Architectures for pose regression and object detection

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

Outline

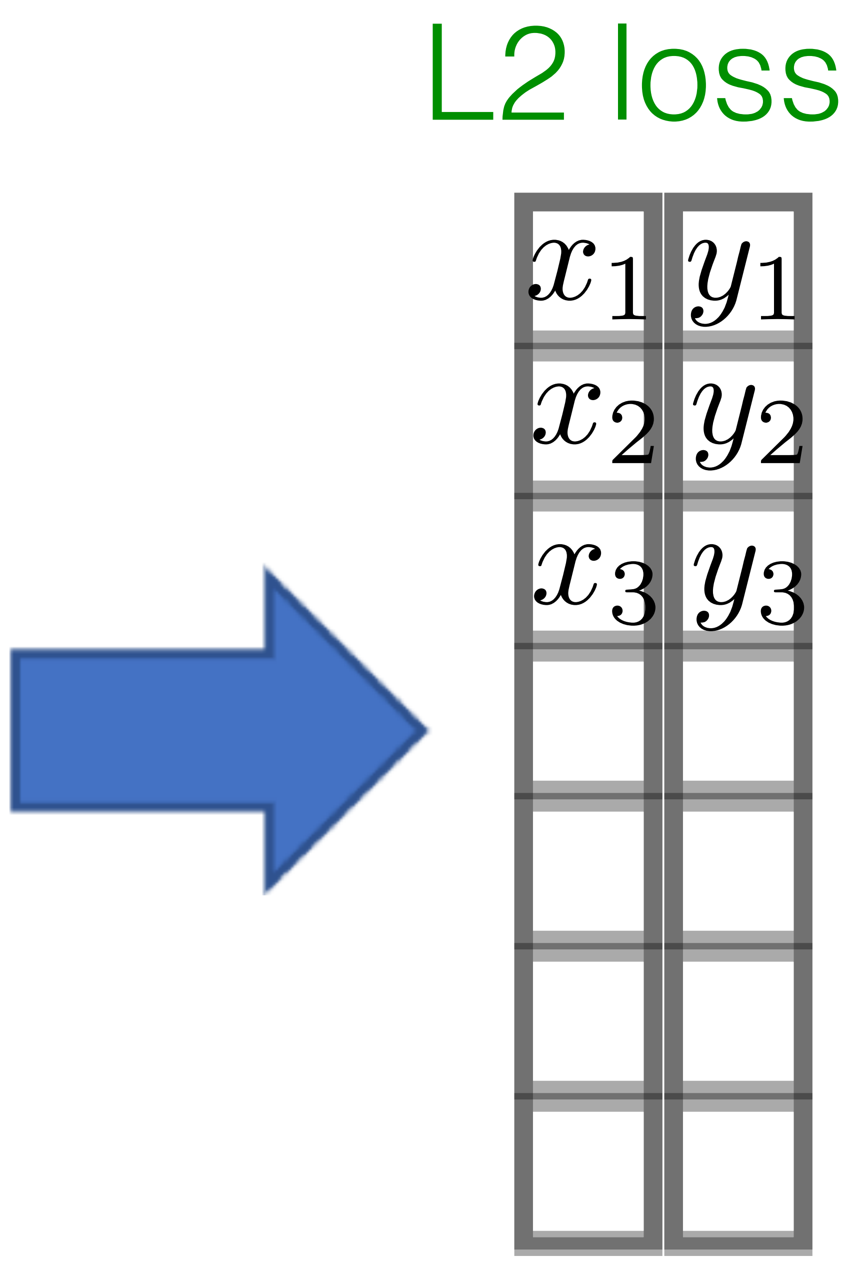
- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Architectures of regression networks
- Architectures of feature matching networks

Pose regression: approach I

L2 loss

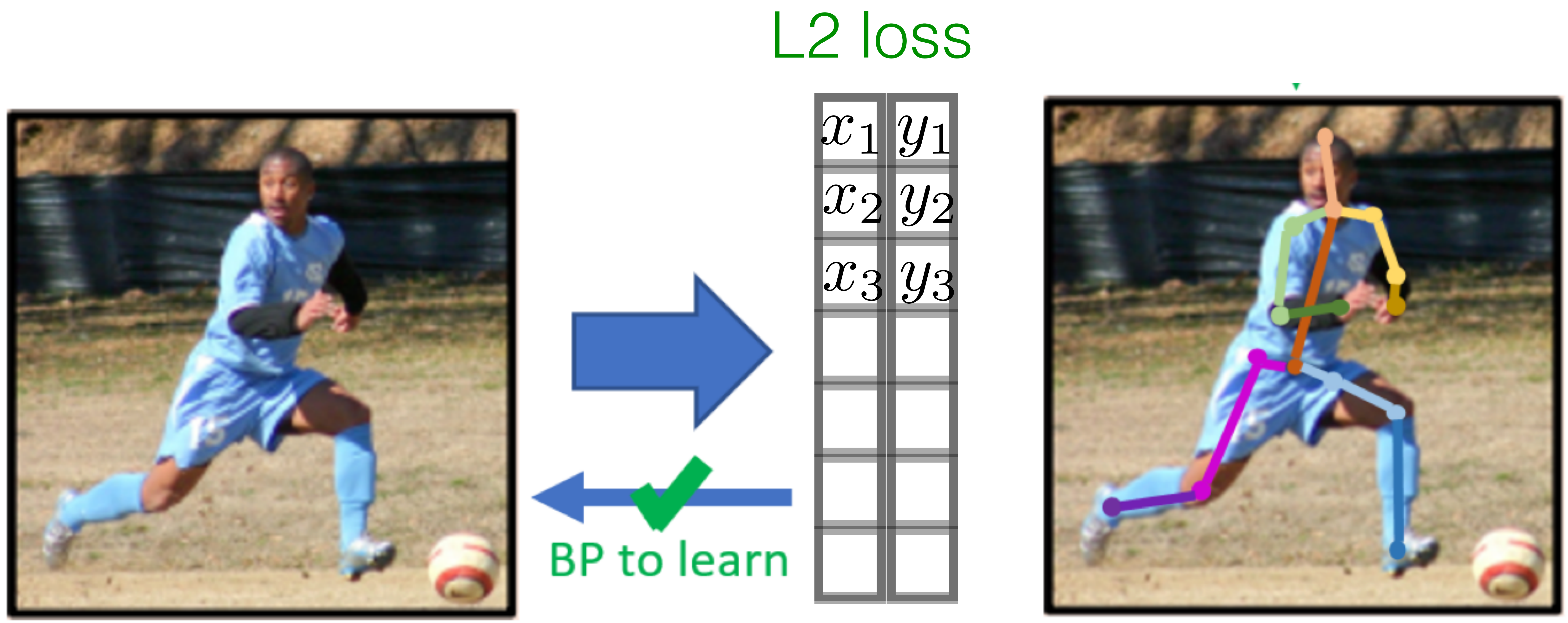
Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

Pose regression: approach I



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

Pose regression: approach I



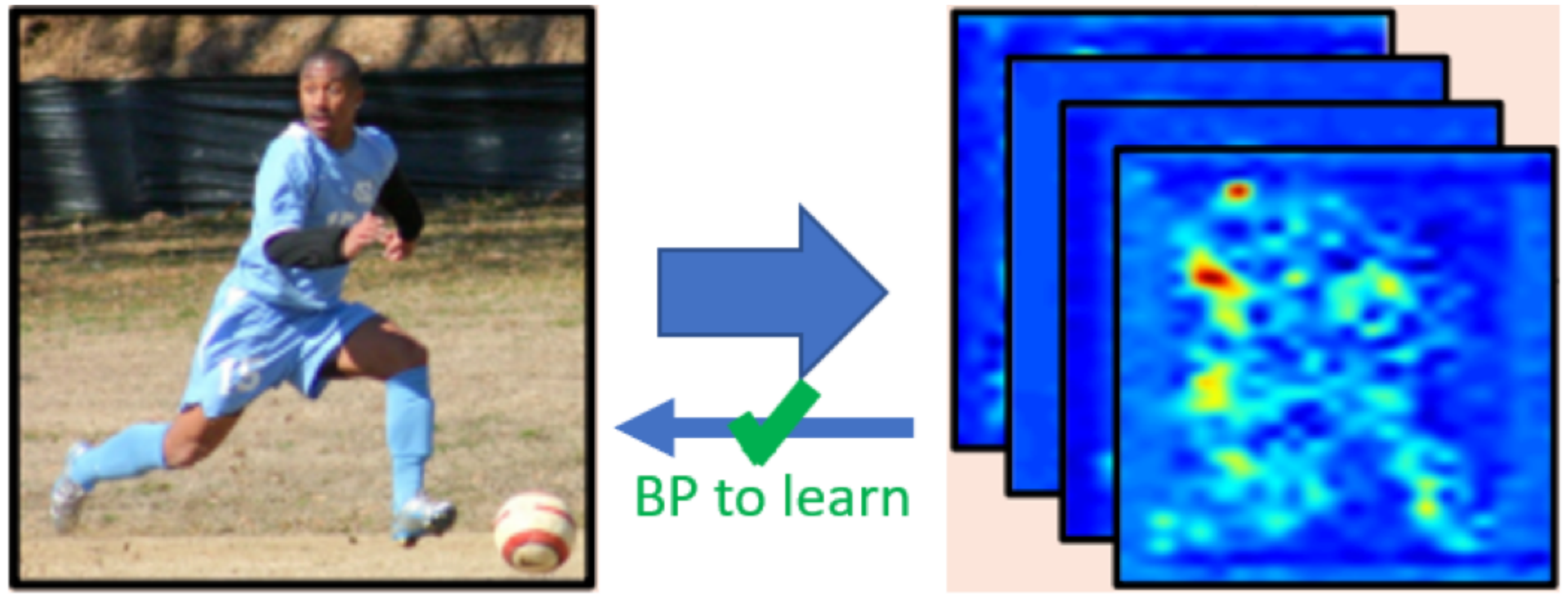
Joint regression:

- 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 regression: approach II

cross-entropy loss



$\mathbf{H}_k$: Heatmap

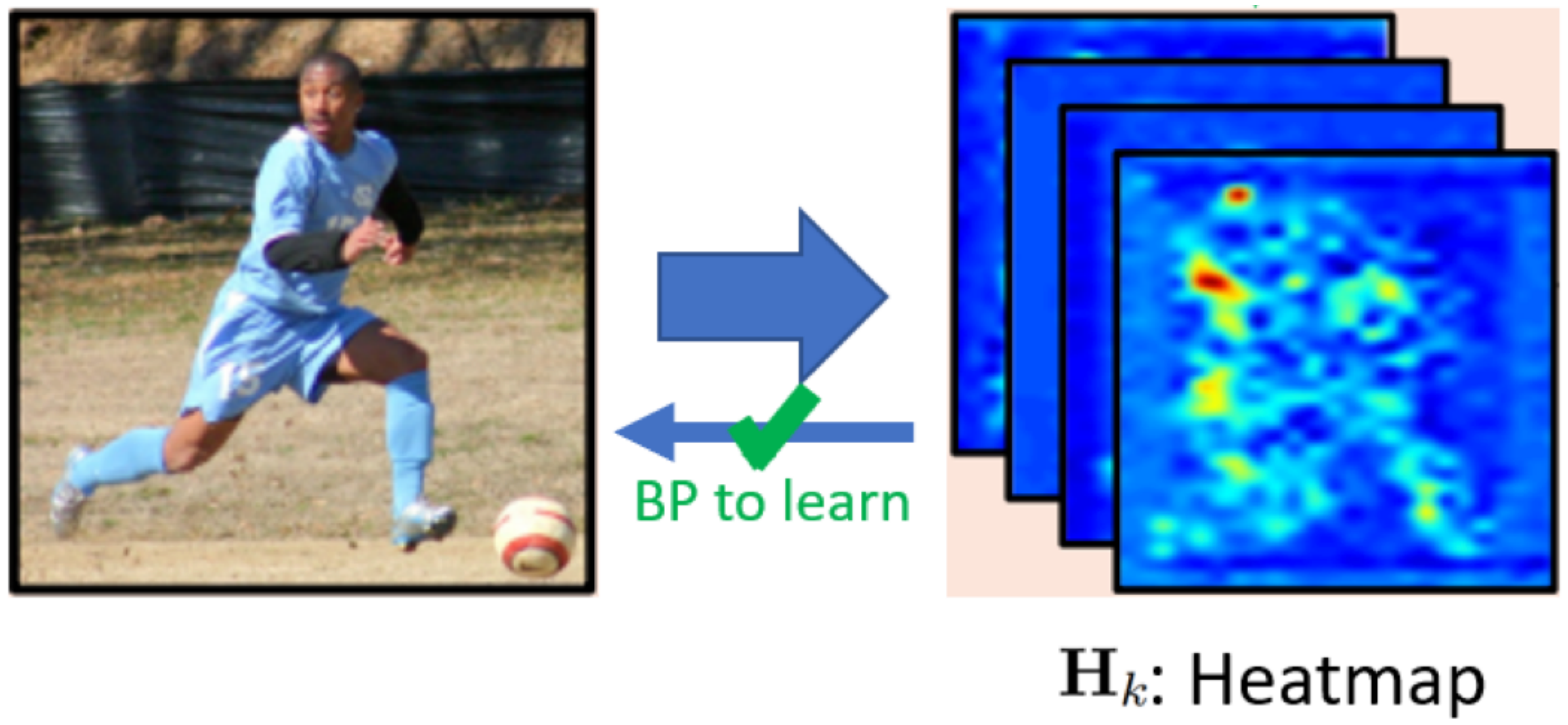
Joint segmentation:

- ConvNet first estimates N joint's heat maps $\mathbf{H}_k,\ k = 1 \dots N$ (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 regression: approach II

cross-entropy loss



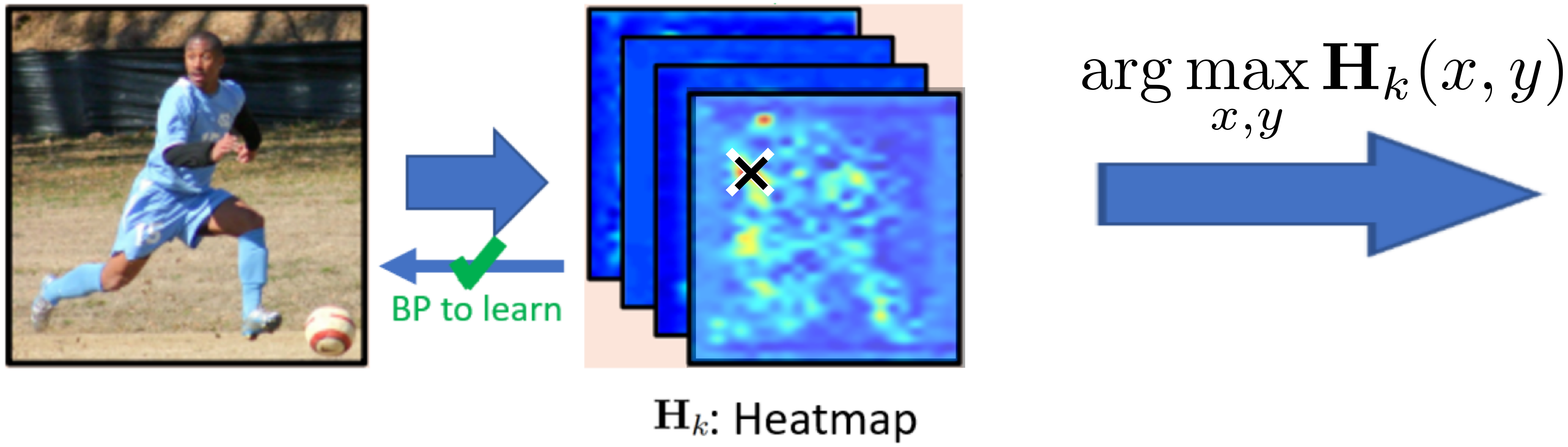
Joint segmentation:

- estimate joint position as position of heatmap maximum

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

Pose regression: approach II

cross-entropy loss

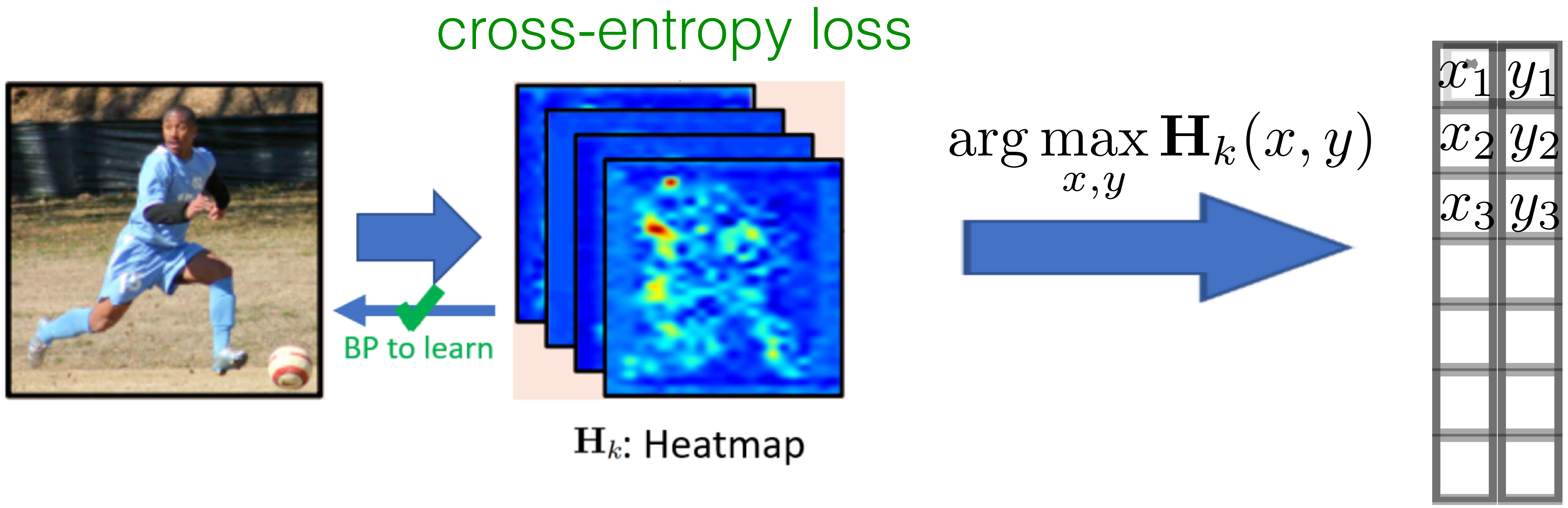


Joint segmentation:

- estimate joint position as position of heatmap maximum

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

Pose regression: approach II

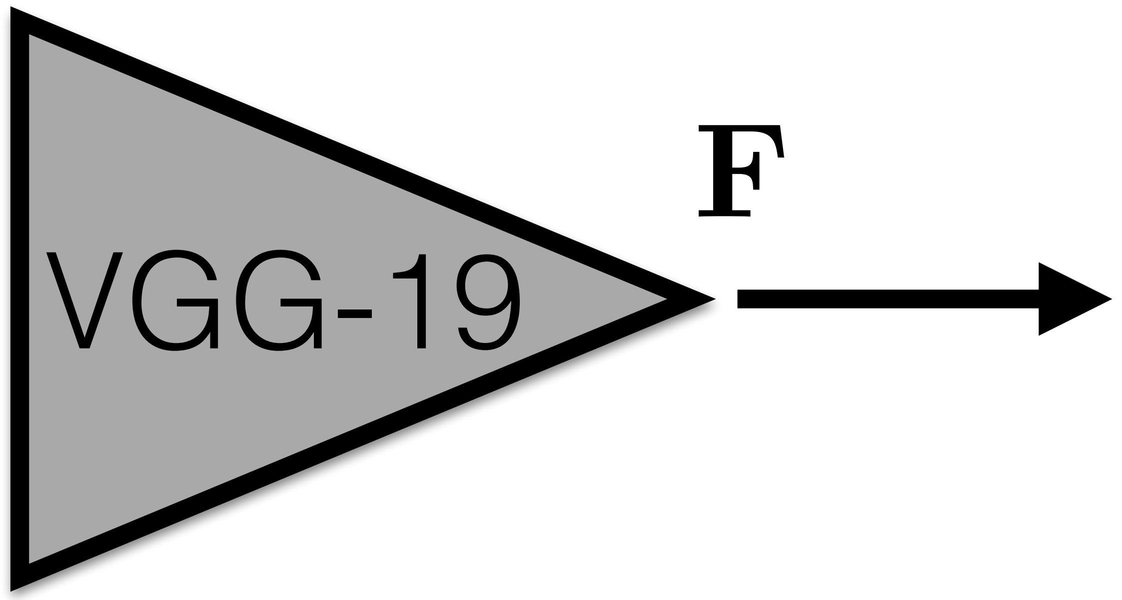


Joint segmentation:

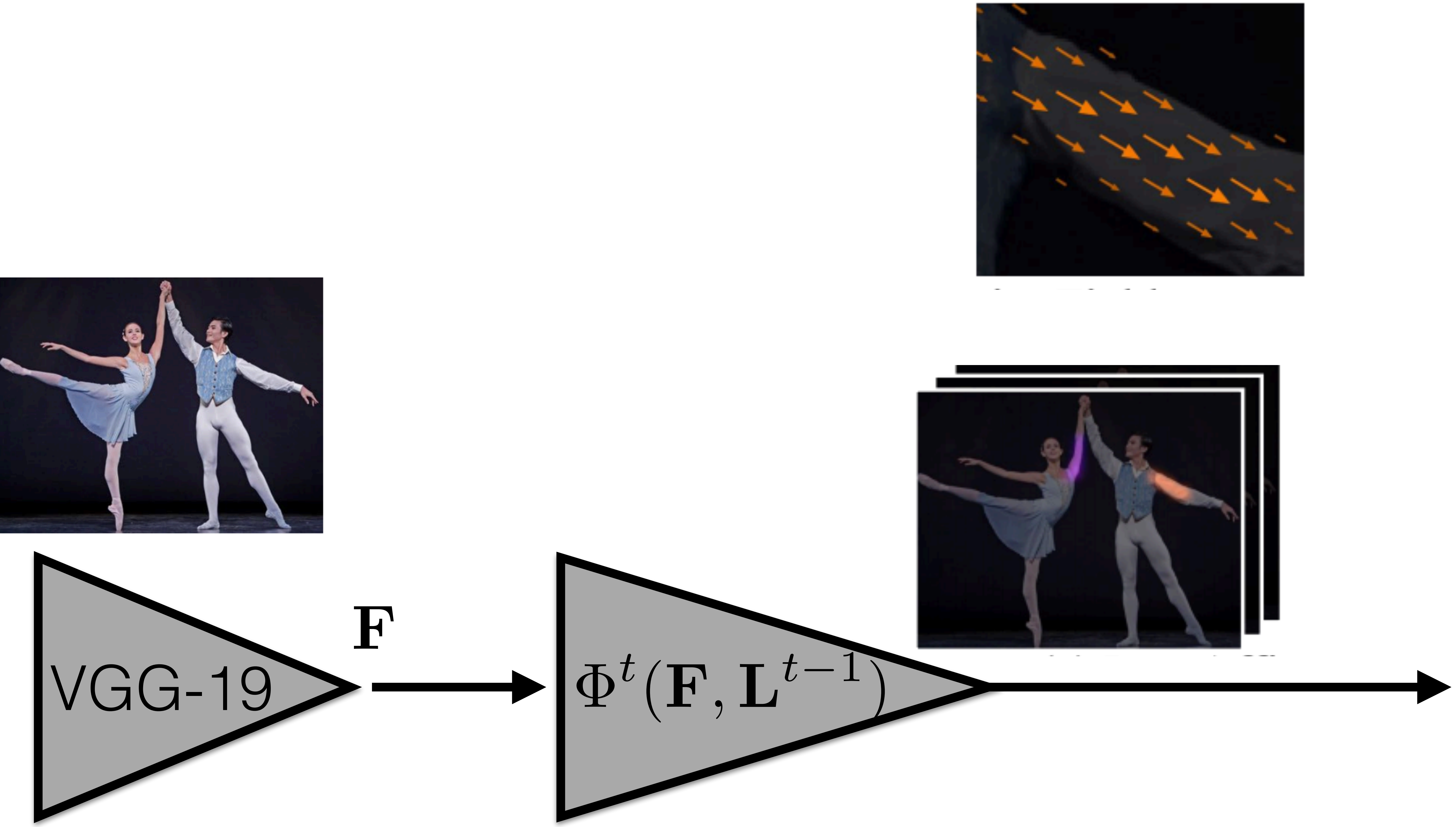
- estimate joint position as position of heatmap maximum

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

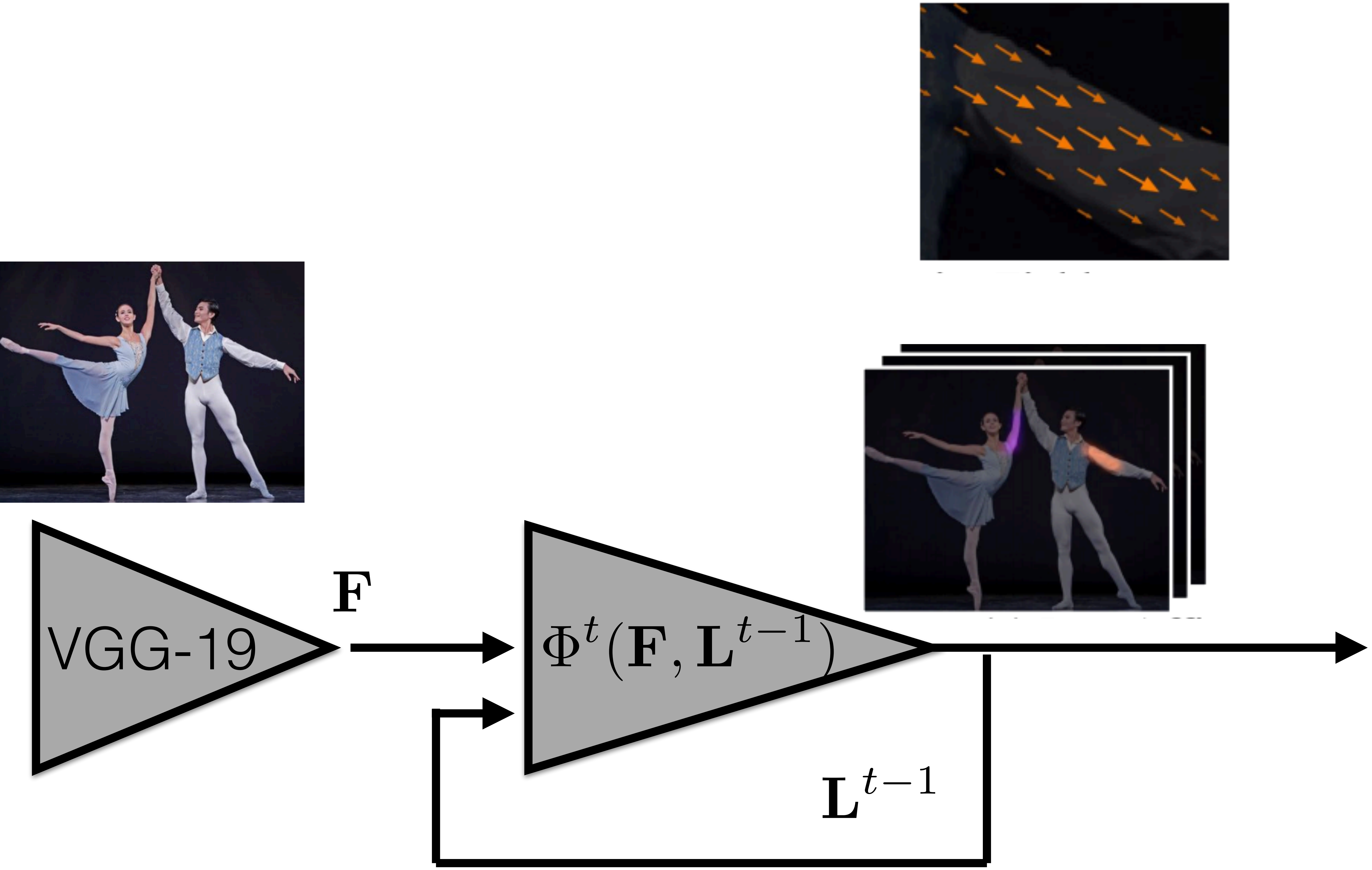
OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf



OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf

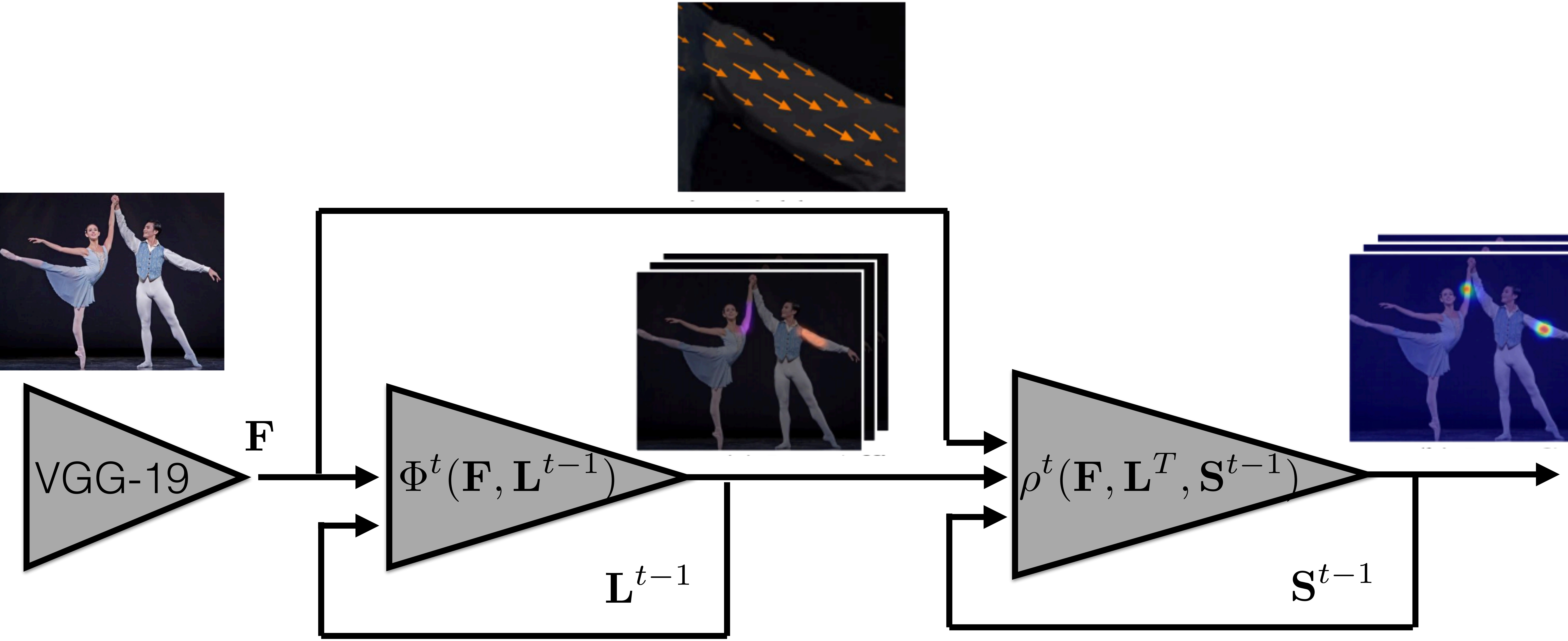


OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf



OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf

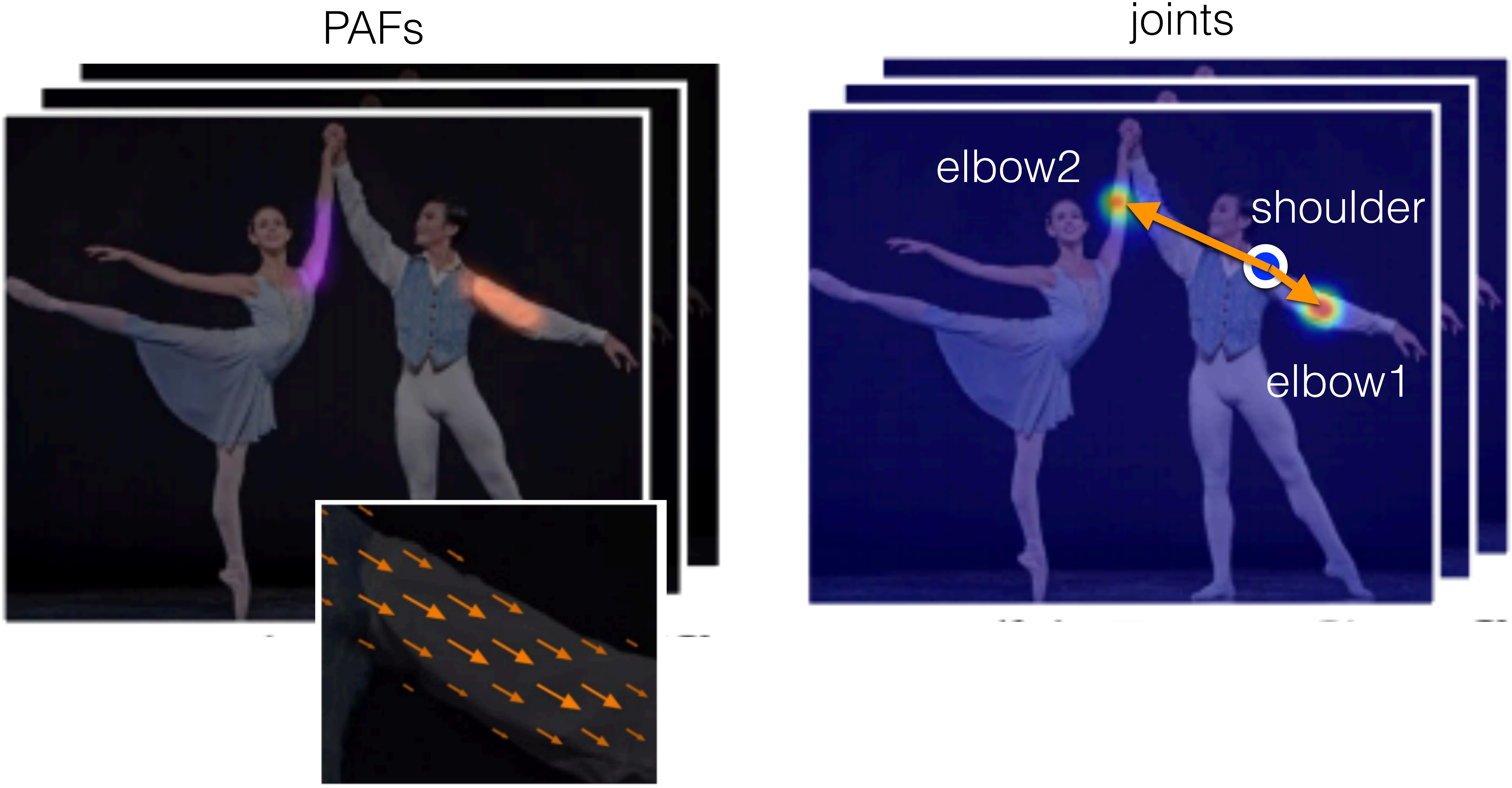
OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf



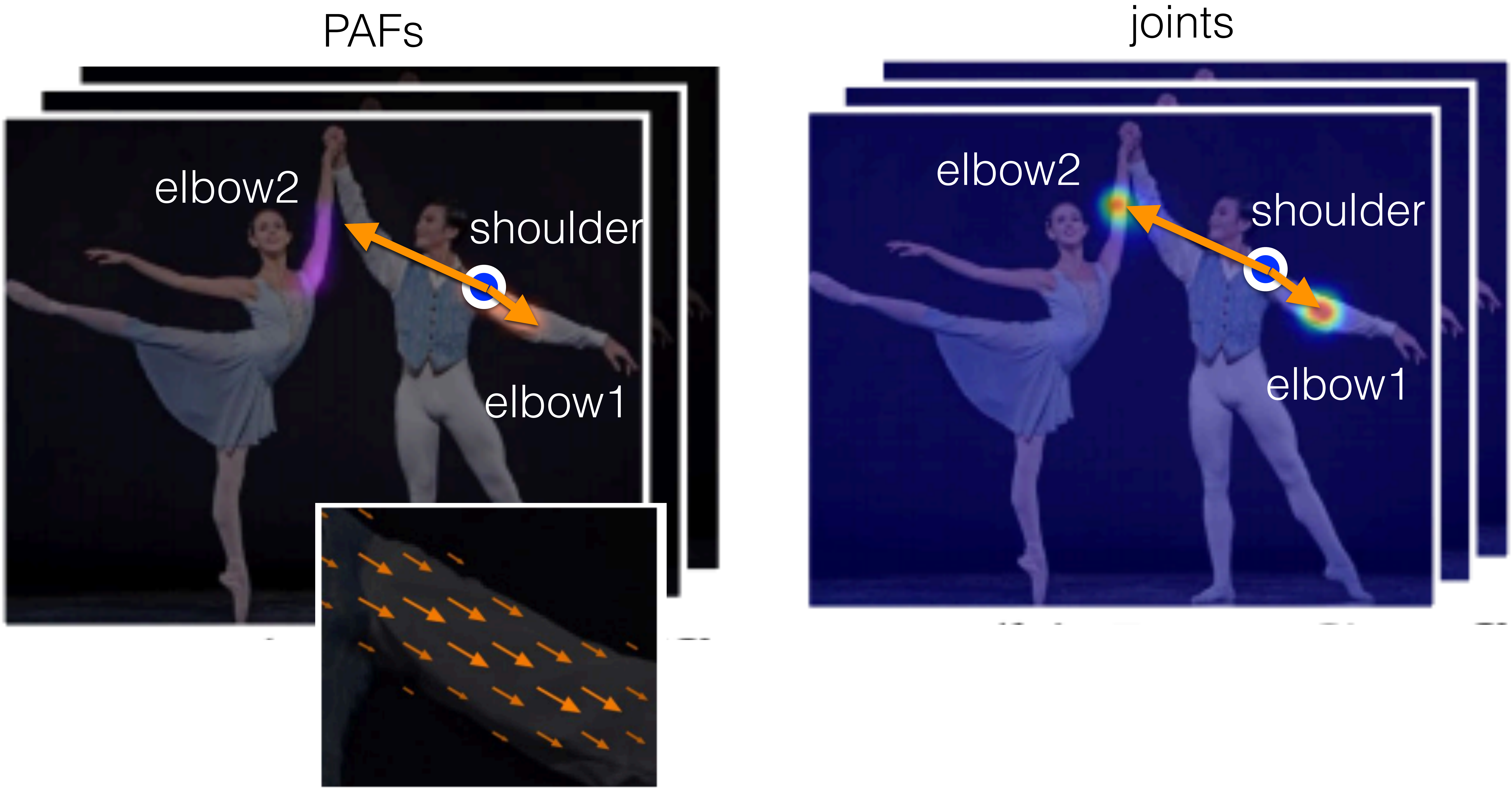
OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf

$$\mathcal{L} = \sum_c \sum_{\mathbf{p}} \|\mathbf{L}_c^*(\mathbf{p}) - \mathbf{L}_c^{t-1}(\mathbf{p})\|_2^2 + \sum_c \sum_{\mathbf{p}} \|\mathbf{S}_c^*(\mathbf{p}) - \mathbf{S}_c^{t-1}(\mathbf{p})\|_2^2$$

