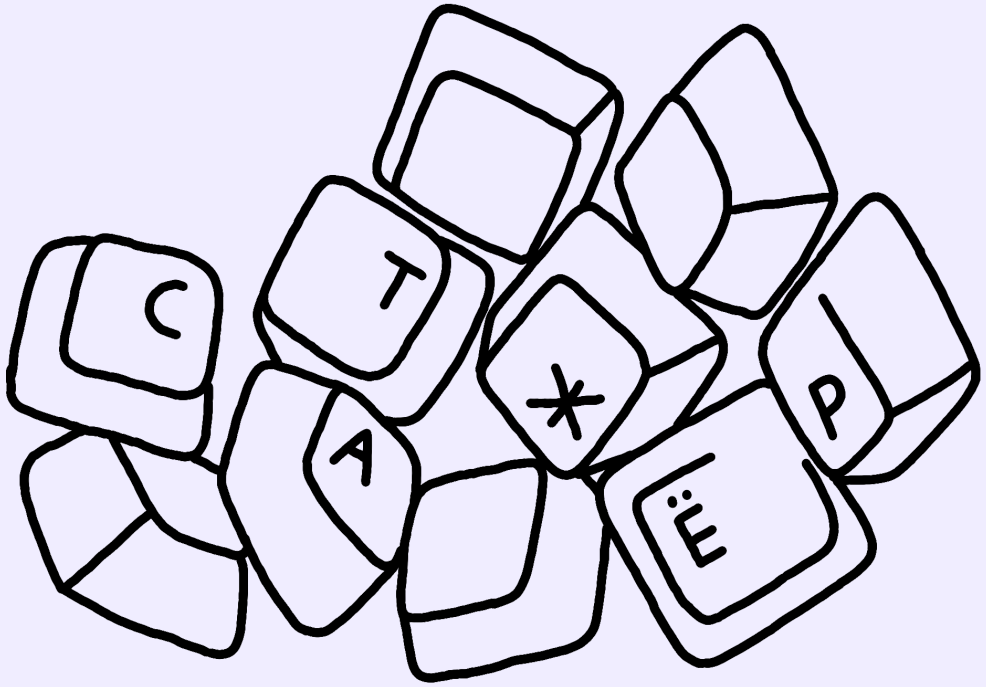
Detection & Segmentation

YOUNG & YANDEX

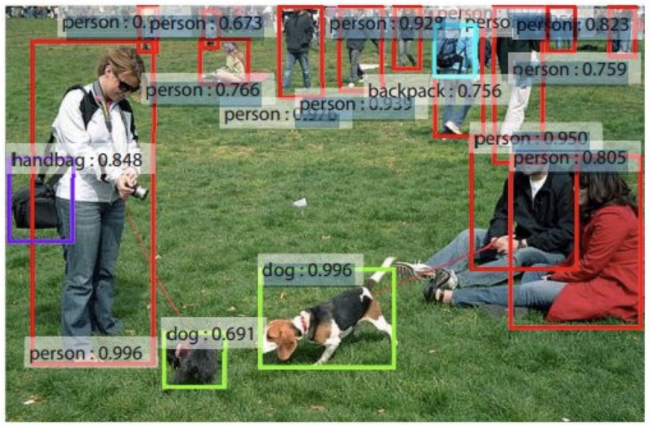
Радослав Нейчев
Выпускник и преподаватель ШАД и МФТИ,
руководитель группы ML-разработки в Яндексе,
основатель girafe ai

Computer Vision tasks



01

Computer Vision tasks



- Classification
- Localisation
- Detection

- Semantic Segmentation
- Instance Segmentation

- Style Transfer
- Adversarial Attacks

- Human Pose Estimation
- Person Re-Identification
- Object Tracking
- ……

Classification

Cat

Input image (C x H x W)

Output

class label (int)

Localization

Input image (C x H x W)

