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
<u>Steel drum</u>
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

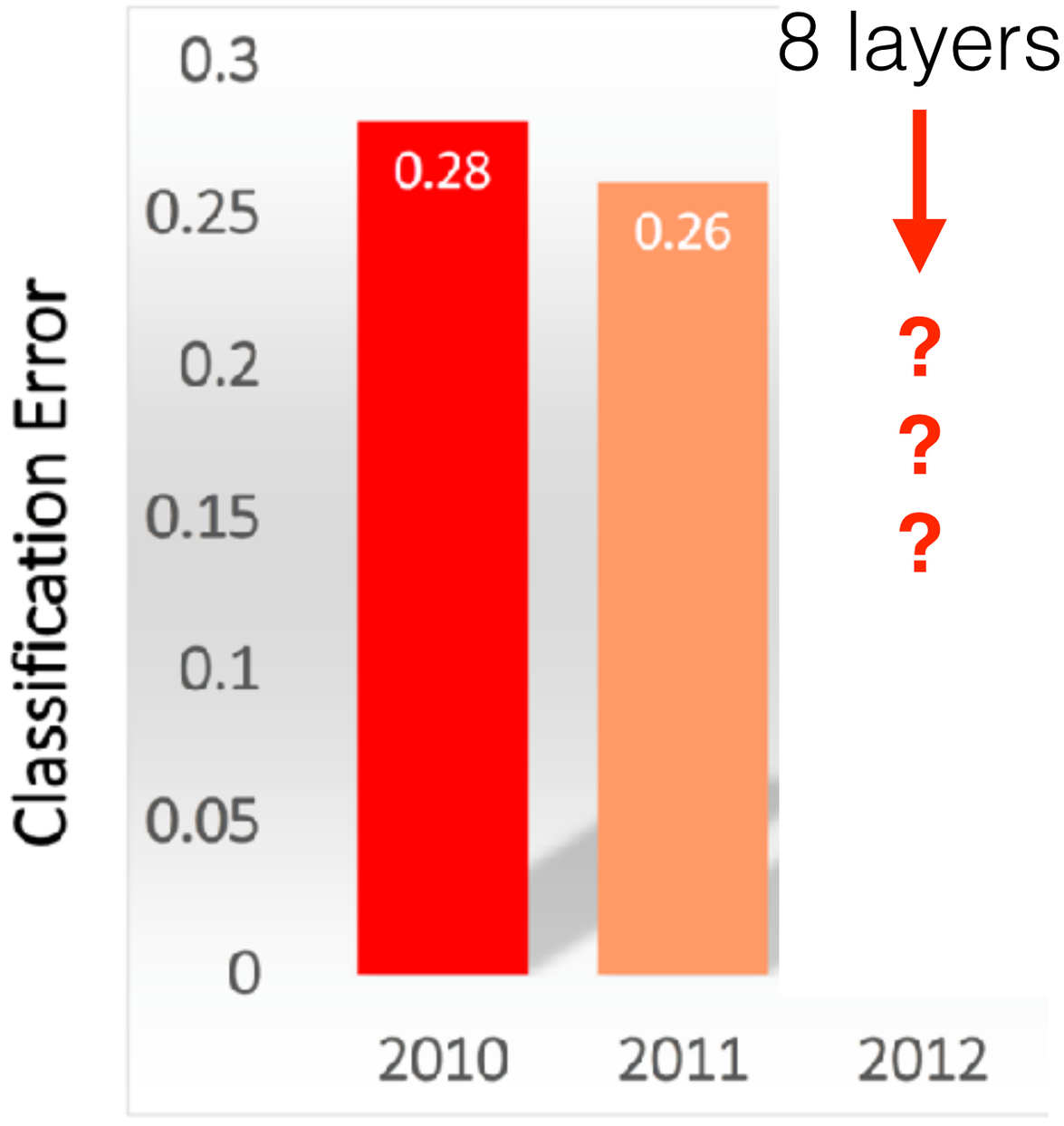
✗

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

IMAGENET

Classification results

AlexNet
8 layers

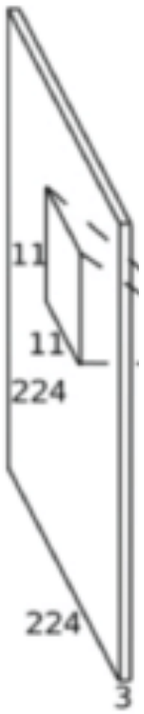


?

?

?

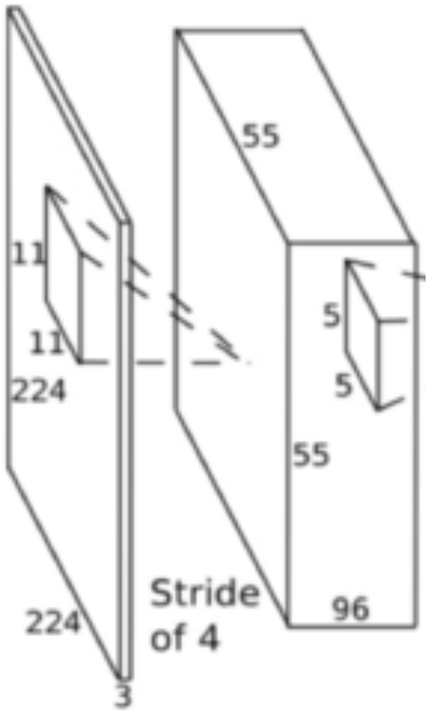
AlexNet on ImageNet 2012 (**over 27k citations !!!**)



- 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

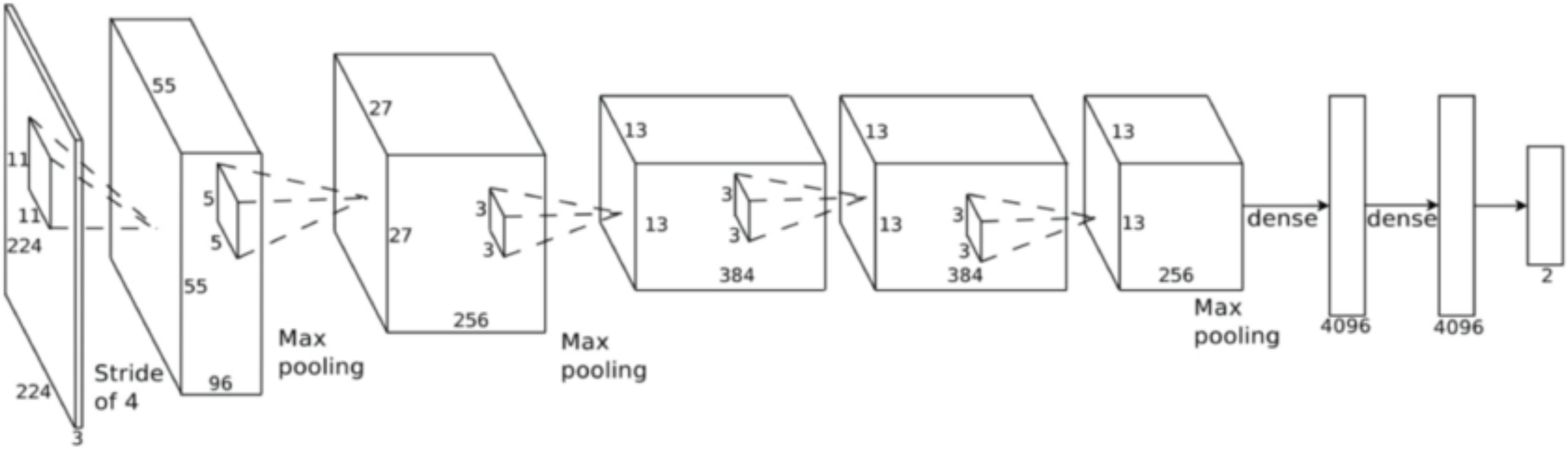
AlexNet on ImageNet 2012 (**over 27k citations !!!**)



- 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 !!!**)



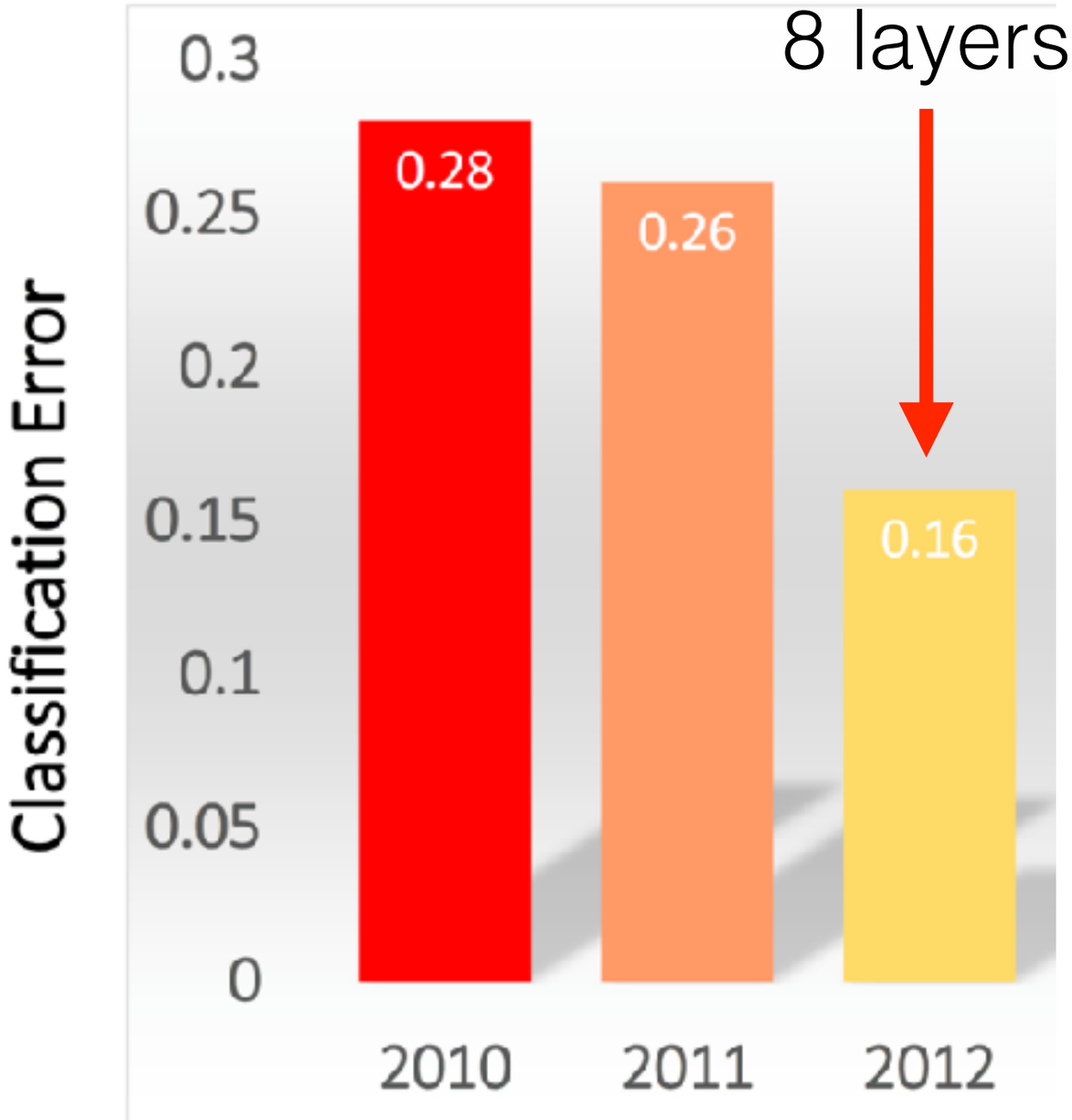
- Param in layer1 (conv, 96 11x11 filters, stride=4, pad=0)?
- Param in layer2 (maxp,3x3 filters, stride=2, pad=0)?
- Param in layer3 (conv, 256 5x5 filters, stride=1, pad=2?
- Parameters in total: 60M, Depth: 8 layers

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

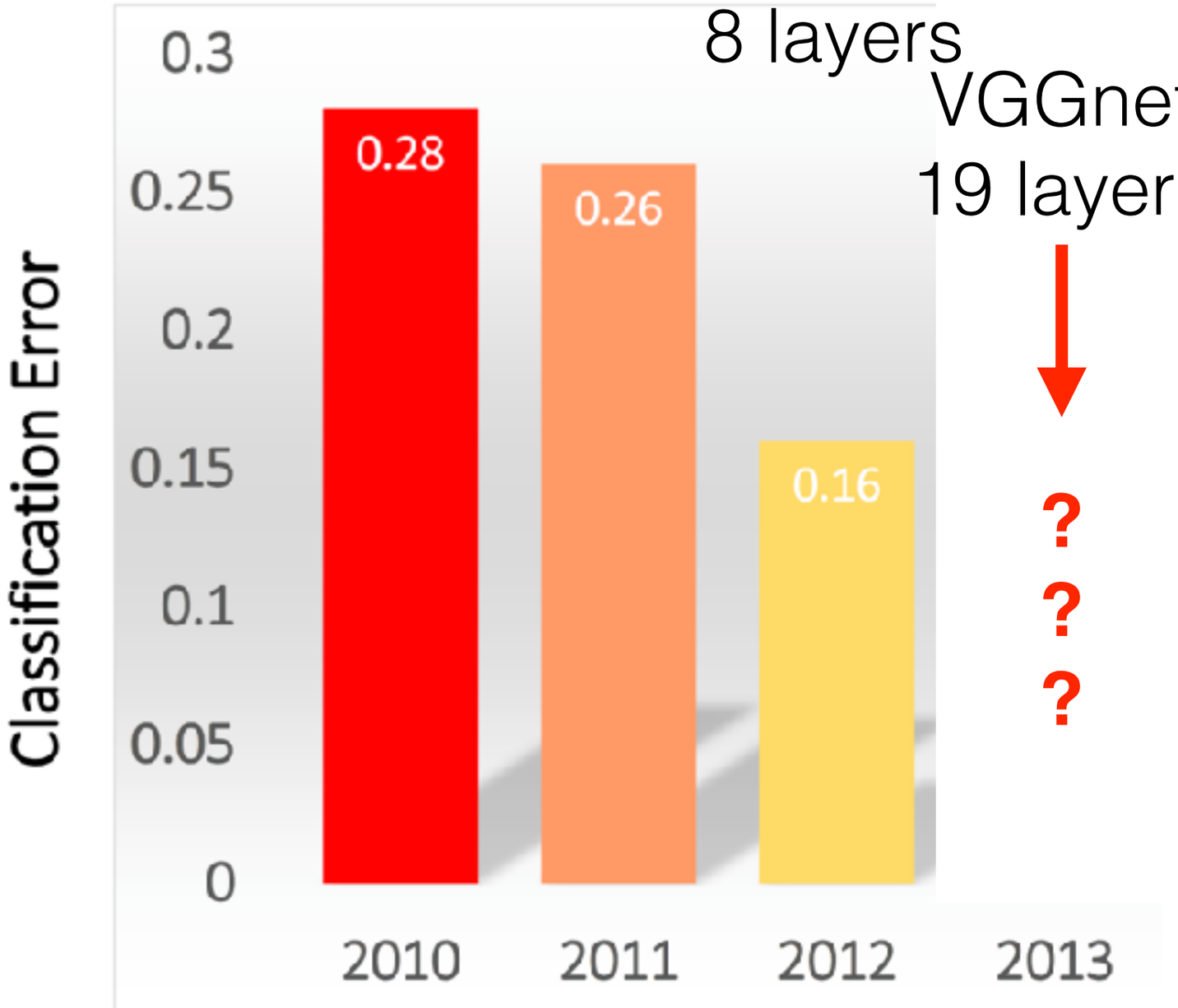


IMAGENET

Classification results

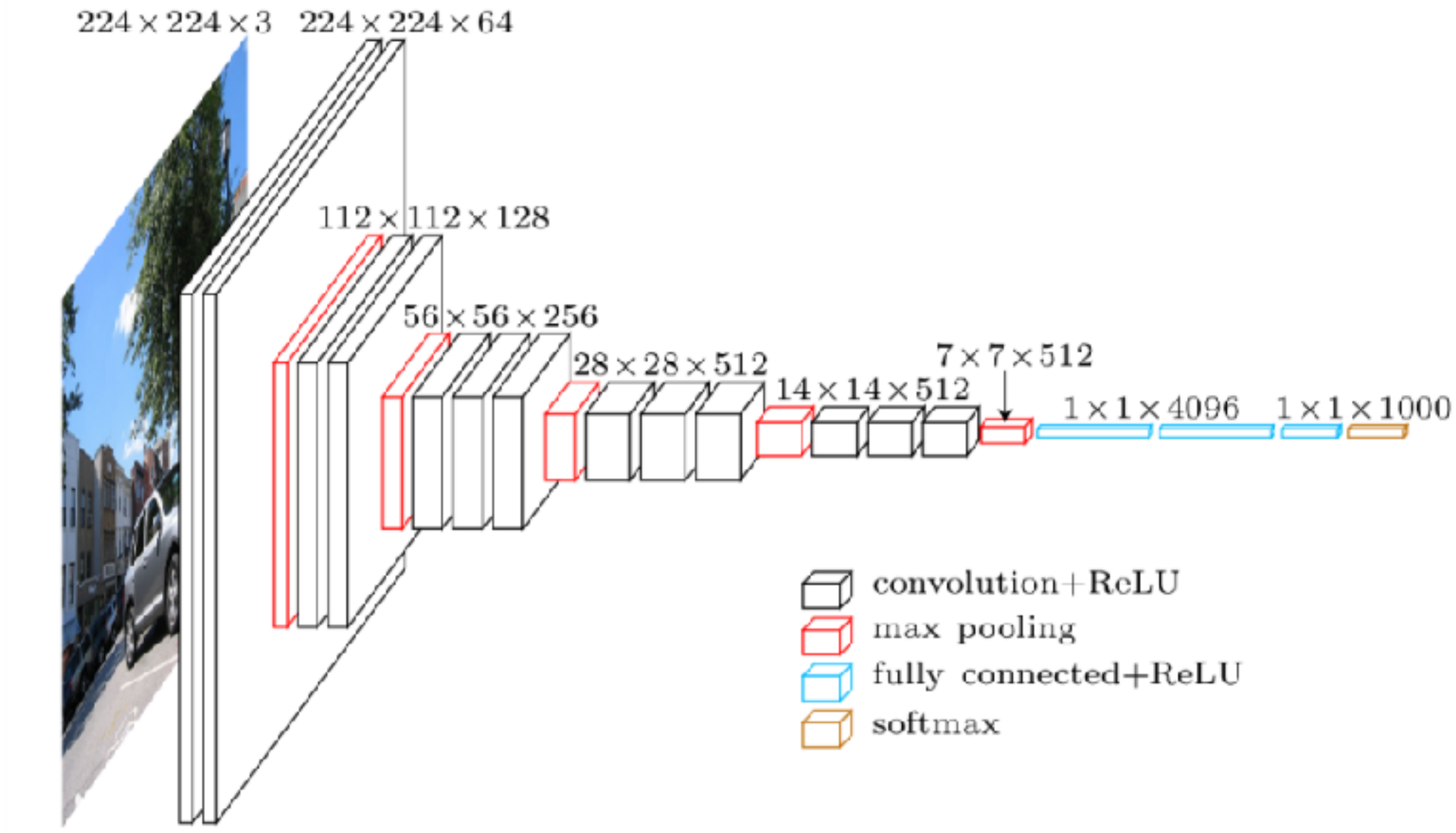
AlexNet
8 layers

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

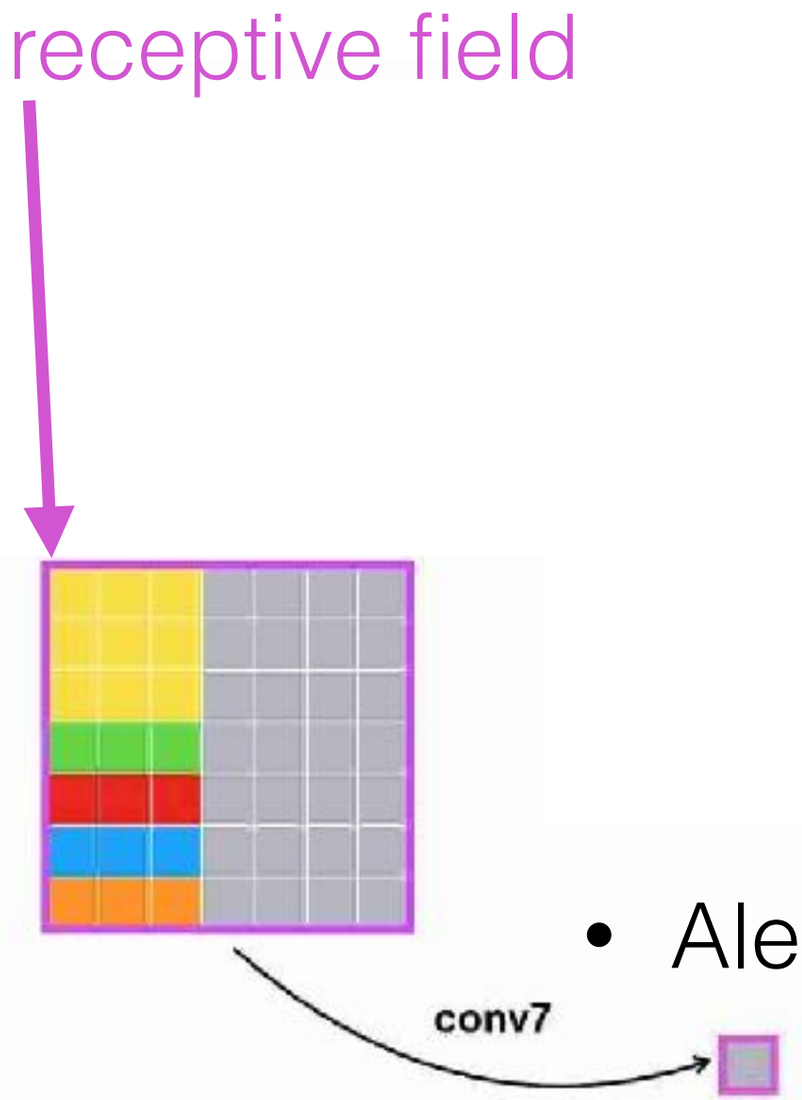
VGGNet vs AlexNet

- 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

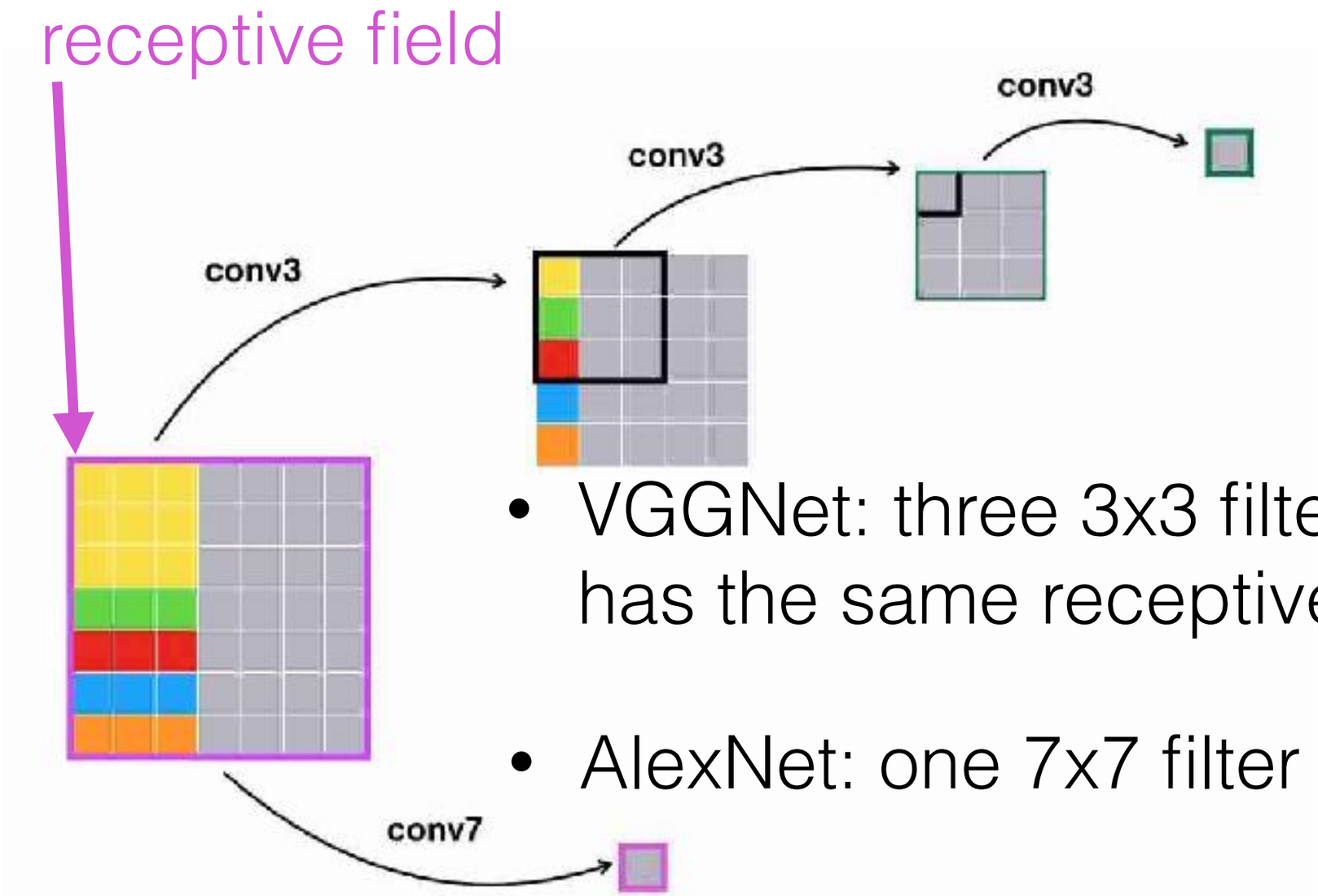
VGGNet vs AlexNet



- 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

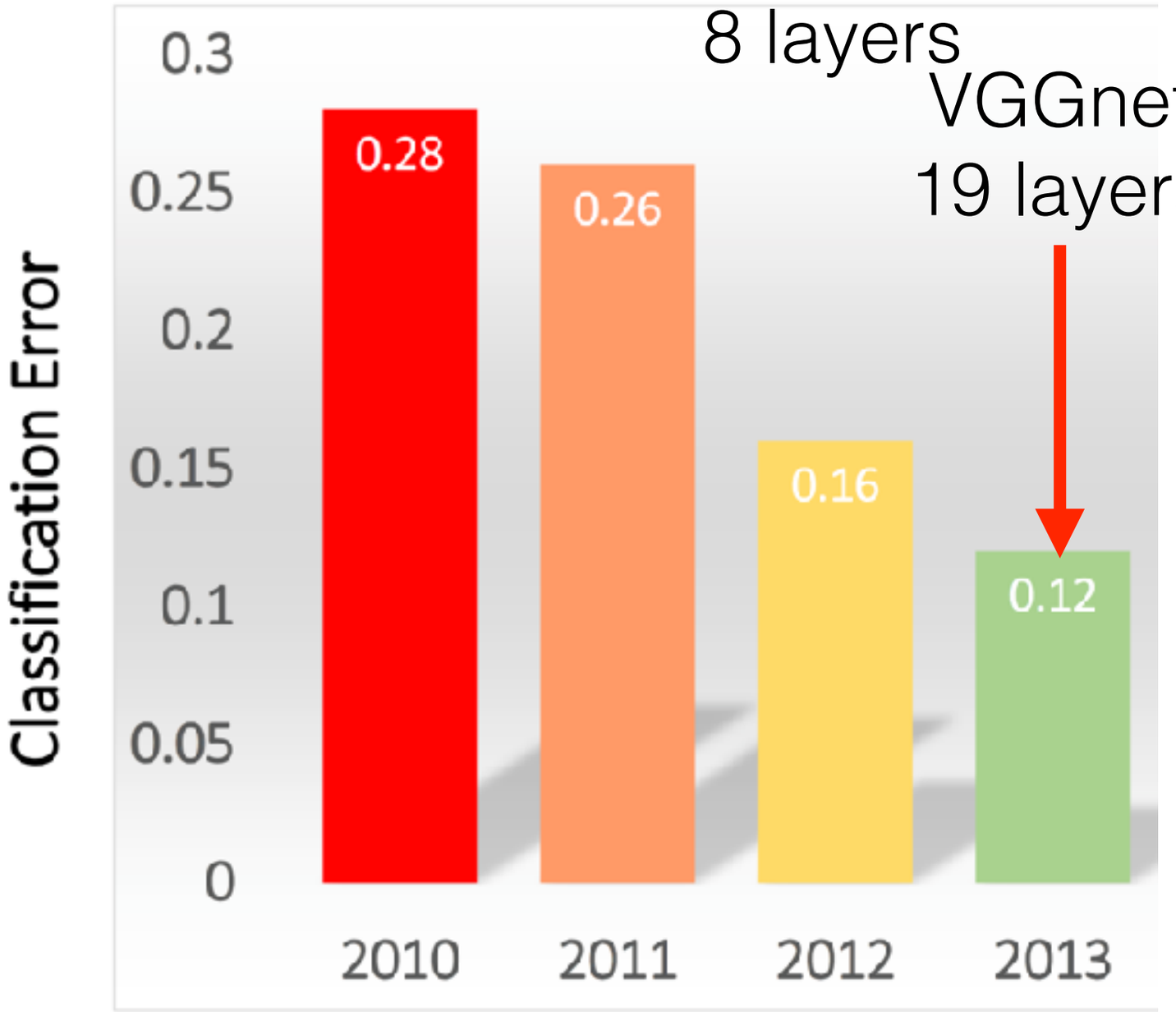


- VGGNet: three 3x3 filters (3x9+3 params) has the same receptive field
- AlexNet: one 7x7 filter (49+1 params)

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

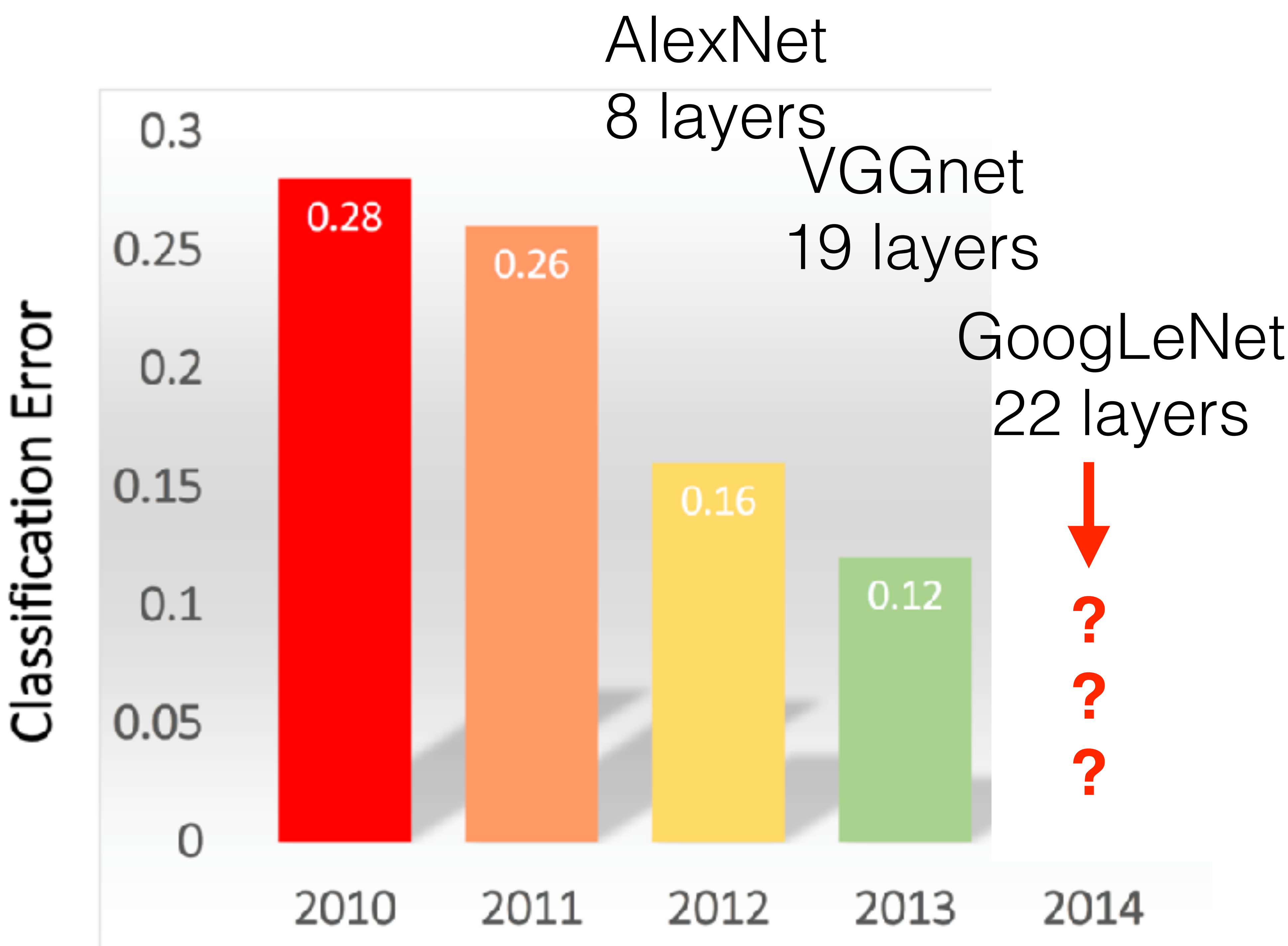
IMAGENET

Classification results

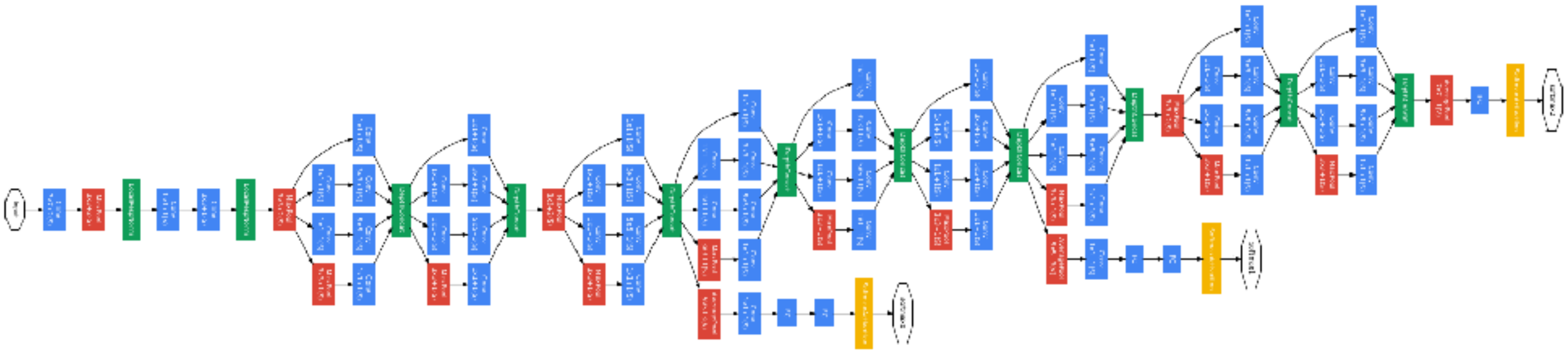


IMAGENET

Classification results

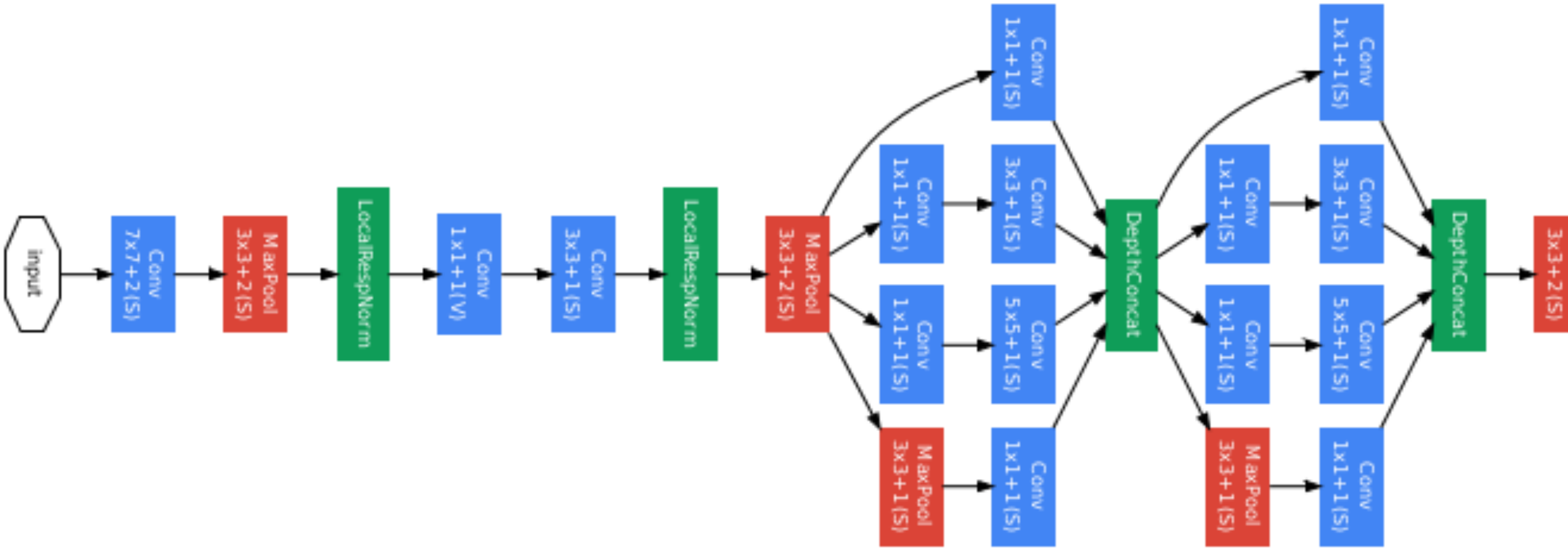


GoogLeNet



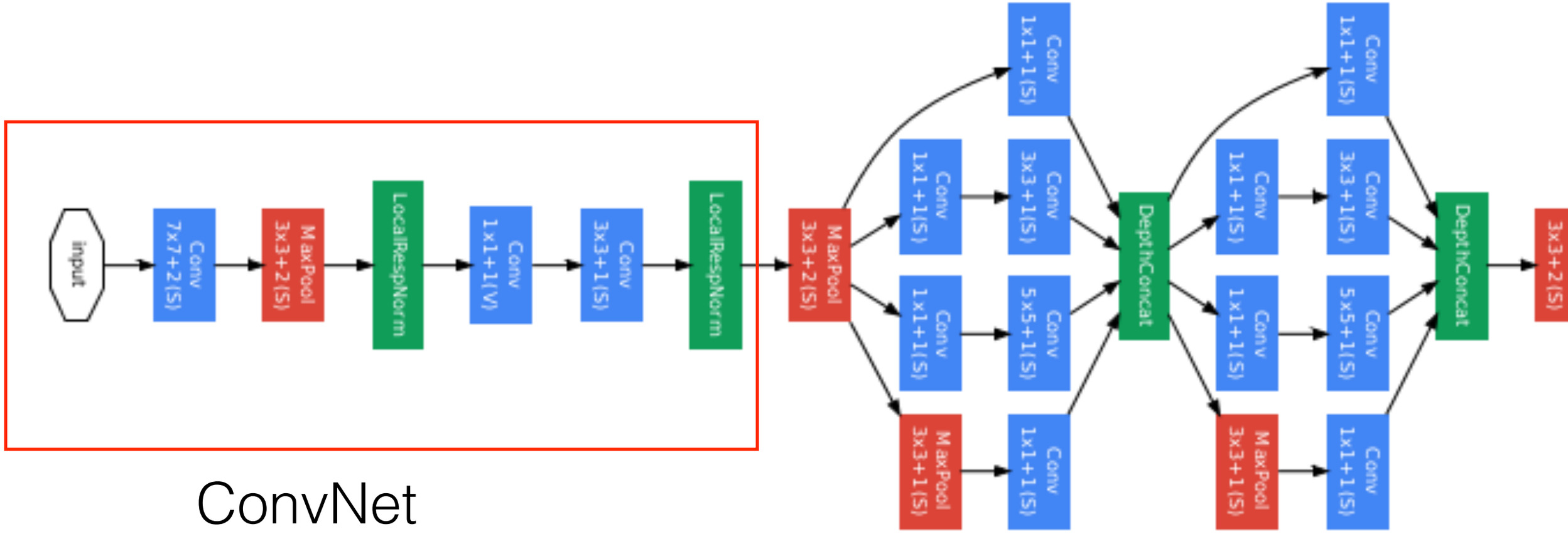
Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

GoogLeNet



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

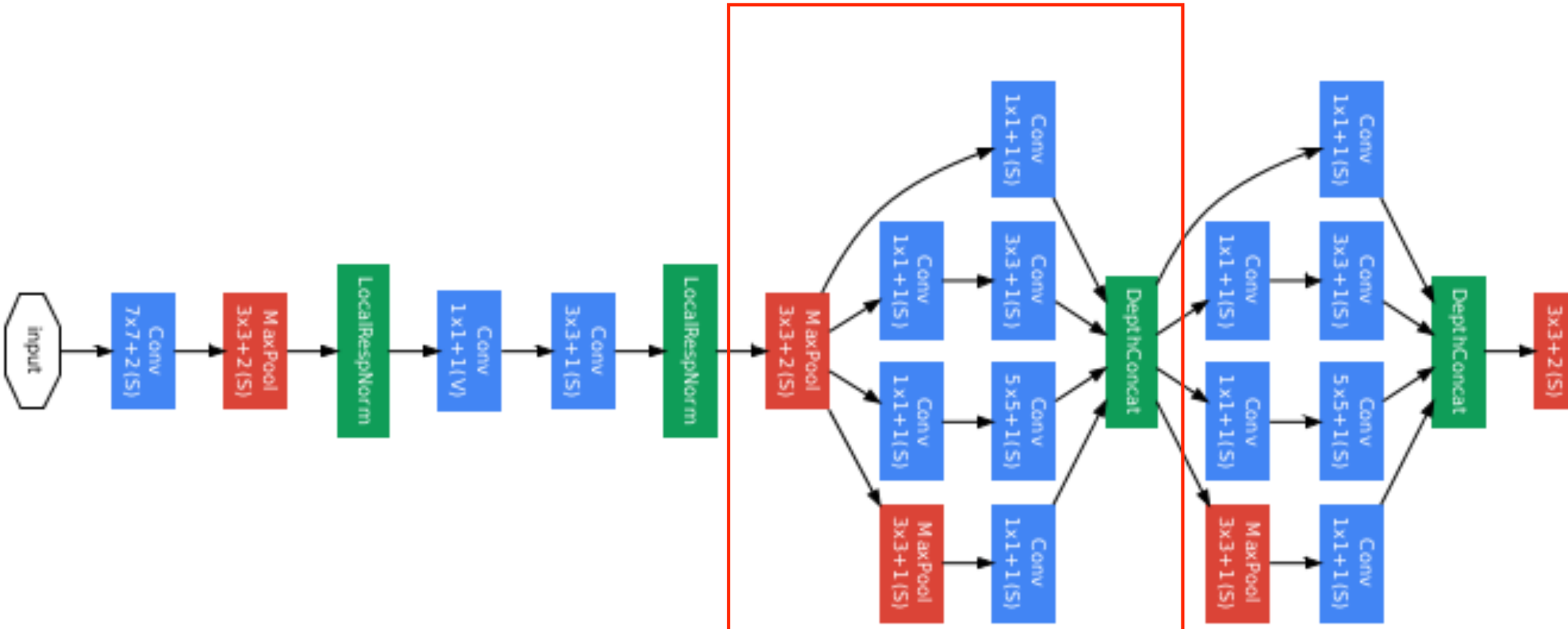
GoogLeNet



ConvNet

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

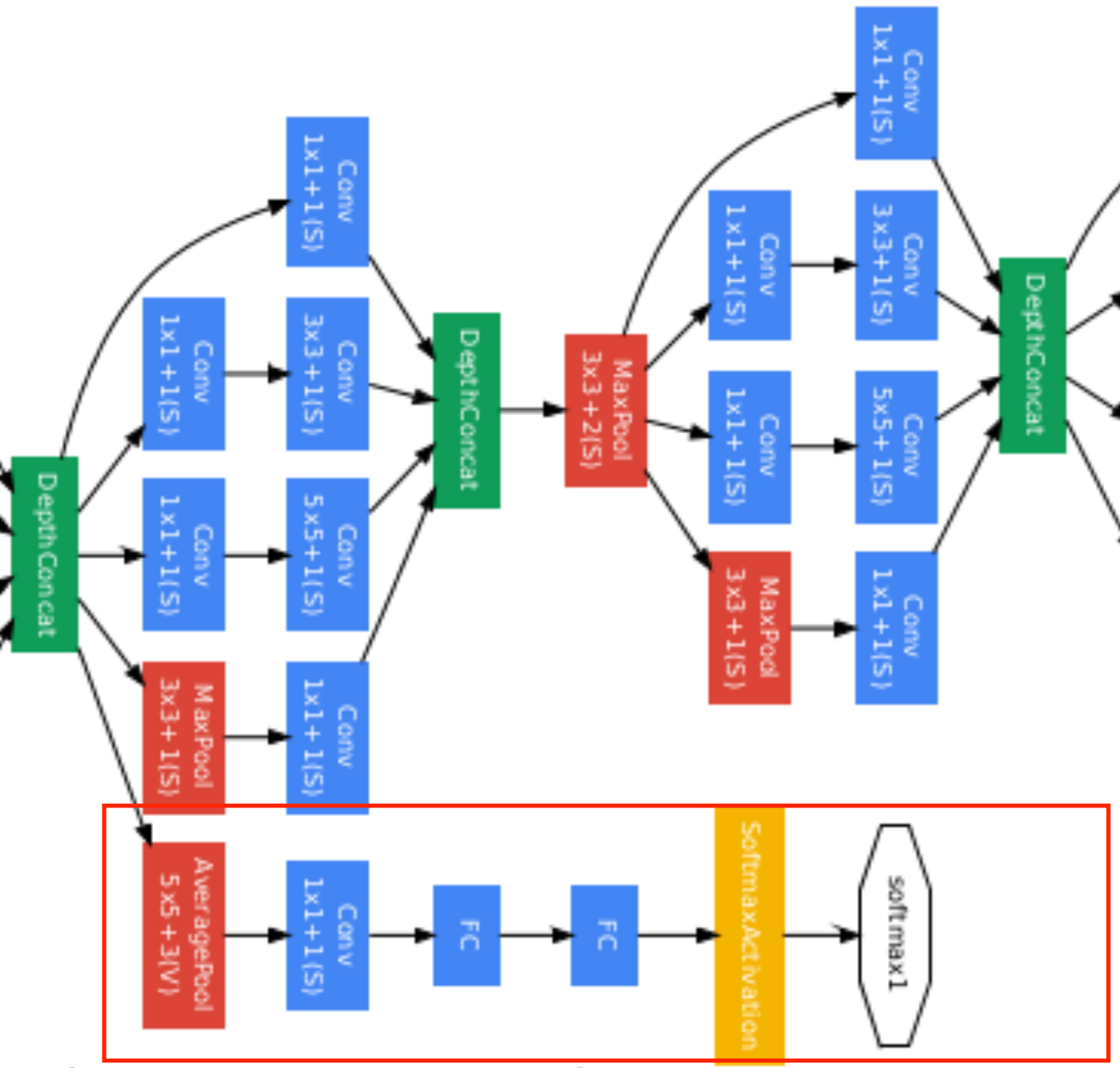
GoogLeNet



Inception module

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

GoogLeNet

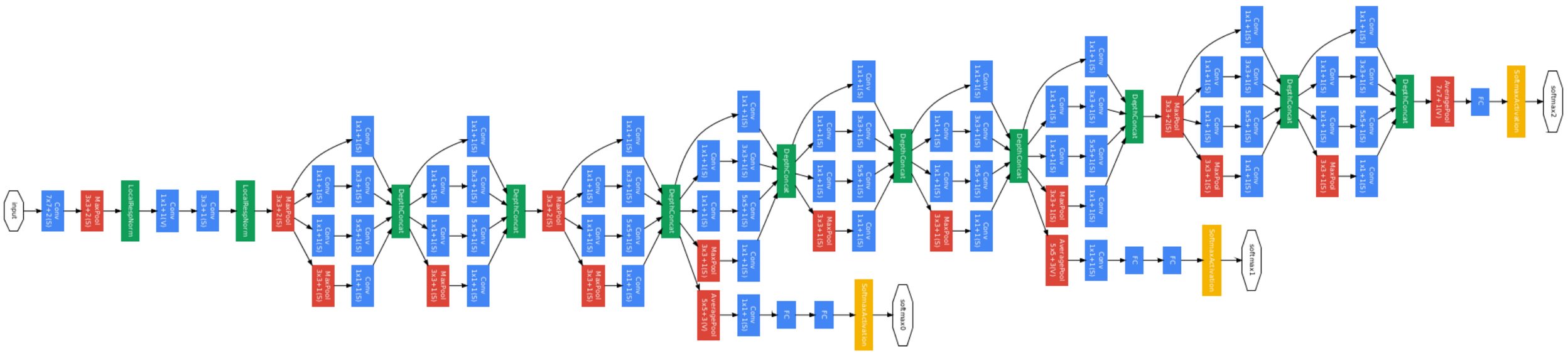


Additional loss layer which injects the gradient inside

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

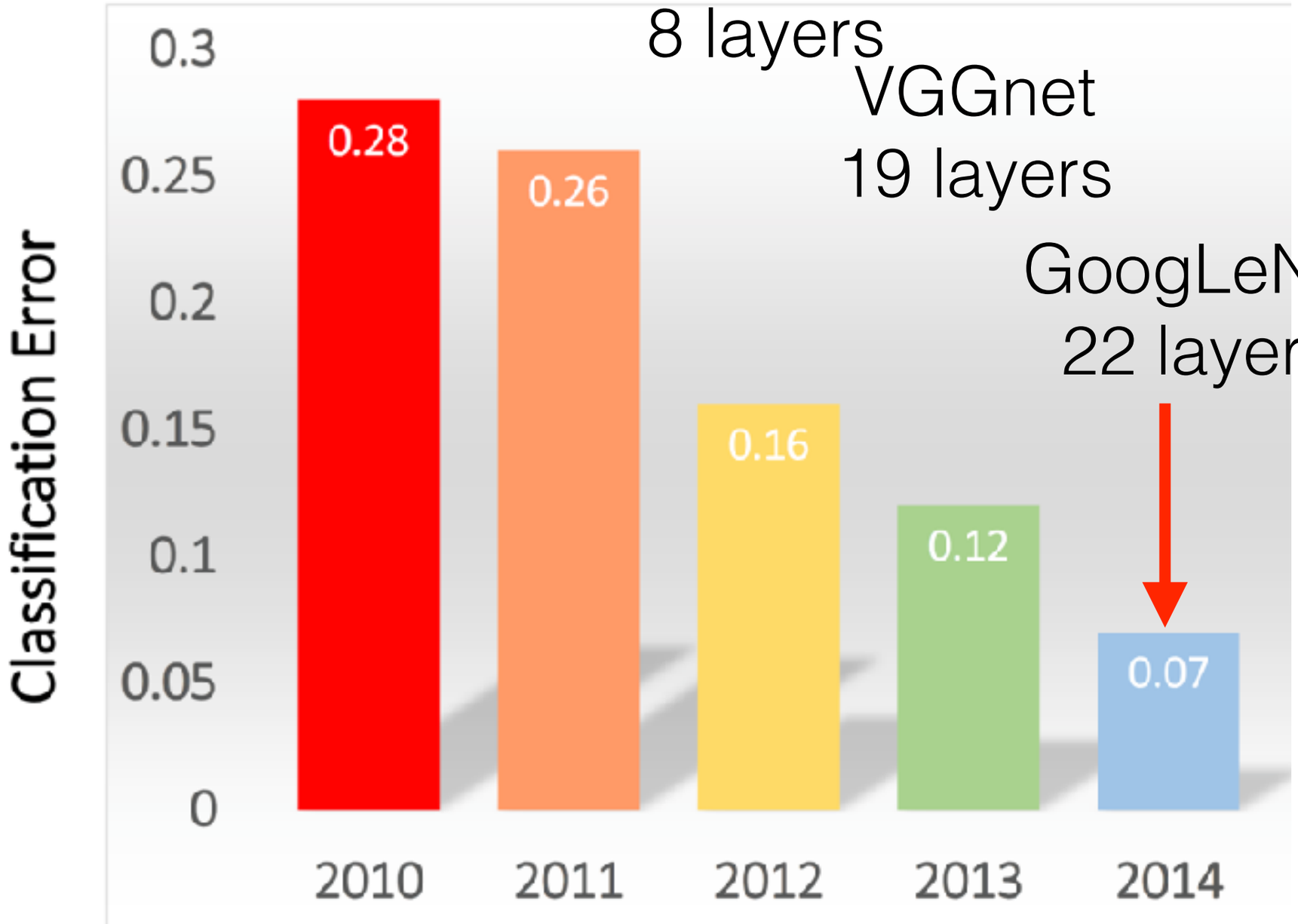
IMAGENET

Classification results

AlexNet
8 layers

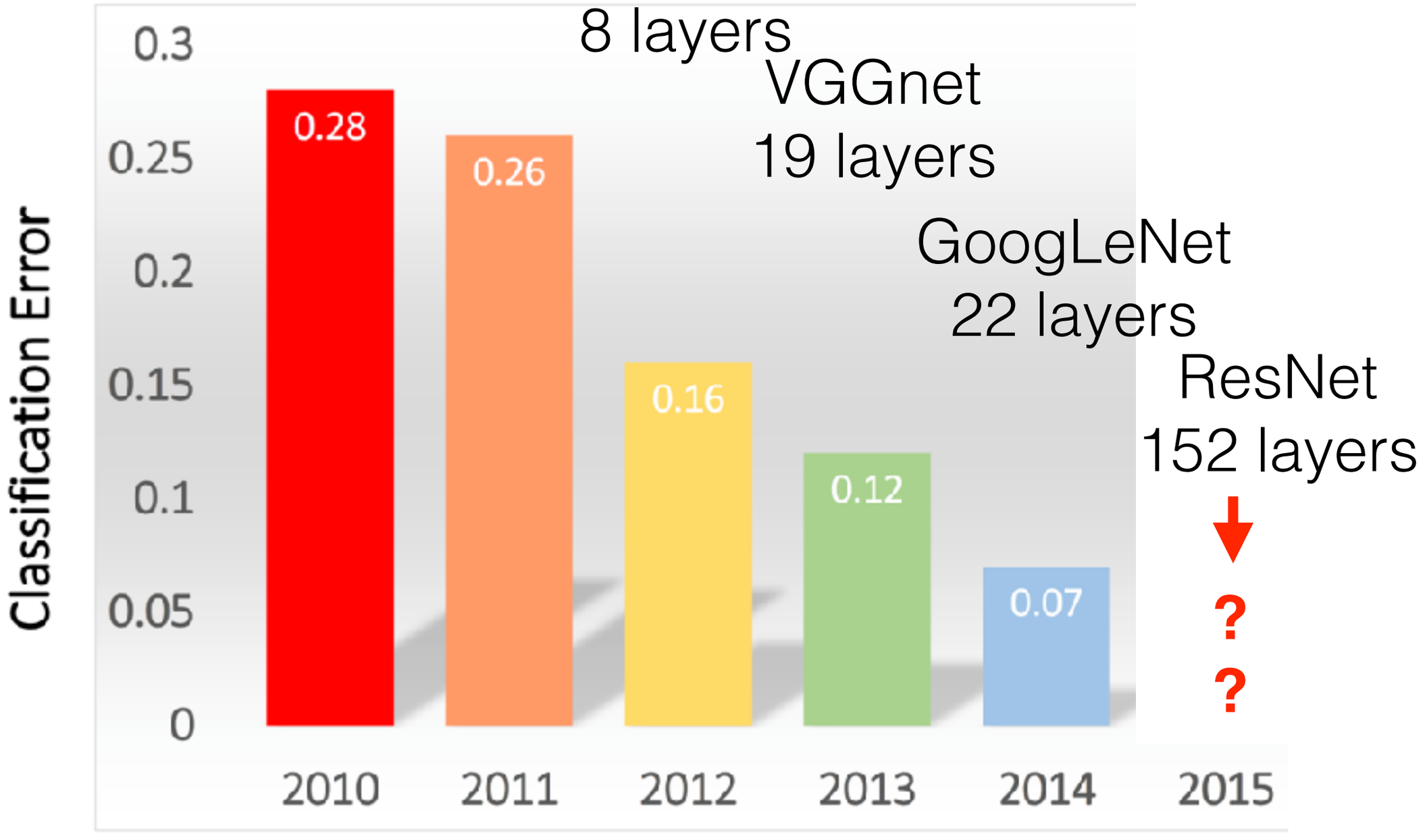
VGGnet
19 layers

GoogLeNet
22 layers



IMAGENET

Classification results



AlexNet
8 layers

VGGnet
19 layers

GoogLeNet
22 layers

ResNet
152 layers

?
?

