GoogLeNet
22 layers

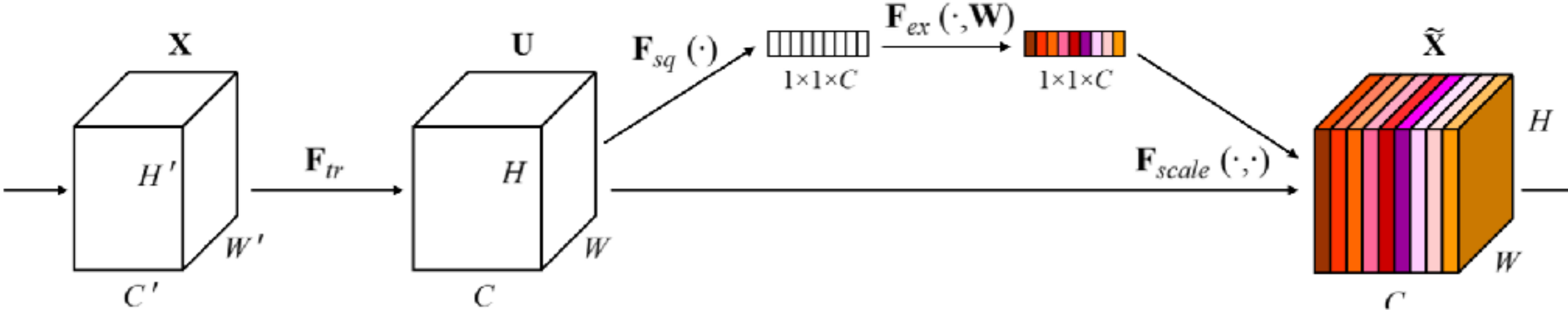
ResNet
152 layers

Human error around 5%

Squeeze and Excitation Networks [Hu et al, CVPR oral, 2017] https://arxiv.org/pdf/1709.01507.pdf

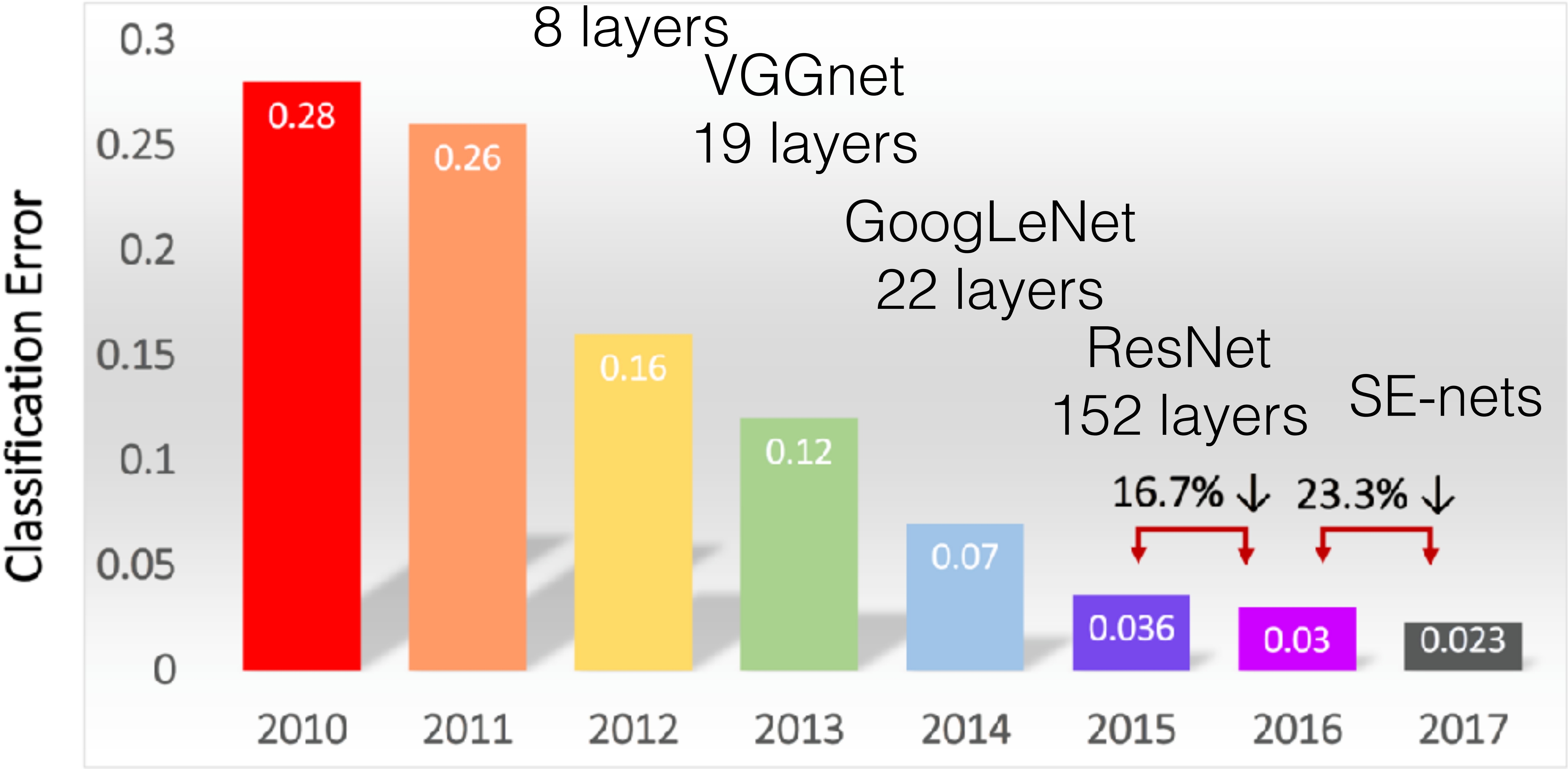
- Winner of ILSVRC 2017
- Enhancement of ResNet, InceptionNet and DenseNet architectures by SE blocks consistently decrease error on ImageNet, COCO, …

Squeeze and Excitation block



IMAGENET

Classification results



AlexNet
8 layers

VGGnet
19 layers

GoogLeNet
22 layers

ResNet
152 layers

SE-nets

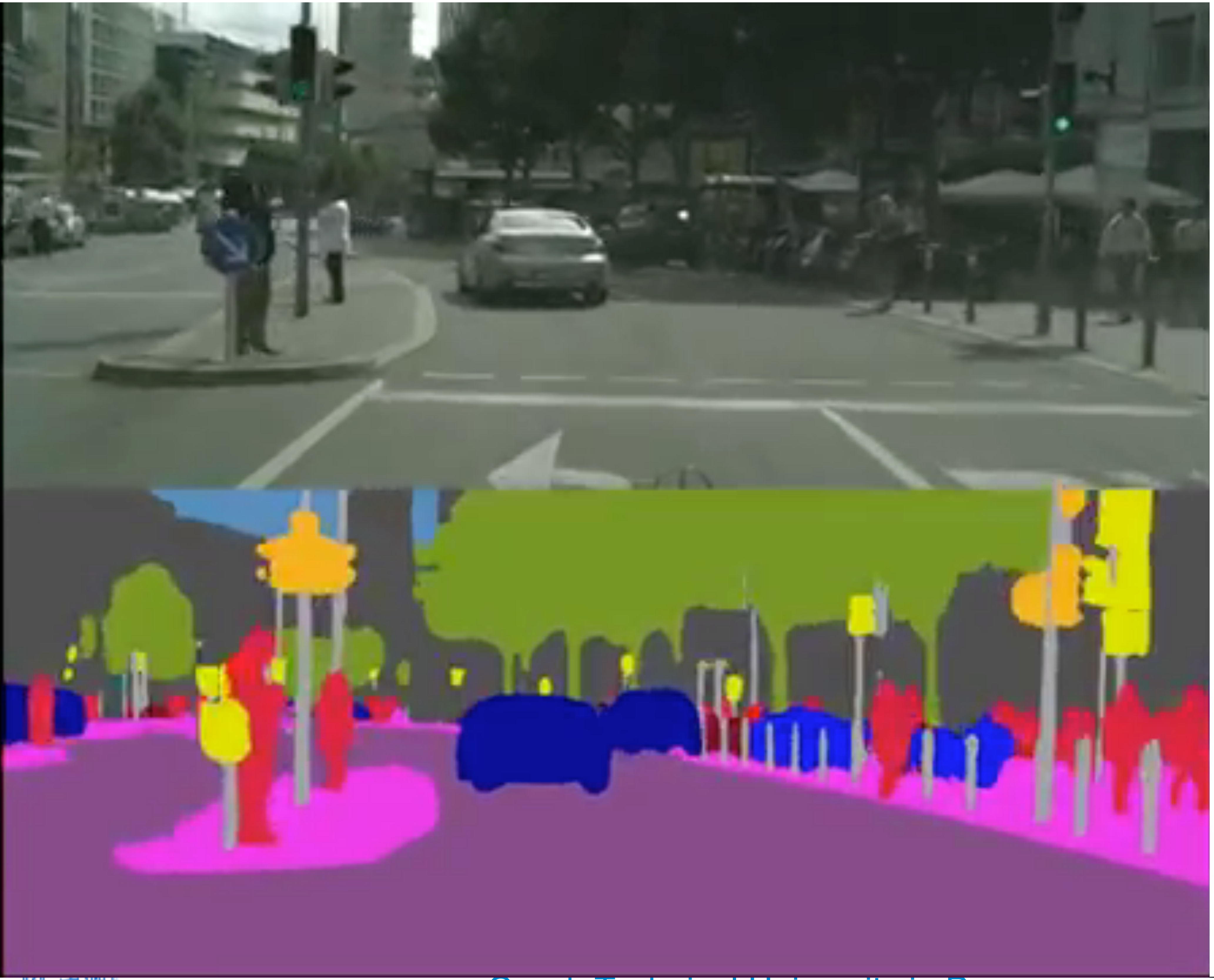
Summary classification architectures

- It seems that the deeper the better
- ResNet is easy, well-studied architecture=> consider as a starting point
- You should be careful about combining DropOut with BN https://arxiv.org/abs/1801.05134
- Capsule networks https://medium.com/ai^{3}-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Outline

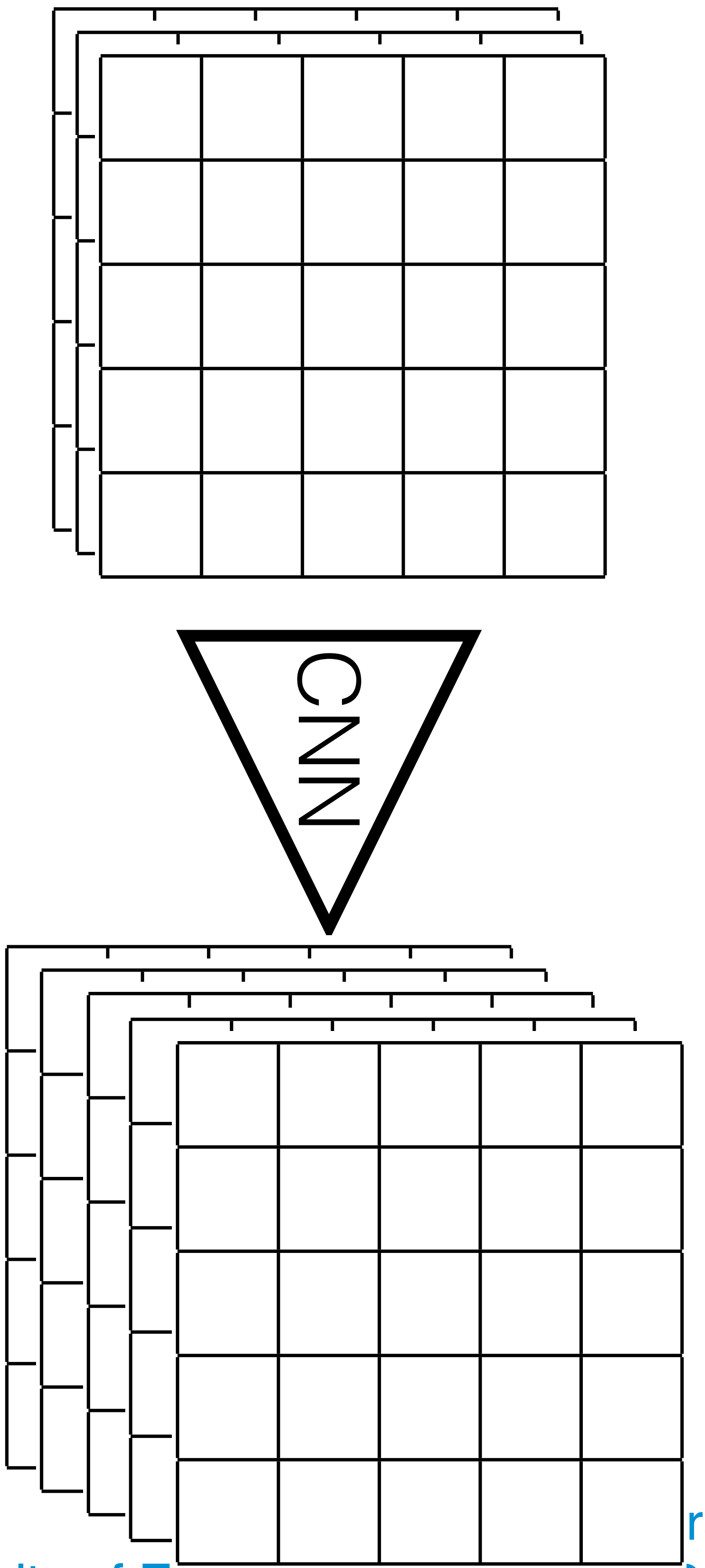
- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Architectures of regression networks
- Architectures of feature matching networks

Semantic segmentation



- 🟪 road
- 🟪 sideway
- 🟥 pedestrian
- 🟨 traffic sign
- 🟩 trees
- 🟦 sky

Semantic segmentation

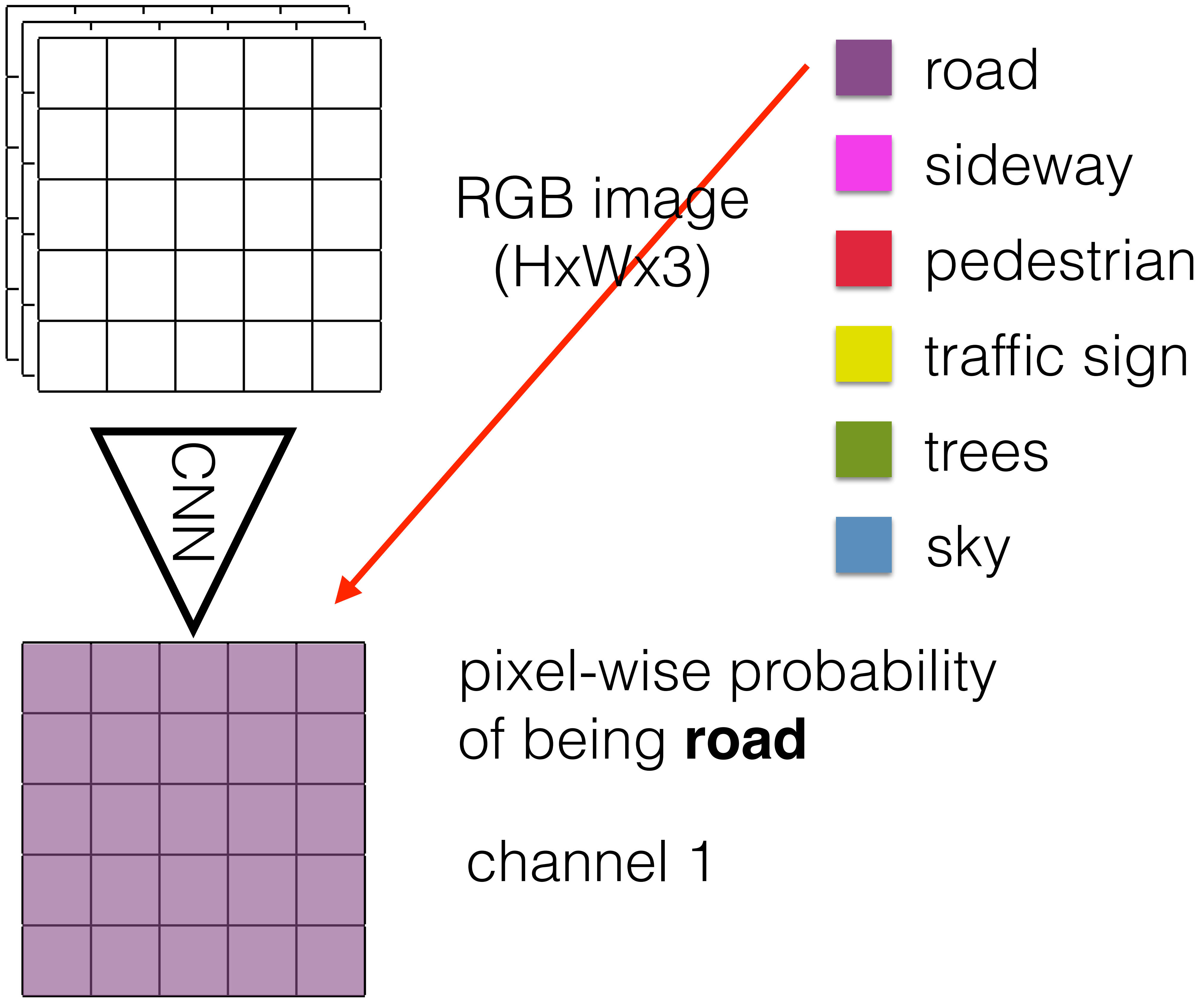


RGB image
(HxWx3)

labels
(HxWxN)

- 🟪 road
- 🟪 sideway
- 🟥 pedestrian
- 🟨 traffic sign
- 🟩 trees
- 🟦 sky

Semantic segmentation



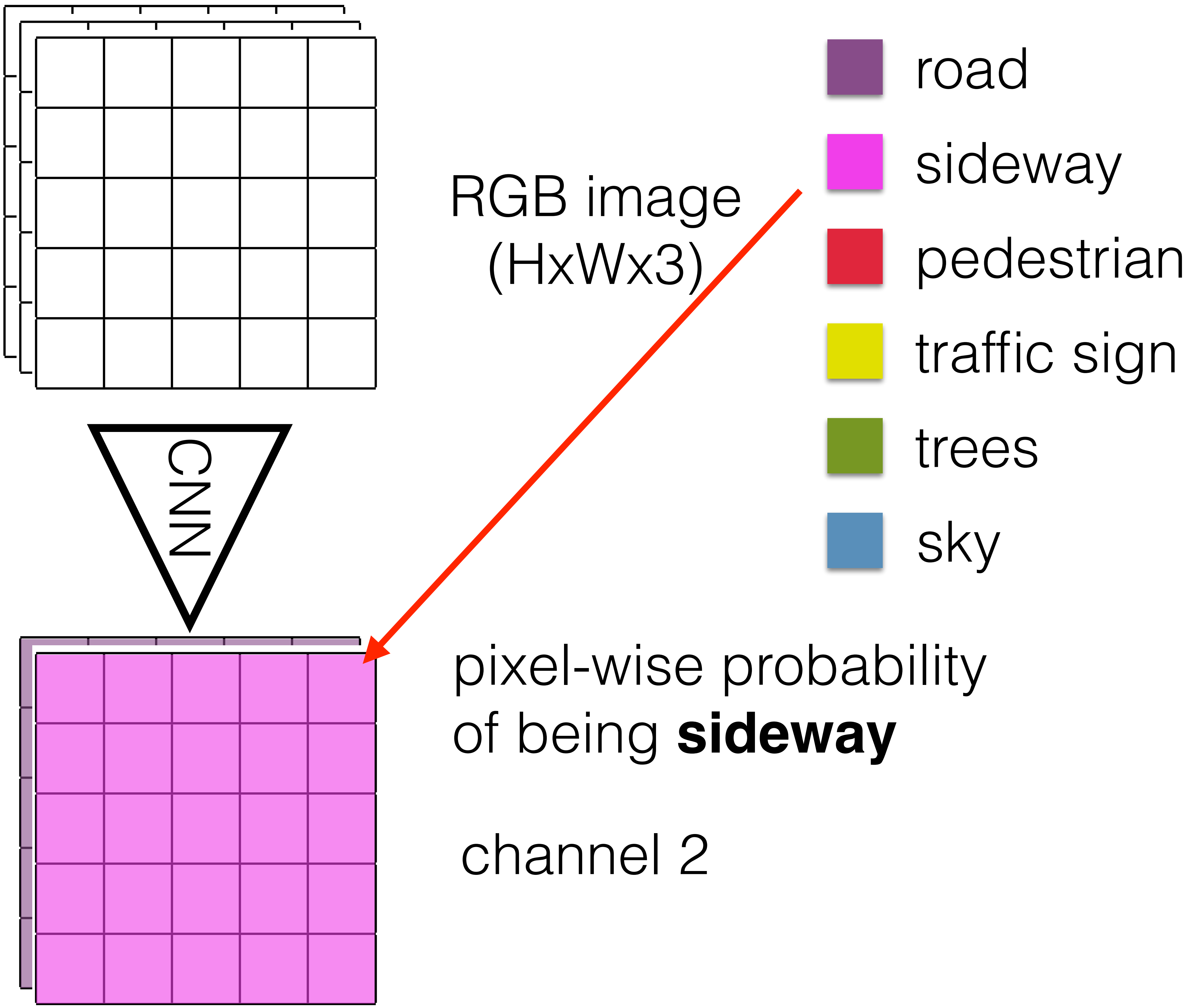
RGB image
(HxWx3)

🟪 road
🟪 sideway
🟥 pedestrian
🟨 traffic sign
🟩 trees
🟦 sky

pixel-wise probability
of being **road**

channel 1

Semantic segmentation



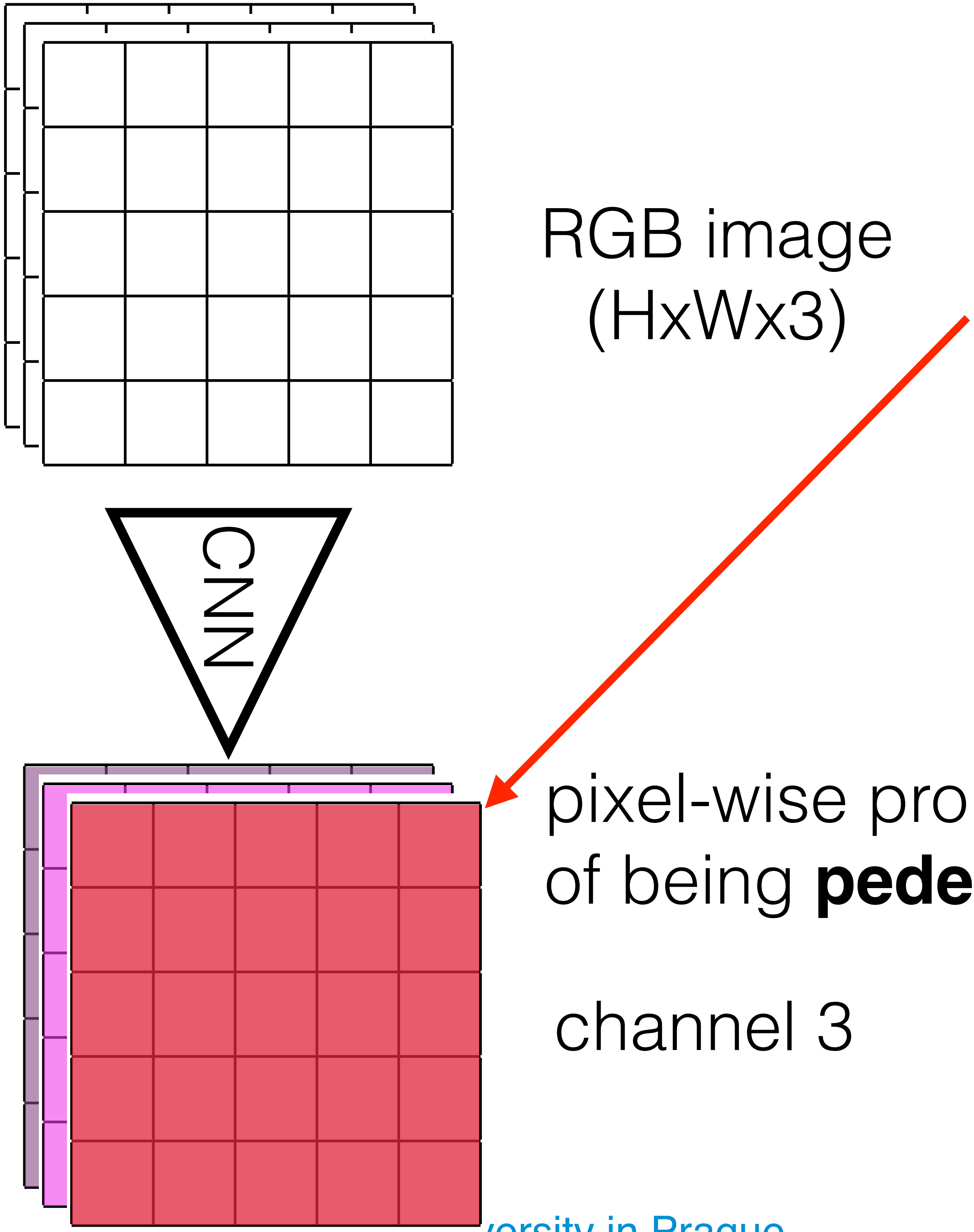
RGB image (HxWx3)

- 🟪 road
- 🟪 sideway
- 🟥 pedestrian
- 🟨 traffic sign
- 🟩 trees
- 🟦 sky

pixel-wise probability of being **sideway**

channel 2

Semantic segmentation



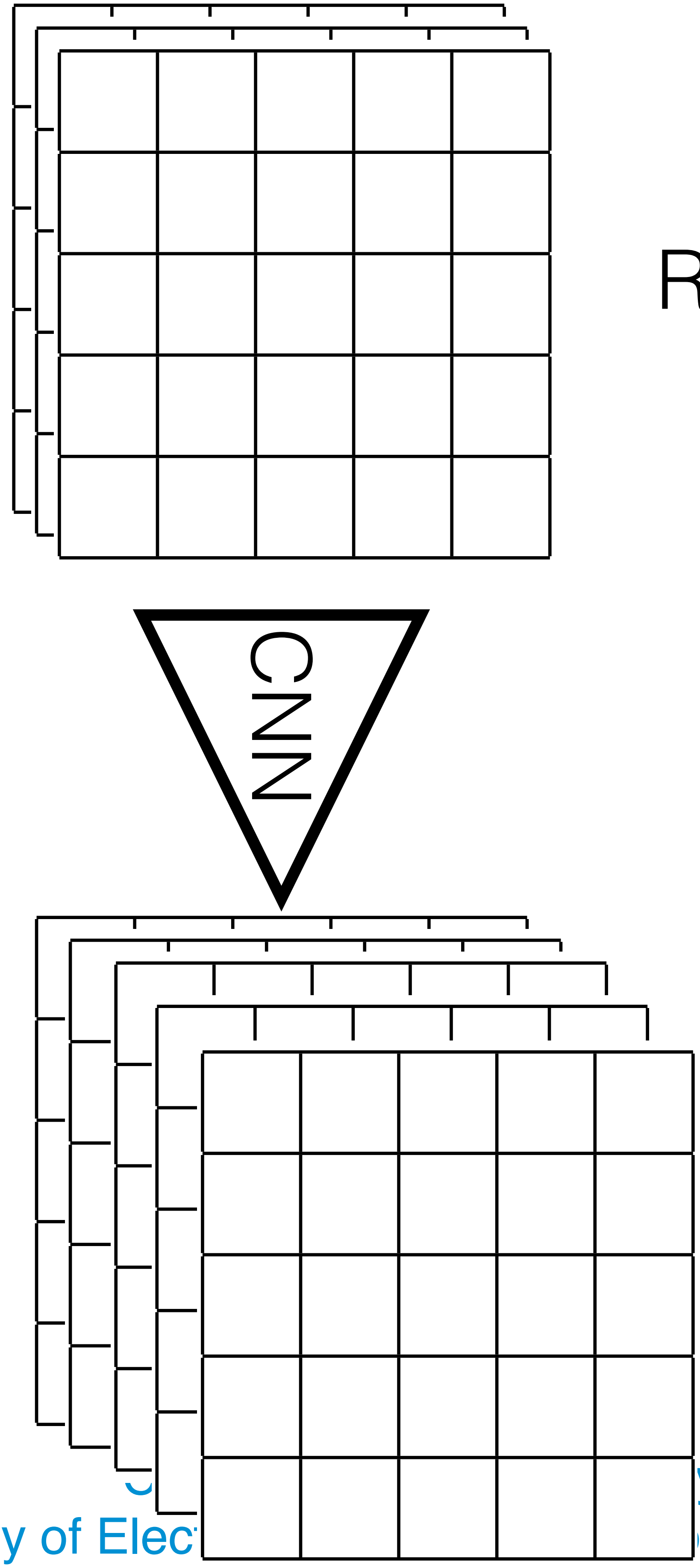
RGB image
(HxWx3)

- 🟪 road
- 🟪 sideway
- 🟥 pedestrian
- 🟨 traffic sign
- 🟩 trees
- 🟦 sky

pixel-wise probability
of being **pedestrian**

channel 3

Semantic segmentation

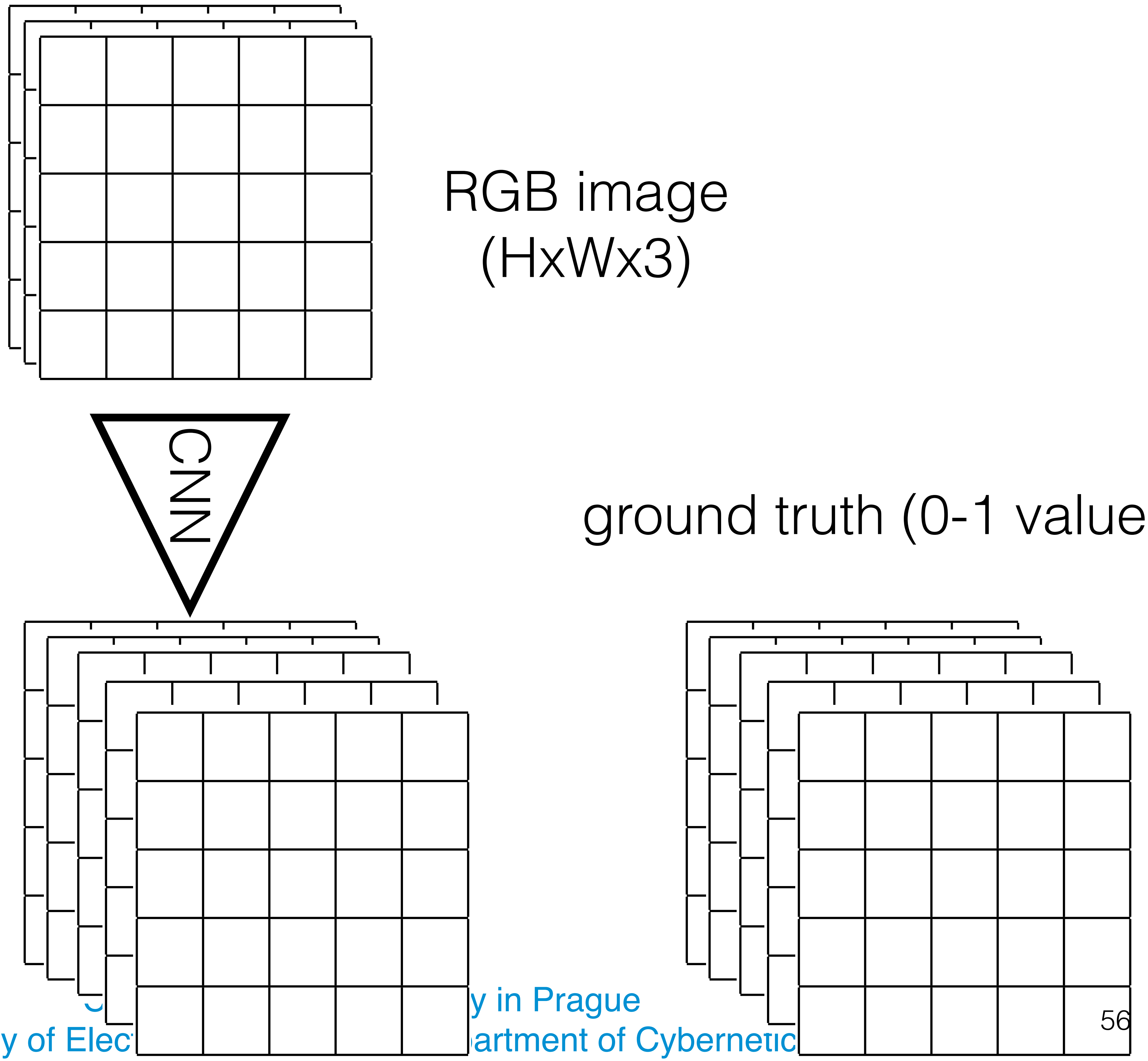


RGB image
(HxWx3)

- 🟪 road
- 🟪 sideway
- 🟥 pedestrian
- 🟨 traffic sign
- 🟩 trees
- 🟦 sky

as many output channels
as semantic labels

Semantic segmentation

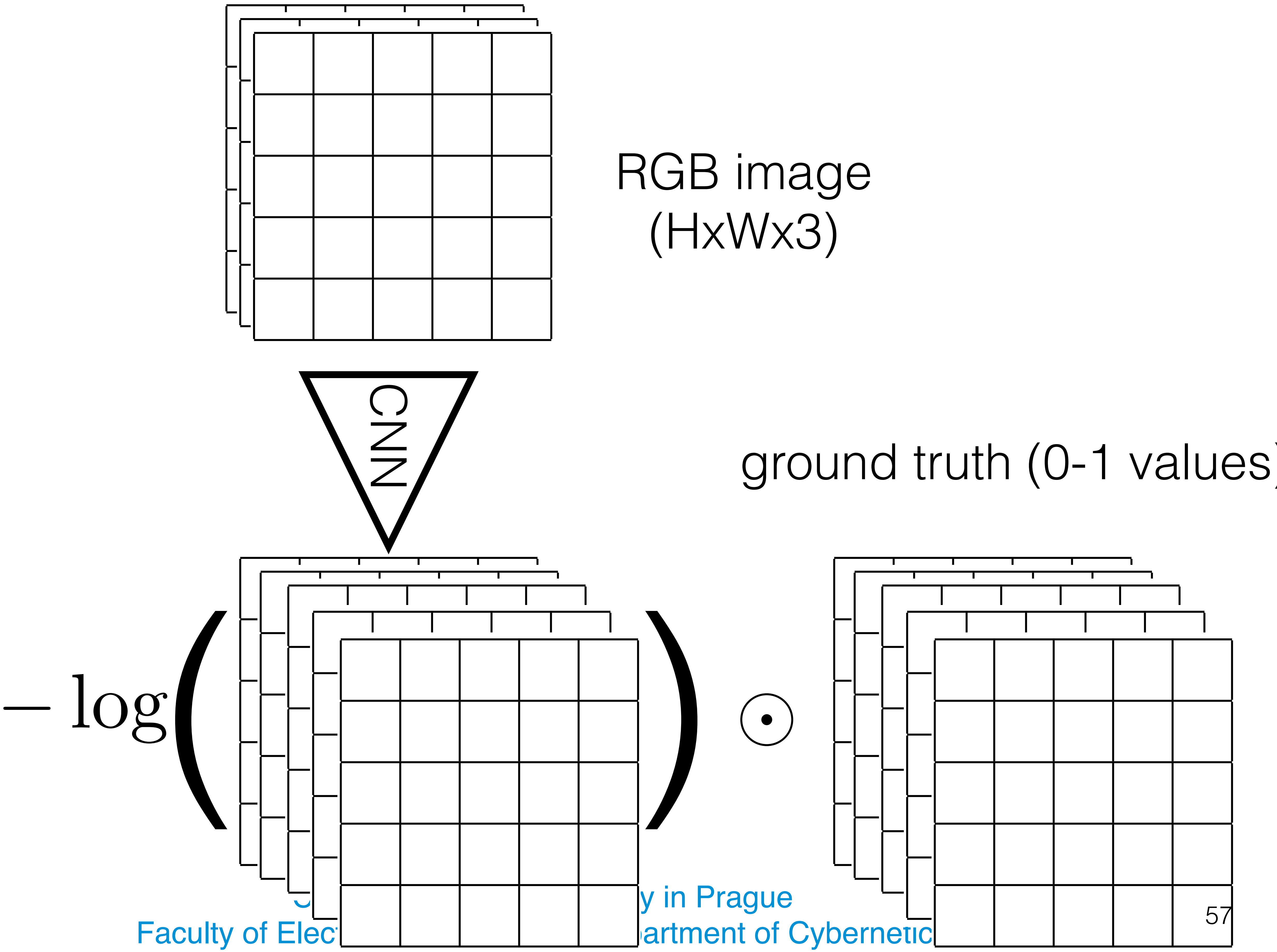


RGB image
(HxWx3)

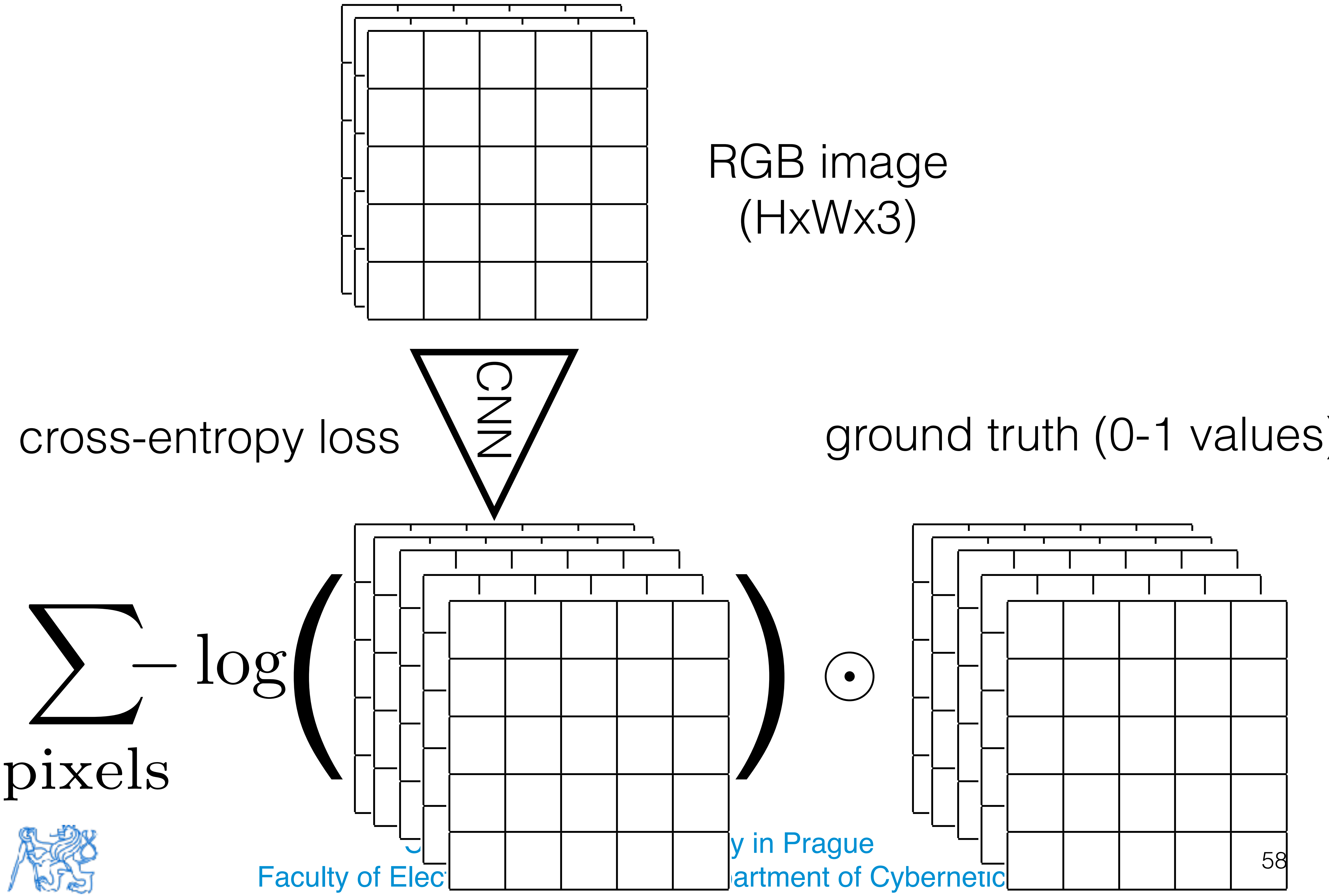
CNN

ground truth (0-1 values

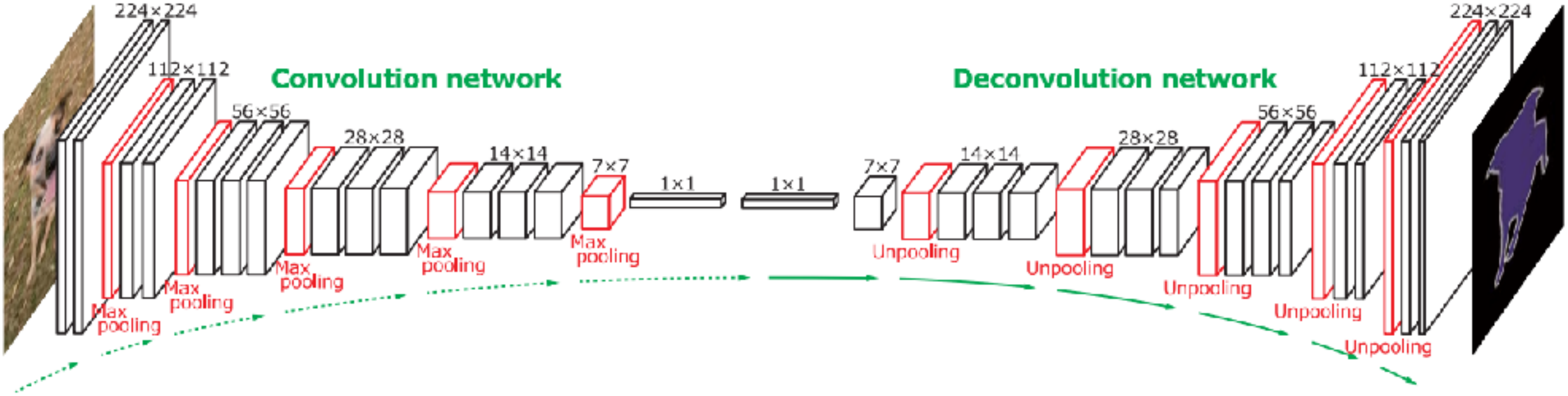
Semantic segmentation



Semantic segmentation



Semantic segmentation



- Loss: cross entropy loss summed over all pixels
- Convolution layers:
  - decrease spatial resolution
  - increase number of channels
- Deconvolution layers: exactly opposite

[Noh et al ICCV 2015] https://arxiv.org/pdf/1505.04366.pdf

Deconvolution

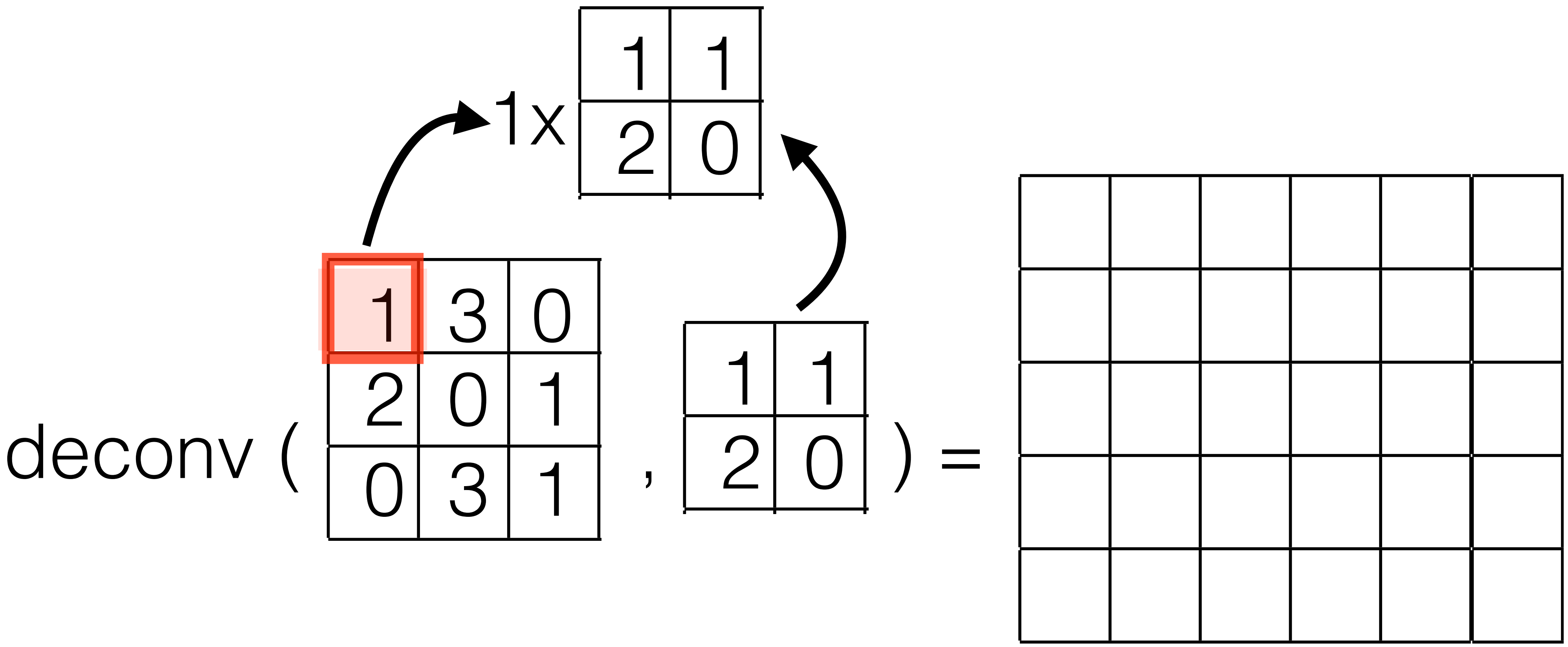
$$\text{deconv}\left(\begin{array}{|c|c|c|}\hline 1 & 3 & 0 \\ \hline 2 & 0 & 1 \\ \hline 0 & 3 & 1 \\ \hline\end{array}\,,\;\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 2 & 0 \\ \hline\end{array}\right) = \begin{array}{|c|c|c|c|c|c|}\hline \; & \; & \; & \; & \; & \; \\ \hline \; & \; & \; & \; & \; & \; \\ \hline \; & \; & \; & \; & \; & \; \\ \hline \; & \; & \; & \; & \; & \; \\ \hline \; & \; & \; & \; & \; & \; \\ \hline\end{array}$$

image (3x3)

kernel (2x2)

output (**6x6**)

Deconvolution



1	1
2	0

deconv (

1	3	0
2	0	1
0	3	1

,

1	1
2	0

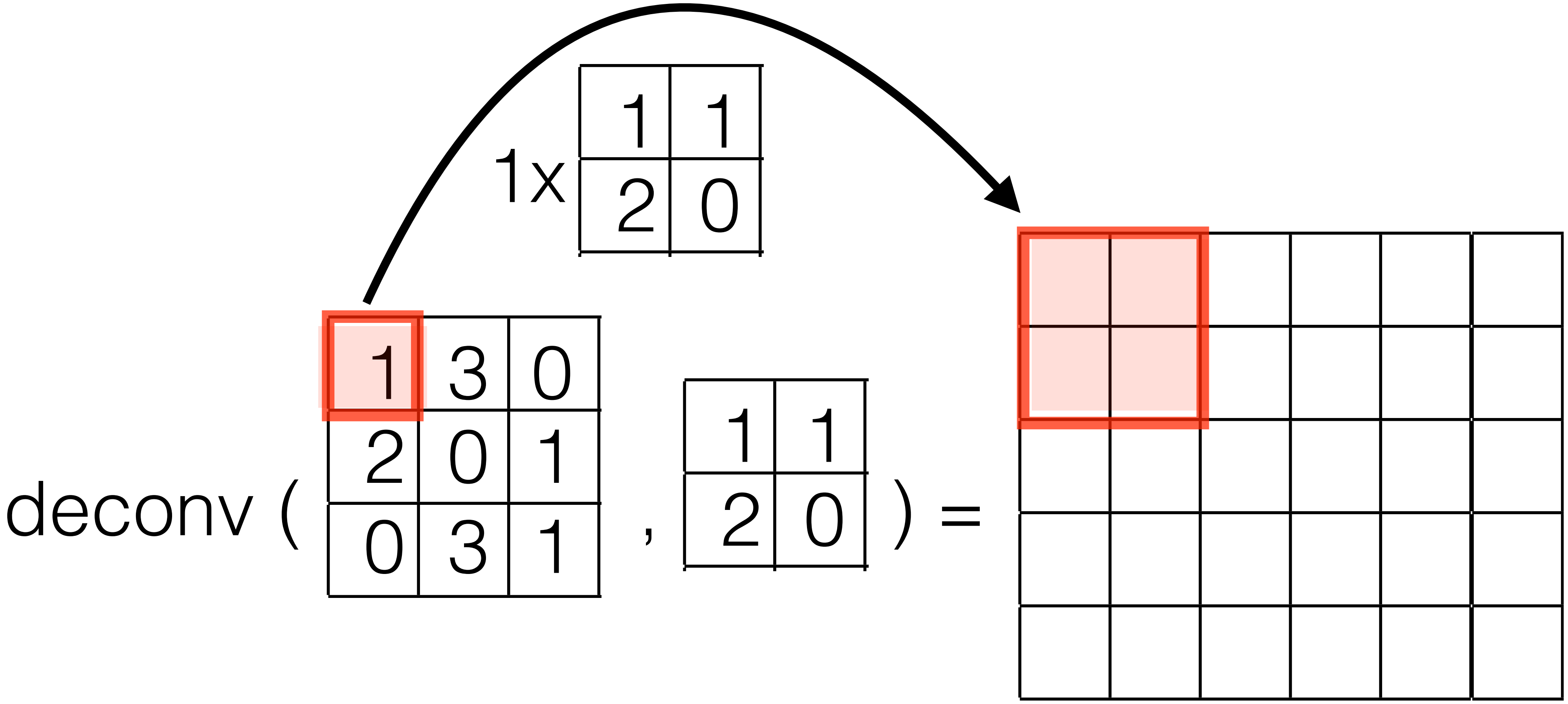
) =

image
(3x3)

kernel
(2x2)

output
(**6x6**)

Deconvolution



$$1 \times \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 2 & 0 \\ \hline\end{array}$$

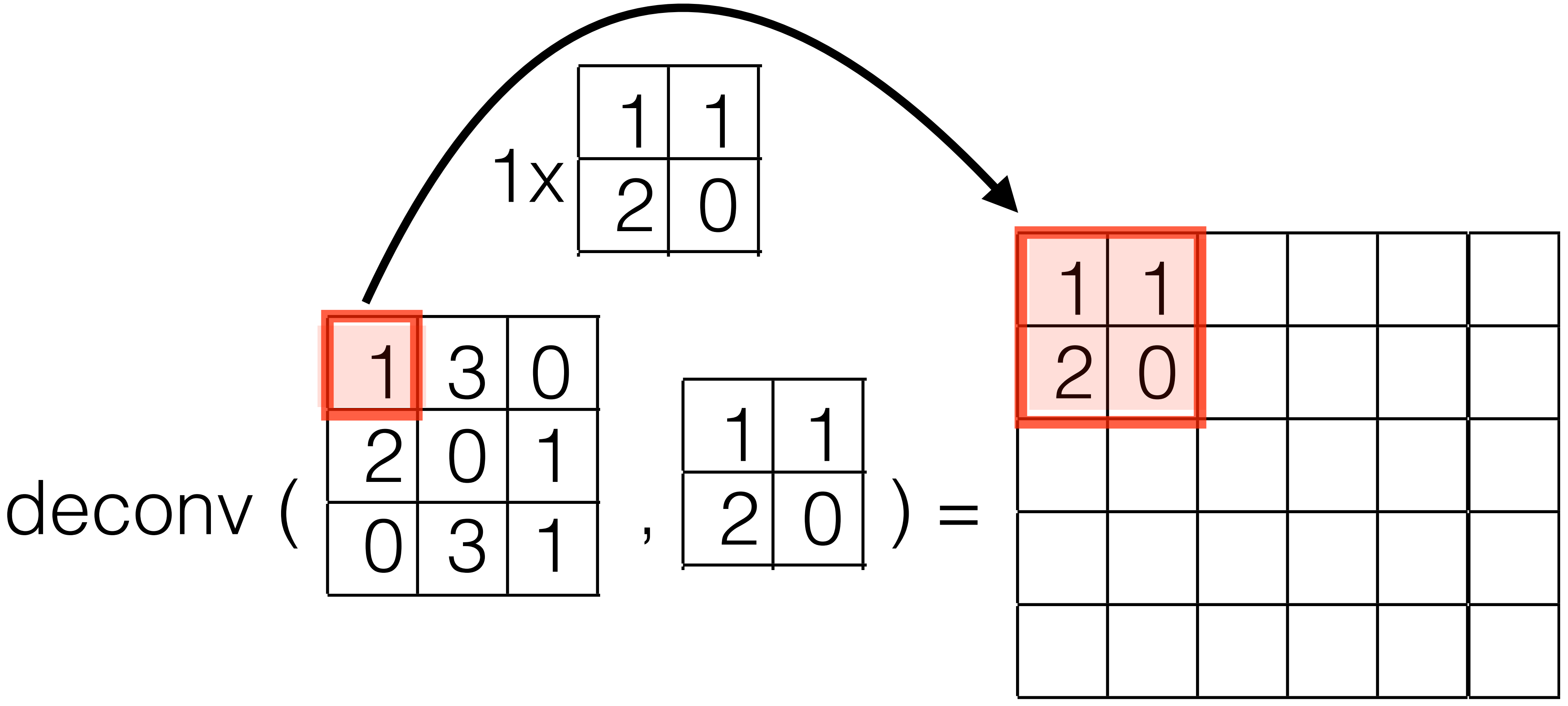
$$\mathrm{deconv}\left(\begin{array}{|c|c|c|}\hline 1 & 3 & 0 \\ \hline 2 & 0 & 1 \\ \hline 0 & 3 & 1 \\ \hline\end{array}\ ,\ \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 2 & 0 \\ \hline\end{array}\right) =$$

image
(3x3)

kernel
(2x2)

output
(**6x6**)