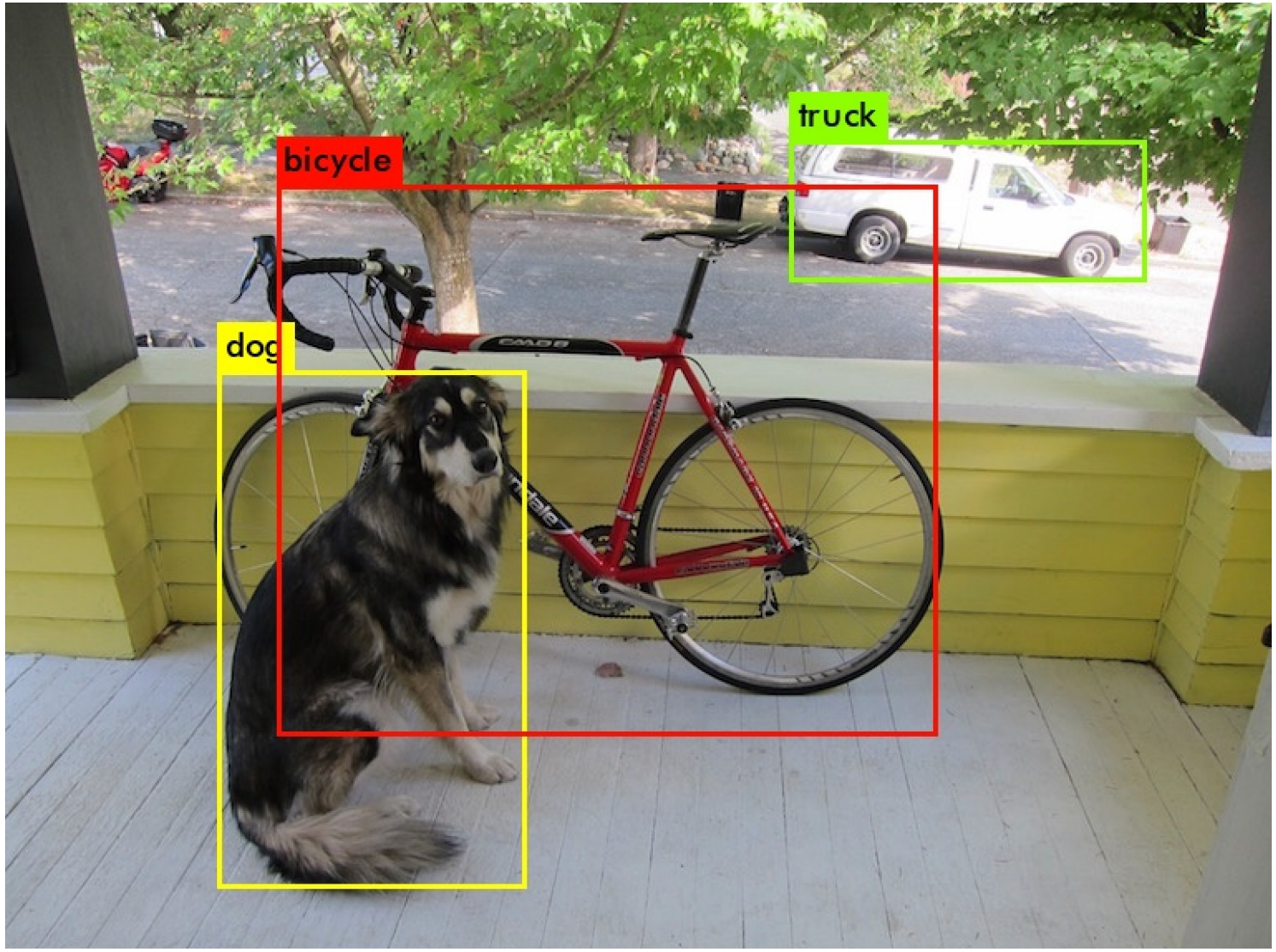
Output

class label (int), BB* coordinates (4 ints)

Assumption

one object per image

Detection



Input image (C x H x W)

Output

class label (int), confidence (0-1 float), BB (4 ints)