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)

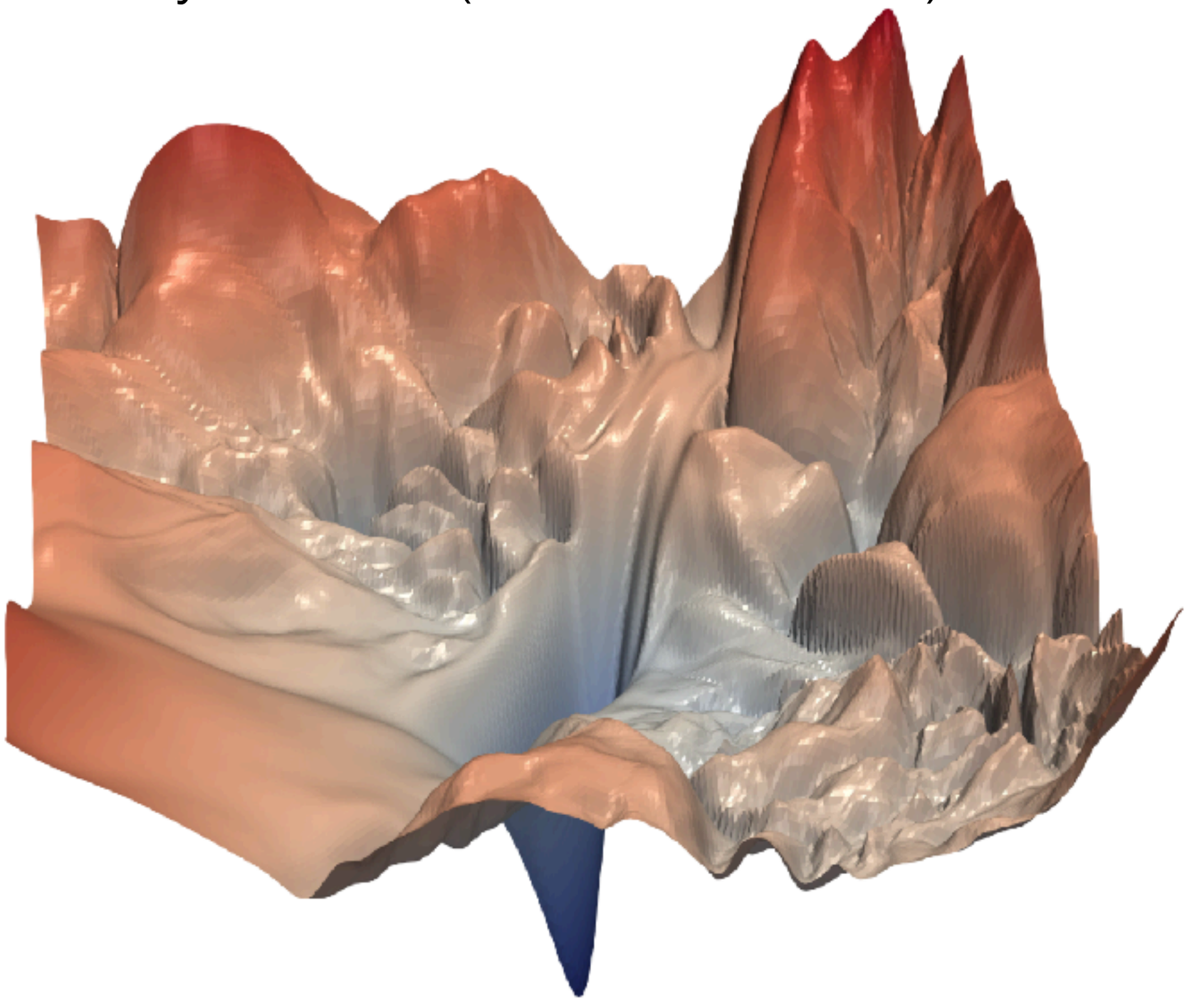
He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385

Visualizing Loss Landscape of Neural Nets

[Li et al, NIPS, 2018] https://arxiv.org/pdf/1712.09913.pdf

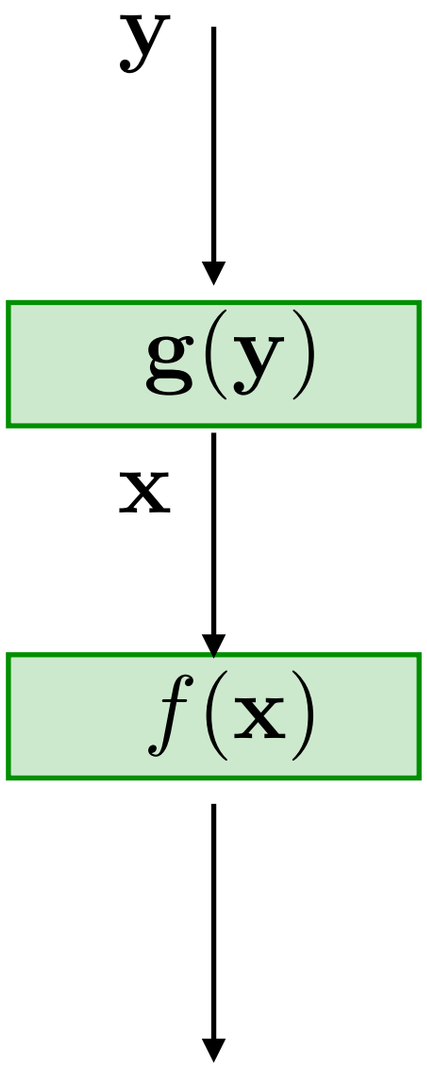
$$f(\alpha, \beta) = \mathcal{L}(\mathbf{w}^* + \alpha\mathbf{u} + \beta\mathbf{v})$$

for randomly chosen (and normalized) directions $\mathbf{u}, \mathbf{v}$



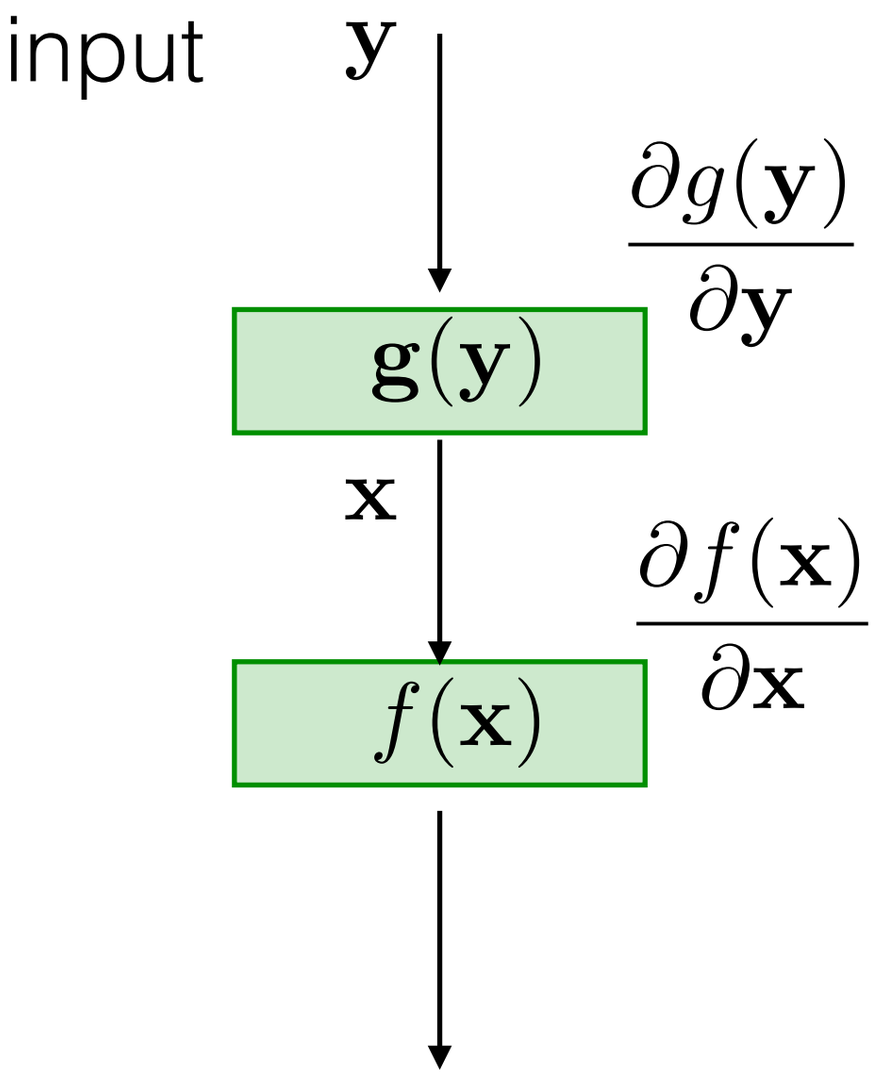
forward pass

input $\mathbf{y}$



output $\mathbf{z} = f(g(\mathbf{y}))$

forward pass:



output $\mathbf{z} = f(g(\mathbf{y}))$

backward pass:

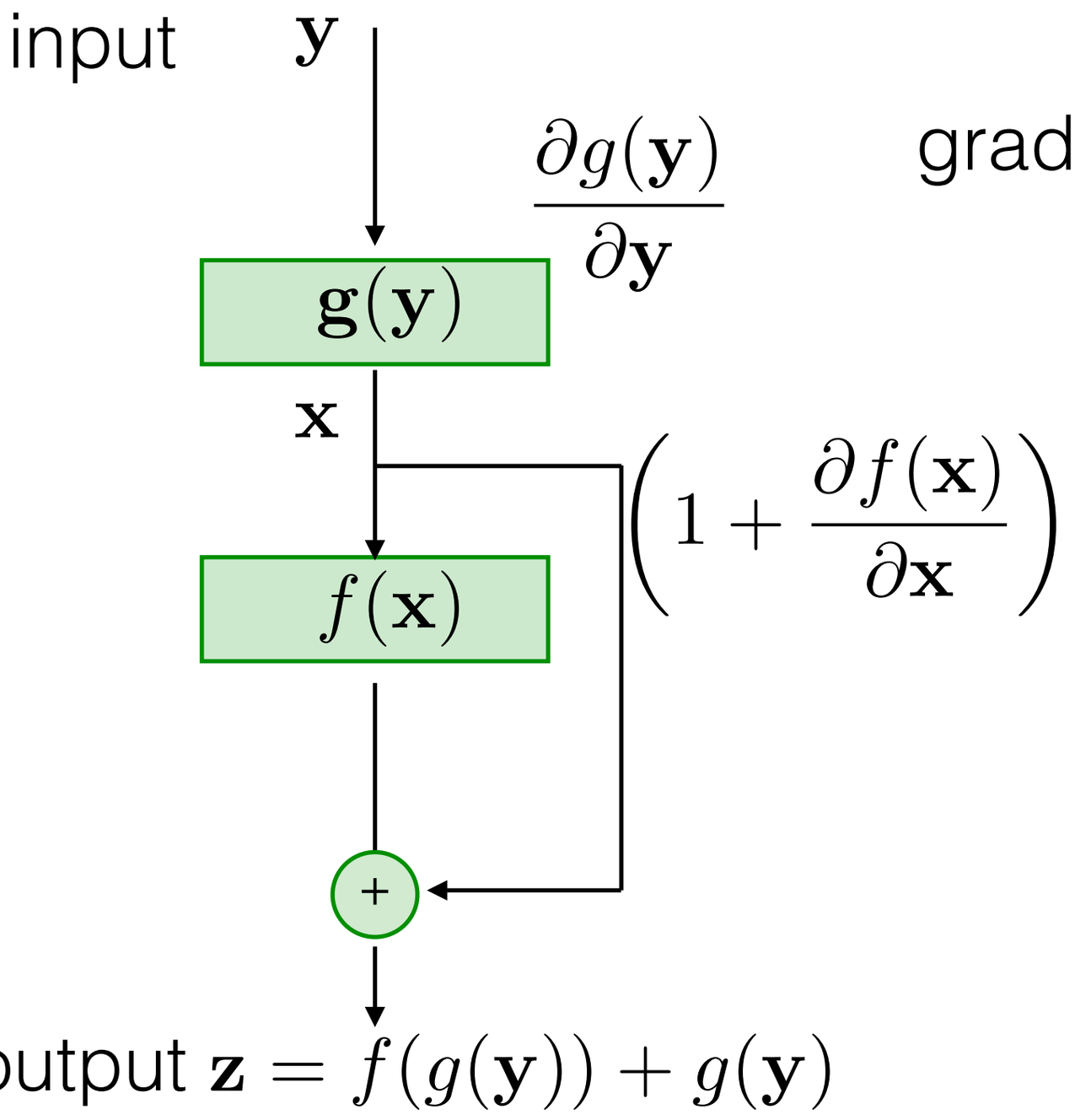
gradient $\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial g(\mathbf{y})}{\partial \mathbf{y}}$

if $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \approx \mathbf{0}$

then gradient is always zero

$$\frac{\partial \mathbf{z}}{\partial \mathbf{y}} \approx \mathbf{0}$$

forward pass:



backward pass:

gradient $\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = \left(1 + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right) \frac{\partial g(\mathbf{y})}{\partial \mathbf{y}}$

if $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \approx \mathbf{0}$

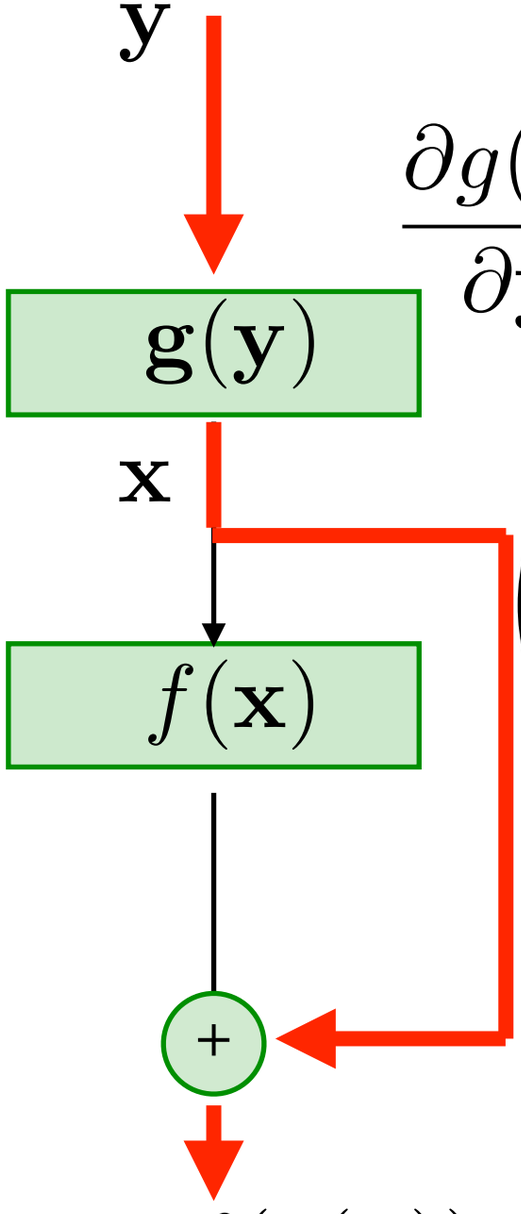
then gradient is

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

forward pass:

backward pass:

input $\mathbf{y}$



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

$$\frac{\partial g(\mathbf{y})}{\partial \mathbf{y}} \left(1 + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right)$$

gradient $$\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = \left(1 + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right) \frac{\partial g(\mathbf{y})}{\partial \mathbf{y}}$$

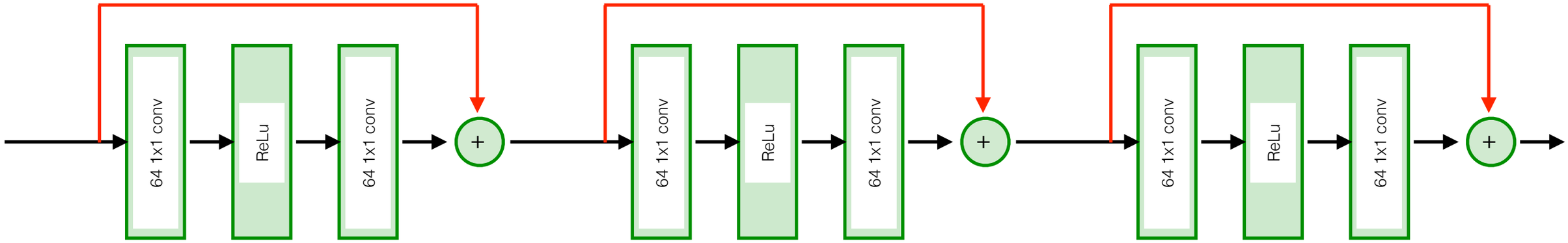
if $$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \approx \mathbf{0}$$

then gradient is

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

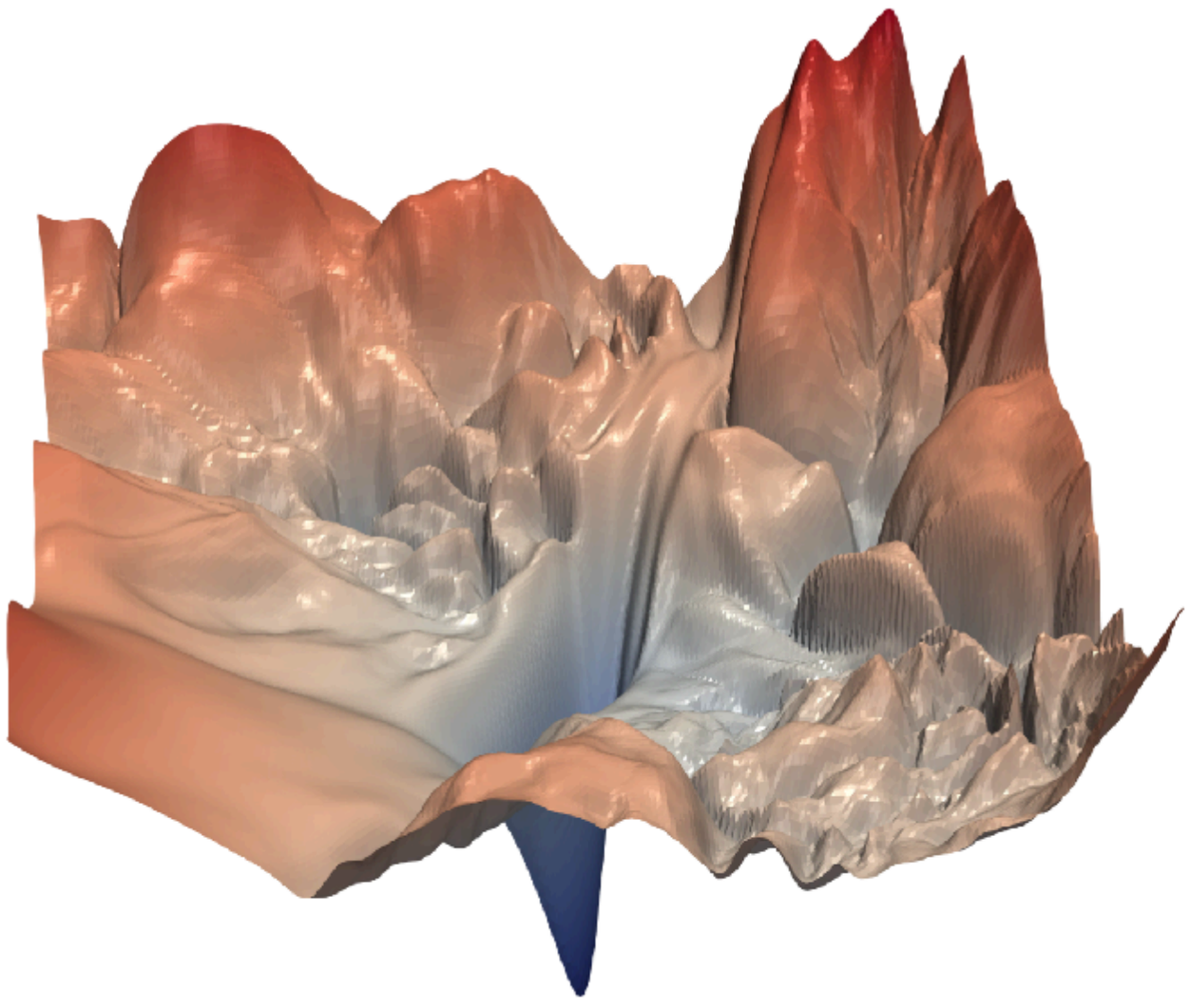
Input values can still influence the output via skip connection !

ResNet - gradient flow

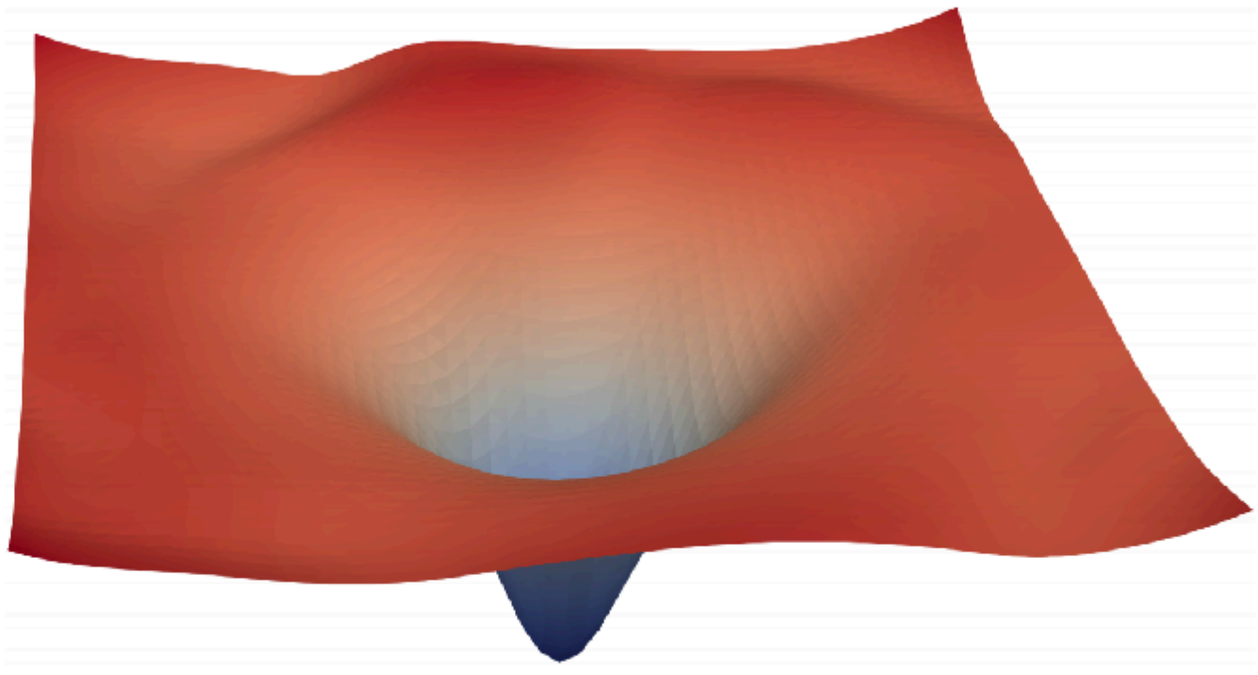


- Skip connections partially avoids diminishing gradient
- The weights from the beginning of the net has strong influence on the output! (via red skip-connections)

Visualizing Loss Landscape of Neural Nets
[Li et al, NIPS, 2018] https://arxiv.org/pdf/1712.09913.pdf



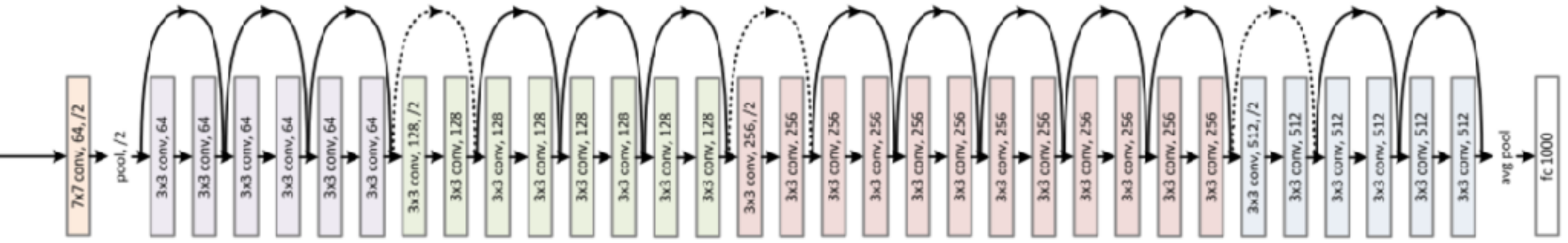
(a) without skip connections



(b) with skip connections

ResNet: deep ConvNet with skip connections

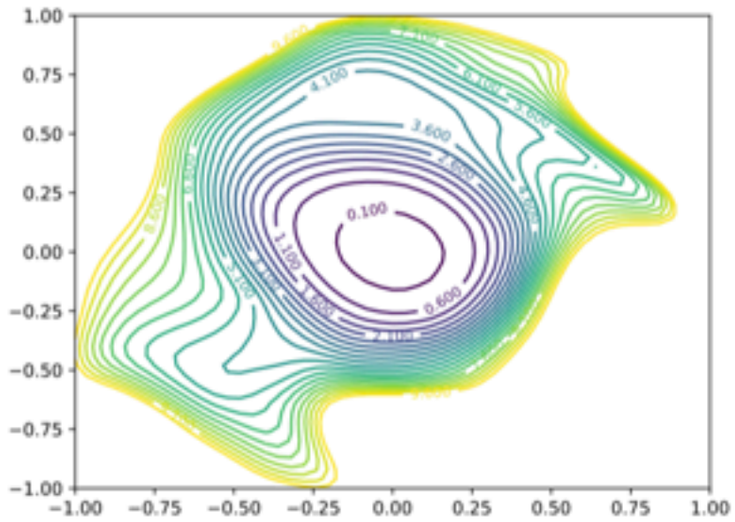
ResNet



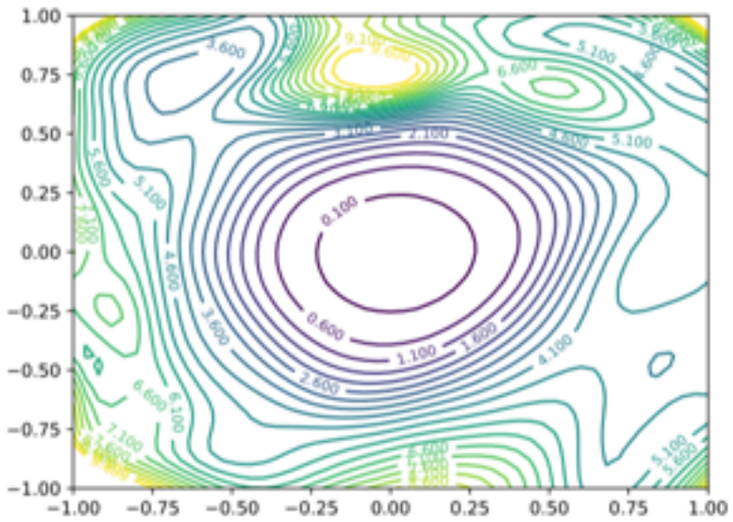
ResNet-NS

ResNet: deep ConvNet with skip connections
[Li et al, NIPS, 2018] https://arxiv.org/pdf/1712.09913.pdf

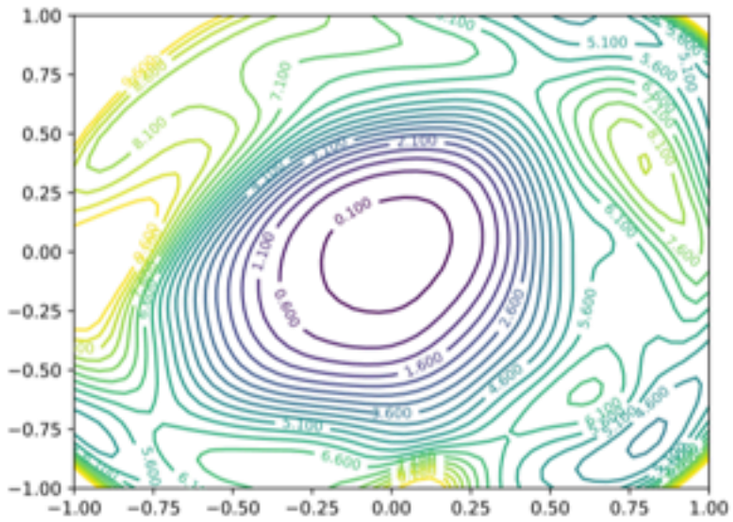
ResNet



(a) ResNet-20, 7.37%

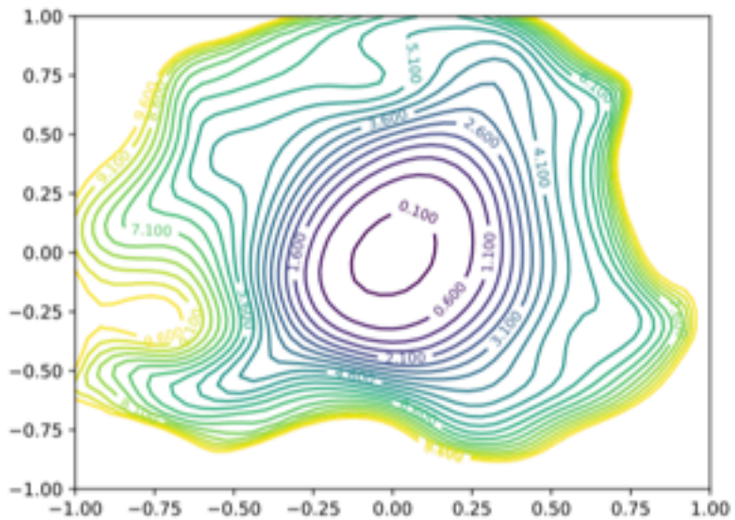


(b) ResNet-56, 5.89%

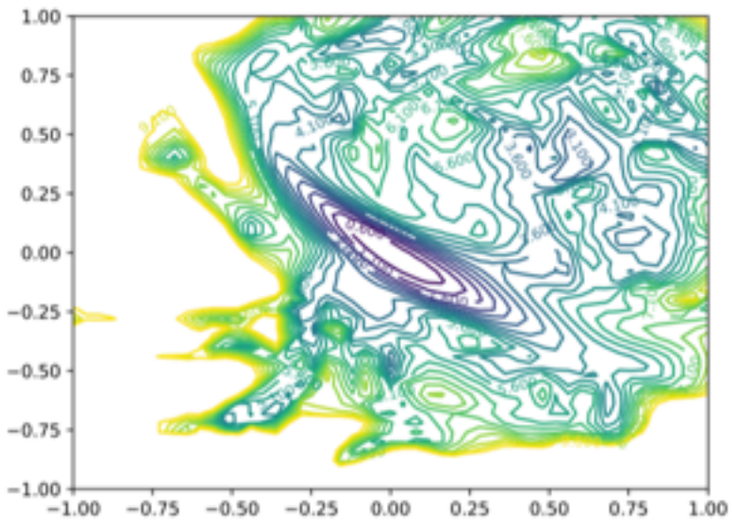


(c) ResNet-110, 5.79%

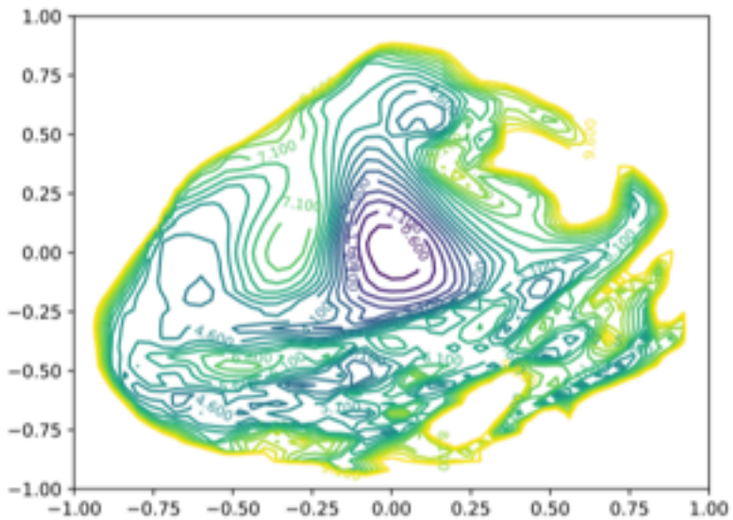
ResNet-NS



(d) ResNet-20-NS, 8.18%



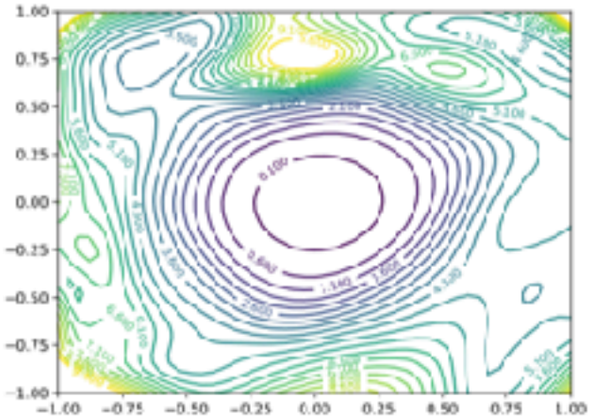
(e) ResNet-56-NS, 13.31%



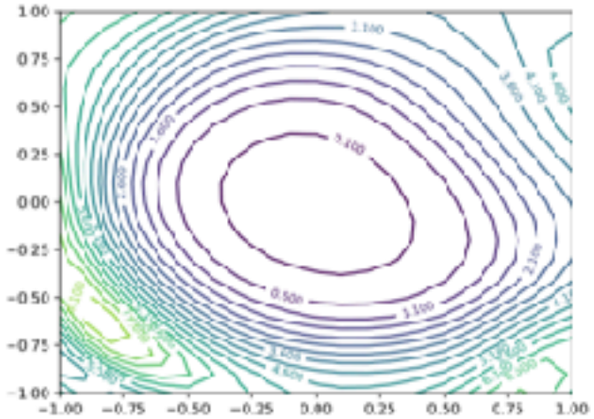
(f) ResNet-110-NS, 16.44%

WideResNet https://arxiv.org/abs/1605.07146

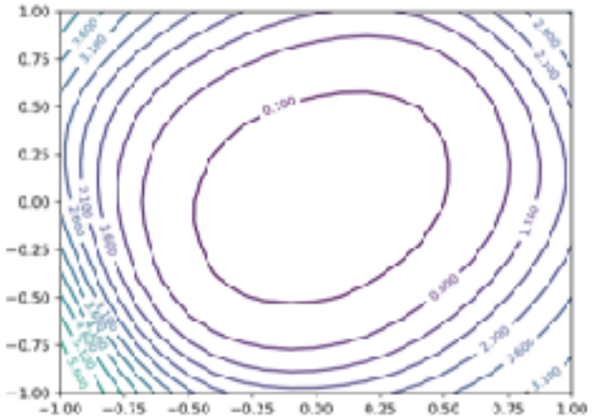
wide-ResNet



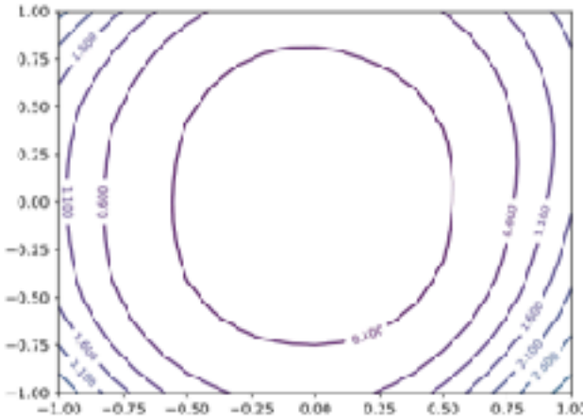
(a) $k = 1$, 5.89%



(b) $k = 2$, 5.07%

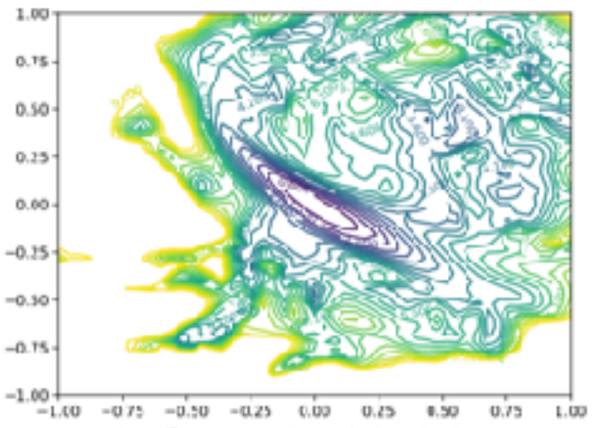


(c) $k = 4$, 4.34%

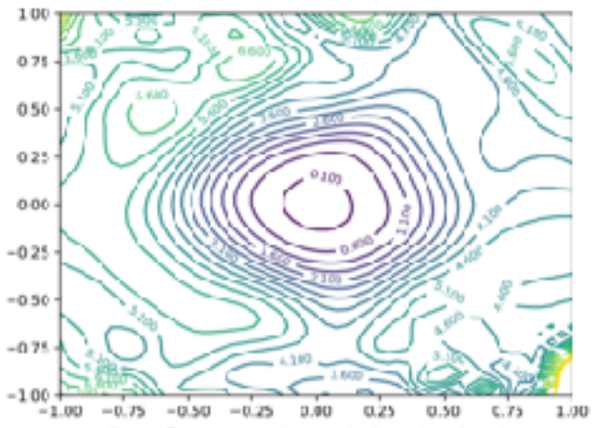


(d) $k = 8$, 3.93%

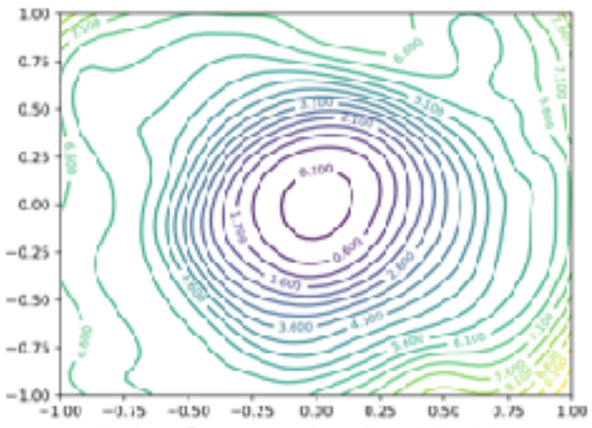
wide-ResNet-NS



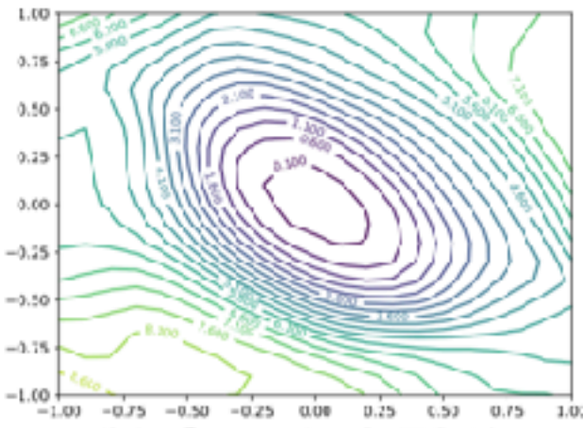
(e) $k = 1$, 13.31%



(f) $k = 2$, 10.26%

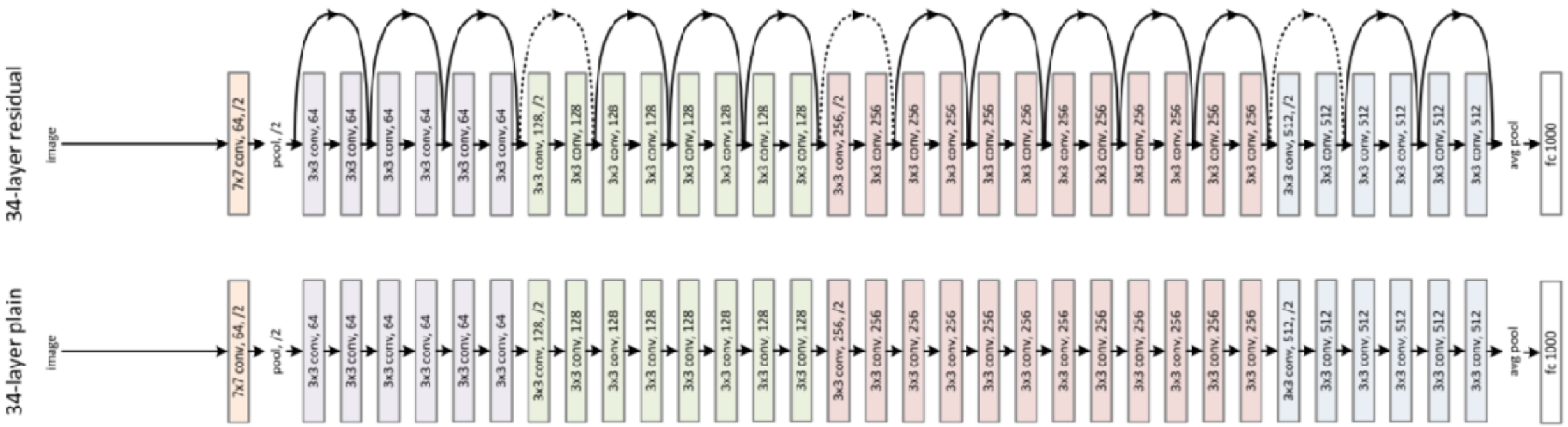


(g) $k = 4$, 9.69%



(h) $k = 8$, 8.70%

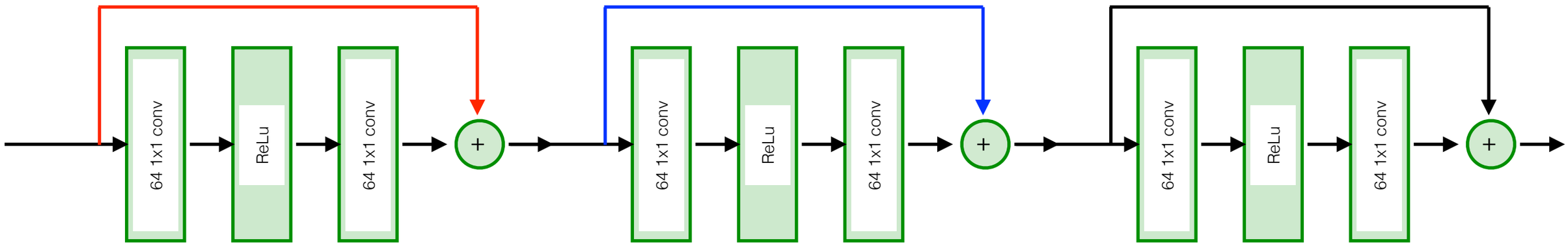
ResNet: deep ConvNet with skip connections



- Competition time about 152 layers ResNet,
- Recently they are able to train 1k layers ResNet
- Initialization with zero weights is meaningful
- Better gradient flow (many independent paths)
- Robustness wrt noise and layer removal
  https://www.kaggle.com/keras/resnet50/home

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

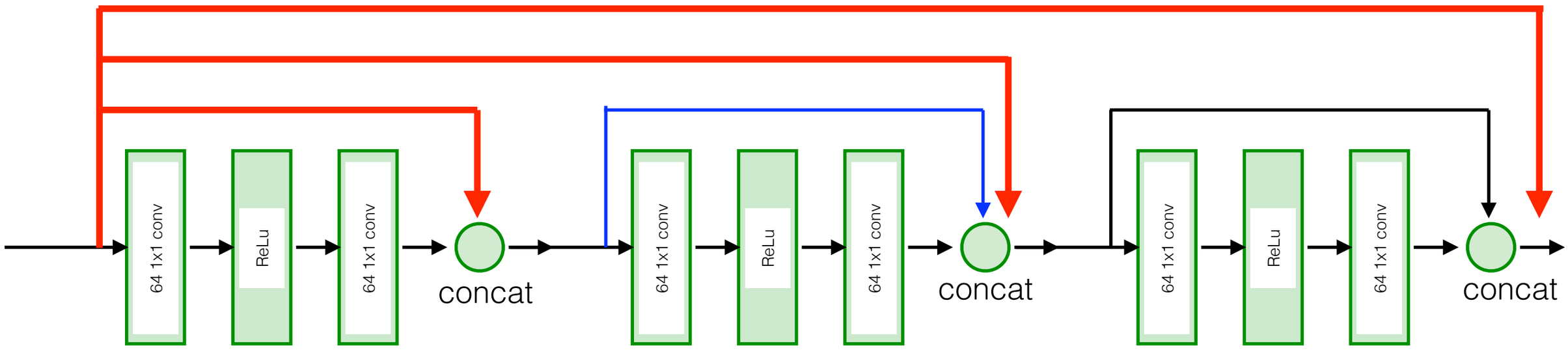
ResNet =>DenseNet



Start with multilayer ResNet architecture

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

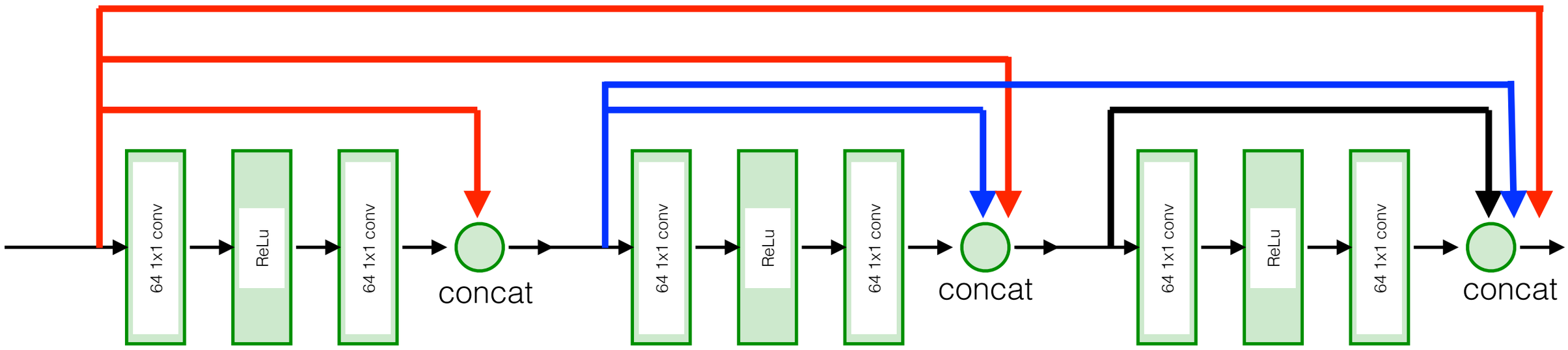
DenseNet



- Directly propagate each feature map to all following layers

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

DenseNet

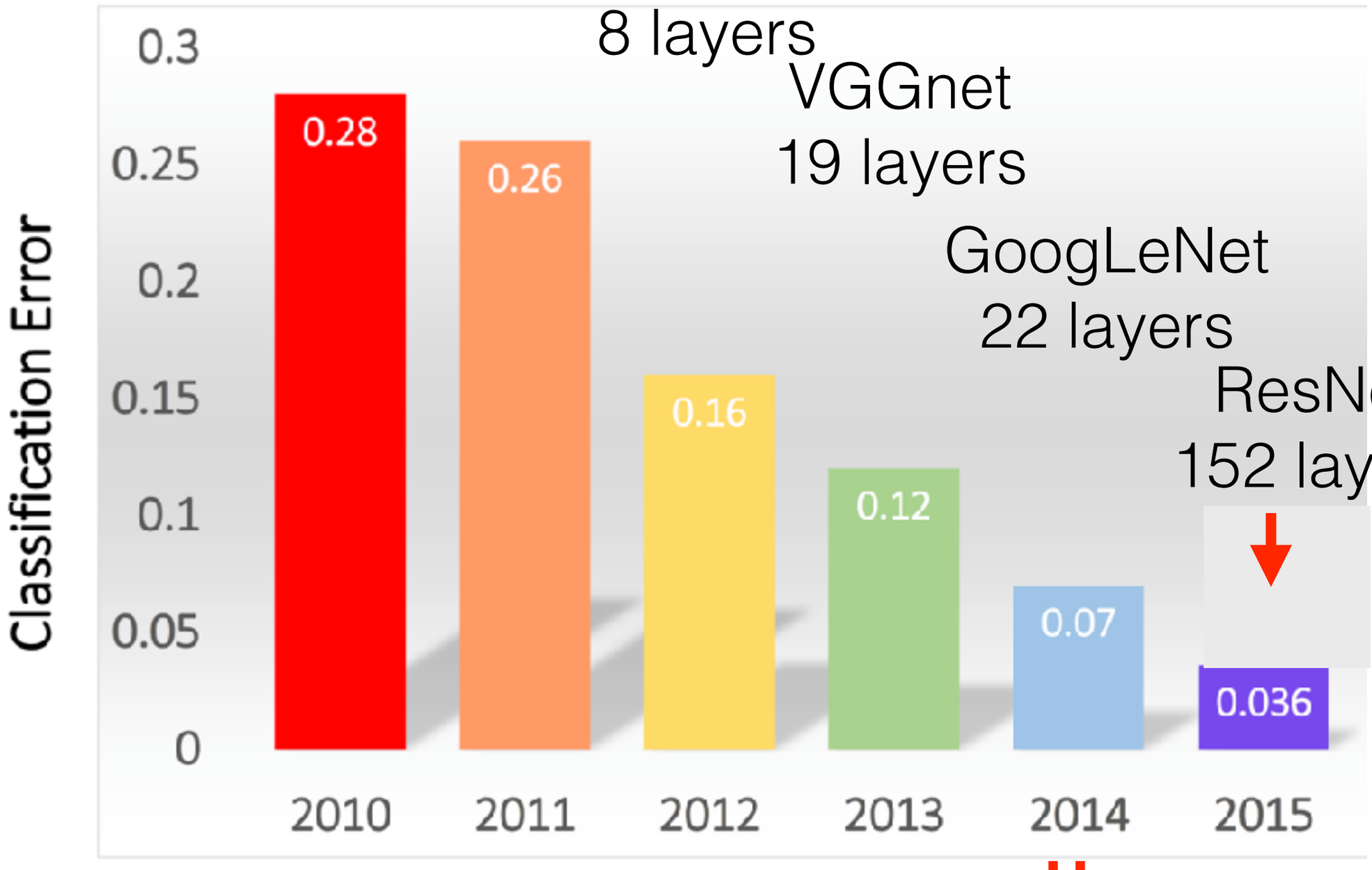


- Directly propagate each feature map to all following layers
- Improves gradient flow in backward pass

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

IMAGENET

Classification results

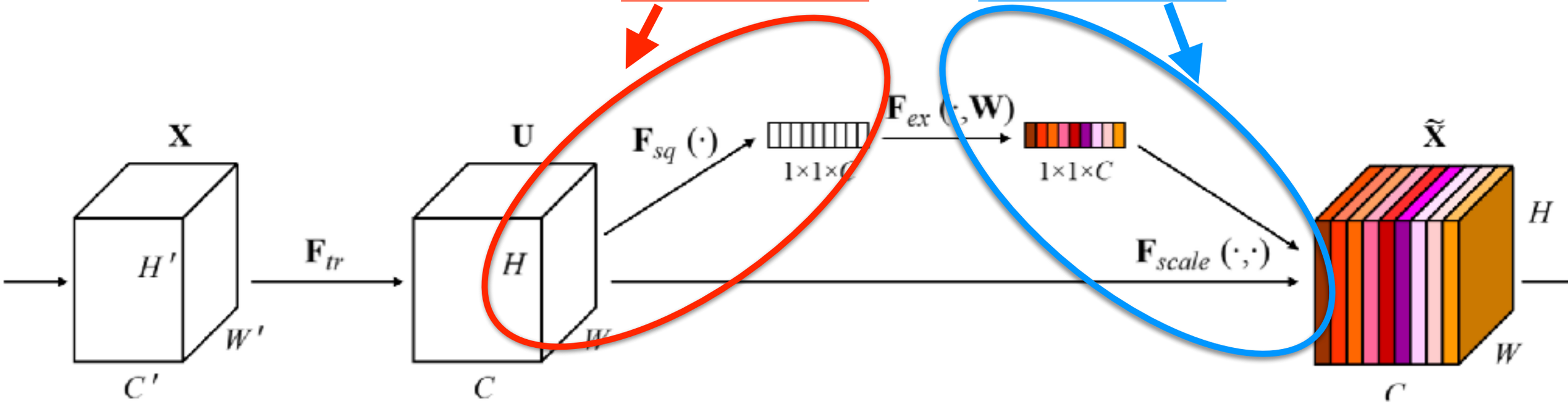


Human error around 5%

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

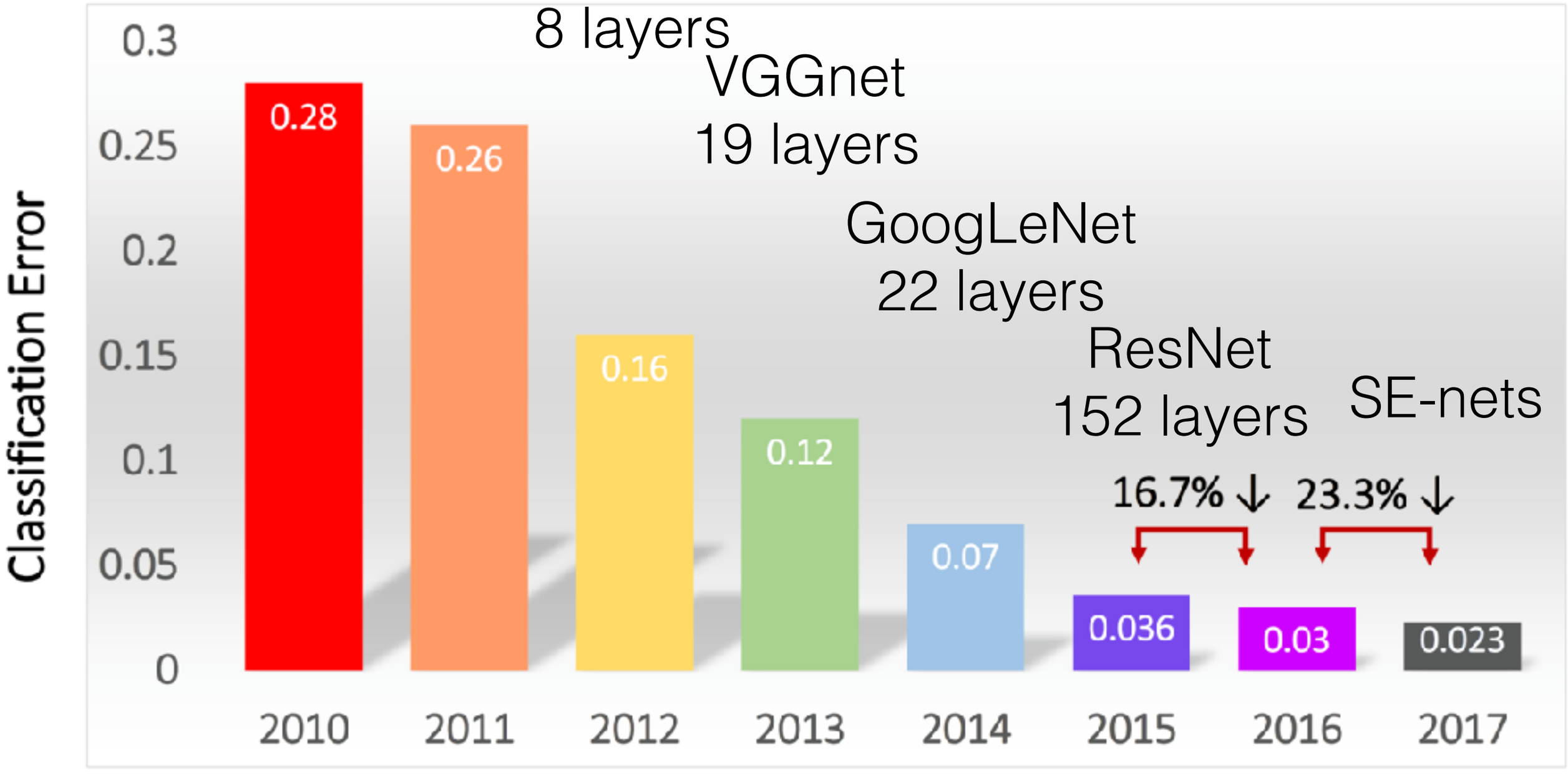
- Winner of ILSVRC 2017
- Enhancement of ResNet, InceptionNet and DenseNet architectures by SE blocks consistently decrease error on ImageNet, COCO, …

Squeeze and Excitation block



IMAGENET

Classification results



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

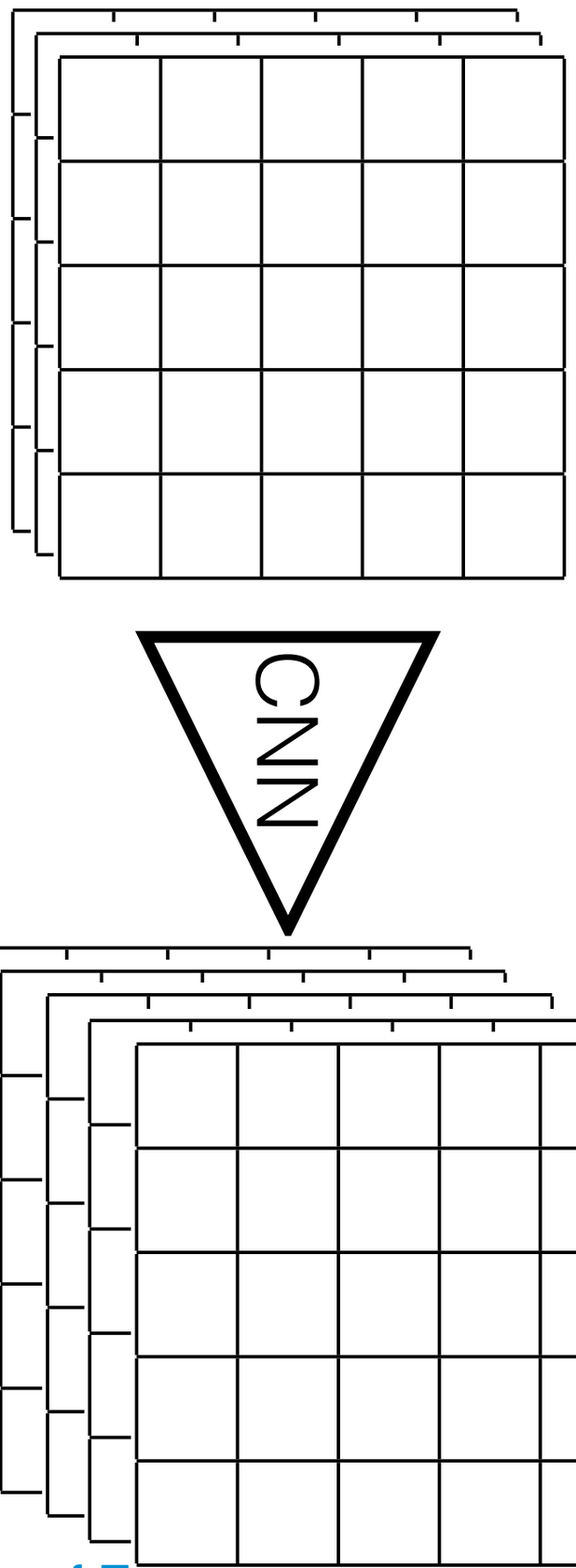
Outline

- 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

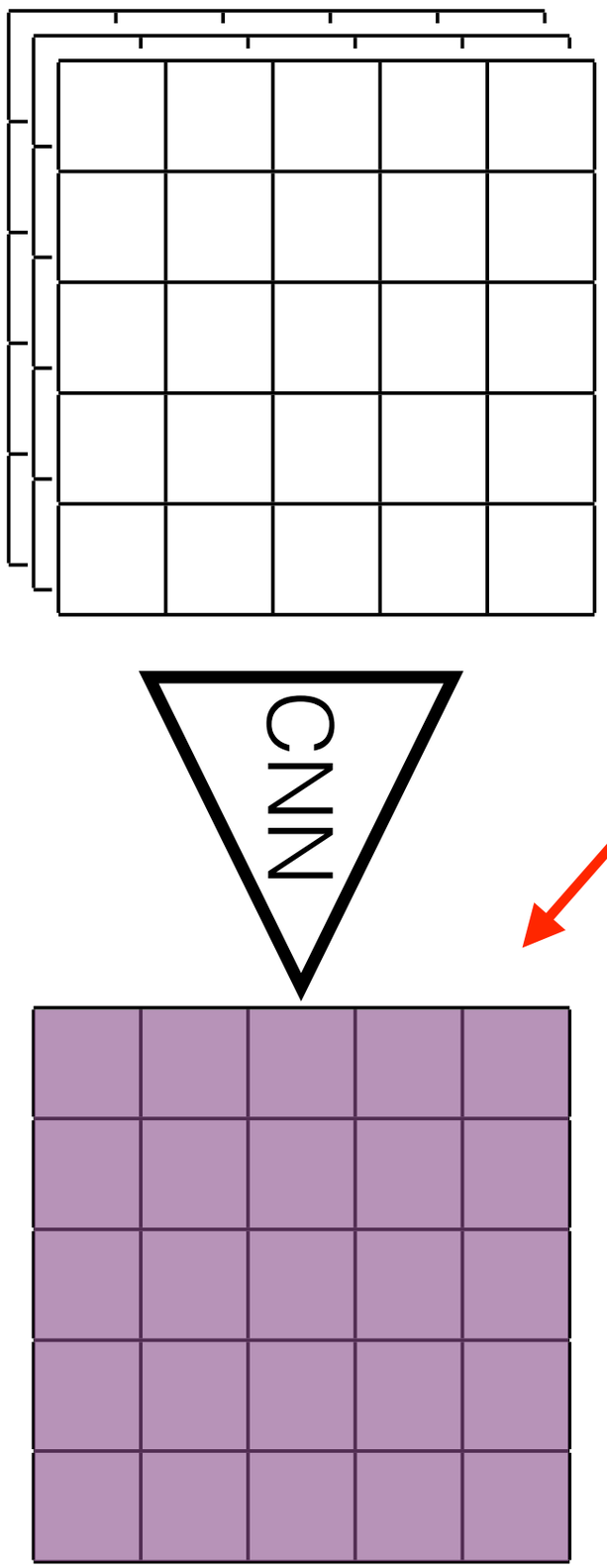


RGB image
(HxWx3)

- road
- sideway
- pedestrian
- traffic sign
- trees
- sky

labels
(HxWxN)