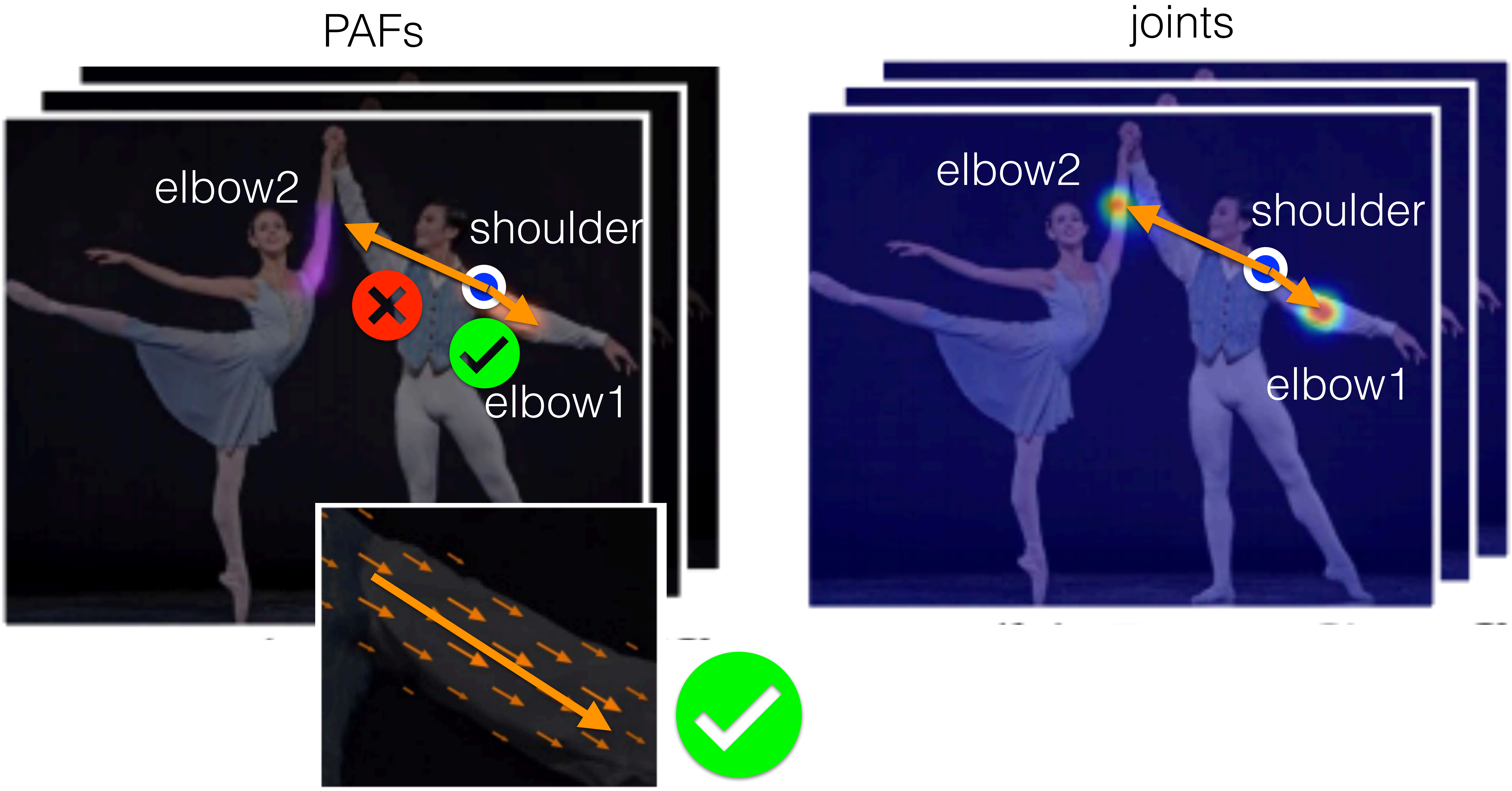
OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf



OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf

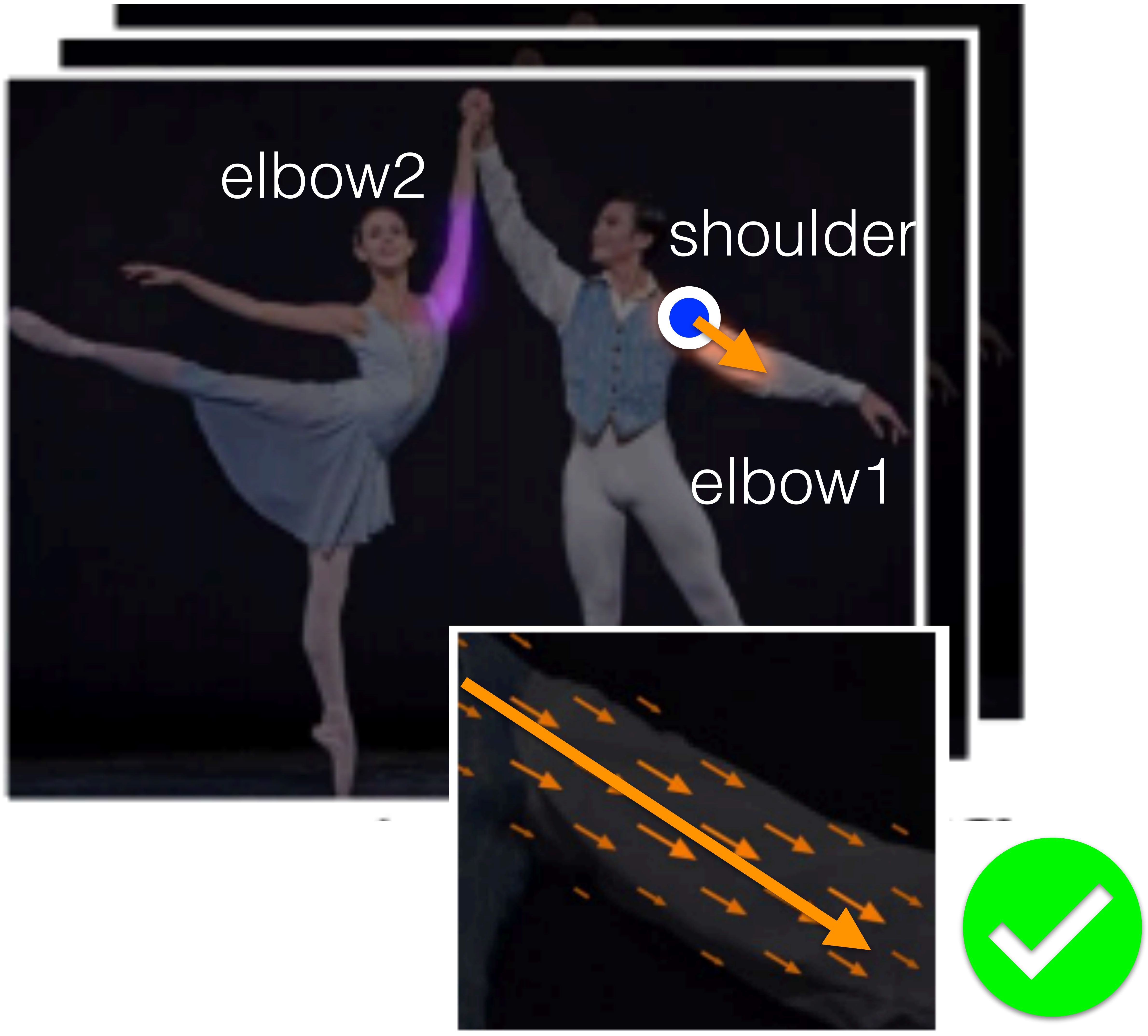


OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf



OpenPose [Cao, TPAMI, 2019]
https://arxiv.org/pdf/1812.08008.pdf

PAFs



output

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

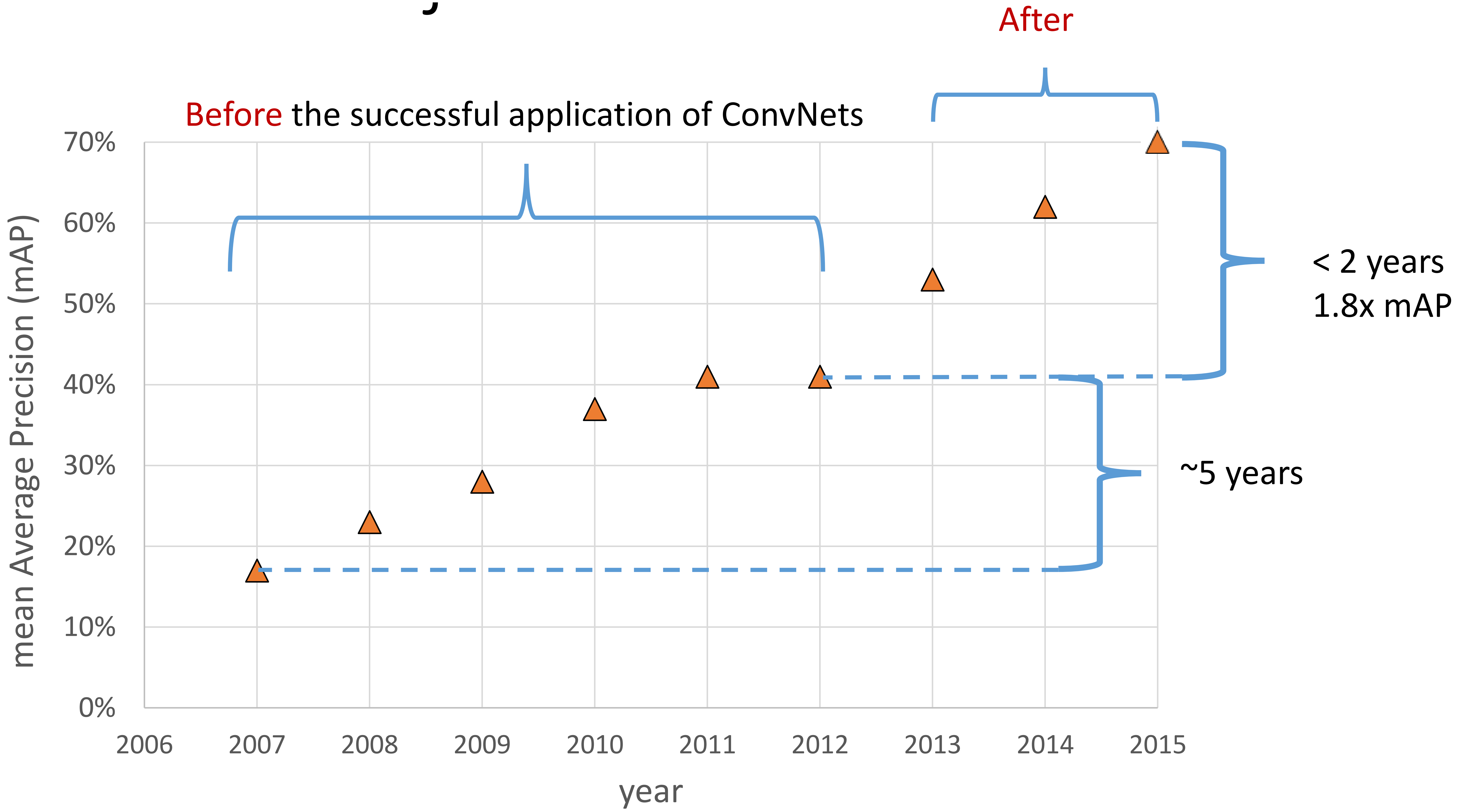
- Realtime Multi-Person 2D Human Pose Estimation using Part Affinity Fields, CVPR 2017 Oral
  https://www.youtube.com/watch?v=pW6nZXeWlGM

- Integral Human Pose Regression [Sun ECCV 2018]
  Microsoft Research
  https://arxiv.org/abs/1711.08229
  https://github.com/JimmySuen/integral-human-pose

Outline

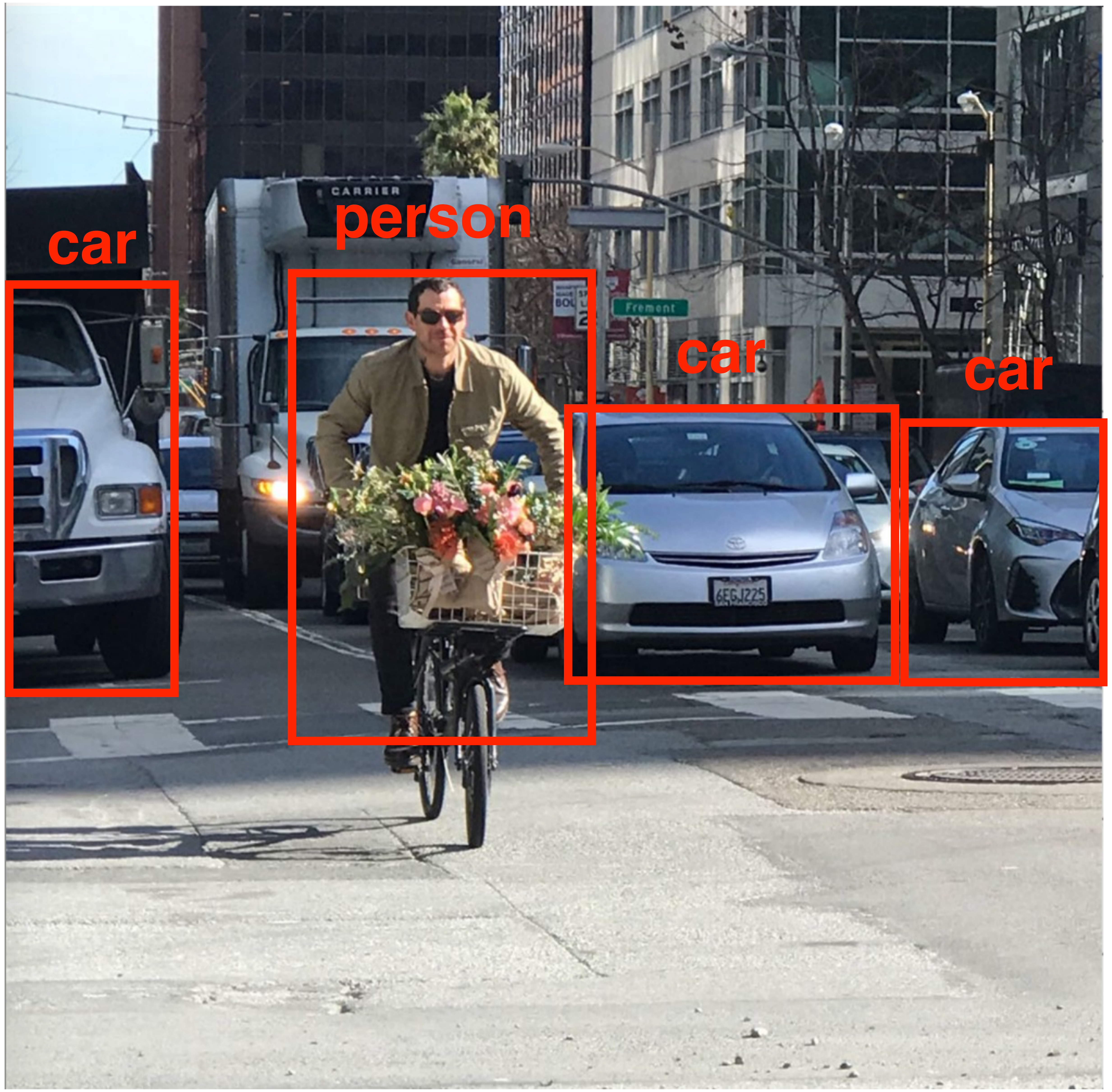
- Architectures of classification networks
- Architectures of segmentation networks
- Architectures of regression networks
- Architectures of detection networks
- Architectures of feature matching networks