Deconvolution



1x

1	1
2	0

deconv (

1	3	0
2	0	1
0	3	1

,

1	1
2	0

) =

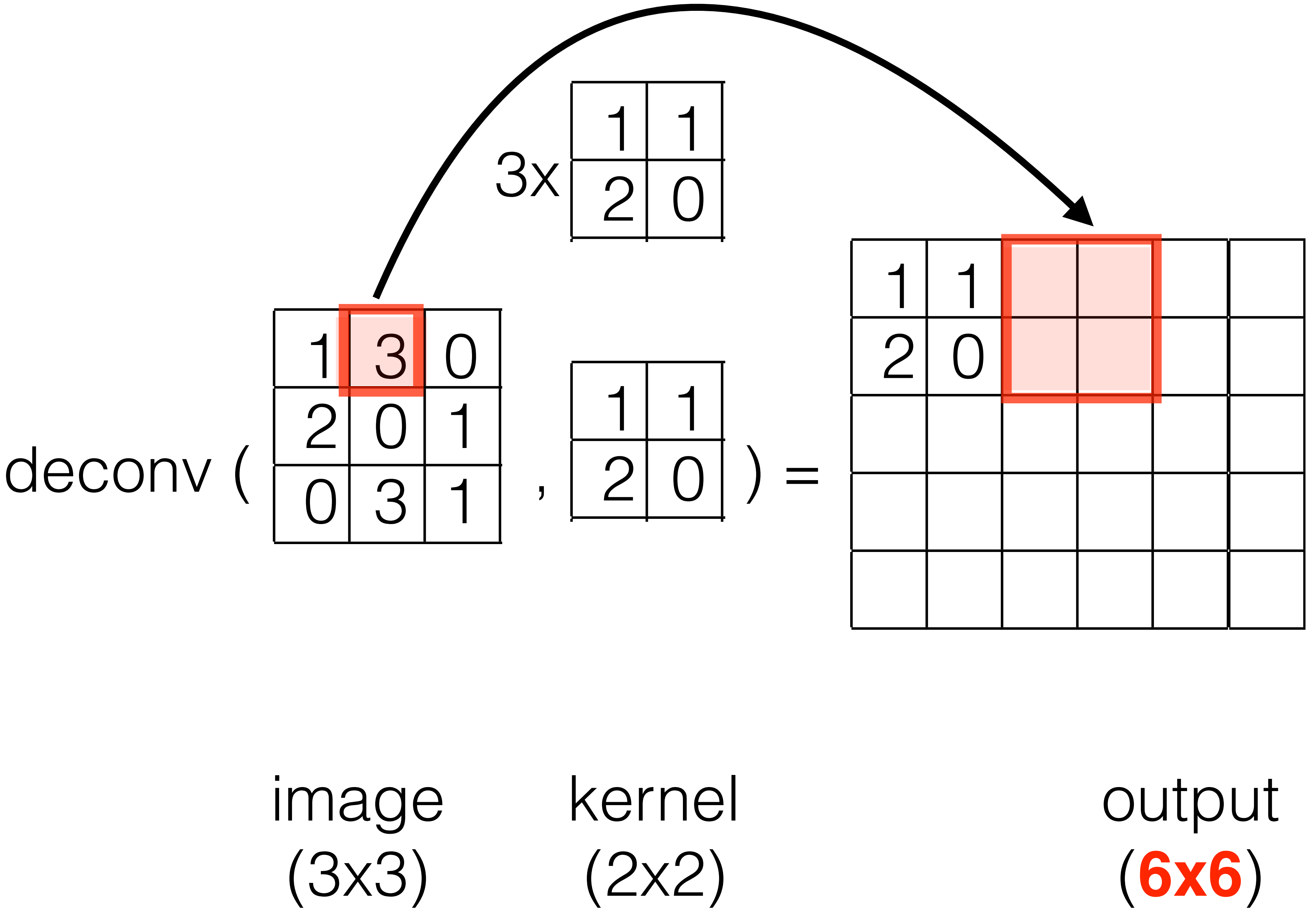
1	1				
2	0				

image (3x3)

kernel (2x2)

output (**6x6**)

Deconvolution



$$3\times\begin{array}{|c|c|}\hline 1 & 1\\ \hline 2 & 0\\ \hline\end{array}$$

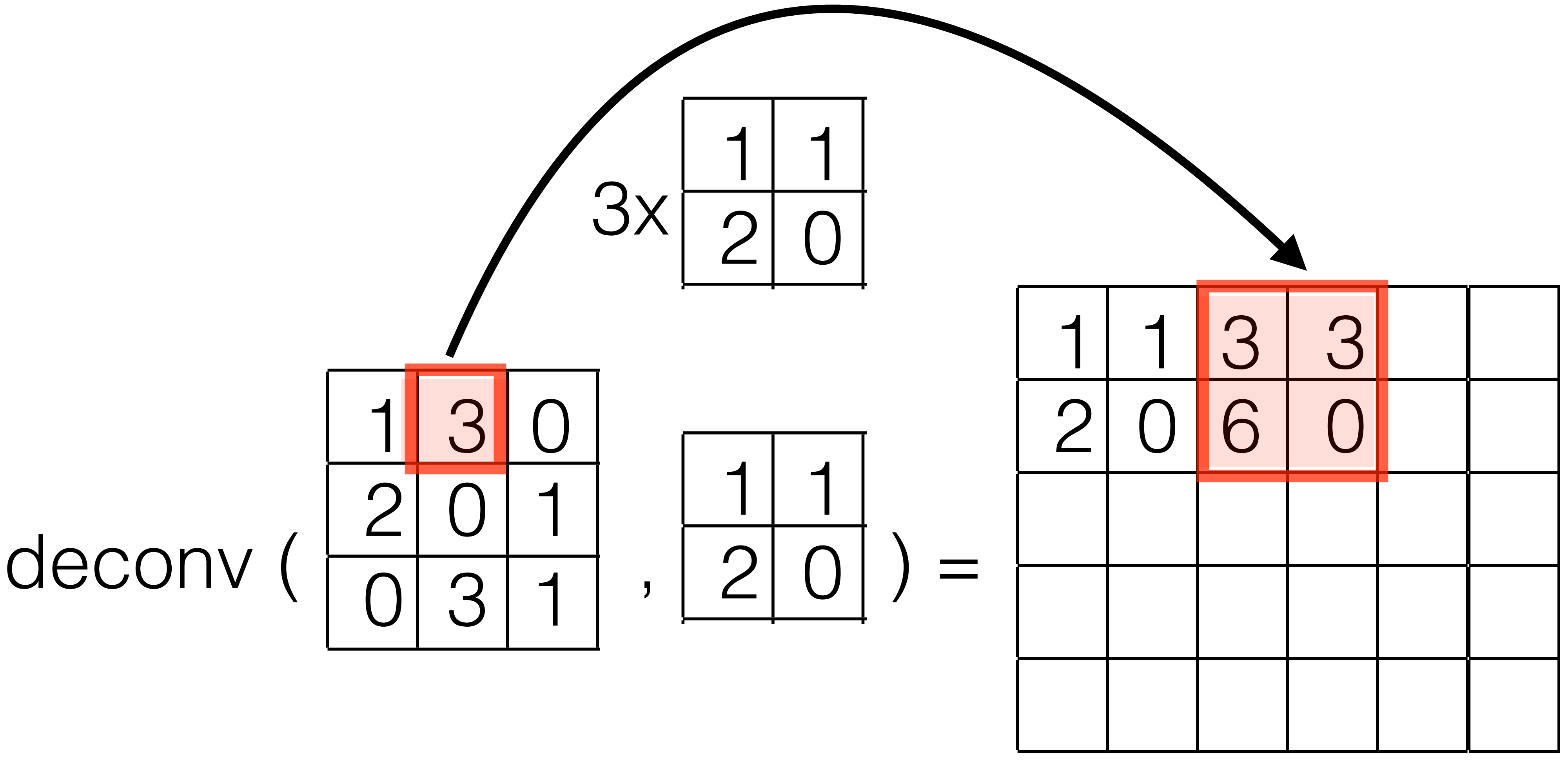
$$\text{deconv}\left(\begin{array}{|c|c|c|}\hline 1 & 3 & 0\\ \hline 2 & 0 & 1\\ \hline 0 & 3 & 1\\ \hline\end{array}\ ,\ \begin{array}{|c|c|}\hline 1 & 1\\ \hline 2 & 0\\ \hline\end{array}\right) = \begin{array}{|c|c|c|c|c|c|}\hline 1 & 1 & & & & \\ \hline 2 & 0 & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline\end{array}$$

image
(3x3)

kernel
(2x2)

output
(**6x6**)

Deconvolution



$$3x\begin{bmatrix}1&1\\2&0\end{bmatrix}$$

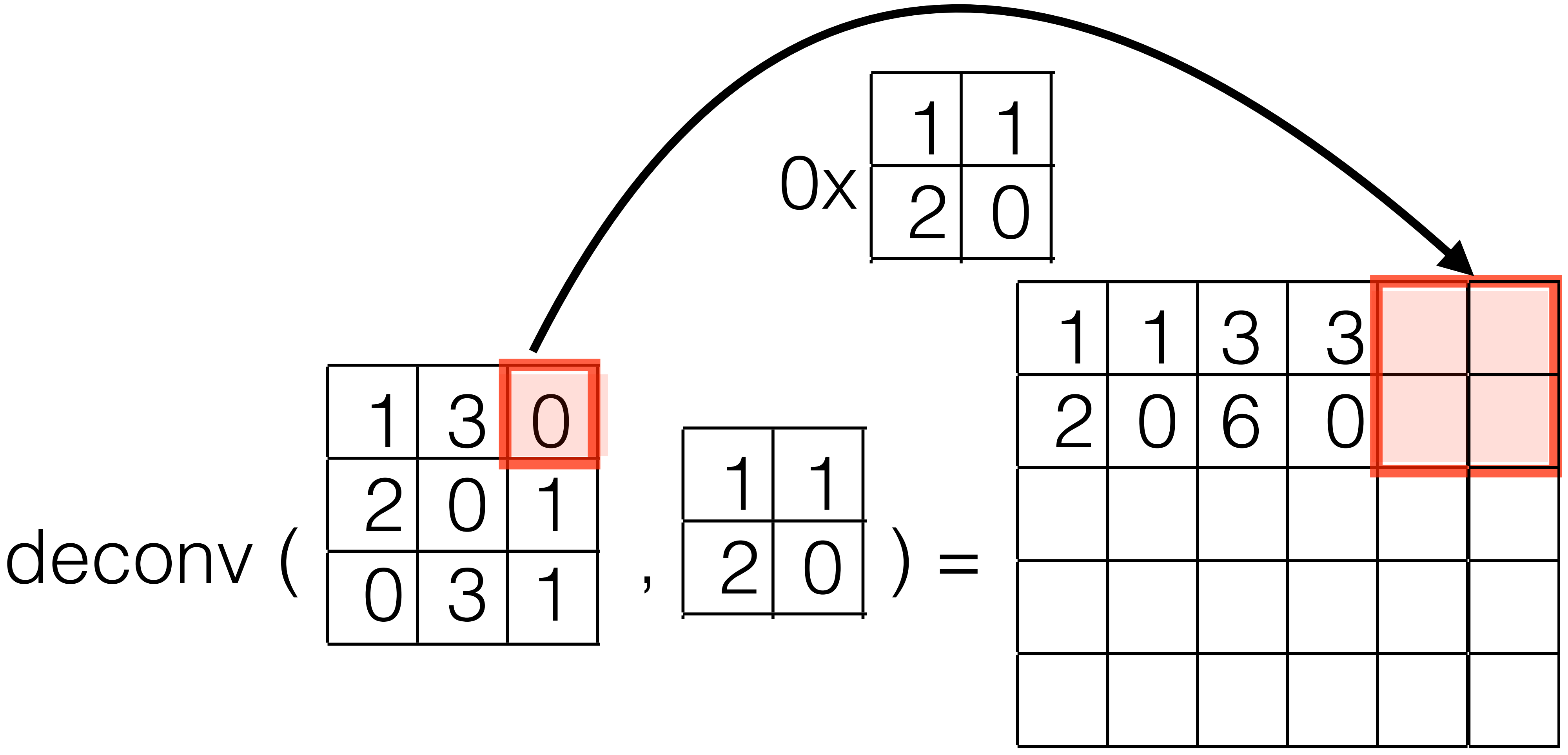
$$\text{deconv}\left(\begin{bmatrix}1&3&0\\2&0&1\\0&3&1\end{bmatrix},\begin{bmatrix}1&1\\2&0\end{bmatrix}\right)=\begin{bmatrix}1&1&3&3&&\\2&0&6&0&&\\&&&&&\\&&&&&\\&&&&&\end{bmatrix}$$

image
(3x3)

kernel
(2x2)

output
(**6x6**)

Deconvolution



$$0\times\begin{array}{|c|c|}\hline 1 & 1\\ \hline 2 & 0\\ \hline\end{array}$$

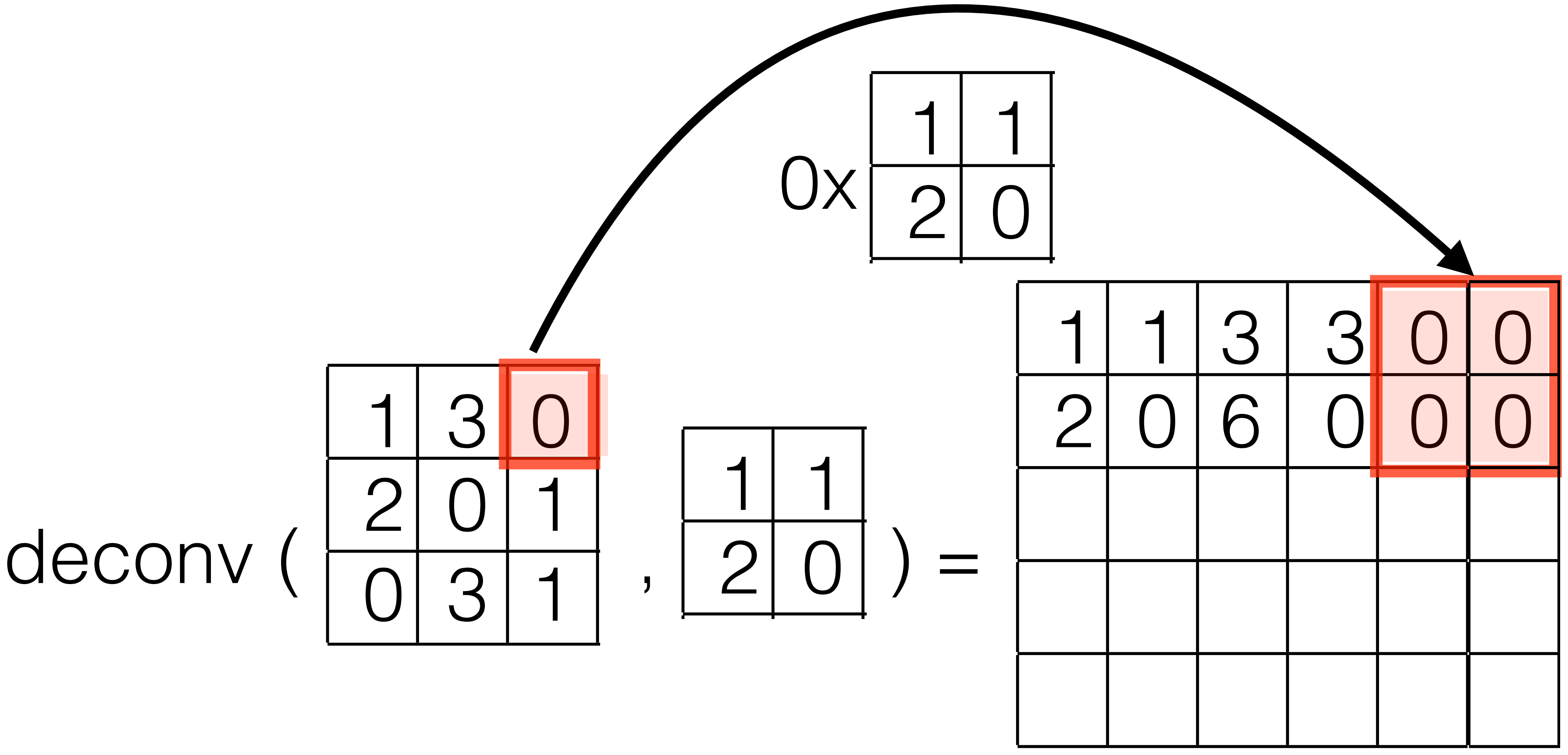
$$\text{deconv}\left(\begin{array}{|c|c|c|}\hline 1 & 3 & 0\\ \hline 2 & 0 & 1\\ \hline 0 & 3 & 1\\ \hline\end{array}\;,\;\begin{array}{|c|c|}\hline 1 & 1\\ \hline 2 & 0\\ \hline\end{array}\right)=\begin{array}{|c|c|c|c|c|c|}\hline 1 & 1 & 3 & 3 & & \\ \hline 2 & 0 & 6 & 0 & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline\end{array}$$

image
(3x3)

kernel
(2x2)

output
(**6x6**)

Deconvolution



0x

1	1
2	0

deconv (

1	3	0
2	0	1
0	3	1

,

1	1
2	0

) =

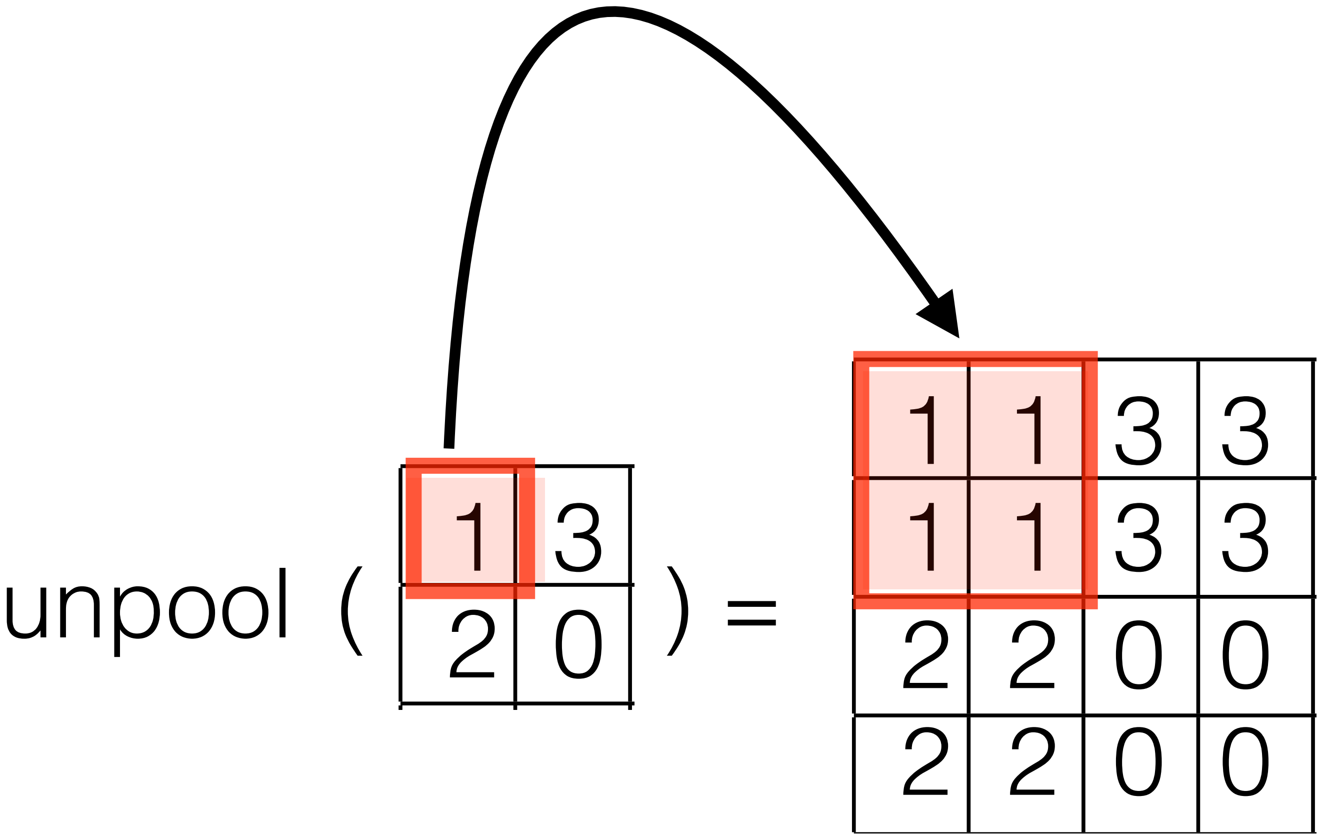
1	1	3	3	0	0
2	0	6	0	0	0

image
(3x3)

kernel
(2x2)

output
(**6x6**)

unpooling



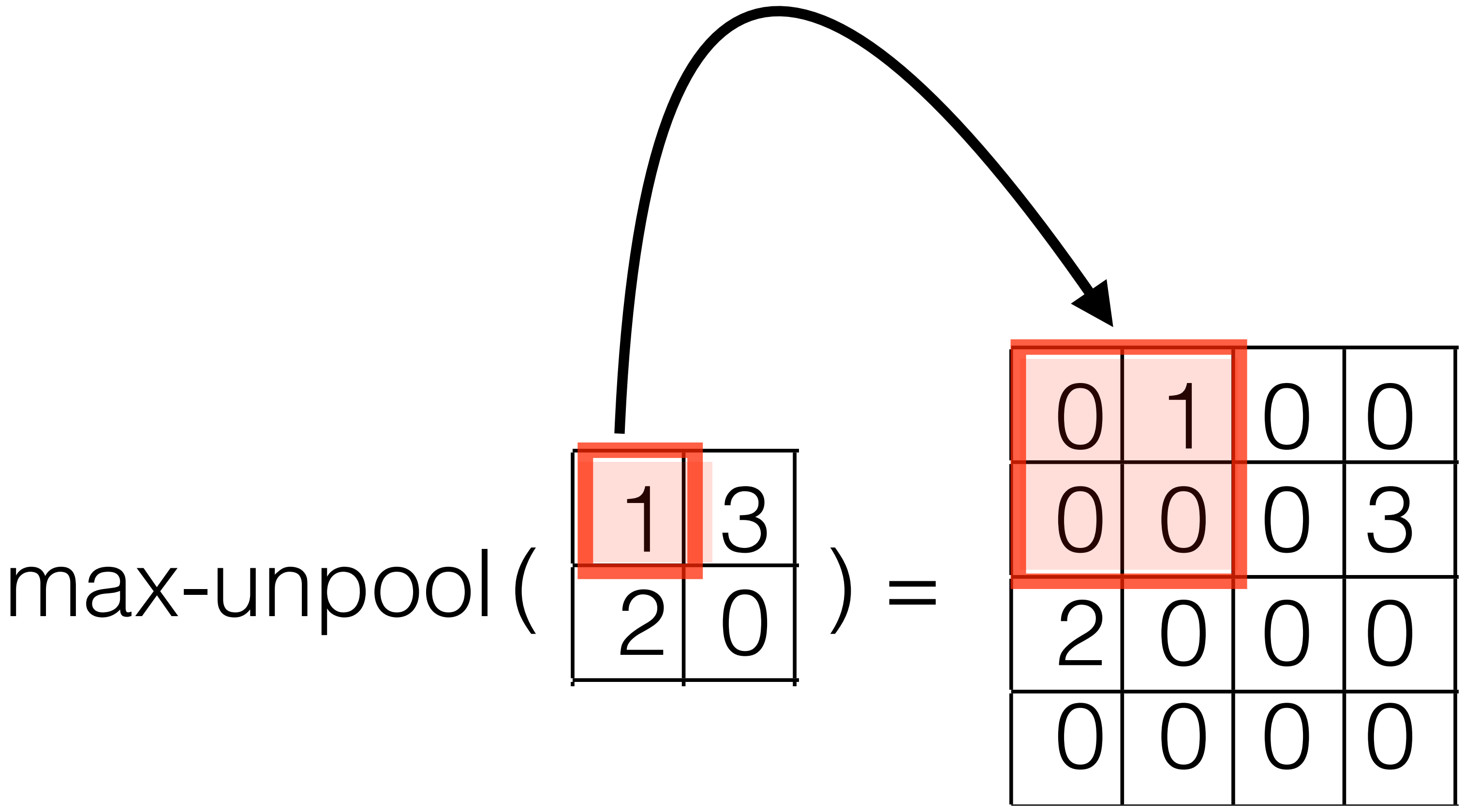
$$\text{unpool}\left(\begin{array}{|c|c|}\hline 1 & 3 \\ \hline 2 & 0 \\ \hline\end{array}\right) = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 3 & 3 \\ \hline 1 & 1 & 3 & 3 \\ \hline 2 & 2 & 0 & 0 \\ \hline 2 & 2 & 0 & 0 \\ \hline\end{array}$$

image
(2x2)

output
(**4x4**)

copy everywhere unpooling

max-unpooling



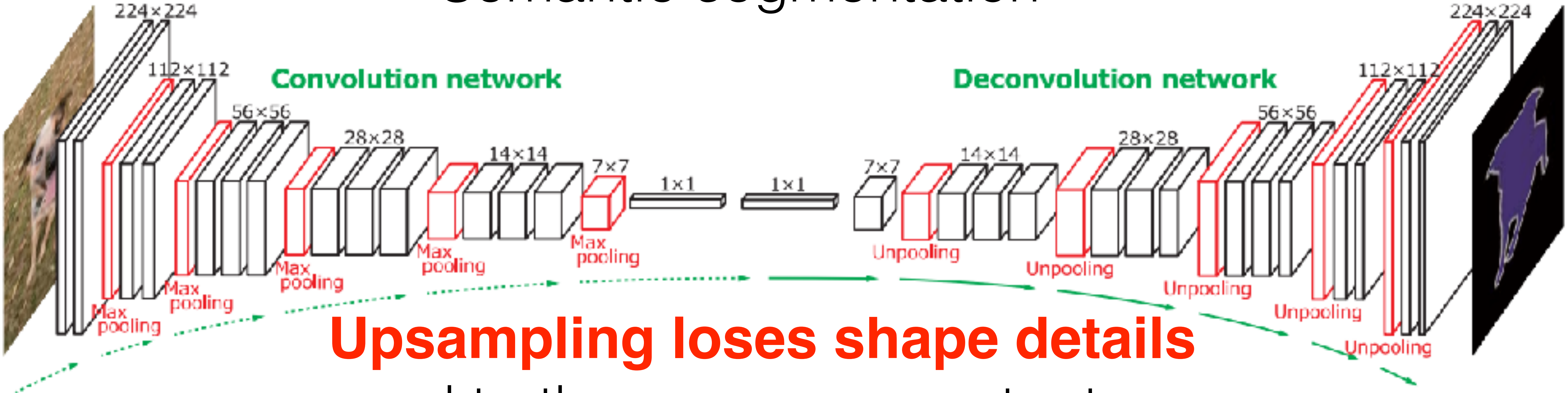
$$\text{max-unpool}\left(\begin{array}{|c|c|}\hline 1 & 3 \\ \hline 2 & 0 \\ \hline\end{array}\right) = \begin{array}{|c|c|c|c|}\hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 3 \\ \hline 2 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline\end{array}$$

image
(2x2)

output
(**4x4**)

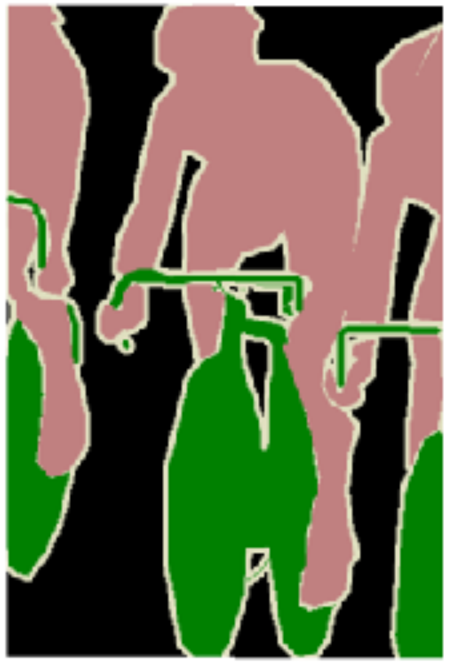
bad-of-nails unpooling
remember position of the maximum from max-pooling layer

Semantic segmentation

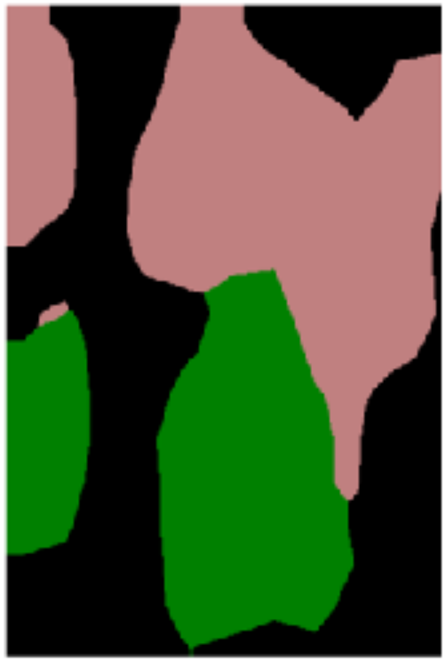


Upsampling loses shape details

ground truth

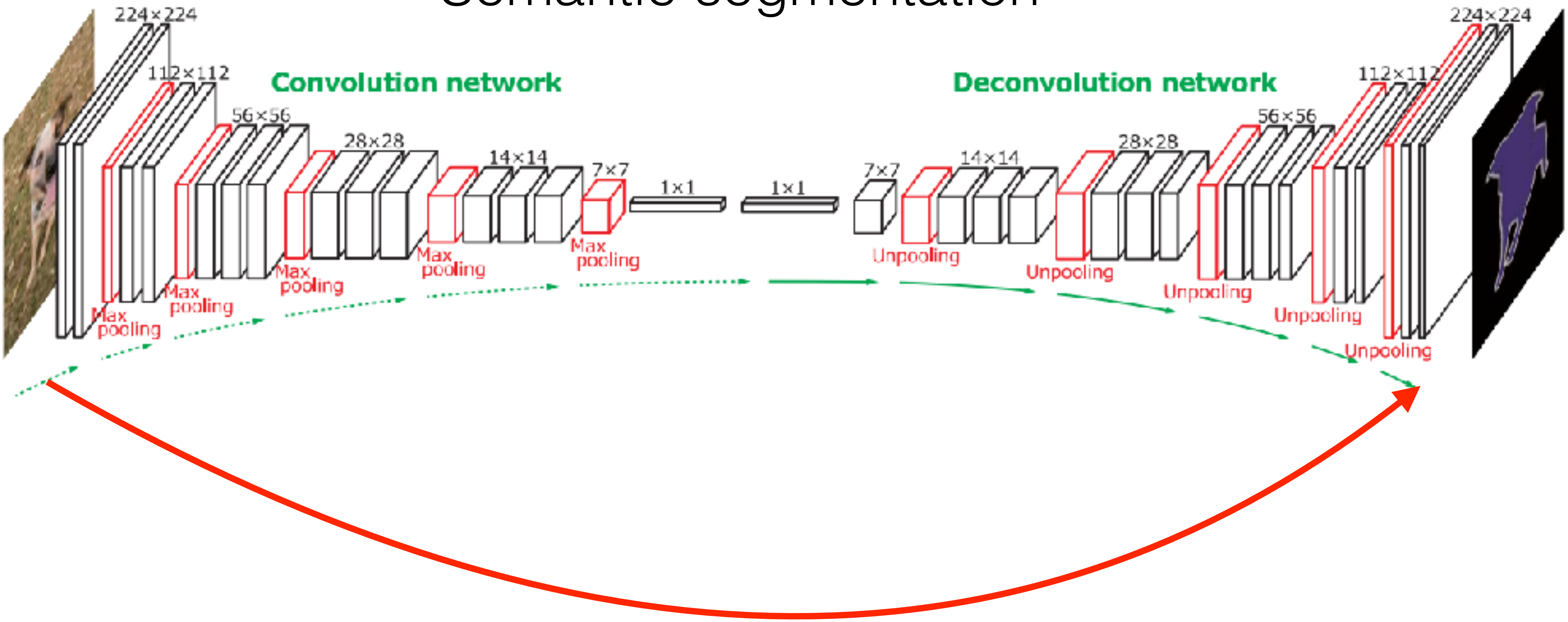


output



[Long et al CVPR 2015] https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

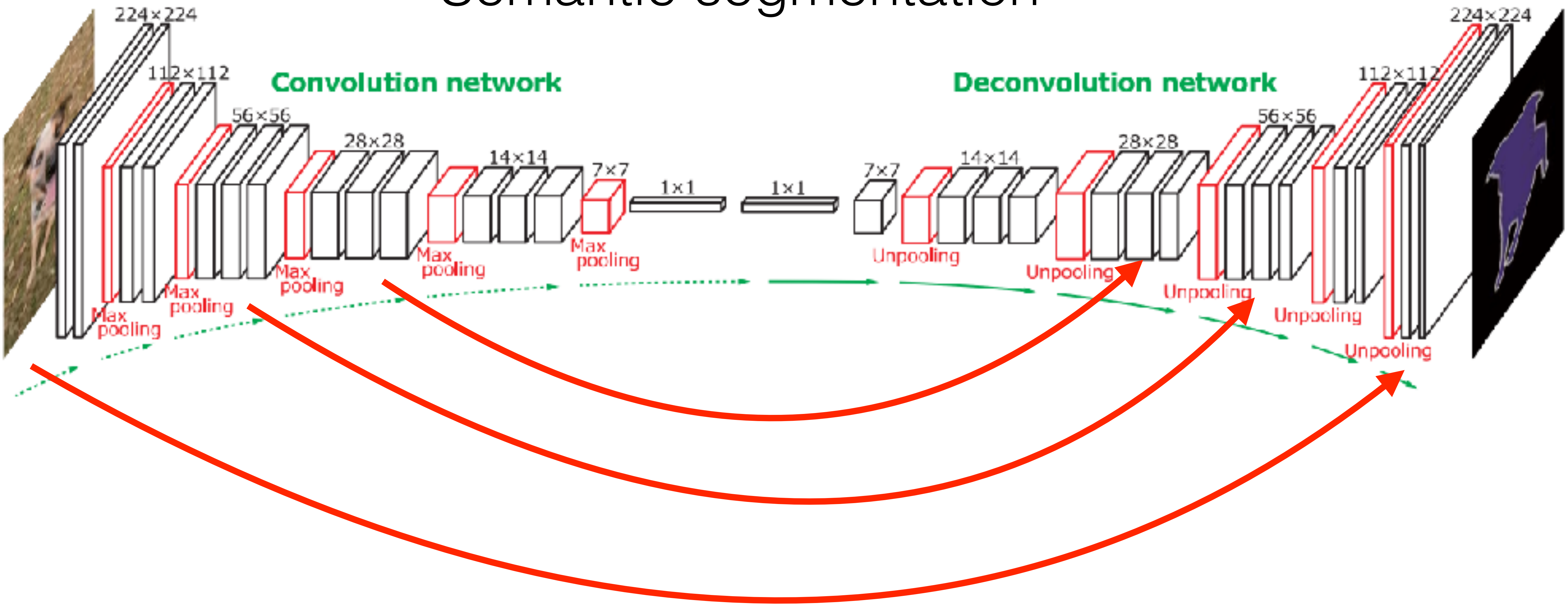
Semantic segmentation



concatenate deconvolution feature map with the original feature map

[Long et al CVPR 2015] https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

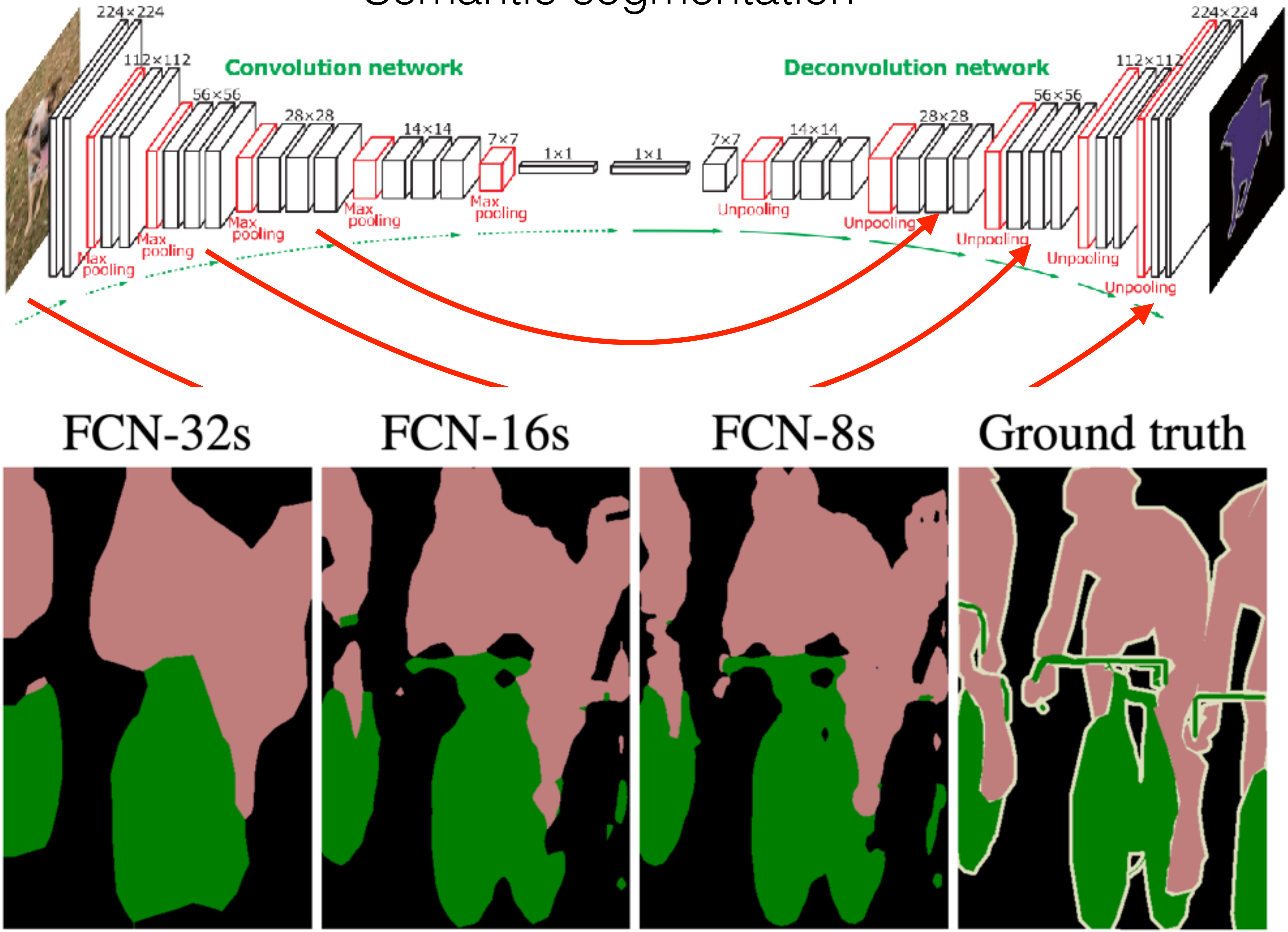
Semantic segmentation



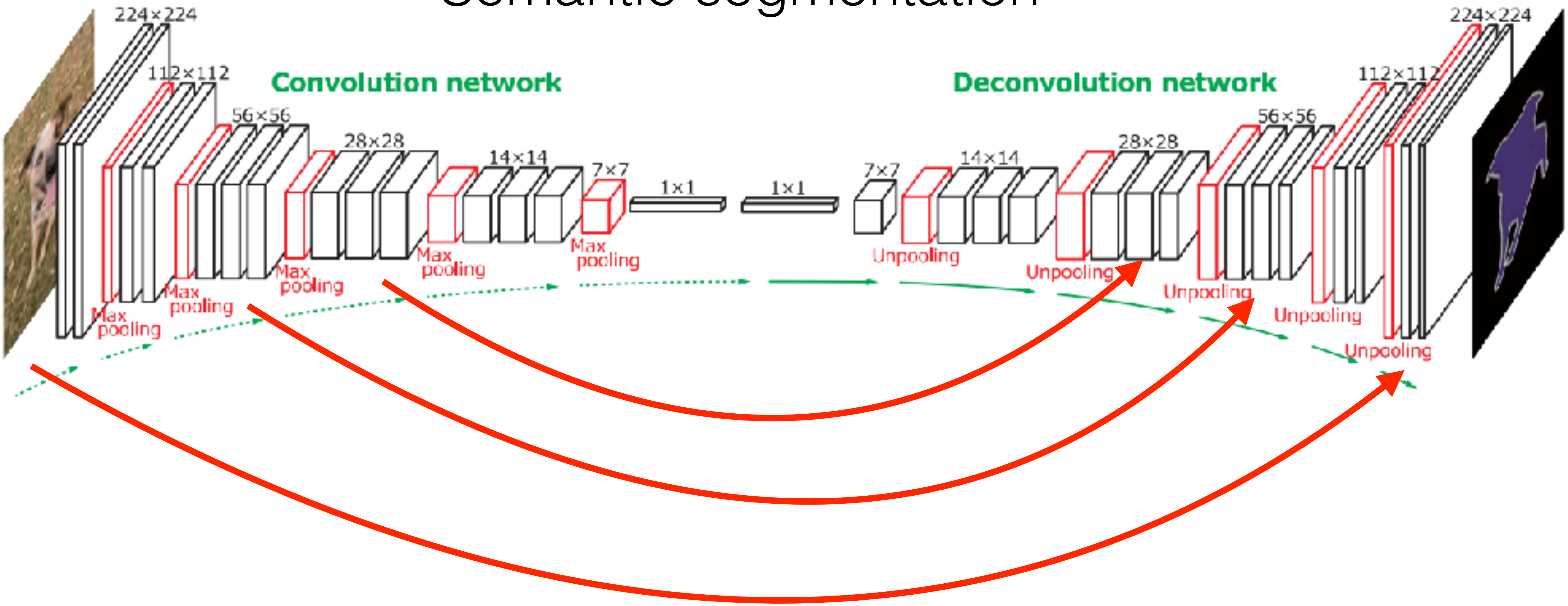
concatenate deconvolution feature map with the original feature map

[Long et al CVPR 2015] https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Semantic segmentation



Semantic segmentation



- Autonomous driving applications require segmentation of objects on very different scales.
- Instead of segmenting on hires images, downsampling, detecting on midres images, downsampling… upsampling
- People introduced atrous convolution

Convolution layer

Dilatation rate = 1

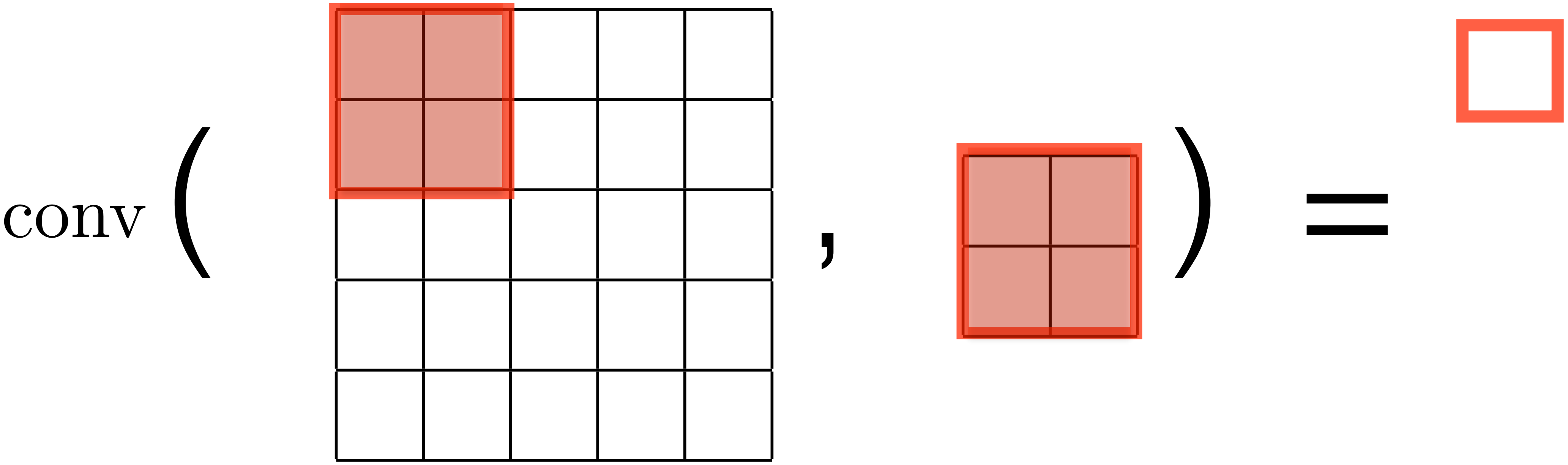


image
(5x5)

kernel
(2x2)

output
(**? x ?**)

Atrous convolution layer

Dilatation rate = 2

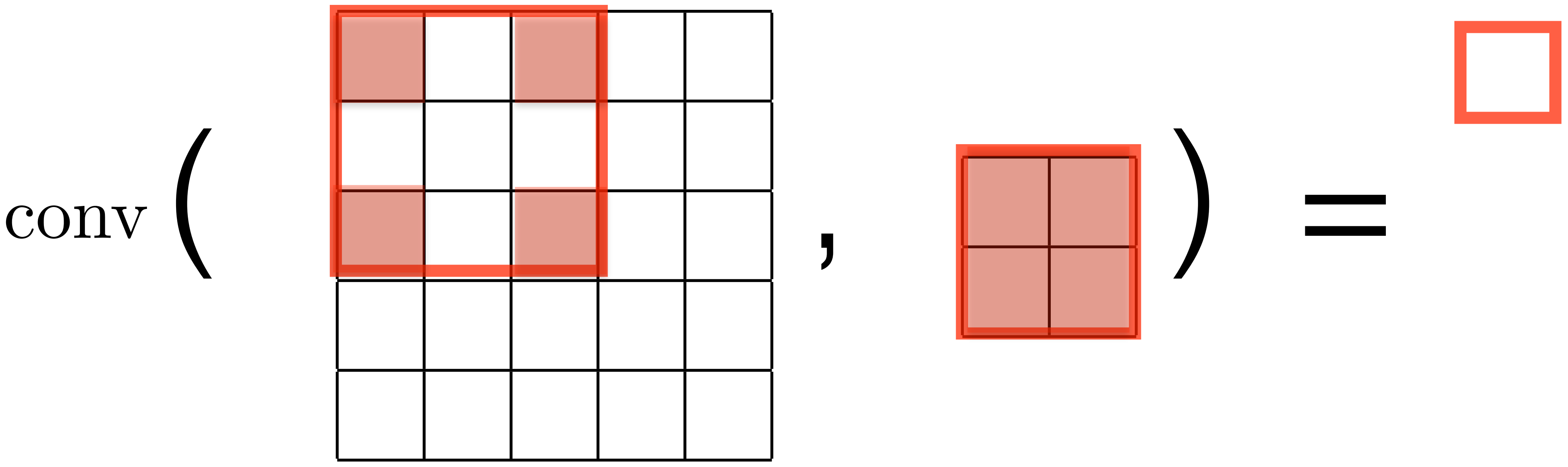
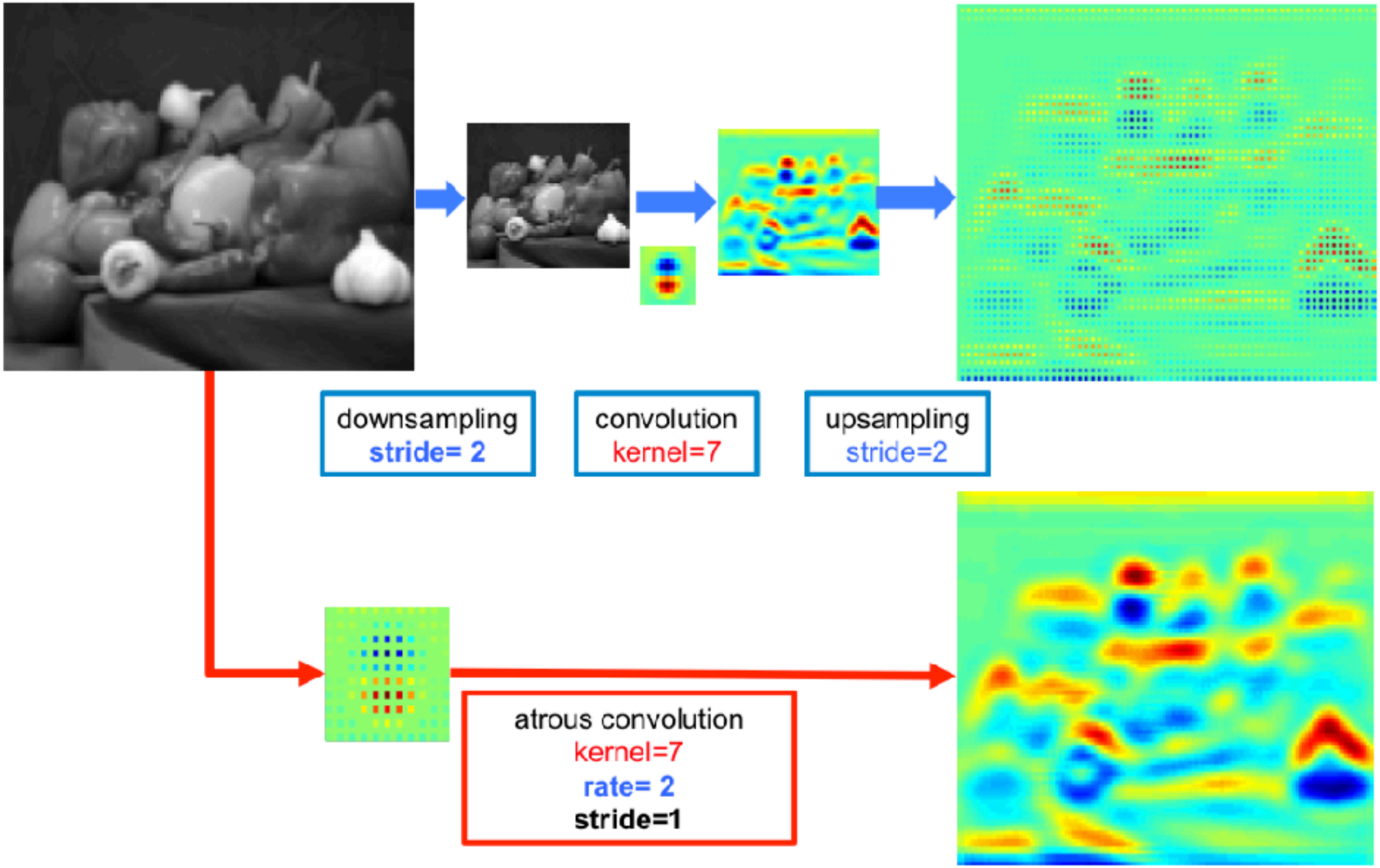


image
(5x5)

kernel
(2x2)

output
(**? x ?**)

Atrous vs standard convolution for segmentation



downsampling
stride= 2

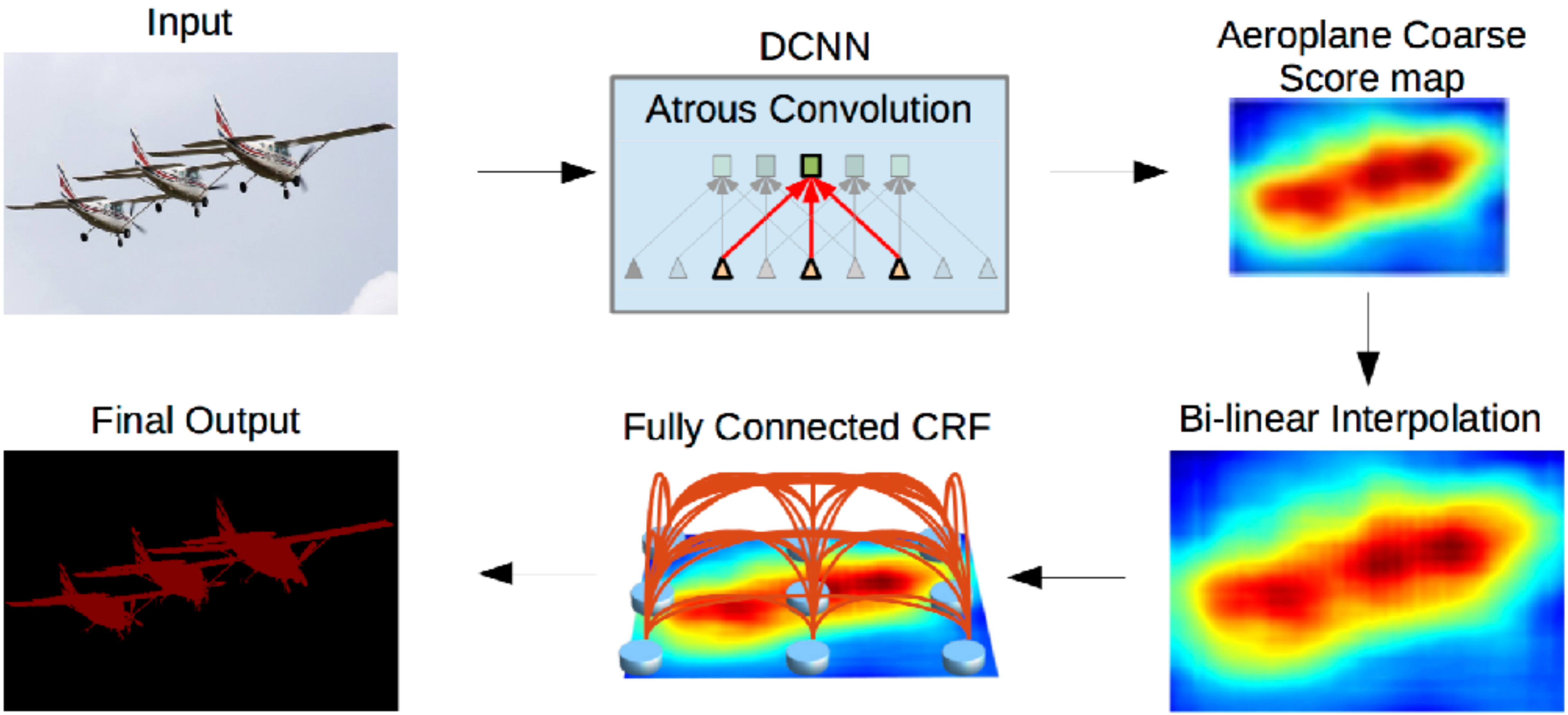
convolution
kernel=7

upsampling
stride=2

atrous convolution
kernel=7
rate= 2
stride=1

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

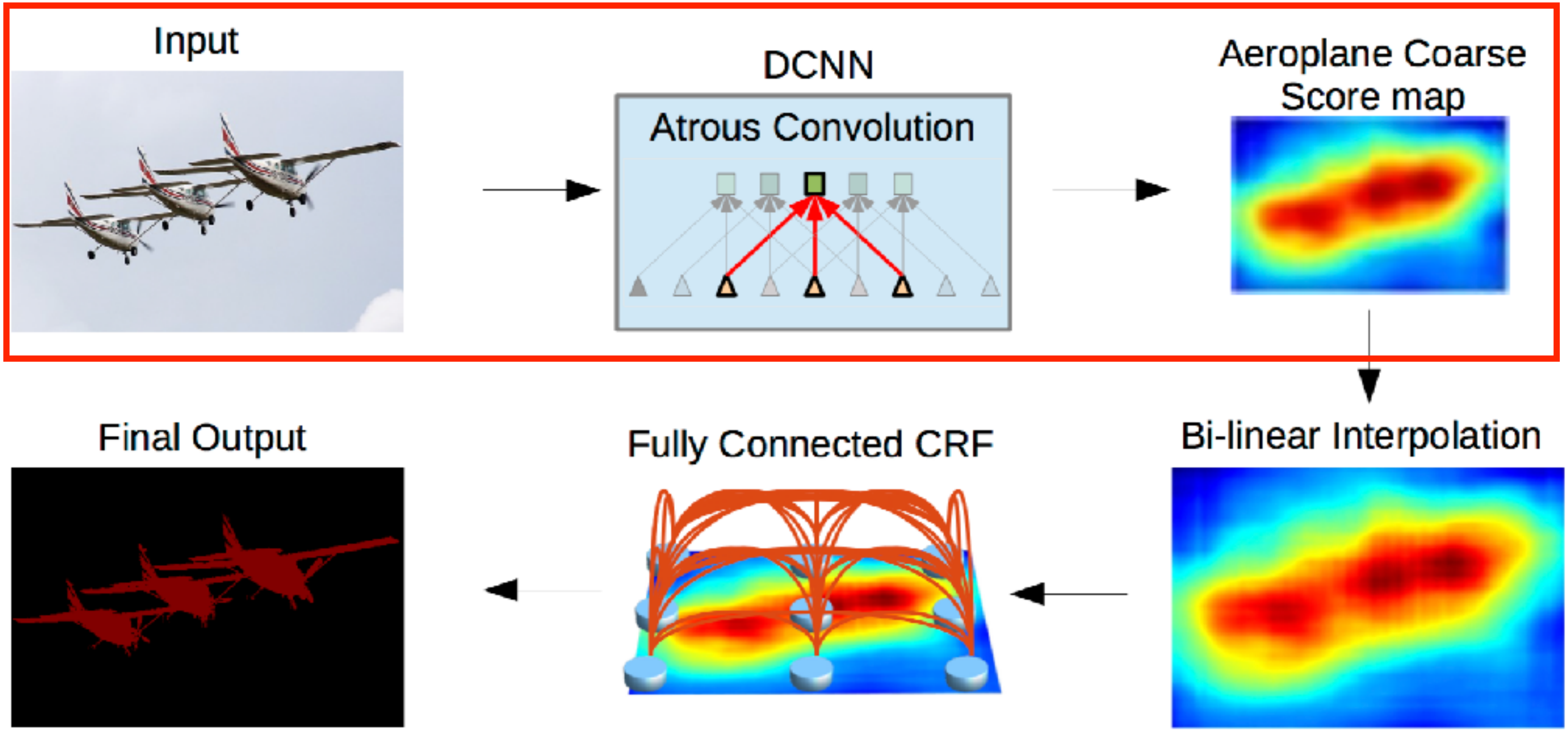
DeepLab v3



- Replace maxpooling by Atrous Convolution
- Replace deconvolutions by bi-linear interp+CRF