Semantic segmentation



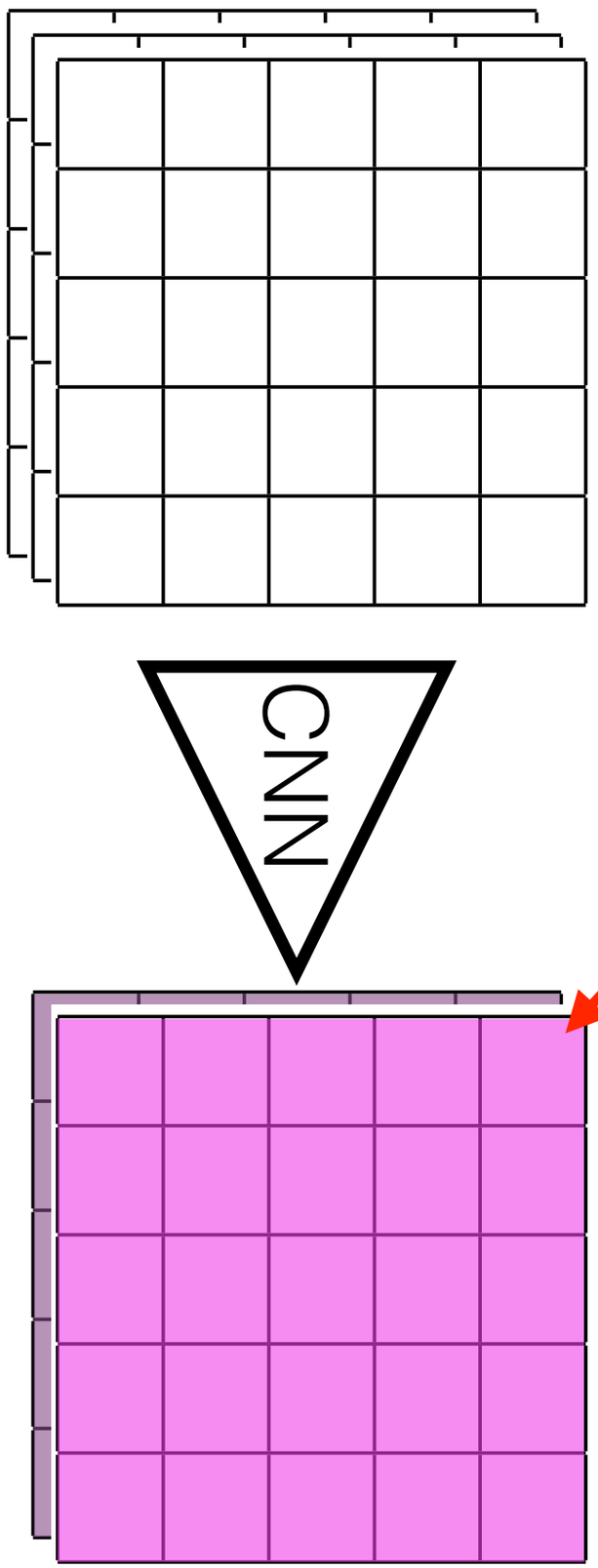
RGB image
(HxWx3)

road
sideway
pedestrian
traffic sign
trees
sky

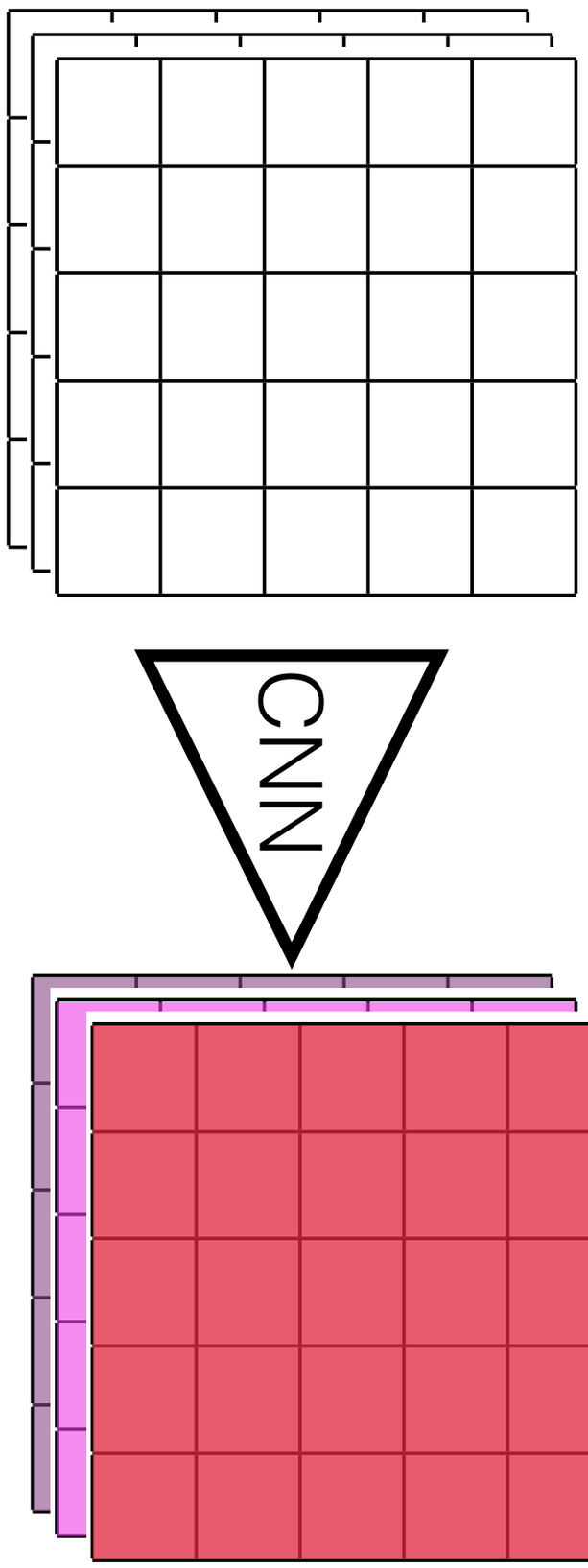
pixel-wise probability
of being **road**

channel 1

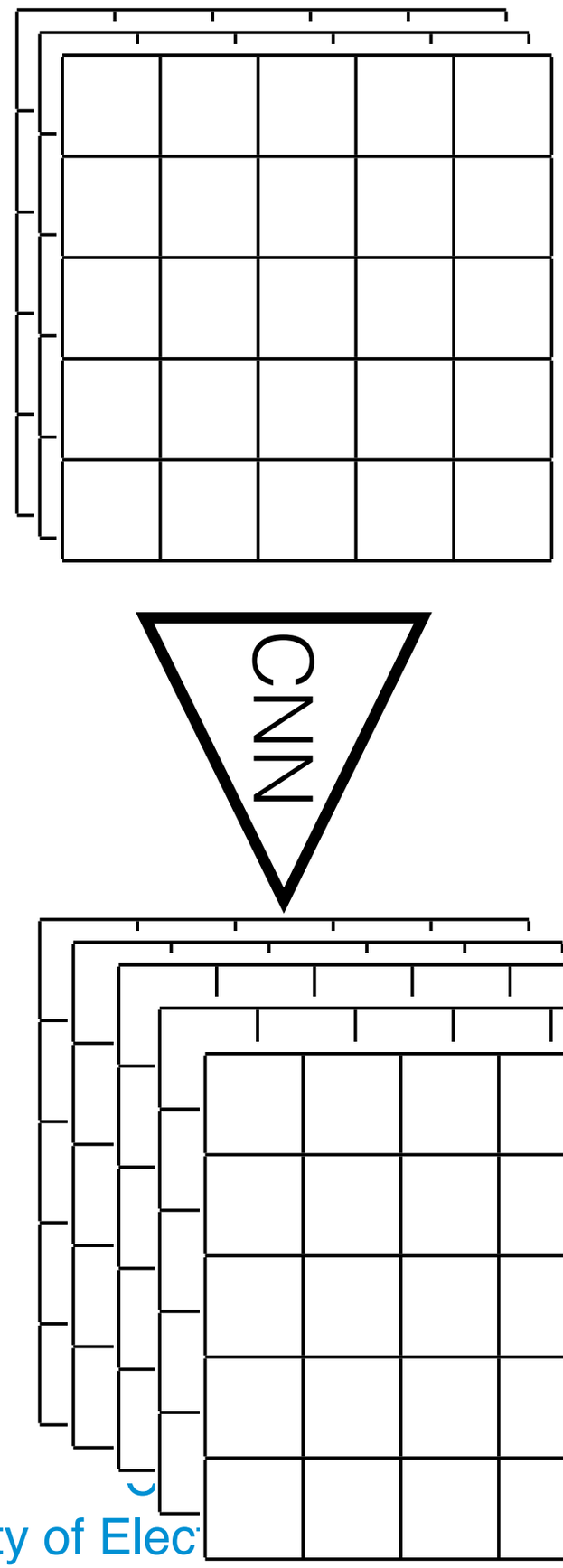
Semantic segmentation



Semantic segmentation



Semantic segmentation

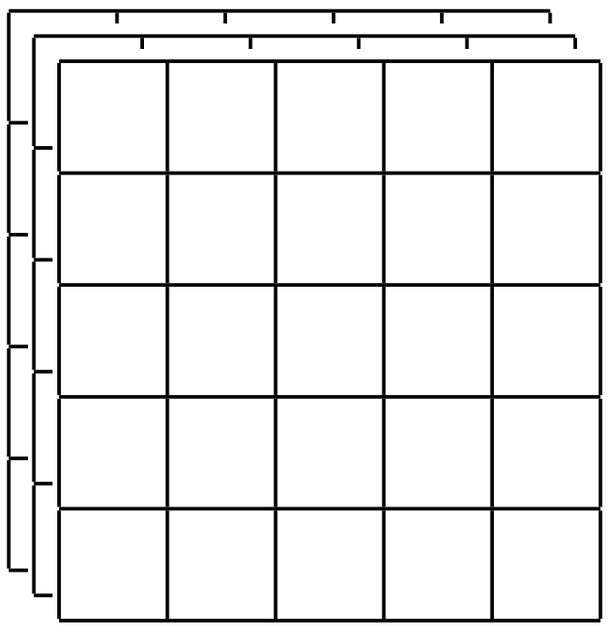


RGB image
(HxWx3)

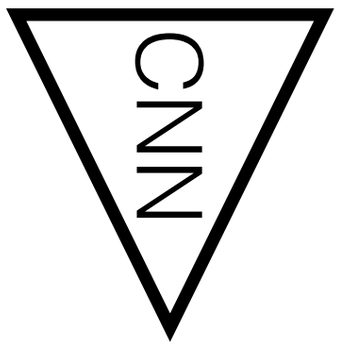
road
sideway
pedestrian
traffic sign
trees
sky

as many output channels
as semantic labels

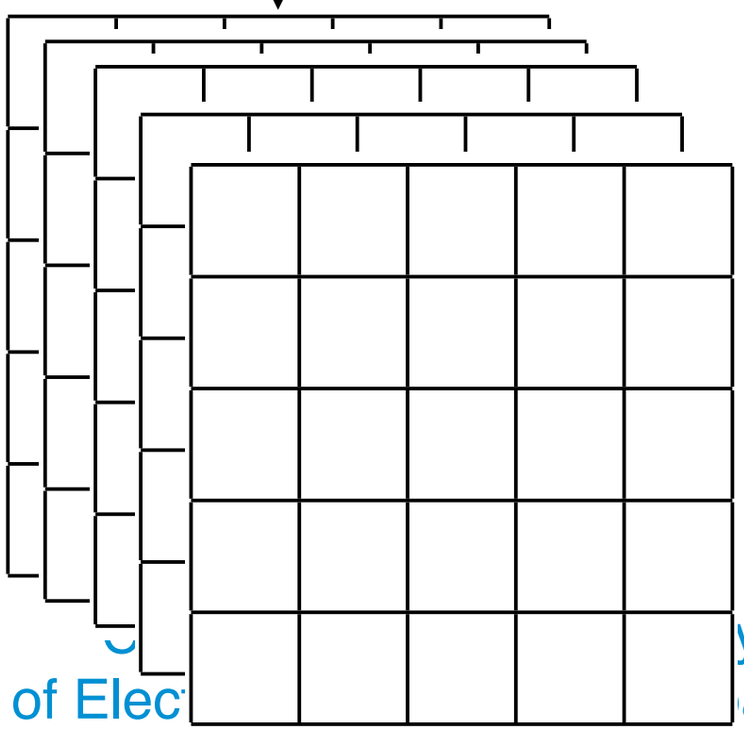
Semantic segmentation

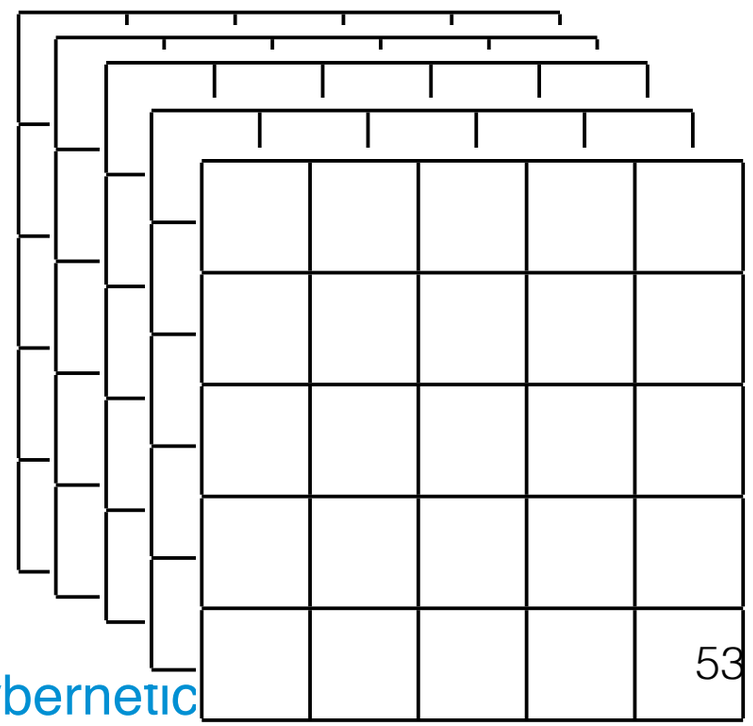


RGB image
(HxWx3)

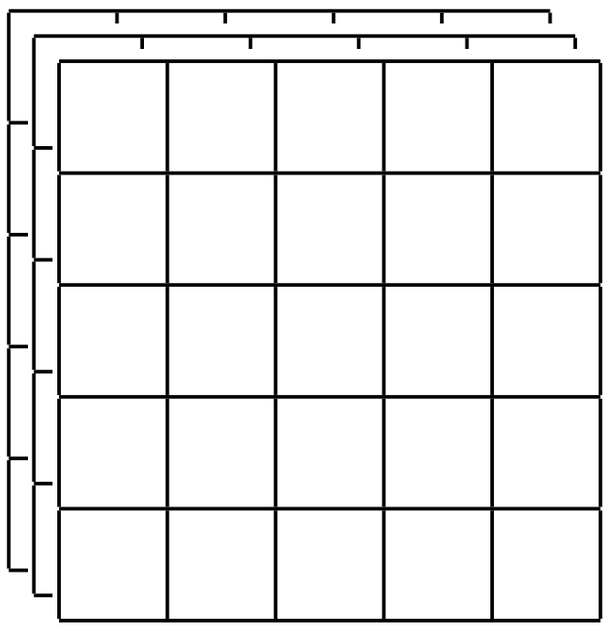


ground truth (0-1 values)

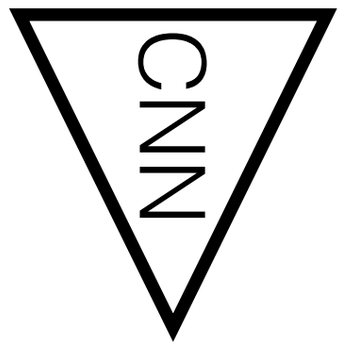




Semantic segmentation

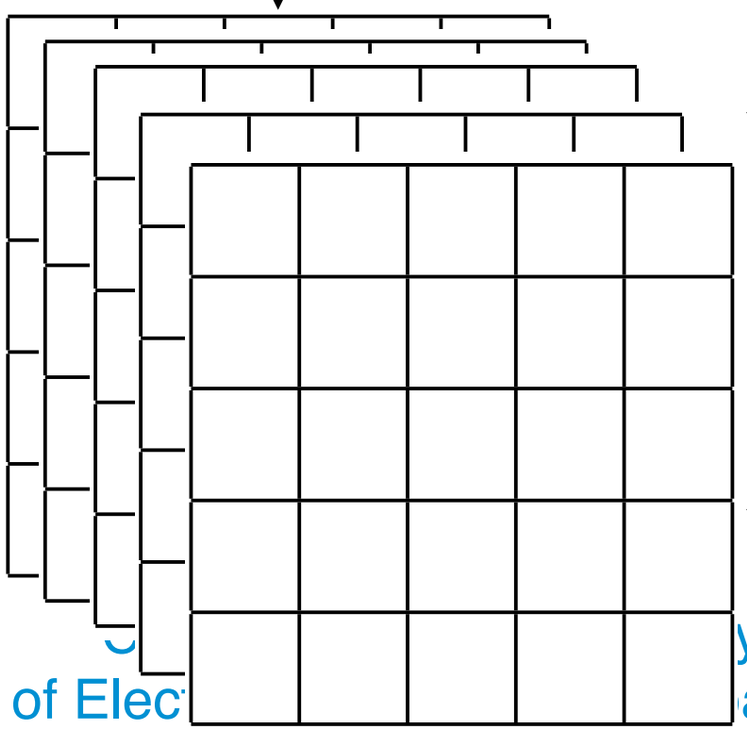


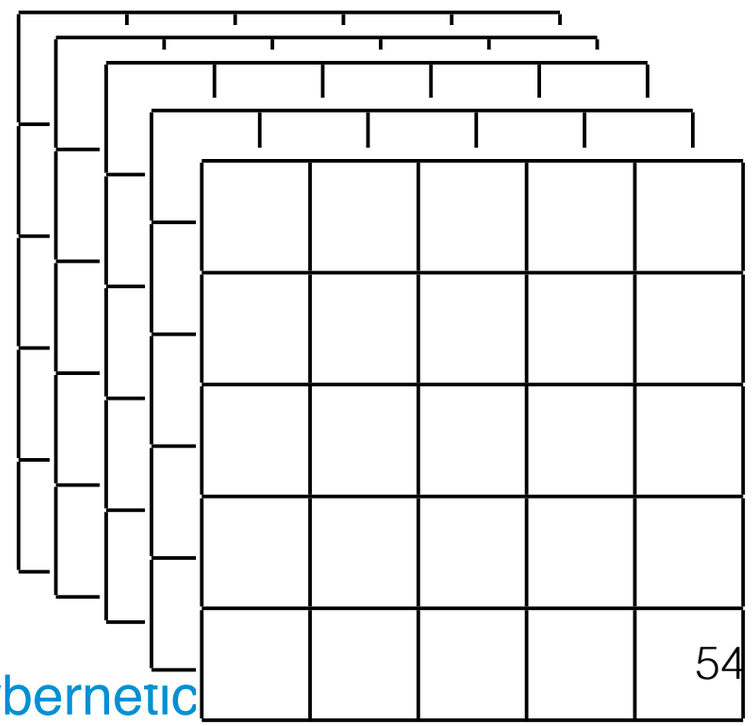
RGB image
(HxWx3)



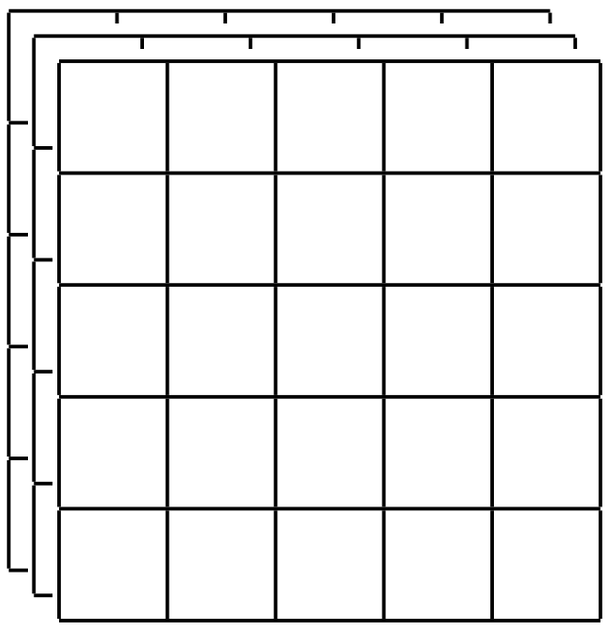
ground truth (0-1 values)

$$-\log\Big(\quad\Big)\odot$$

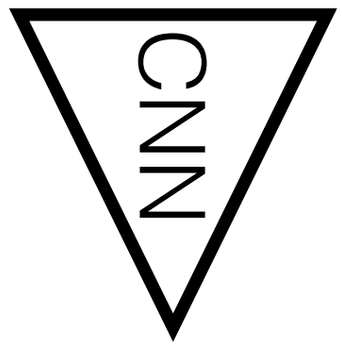




Semantic segmentation



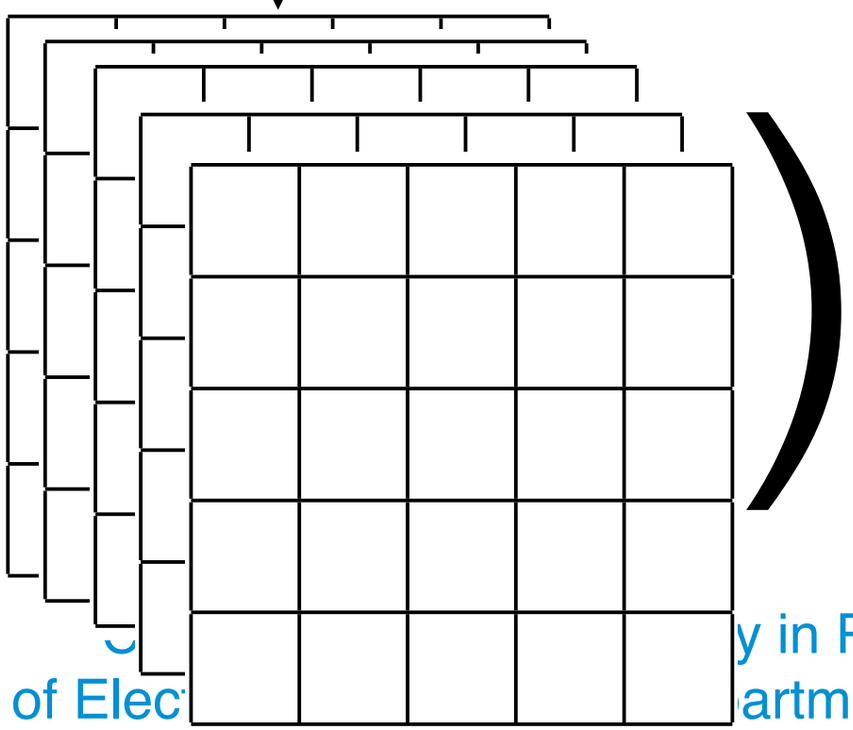
RGB image
(HxWx3)

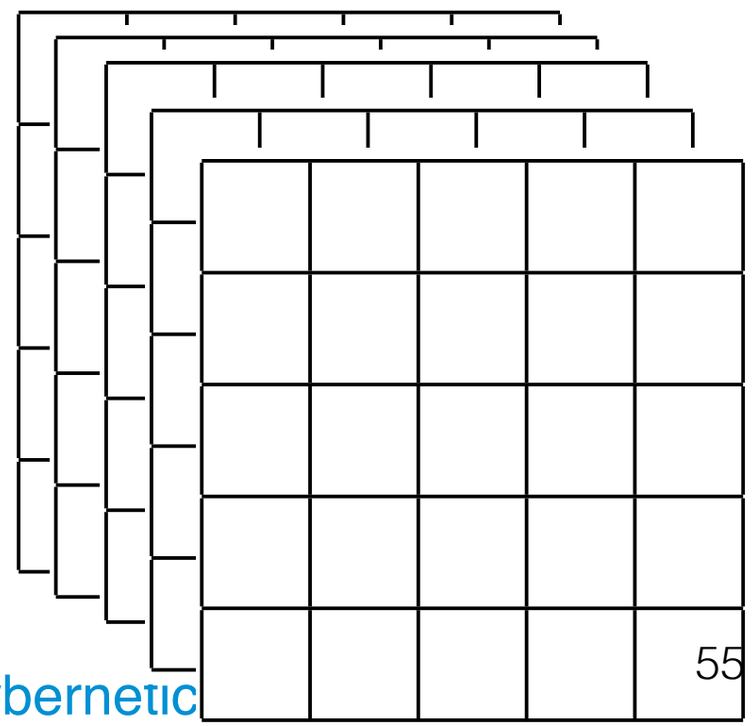


cross-entropy loss

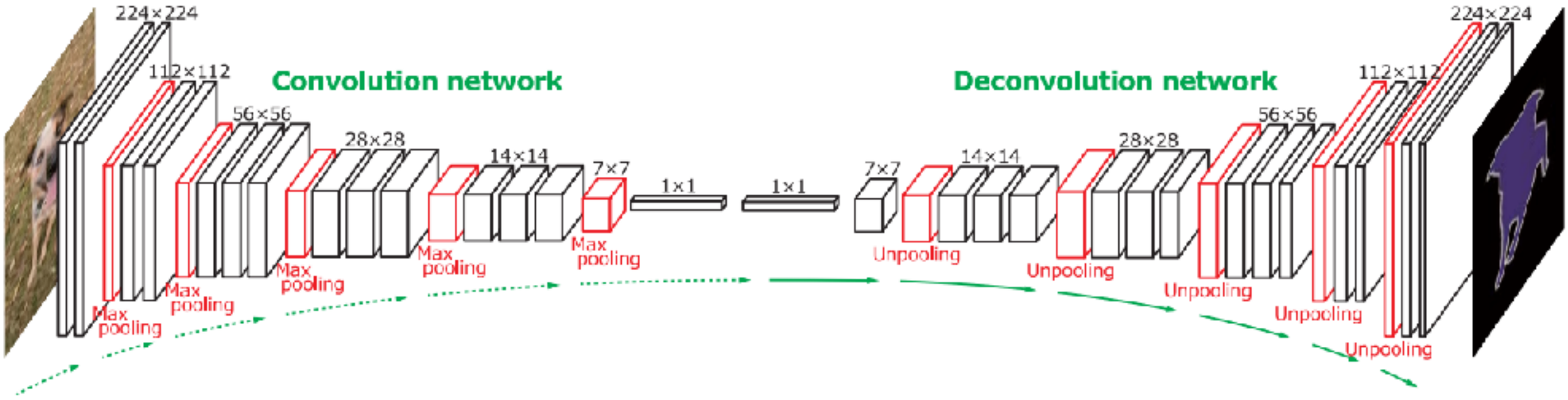
ground truth (0-1 values)

$$\sum_{\text{pixels}} -\log\Big(\quad\Big) \odot$$





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

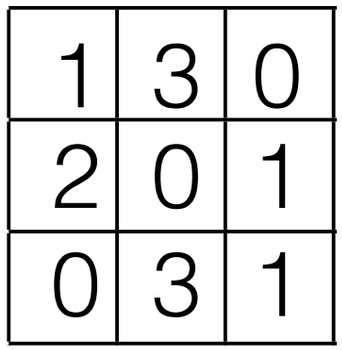
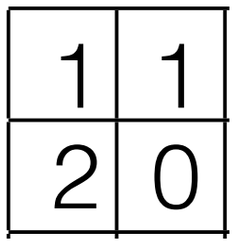
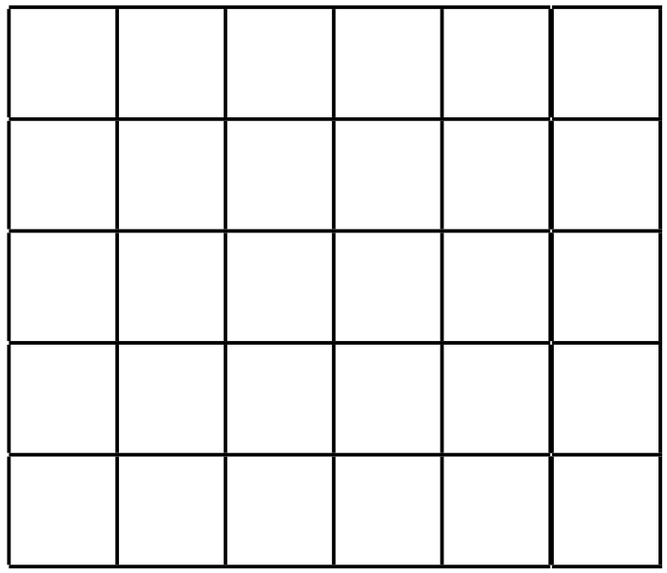
deconv ( , ) = 

image (3x3)

kernel (2x2)

output (**6x6**)

Deconvolution

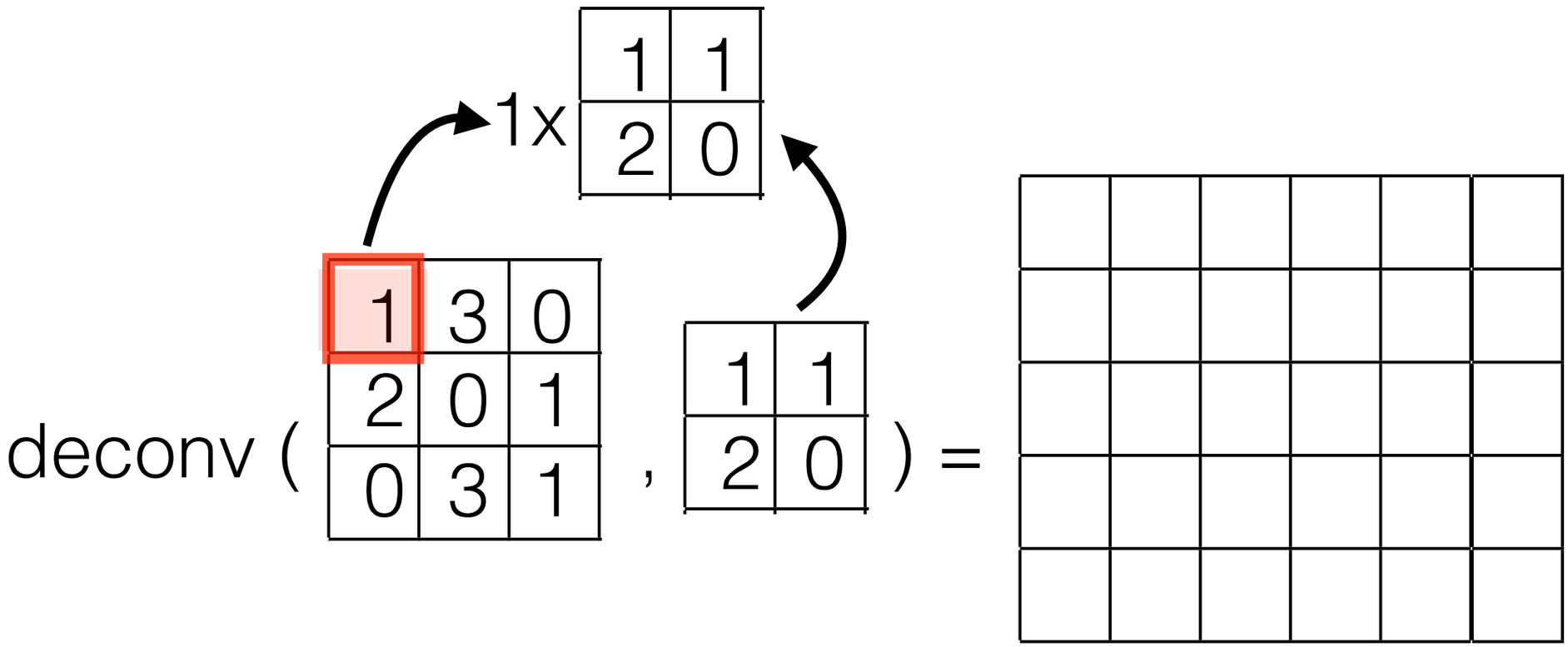


image (3x3)

kernel (2x2)

output (**6x6**)

Deconvolution

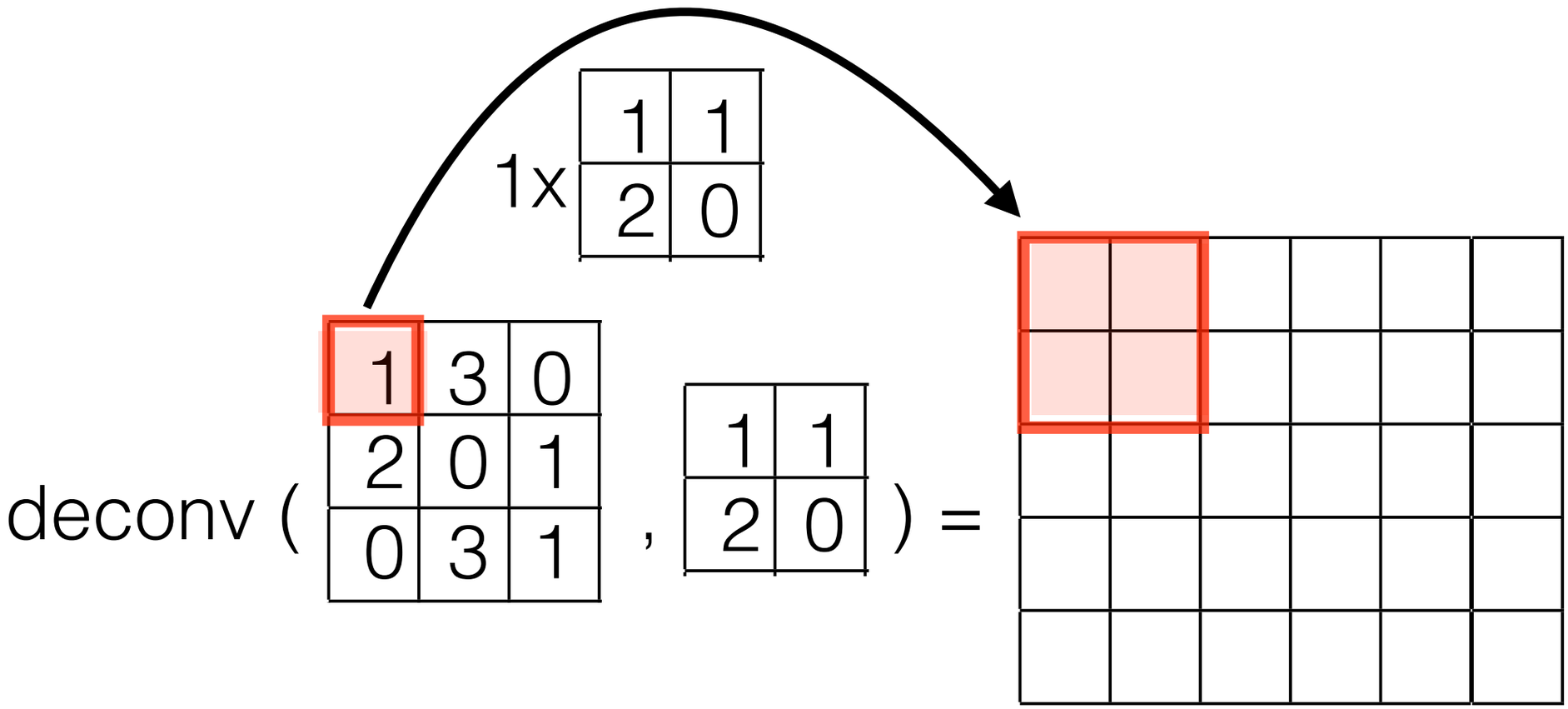


image (3x3) kernel (2x2) output (**6x6**)

Deconvolution

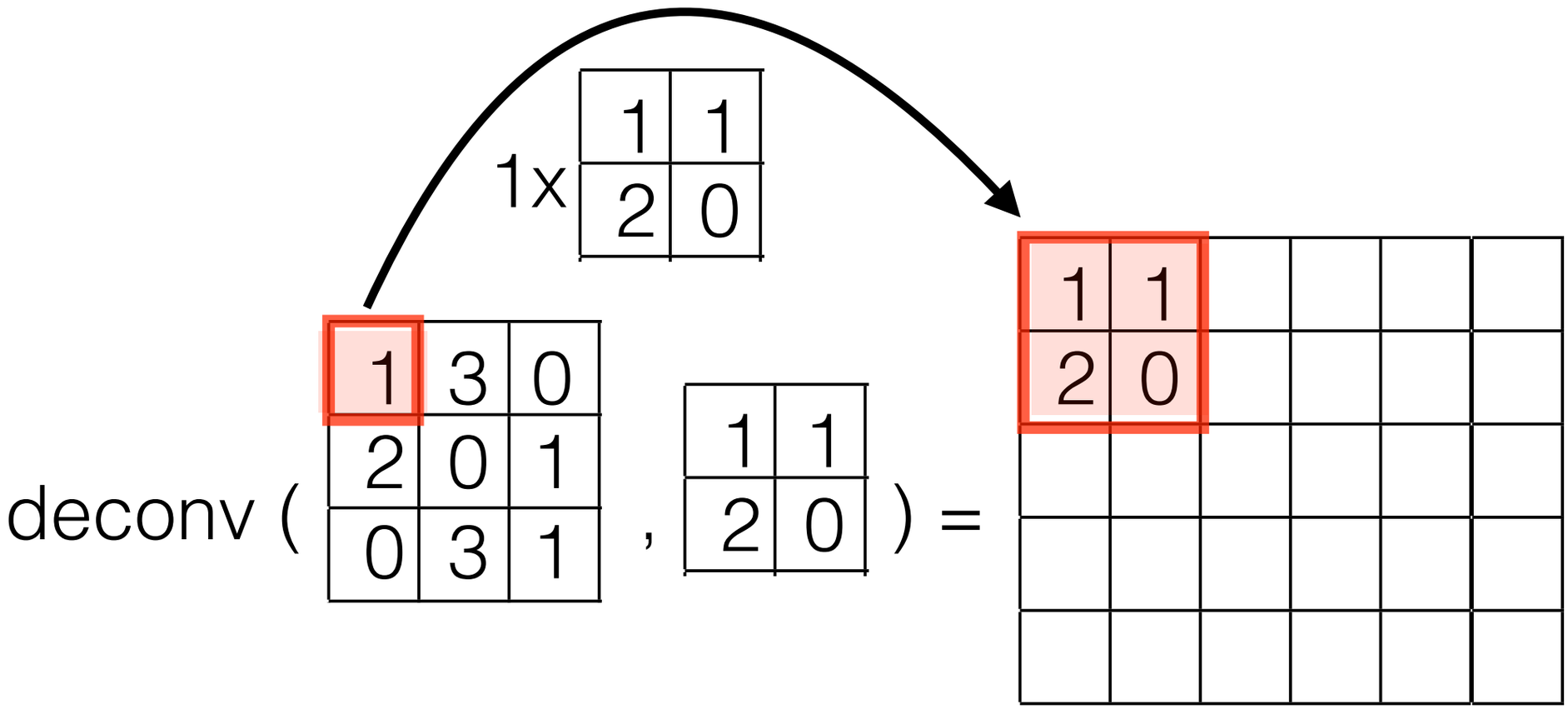


image (3x3) kernel (2x2) output (**6x6**)

Deconvolution

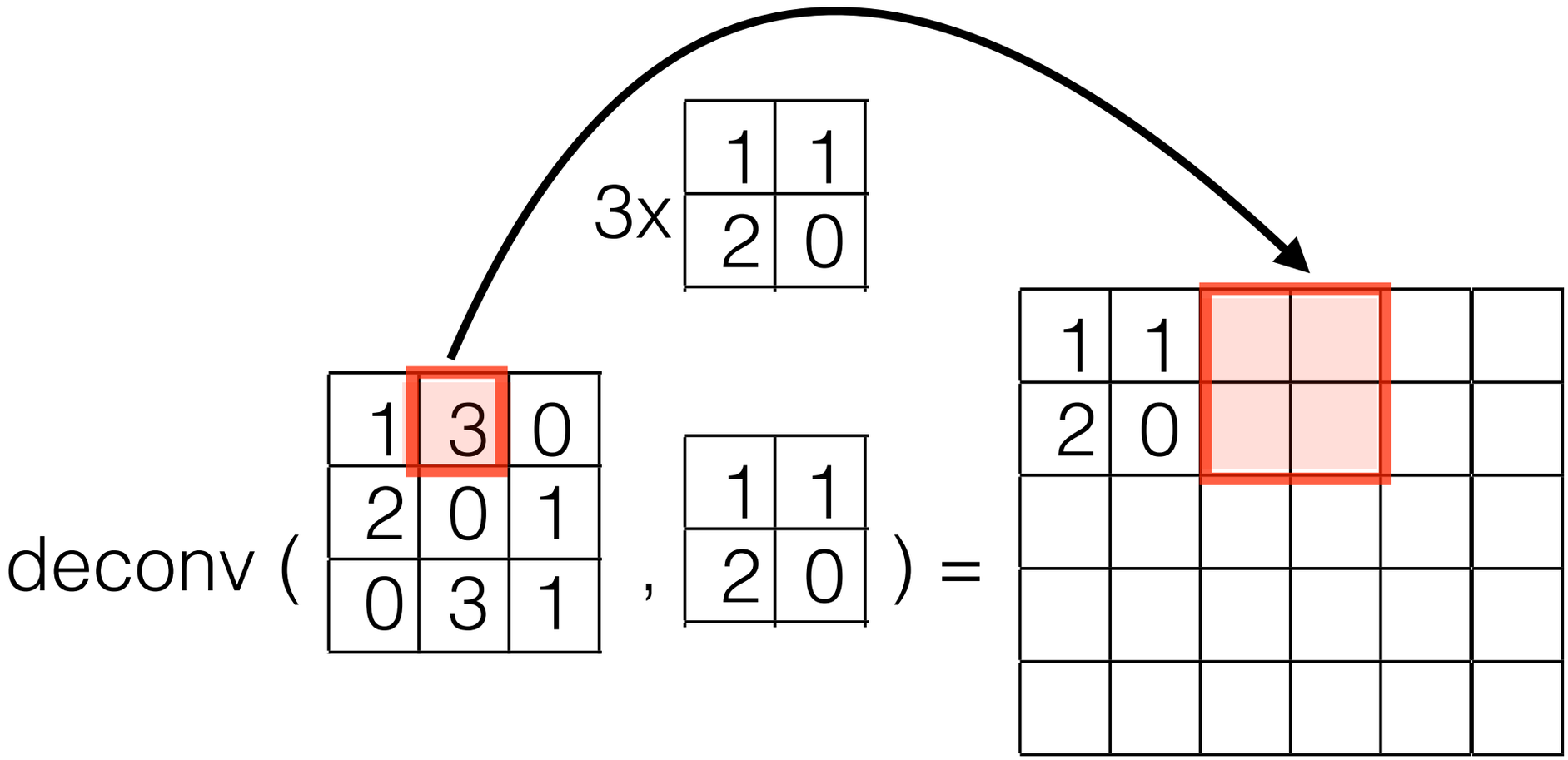


image (3x3) kernel (2x2) output (**6x6**)

Deconvolution

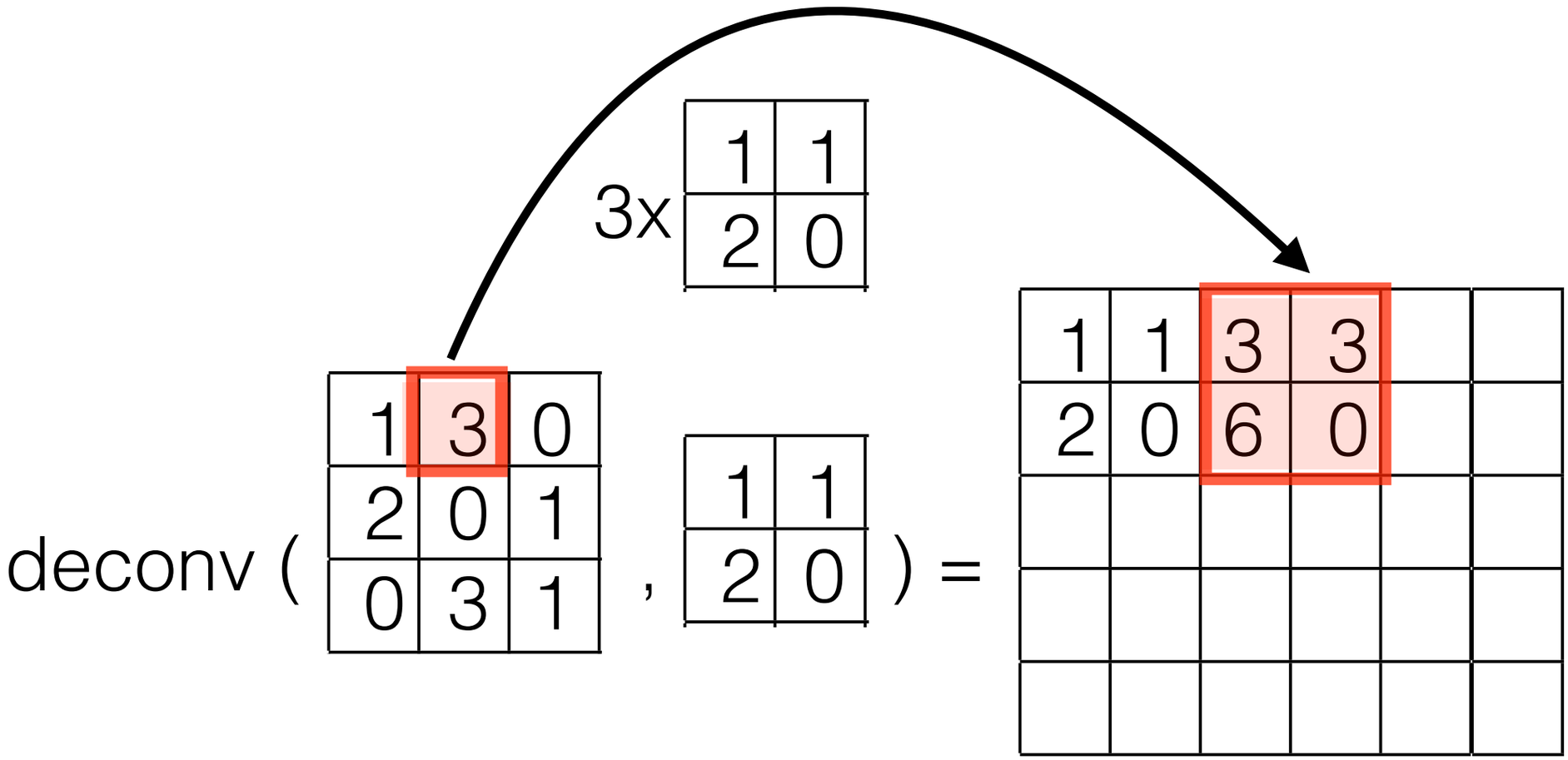


image (3x3) kernel (2x2) output (**6x6**)

Deconvolution

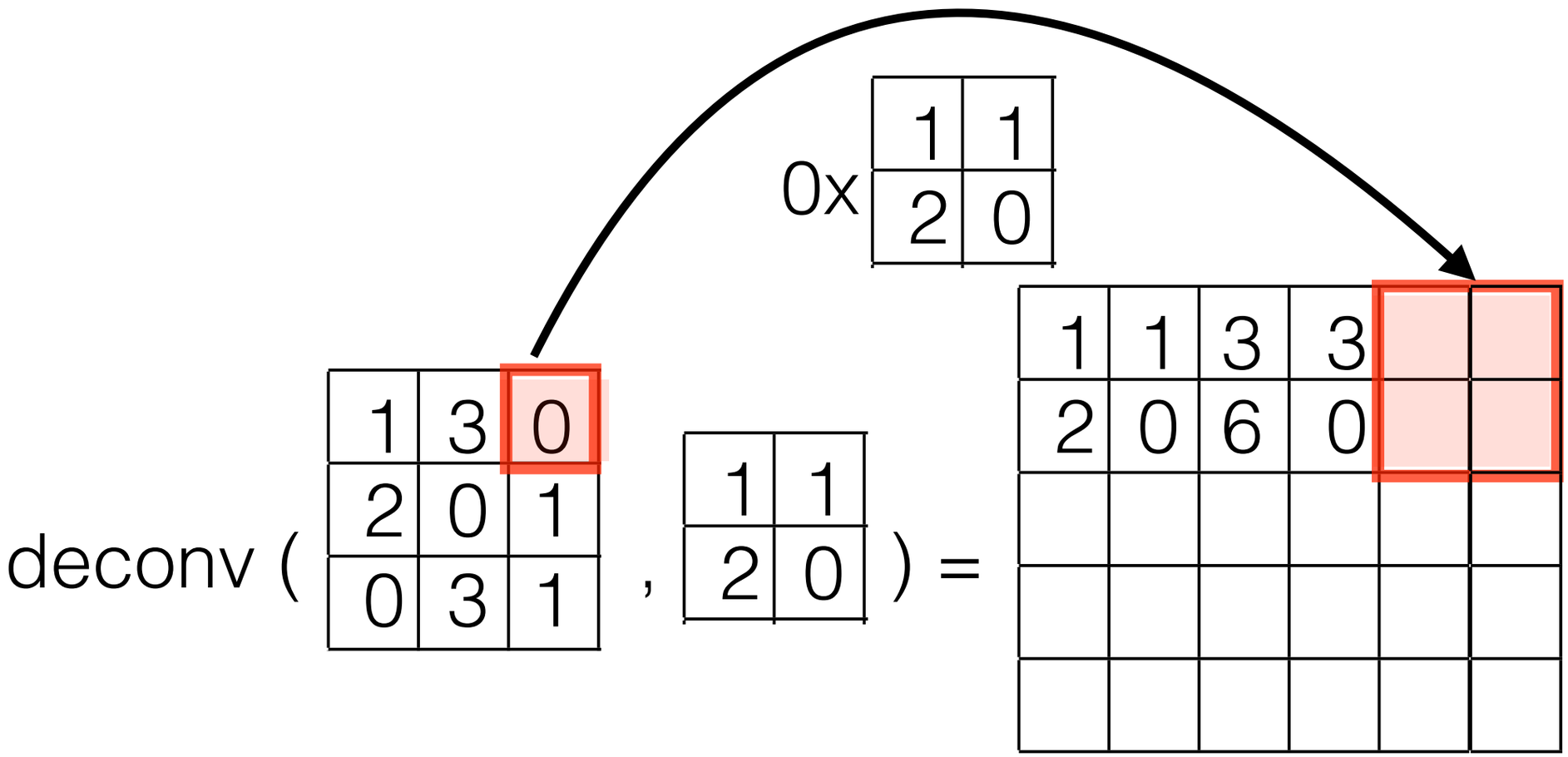


image (3x3) | kernel (2x2) | output (**6x6**)

Deconvolution

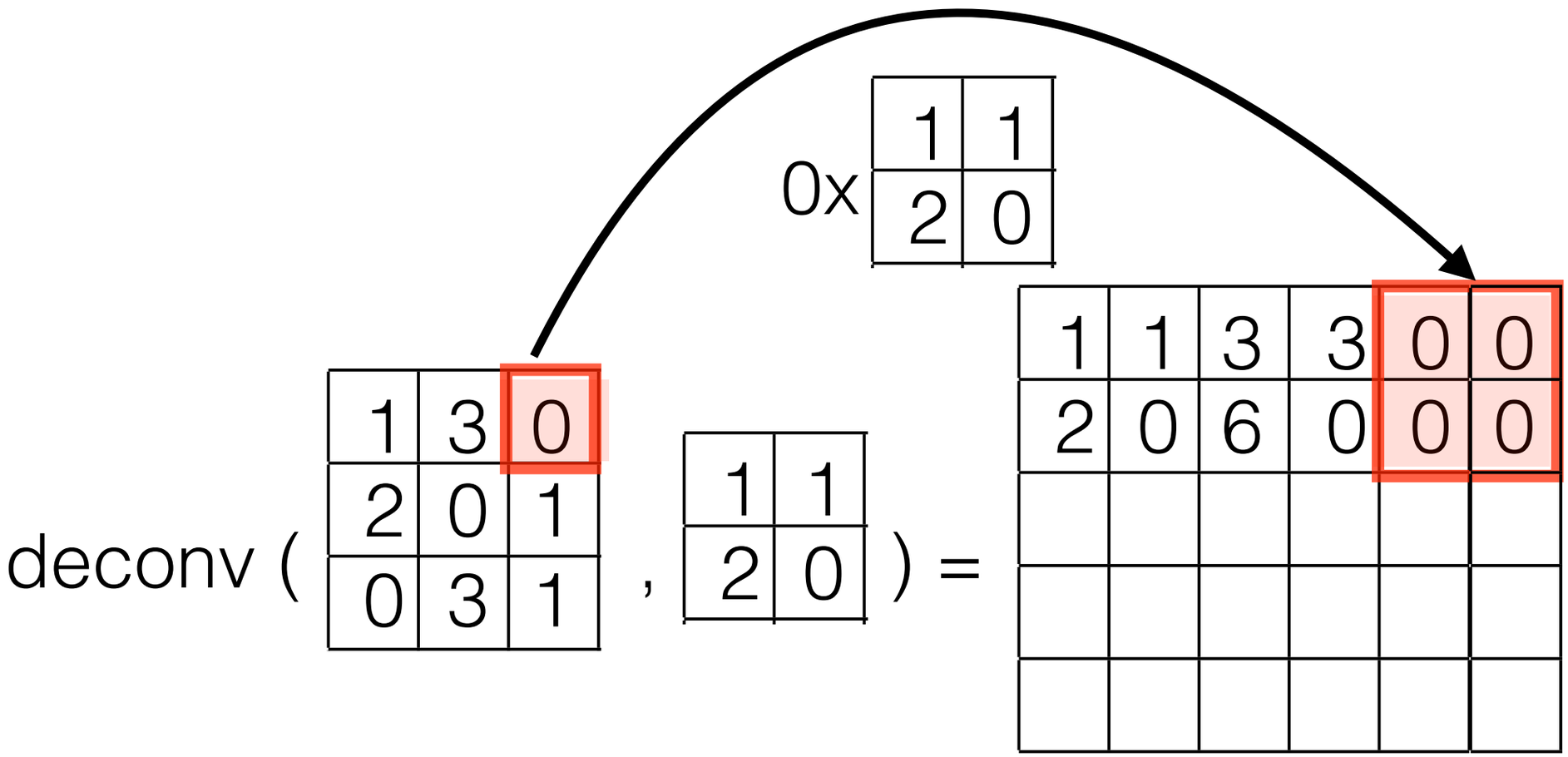


image (3x3) kernel (2x2) output (**6x6**)

unpooling

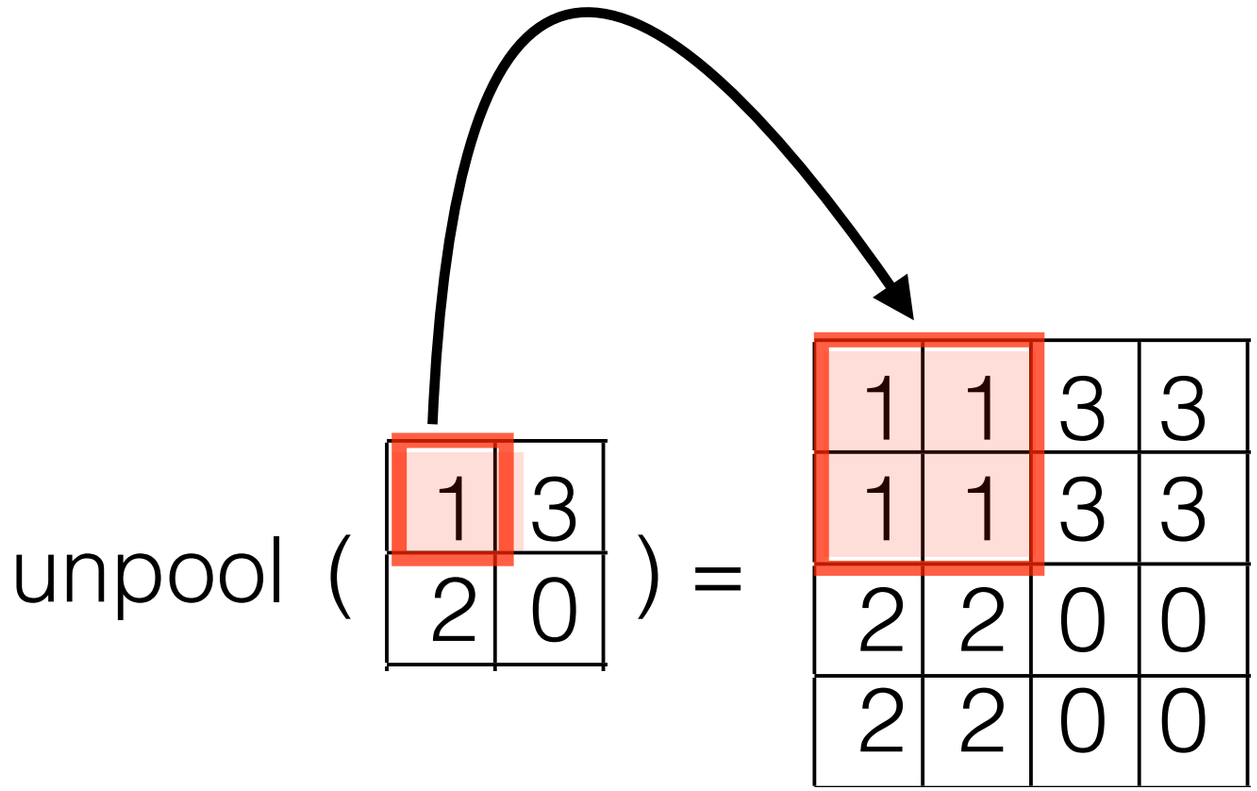


image (2x2)

output (**4x4**)

copy everywhere unpooling

max-unpooling

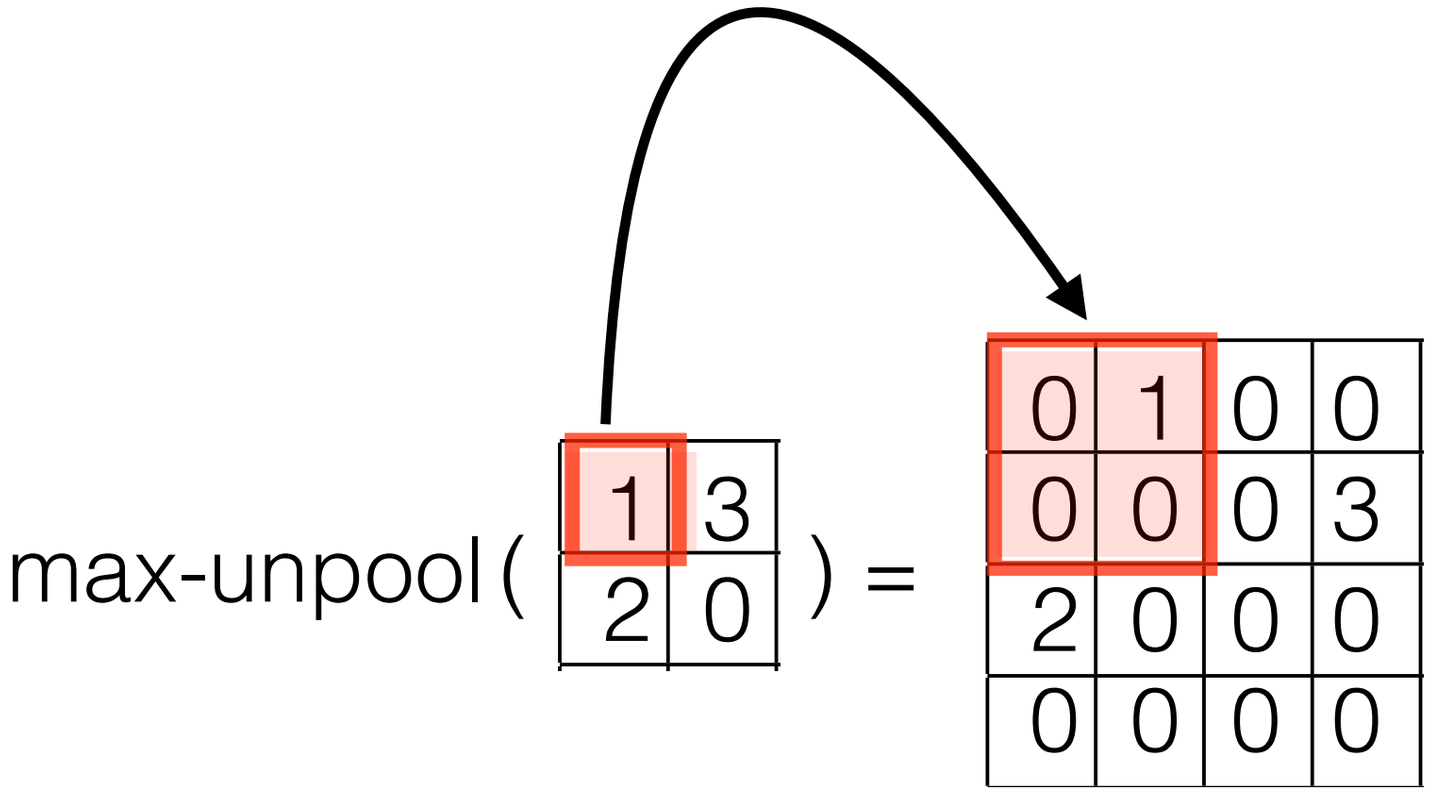
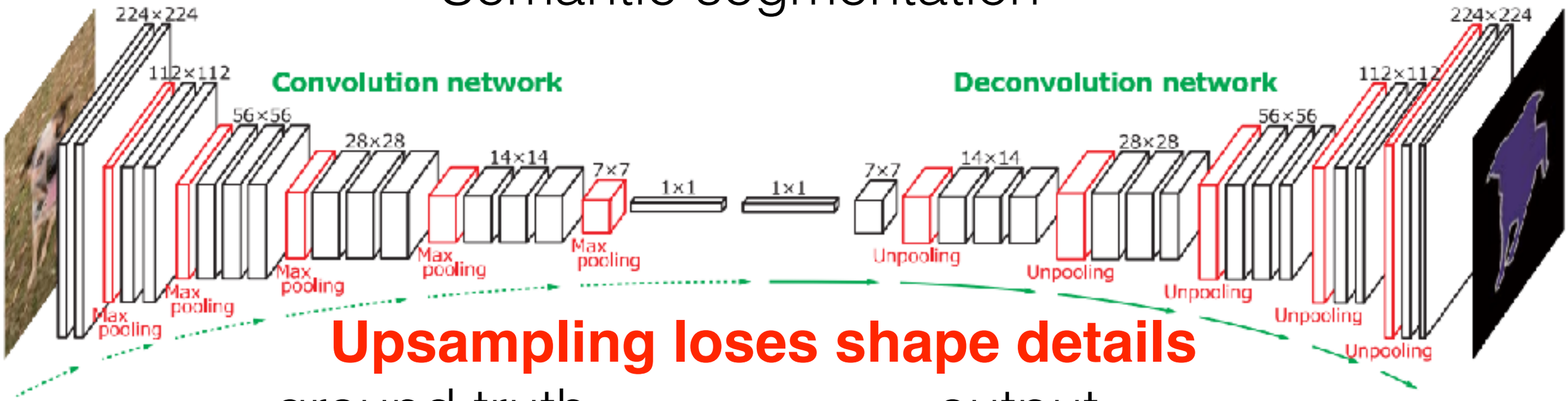


image (2x2)

output (**4x4**)