Pascal VOC object detection challenge

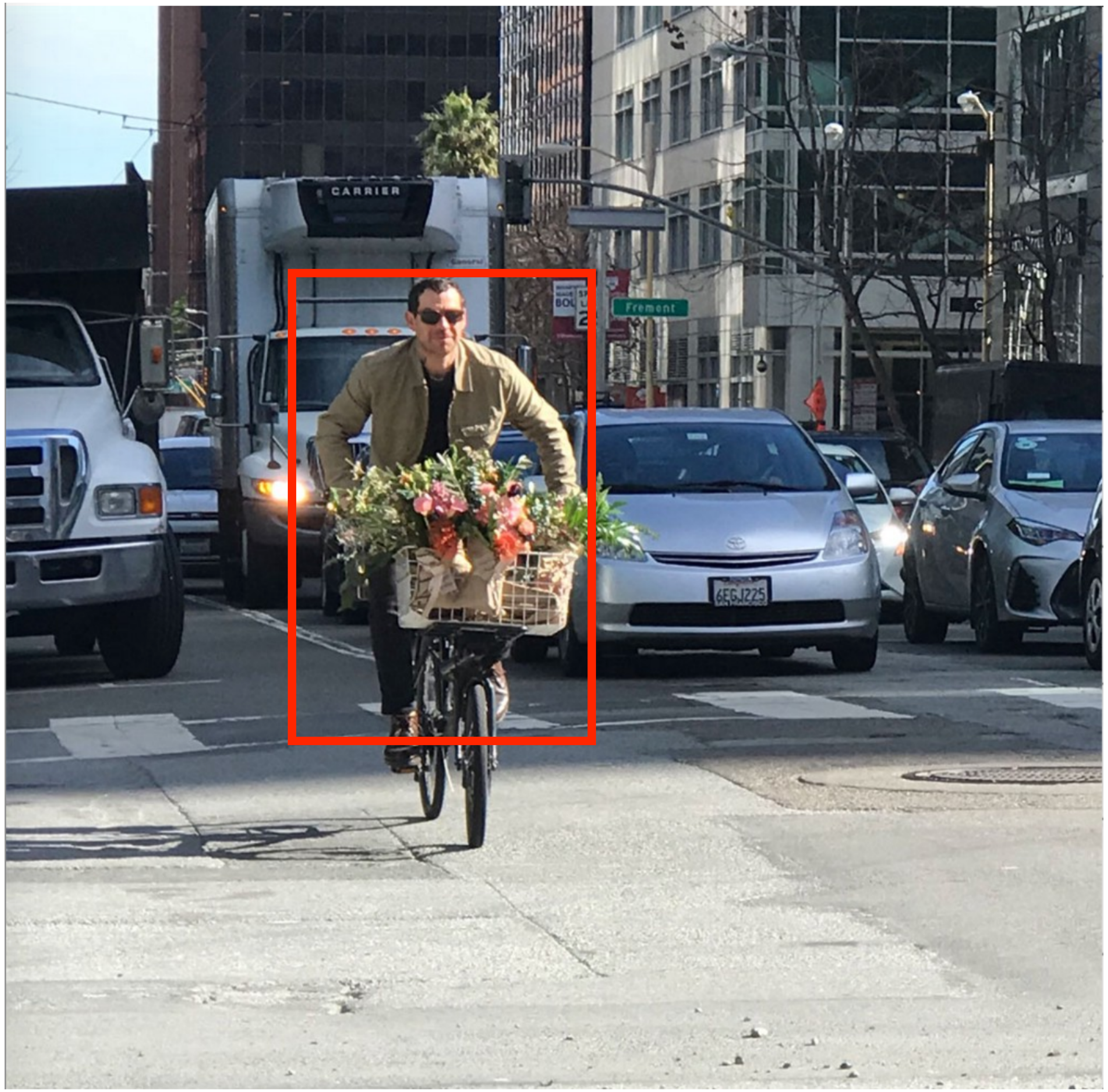


Object detection

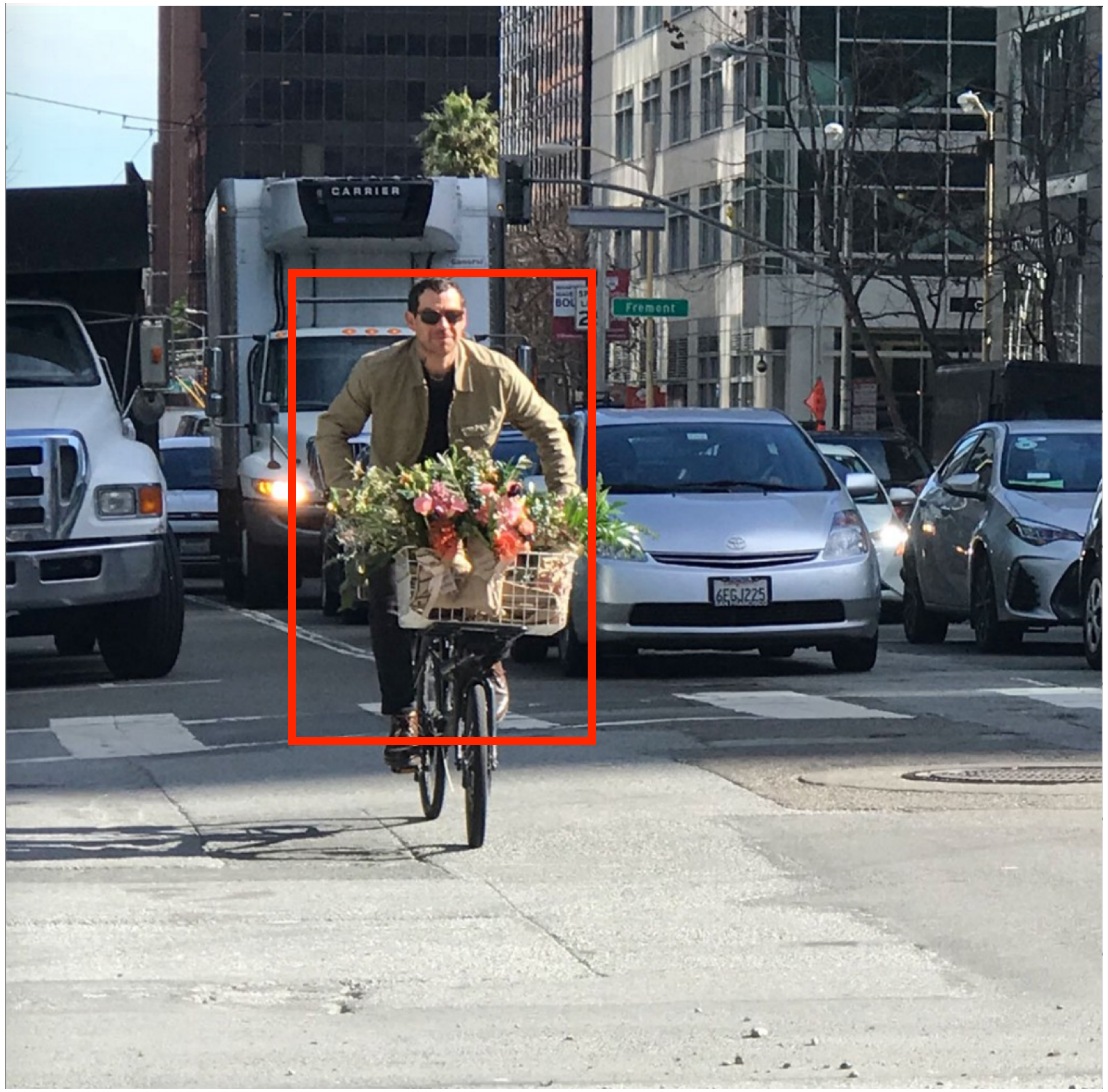
Object detection

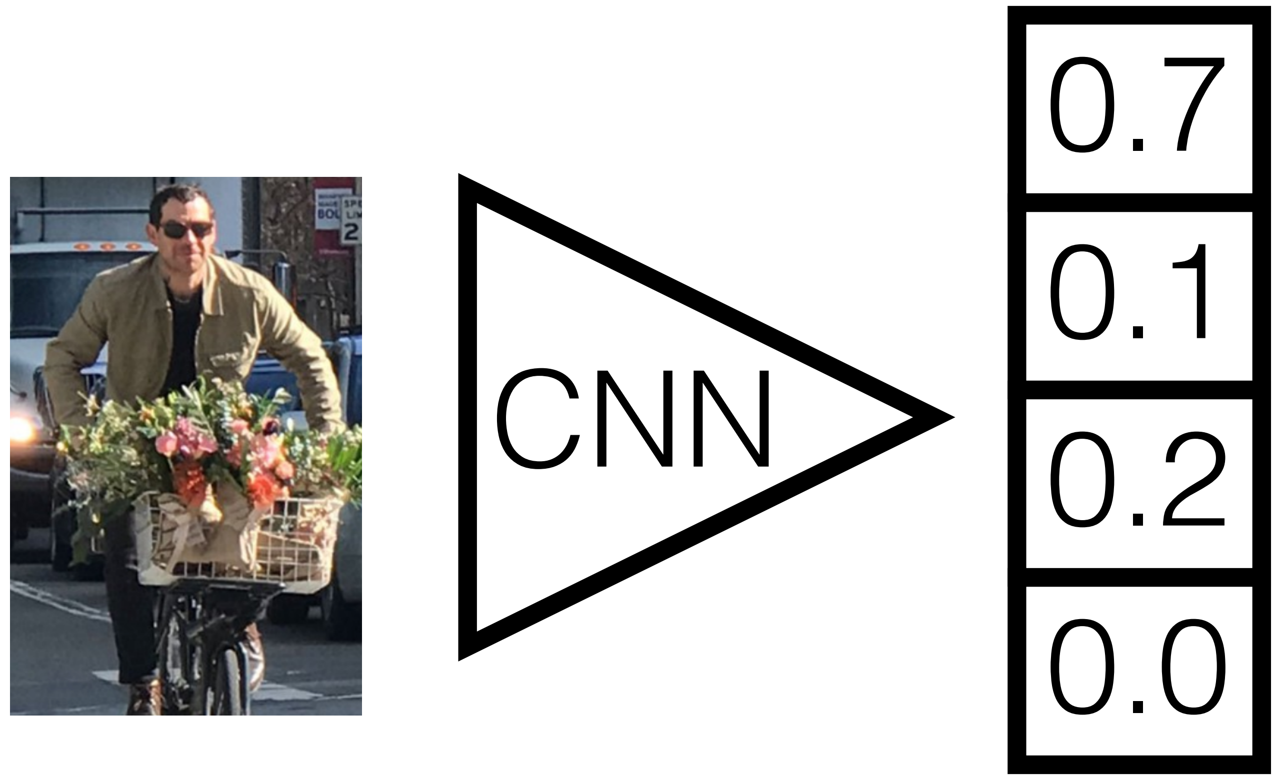


Object detection



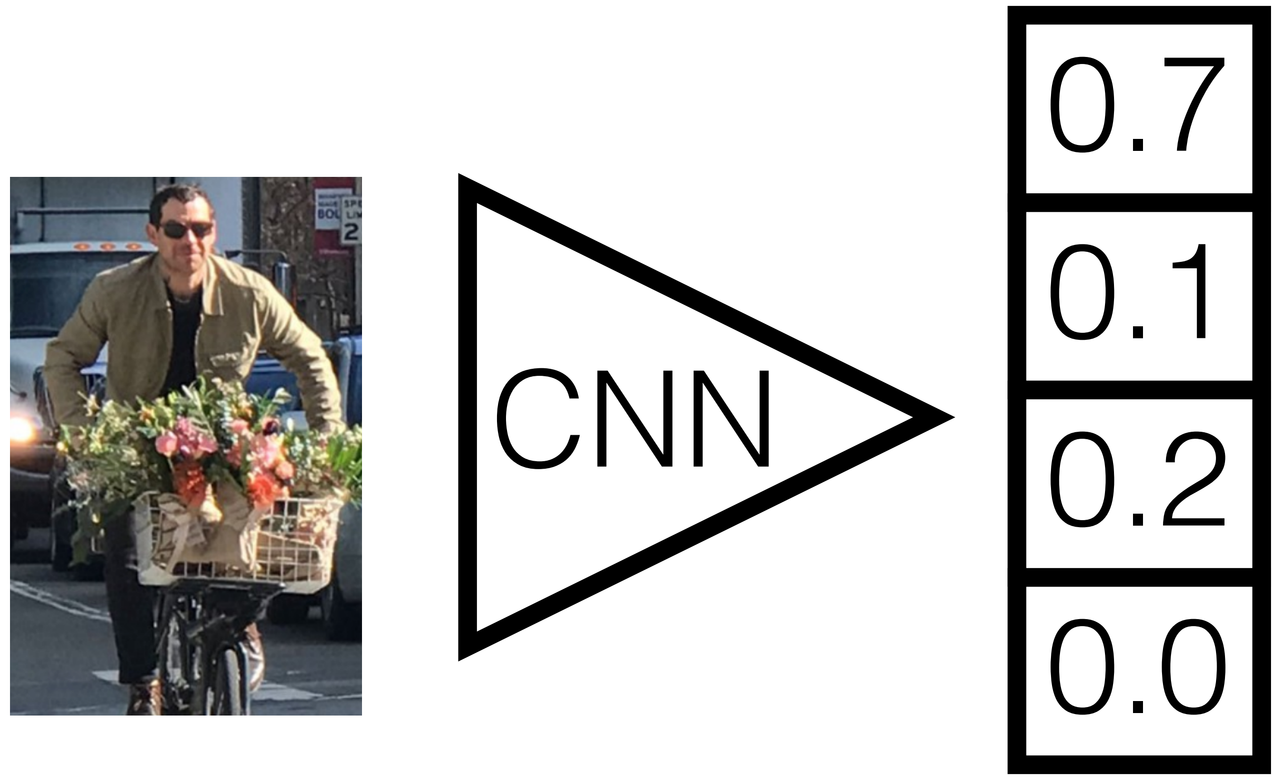
Object detection



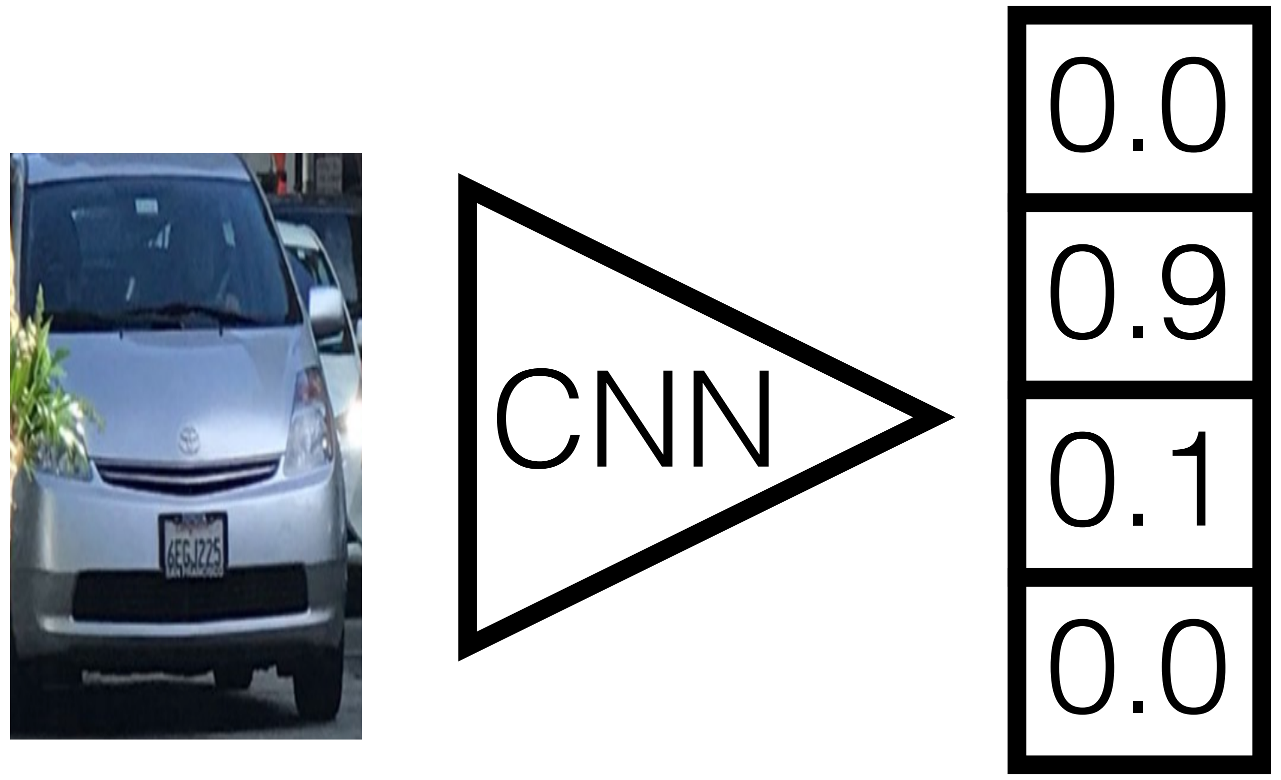


class: person

Object detection



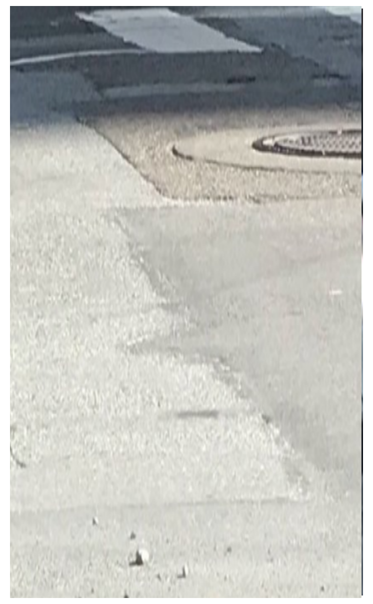
Object detection

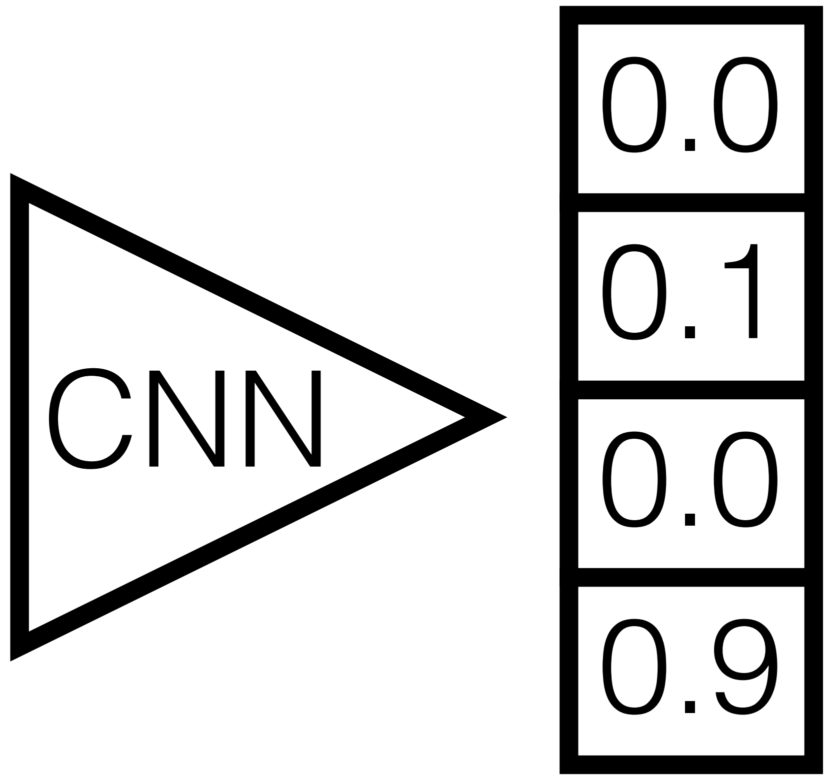


class: car

Object detection

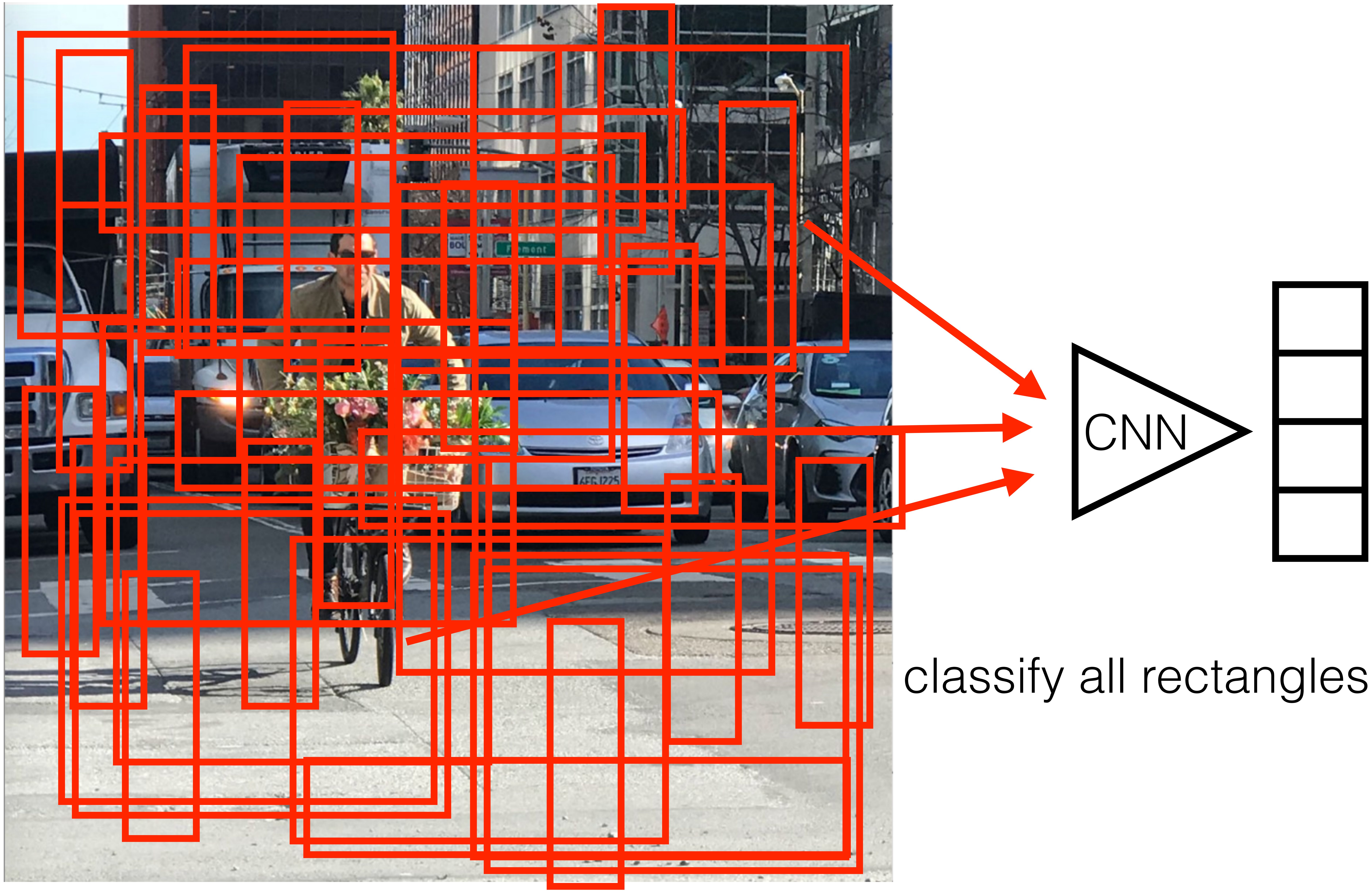
Object detection





class: background

Object detection

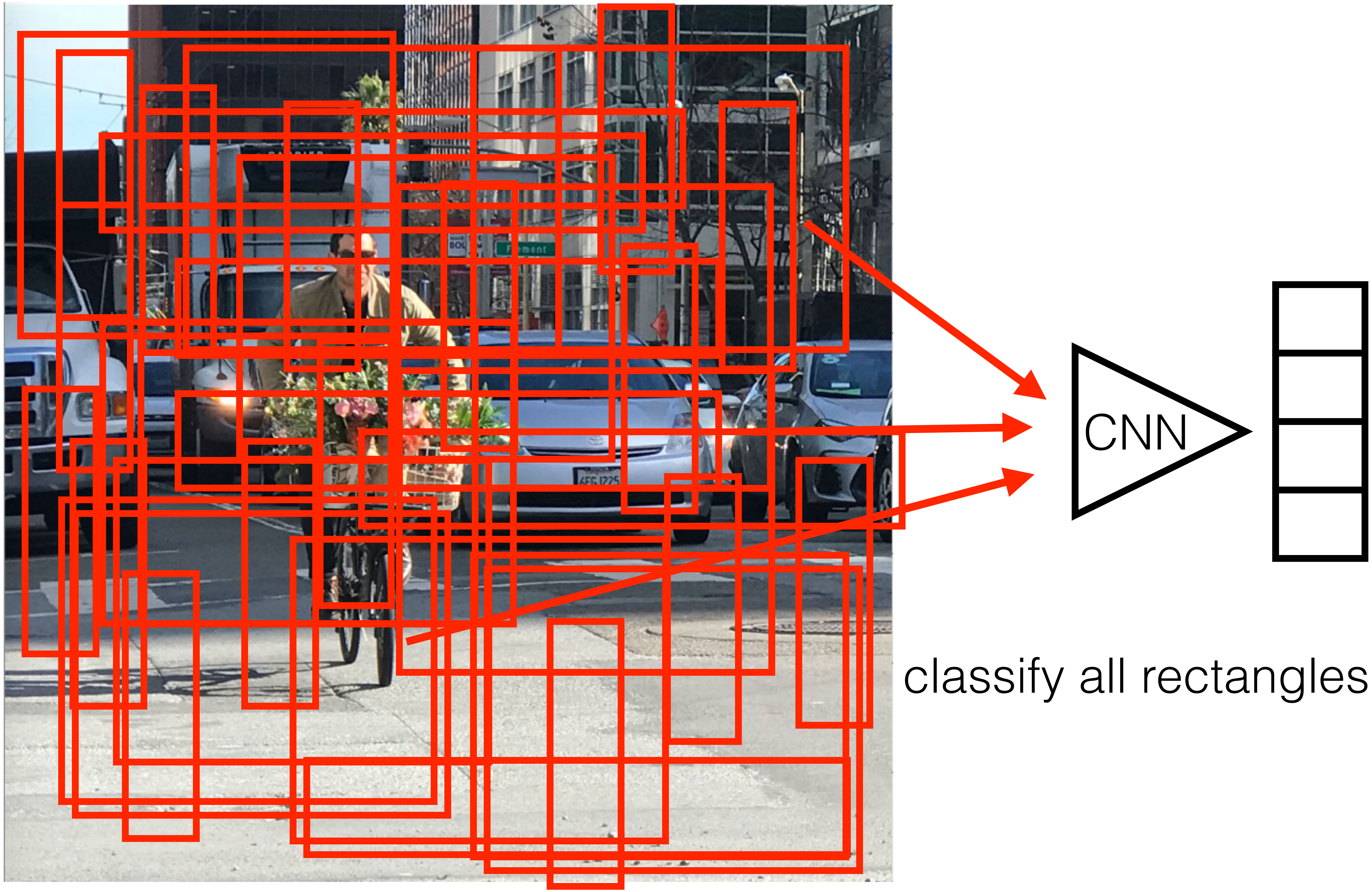


classify all rectangles

Object detection

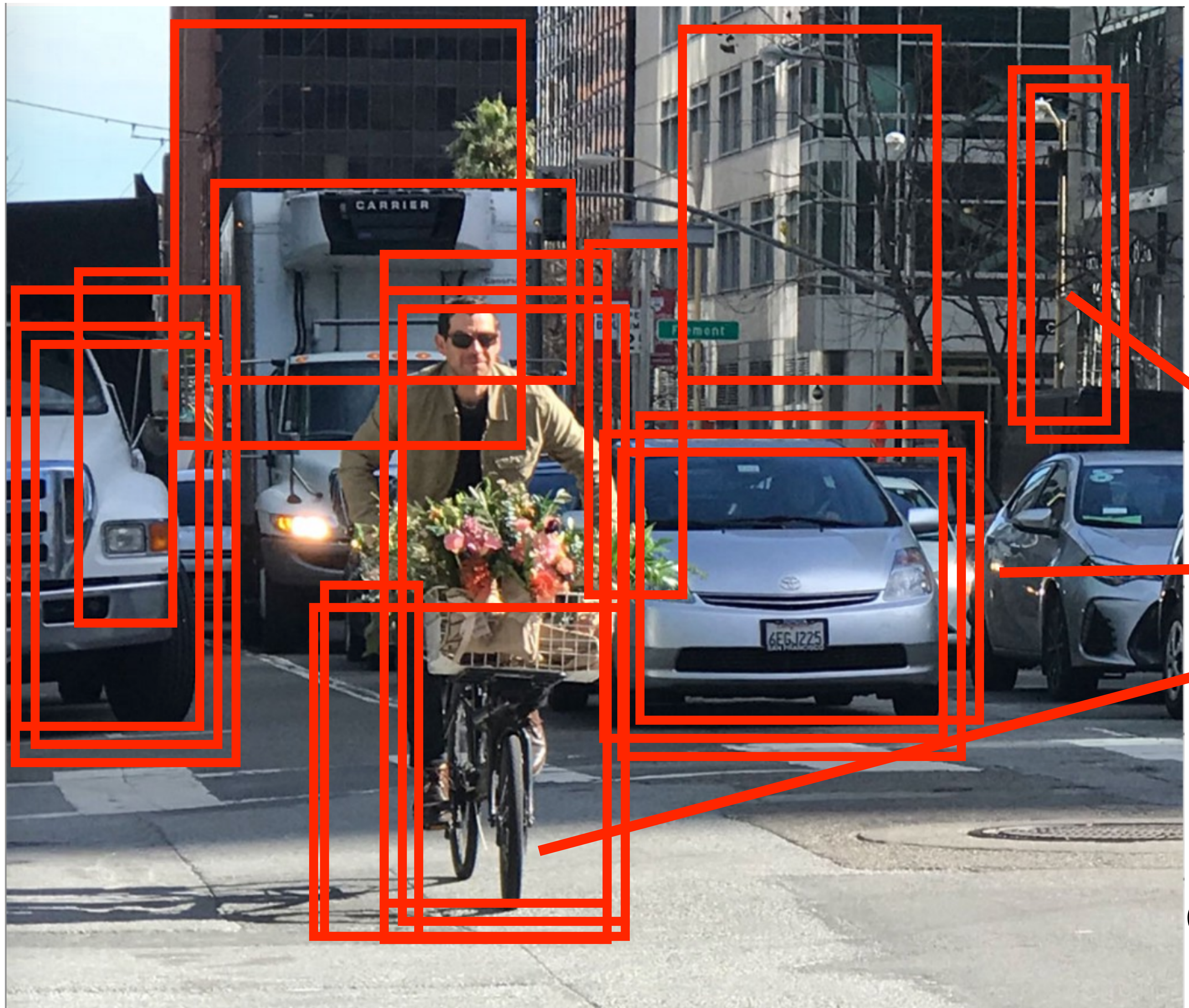
- Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  H x W x Aspect_Ratio x Scales x 0.001 sec = **months**

Object detection

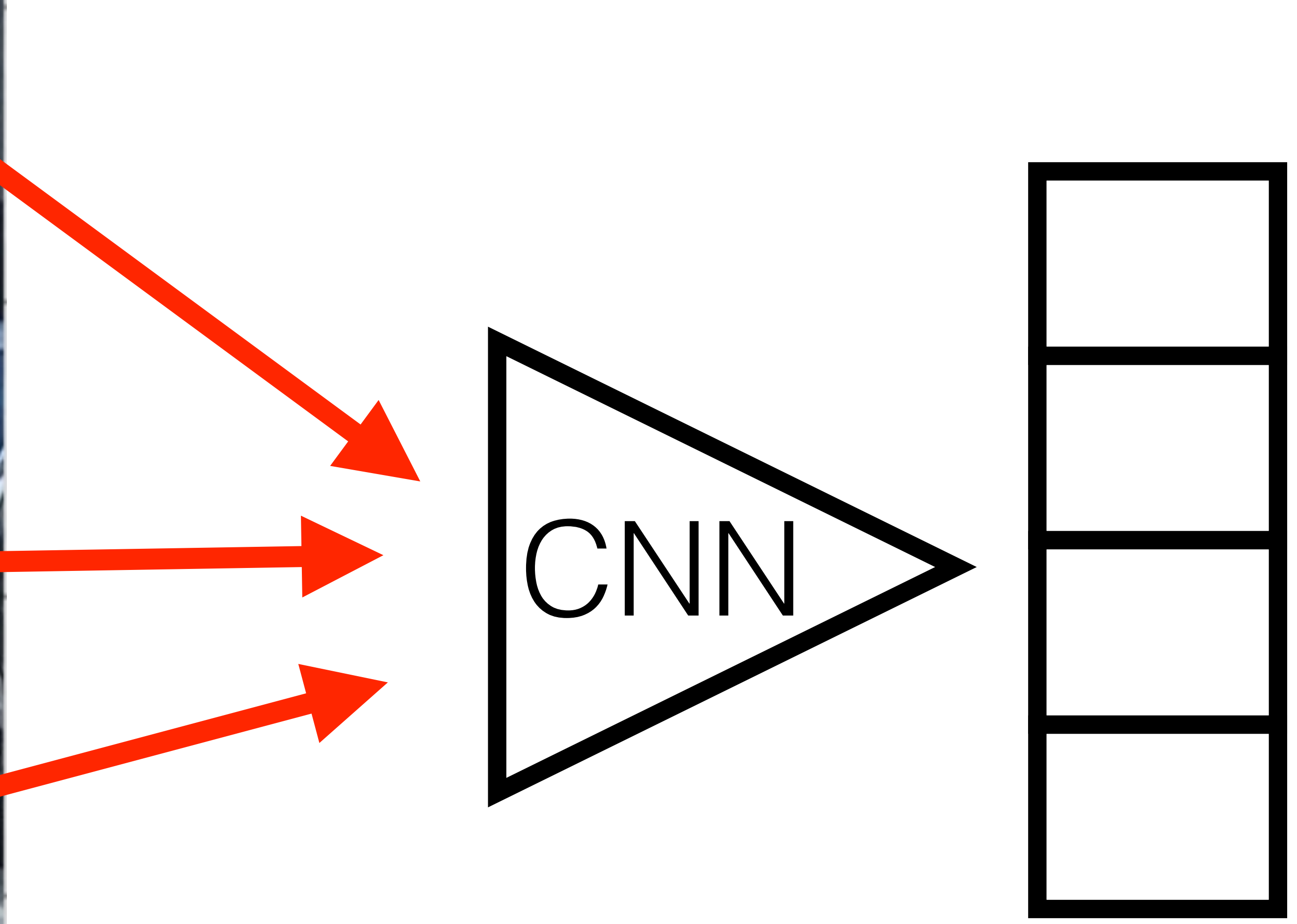


classify all rectangles

Object detection



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

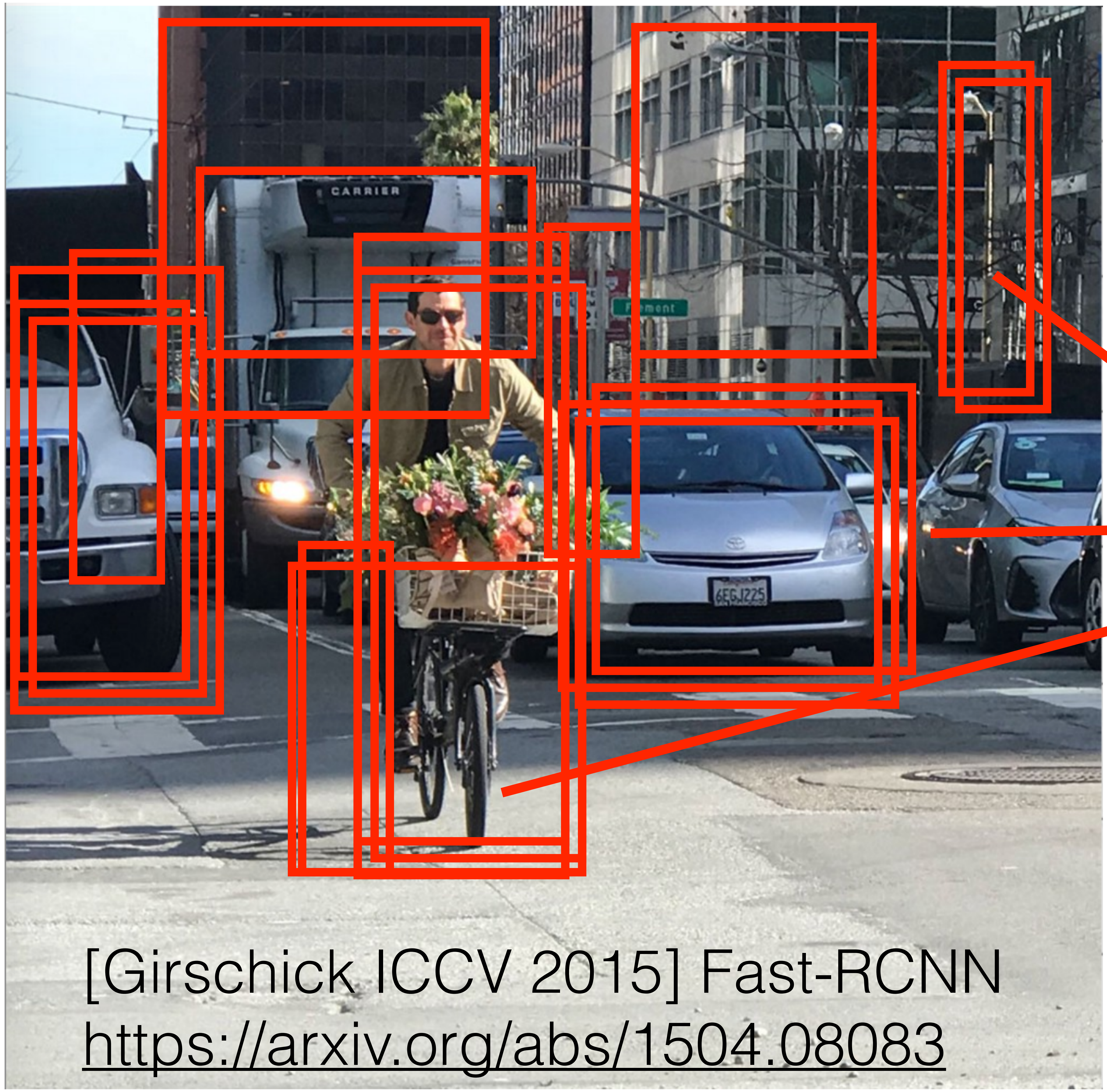


classify + align only 2k region proposals

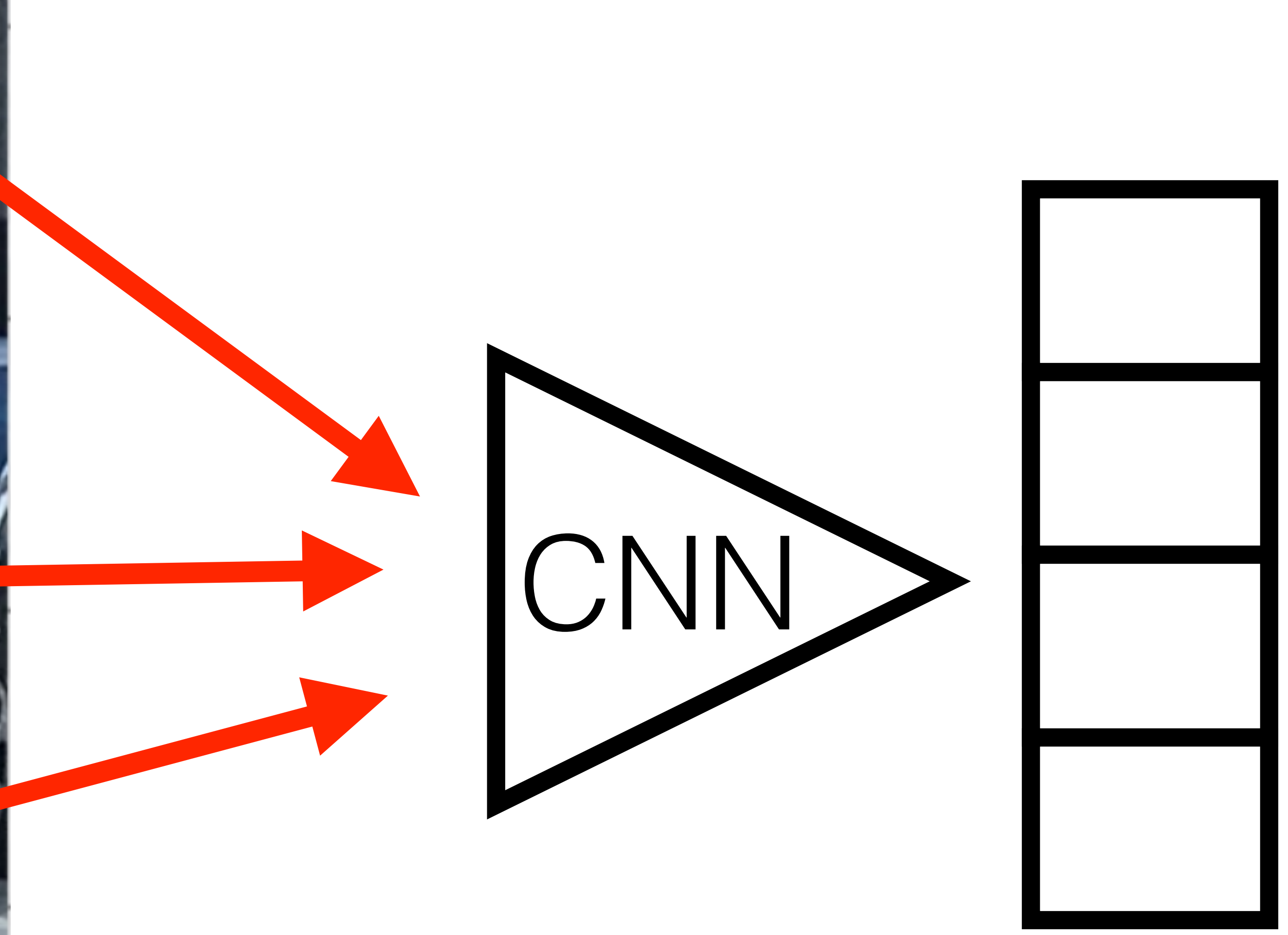
Object detection

- 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

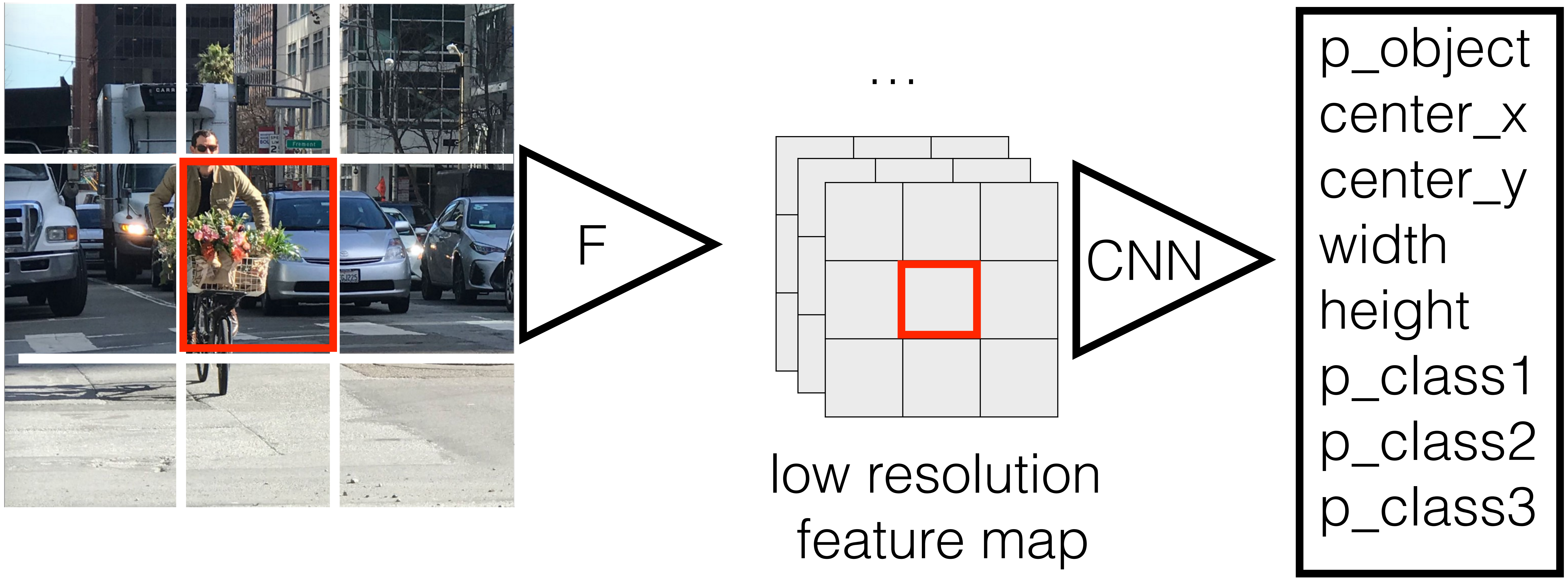


[Girschick ICCV 2015] Fast-RCNN
https://arxiv.org/abs/1504.08083



The selective search for region proposals is computational bottleneck !!!

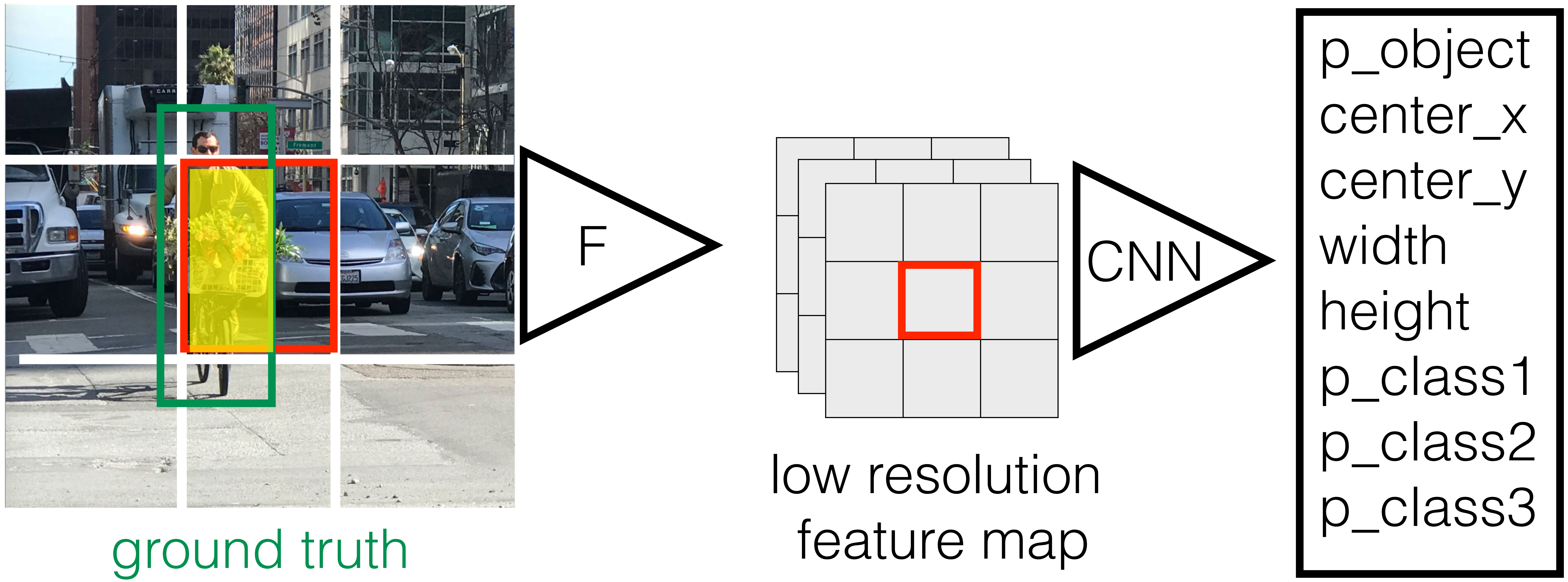
YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



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

YOLO and Faster RCNN architectures

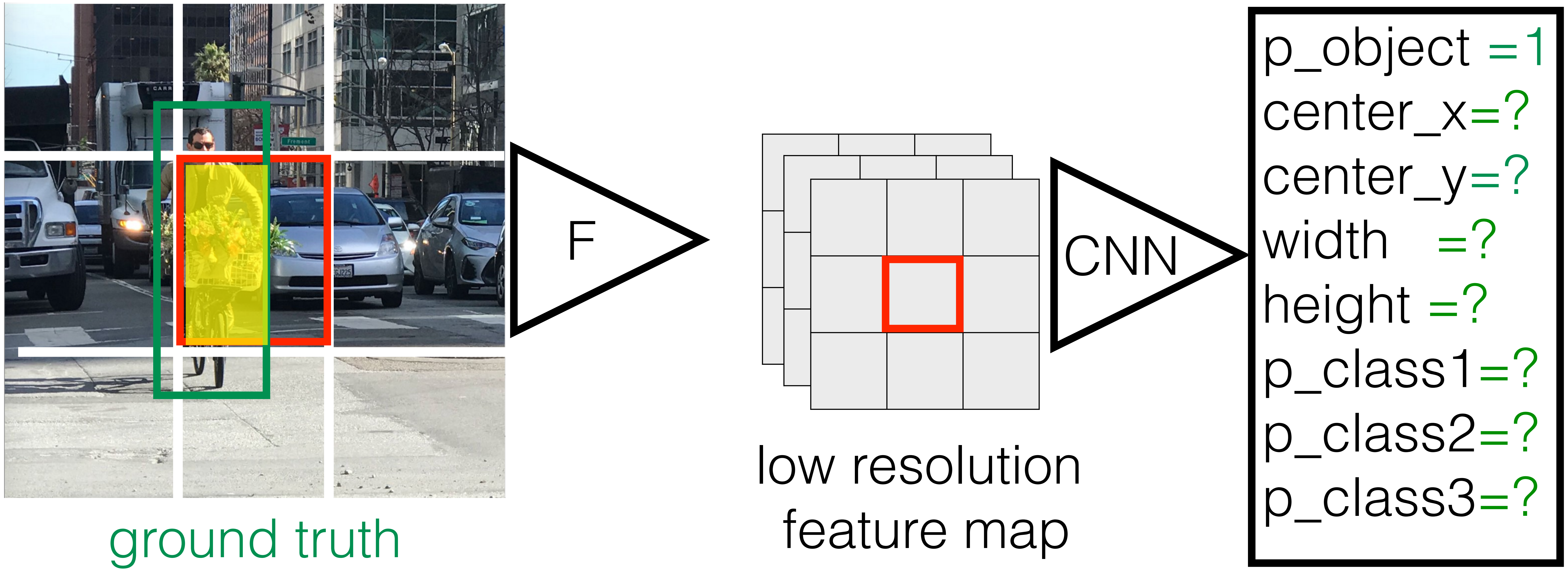
https://arxiv.org/abs/1506.01497



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

YOLO and Faster RCNN architectures

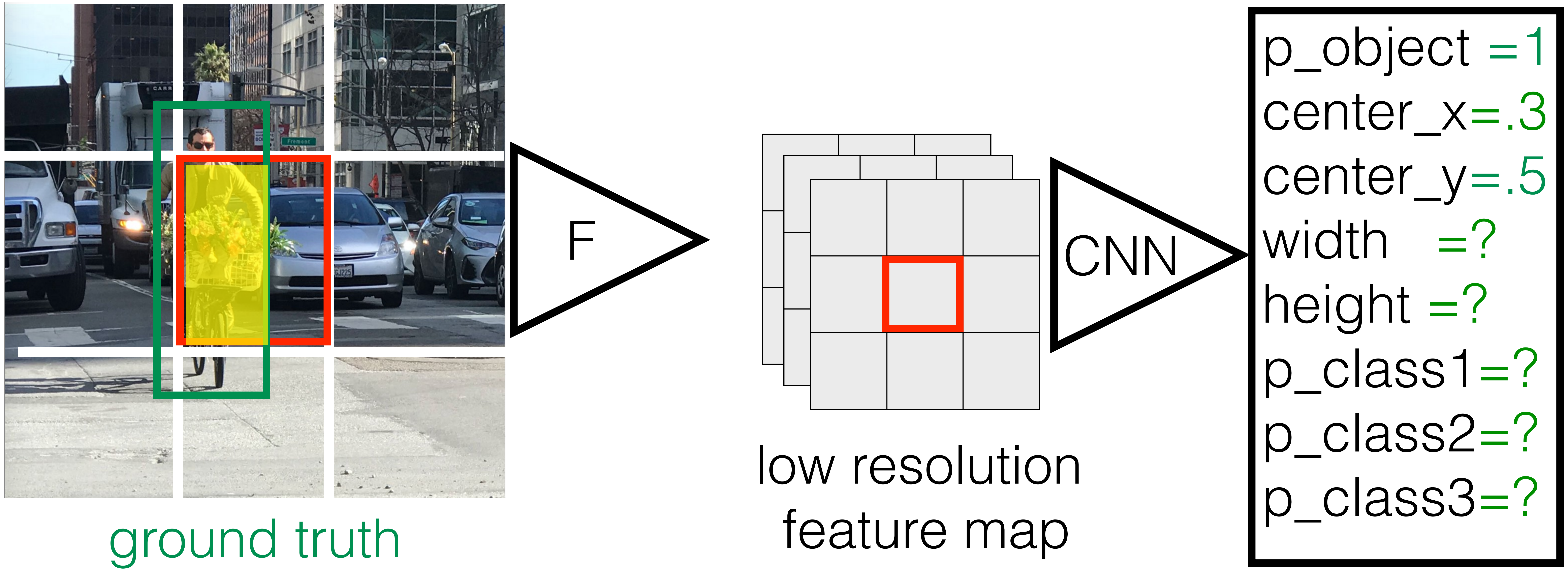
https://arxiv.org/abs/1506.01497



- 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

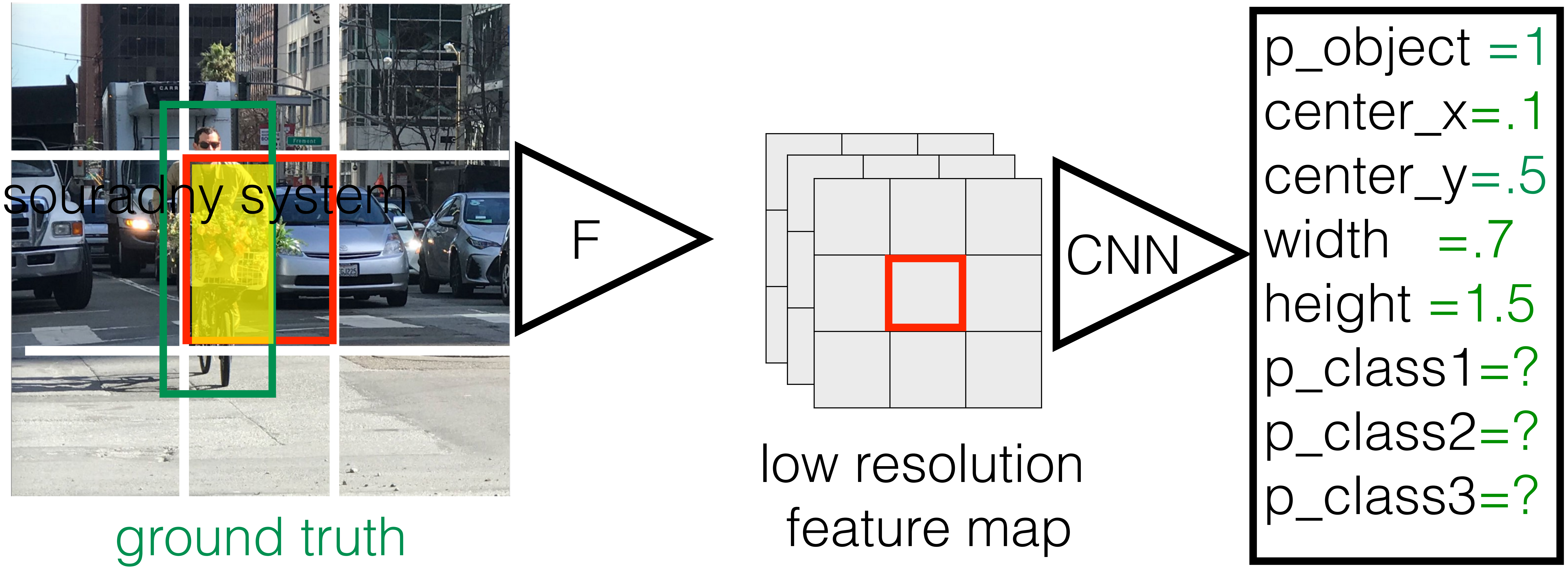


- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- ground truth: bbs with IoU>0.7 are objects,
  bbs with IoU<0.3 not objects