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

DeepLab v3



Input

DCNN

Atrous Convolution

Aeroplane Coarse Score map

Bi-linear Interpolation

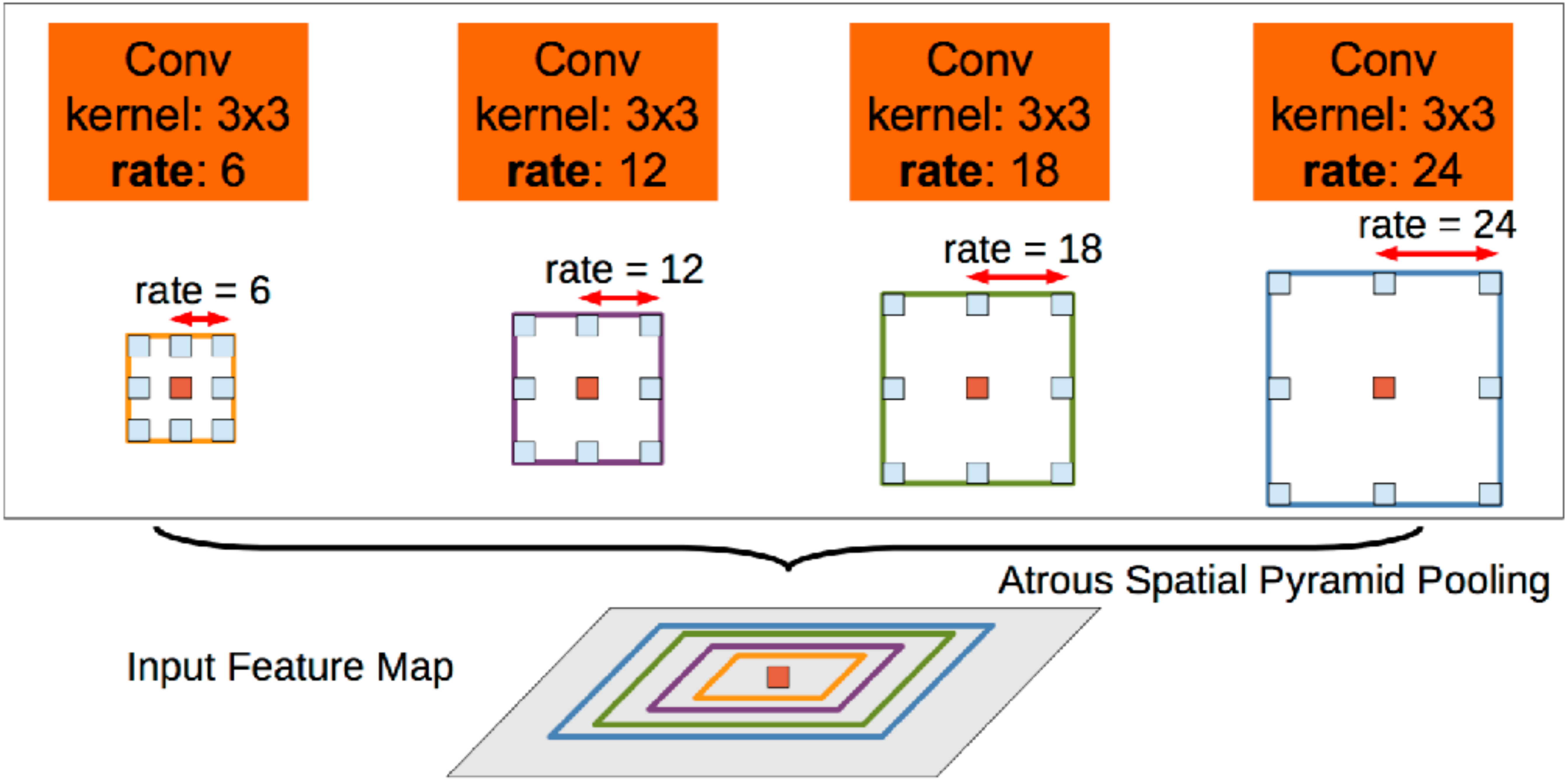
Fully Connected CRF

Final Output

- Replace maxpooling by Atrous Convolution
- Replace deconvolutions by bi-linear interp+CRF

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

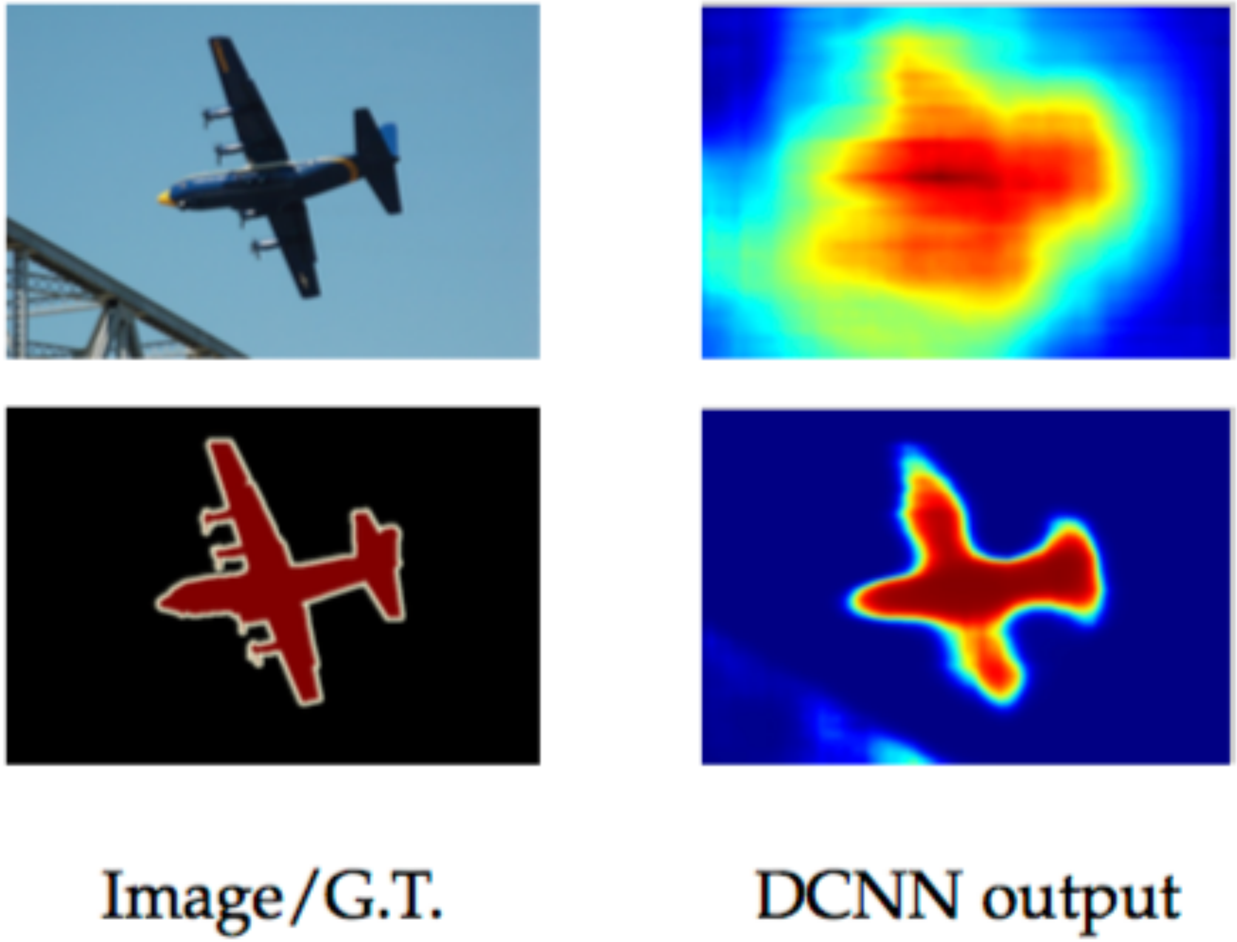
Atrous Spatial Pyramid Pooling (ASPP)



- Similar downsampling effect as maxpooling but it is learnable

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

DeepLab v3- result after DCNN



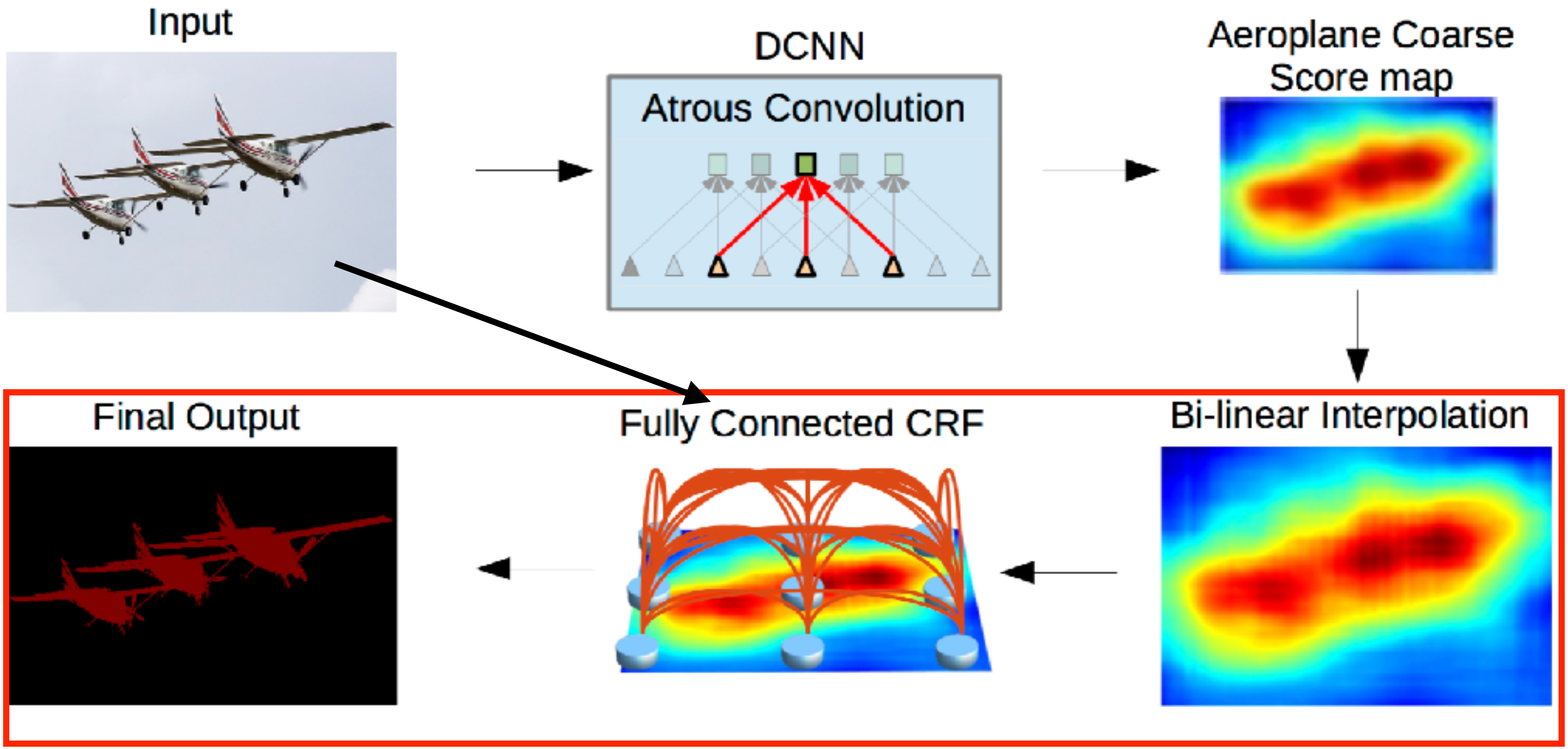
score max (output of the last conv layer before softmax)

belief map (output of the last conv layer after softmax)

- Deep structures are extremely good in recognition tasks but weak in exact border alignment
- CRF refinement needed

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

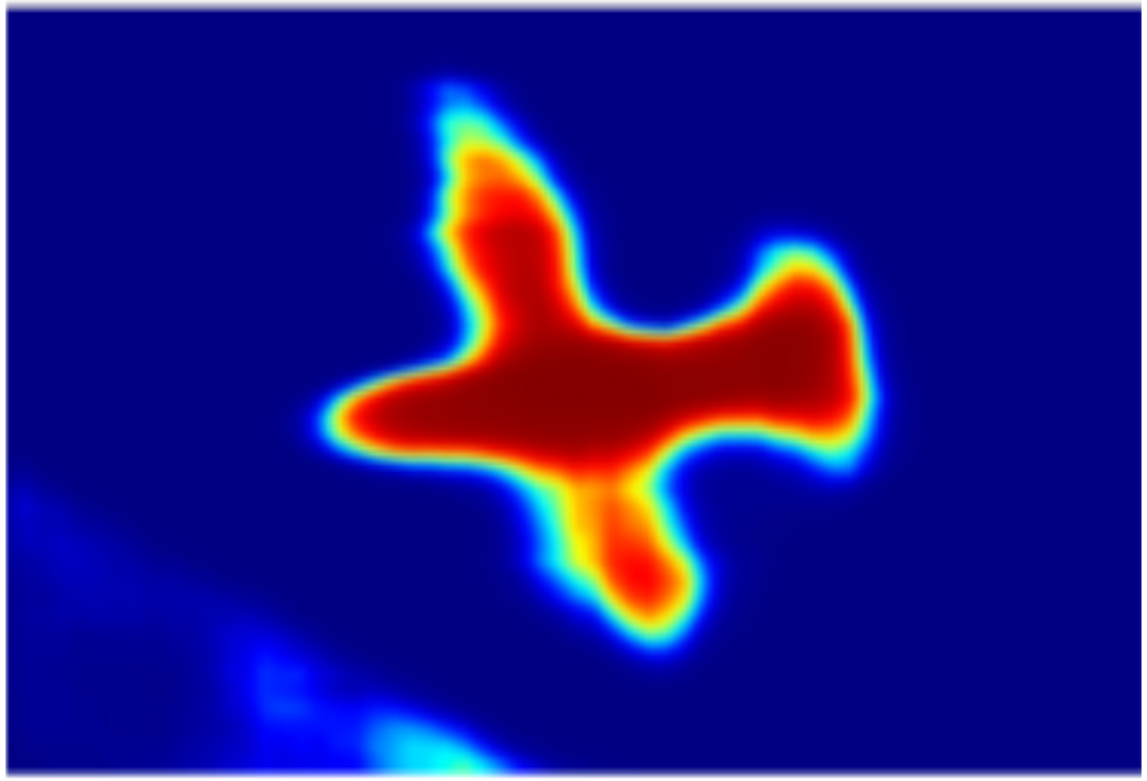
DeepLab v3



- Refinement: increse resolution + do CRF refinement

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

DeepLab v3 - Conditional Random Fields (CRF)

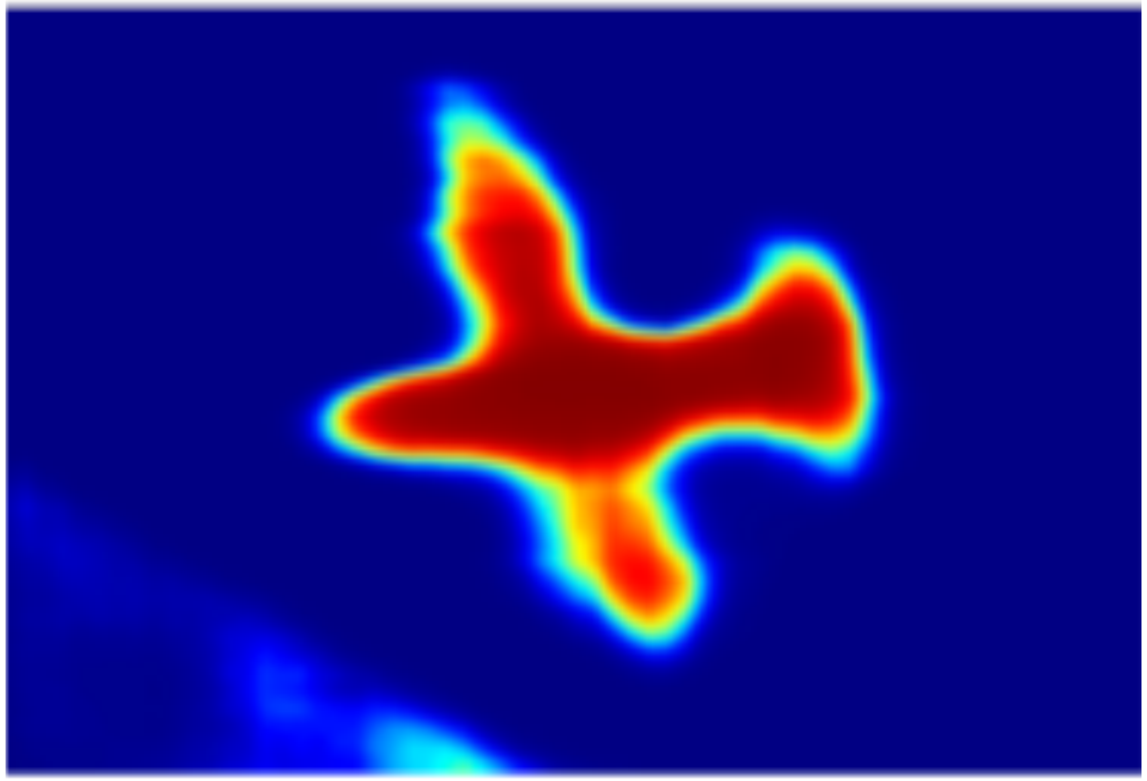


$p(\mathbf{x}_i)$

label of pixel i

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

DeepLab v3 - Conditional Random Fields (CRF)

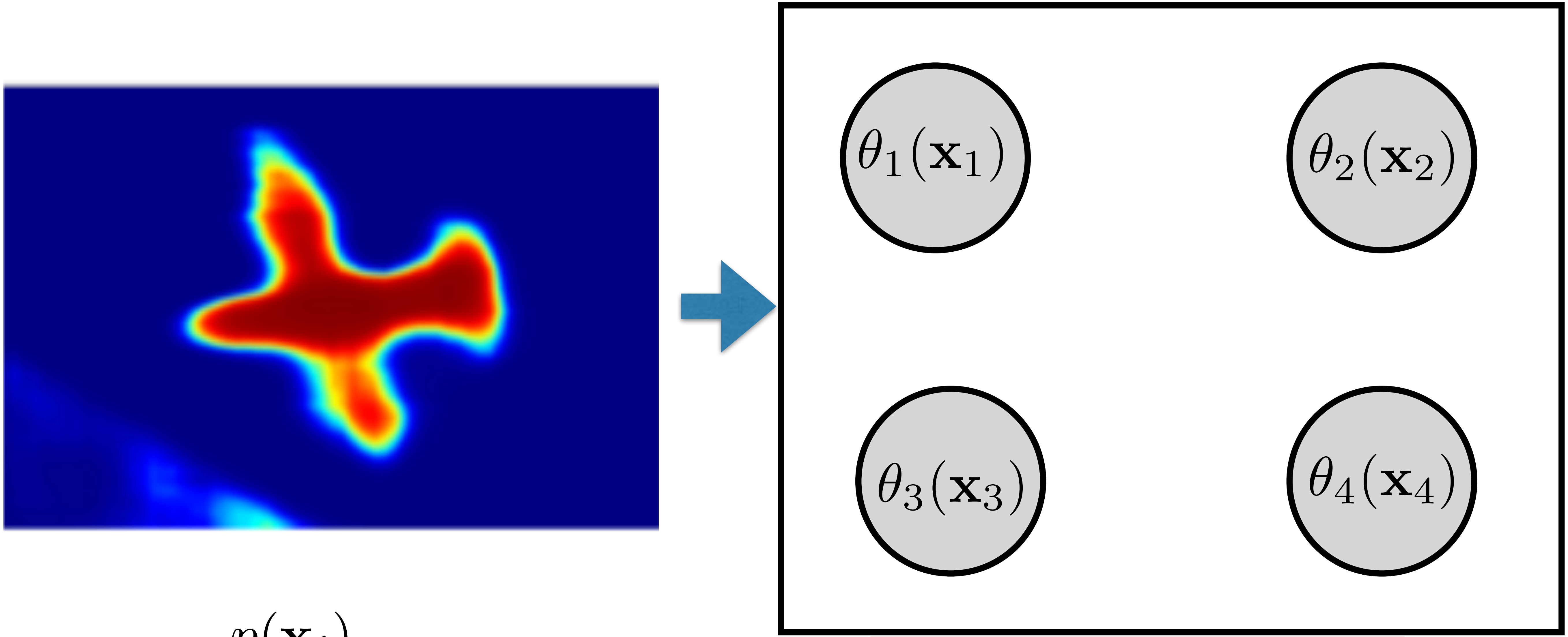


$p(\mathbf{x}_i)$

output of DCNN in pixel i
(probability that pixel i has label $\mathbf{x}_i$)

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

DeepLab v3 - Conditional Random Fields (CRF)



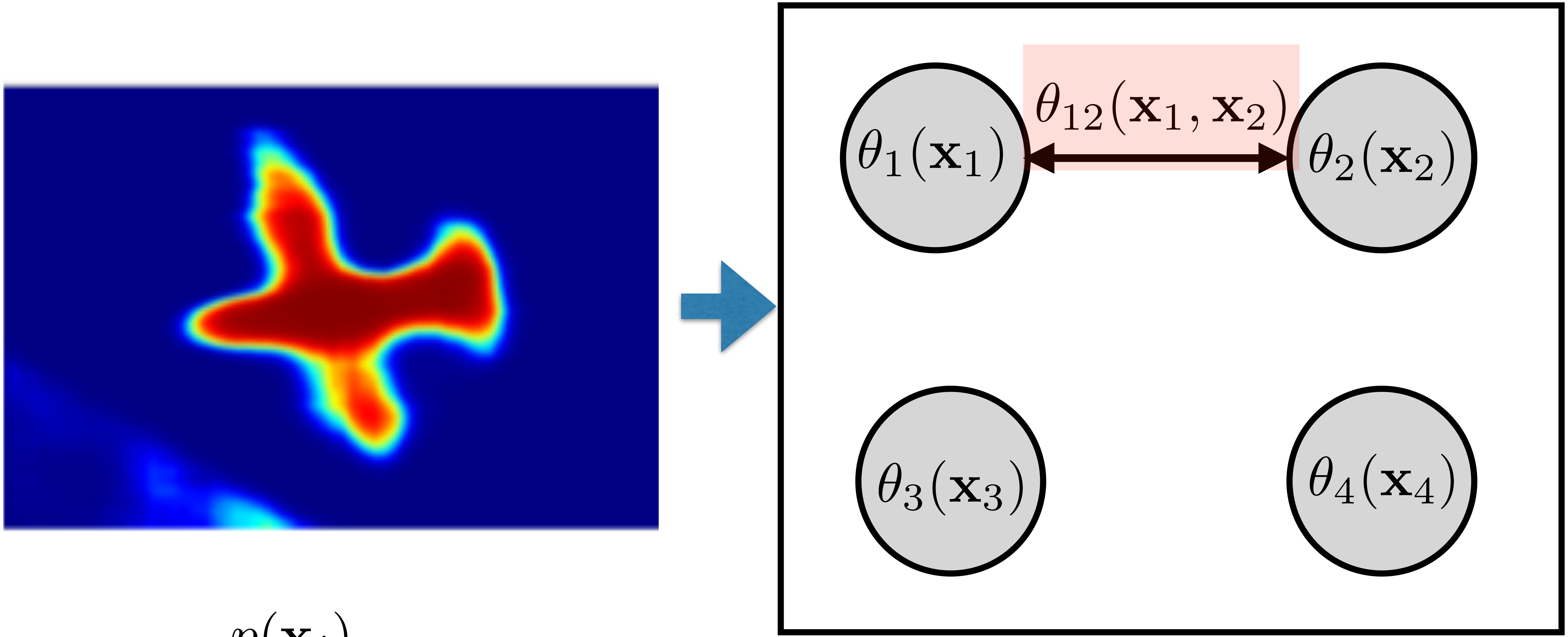
$p(\mathbf{x}_i)$

$\theta_i(\mathbf{x}_i) = -\log(p(\mathbf{x}_i))$

penalty for not following the estimated probability $p(\mathbf{x}_i)$

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

DeepLab v3 - Conditional Random Fields (CRF)



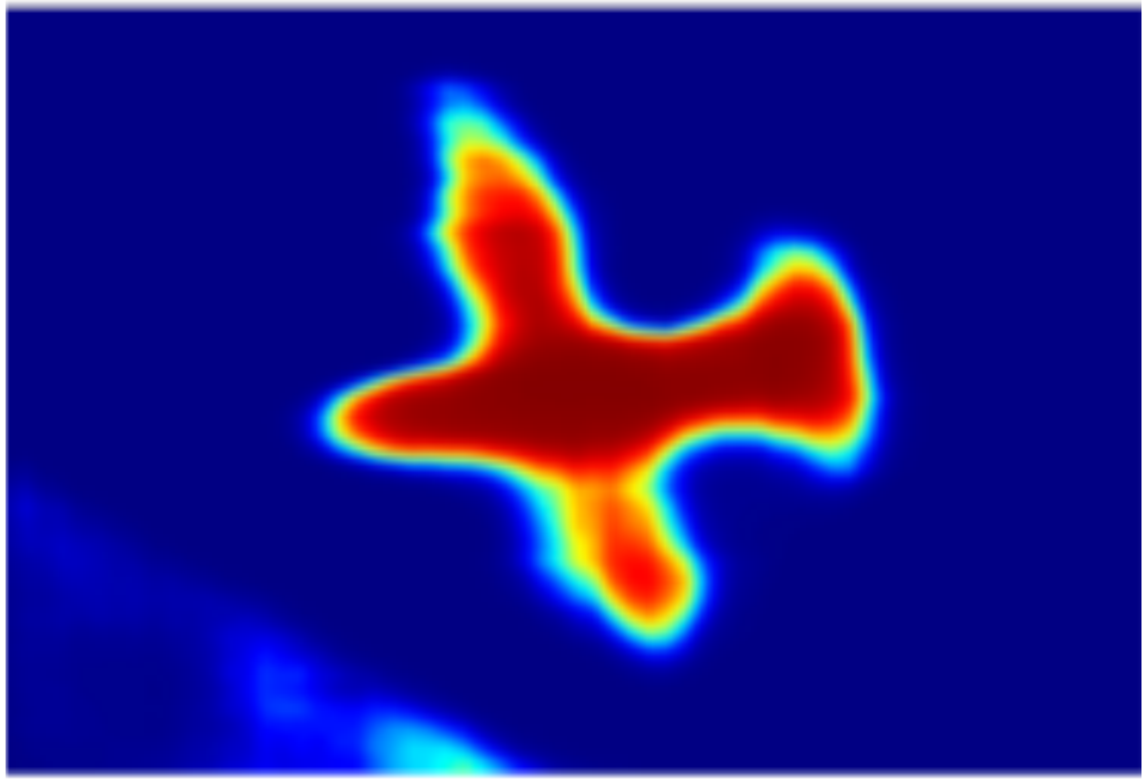
$p(\mathbf{x}_i)$

$$\theta_i(\mathbf{x}_i) = -\log(p(\mathbf{x}_i))$$

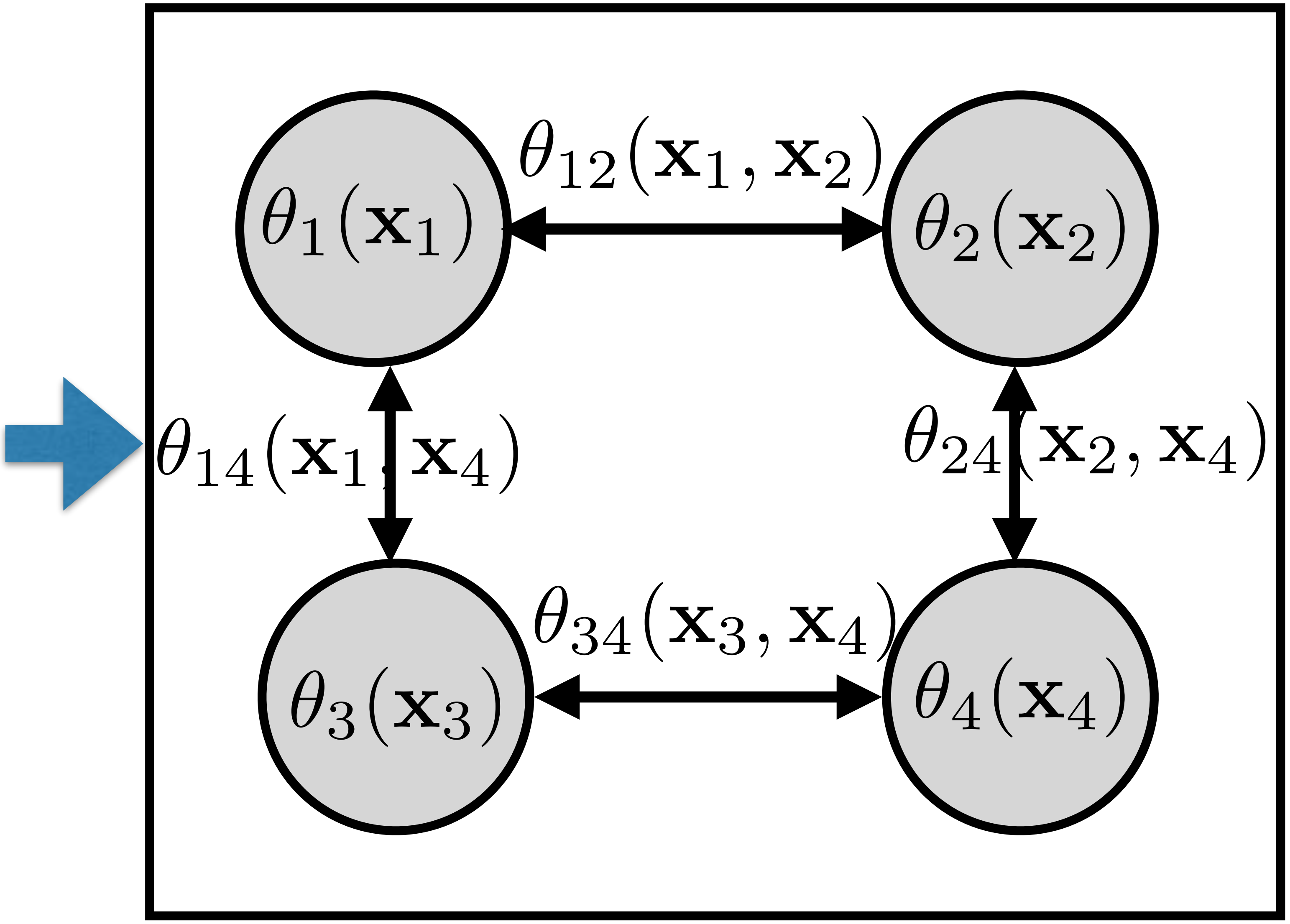
$\theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ = “penalty for dissimilar labels to similar pixels”

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

DeepLab v3 - Conditional Random Fields (CRF)



$$p(\mathbf{x}_i)$$

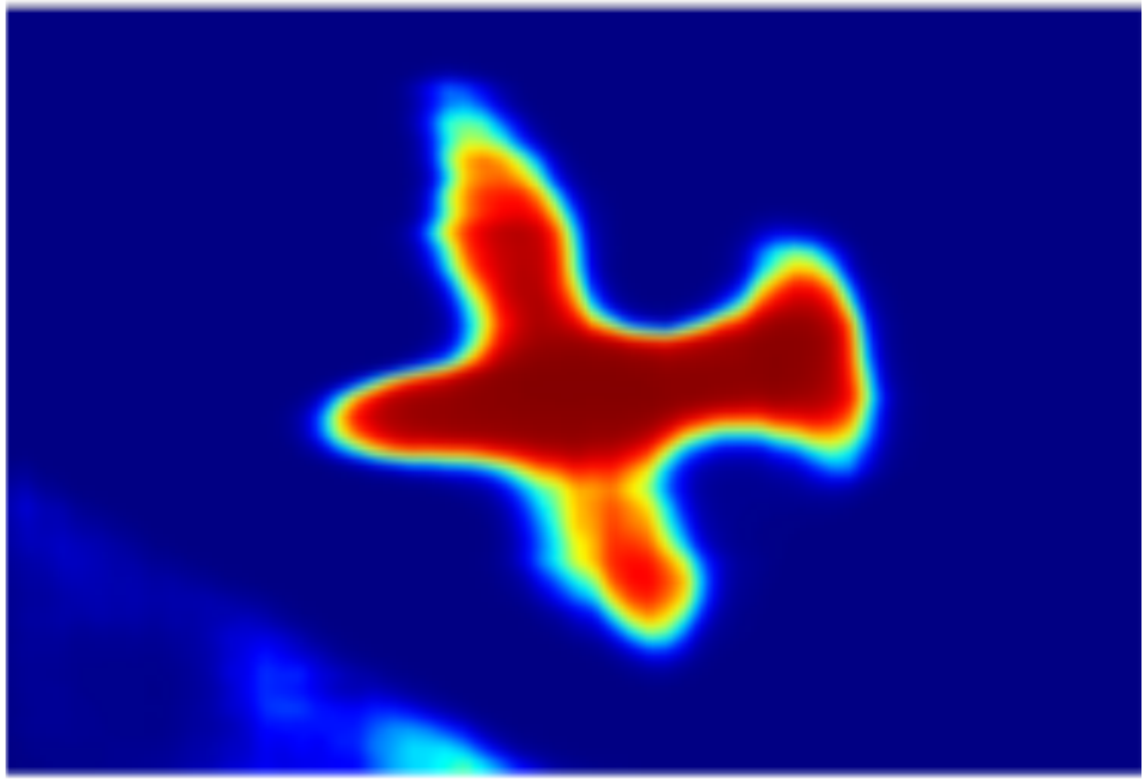

$\theta_1(\mathbf{x}_1)$
$\theta_{12}(\mathbf{x}_1, \mathbf{x}_2)$
$\theta_2(\mathbf{x}_2)$
$\theta_{14}(\mathbf{x}_1, \mathbf{x}_4)$
$\theta_{24}(\mathbf{x}_2, \mathbf{x}_4)$
$\theta_{34}(\mathbf{x}_3, \mathbf{x}_4)$
$\theta_3(\mathbf{x}_3)$
$\theta_4(\mathbf{x}_4)$

$$\theta_i(\mathbf{x}_i) = -\log(p(\mathbf{x}_i))$$

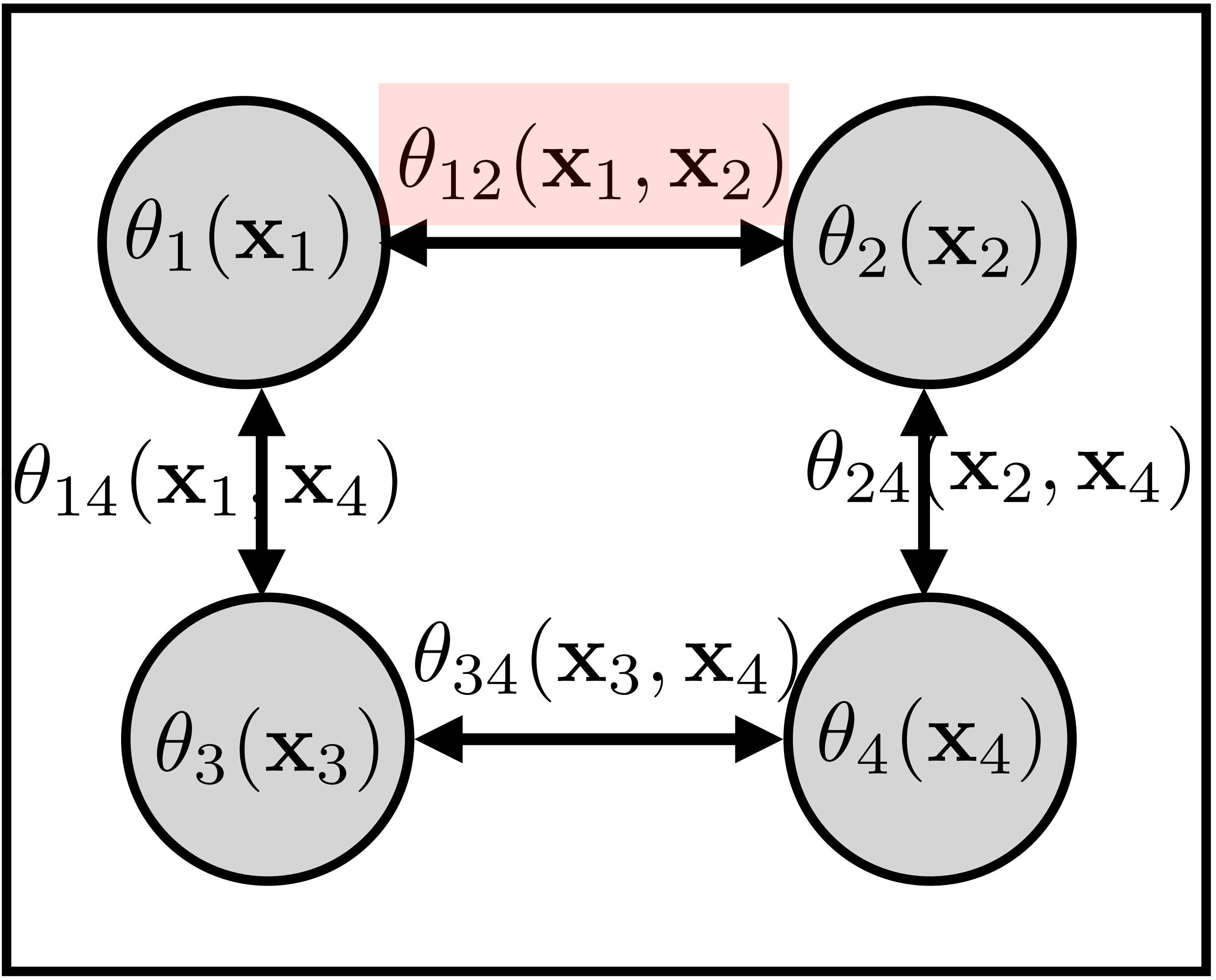
$\theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ = “penalty for dissimilar labels to similar pixels”

[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

DeepLab v3 - Conditional Random Fields (CRF)

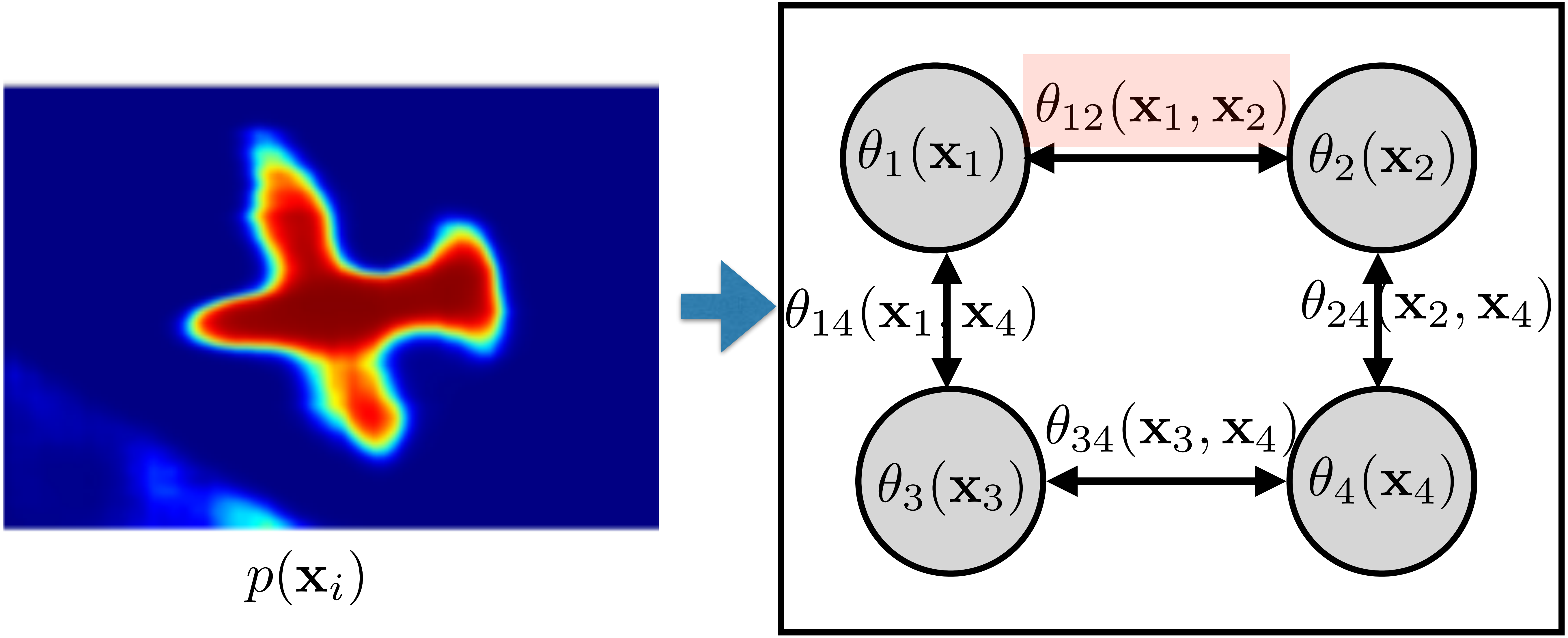


$p(\mathbf{x}_i)$


$\theta_1(\mathbf{x}_1)$
$\theta_{12}(\mathbf{x}_1, \mathbf{x}_2)$
$\theta_2(\mathbf{x}_2)$
$\theta_{14}(\mathbf{x}_1, \mathbf{x}_4)$
$\theta_{24}(\mathbf{x}_2, \mathbf{x}_4)$
$\theta_{34}(\mathbf{x}_3, \mathbf{x}_4)$
$\theta_3(\mathbf{x}_3)$
$\theta_4(\mathbf{x}_4)$

$$\theta_{ij}(x_i, x_j) =$$

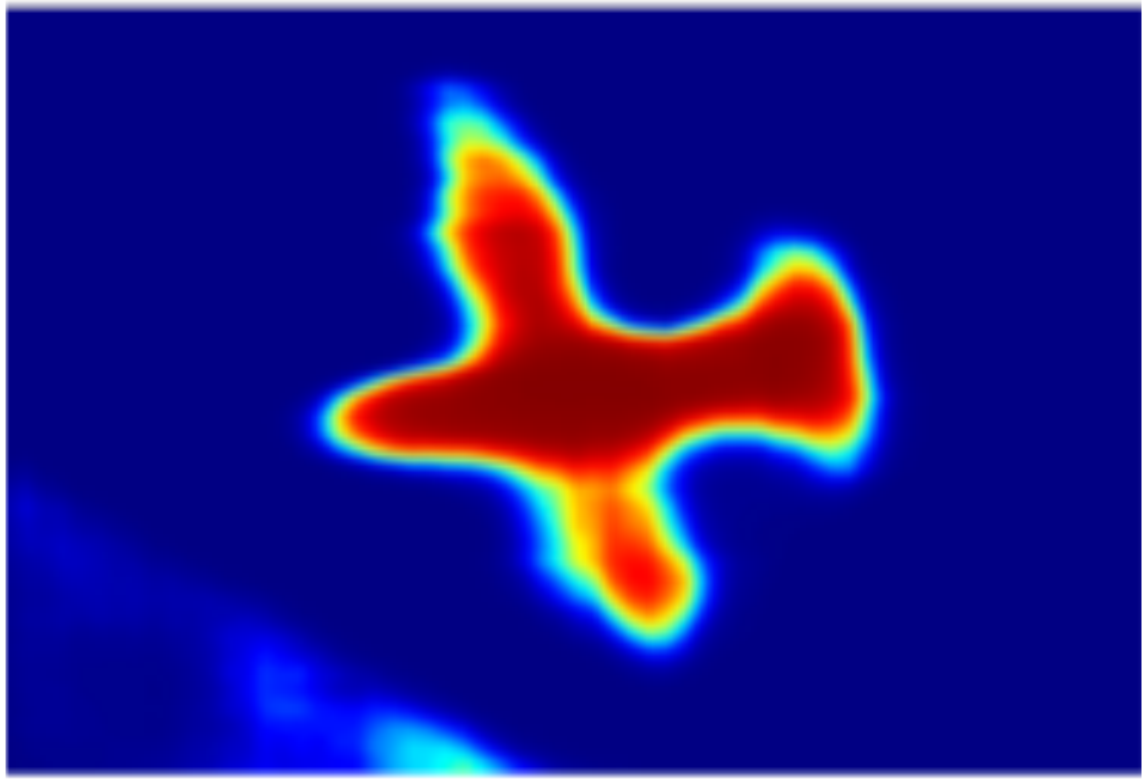
DeepLab v3 - Conditional Random Fields (CRF)


$p(\mathbf{x}_i)$
$\theta_1(\mathbf{x}_1)$
$\theta_{12}(\mathbf{x}_1, \mathbf{x}_2)$
$\theta_2(\mathbf{x}_2)$
$\theta_{14}(\mathbf{x}_1, \mathbf{x}_4)$
$\theta_{24}(\mathbf{x}_2, \mathbf{x}_4)$
$\theta_{34}(\mathbf{x}_3, \mathbf{x}_4)$
$\theta_3(\mathbf{x}_3)$
$\theta_4(\mathbf{x}_4)$

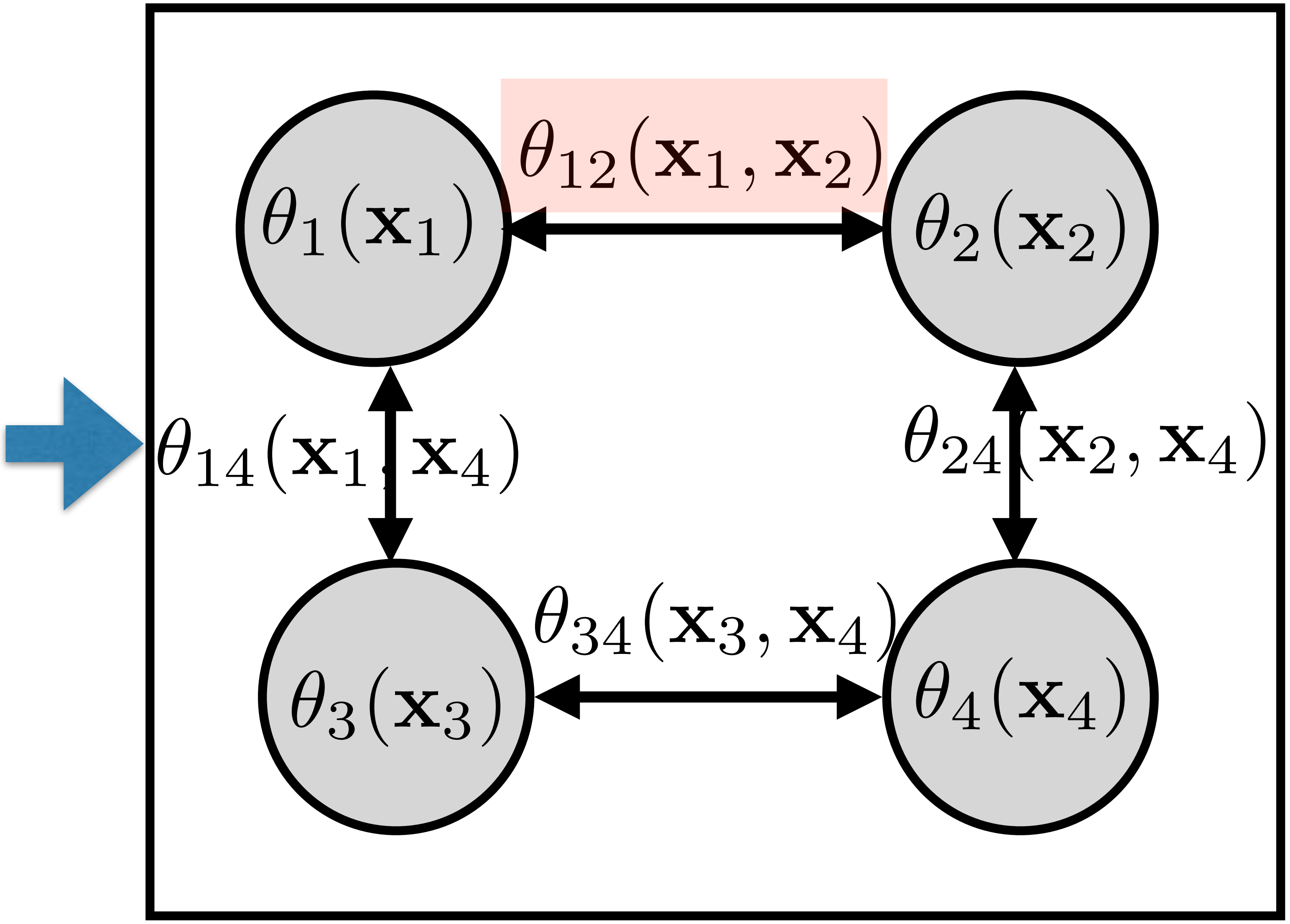
$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \begin{cases} 1 & x_i \neq x_j \\ 0 & x_i = x_j \end{cases}$$

same labels are not penalized

DeepLab v3 - Conditional Random Fields (CRF)



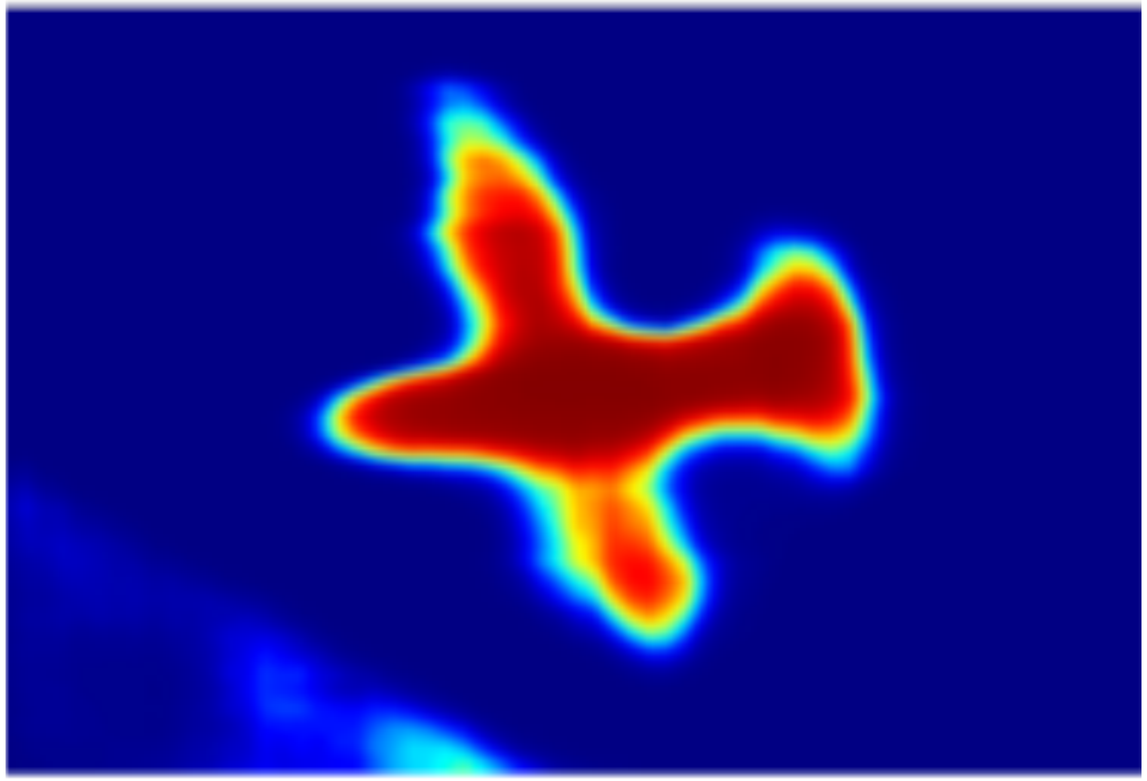
$p(\mathbf{x}_i)$


$\theta_1(\mathbf{x}_1)$
$\theta_{12}(\mathbf{x}_1, \mathbf{x}_2)$
$\theta_2(\mathbf{x}_2)$
$\theta_{14}(\mathbf{x}_1, \mathbf{x}_4)$
$\theta_{24}(\mathbf{x}_2, \mathbf{x}_4)$
$\theta_{34}(\mathbf{x}_3, \mathbf{x}_4)$
$\theta_3(\mathbf{x}_3)$
$\theta_4(\mathbf{x}_4)$