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

Semantic segmentation



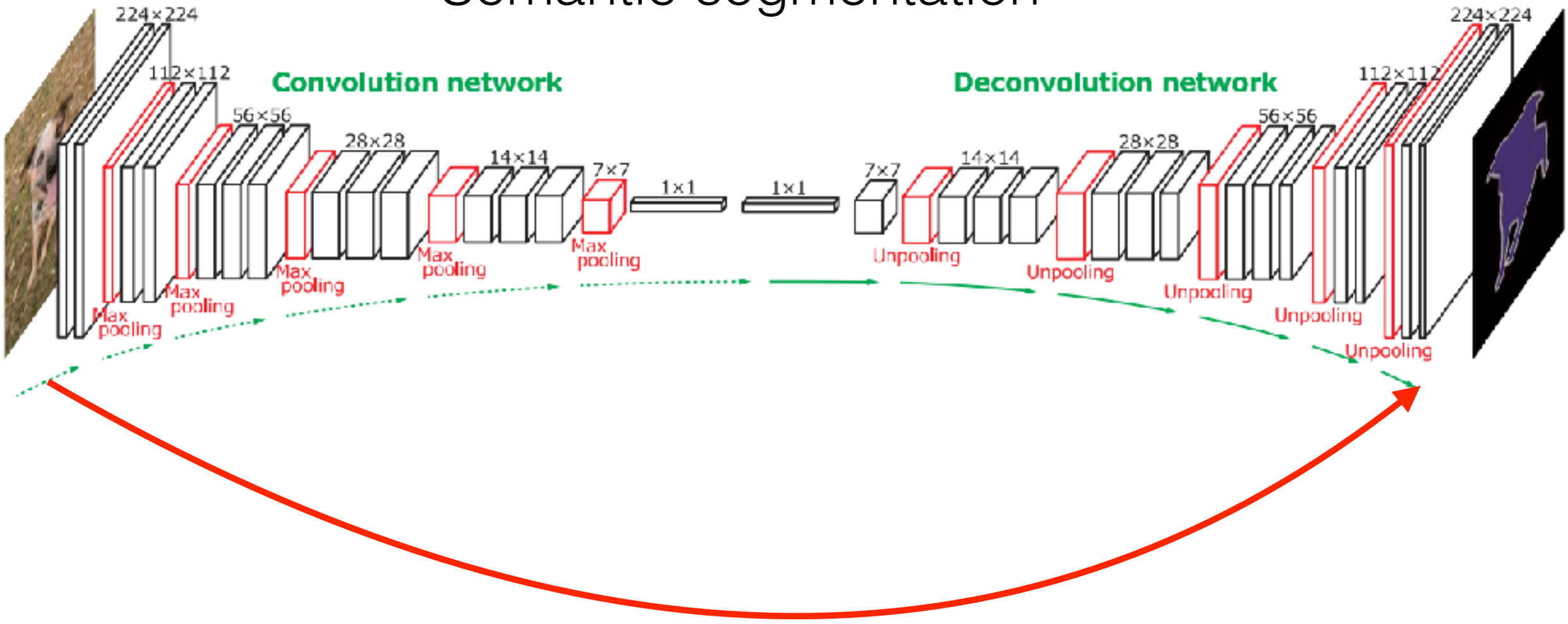
Upsampling loses shape details

ground truth

output

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

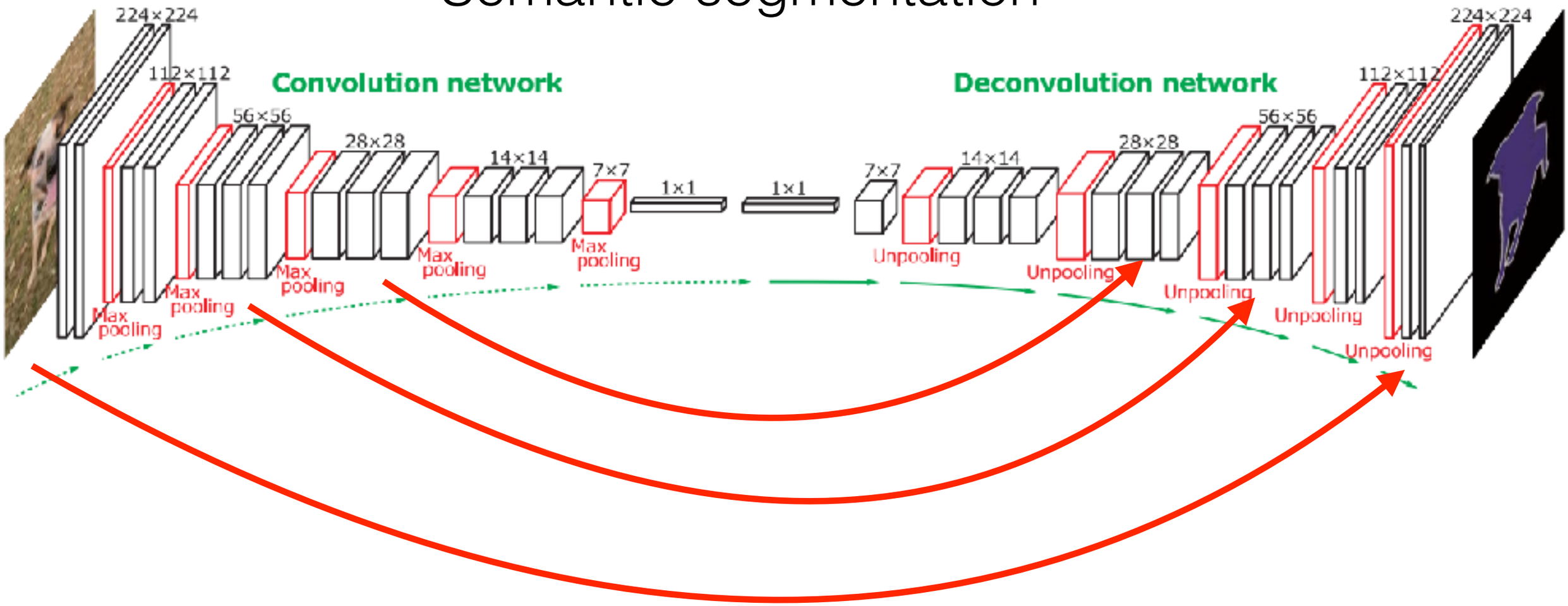
Semantic segmentation



concatenate deconvolution feature map with the original feature map

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

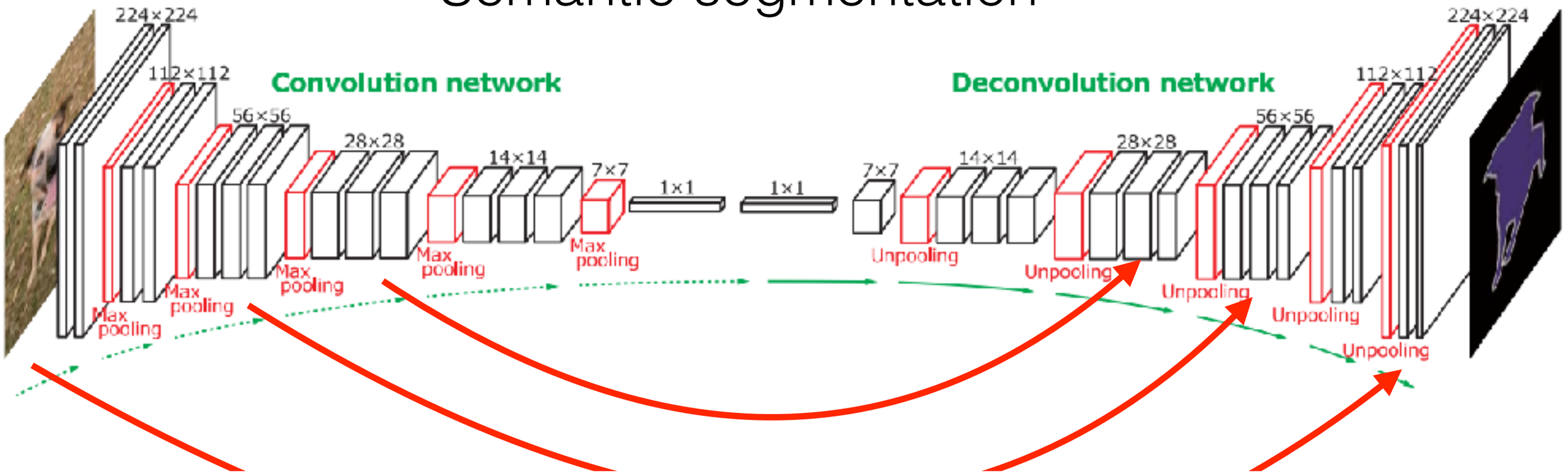
Semantic segmentation



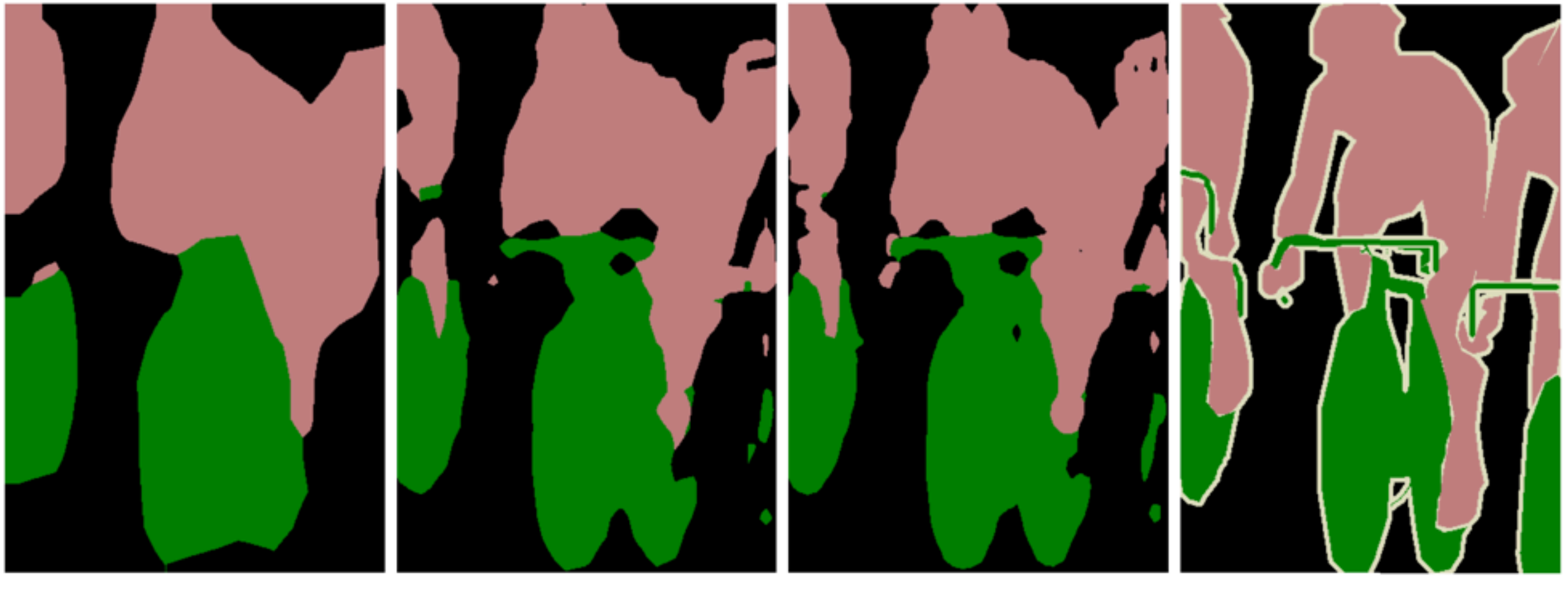
concatenate deconvolution feature map with the original feature map

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

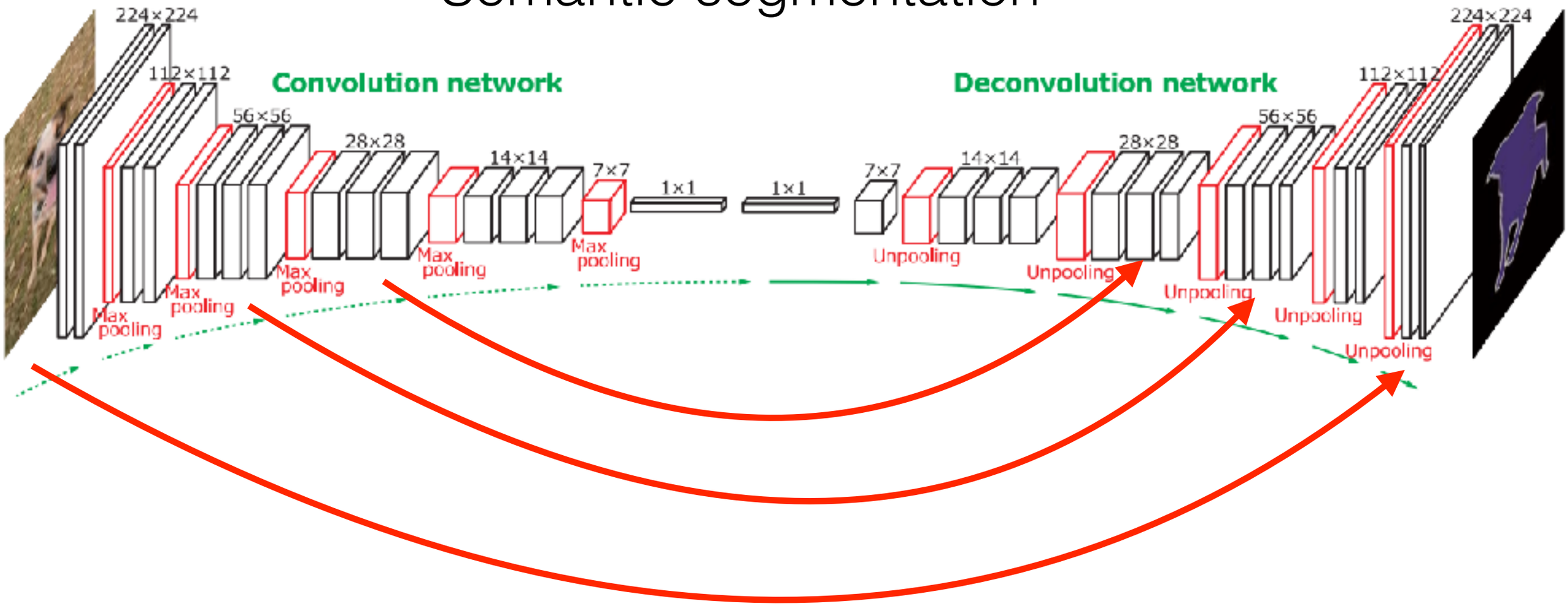
Semantic segmentation



FCN-32s FCN-16s FCN-8s Ground truth

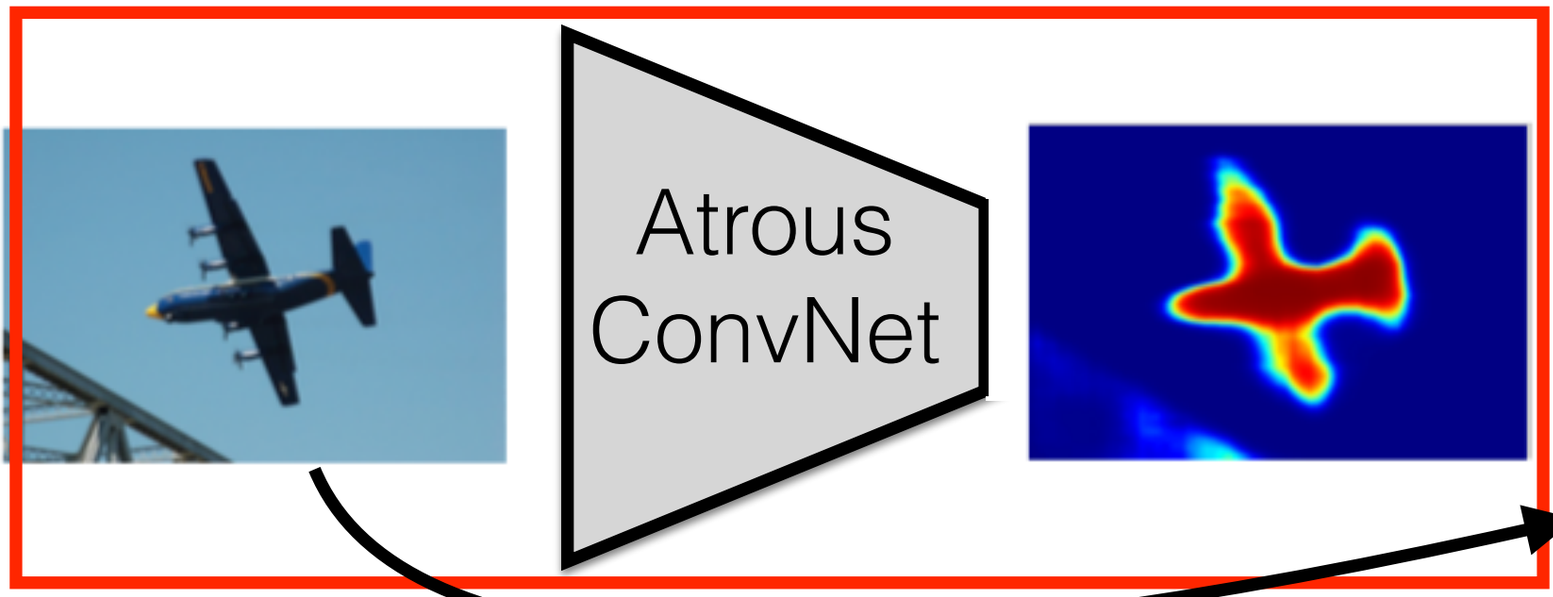


Semantic segmentation



- 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

DeepLab v3



- Replace “maxpooling+conv” by “atrous convolution”
- Replace deconvolutions by bi-linear interp+CRF

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

Convolution layer

Dilatation rate = 1

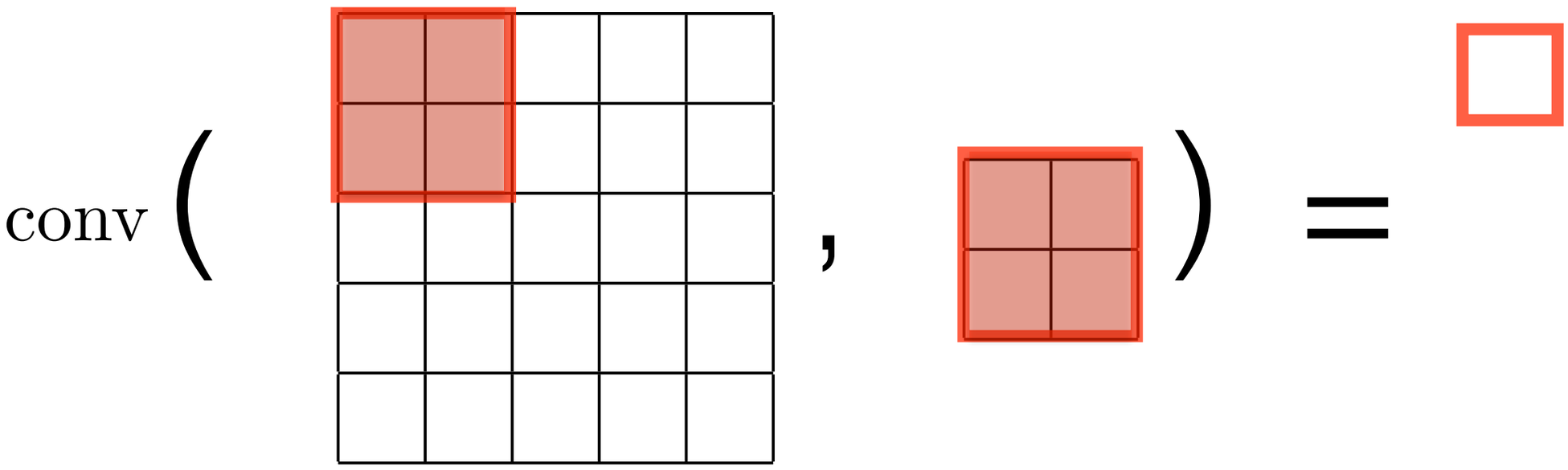


image
(5x5)

kernel
(2x2)

output
(**? x ?**)

Atrous convolution layer

Dilatation rate = 2

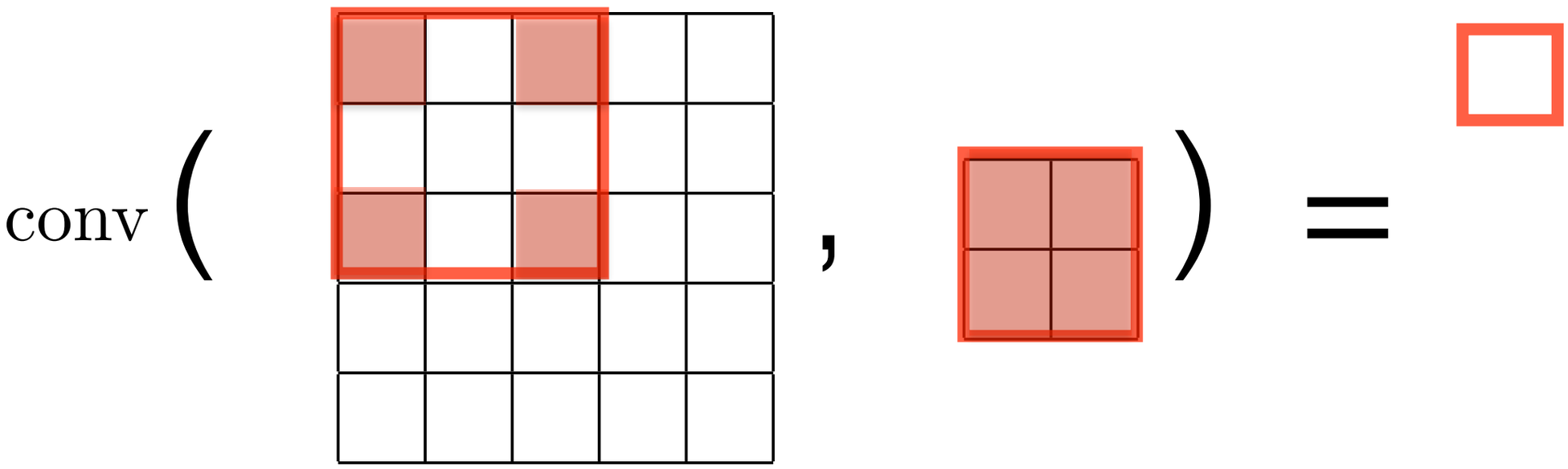
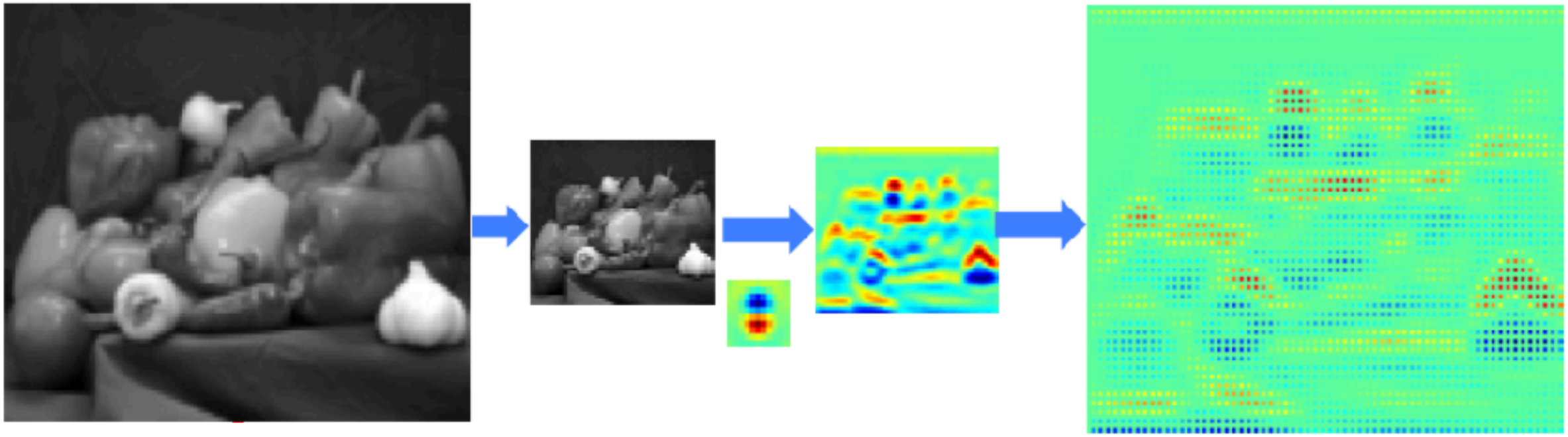


image
(5x5)

kernel
(2x2)

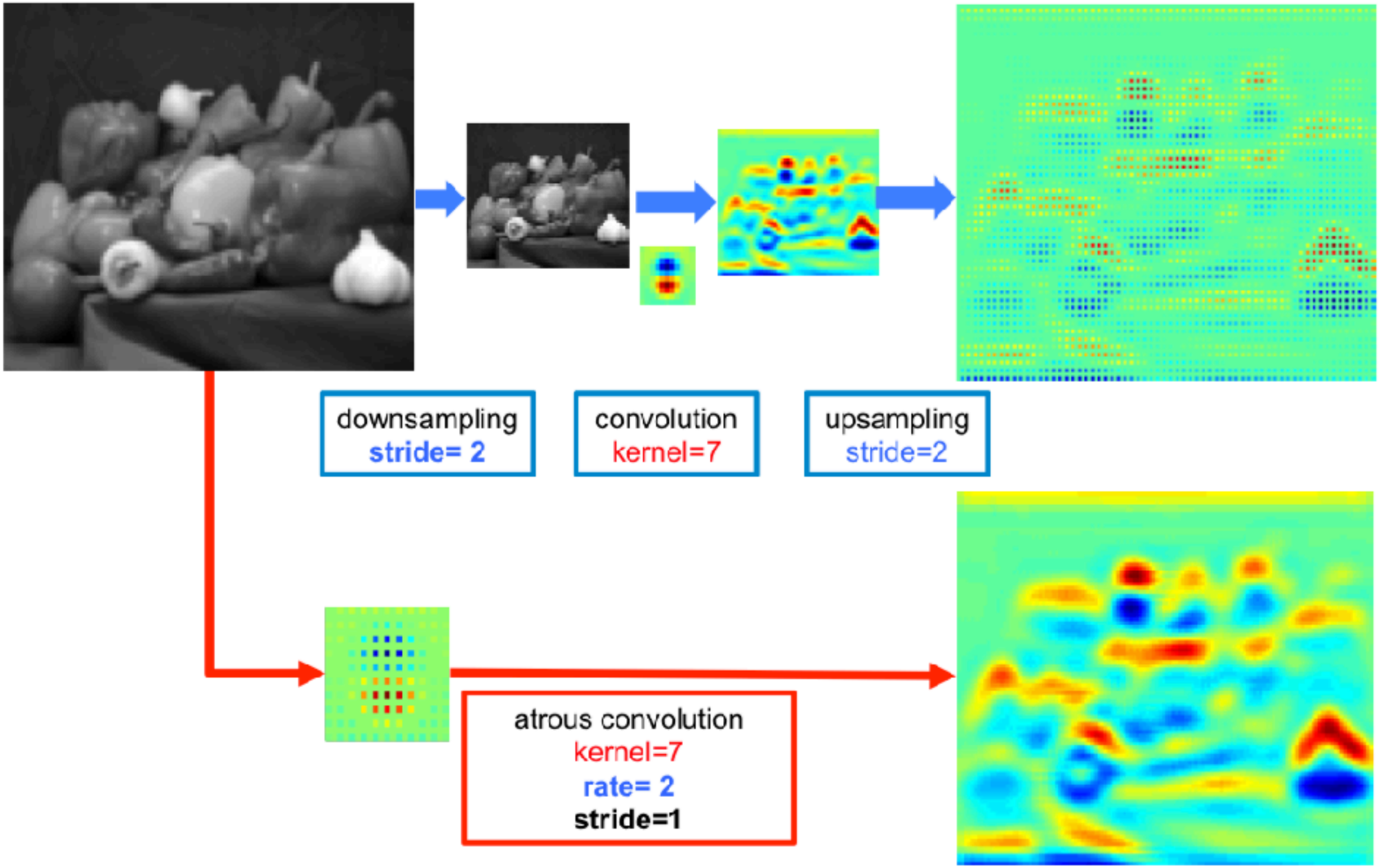
output
(**? x ?**)

Atrous vs standard convolution for segmentation



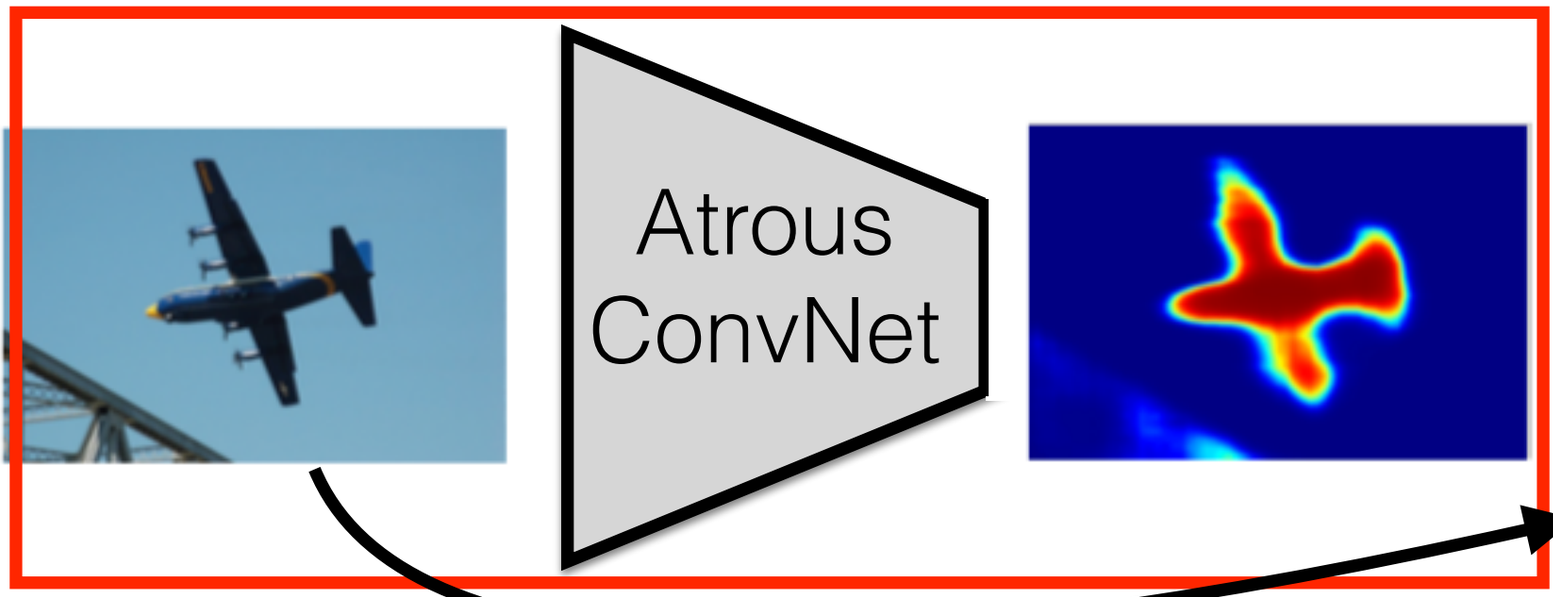
[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

Atrous vs standard convolution for segmentation



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

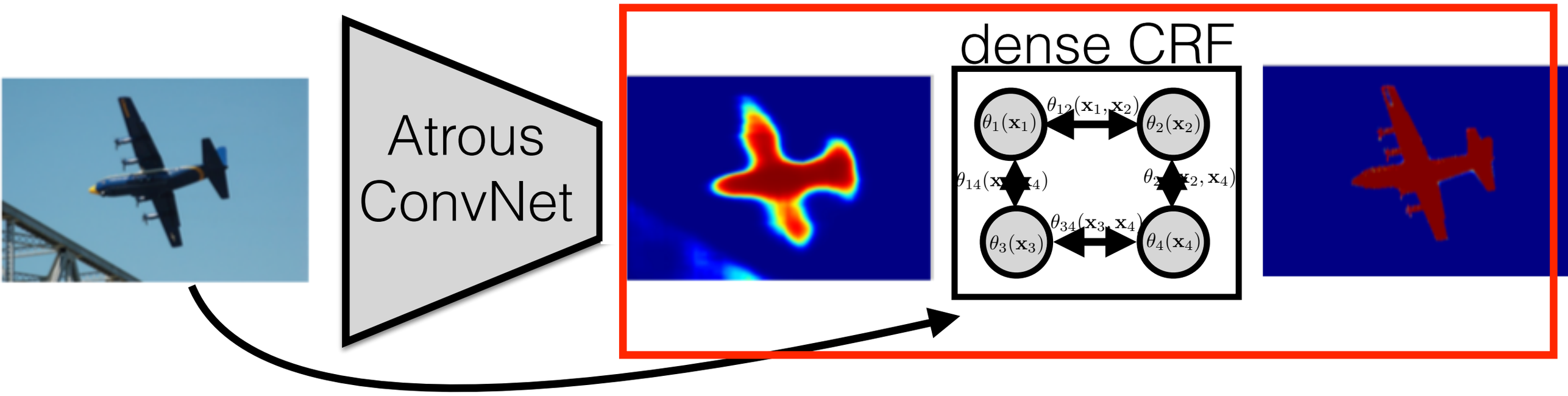
DeepLab v3



- Replace “maxpooling+conv” by “atrous convolution”
- Replace deconvolutions by bi-linear interp+CRF

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

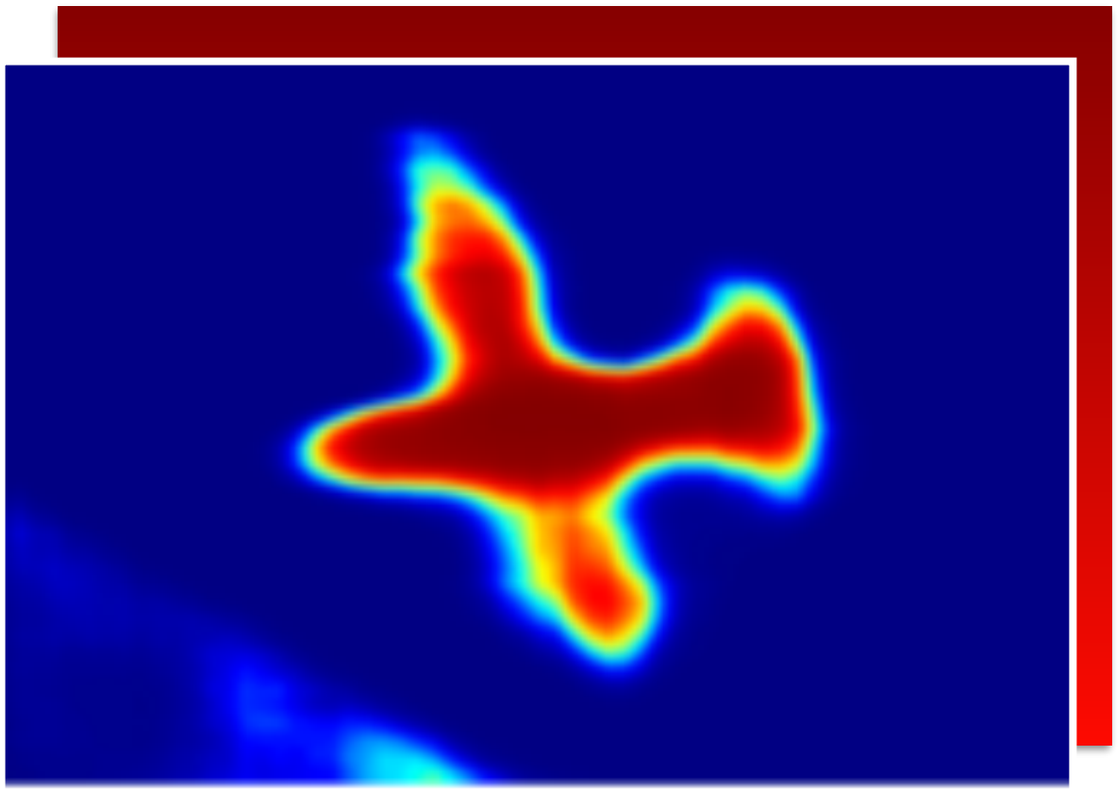
DeepLab v3



- 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 - Conditional Random Fields (CRF)

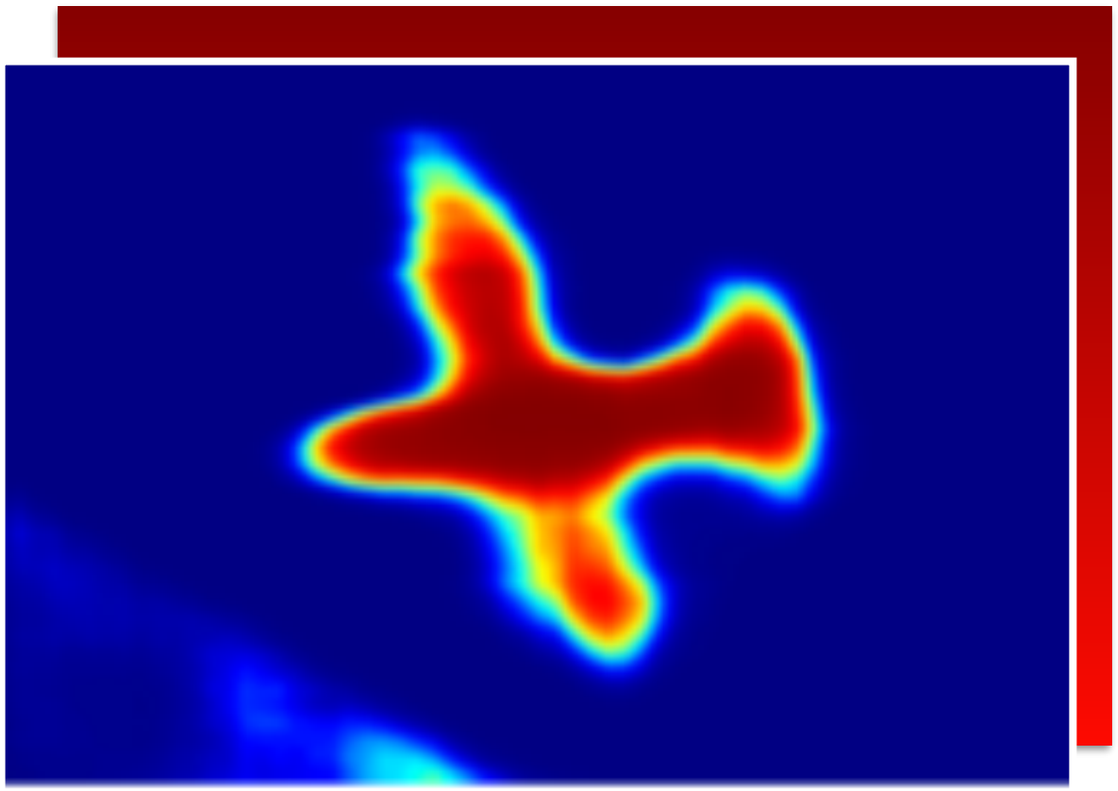


$p(\mathbf{x}_i)$

label of pixel i
$\mathbf{x}_i \in \{0, 1\}$

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

DeepLab v3 - Conditional Random Fields (CRF)

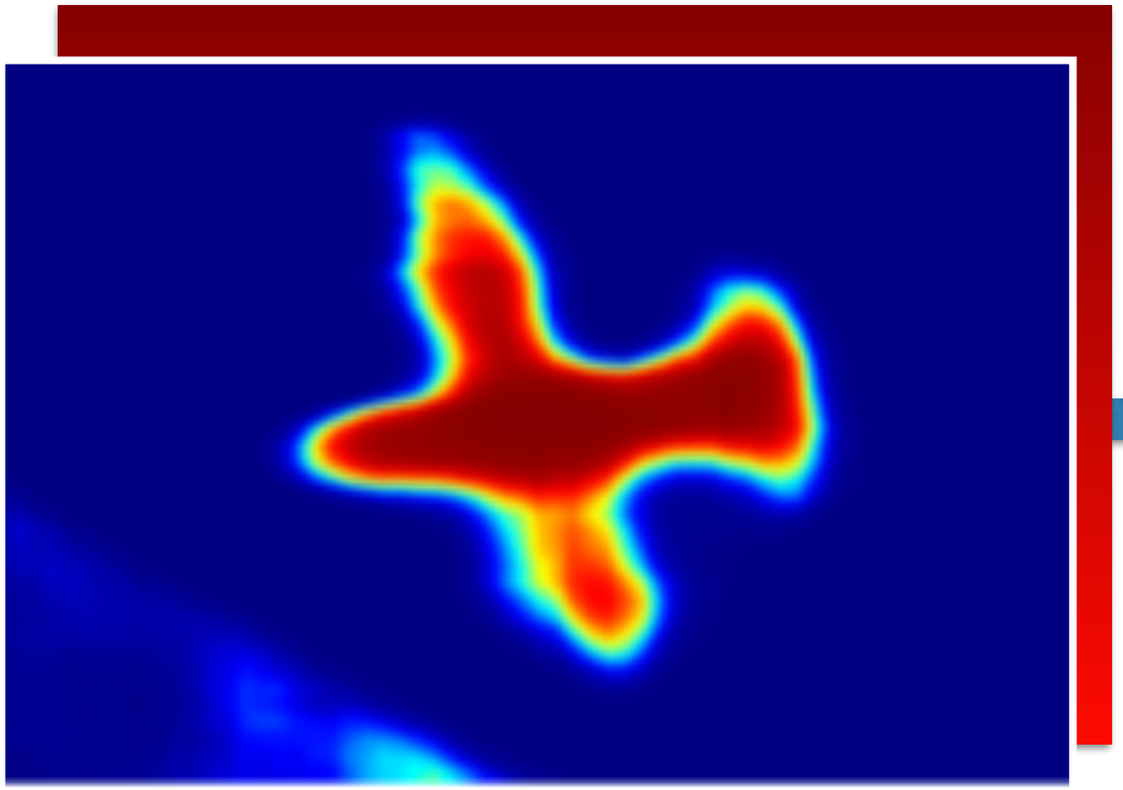


$p(\mathbf{x}_i)$

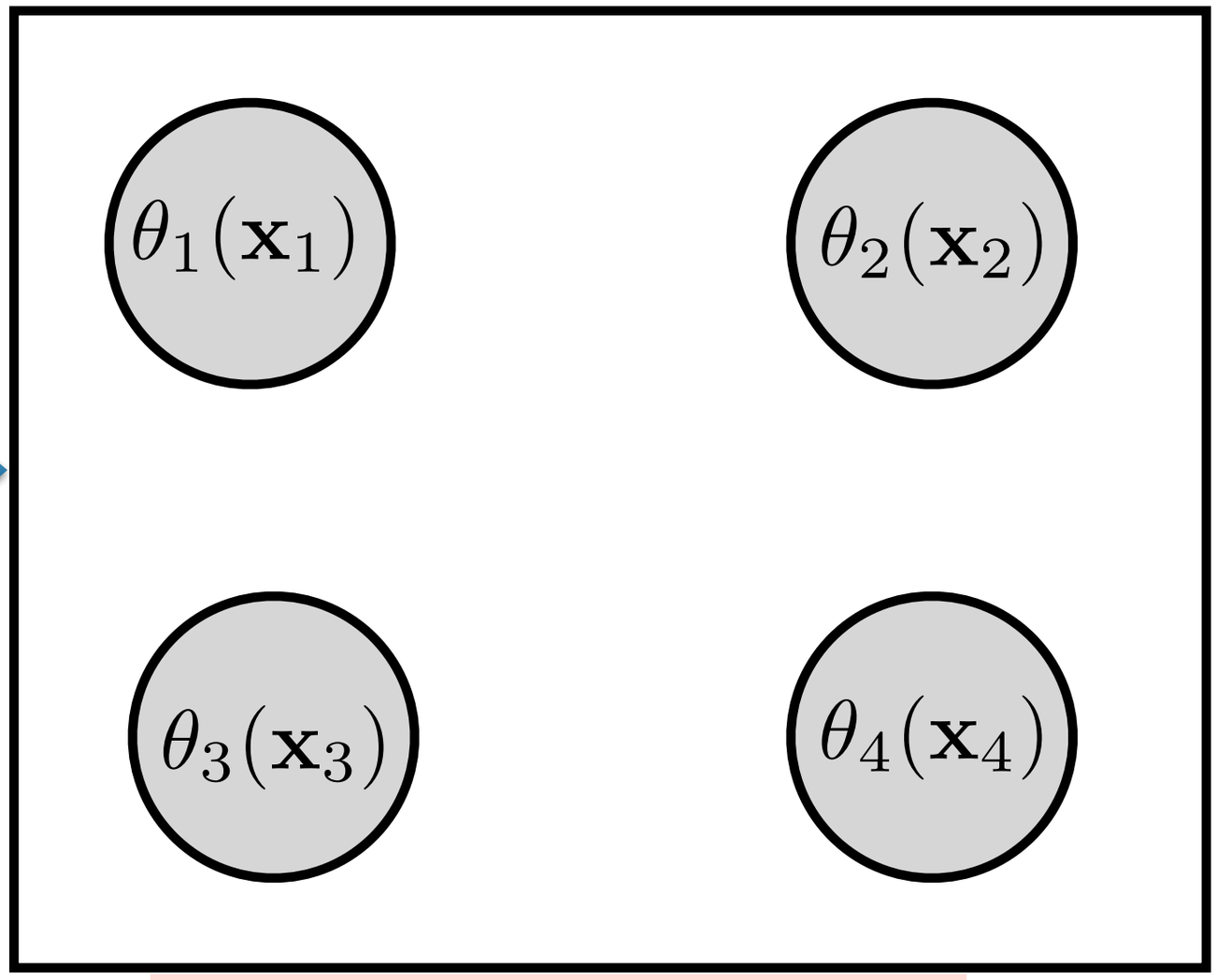
output of DCNN in pixel i
(probability that pixel i has label $\mathbf{x}_i$)

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

DeepLab v3 - Conditional Random Fields (CRF)



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



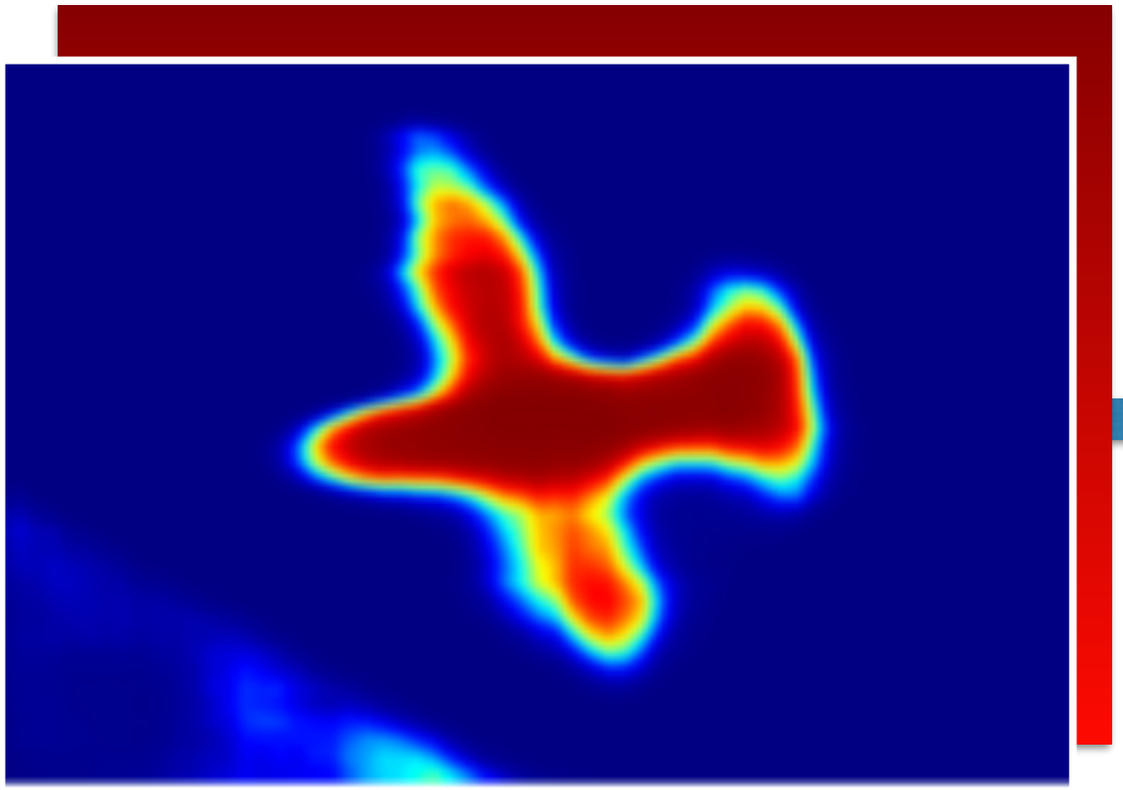
$$\theta_i(\mathbf{x}_i) = -\log(p(\mathbf{x}_i))$$

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

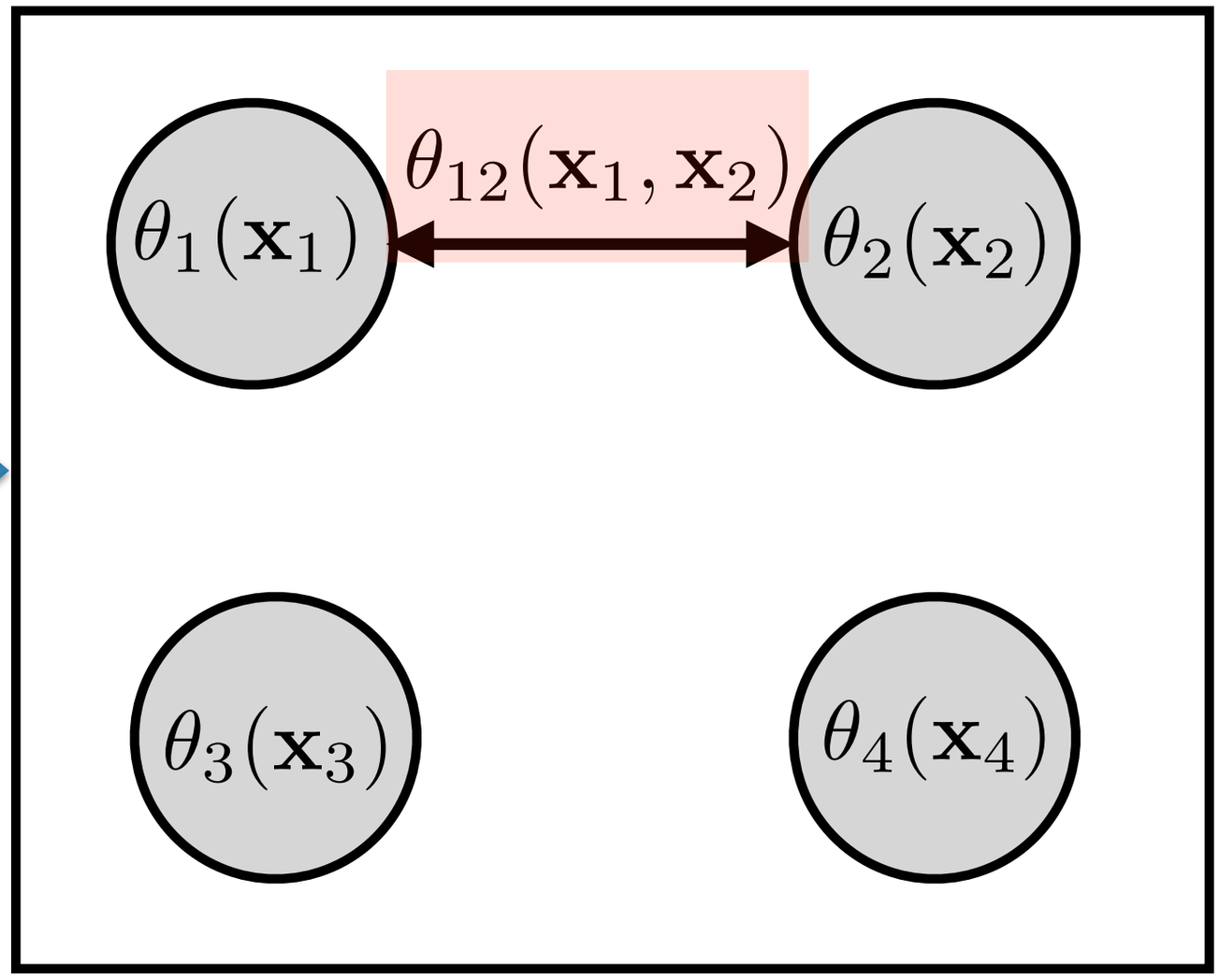
$$E(\mathbf{x}) = \sum_i \theta_i(\mathbf{x}_i) \Rightarrow \arg\min_{\mathbf{x}\in\{0,1\}^N} E(\mathbf{x}) \Rightarrow \text{greedy solution}$$

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

DeepLab v3 - Conditional Random Fields (CRF)



$p(\mathbf{x}_i)$

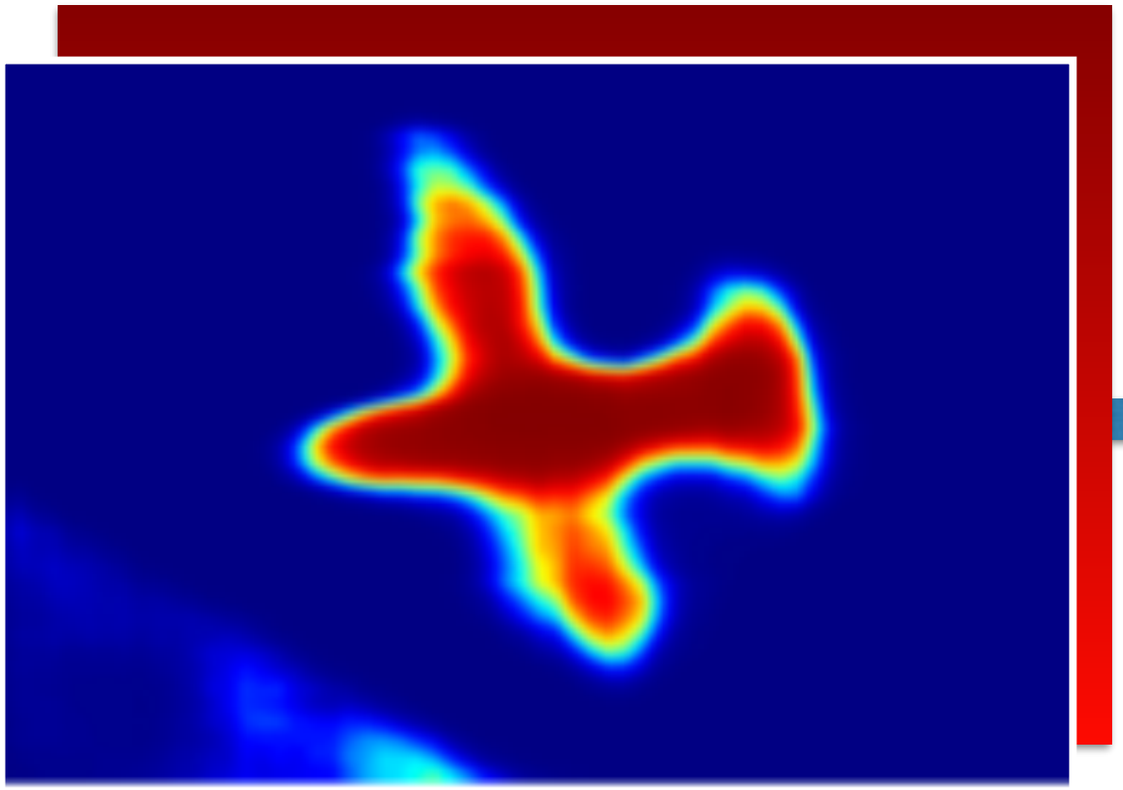


$$\theta_i(\mathbf{x}_i) = -\log(p(\mathbf{x}_i))$$

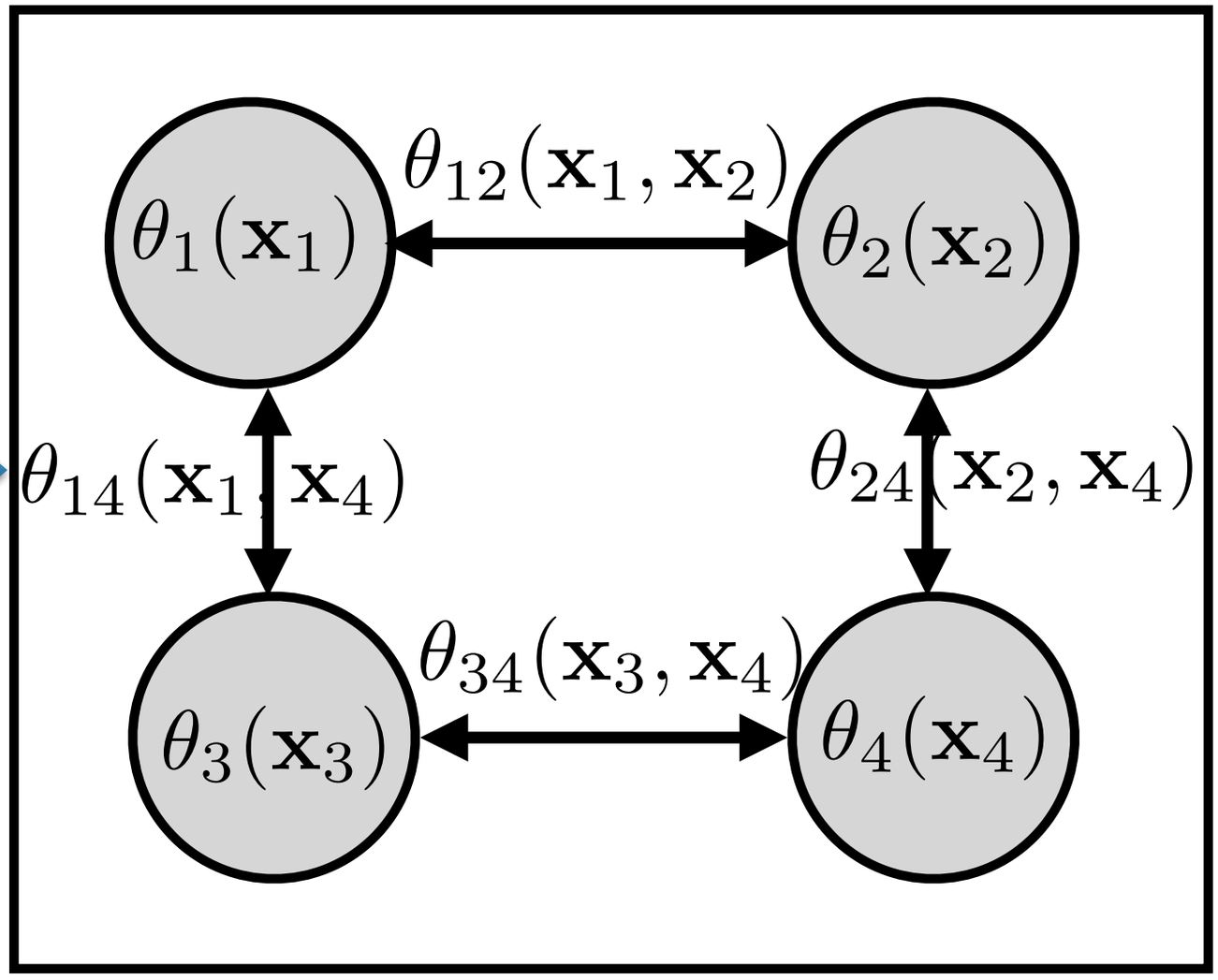
$\theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ = “penalty for dissimilar labels to similar pixels”

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

DeepLab v3 - Conditional Random Fields (CRF)