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497

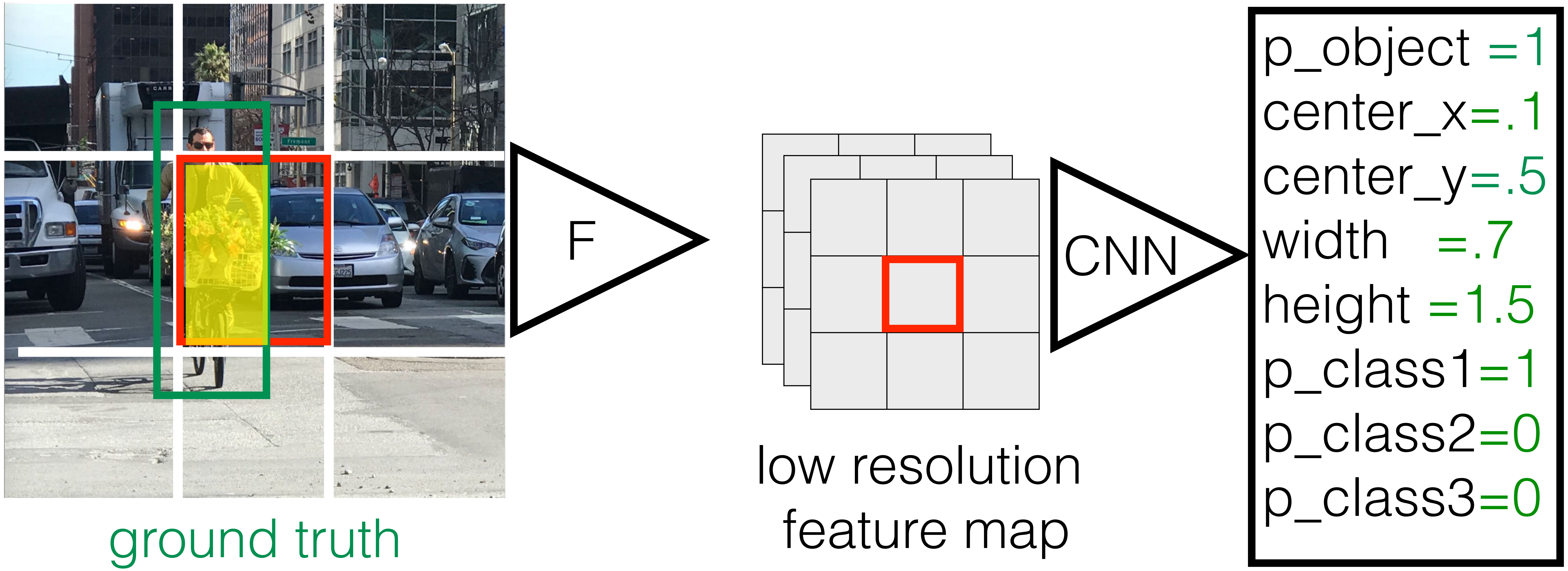
Dokreslit souradny system



- 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

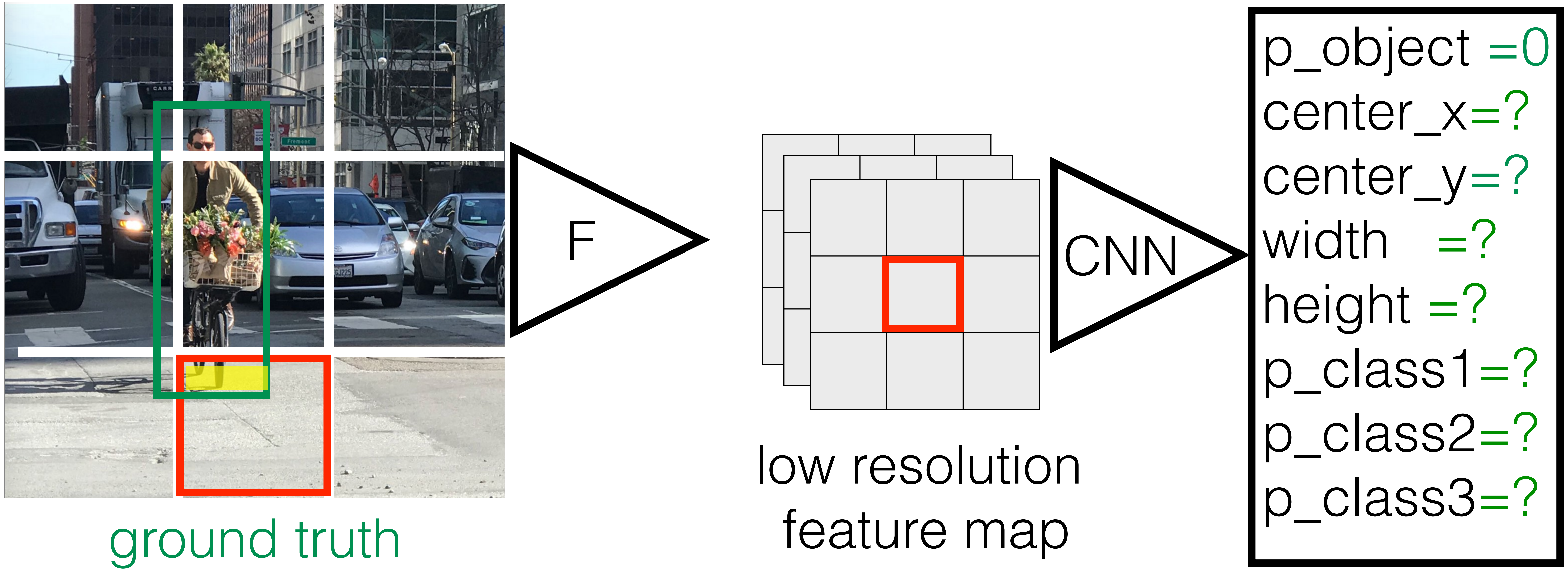
https://arxiv.org/abs/1506.01497



- 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

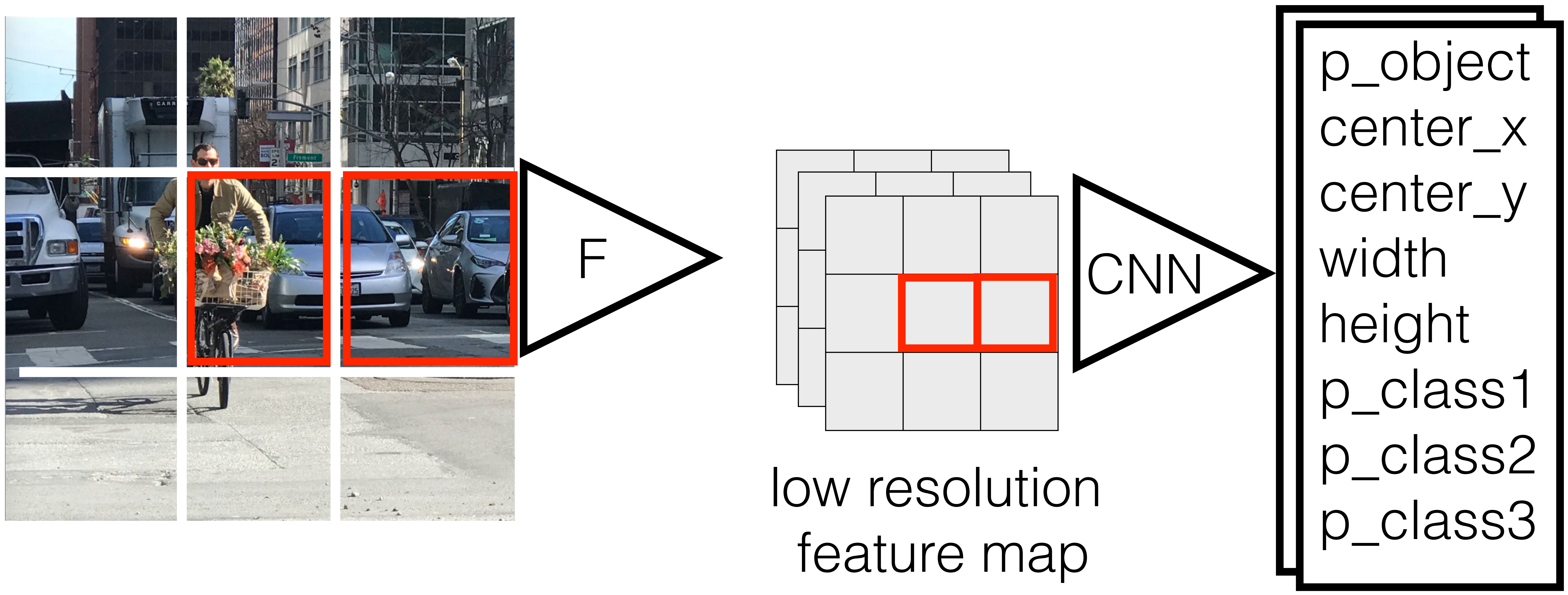


ground truth

low resolution feature map

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

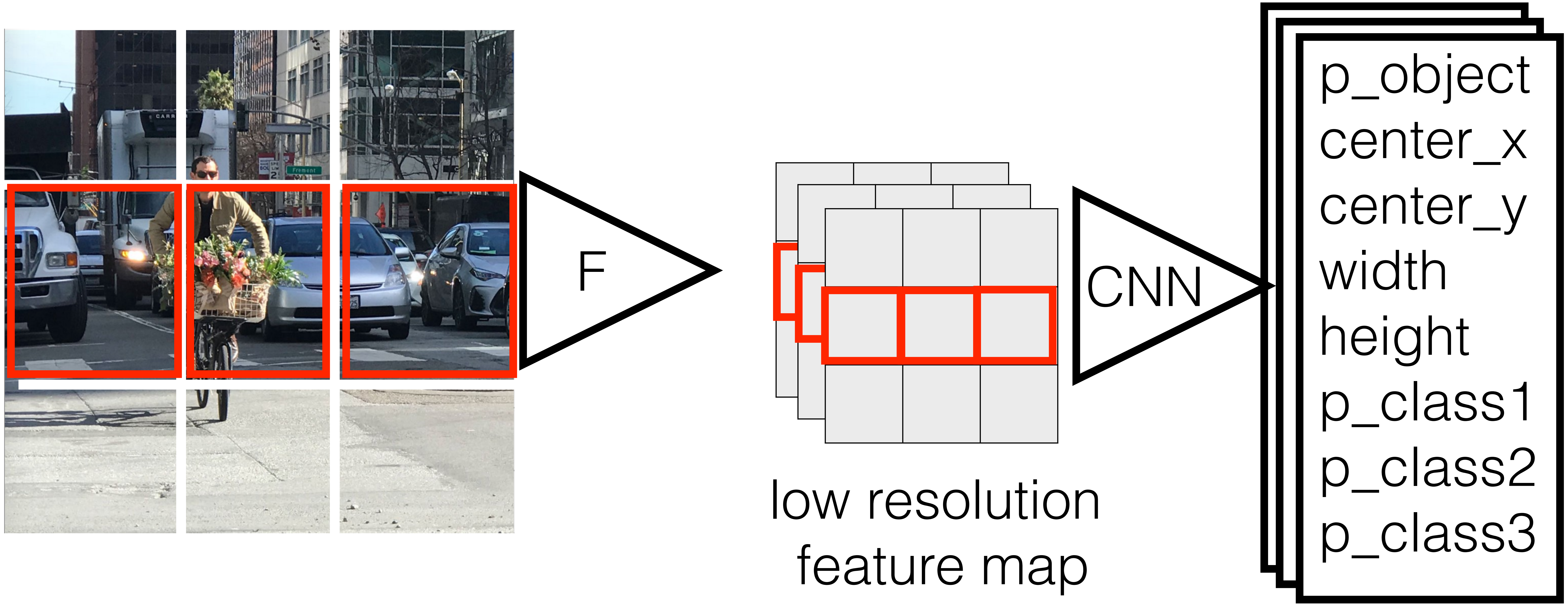
YOLO and Faster RCNN architectures
https://arxiv.org/abs/1506.01497



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

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497

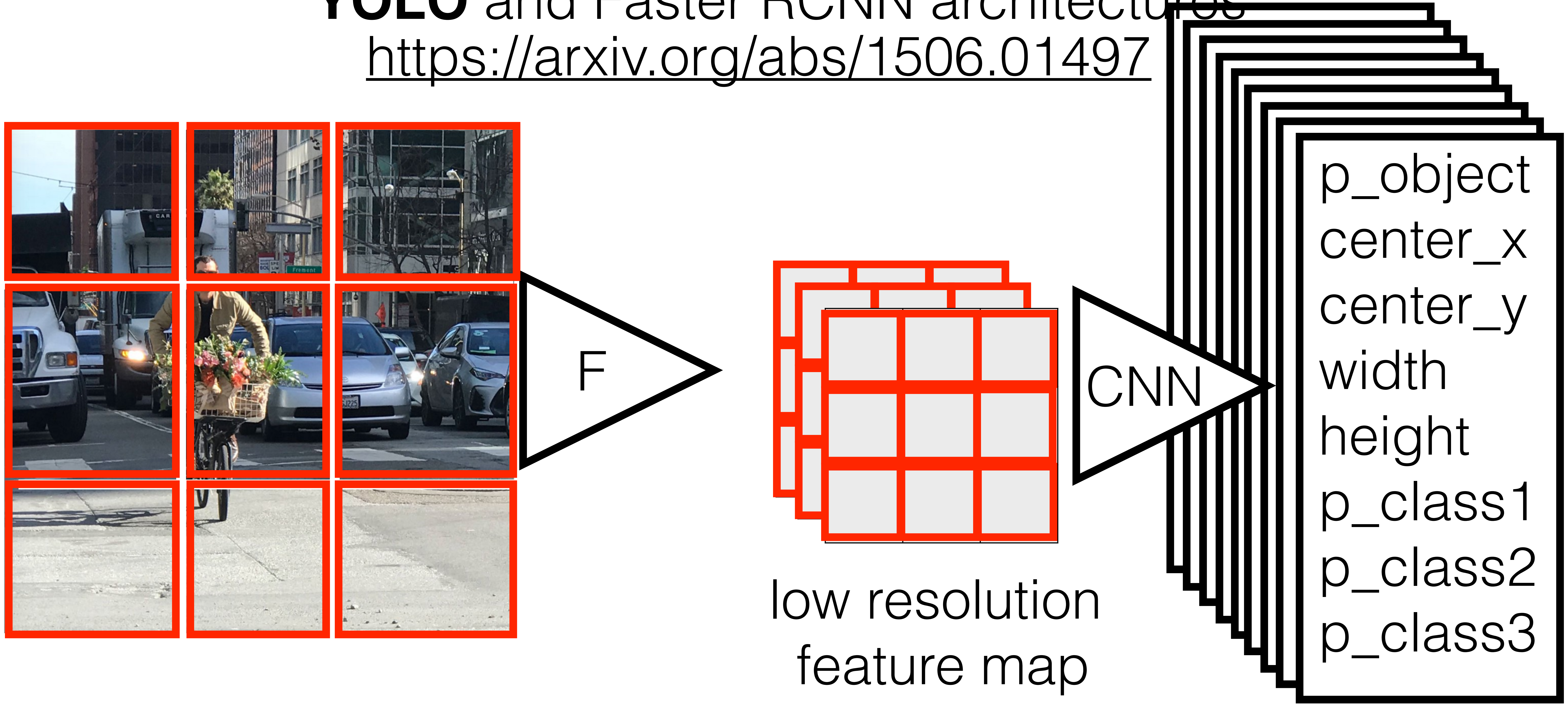


- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- each sub-image has its own output

Do you see any problem?

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497

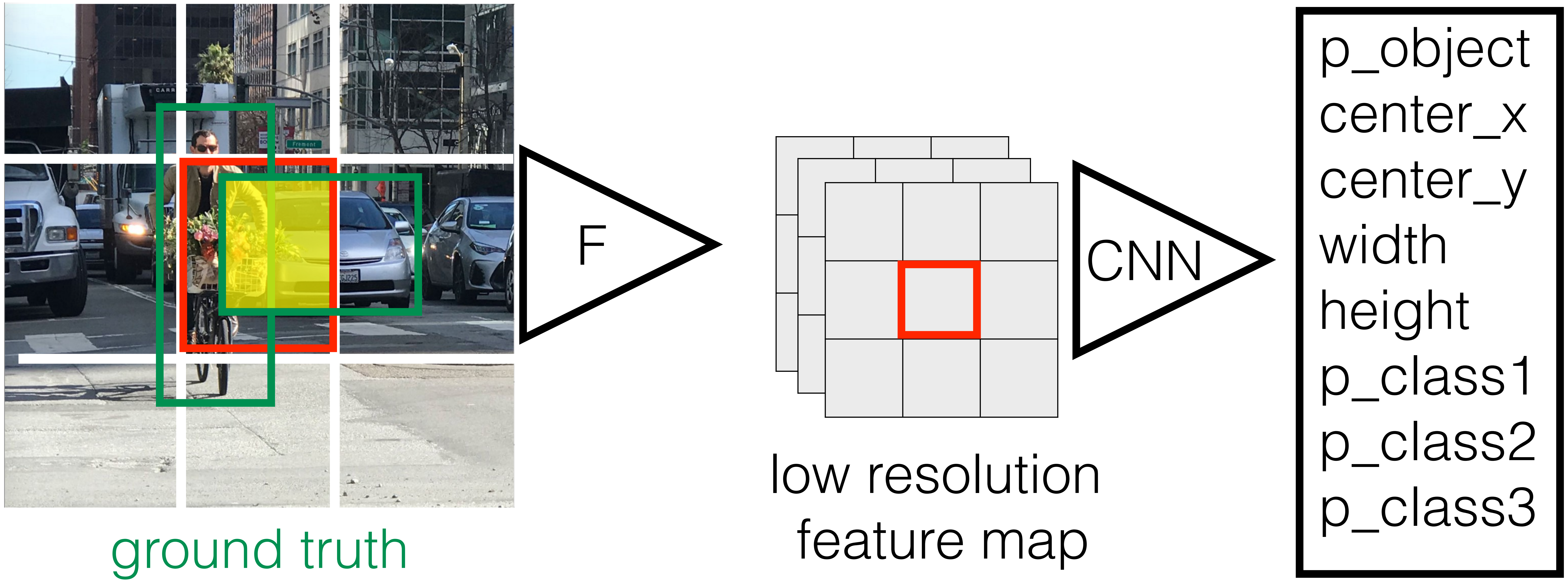


- divide image into 3x3 sub images
- predict relative position, objectness, class for each sub-im
- each sub-image has its own output

Do you see any problem?

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497



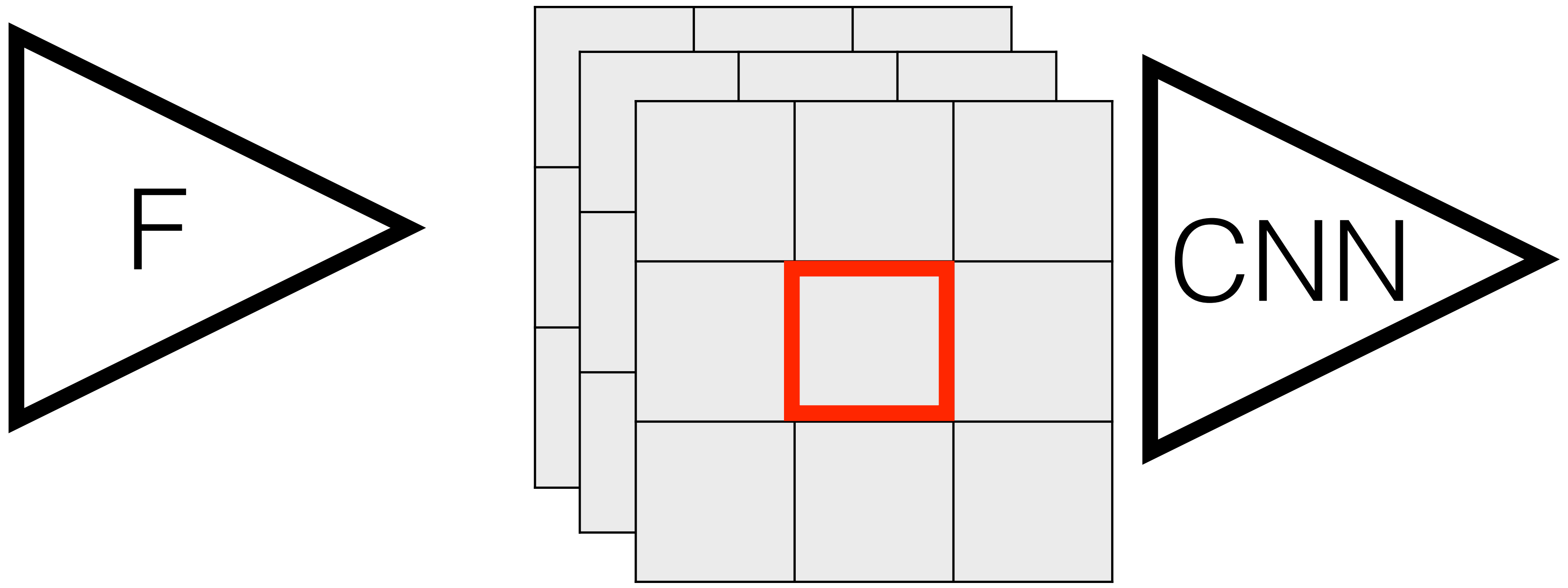
ground truth

- 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
  bbs with IoU<0.3 not objects one sub-im

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497

ground truth



low resolution feature map

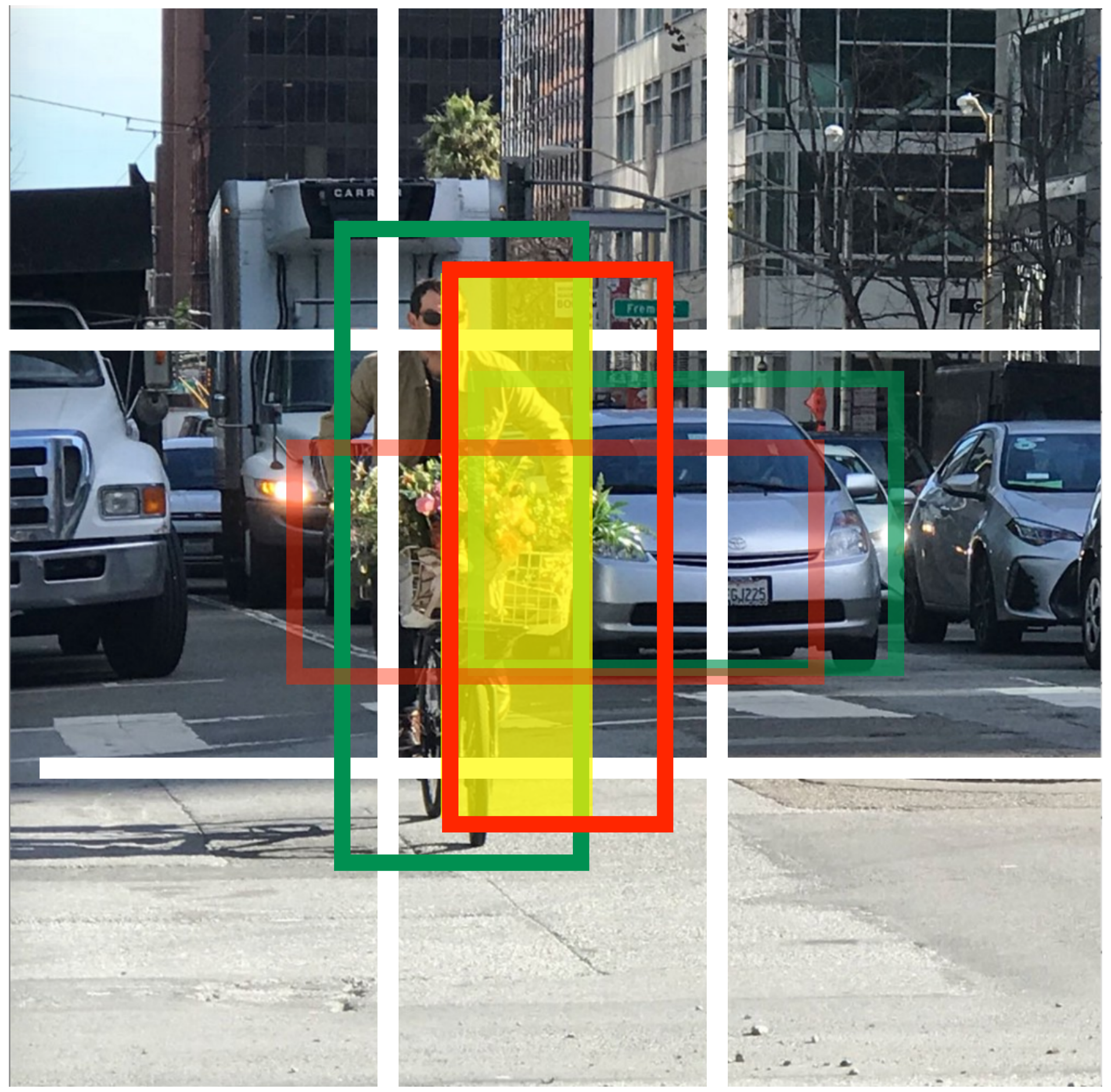
p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

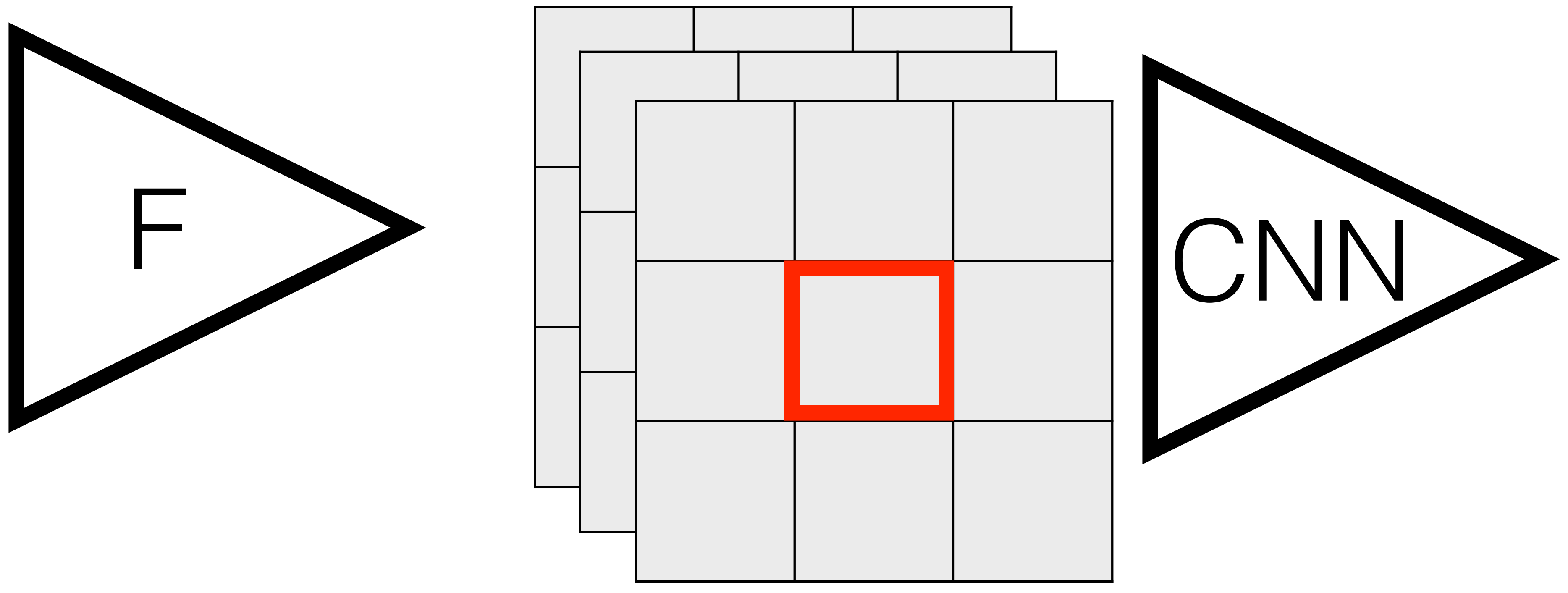
Introduce anchor bounding boxes

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497



ground truth



low resolution feature map

p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

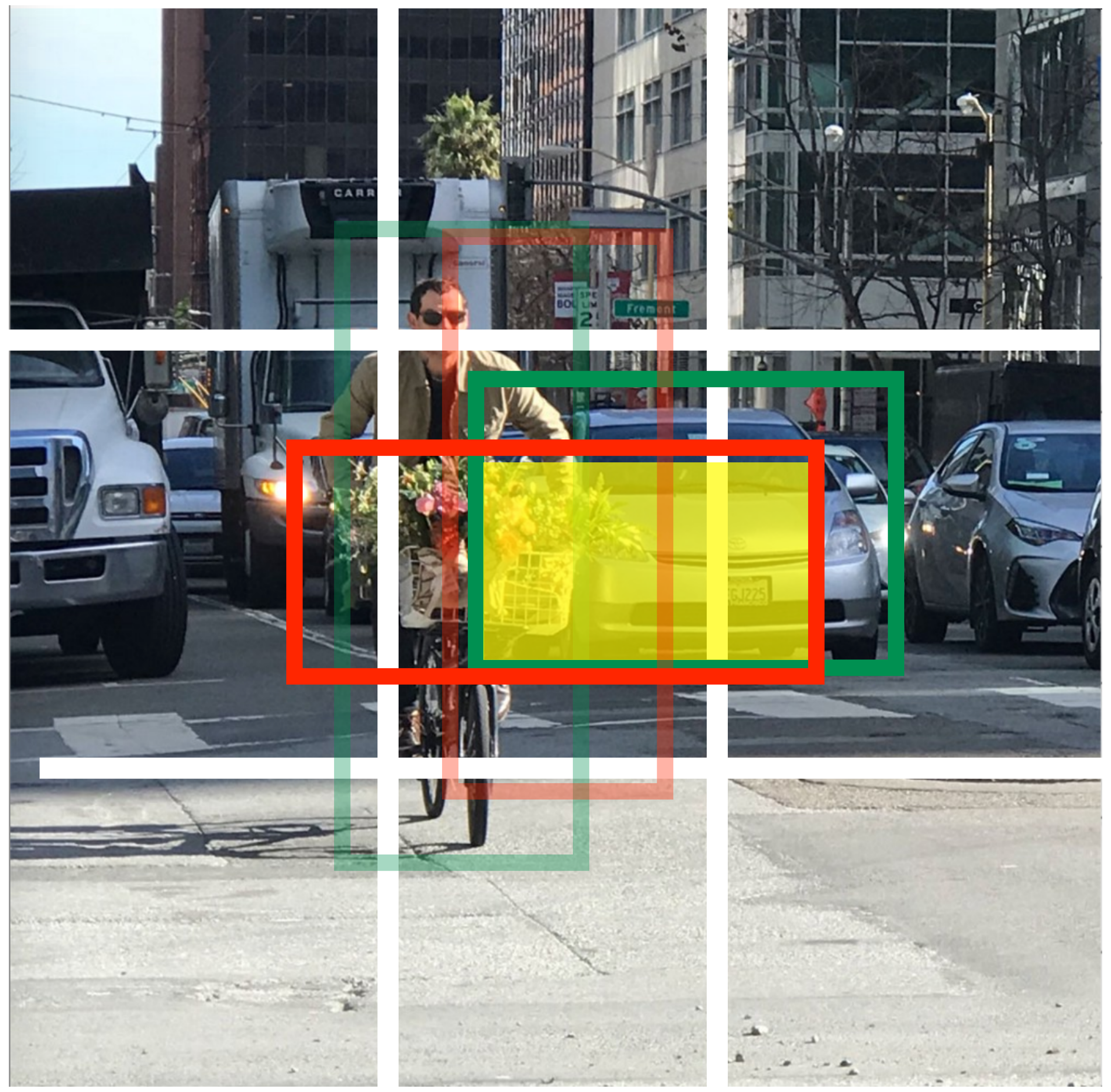
p_object
center_x
center_y
width
height
p_class1
p_class2
p_class3

Introduce anchor bounding boxes

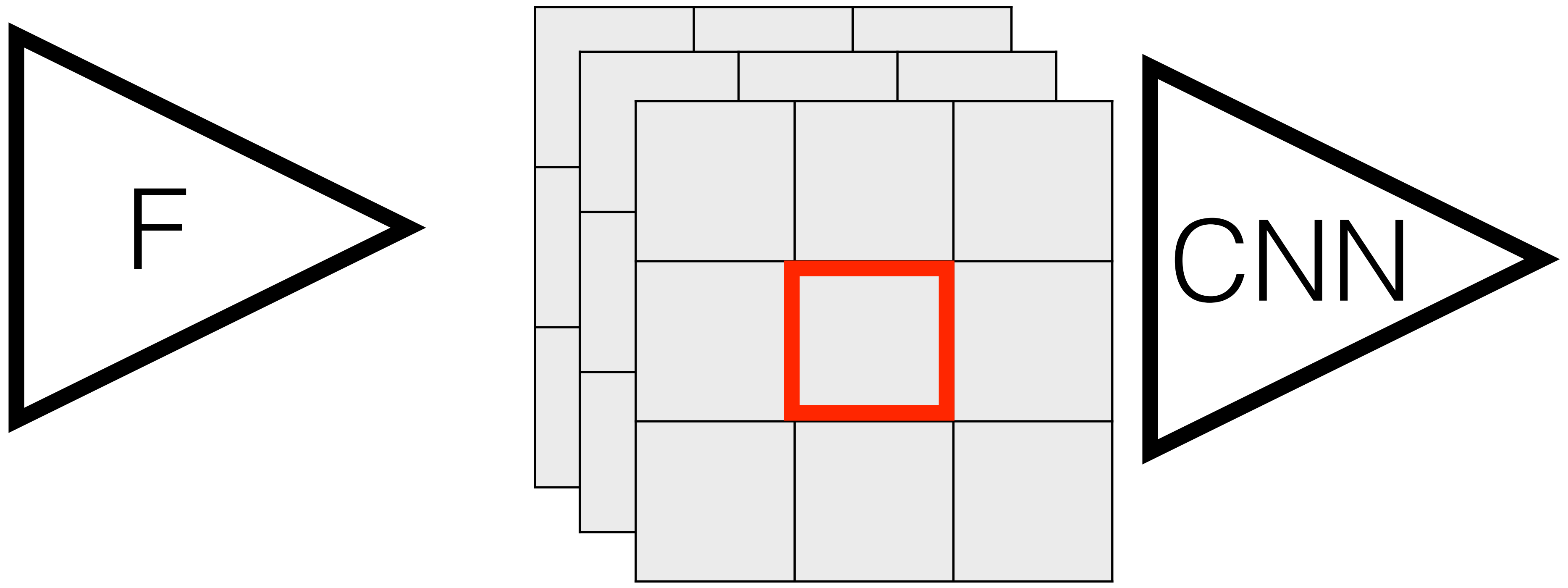
- for each anchor bb CNN predicts:
  - its “alignment with gt” (regression loss)
  - its “objectness”+”class” (classification loss)

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497



ground truth



low resolution feature map

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)

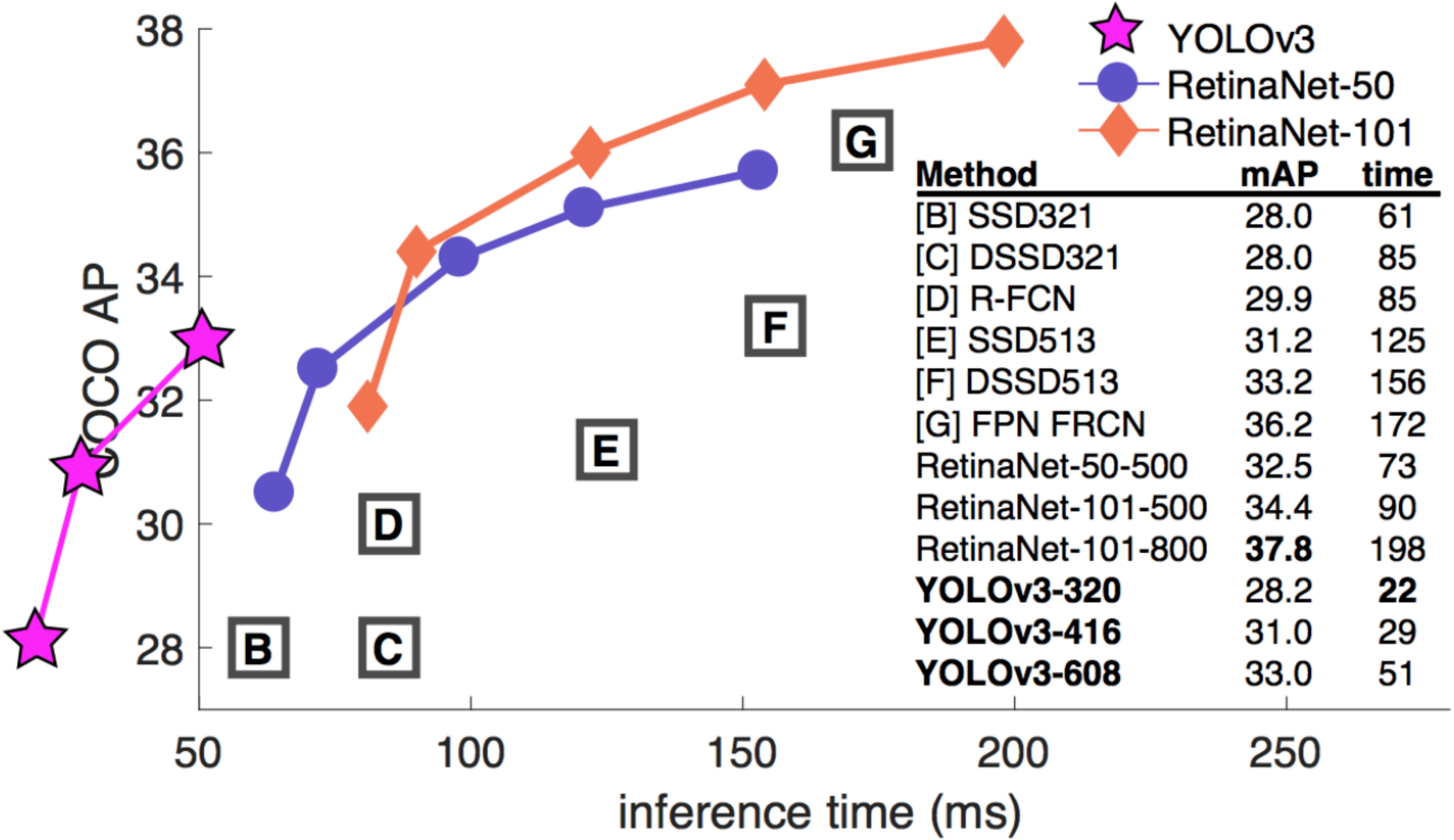
Object detection

- 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**
- Perform region proposal by CNN => **0.05-0.2 sec**