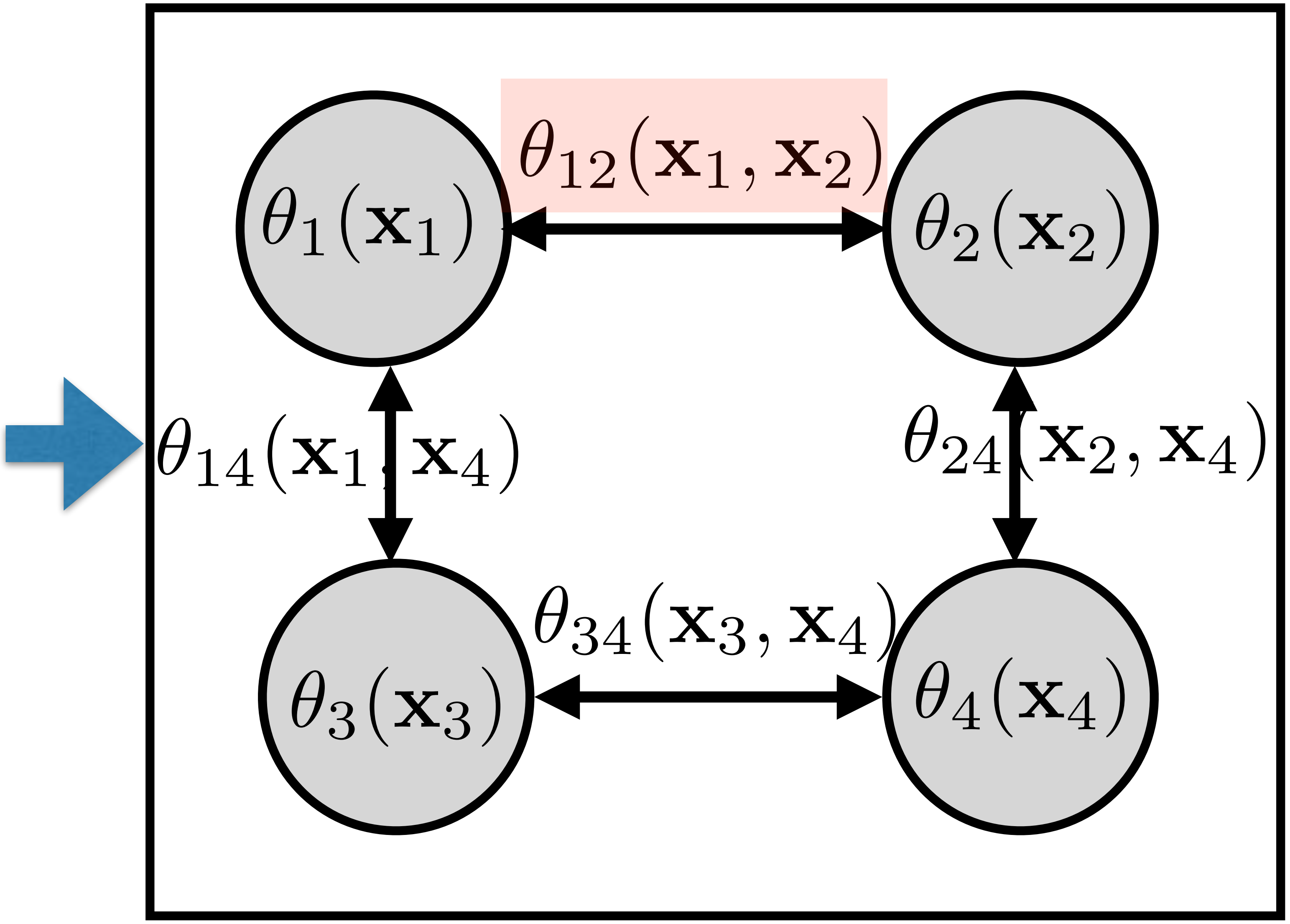
$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j)\Bigg[w_1 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2}\right)$$

high penalty for different labels, when pixels are
(i) spatially close and (ii) has similar color

DeepLab v3 - Conditional Random Fields (CRF)



$p(\mathbf{x}_i)$

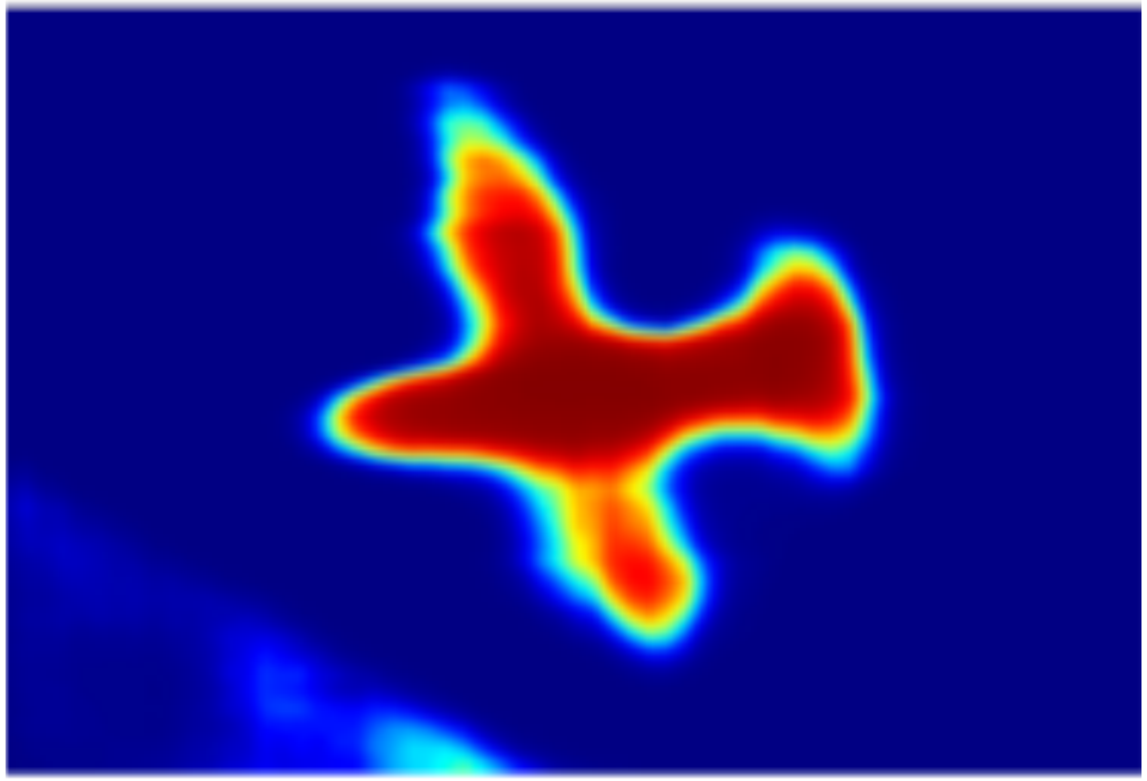


$\theta_{12}(\mathbf{x}_1, \mathbf{x}_2)$
$\theta_1(\mathbf{x}_1)$
$\theta_2(\mathbf{x}_2)$
$\theta_{14}(\mathbf{x}_1, \mathbf{x}_4)$
$\theta_{24}(\mathbf{x}_2, \mathbf{x}_4)$
$\theta_{34}(\mathbf{x}_3, \mathbf{x}_4)$
$\theta_3(\mathbf{x}_3)$
$\theta_4(\mathbf{x}_4)$

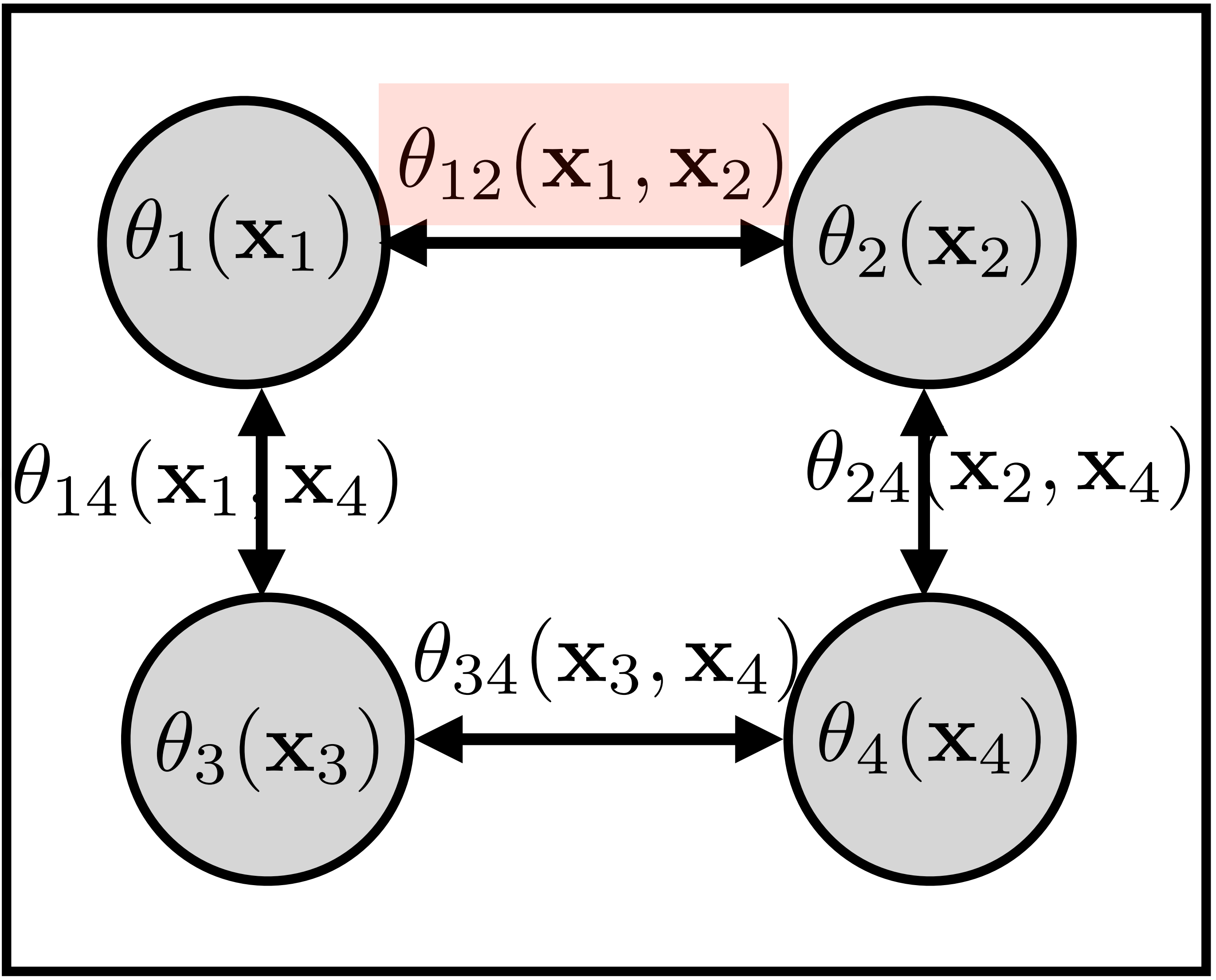
$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \left[w_1 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2} \right) \right.$$

high penalty for different labels, when pixels are (i) spatially close and (ii) has similar color

DeepLab v3 - Conditional Random Fields (CRF)



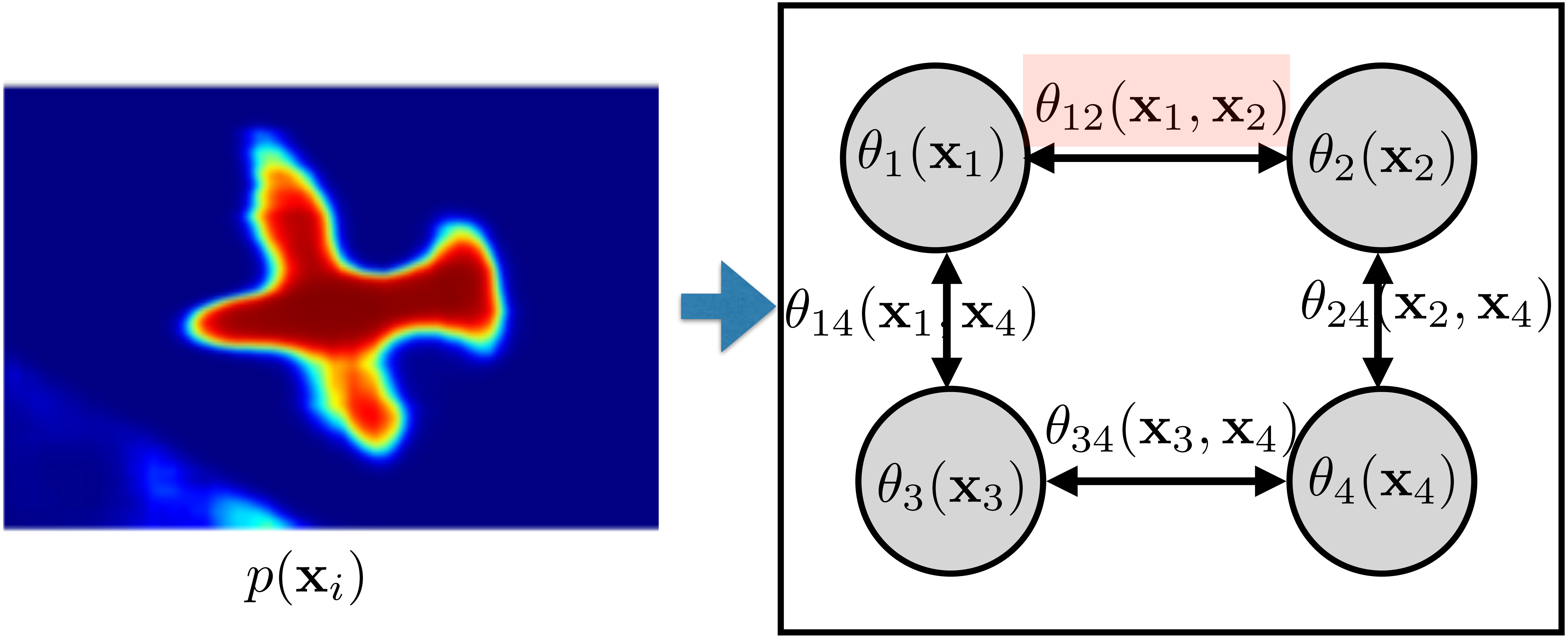
$p(\mathbf{x}_i)$


$\theta_{12}(\mathbf{x}_1, \mathbf{x}_2)$
$\theta_1(\mathbf{x}_1)$
$\theta_2(\mathbf{x}_2)$
$\theta_{14}(\mathbf{x}_1, \mathbf{x}_4)$
$\theta_{24}(\mathbf{x}_2, \mathbf{x}_4)$
$\theta_{34}(\mathbf{x}_3, \mathbf{x}_4)$
$\theta_3(\mathbf{x}_3)$
$\theta_4(\mathbf{x}_4)$

$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \left[w_1 \exp \left(- \frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2} \right) \right.$$

high penalty for different labels, when pixels are (i) spatially close and (ii) has similar color

DeepLab v3 - Conditional Random Fields (CRF)

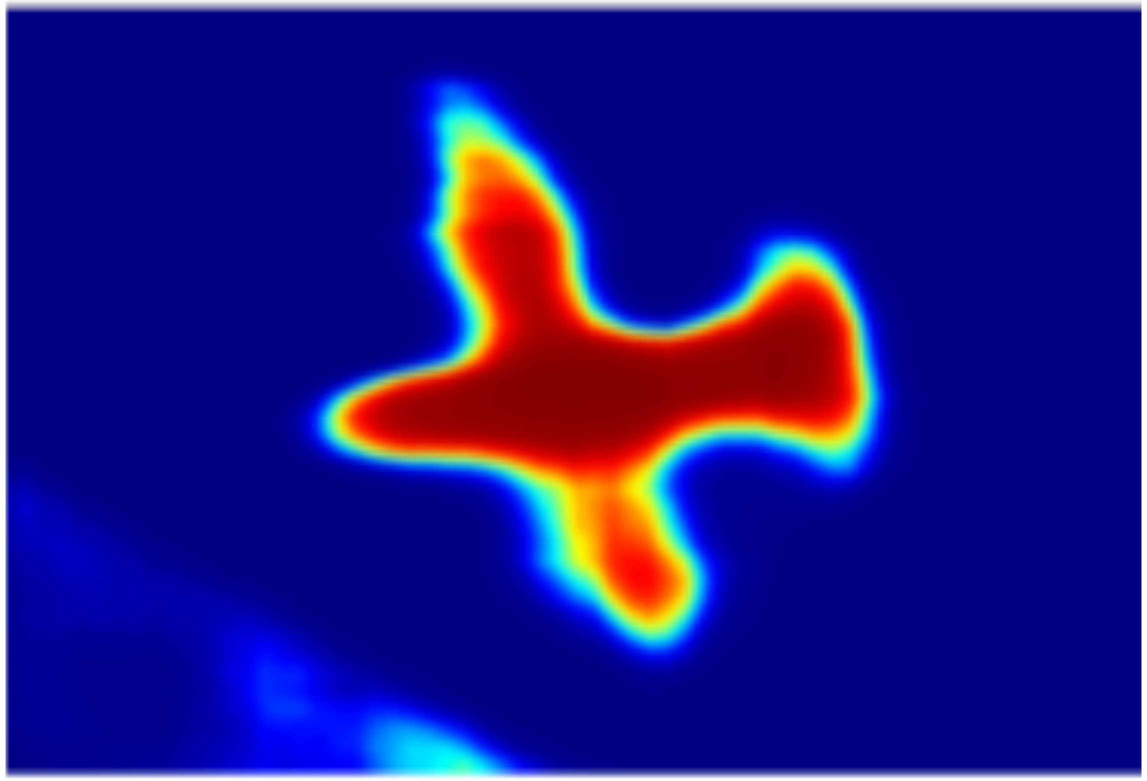

$p(\mathbf{x}_i)$
$\theta_{12}(\mathbf{x}_1, \mathbf{x}_2)$
$\theta_1(\mathbf{x}_1)$
$\theta_2(\mathbf{x}_2)$
$\theta_{14}(\mathbf{x}_1, \mathbf{x}_4)$
$\theta_{24}(\mathbf{x}_2, \mathbf{x}_4)$
$\theta_{34}(\mathbf{x}_3, \mathbf{x}_4)$
$\theta_3(\mathbf{x}_3)$
$\theta_4(\mathbf{x}_4)$

$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j)\left[w_1 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2}\right)\right.$$

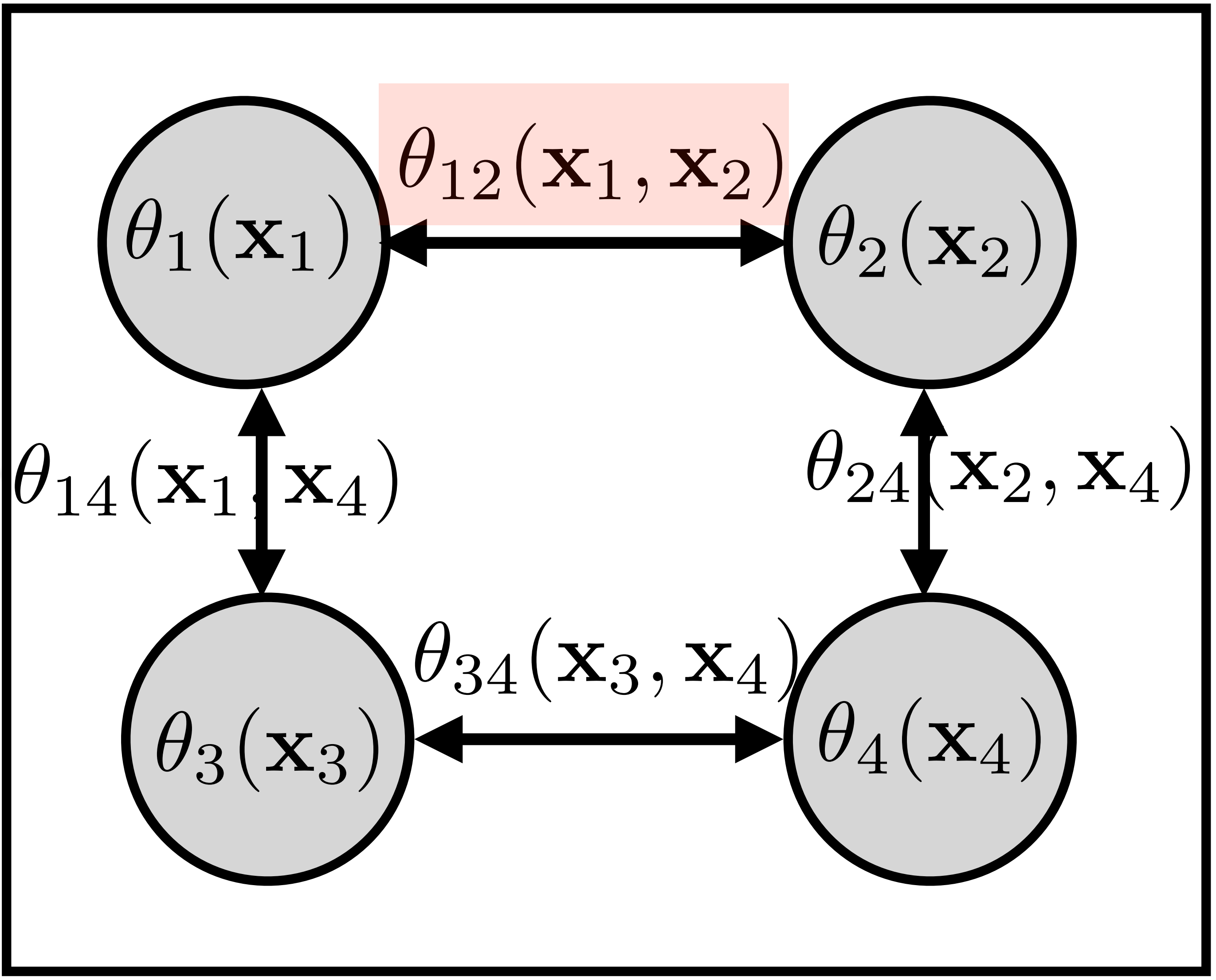
another penalty for close pixels

$$\left. + w_2 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\gamma^2}\right)\right]$$

DeepLab v3 - Conditional Random Fields (CRF)



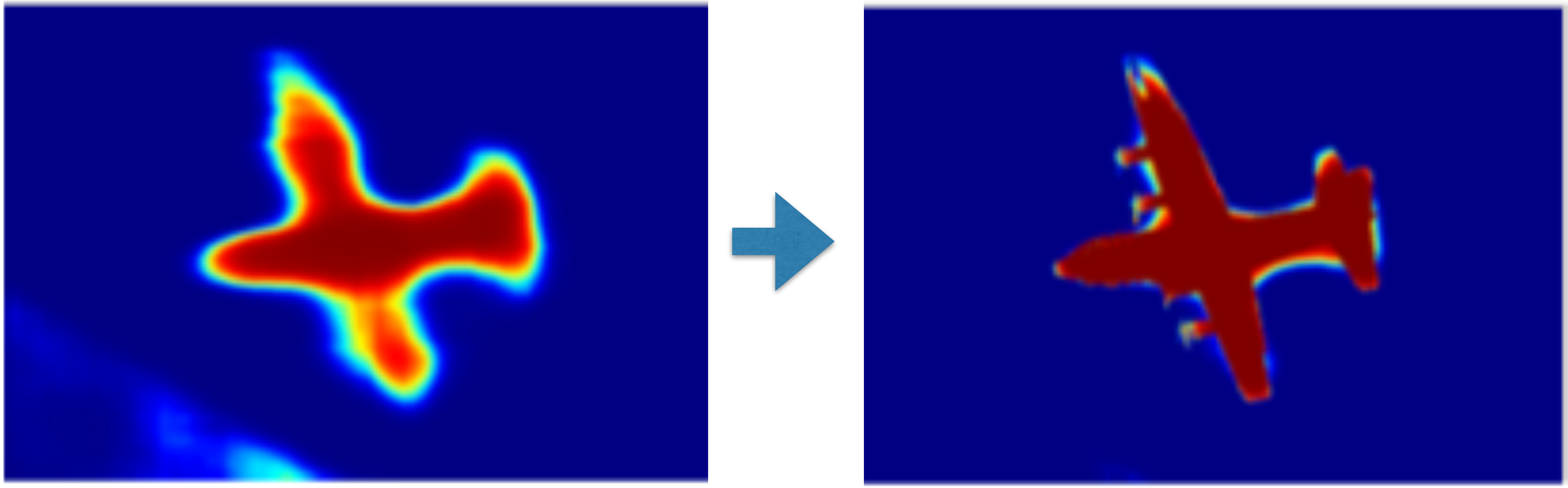
$p(\mathbf{x}_i)$


$\theta_{12}(\mathbf{x}_1, \mathbf{x}_2)$
$\theta_1(\mathbf{x}_1)$
$\theta_2(\mathbf{x}_2)$
$\theta_{14}(\mathbf{x}_1, \mathbf{x}_4)$
$\theta_{24}(\mathbf{x}_2, \mathbf{x}_4)$
$\theta_{34}(\mathbf{x}_3, \mathbf{x}_4)$
$\theta_3(\mathbf{x}_3)$
$\theta_4(\mathbf{x}_4)$

$$E(\mathbf{x}) = \sum_i \theta_i(\mathbf{x}_i) + \sum_{ij} \theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)$$

$$\arg\min_{\mathbf{x}} E(\mathbf{x})$$

DeepLab v3 - Conditional Random Fields (CRF)



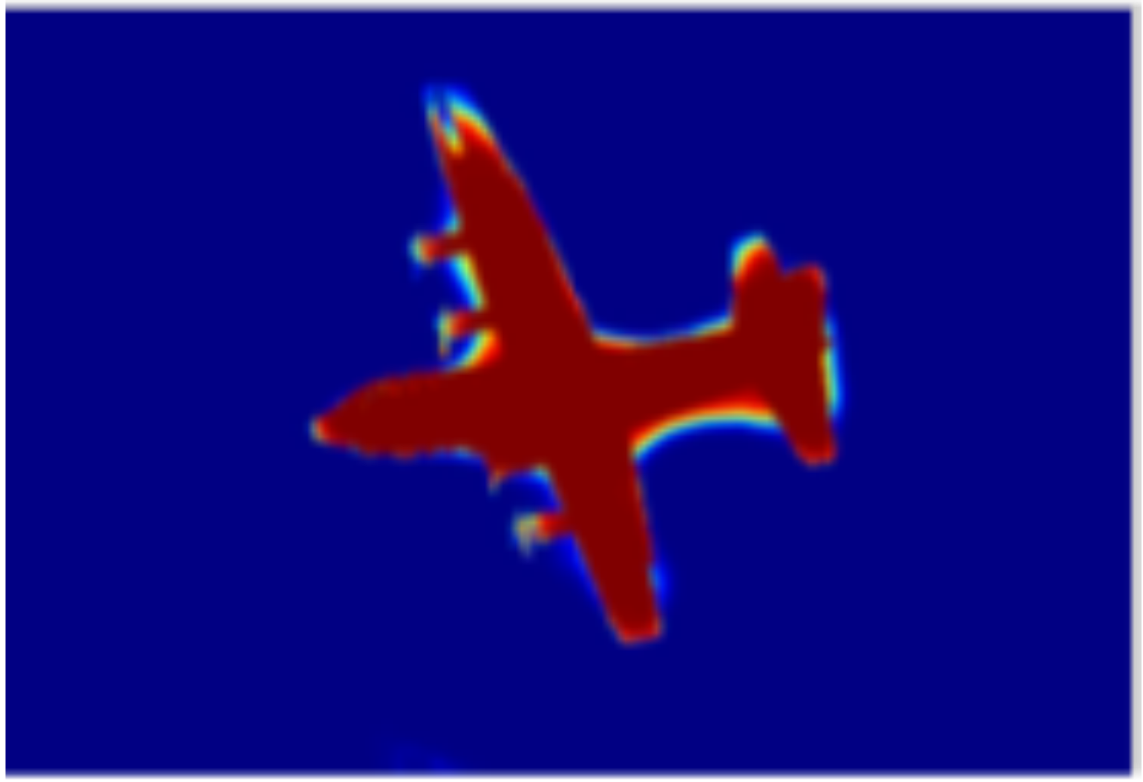
$p(\mathbf{x}_i)$

$\arg\min_{\mathbf{x}} E(\mathbf{x})$

$$E(\mathbf{x}) = \sum_i \theta_i(\mathbf{x}_i) + \sum_{ij} \theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)$$

- Direct optimization complicated (NP complete)
- You can fix $\mathbf{x}_j$ and find approx. closed form solution over $\mathbf{x}_i$

DeepLab v3 - Conditional Random Fields (CRF)



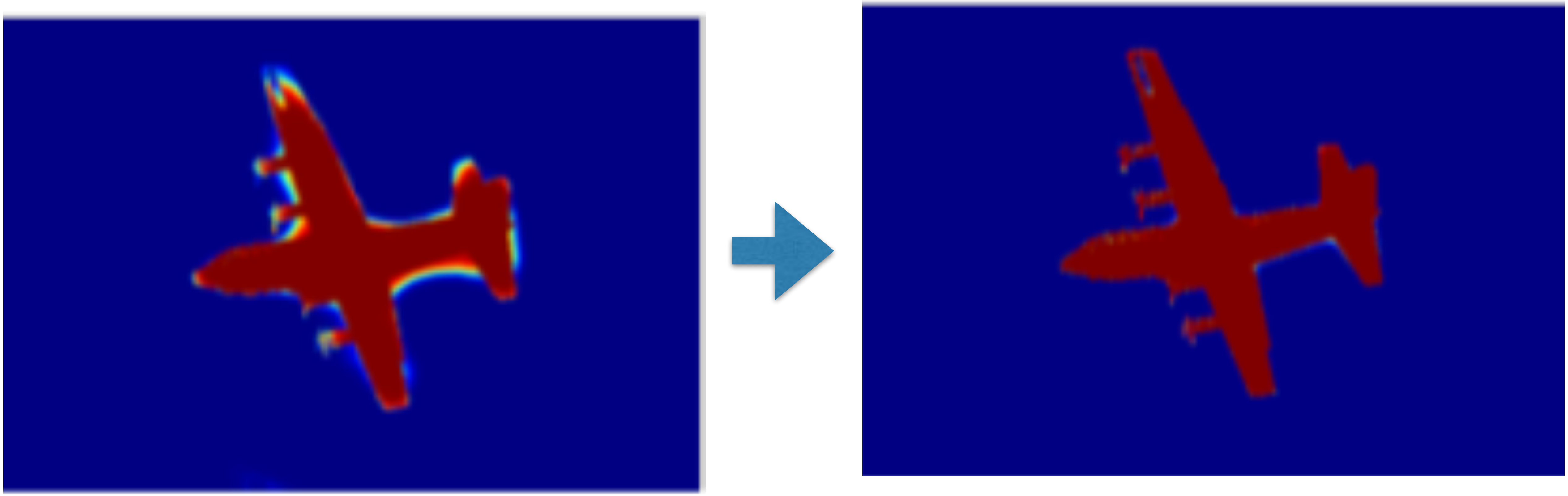
$p(\mathbf{x}_i)$

$\arg\min_{\mathbf{x}} E(\mathbf{x})$

$$E(\mathbf{x}) = \sum_i \theta_i(\mathbf{x}_i) + \sum_{ij} \theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)$$

iterate

DeepLab v3 - Conditional Random Fields (CRF)



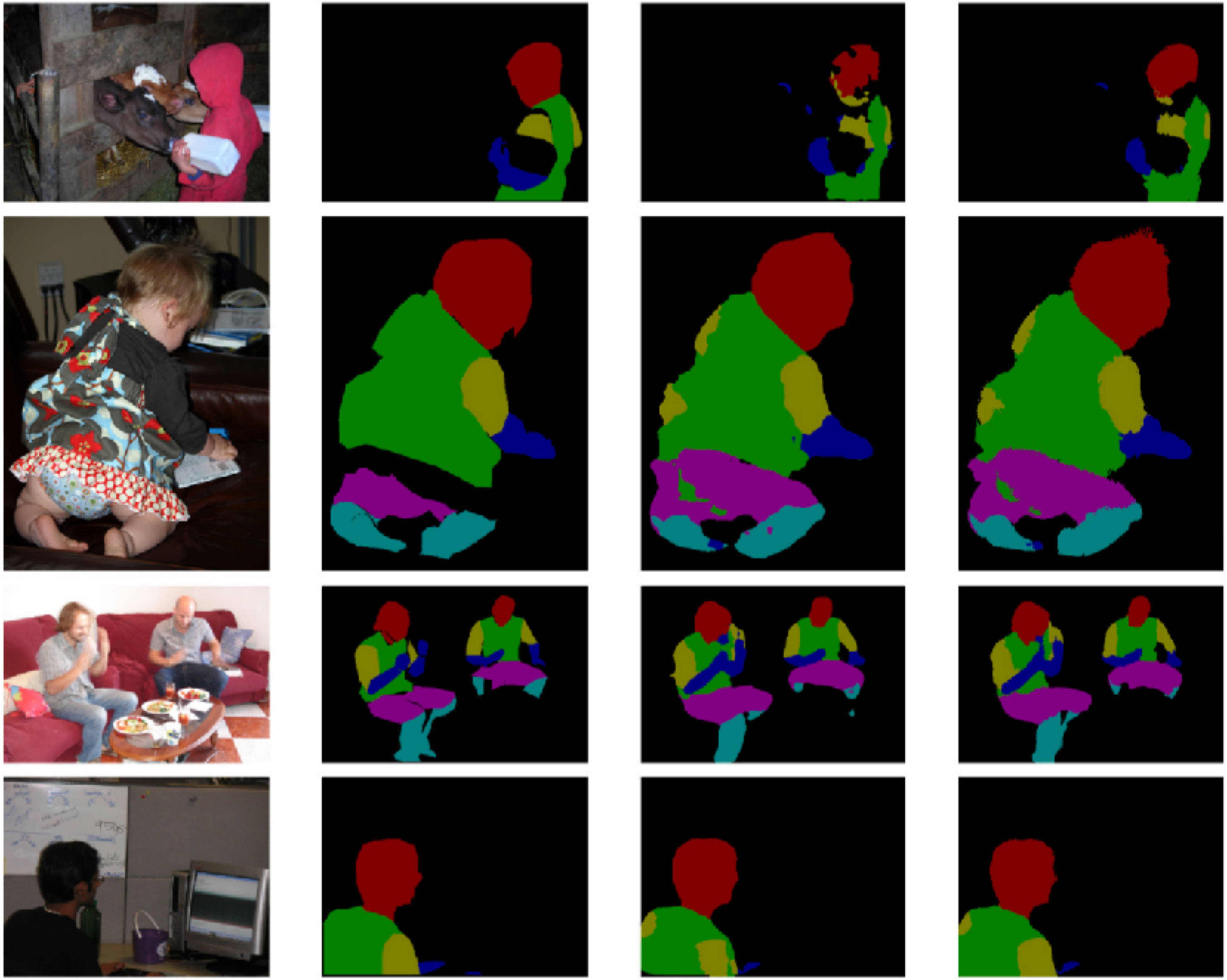
$p(\mathbf{x}_i)$

$\arg\min_{\mathbf{x}} E(\mathbf{x})$

$$E(\mathbf{x}) = \sum_i \theta_i(\mathbf{x}_i) + \sum_{ij} \theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)$$

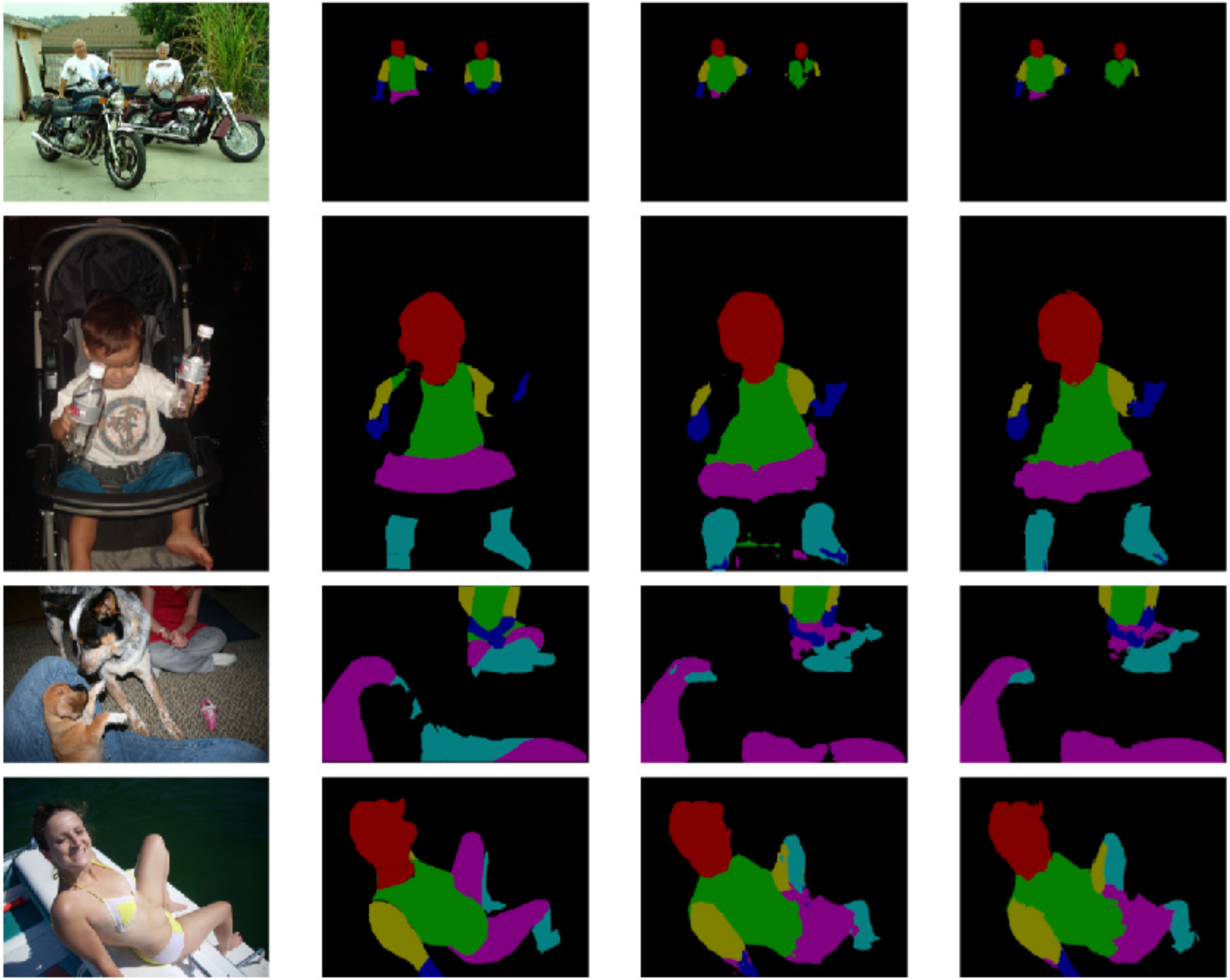
iterate (result shown after 10 iterations)

DeepLab v3 - results



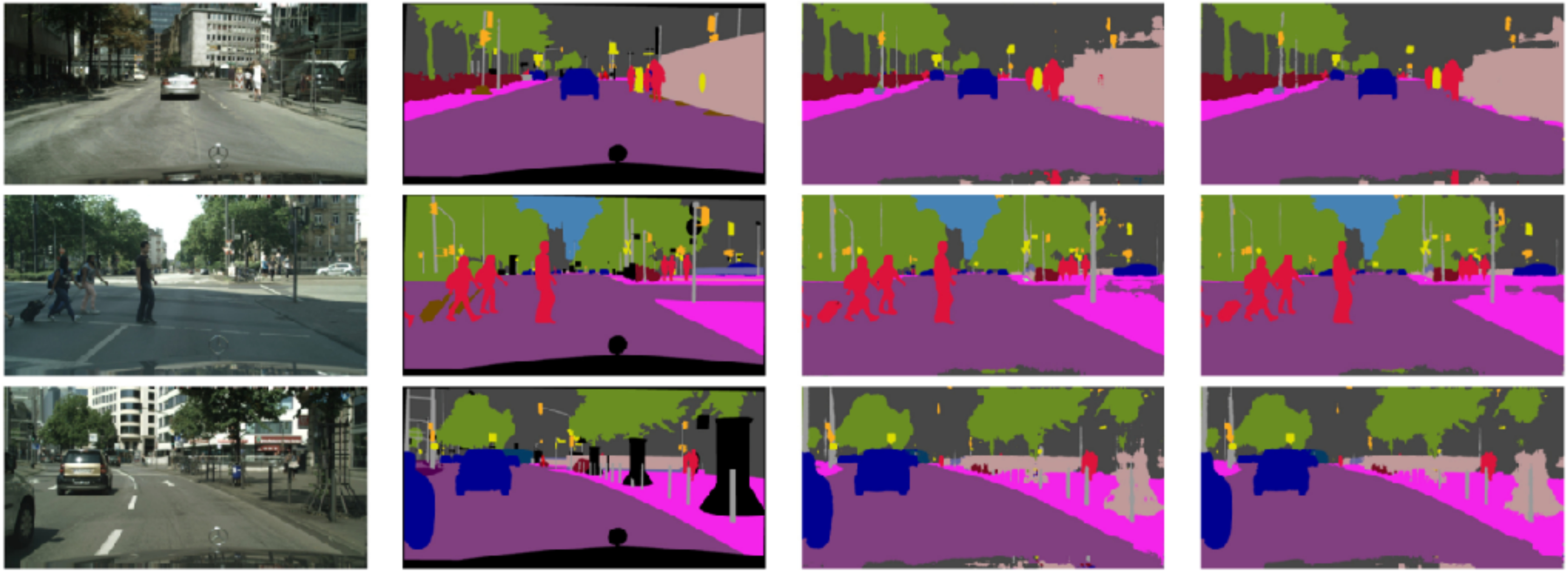
(a) Image (b) G.T. (c) Before CRF (d) After CRF

DeepLab v3 - results



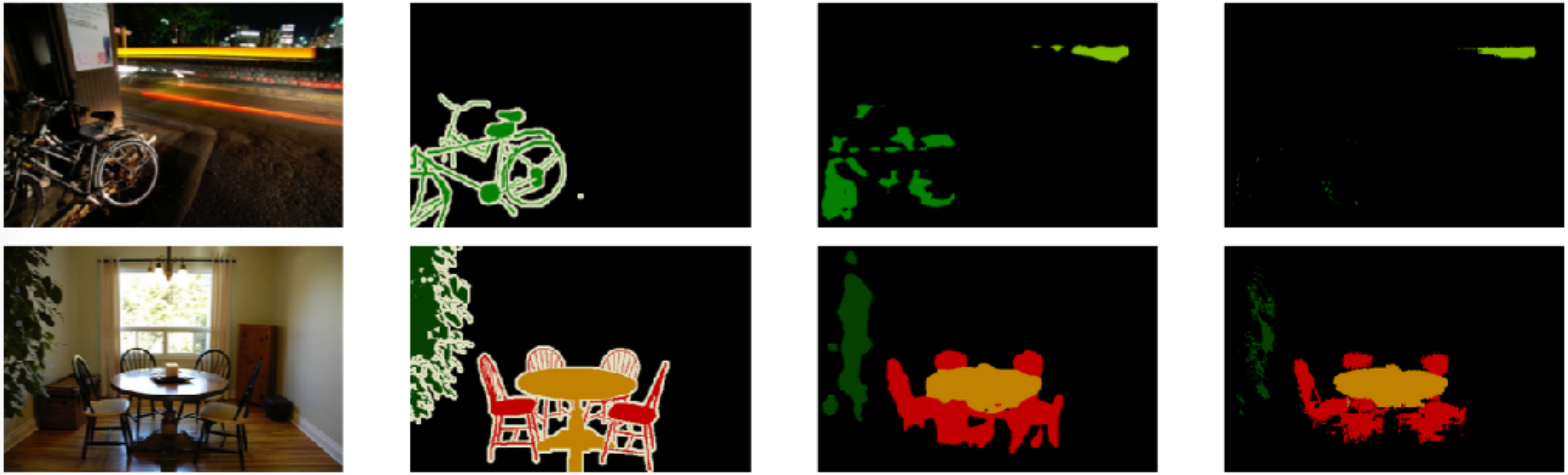
(a) Image (b) G.T. (c) Before CRF (d) After CRF

DeepLab v3 - results



(a) Image (b) G.T. (c) Before CRF (d) After CRF

DeepLab v3 - results



(a) Image (b) G.T. (c) Before CRF (d) After CRF

CRF failure cases

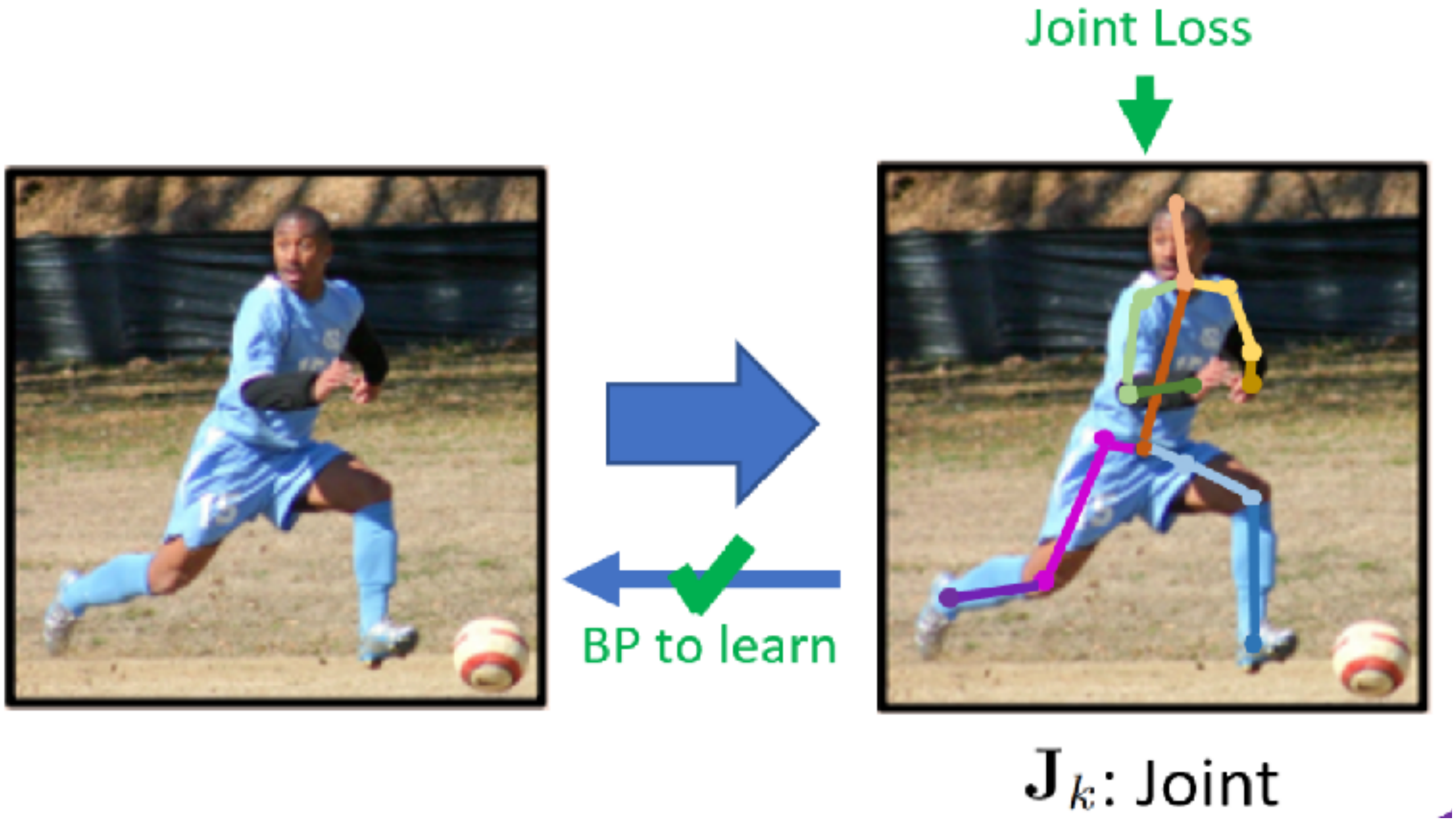
DeepLab v3 - summary

- significantly outperforms state-of-the-art on several datasets
- CRF improves mIOU about 2%
- ASPP improves mIOU about 3%
- codes available: https://github.com/tensorflow/models/tree/master/research/deeplab
- state-of-the-art benchmarks: http://www.robustvision.net/leaderboard.php?benchmark=semantic
- Dense ASPP [Yang et a; CVPR 2018] http://openaccess.thecvf.com/content_cvpr_2018/papers/Yang_DenseASPP_for_Semantic_CVPR_2018_paper.pdf

Outline

- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Architectures of regression networks
- Architectures of feature matching networks

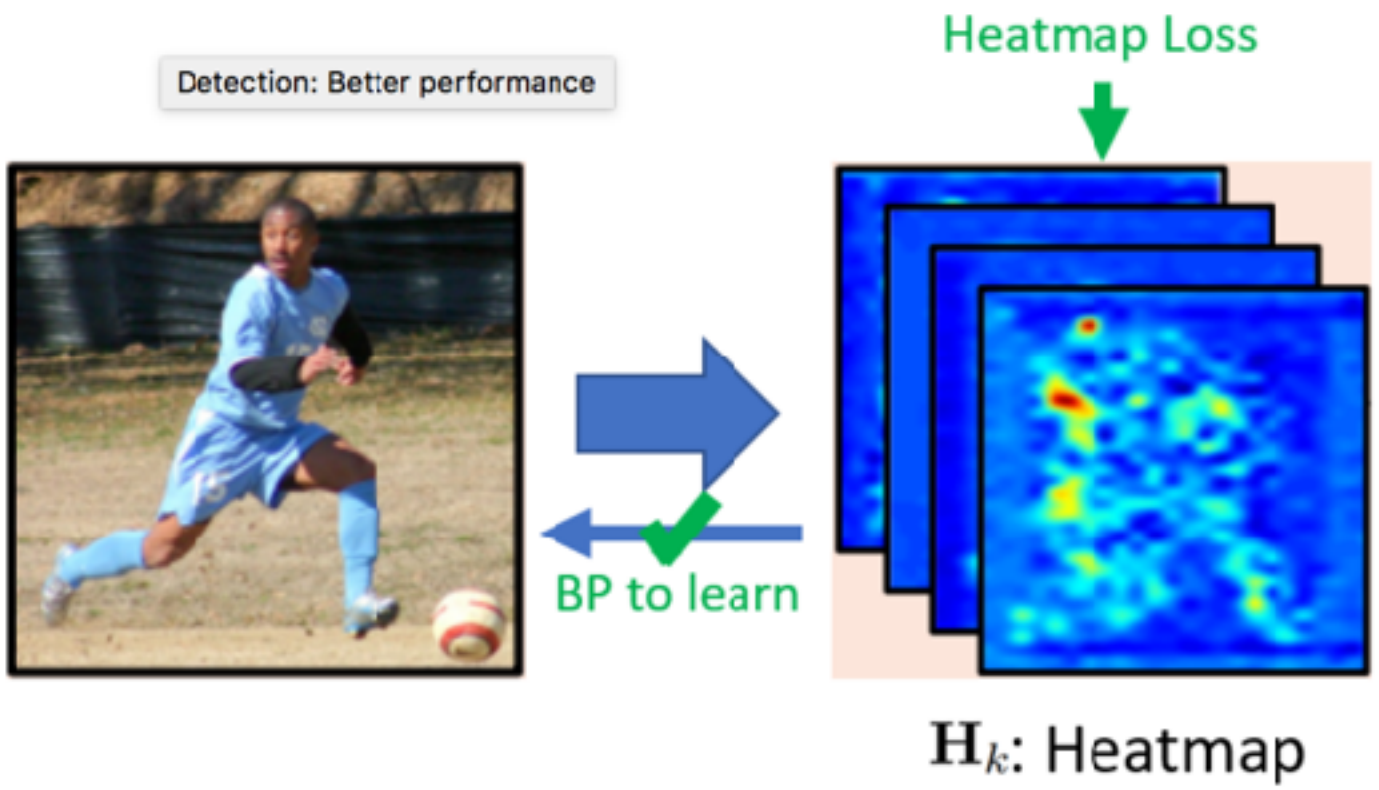
Pose regression baseline



- ConvNet directly estimates joint positions (2xN real numbers)
- Straightforward learning directly minimize L2 loss over all joint positions (2D/3D).

Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

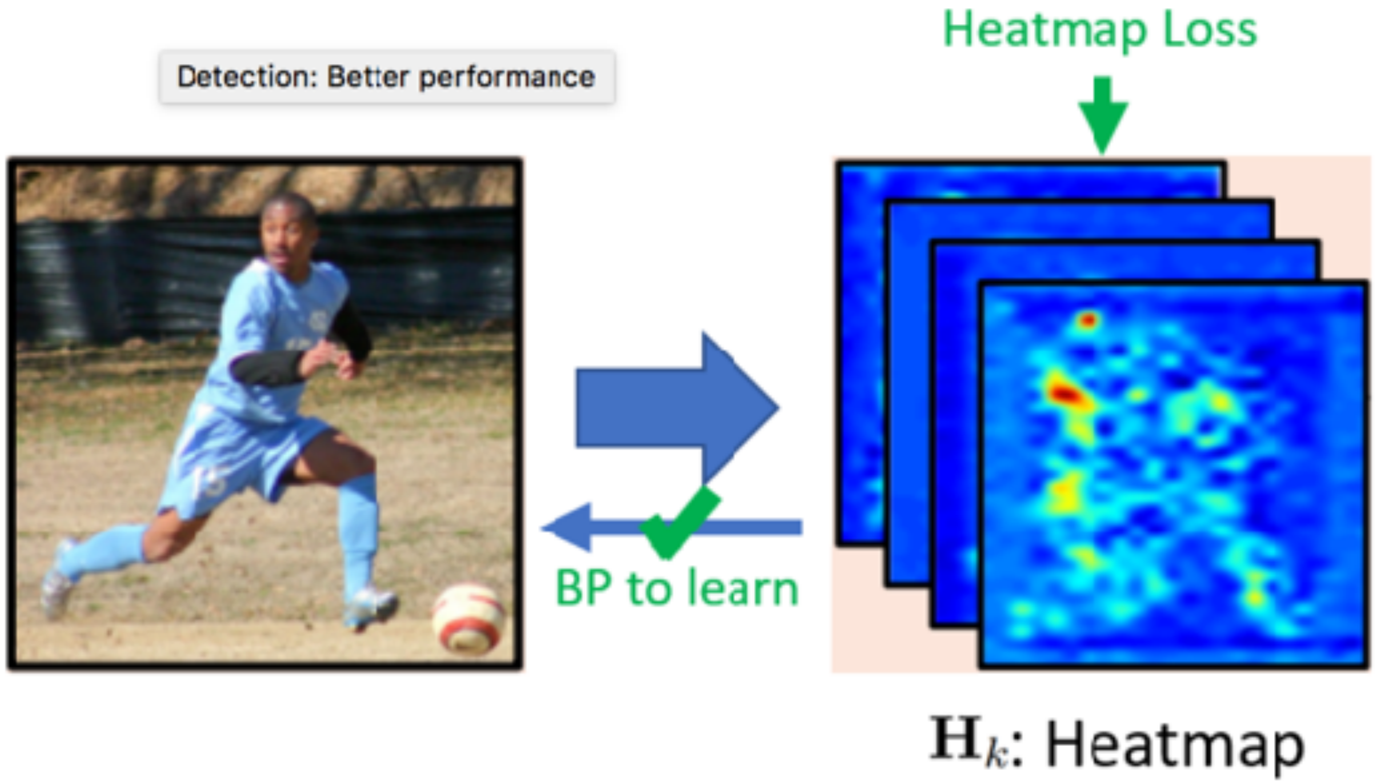
Pose detection baseline



- ConvNet first estimates N joint's heat maps $\mathbf{H}_k,\ k = 1 \ldots N$ (i.e. N 2D-images or N 3D-arrays)
- Learning minimizes segmentation loss over the N images

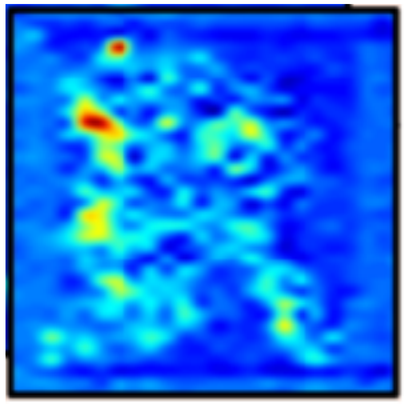
Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

Pose detection baseline



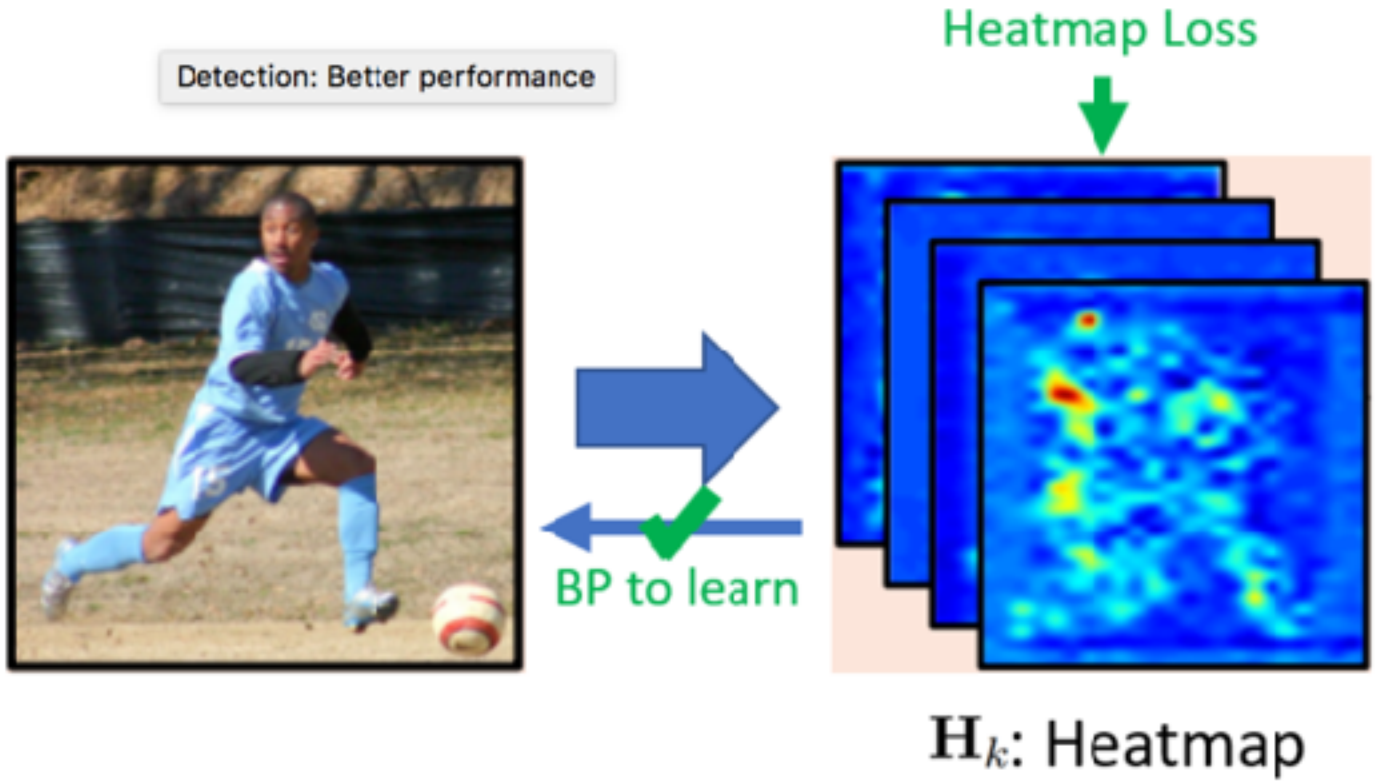
- estimate joint position as position of heatmap maximum

$\mathbf{H}_k$

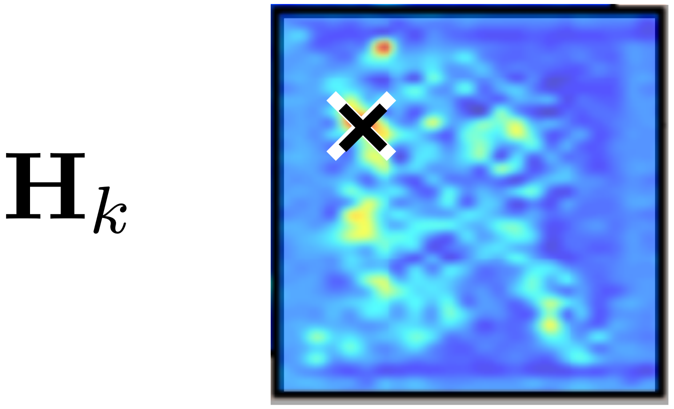


Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

Pose detection baseline

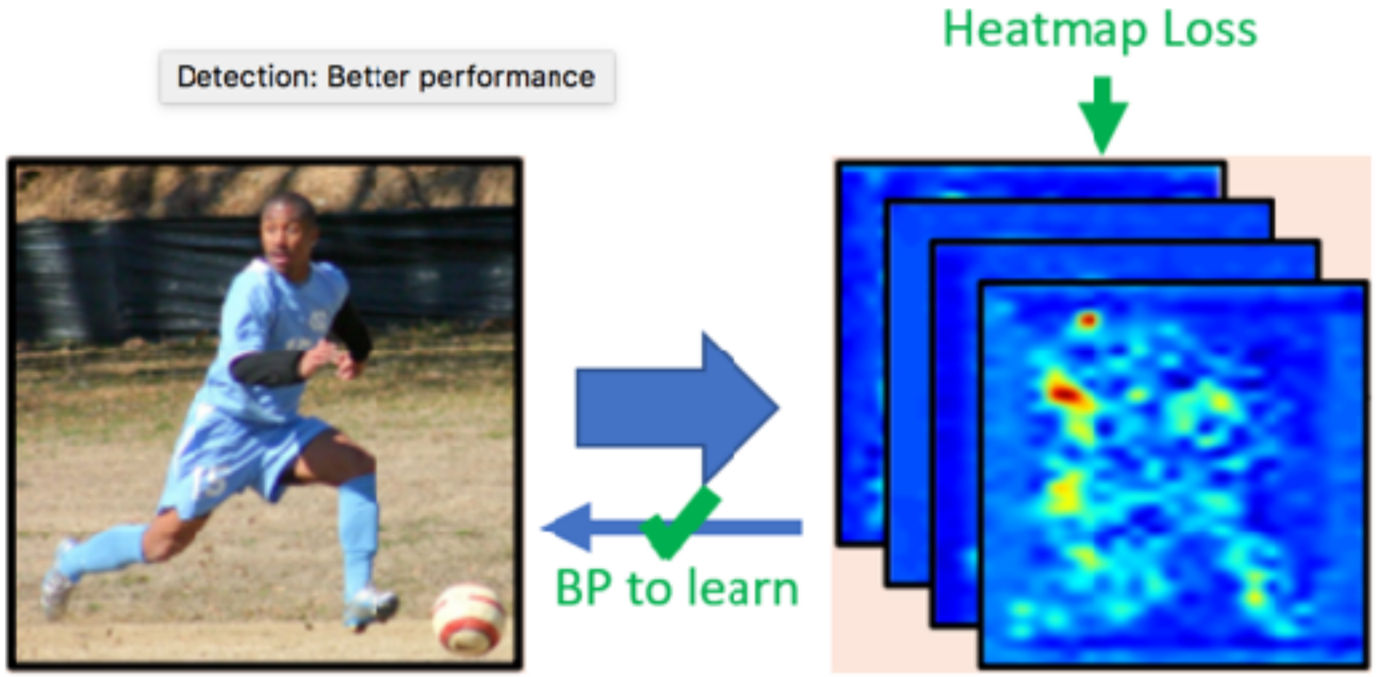


- estimate joint position as position of heatmap maximum



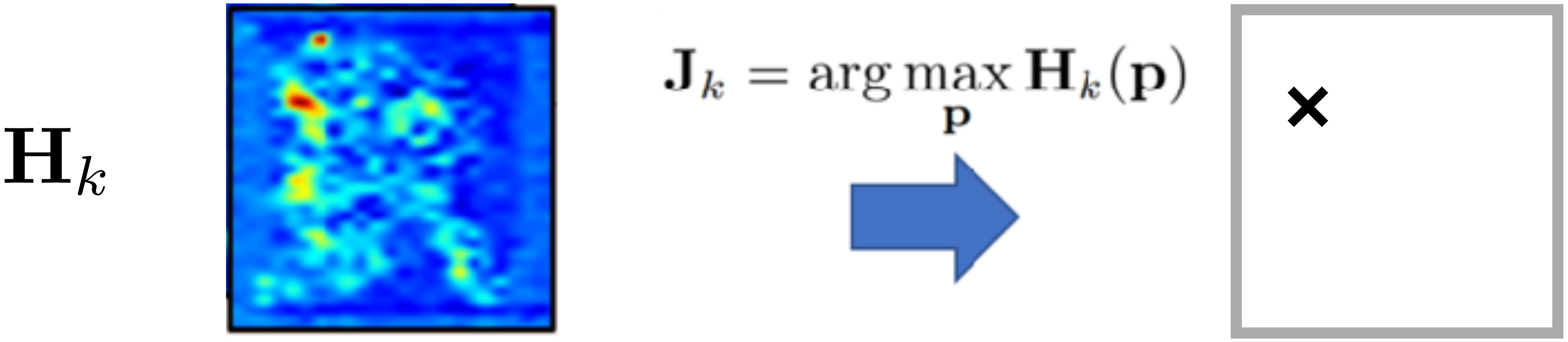
Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

Pose detection baseline



$\mathbf{H}_k$: Heatmap

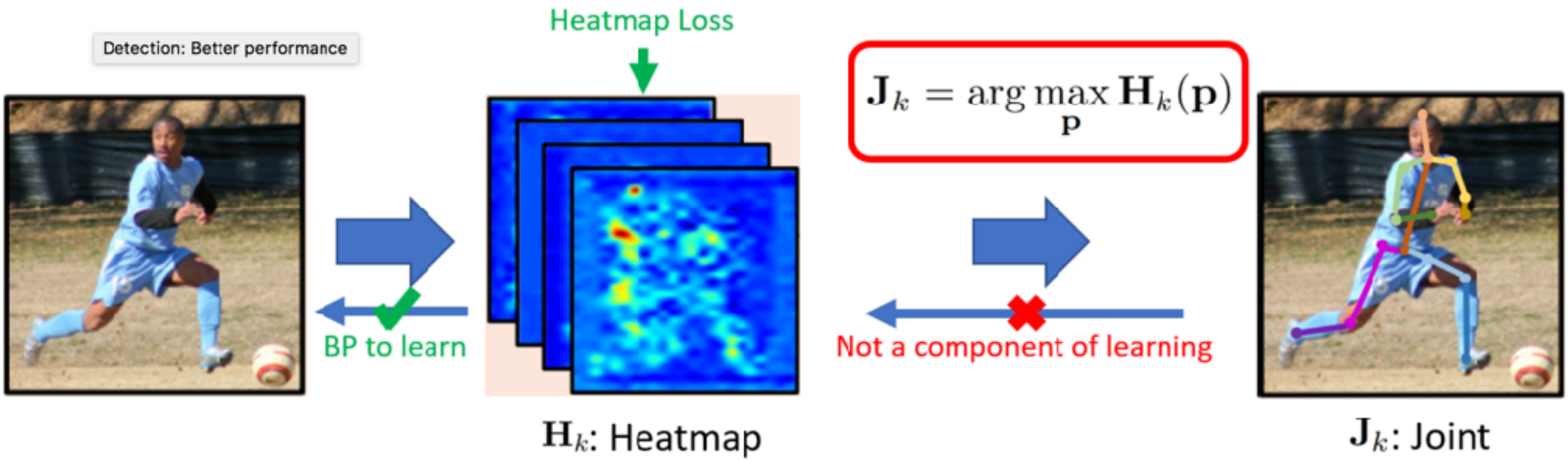
- estimate joint position as position of heatmap maximum



$$\mathbf{J}_k = \arg\max_{\mathbf{p}} \mathbf{H}_k(\mathbf{p})$$

Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

Pose detection baseline



Detection: Better performance

Heatmap Loss

BP to learn

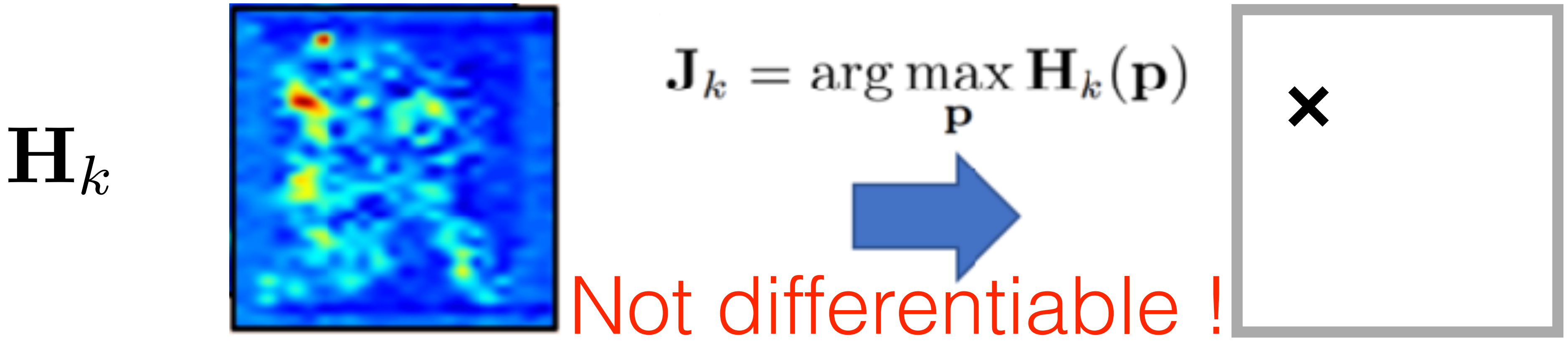
$$\mathbf{J}_k = \arg\max_{\mathbf{p}} \mathbf{H}_k(\mathbf{p})$$

Not a component of learning

$\mathbf{H}_k$: Heatmap

$\mathbf{J}_k$: Joint

- estimate joint position as position of heatmap maximum



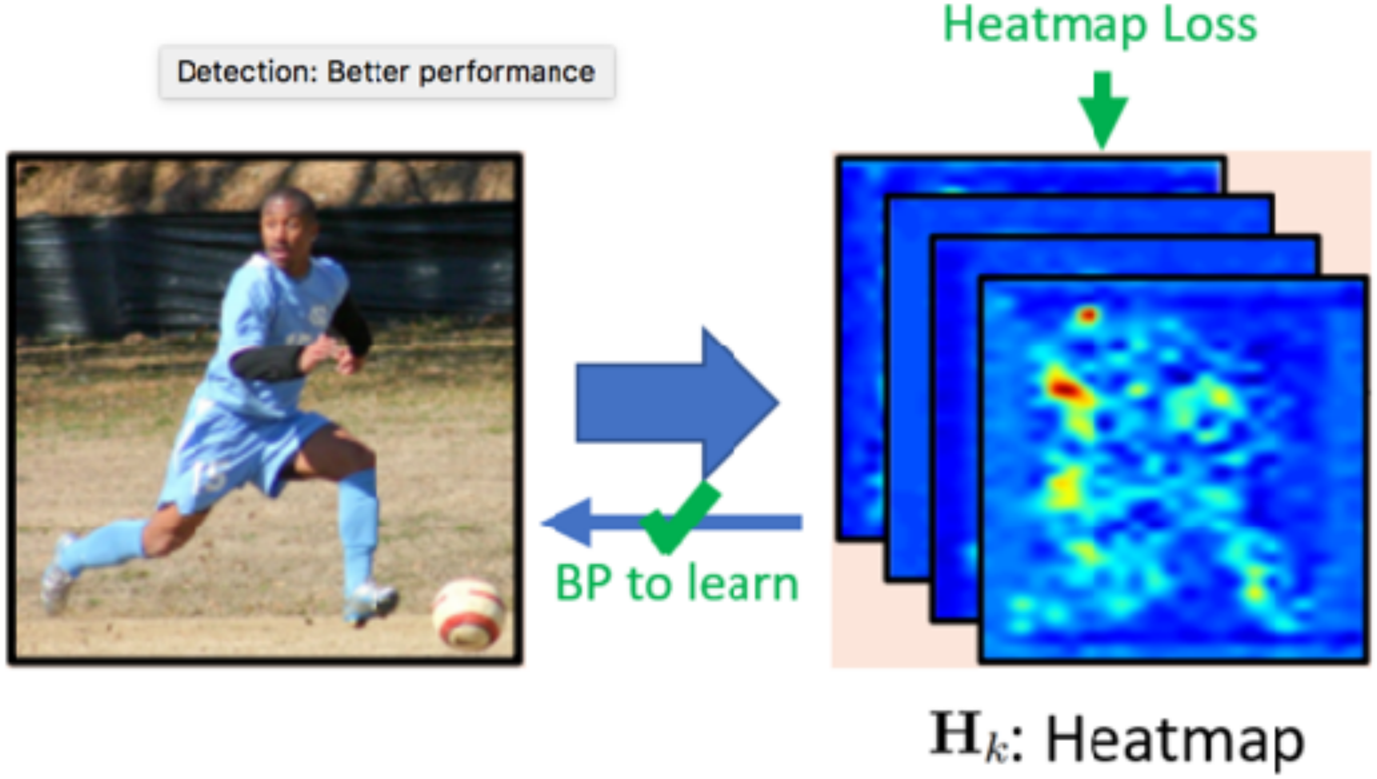
$\mathbf{H}_k$

$$\mathbf{J}_k = \arg\max_{\mathbf{p}} \mathbf{H}_k(\mathbf{p})$$

Not differentiable !

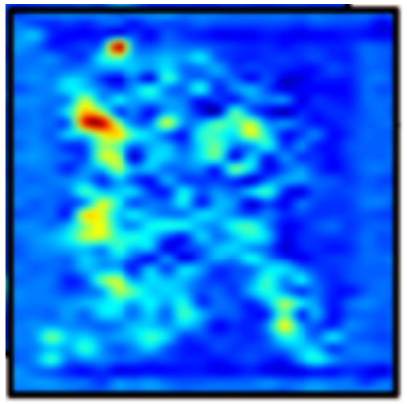
Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

Pose detection baseline



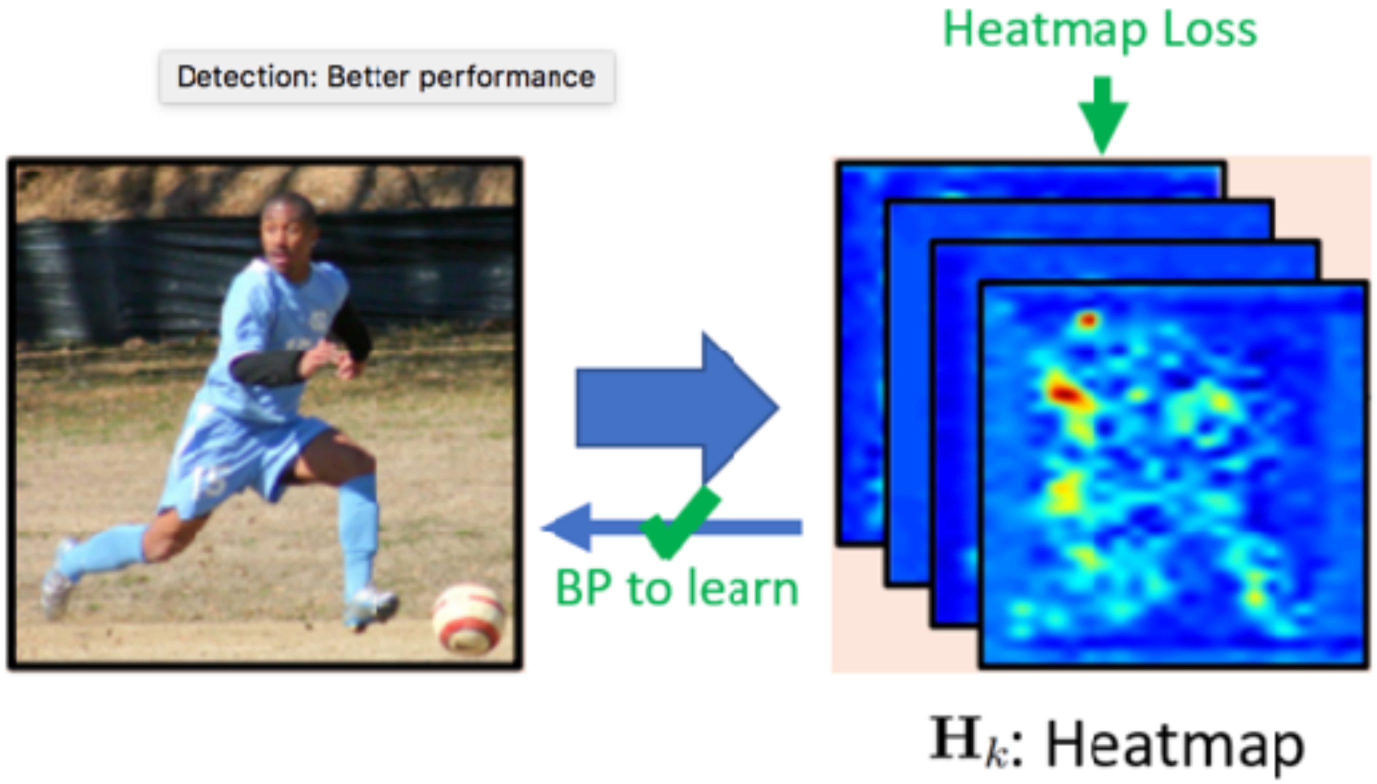
- estimate joint position as expected value in heatmap

$\mathbf{H}_k$

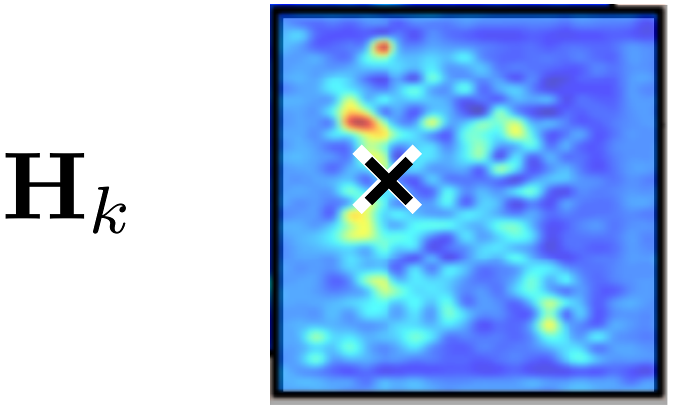


Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

Pose detection baseline



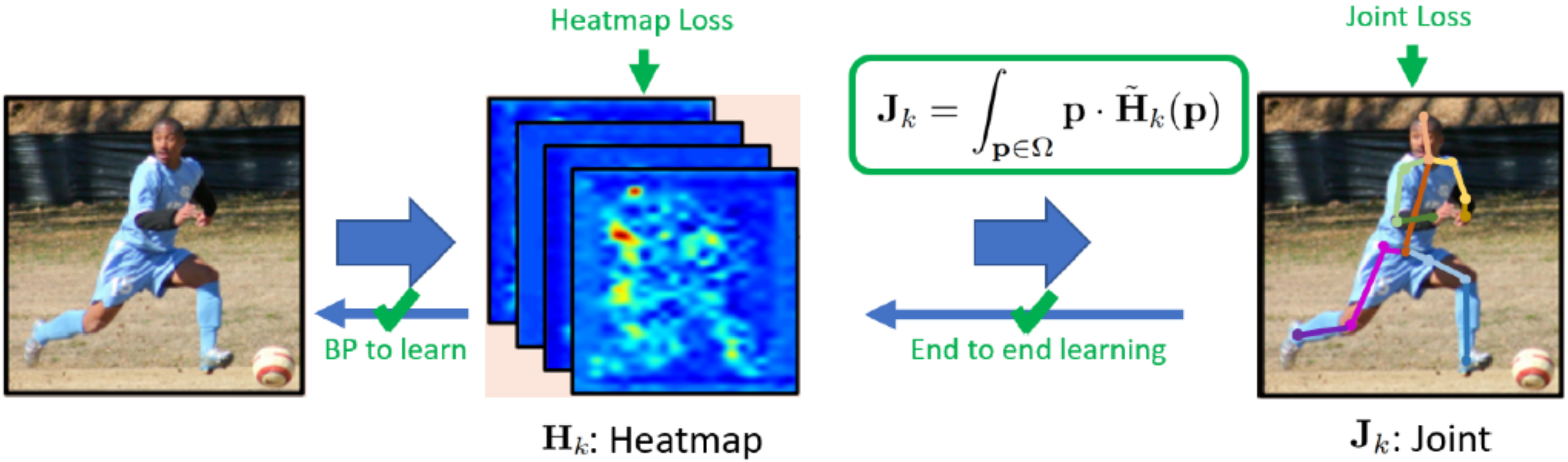
- estimate joint position as expected value in heatmap



$$\mathbf{J}_k = \int_{\mathbf{p}\in\Omega} \mathbf{p} \cdot \tilde{\mathbf{H}}_k(\mathbf{p})$$

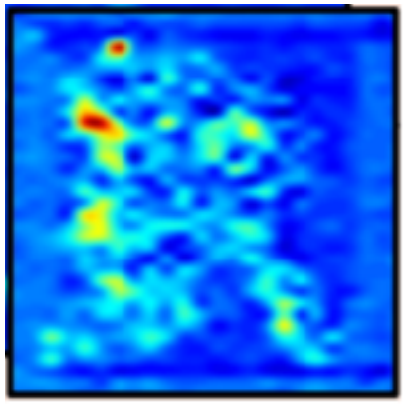
Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

Pose detection baseline

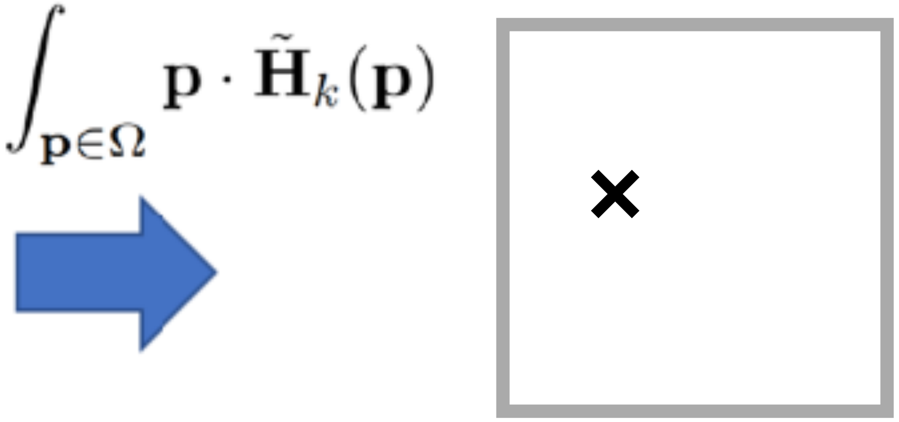


- estimate joint position as expected value in heatmap

$\mathbf{H}_k$

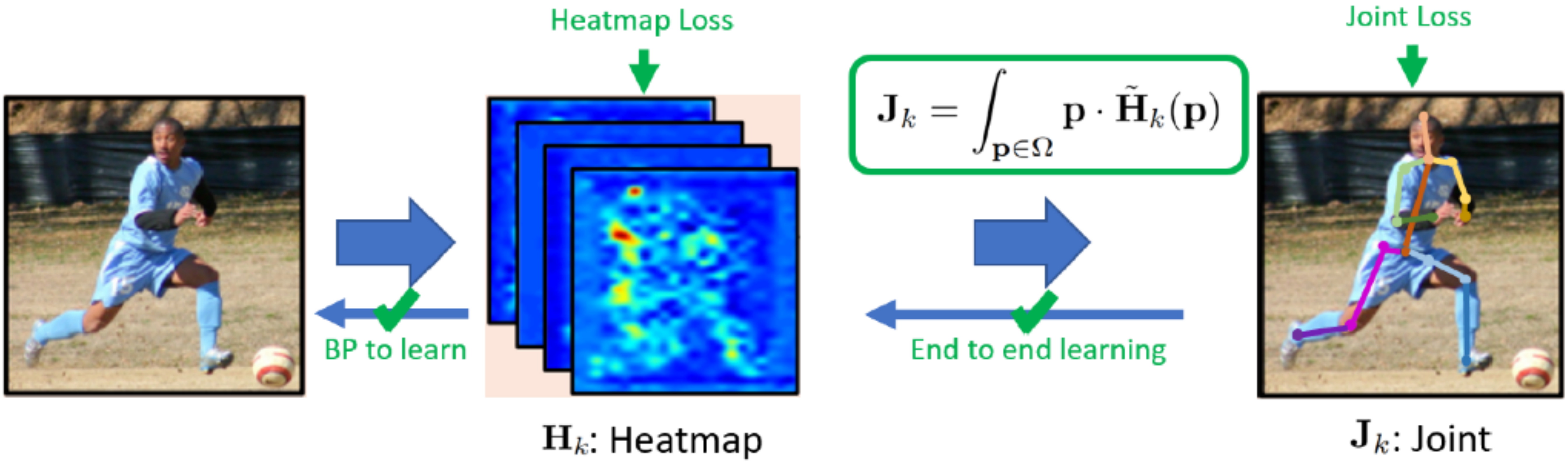


$$\mathbf{J}_k = \int_{\mathbf{p}\in\Omega} \mathbf{p} \cdot \tilde{\mathbf{H}}_k(\mathbf{p})$$



Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

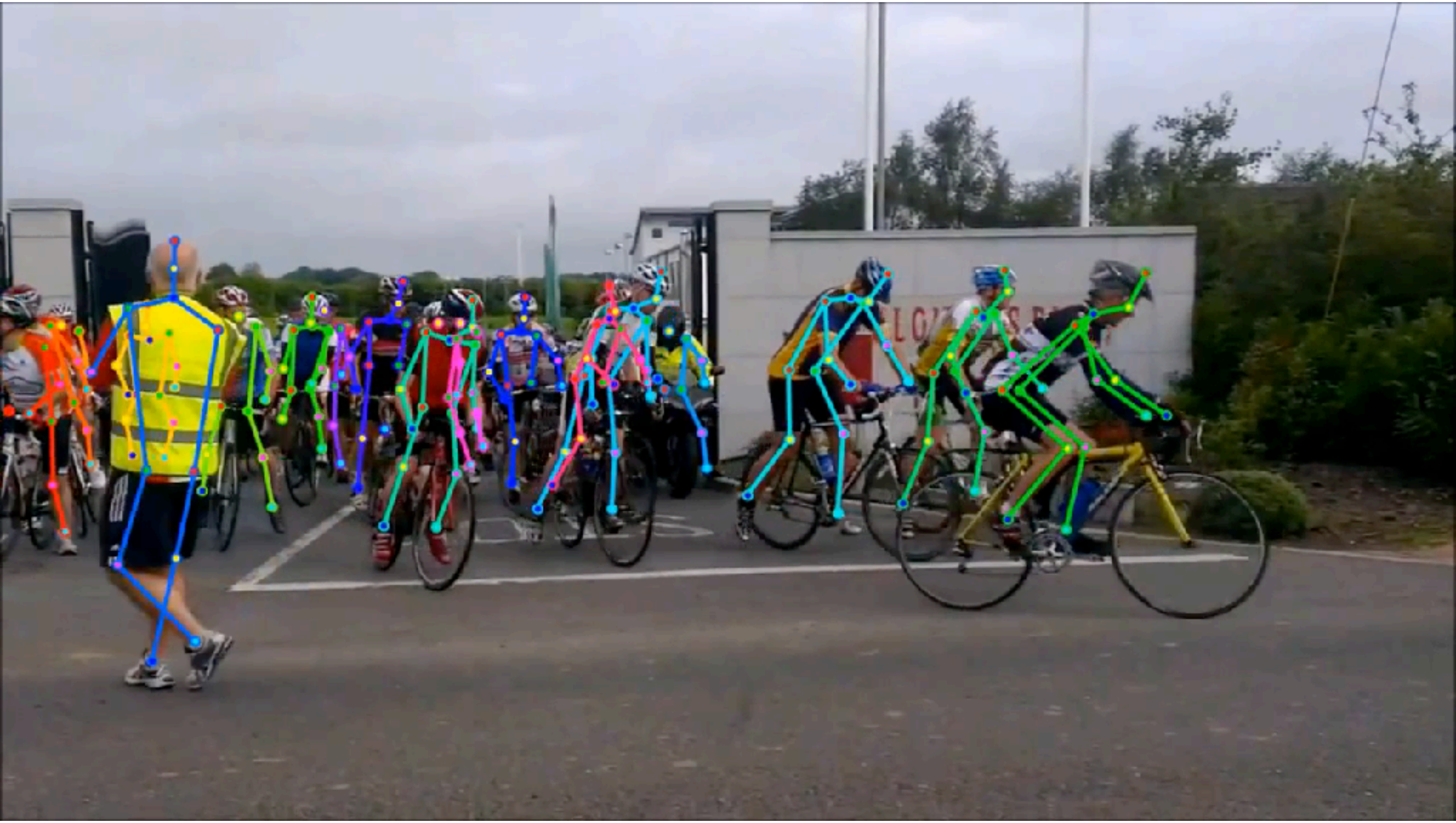
Pose detection+regression baseline



- ConvNet first estimates N joint's heat maps $\mathbf{H}_k,\ k=1\dots N$ (i.e. N 2D-images or N 3D-arrays)
- Learning minimizes segmentation loss over the N images
- Joints positions (p) estimated as expected value in $\mathbf{H}_k$

Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

PoseTrack challenge (ICCV 2017/ECCV 2018)
https://posetrack.net



Pose regression references

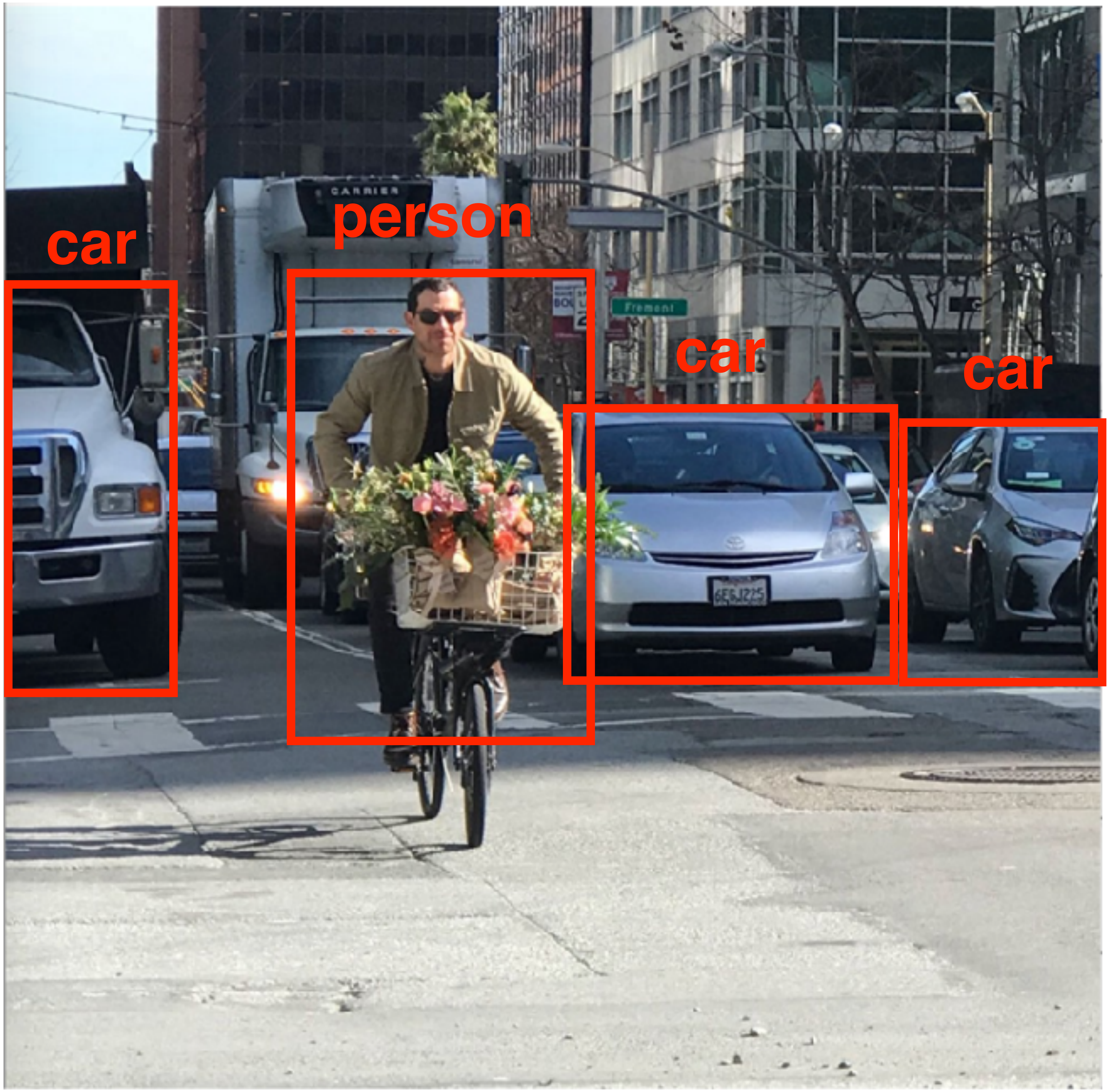
- PoseTrack benchmark a datasets
  https://posetrack.net
- Guler et al. (Facebook Research), DensePose
  https://arxiv.org/abs/1802.00434
  https://github.com/facebookresearch/Densepose
  https://www.youtube.com/watch?v=EMjPqgLX14A&feature=youtu.be
- Realtime Multi-Person 2D Human Pose Estimation using Par
  Affinity Fields, CVPR 2017 Oral
  https://www.youtube.com/watch?v=pW6nZXeWlGM
- Integral Human Pose Regression [Sun ECCV 2018]
  Microsoft Research
  https://arxiv.org/abs/1711.08229
  https://github.com/JimmySuen/integral-human-pose

Outline

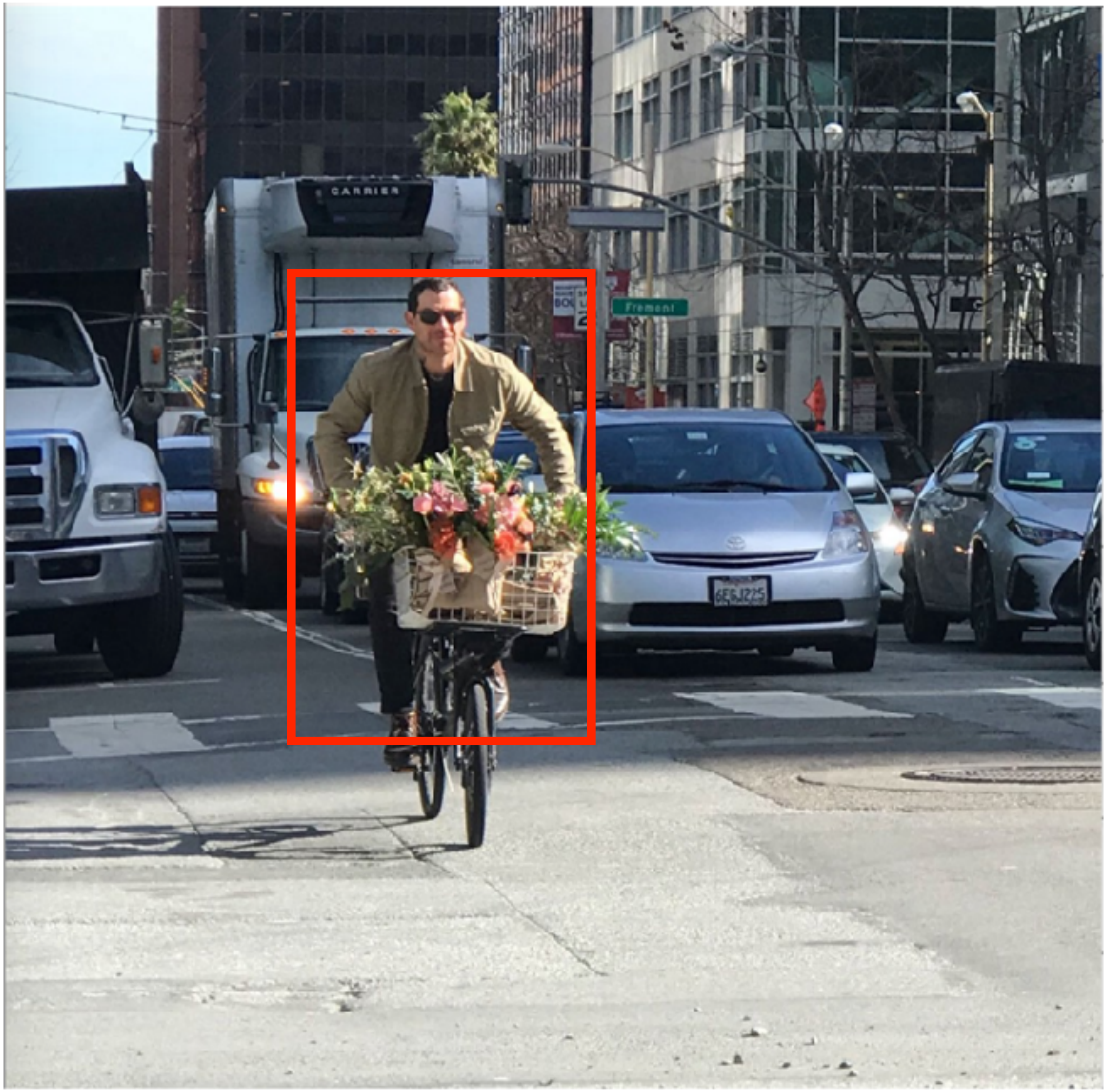
- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Architectures of feature matching networks