$p(\mathbf{x}_i)$

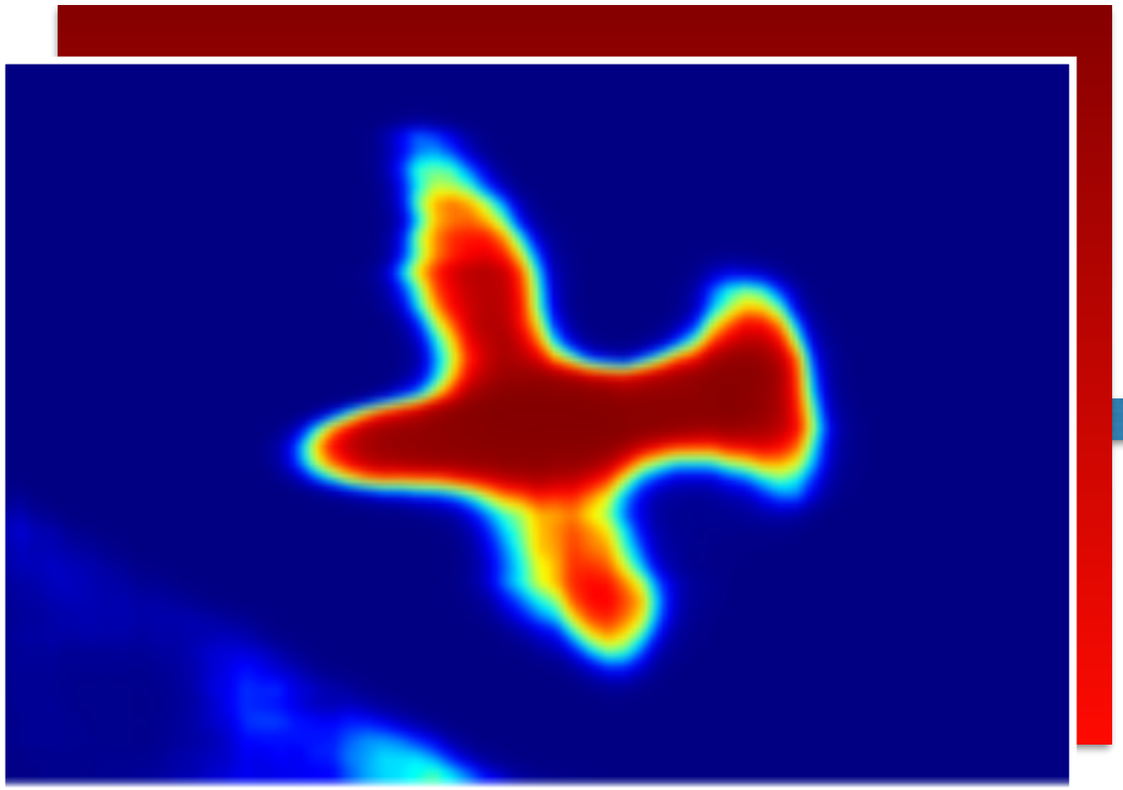


$$\theta_i(\mathbf{x}_i) = -\log(p(\mathbf{x}_i))$$

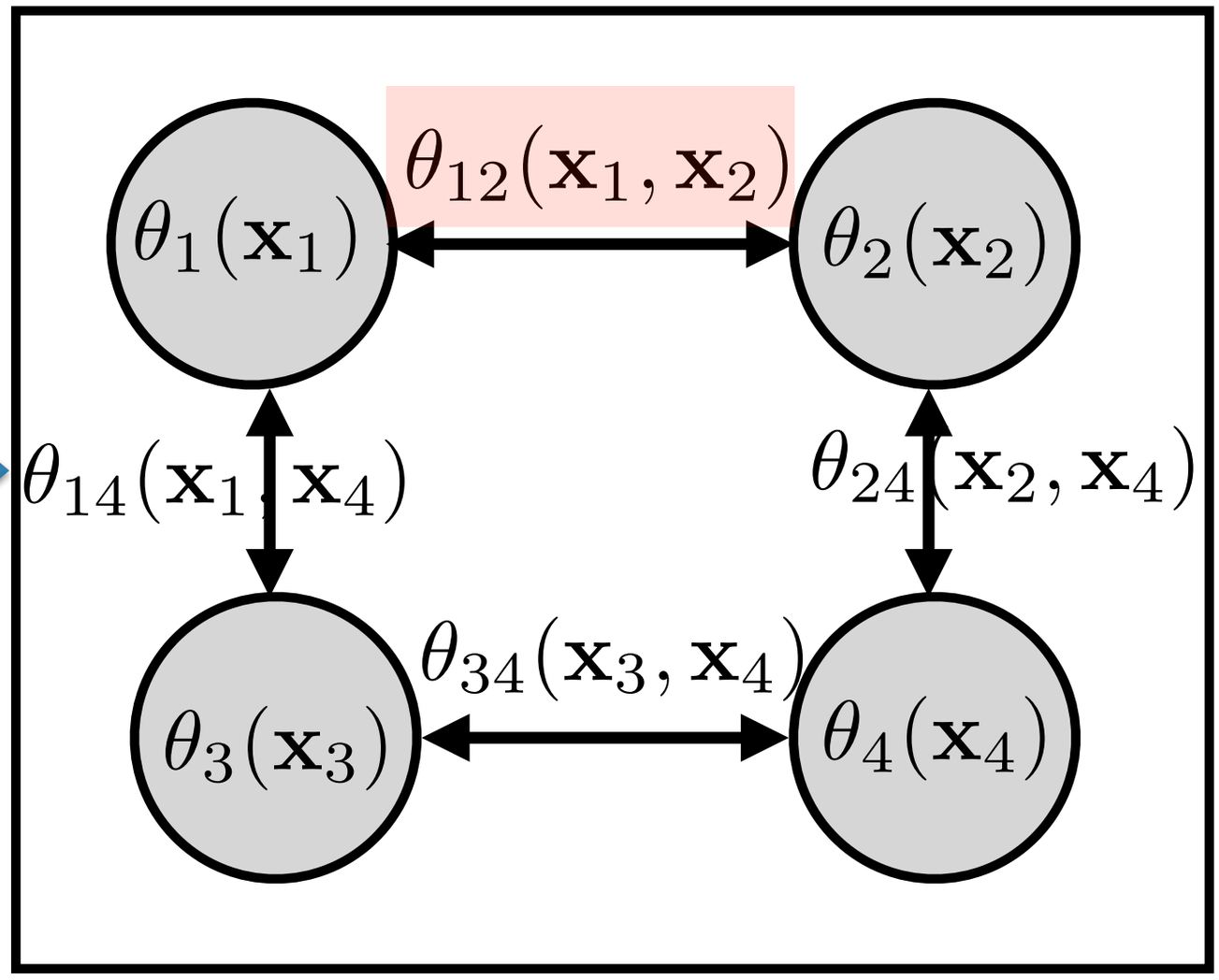
$\theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ = “penalty for dissimilar labels to similar pixels”

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

DeepLab v3 - Conditional Random Fields (CRF)

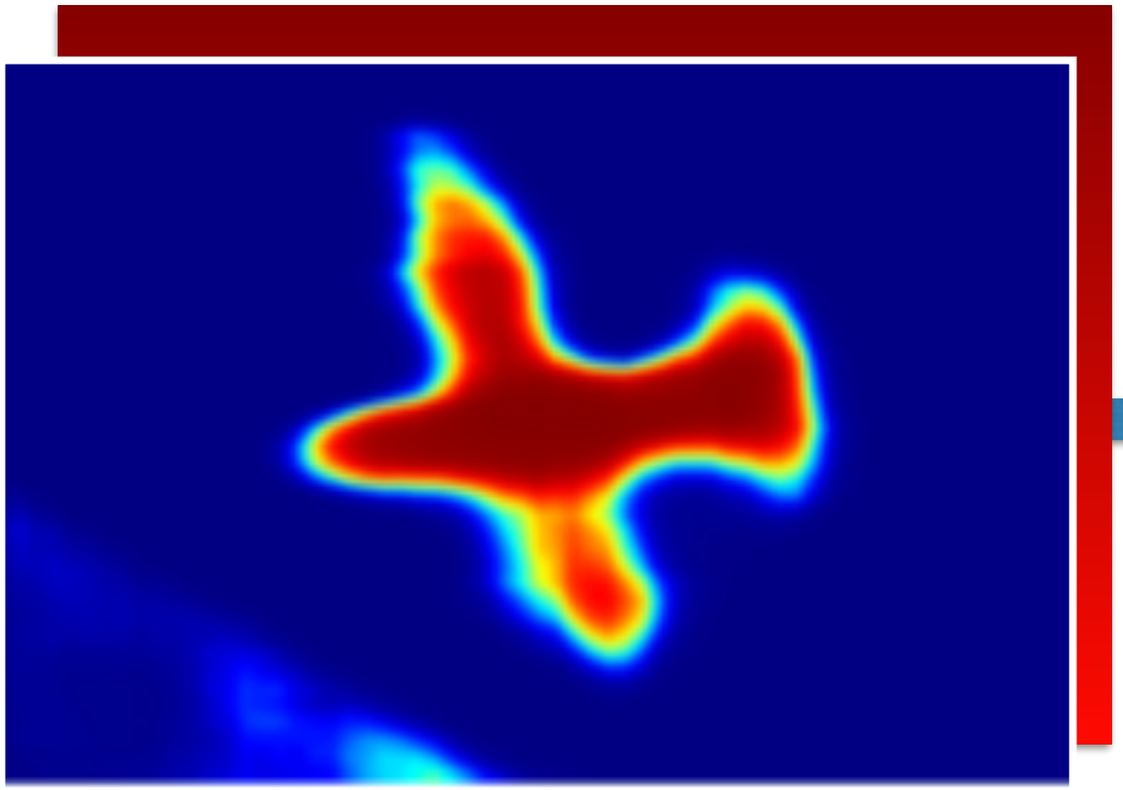


$p(\mathbf{x}_i)$

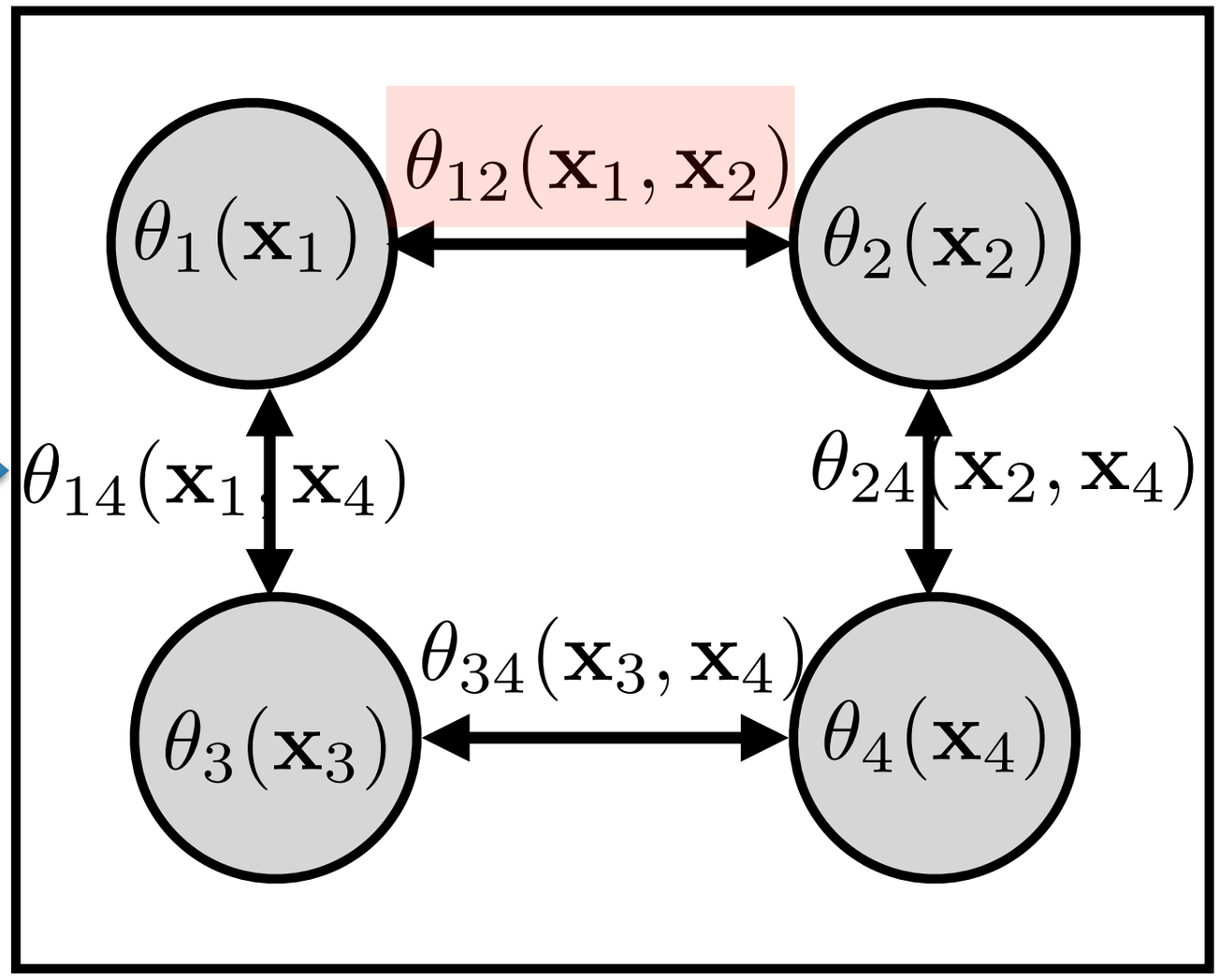


$$\theta_{ij}(x_i, x_j) =$$

DeepLab v3 - Conditional Random Fields (CRF)



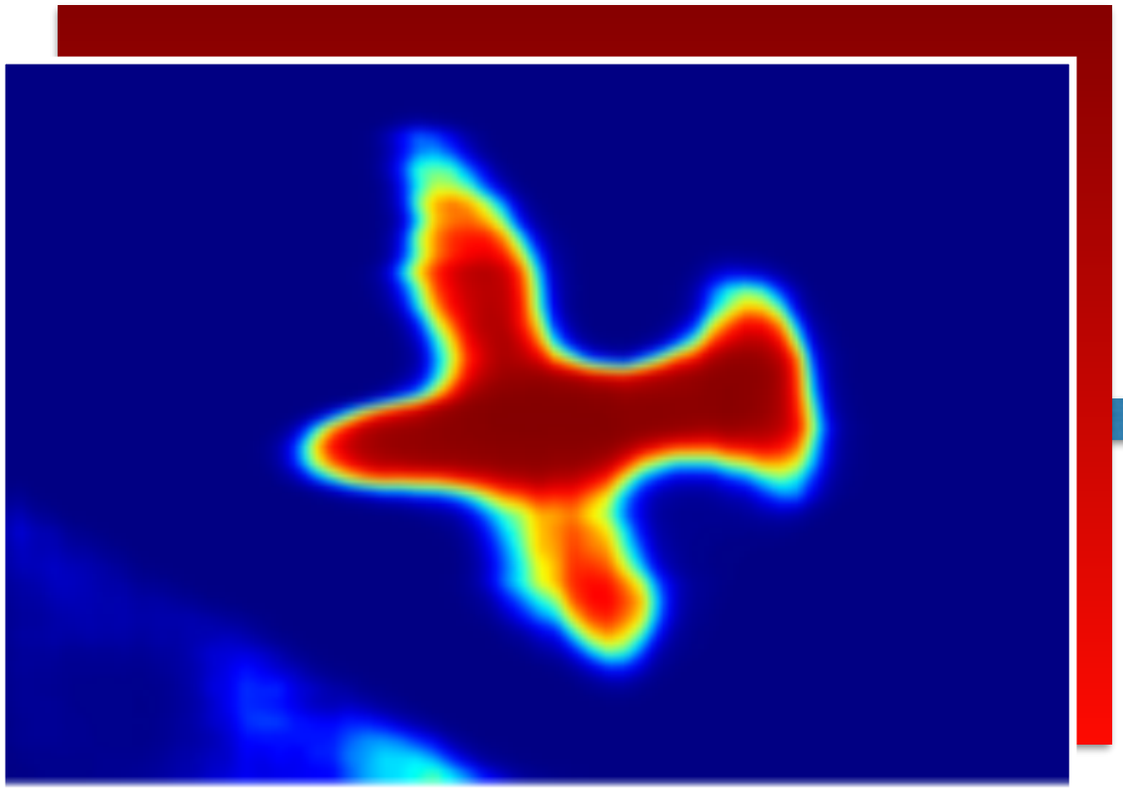
$p(\mathbf{x}_i)$



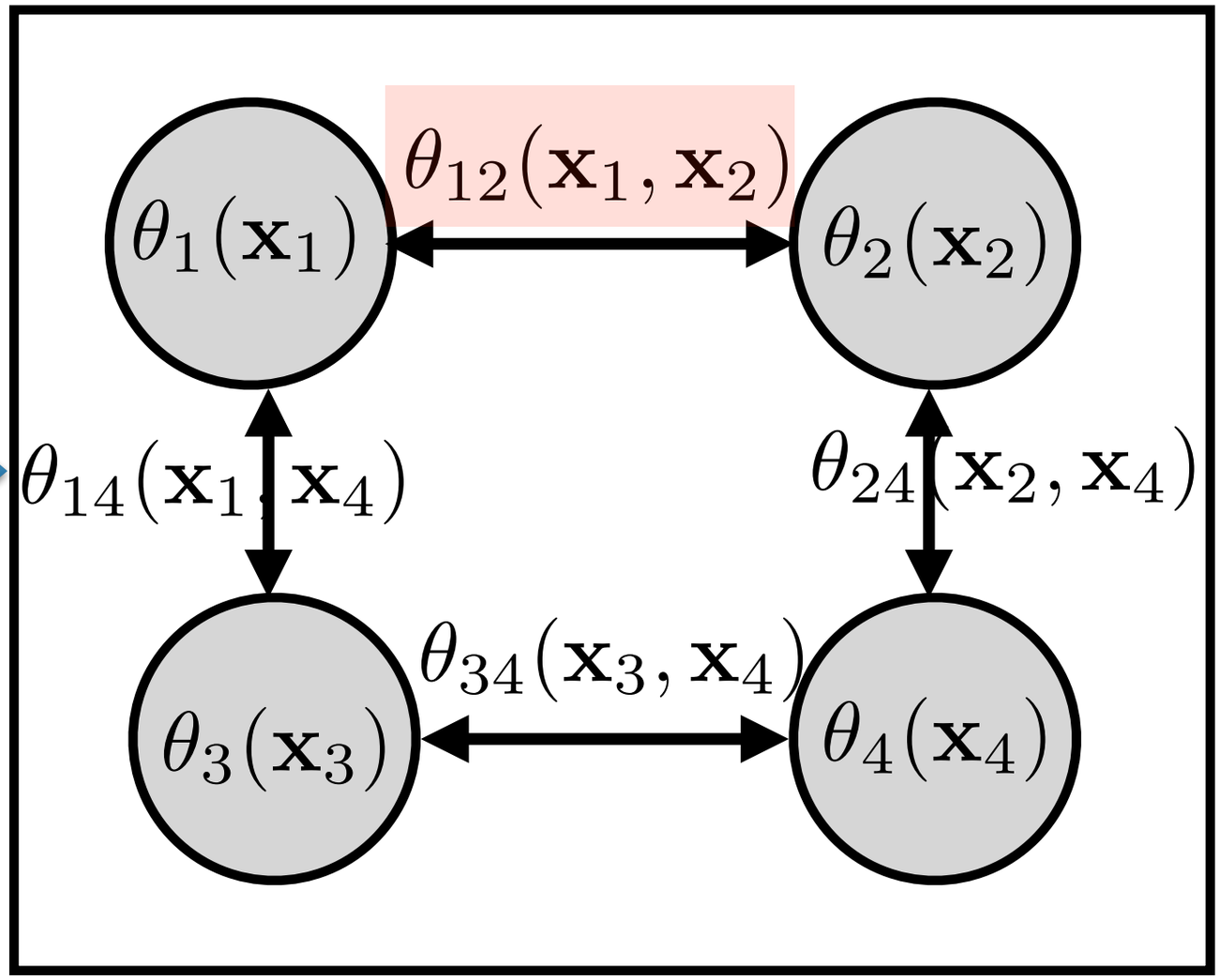
$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \begin{cases} 1 & x_i \neq x_j \\ 0 & x_i = x_j \end{cases}$$

same labels are not penalized

DeepLab v3 - Conditional Random Fields (CRF)



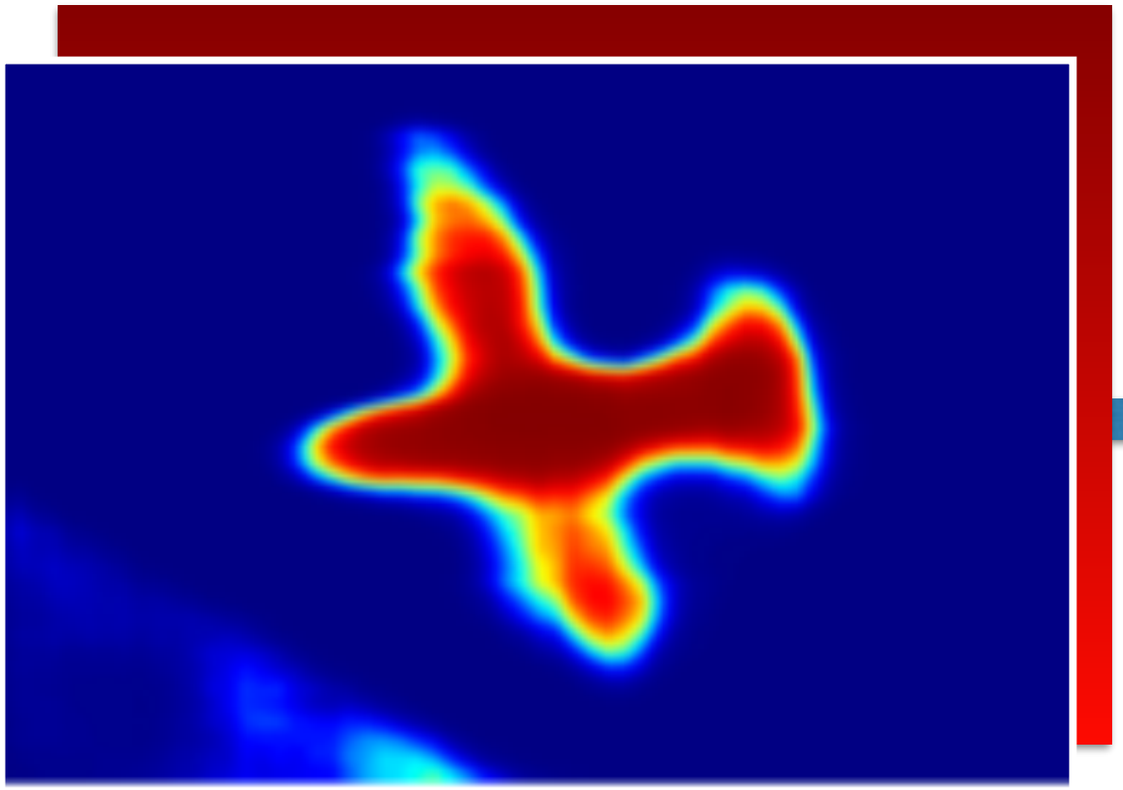
$p(\mathbf{x}_i)$



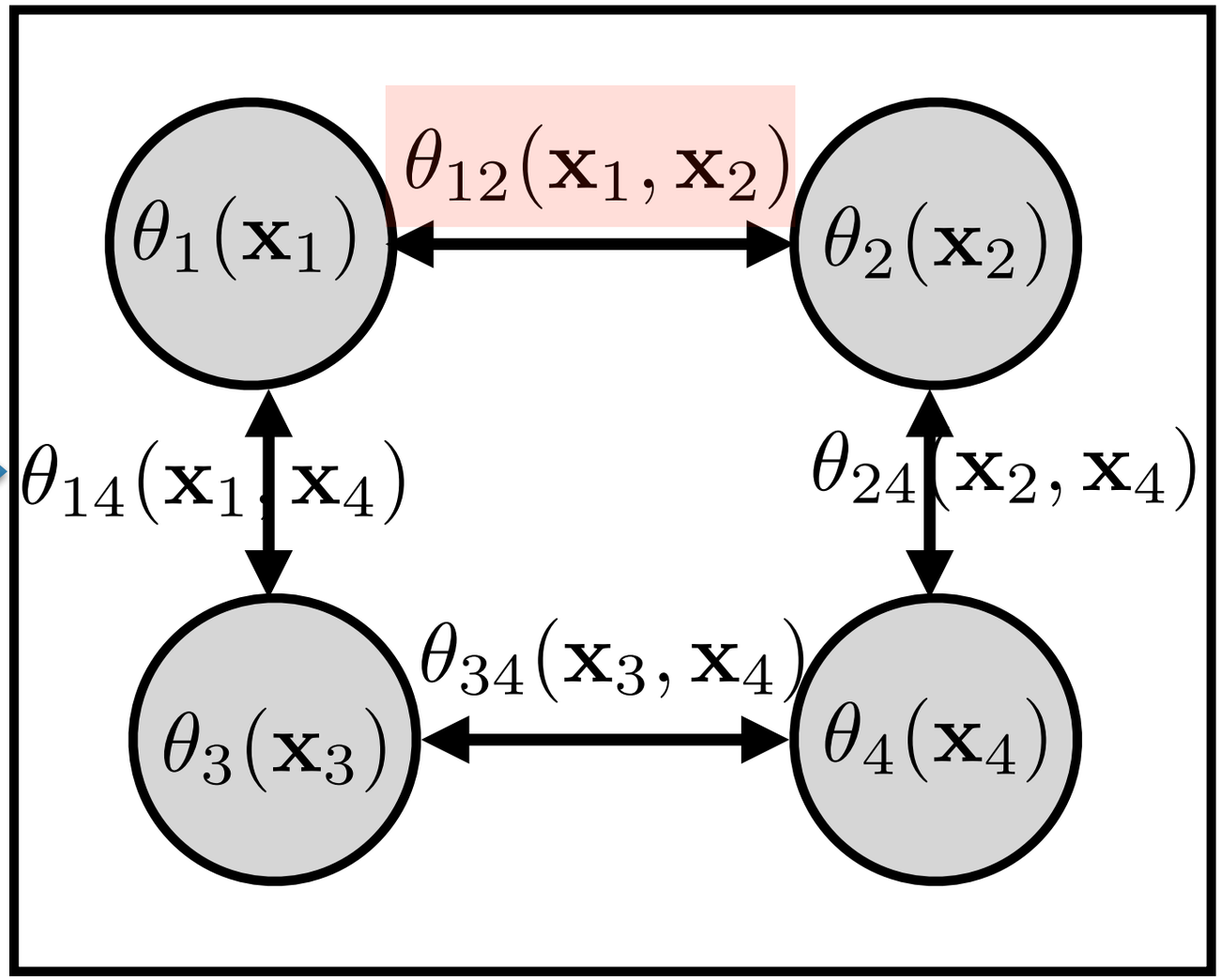
$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \left[w_1 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2} \right) \right.$$

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

DeepLab v3 - Conditional Random Fields (CRF)



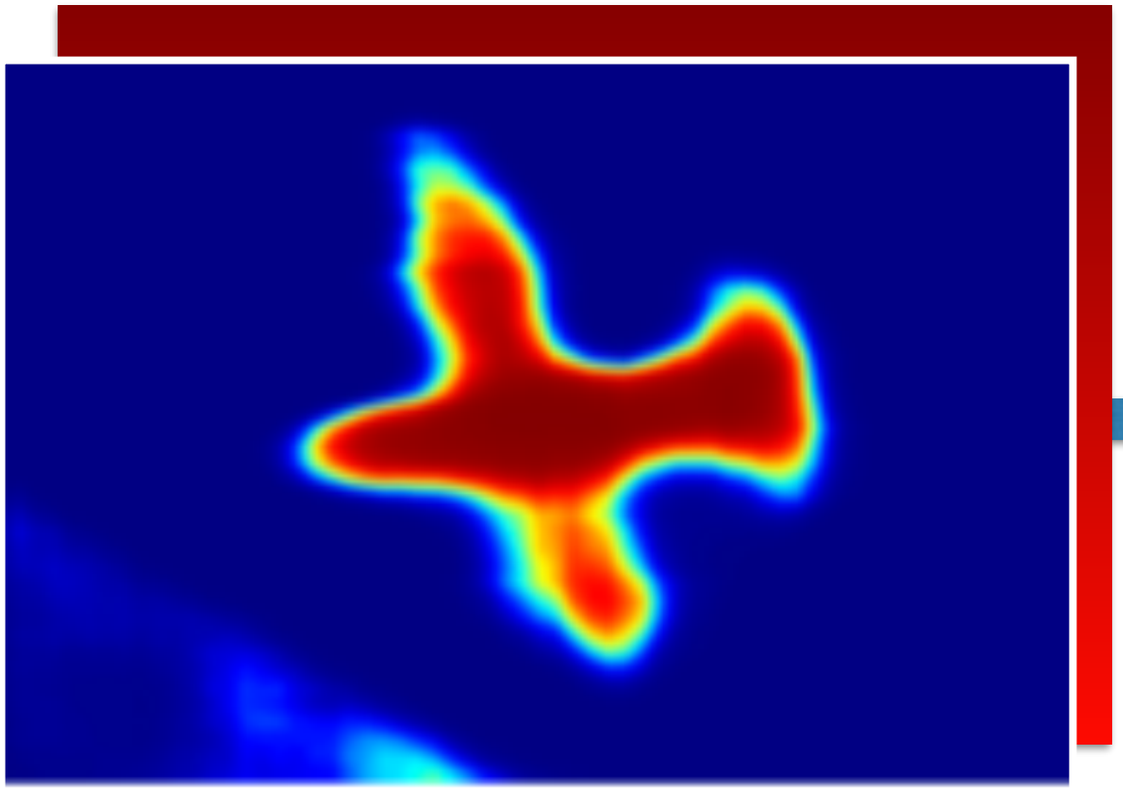
$p(\mathbf{x}_i)$



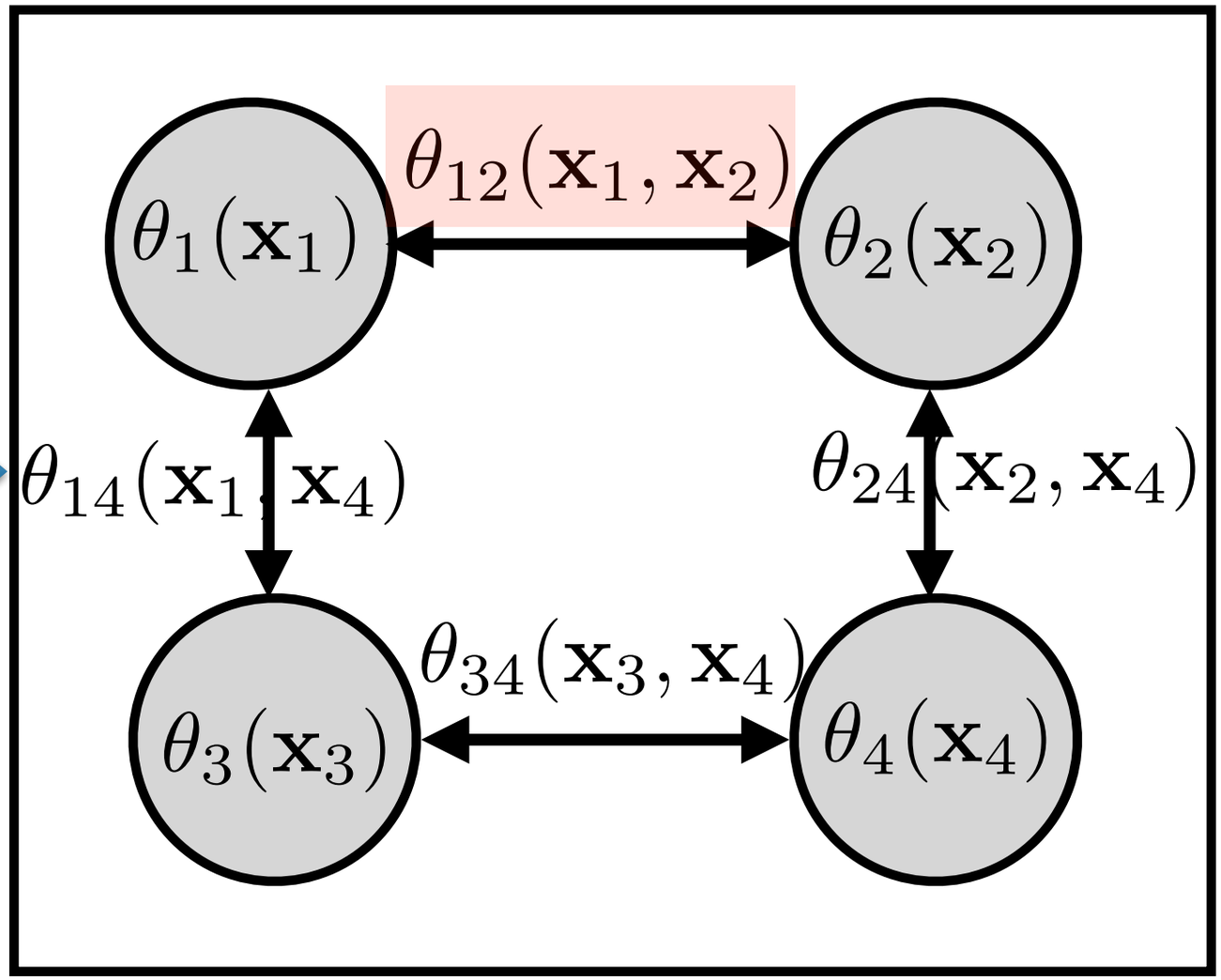
$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \left[w_1 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2} \right) \right.$$

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

DeepLab v3 - Conditional Random Fields (CRF)



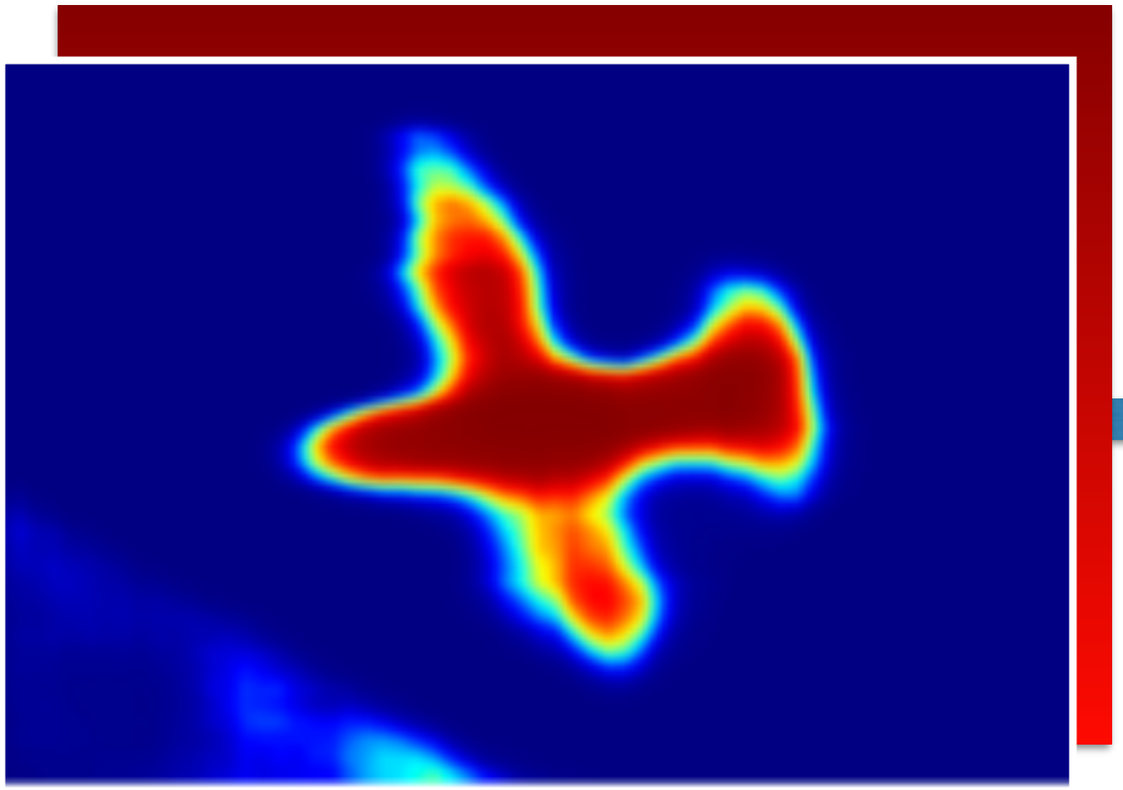
$p(\mathbf{x}_i)$



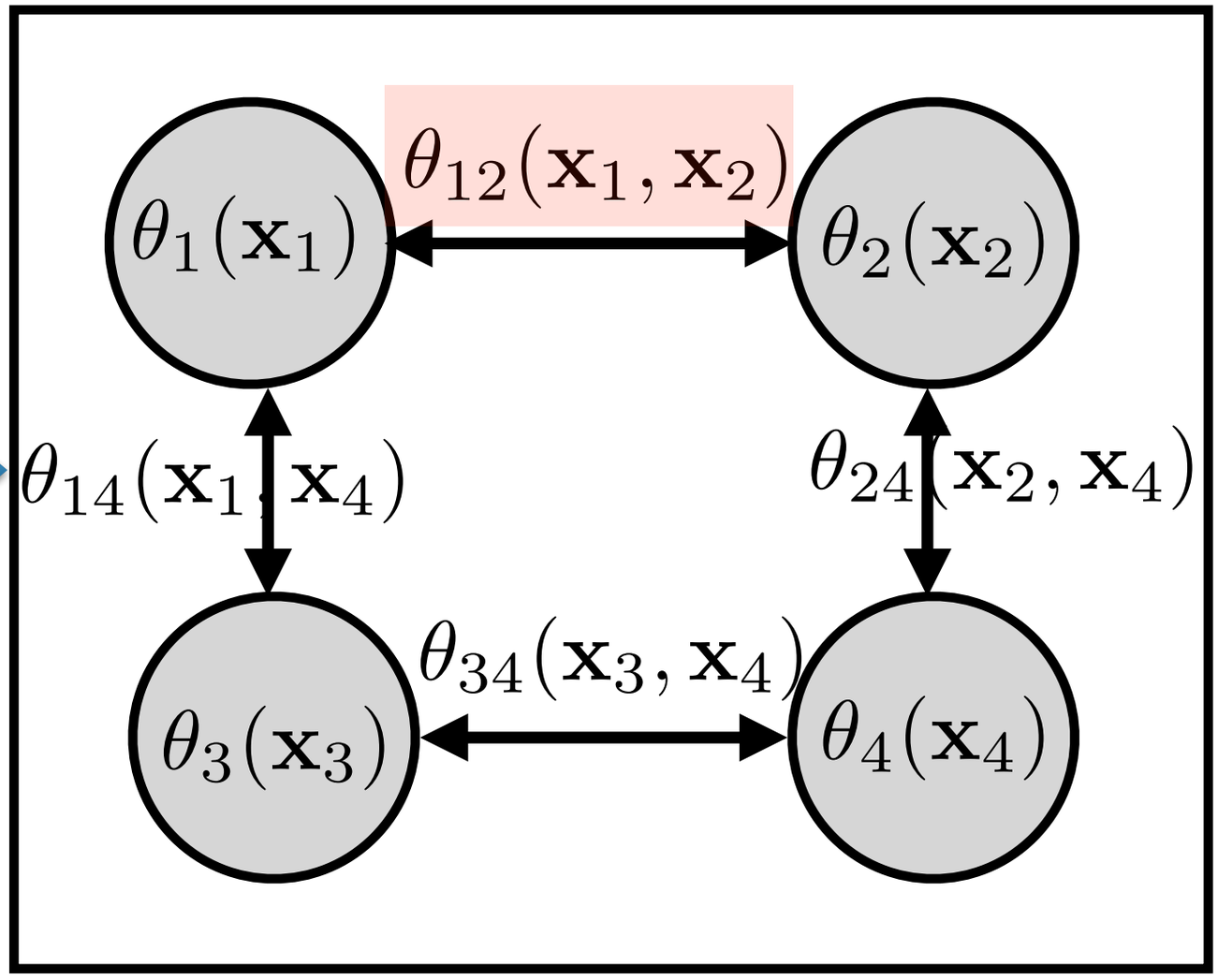
$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \left[w_1 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2} \right) \right.$$

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

DeepLab v3 - Conditional Random Fields (CRF)



$p(\mathbf{x}_i)$

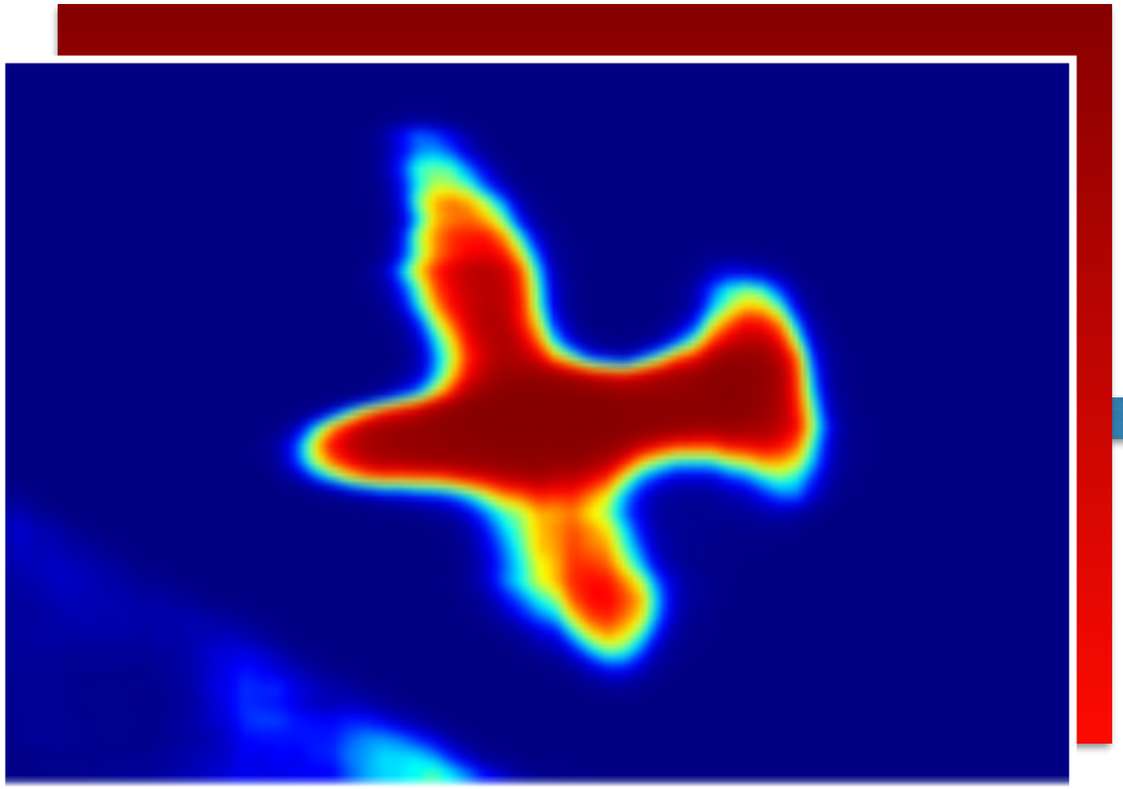


$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \left[w_1 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2} \right) \right.$$

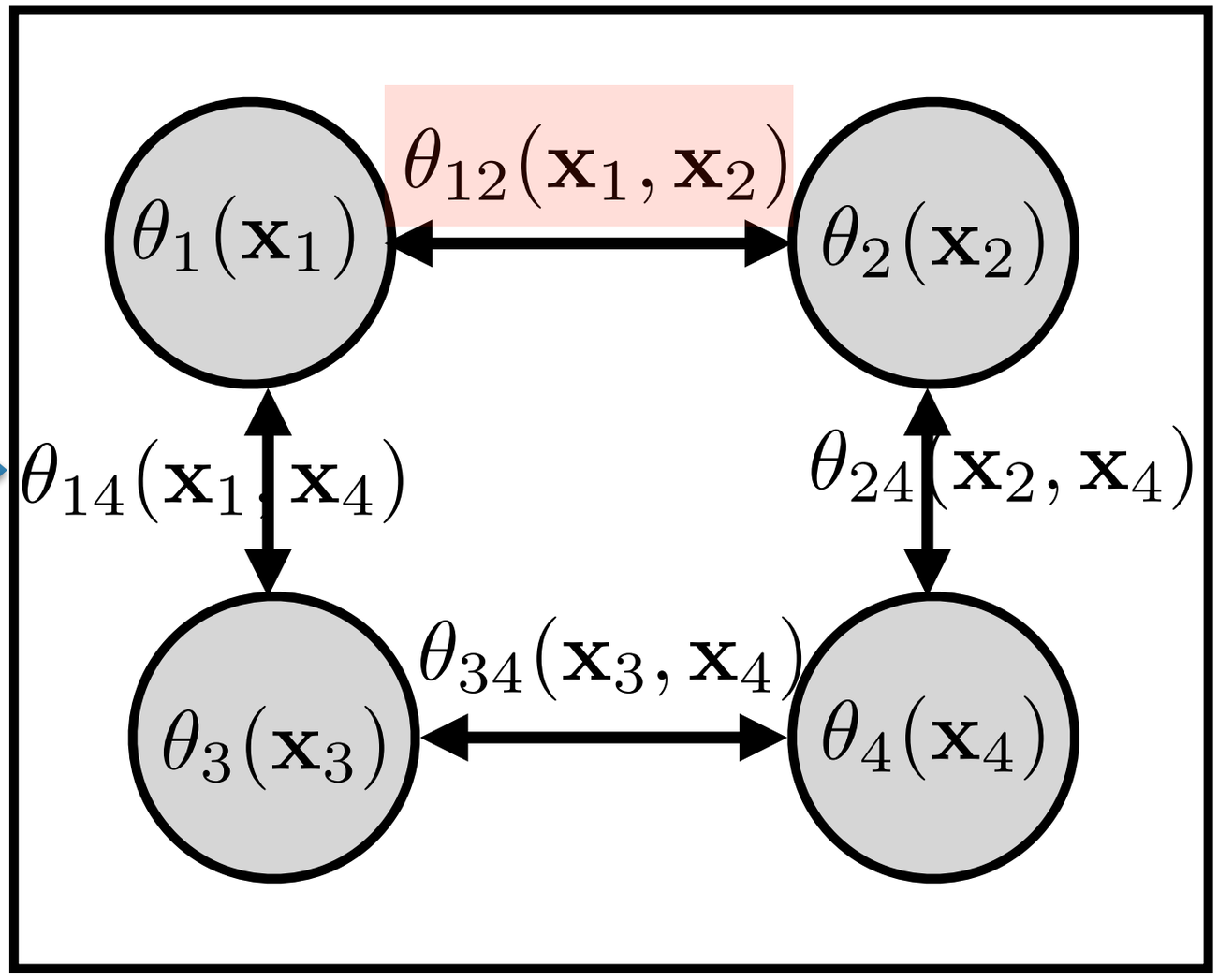
another penalty for close pixels

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

DeepLab v3 - Conditional Random Fields (CRF)



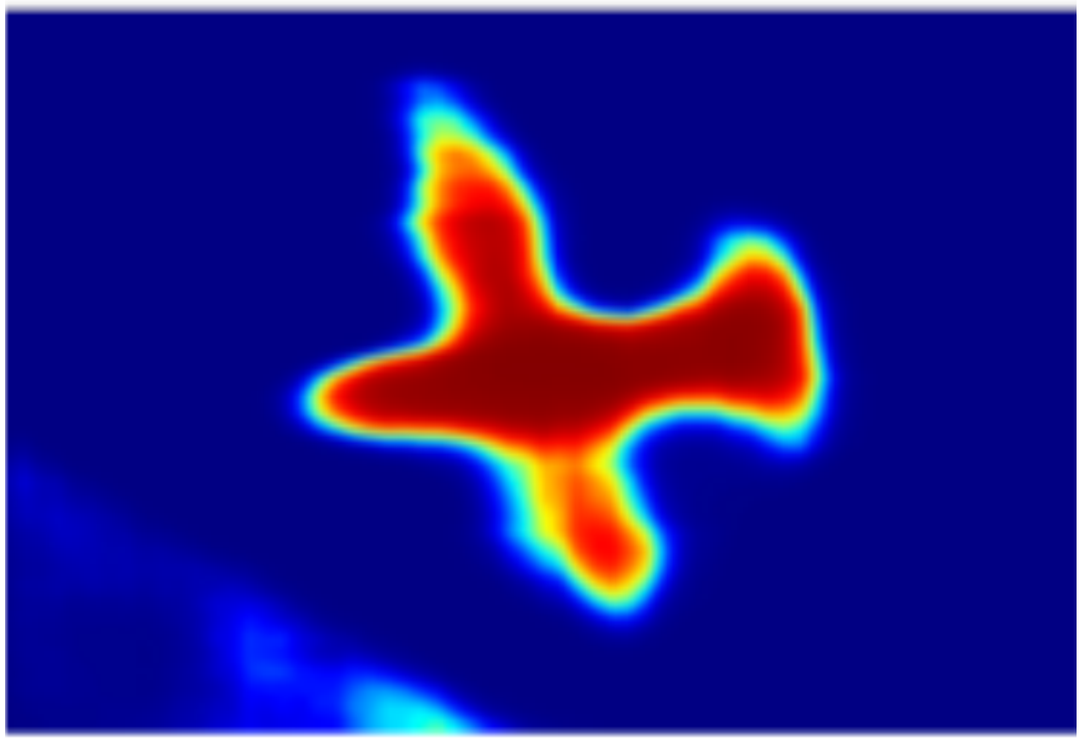
$p(\mathbf{x}_i)$



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

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

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)

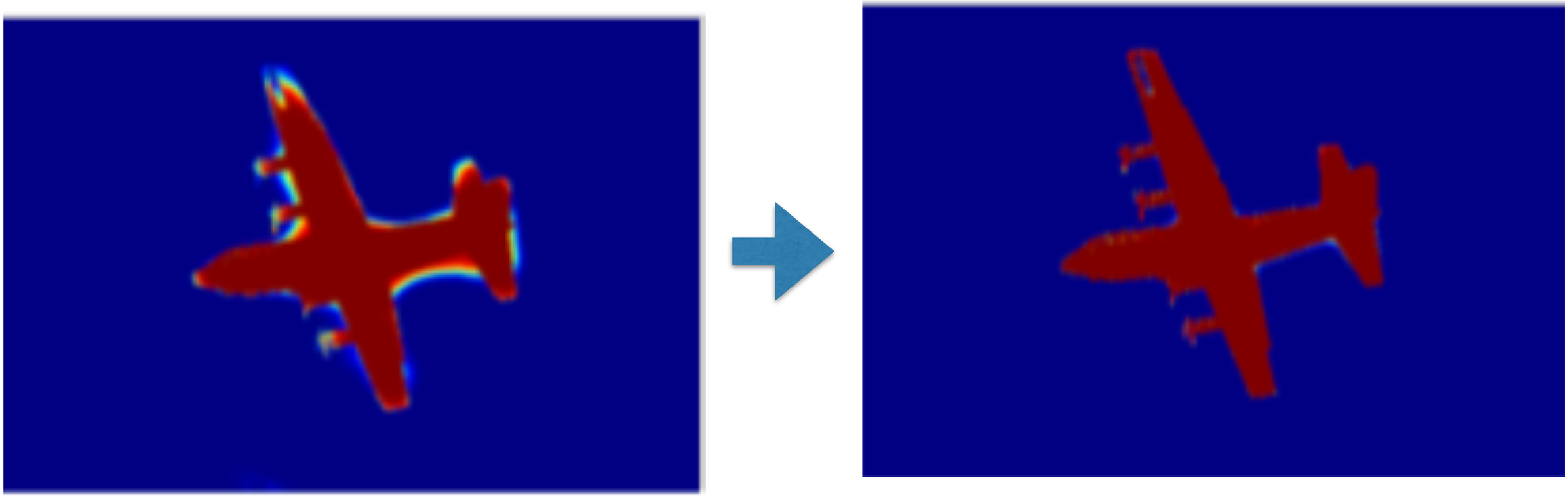
$p(\mathbf{x}_i)$

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

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

iterate

DeepLab v3 - Conditional Random Fields (CRF)



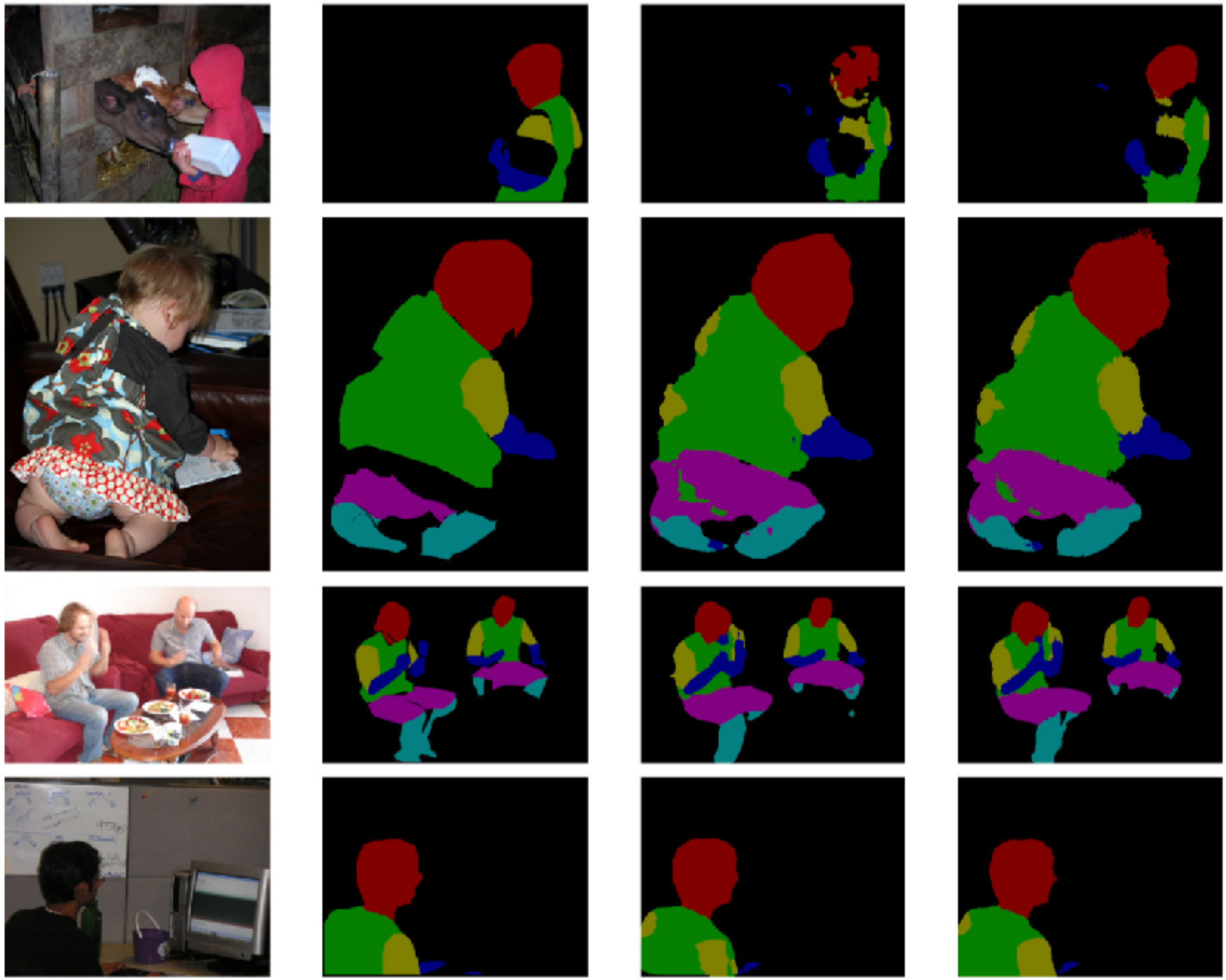
$p(\mathbf{x}_i)$

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

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

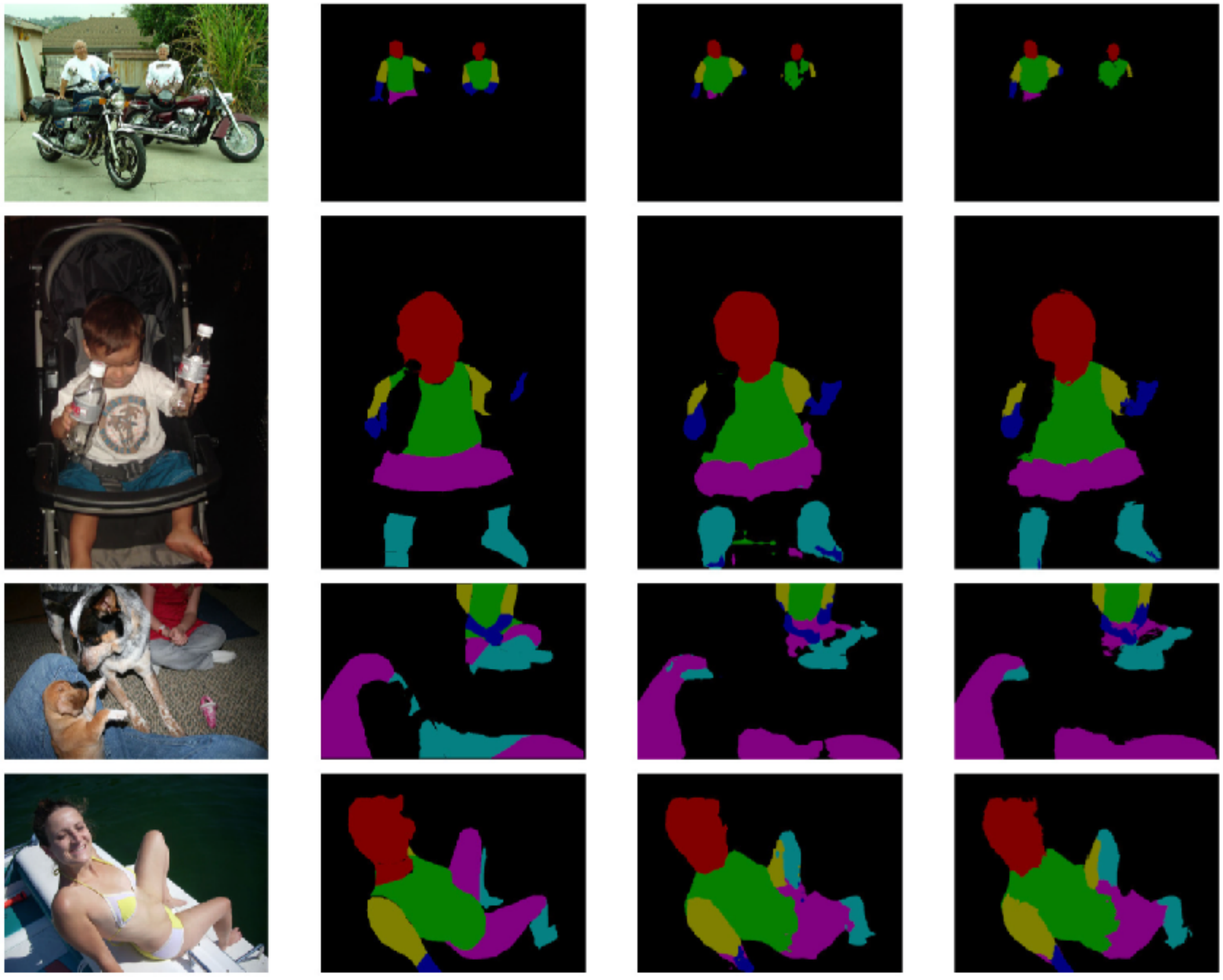
iterate (result shown after 10 iterations)

DeepLab v3 - results



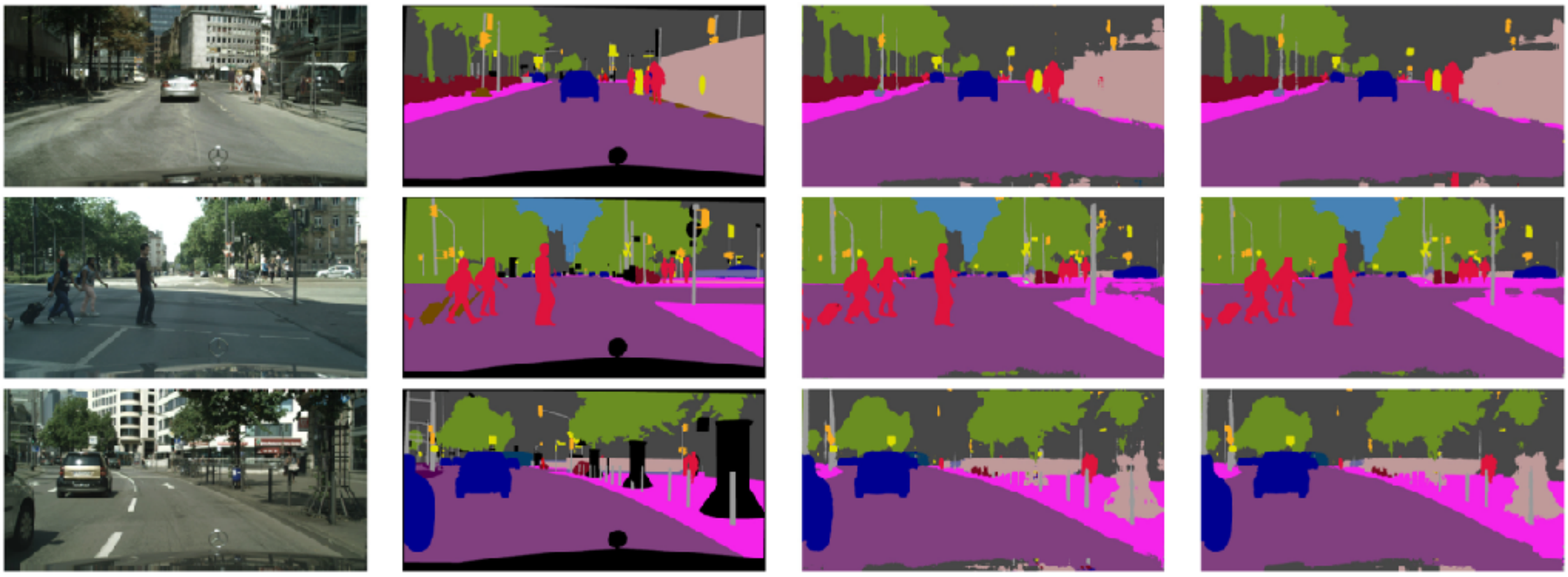
(a) Image (b) G.T. (c) Before CRF (d) After CRF