[Faster RCNN 2017] https://arxiv.org/abs/1506.01497 (slower, works for smaller objs)

[Redmont CVPR 2018], https://arxiv.org/abs/1804.02767 (faster, small obj. problems)

Method	**mAP-50**	**time**
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

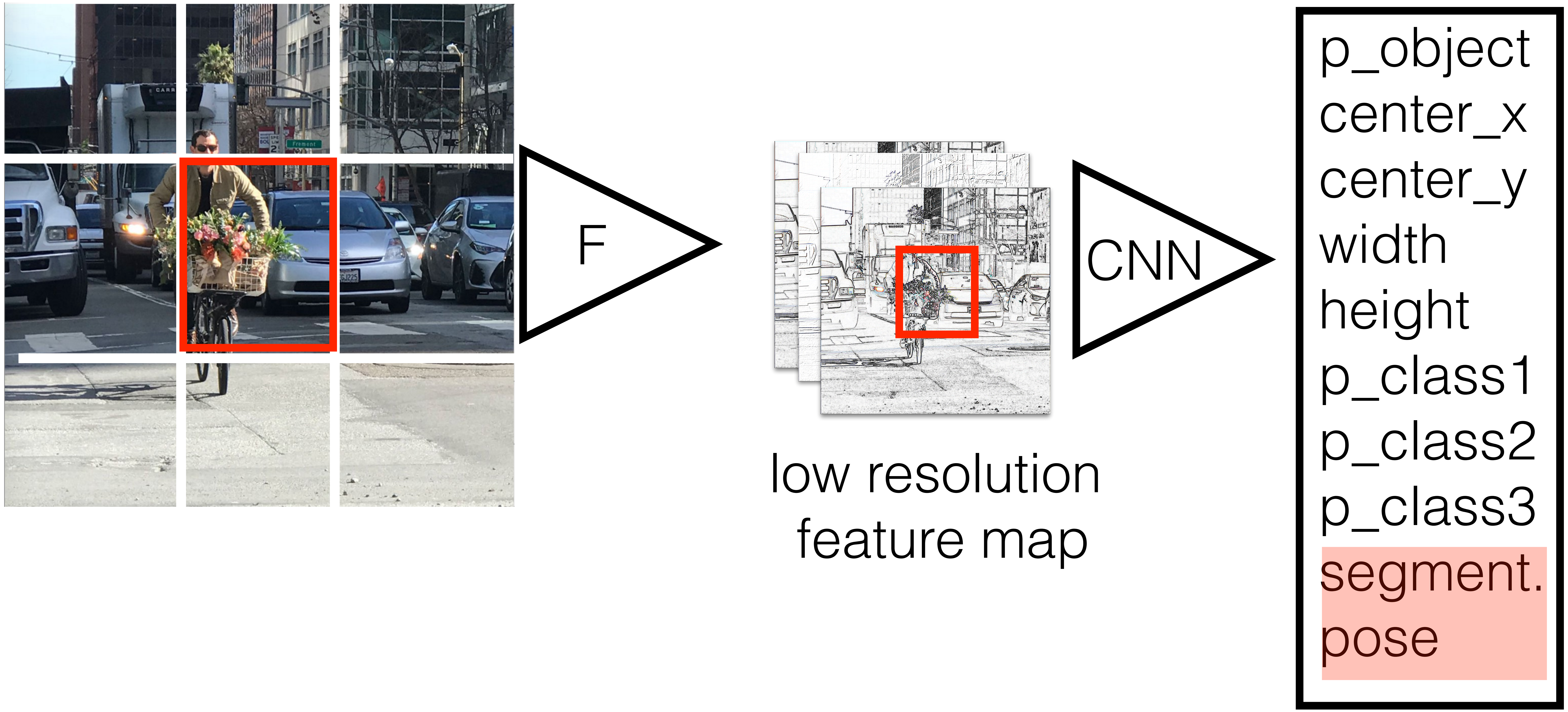

YOLOv3
RetinaNet-50
RetinaNet-101
Method mAP time
[B] SSD321 28.0 61
[C] DSSD321 28.0 85
[D] R-FCN 29.9 85
[E] SSD513 31.2 125
[F] DSSD513 33.2 156
[G] FPN FRCN 36.2 172
RetinaNet-50-500 32.5 73
RetinaNet-101-500 34.4 90
RetinaNet-101-800 37.8 198
YOLOv3-320 28.2 22
YOLOv3-416 31.0 29
YOLOv3-608 33.0 51
COCO AP
inference time (ms)
B
C
D
E
F
G
28
30
32
34
36
38
50
100
150
200
250

Deep convolutional - object detection

[Redmont CVPR 2018], https://arxiv.org/abs/1804.02767
code: https://pjreddie.com/darknet/yolo/

YOLO and Faster RCNN architectures

https://arxiv.org/abs/1506.01497



[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

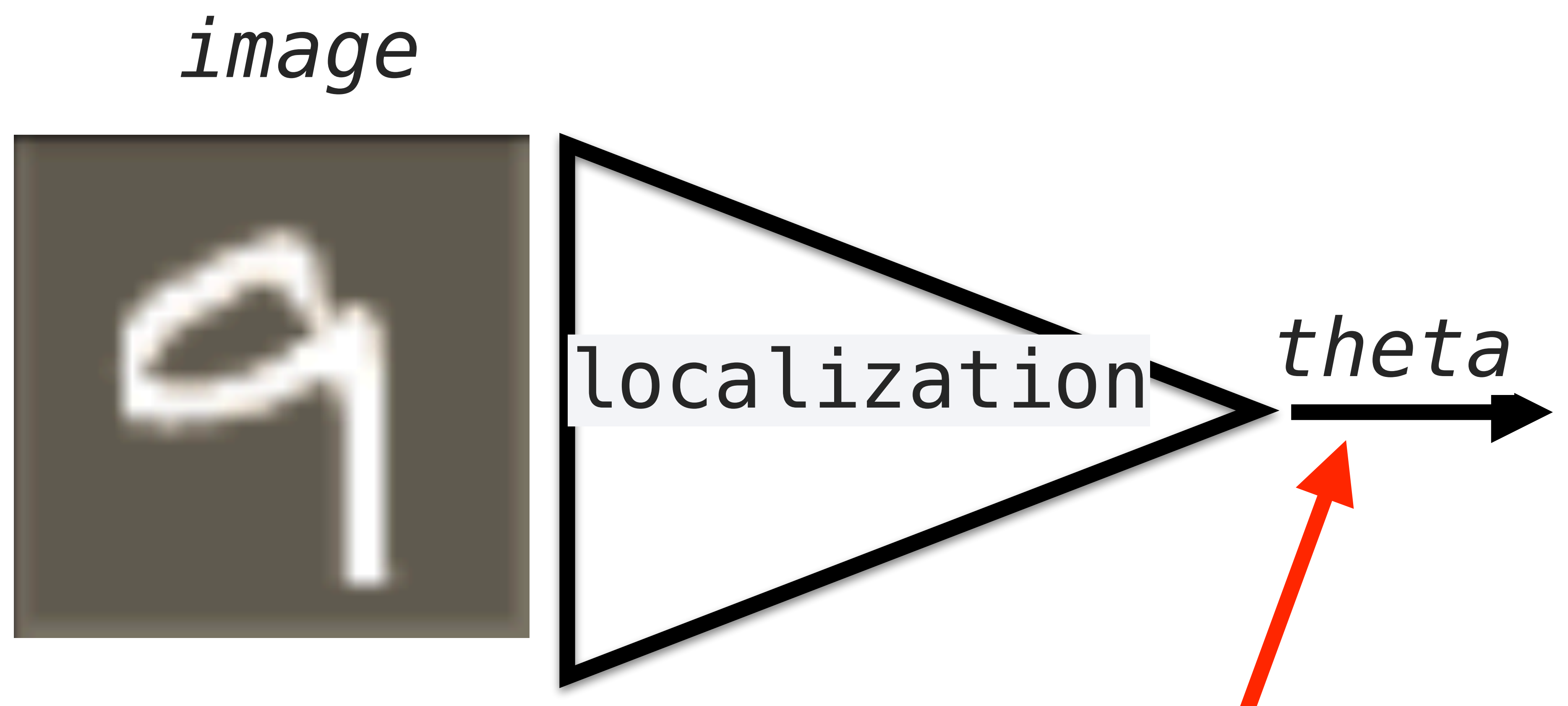
Mask RCNN - results

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

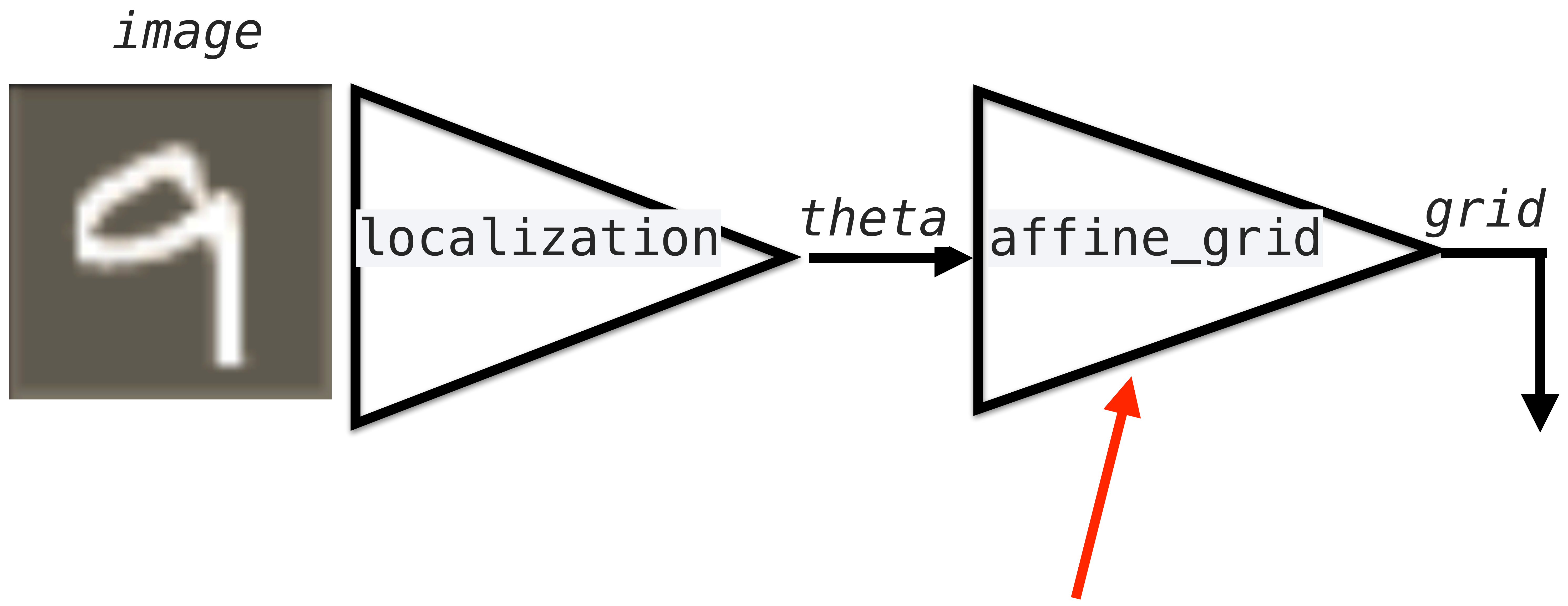
Spatial Transformer networks [Jaderberg 2016]

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



estimate parameters of 2D similarity transformation

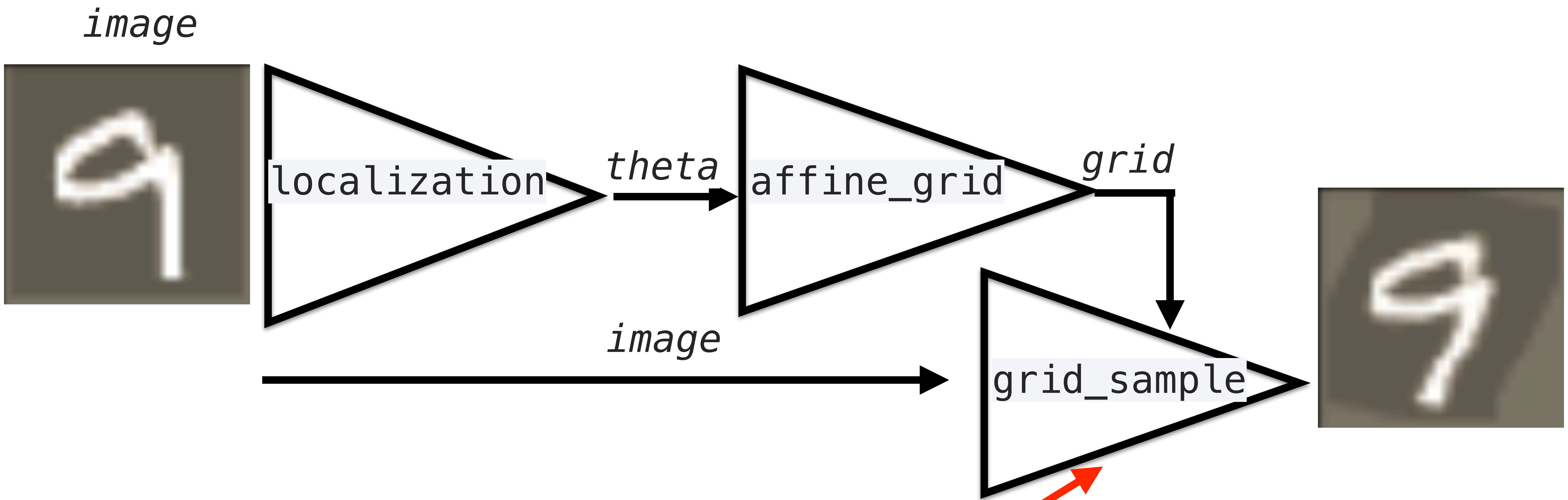
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



estimate pixel-wise correspondences of the 2D similarity transformation

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

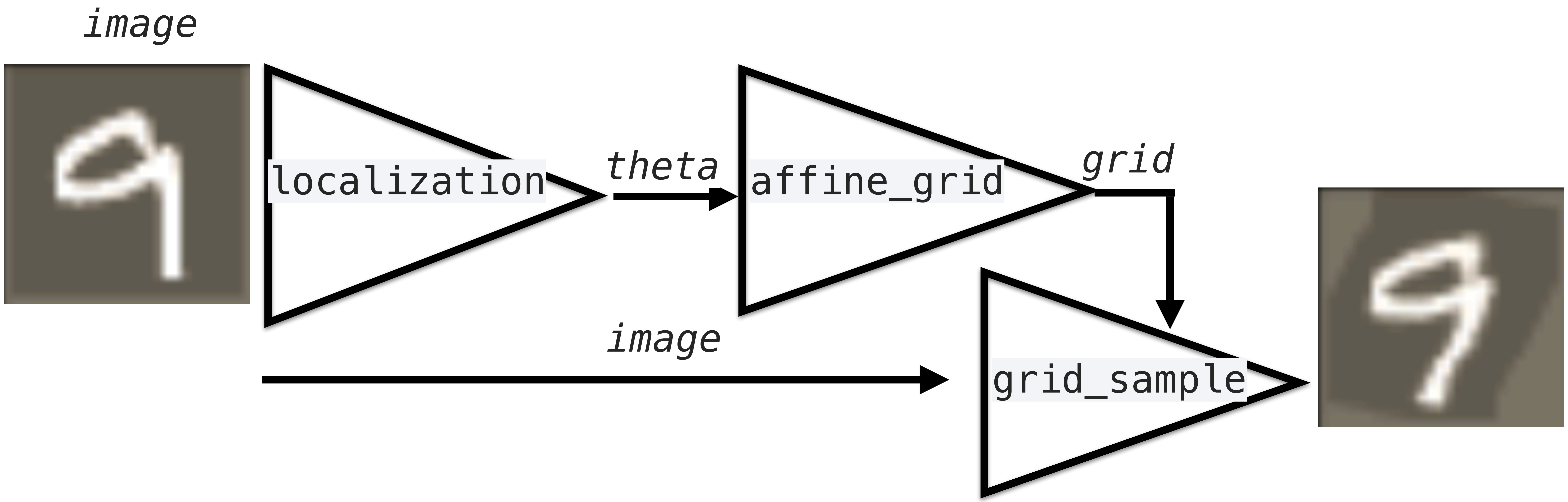
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf




```
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)
```

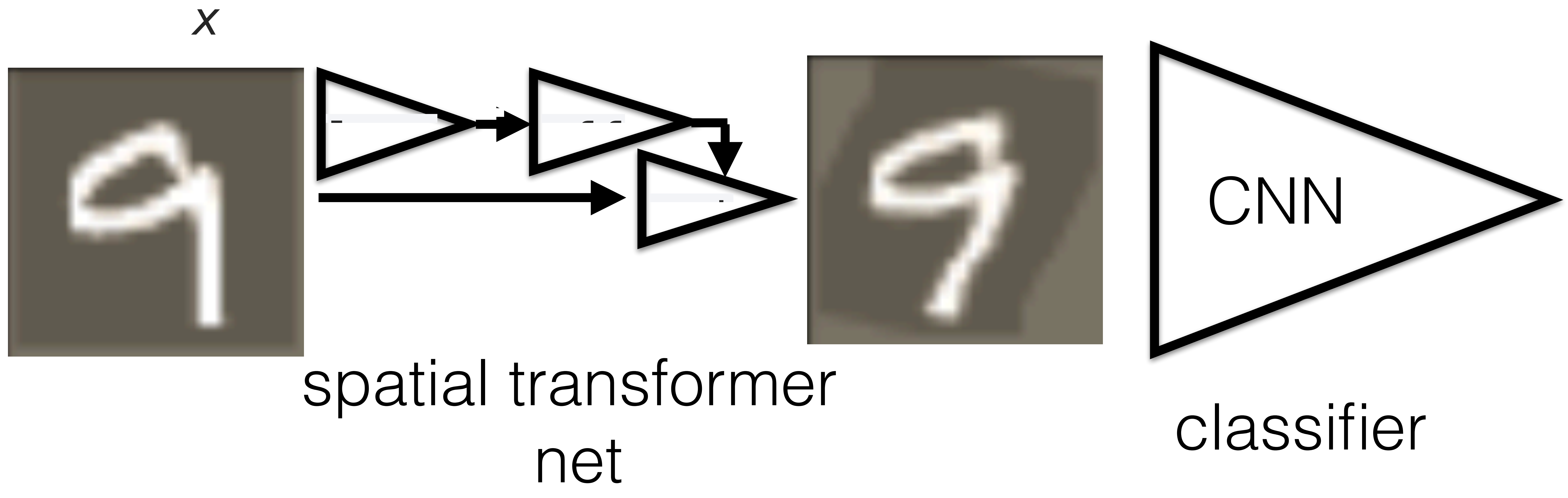
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



