# 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
<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: | Output: |
|---|---|
| Scale | Scale |
| T-shirt | T-shirt |
| **Steel drum** | Giant panda |
| Drumstick | Drumstick |
| Mud turtle | Mud turtle |

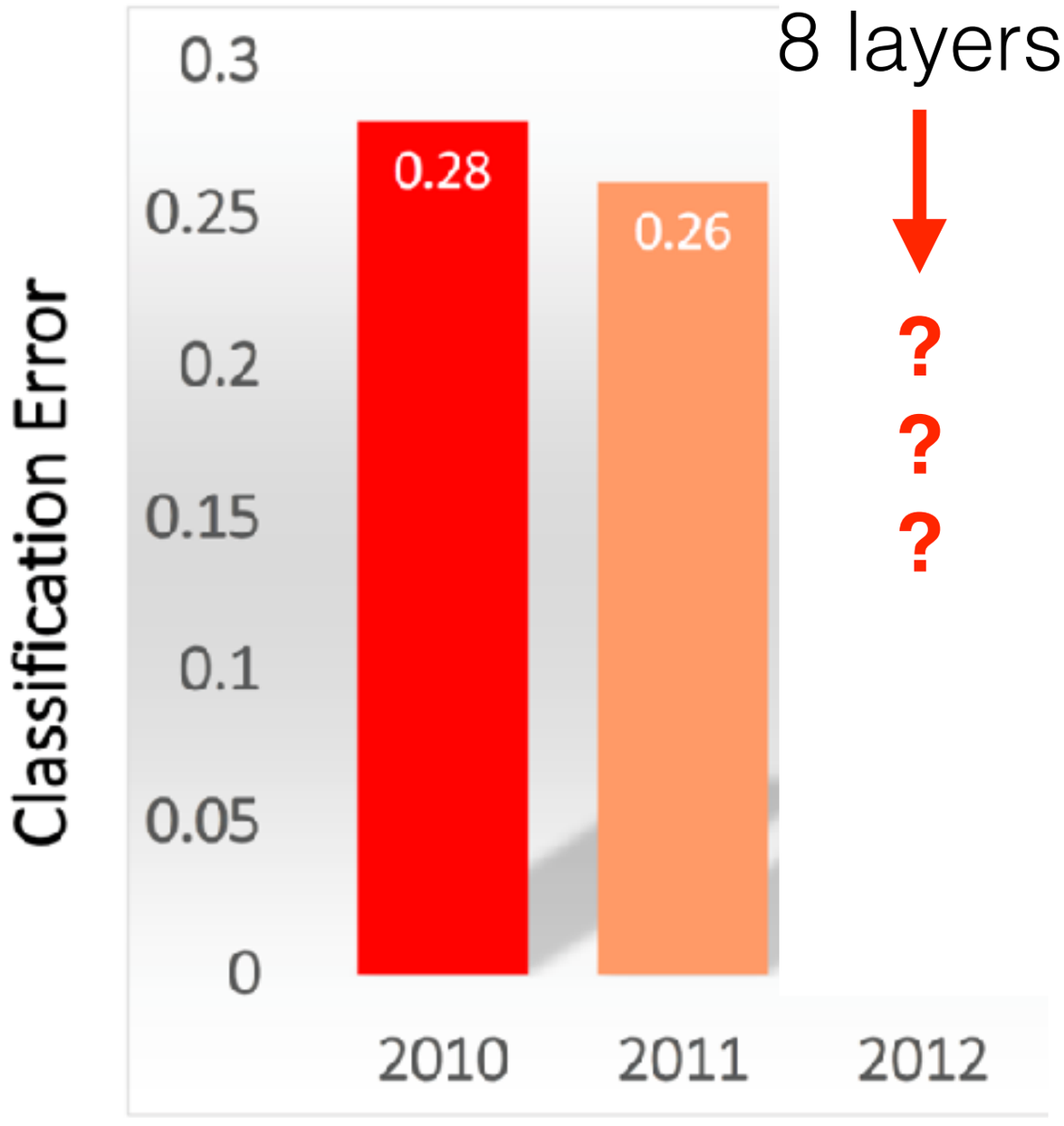
✓ ✗

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

IMAGENET

# Classification results

AlexNet
8 layers

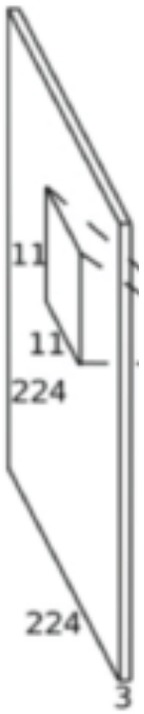



?

?

?

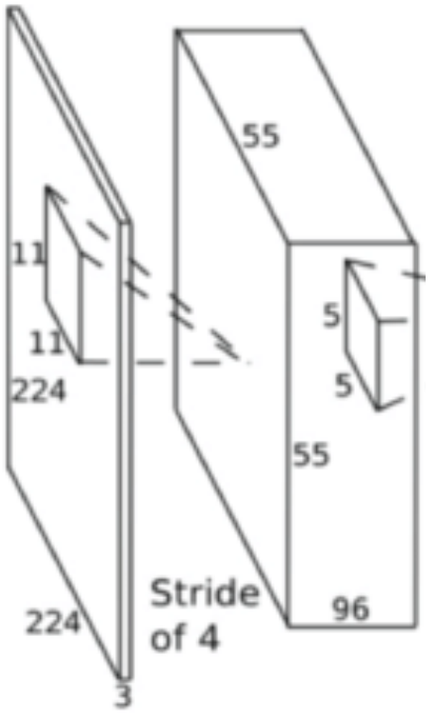
# AlexNet on ImageNet 2012 (**over 27k citations !!!**)




- 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

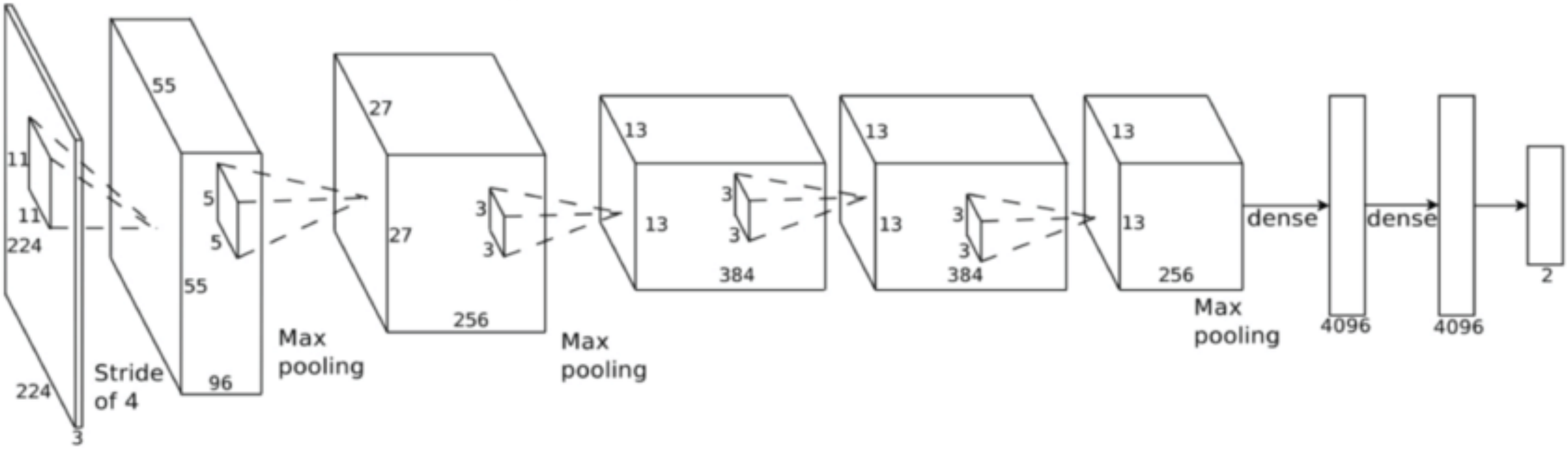
# AlexNet on ImageNet 2012 (**over 27k citations !!!**)




- 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 !!!**)



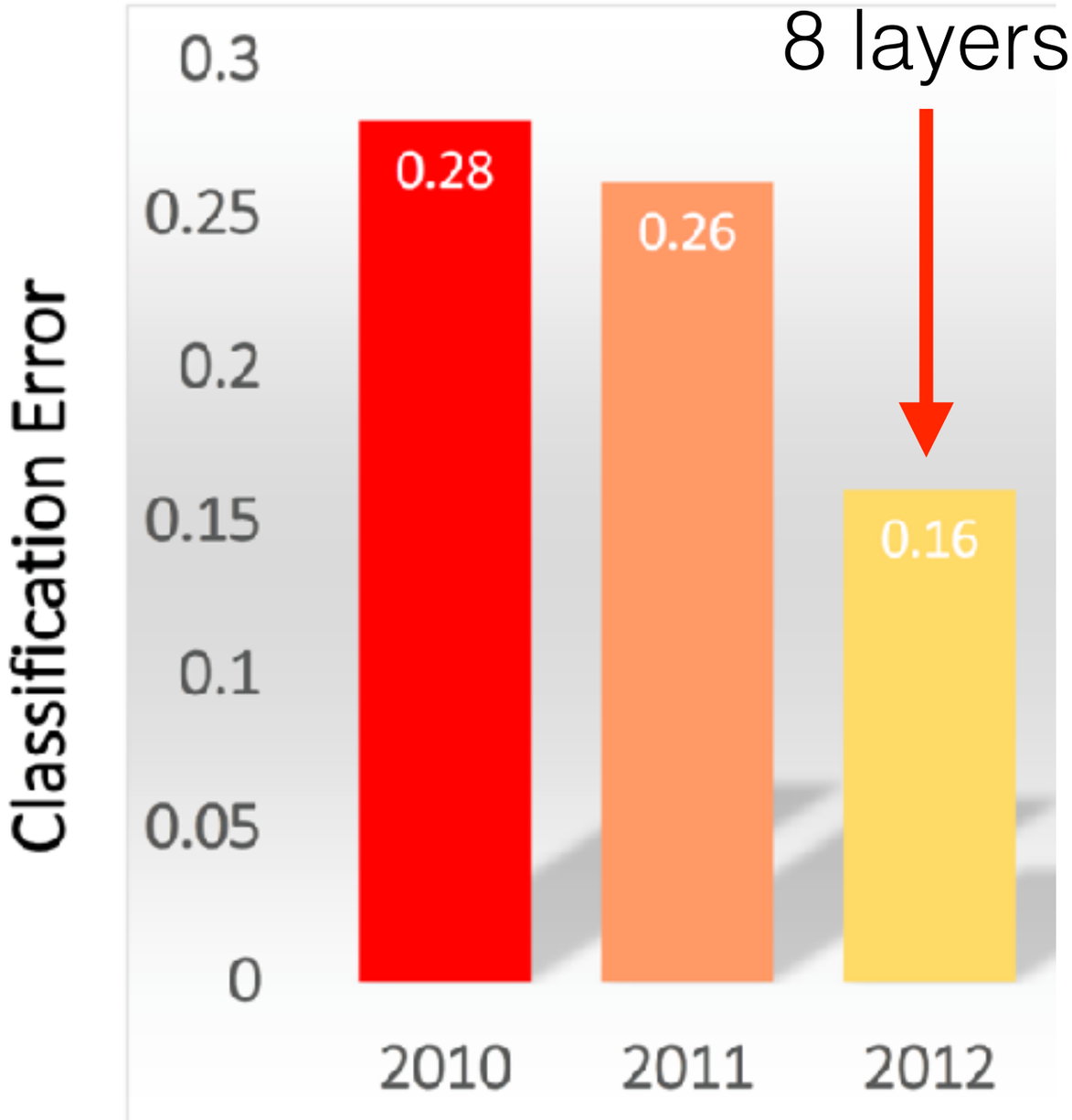

- Param in layer1 (conv, 96 11x11 filters, stride=4, pad=0)?
- Param in layer2 (maxp,3x3 filters, stride=2, pad=0)?
- Param in layer3 (conv, 256 5x5 filters, stride=1, pad=2?
- Parameters in total: 60M, Depth: 8 layers

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

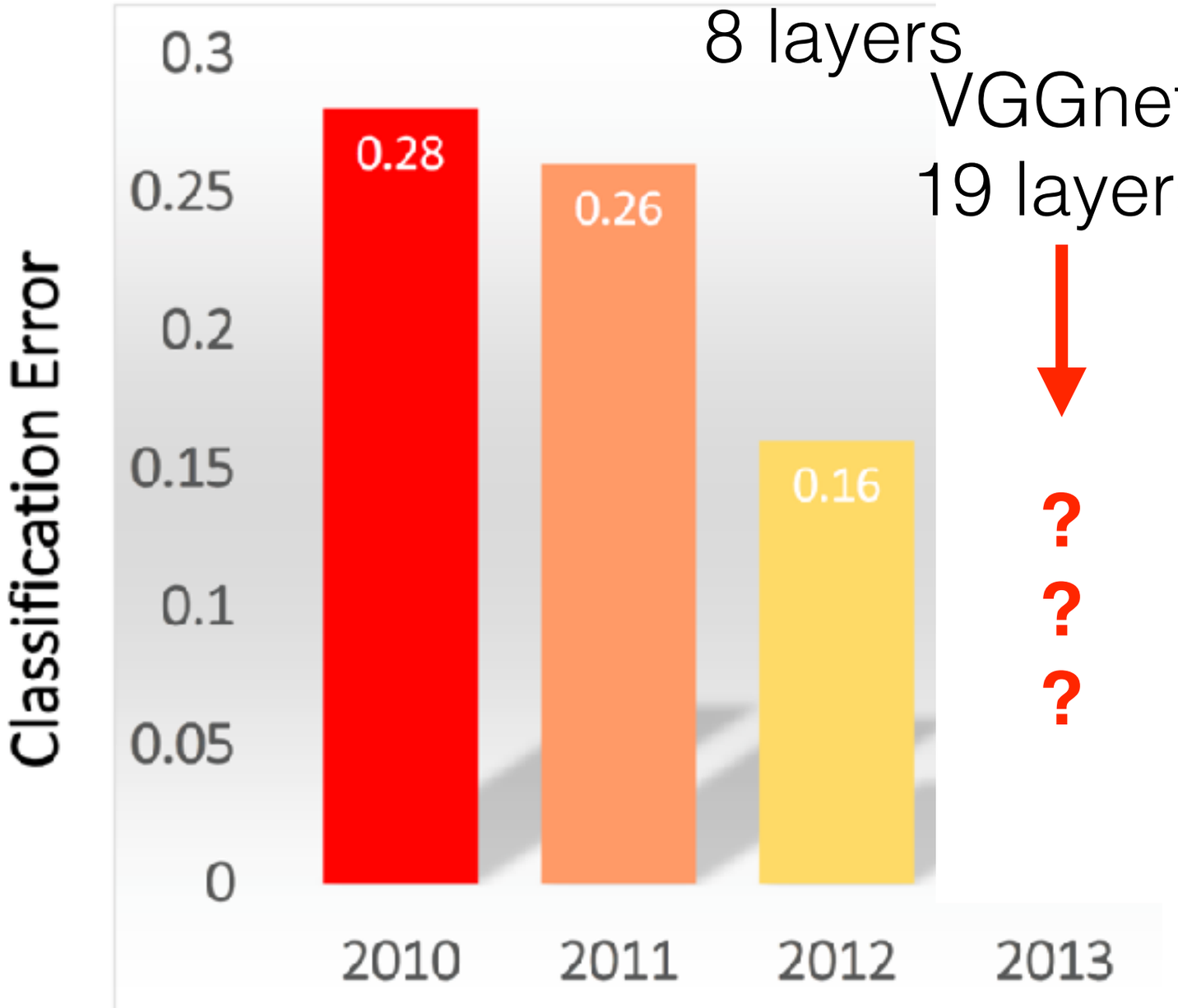
# Classification results

AlexNet
8 layers



IMAGENET

# Classification results

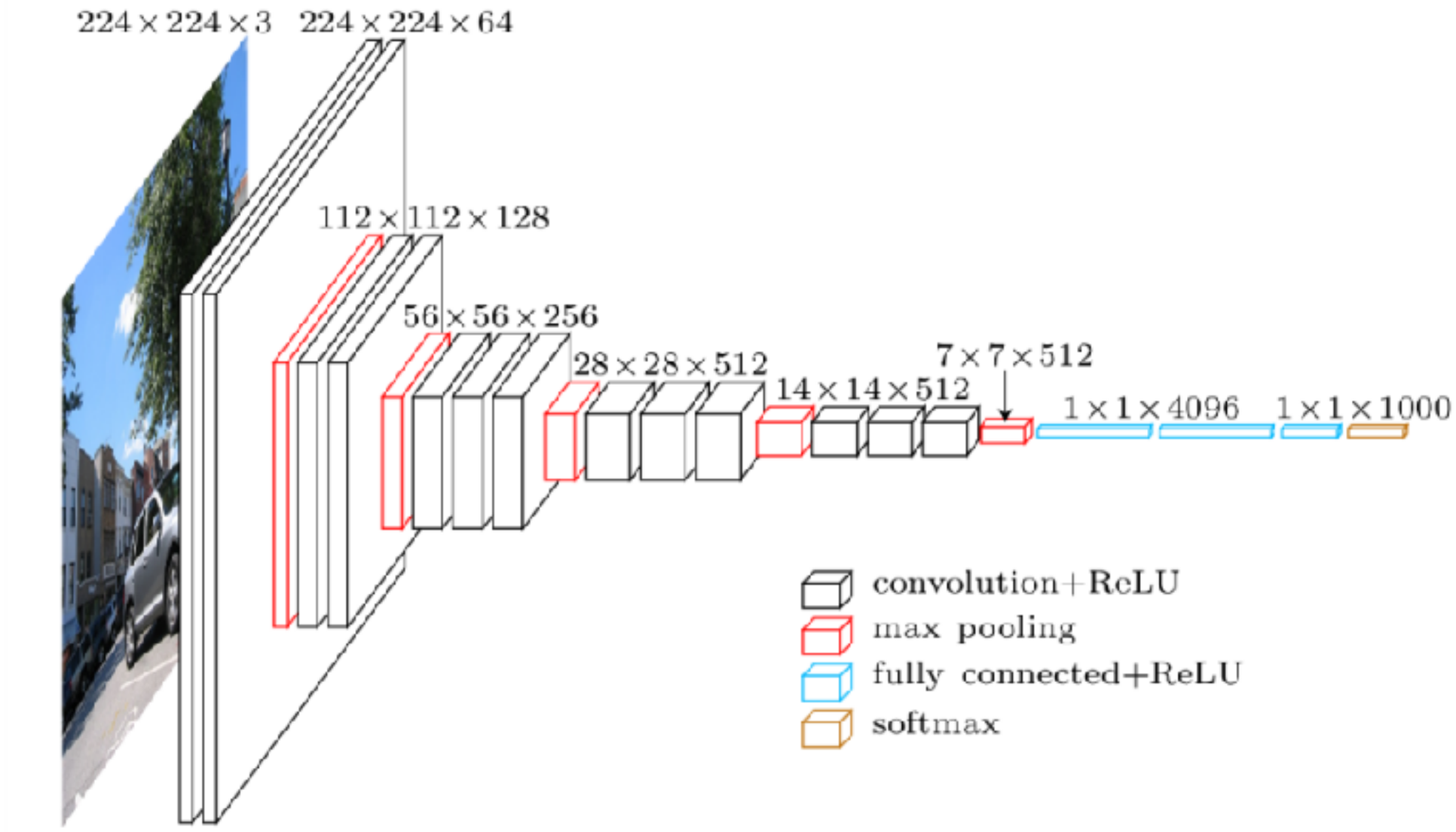



AlexNet
8 layers

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

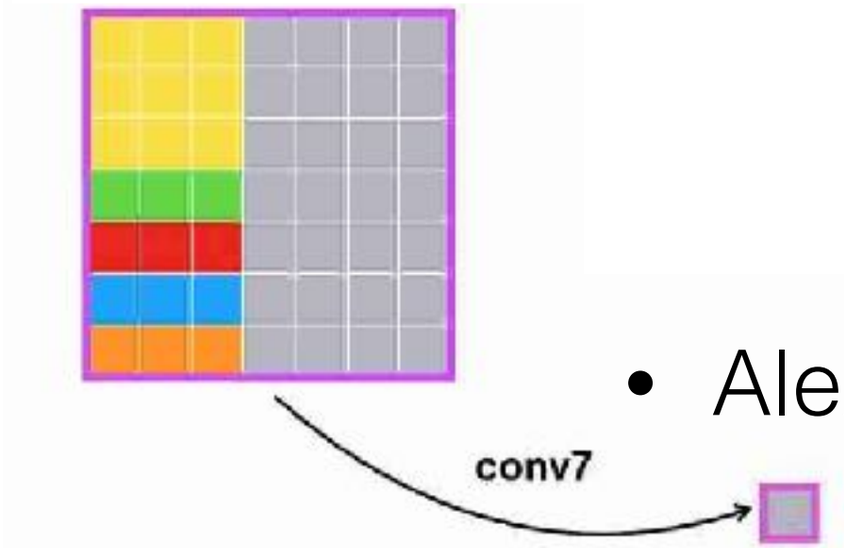
# VGGNet vs AlexNet


- 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

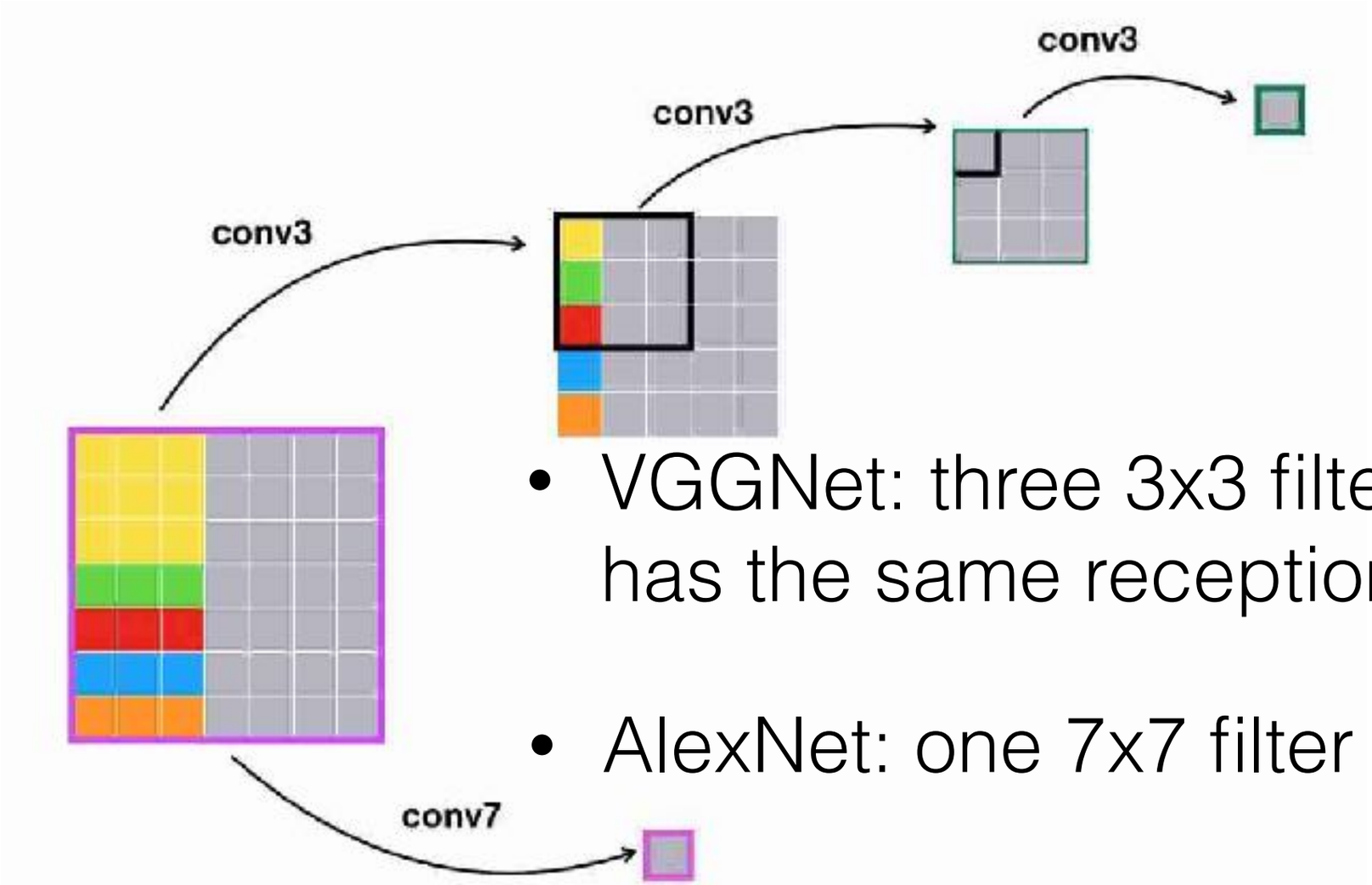
# VGGNet vs AlexNet




- 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

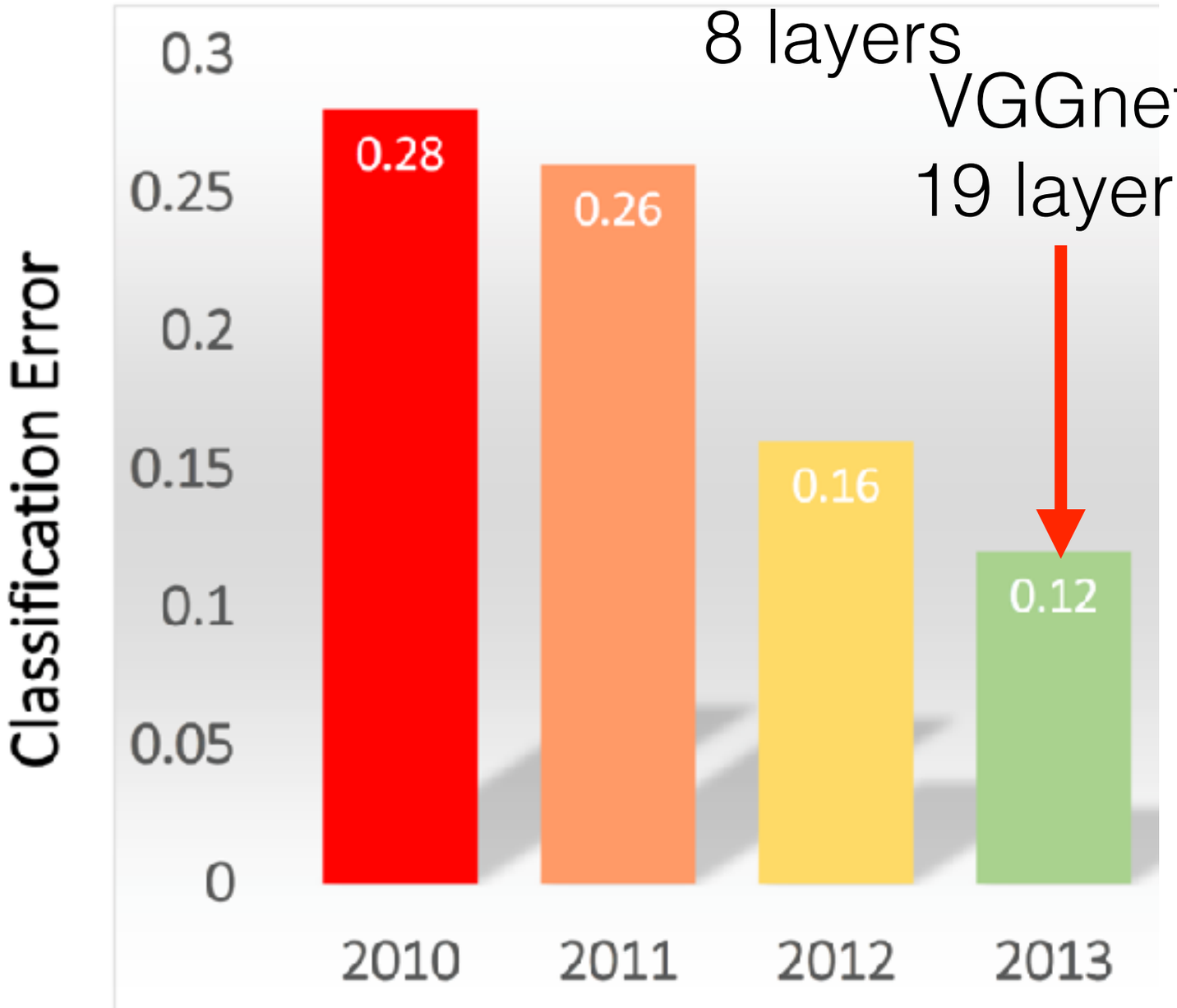



- VGGNet: three 3x3 filters (3x9+3 params) has the same reception filed
- AlexNet: one 7x7 filter (49+1 params)

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



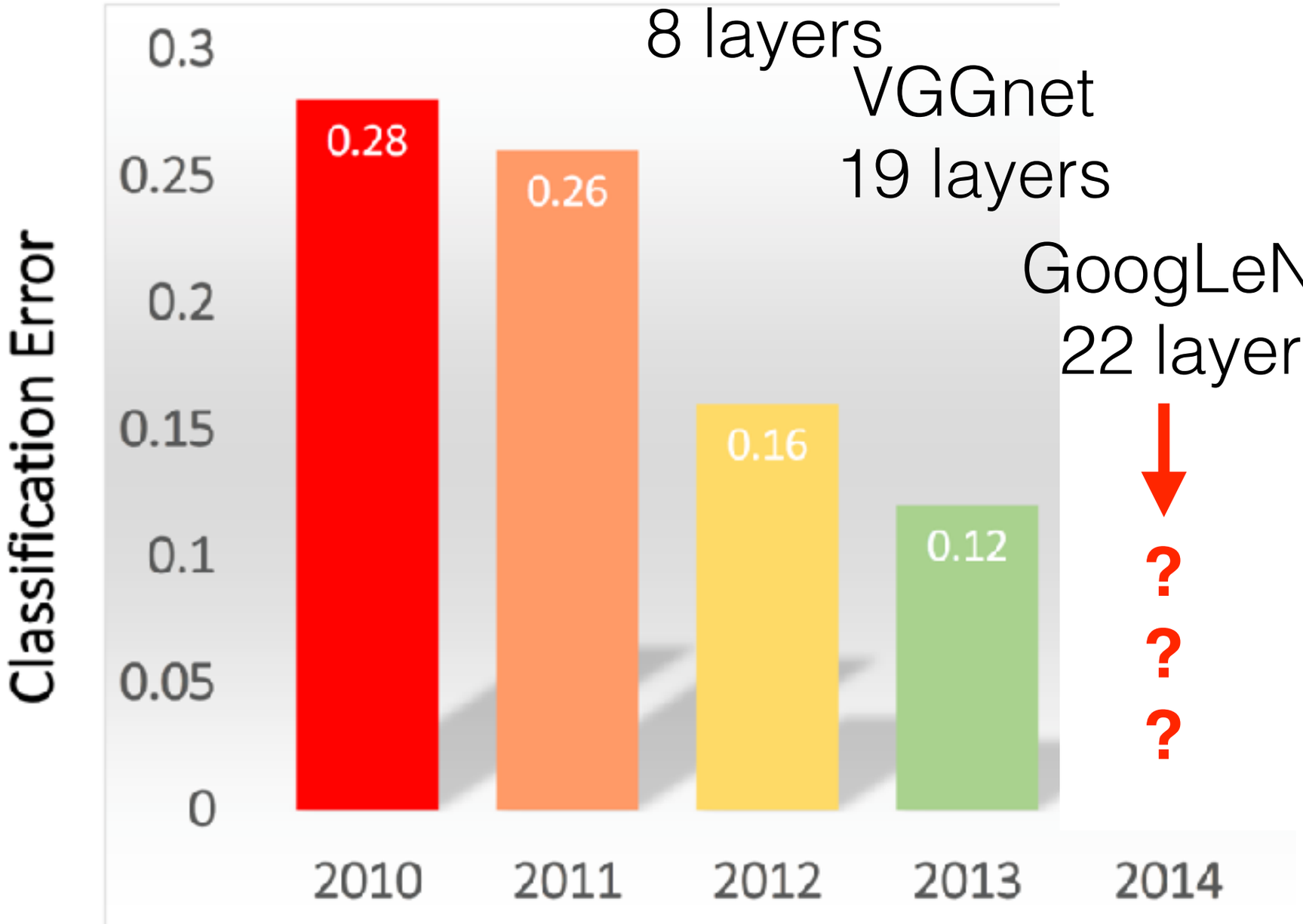
IMAGENET

# Classification results

AlexNet
8 layers

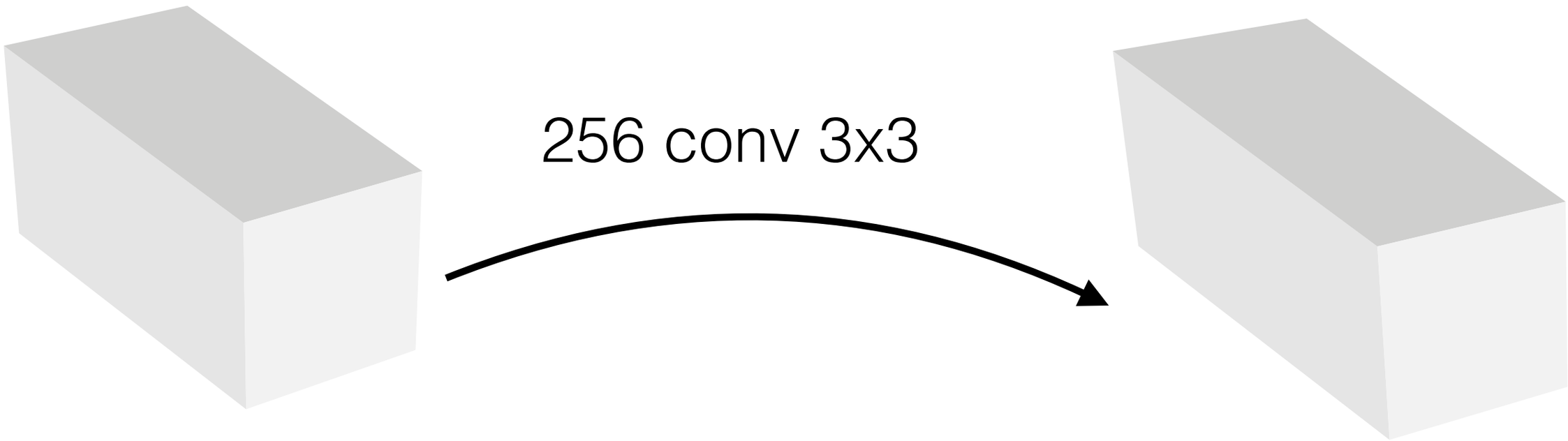
VGGnet
19 layers

GoogLeNet
22 layers



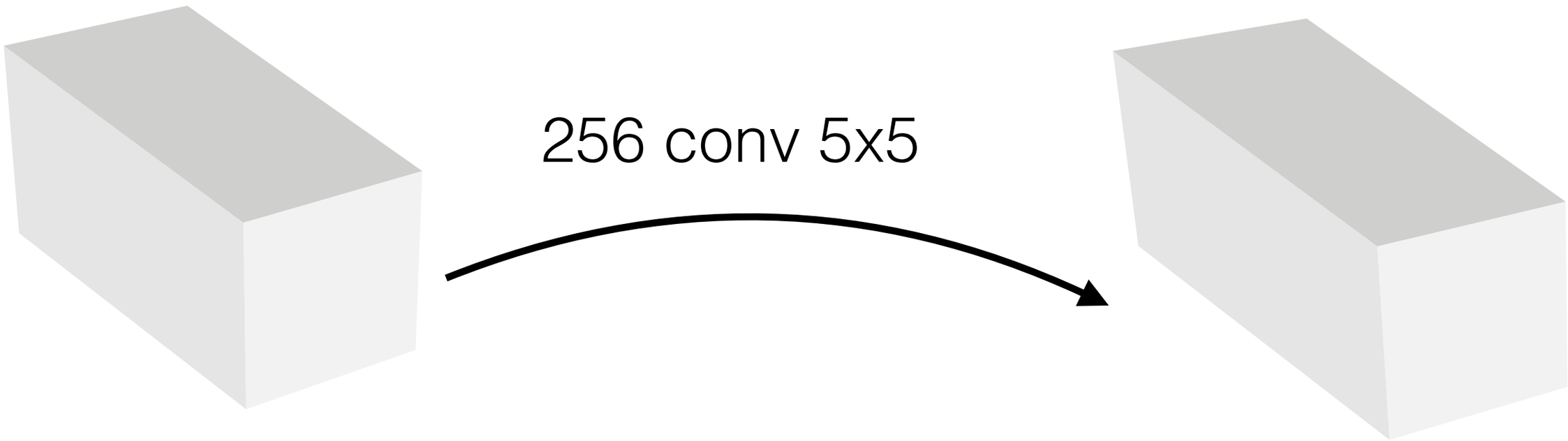

?
?
?

# GoogLeNet: concatenation of inception modules:



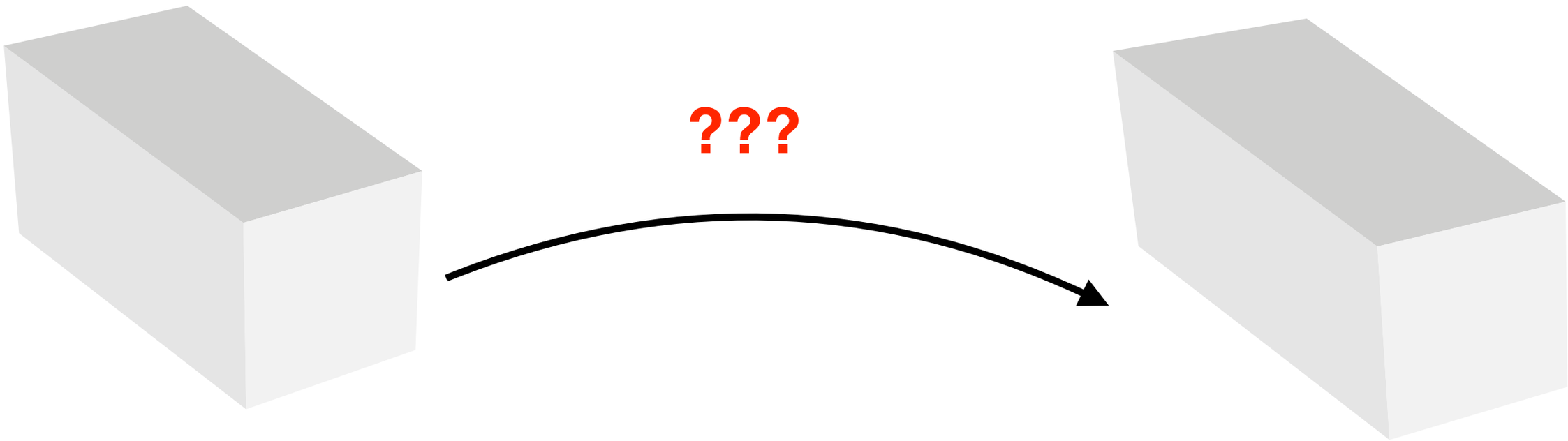

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

# GoogLeNet: concatenation of inception modules:




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

# GoogLeNet: concatenation of inception modules:

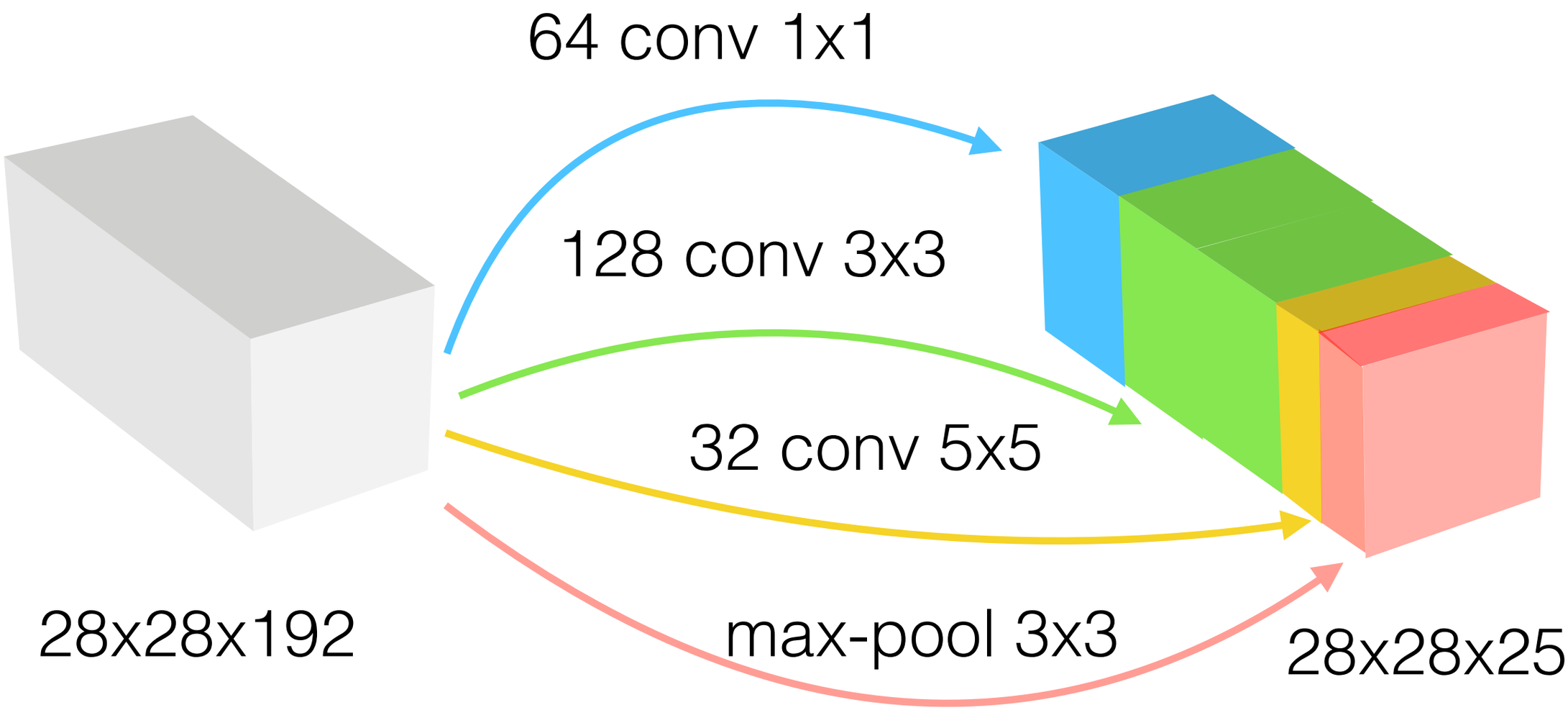



28x28x192

28x28x256

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

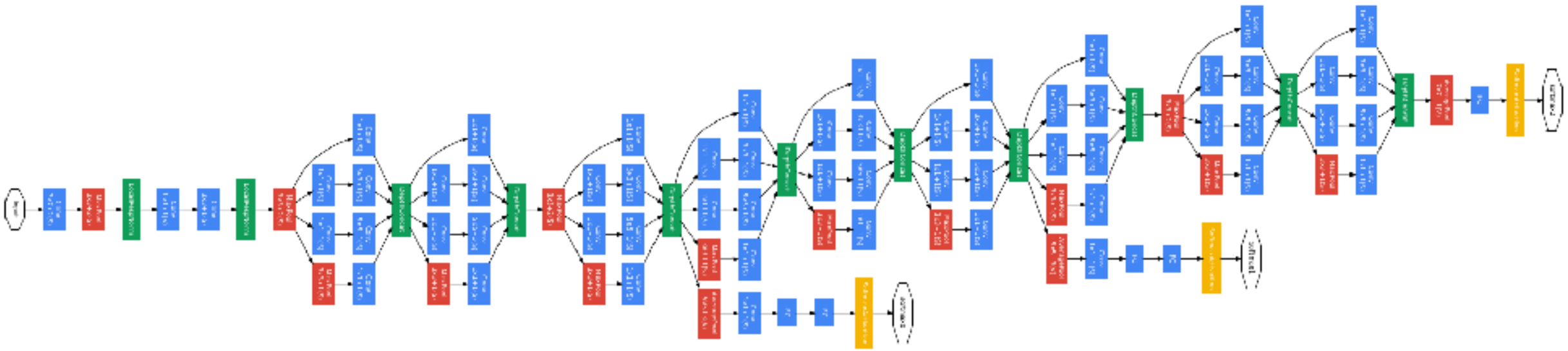
# GoogLeNet: concatenation of inception modules:




**Too many operations! => simplification using 1x1 conv**

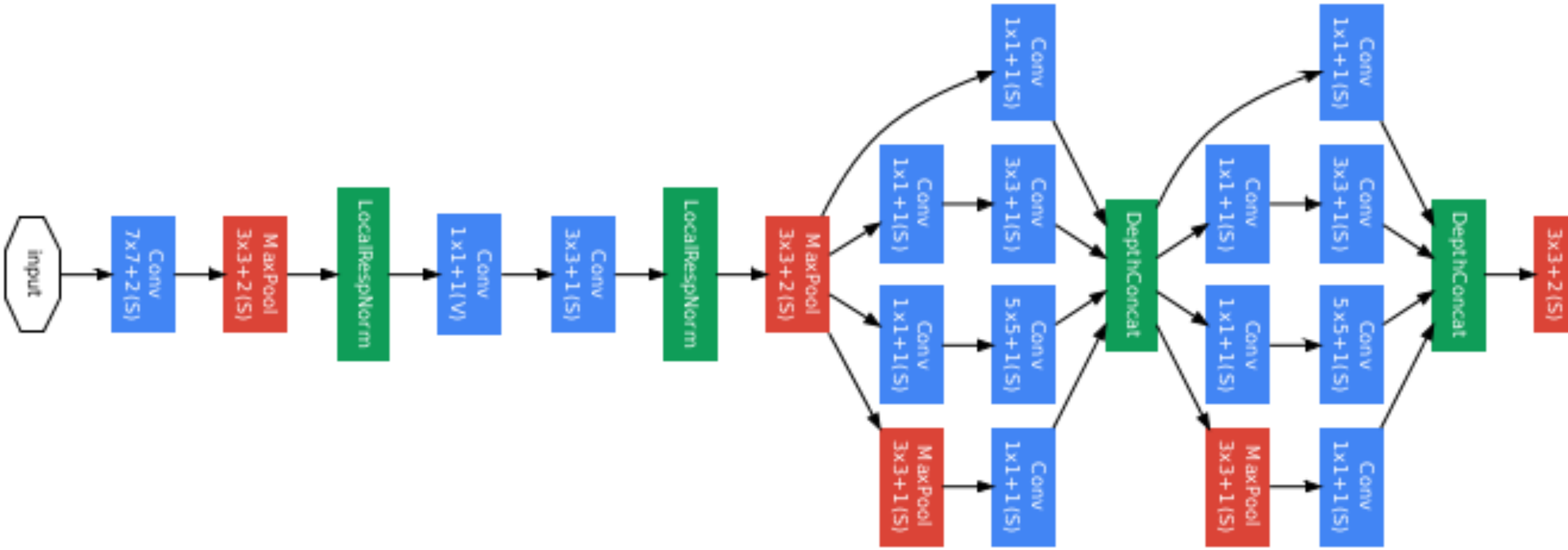
Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

# GoogLeNet



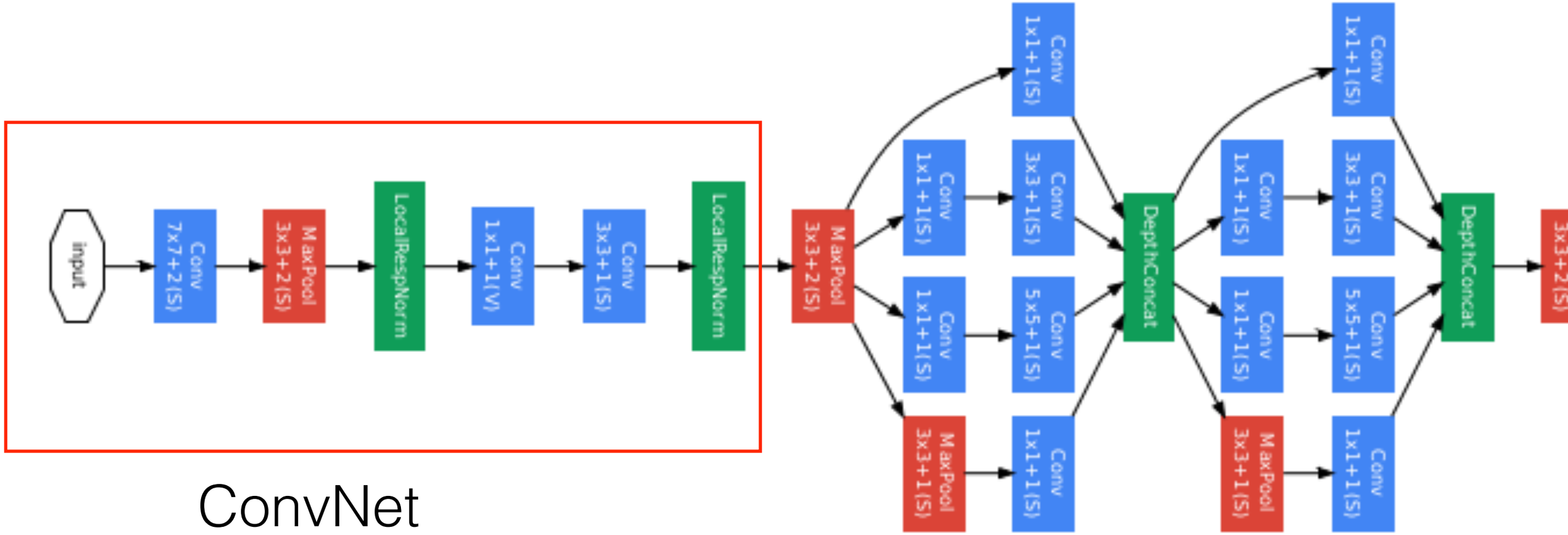
Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