DeepLab v3 - results



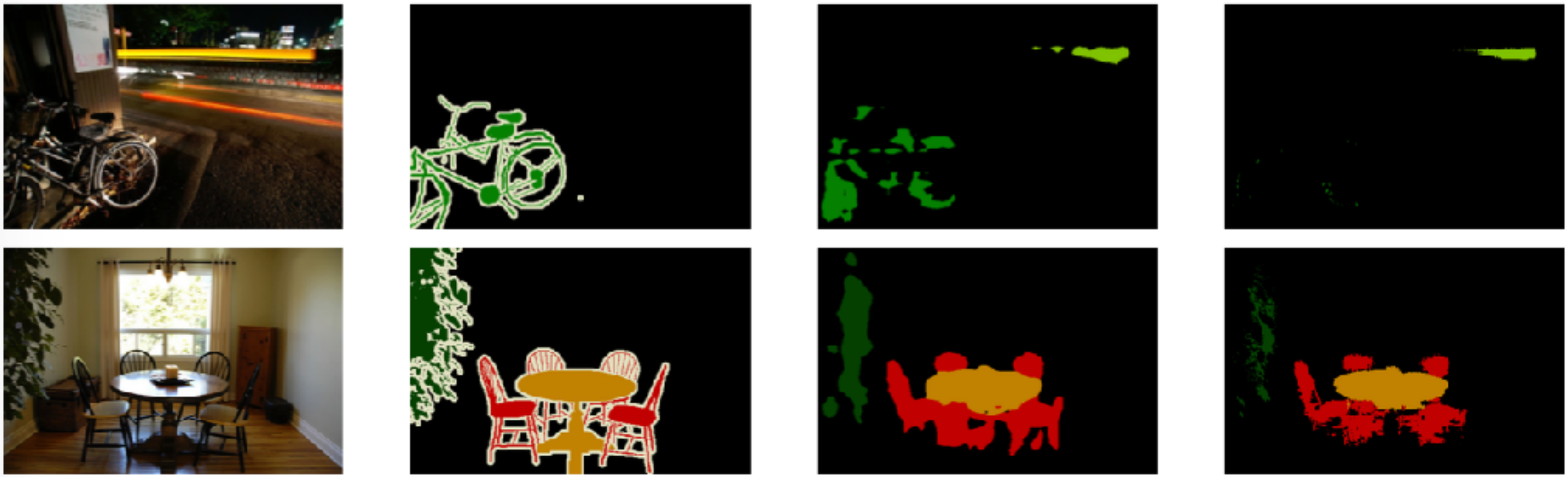
(a) Image (b) G.T. (c) Before CRF (d) After CRF

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

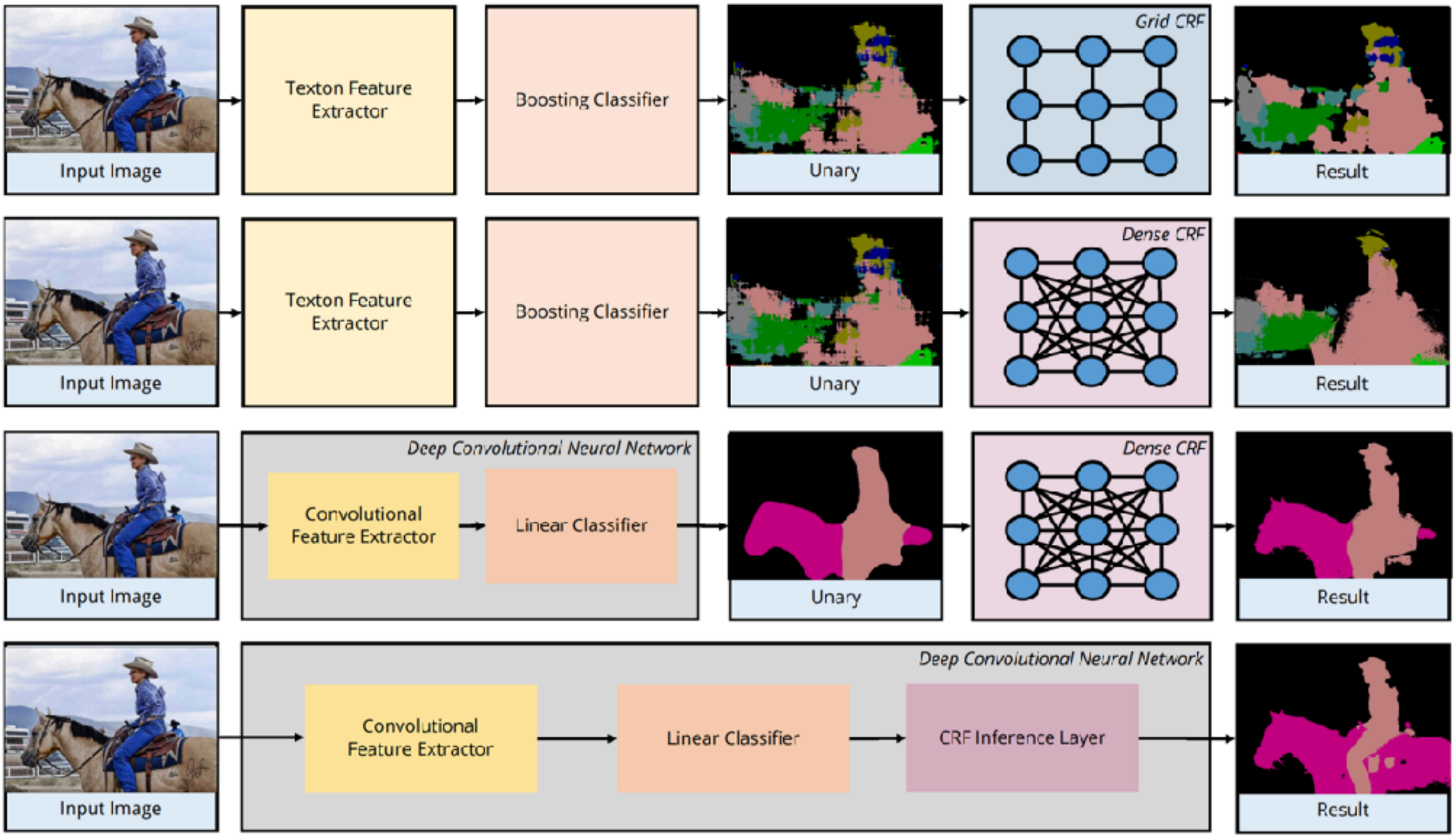
- significantly outperforms state-of-the-art on several datasets
- CRF improves mIOU about 2%
- ASPP improves mIOU about 3%
- codes available: https://github.com/tensorflow/models/tree/master/research/deeplab
- state-of-the-art benchmarks: http://www.robustvision.net/leaderboard.php?benchmark=semantic

- PyTorch implementation of differentiable ConvCRF layer [Teichmann & Cipolla, BMVC, 2019]
  https://arxiv.org/pdf/1805.04777.pdf

```
# Running ConvCRF 10 times and average total time
pred = gausscrf.forward(unary=unary_var, img=img_var)
```

Segmentation summary, [Torr et al. SPM, 2018]

http://www.robots.ox.ac.uk/~tvg/publications/2017/CRFMeetCNN4SemanticSegmentation.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

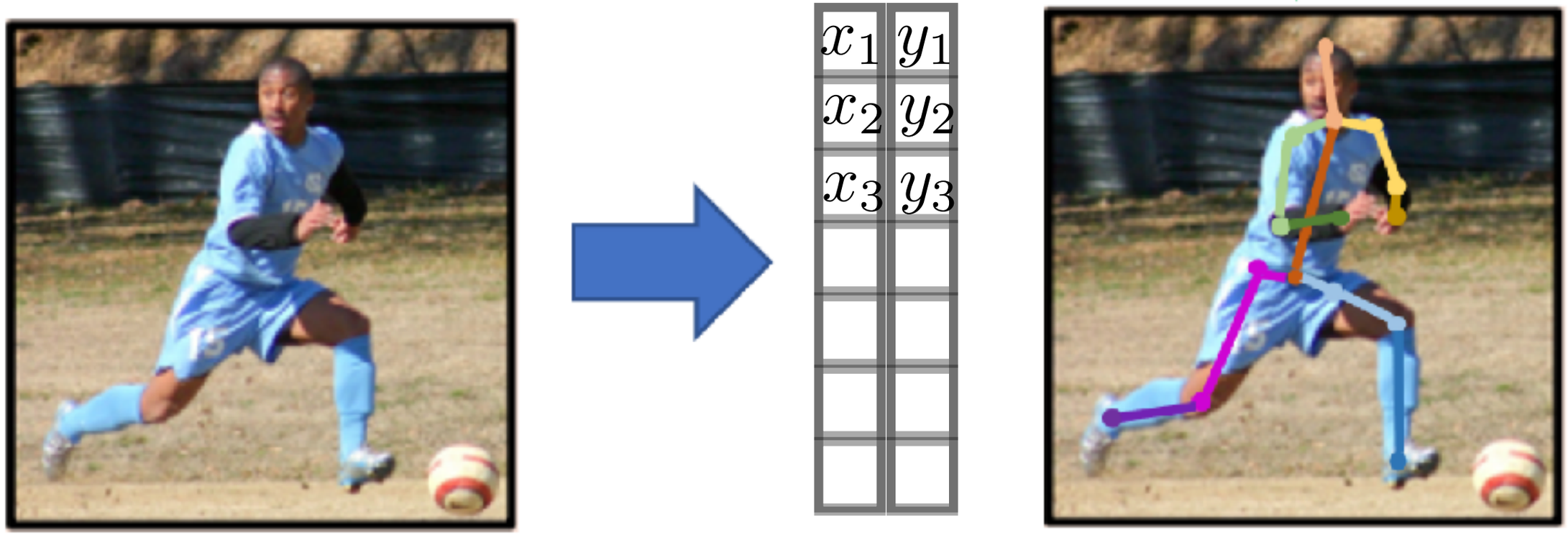
L2 loss

x_1	y_1
x_2	y_2
x_3	y_3

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

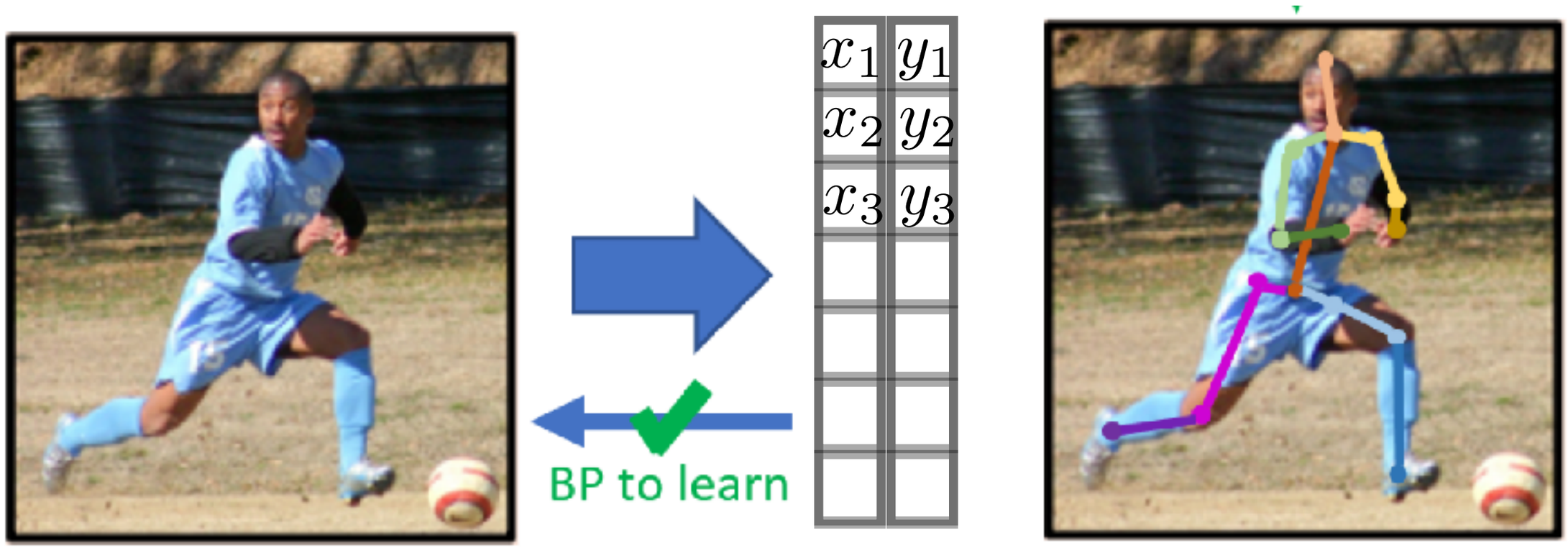
Pose regression baseline

L2 loss



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

Pose regression baseline



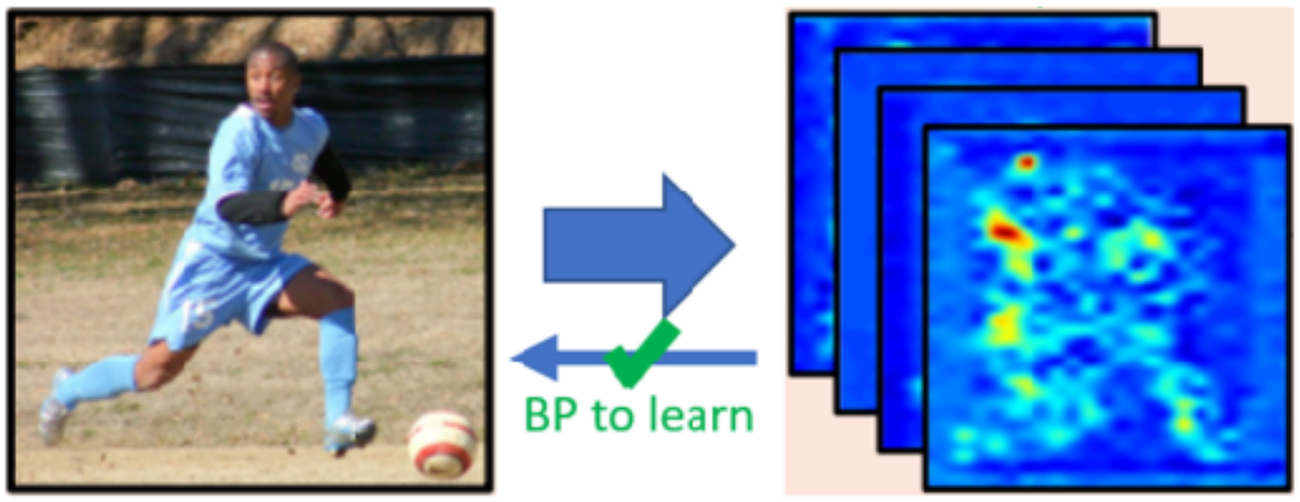
Joint regression:

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

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

Pose detection baseline

cross-entropy loss



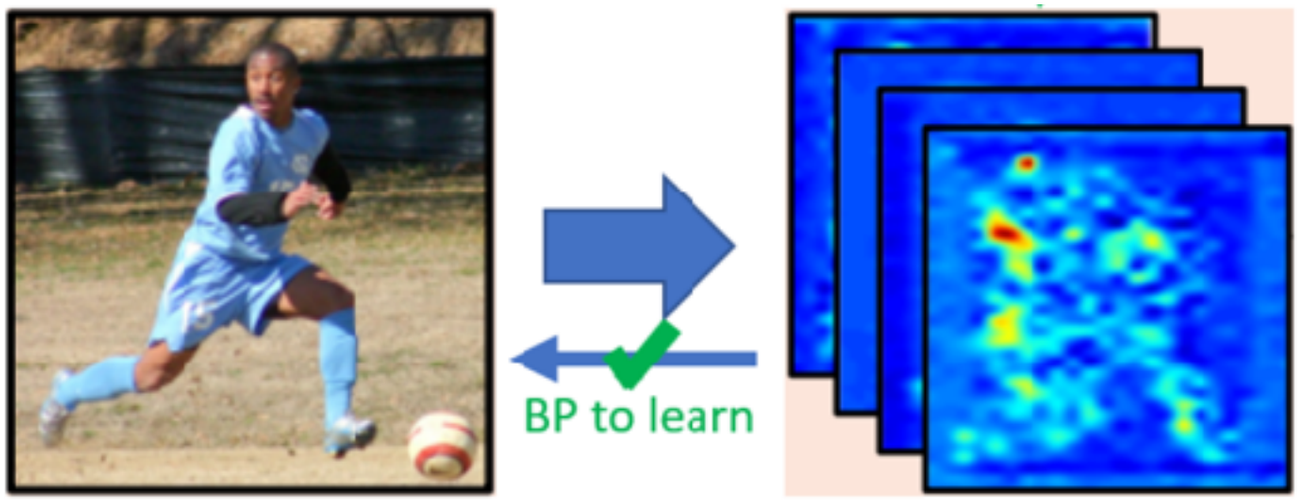
Joint segmentation:

- ConvNet first estimates N joint's heat maps $\mathbf{H}_k,\ k = 1 \ldots N$ (i.e. N 2D-images)
- 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

cross-entropy loss



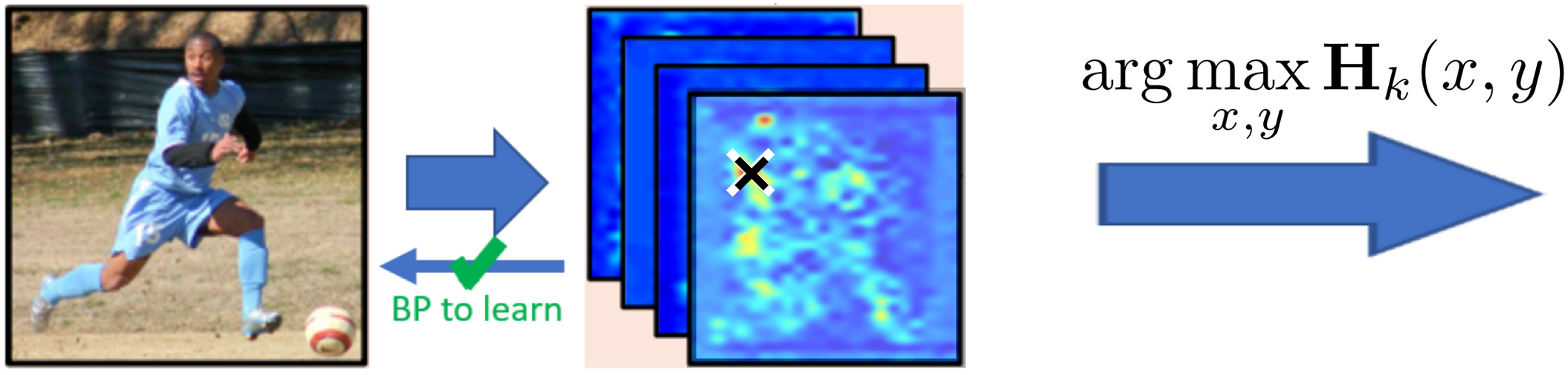
Joint segmentation:

- 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

cross-entropy loss



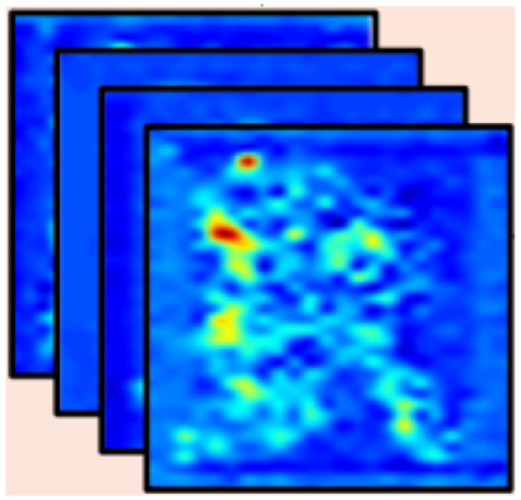
Joint segmentation:

- estimate joint position as position of heatmap maximum

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

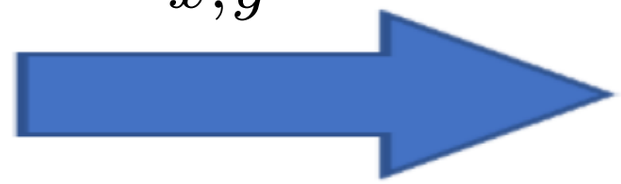
Pose detection baseline

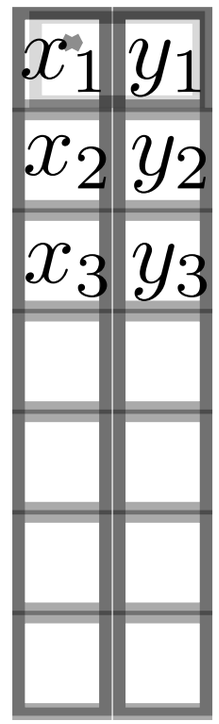
cross-entropy loss



$\mathbf{H}_k$: Heatmap

$$\arg\max_{x,y} \mathbf{H}_k(x,y)$$





Joint segmentation:

- estimate joint position as position of heatmap maximum

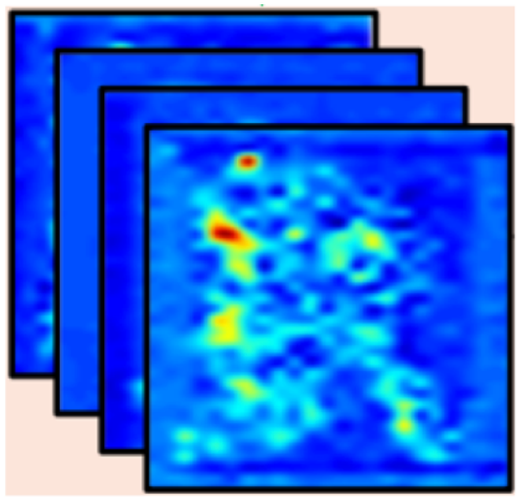
Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

Pose detection baseline

cross-entropy loss

L2 loss

BP to learn



$\mathbf{H}_k$: Heatmap

$$\arg\max_{x,y} \mathbf{H}_k(x,y)$$

Not differentiable !

x_1	y_1
x_2	y_2
x_3	y_3

Joint regression + segmentation:

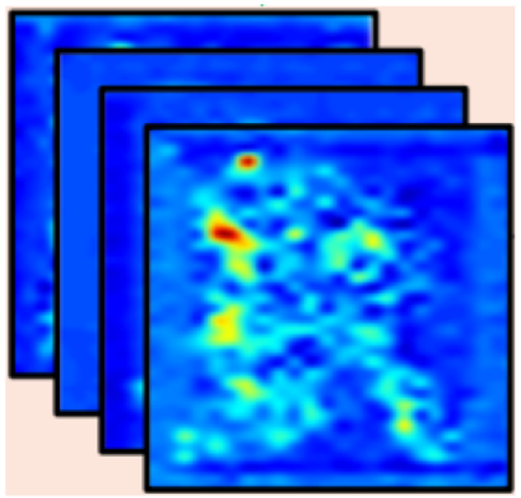
- estimate joint position as position of heatmap maximum

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

Pose detection baseline

BP to learn

cross-entropy loss

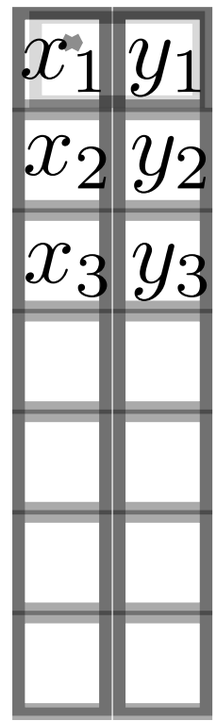


$\mathbf{H}_k$: Heatmap

$$\sum_x \sum_y \mathbf{H}_k(x,y) \cdot [x,y]$$

Differentiable !

L2 loss



Joint regression + segmentation:

- estimate joint position as expected value in heatmap
- learning = minimize cross-entropy + L2 loss

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

heatmap-weighted center-of-gravity layer

grid – fixed tensors

heatmap

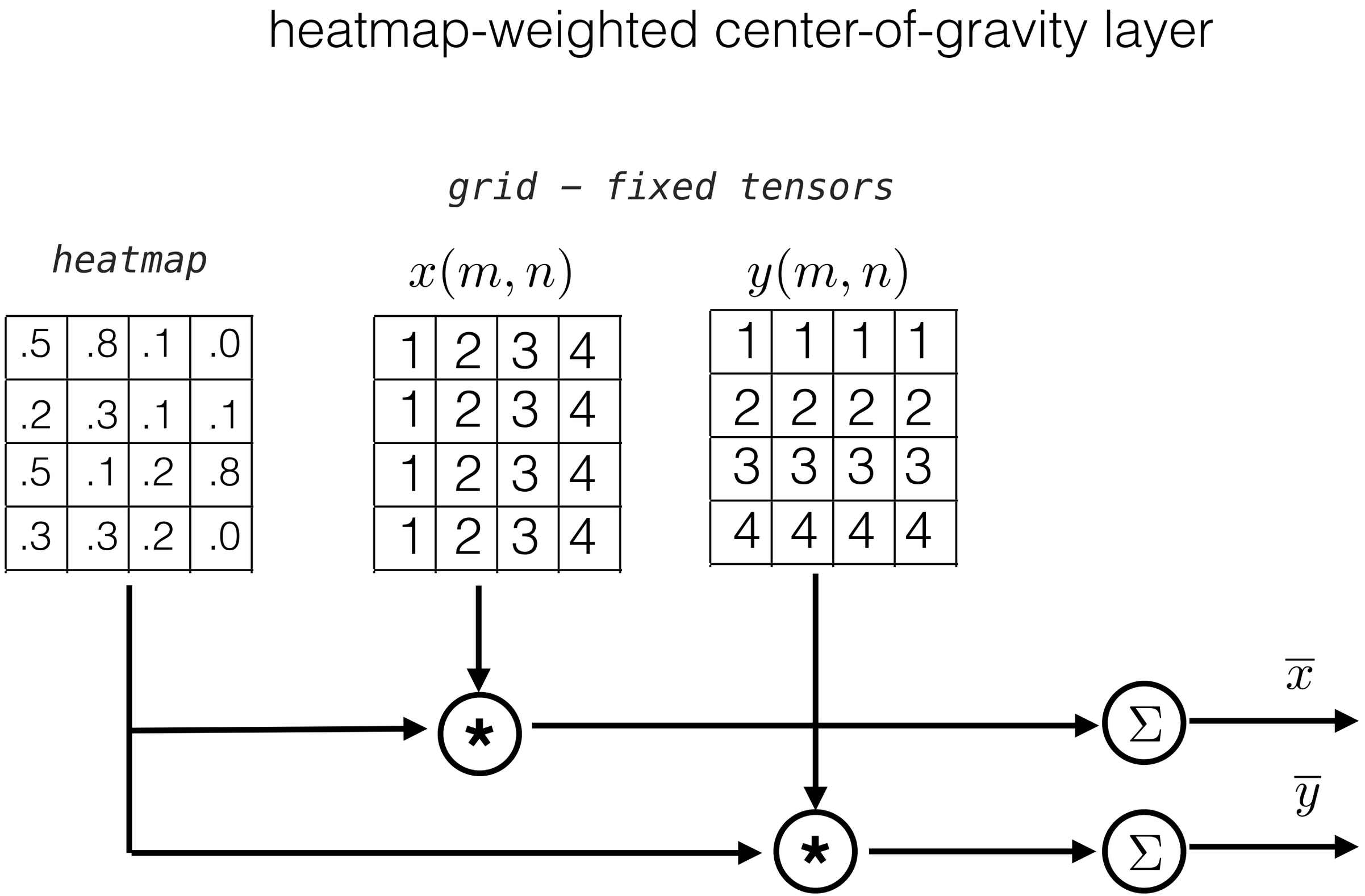
.5	.8	.1	.0
.2	.3	.1	.1
.5	.1	.2	.8
.3	.3	.2	.0

$x(m,n)$

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

$y(m,n)$

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



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

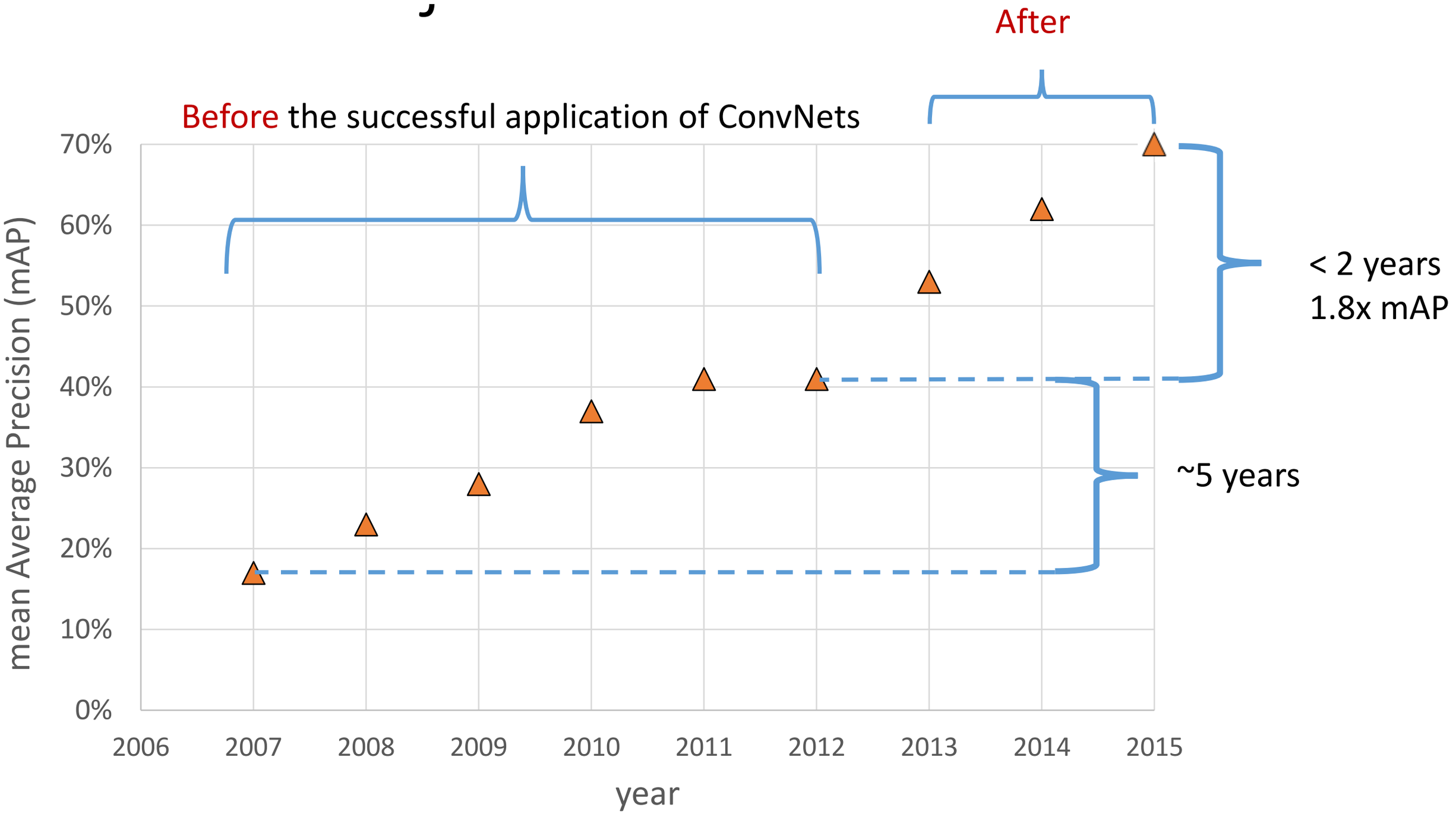
Pose regression references

- 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

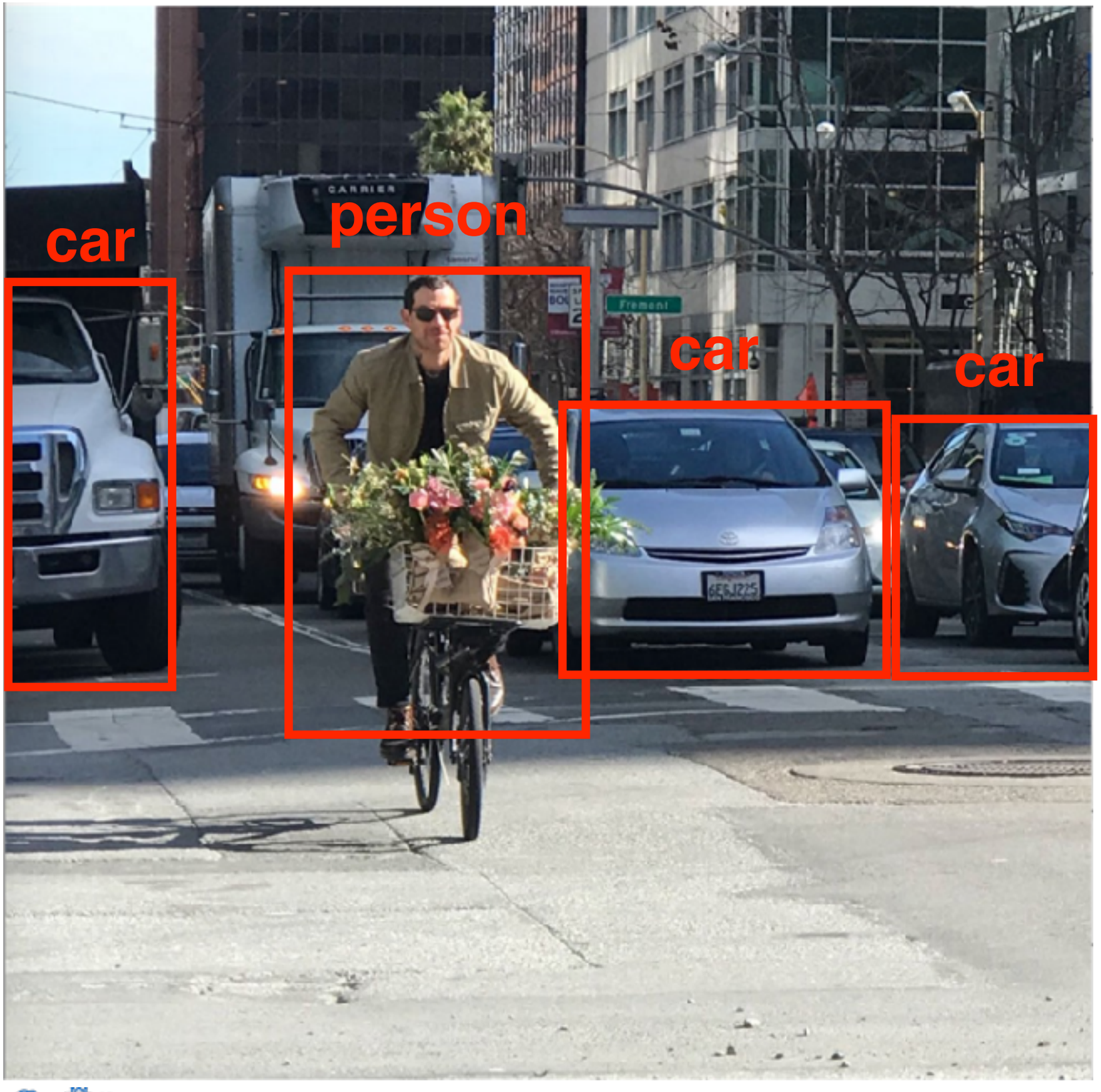
- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Architectures of feature matching networks

Pascal VOC object detection challenge



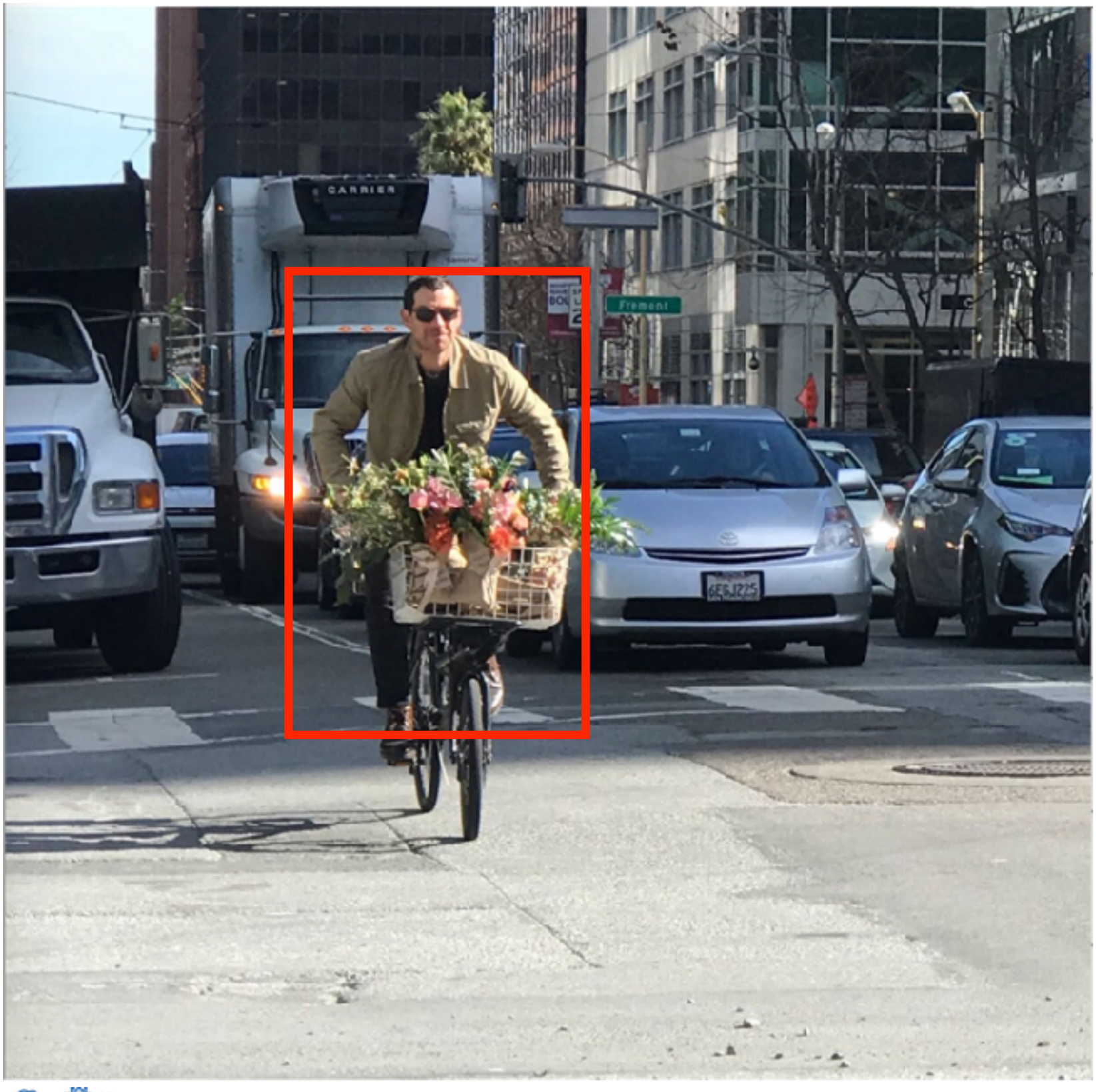
Object detection

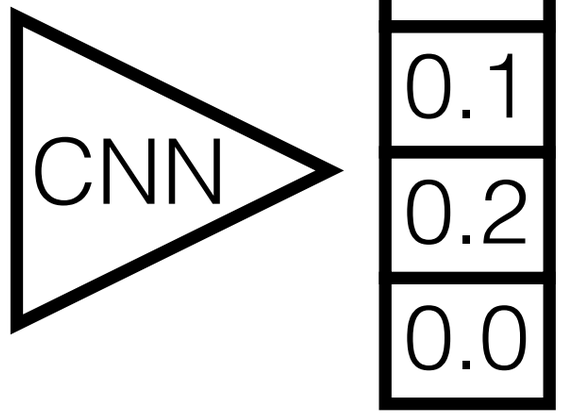
Object detection



Object detection

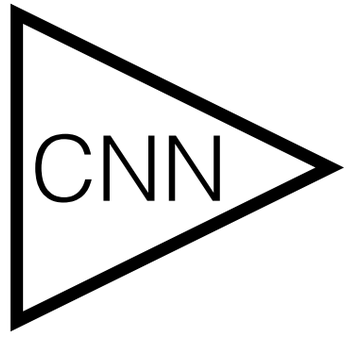
Object detection

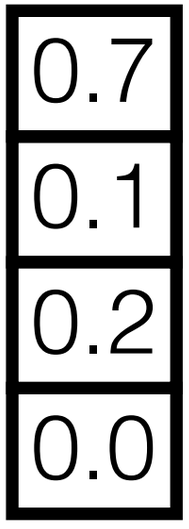




class: person

Object detection





Object detection

CNN

0.0
0.9
0.1
0.0

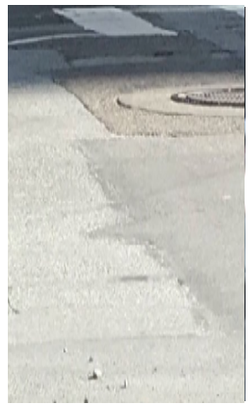
class: car

Object detection

CNN

0.0
0.9
0.1
0.0

Object detection

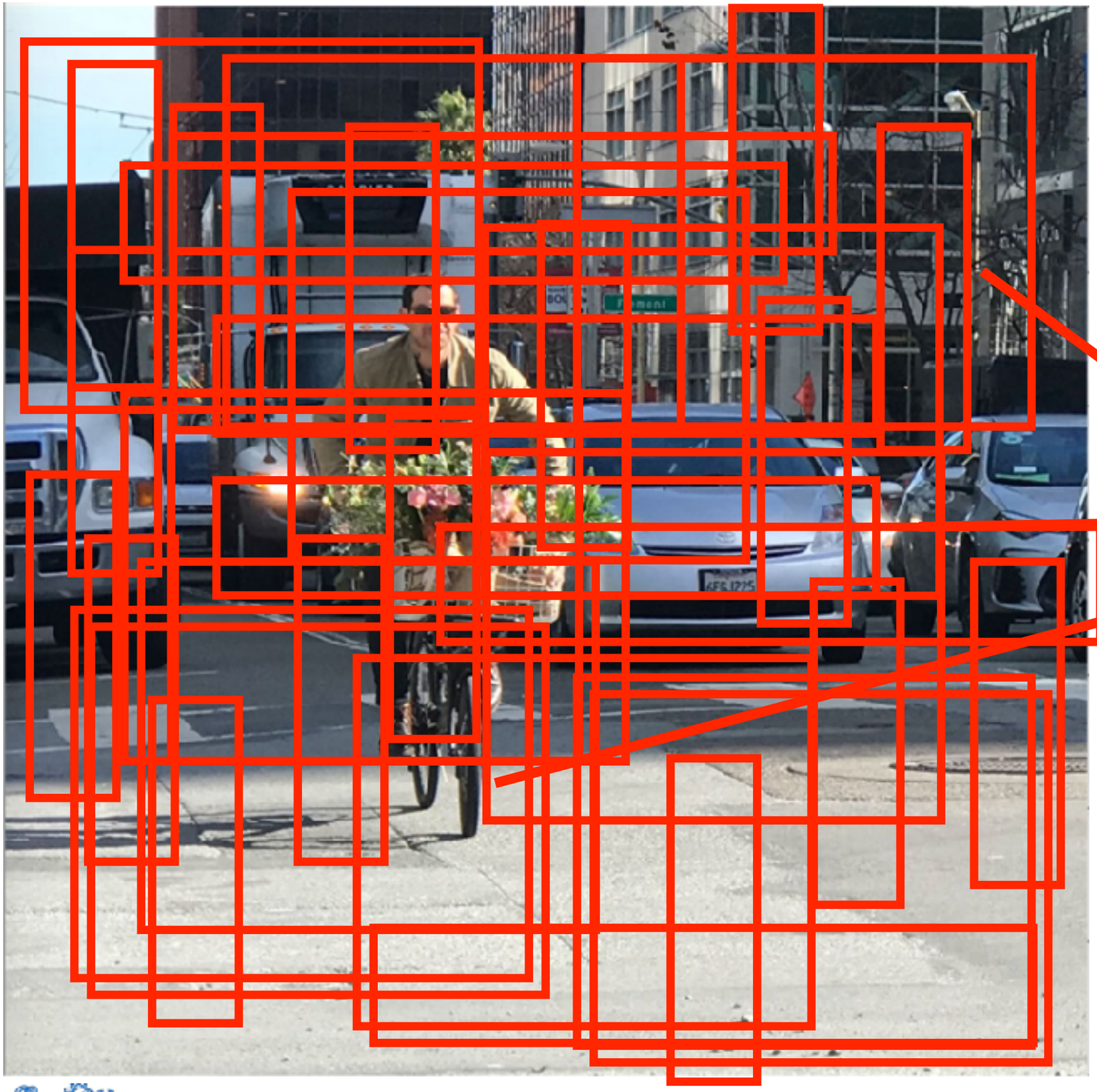


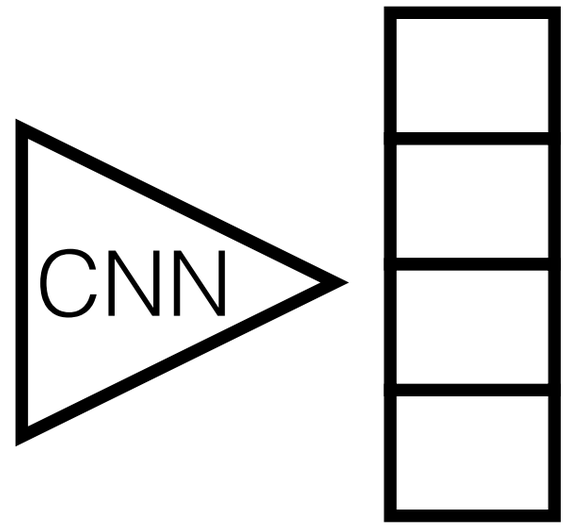
CNN

0.0
0.1
0.0
0.9

class: background

Object detection



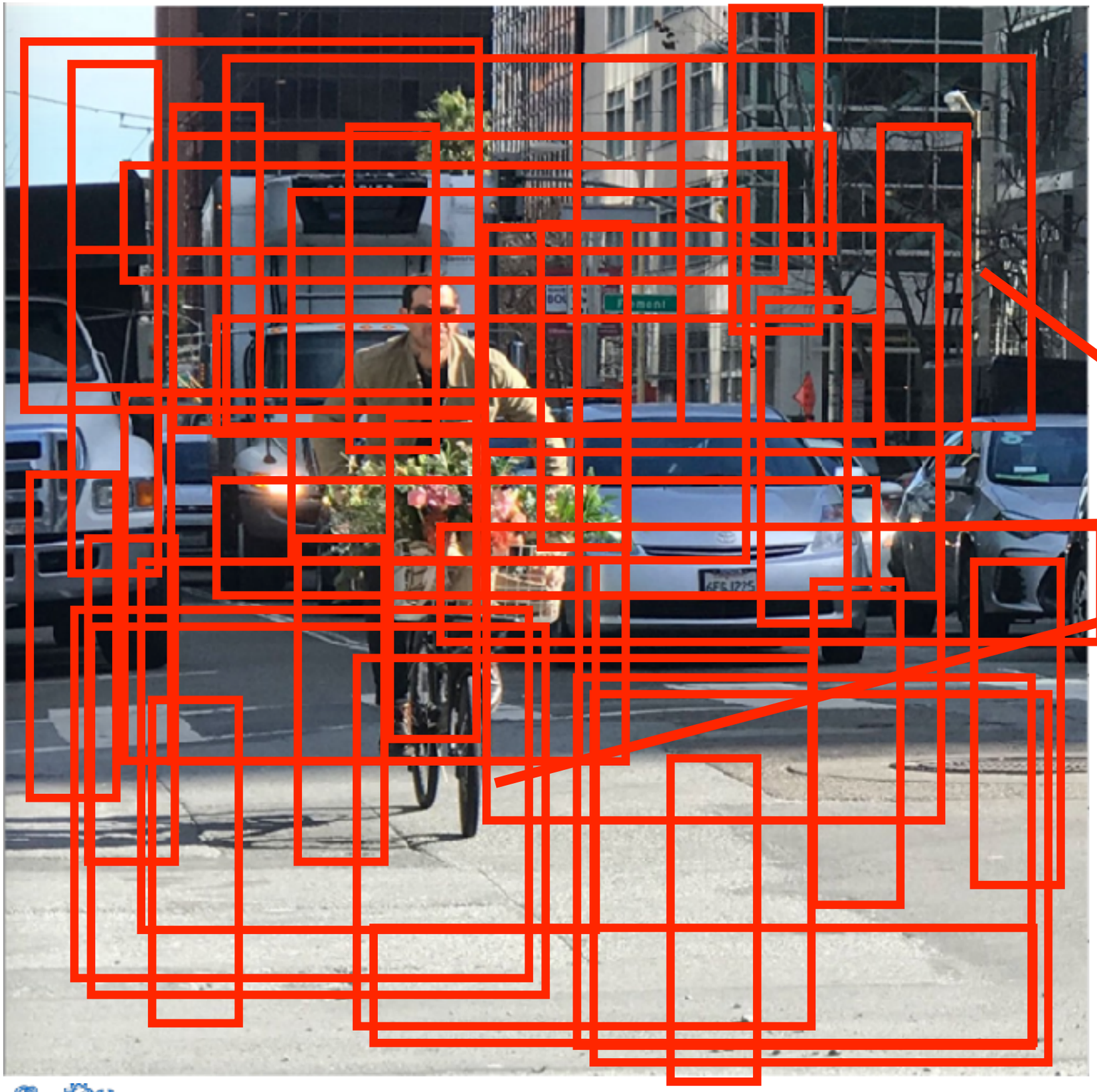


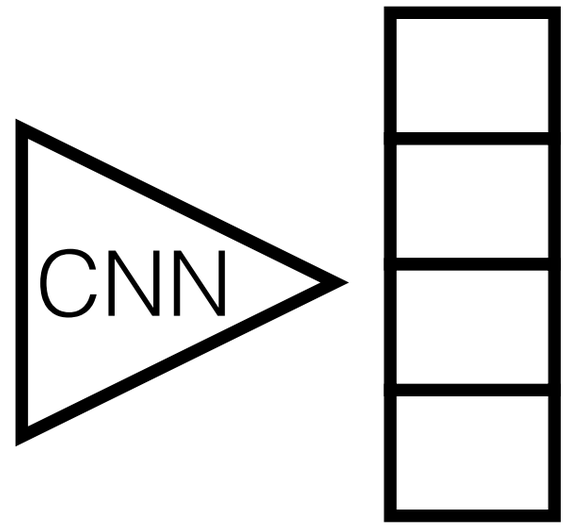
classify all rectangles

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**

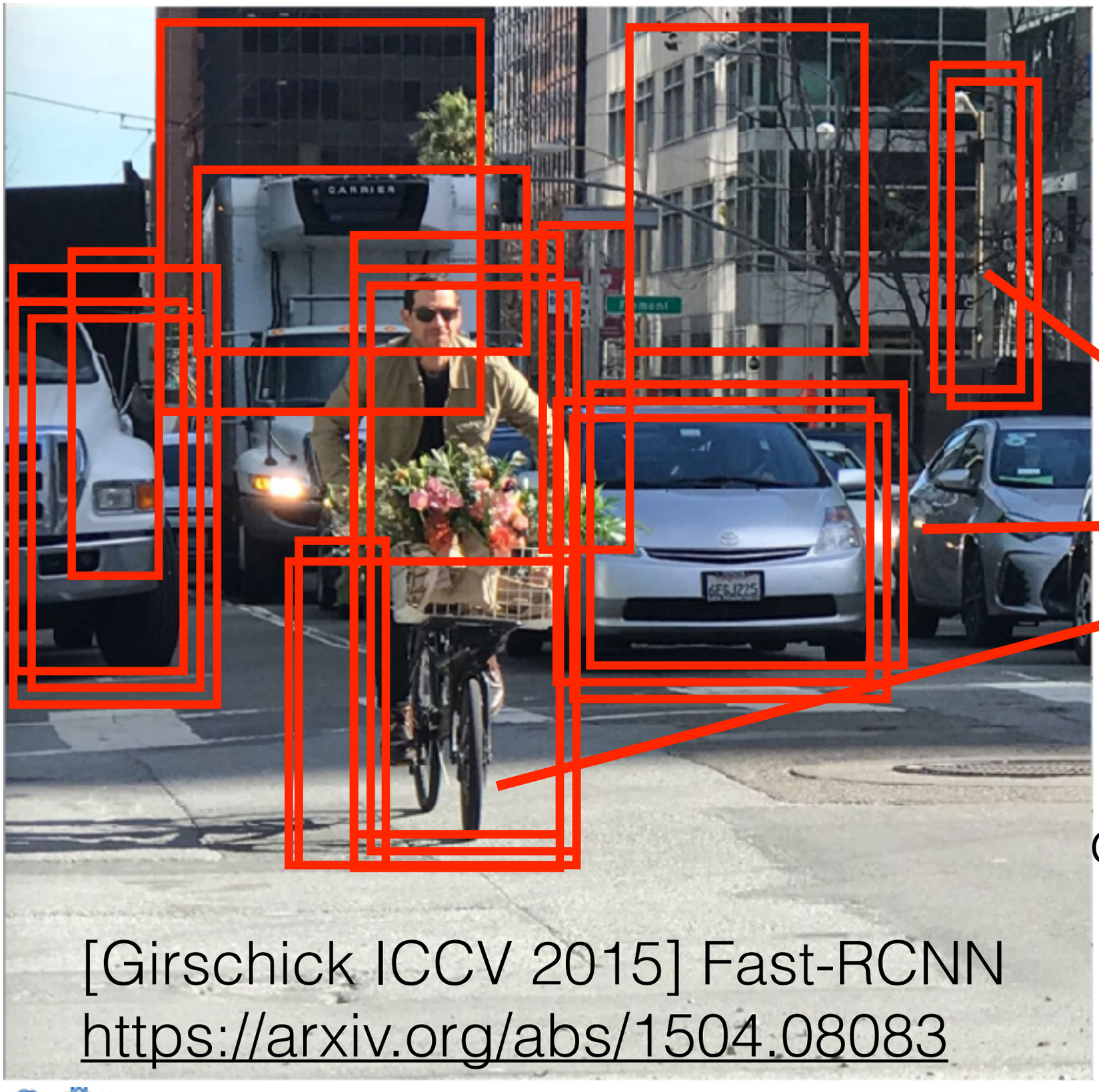
Object detection





classify all rectangles

Object detection



CNN

classify + align only 2k region proposals

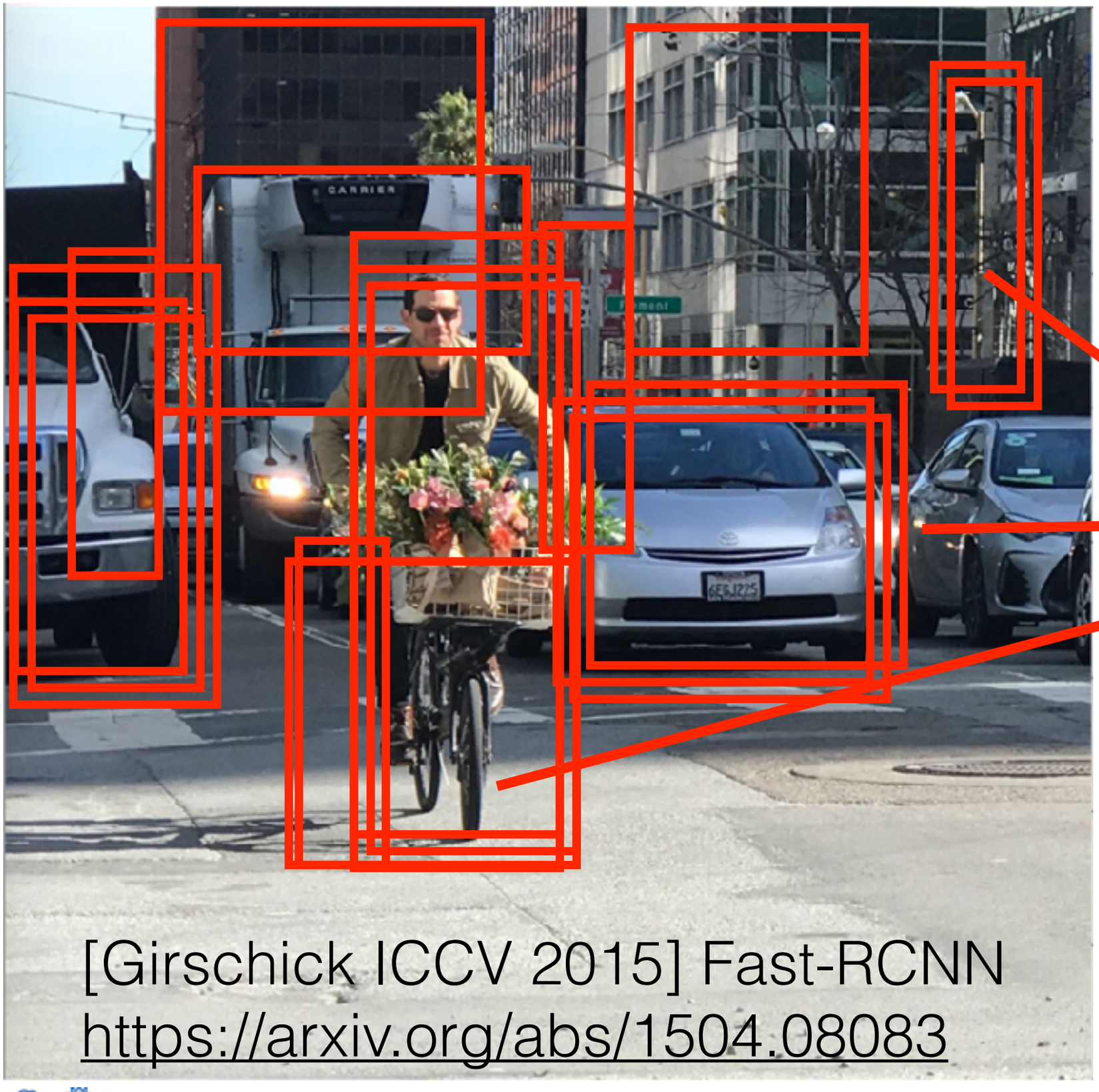
[Girschick ICCV 2015] Fast-RCNN
https://arxiv.org/abs/1504.08083

Object detection

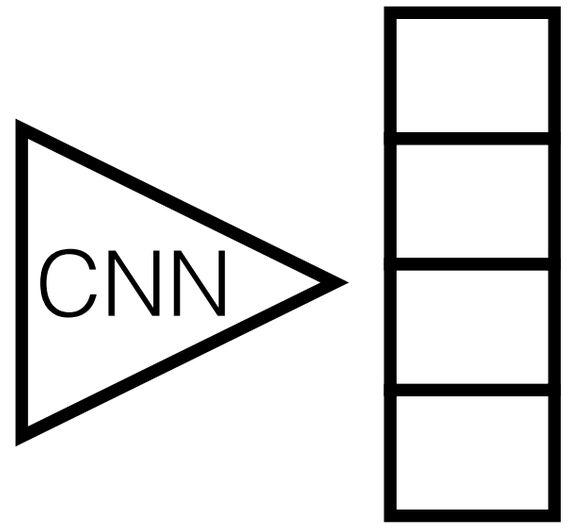
- Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  H x W x Aspect_Ratio x Scales x 0.001 sec = **months**
- Instead we can use elementary signal processing method to extract only 2k viable candidates:
  [Girschick ICCV 2015], Fast-RCNN
  https://arxiv.org/abs/1504.08083

(find 2k cand.) + (2k cand. x 0.001 sec) = **47+2 sec = 49 sec**

Object detection



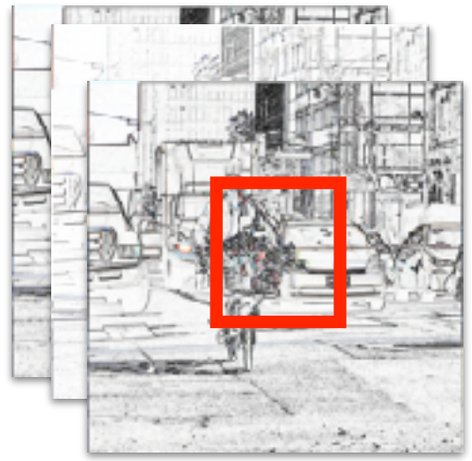
[Girschick ICCV 2015] Fast-RCNN
https://arxiv.org/abs/1504.08083



The search for region proposals is computational bottleneck !!!