Object detection

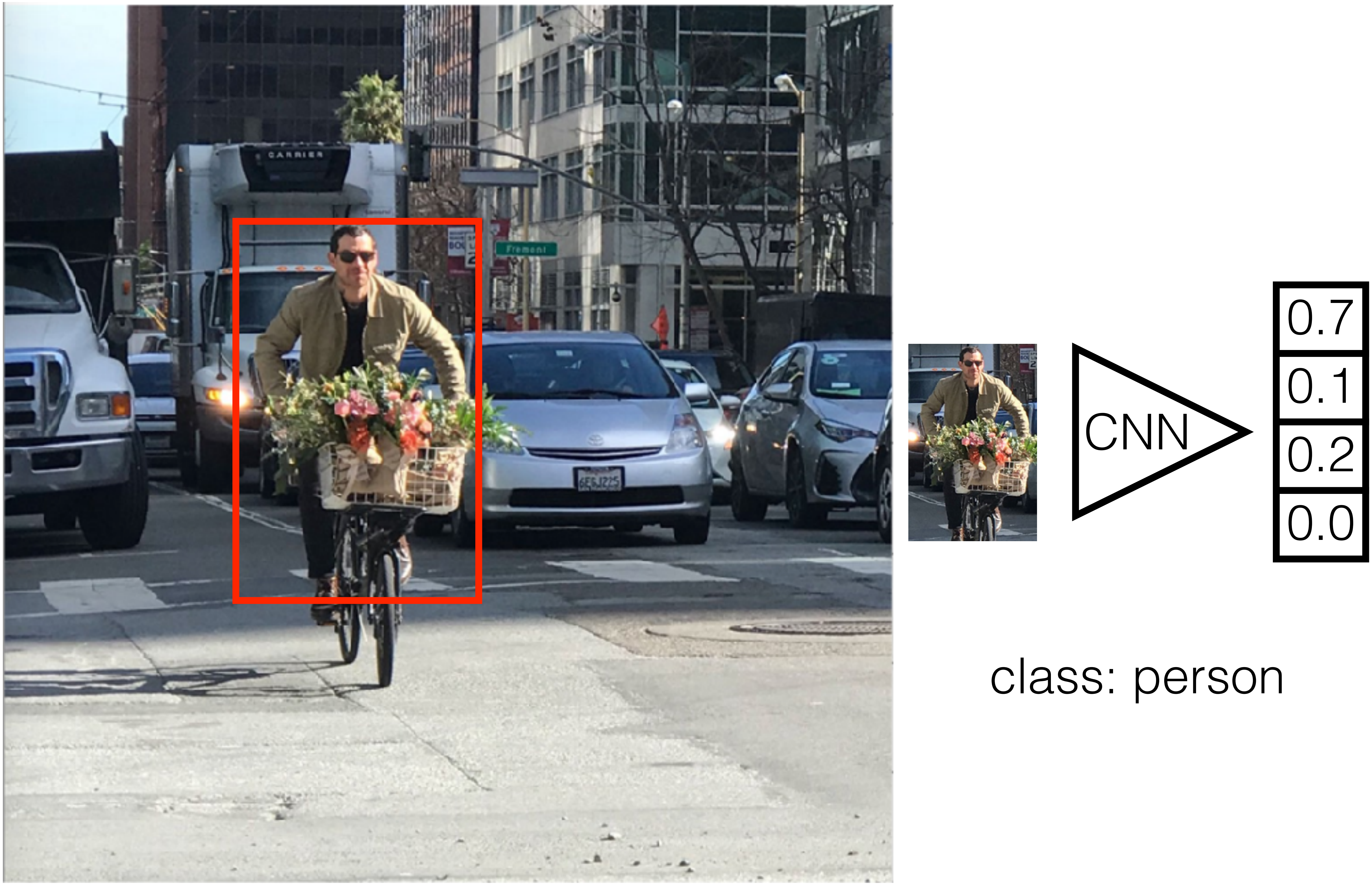
Object detection



Object detection

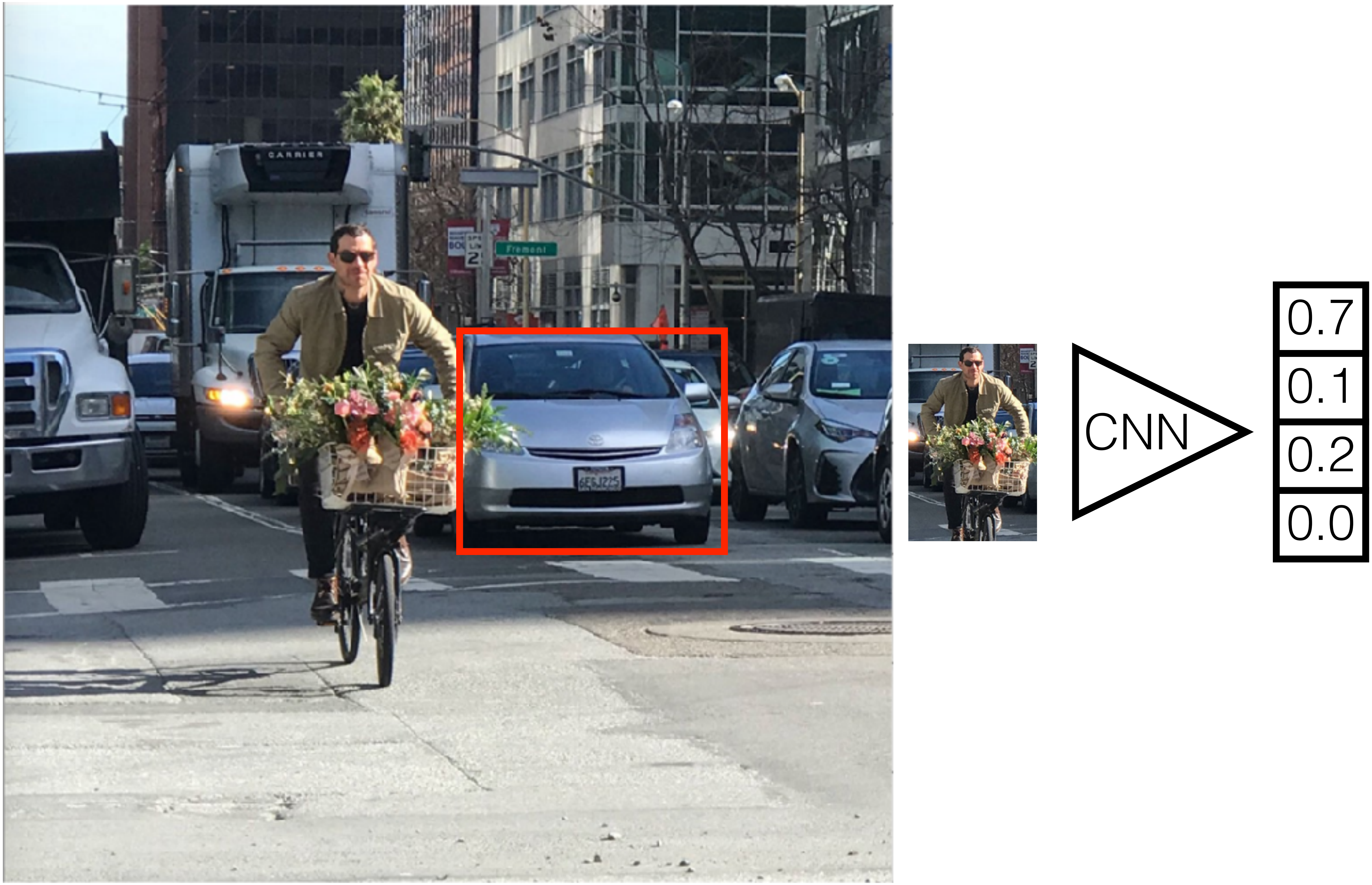


Object detection

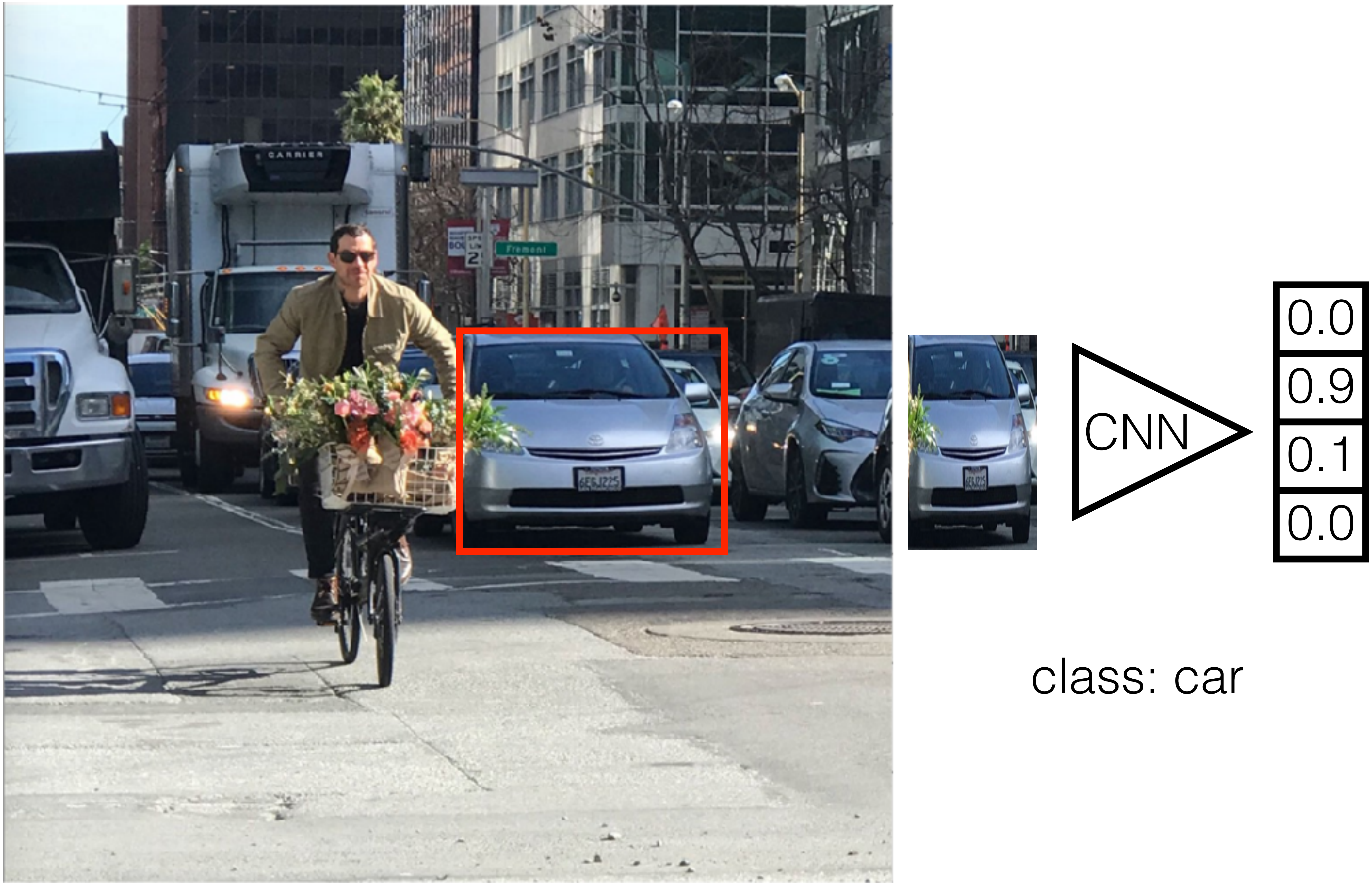


class: person

Object detection

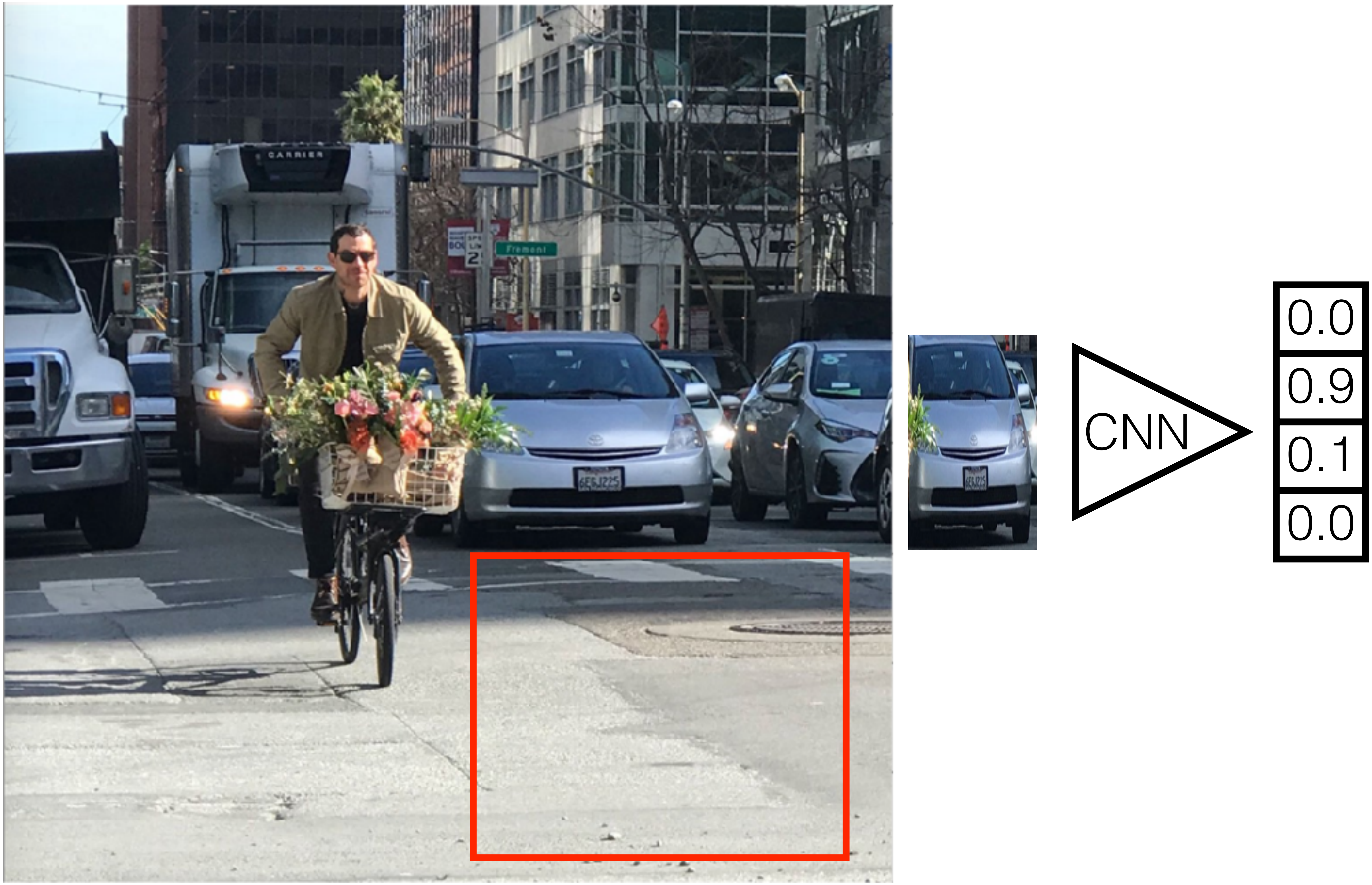


Object detection

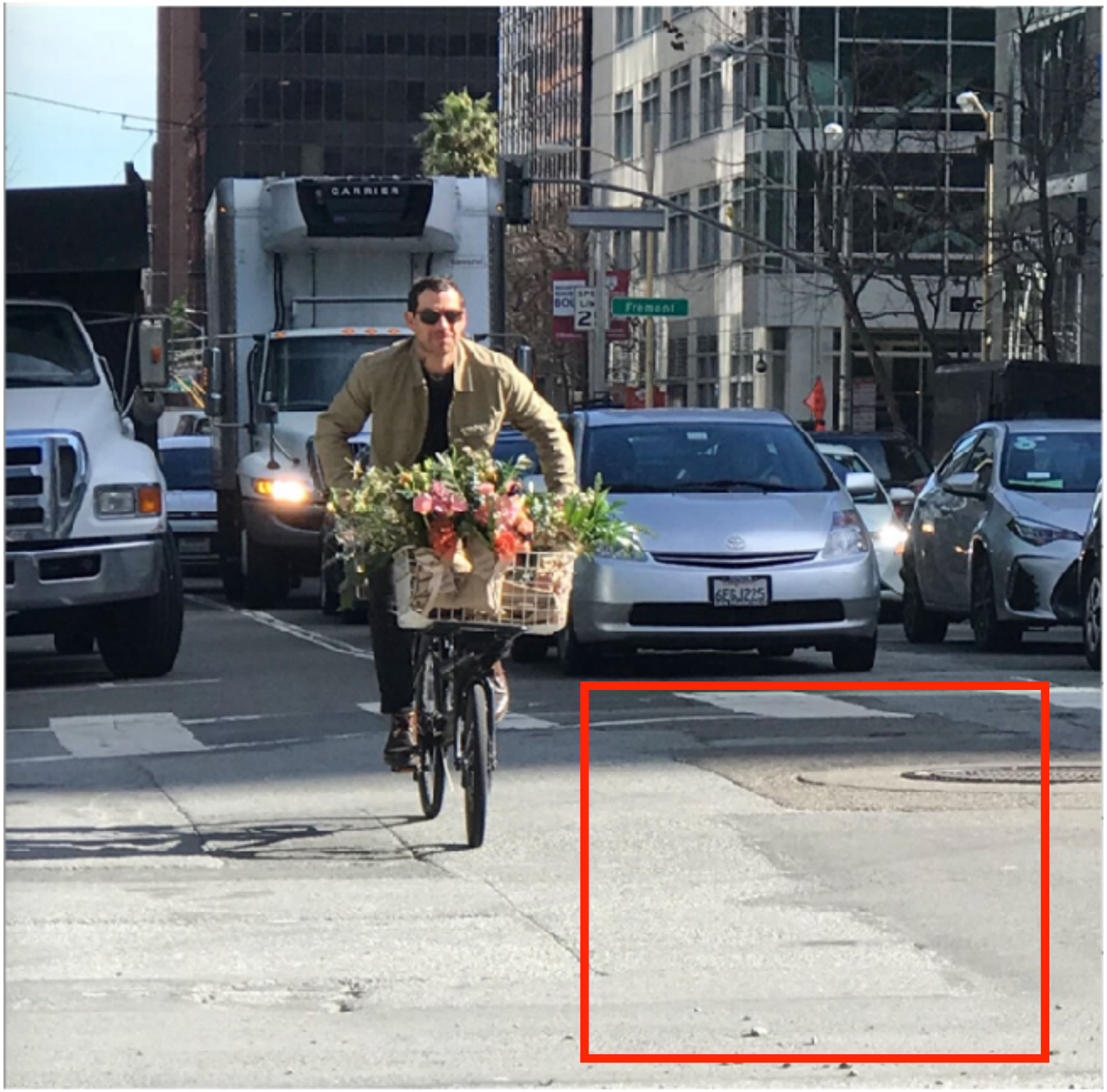


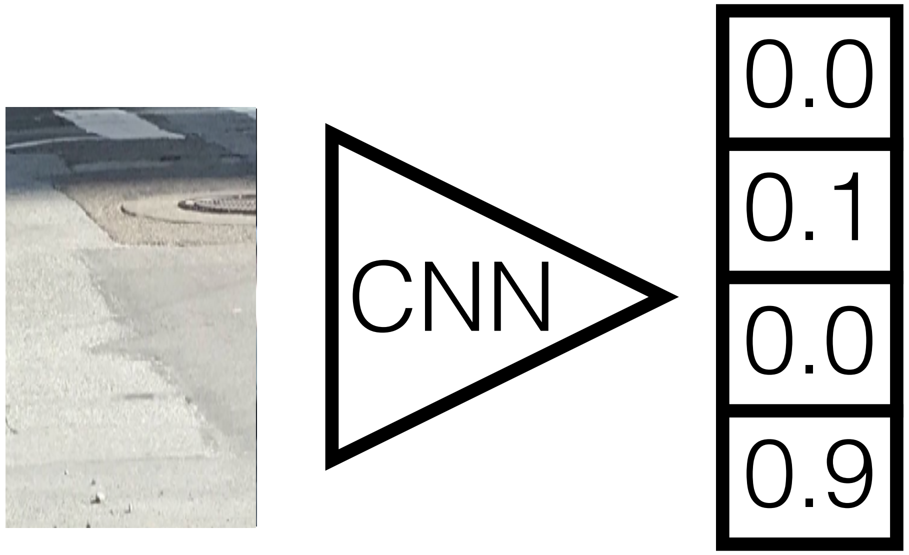
class: car

Object detection



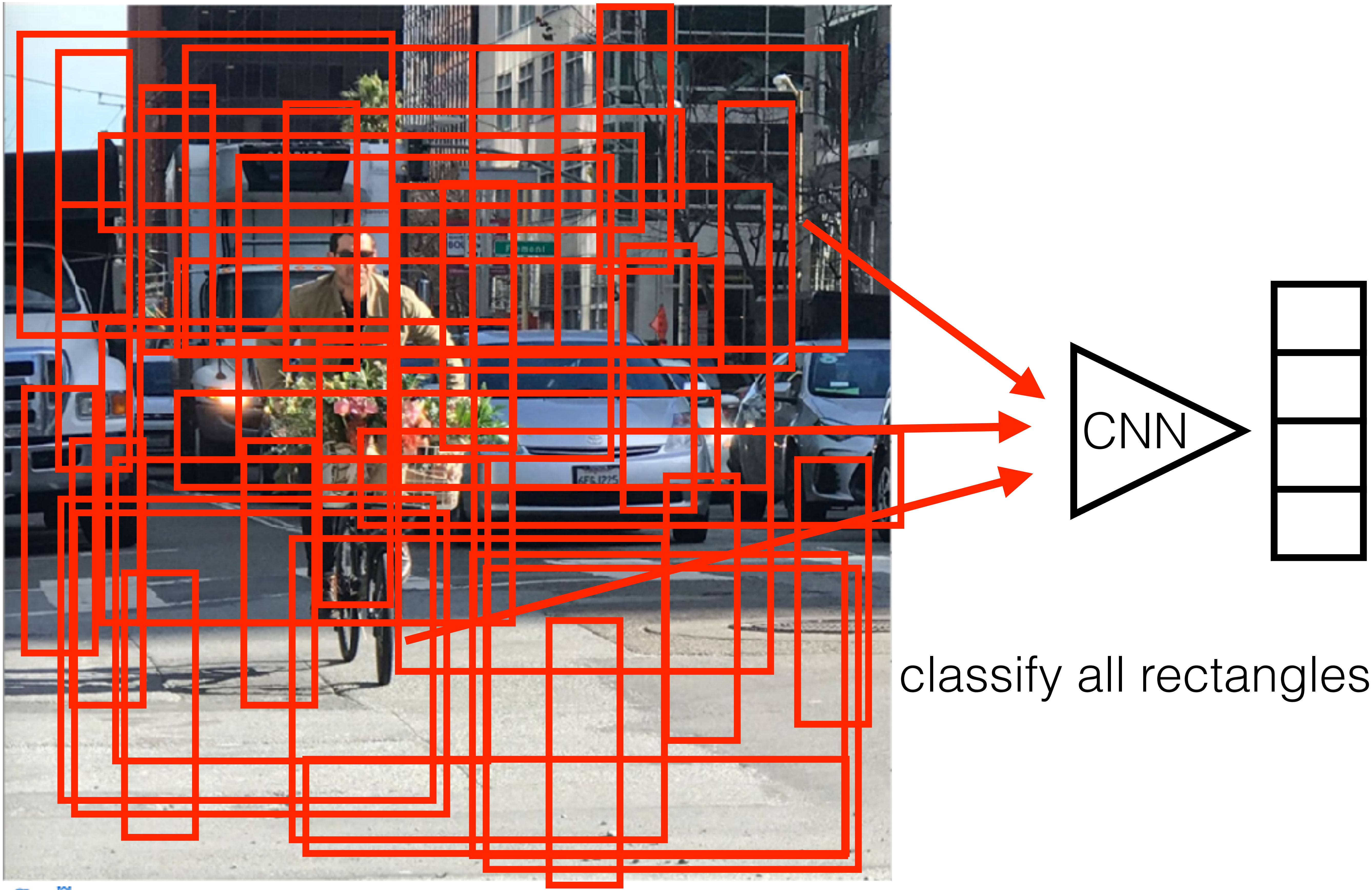
Object detection





class: background

Object detection

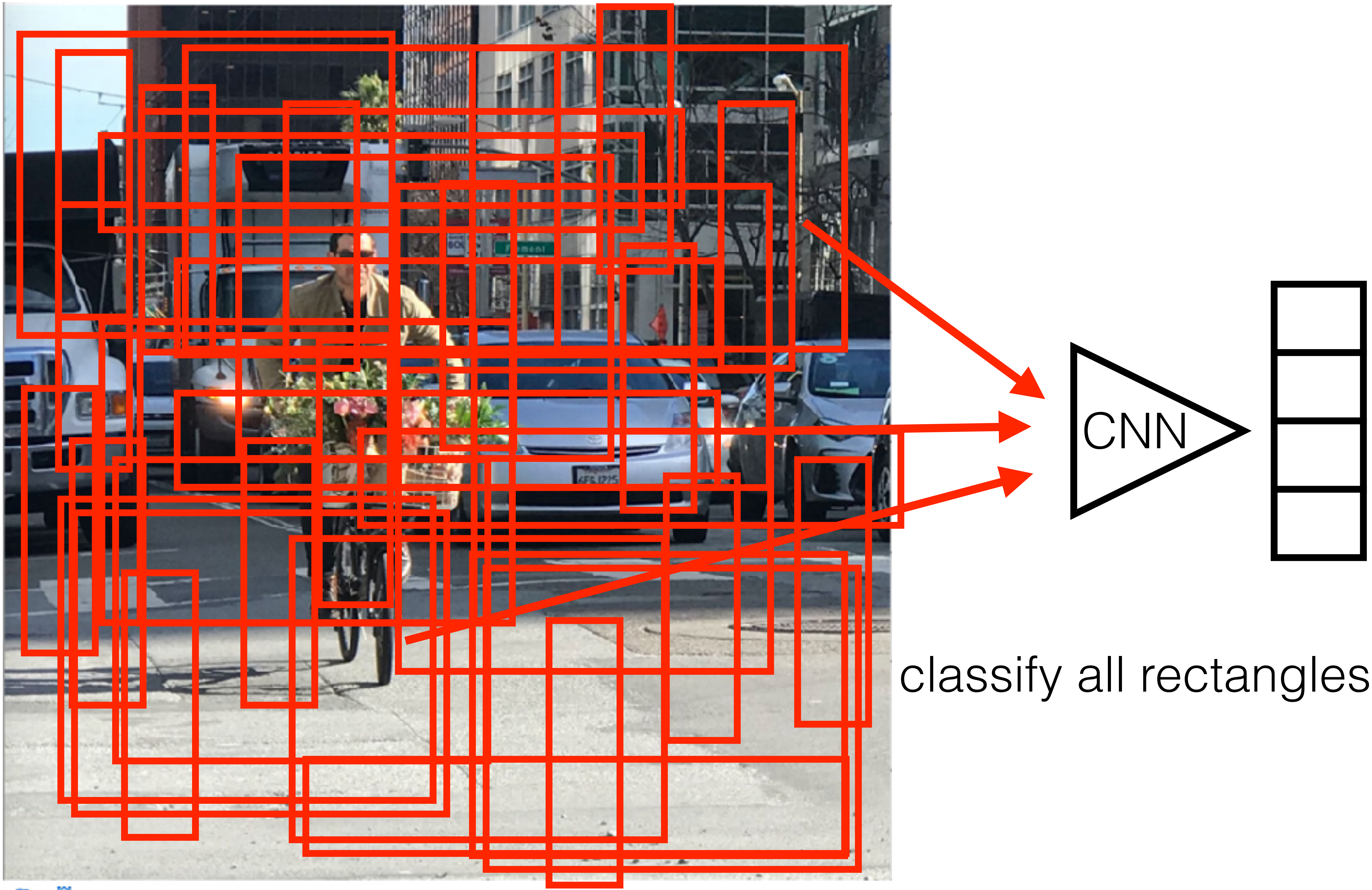


classify all rectangles

Object detection

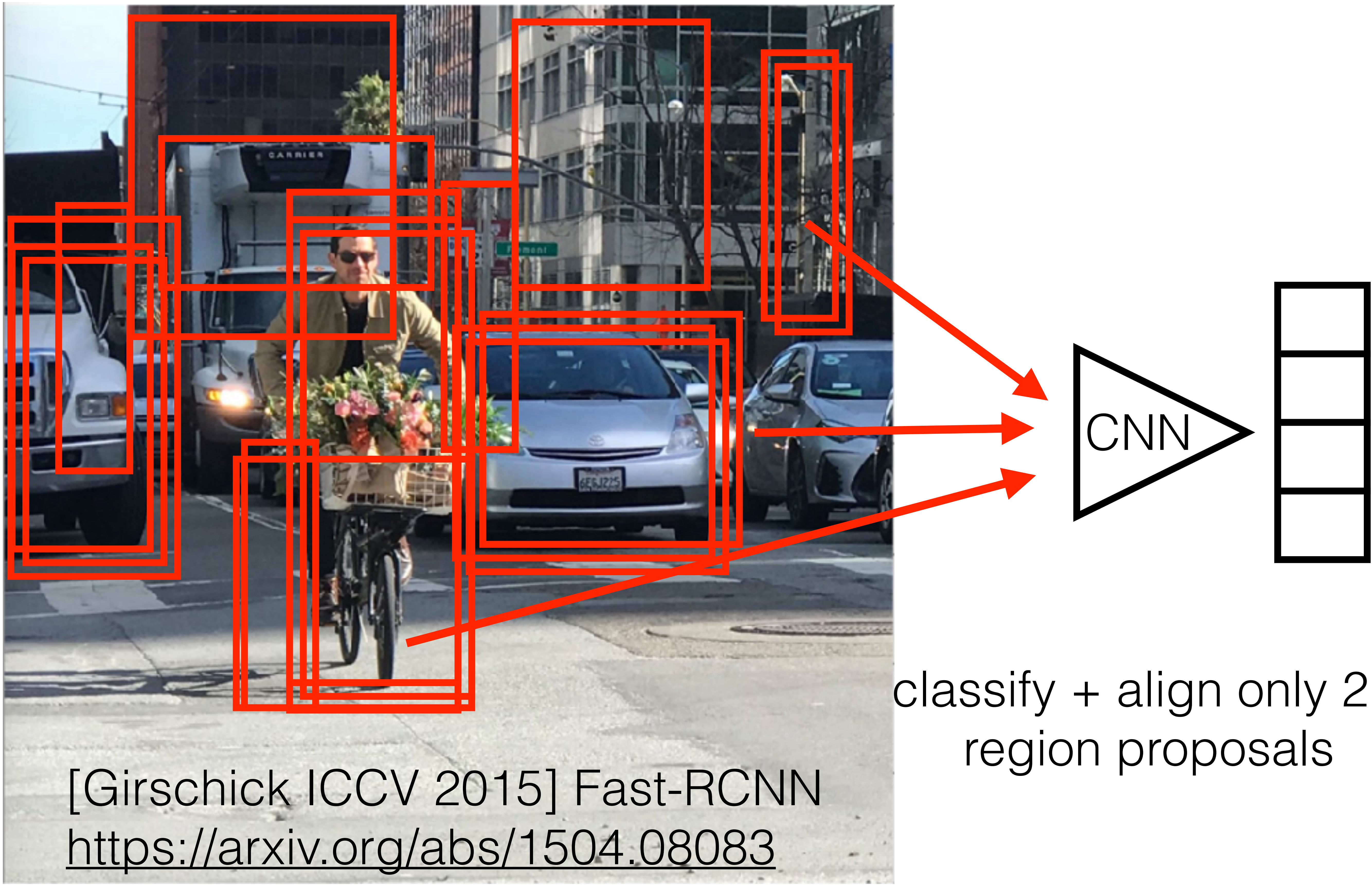
- Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  H x W x Aspect_Ratio x Scales x 0.001 sec = **months**

Object detection



classify all rectangles

Object detection



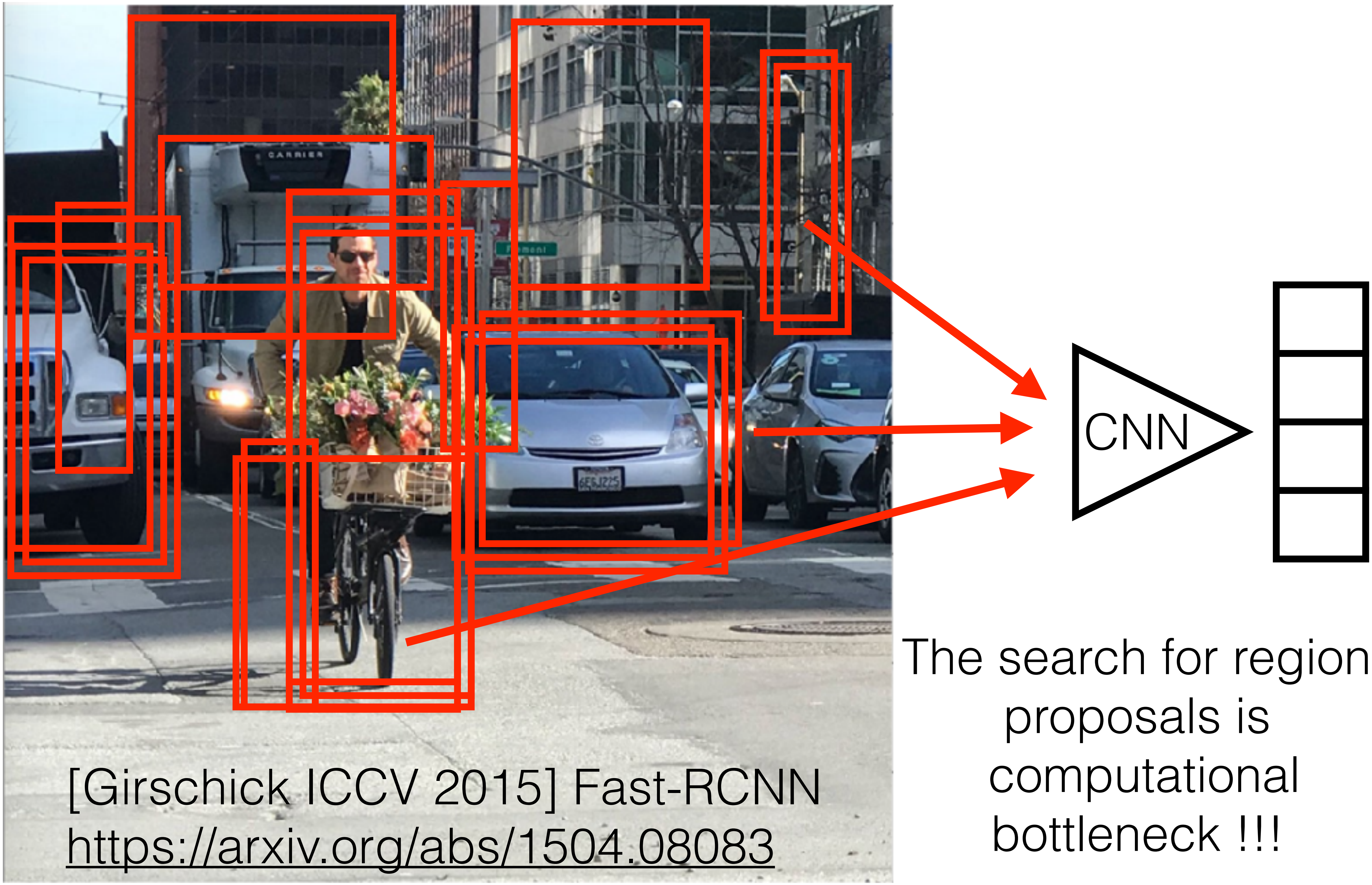
classify + align only 2k
region proposals

[Girschick ICCV 2015] Fast-RCNN
https://arxiv.org/abs/1504.08083

Object detection

- Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  H x W x Aspect_Ratio x Scales x 0.001 sec = **months**
- Instead we can use elementary signal processing method to extract only 2k viable candidates:
  [Girschick ICCV 2015], Fast-RCNN
  https://arxiv.org/abs/1504.08083
  (find 2k cand.) + (2k cand. x 0.001 sec) = **47+2 sec = 49 sec**

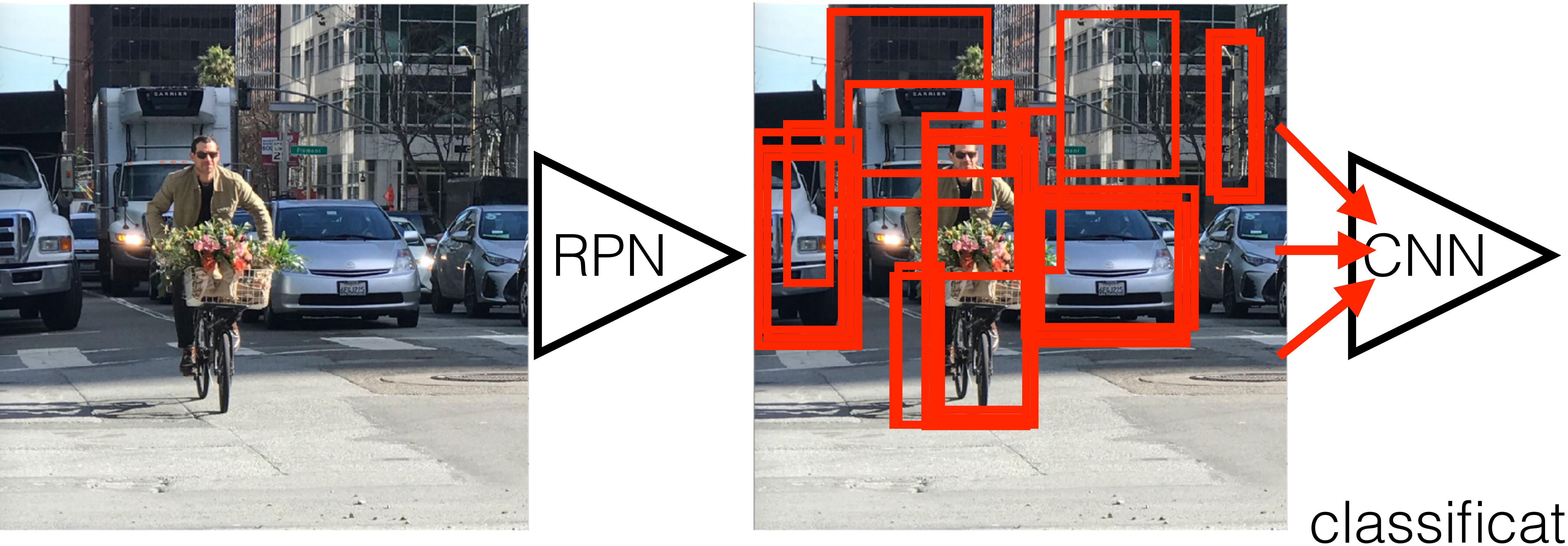
Object detection



[Girschick ICCV 2015] Fast-RCNN
https://arxiv.org/abs/1504.08083

The search for region proposals is computational bottleneck !!!

Object detection

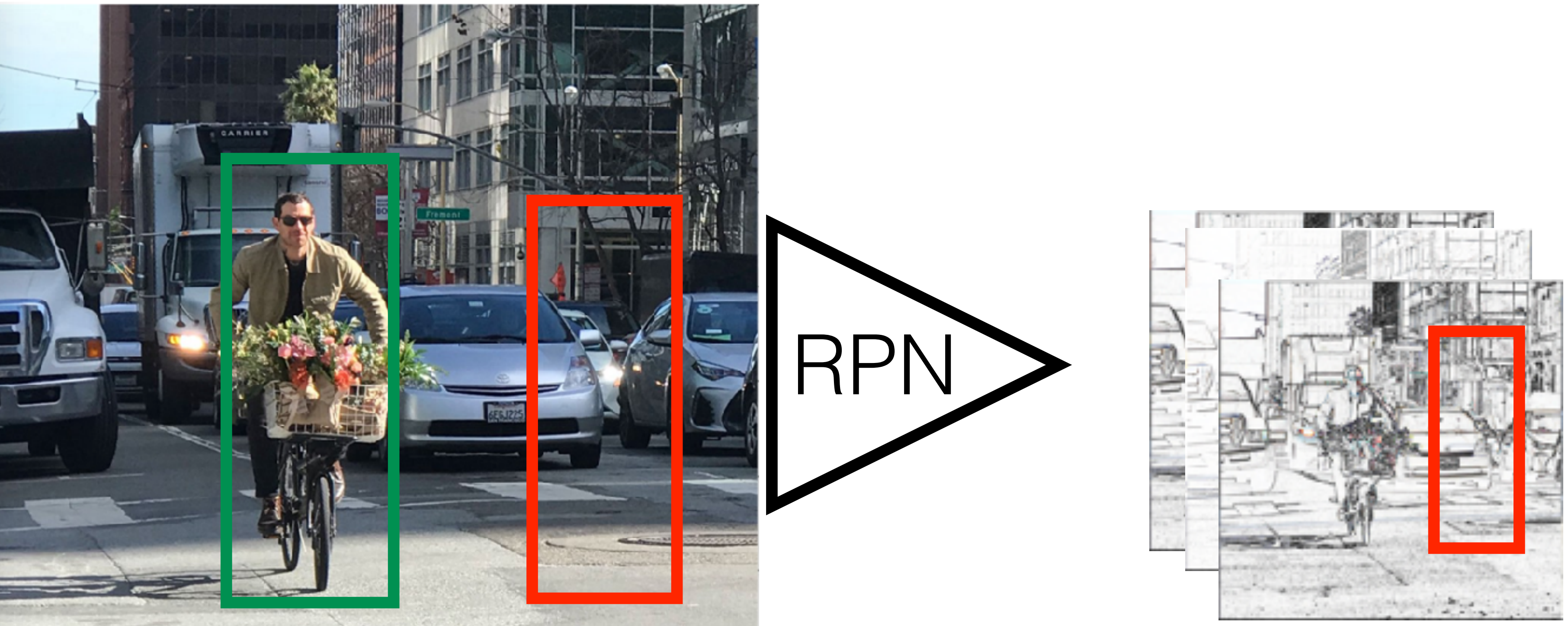


region proposal net
(output: 2k proposals)

classification
+
alignment
net

Faster-RCNN https://arxiv.org/abs/1506.01497

Region Proposal Net (RPN)



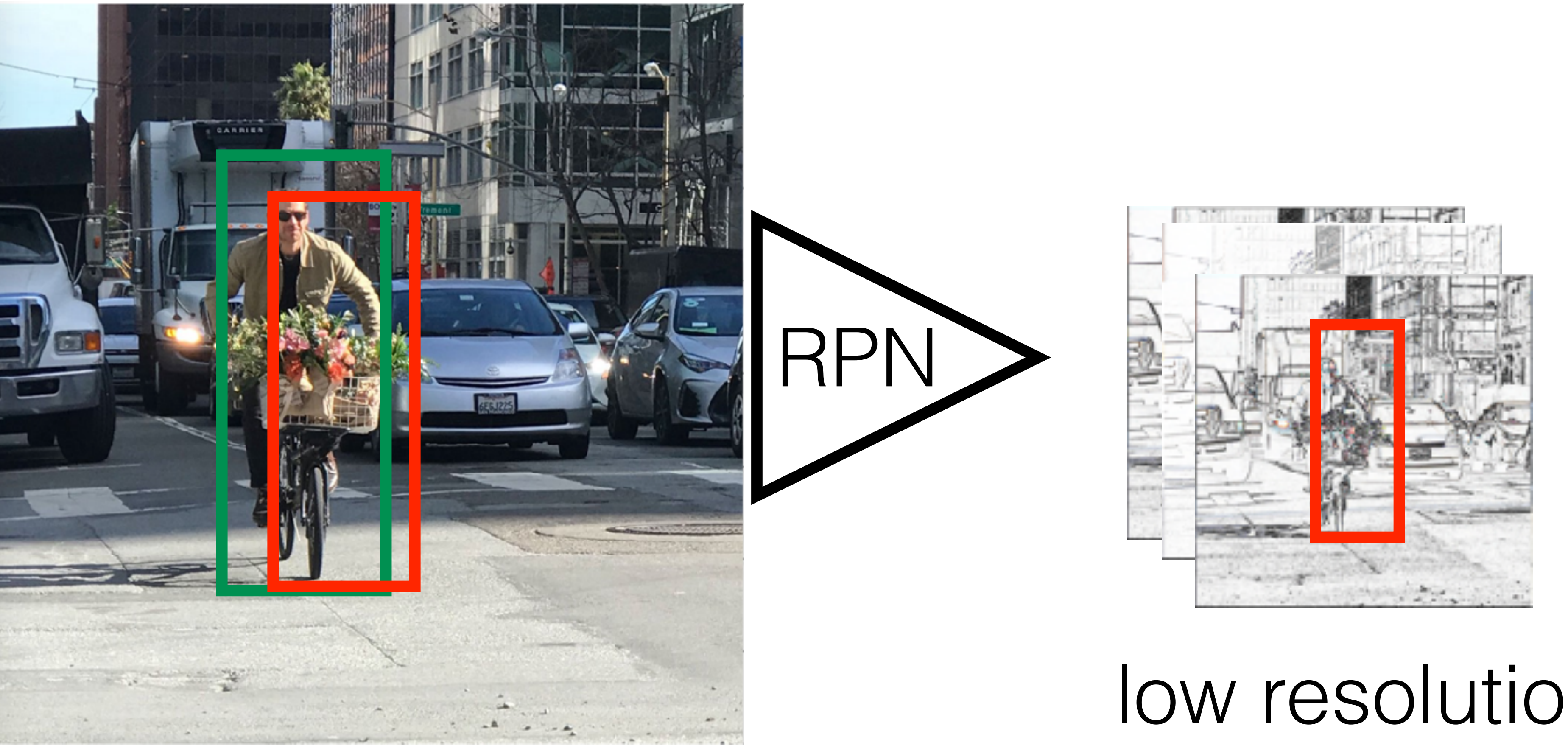
ground truth

low resolution
feature map

- generate bounding which corresponds to discrete positions in low resolution feature maps and measure IoU

Faster-RCNN https://arxiv.org/abs/1506.01497

Region Proposal Net (RPN)



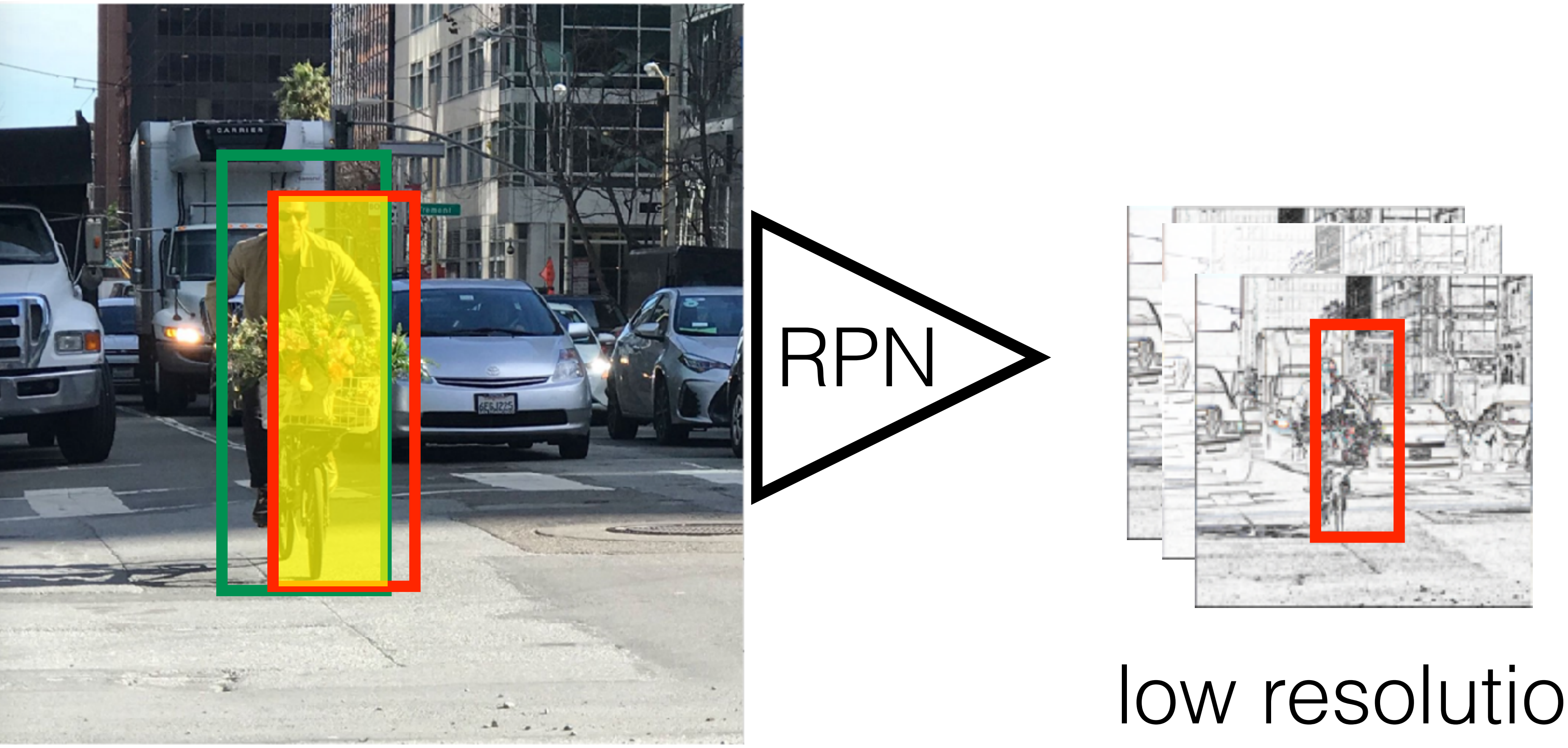
ground truth

low resolution feature map

- generate bounding which corresponds to discrete positions in low resolution feature maps and measure IoU

Faster-RCNN https://arxiv.org/abs/1506.01497

Region Proposal Net (RPN)



RPN

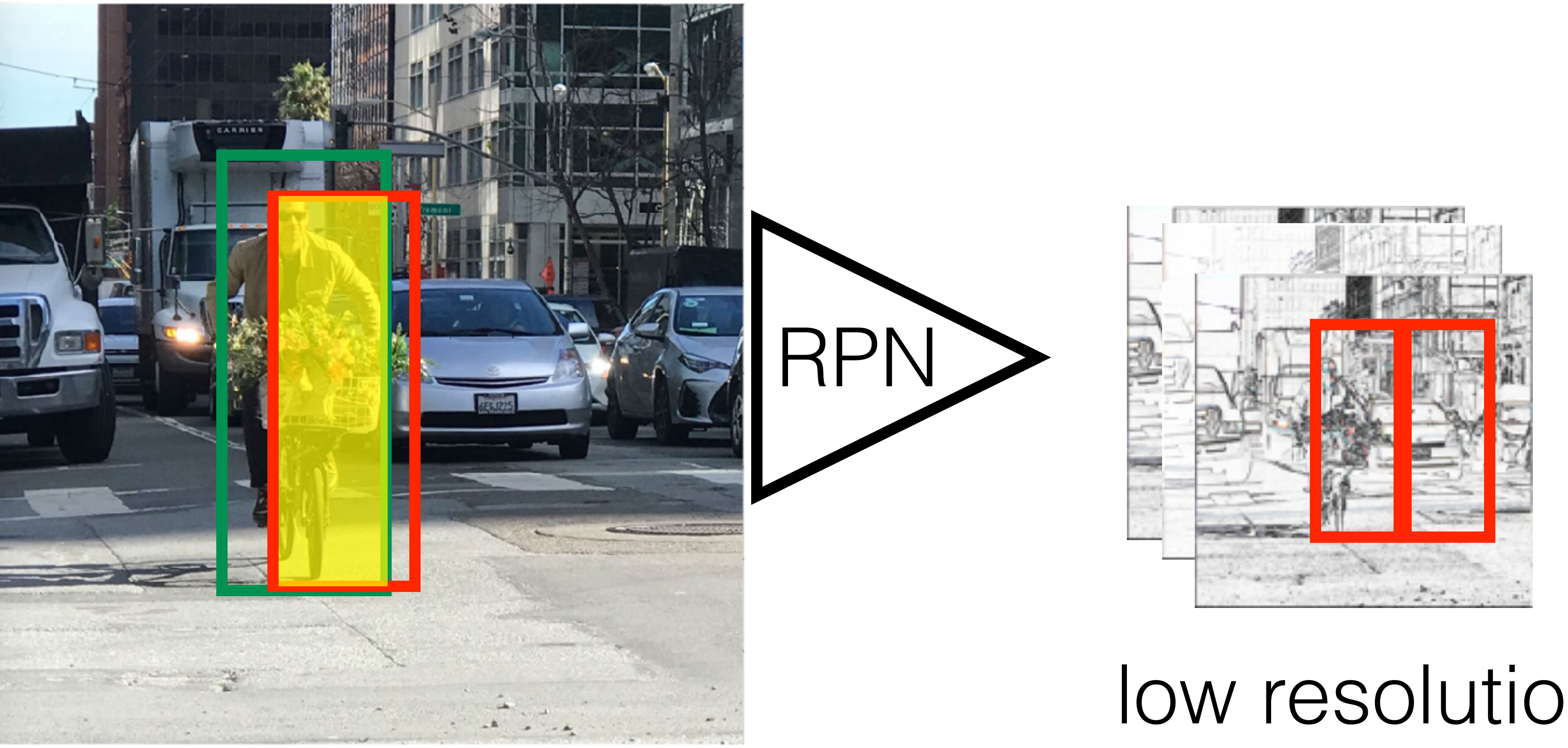
low resolution feature map

ground truth

- generate bounding which corresponds to discrete positions in low resolution feature maps and measure IoU

Faster-RCNN https://arxiv.org/abs/1506.01497

Region Proposal Net (RPN)



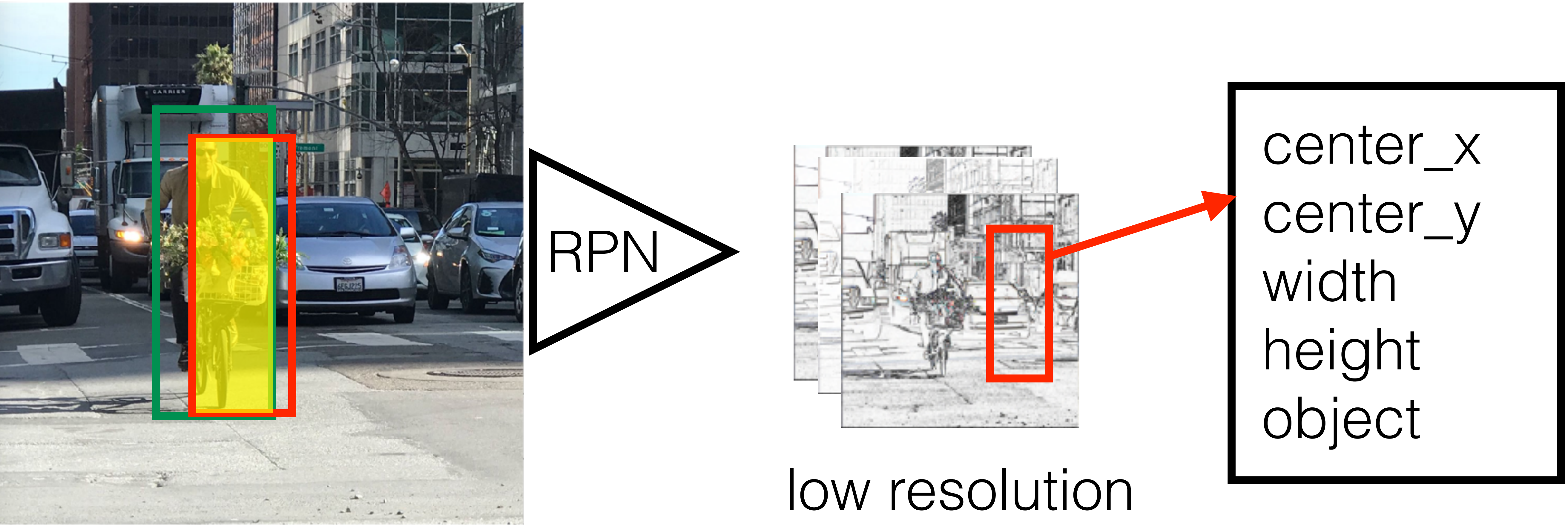
ground truth

low resolution feature map

- generate bounding which corresponds to discrete positions in low resolution feature maps and measure IoU
- bbs with IoU>0.7 are objects,
  bbs with IoU<0.3 not objects

Faster-RCNN https://arxiv.org/abs/1506.01497

Region Proposal Net (RPN)



center_x
center_y
width
height
object

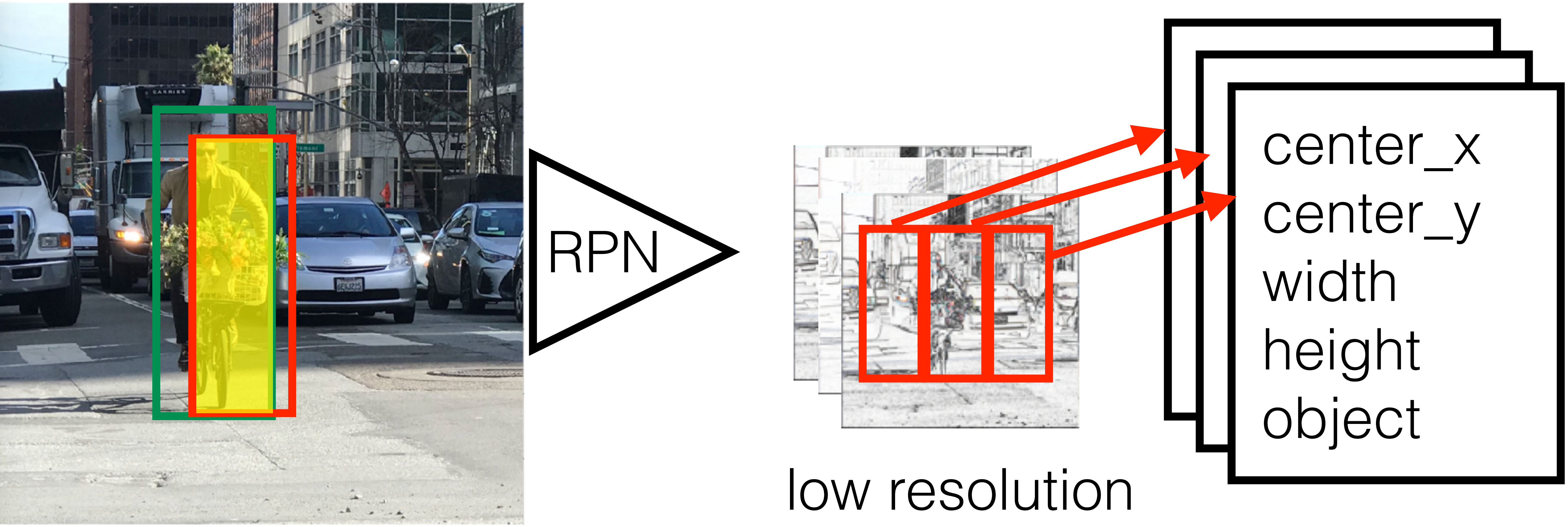
ground truth

low resolution
feature map

- for each discrete bb RPN predicts:
  - its “alignment with gt” (regression loss)
  - its “objectness" (classification loss)

Faster-RCNN https://arxiv.org/abs/1506.01497

Region Proposal Net (RPN)



RPN

center_x
center_y
width
height
object

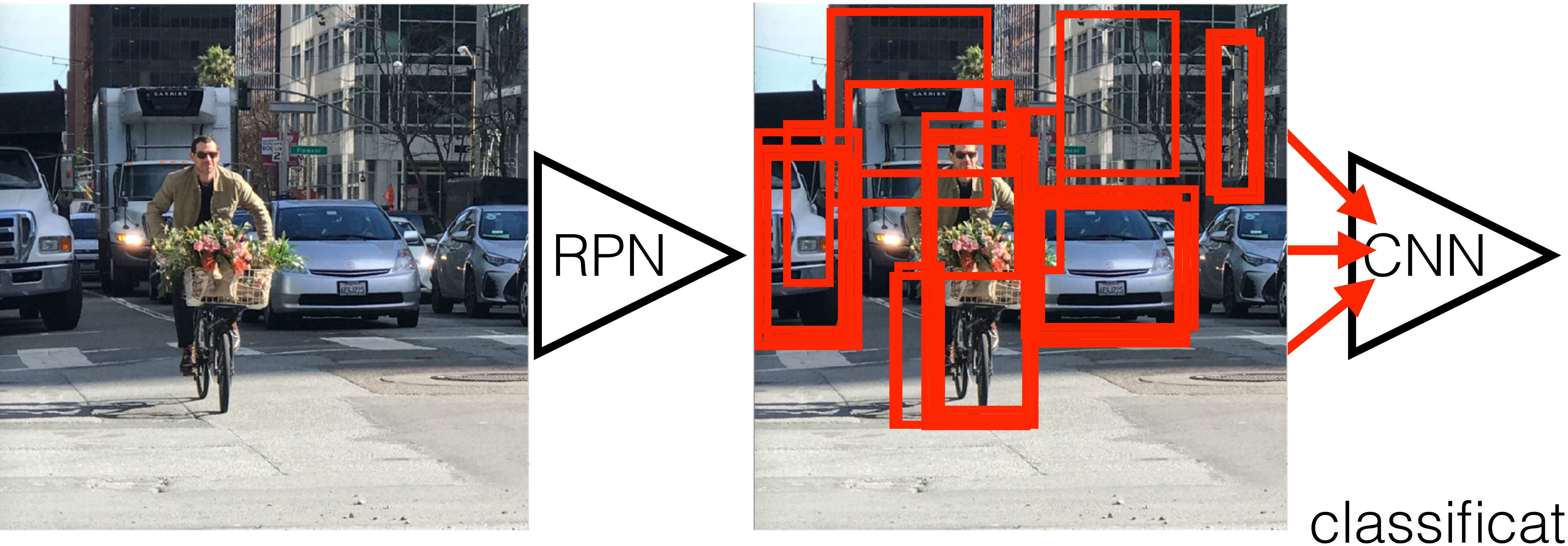
ground truth

low resolution feature map

- for each discrete bb RPN predicts:
  - its “alignment with gt” (regression loss)
  - its “objectness" (classification loss)

Faster-RCNN https://arxiv.org/abs/1506.01497

Object detection



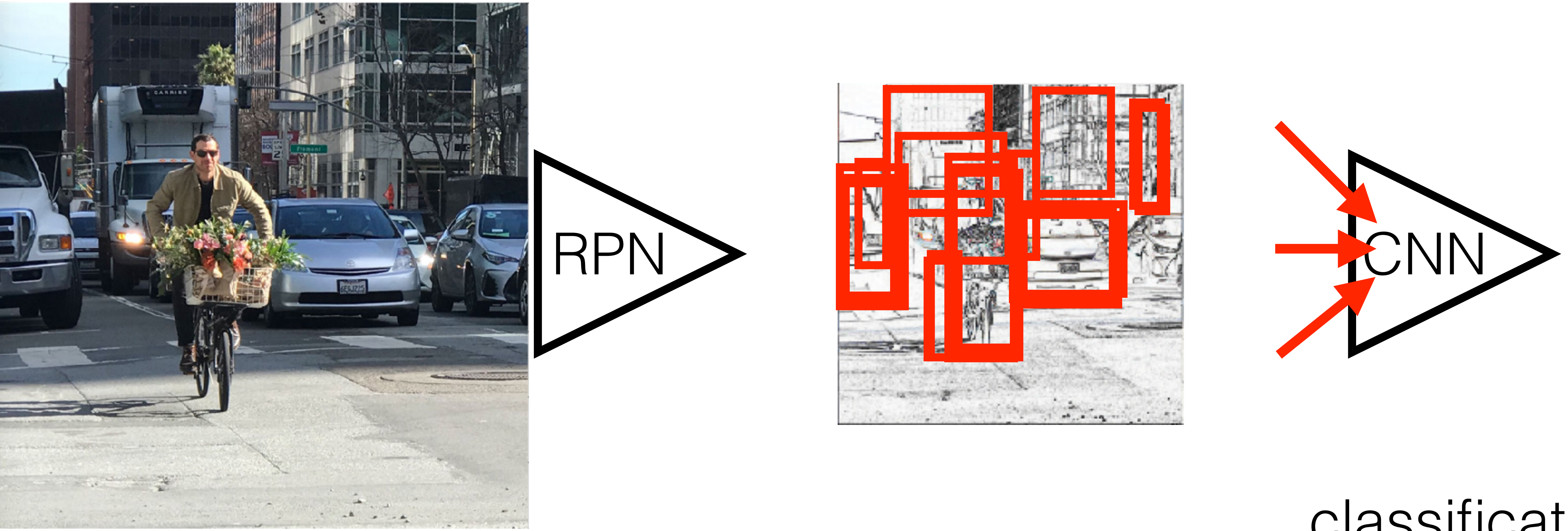
region proposal net
(output: 2k proposals)

classification
+
alignment
net

Save computational power by reusing RPN feature maps

Faster-RCNN https://arxiv.org/abs/1506.01497

Object detection



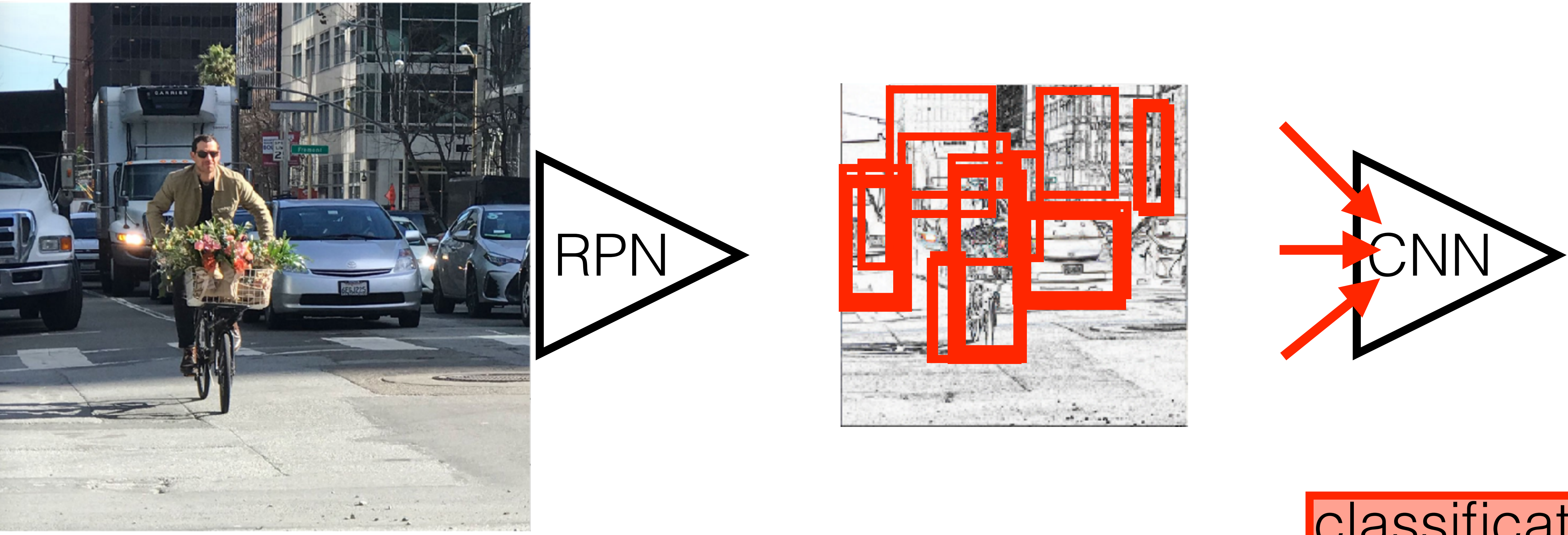
region proposal net
(output: 2k proposals)

classification
+
alignment
net

Save computational power by reusing RPN feature maps

Faster-RCNN https://arxiv.org/abs/1506.01497

Object detection



RPN

CNN

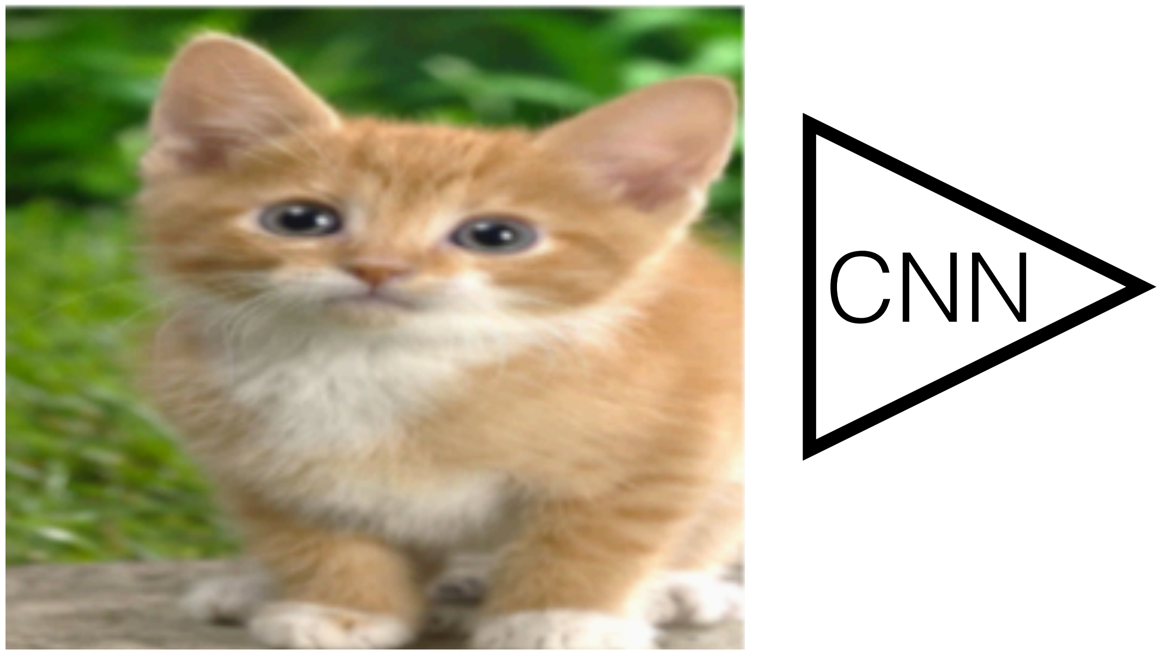
region proposal net
(output: 2k proposals)

classification
+
alignment
net

Faster-RCNN https://arxiv.org/abs/1506.01497

Object detection

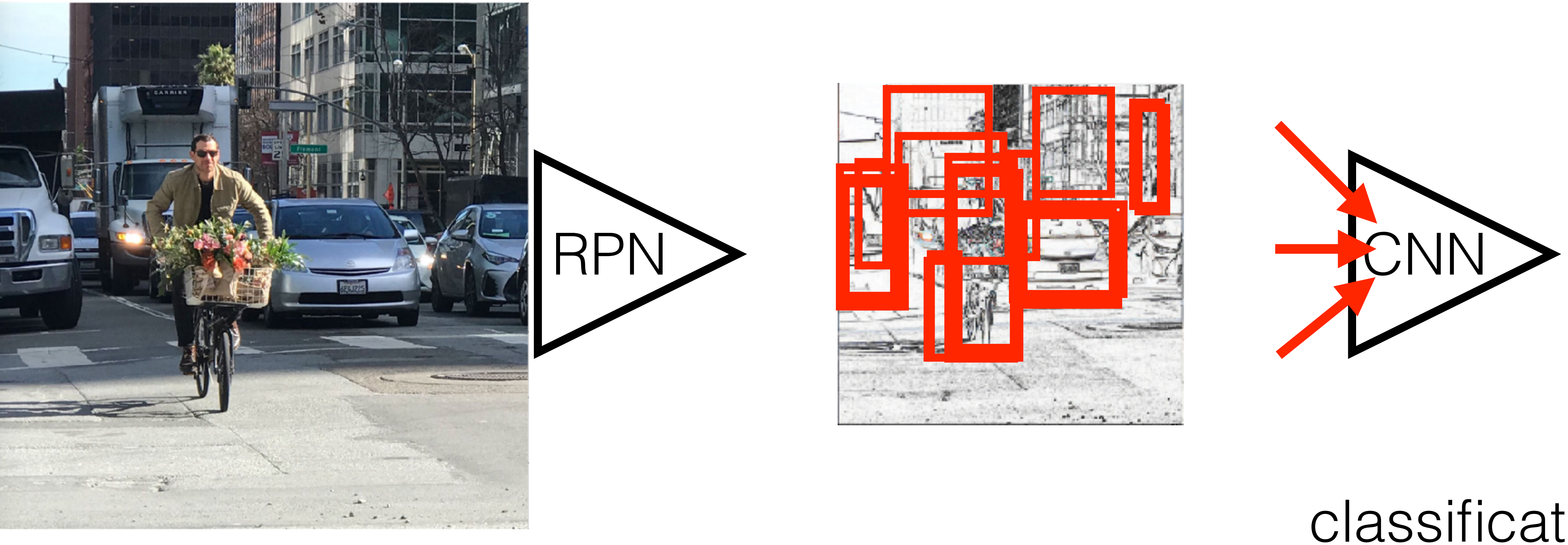
classification:



0.1	person
0.6	cat
0.2	house
0.1	background

Faster-RCNN https://arxiv.org/abs/1506.01497

Object detection



RPN

CNN

region proposal net
(output: 2k proposals)

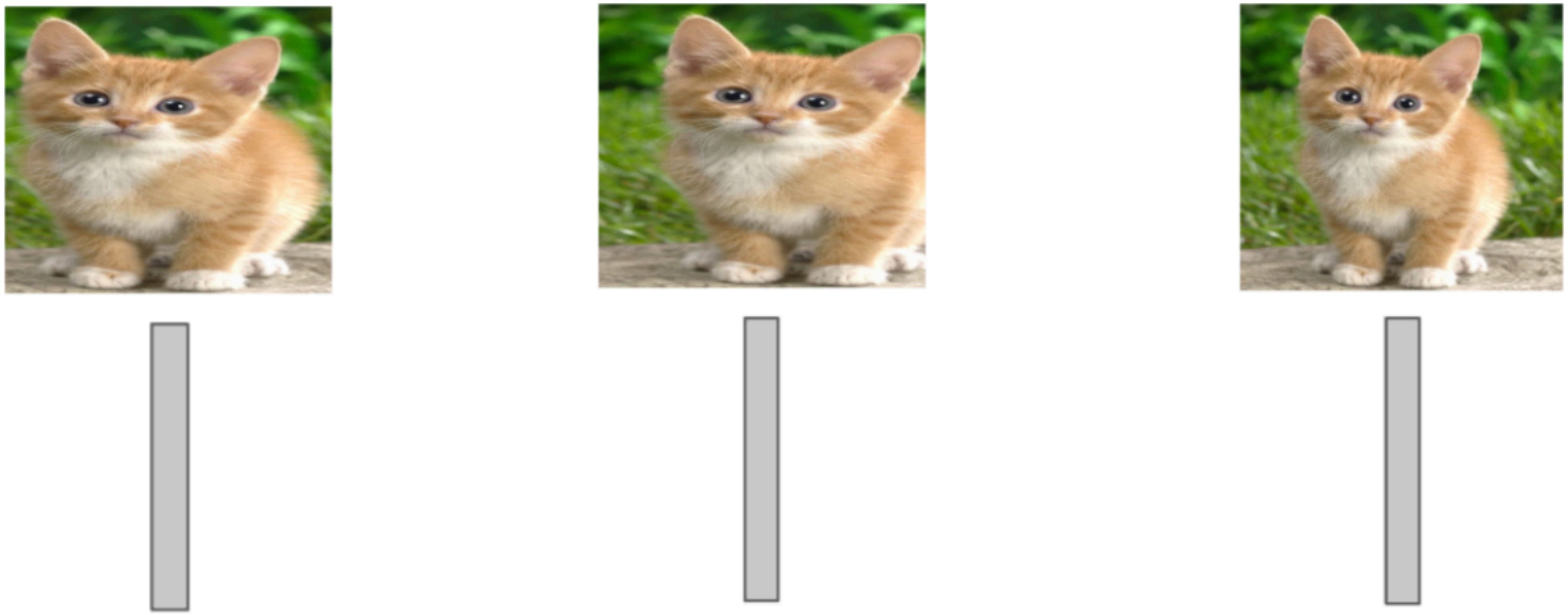
classification
+
alignment
net

Save computational power by reusing RPN feature maps

Faster-RCNN https://arxiv.org/abs/1506.01497

Object detection

alignment: $[\Delta\mathrm{x}, \Delta\mathrm{y}, \Delta\mathrm{w}, \Delta\mathrm{h}]$



(0, 0, 0, 0)
Proposal is good

(.25, 0, 0, 0)
Proposal too far to left

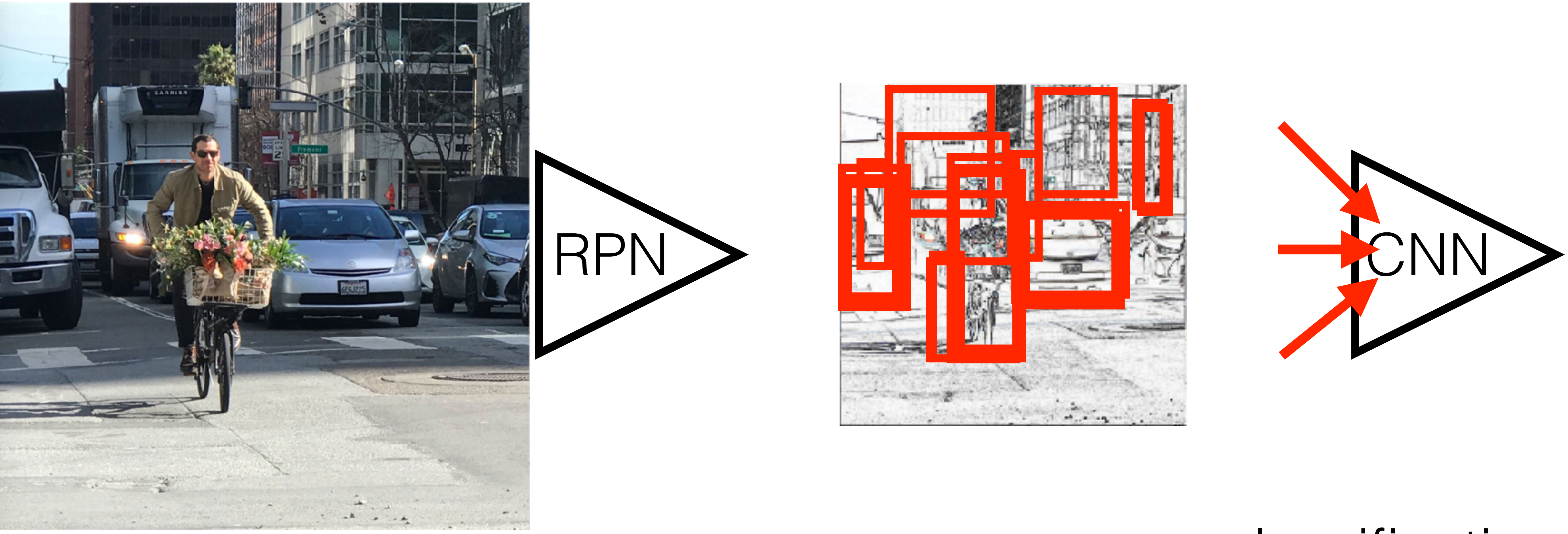
(0, 0, -0.125, 0)
Proposal too wide

Faster-RCNN https://arxiv.org/abs/1506.01497

Object detection

- Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  H x W x Aspect_Ratio x Scales x 0.001 sec = **months**
- Instead we can use elementary signal processing method to extract only 2k viable candidates:
  [Girschick ICCV 2015], Fast-RCNN
  https://arxiv.org/abs/1504.08083
  (find 2k cand.) + (2k cand. x 0.001 sec) = **47+2 sec = 49 sec**
- Do region proposal by CNN => **0.3 + 2 = 2.3 sec**

Object detection

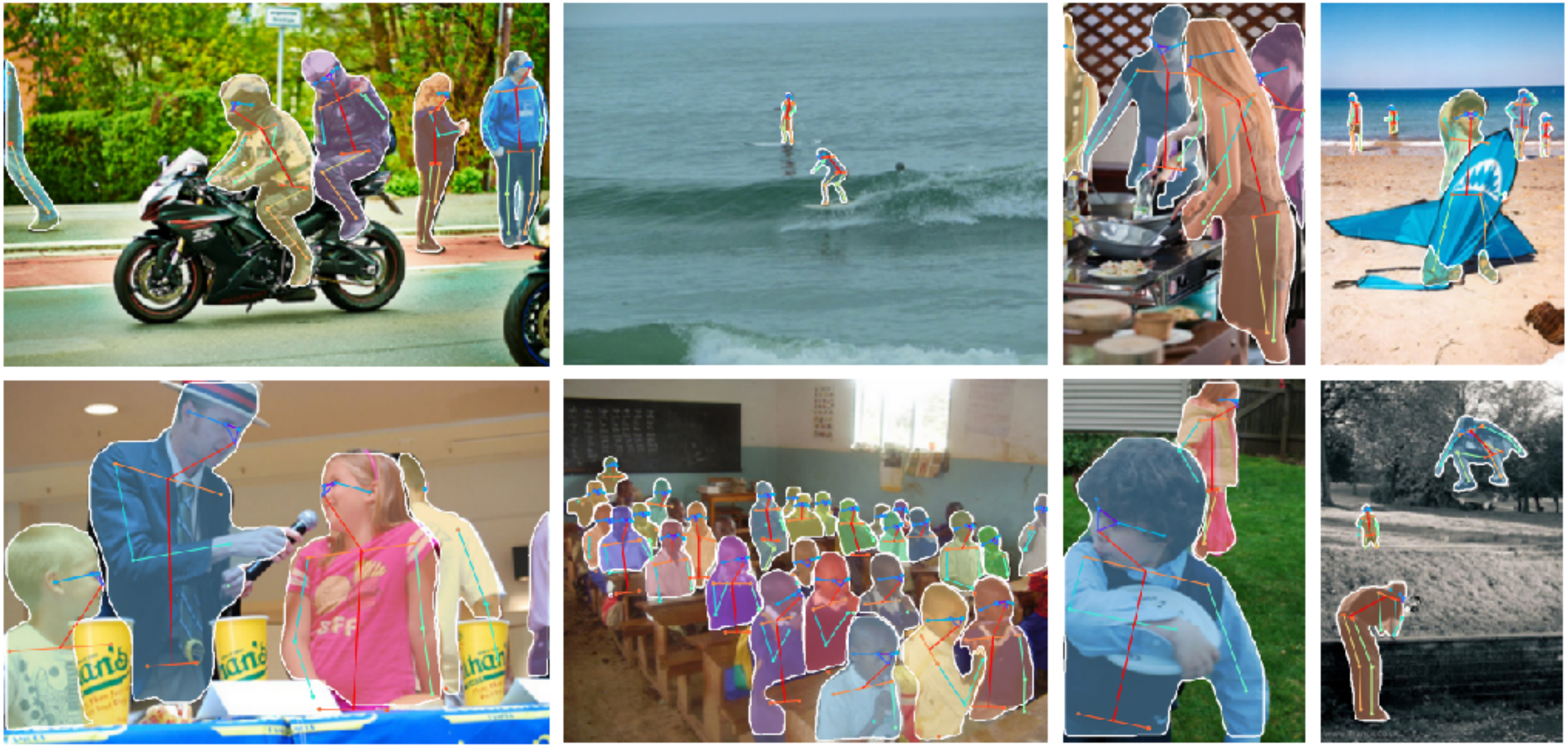


region proposal net
(output: 2k proposals)

[He et al CVPR 2017] Mask-RCNN
https://arxiv.org/abs/1703.06870

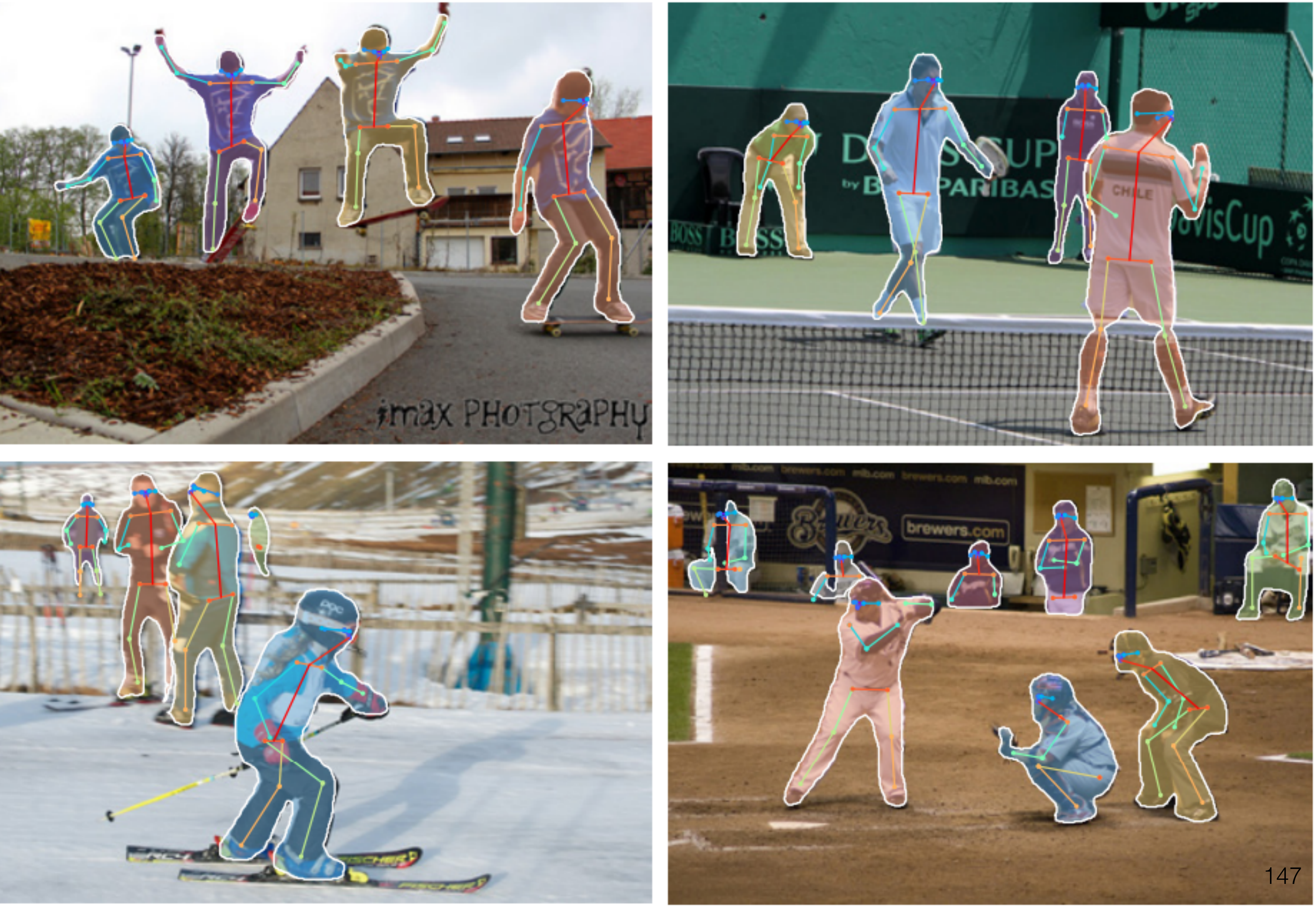
classification
+
alignment
+
segmentation mask
+
pose regression

Mask RCNN - results



[He et al CVPR 2017] Mask-RCNN
https://arxiv.org/abs/1703.06870

Mask RCNN - results



Object detection

- Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  H x W x Aspect_Ratio x Scales x 0.001 sec = **months**
- Instead we can use elementary signal processing method to extract only 2k viable candidates:
  [Girschick ICCV 2015], Fast-RCNN
  https://arxiv.org/abs/1504.08083
  (find 2k cand.) + (2k cand. x 0.001 sec) = **47+2 sec = 49 se**
- Do region proposal by CNN => **0.3 + 2 = 2.3 sec**
- Similar idea but more efficient implementation YOLO/SSD: about **0.2 sec**
  code: https://pjreddie.com/darknet/yolo/
  [Redmont CVPR 2018], https://arxiv.org/abs/1804.02767

[Liu ECCV 2016], https://arxiv.org/abs/1512.02325

Deep convolutional - object detection

Conclusion and links

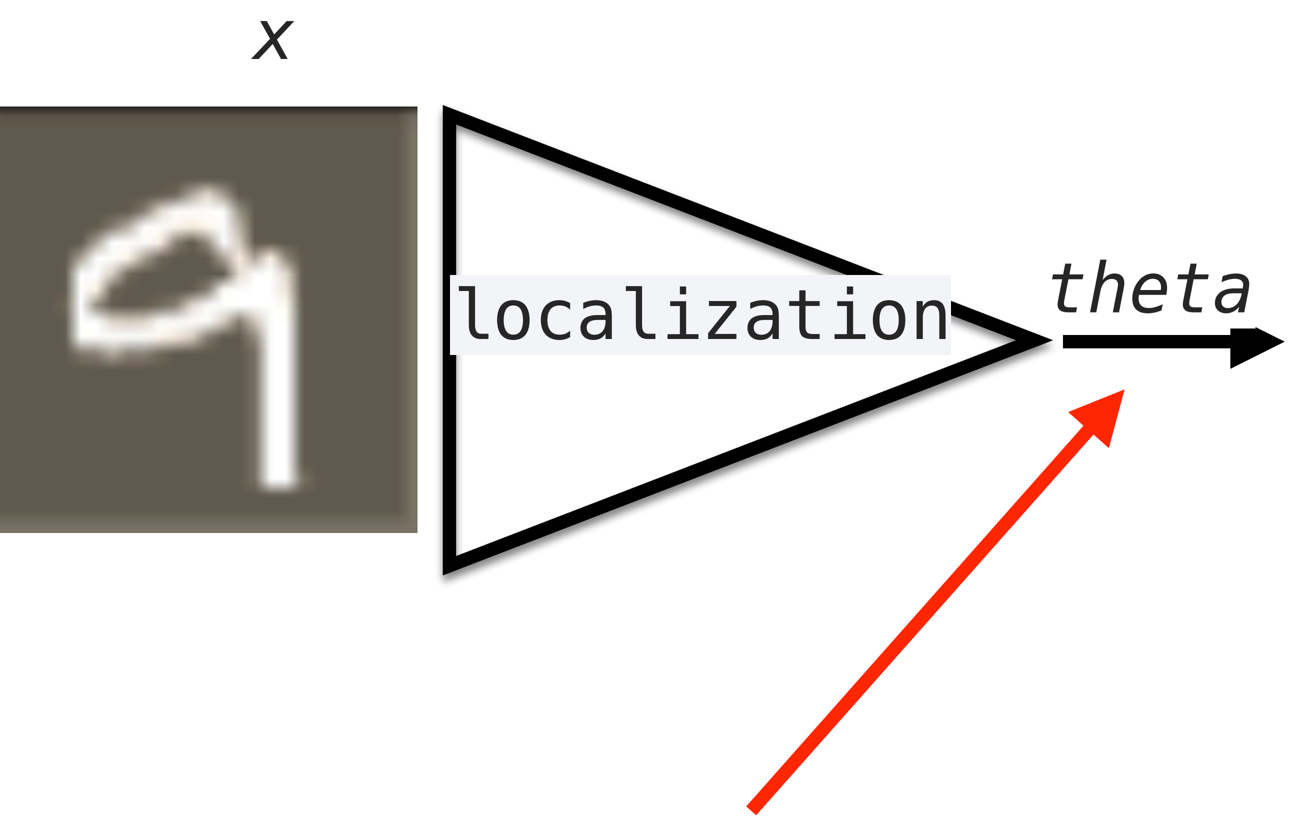
- Many datasets/challenges/results:
  - http://www.robustvision.net
  - http://mscoco.org
  - http://www.image-net.org
  - http://host.robots.ox.ac.uk/pascal/VOC/

- Comparison: Yolo v2/v3, DeepLab v3, MaskRCNN (30min) https://www.youtube.com/watch?v=s8Ui_kV9dhw

Outline

- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Spatial Transformer networks
- Architectures of feature matching networks

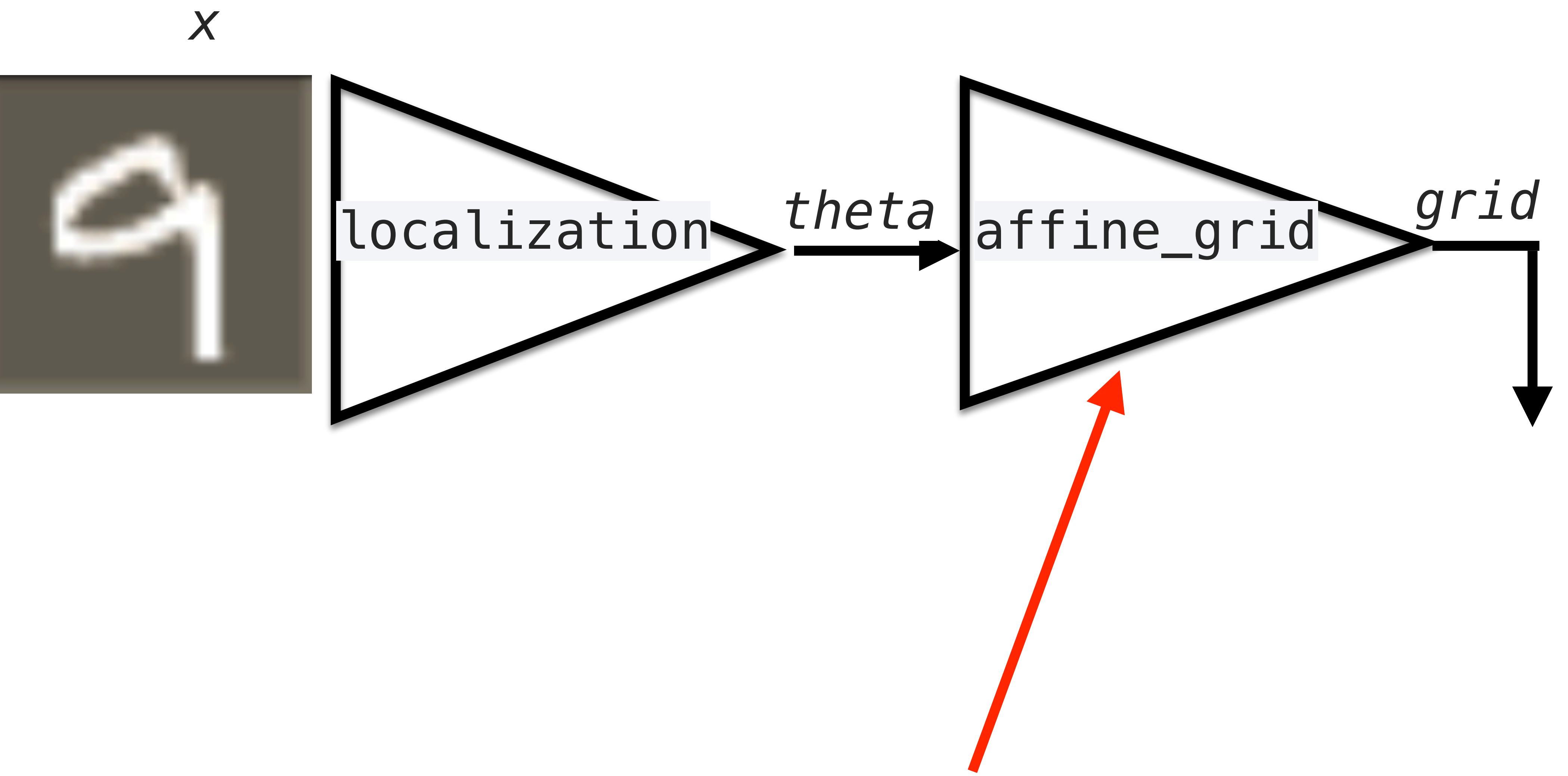
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



2D similarity transformation

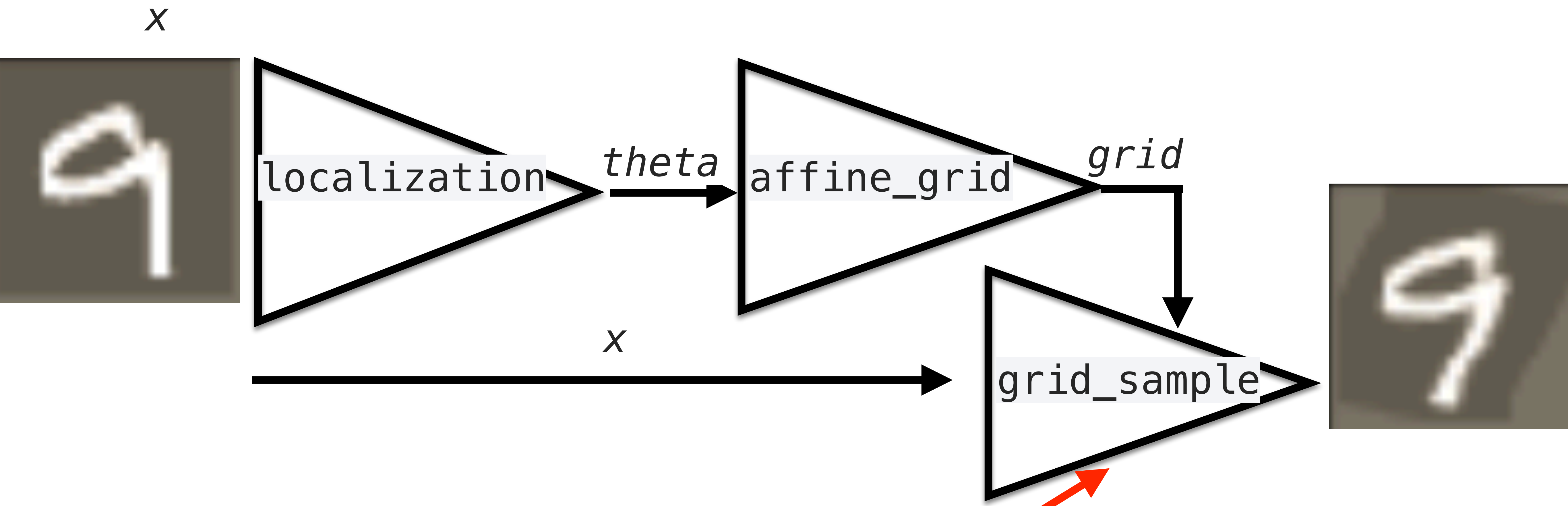
$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) & \theta_2 \\ -\sin(\theta_1) & \cos(\theta_1) & \theta_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



```
torch.nn.functional.affine_grid(theta, size, align_corners=None)
```

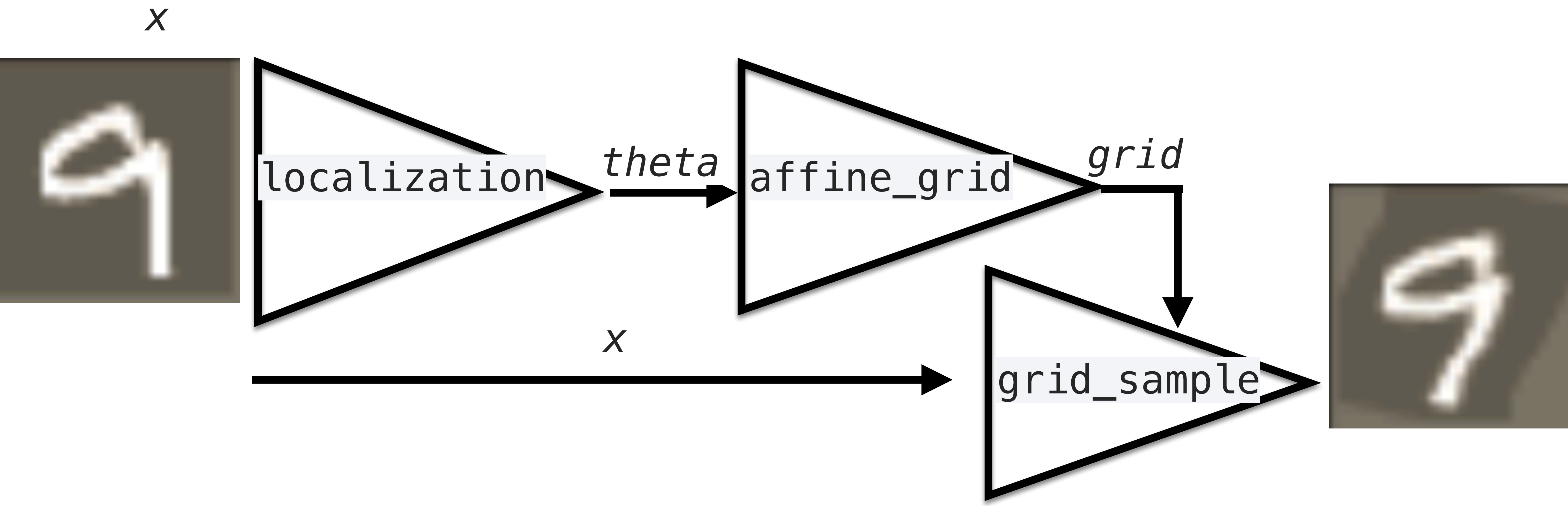
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



```
torch.nn.functional.affine_grid(theta, size, align_corners=None)

torch.nn.functional.grid_sample(input, grid, mode='bilinear',
                                padding_mode='zeros', align_corners=None)
```

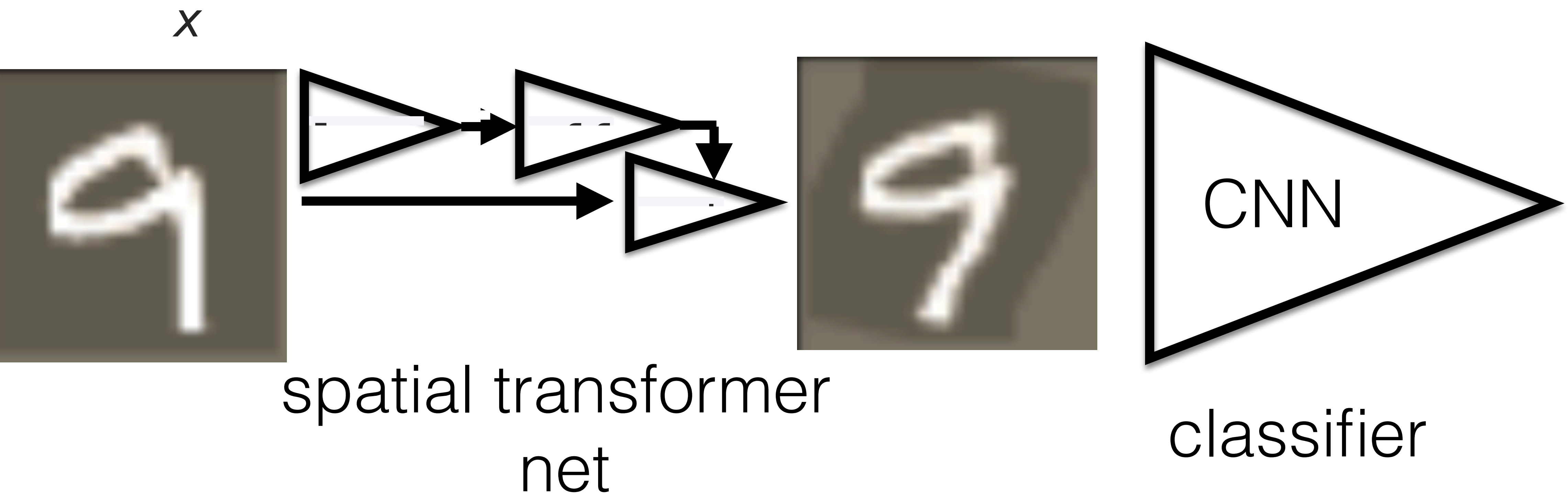
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



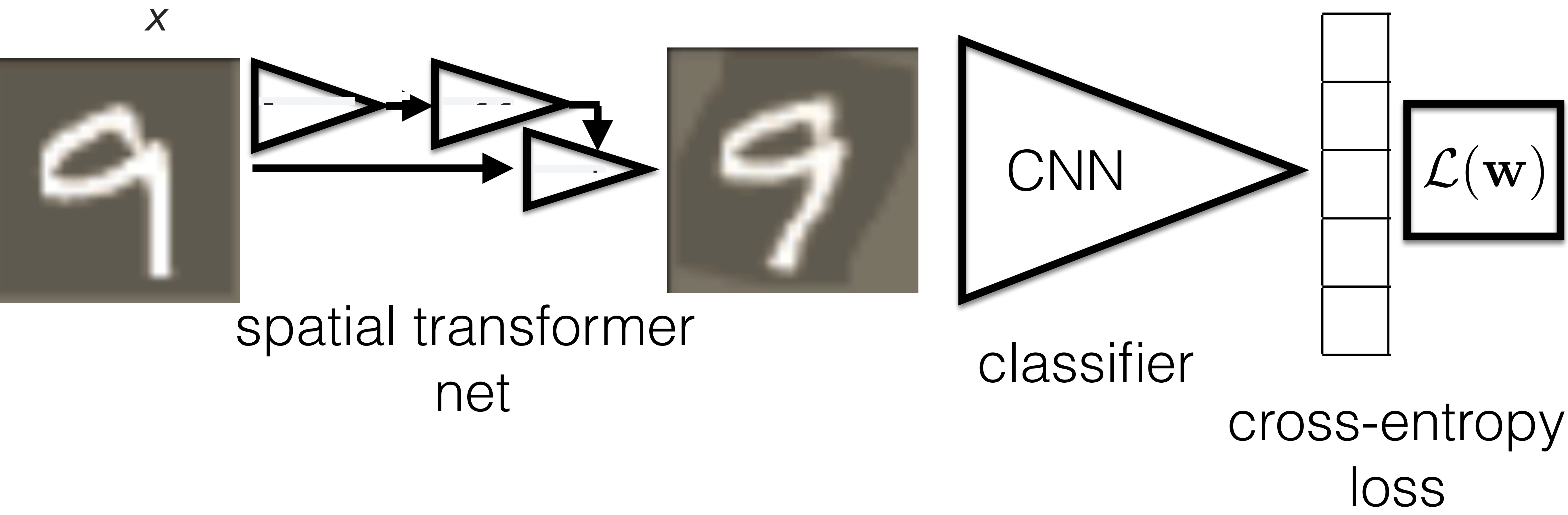
```
torch.nn.functional.affine_grid(theta, size, align_corners=None)

torch.nn.functional.grid_sample(input, grid, mode='bilinear',
                                padding_mode='zeros', align_corners=None)
```

Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



x

spatial transformer net

CNN

classifier

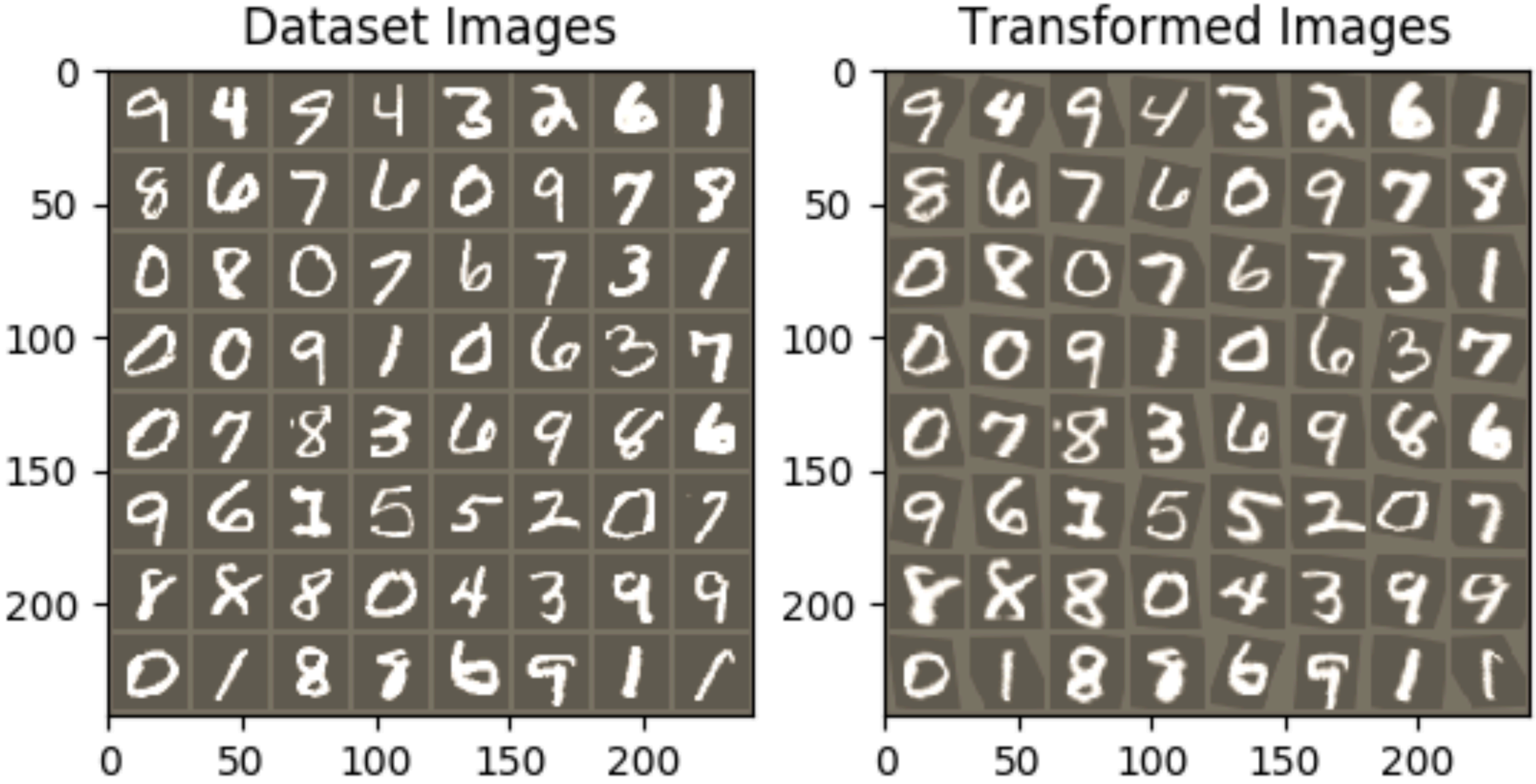
$\mathcal{L}(\mathbf{w})$

cross-entropy loss

Backpropagation learns also STN weights, which perform the most suitable transformation for the classification task

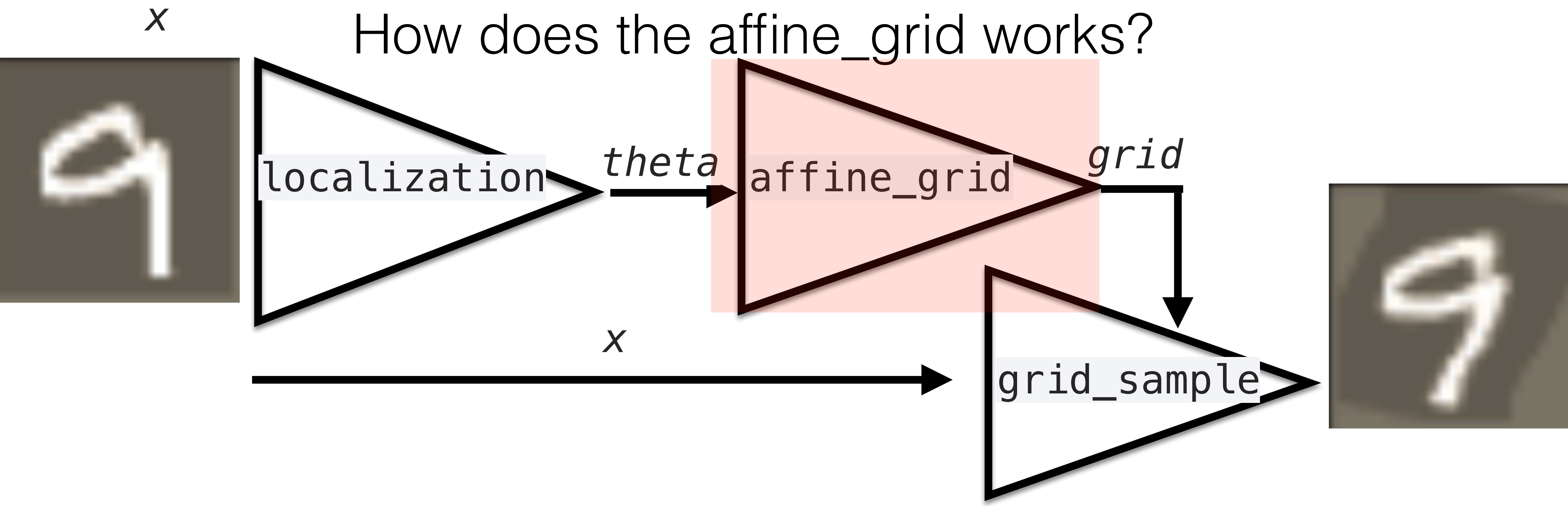
Spatial Transformer networks

https://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html



Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?



$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) & \theta_2 \\ -\sin(\theta_1) & \cos(\theta_1) & \theta_3 \end{bmatrix} \begin{bmatrix} m \\ n \\ 1 \end{bmatrix}$$

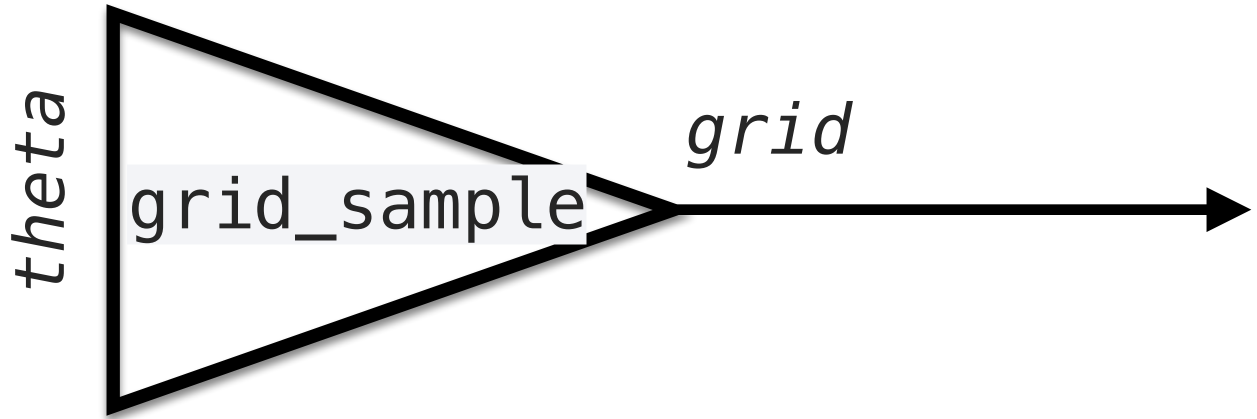
coordinates of all pixels in the input pixels (fixed)

coordinates of all pixels in the transformed (grid)

Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Can we translate image U by 1 pixel up?



$x(m,n)$

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

$y(m,n)$

2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5

$\theta_1 = 0$

$\theta_2 = 0$

$\theta_3 = 1$

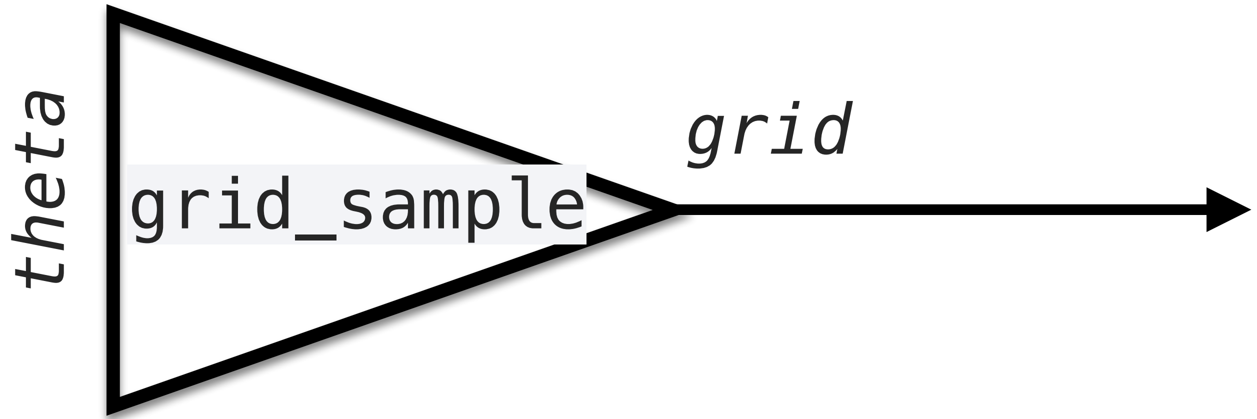
$$\begin{bmatrix} x(m,n) \\ y(m,n) \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) & \theta_2 \\ -\sin(\theta_1) & \cos(\theta_1) & \theta_3 \end{bmatrix} \begin{bmatrix} m \\ n \\ 1 \end{bmatrix}$$

Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Can we translate image U by 1 pixel up?



$x(m,n)$

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

$y(m,n)$

2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5

$\theta_1 = 0$

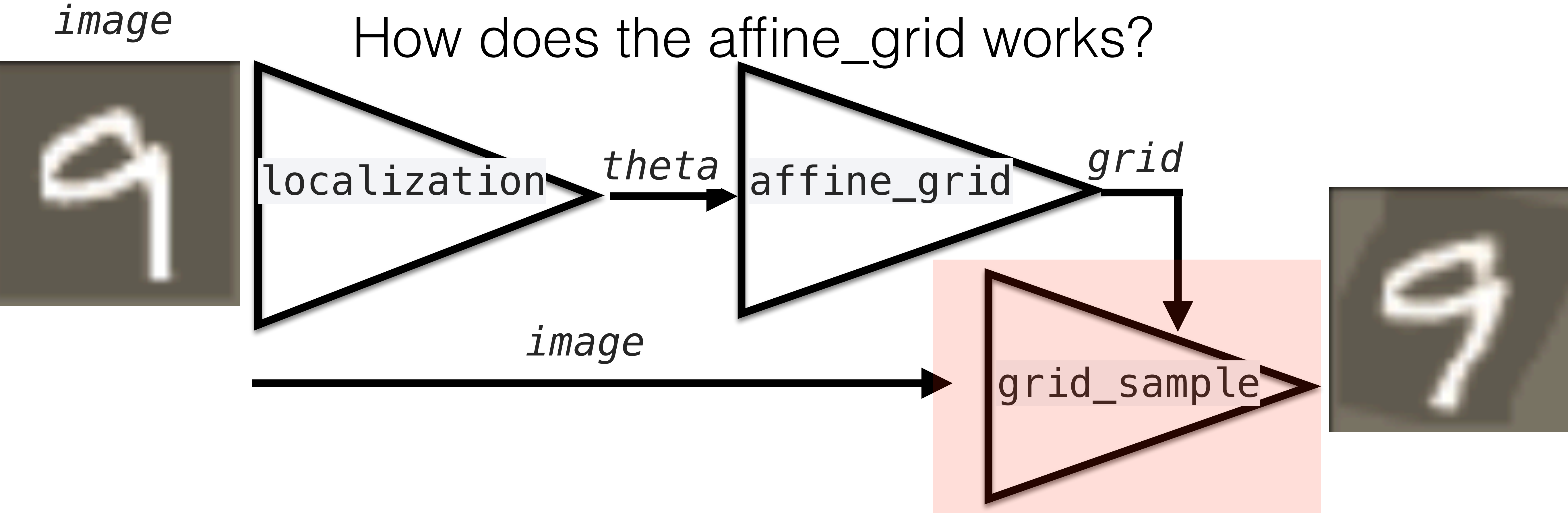
$\theta_2 = 0$

$\theta_3 = 1$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) & \theta_2 \\ -\sin(\theta_1) & \cos(\theta_1) & \theta_3 \end{bmatrix} \begin{bmatrix} m \\ n \\ 1 \end{bmatrix}$$

Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

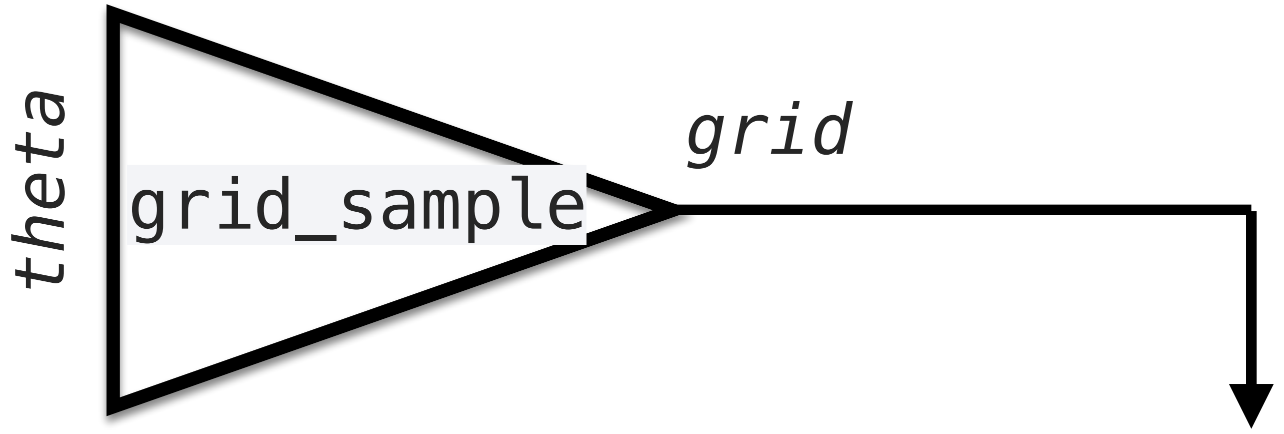


Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Can we translate image U by 1 pixel up?



$x(m,n)$

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

$y(m,n)$

2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5

$$\begin{array}{c|cccc} & & x & & \\ \hline & 1 & 0 & 0 & 2 \\ y & 6 & 3 & 2 & 1 \\ & 0 & 1 & 0 & 3 \\ & 1 & 3 & 1 & 2 \end{array} \otimes \begin{array}{c|cccc} & & u & & \\ \hline & & & & \\ v & & & & \\ & & & & \\ & & & & \end{array} = \begin{array}{c|cccc} & m=1 & & & \\ \hline n=1 & ? & & & \\ & & & & \\ & & & & \\ & & & & \end{array}$$

input image (x,y)

kernel (u,v)

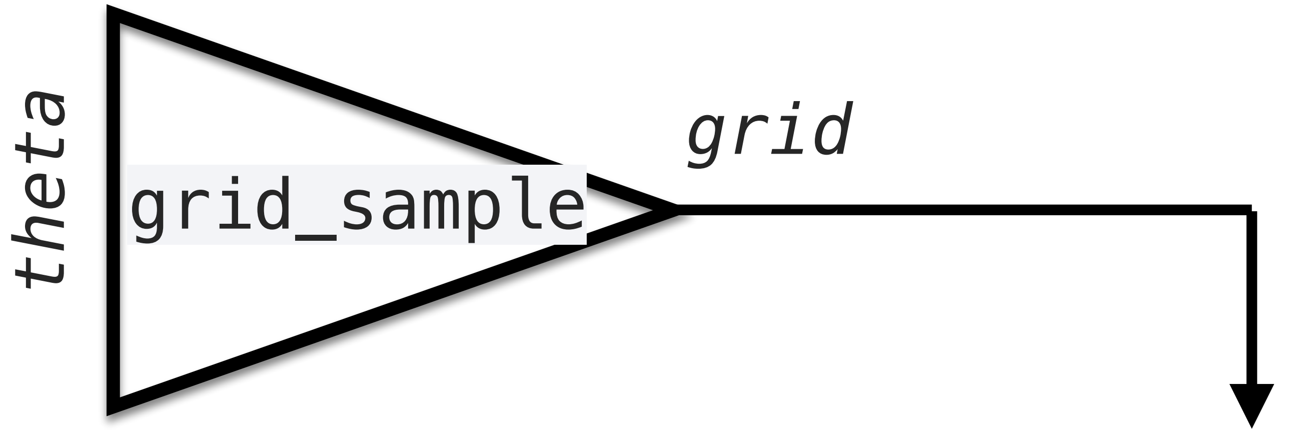
output image (m,n)

Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Can we translate image U by 1 pixel up?