# GoogLeNet




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

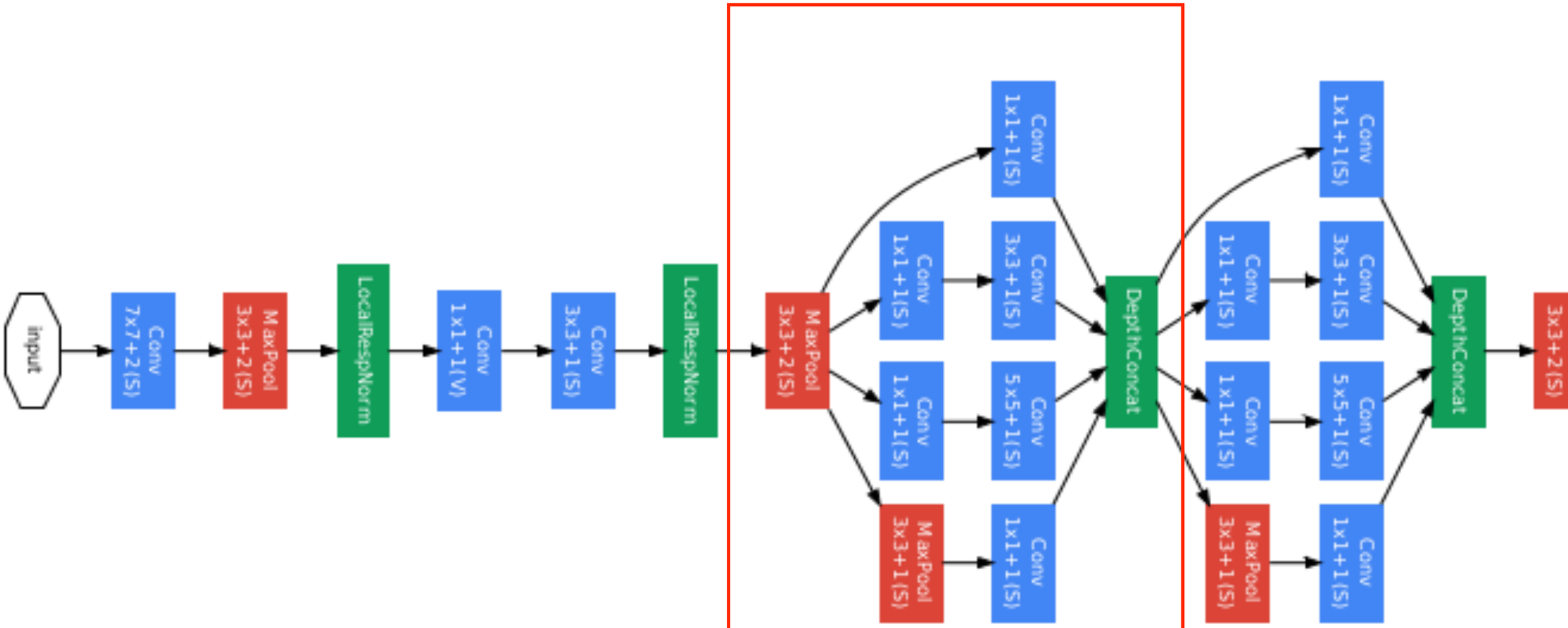
# GoogLeNet




ConvNet

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

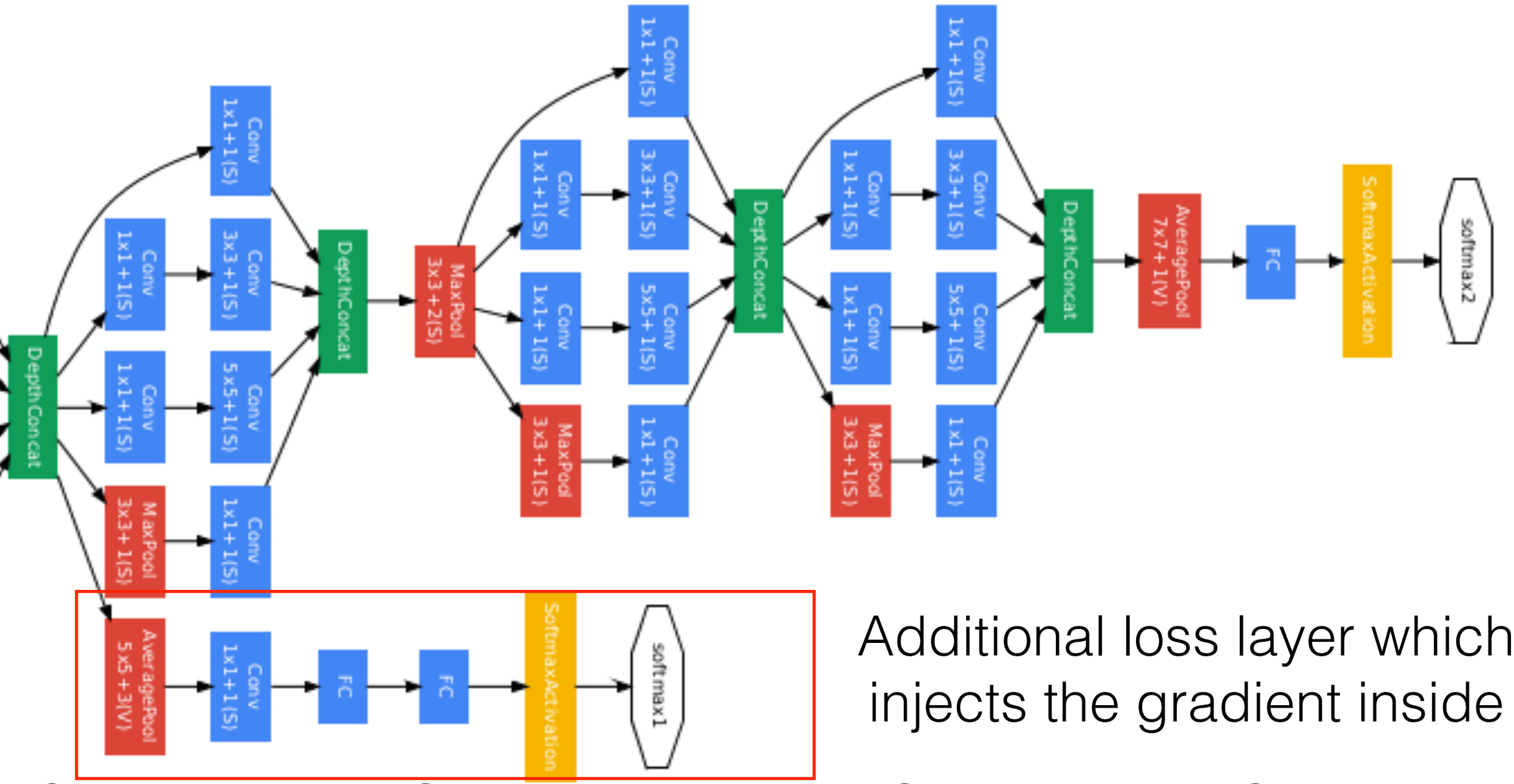
# GoogLeNet




Inception module

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

# GoogLeNet

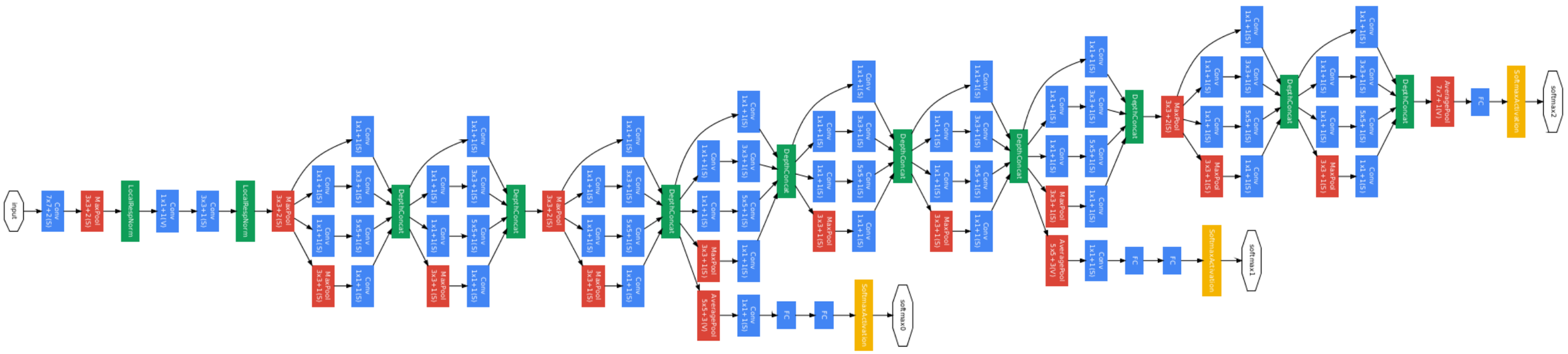



Additional loss layer which injects the gradient inside

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



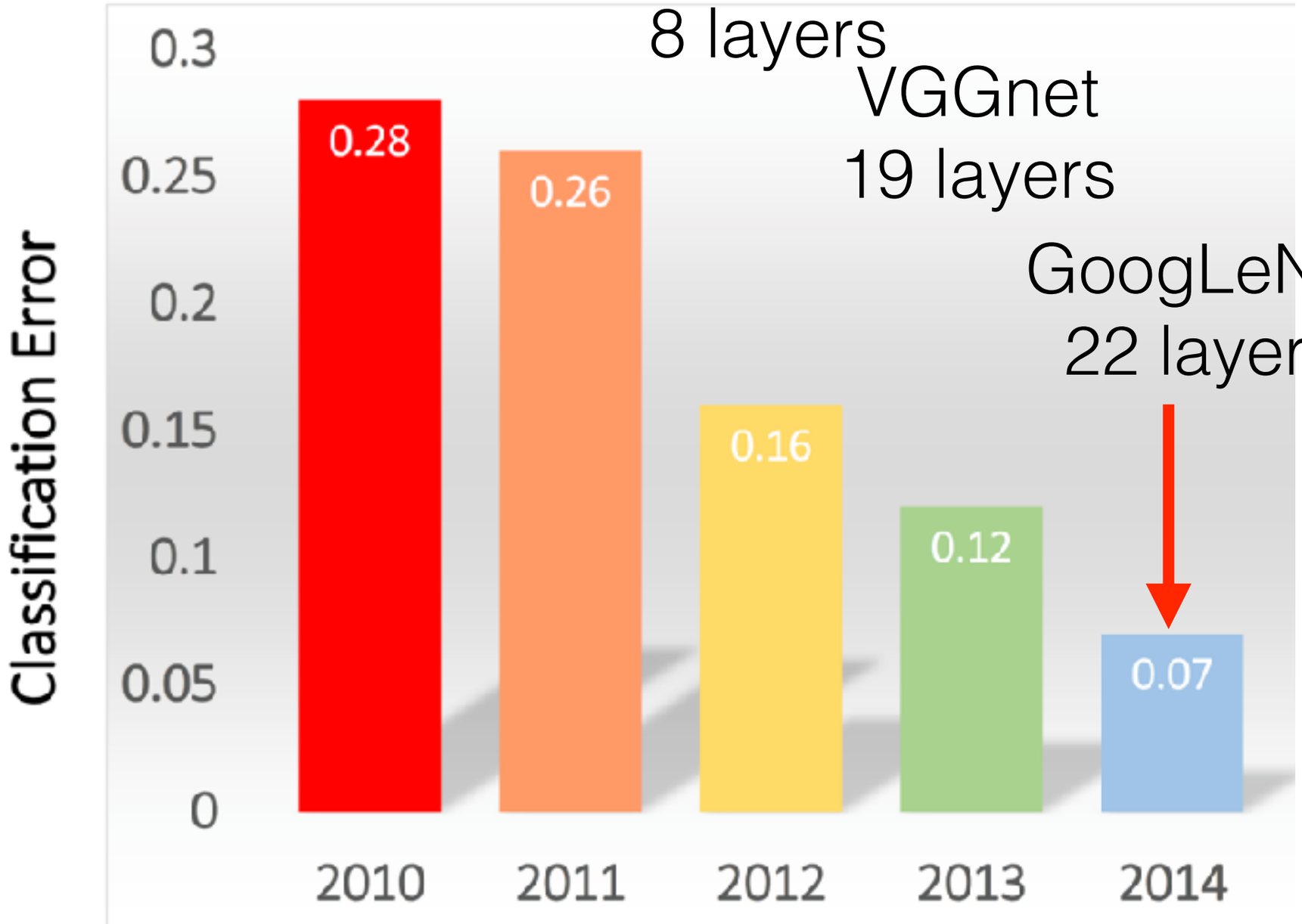

- 12x fewer parameters than AlexNet
- depth 22 layers
- training: few high-end GPU about a week

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

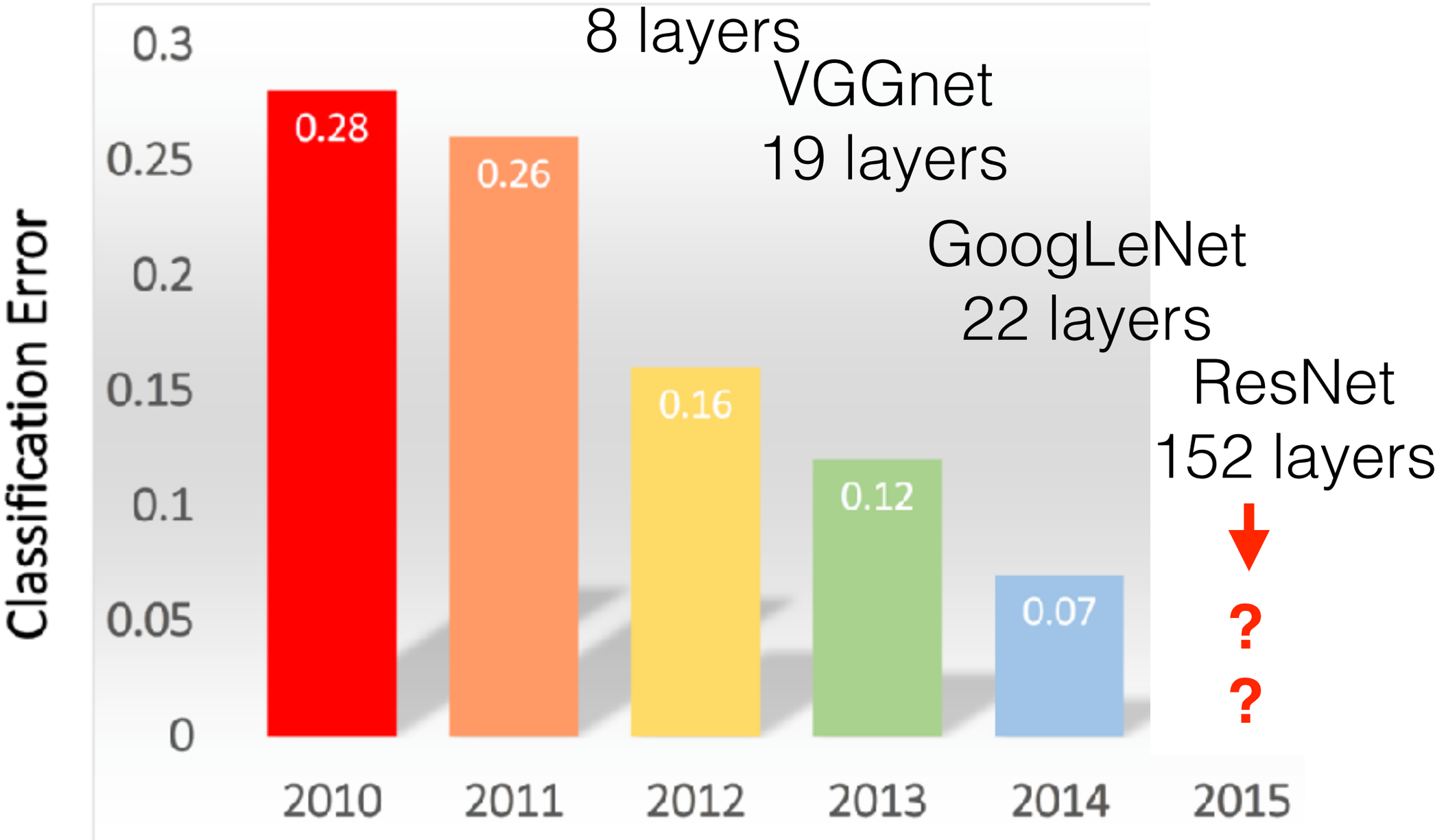
IMAGENET

# Classification results



IMAGENET

# Classification results




AlexNet
8 layers

VGGnet
19 layers

GoogLeNet
22 layers

ResNet
152 layers

?
?

**The main idea is as follows:**

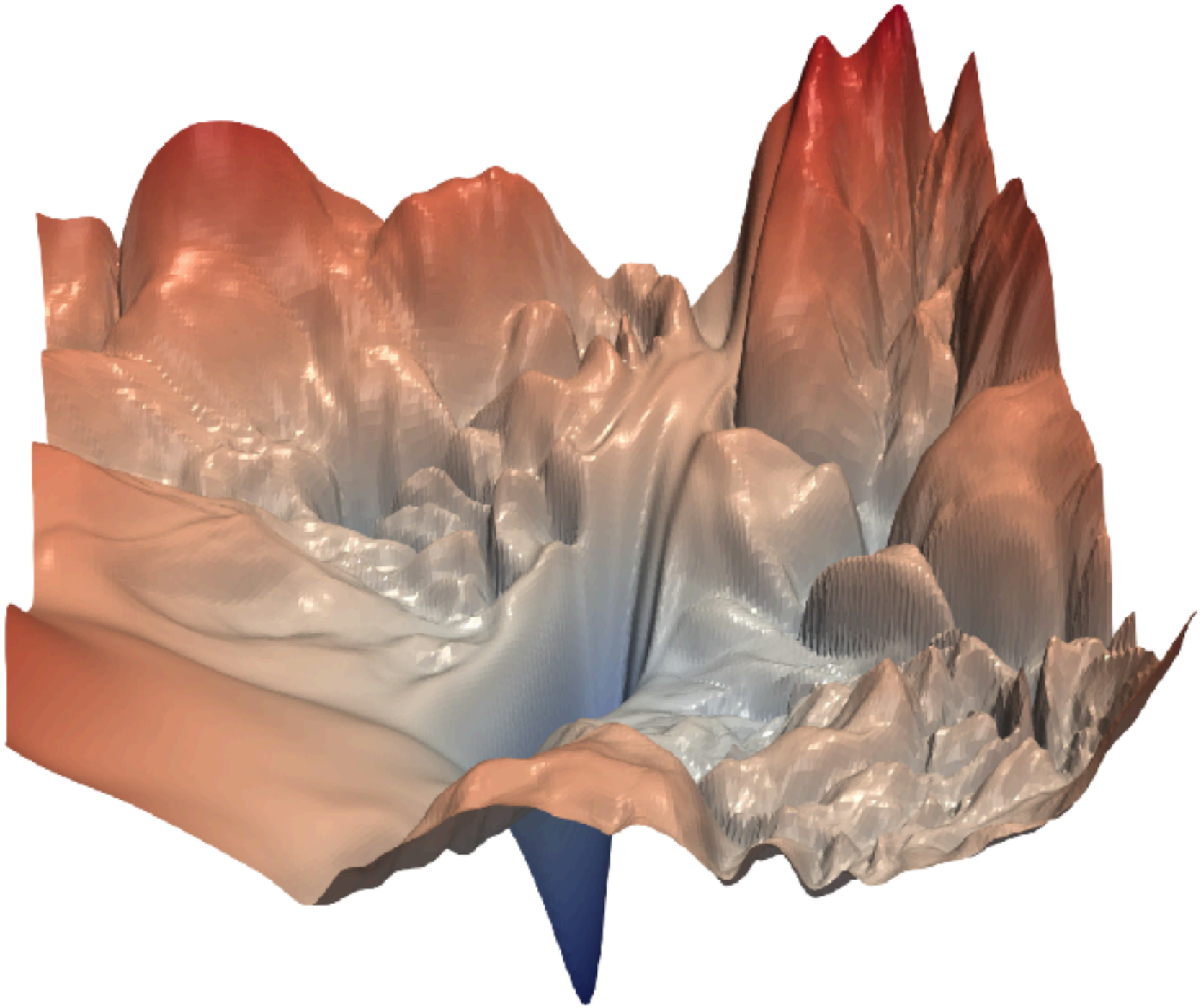

Well said Leo, well said

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

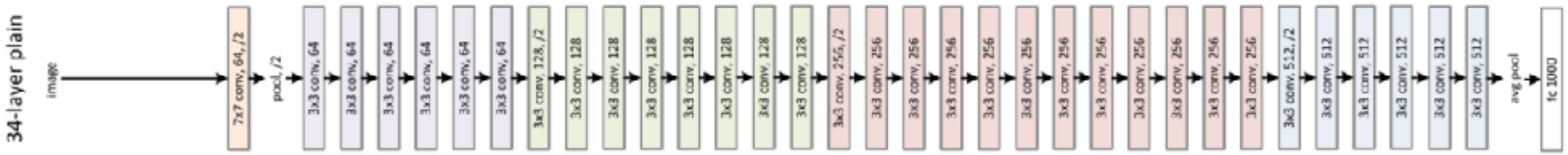
He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385

# Visualizing Loss Landscape of Neural Nets

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

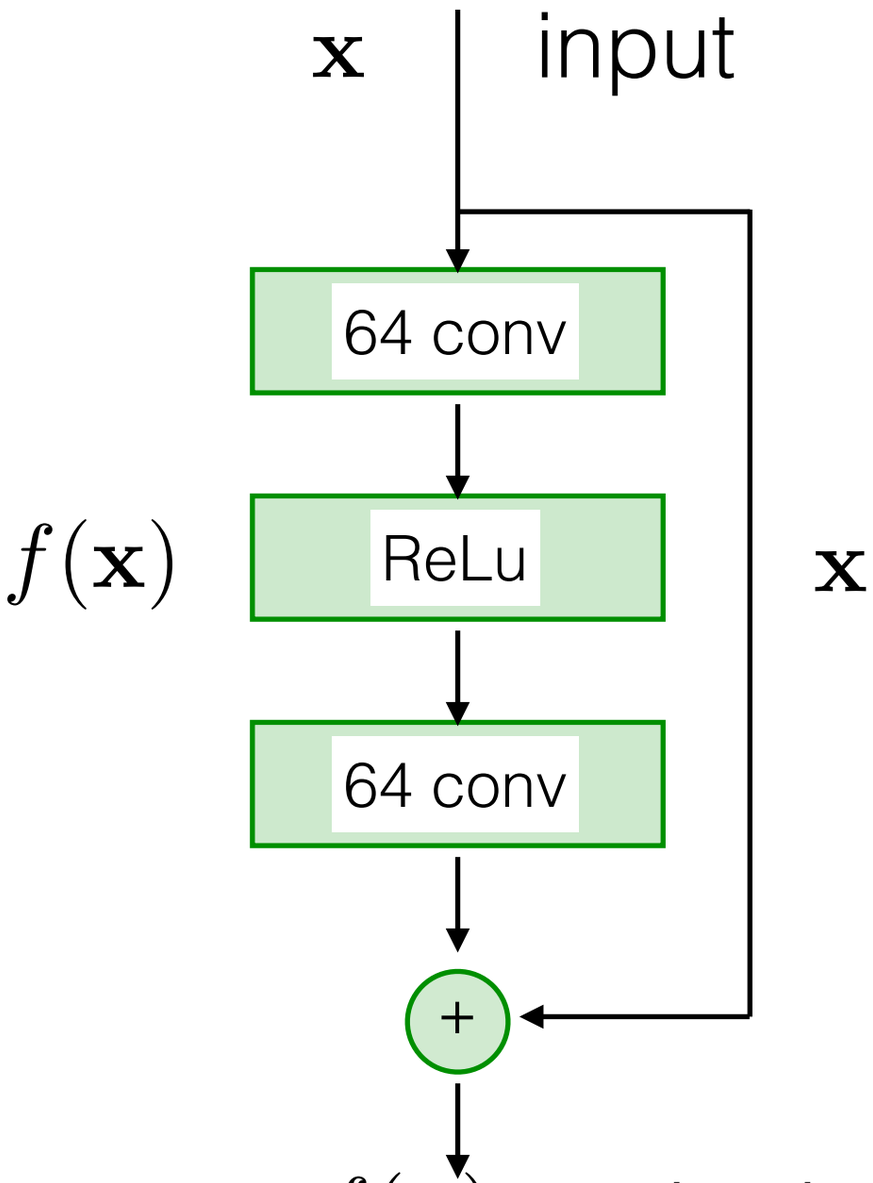


# ResNet



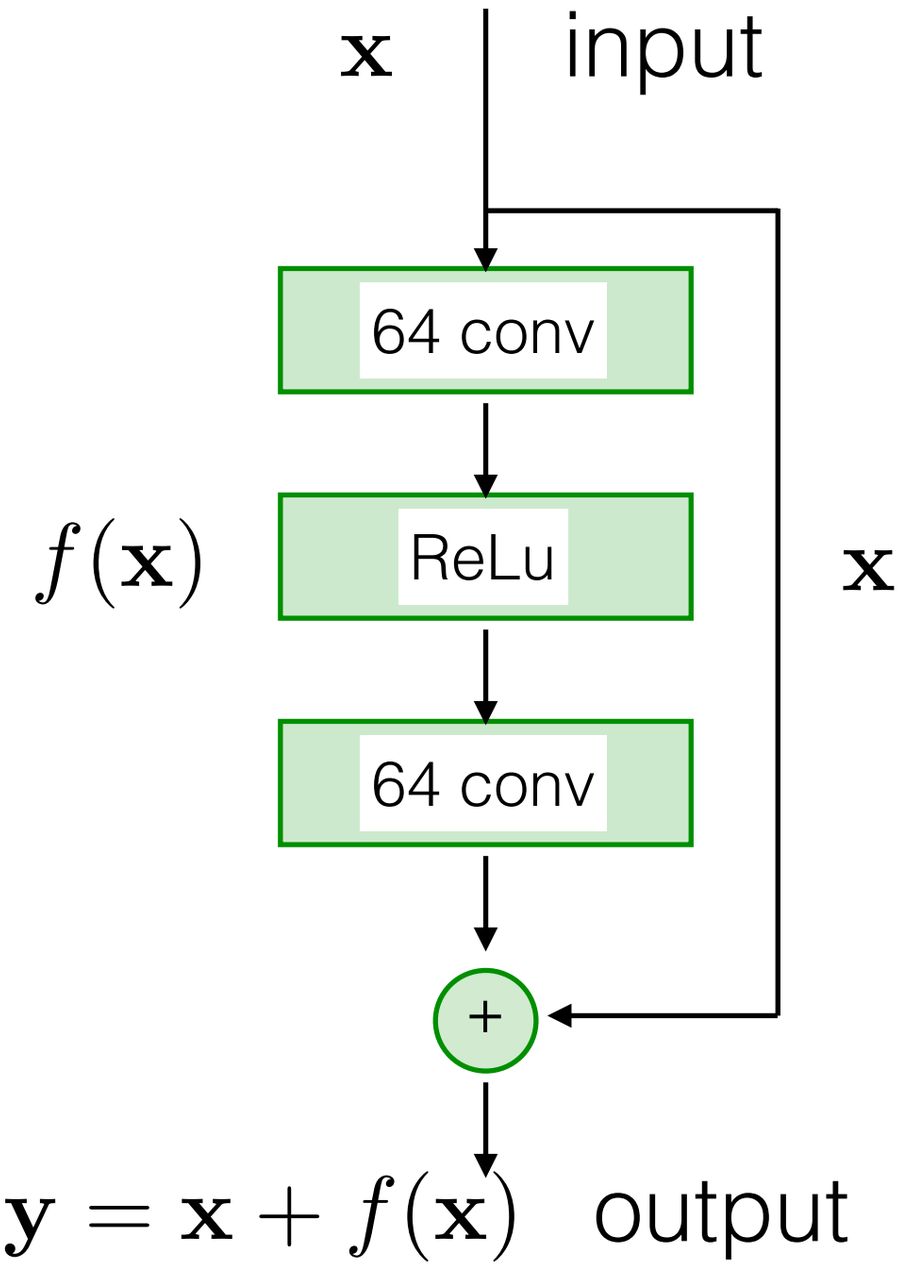

- Gradient in deep nets vanishes quickly
- In straightforward conv architecture the weights from the beginning of the net has minor influence on the output !!!
- In backward-pass the gradient of weights in the first layer is computed by multiplication of the all following gradients => prone to diminish!

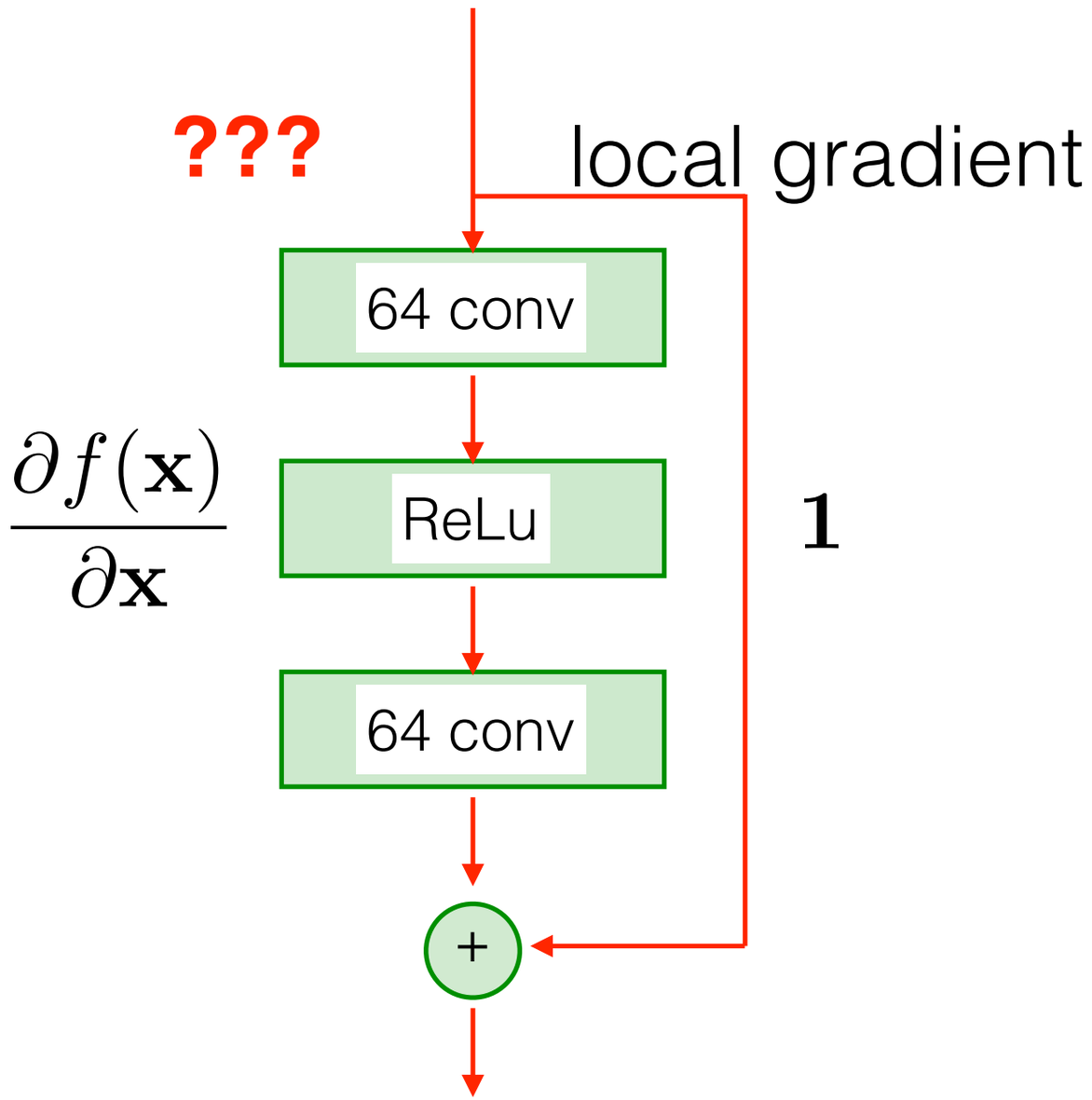
# ResNet: skip connections layer preserve gradient




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

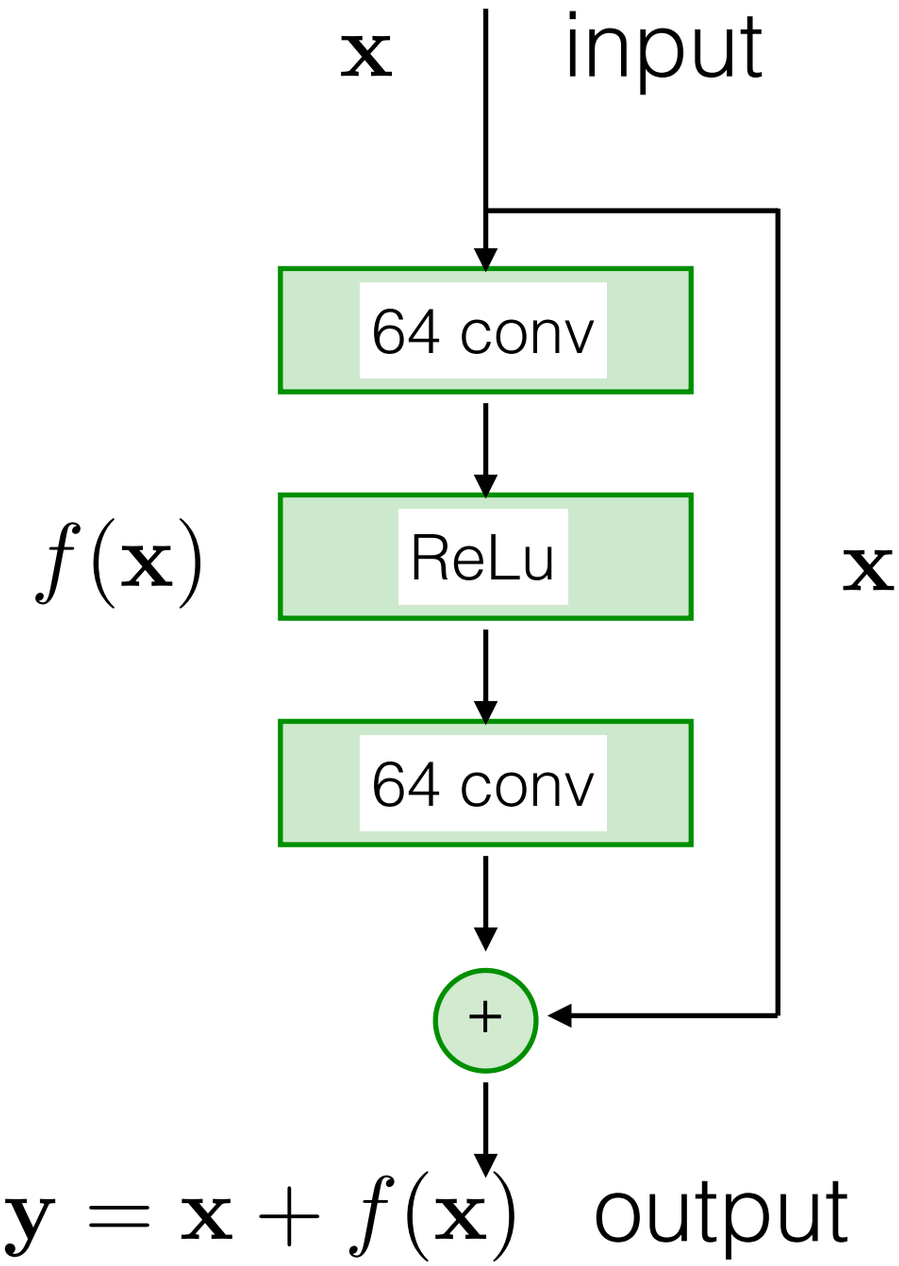
forward pass

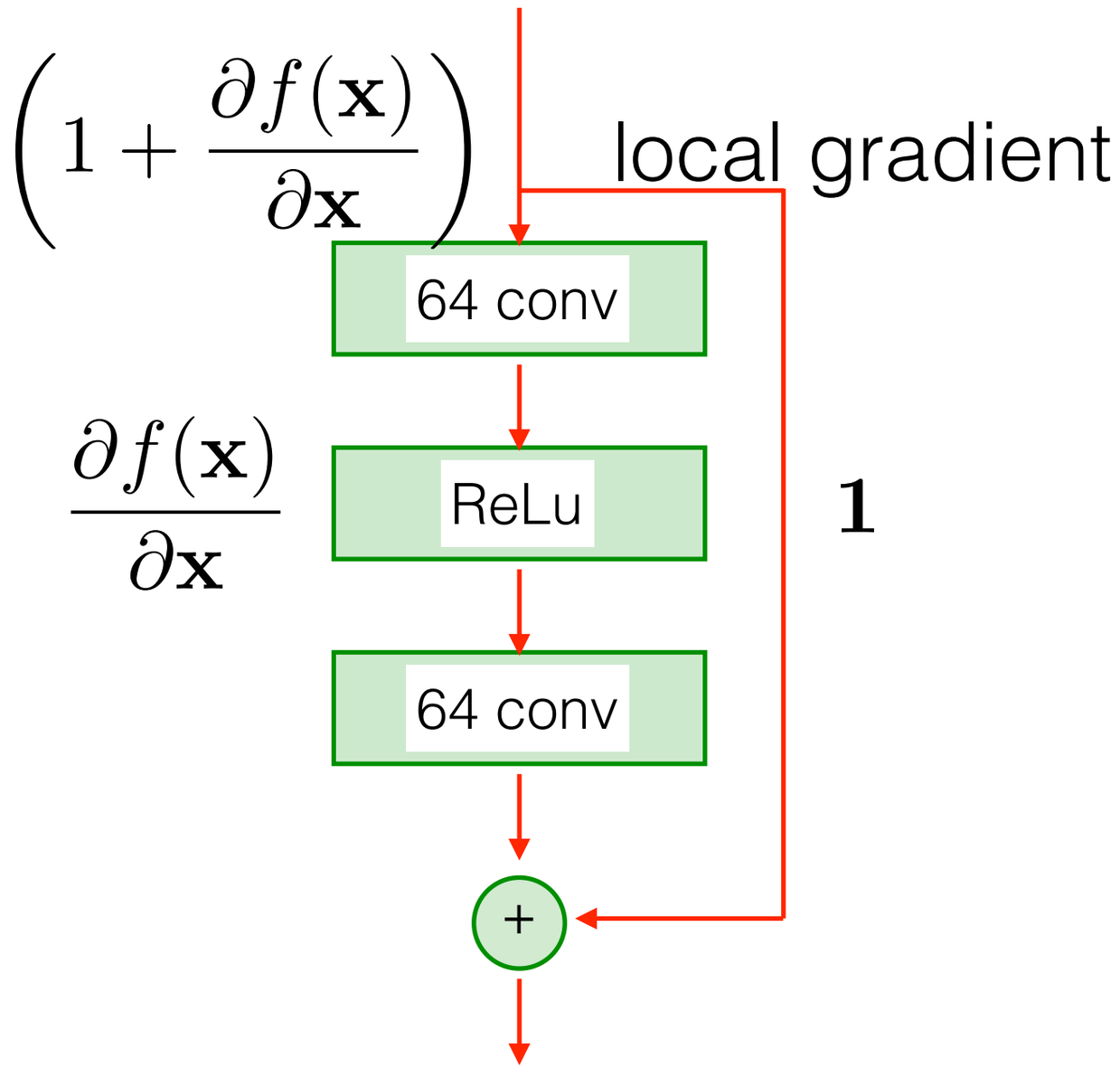



He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385

forward pass



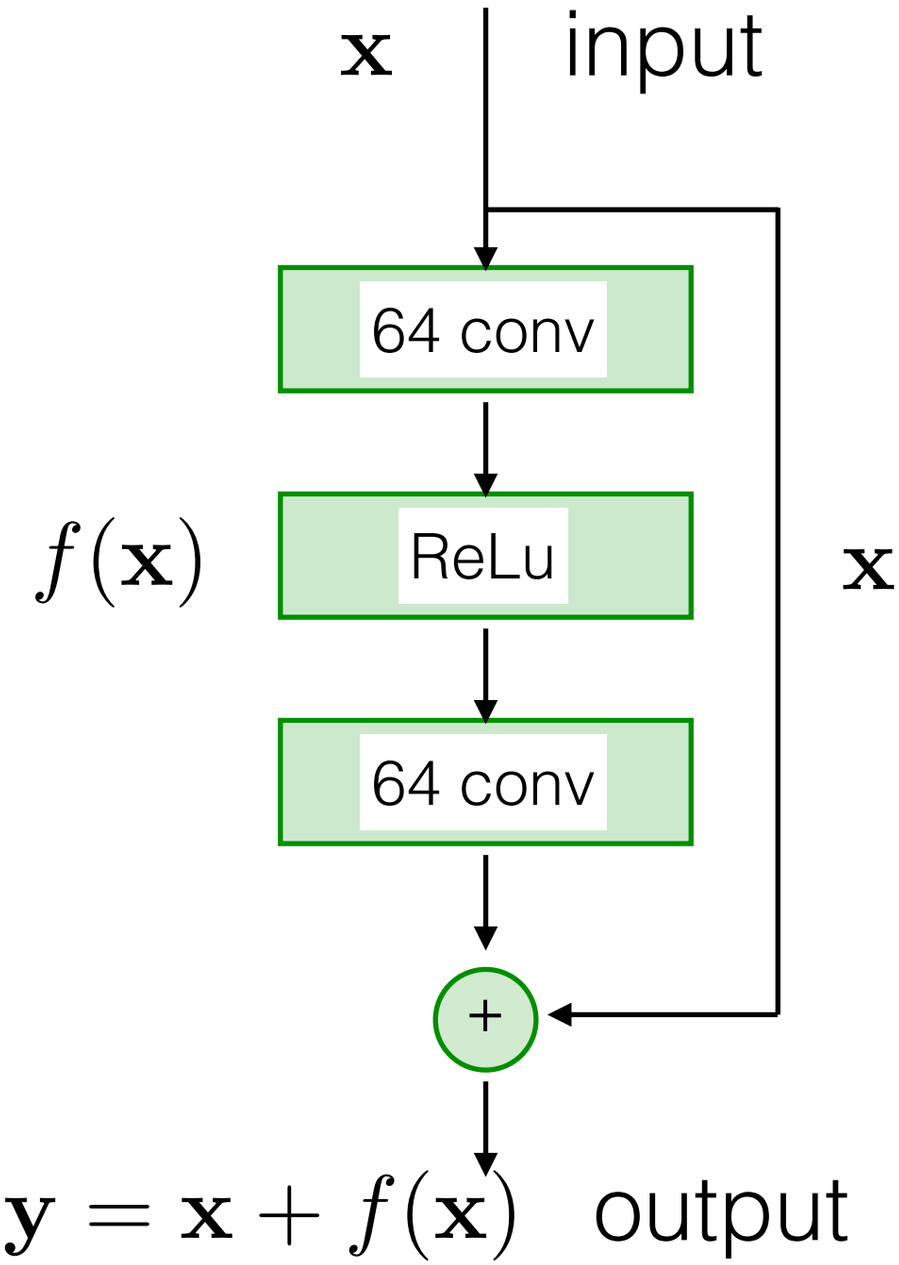



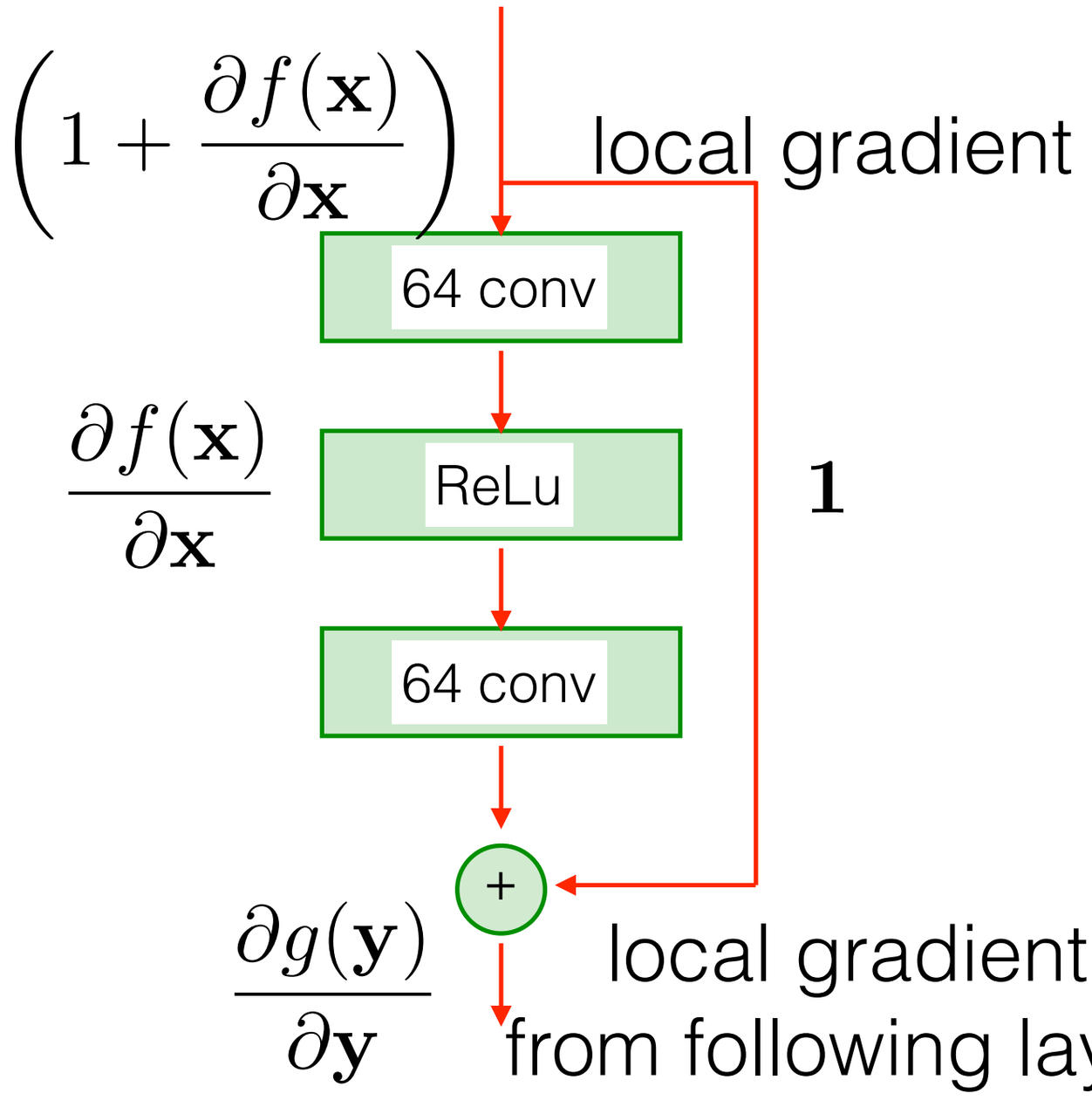

He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385

forward pass

backward pass

???