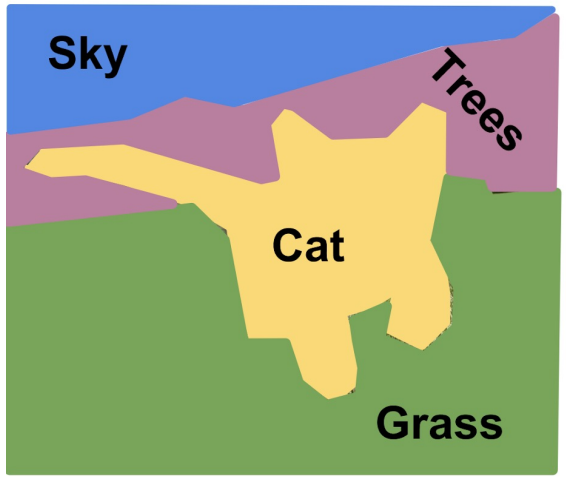
……

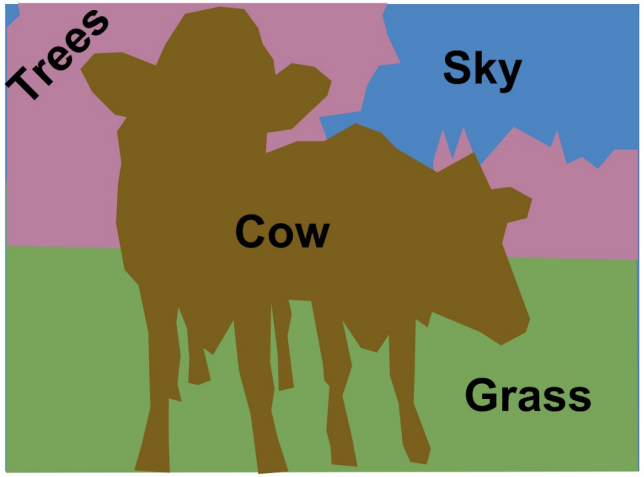
class label (int), confidence (0-1 float), BB (4 ints)

Assumption

multiple objects per image

Semantic Segmentation





Input image (C x H x W)

Output

mask of classes (1 x H x W)

Assumption

just class label for each pixel

Instance Segmentation

Input image (C x H x W)

Output

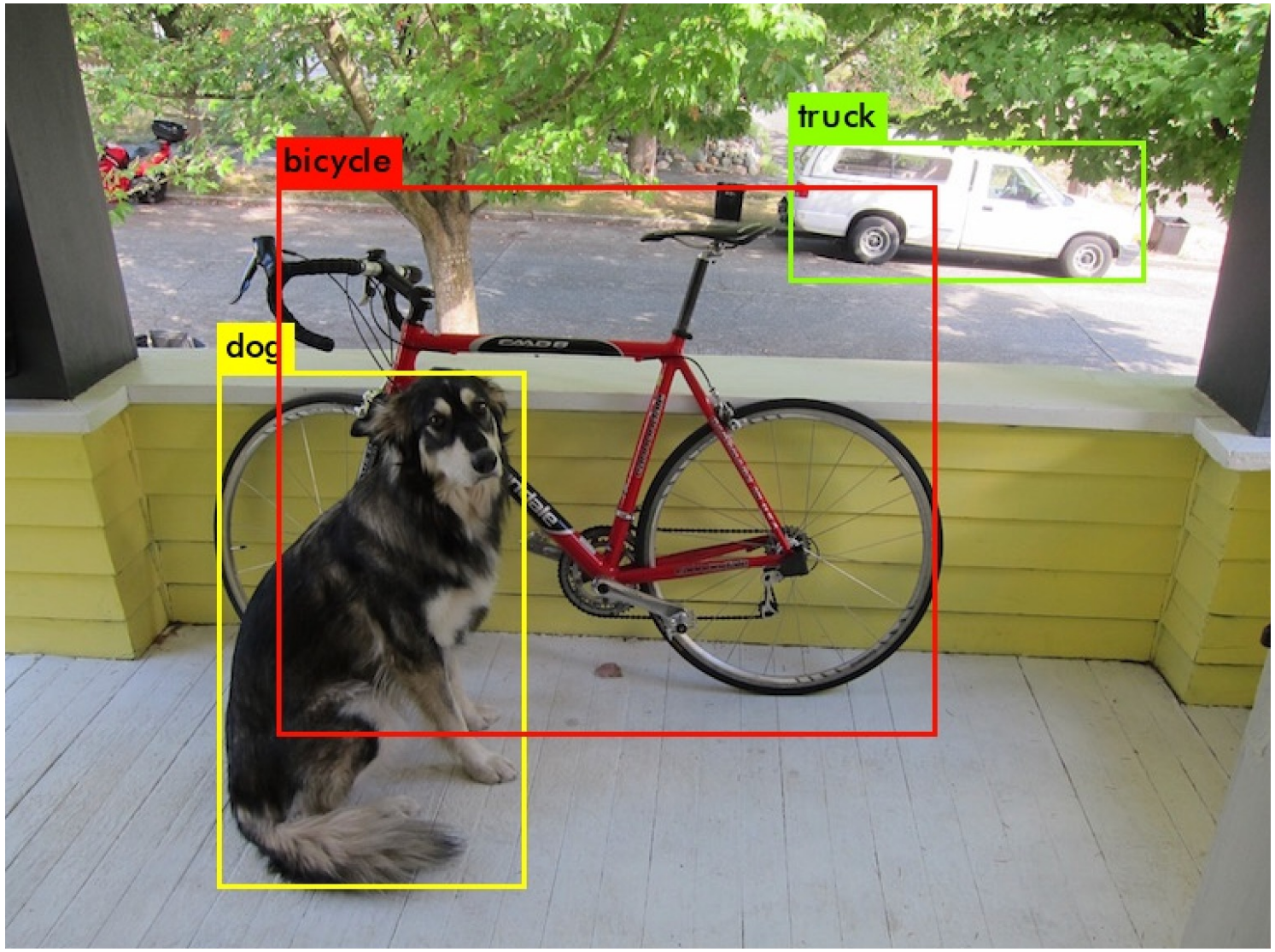
mask of class instances
(detection + 1 x H x W)