$x(m,n)$

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

$y(m,n)$

2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5

x

y				
	1	0	0	2
	6	3	2	1
	0	1	0	3
	1	3	1	2

input image (x,y)

$\otimes$

u

v				

kernel (u,v)

$=$

	$m=1$			
$n=1$	?			

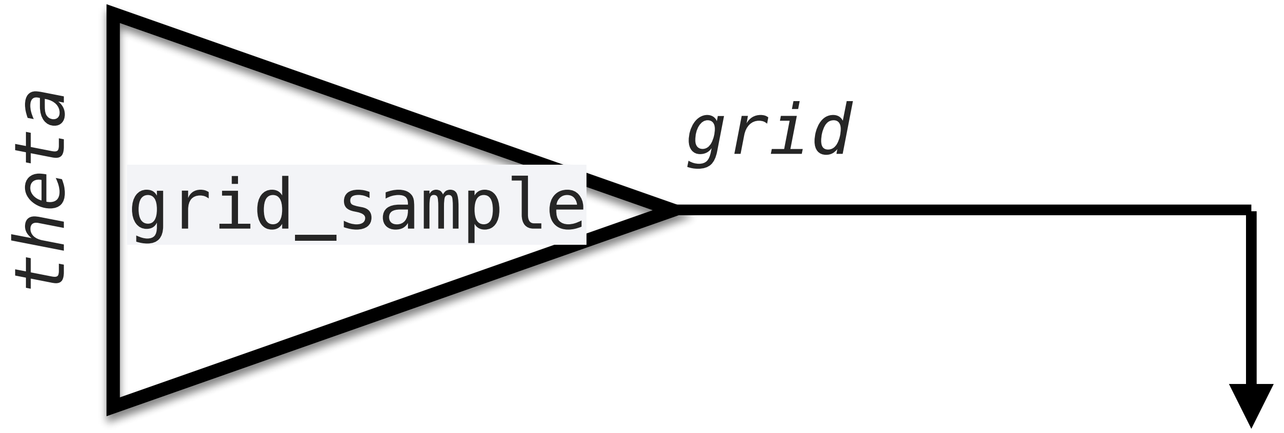
output image (m,n)

Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Can we translate image U by 1 pixel up?



$x(m,n)$

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

$y(m,n)$

2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5

$$\begin{array}{c|cccc} & & x & & \\ \hline & 1 & 0 & 0 & 2 \\ y & 6 & 3 & 2 & 1 \\ & 0 & 1 & 0 & 3 \\ & 1 & 3 & 1 & 2 \end{array} \otimes \begin{array}{c|cccc} & & u & & \\ \hline & 0 & 0 & 0 & 0 \\ v & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \end{array} = \begin{array}{c|cccc} & m=1 & & & \\ \hline n=1 & ? & & & \\ & & & & \\ & & & & \\ & & & & \end{array}$$

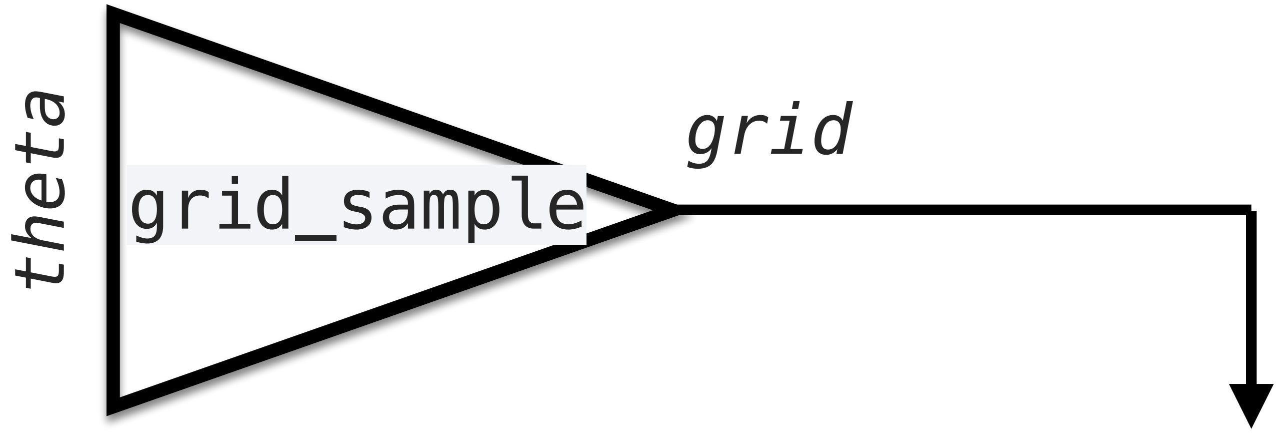
$$\kappa_{(m,n)}(u,v) = \delta\big(u - x(m,n)\big) \cdot \delta\big(v - y(m,n)\big)$$

Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Can we translate image U by 1 pixel up?



$x(m,n)$

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

$y(m,n)$

2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5

$$y\ \begin{array}{cccc} \multicolumn{4}{c}{x} \\ 1 & 0 & 0 & 2 \\ 6 & 3 & 2 & 1 \\ 0 & 1 & 0 & 3 \\ 1 & 3 & 1 & 2 \end{array} \otimes \ v\ \begin{array}{cccc} \multicolumn{4}{c}{u} \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} = \begin{array}{c|cccc} & & m=2 & & \\ n=1 & 6 & \boxed{3} & & \\ & & & & \\ & & & & \\ & & & & \end{array}$$

$$\kappa_{(m,n)}(u,v) = \delta\big(u - x(m,n)\big) \cdot \delta\big(v - y(m,n)\big)$$

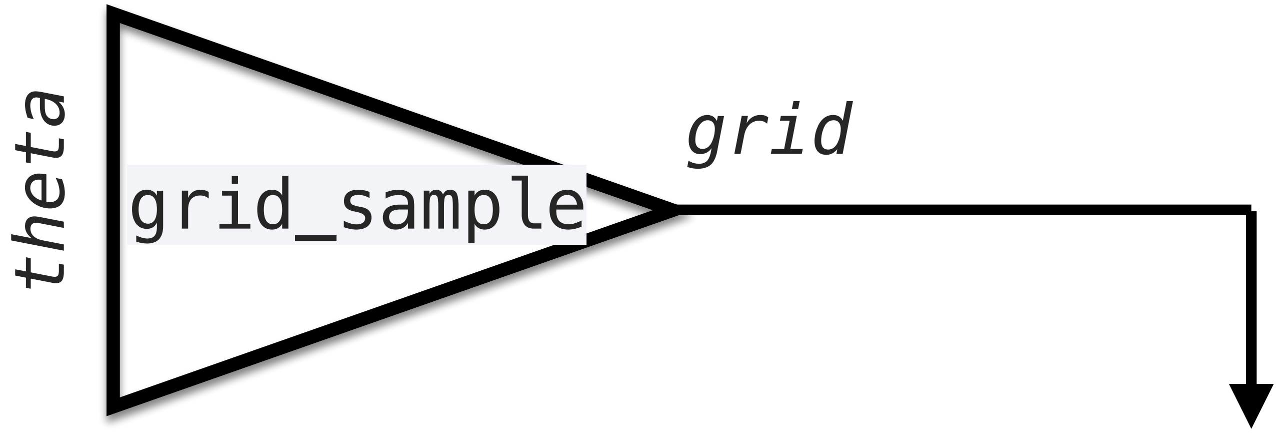
Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Can we translate image U by 1 pixel up?

$$x(m,n) = \begin{array}{|c|c|c|c|} \hline 1&2&3&4\\ \hline 1&2&3&4\\ \hline 1&2&3&4\\ \hline 1&2&3&4\\ \hline \end{array} \qquad y(m,n) = \begin{array}{|c|c|c|c|} \hline 2&2&2&2\\ \hline 3&3&3&3\\ \hline 4&4&4&4\\ \hline 5&5&5&5\\ \hline \end{array}$$



$$\begin{array}{c|cccc} & & x & & \\ \hline & 1&0&0&2\\ y & 6&3&2&1\\ & 0&1&0&3\\ & 1&3&1&2 \end{array} \otimes \begin{array}{c|cccc} & & u & & \\ \hline & 0&0&0&0\\ v & 0&0&0&0\\ & 0&0&0&0\\ & 0&0&0&0 \end{array} = \begin{array}{c|cccc} & & m=2 & & \\ \hline n=1 & 6&3&2&1\\ & 0&1&0&3\\ & 1&3&1&2\\ & 0&0&0&0 \end{array}$$

$$\kappa_{(m,n)}(u,v) = \delta\big(u - x(m,n)\big) \cdot \delta\big(v - y(m,n)\big)$$

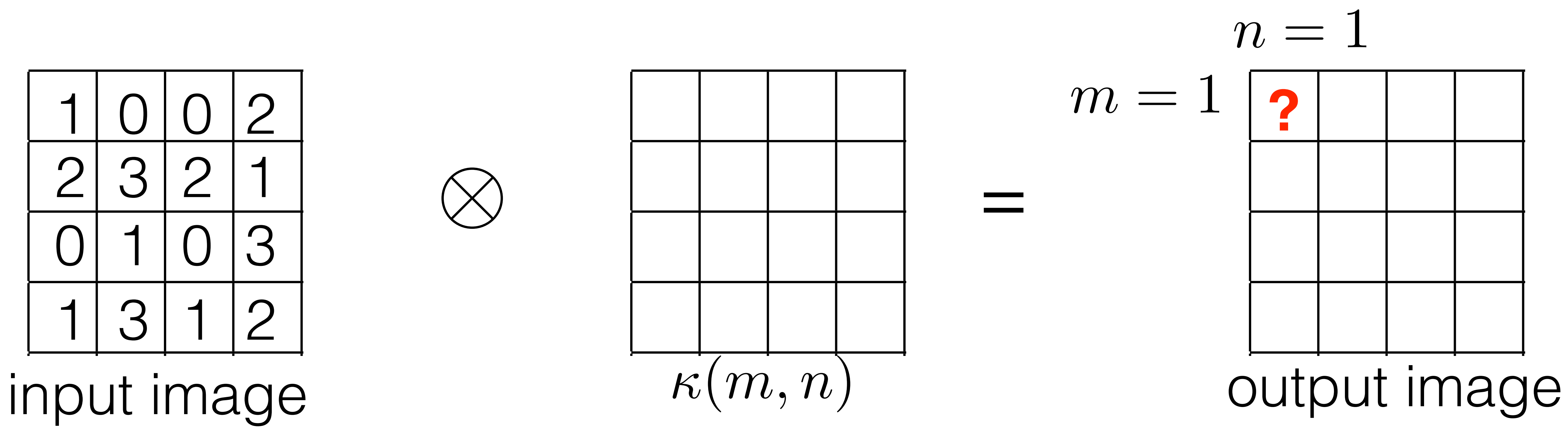
Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image translation:

Can we translate image U by 1/2 pixel?



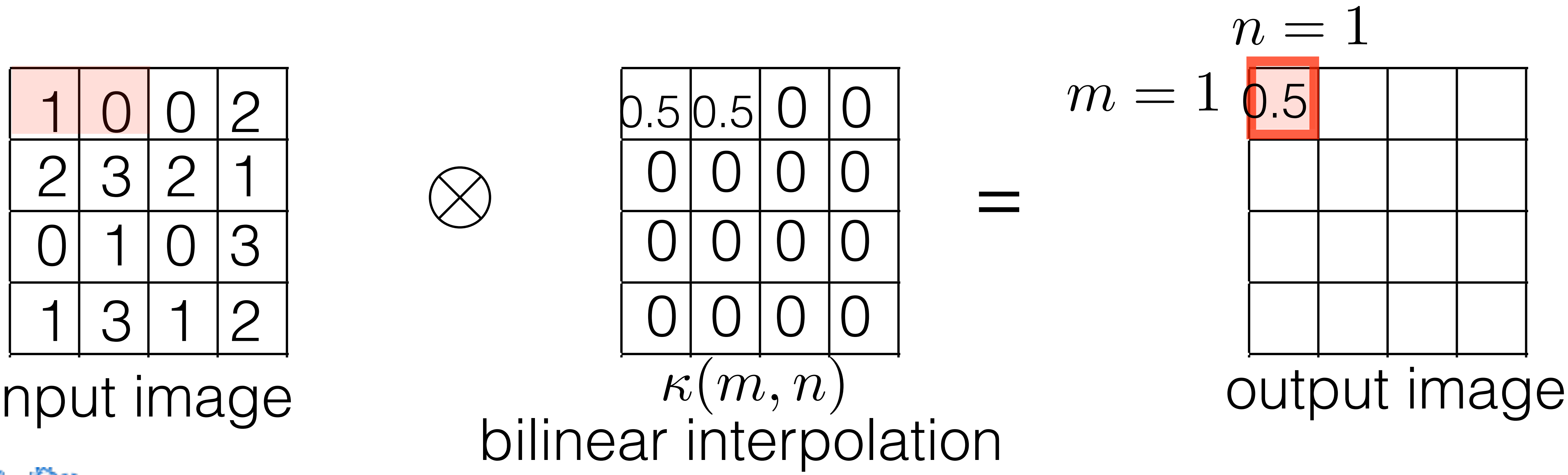
1	0	0	2
2	3	2	1
0	1	0	3
1	3	1	2

Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image translation:

Can we translate image U by 1/2 pixel?



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

input image

$\otimes$

0.5	0.5	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$\kappa(m,n)$

bilinear interpolation

$=$

$n = 1$

$m = 1$

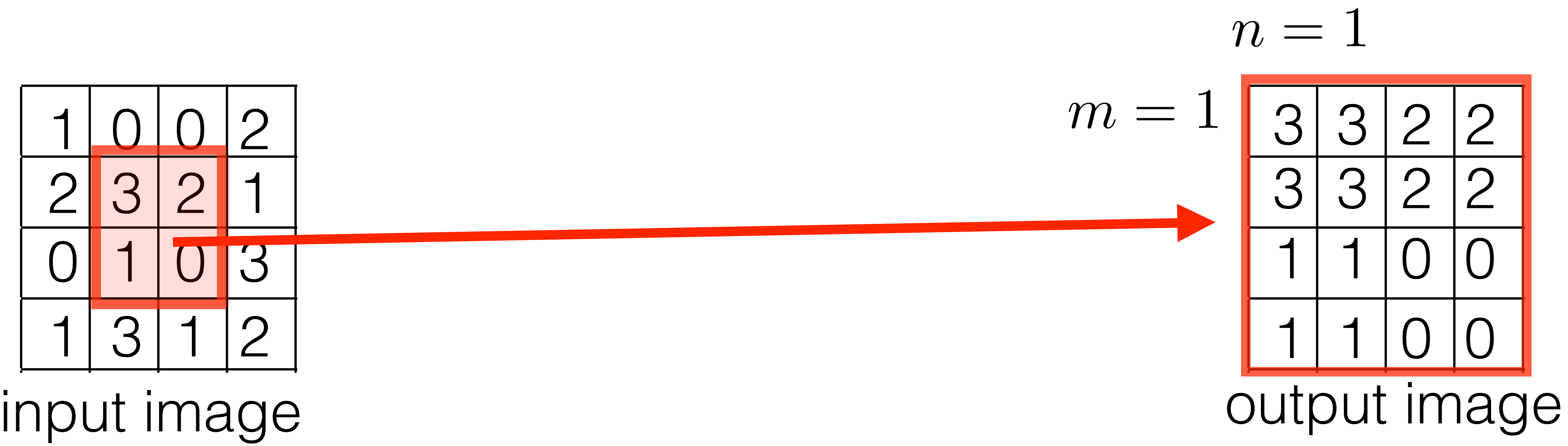
0.5			

output image

Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:



$n = 1$

$m = 1$

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

input image

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

output image

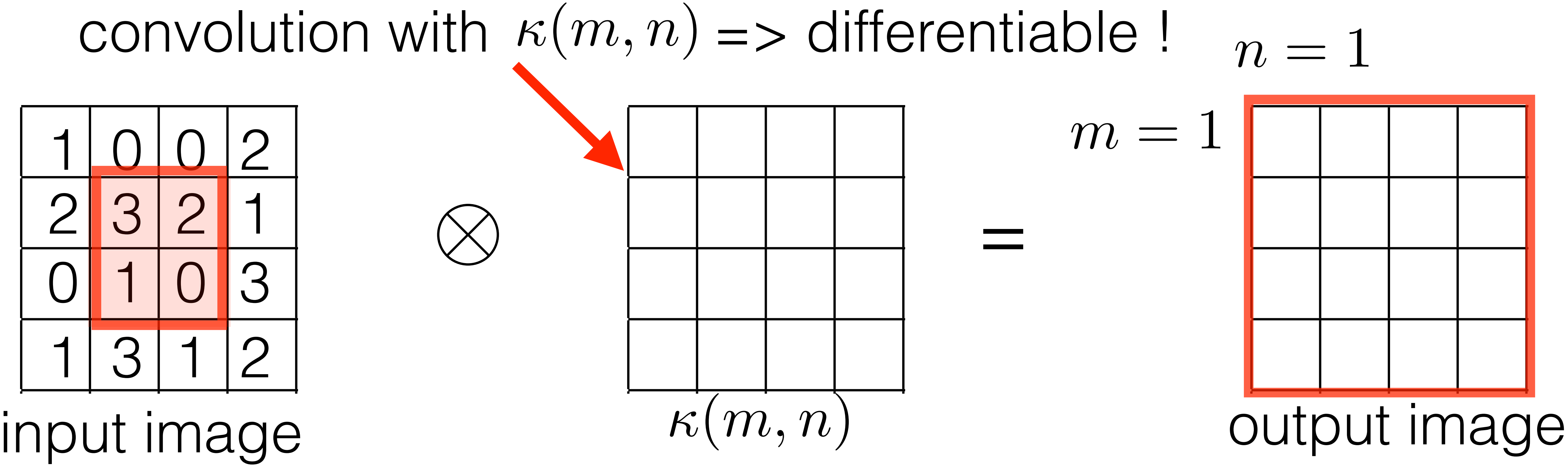
Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:

convolution with $\kappa(m,n)$ => differentiable !

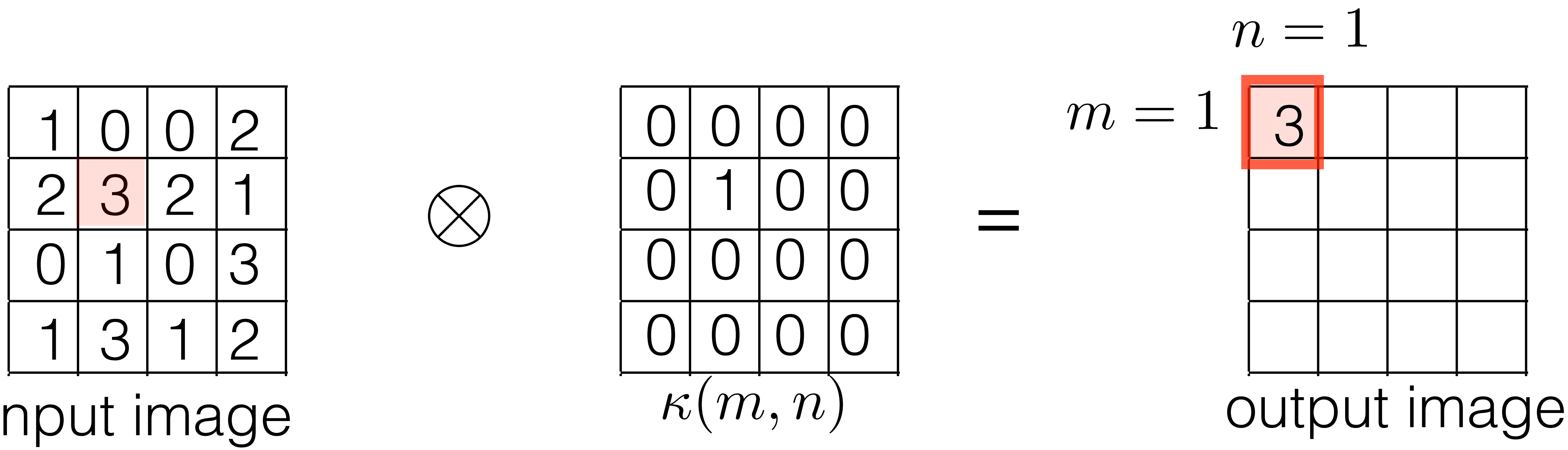


1	0	0	2
2	3	2	1
0	1	0	3
1	3	1	2

Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:



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

input image

$\otimes$

0	0	0	0
0	1	0	0
0	0	0	0
0	0	0	0

$\kappa(m, n)$

=

$n = 1$

$m = 1$

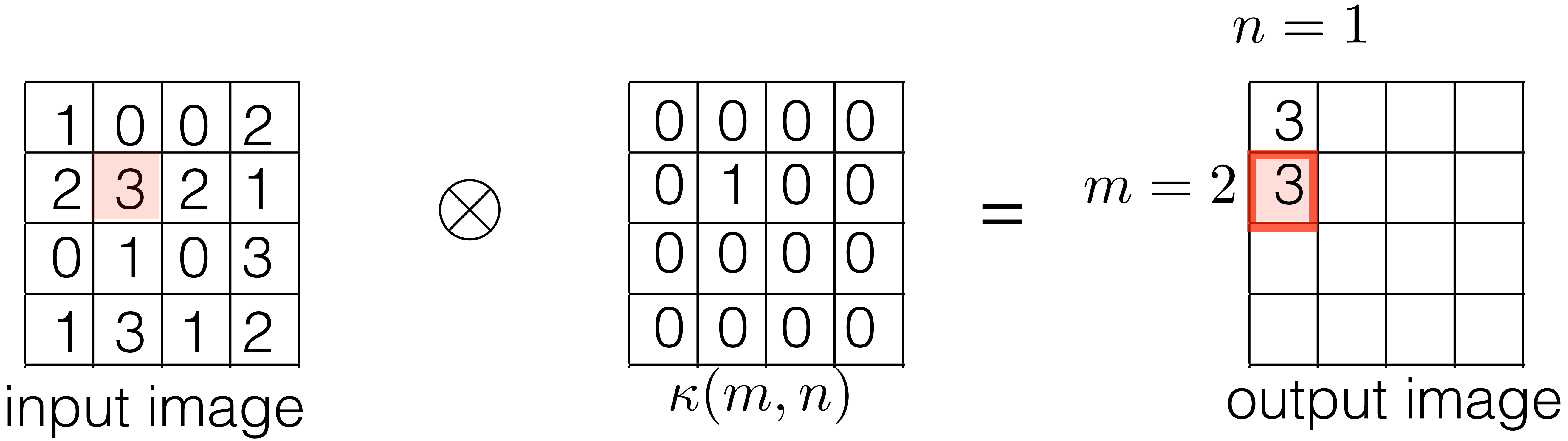
3			

output image

Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:



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

input image

$\otimes$

0	0	0	0
0	1	0	0
0	0	0	0
0	0	0	0

$\kappa(m, n)$

=

$n = 1$

$m = 2$

3			
3			

output image

Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:

$$\underset{\text{input image}}{\begin{array}{|c|c|c|c|}\hline 1&0&0&2\\\hline 2&3&2&1\\\hline 0&1&0&3\\\hline 1&3&1&2\\\hline\end{array}} \otimes \underset{\kappa(m,n)}{\begin{array}{|c|c|c|c|}\hline 0&0&0&0\\\hline 0&0&0&0\\\hline 0&1&0&0\\\hline 0&0&0&0\\\hline\end{array}} = \underset{\text{output image}}{\begin{array}{c|c|c|c|c|} & n=1 & & & \\\hline & 3 & & & \\\hline & 3 & & & \\\hline m=3 & \boxed{1} & & & \\\hline & & & & \\\hline\end{array}}$$

Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:

$$\underbrace{\begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 2 \\ \hline 2 & 3 & 2 & 1 \\ \hline 0 & 1 & 0 & 3 \\ \hline 1 & 3 & 1 & 2 \\ \hline \end{array}}_{\text{input image}} \otimes \underbrace{\begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array}}_{\kappa(m,n)} = \underbrace{\begin{array}{c|c|c|c|c|} & n=1 & & & \\ \hline & 3 & & & \\ \hline & 3 & & & \\ \hline & 1 & & & \\ \hline m=4 & \boxed{1} & & & \\ \hline \end{array}}_{\text{output image}}$$

Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:

$$\underset{\text{input image}}{\begin{array}{|c|c|c|c|}\hline 1&0&0&2\\\hline 2&3&2&1\\\hline 0&1&0&3\\\hline 1&3&1&2\\\hline\end{array}} \otimes \underset{\kappa(m,n)}{\begin{array}{|c|c|c|c|}\hline 0&0&0&0\\\hline 0&0&0&0\\\hline 0&1&0&0\\\hline 0&0&0&0\\\hline\end{array}} = \underset{\text{output image}}{\begin{array}{c|c|c|c|c|} \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{n=2} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} \\ \cline{2-5} & 3&3&&\\ \cline{2-5} &3&3&&\\ \cline{2-5} &1&1&&\\ \cline{2-5} m=4 &1&\boxed{1}&&\\ \cline{2-5}\end{array}}$$

Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:

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

input image

$\otimes$

0	0	0	0
0	0	0	0
0	1	0	0
0	0	0	0

$\kappa(m,n)$

$=$

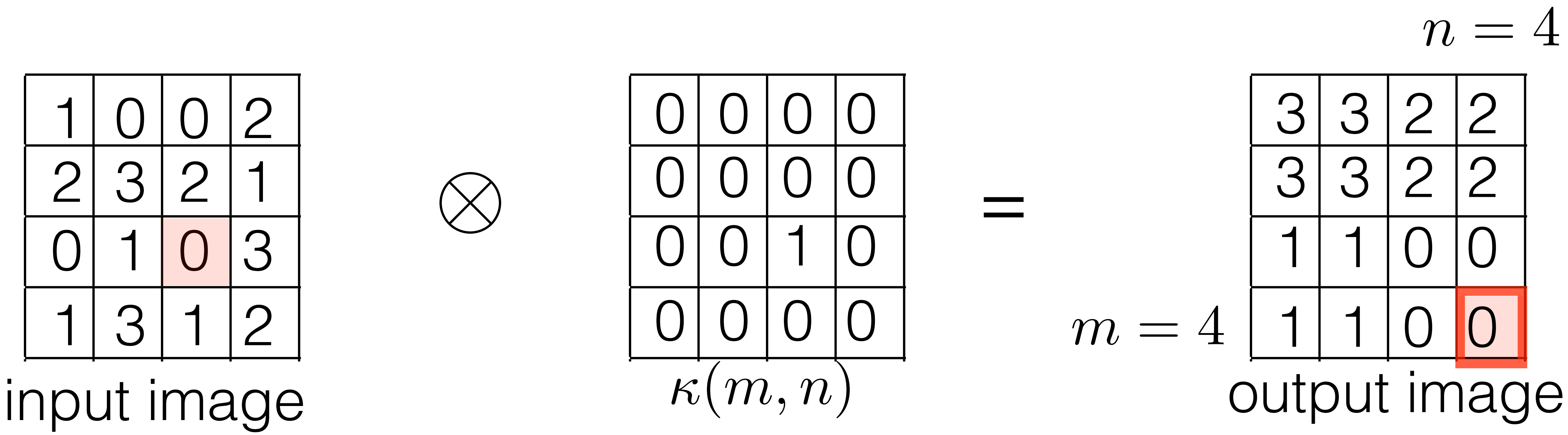
$$\begin{array}{c|cccc} & & n=2 & & \\ \hline & 3 & 3 & & \\ & 3 & 3 & & \\ & 1 & 1 & & \\ m=4 & 1 & 1 & & \end{array}$$

output image

Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:



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

input image

$\otimes$

0	0	0	0
0	0	0	0
0	0	1	0
0	0	0	0

$\kappa(m, n)$

$=$

$n = 4$

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

$m = 4$

output image

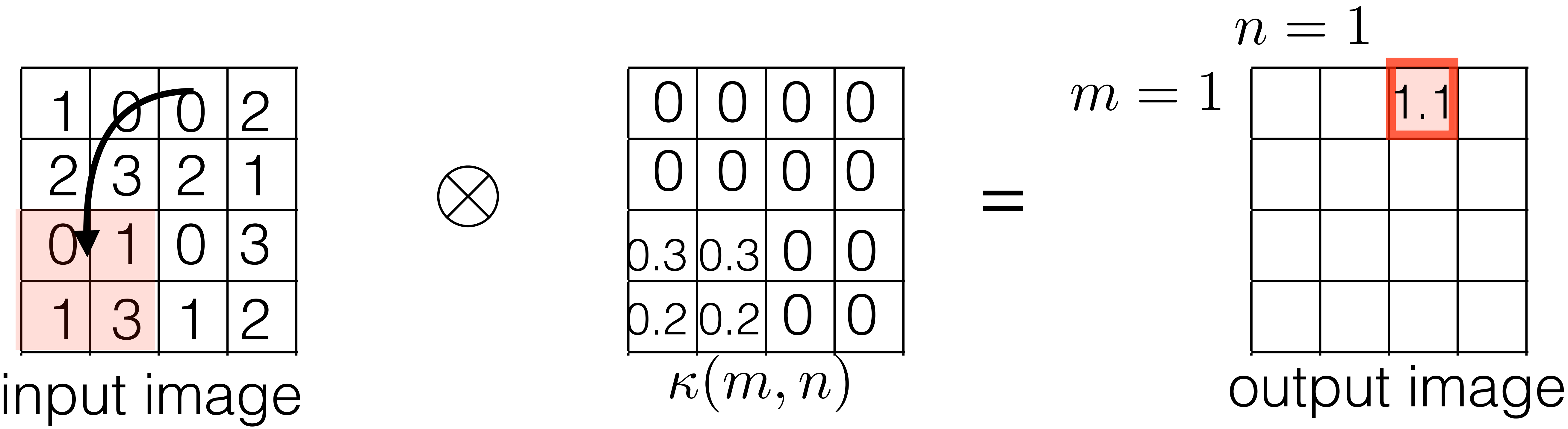
Spatial Transformer networks [Jaderberg 2016]

https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image translation:

What about rotation?



input image

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

$\otimes$

$\kappa(m,n)$

0	0	0	0
0	0	0	0
0.3	0.3	0	0
0.2	0.2	0	0

=

$n = 1$

$m = 1$

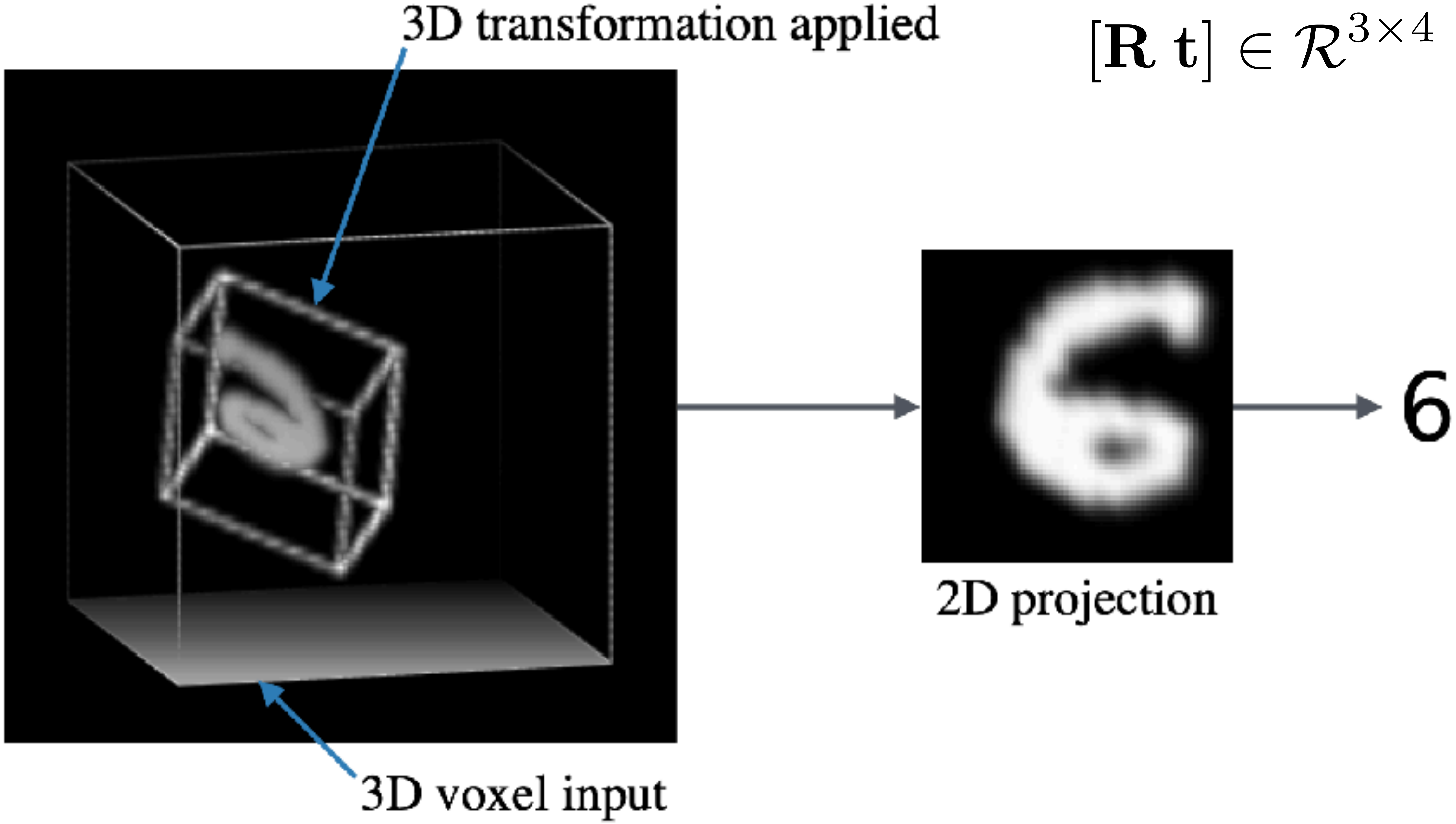
output image

		1.1	

Spatial Transformer networks [Jaderberg 2016] https://arxiv.org/pdf/1506.02025.pdf

Also implemented 3D affine_grid transformation layer:

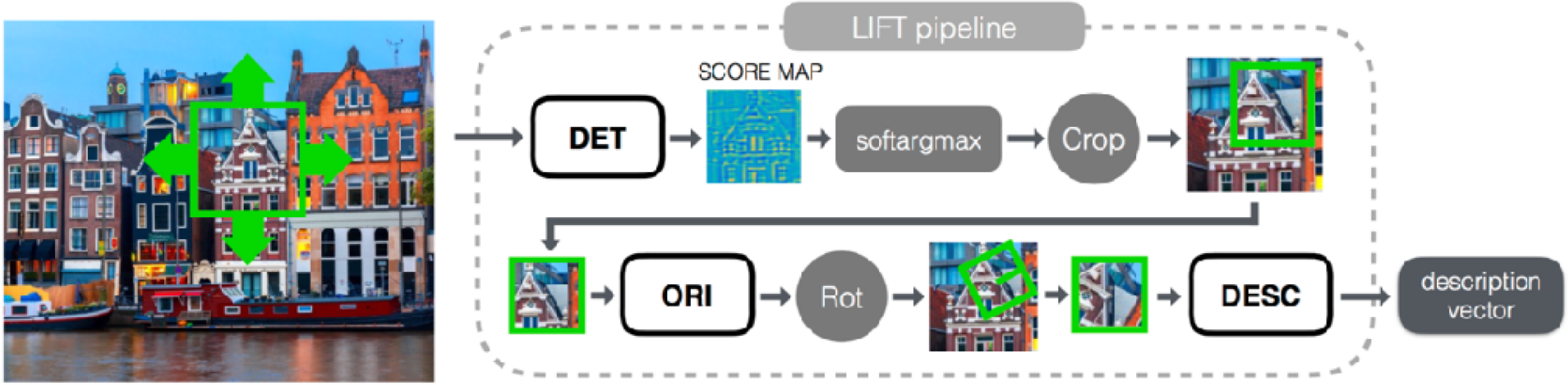
$[\mathbf{R}\ \mathbf{t}] \in \mathcal{R}^{3\times 4}$



Outline

- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Spatial Transformer networks
- Architectures of feature matching networks

LIFT: Learnable Invariant Feature Descriptors [Yi et al ECCV 2016] https://arxiv.org/abs/1603.09114



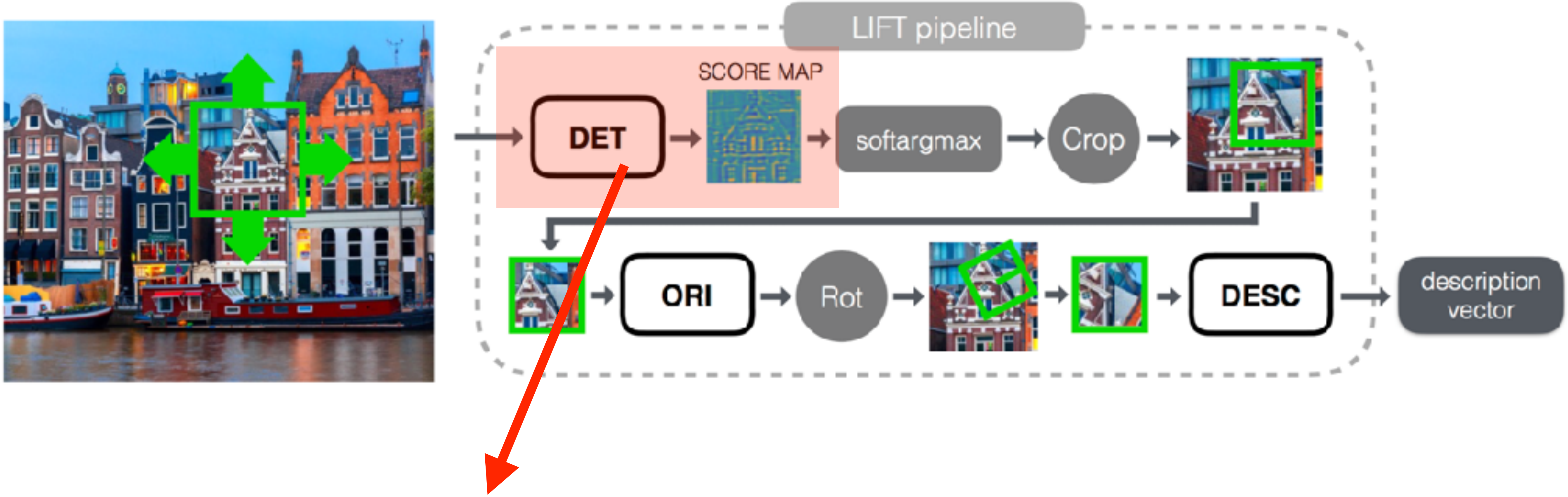
Input: RGB image

Output: set of detected feature points with descriptors

Descriptor is vector which is:

- similar for corresponding points
- and dissimilar for not corresponding points.

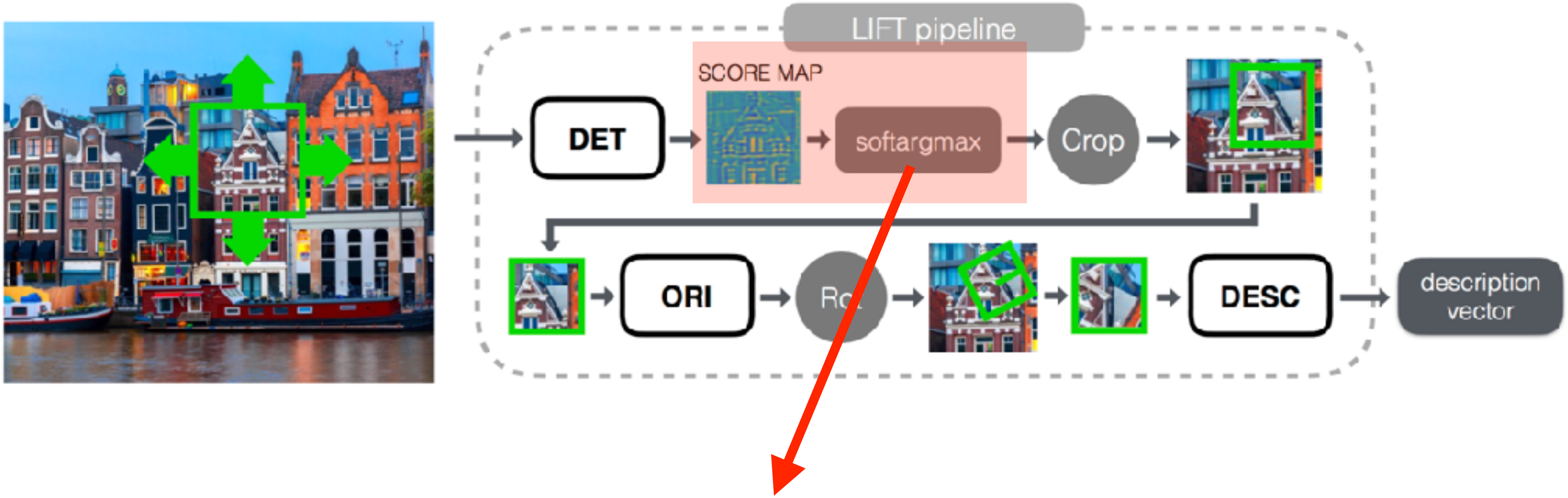
LIFT: Learnable Invariant Feature Descriptors [Yi et al ECCV 2016] https://arxiv.org/abs/1603.09114

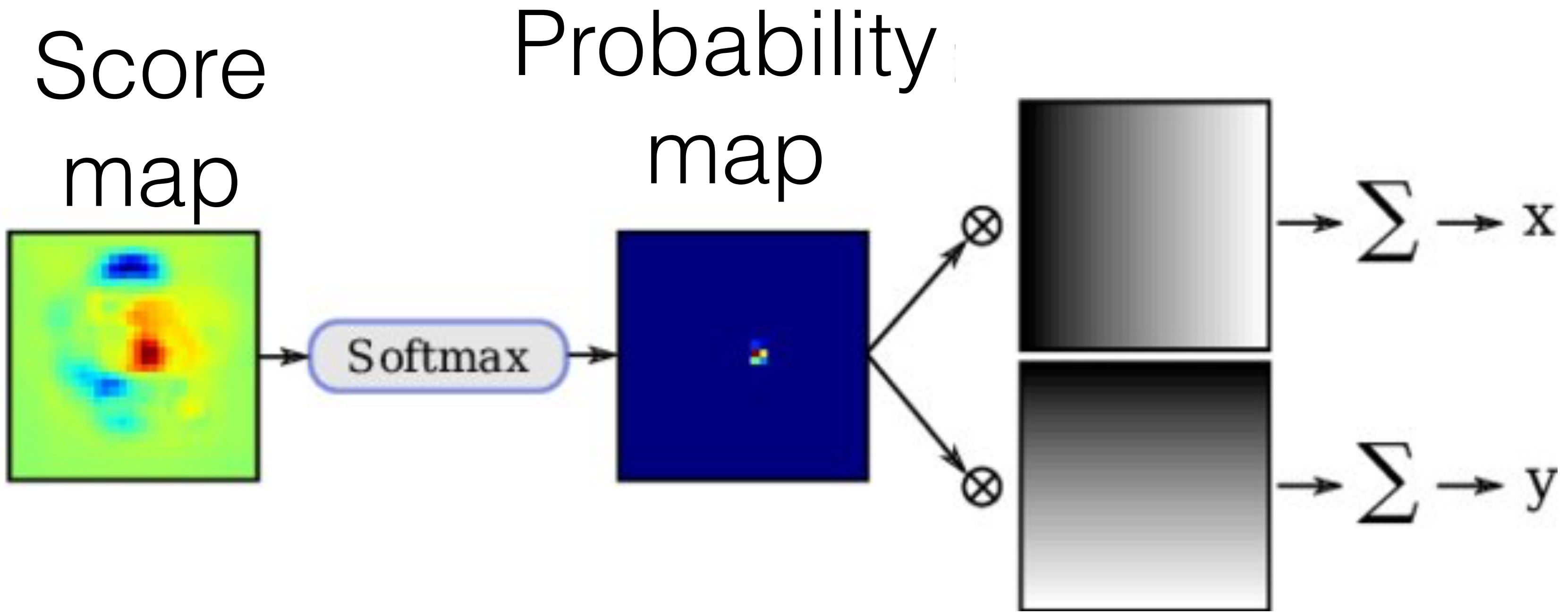


Segmentation CNN for pixel-wise two-class labelling

- class 1: “suitable feature point”
- class 2: “unsuitable feature point”

LIFT: Learnable Invariant Feature Descriptors [Yi et al ECCV 2016] https://arxiv.org/abs/1603.09114

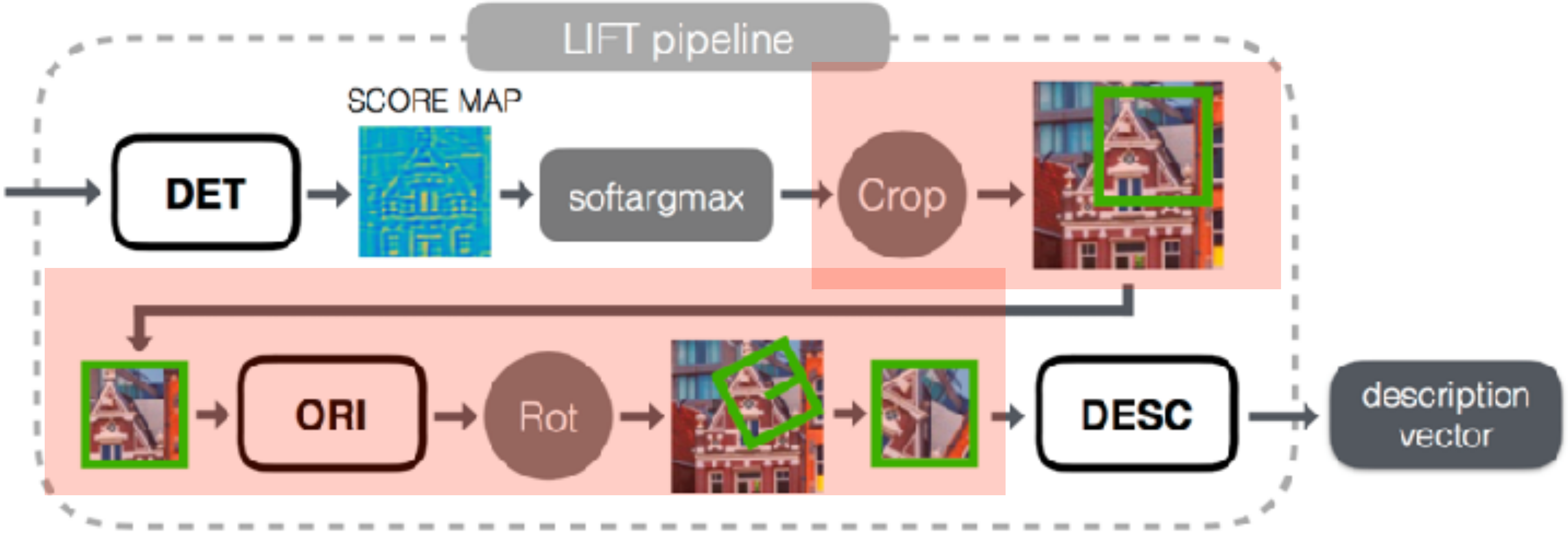




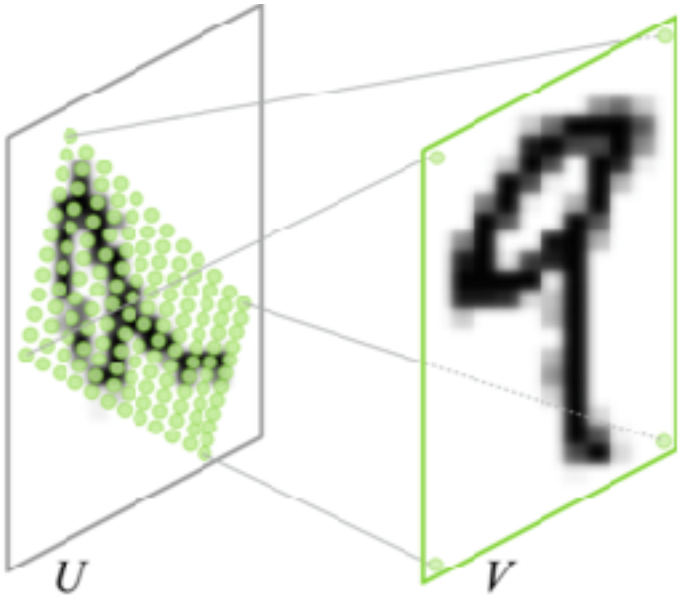
Expected value of x and y under the prob. distr.

Learnable Invariant Feature Descriptor (LIFThttps://www.researchgate.net/publication/323410987_2D3D_Pose_Estimation_and_Action_Recognition_using_Multitask_Deep_Learning/figures?lo=1)

LIFT: Learnable Invariant Feature Descriptors [Yi et al ECCV 2016] https://arxiv.org/abs/1603.09114



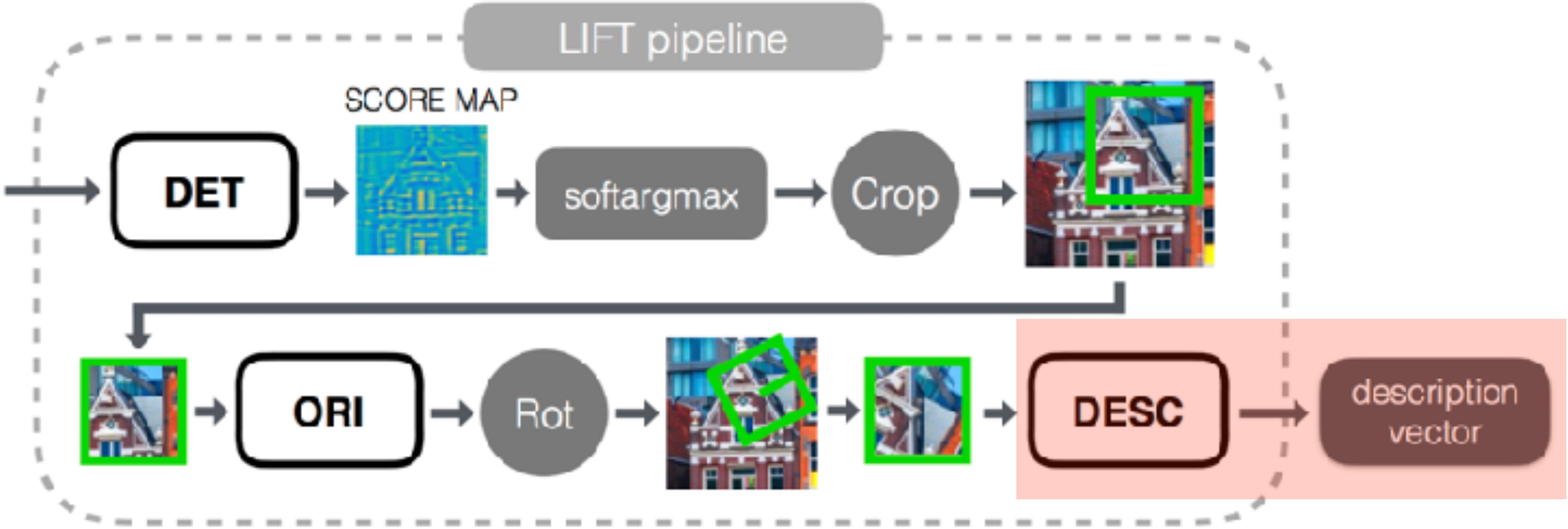
Spatial Transformer Network



Bilinear approximation of affine transformation is differentiable !

[Jaderberg, 2016] https://arxiv.org/pdf/1506.02025.pdf

LIFT: Learnable Invariant Feature Descriptors [Yi et al ECCV 2016] https://arxiv.org/abs/1603.09114



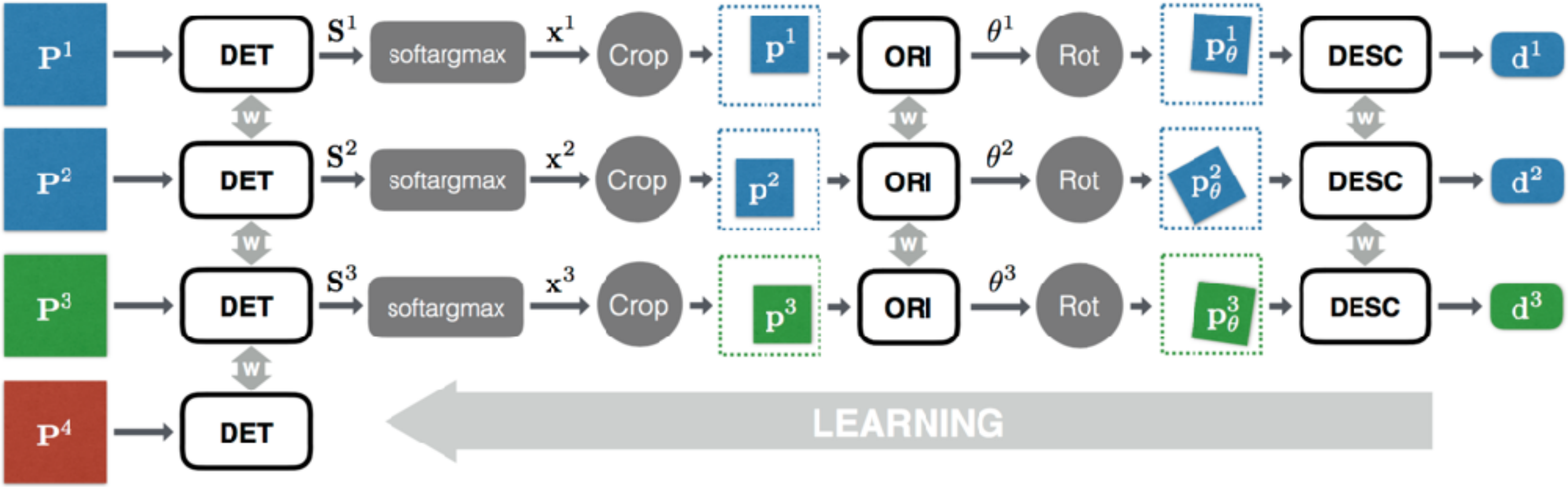
- Trained in end-to-end manner
- Ground truth correspondences for training obtained from SfM and webcameras
- Training set consists of four-touples:

P^1 P^2 Two corresponding patches on distinctive points

P^3 One not corresponding patch on a distinctive point

P^4 One patch on a not distinctive point

LIFT: Learnable Invariant Feature Descriptors [Yi et al ECCV 2016] https://arxiv.org/abs/1603.09114



- All patches are fed into the network and differentiable loss
- Loss makes:
  - d1 and d2 as close as possible,
  - d3 as far as possible (from d1 and d2)
  - DET to have high response on p1,p2,p3 and small on p4

Summary architectures

- Deeper architectures, with many small kernels with skip-connections (e.g. ResNet, DenseNet) seems reasonable
- Decreasing the spatial resolution while increasing spatial resolution allows to exploit context.
- Atrous spatial pyramid seems to be viable replacement for max-pooling
- Argmax is not differentiable, but it can be replaced by expected value.
- Any affine transformation can be tackled by Spatial Transform Layer
- Divide and Conquer strategy with as many as possible auxiliary losses seems to work well on many problems
- A lot of dark-magic needed for successful training