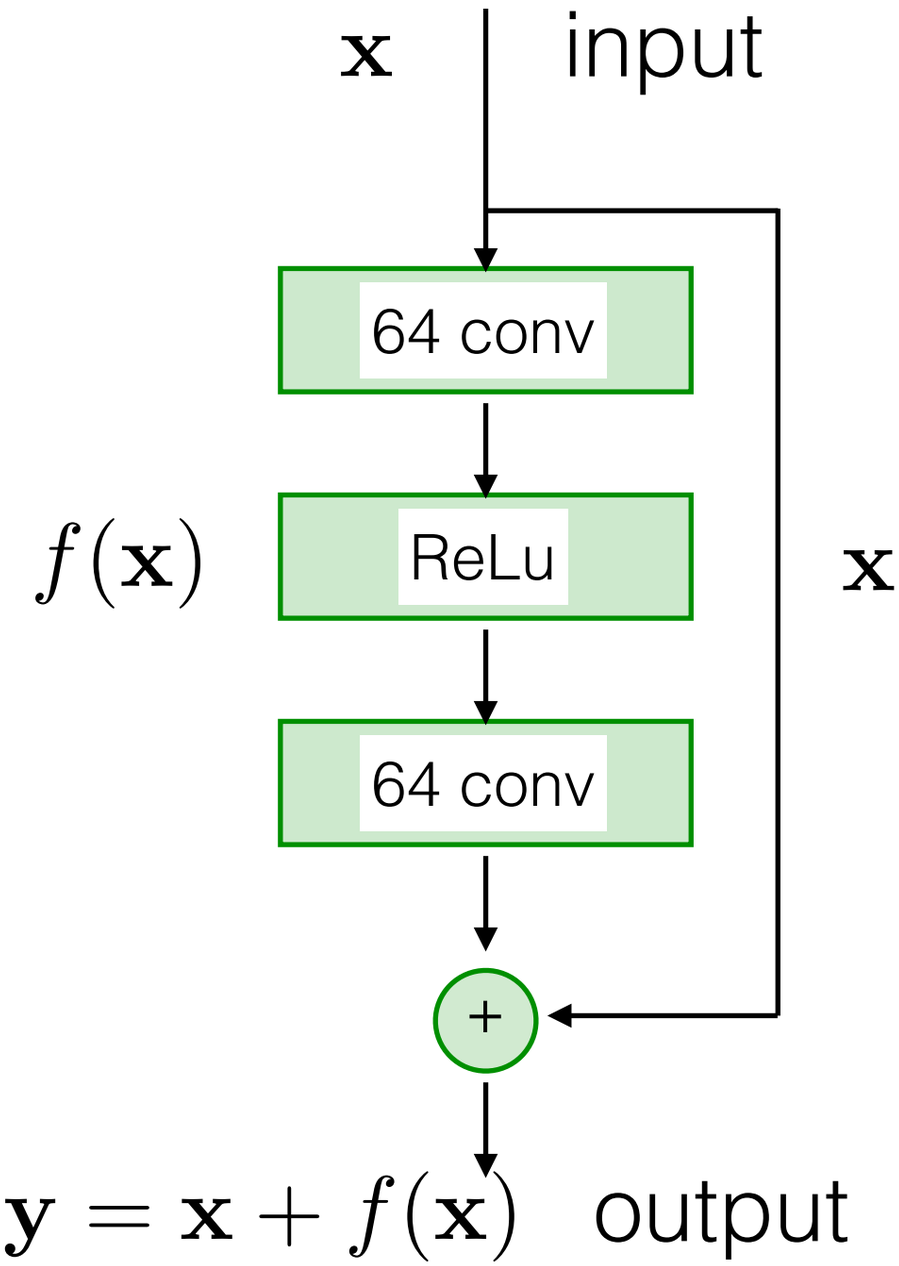




He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385

forward pass

backward pass

$\mathbf{x}$ input




$f(\mathbf{x})$

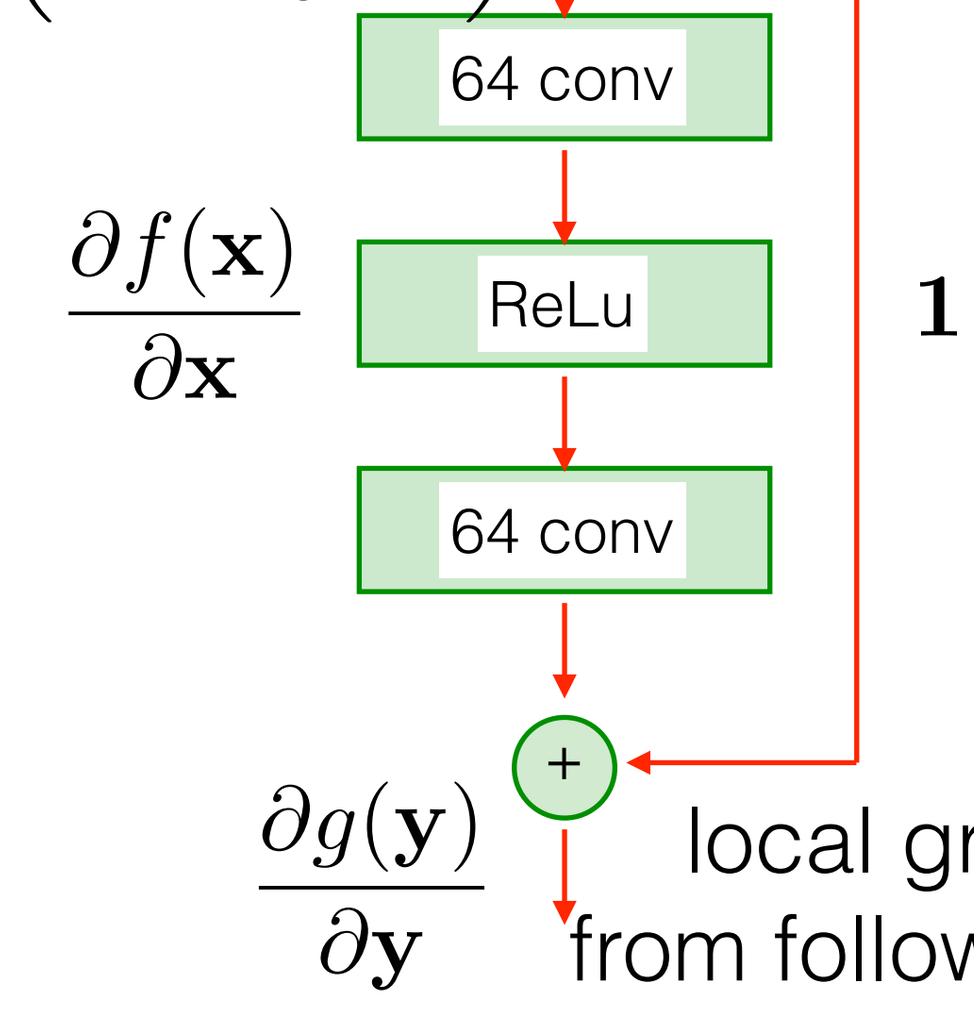
$\mathbf{x}$

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

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

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

local gradient



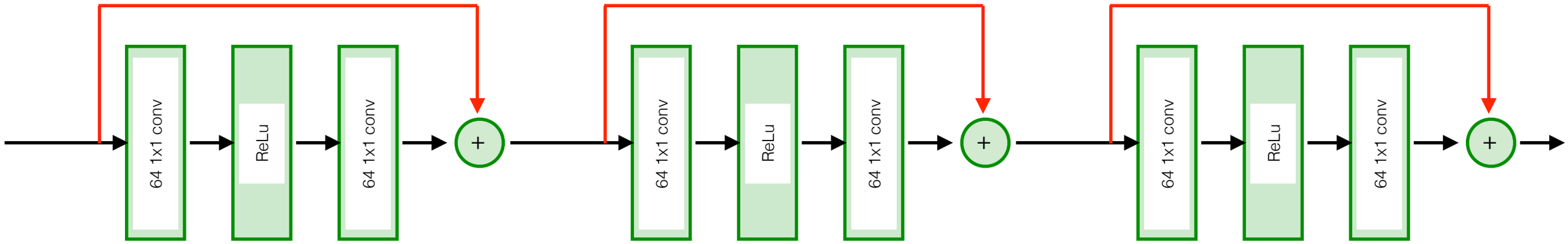

$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$

$\mathbf{1}$

$\frac{\partial g(\mathbf{y})}{\partial \mathbf{y}}$ local gradient from following layer

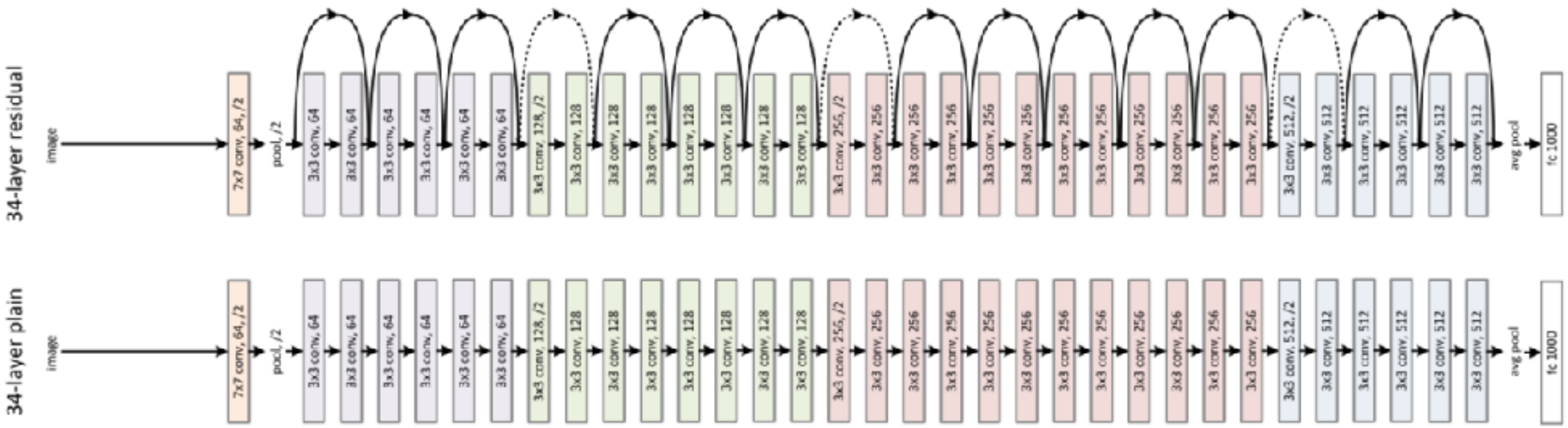
He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385

# 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



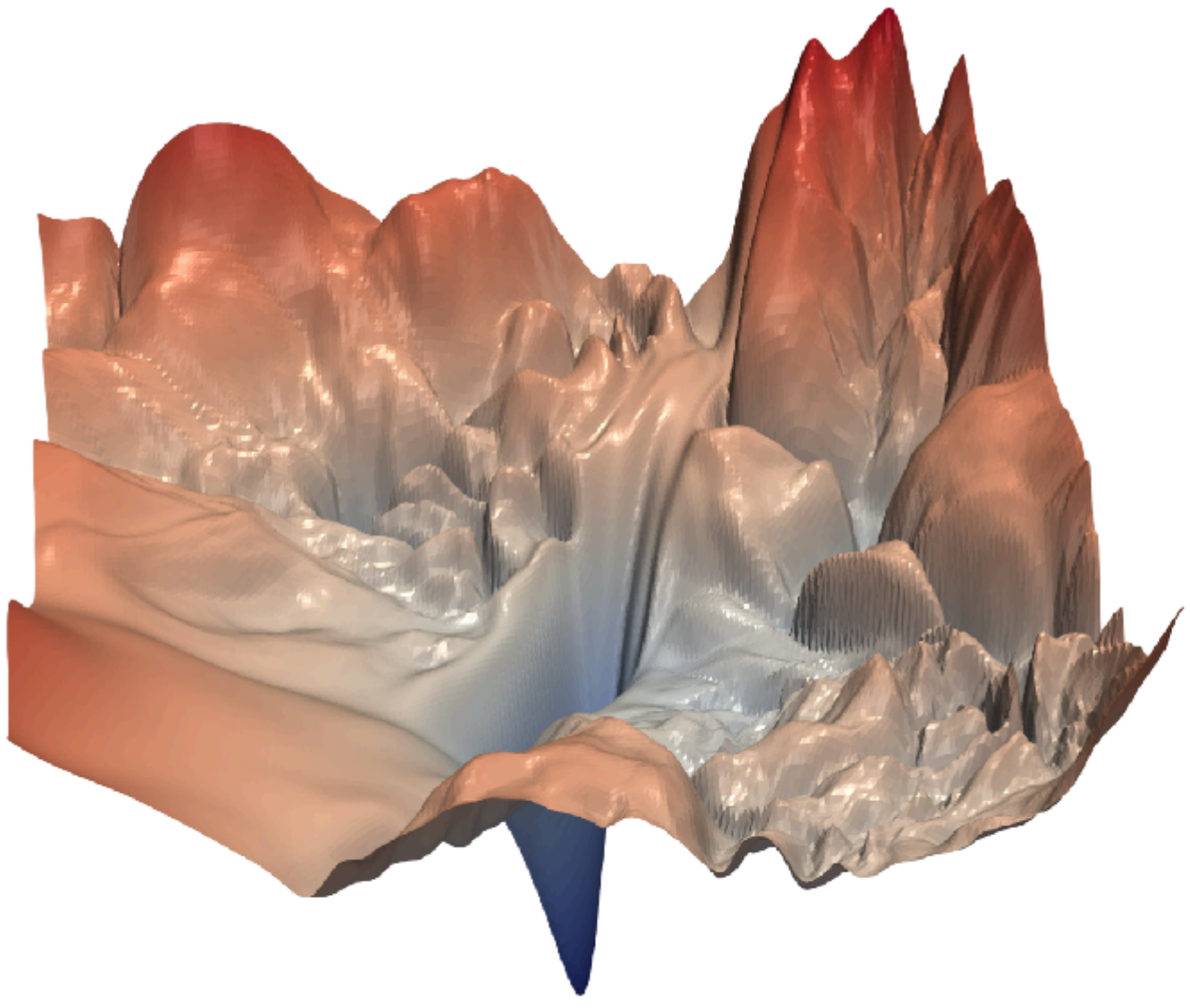

- Competition time about 152 layers ResNet,
- Recently they are able to train 1k layers ResNet
- Initialization with zero weights is meaningful
- Better gradient flow

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

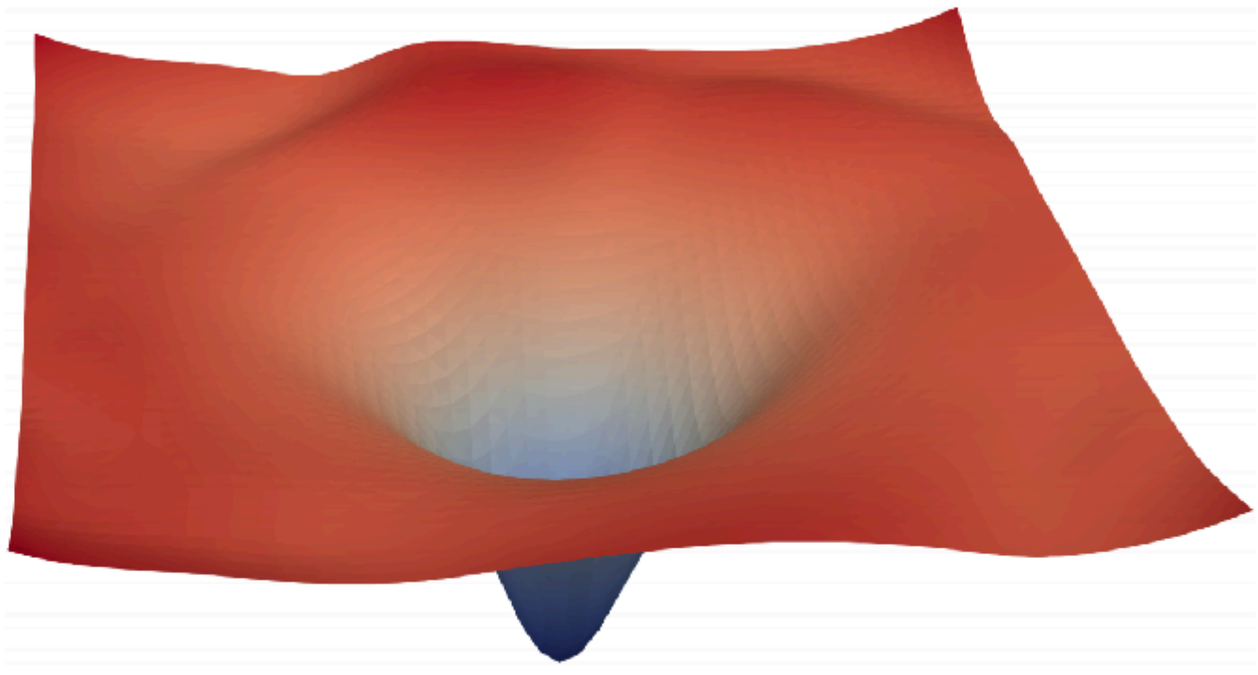
He et al. Going Deeper with Convolutions, CVPR, 2015
https://arxiv.org/abs/1512.03385

# Visualizing Loss Landscape of Neural Nets

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

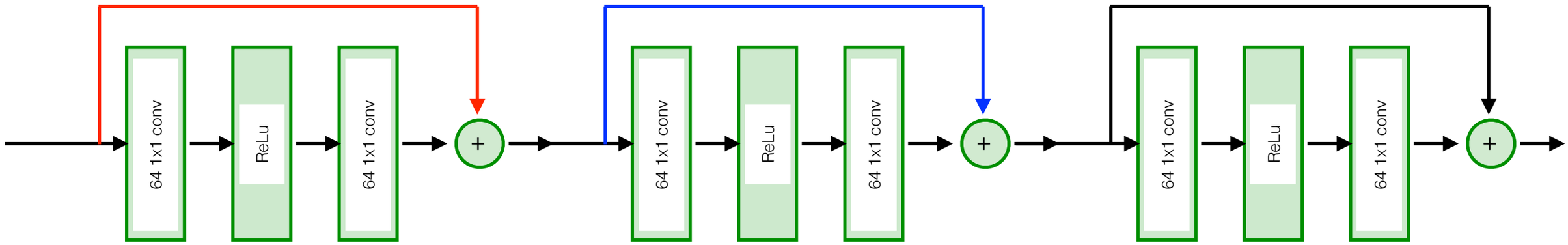


(a) without skip connections



(b) with skip connections

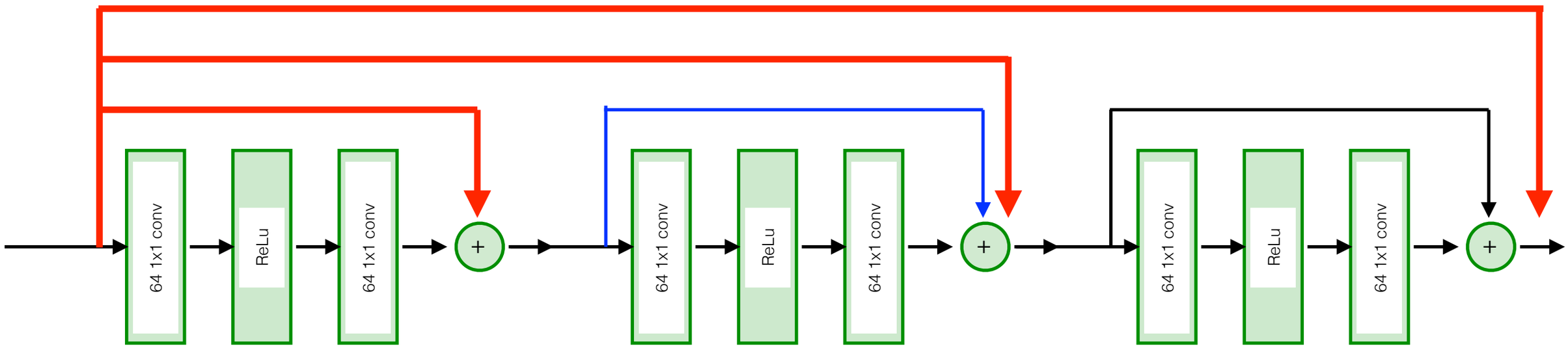
# ResNet =>DenseNet




Start with multilayer ResNet architecture

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

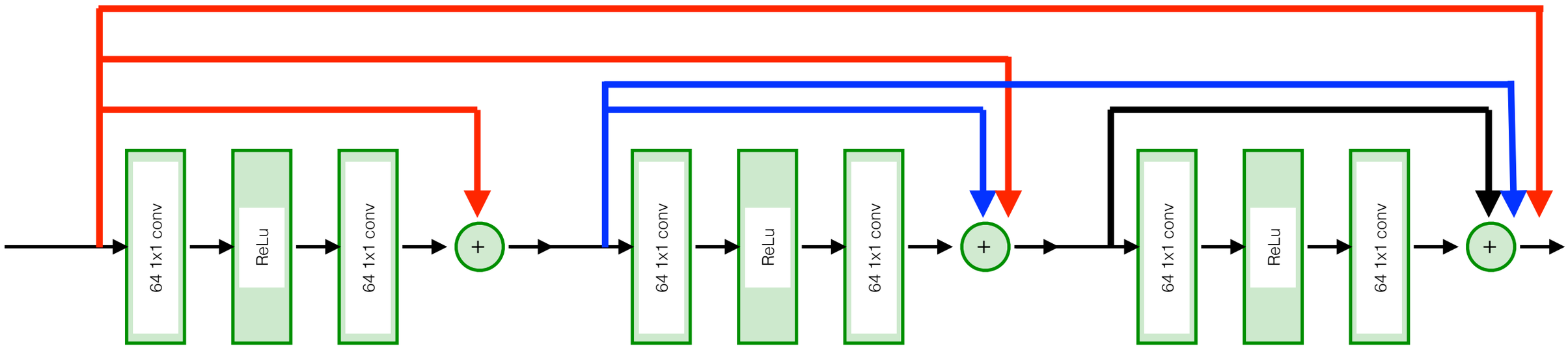
# DenseNet




- Directly propagate each feature map to all following layers

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

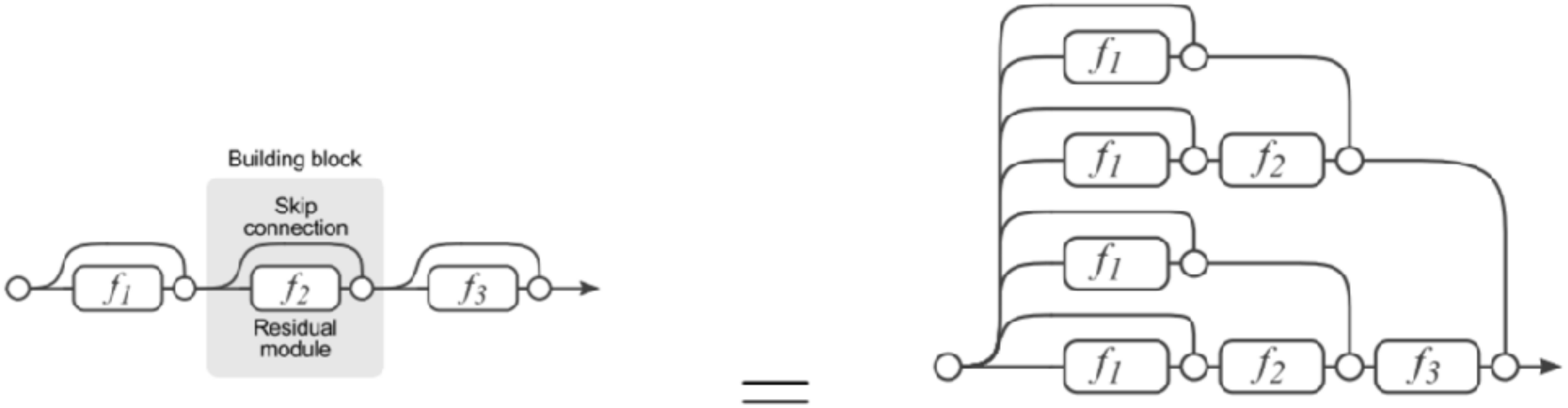
# DenseNet




- 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

# DenseNet



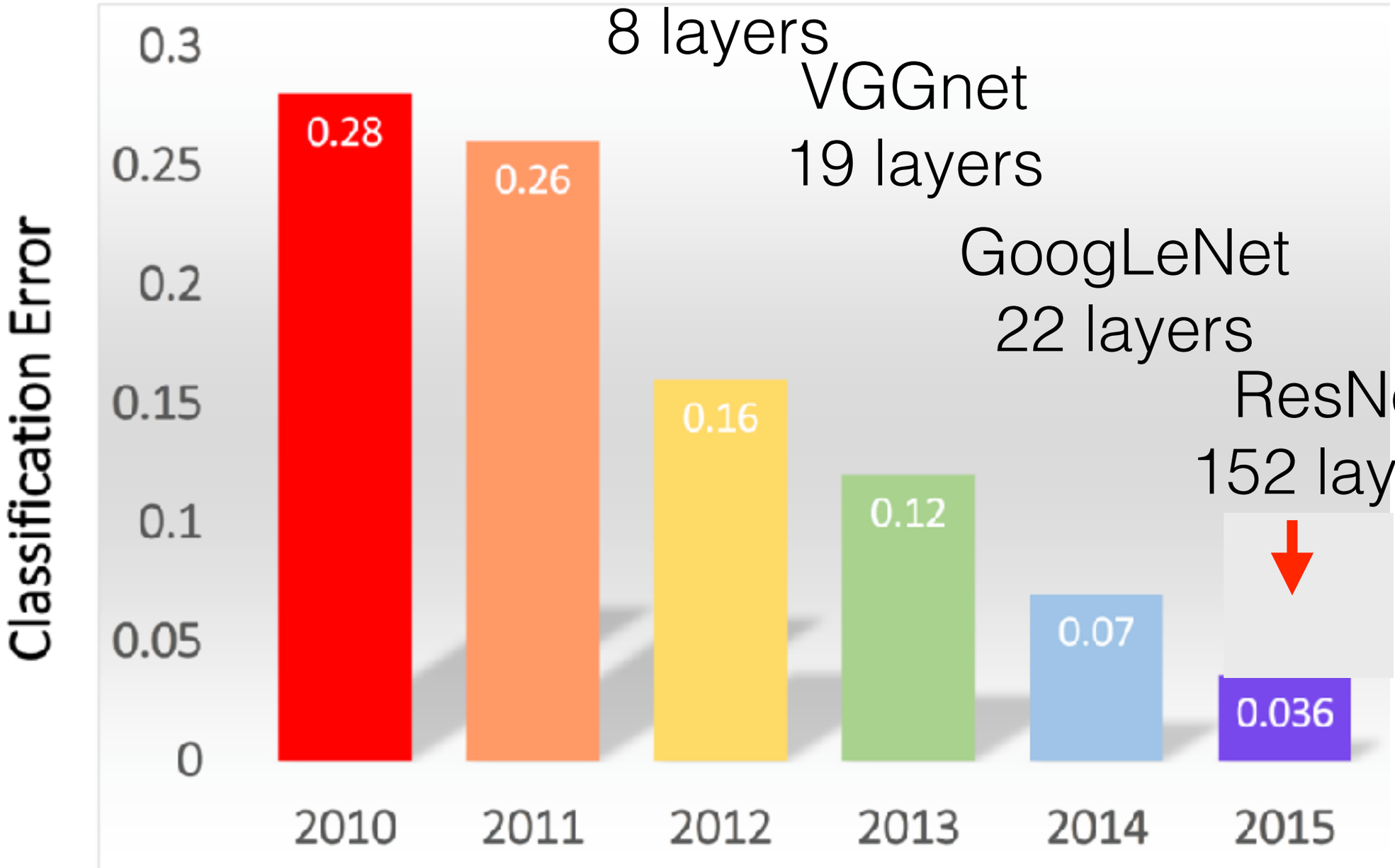

(a) Conventional 3-block residual network

(b) Unraveled view of (a)

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

IMAGENET

# Classification results

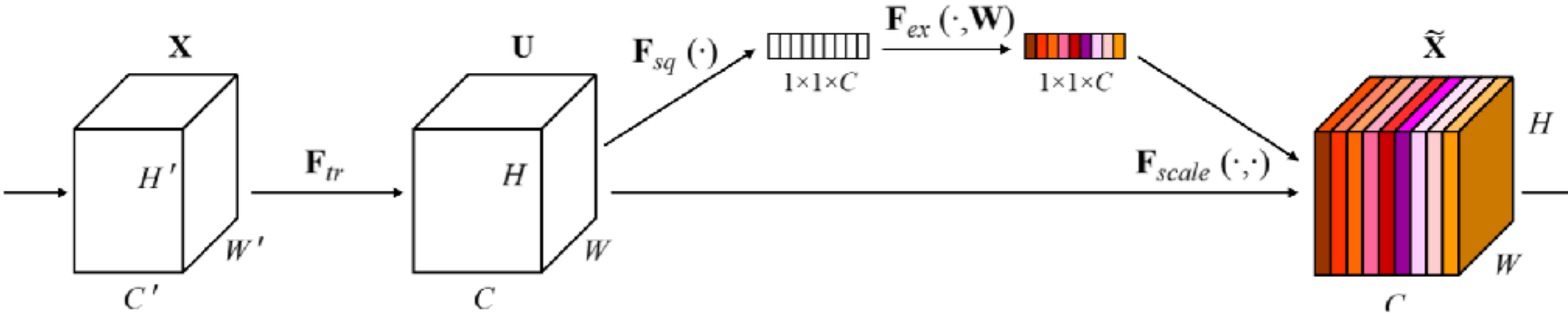



**Human error around 5%**

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

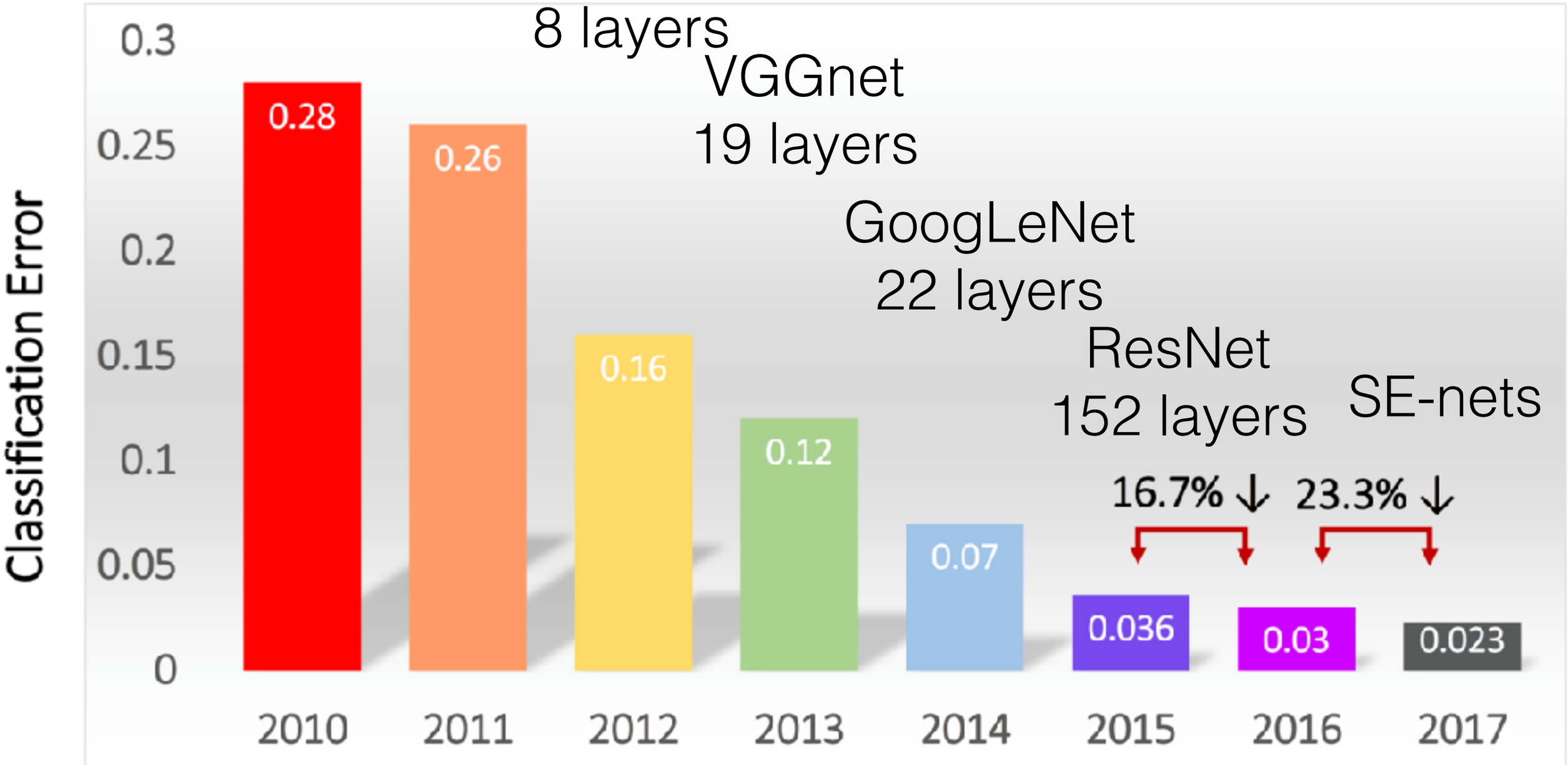
- Winner of ILSVRC 2017
- Enhancement of ResNet, InceptionNet and DenseNet architectures by SE blocks consistently decrease error on ImageNet, COCO, …

# Squeeze and Excitation block



IMAGENET

# Classification results



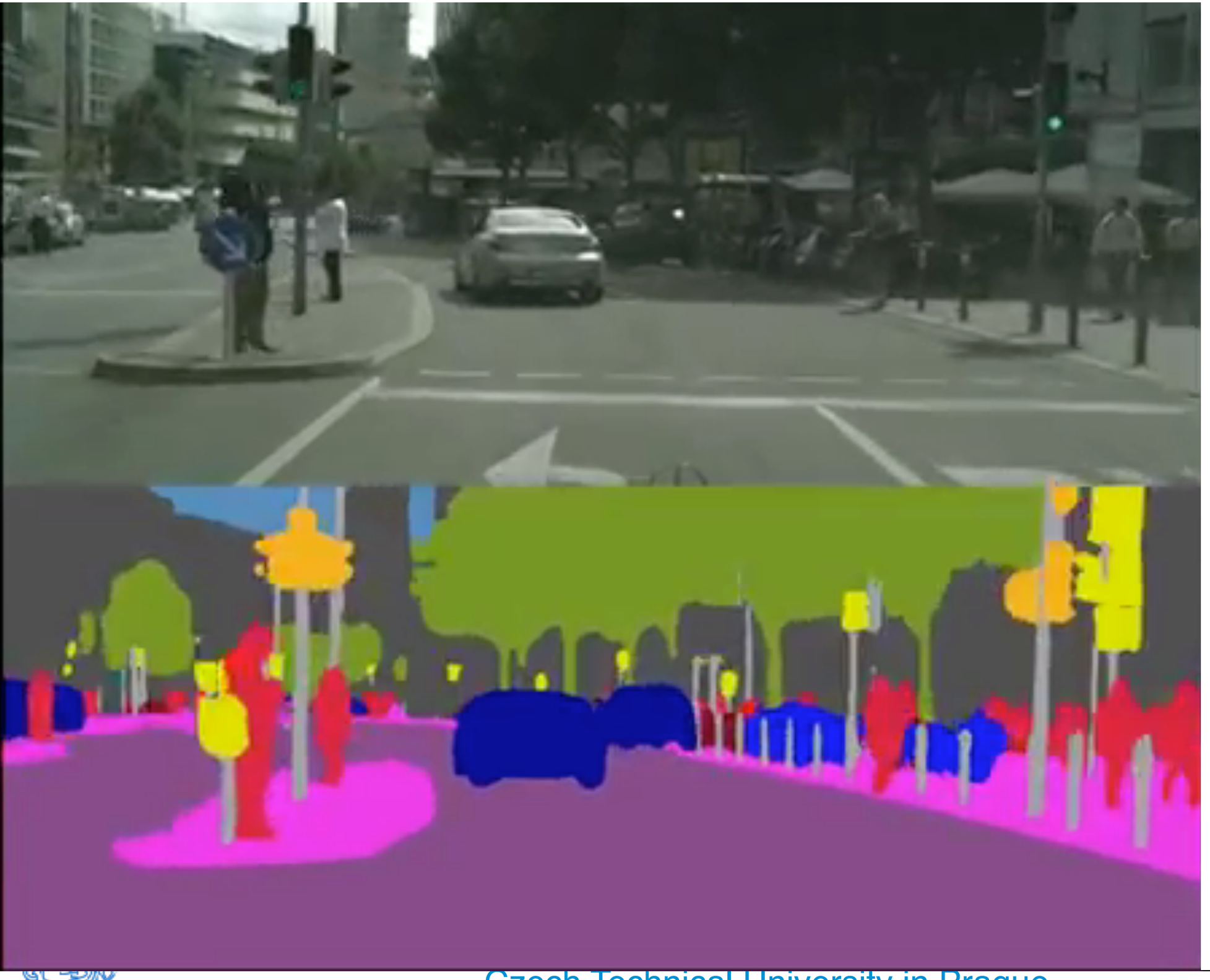
# Summary classification architectures

- 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
- Capsule networks https://medium.com/ai$^3$-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

# Outline

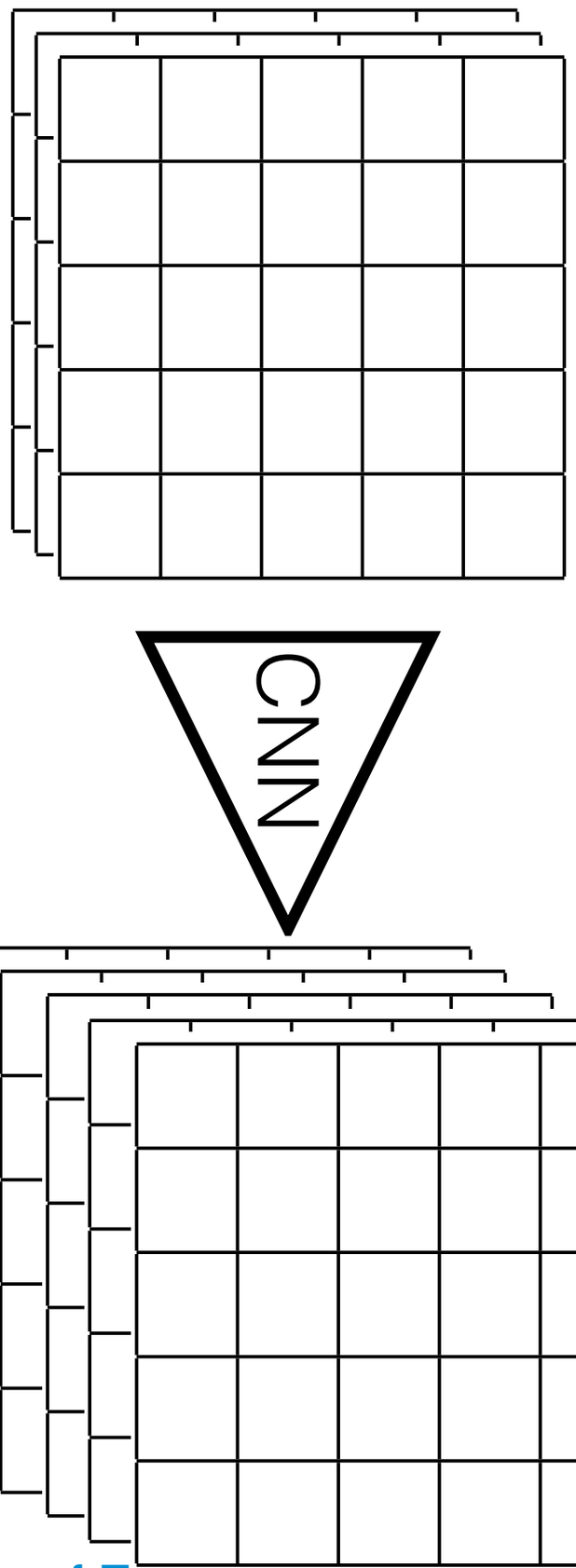
- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Architectures of regression networks
- Architectures of feature matching networks

# Semantic segmentation



road
sideway
pedestrian
traffic sign
trees
sky

# Semantic segmentation

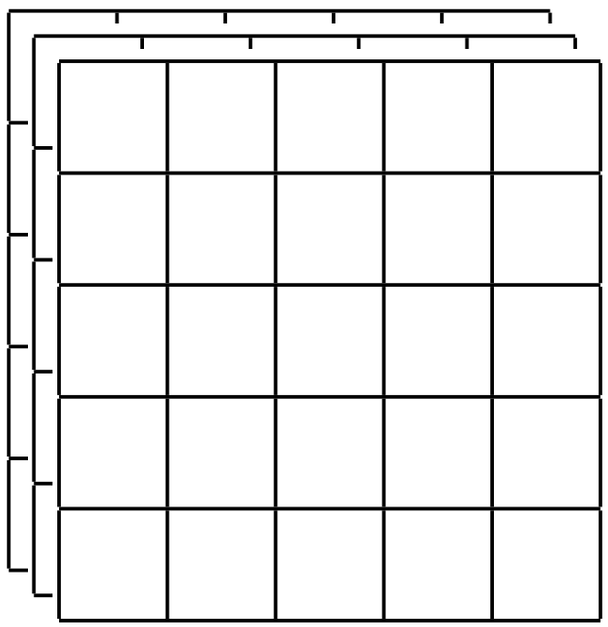



RGB image
(HxWx3)

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

labels
(HxWxN)

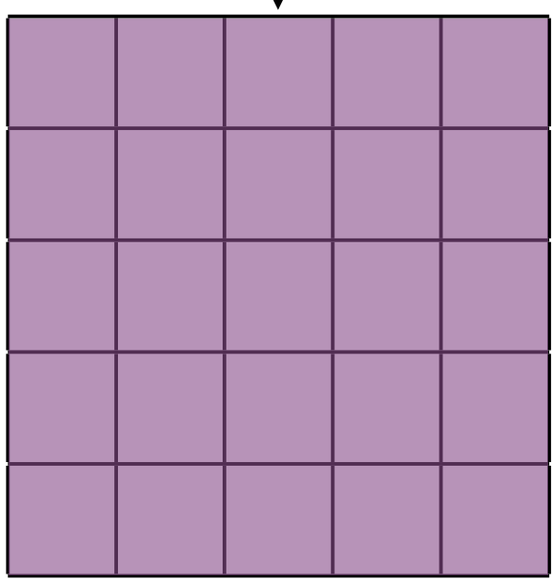
# Semantic segmentation



CNN

RGB image
(HxWx3)

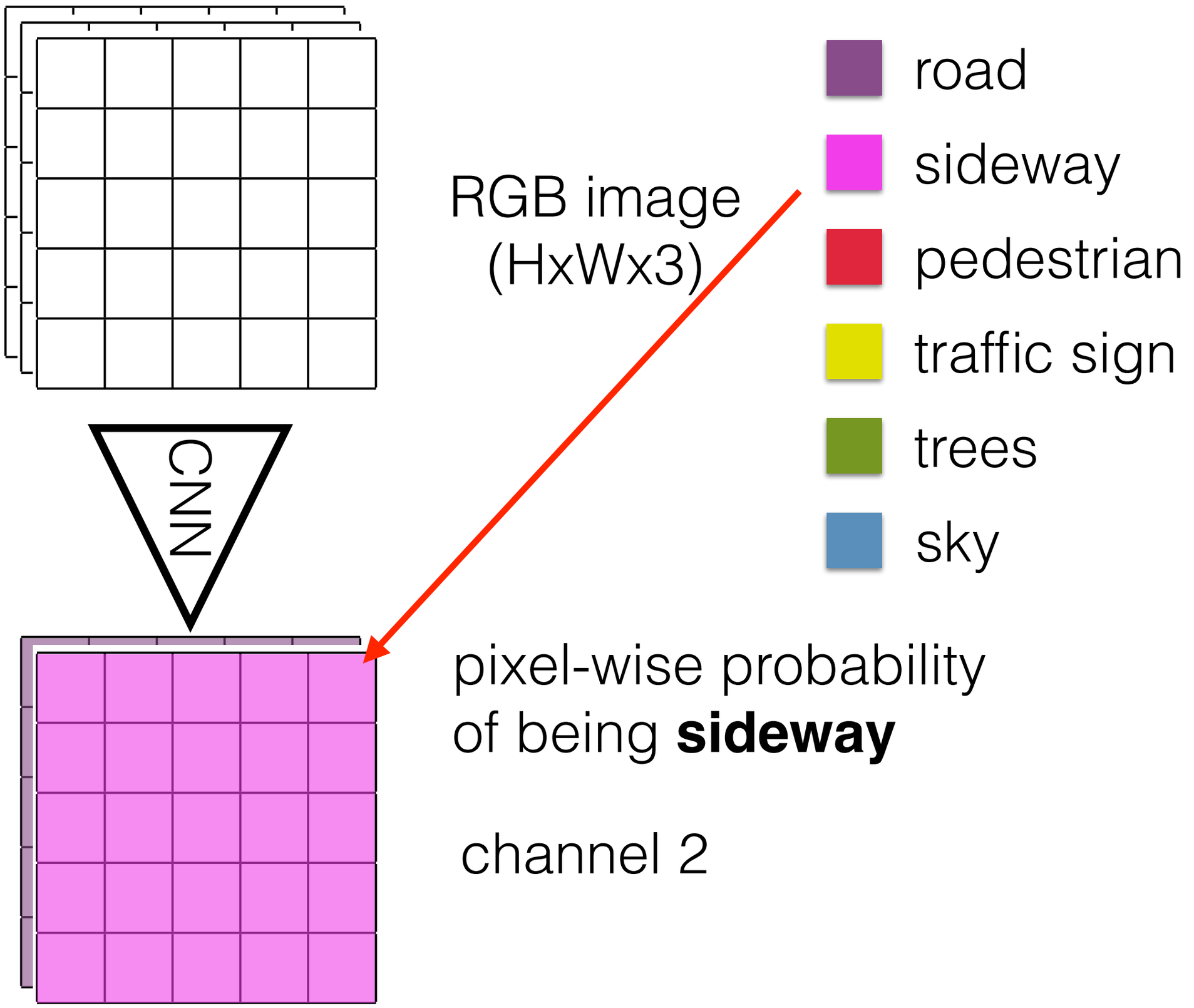
road
sideway
pedestrian
traffic sign
trees
sky