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497

RPN



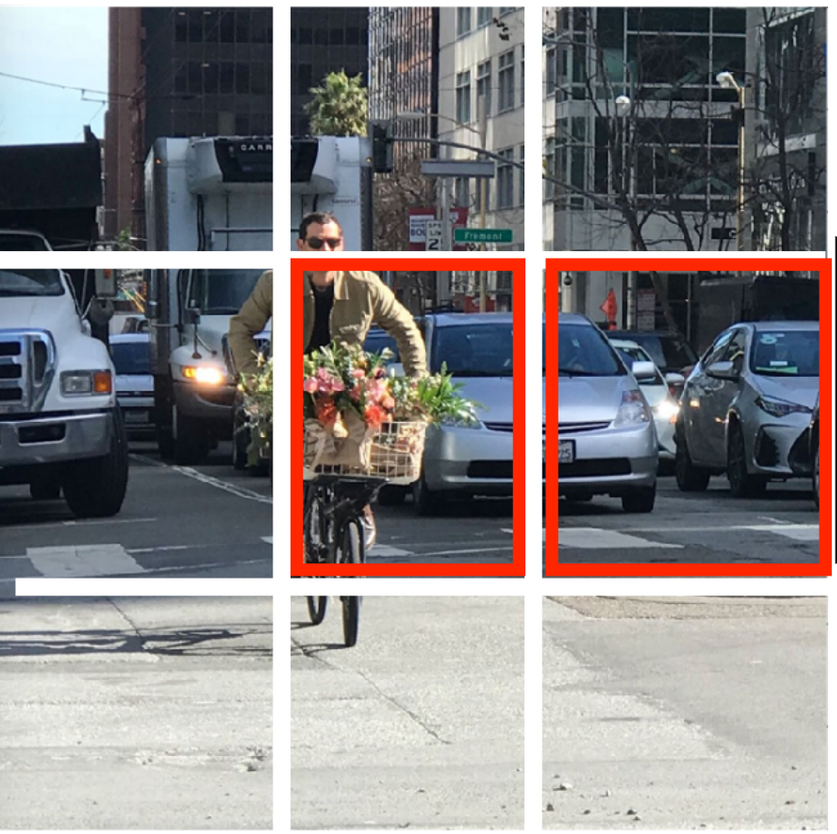
low resolution
feature map

CNN

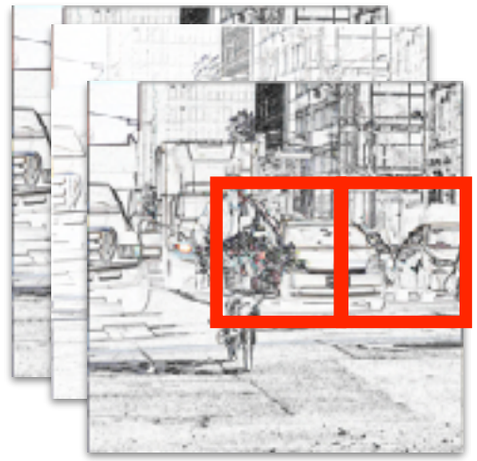
p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



RPN



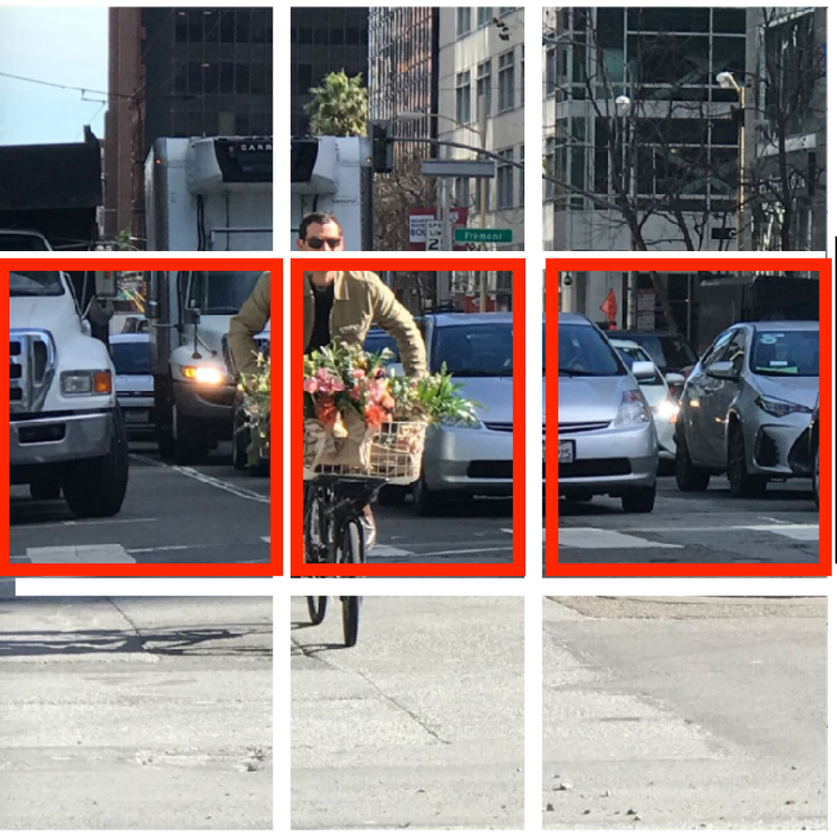
low resolution
feature map

CNN

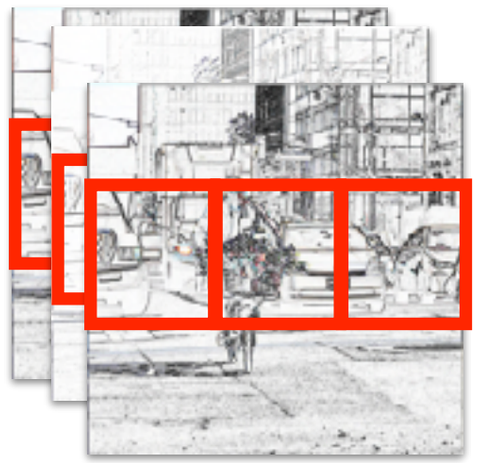
p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



RPN



low resolution
feature map

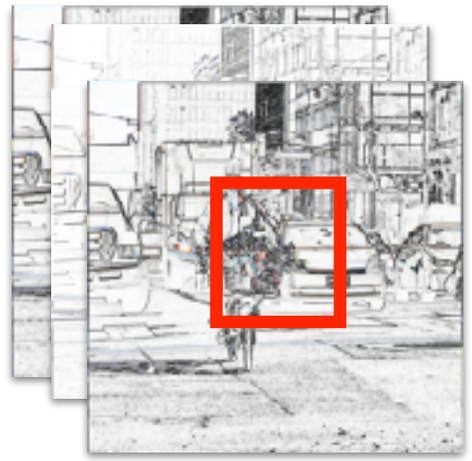
CNN

p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497

RPN



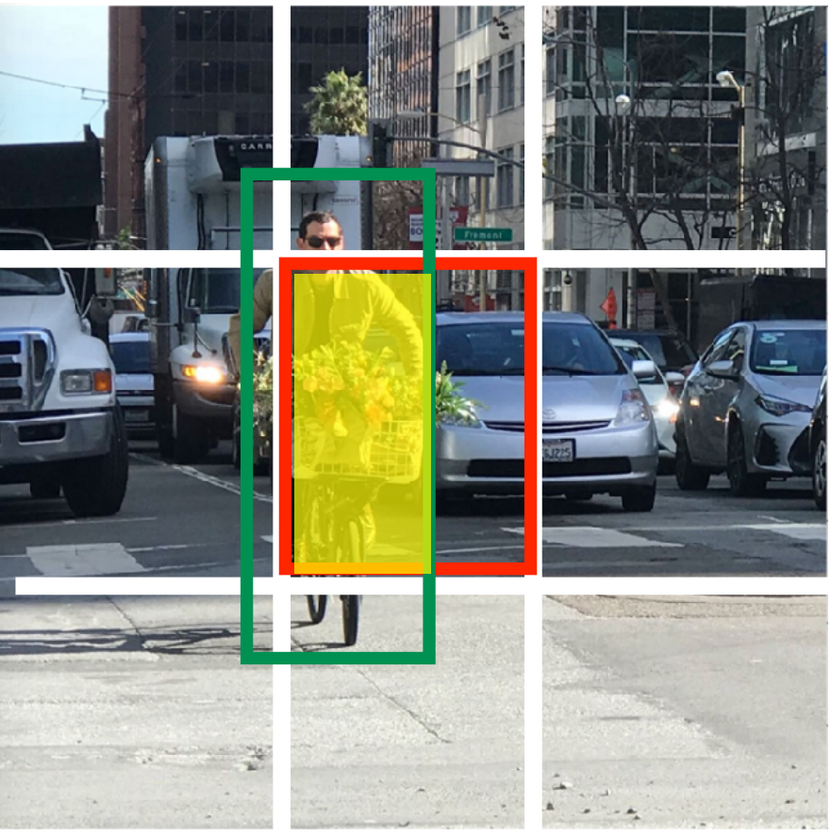
low resolution
feature map

CNN

p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

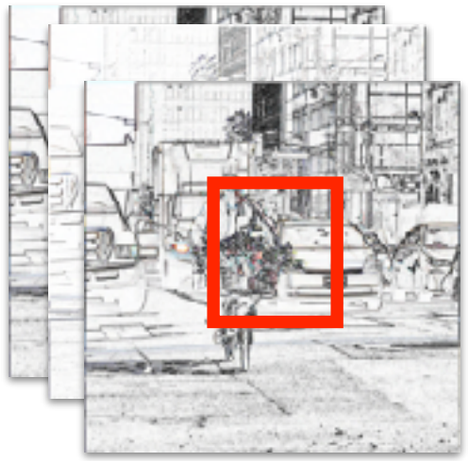
- divide image into 3x3 sub-images
- predict relative position, objectness, class for each sub-im

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



ground truth

RPN



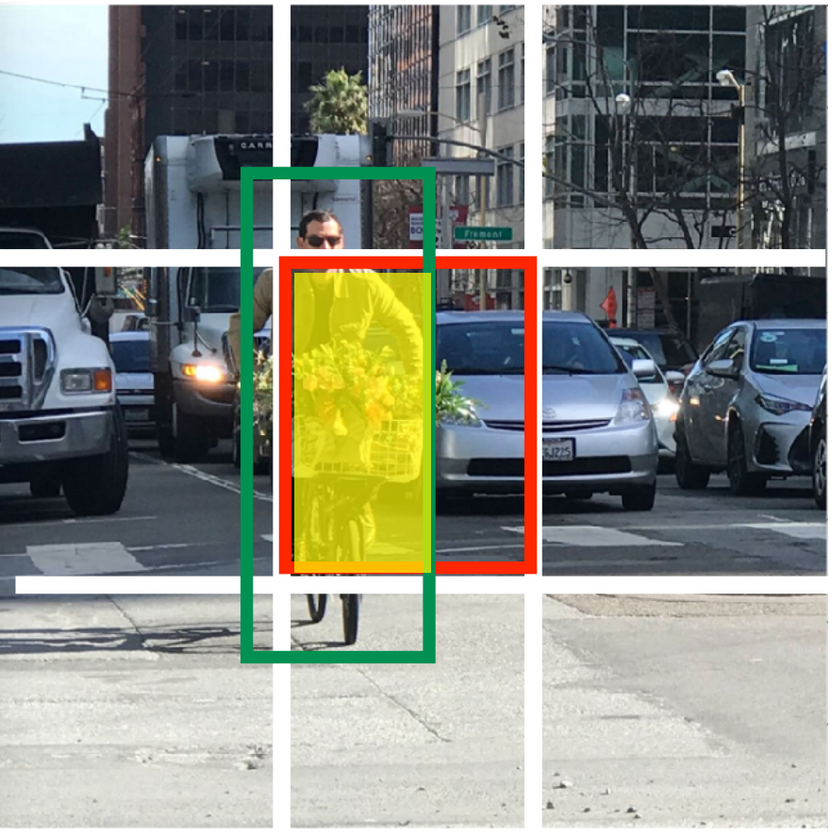
low resolution
feature map

CNN

p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

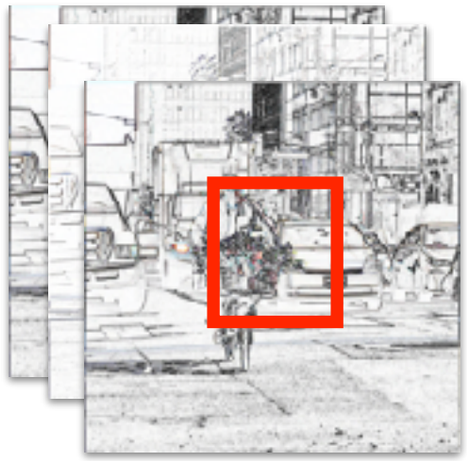
- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- learn from ground truth

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



ground truth

RPN



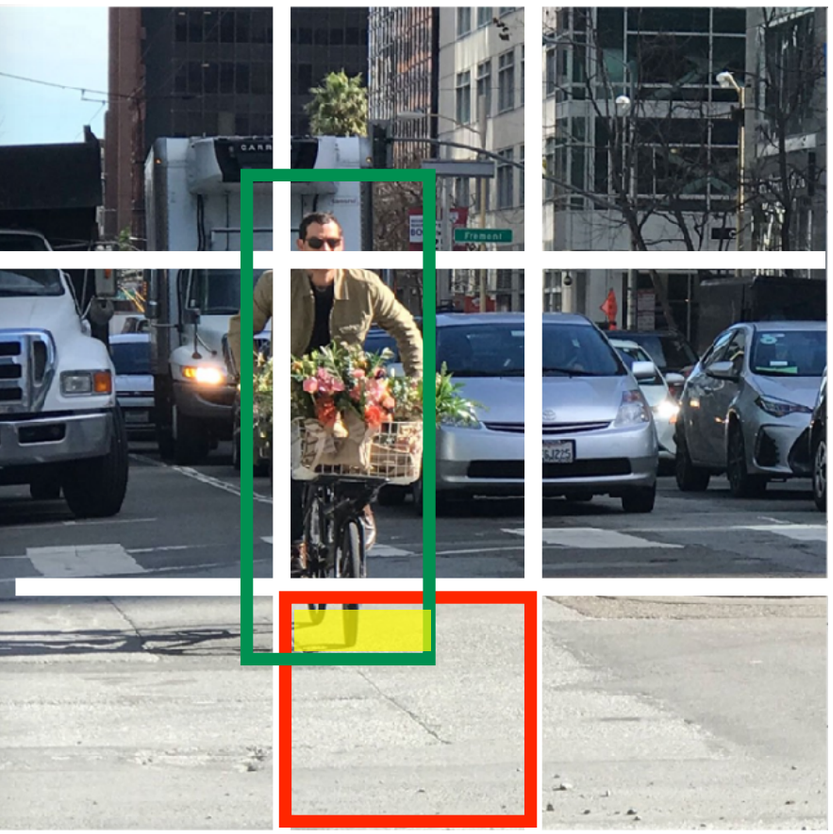
low resolution
feature map

CNN

p_object =1
center_x=.3
center_y=.5
width =.7
height =1.5
p_class1=1
p_class2=0
p_class3=0

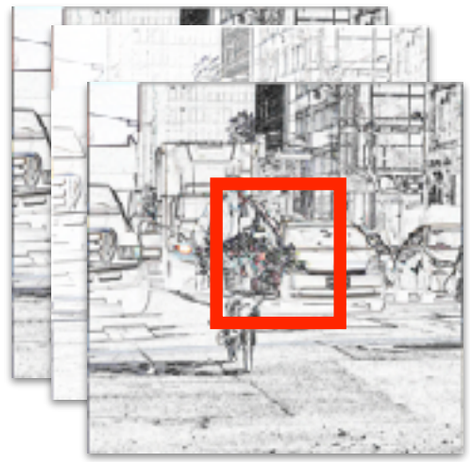
- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- ground truth: bbs with IoU>0.7 are objects,
  bbs with IoU<0.3 not objects

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



ground truth

RPN



low resolution
feature map

CNN

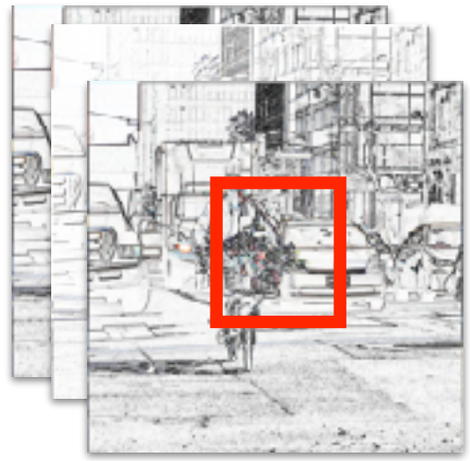
p_object =0
center_x=?
center_y=?
width =?
height =?
p_class1=?
p_class2=?
p_class3=?

- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- ground truth: bbs with IoU>0.7 are objects,
  bbs with IoU<0.3 not objects

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497

ground truth

RPN



low resolution
feature map

CNN

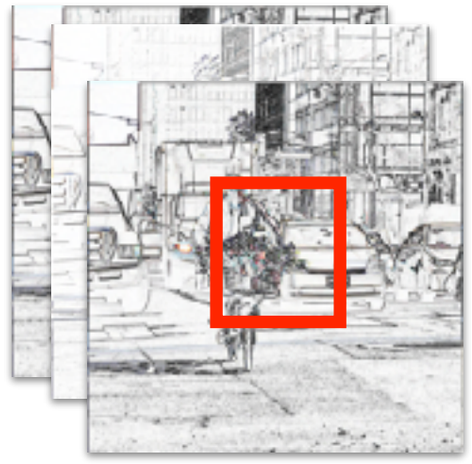
p_object =1
center_x=.3
center_y=.5
width =.7
height =1.5
p_class1=**1**
p_class2=**1**
p_class3=0

- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- ground truth: bbs with IoU>0.7 are objects, => more obj in one sub-im
  bbs with IoU<0.3 not objects

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497

ground truth

RPN



low resolution
feature map

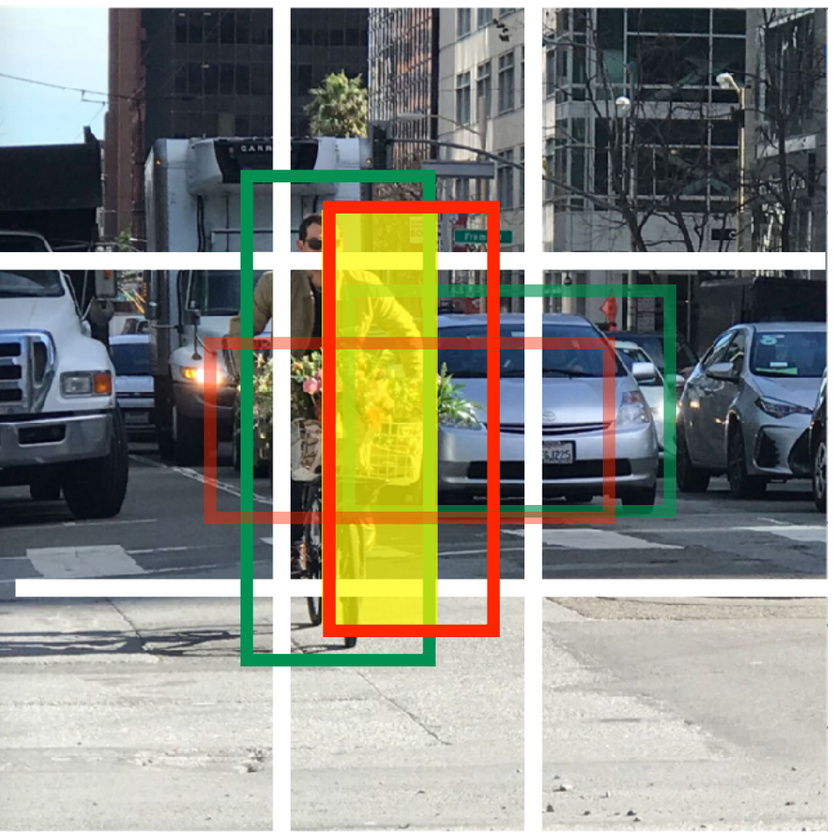
CNN

p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

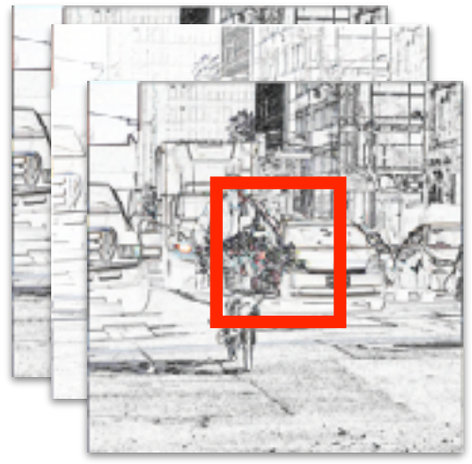
Introduce anchor bounding boxes

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



ground truth

RPN



low resolution
feature map

CNN

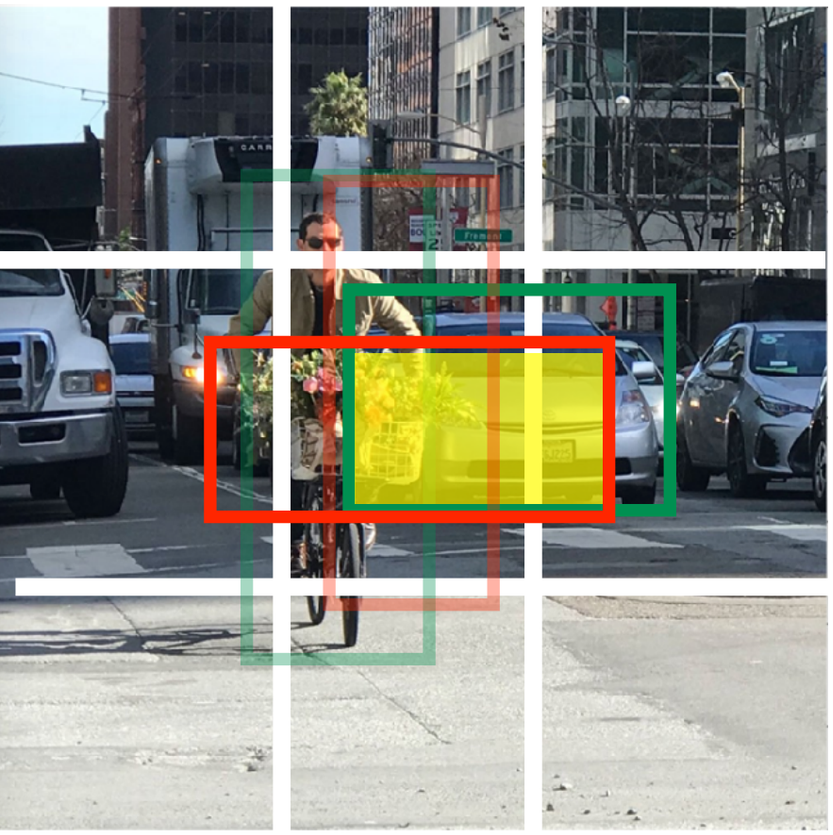
p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

Introduce anchor bounding boxes

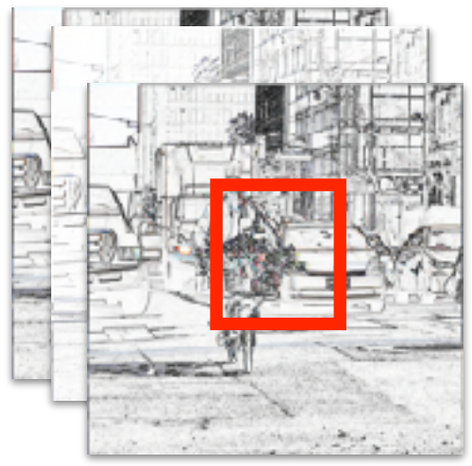
- for each anchor bb CNN predicts:
  - its “alignment with gt” (regression loss)
  - its “objectness”+”class” (classification loss)

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



ground truth

RPN



low resolution
feature map

CNN

p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

Introduce anchor bounding boxes

- for each anchor bb CNN predicts:
  - its “alignment with gt” (regression loss)
  - its “objectness”+”class” (classification loss)

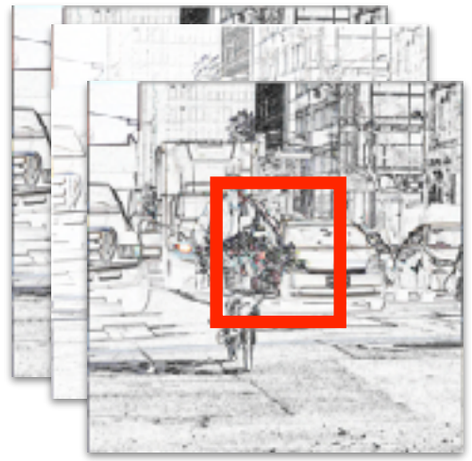
Object detection

- 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 => **0.1 sec**
  [Faster RCNN 2017] https://arxiv.org/abs/1506.01497
  [Redmont CVPR 2018], https://arxiv.org/abs/1804.02767
  code: https://pjreddie.com/darknet/yolo/

Deep convolutional - object detection

YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497

RPN



low resolution
feature map

CNN

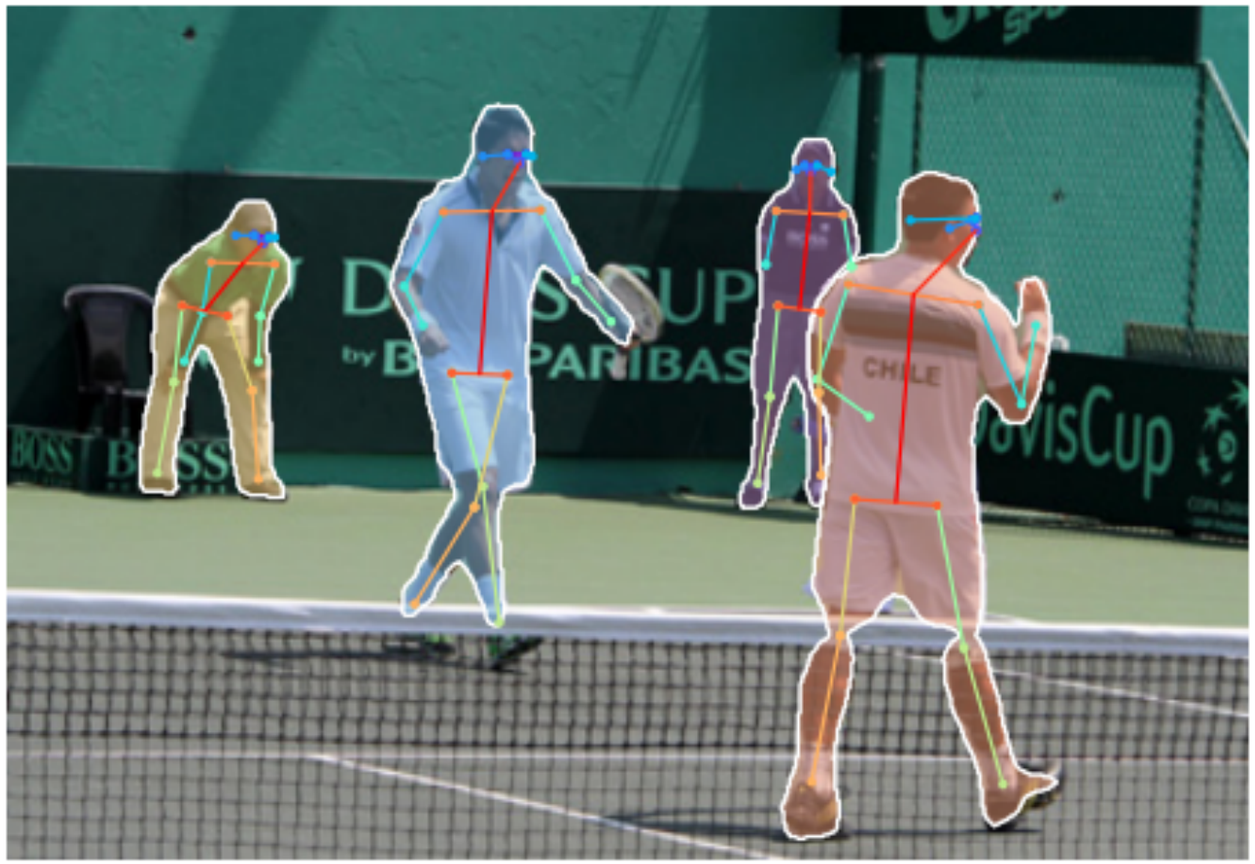
p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3
segment.
pose

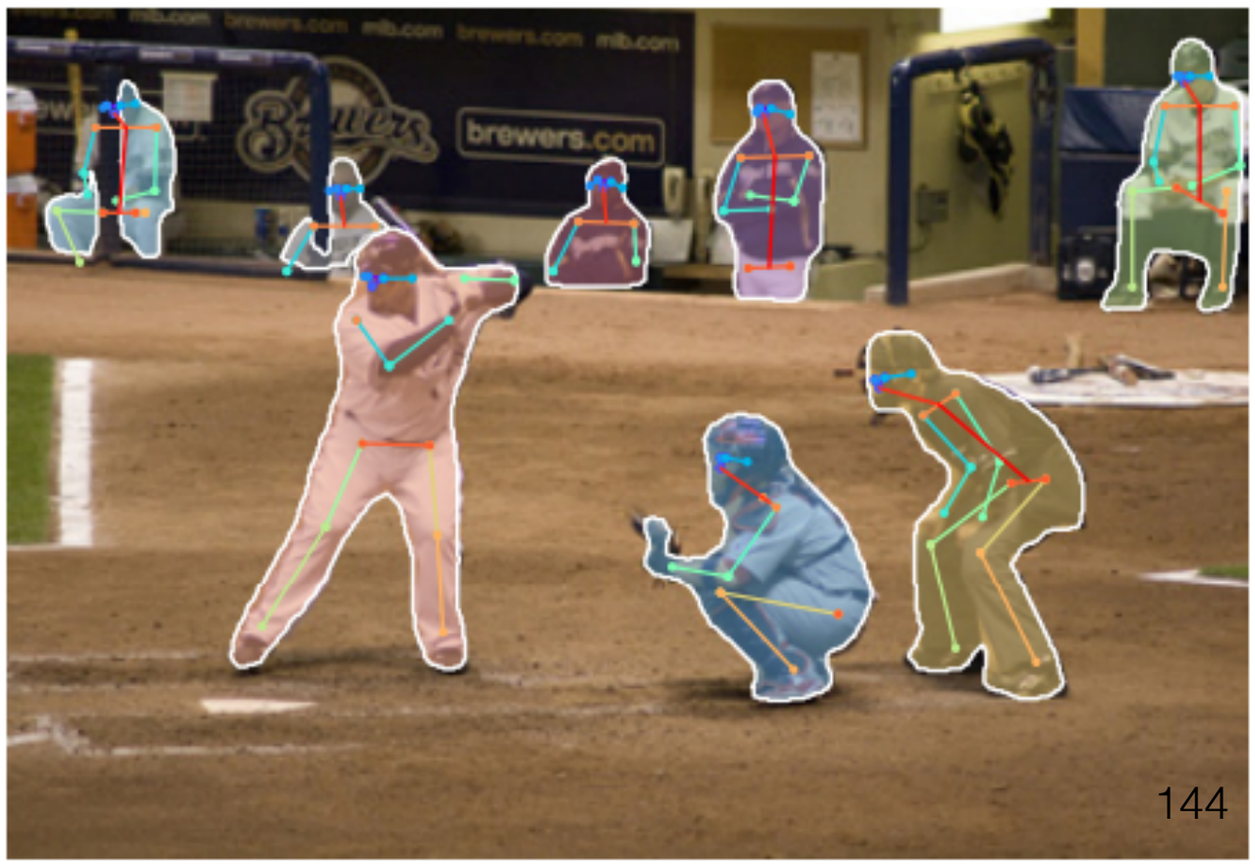
[He et al CVPR 2017] Mask-RCNN
https://arxiv.org/abs/1703.06870

Mask RCNN - results

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

Mask RCNN - results





Conclusion and links

- COCO Leaderboard:

https://paperswithcode.com/sota/object-detection-on-coco

- 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

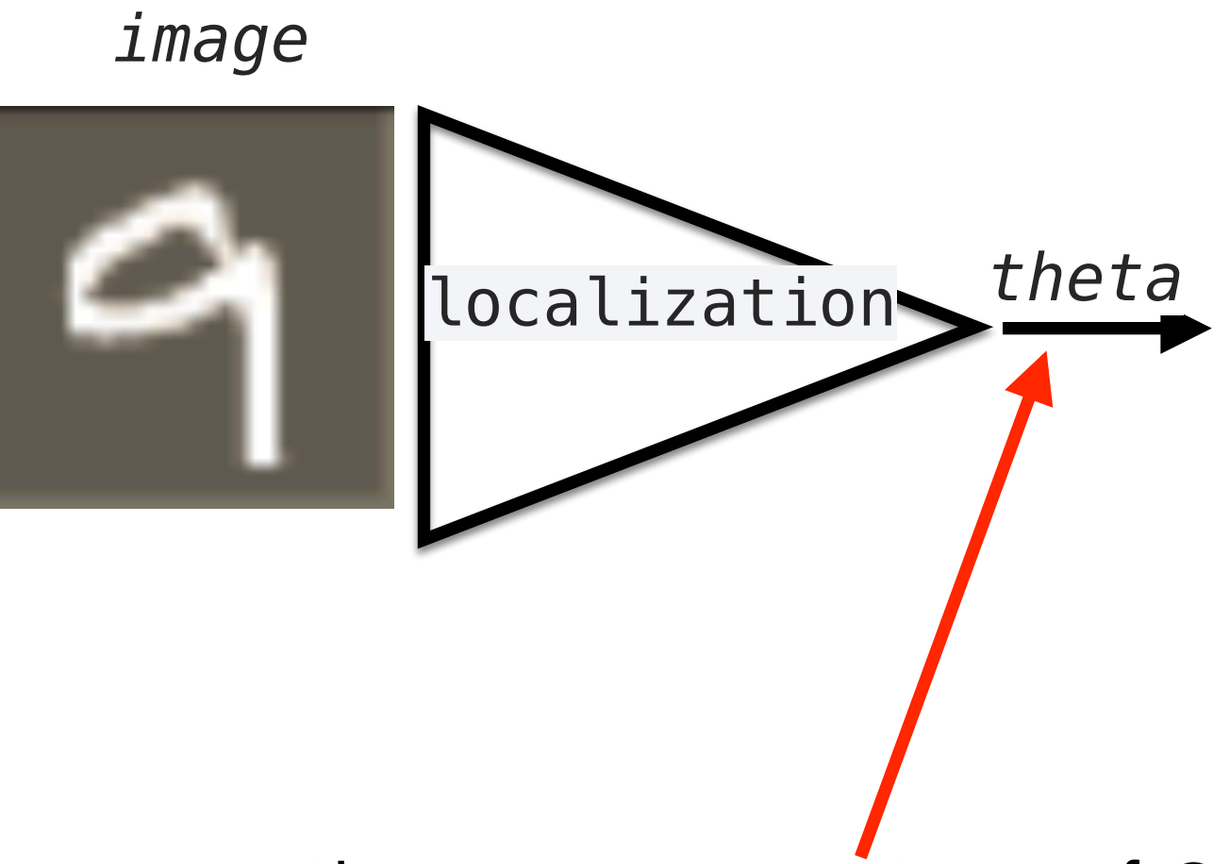
https://everitt257.github.io/post/2018/08/10/object_detection.html#:~:text=YOLO%20stands%20for%20You%20Only,regression%20at%20the%20same%20time.

Efficient scaling: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

Outline

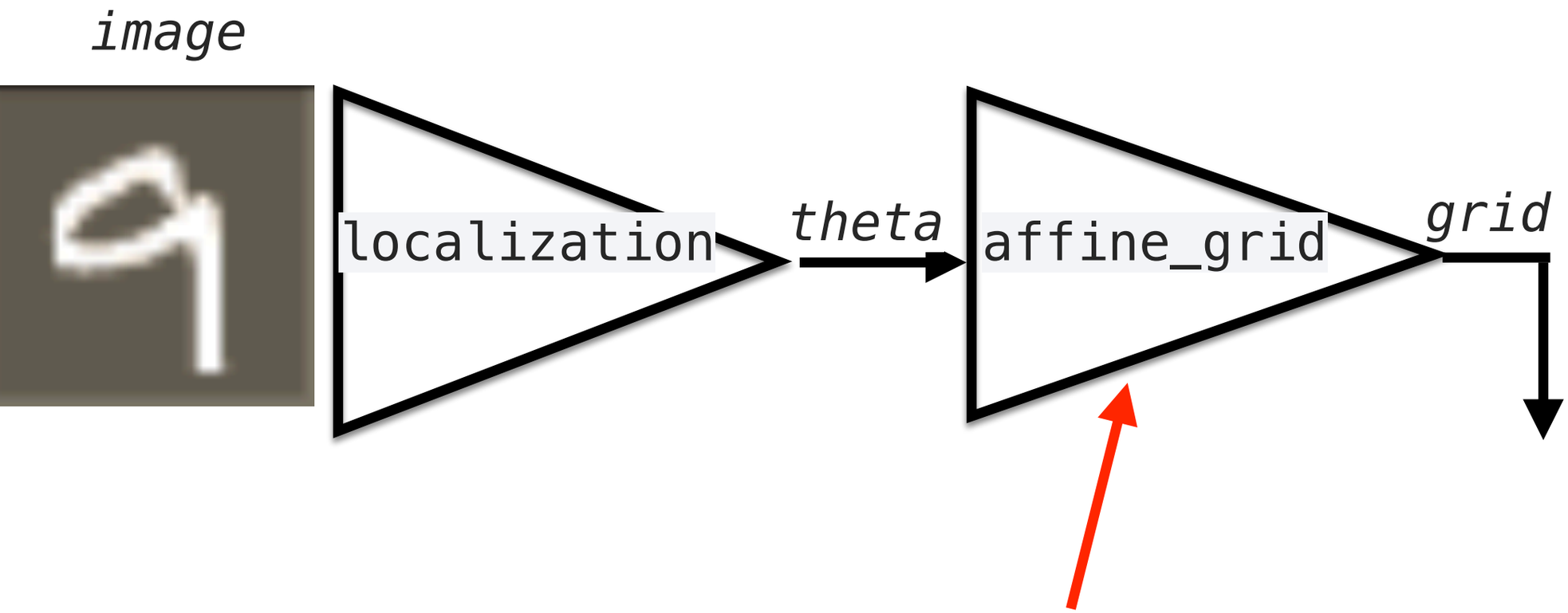
- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Spatial Transformer networks
- Architectures of feature matching networks

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



estimate parameters of 2D similarity transformation

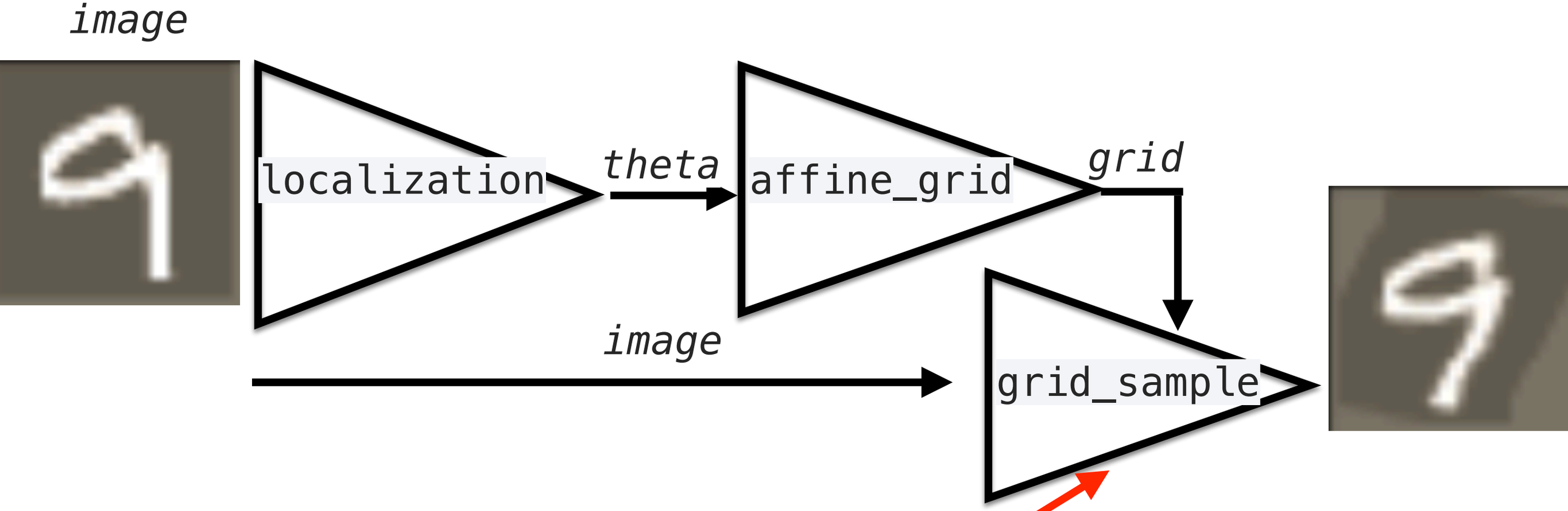
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



estimate pixel-wise correspondences of
the 2D similarity transformation

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

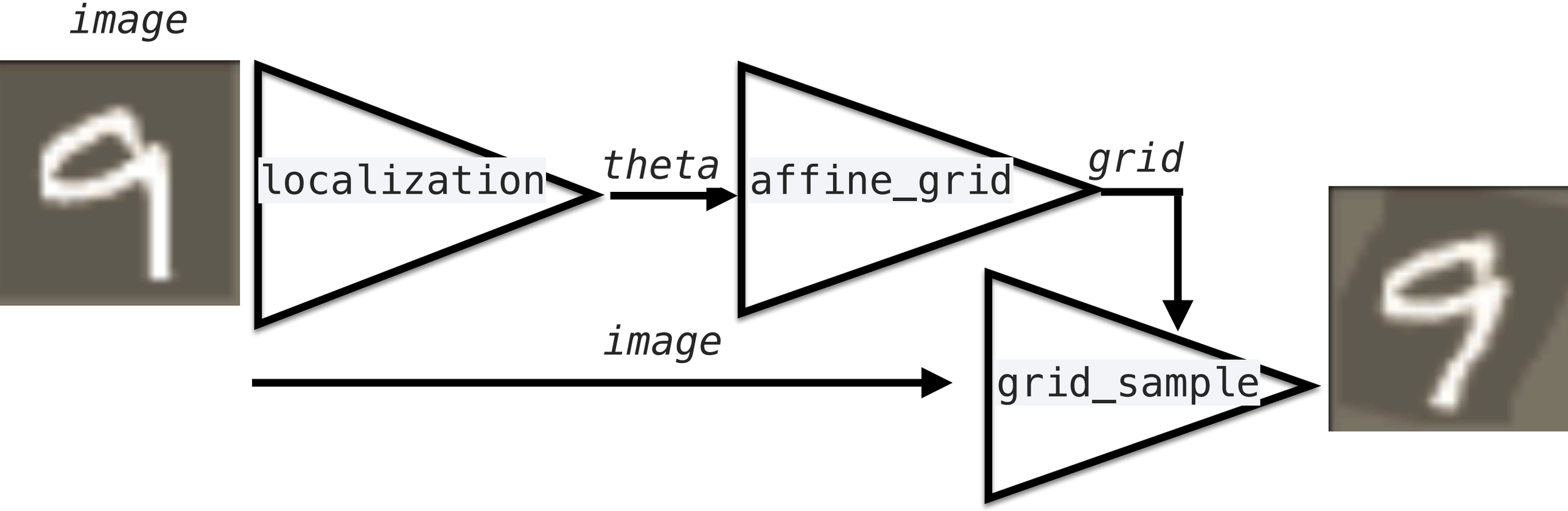
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf




```
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)
```

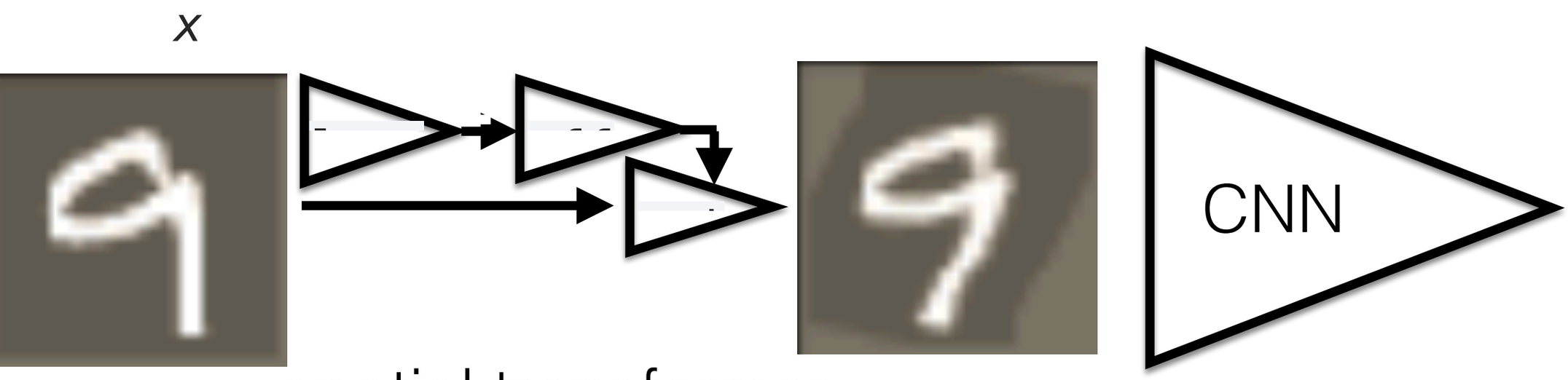
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf




```
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)
```

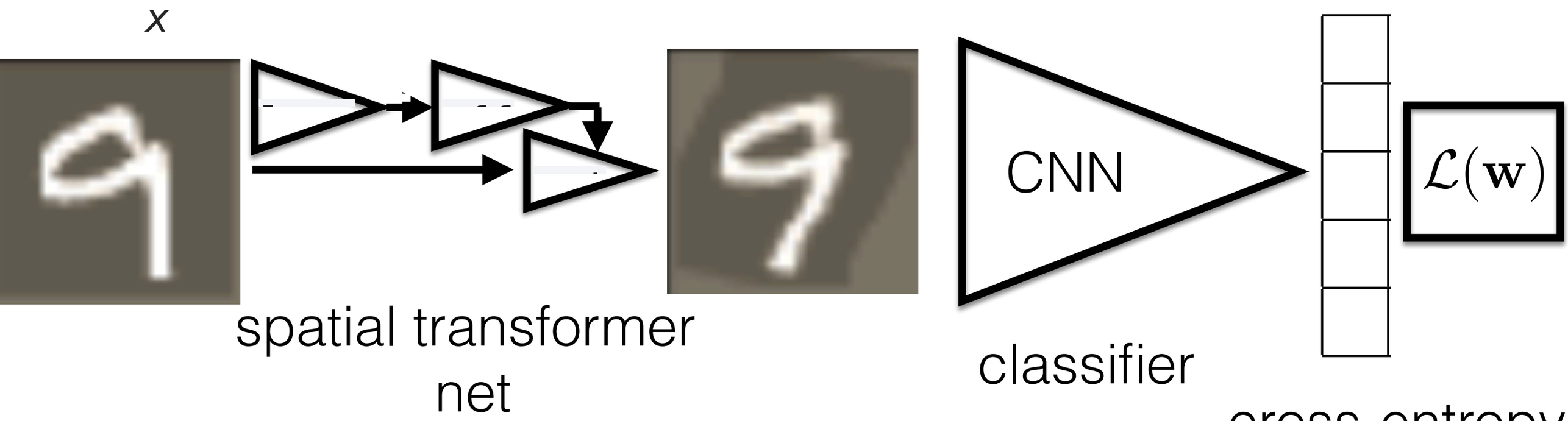
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



spatial transformer net

classifier

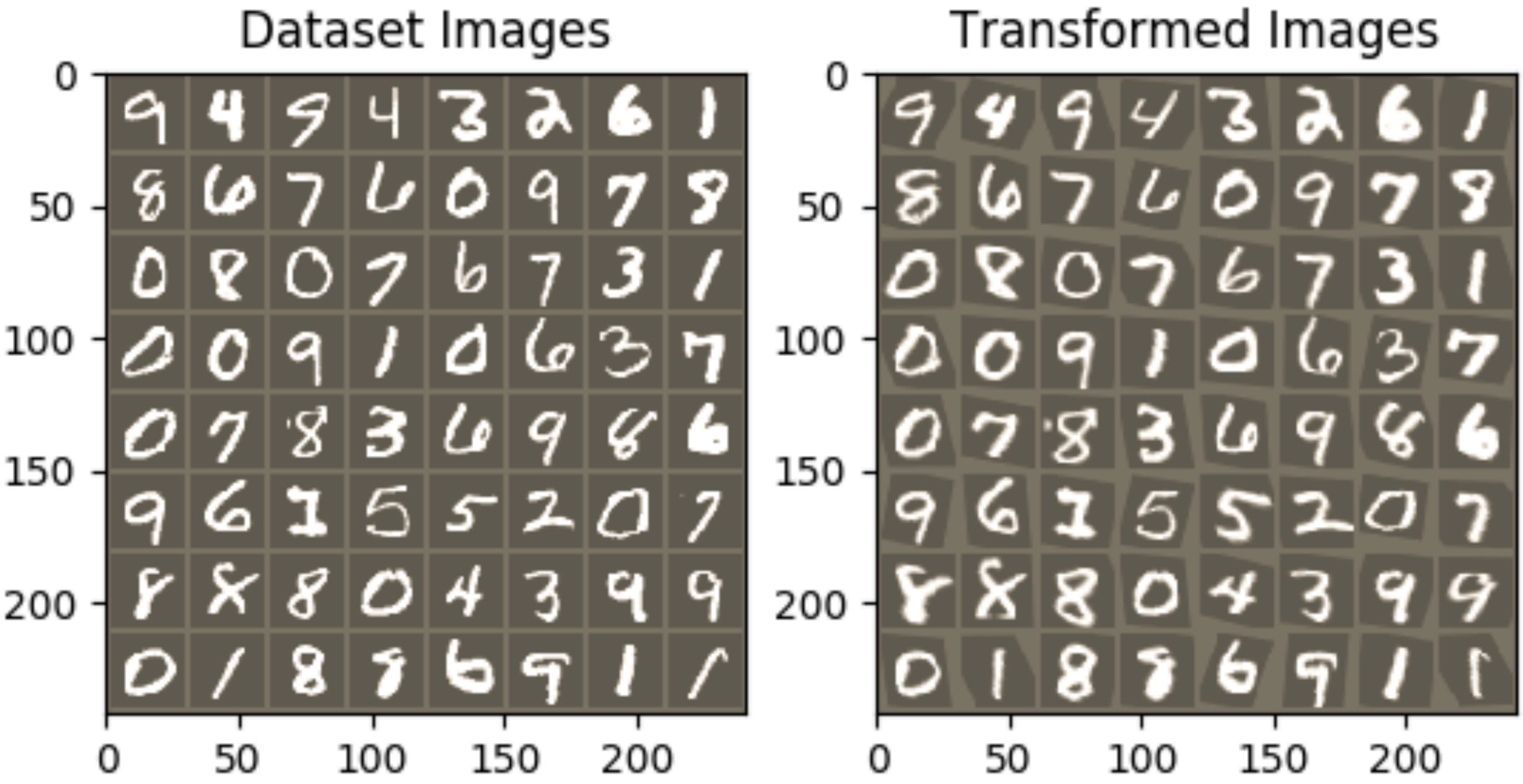
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



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

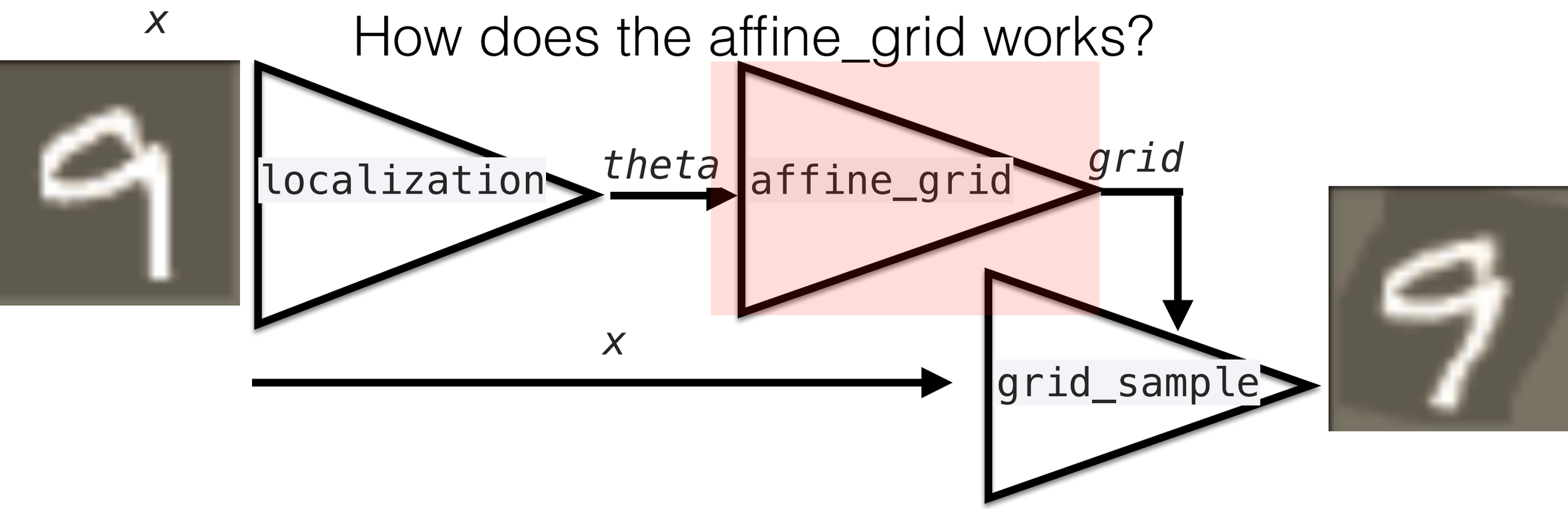
Spatial Transformer networks

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)

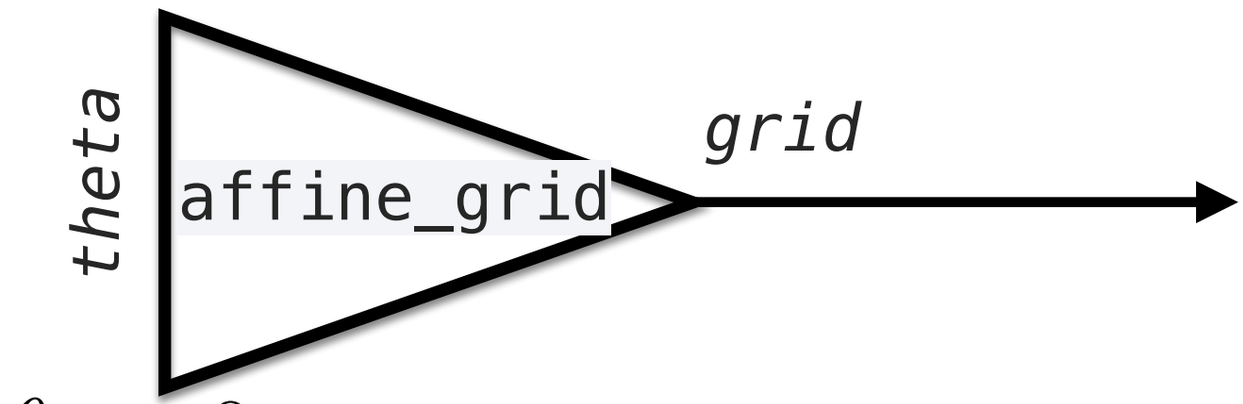
coordinates of all pixels in the transformed (grid)

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?



$$\theta_1 = 0$$

$$\theta_2 = 0$$

$$\theta_3 = 1$$

$x(m,n)$

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

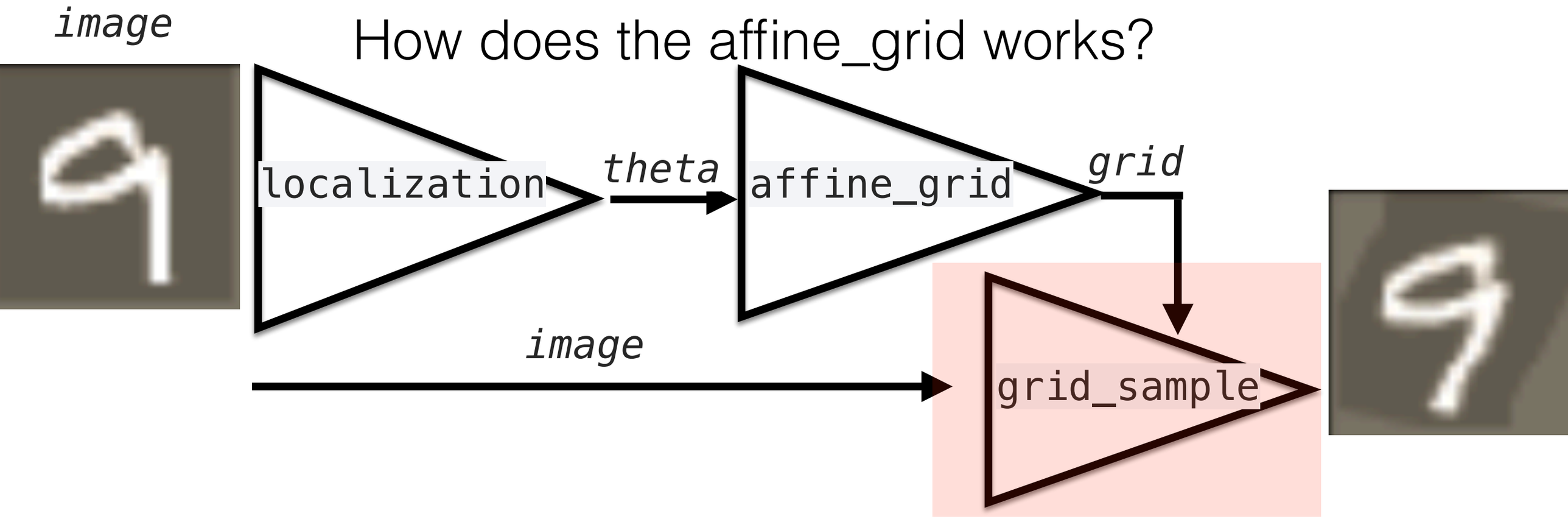
$y(m,n)$

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

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

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

How does the affine_grid works?

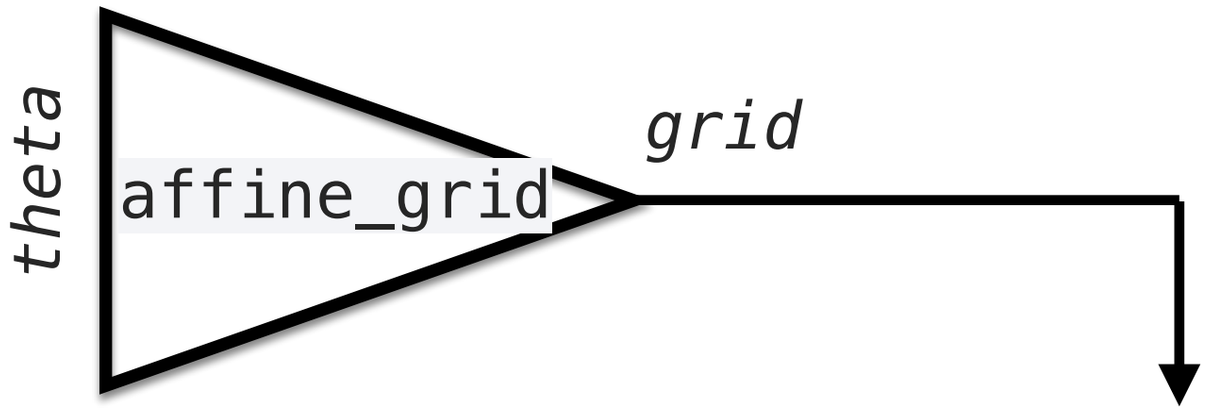


Spatial Transformer networks [Jaderberg 2016]

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

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

y

input image
(x, y)

$\otimes$

u

v

kernel
(u, v)

$=$

$m = 1$

$n = 1$

?			