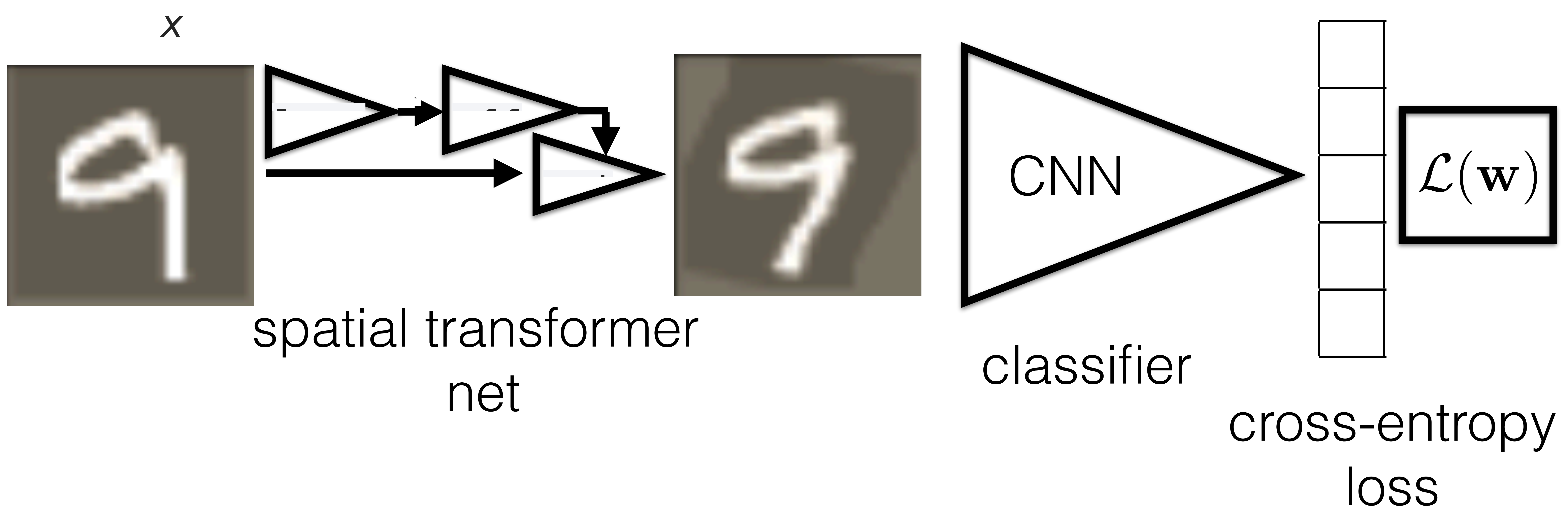
`torch.nn.functional.affine_grid(theta, size, align_corners=None)`

```
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



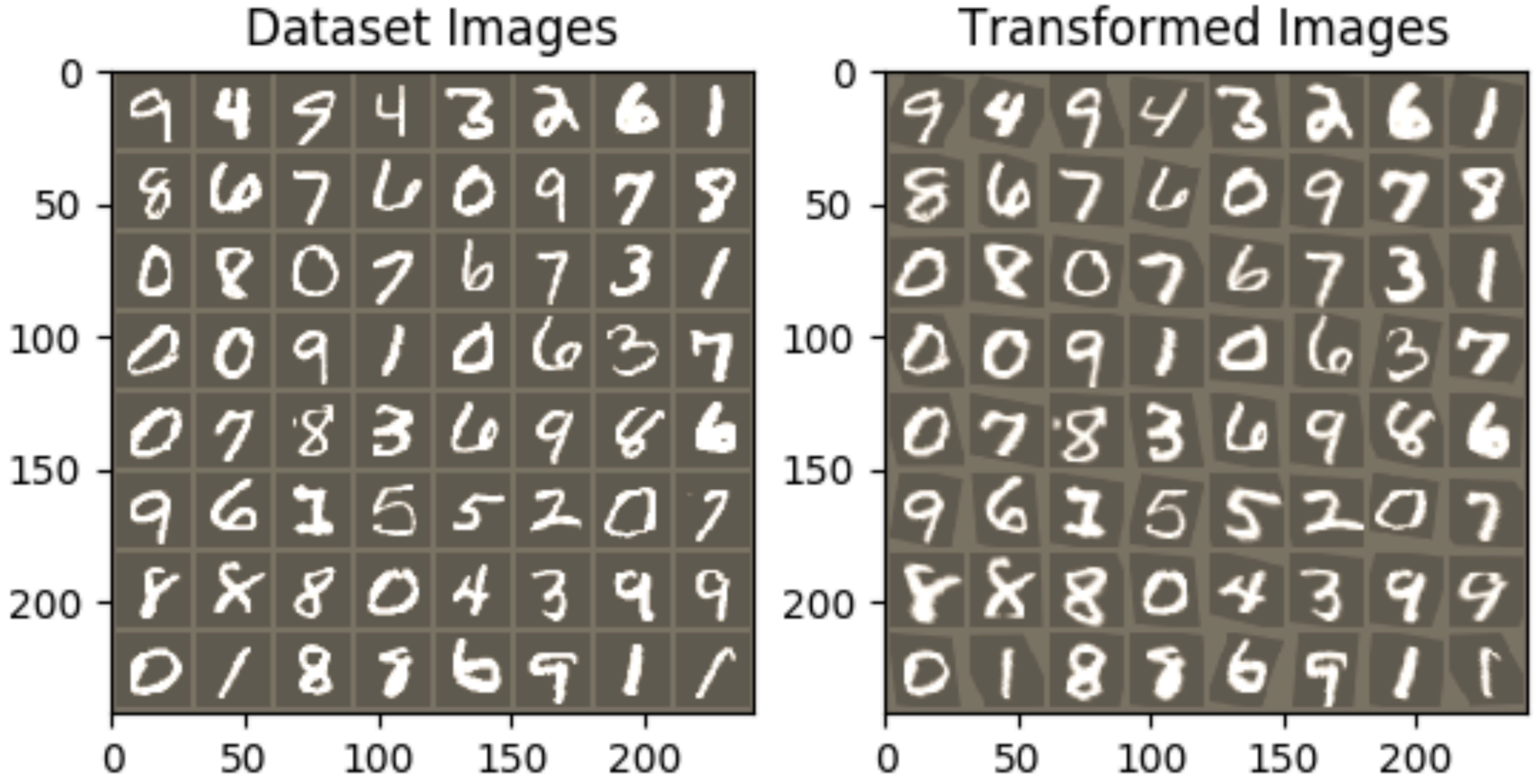
Spatial Transformer networks [Jaderberg 2016]
https://arxiv.org/pdf/1506.02025.pdf



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

Spatial Transformer networks

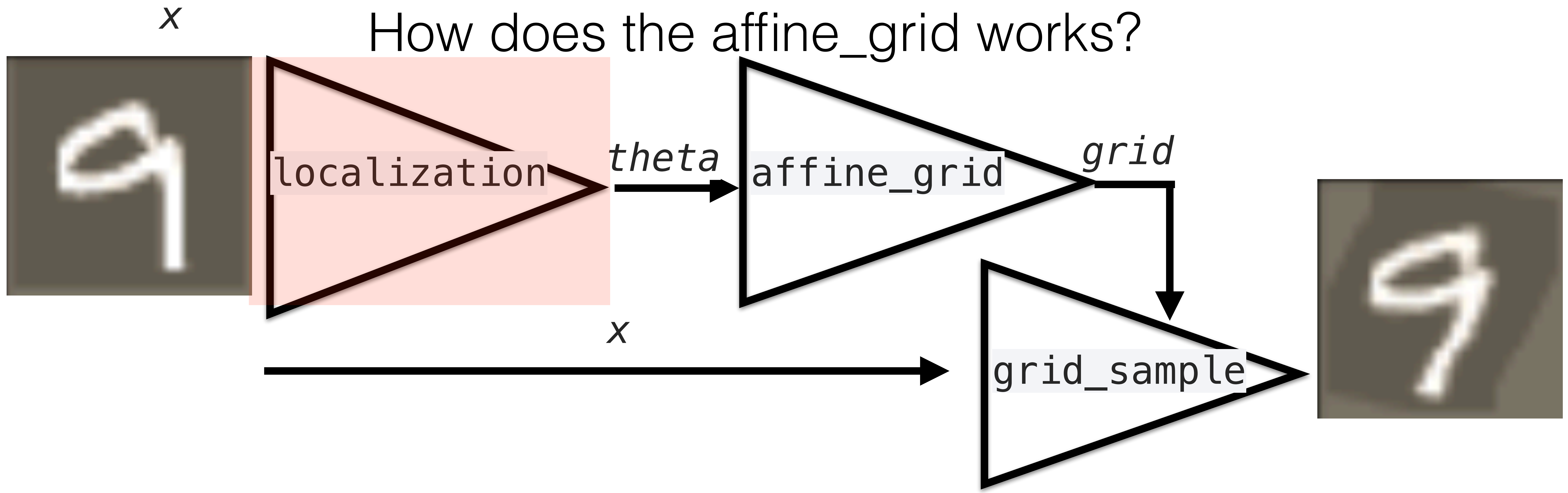
https://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html



Spatial Transformer networks [Jaderberg 2016]

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

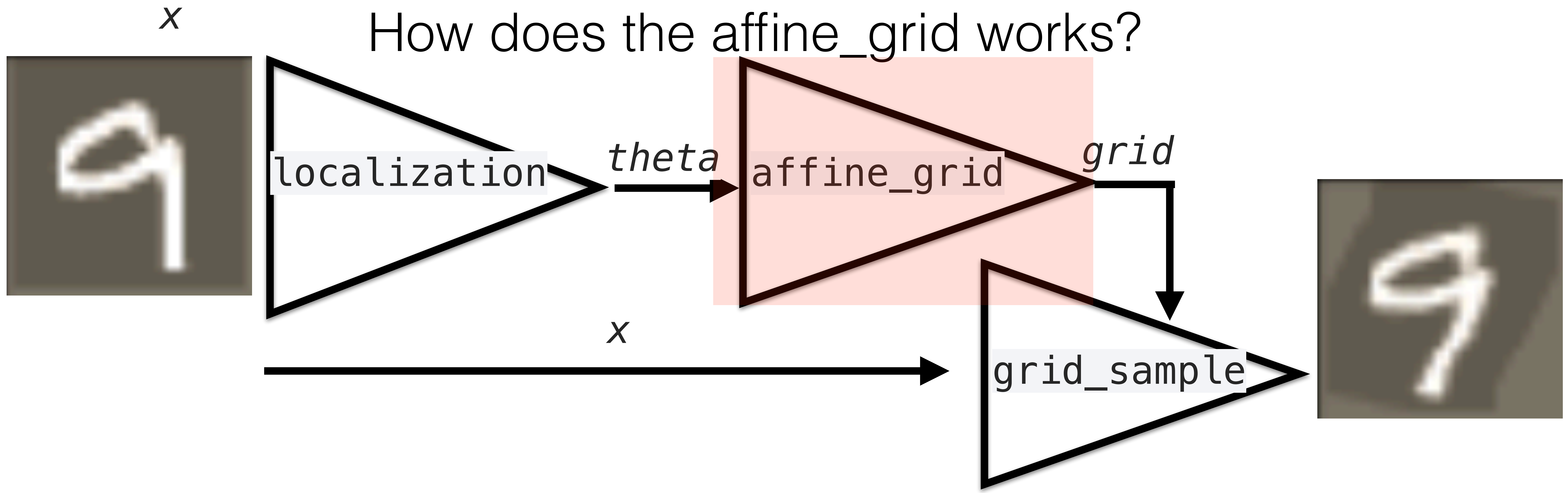
How does the affine_grid works?

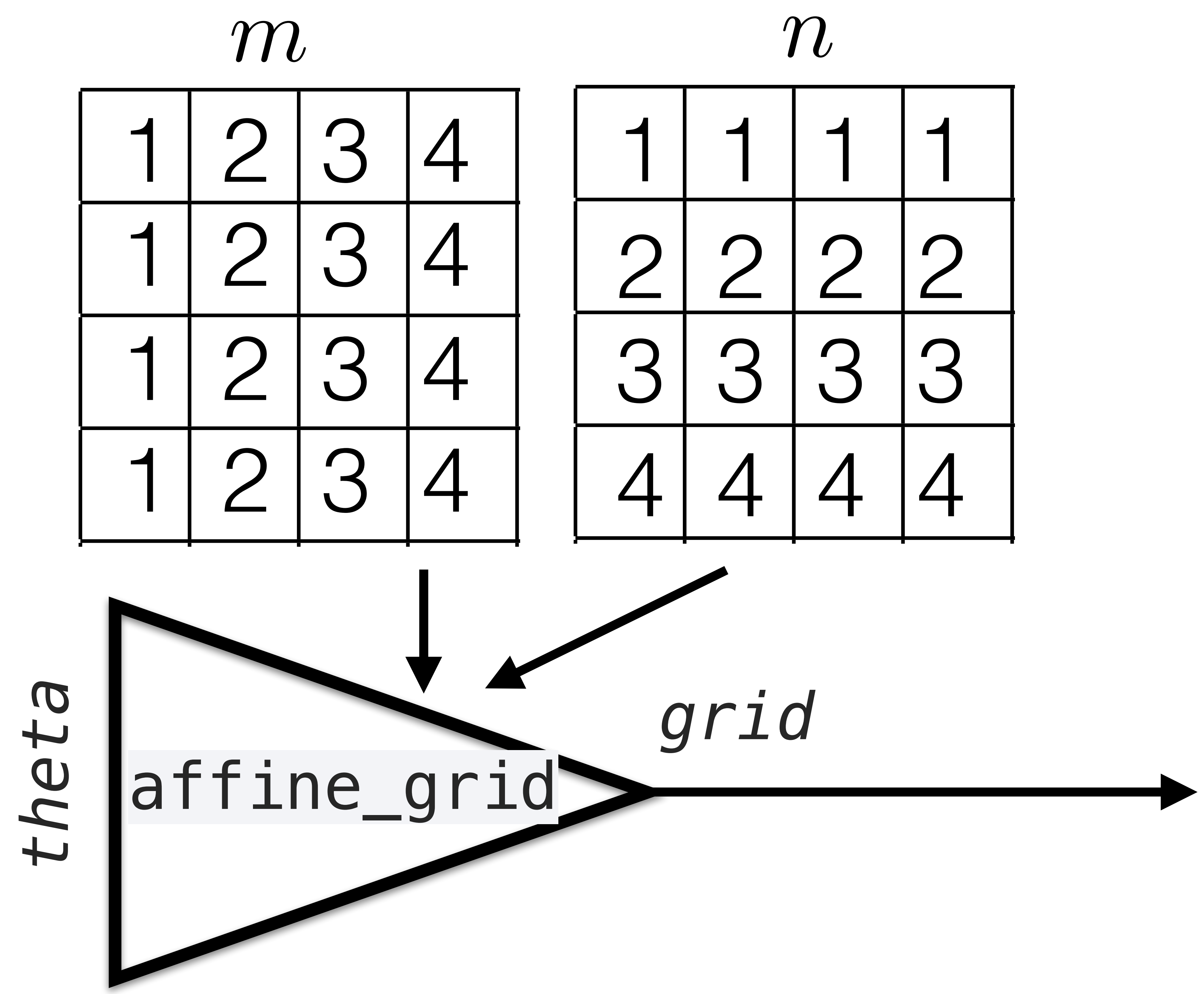


Spatial Transformer networks [Jaderberg 2016]

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

How does the affine_grid works?





θ_1

θ_2

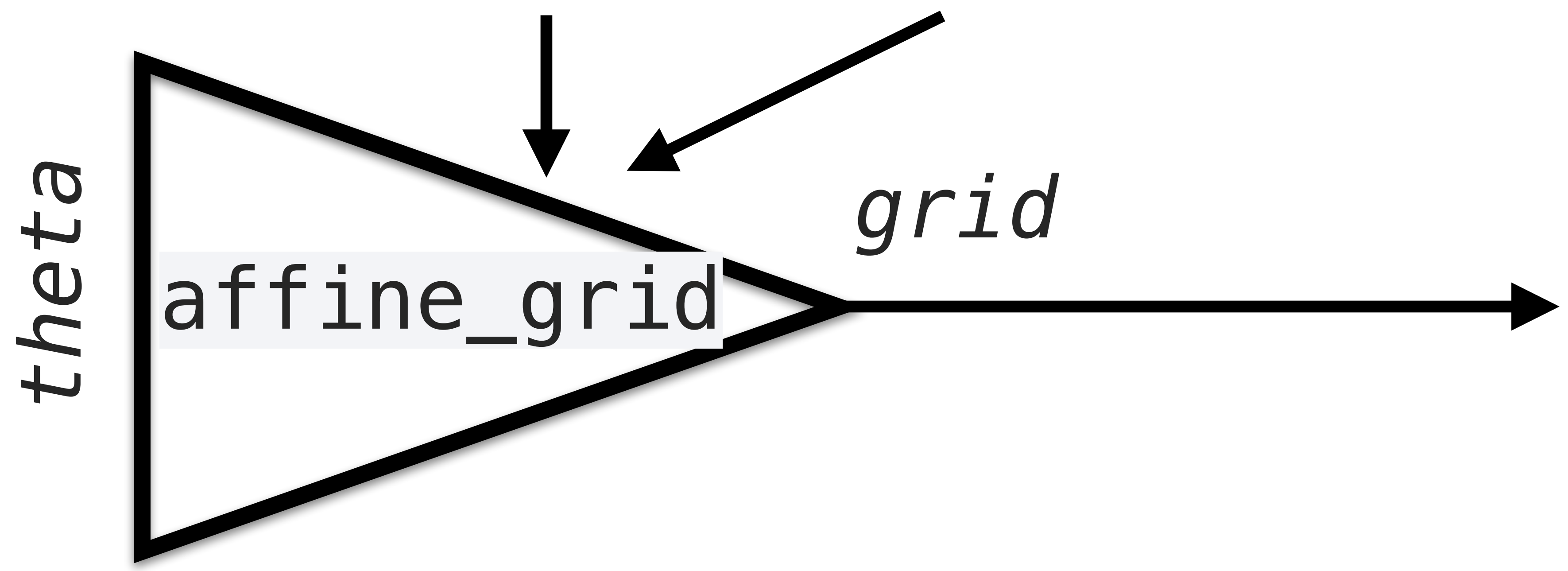
θ_3

m

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

n

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

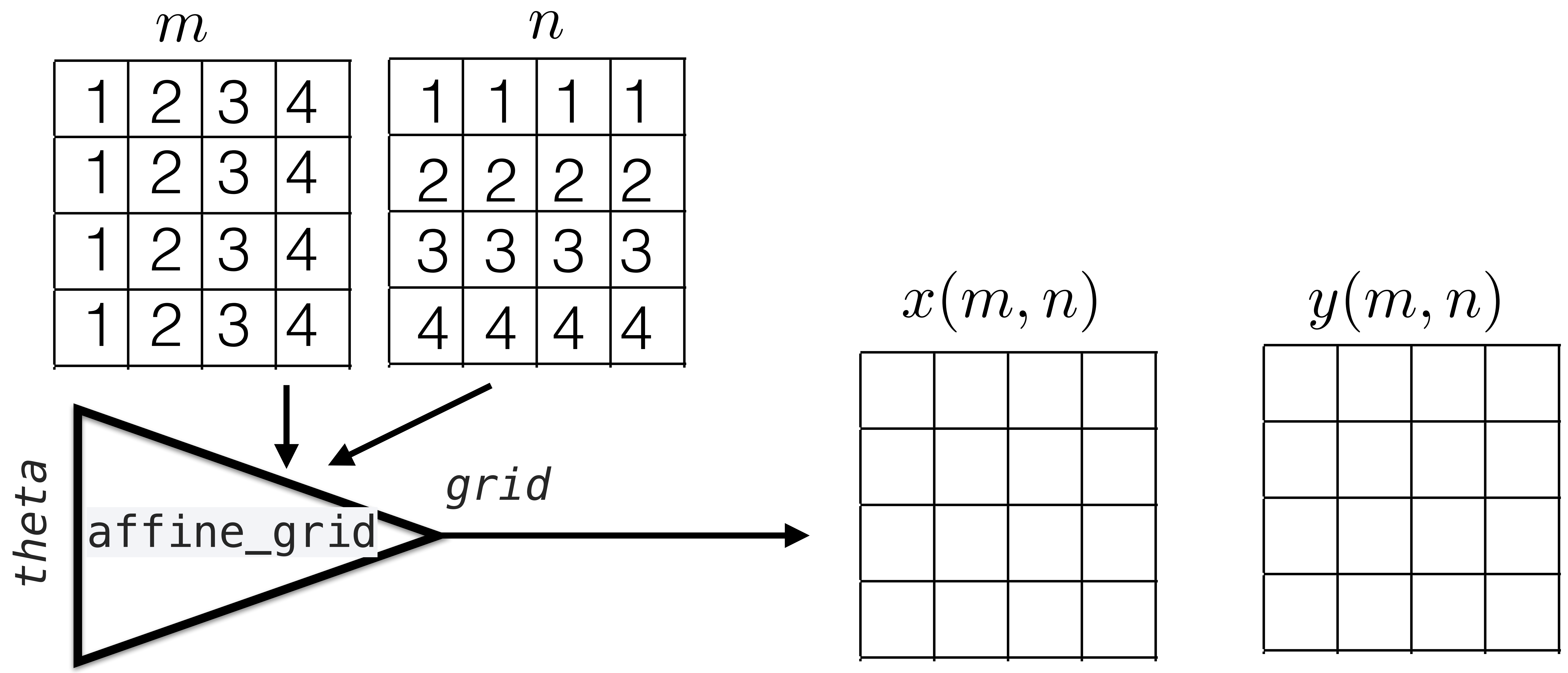


θ_1

θ_2

θ_3

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



θ_1

θ_2

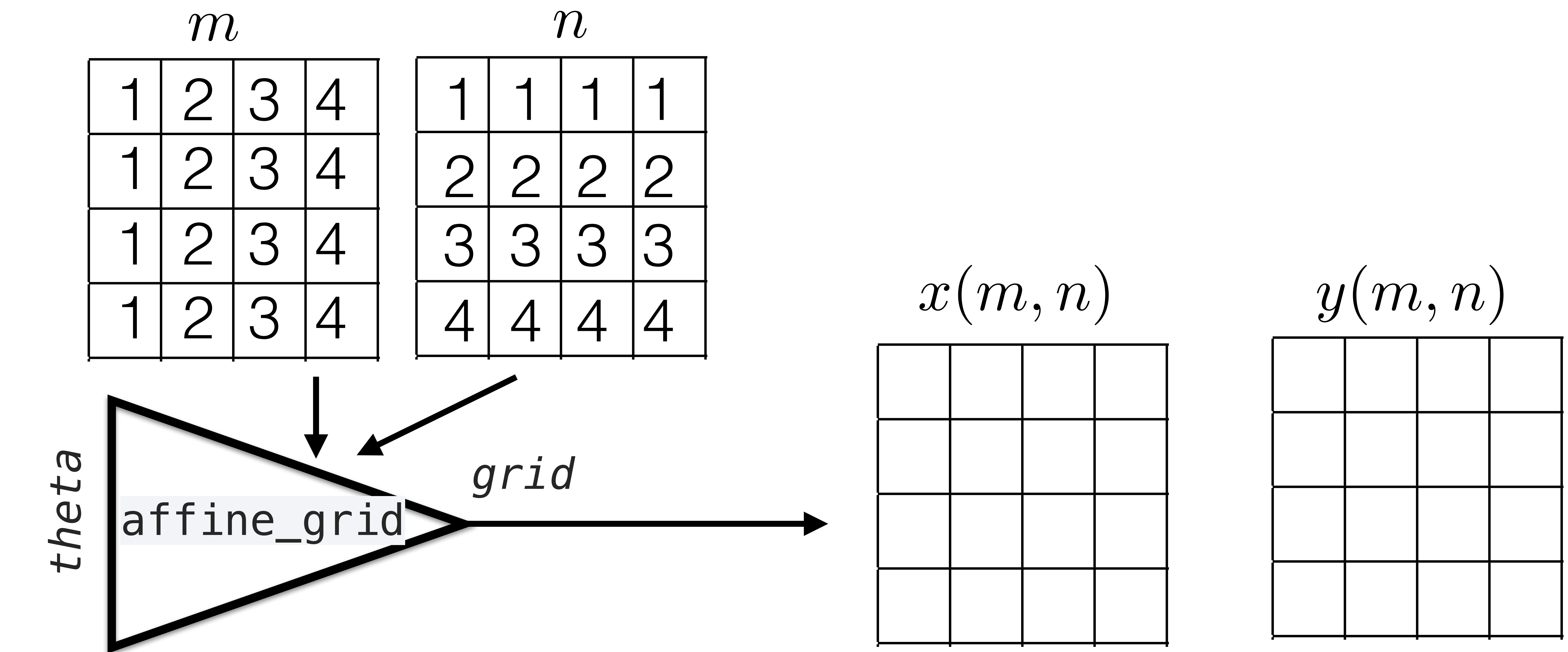
θ_3

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

Output arrays x(m,n) and y(m,n) say:

where should we read the output (m,n)-pixel from the original image

Can we translate image by 1 pixel up?



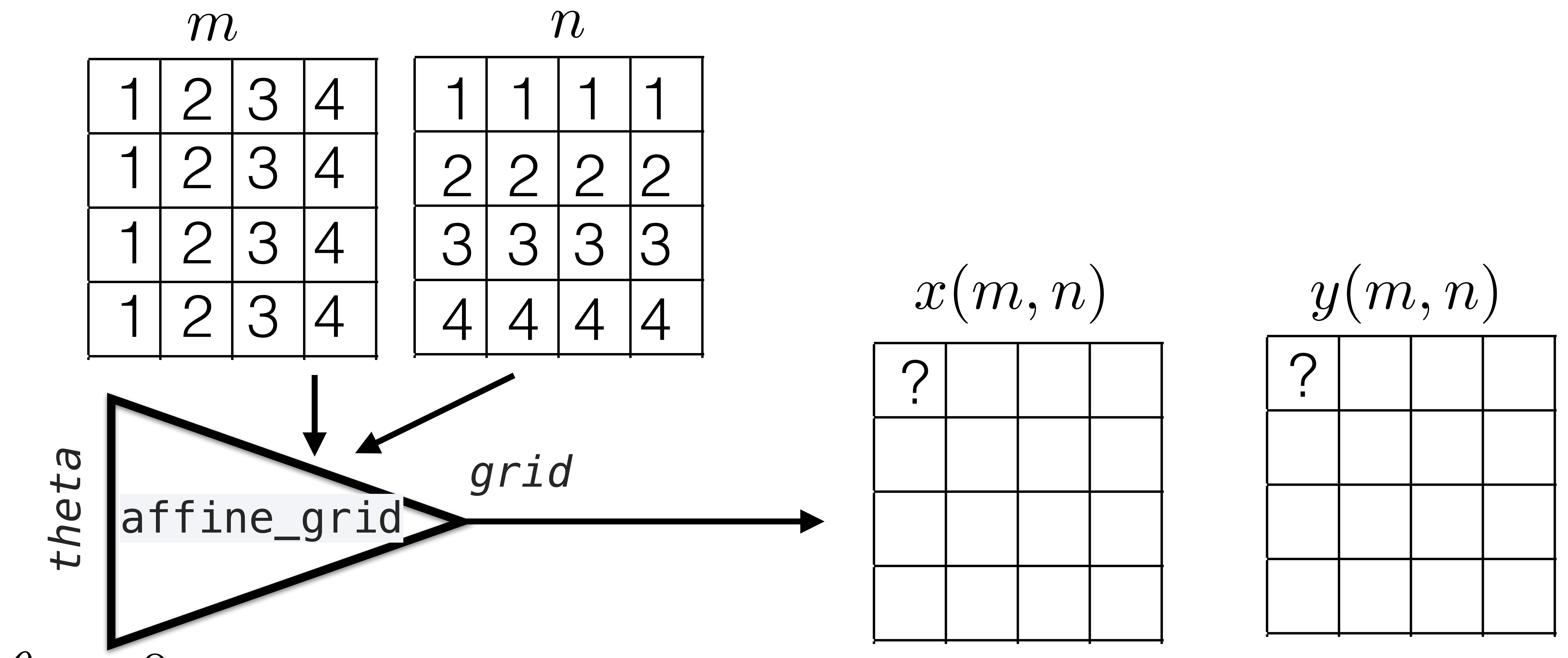
$$\theta_1 = 0$$
$$\theta_2 = 0$$
$$\theta_3 = 1$$

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

Output arrays x(m,n) and y(m,n) say:

where should we read the output (m,n)-pixel from the original image

Can we translate image by 1 pixel up?



$\theta_1 = 0$

$\theta_2 = 0$

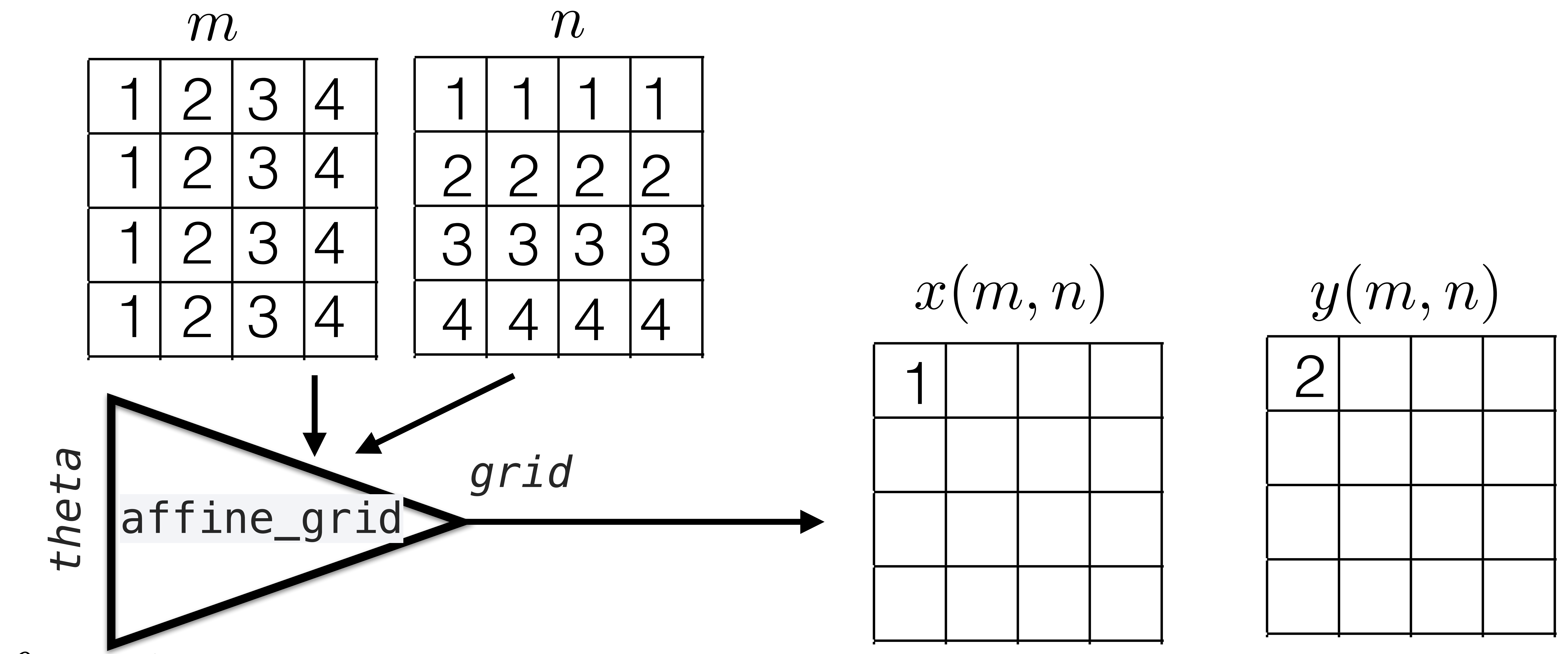
$\theta_3 = 1$

$$\begin{bmatrix} x(m,n) \\ y(m,n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} m \\ n \\ 1 \end{bmatrix}$$

Output arrays x(m,n) and y(m,n) say:

where should we read the output (m,n)-pixel from the original image

Can we translate image by 1 pixel up?



$\theta_1 = 0$

$\theta_2 = 0$

$\theta_3 = 1$

$$\begin{bmatrix} x(m,n) \\ y(m,n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} m \\ n \\ 1 \end{bmatrix}$$

Output arrays x(m,n) and y(m,n) say:

where should we read the output (m,n)-pixel from the original image

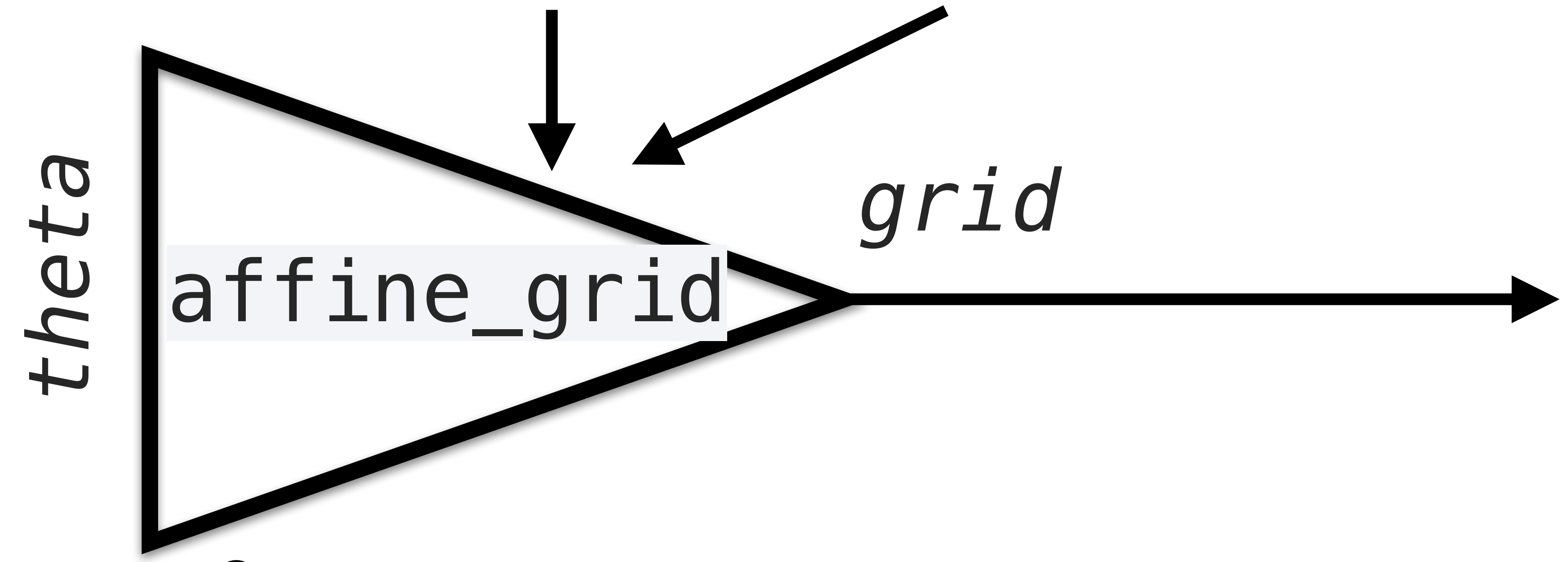
Can we translate image by 1 pixel up?

m

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

n

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



$x(m,n)$

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

$y(m,n)$

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

$\theta_1 = 0$

$\theta_2 = 0$

$\theta_3 = 1$

$$\begin{bmatrix} x(m,n) \\ y(m,n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} m \\ n \\ 1 \end{bmatrix}$$

Output arrays x(m,n) and y(m,n) say:

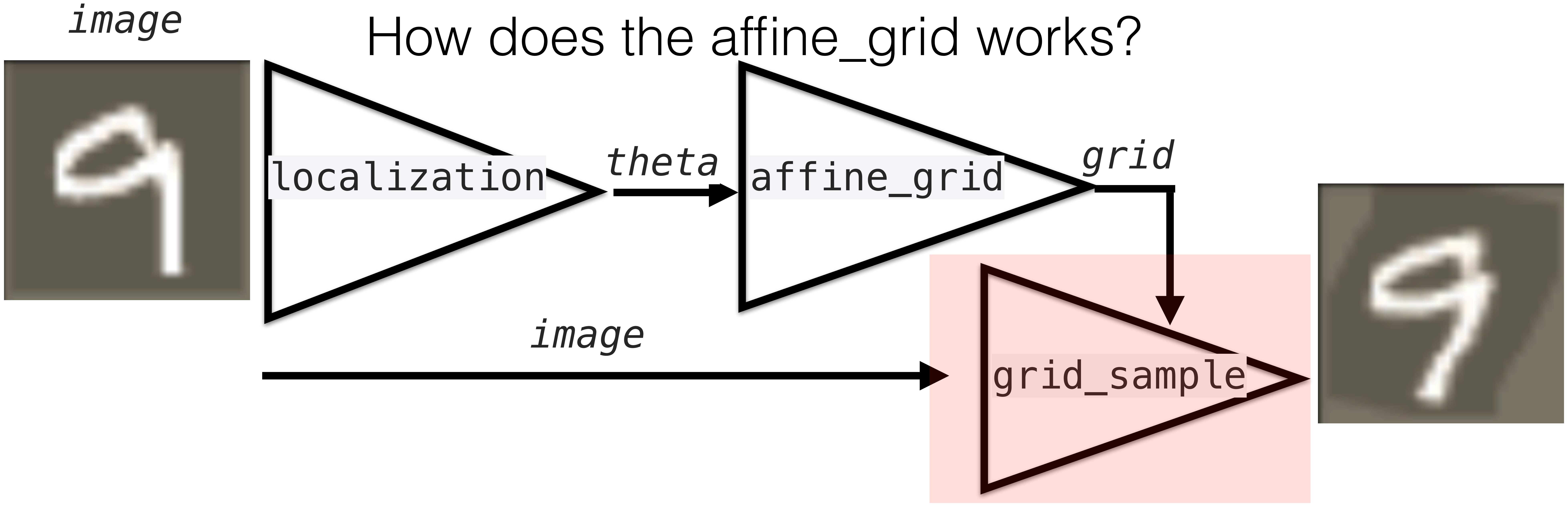
where should we read the output (m,n)-pixel from the original image

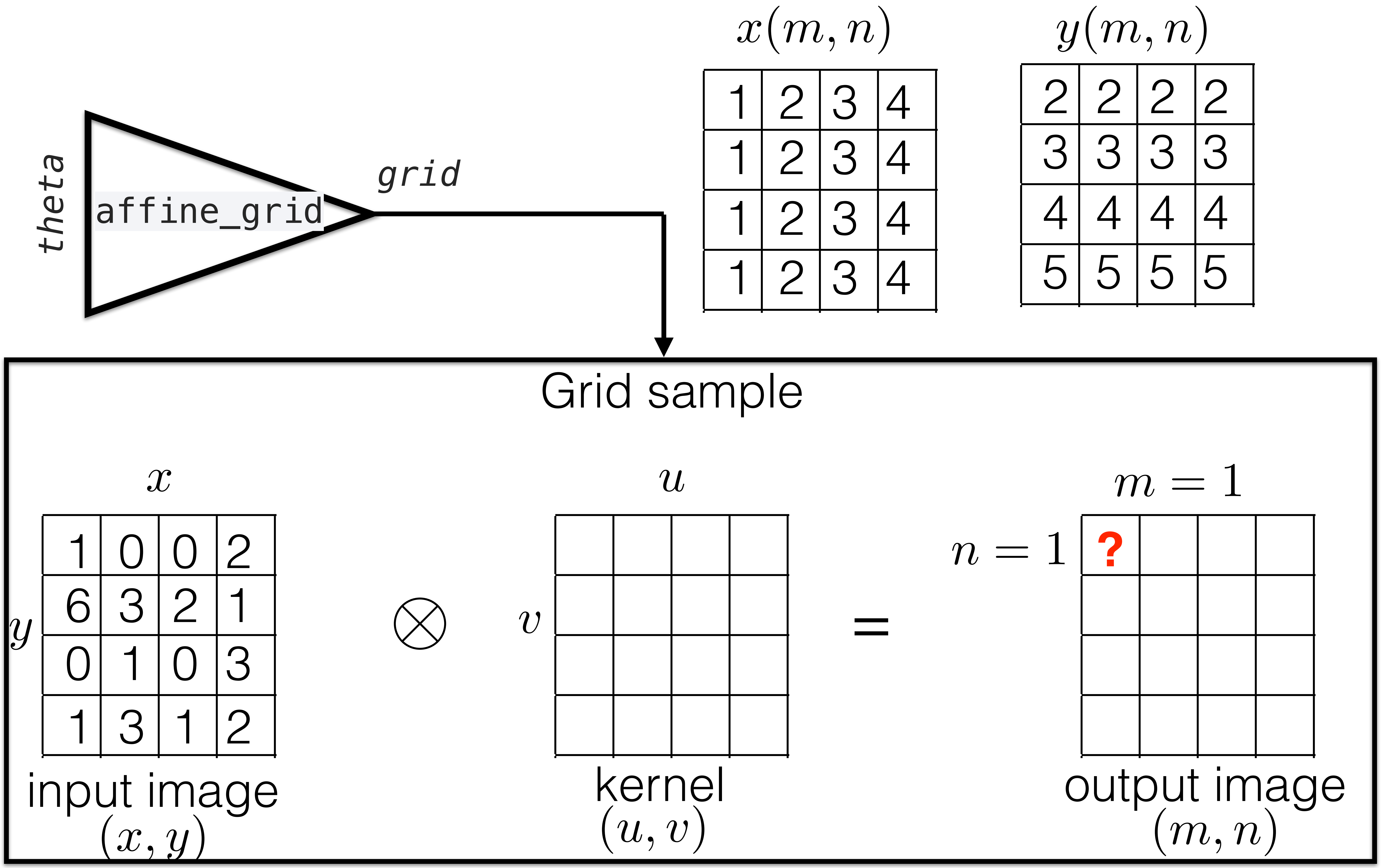
Can we translate image by 1 pixel up?

Spatial Transformer networks [Jaderberg 2016]

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

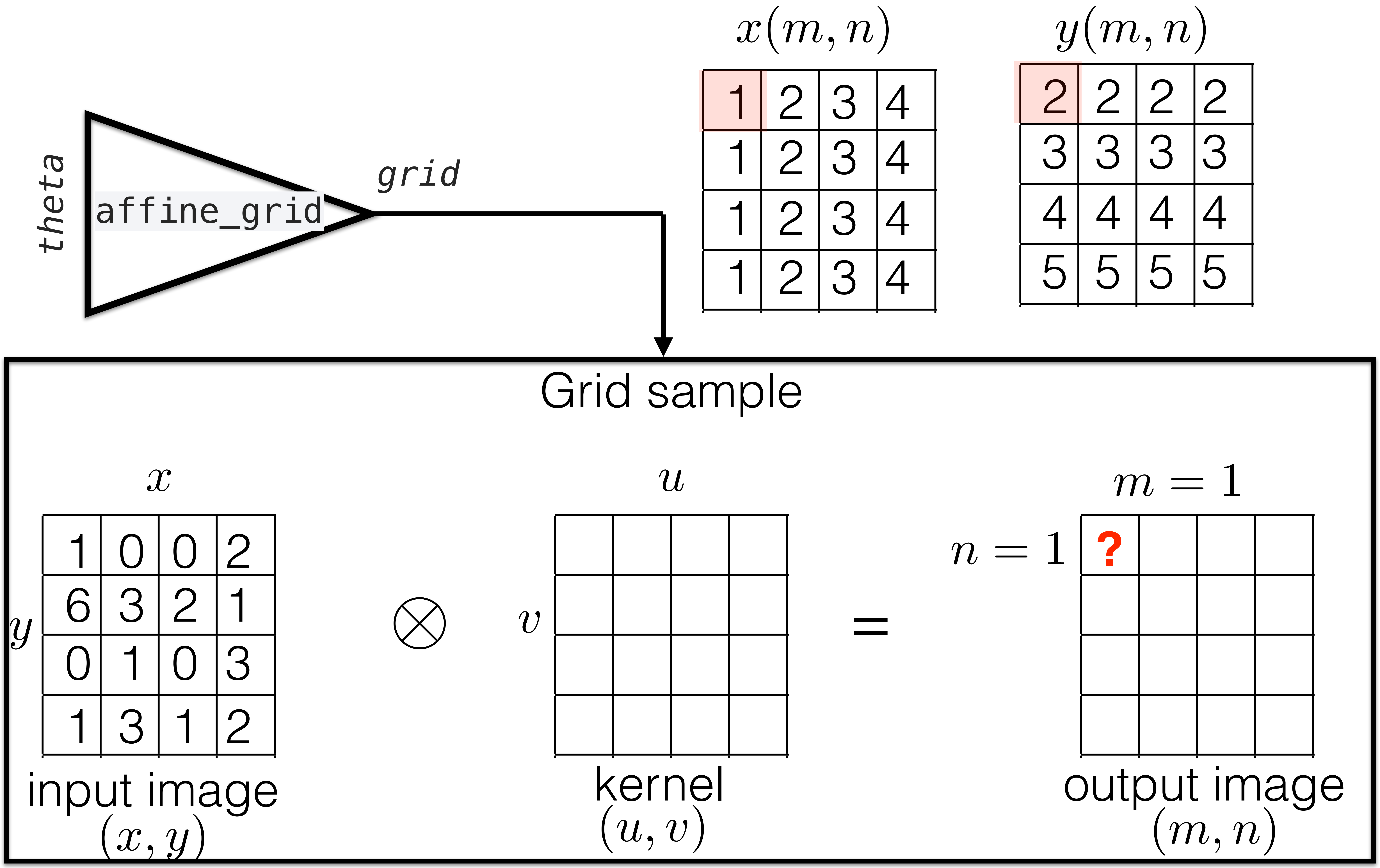
How does the affine_grid works?

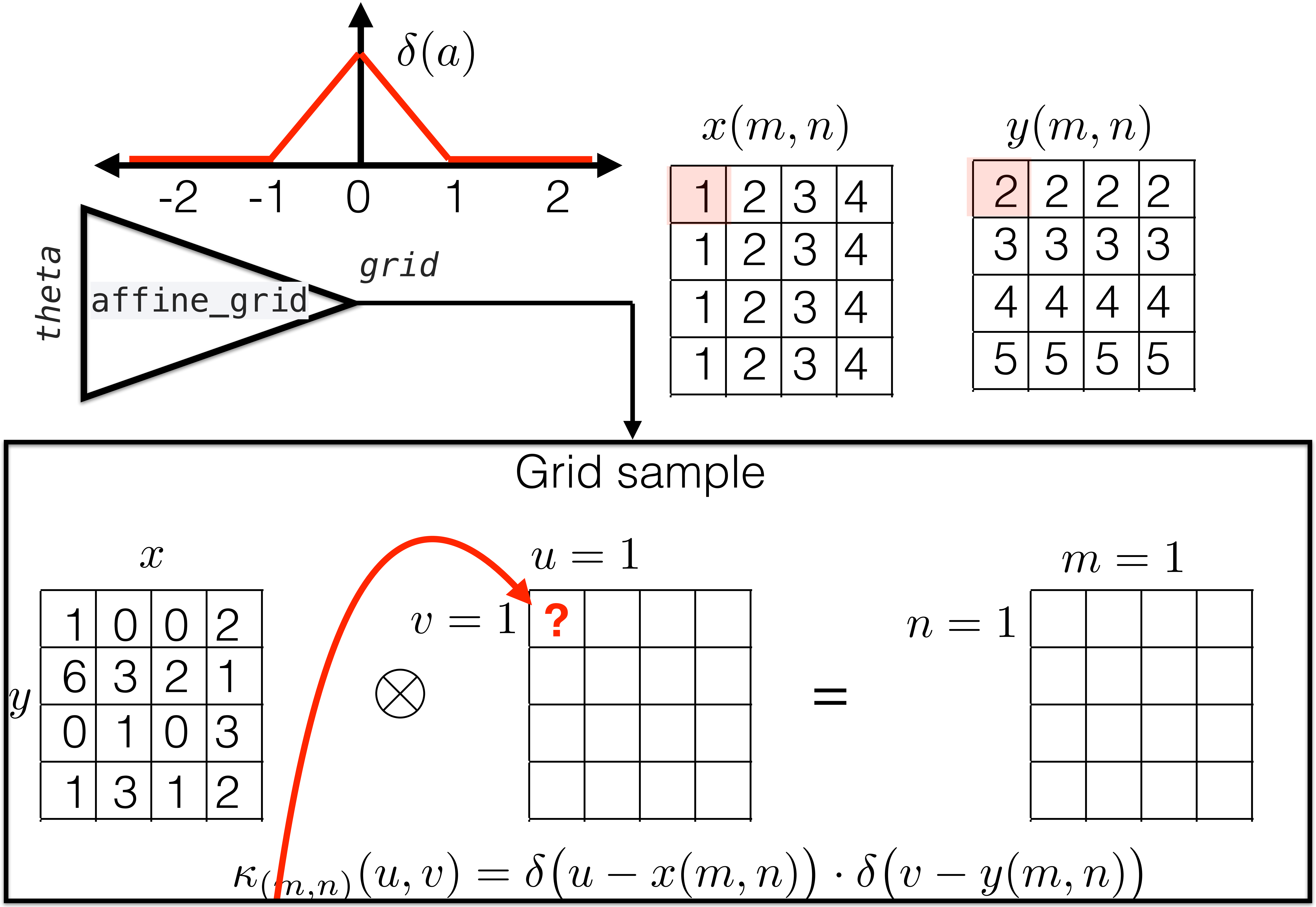




theta
affine_grid
grid
$x(m,n)$
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
$y(m,n)$
2 2 2 2
3 3 3 3
4 4 4 4
5 5 5 5
Grid sample
x
y
1 0 0 2
6 3 2 1
0 1 0 3
1 3 1 2
input image
(x,y)
$\otimes$
u
v
kernel
(u,v)
=
$m=1$
$n=1$
?
output image
(m,n)

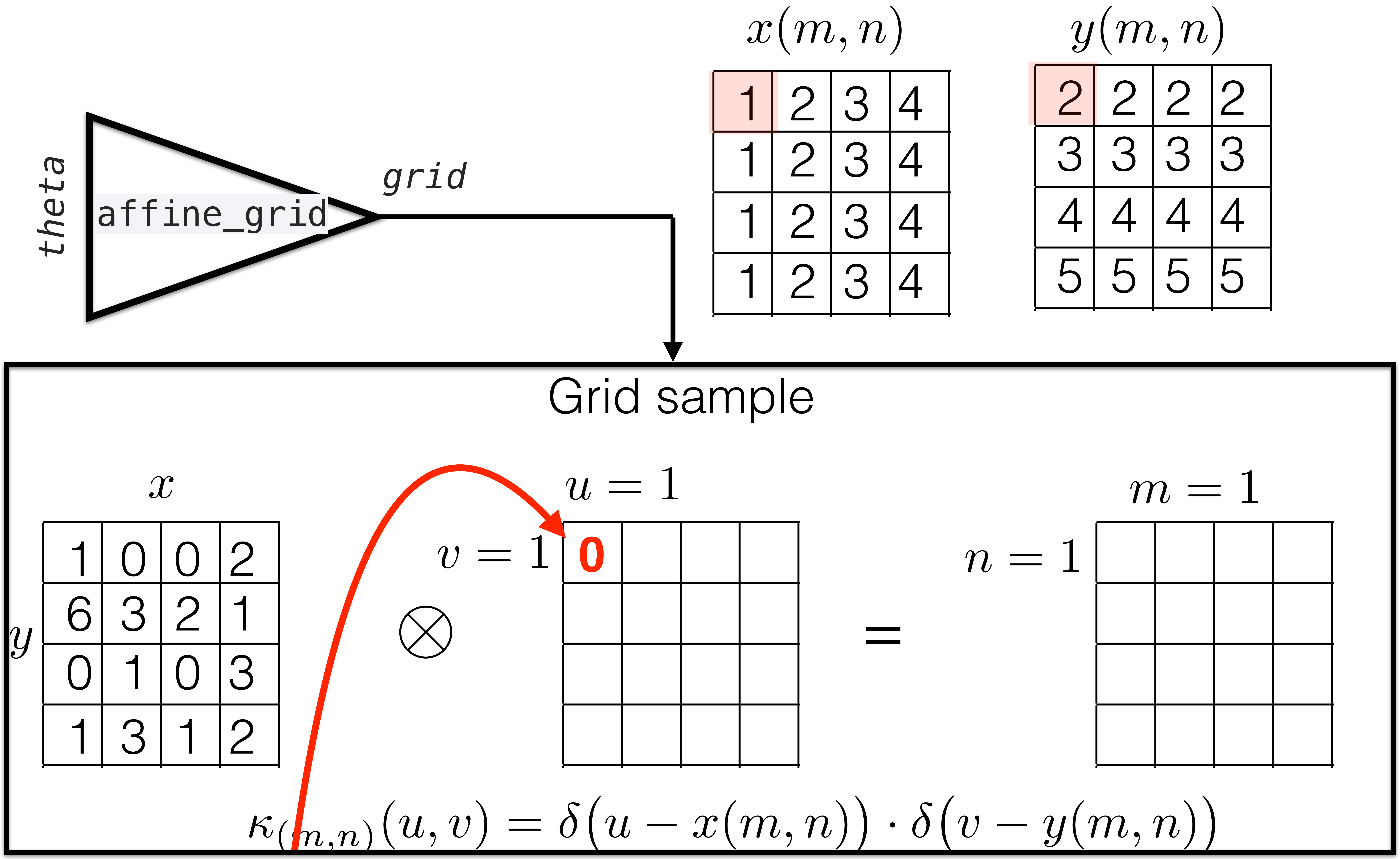
Can we translate image by 1 pixel up?



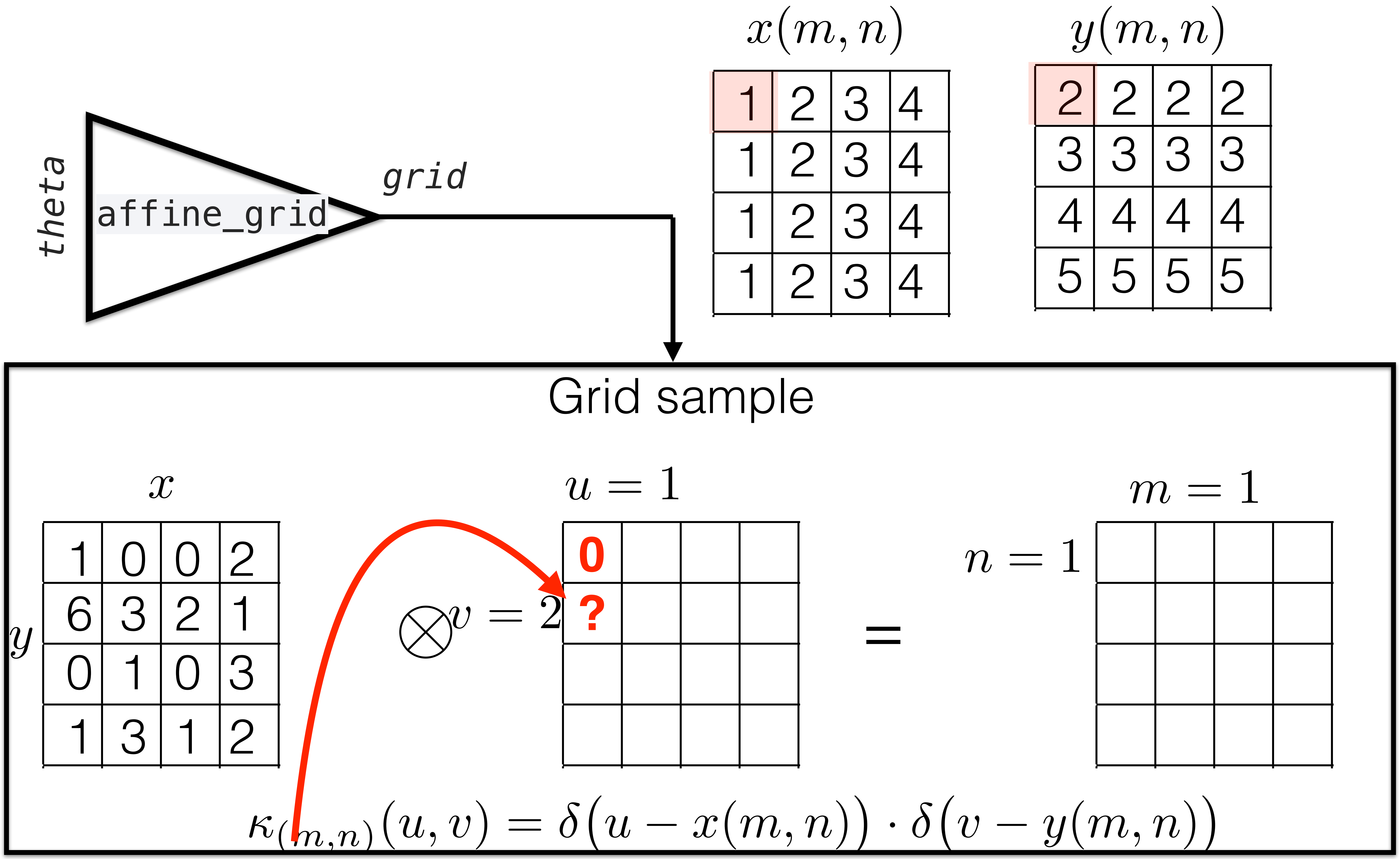


$\delta(a)$
-2 -1 0 1 2
theta
affine_grid
grid
$x(m,n)$
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
$y(m,n)$
2 2 2 2
3 3 3 3
4 4 4 4
5 5 5 5
Grid sample
x
y
1 0 0 2
6 3 2 1
0 1 0 3
1 3 1 2
$u = 1$
$v = 1$
?
$\otimes$
=
$m = 1$
$n = 1$
$\kappa_{(m,n)}(u,v) = \delta\big(u - x(m,n)\big) \cdot \delta\big(v - y(m,n)\big)$

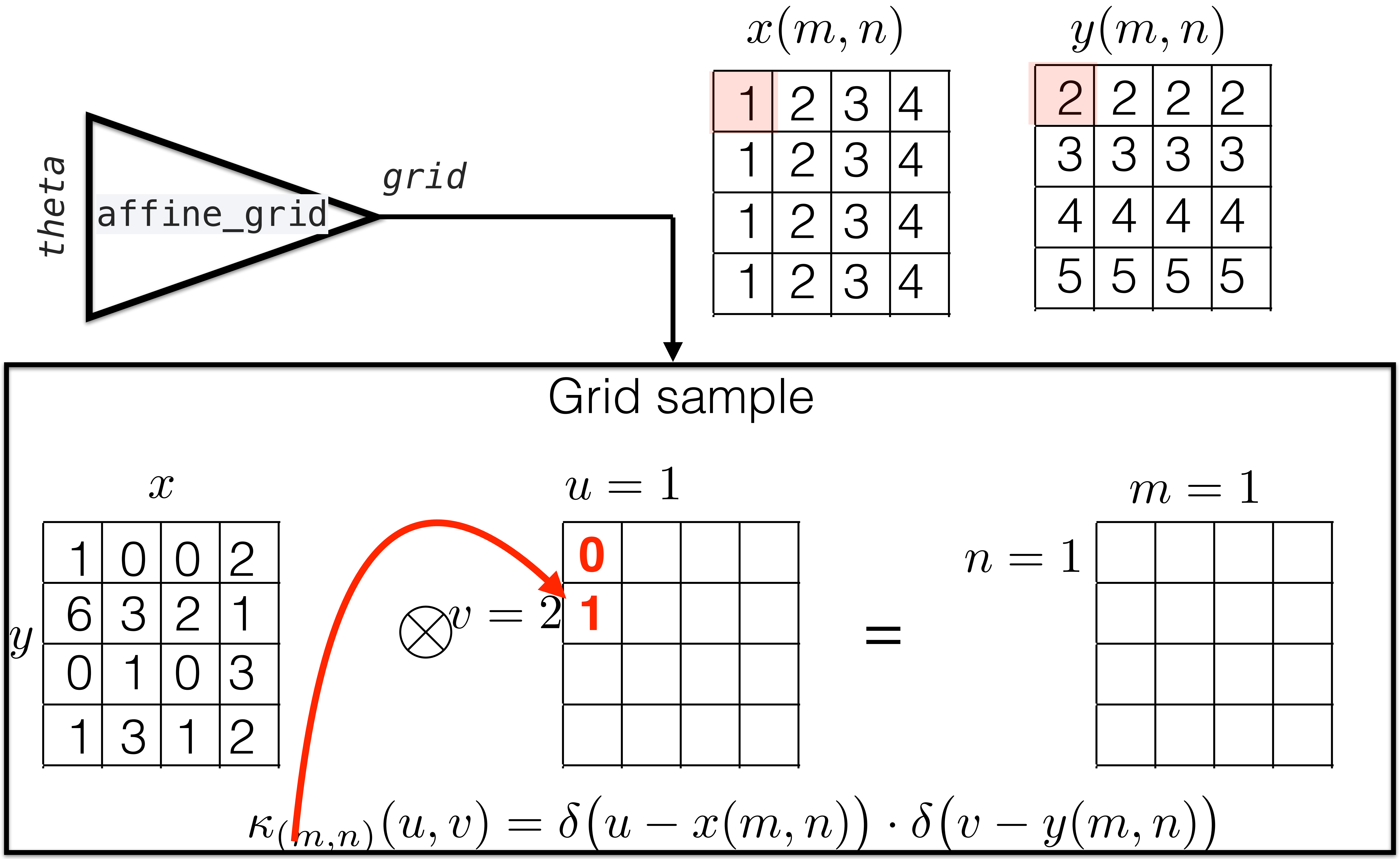
Can we translate image by 1 pixel up?