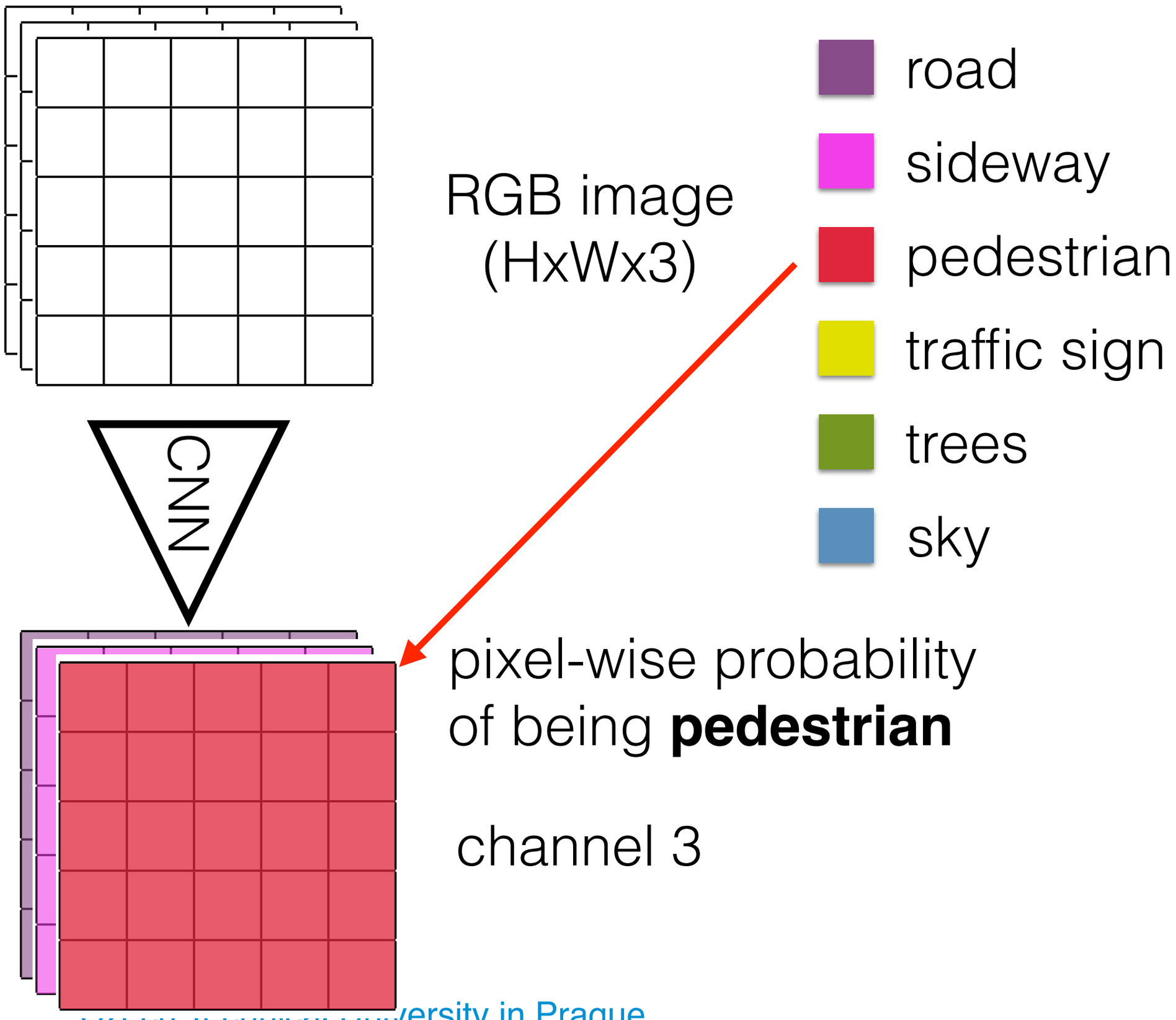
pixel-wise probability
of being **road**

channel 1

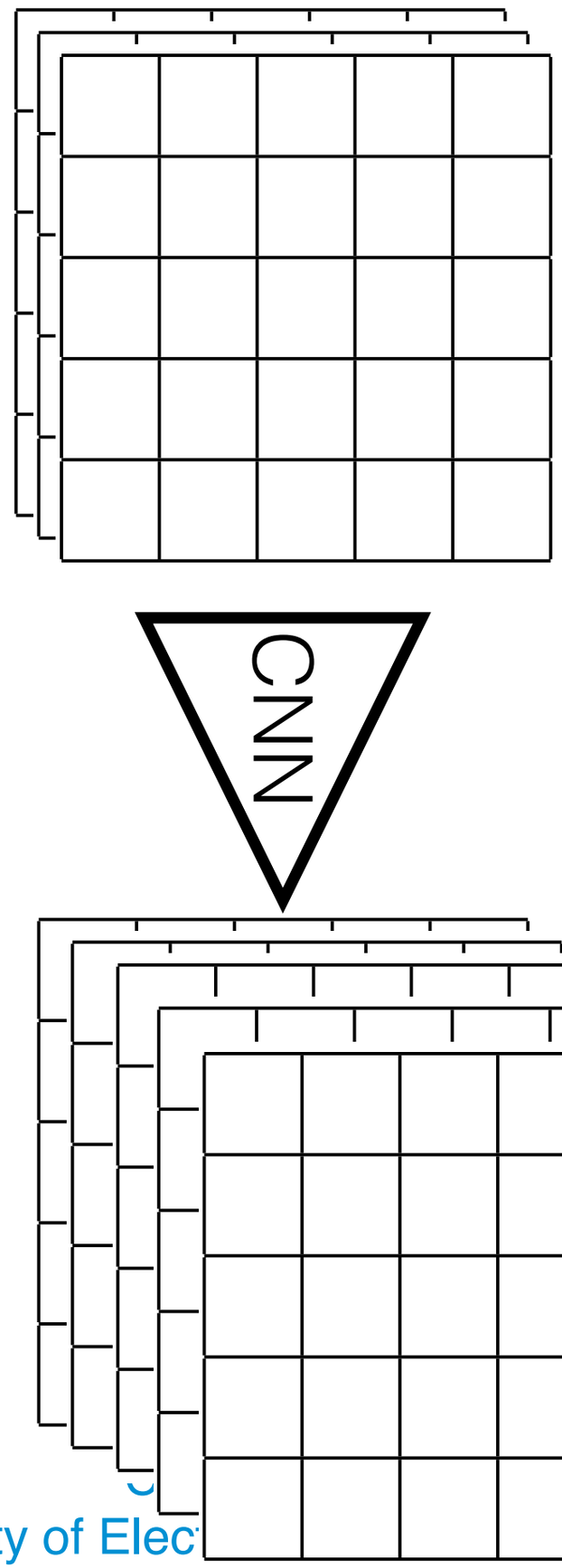
# Semantic segmentation



# Semantic segmentation



# Semantic segmentation

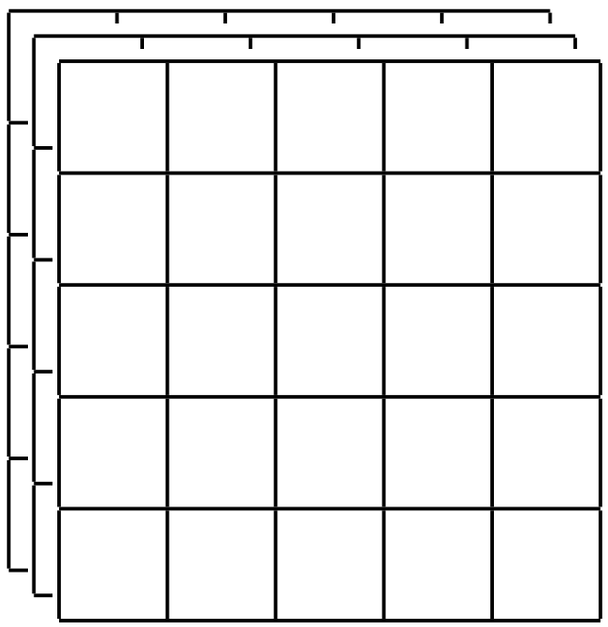



RGB image
(HxWx3)

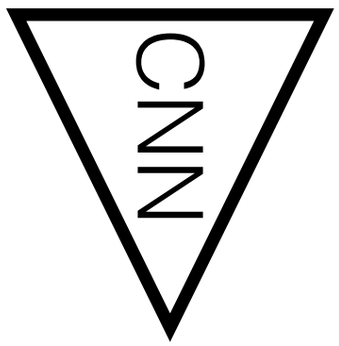
road
sideway
pedestrian
traffic sign
trees
sky

as many output channels
as semantic labels

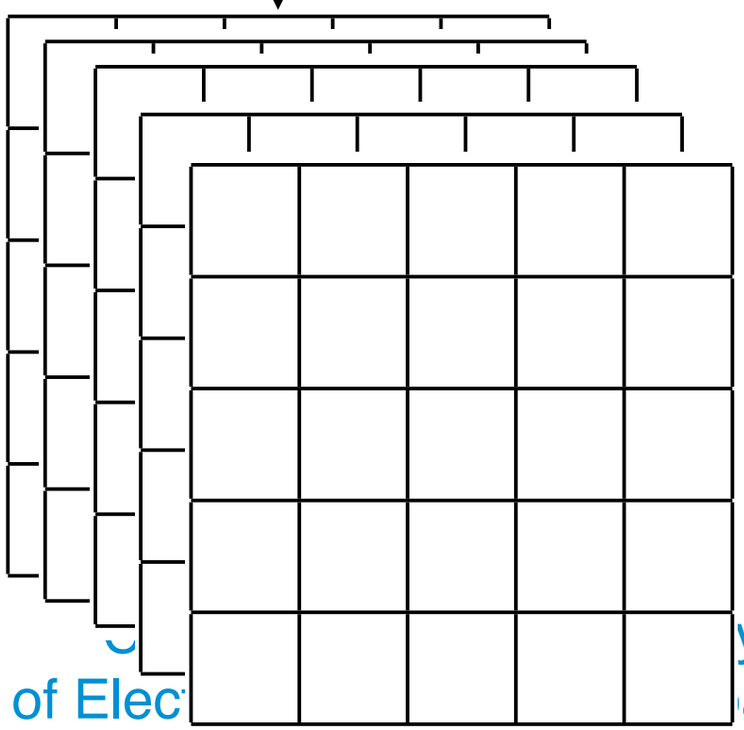
# Semantic segmentation

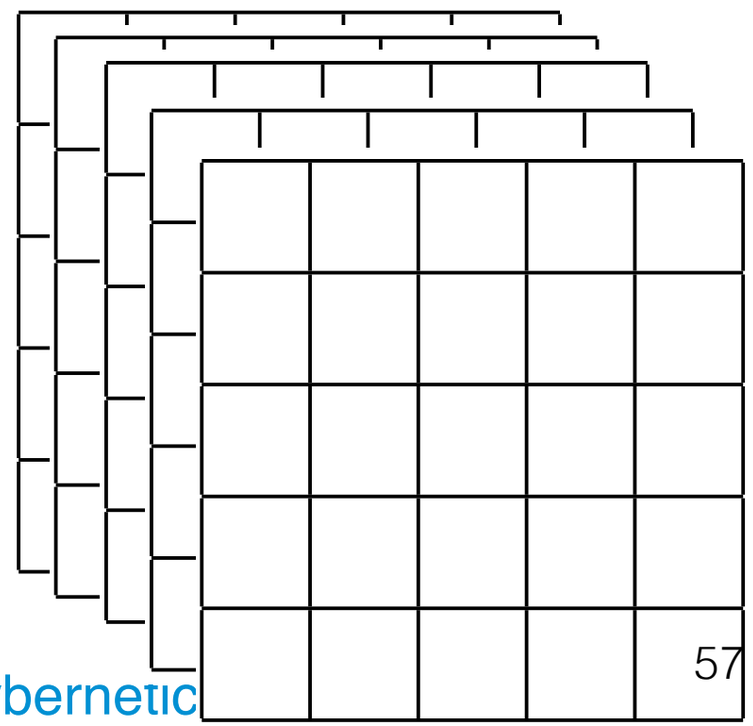


RGB image
(HxWx3)

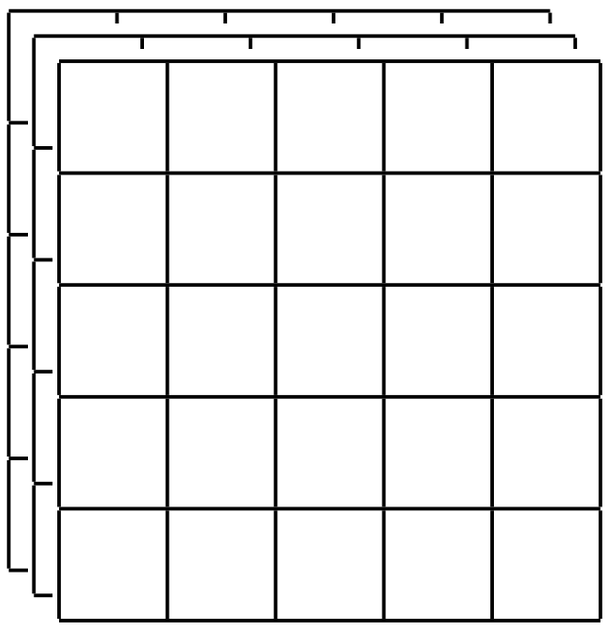



ground truth (0-1 values)

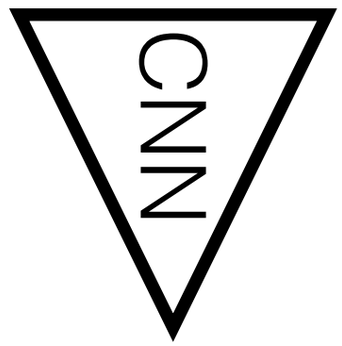




# Semantic segmentation

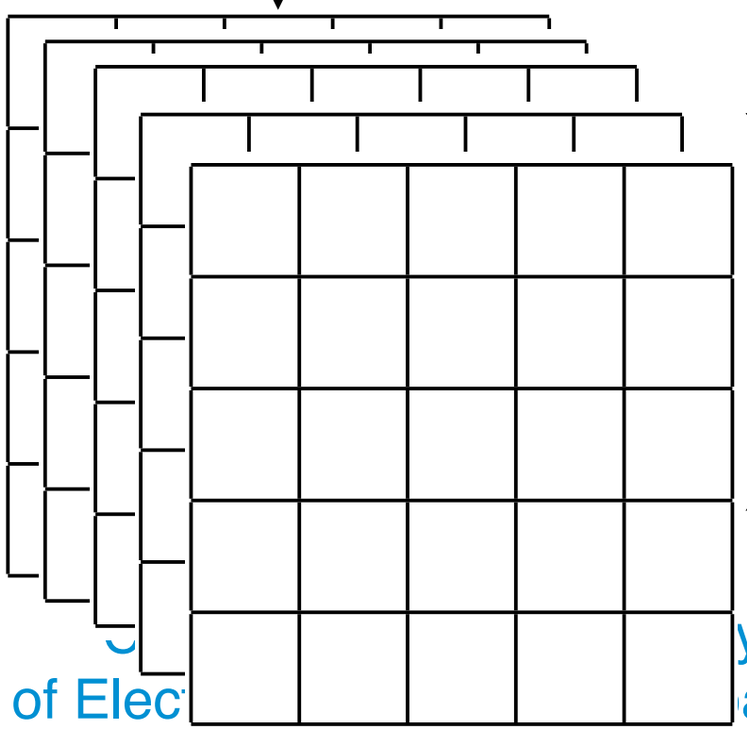


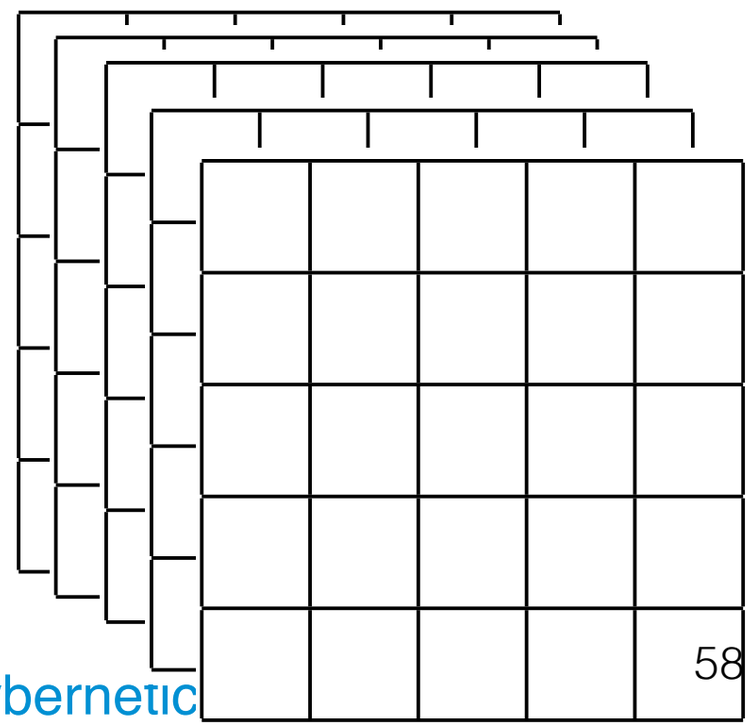
RGB image
(HxWx3)



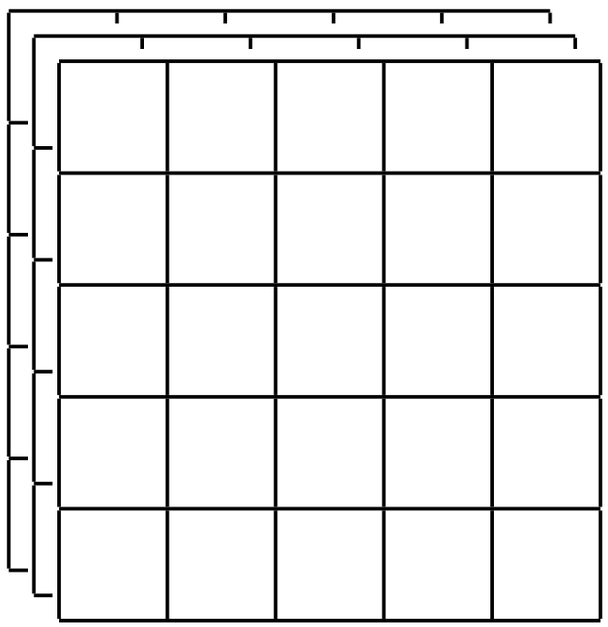

ground truth (0-1 values)

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





# Semantic segmentation



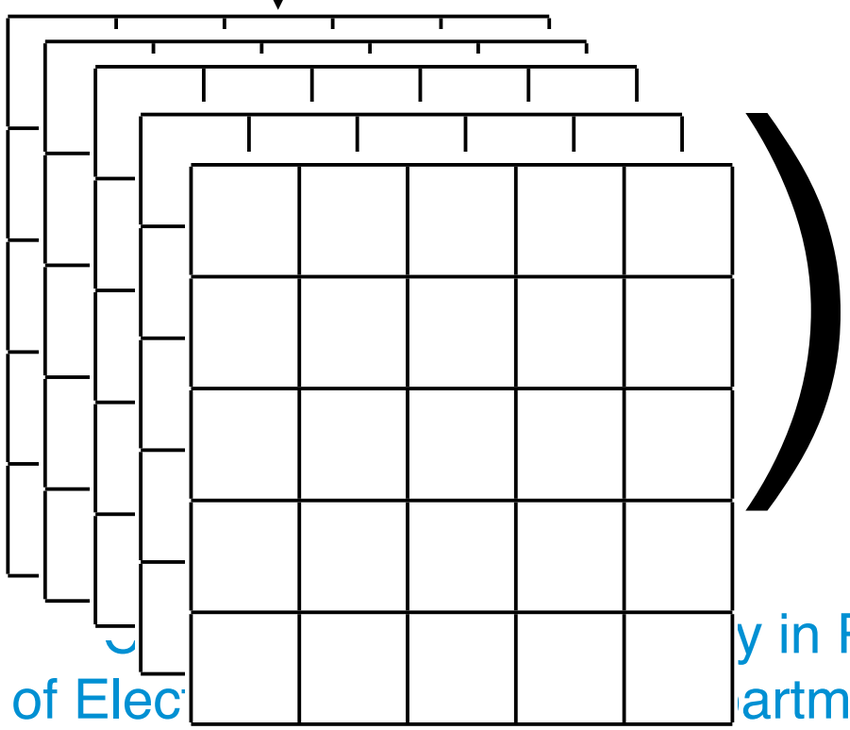
RGB image
(HxWx3)

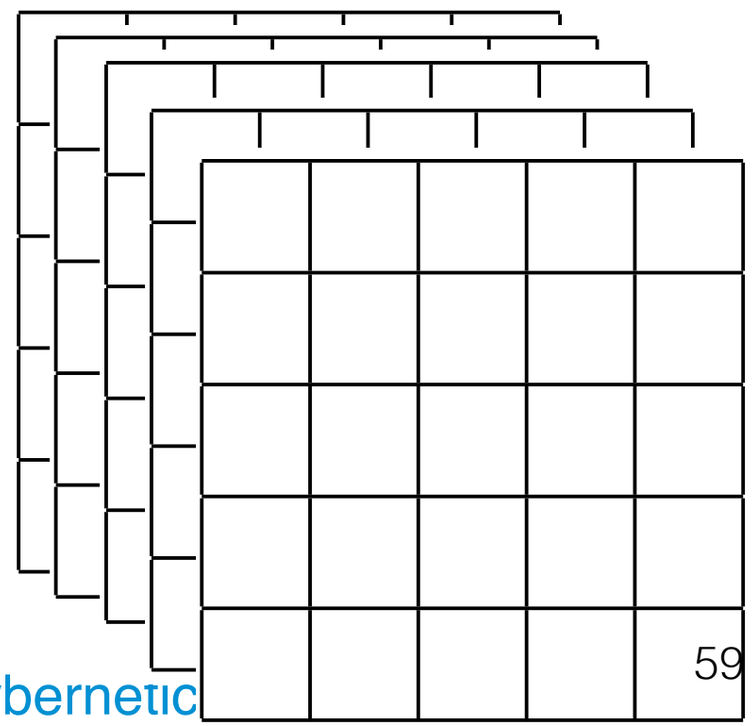
CNN

cross-entropy loss

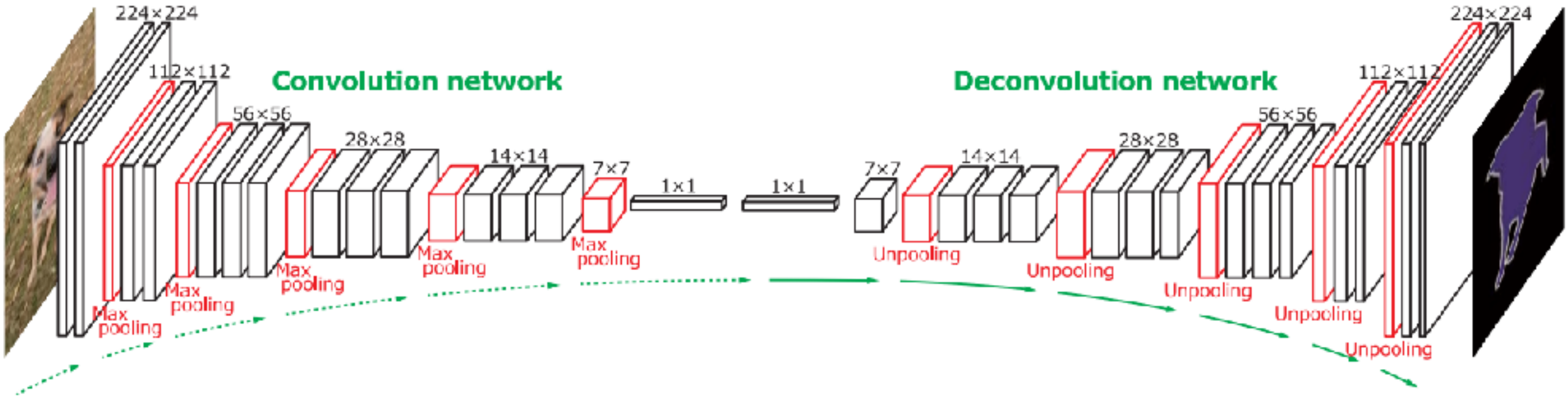
ground truth (0-1 values)

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





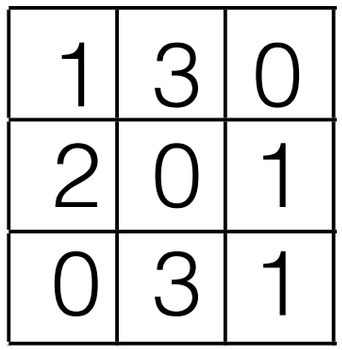
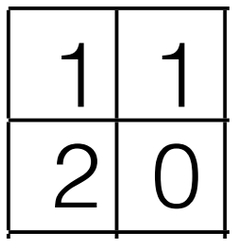
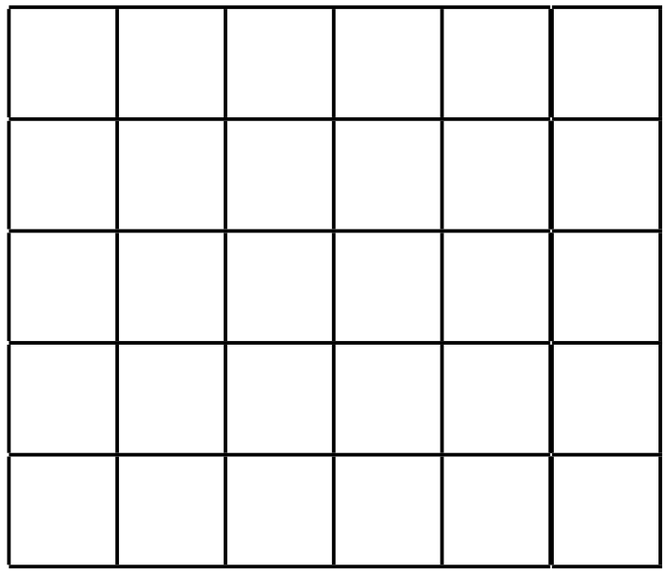
# Semantic segmentation




- 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 (  ,  ) = 

| | | |
|---|---|---|
| 1 | 3 | 0 |
| 2 | 0 | 1 |
| 0 | 3 | 1 |

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

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

# Deconvolution

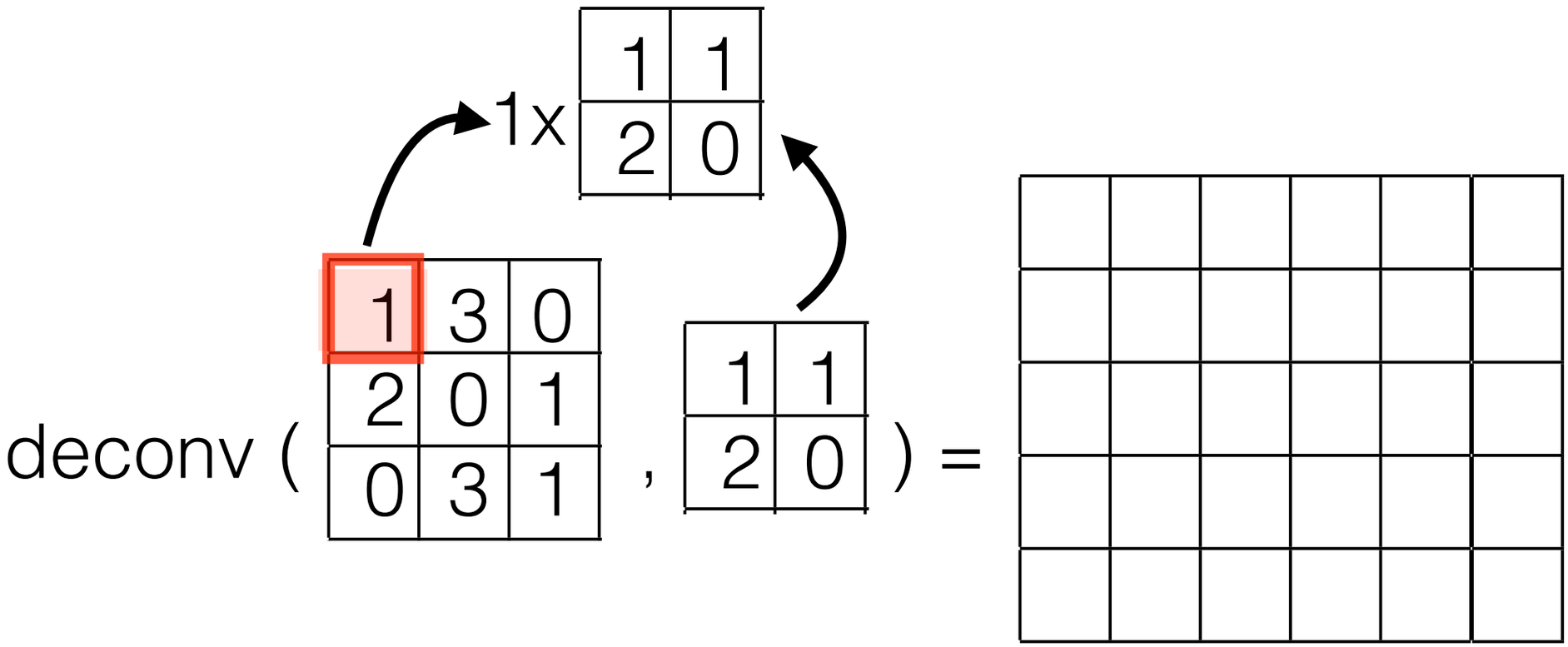



image (3x3)

kernel (2x2)

output (**6x6**)

# Deconvolution

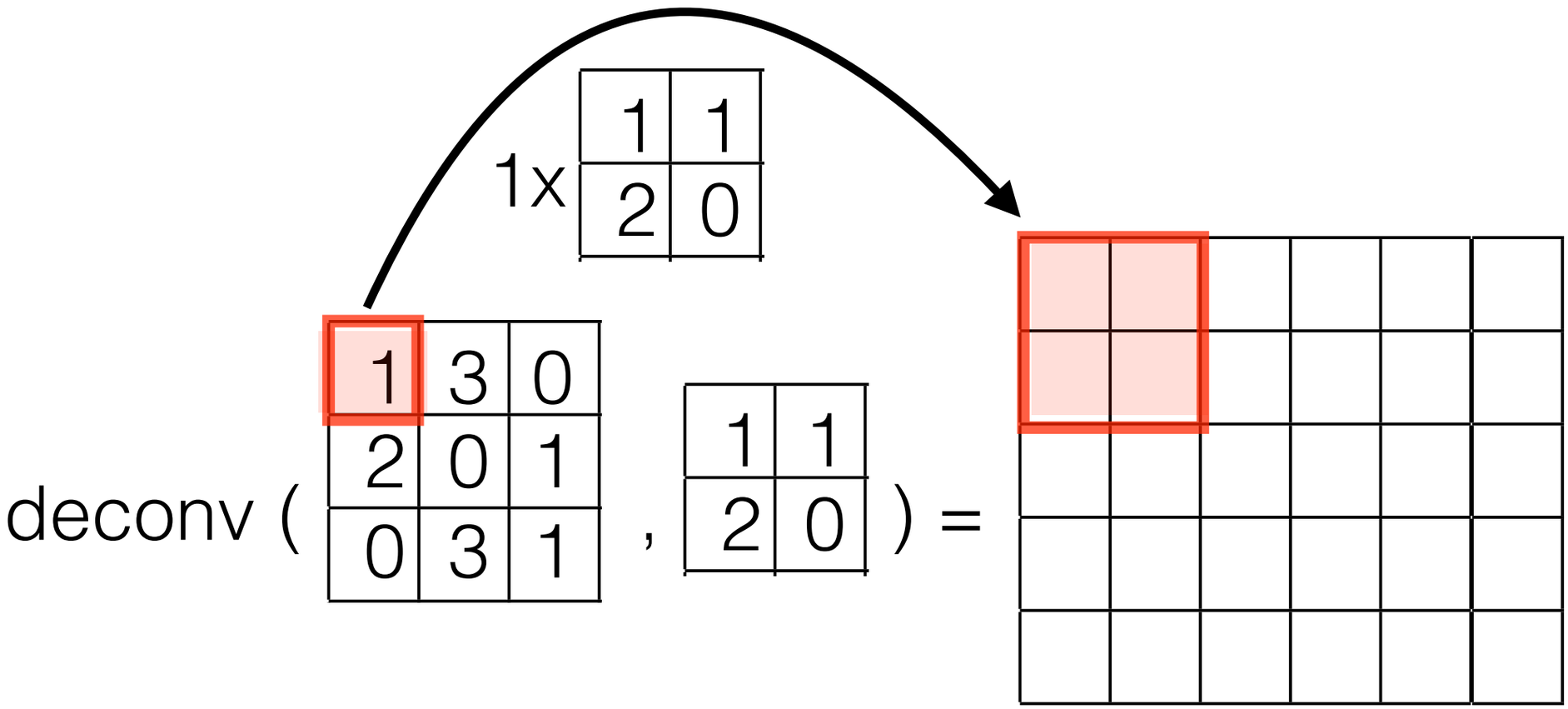



image (3x3)

kernel (2x2)

output (**6x6**)

# Deconvolution

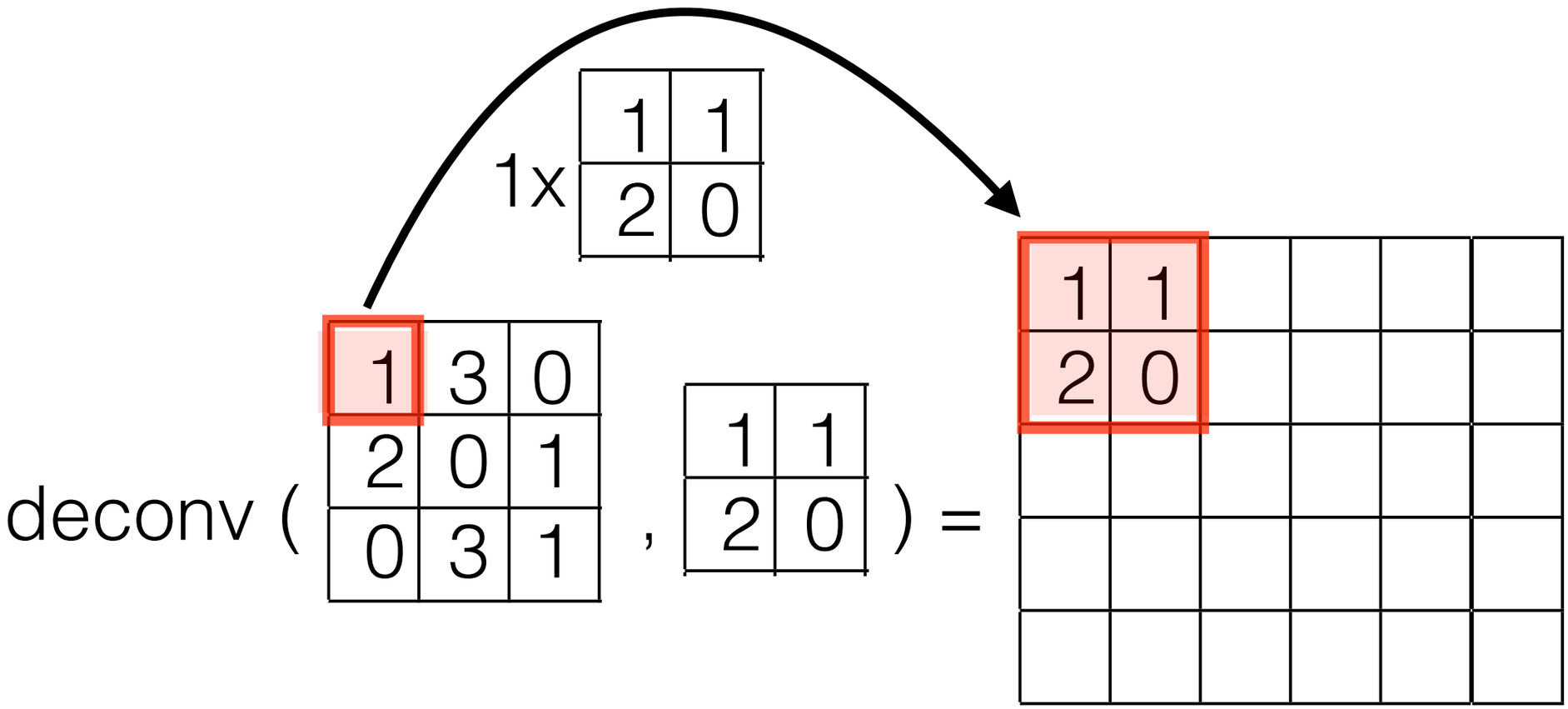



image (3x3) kernel (2x2) output (**6x6**)

# Deconvolution

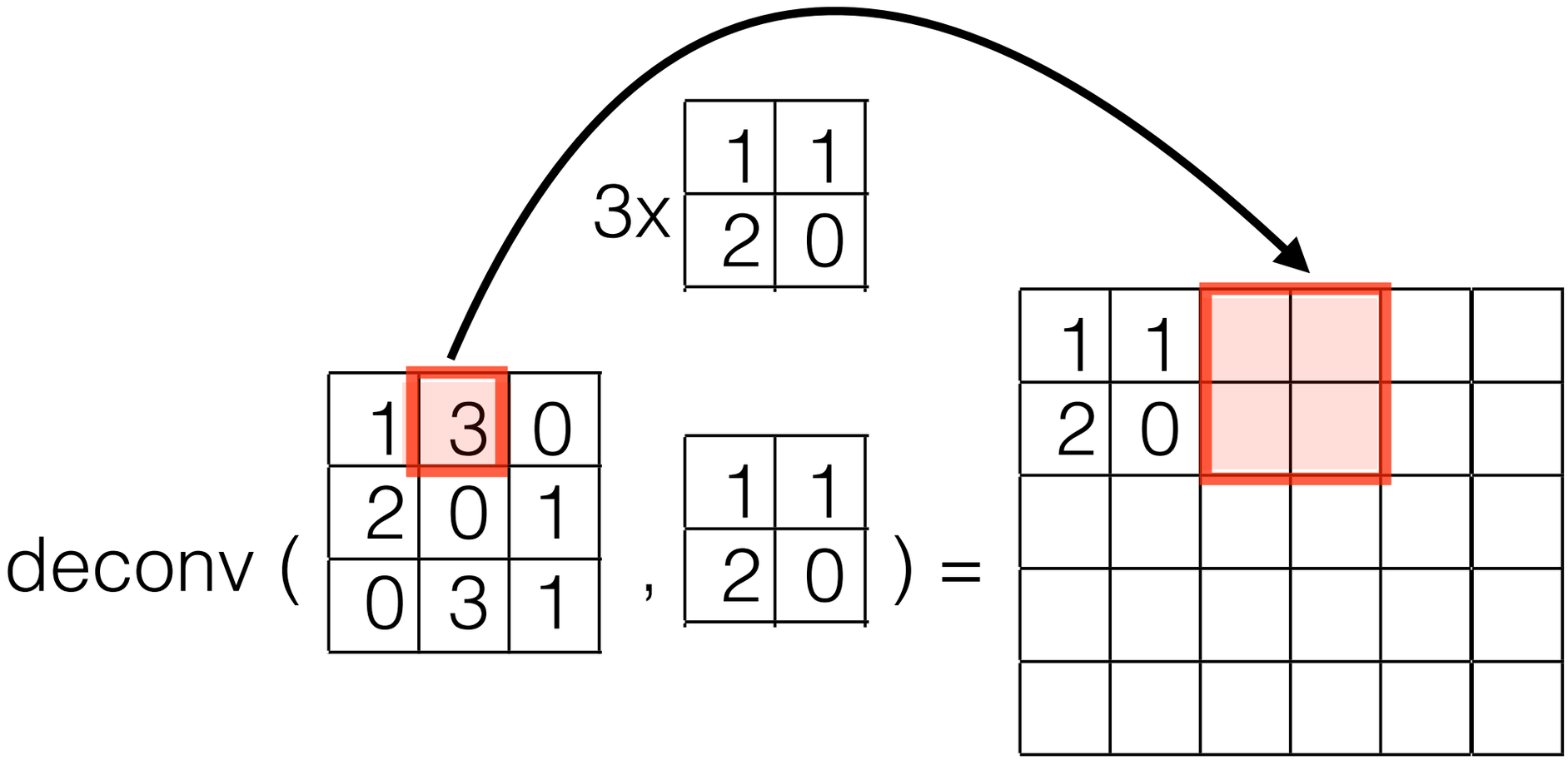



image (3x3)

kernel (2x2)

output (**6x6**)

# Deconvolution

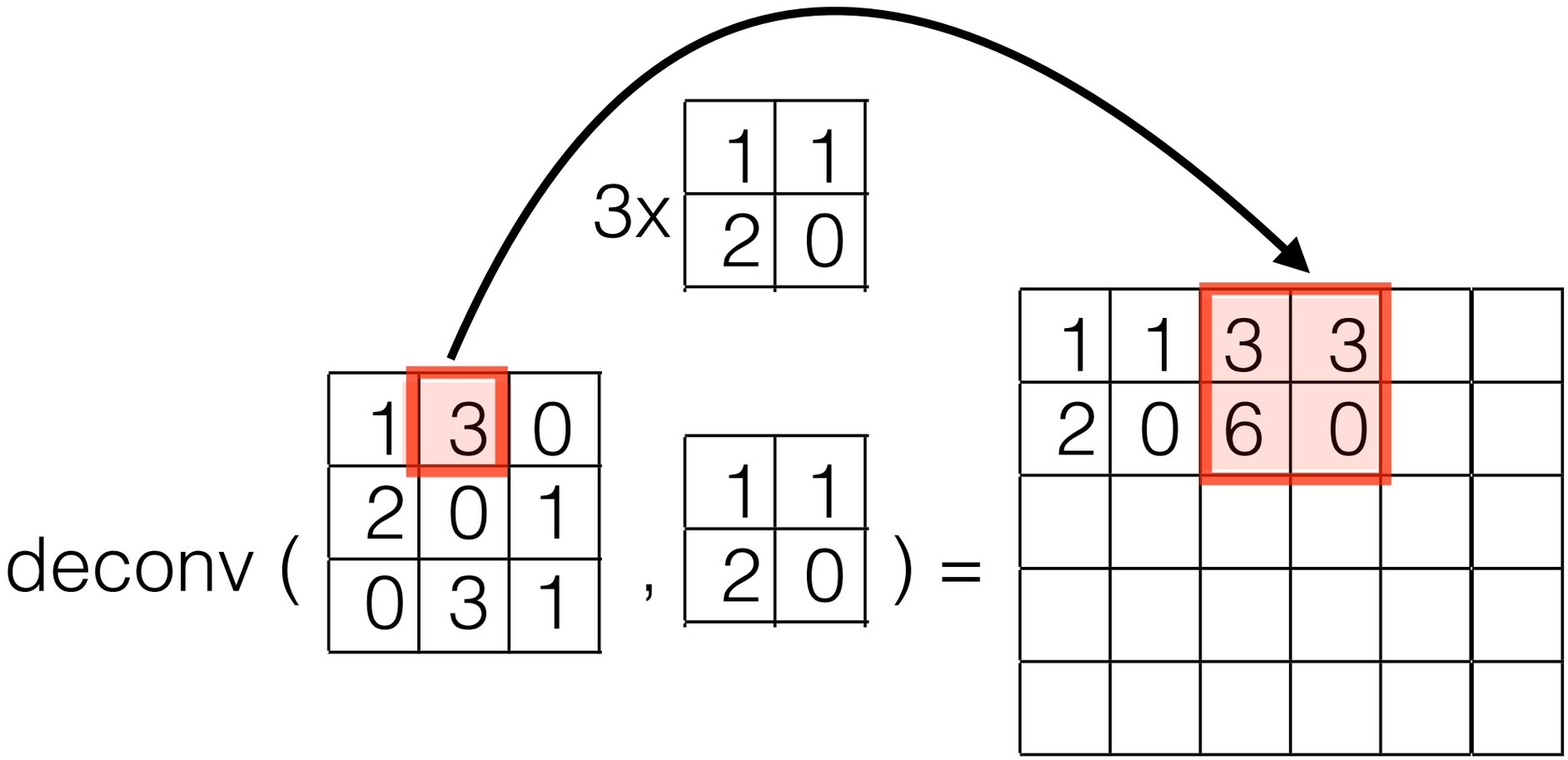



image (3x3) kernel (2x2) output (**6x6**)

# Deconvolution

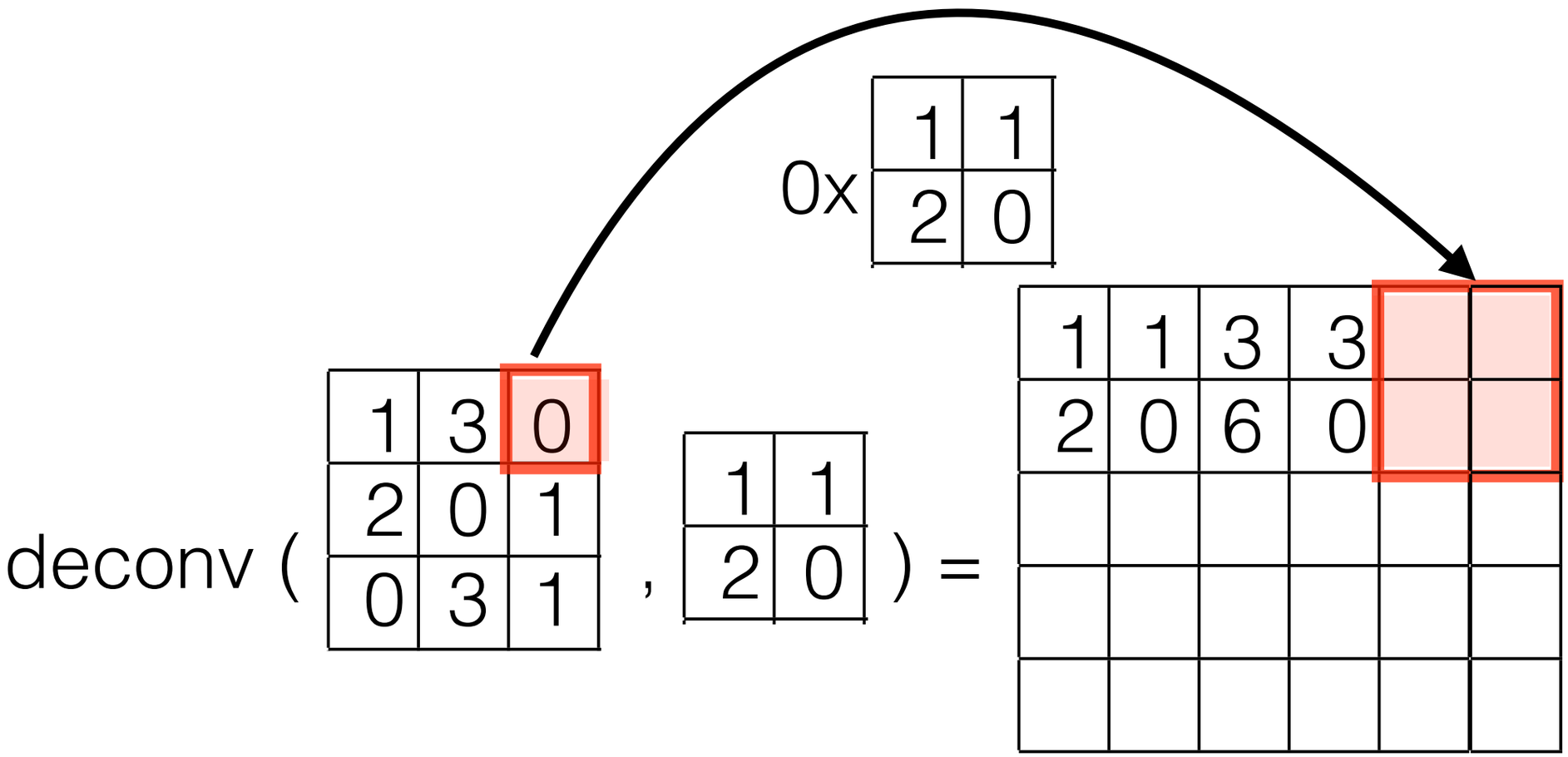



image (3x3) kernel (2x2) output (**6x6**)

# Deconvolution

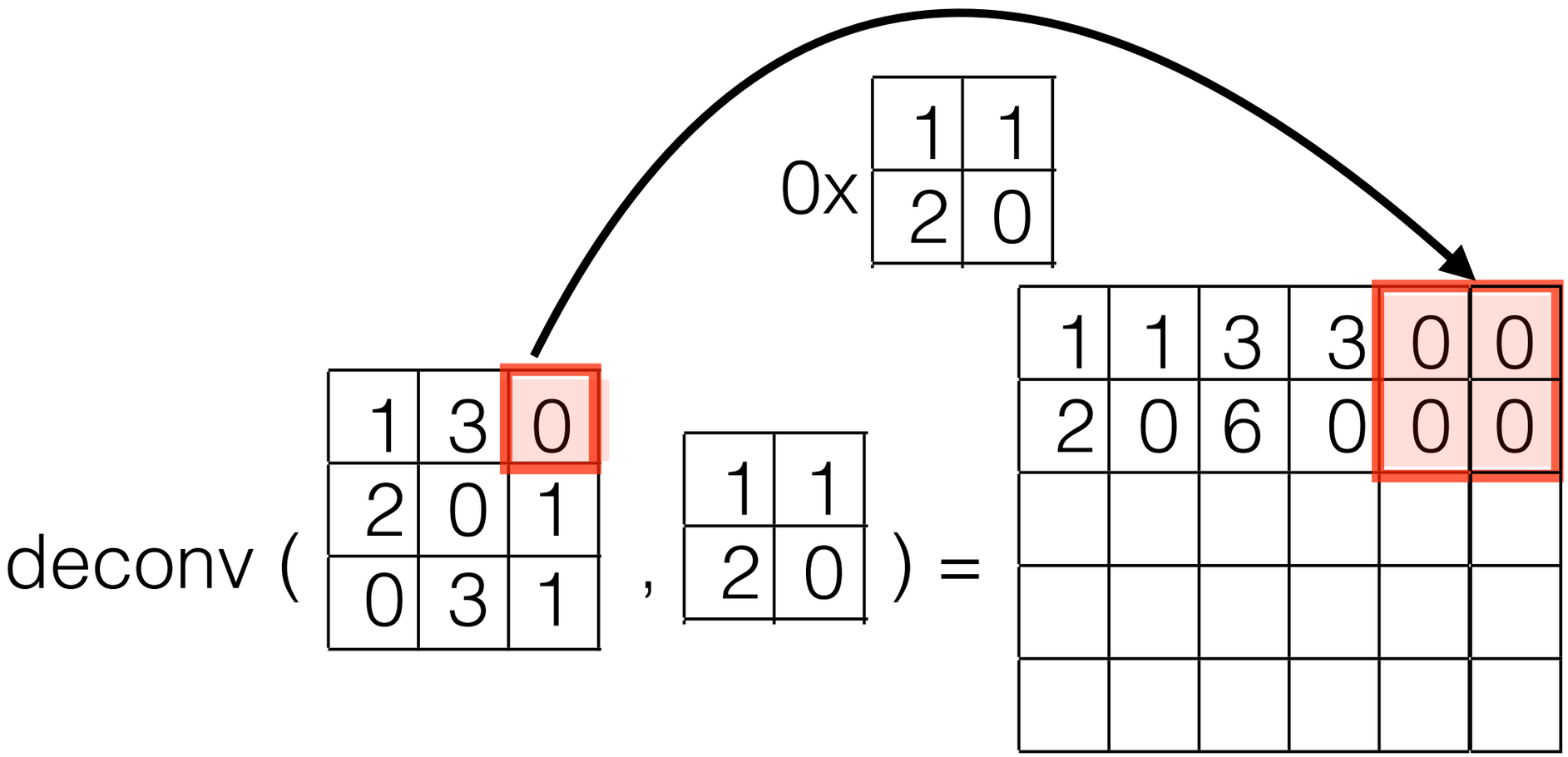



image (3x3) kernel (2x2) output (**6x6**)