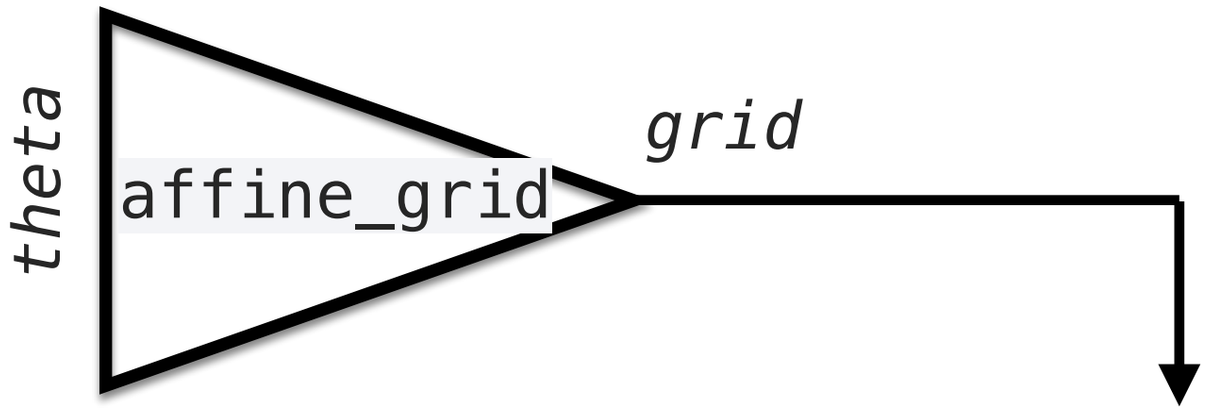
output image
(m, n)

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

x

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

y

input image (x, y)

$\otimes$

u

v

kernel (u, v)

$=$

$m = 1$

$n = 1$

?			

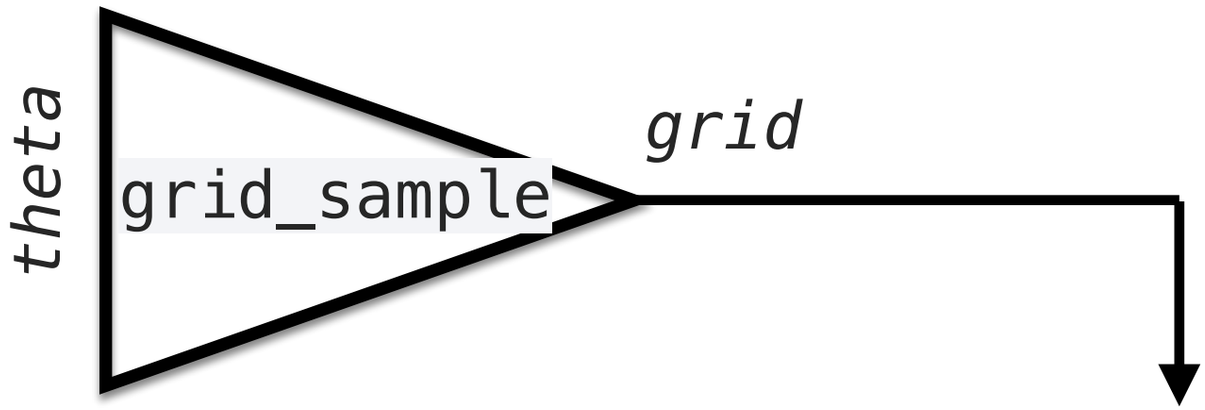
output image (m, n)

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

x

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

y

$\otimes$

u

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

v

$=$

$m = 1$

$n = 1$

?			

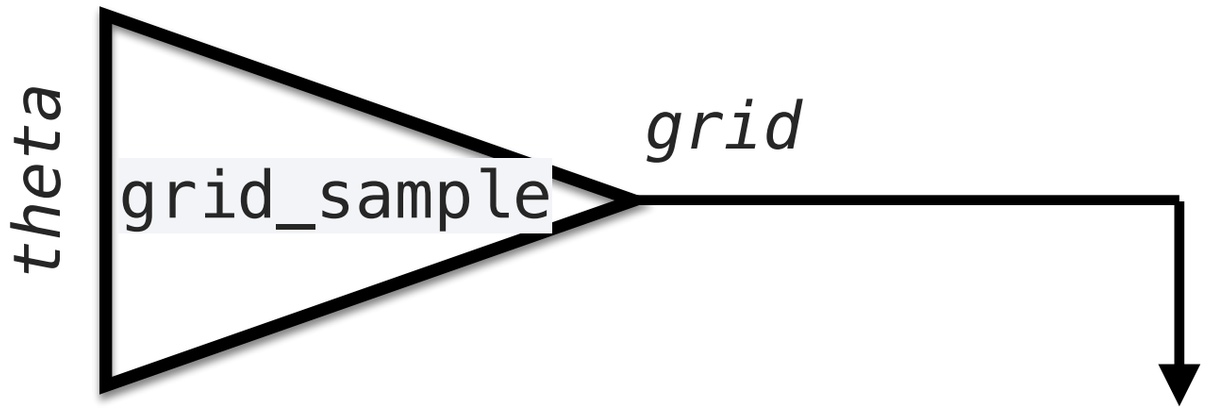
$$\kappa_{(m,n)}(u,v) = \delta\big(u - x(m,n)\big) \cdot \delta\big(v - y(m,n)\big)$$

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

x

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

y

$\otimes$

u

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

v

$=$

$m = 2$

$n = 1$

6	3		

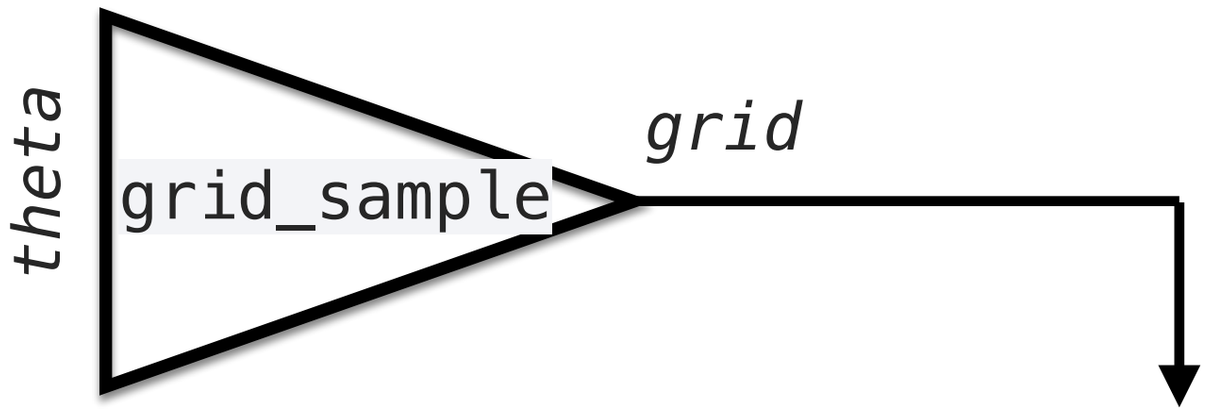
$$\kappa_{(m,n)}(u,v) = \delta\big(u - x(m,n)\big) \cdot \delta\big(v - y(m,n)\big)$$

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

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

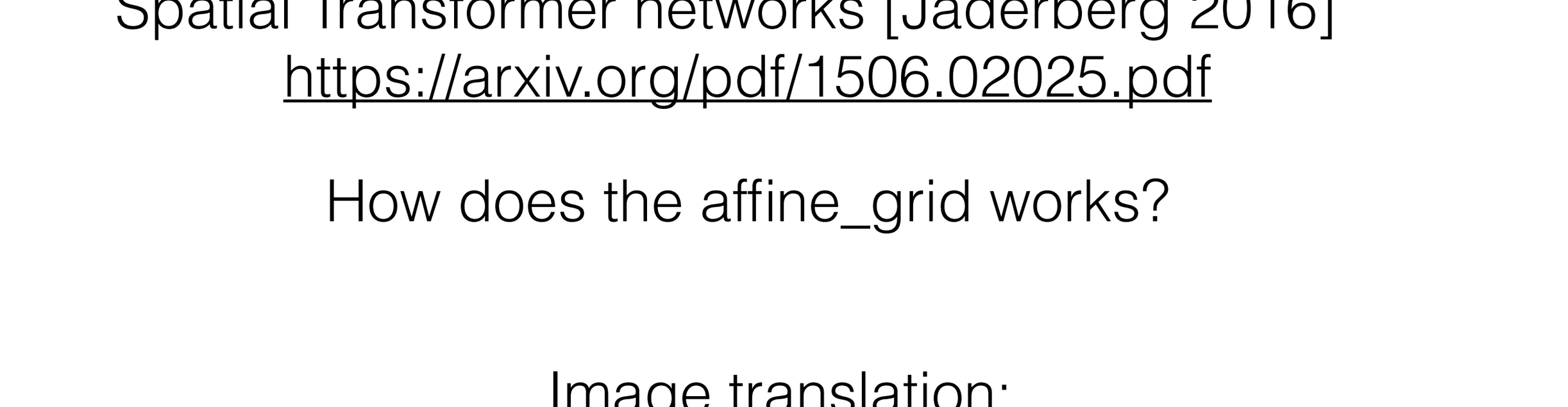
$$\kappa_{(m,n)}(u,v) = \delta\big(u - x(m,n)\big) \cdot \delta\big(v - y(m,n)\big)$$

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?

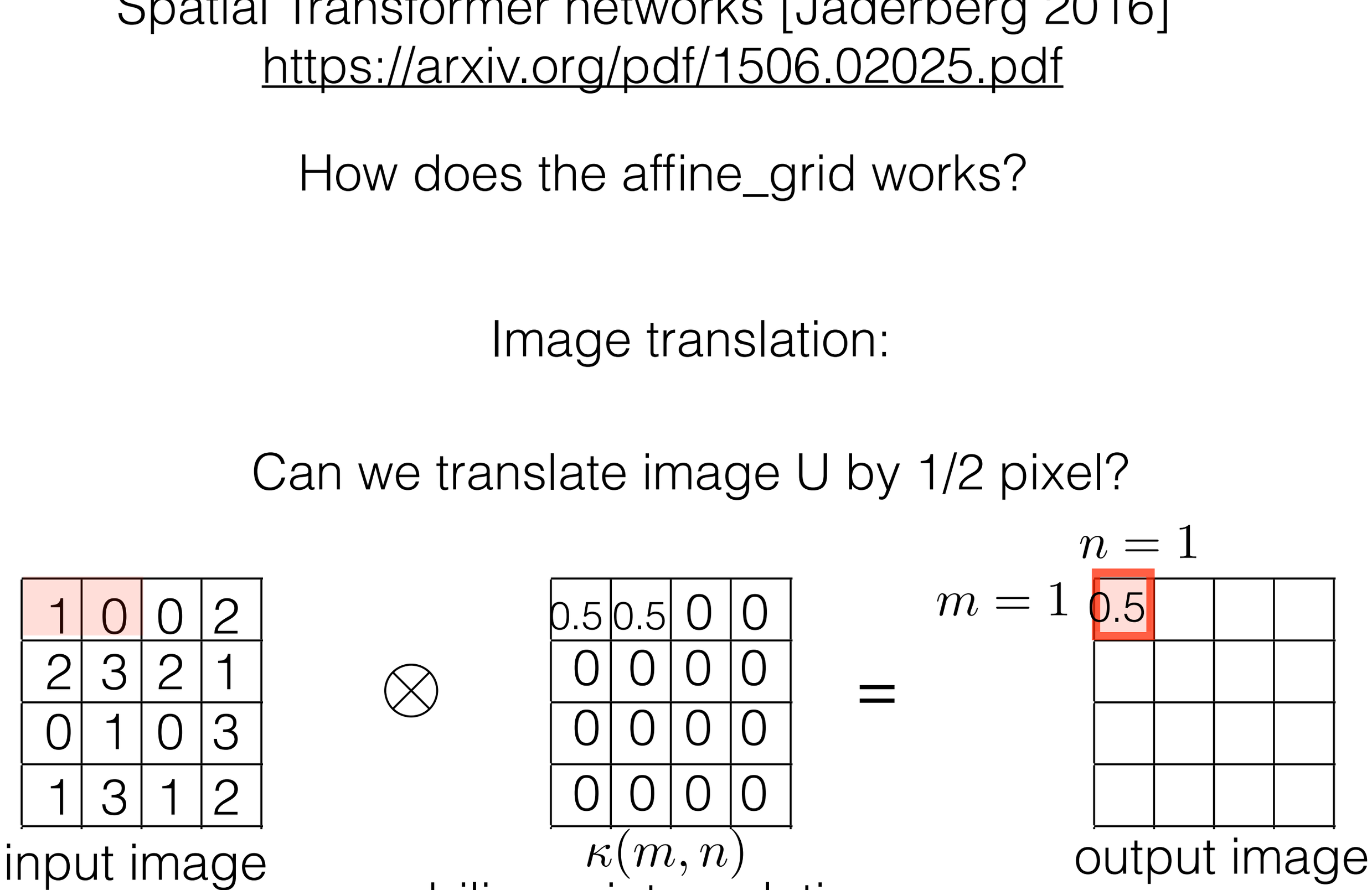


Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image translation:

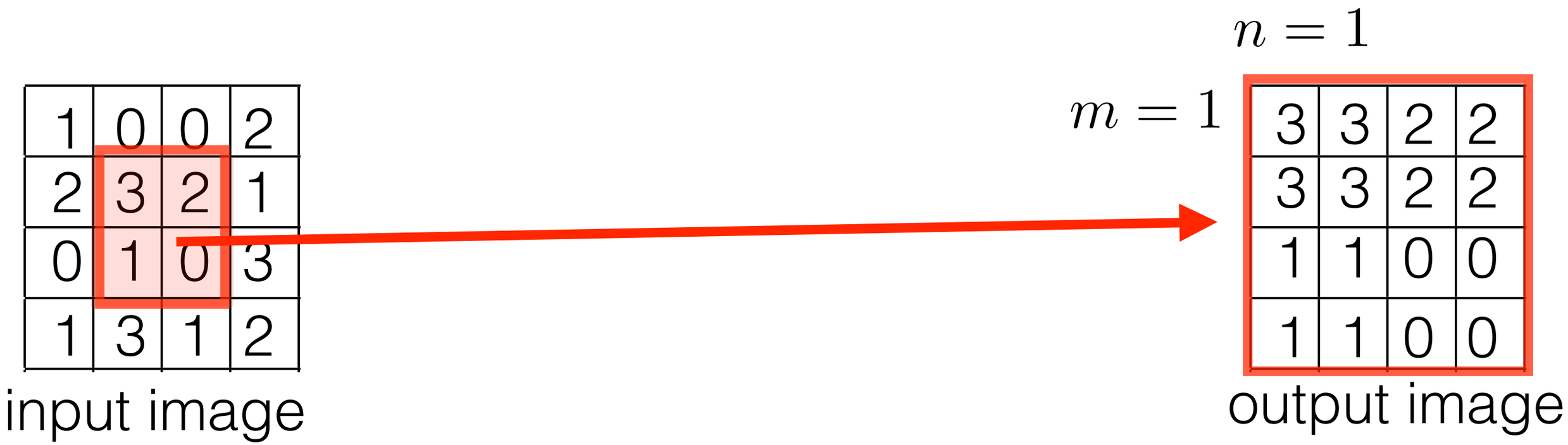
Can we translate image U by 1/2 pixel?



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

How does the affine_grid works?

Image crop:

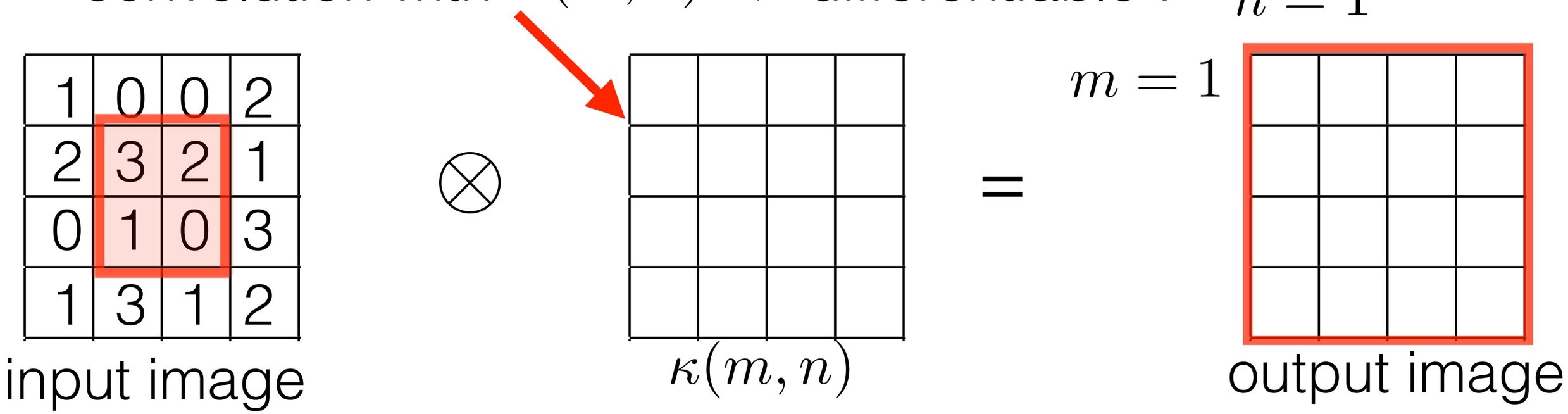


Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:

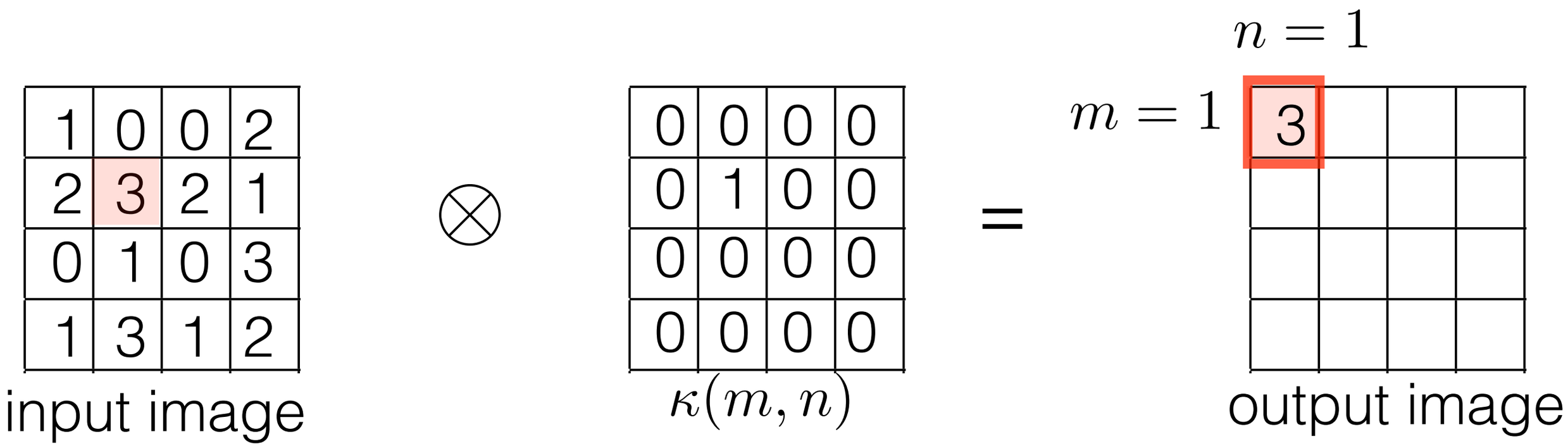
convolution with $\kappa(m,n)$ => differentiable !



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

How does the affine_grid works?

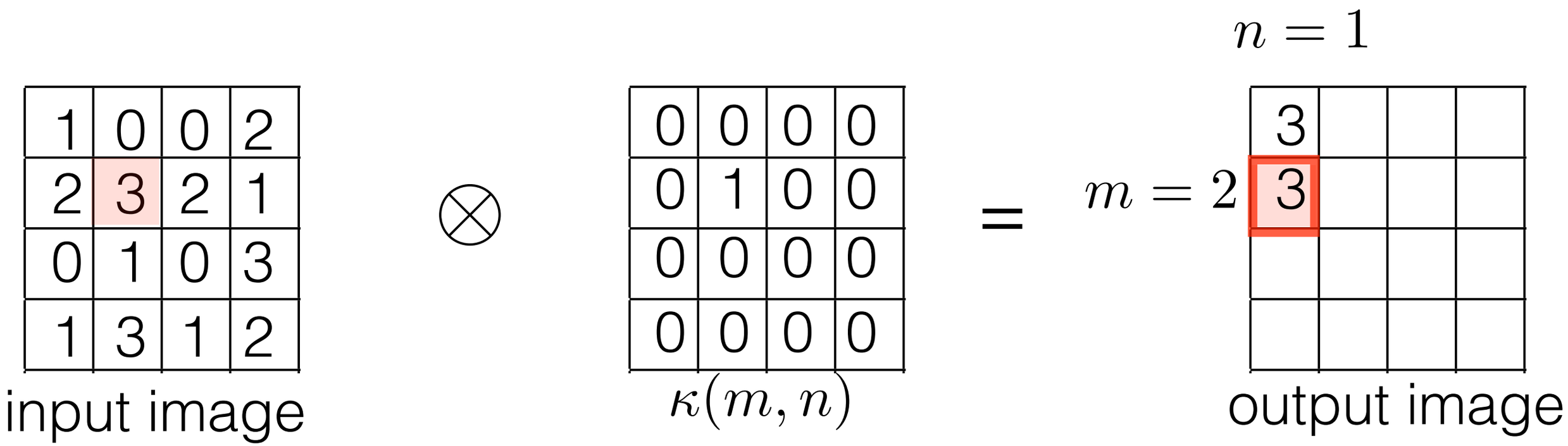
Image crop:



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

How does the affine_grid works?

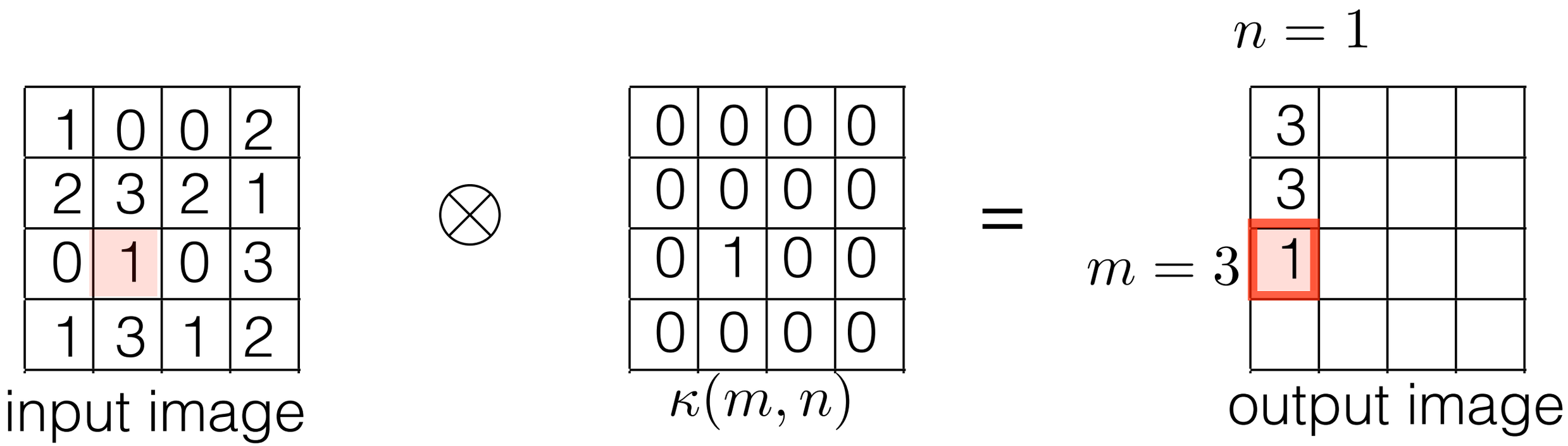
Image crop:



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

How does the affine_grid works?

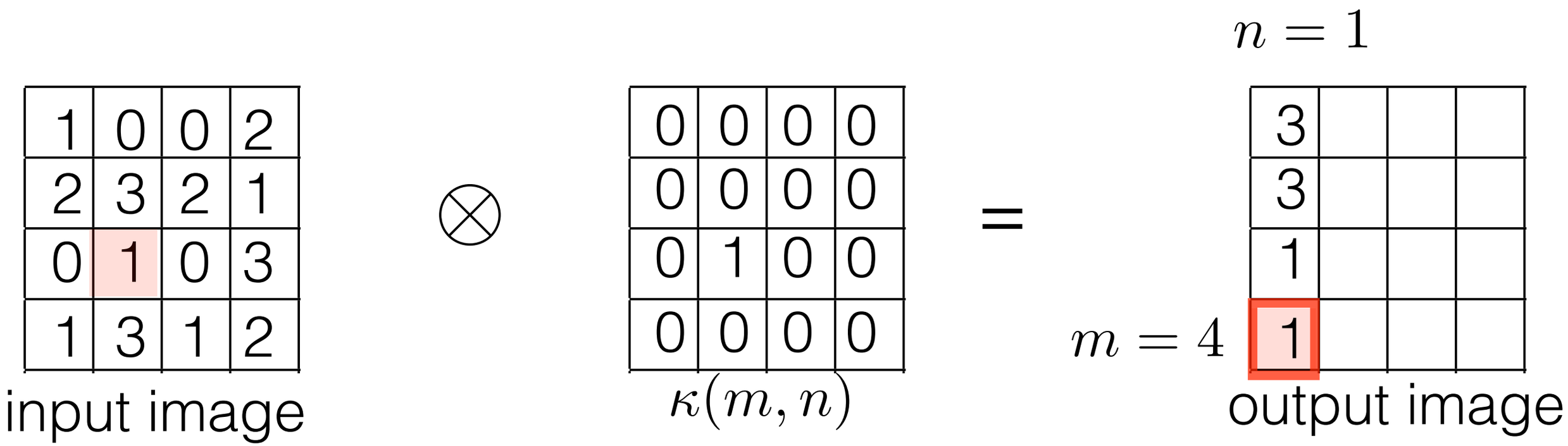
Image crop:



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

How does the affine_grid works?

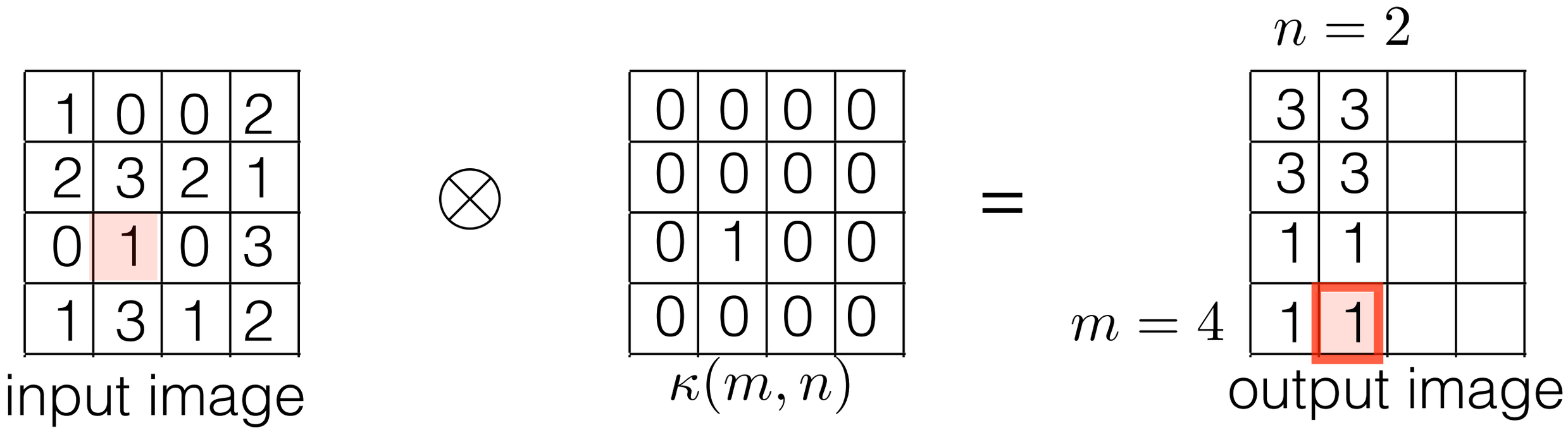
Image crop:



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

How does the affine_grid works?

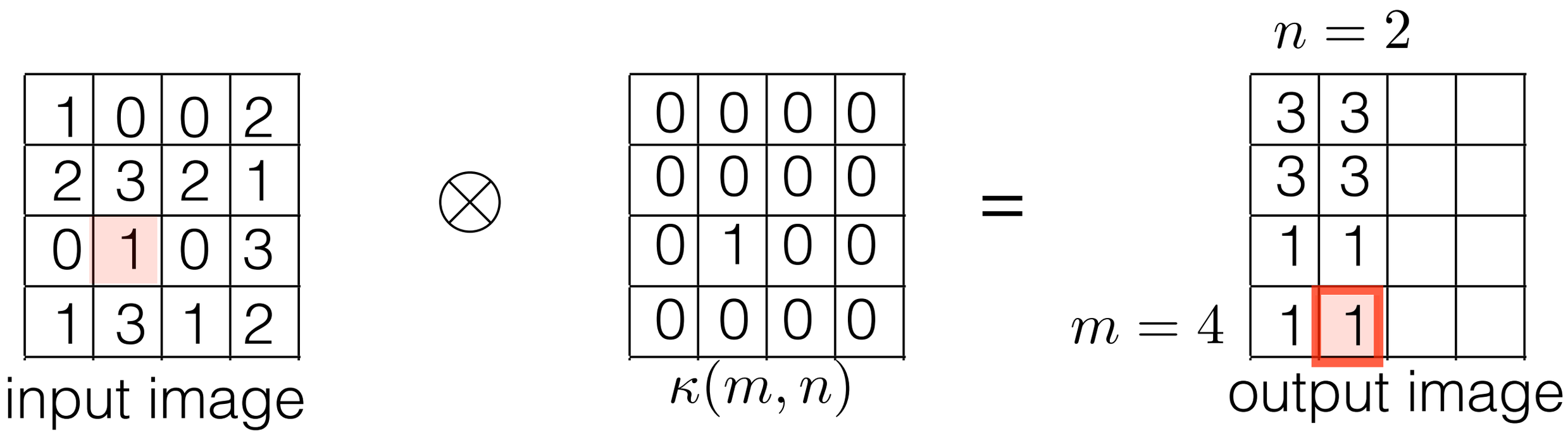
Image crop:



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

How does the affine_grid works?

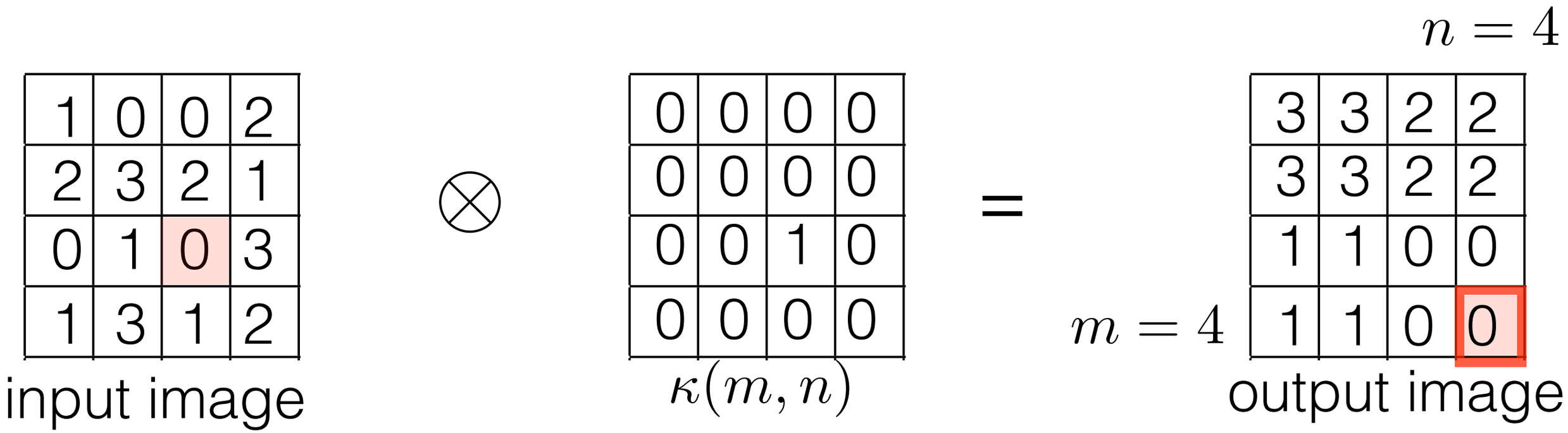
Image crop:



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

How does the affine_grid works?

Image crop:

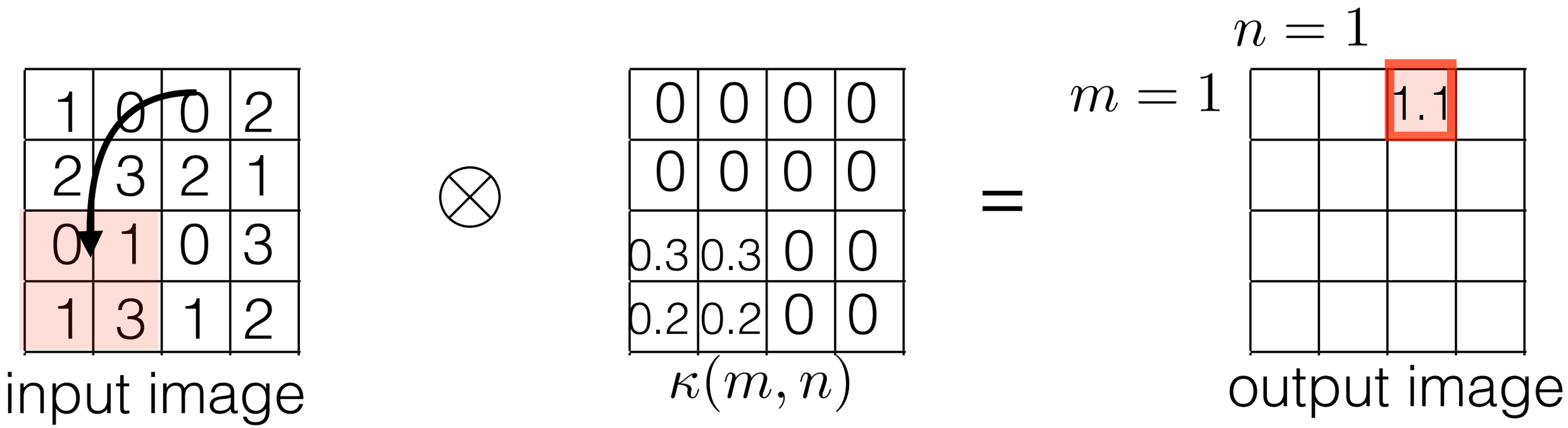


Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image translation:

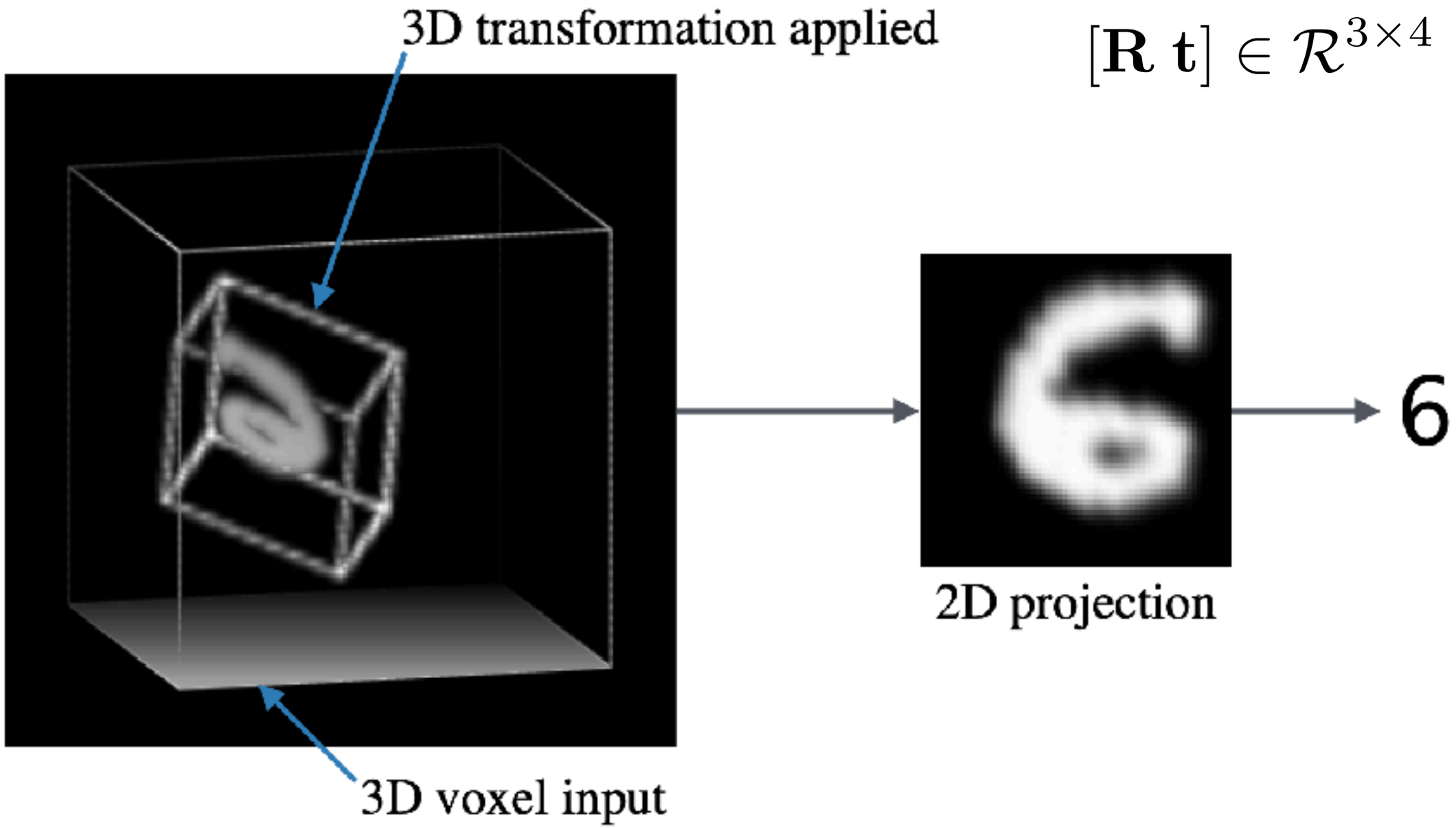
What about rotation?



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

Also implemented 3D affine_grid transformation layer:

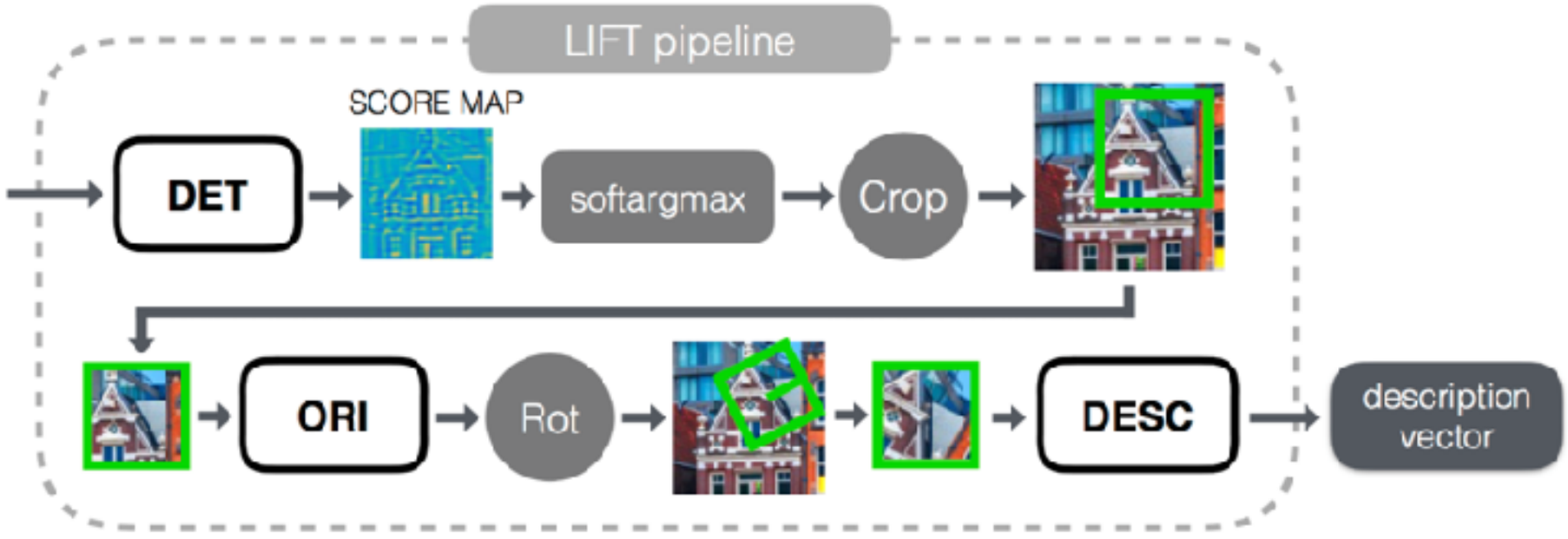
$$[\mathbf{R}\ \mathbf{t}] \in \mathcal{R}^{3\times 4}$$



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

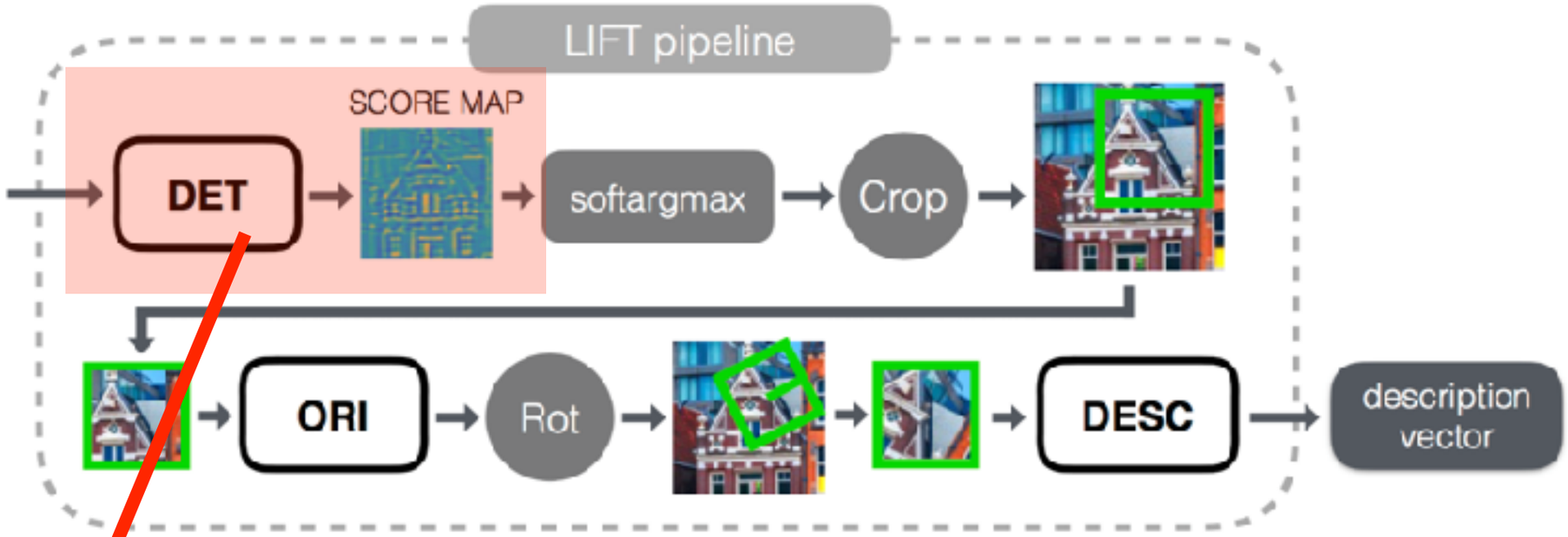


Input: RGB image

Output: set of detected feature points with descriptors

Descriptor is vector which is:
- similar for corresponding points
- and dissimilar for not corresponding points.

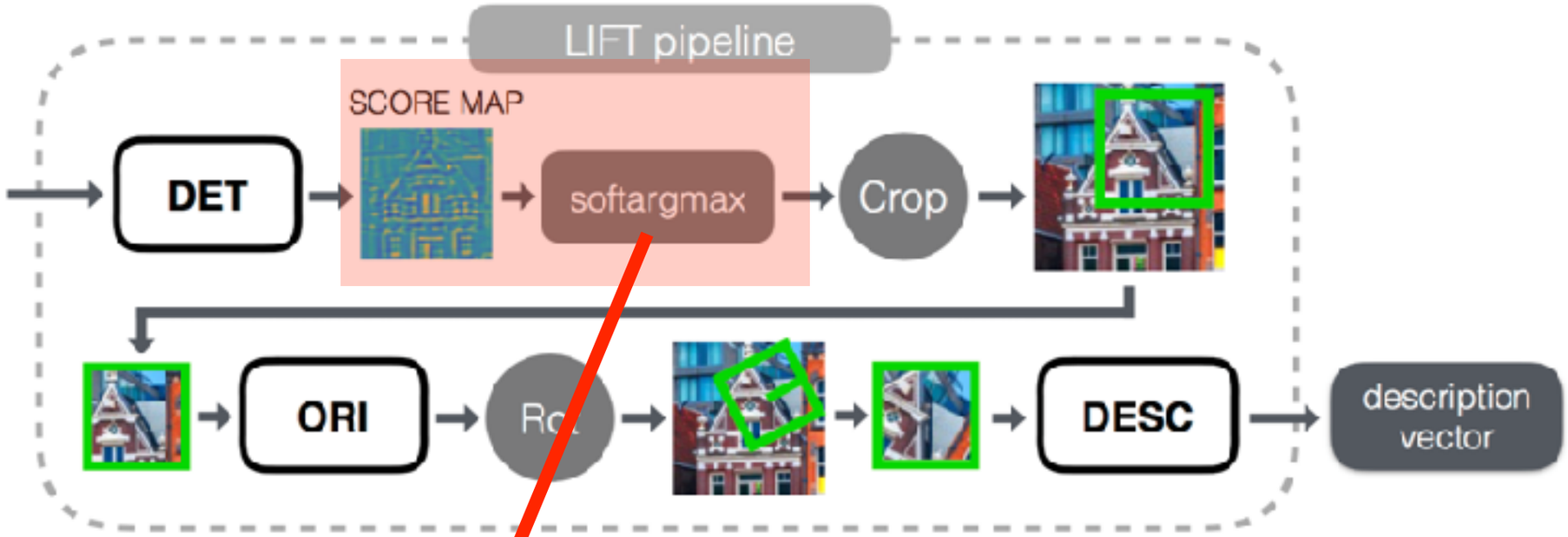
LIFT: Learnable Invariant Feature Descriptors
[Yi et al ECCV 2016] https://arxiv.org/abs/1603.09114

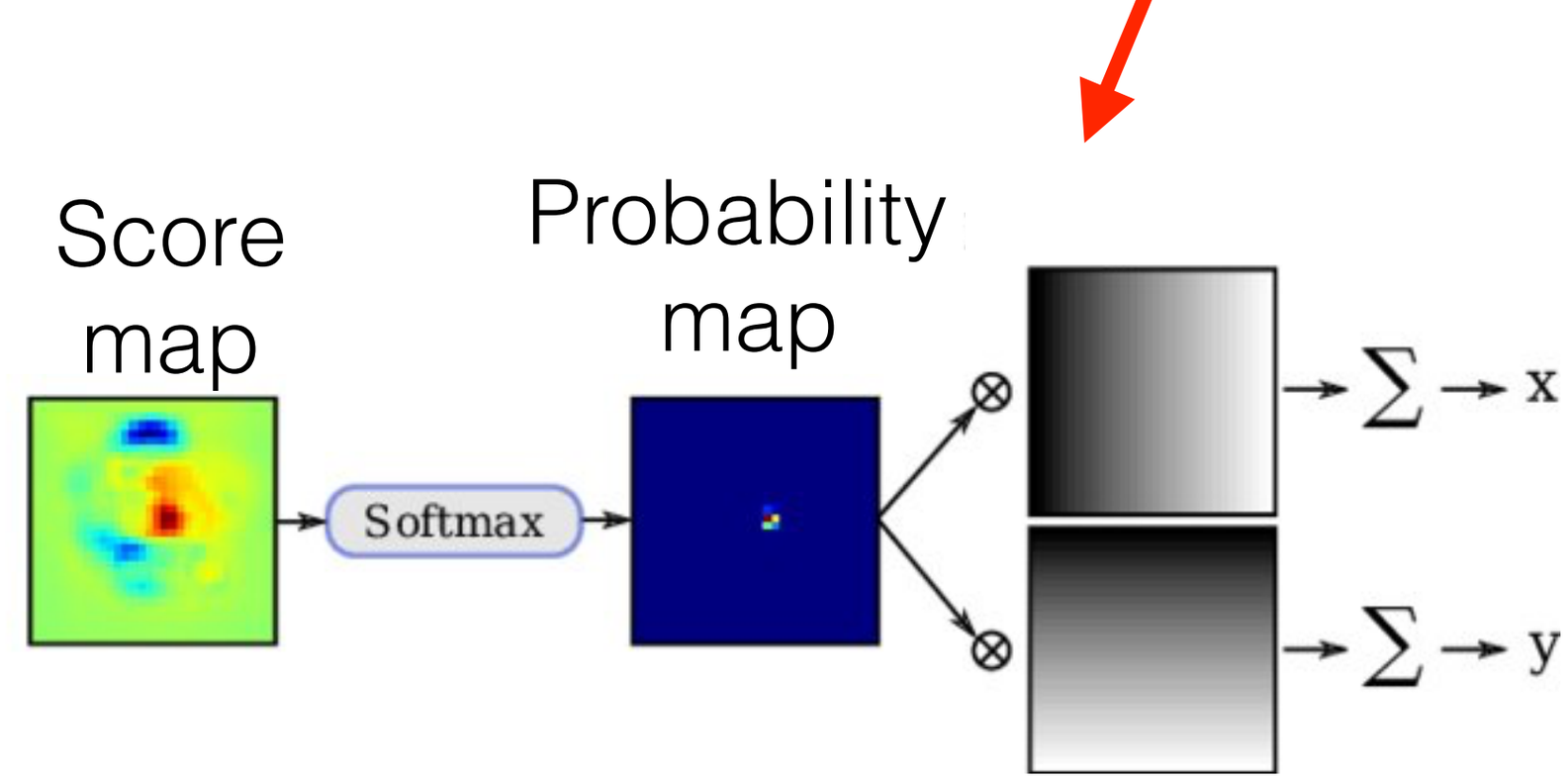


Segmentation CNN for pixel-wise two-class labelling

- 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

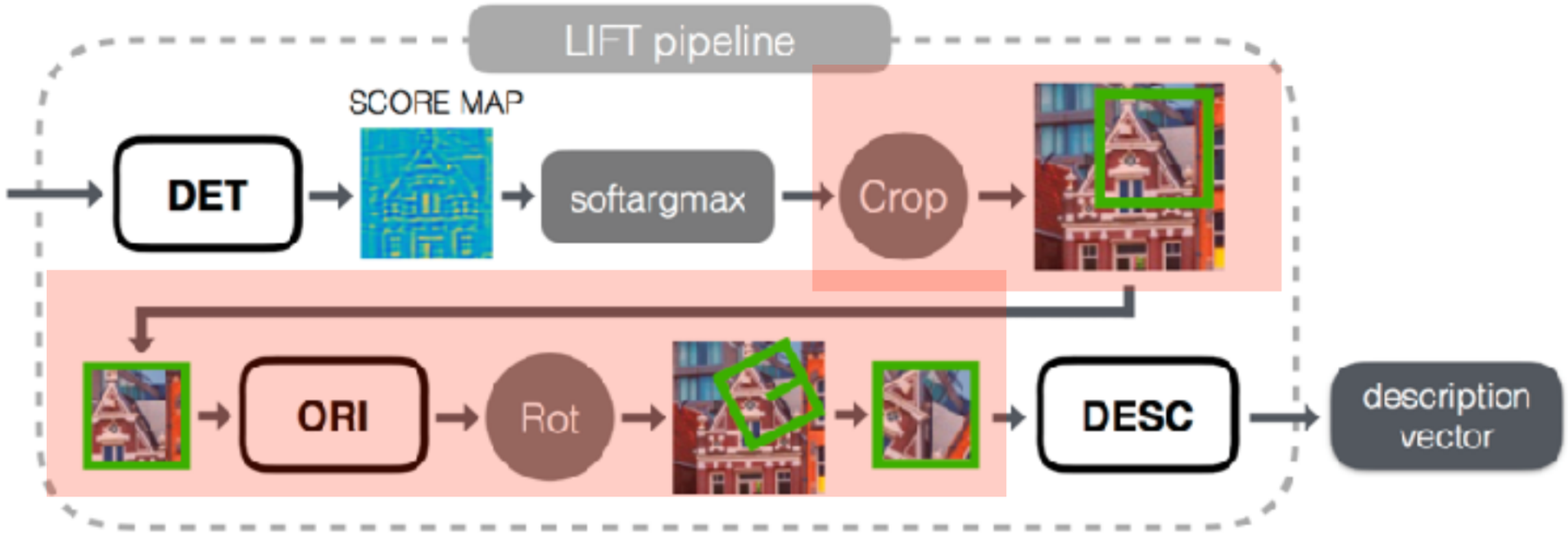




Expected value of x and y under the prob. distr.

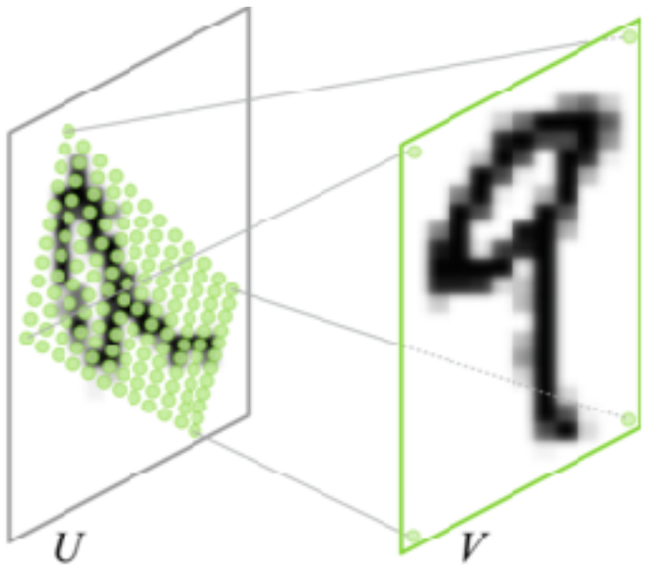
Learnable Invariant Feature Descriptor (LIFThttps://www.researchgate.net/publication/323410987_2D3D_Pose_Estimation_and_Action_Recognition_using_Multitask_Deep_Learning/figures?lo=1)

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



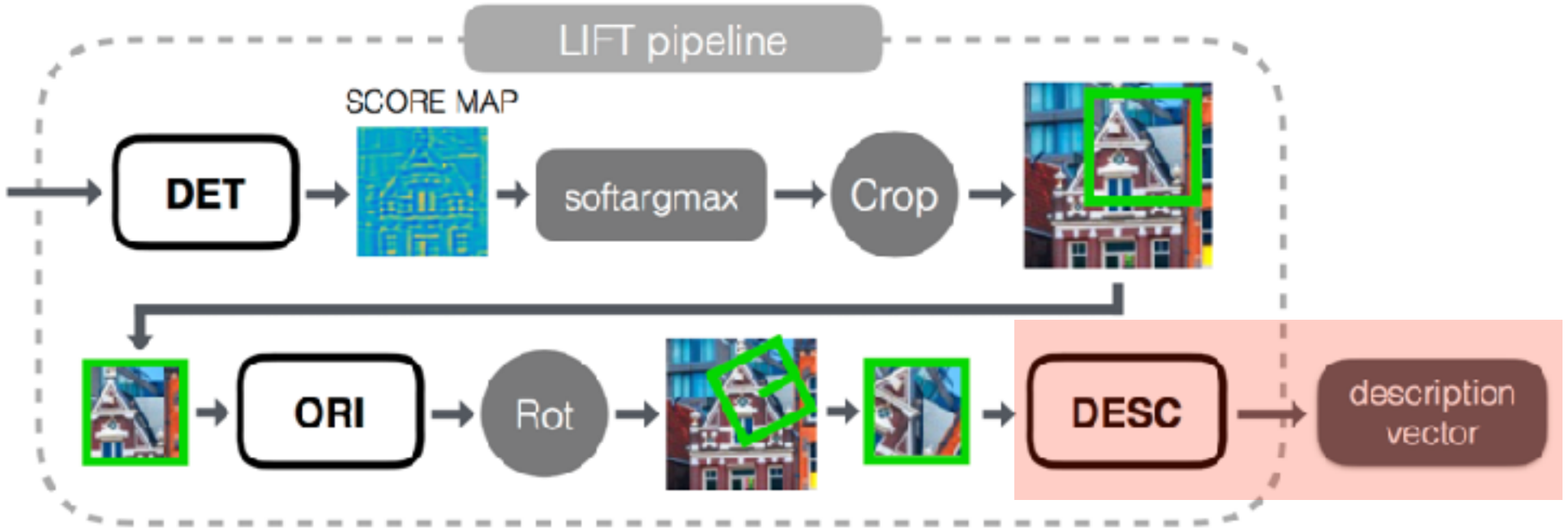
Spatial Transformer Network

Bilinear approximation of affine transformation is differentiable !

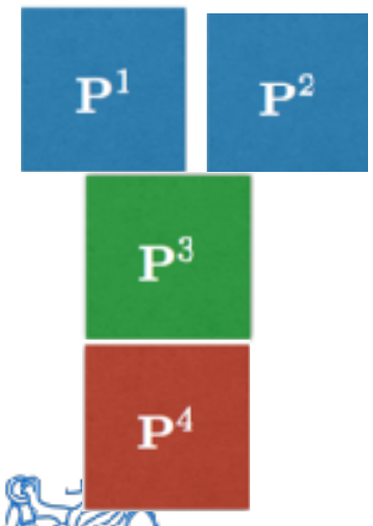


[Jaderberg, 2016] https://arxiv.org/pdf/1506.02025.pdf

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:

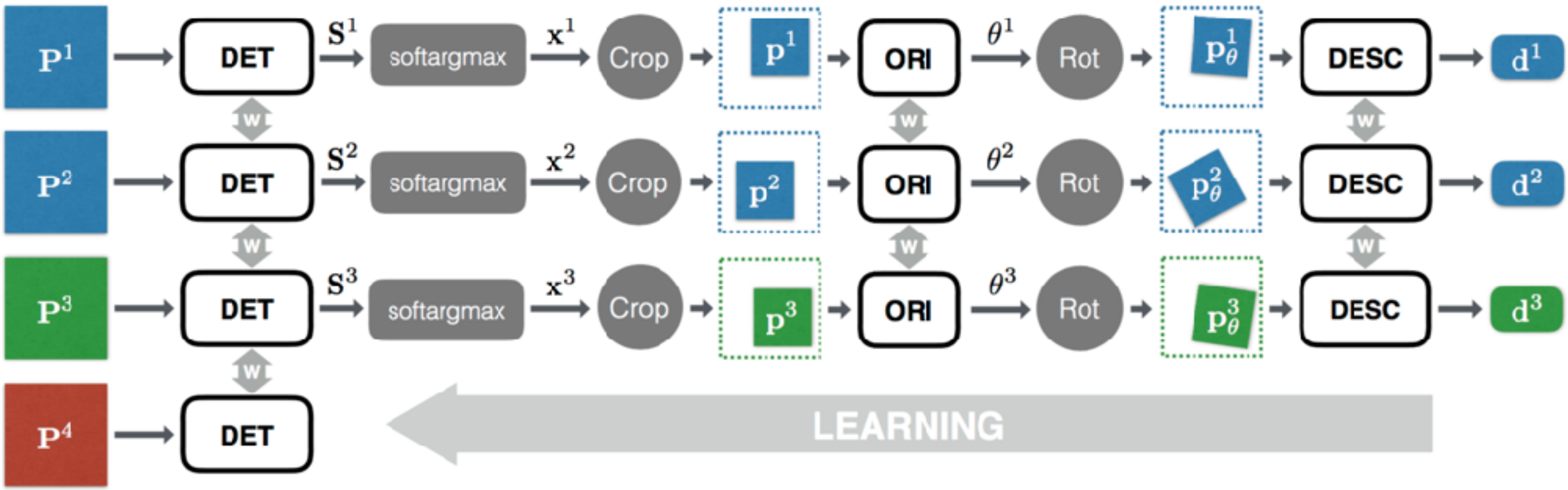


Two corresponding patches on distinctive points

One not corresponding patch on a distinctive point

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