Assumption
separated masks for each distinct object

Detection metrics

02

Detection



Input image (C x H x W)

Output

class label (int), confidence (0-1 float), BB (4 ints)

……

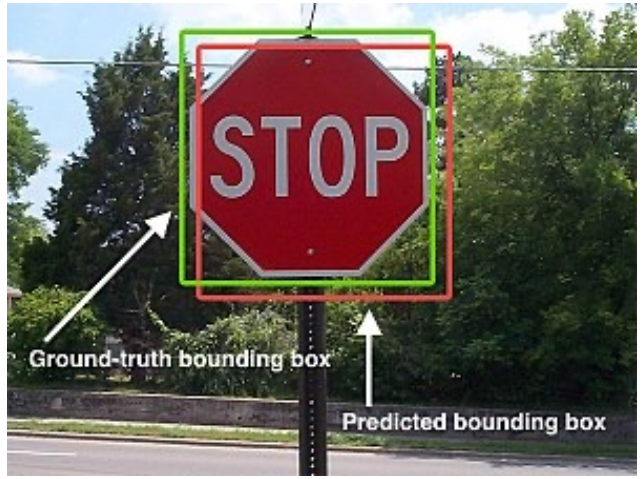
class label (int), confidence (0-1 float), BB (4 ints)

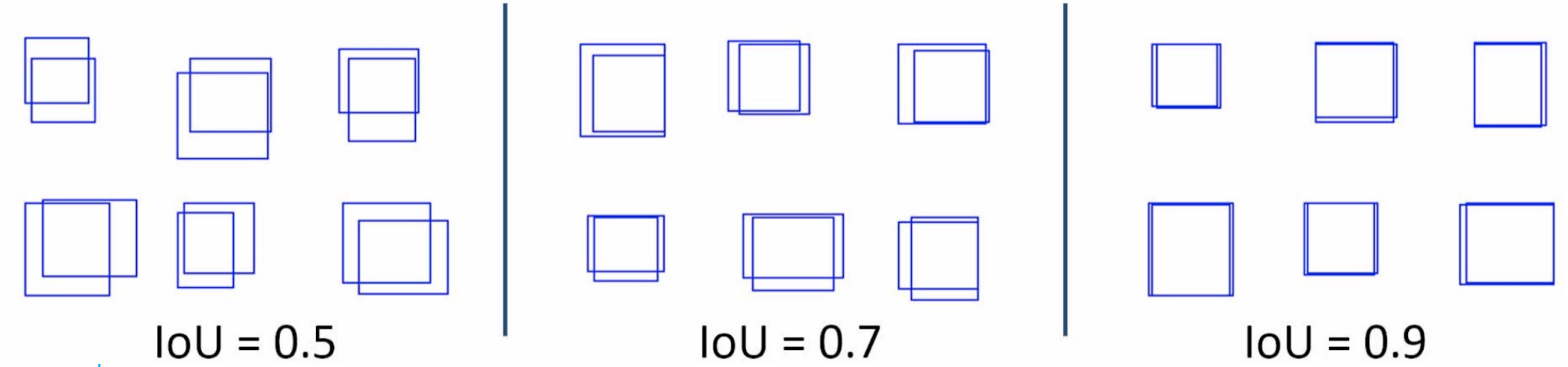
Assumption

multiple objects per image

Intersection over Union (IoU)