# unpooling

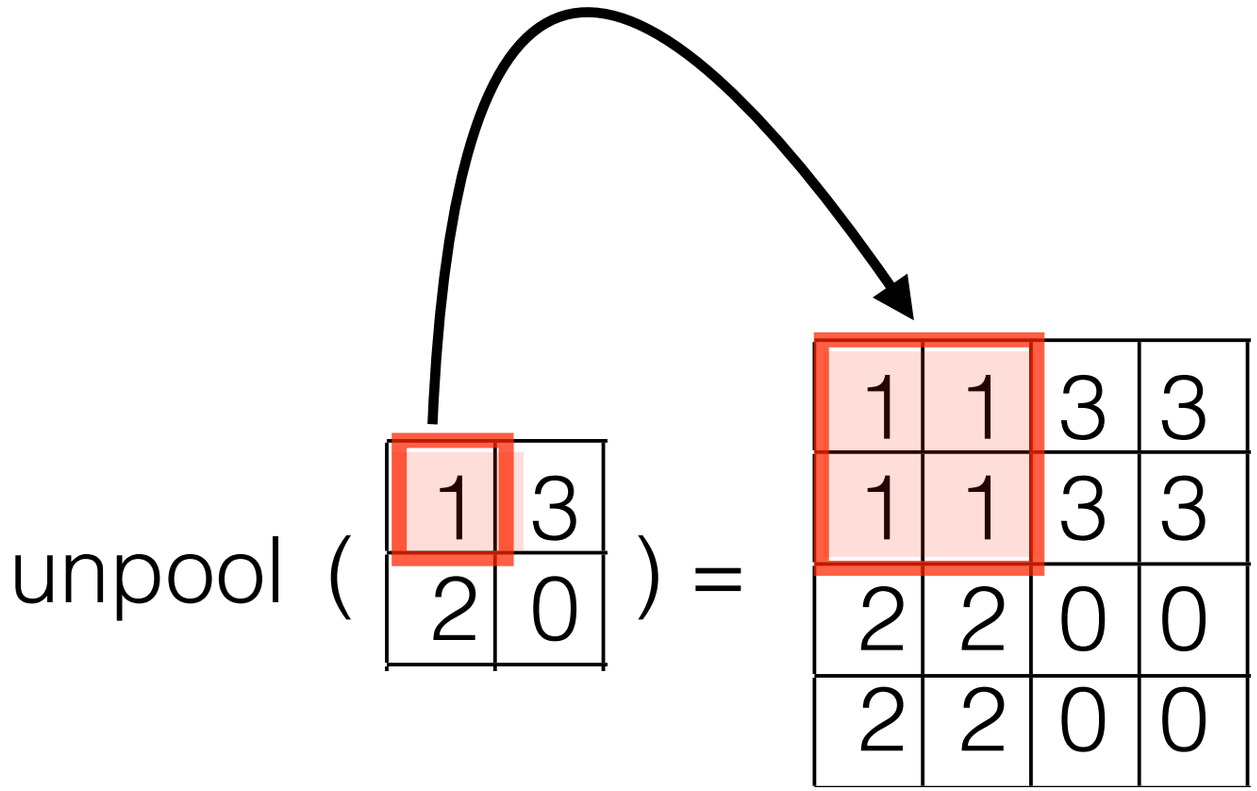



image
(2x2)

output
(**4x4**)

# max-unpooling

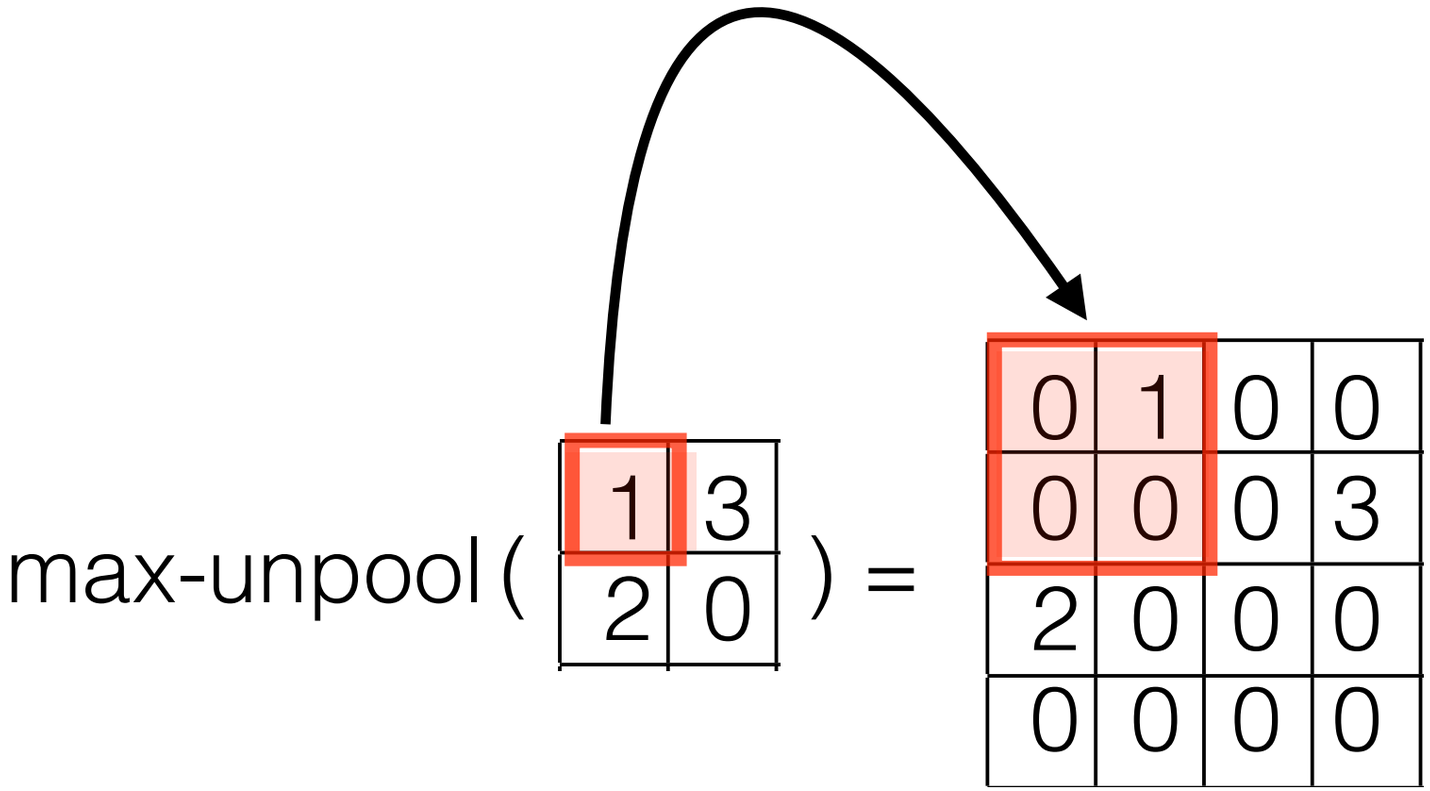
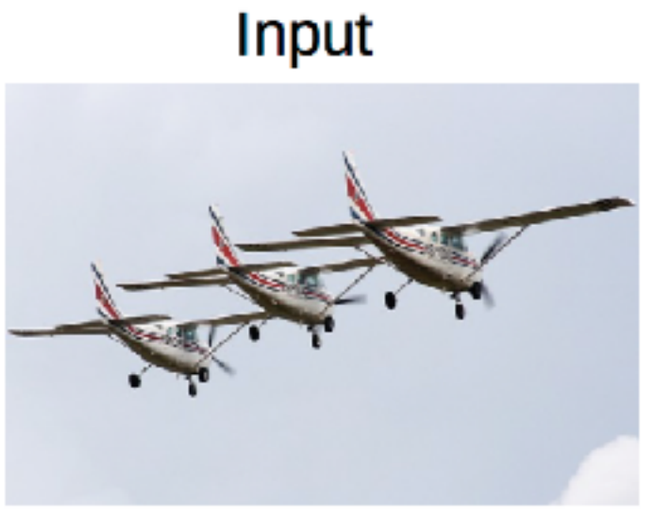



image (2x2)

output (**4x4**)

remember position of the maximum from max-pooling layer

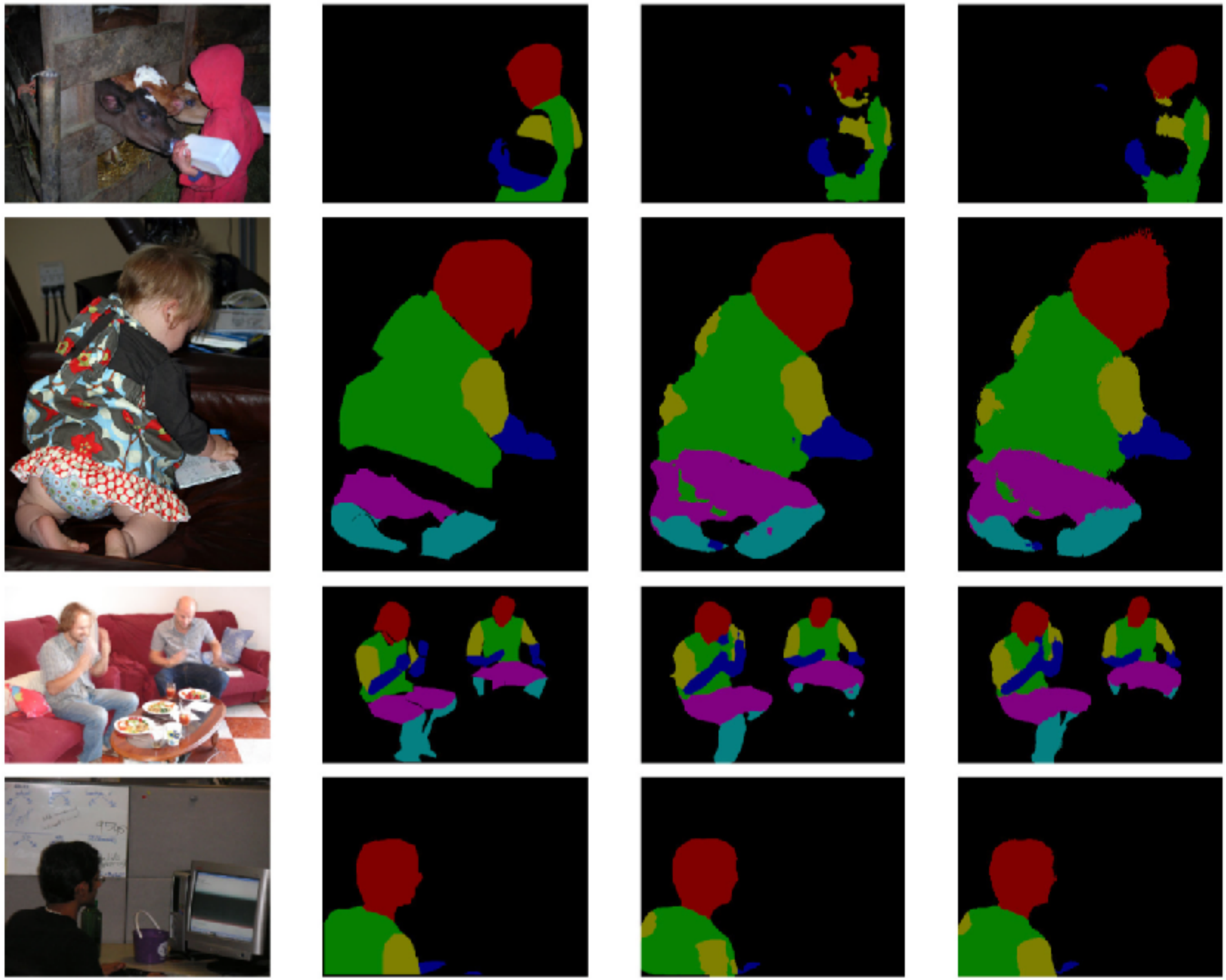
# DeepLab v3




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

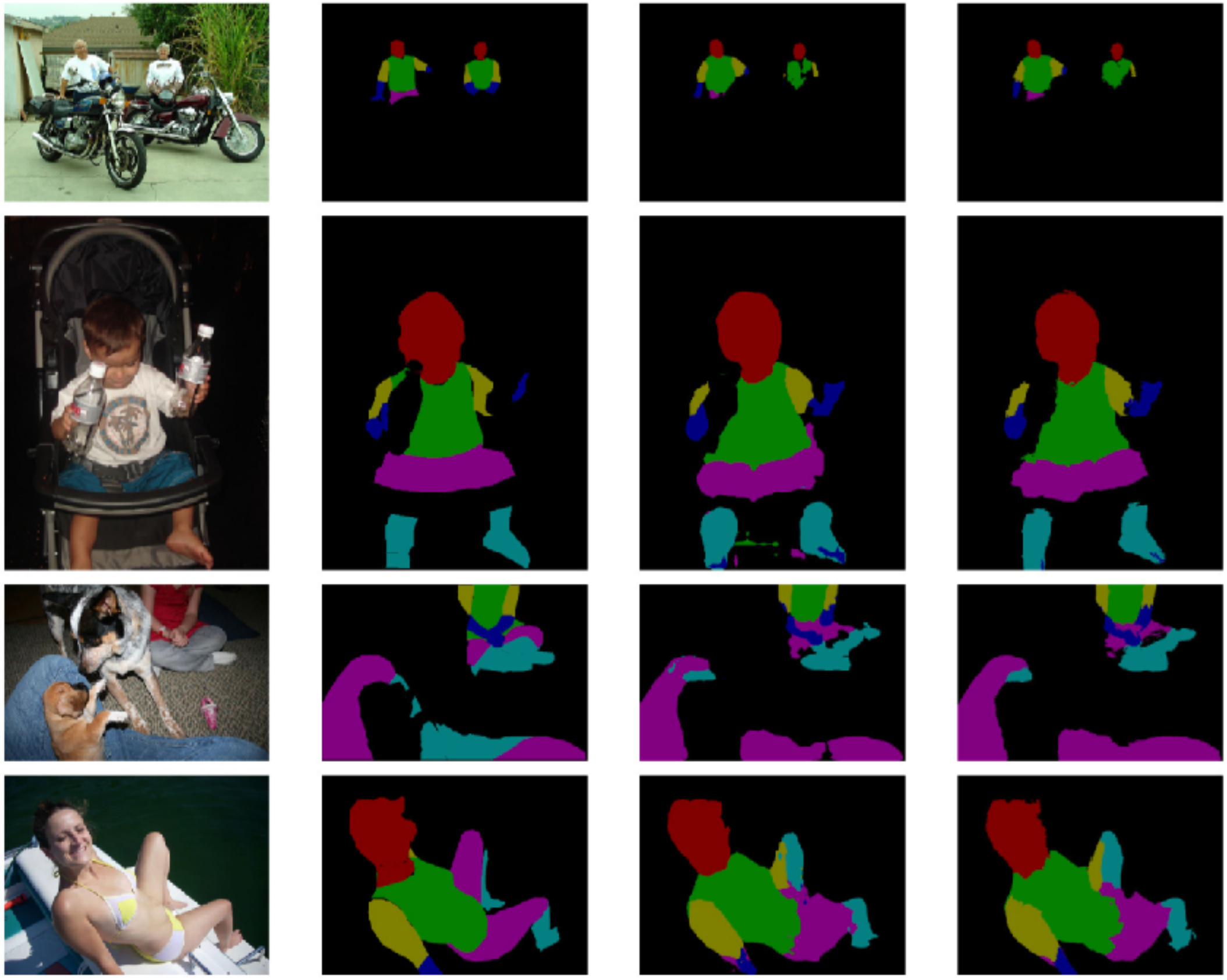
[Chen et al. TPAMI 2018] https://arxiv.org/pdf/1606.00915.pdf

# DeepLab v3 - results



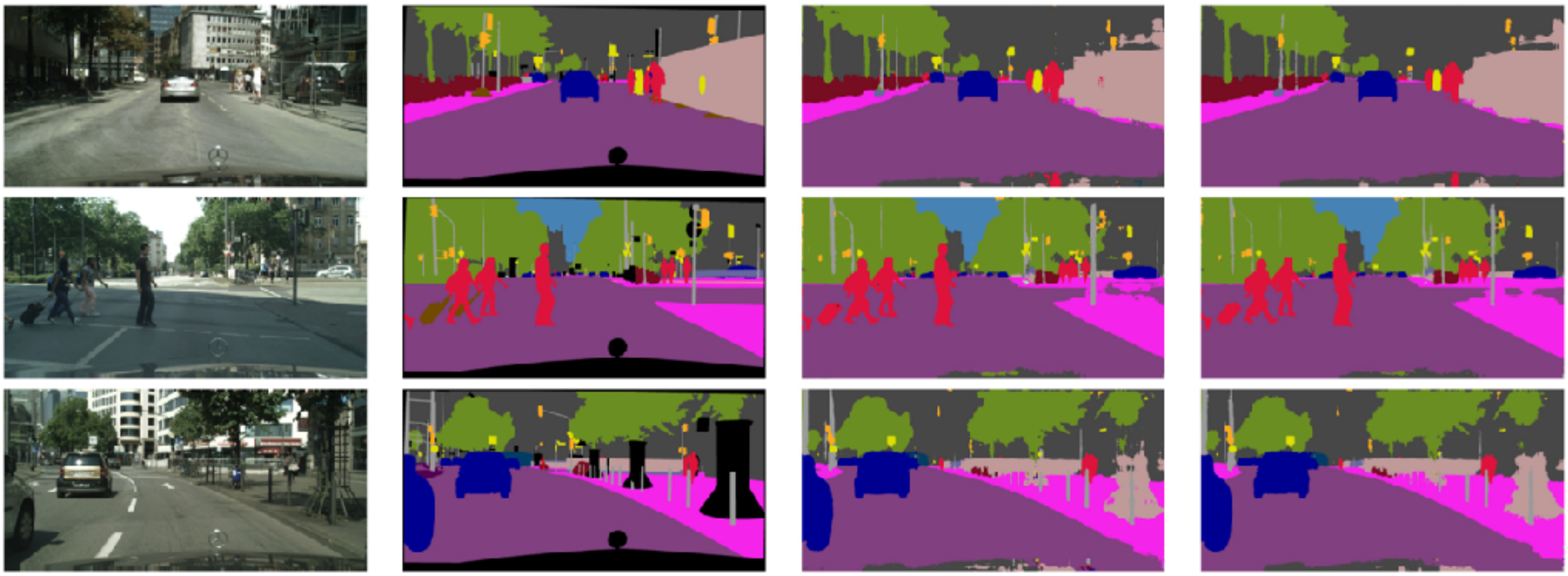
(a) Image (b) G.T. (c) Before CRF (d) After CRF

# DeepLab v3 - results



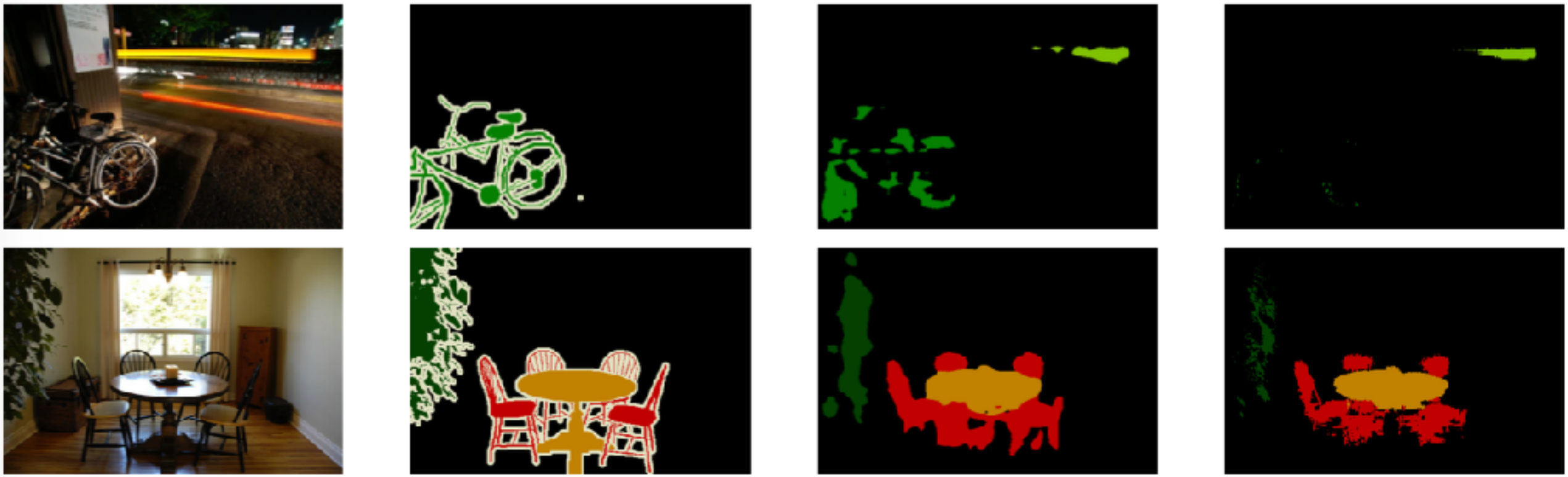
(a) Image (b) G.T. (c) Before CRF (d) After CRF

# 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

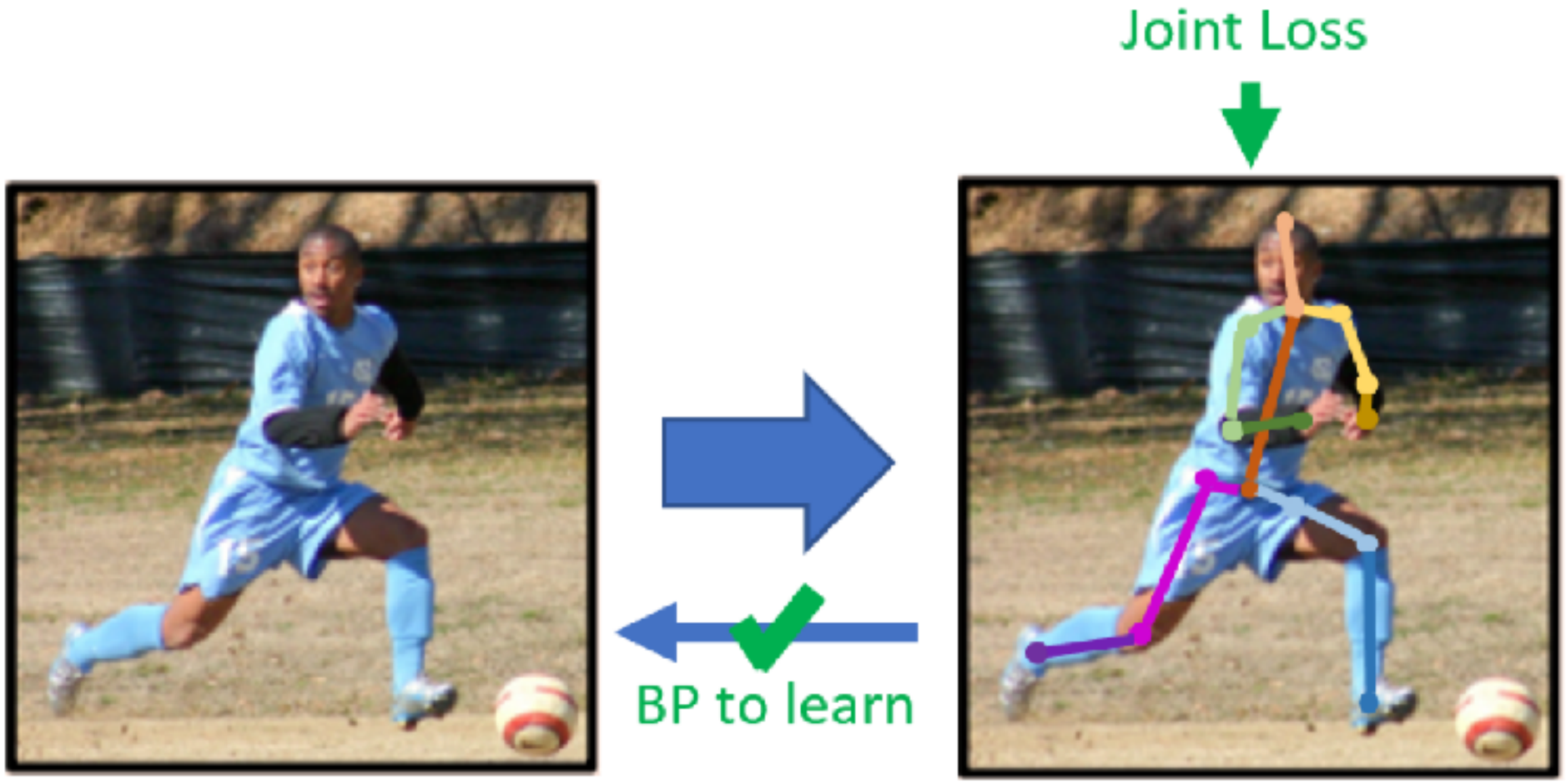
# DeepLab v3 - summary

- 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

# 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

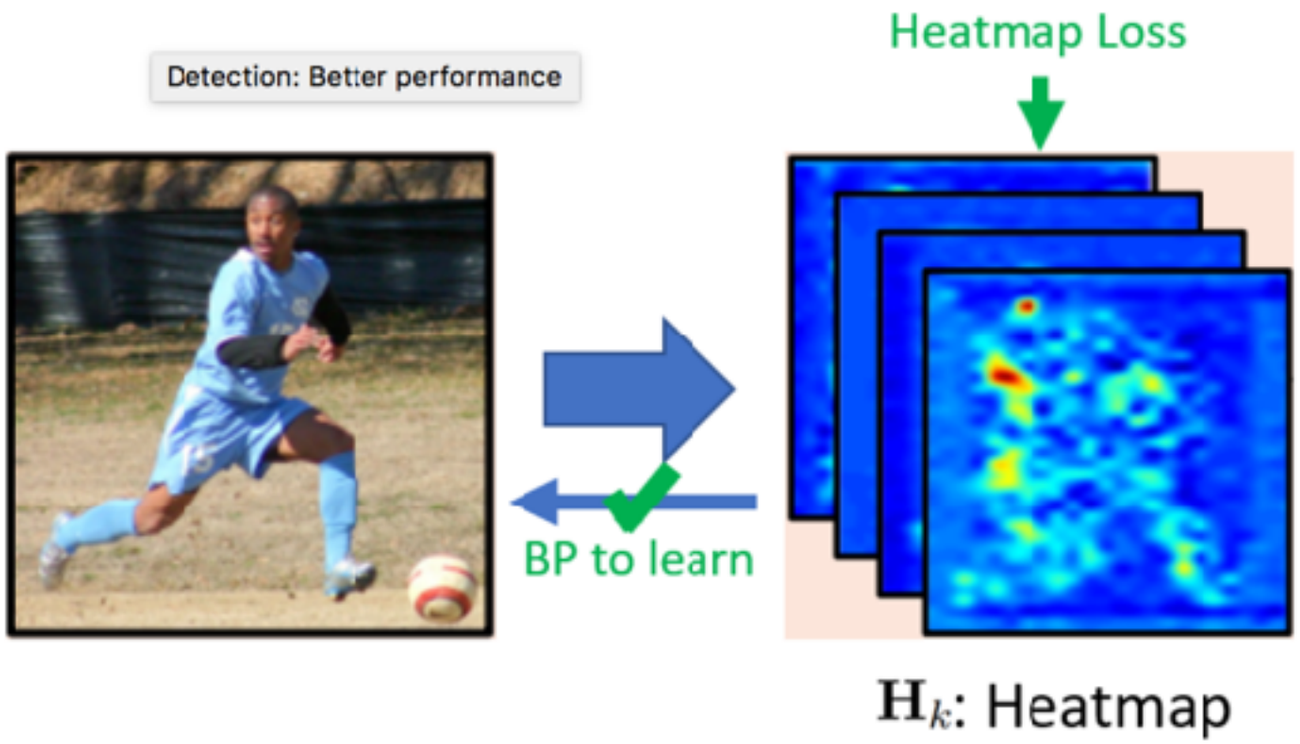
# Pose regression baseline




- 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

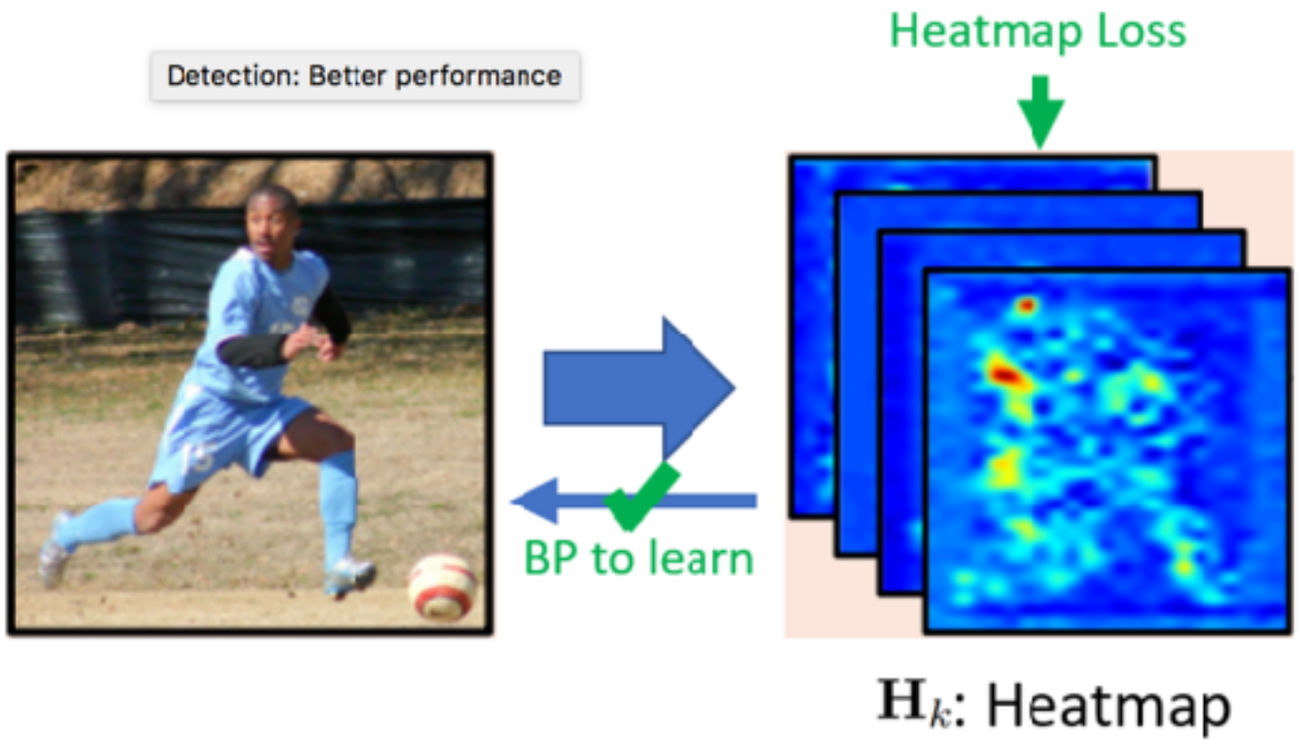
# Pose detection baseline



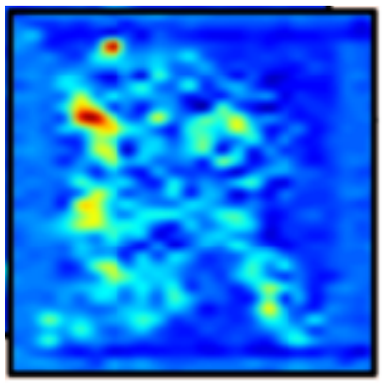

- ConvNet first estimates N joint's heat maps $\mathbf{H}_k,\ k = 1 \ldots N$ (i.e. N 2D-images or N 3D-arrays)
- Learning minimizes segmentation loss over the N images

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

# Pose detection baseline

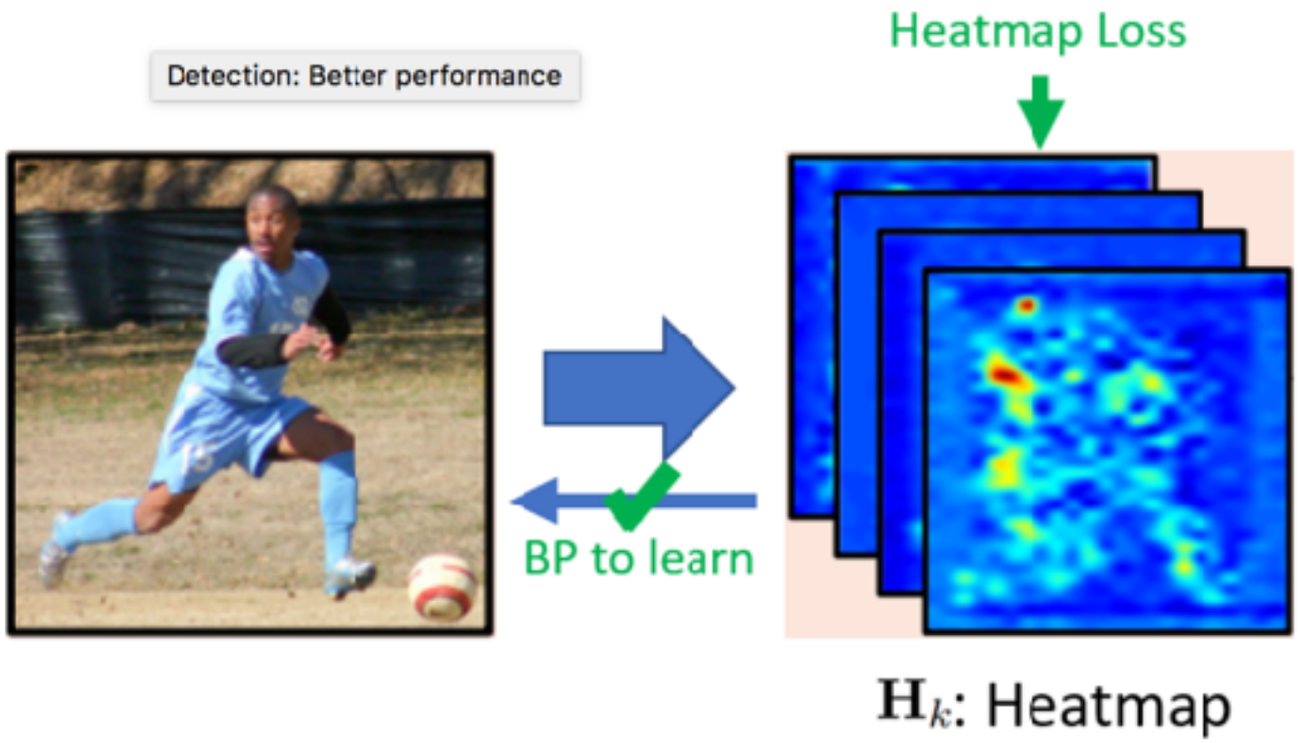



- estimate joint position as position of heatmap maximum

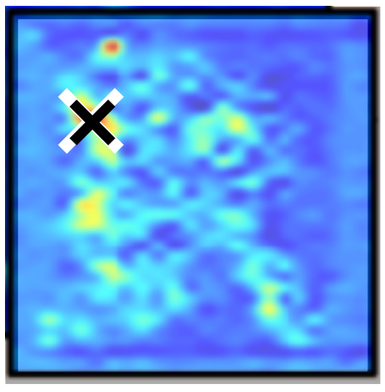
$\mathbf{H}_k$ 

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

# Pose detection baseline

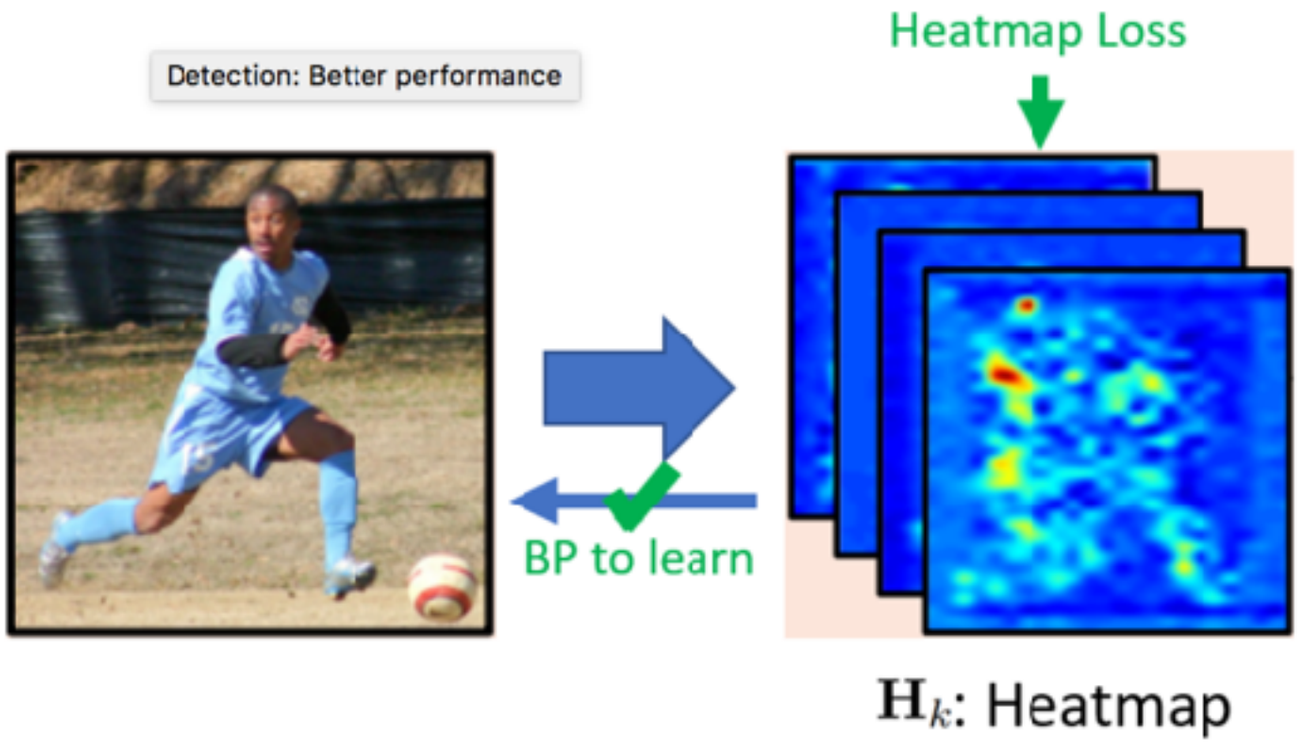



- estimate joint position as position of heatmap maximum

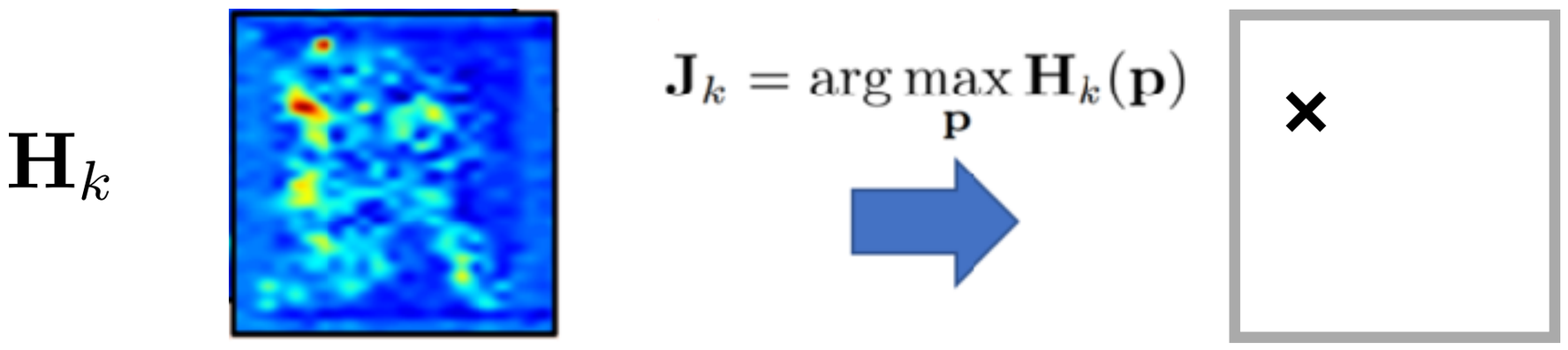
$\mathbf{H}_k$ 

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

# Pose detection baseline

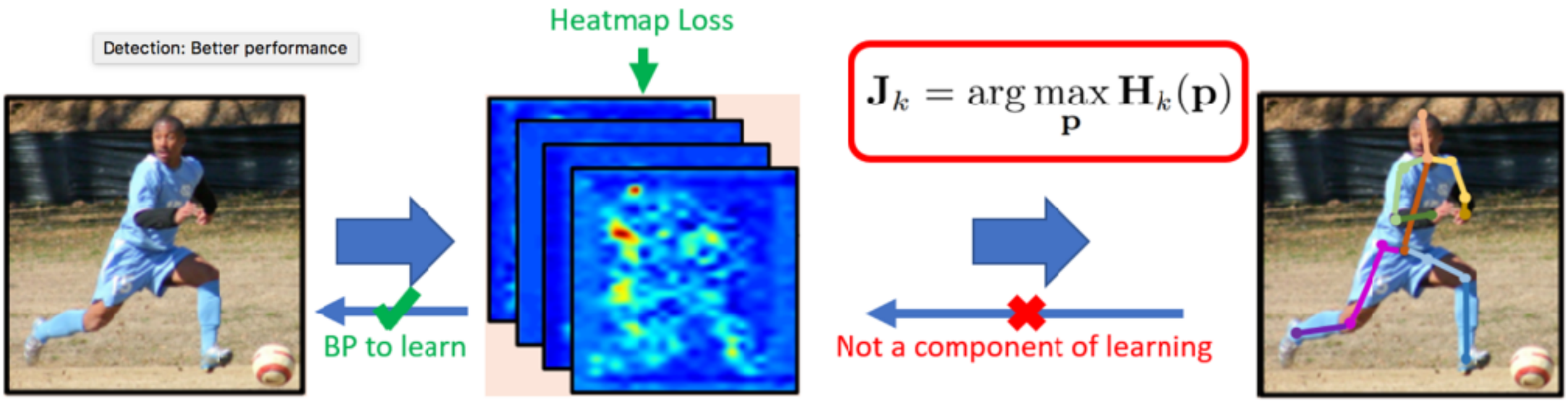



- estimate joint position as position of heatmap maximum

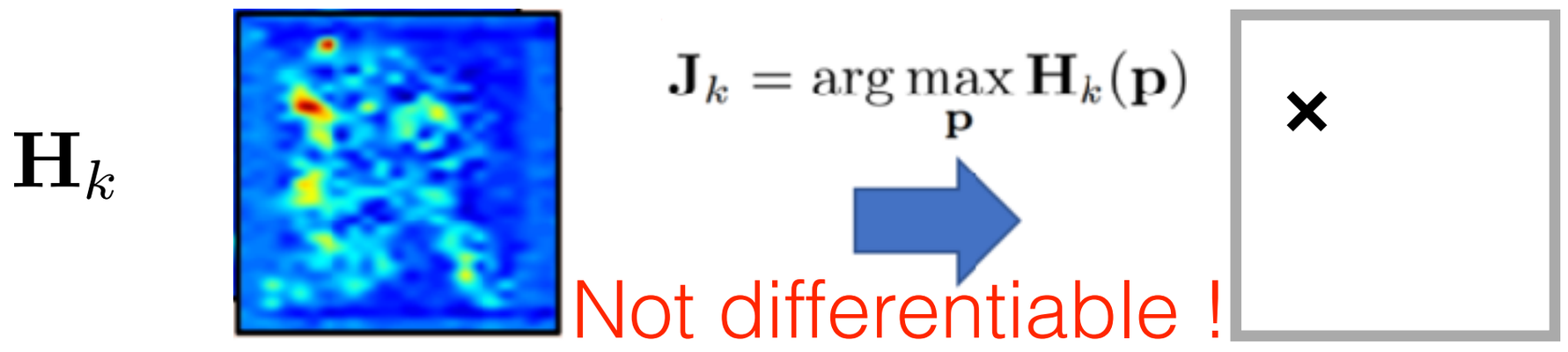



Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

# Pose detection baseline

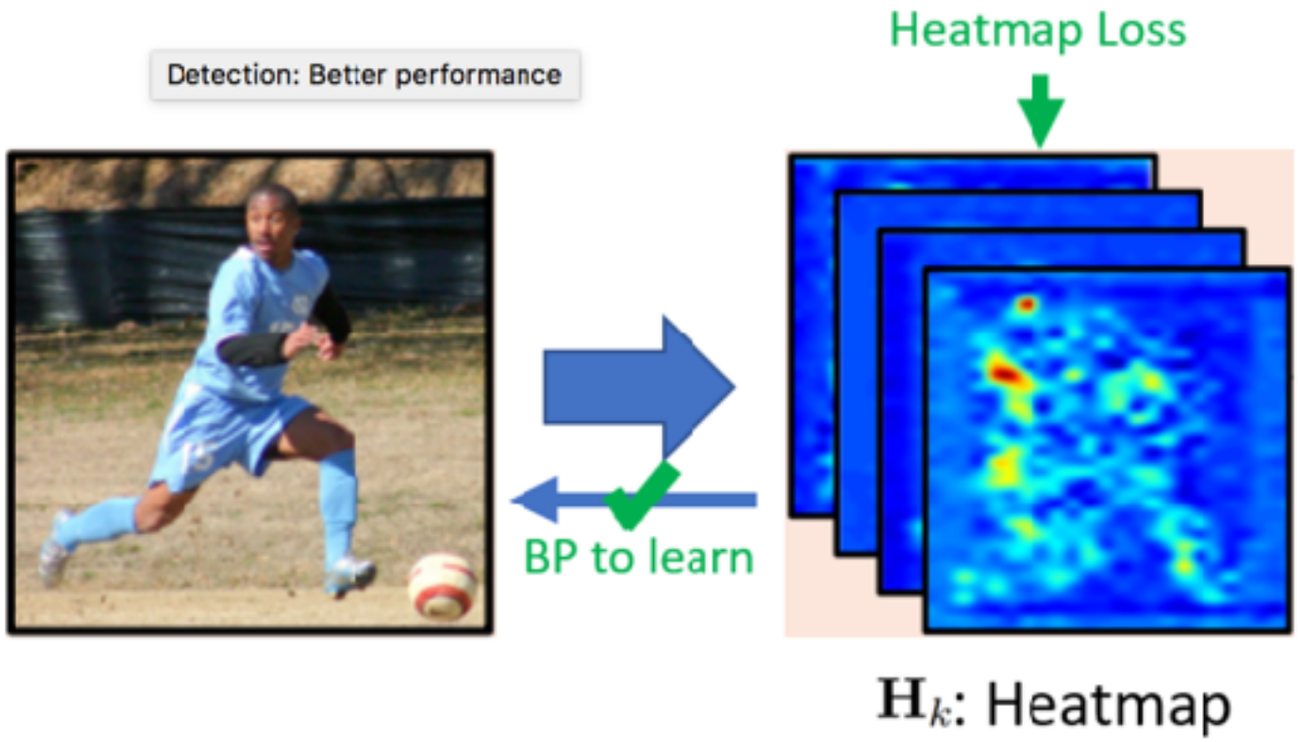



- estimate joint position as position of heatmap maximum

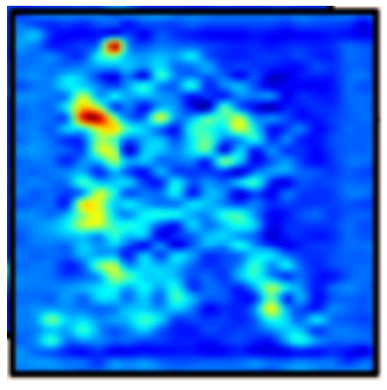



Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

# Pose detection baseline

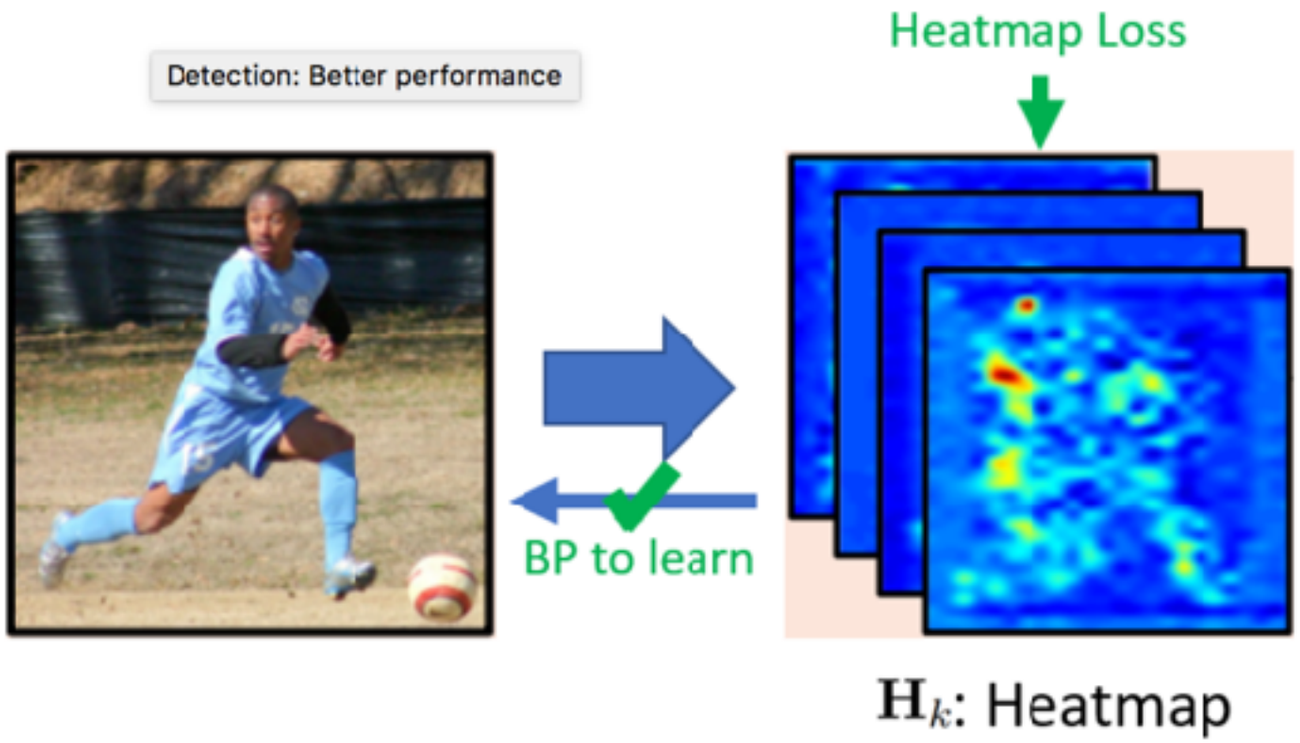



- estimate joint position as expected value in heatmap

$\mathbf{H}_k$ 

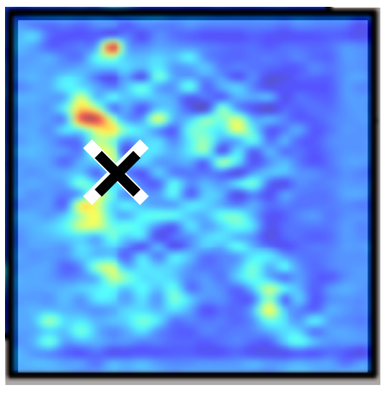
Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