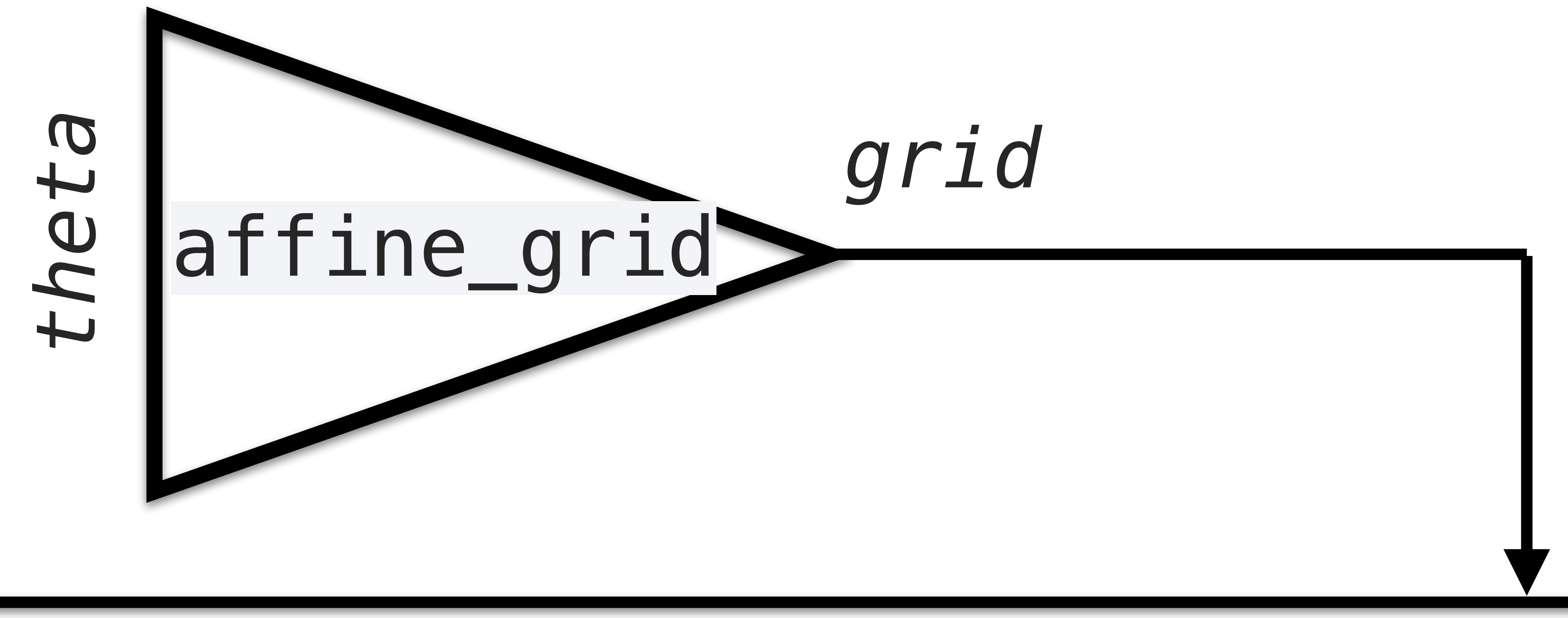
Can we translate image by 1 pixel up?



Can we translate image by 1 pixel up?



Can we translate image 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

Grid sample

x (columns), y (rows)

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

$\otimes$

u (columns), v (rows)

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

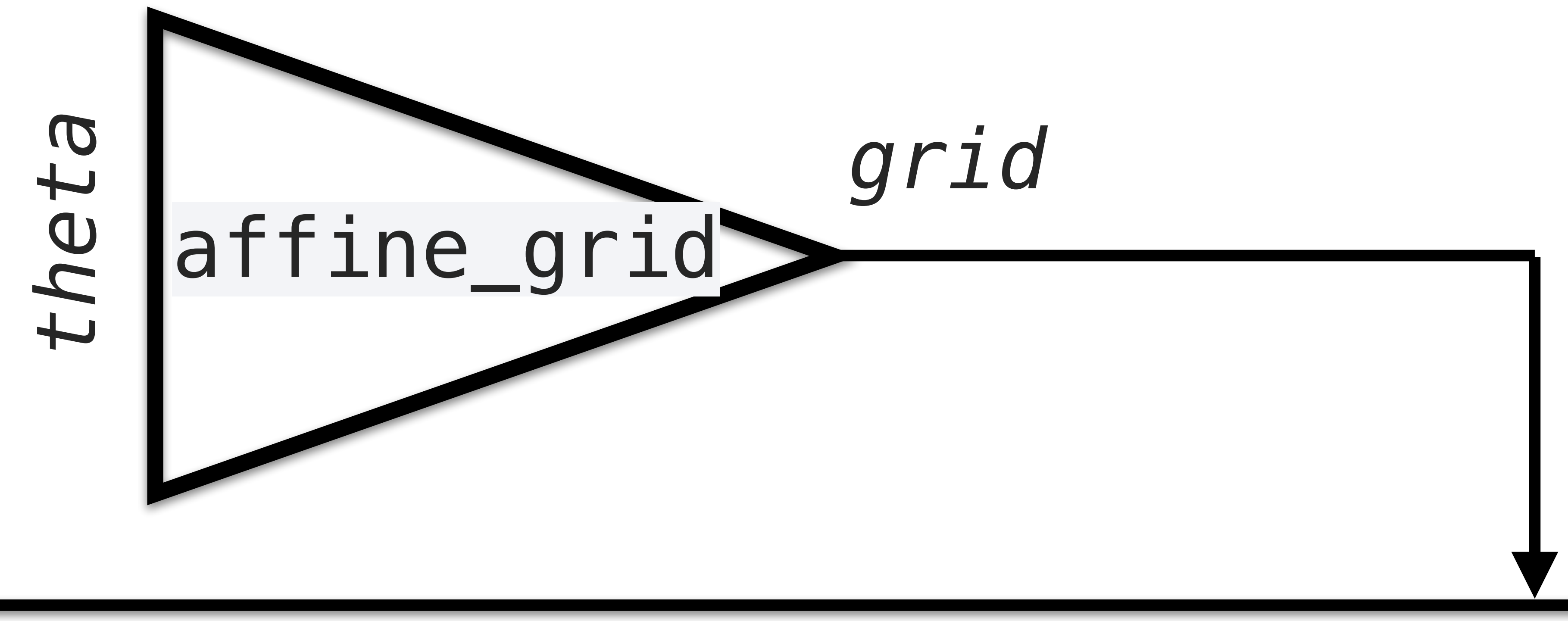
$=$

$m = 1$, $n = 1$

?			

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

Can we translate image by 1 pixel up?



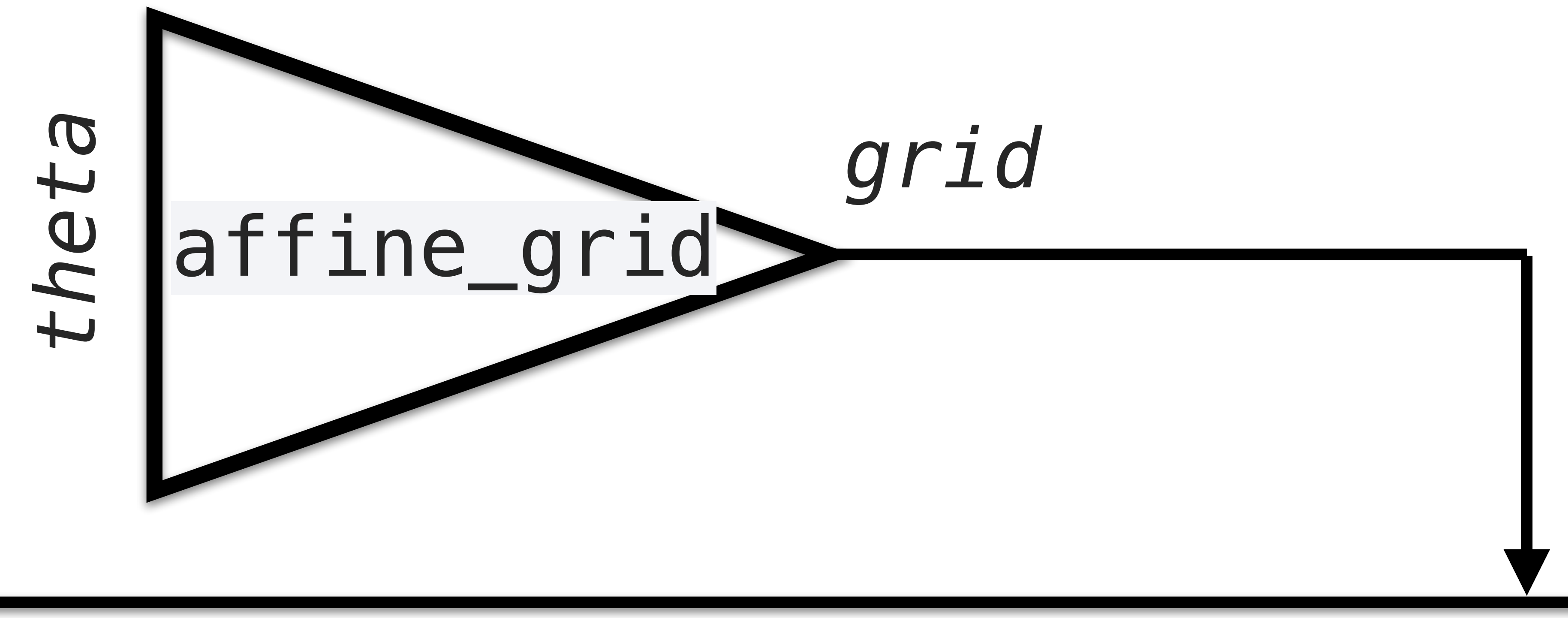
$$x(m,n) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix} \qquad y(m,n) = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 \end{bmatrix}$$

Grid sample

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

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

Can we translate image 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

Grid sample

x (columns), y (rows)

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

$\otimes$

u (columns), v (rows)

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

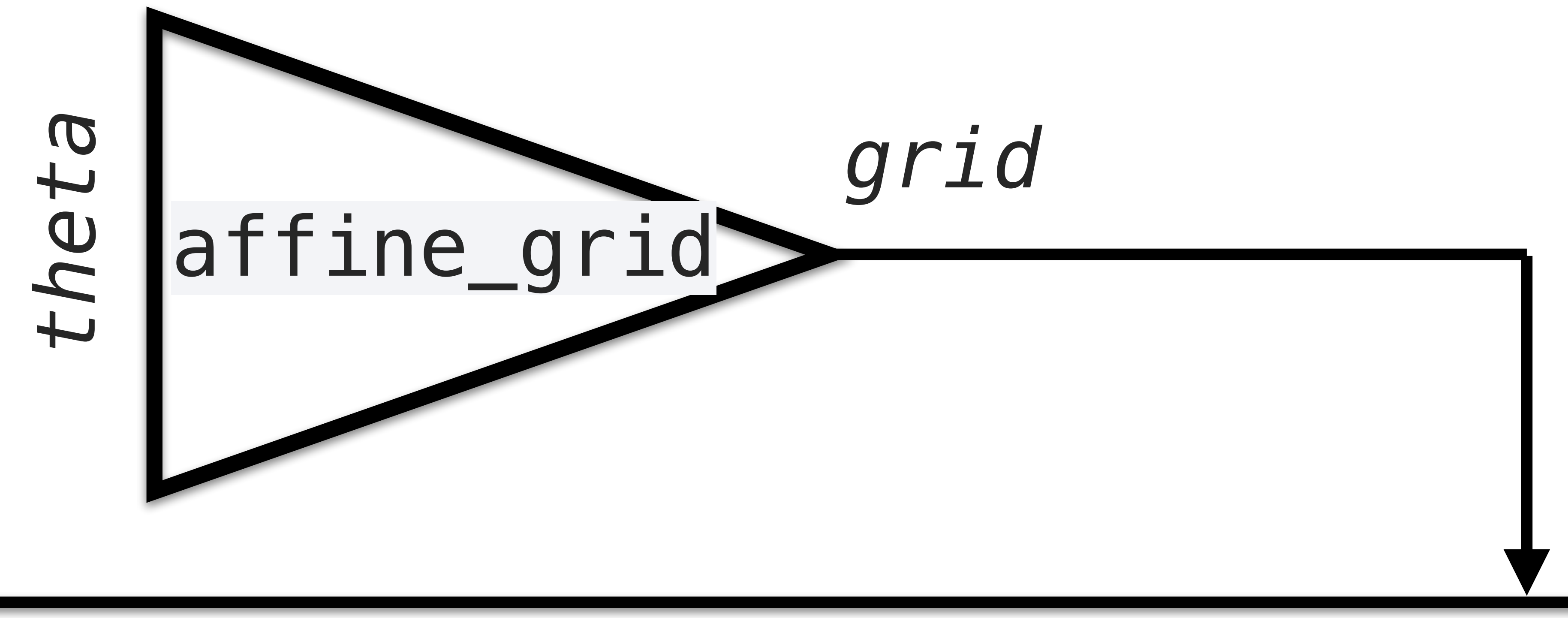
=

$m = 2$, $n = 1$

6	$\boxed{3}$		

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

Can we translate image by 1 pixel up?



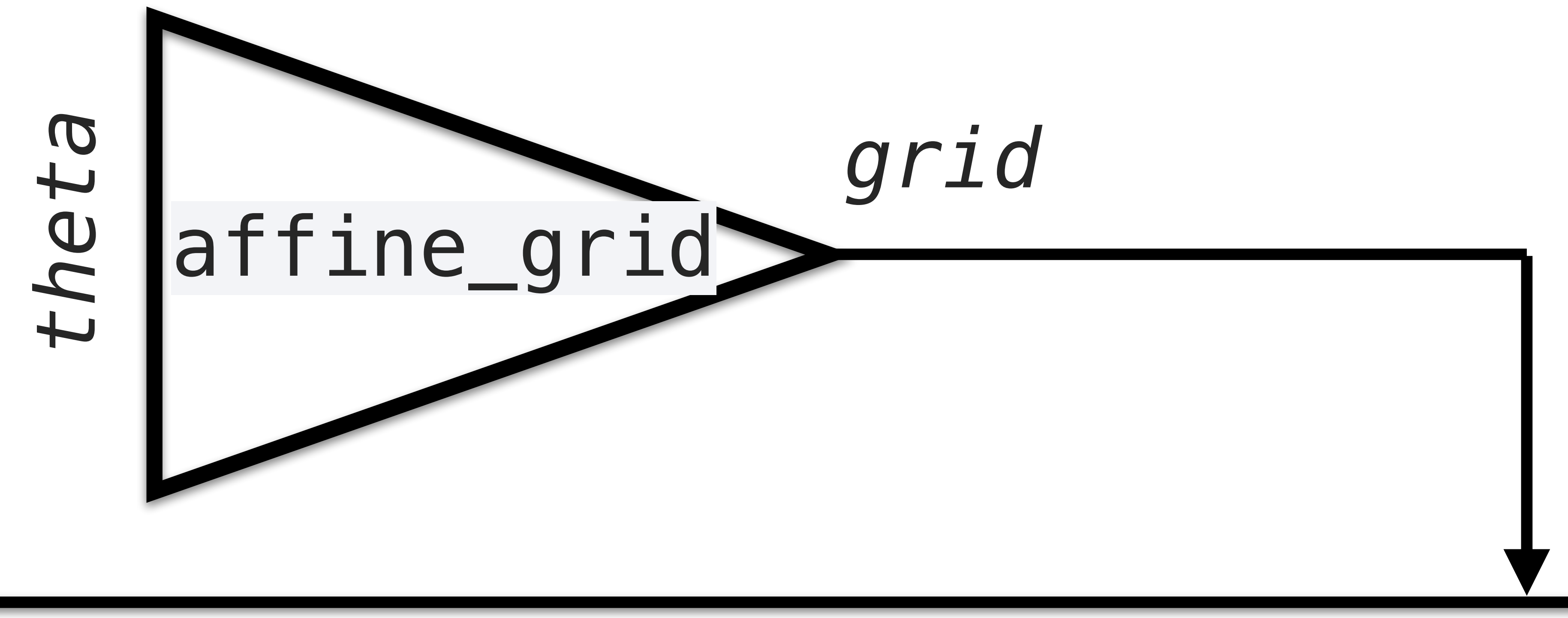
$$x(m,n) = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & 4 \\ \hline\end{array} \qquad y(m,n) = \begin{array}{|c|c|c|c|}\hline 2 & 2 & 2 & 2 \\ \hline 3 & 3 & 3 & 3 \\ \hline 4 & 4 & 4 & 4 \\ \hline 5 & 5 & 5 & 5 \\ \hline\end{array}$$

Grid sample

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

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

Can we translate image by 1/2 pixel up?



$x(m,n)$

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

$y(m,n)$

1.5	2.5	3.5	4.5
1.5	2.5	3.5	4.5
1.5	2.5	3.5	4.5
1.5	2.5	3.5	4.5

Grid sample

x (columns), y (rows):

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

$\otimes$

u (columns), v (rows):

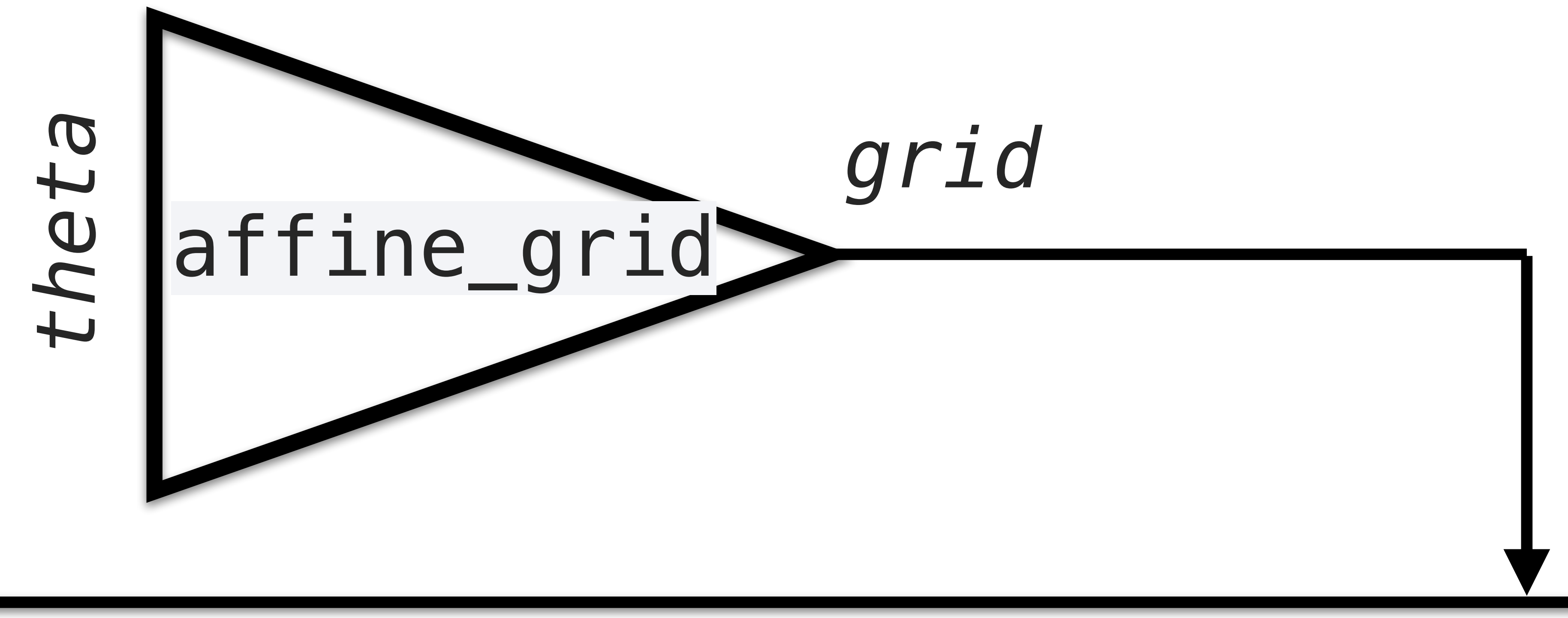
?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

$=$

$m = 2$, $n = 1$

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

Can we translate image by 1/2 pixel up?



$x(m,n)$

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

$y(m,n)$

1.5	2.5	3.5	4.5
1.5	2.5	3.5	4.5
1.5	2.5	3.5	4.5
1.5	2.5	3.5	4.5

Grid sample

x (columns), y (rows)

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

$\otimes$

u (columns), v (rows)

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

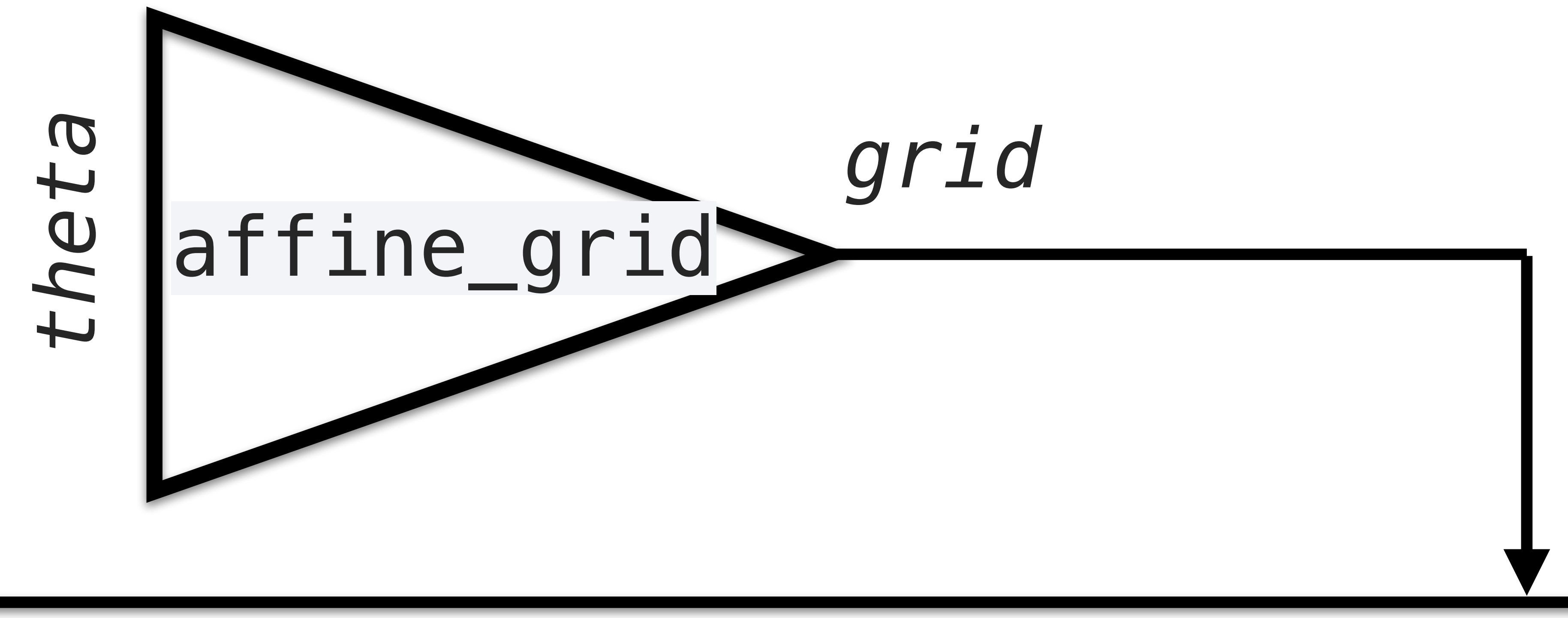
$=$

$m = 2$, $n = 1$

?			

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

Can we translate image by 1/2 pixel up?



$x(m,n)$

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

$y(m,n)$

1.5	2.5	3.5	4.5
1.5	2.5	3.5	4.5
1.5	2.5	3.5	4.5
1.5	2.5	3.5	4.5

Grid sample

x (rows indexed by y)

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

$\otimes$

u (rows indexed by v)

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

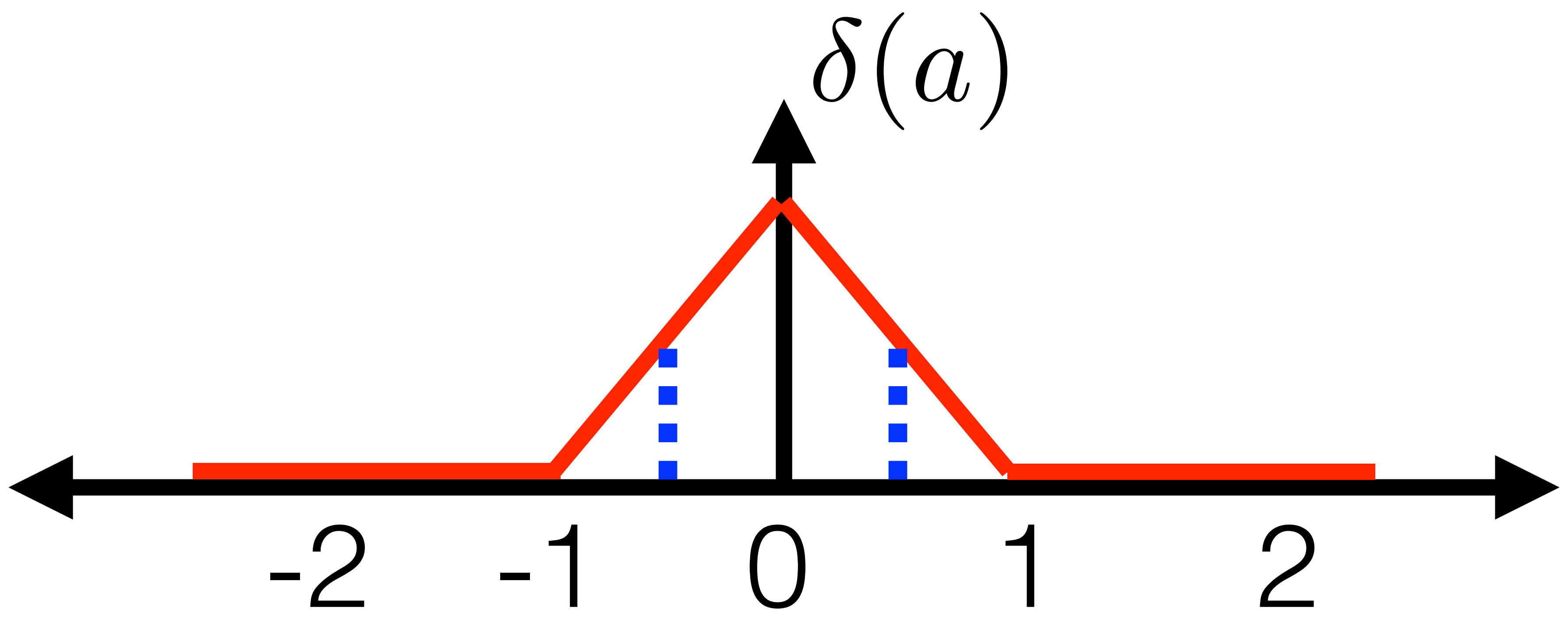
$=$

$m = 2$, $n = 1$

3.5			

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

Can we rotate image?



Grid sample

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

$\otimes$

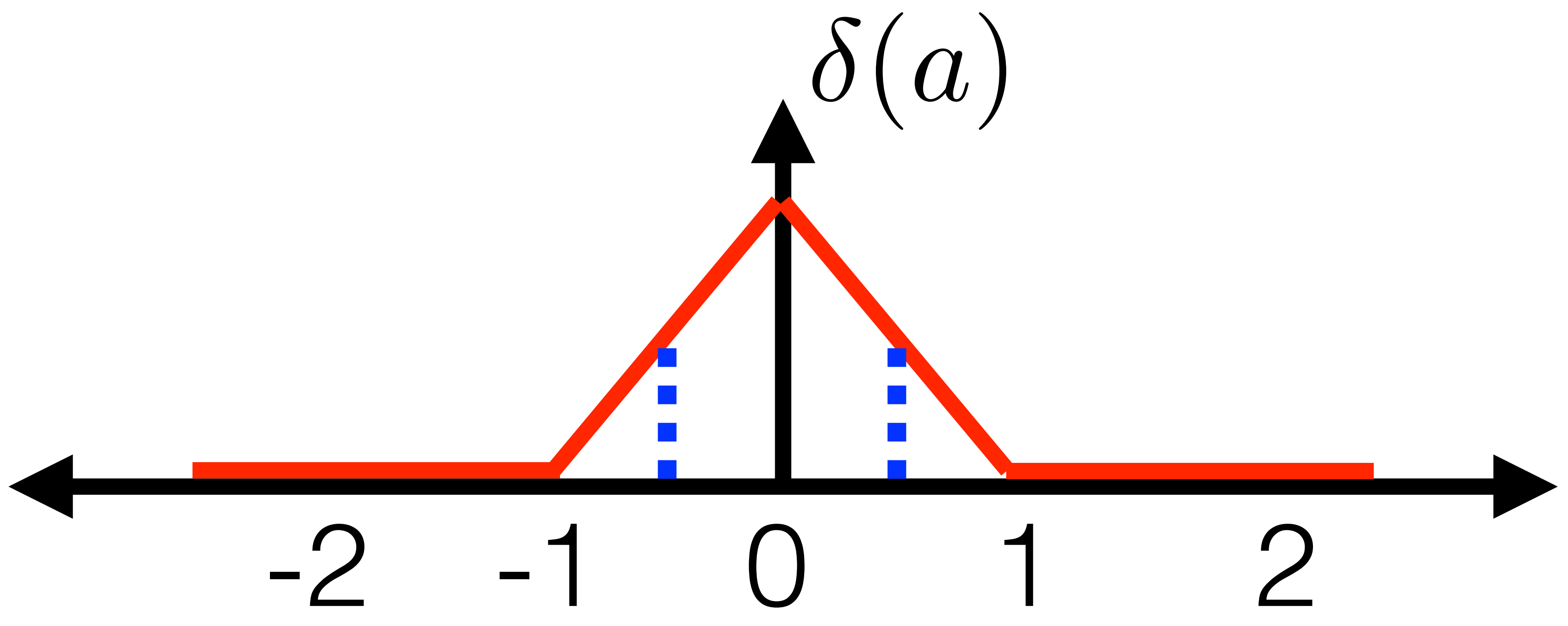
0	0	0	0
0	0	0	0
0.3	0.3	0	0
0.2	0.2	0	0

=

		1.1	

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

Can we rotate image?



Grid sample

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

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

$\otimes$

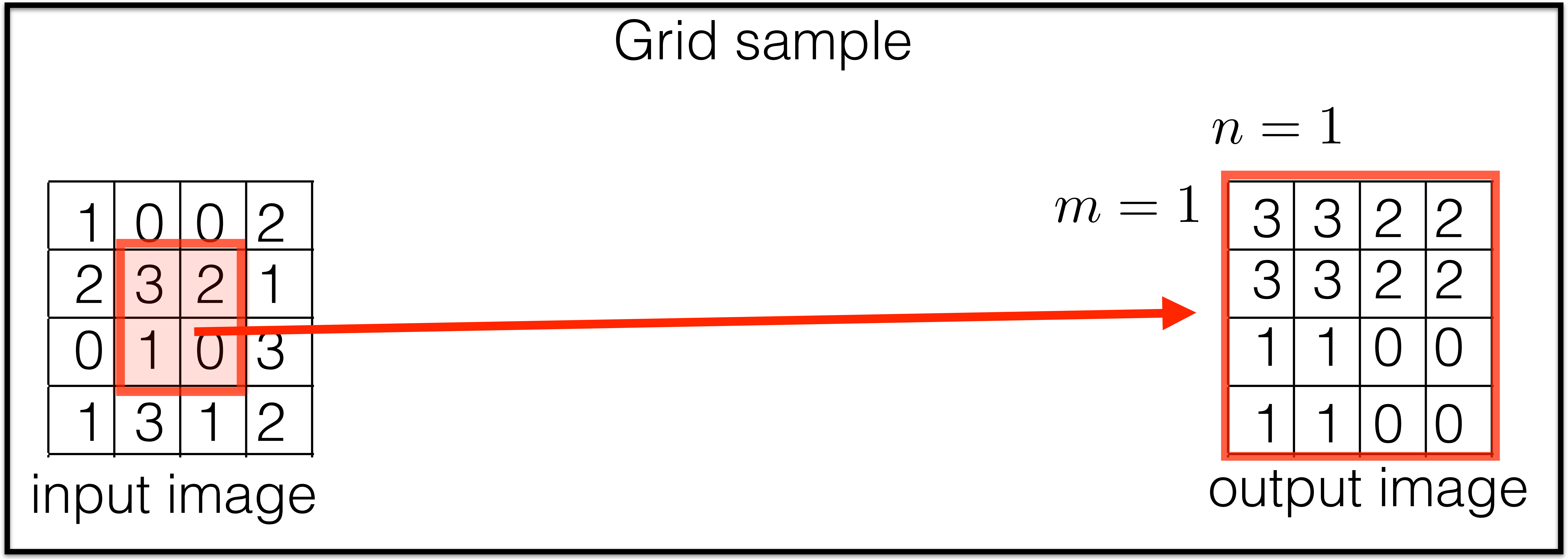
0	0	0	0
0	0	0	0
0.3	0.3	0	0
0.2	0.2	0	0

=

		1.1	

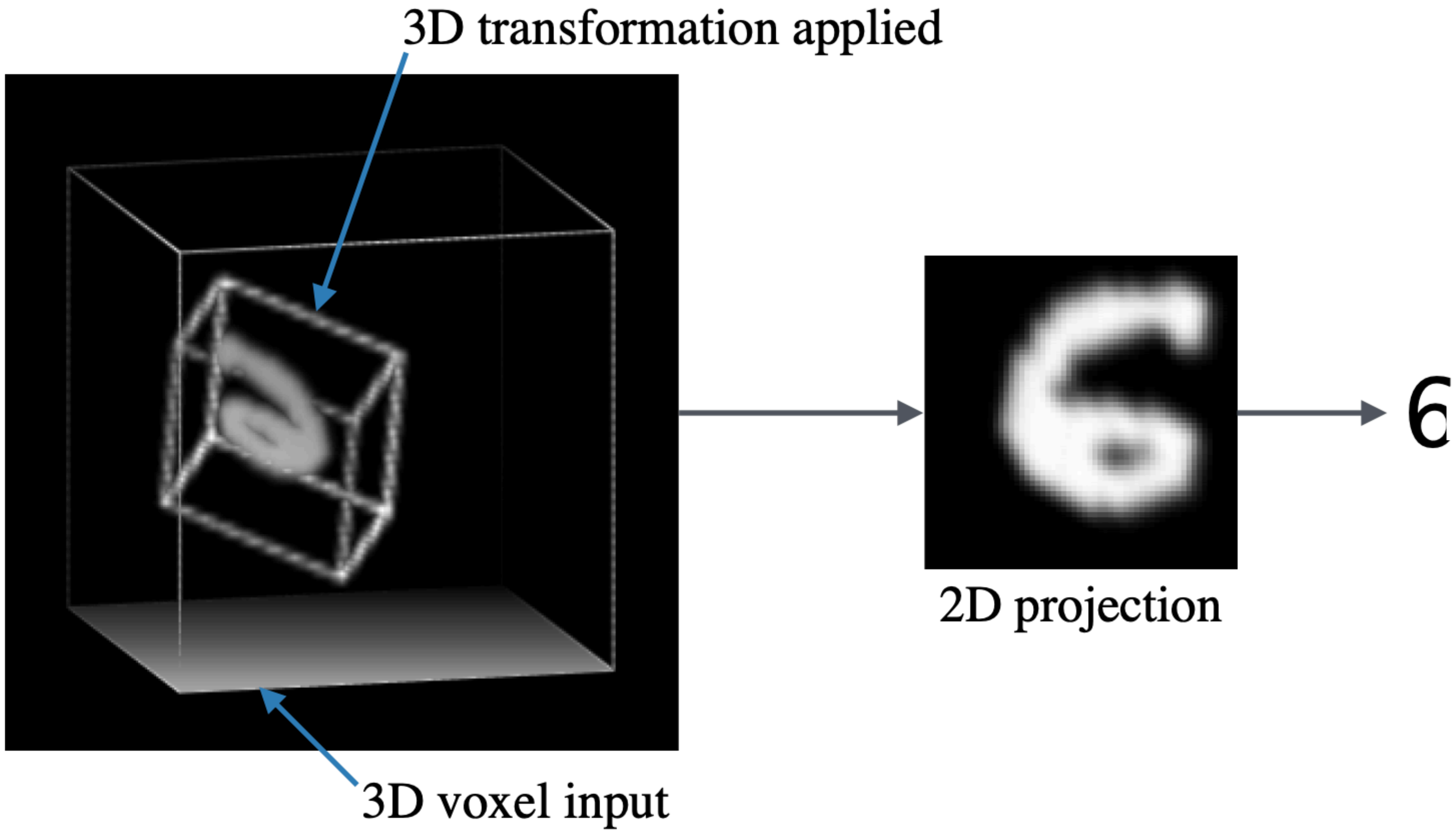
$$\kappa_{(m,n)}(u,v) = \delta\big(u - x(m,n)\big) \cdot \delta\big(v - y(m,n)\big)$$

Can we crop or upsample sub-image?



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

Also implemented 3D affine_grid transformation layer: $[\mathbf{R}\ \mathbf{t}] \in \mathcal{R}^{3\times4}$



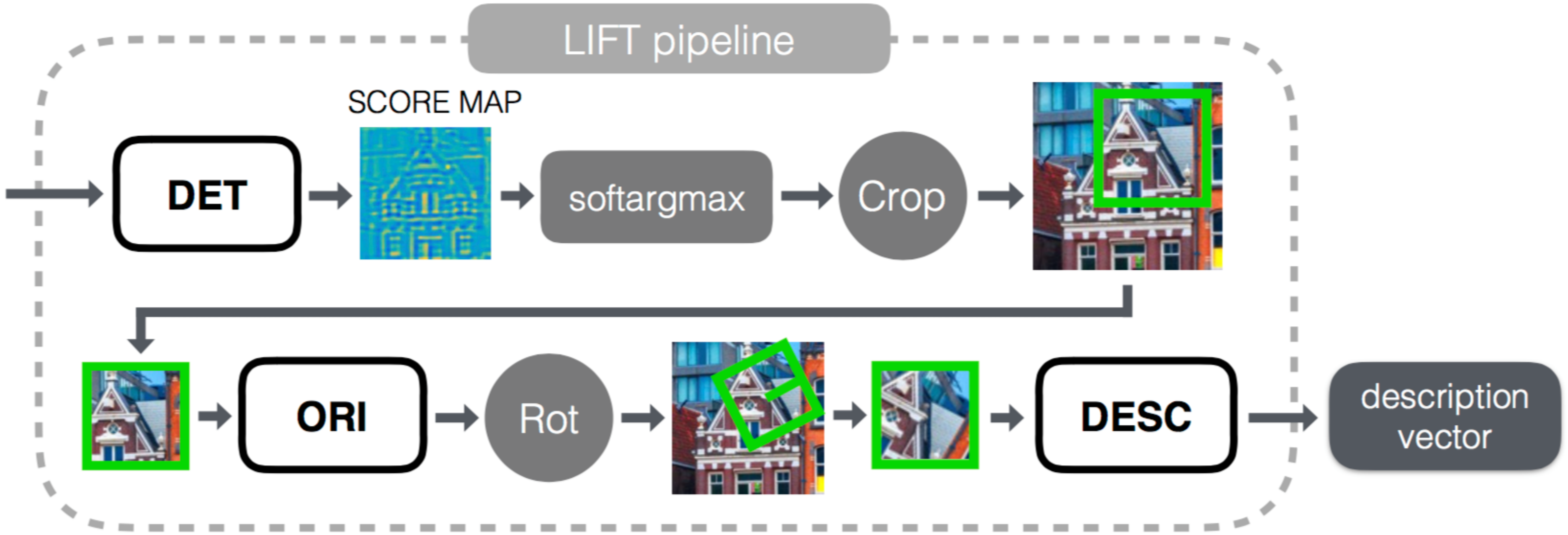