# Pose detection baseline




- estimate joint position as expected value in heatmap

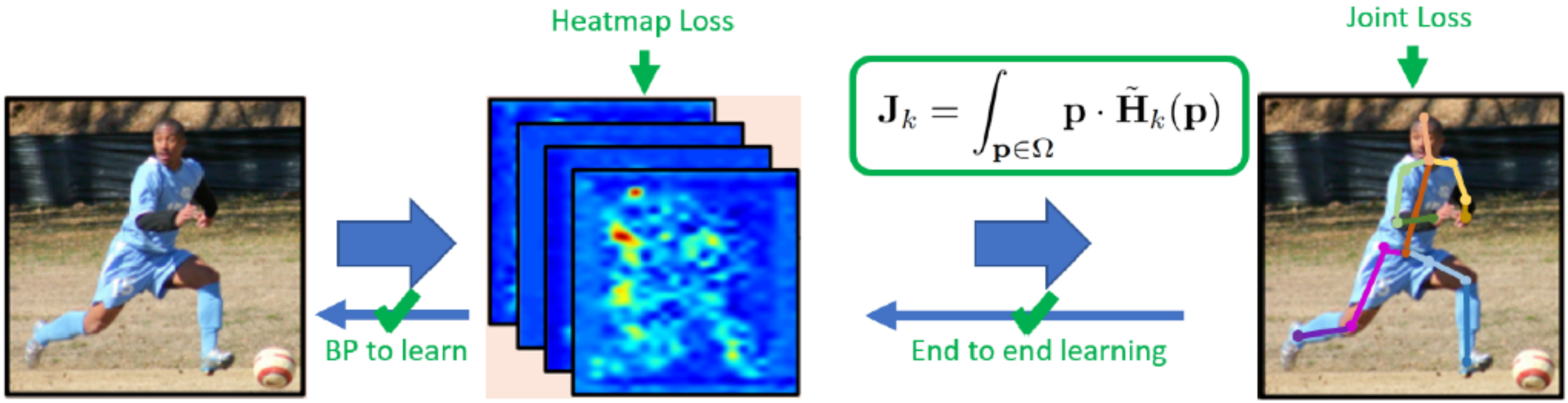
$\mathbf{H}_k$



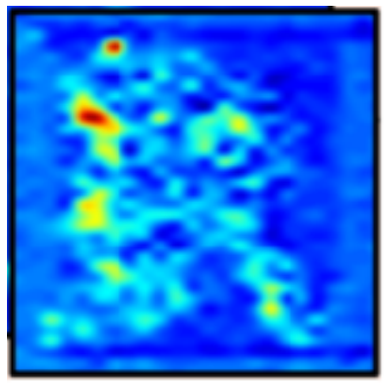
$$\mathbf{J}_k = \int_{\mathbf{p}\in\Omega} \mathbf{p} \cdot \tilde{\mathbf{H}}_k(\mathbf{p})$$

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

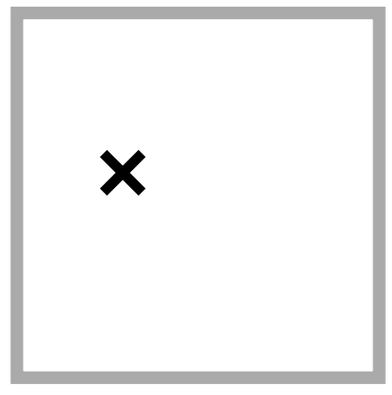
# Pose detection baseline




- estimate joint position as expected value in heatmap

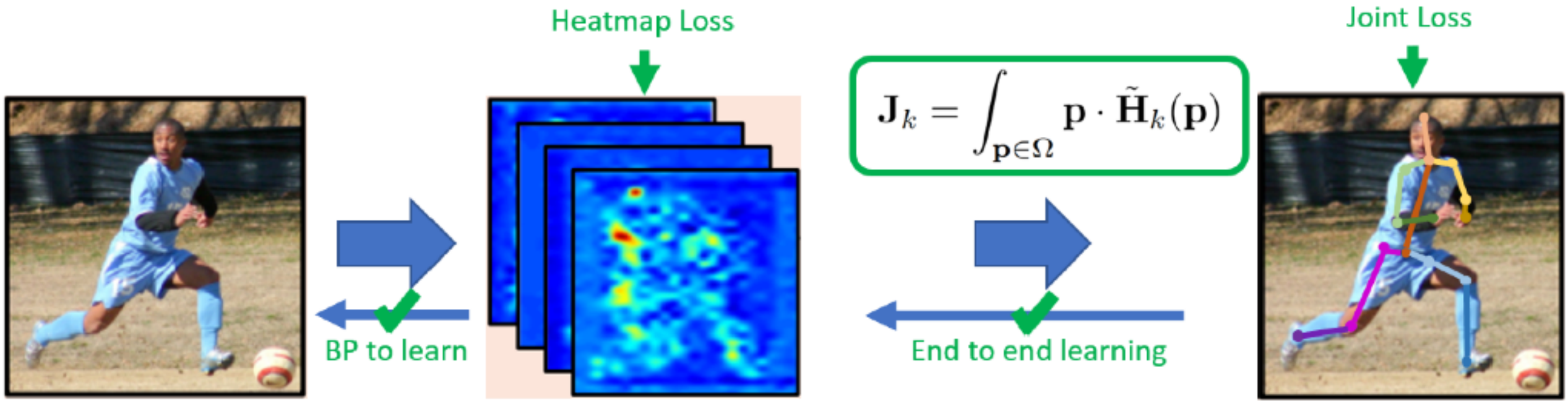
$\mathbf{H}_k$ 

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



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 $\mathbf{H}_k,\ k = 1 \ldots N$ (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

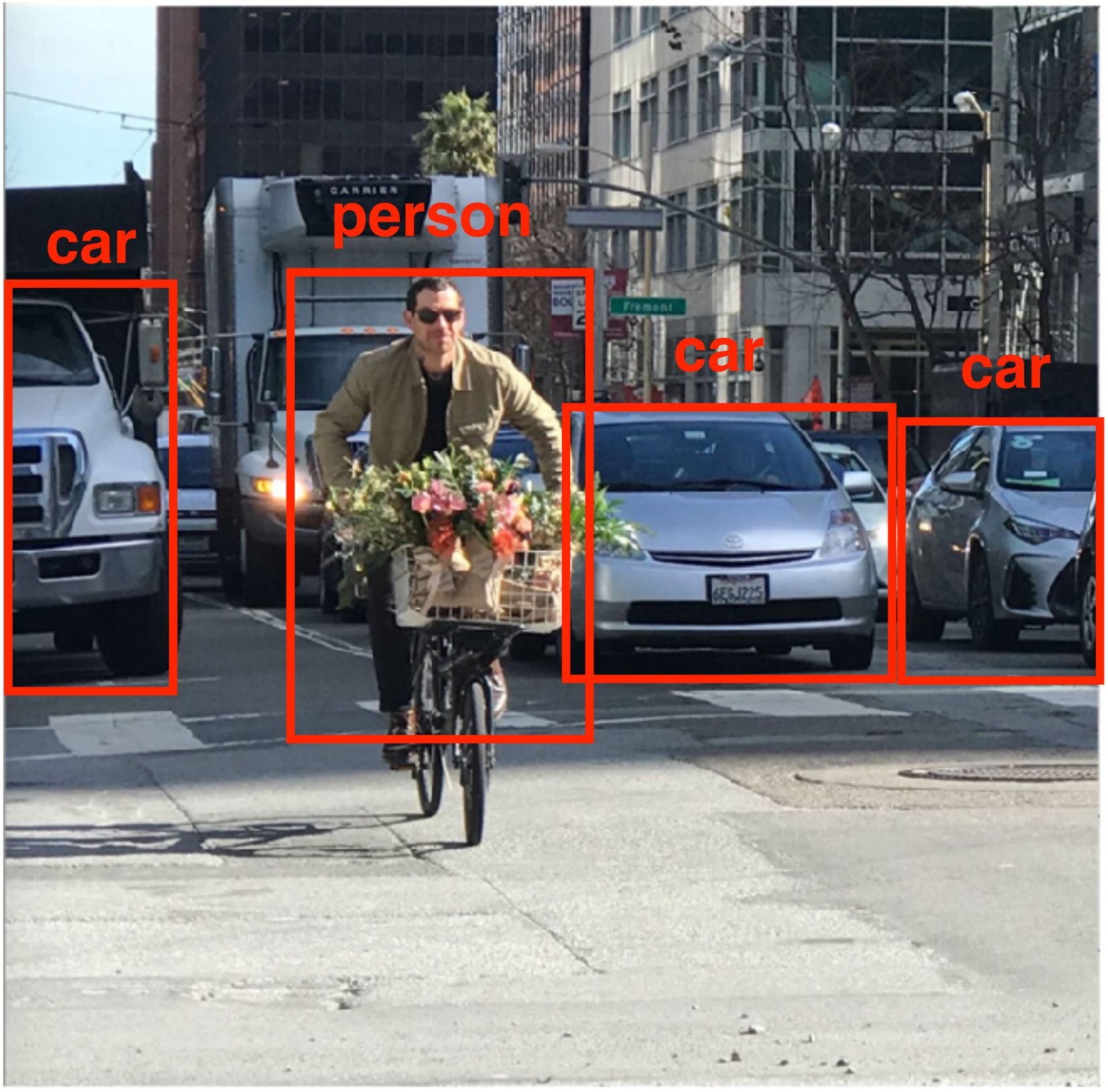
- PoseTrack benchmark a datasets
  https://posetrack.net
- Guler et al. (Facebook Research), DensePose
  https://arxiv.org/abs/1802.00434
  https://github.com/facebookresearch/Densepose
  https://www.youtube.com/watch?v=EMjPqgLX14A&feature=youtu.be
- Realtime Multi-Person 2D Human Pose Estimation using 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

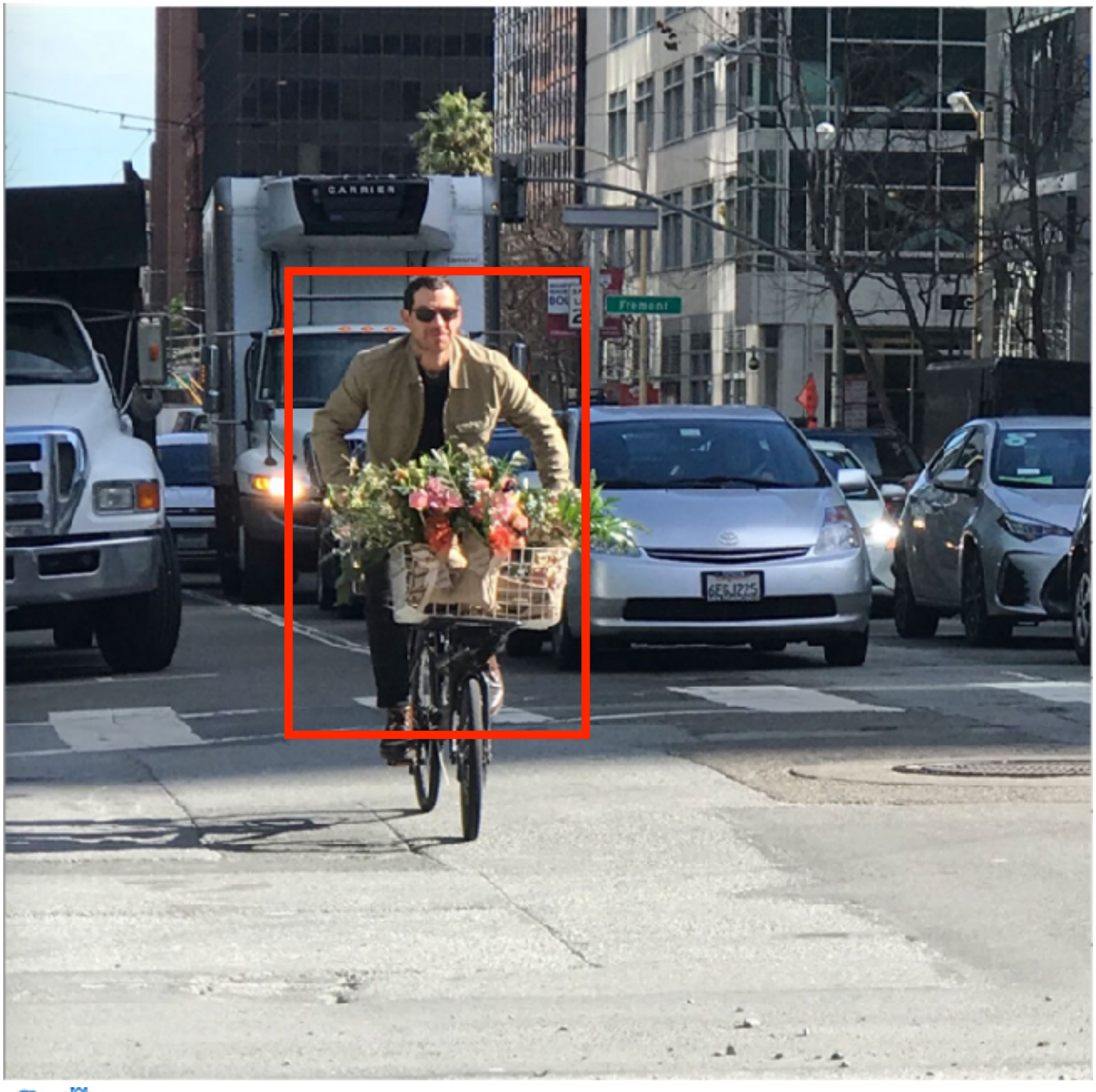
# Object detection

# Object detection



# Object detection

# Object detection

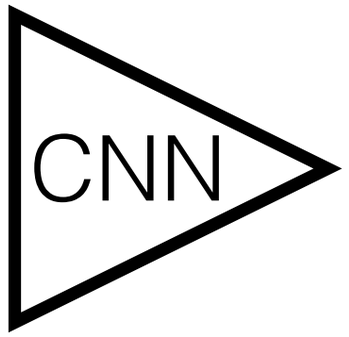


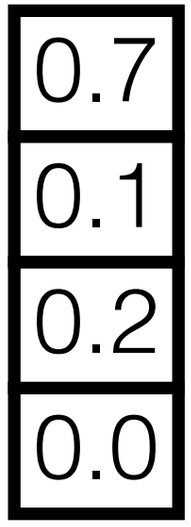

CNN

0.7
0.1
0.2
0.0

class: person

# Object detection



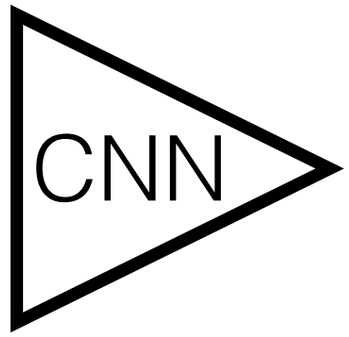
# Object detection

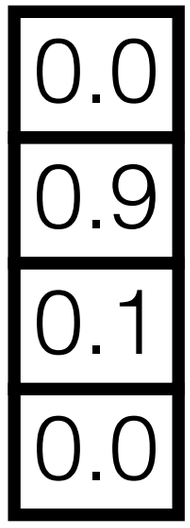
CNN

0.0
0.9
0.1
0.0

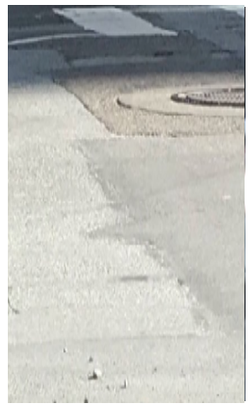
class: car

# Object detection



# Object detection

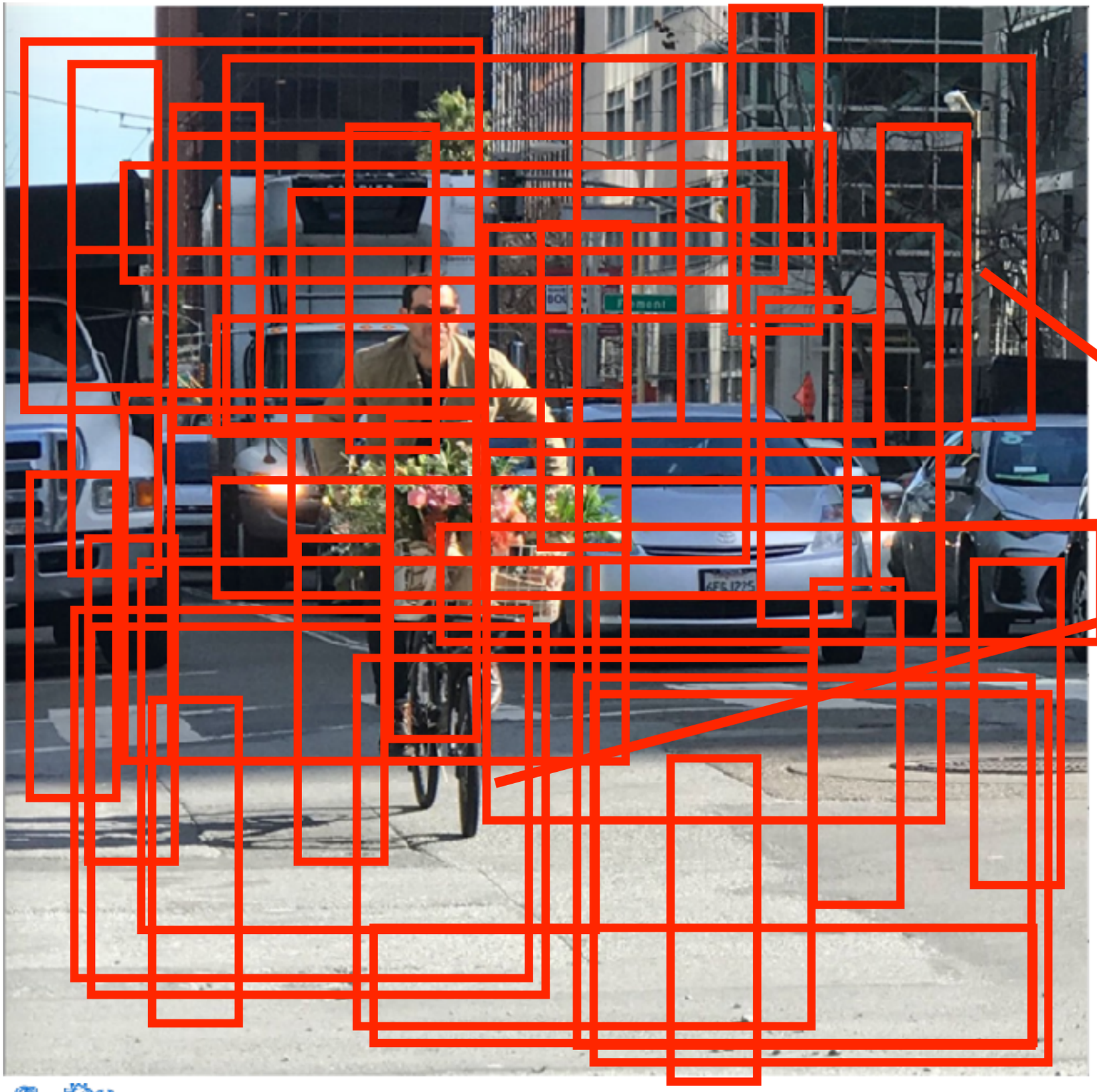



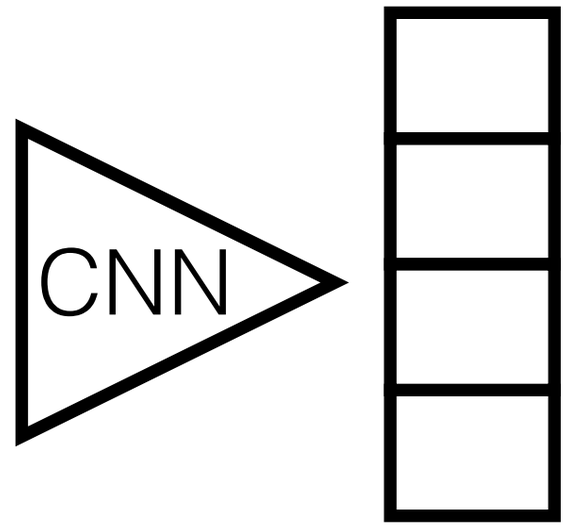
CNN

0.0
0.1
0.0
0.9

class: background

# Object detection



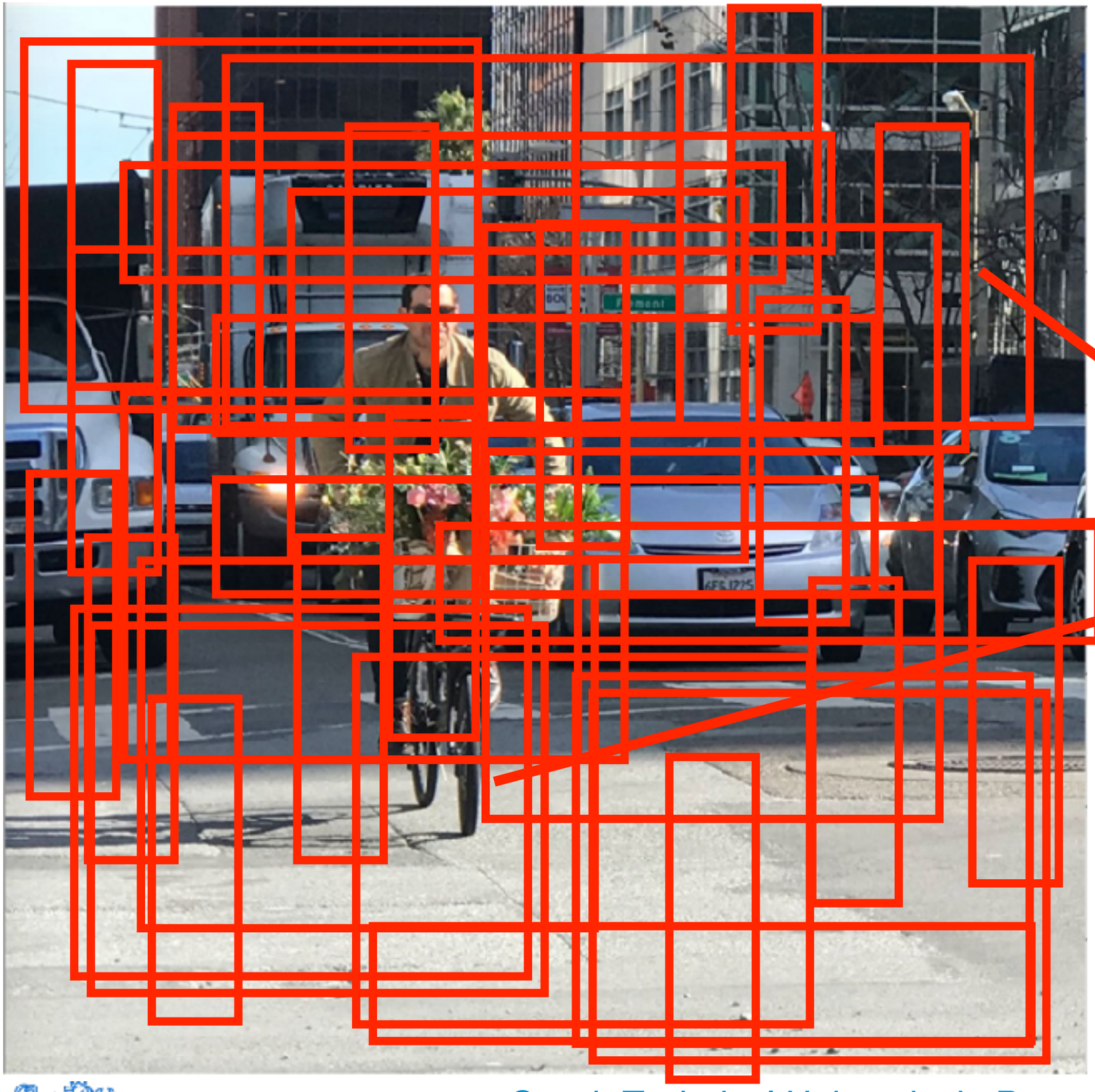


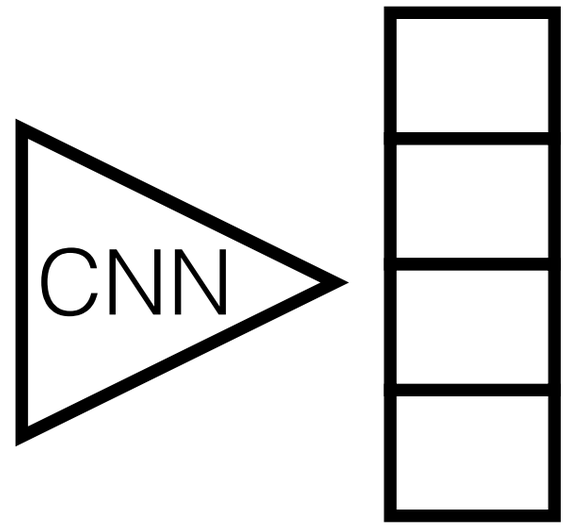

classify all rectangles

# 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**

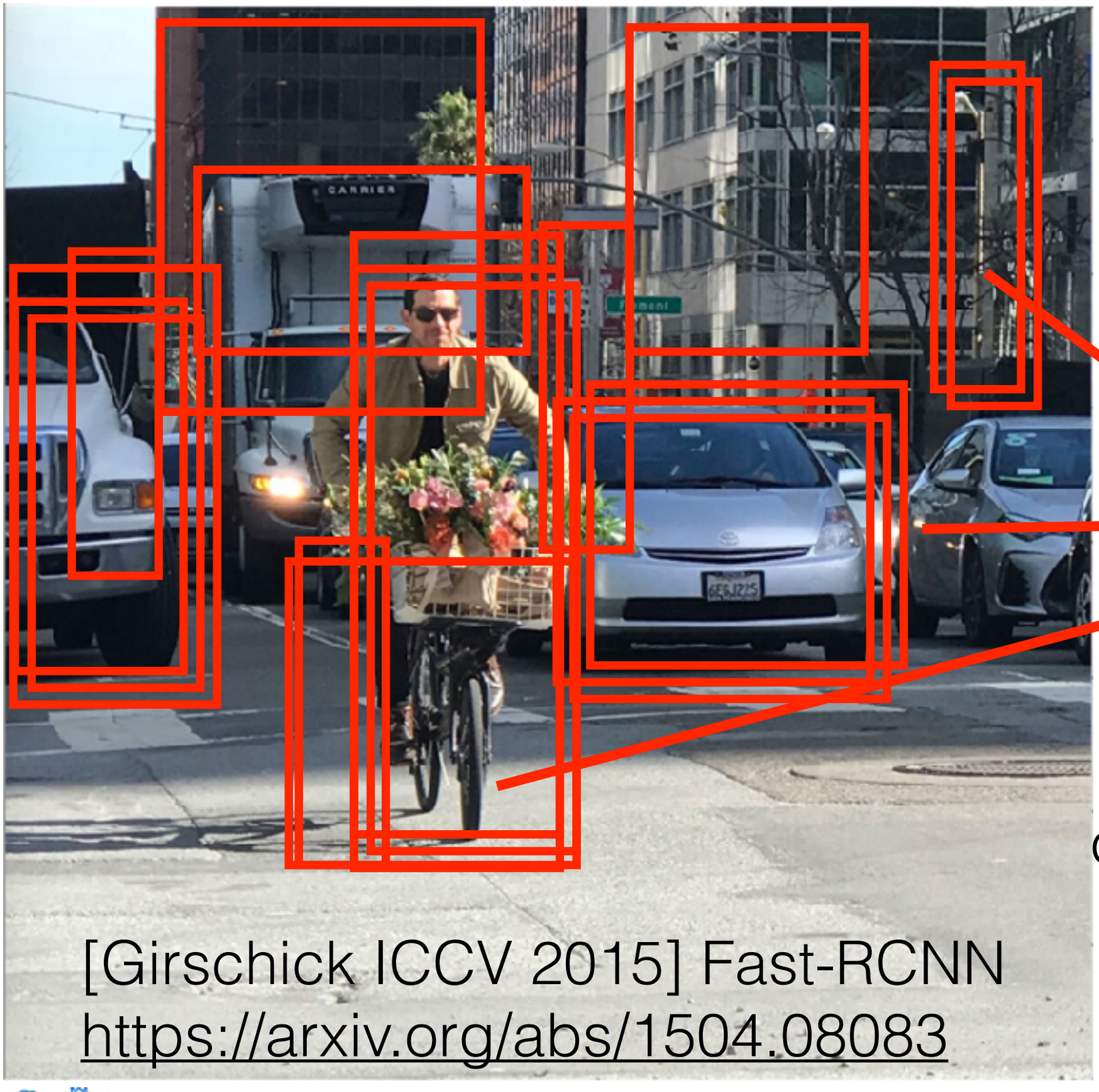
# Object detection






classify all rectangles

# Object detection



CNN

classify + align only 2k region proposals

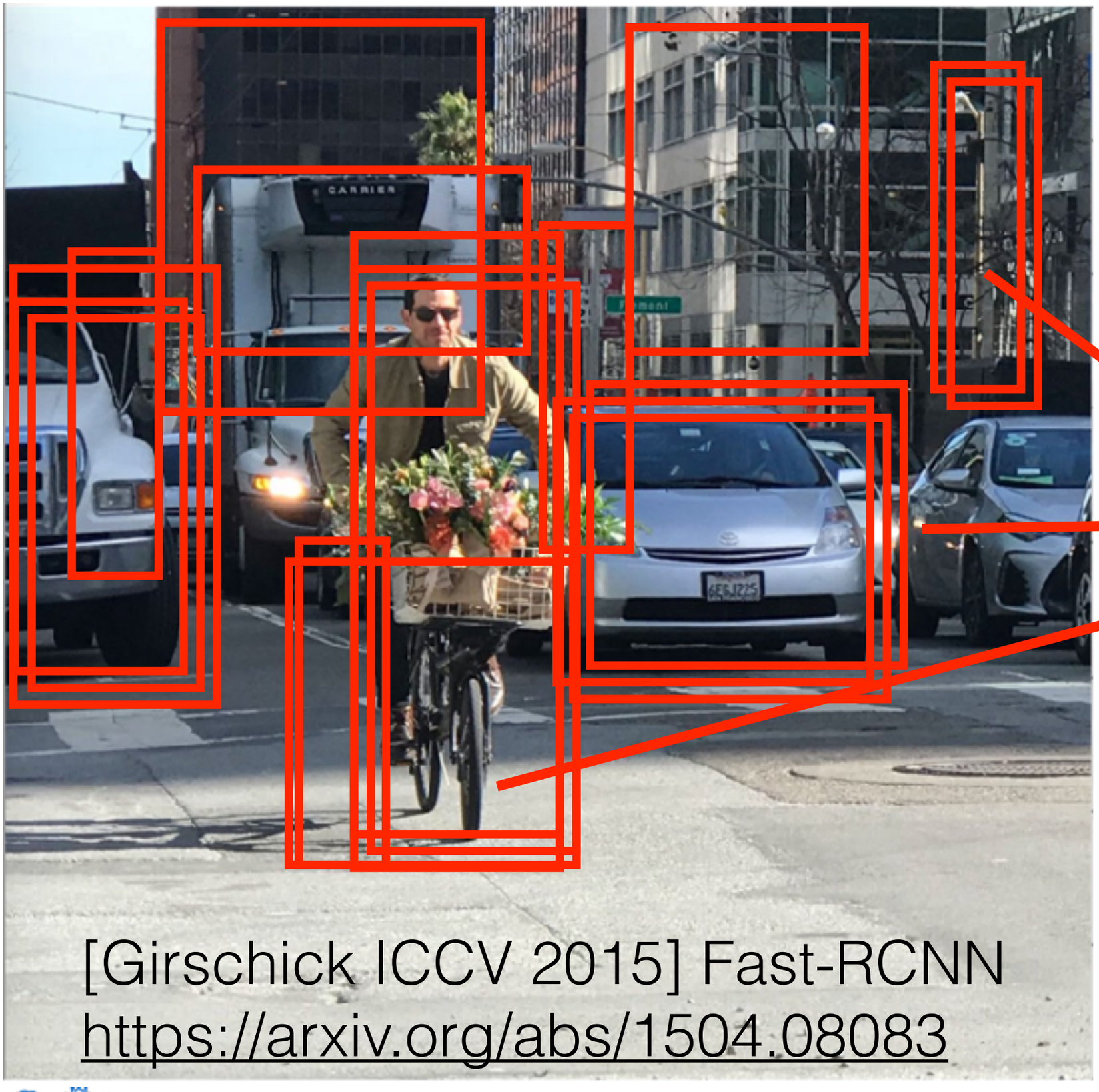
[Girschick ICCV 2015] Fast-RCNN
https://arxiv.org/abs/1504.08083

# Object detection

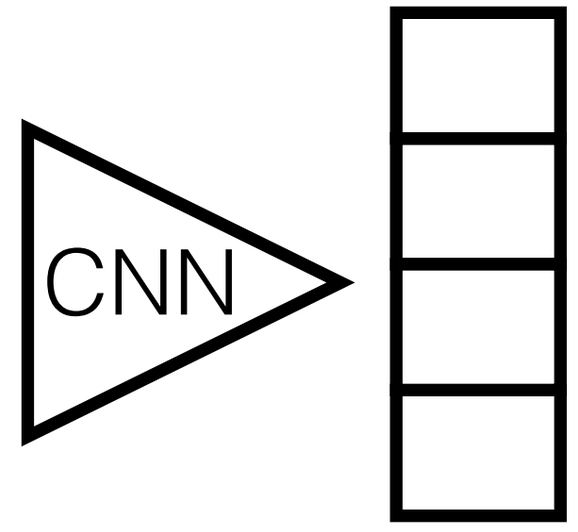
- Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  H x W x Aspect_Ratio x Scales x 0.001 sec = **months**
- Instead we can use elementary signal processing method to extract only 2k viable candidates:
  [Girschick ICCV 2015], Fast-RCNN
  https://arxiv.org/abs/1504.08083

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

# Object detection



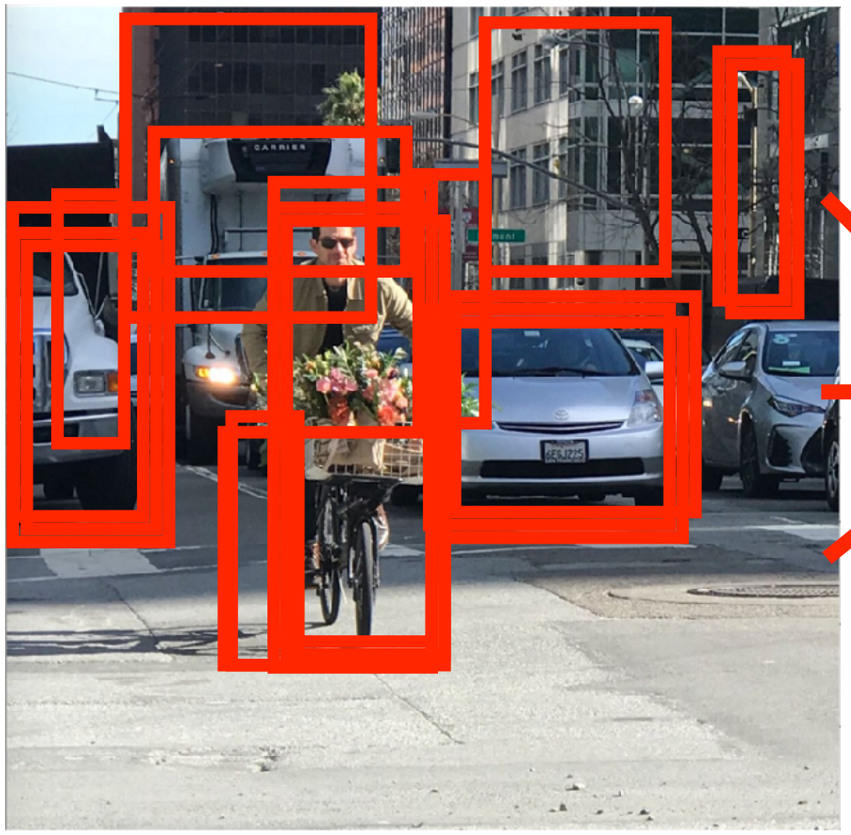
[Girschick ICCV 2015] Fast-RCNN
https://arxiv.org/abs/1504.08083




The search for region proposals is computational bottleneck !!!

# Object detection

RPN



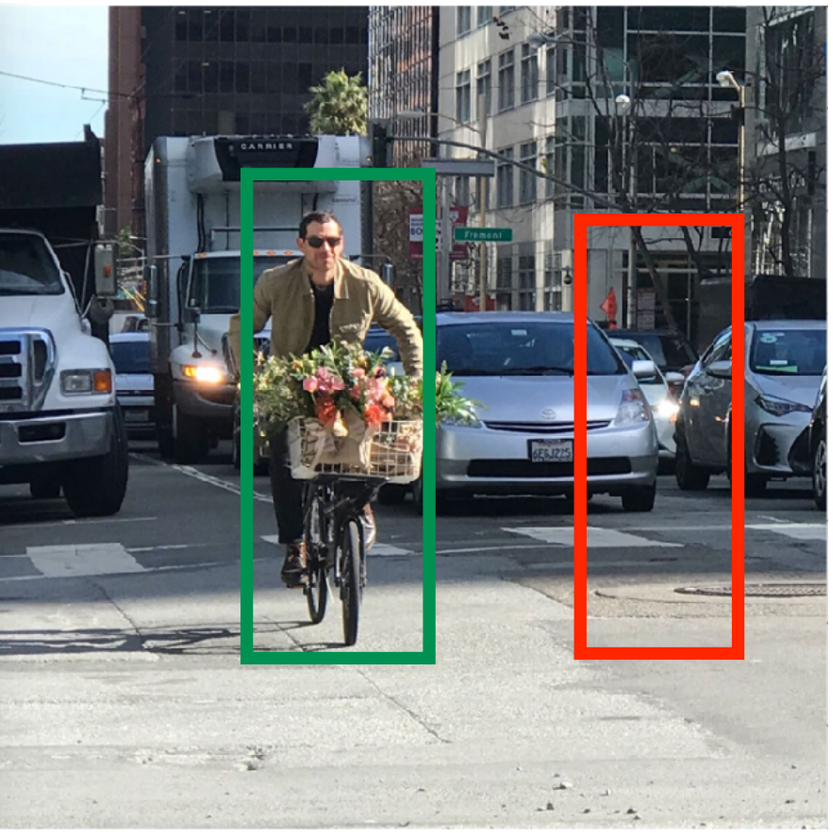
CNN

region proposal net
(output: 2k proposals)

classification
+
alignment
net

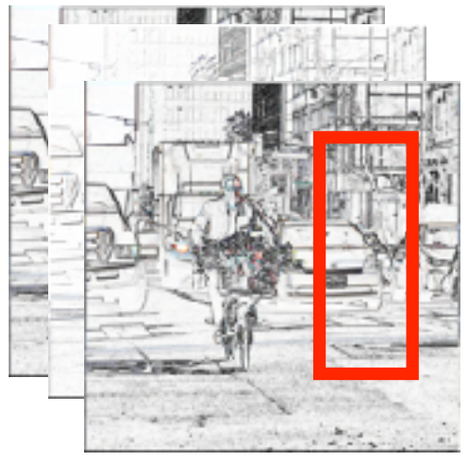
Faster-RCNN https://arxiv.org/abs/1506.01497

# Region Proposal Net (RPN)



ground truth

RPN

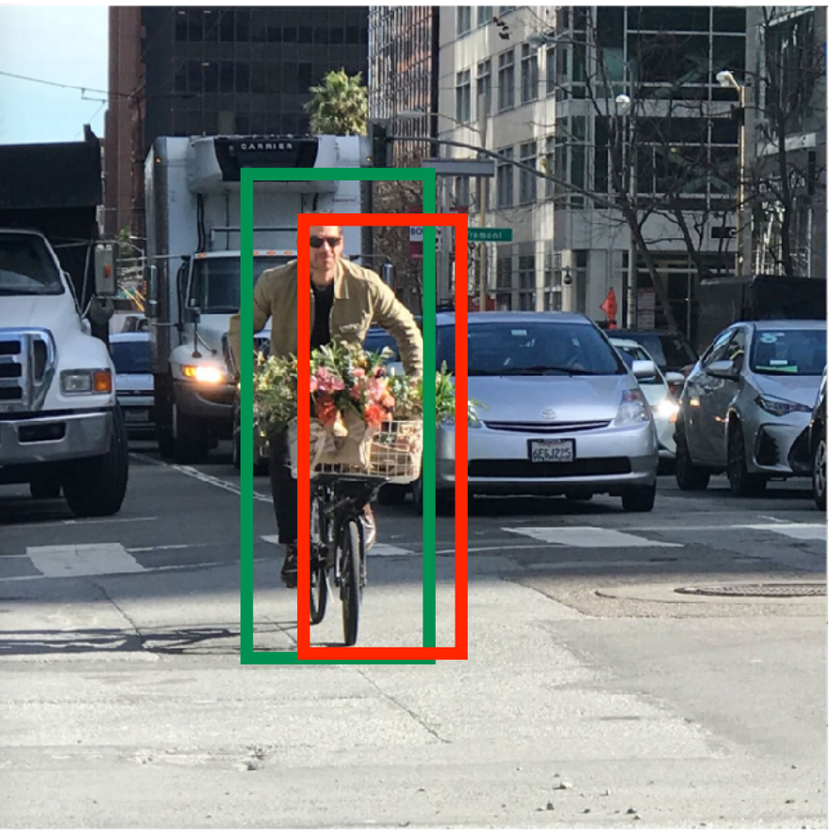


low resolution
feature map

- 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

RPN

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

RPN

low resolution
feature map

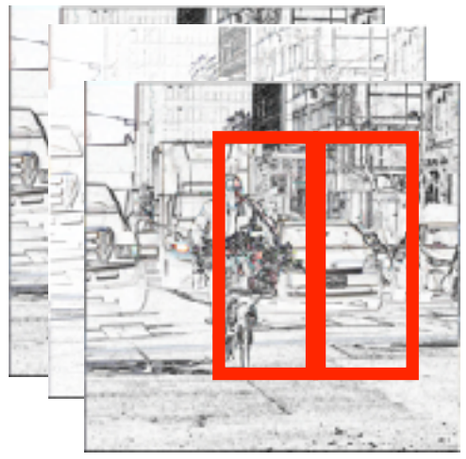
- generate bounding which corresponds to discrete positions in low resolution feature maps and measure IoU

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

# Region Proposal Net (RPN)

ground truth

RPN



low resolution
feature map

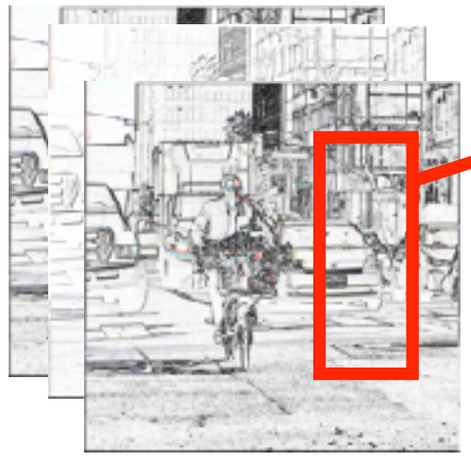
- generate bounding which corresponds to discrete positions in low resolution feature maps and measure IoU
- bbs with IoU>0.7 are objects,
  bbs with IoU<0.3 not objects

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

# Region Proposal Net (RPN)

ground truth

RPN



low resolution
feature map

center_x
center_y
width
height
object

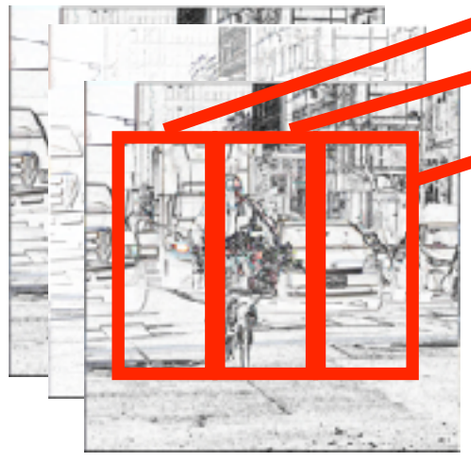
- for each discrete bb RPN predicts:
  - its “alignment with gt” (regression loss)
  - its “objectness" (classification loss)

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

# Region Proposal Net (RPN)

ground truth

RPN



low resolution
feature map

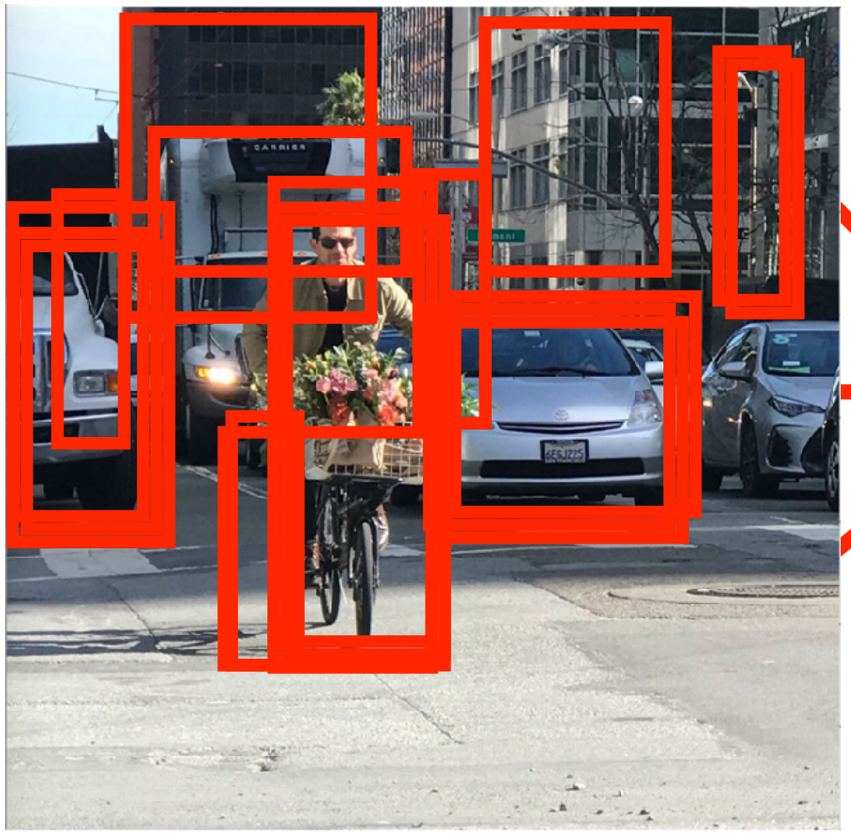
center_x
center_y
width
height
object

- 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

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


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:

CNN

| | |
|---|---|
| 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)

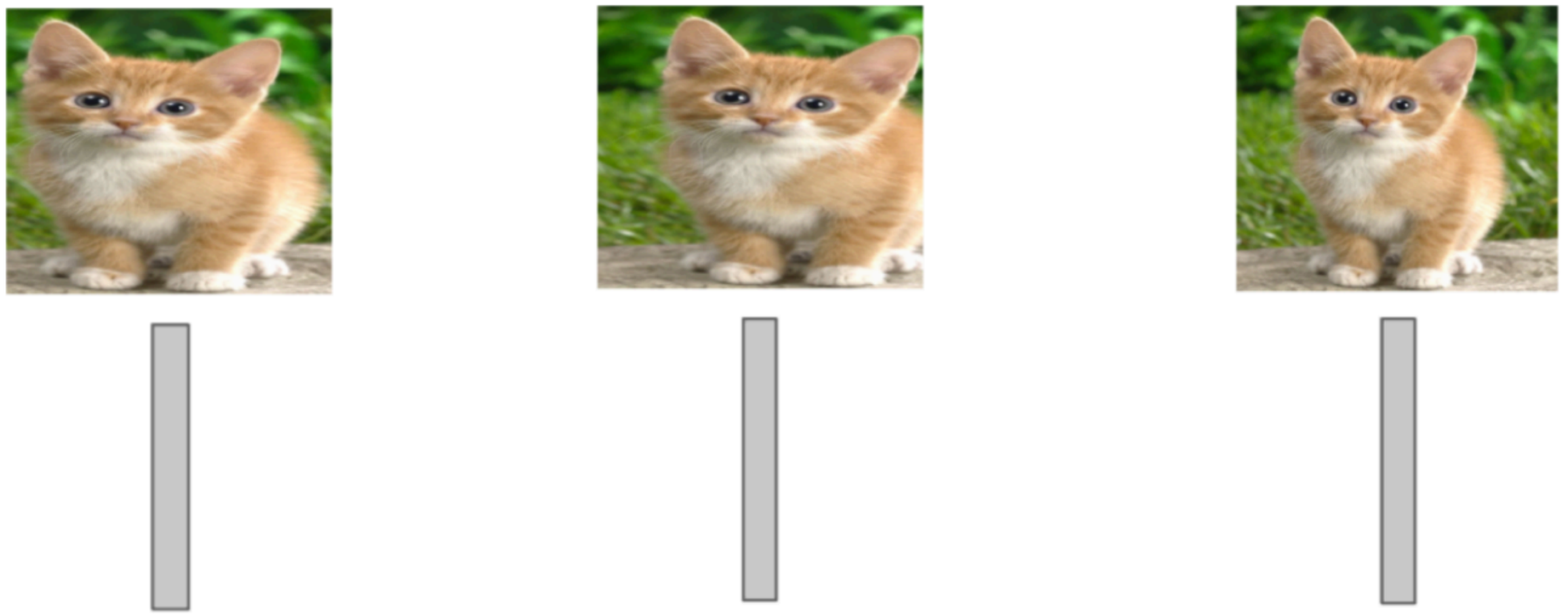
classification
+
alignment
net

Save computational power by reusing RPN feature maps

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

# Object detection

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



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

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

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

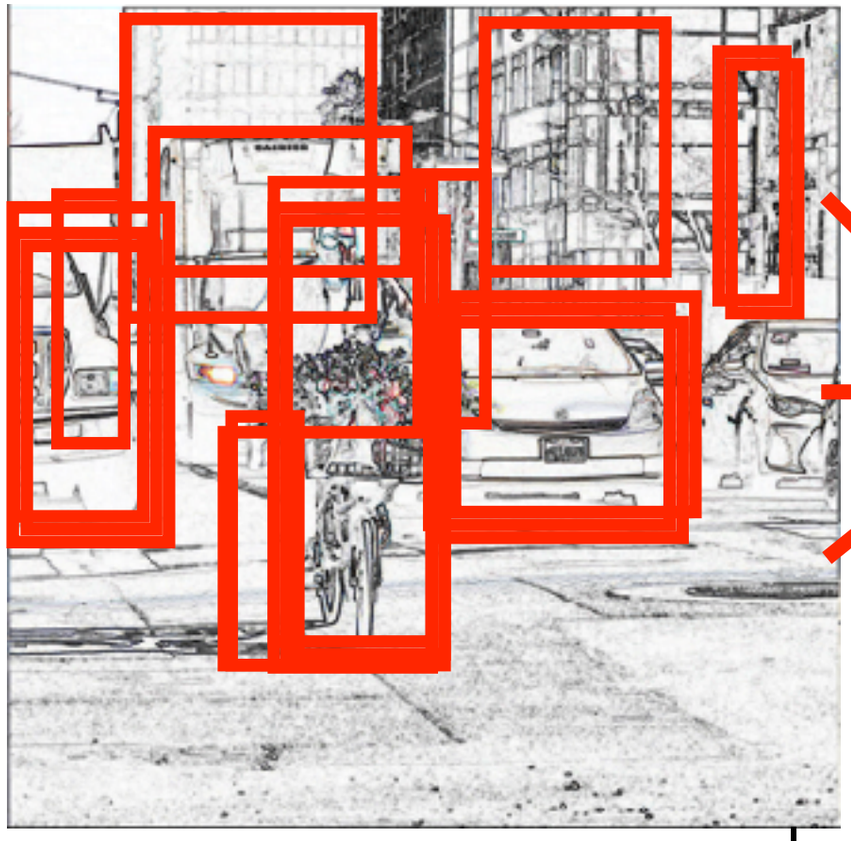
Faster-RCNN https://arxiv.org/abs/1506.01497

# 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

RPN



CNN

region proposal net
(output: 2k proposals)

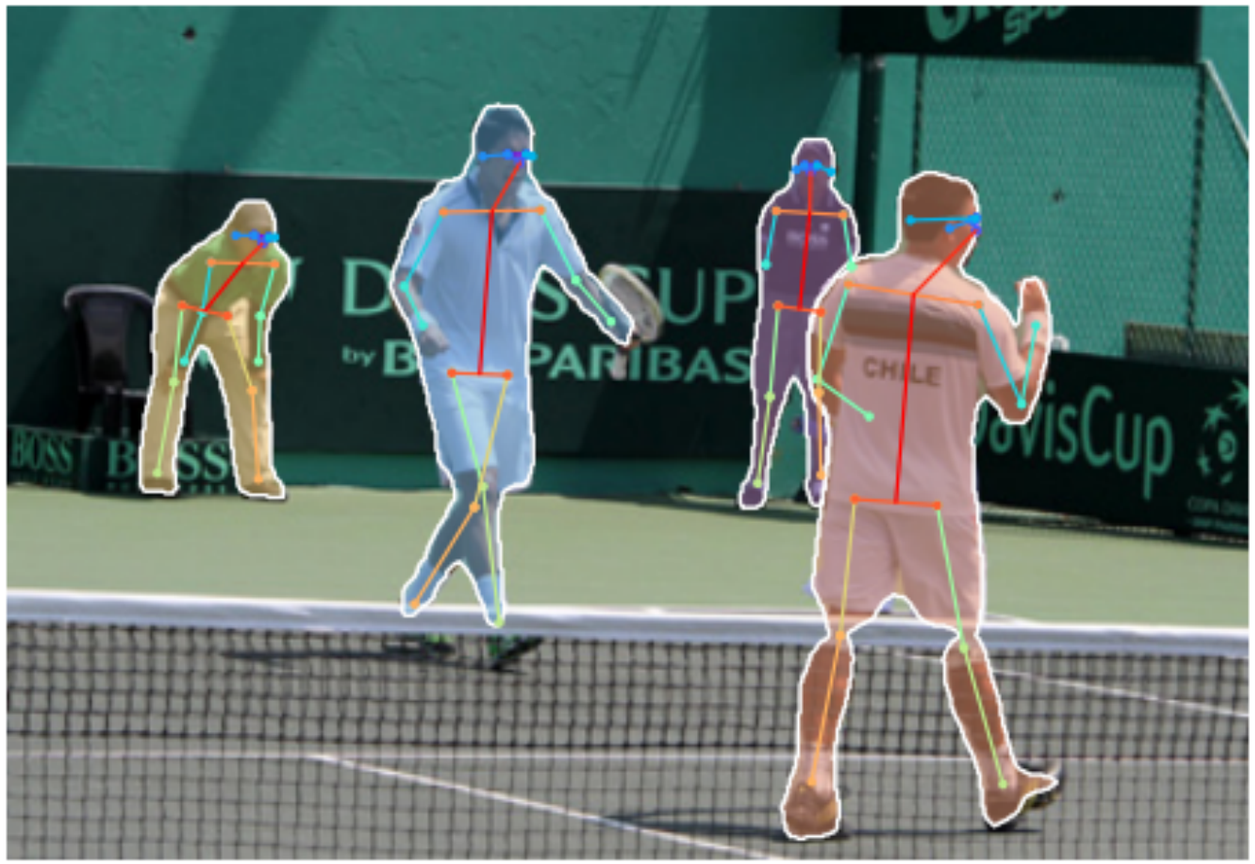
[He et al CVPR 2017] Mask-RCNN
https://arxiv.org/abs/1703.06870

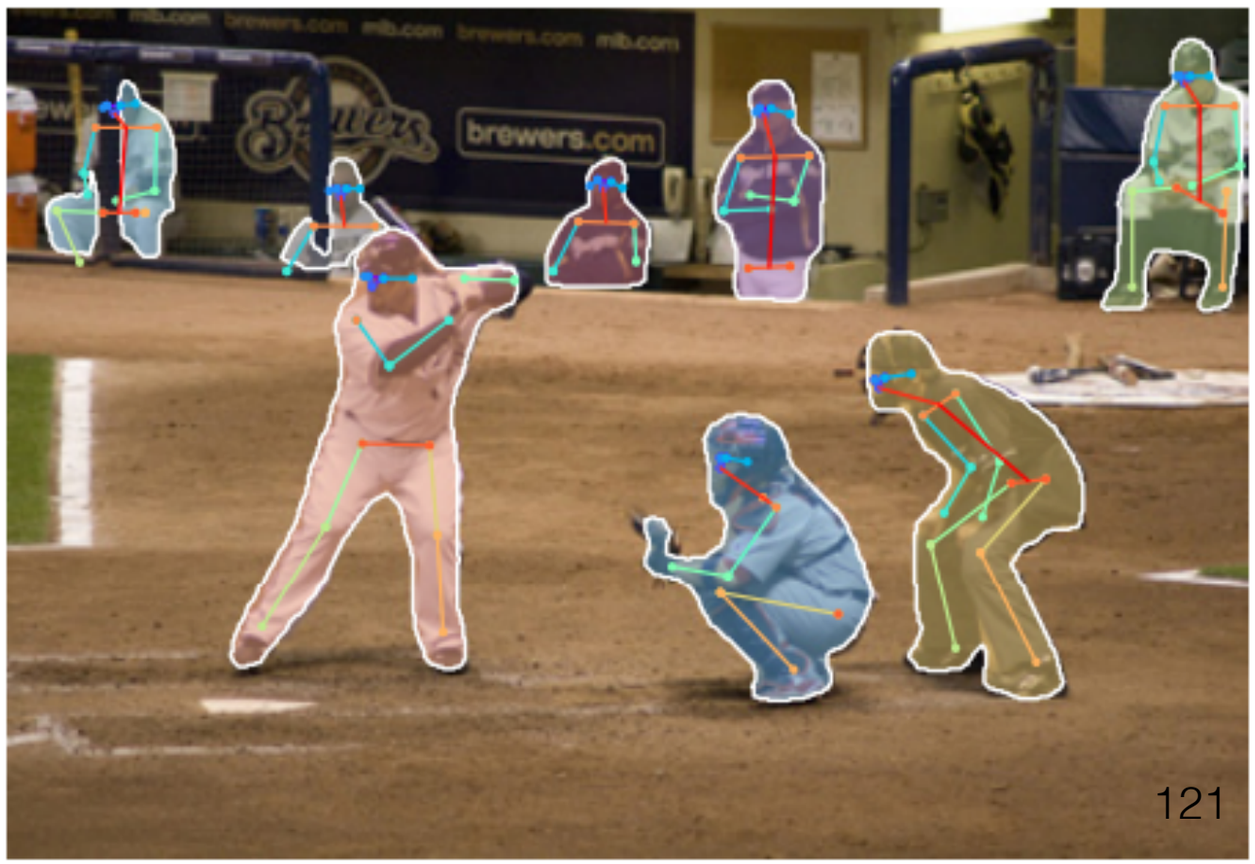
classification
+
alignment
+
segmentation mask
+
pose regression

# Mask RCNN - results

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

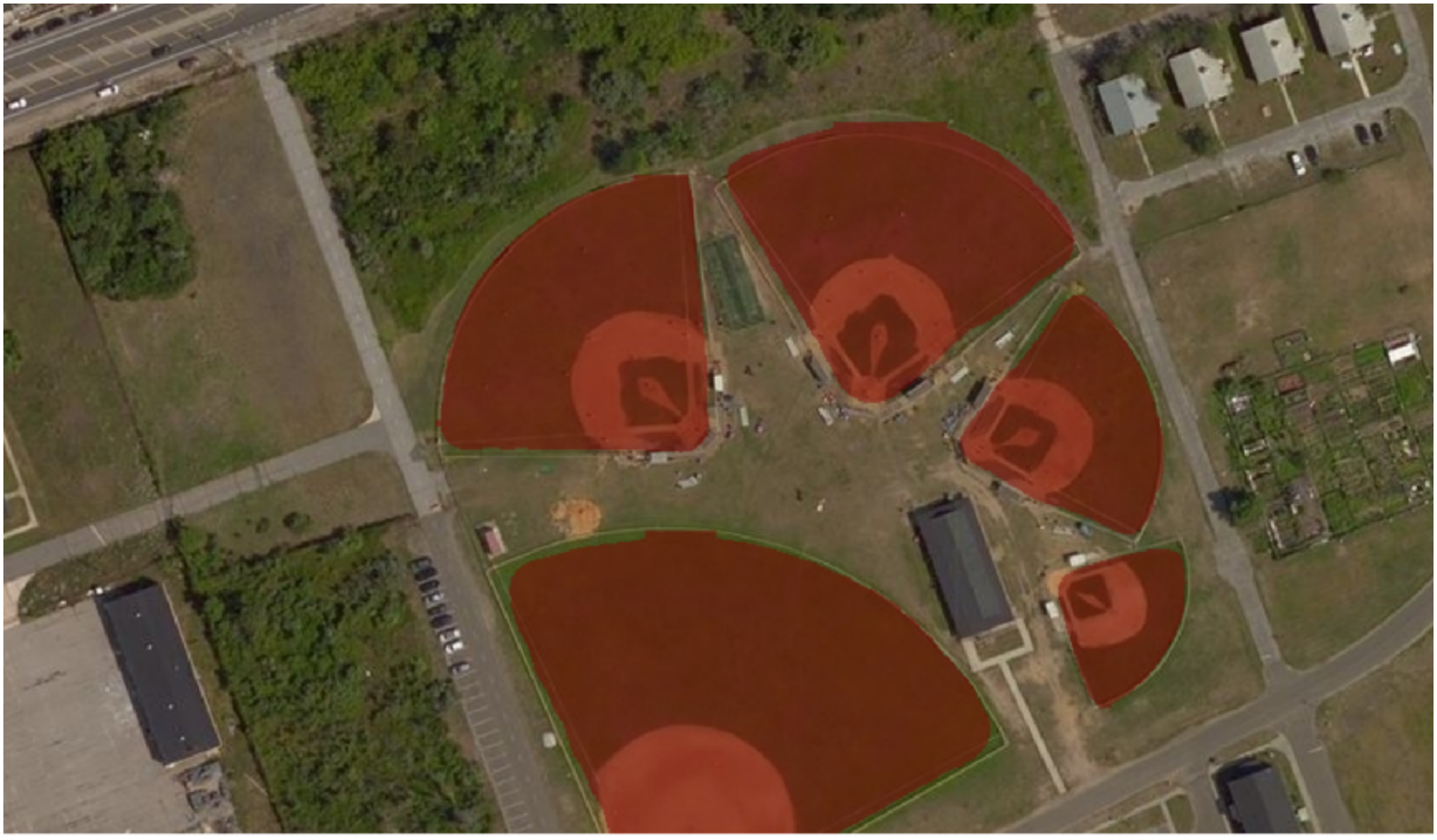
# Mask RCNN - results

# Mask RCNN - applications

## Playground detection for OpenStreetMaps



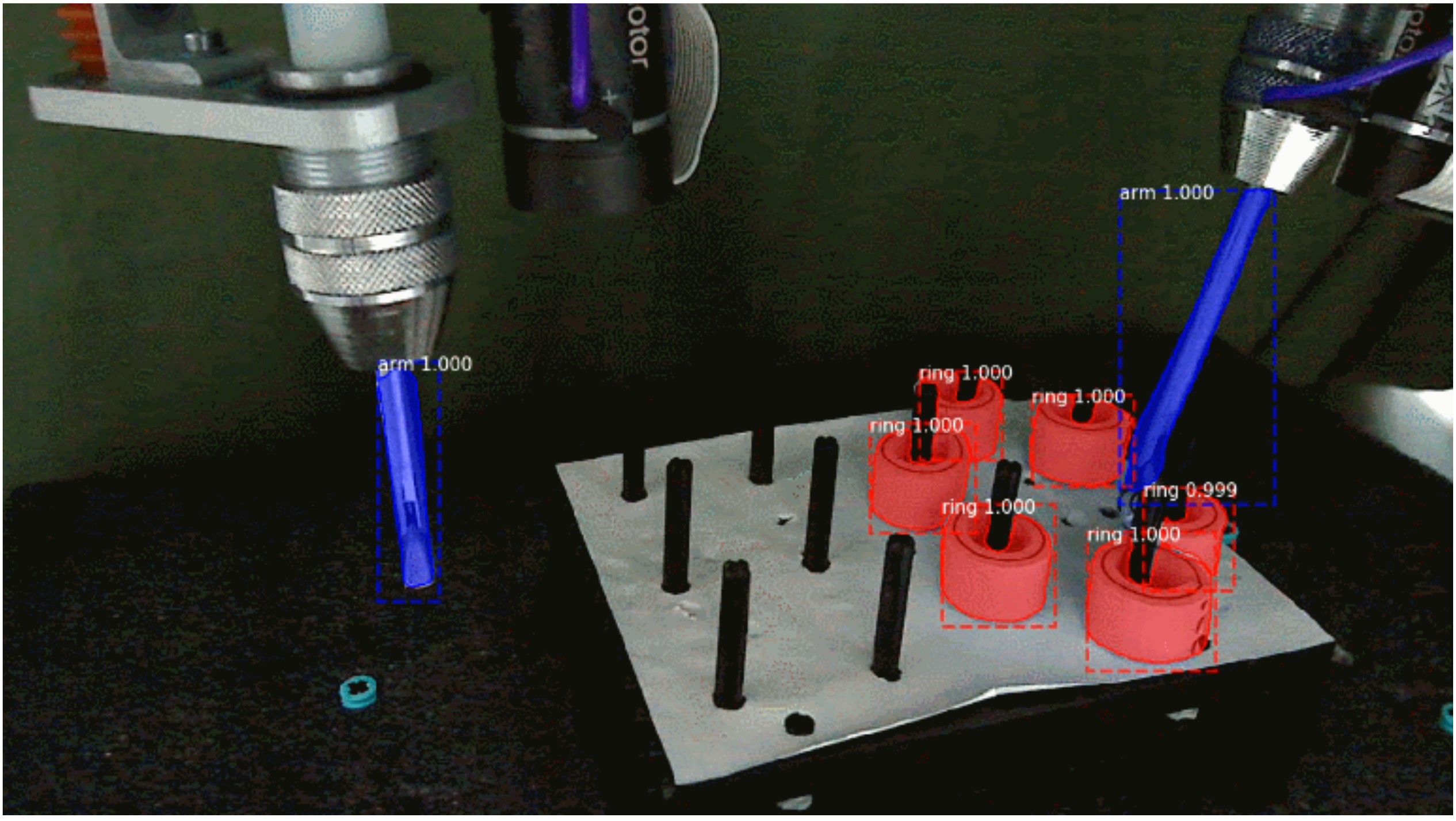
https://github.com/jremillard/images-to-osm

# Mask RCNN - applications
# Mapping challenge


https://github.com/crowdAI/crowdai-mapping-challenge-mask-rcnn

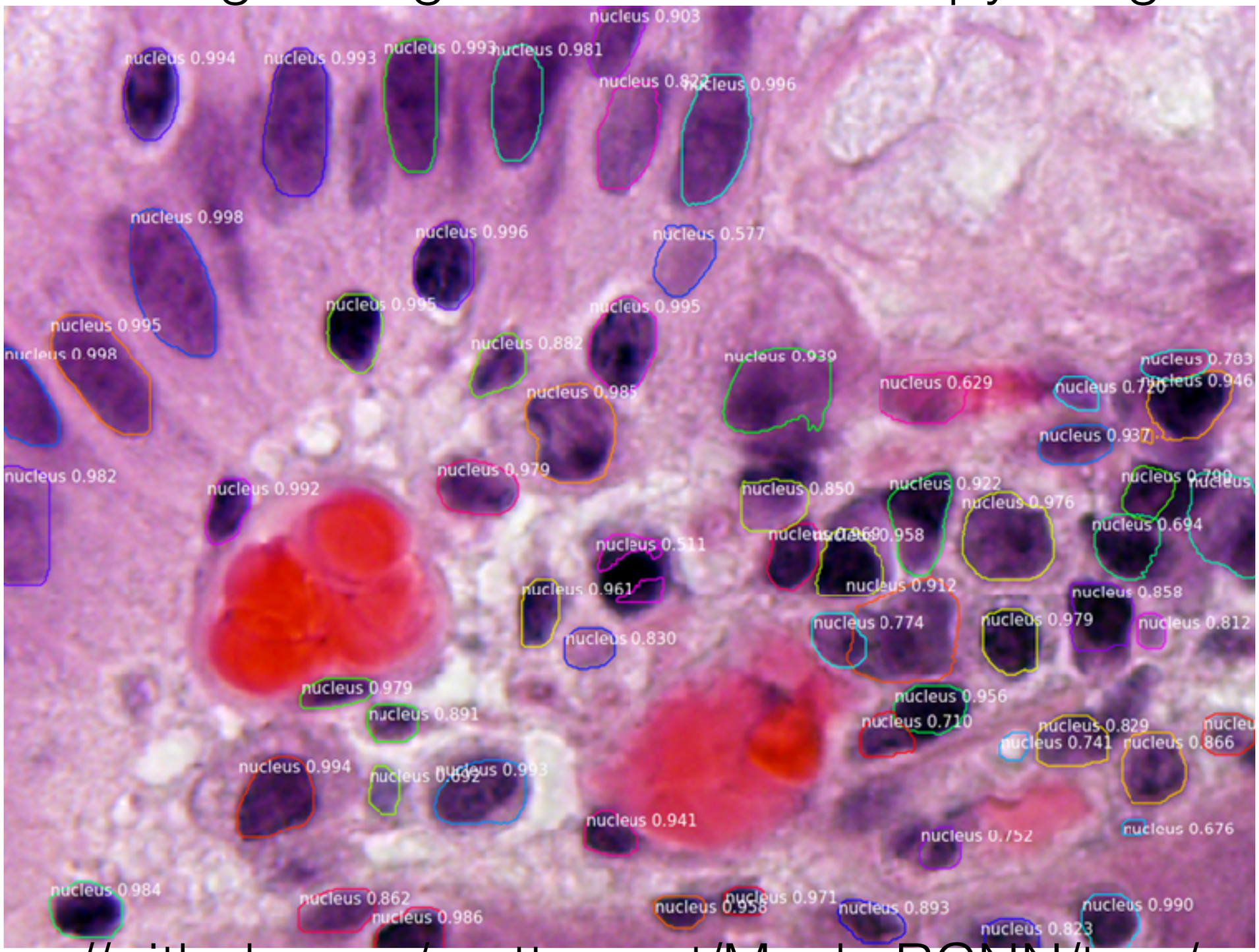
# Mask RCNN - applications

## Detection and segmentation for surgery robot



# Mask RCNN - applications

## Distinguishing nuclei in microscopy images




https://github.com/matterport/Mask_RCNN/tree/master/samples/nucleus

# Mask RCNN - applications


https://github.com/huuuuusy/Mask-RCNN-Shiny

# Object detection

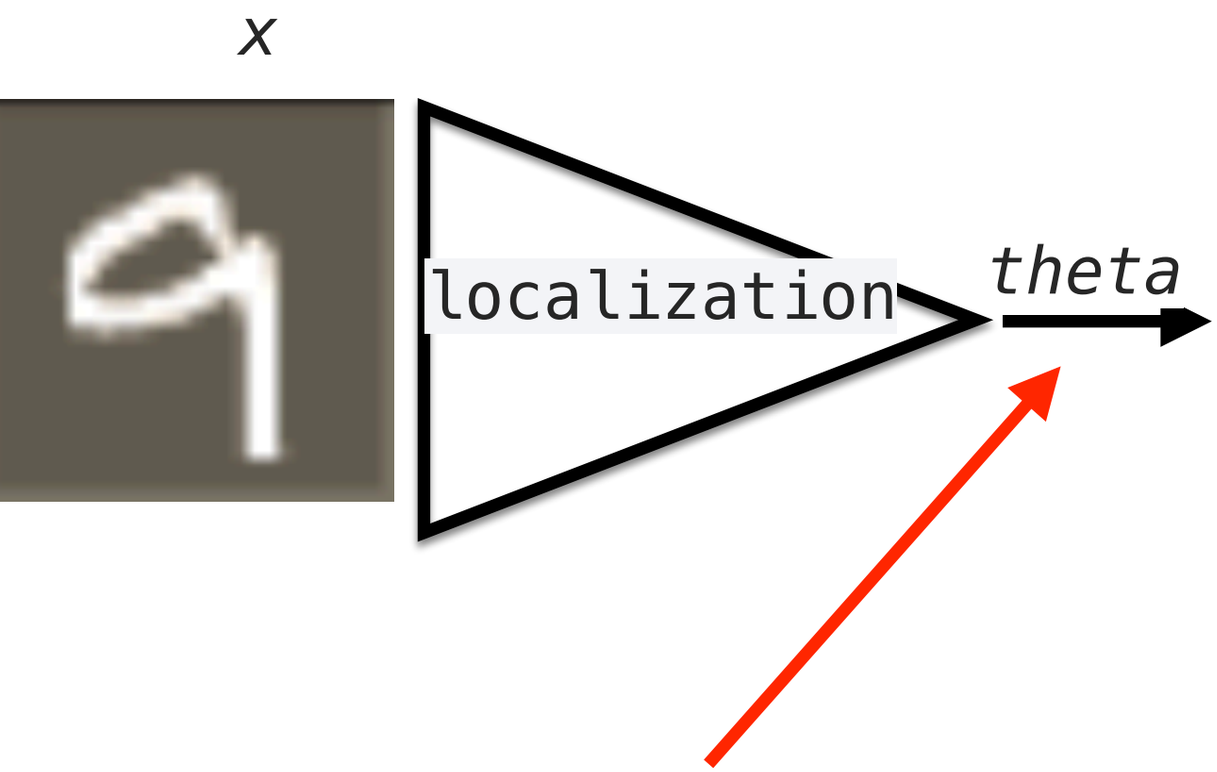
- Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  H x W x Aspect_Ratio x Scales x 0.001 sec = **months**
- Instead we can use elementary signal processing method to extract only 2k viable candidates:
  [Girschick ICCV 2015], Fast-RCNN
  https://arxiv.org/abs/1504.08083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

# Outline

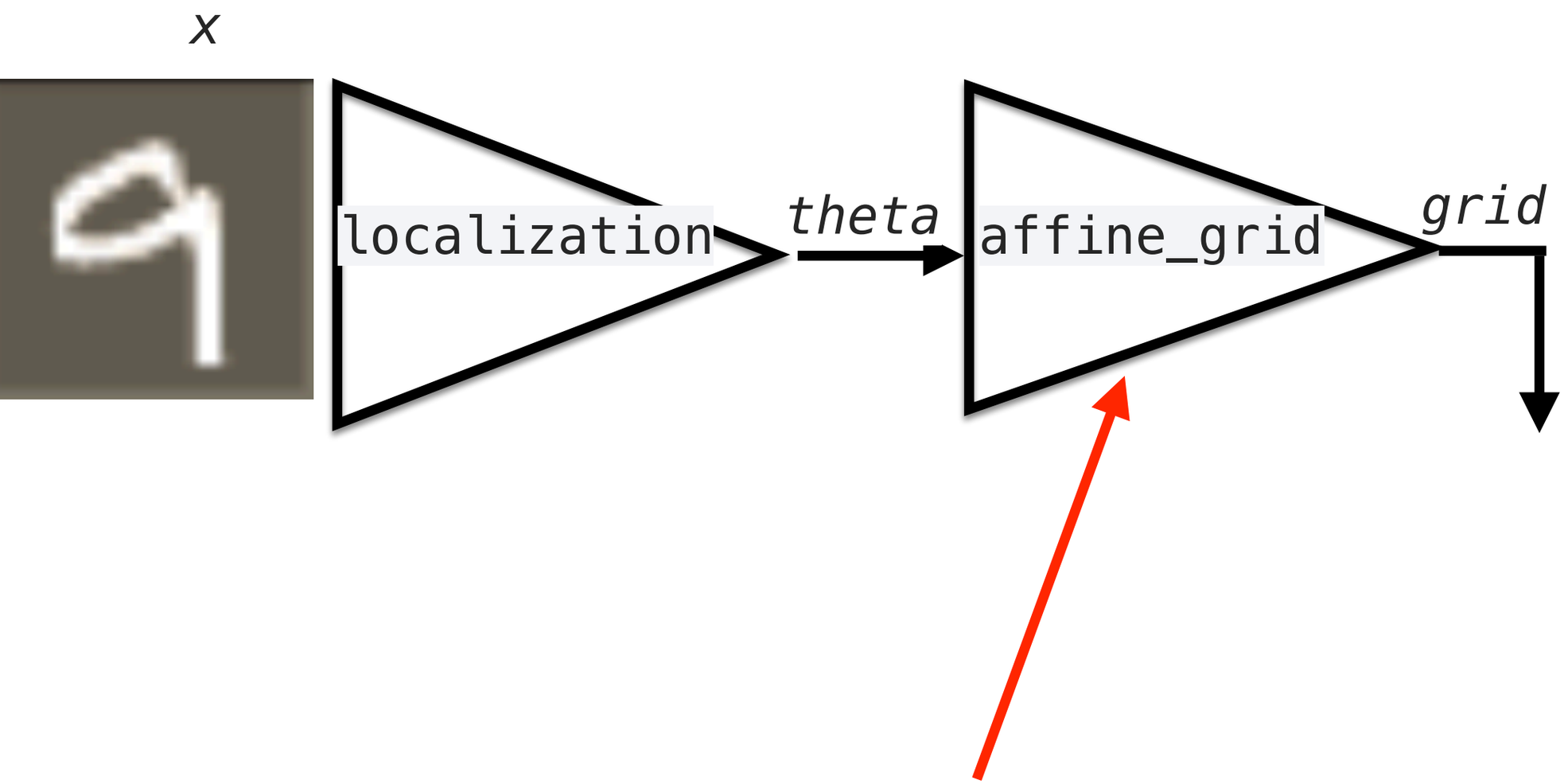
- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Spatial Transformer networks
- Architectures of feature matching networks

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



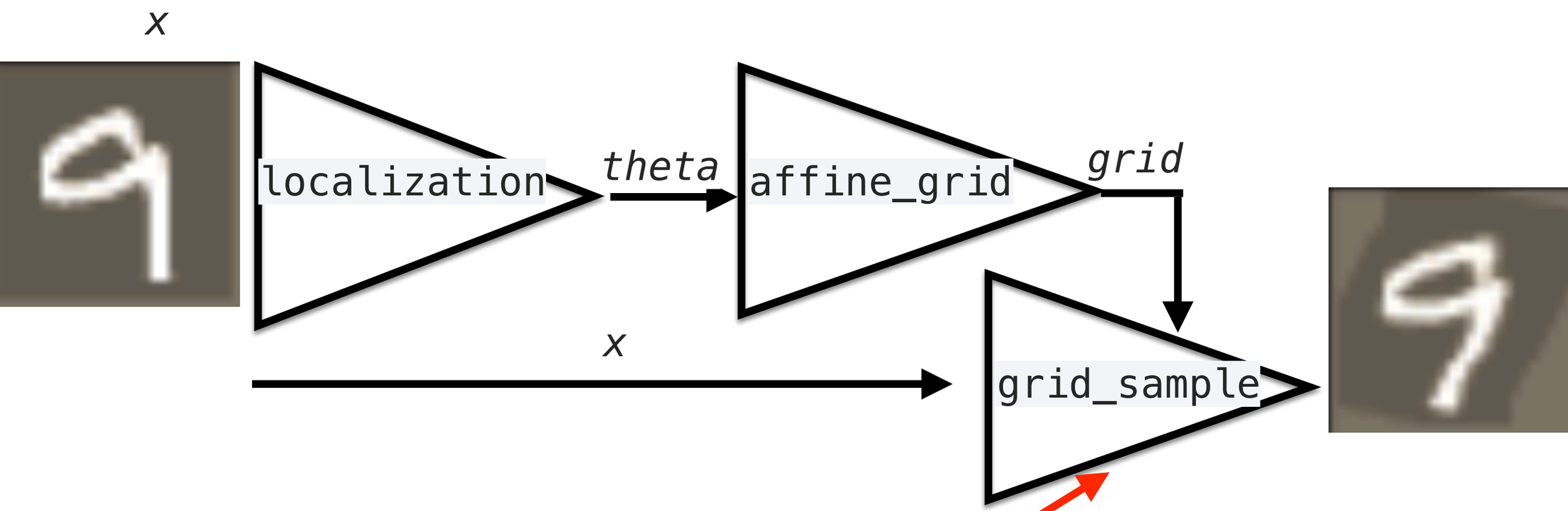

$$[\mathbf{R}\ \mathbf{t}] \in \mathcal{R}^{2\times 3}$$

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




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

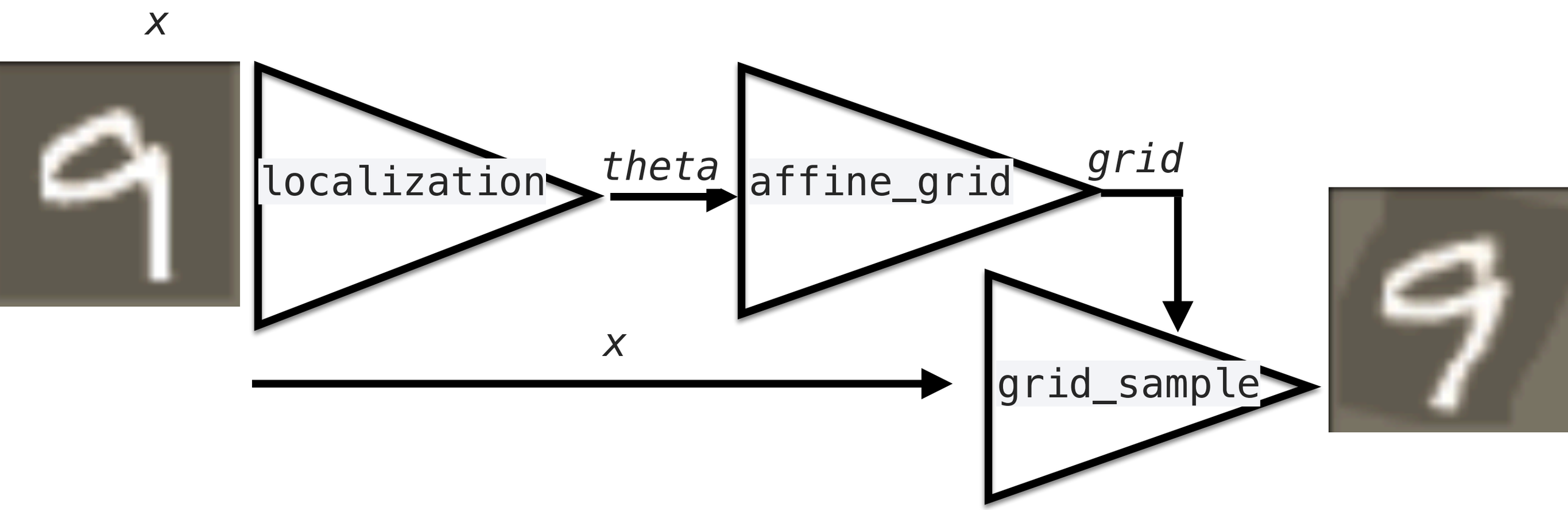
# Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf




```
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)
```

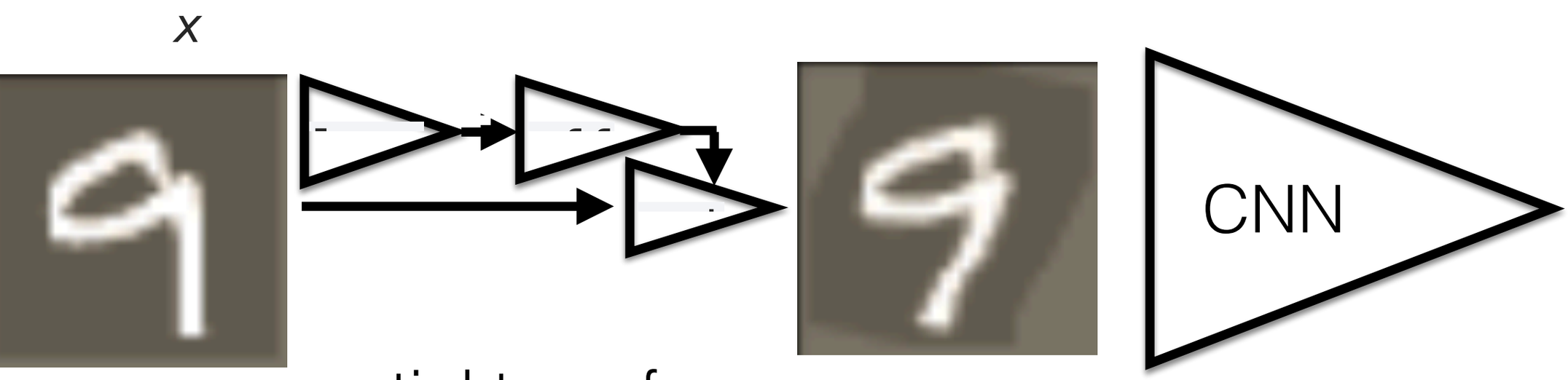
# Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf




```
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)
```

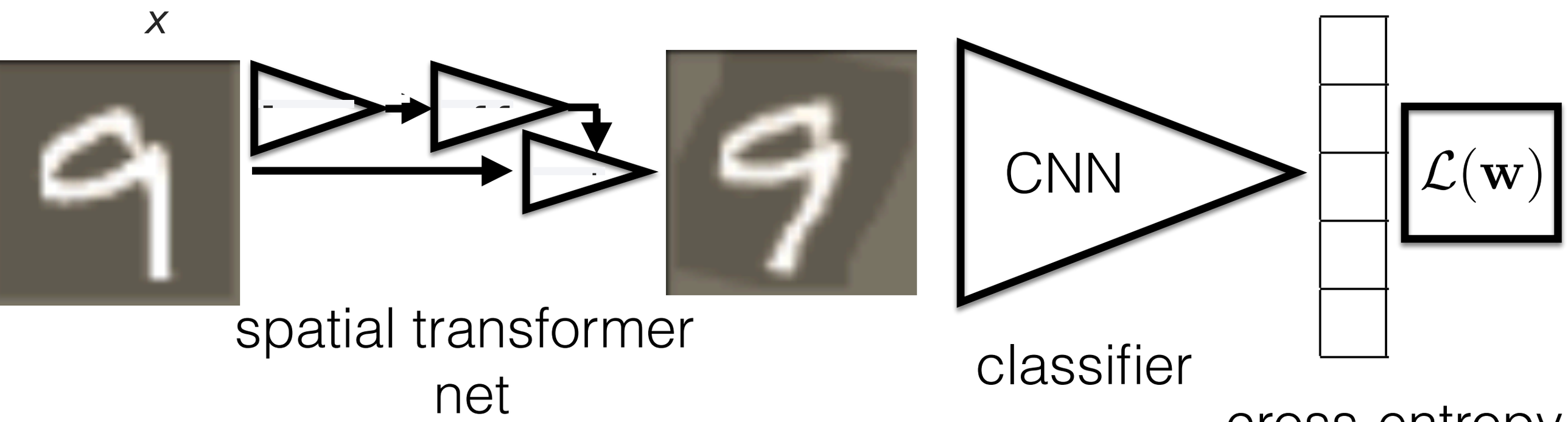
# Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf




spatial transformer net

classifier

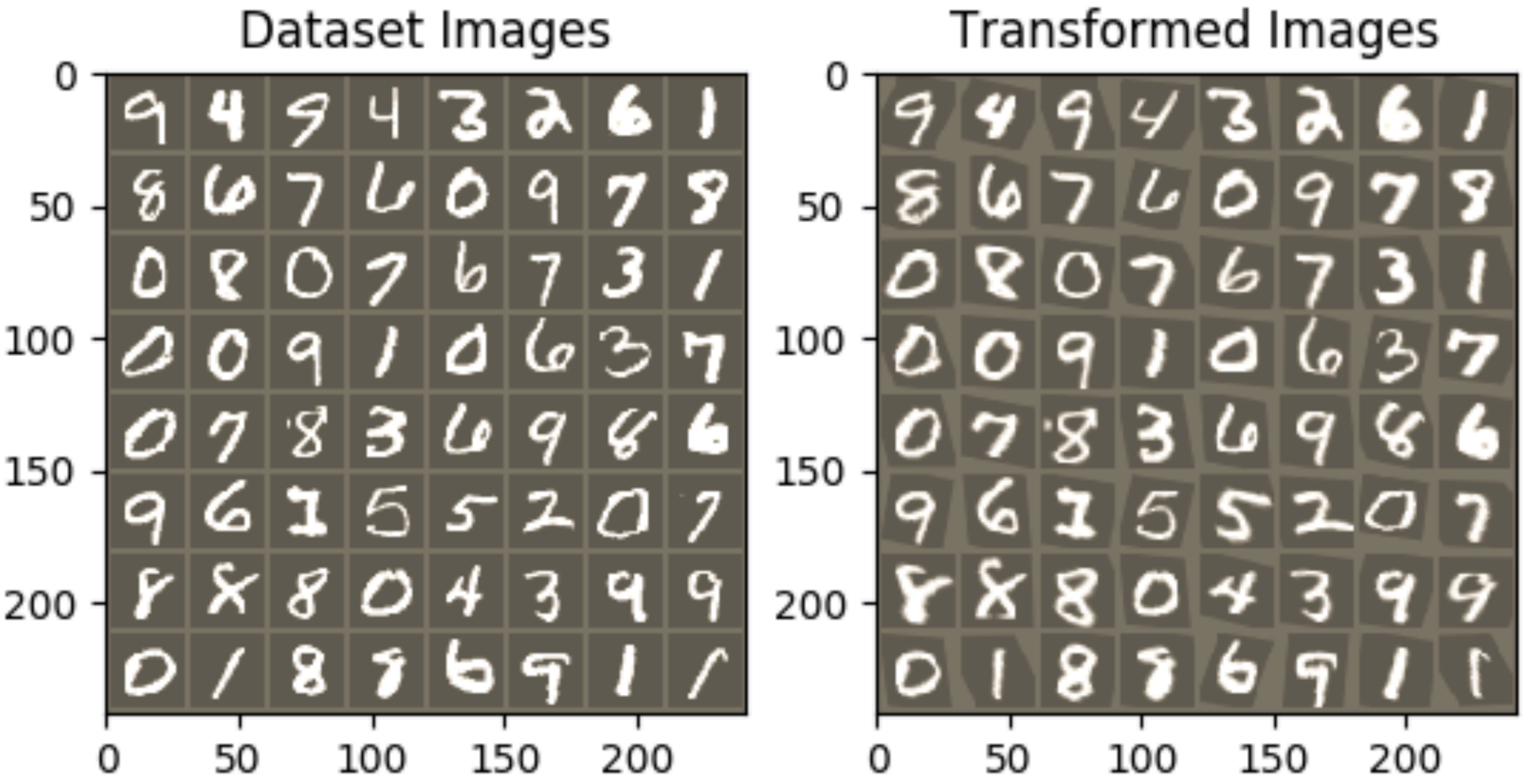
# Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf




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

# Spatial Transformer networks

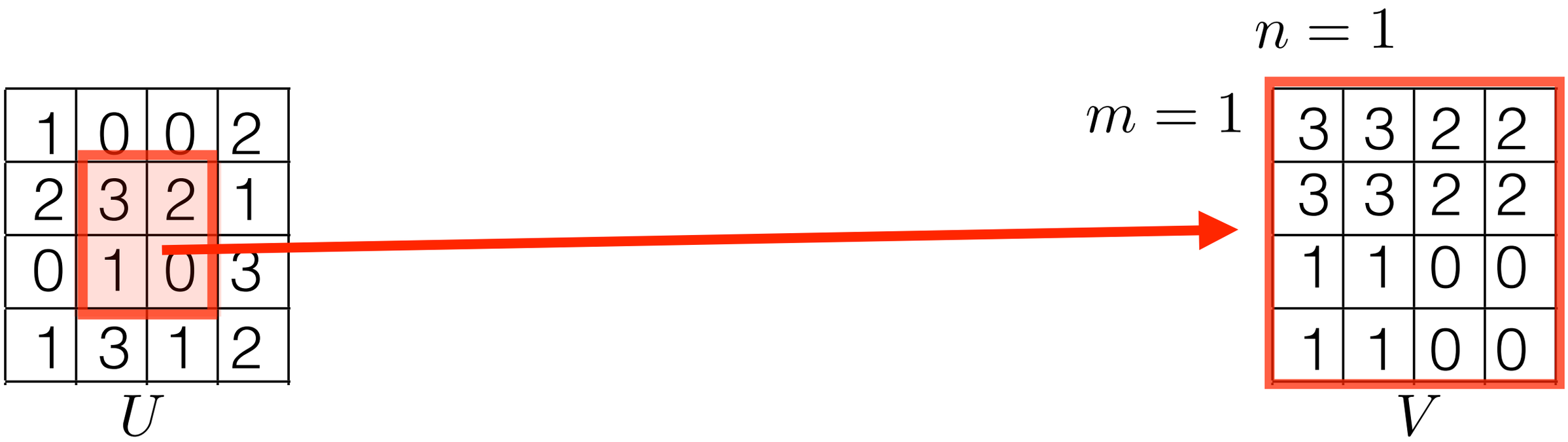
https://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html



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

How does the affine_grid works?