```
torch.nn.functional.affine_grid(theta, size, align_corners=None)

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

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

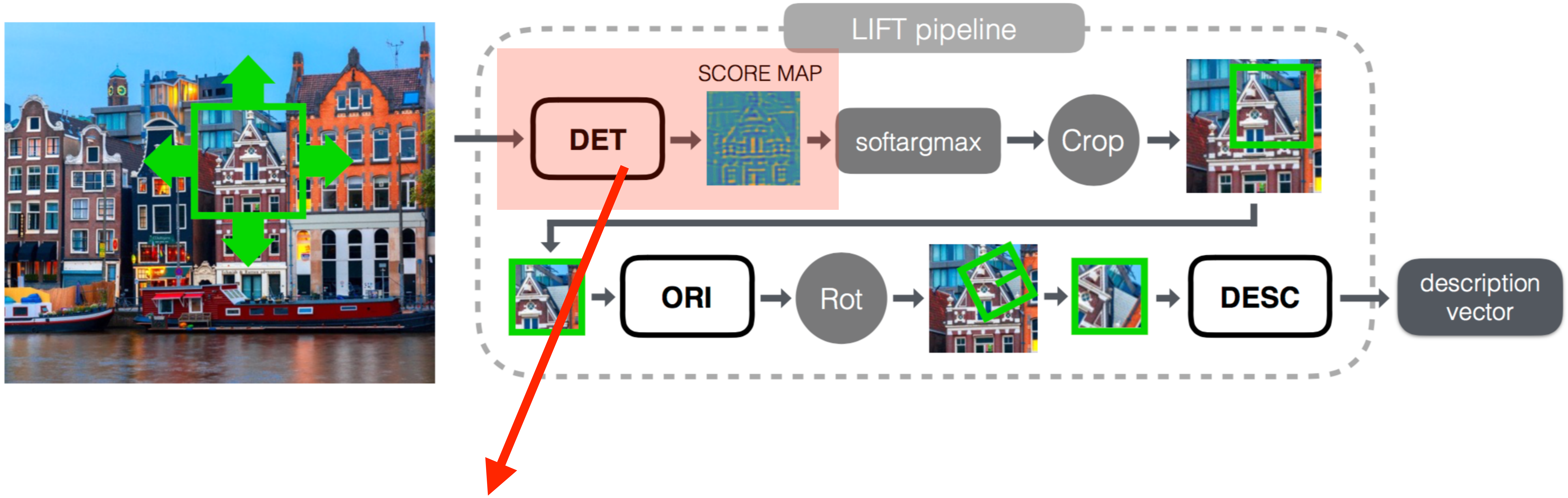


Input: RGB image

Output: set of detected feature points with descriptors

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

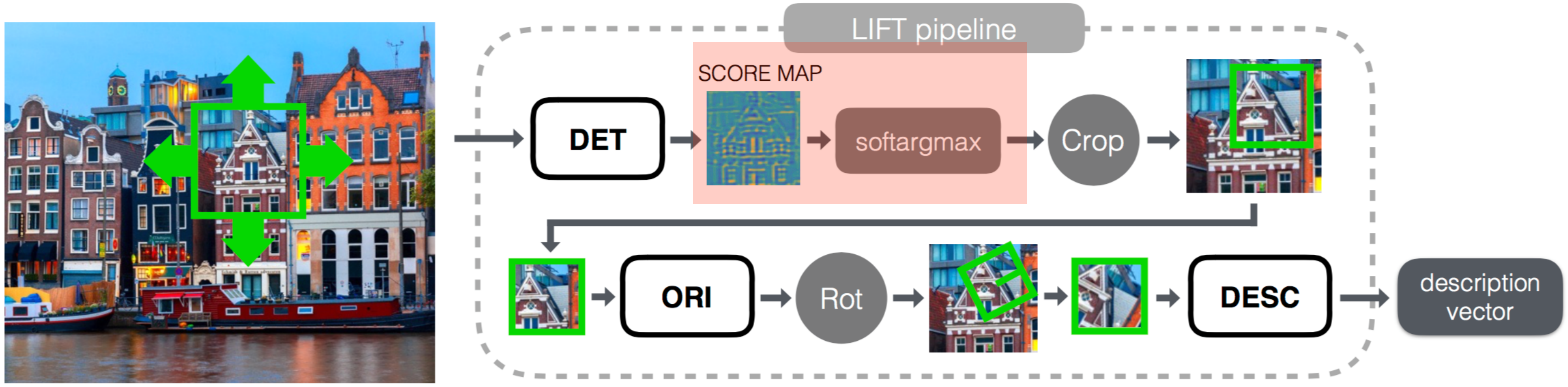
LIFT: Learnable Invariant Feature Descriptors
[Yi et al ECCV 2016] https://arxiv.org/abs/1603.09114



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

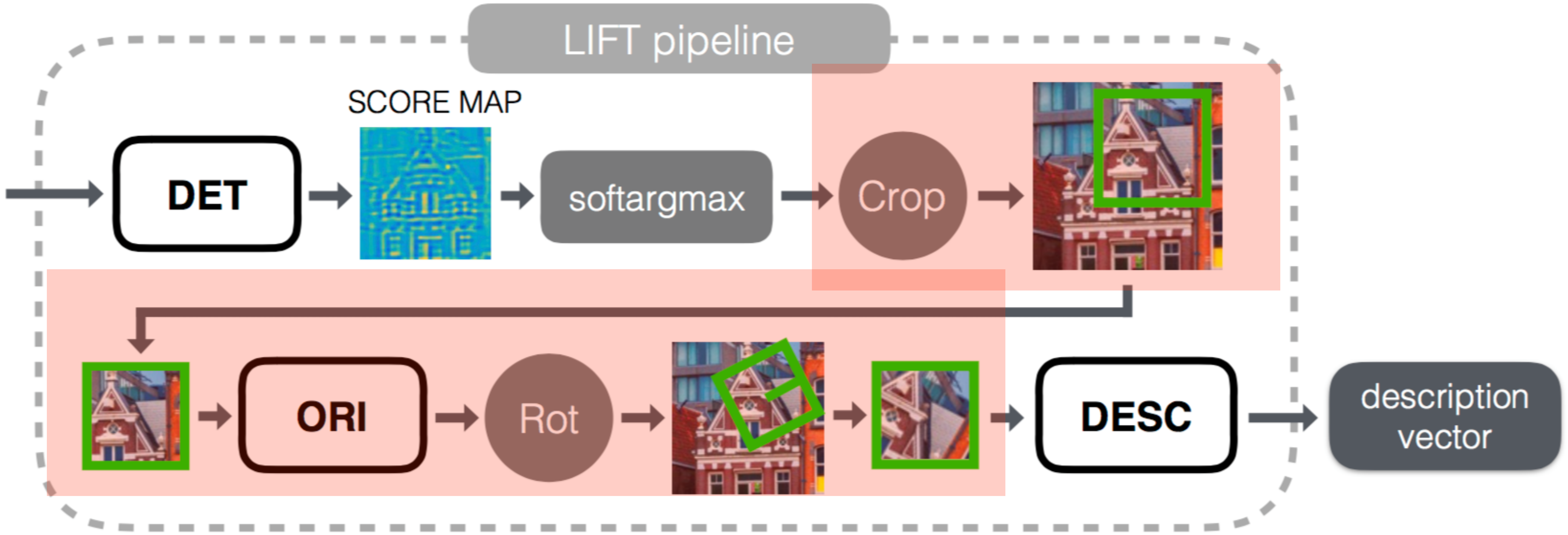


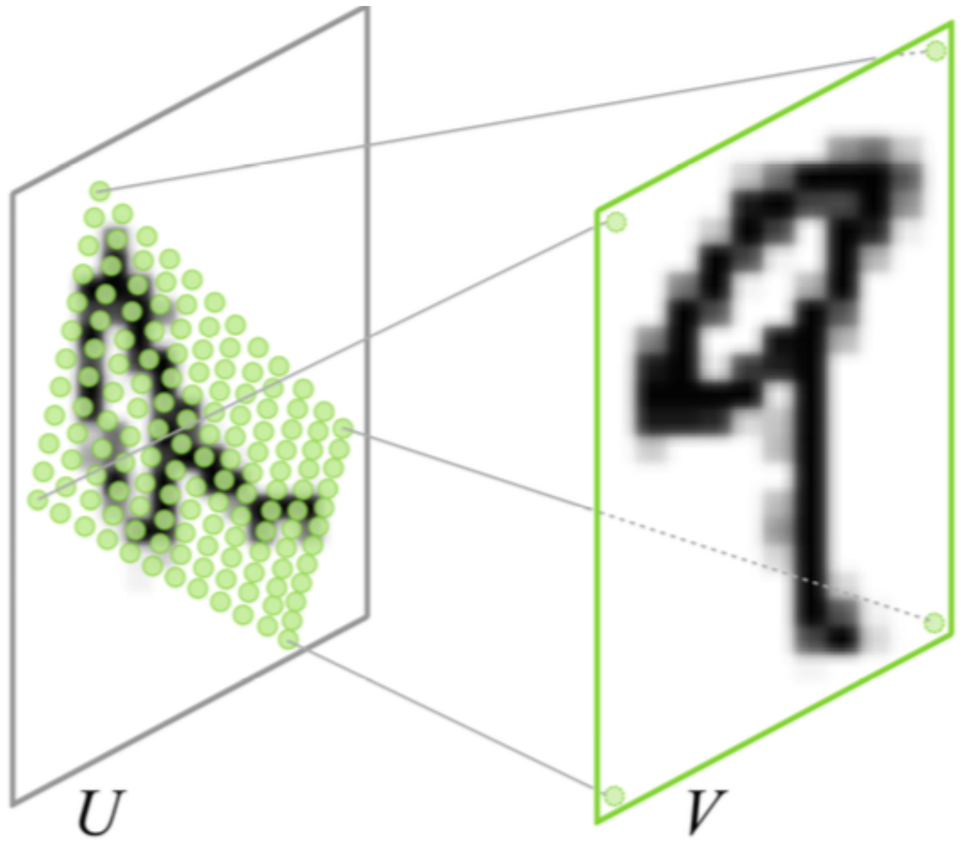
Score map

$$\text{softargmax}\left(\mathbf{S}\right) = \frac{\sum_{\mathbf{y}} \exp(\beta \mathbf{S}(\mathbf{y}))\mathbf{y}}{\sum_{\mathbf{y}} \exp(\beta \mathbf{S}(\mathbf{y}))}$$

Expected value of y weighted by the softmax prob. distr.

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



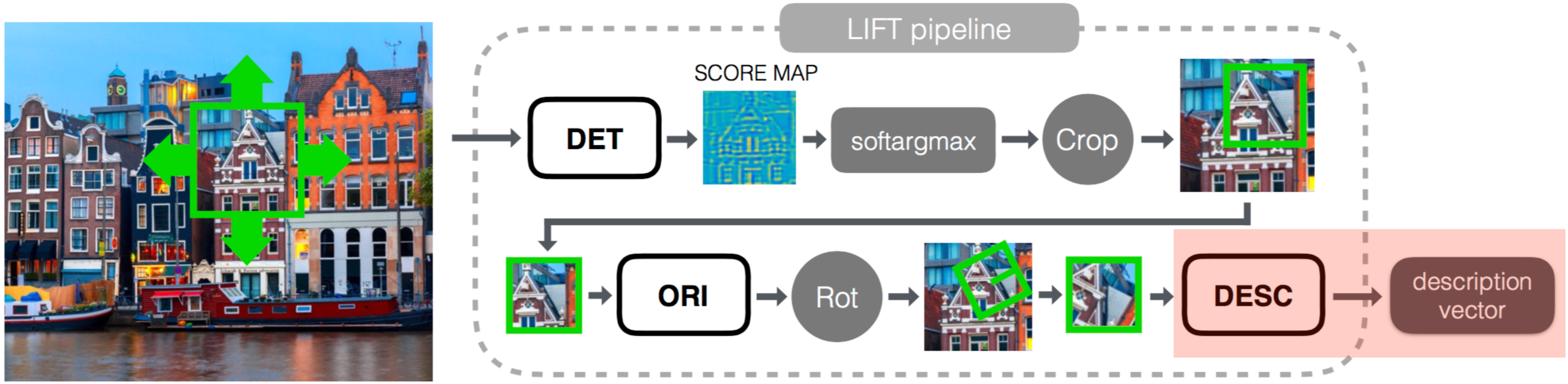


Spatial Transformer Network

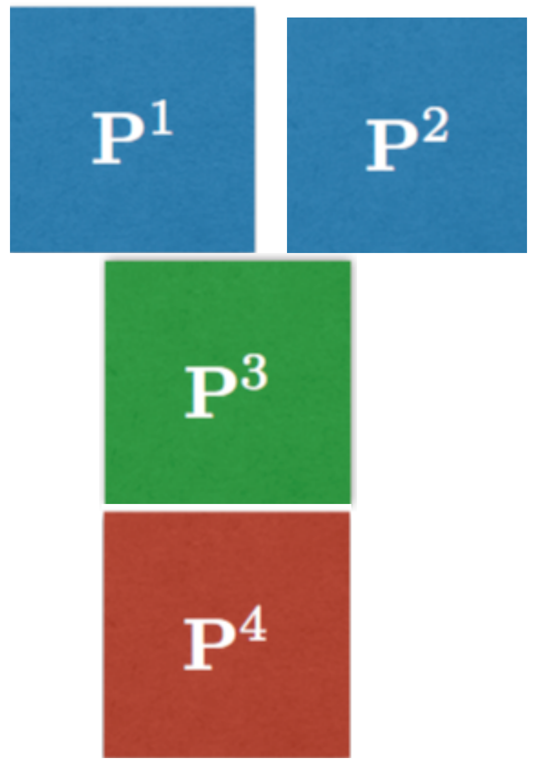
Bilinear approximation of affine transformation is differentiable !

[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-tuples:

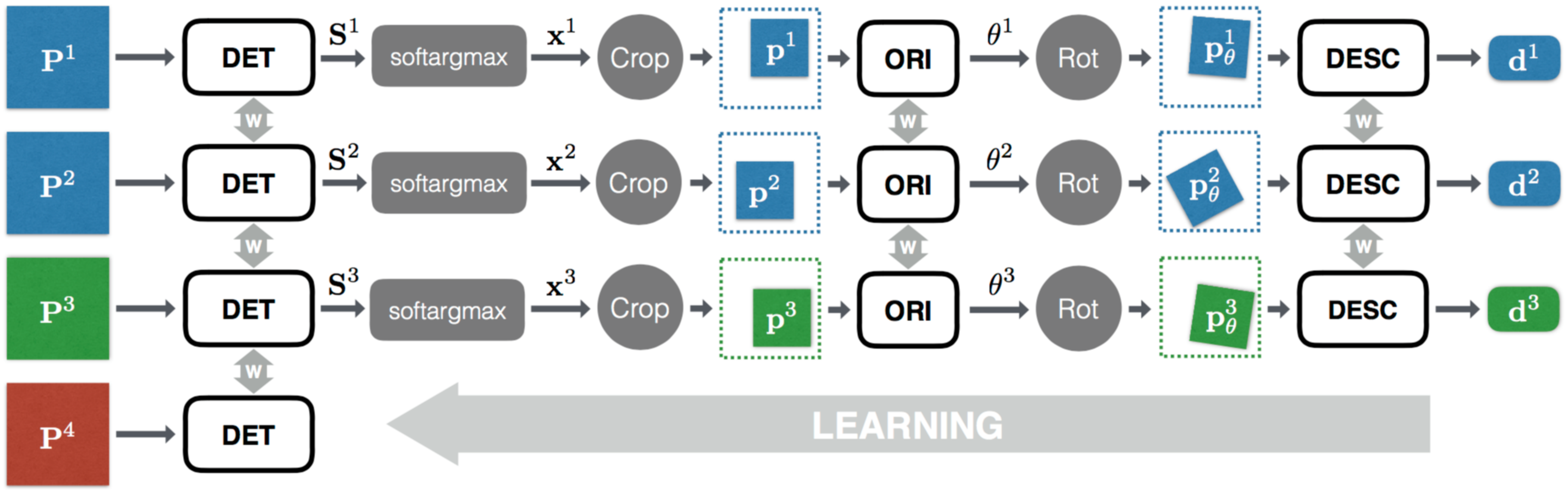


Two corresponding patches on distinctive points

One not corresponding patch on a distinctive point (has different corr.)

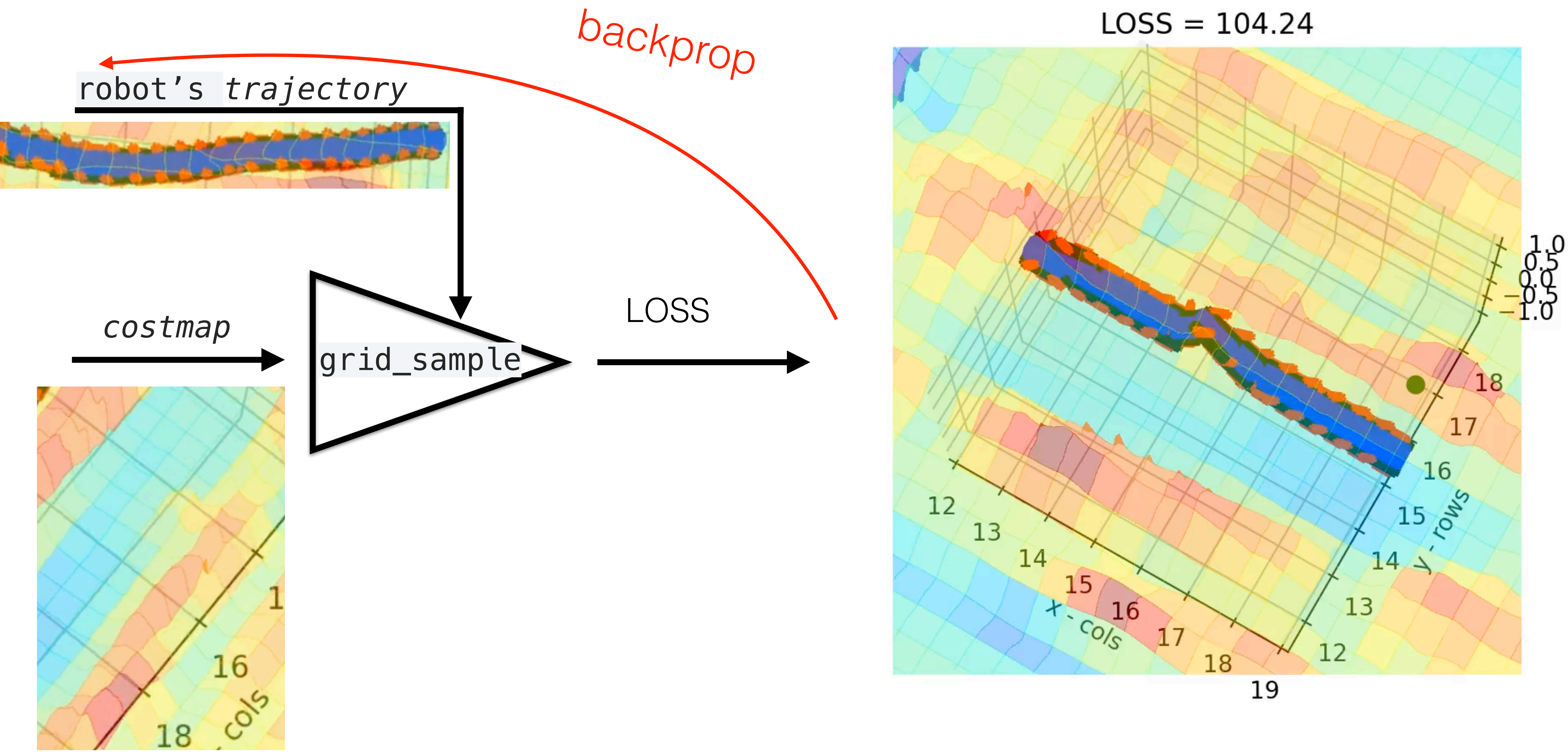
One patch on a not distinctive point (has no correspondence at all)

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

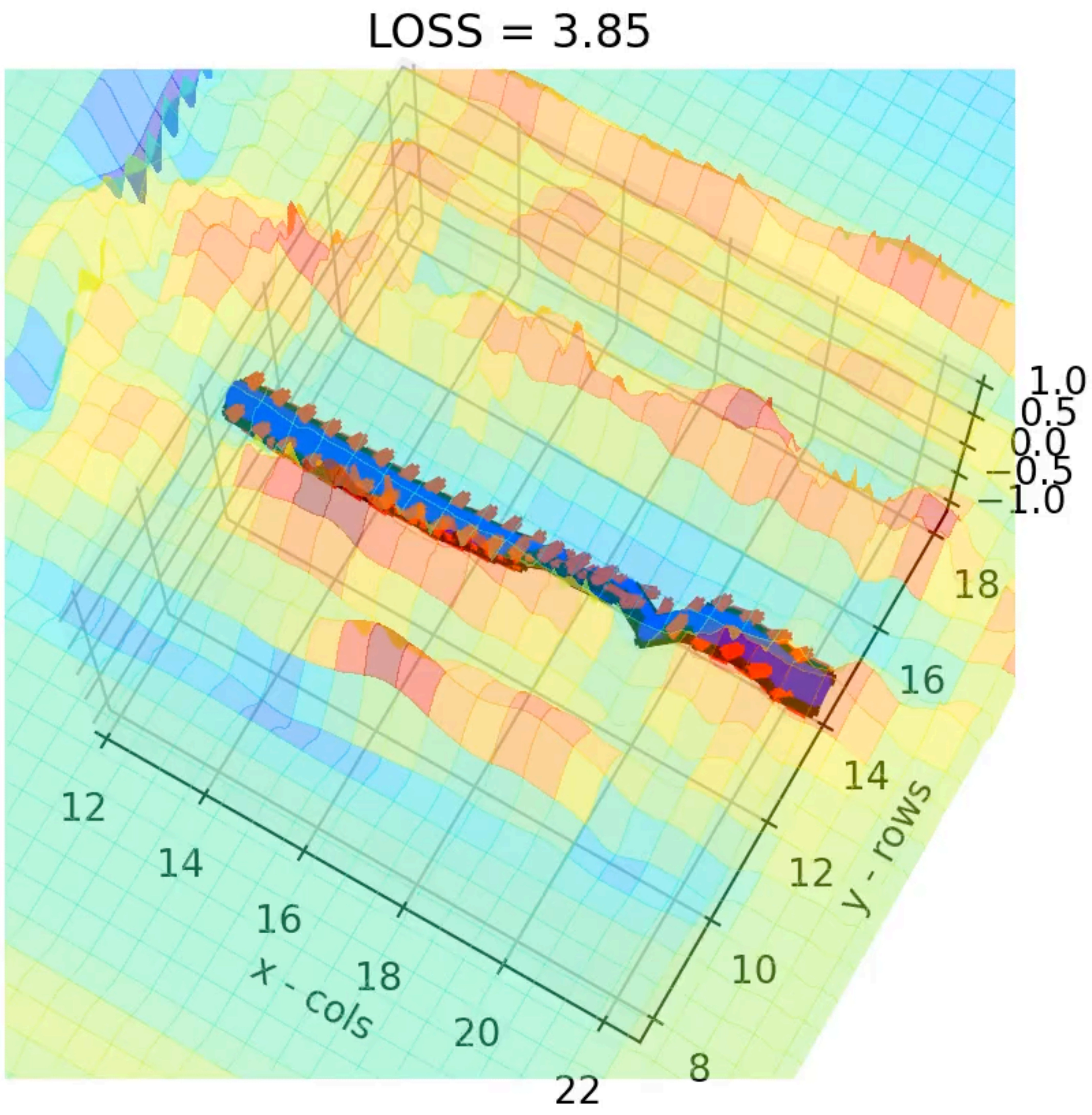


- All patches are fed into the network and differentiable loss is backpropagated
- Loss makes:
  - d1 and d2 as close as possible,
  - d3 as far as possible (from d1 and d2)
  - DET to have high response on p1,p2,p3 and small on p4

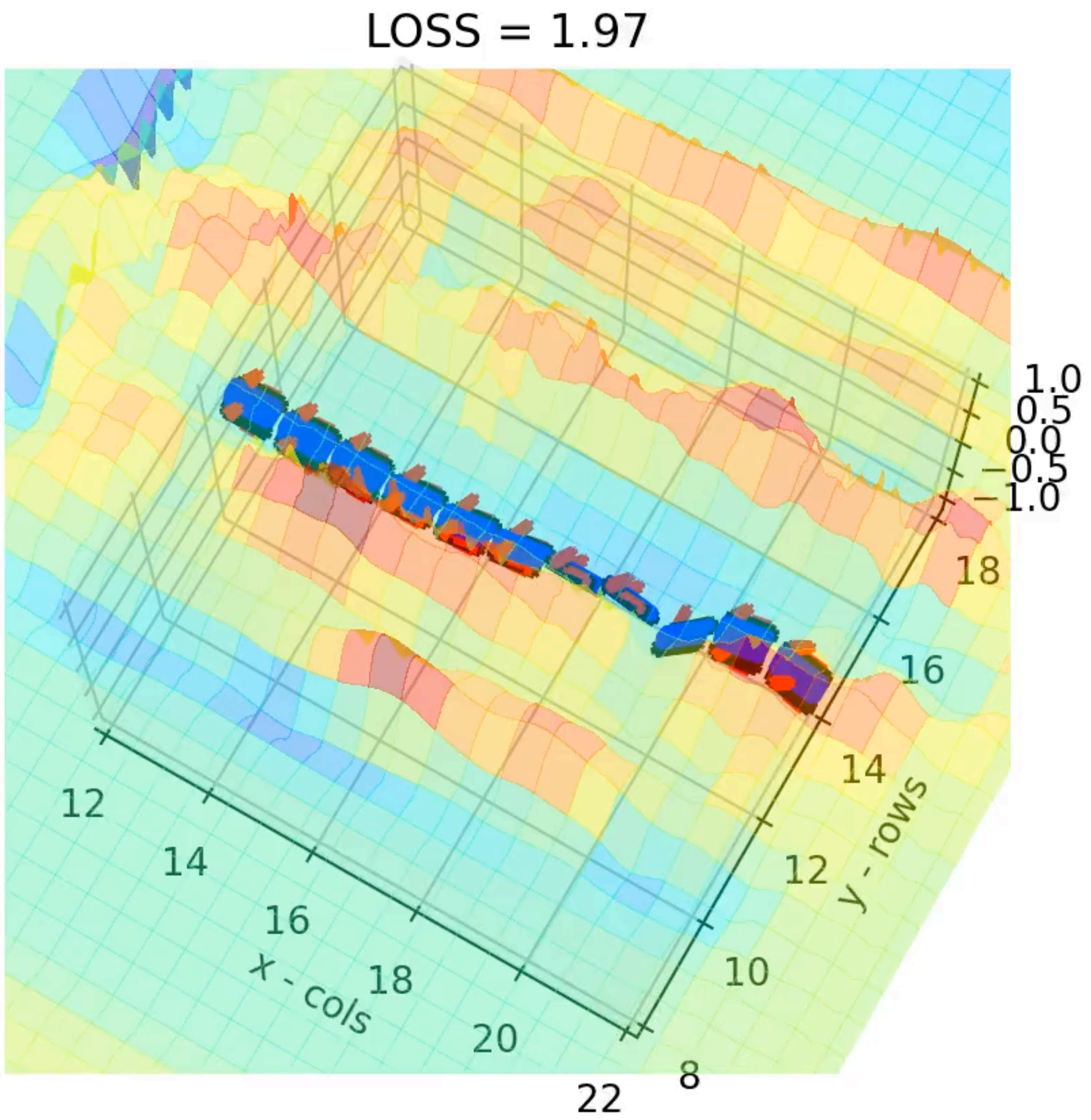
Trajectory optimization



Trajectory optimization



dense trajectory (21 samples)



sparse trajectory (11 samples)

Summary architectures

- Deeper architectures, with small kernels, skip-connections and batch-norms (e.g. ResNet, DenseNet) seems reasonable.
- Reception field of pixels in a feature map determine their ability to infer from spatial context.
- Atrous spatial pyramid seems to be viable replacement for max-pooling
- Argmax is not differentiable, but it can be replaced by expected value.
- Spatial Transform Layer allows to capture spatial transformation (e.g. interpolation, cropping, projectivetransformation of rigi bodies,…)
- A lot of dark-magic needed for successful training