Image crop:

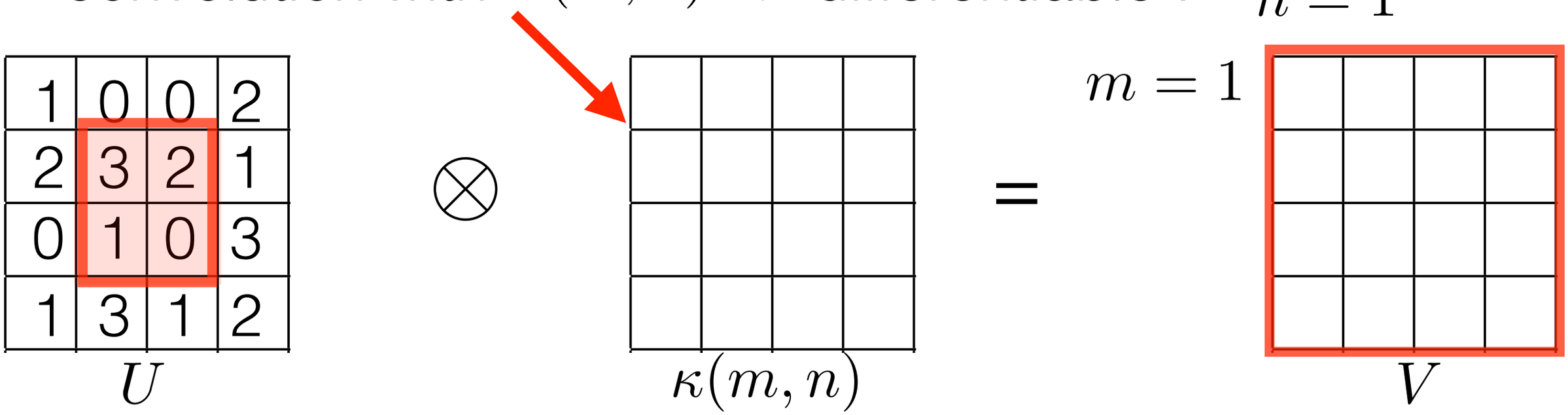


Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image crop:

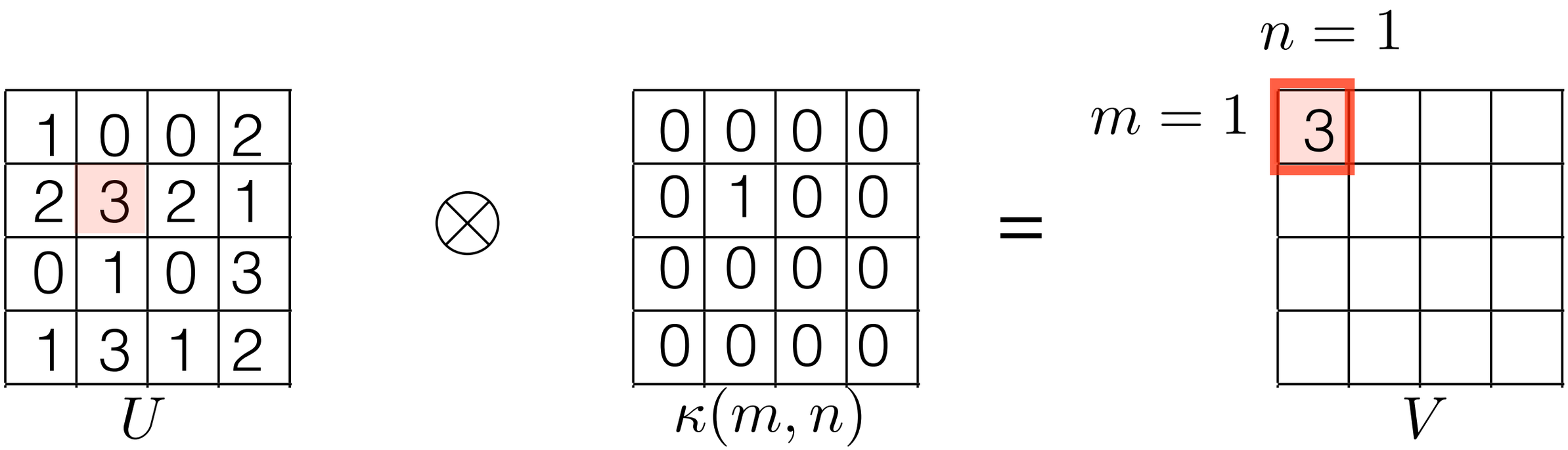
convolution with $\kappa(m,n)$ => differentiable !



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

How does the affine_grid works?

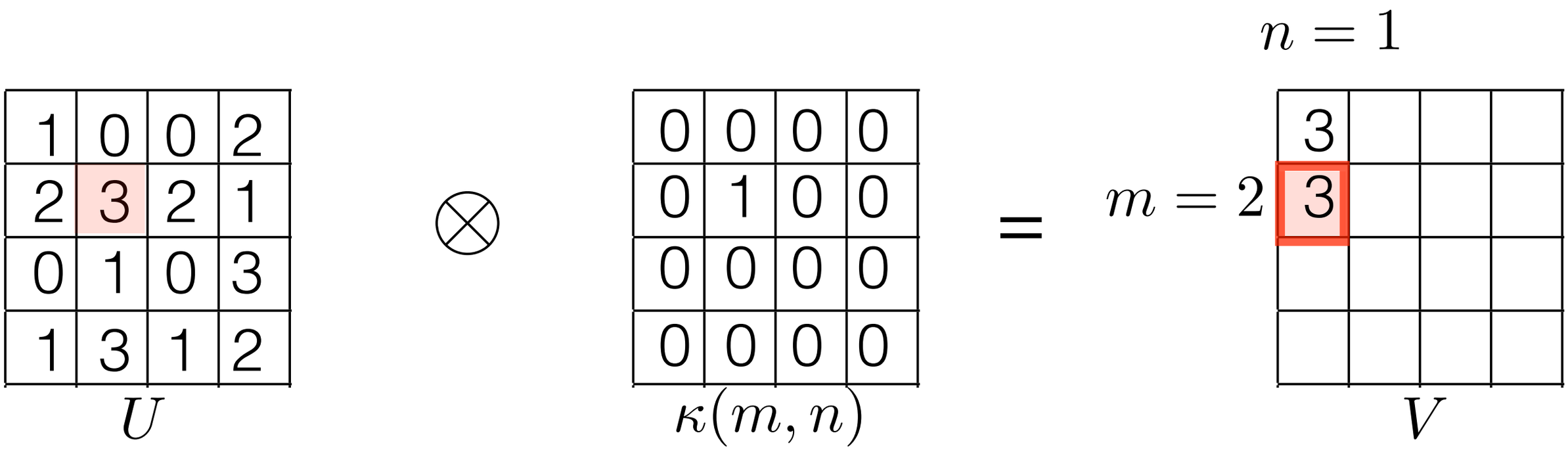
Image crop:



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

How does the affine_grid works?

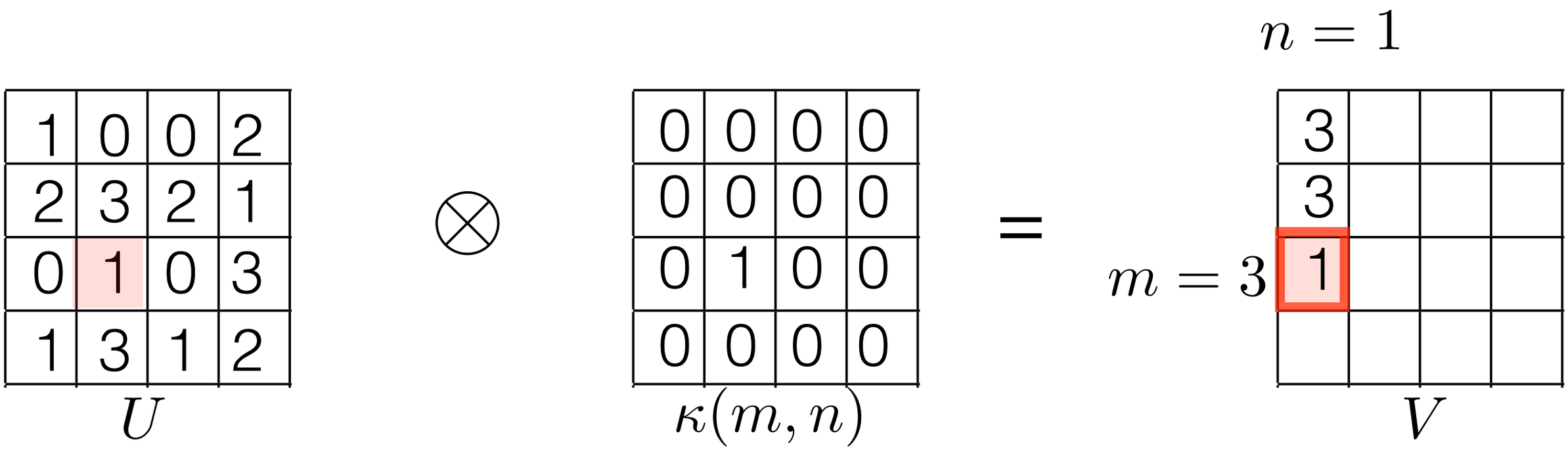
Image crop:



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

How does the affine_grid works?

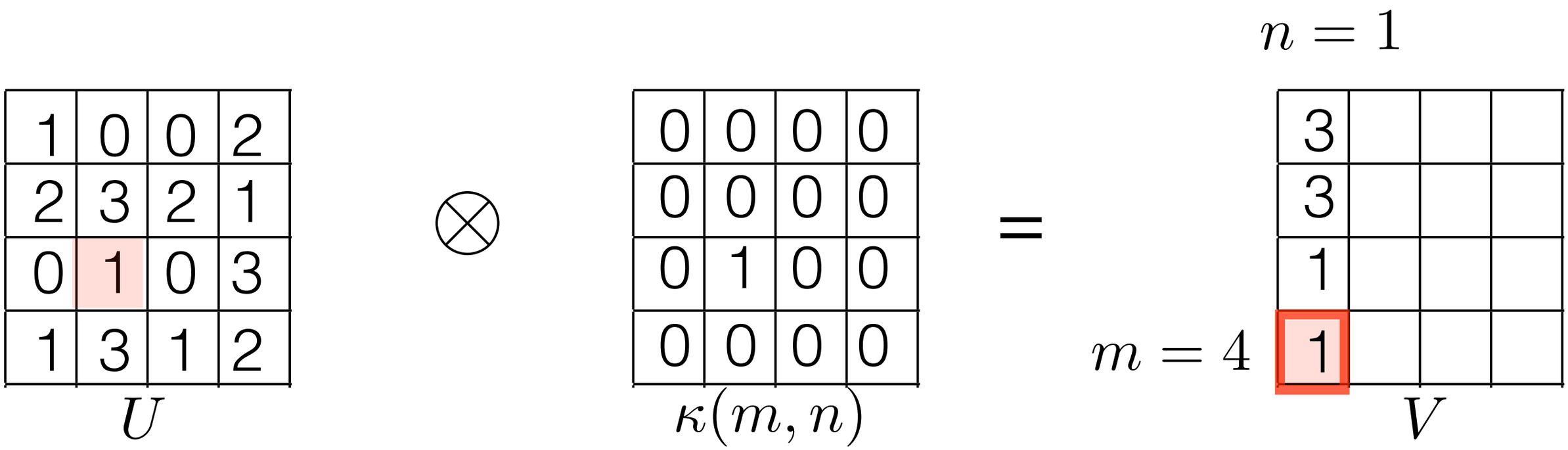
Image crop:



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

How does the affine_grid works?

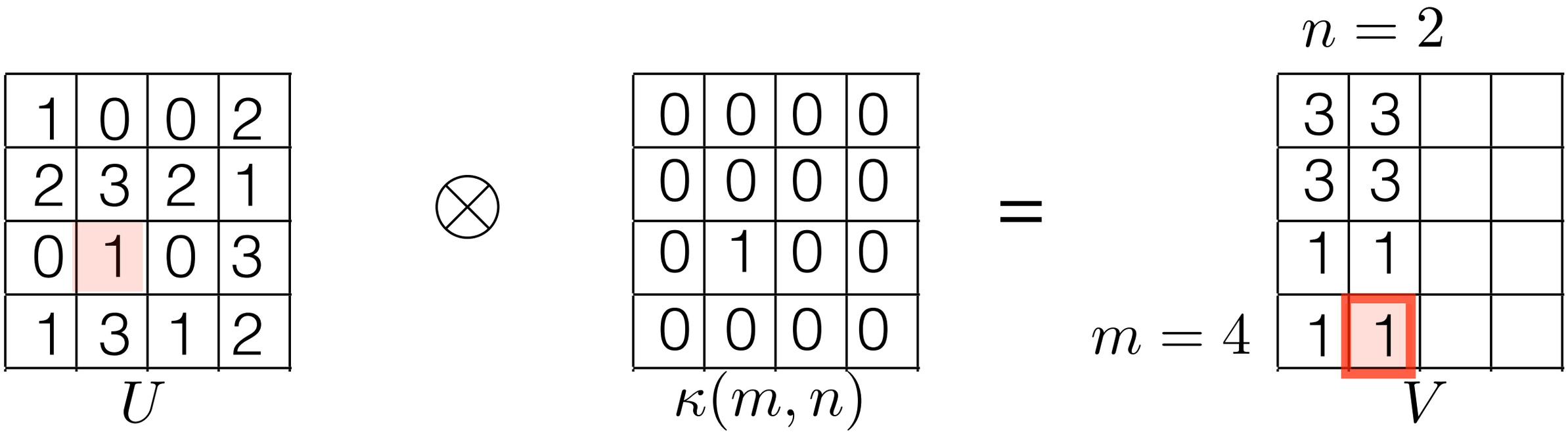
Image crop:



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

How does the affine_grid works?

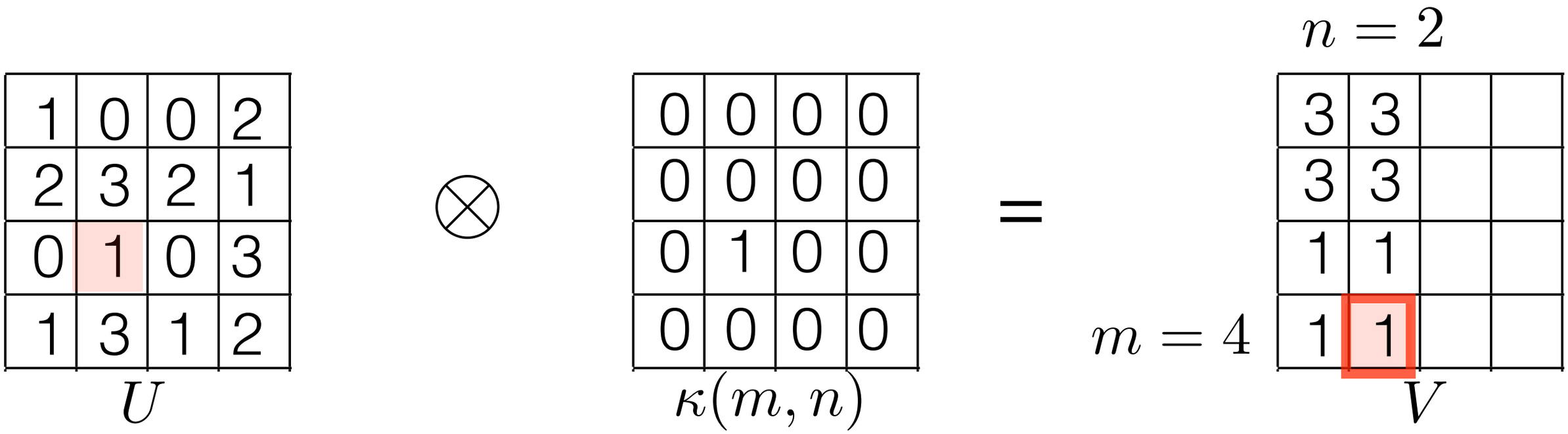
Image crop:



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:

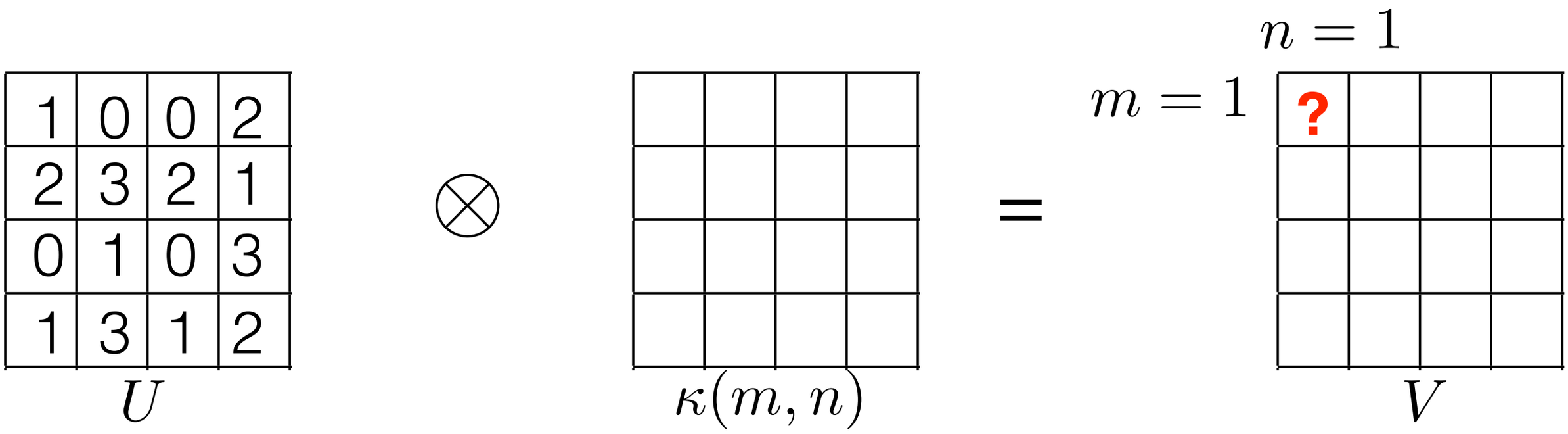
$$\underbrace{\begin{array}{|c|c|c|c|}\hline 1&0&0&2\\\hline 2&3&2&1\\\hline 0&1&0&3\\\hline 1&3&1&2\\\hline\end{array}}_{U} \otimes \underbrace{\begin{array}{|c|c|c|c|}\hline 0&0&0&0\\\hline 0&0&0&0\\\hline 0&0&1&0\\\hline 0&0&0&0\\\hline\end{array}}_{\kappa(m,n)} = \begin{array}{c} \\ \\ \\ m=4 \end{array} \underbrace{\begin{array}{|c|c|c|c|}\hline 3&3&2&2\\\hline 3&3&2&2\\\hline 1&1&0&0\\\hline 1&1&0&0\\\hline\end{array}}_{V} \quad n=4$$

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?

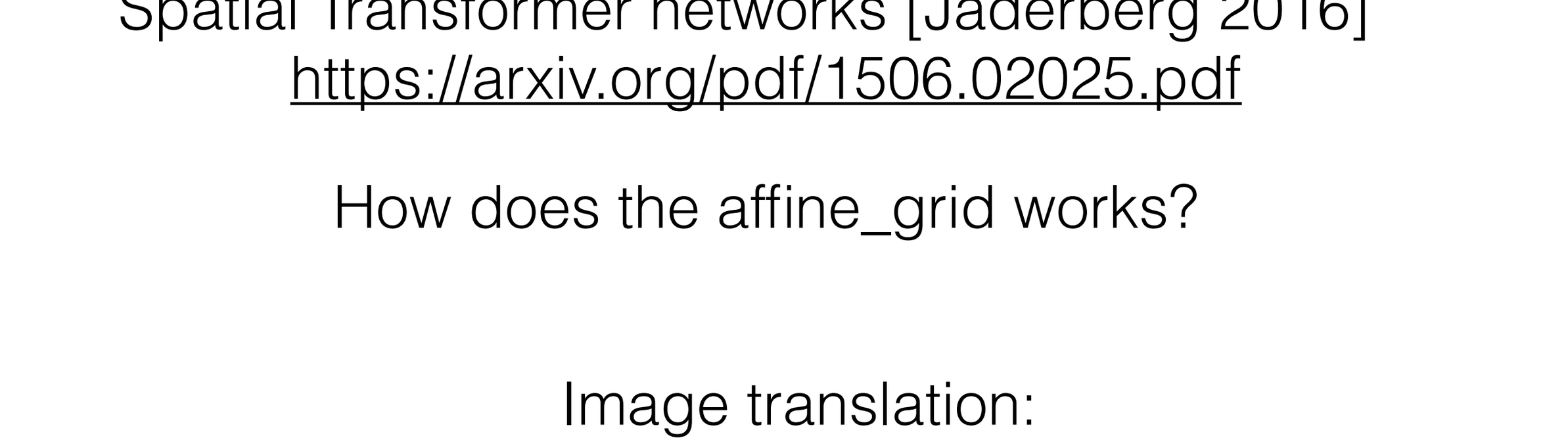


Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image translation:

Can we translate image U by 1/2 pixel?

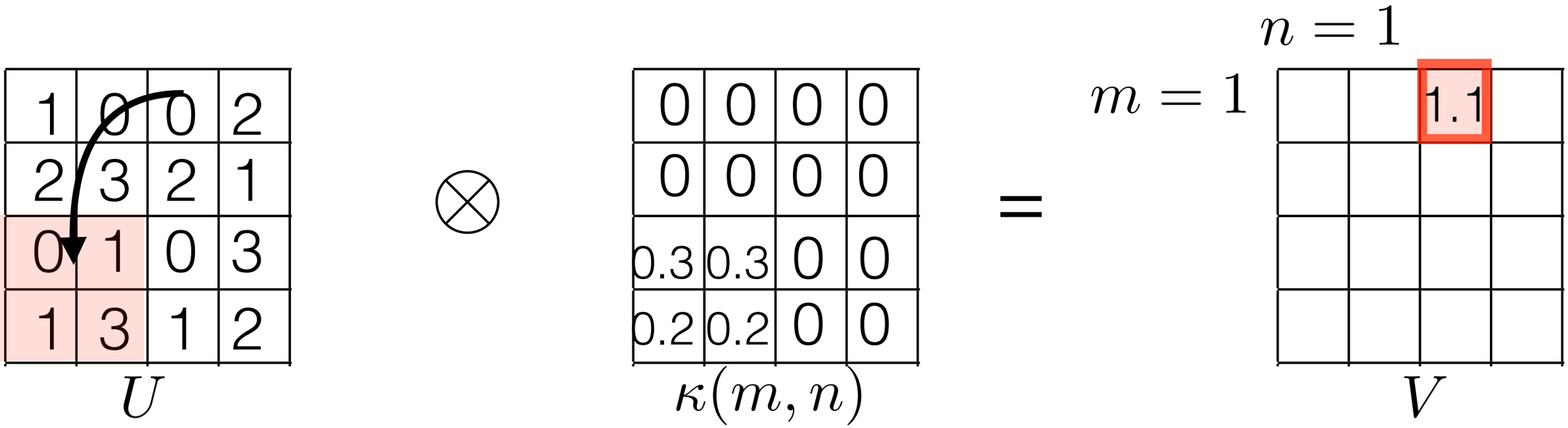


Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf

How does the affine_grid works?

Image translation:

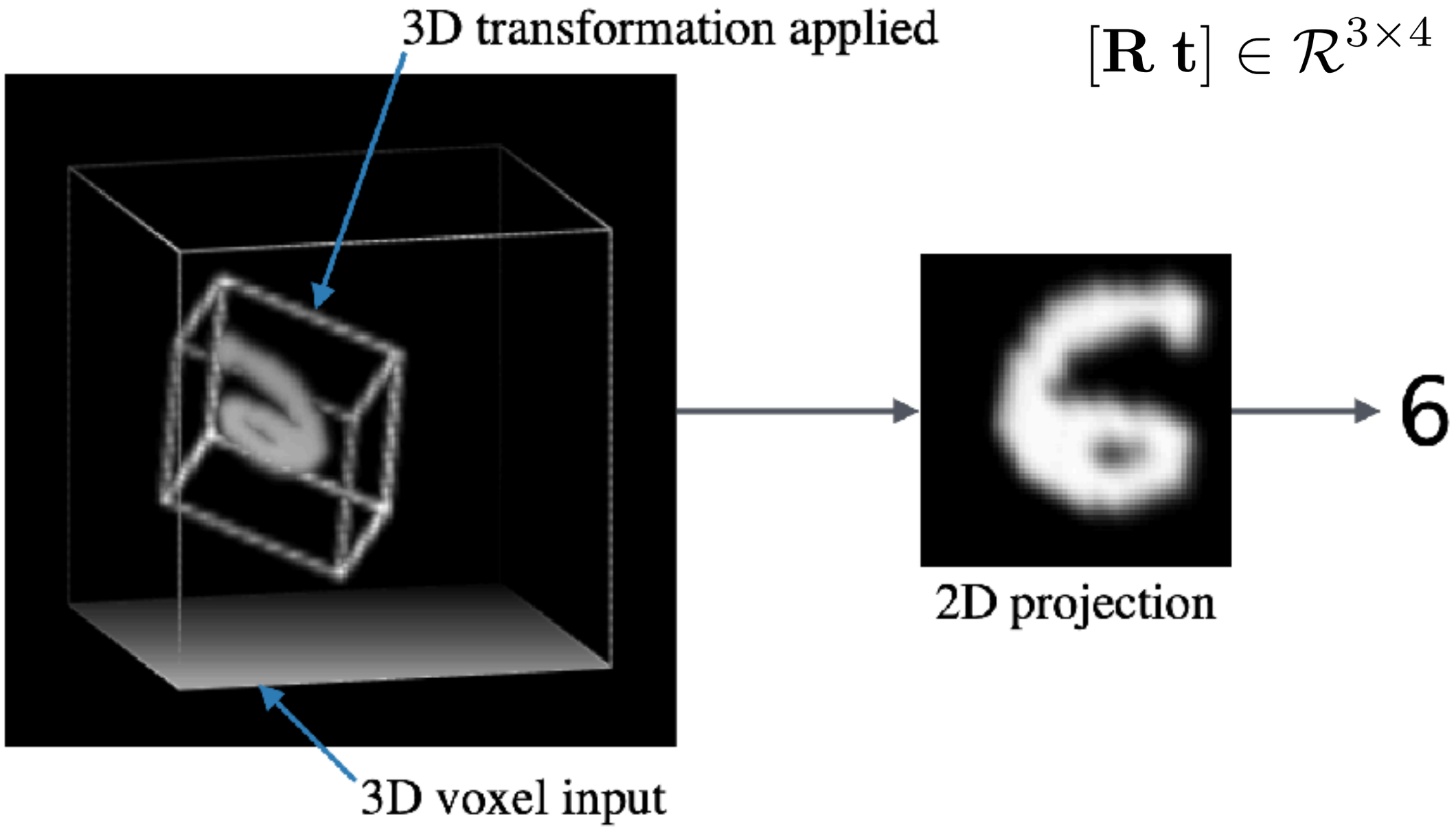
What about rotation?



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

Also implemented 3D affine_grid transformation layer:

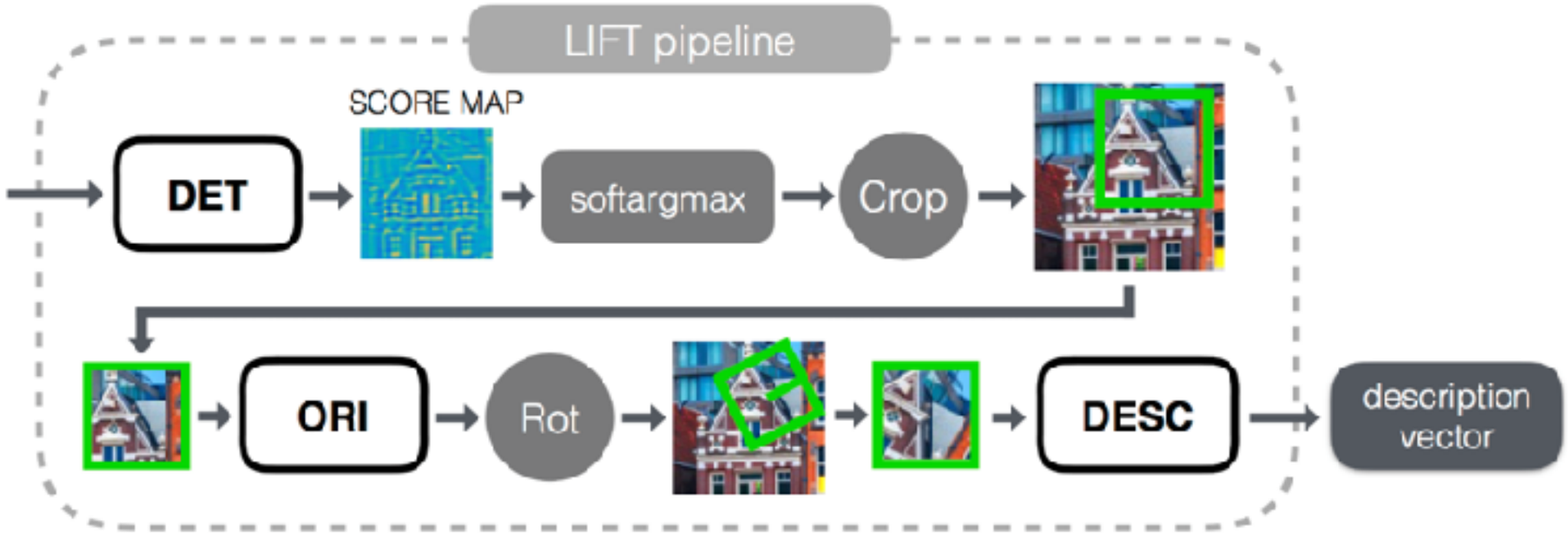
$$[\mathbf{R}\ \mathbf{t}] \in \mathcal{R}^{3\times 4}$$



# 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

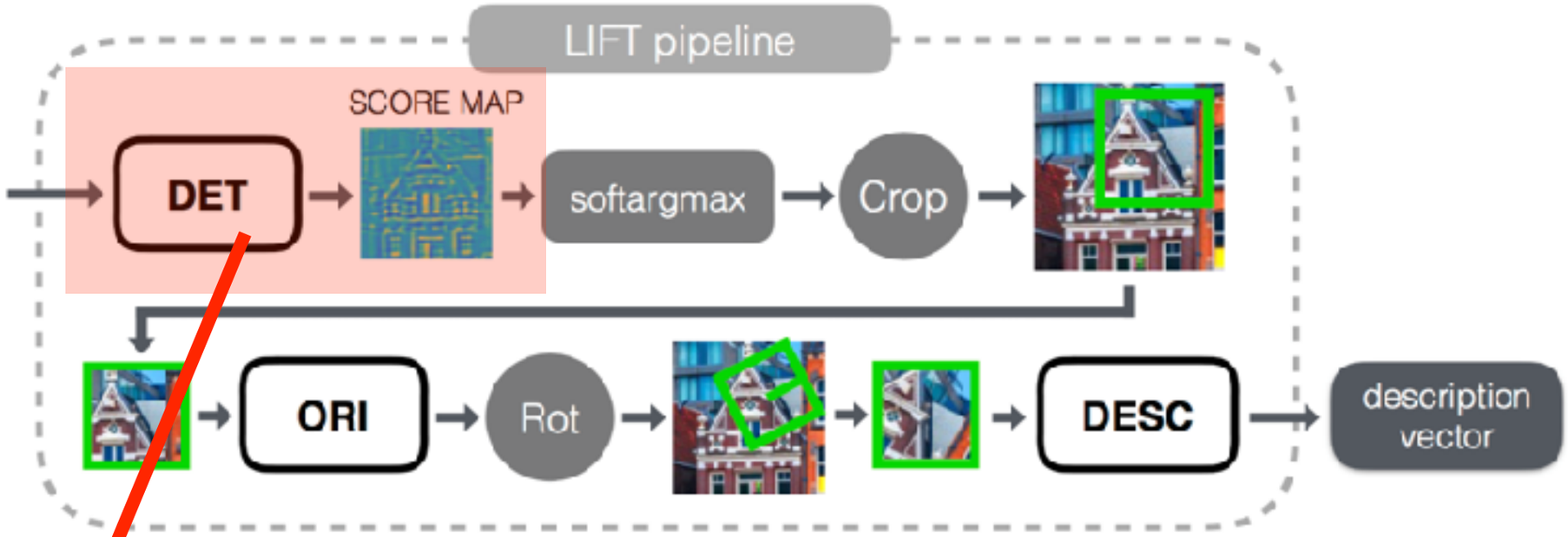



Input: RGB image

Output: set of detected feature points with descriptors

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

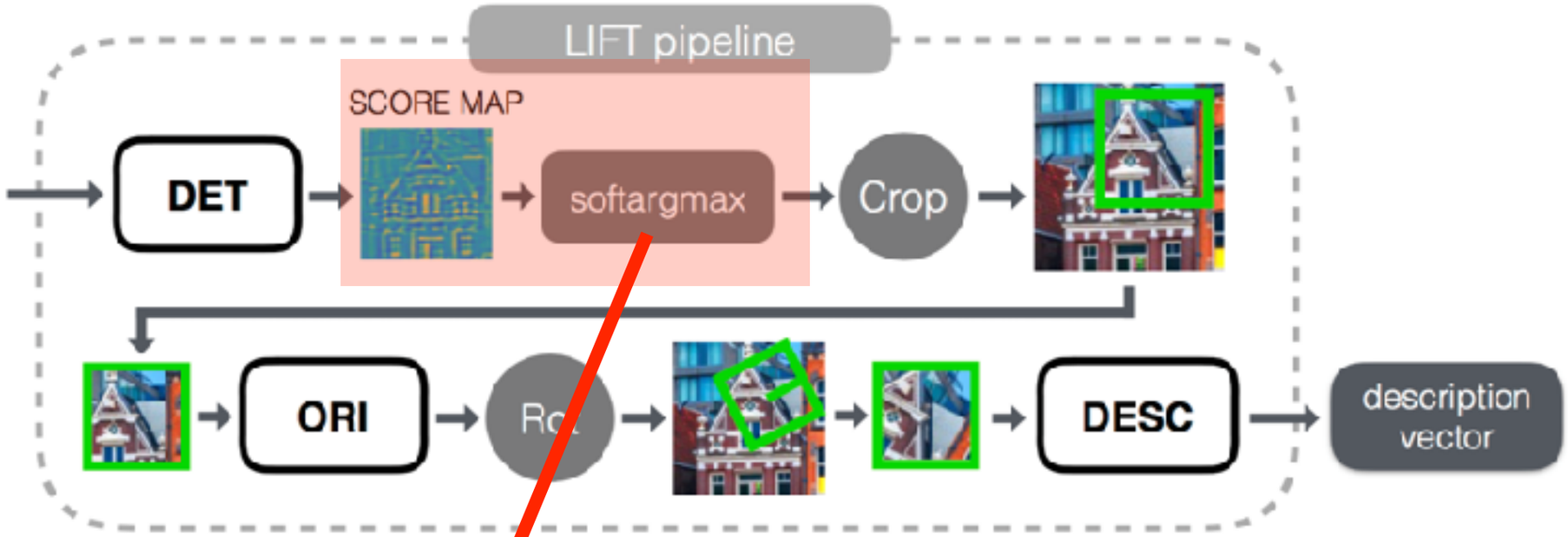
# LIFT: Learnable Invariant Feature Descriptors
# [Yi et al ECCV 2016] https://arxiv.org/abs/1603.09114

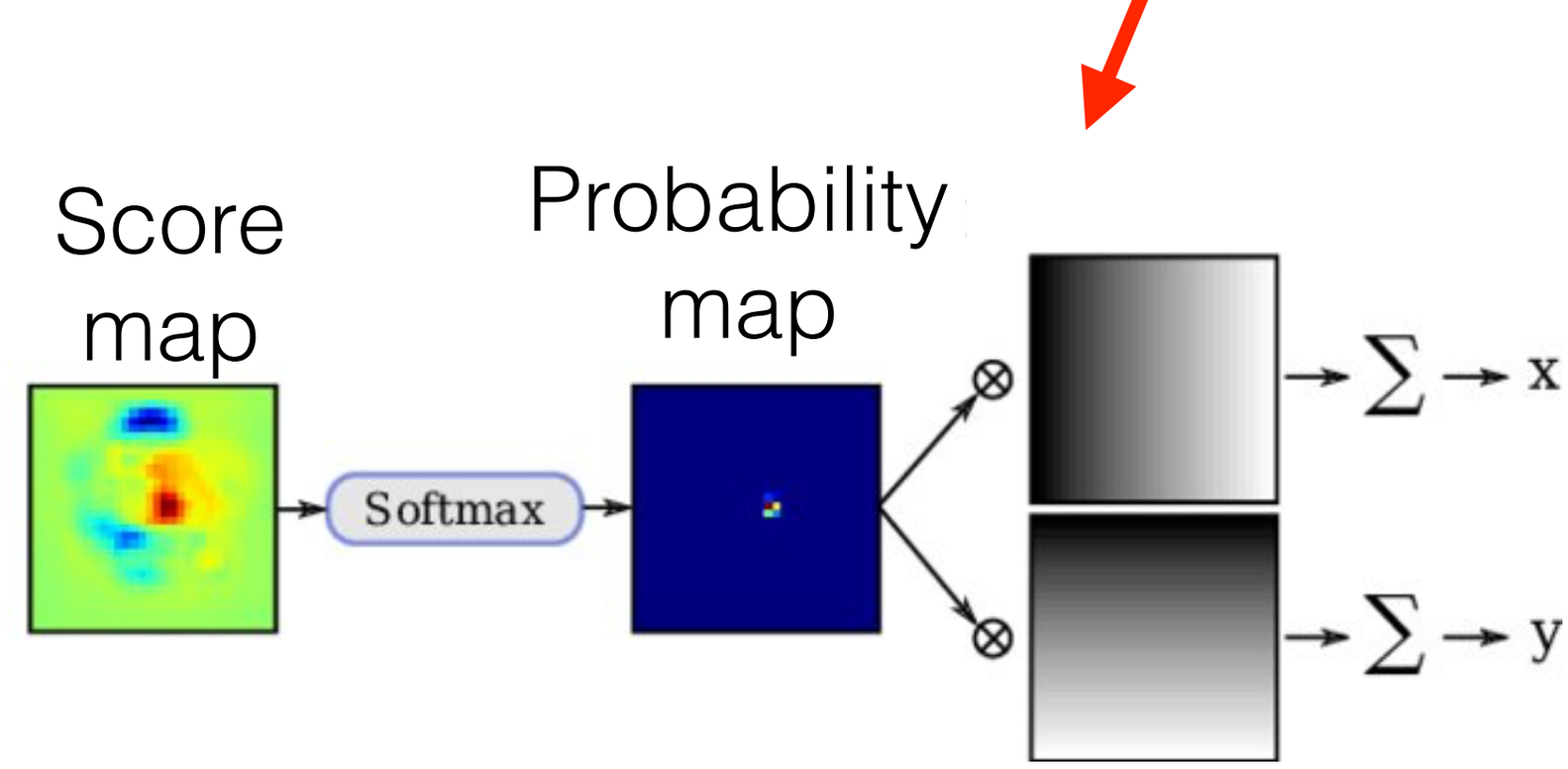



Segmentation CNN for pixel-wise two-class labelling

- 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

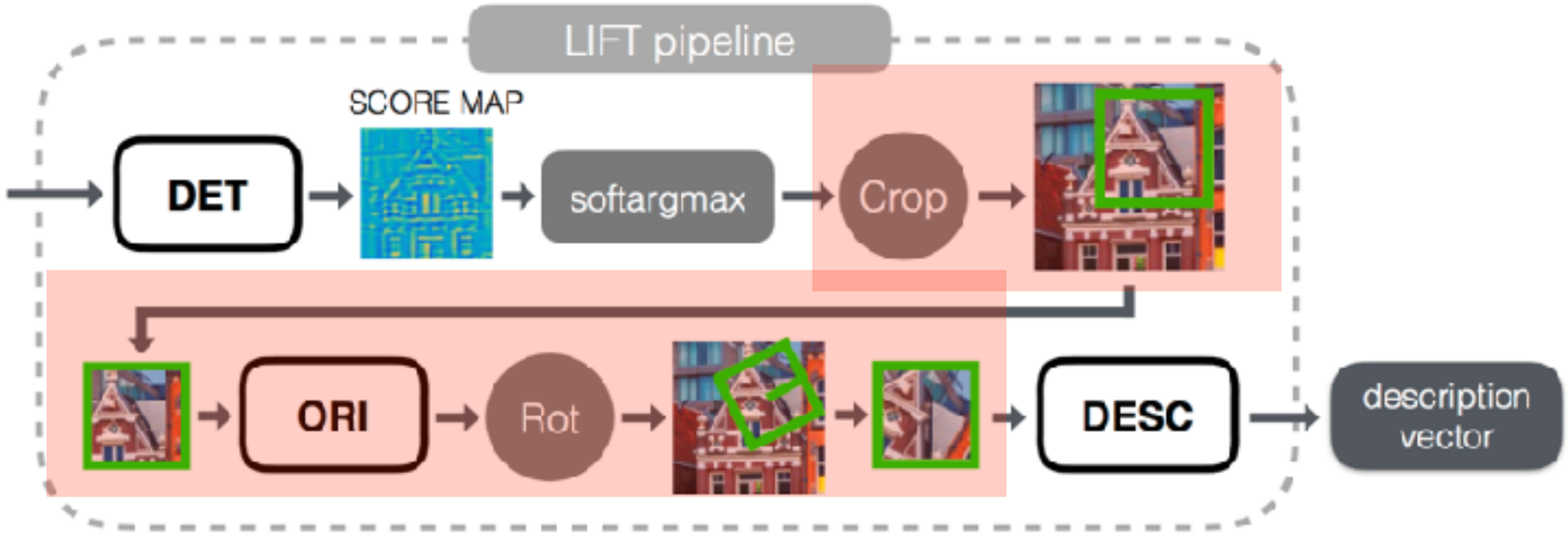



Expected value of x and y under the prob. distr.

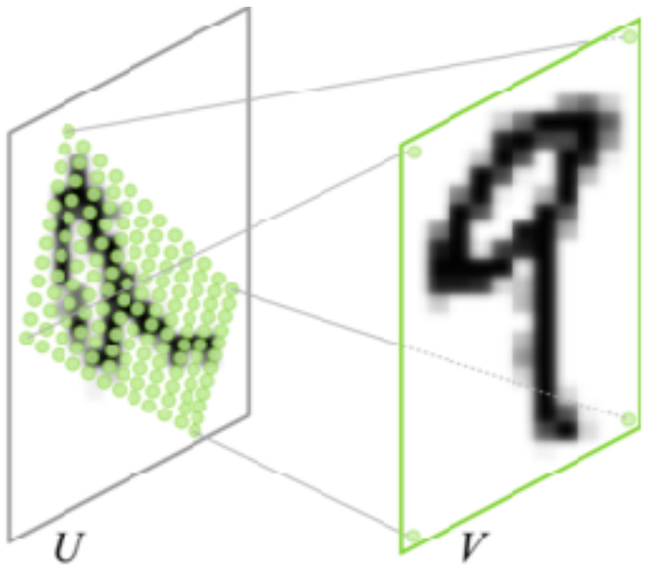
Learnable Invariant Feature Descriptor (LIFThttps://www.researchgate.net/publication/323410987_2D3D_Pose_Estimation_and_Action_Recognition_using_Multitask_Deep_Learning/figures?lo=1)

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



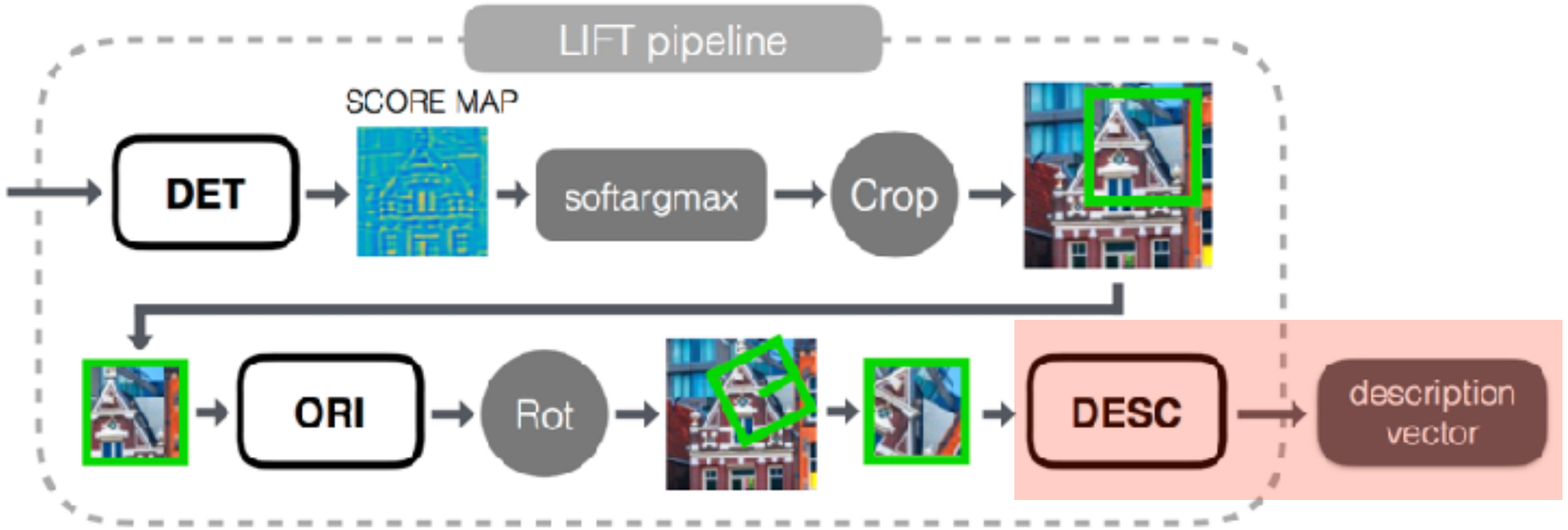

## Spatial Transformer Network

Bilinear approximation of affine transformation is differentiable !

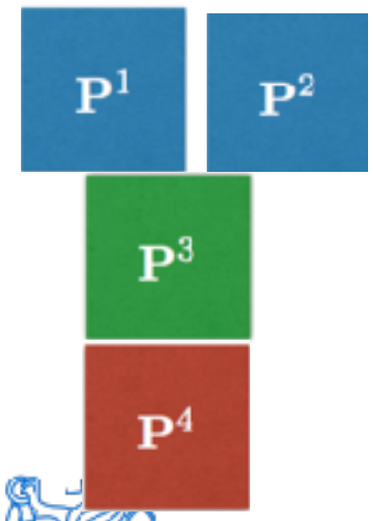



[Jaderberg, 2016] https://arxiv.org/pdf/1506.02025.pdf

# 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:

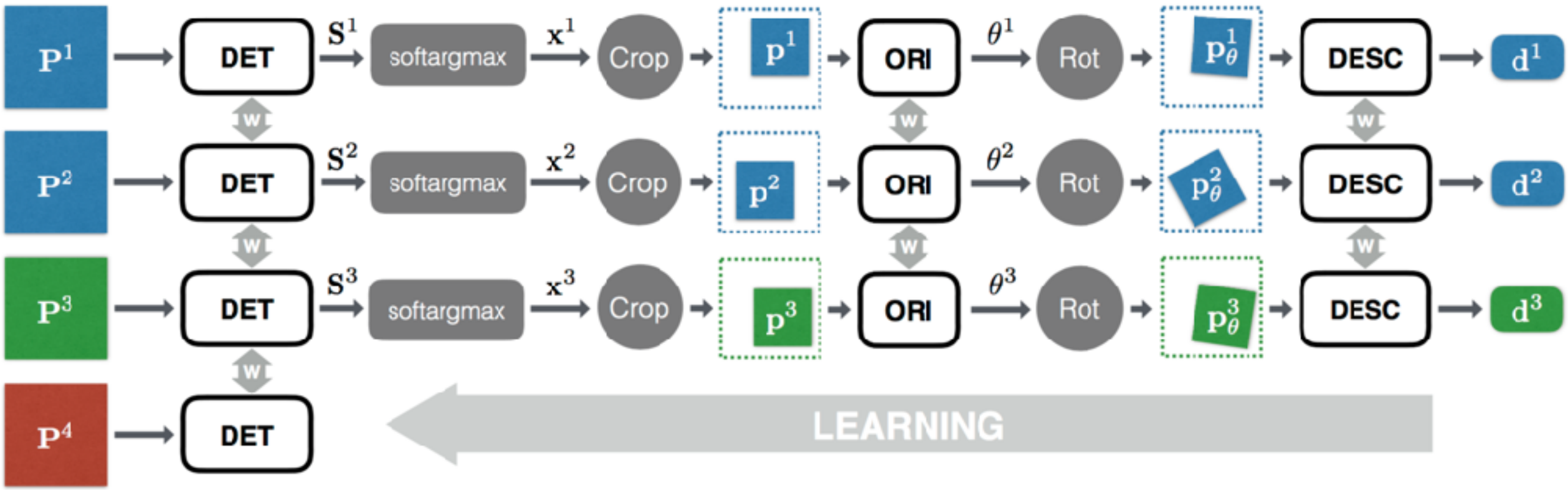



Two corresponding patches on distinctive points

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