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

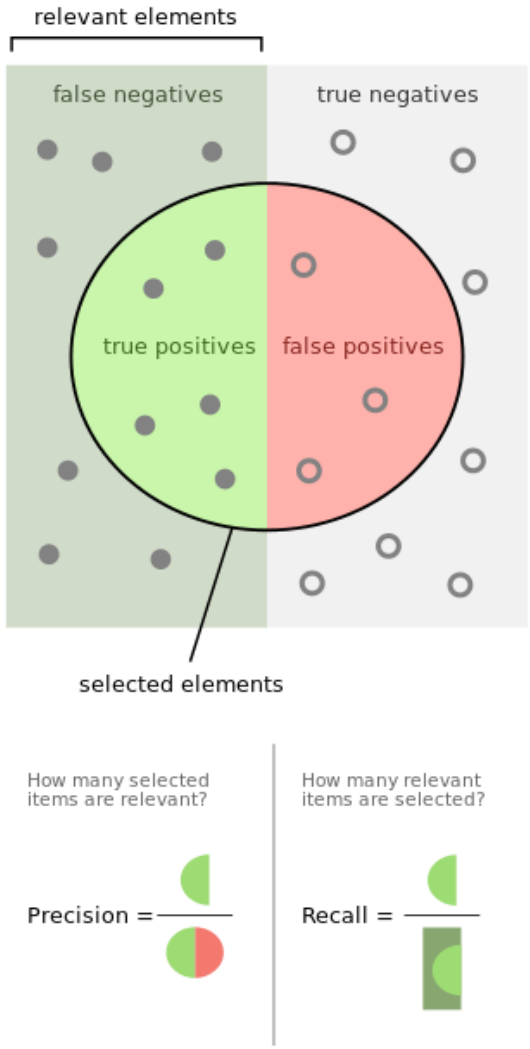




aka Jaccard index

Source: https://pyimagesearch.com

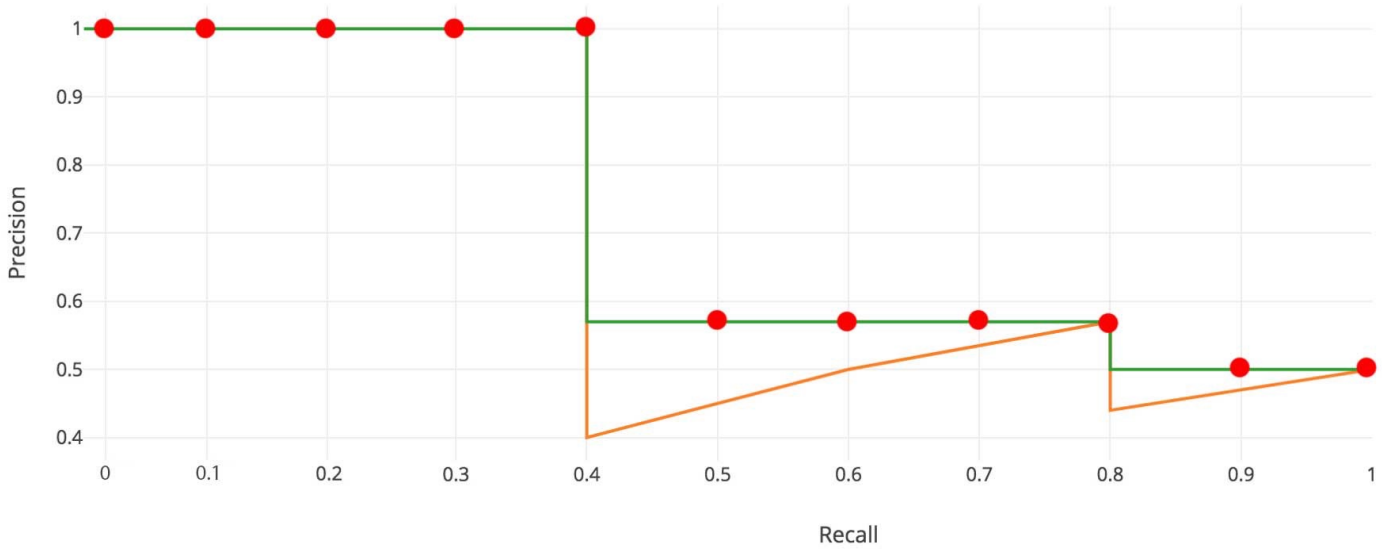
mean Average Precision (mAP)



Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

mean Average Precision (mAP)

mAP is mean of AP over all classes



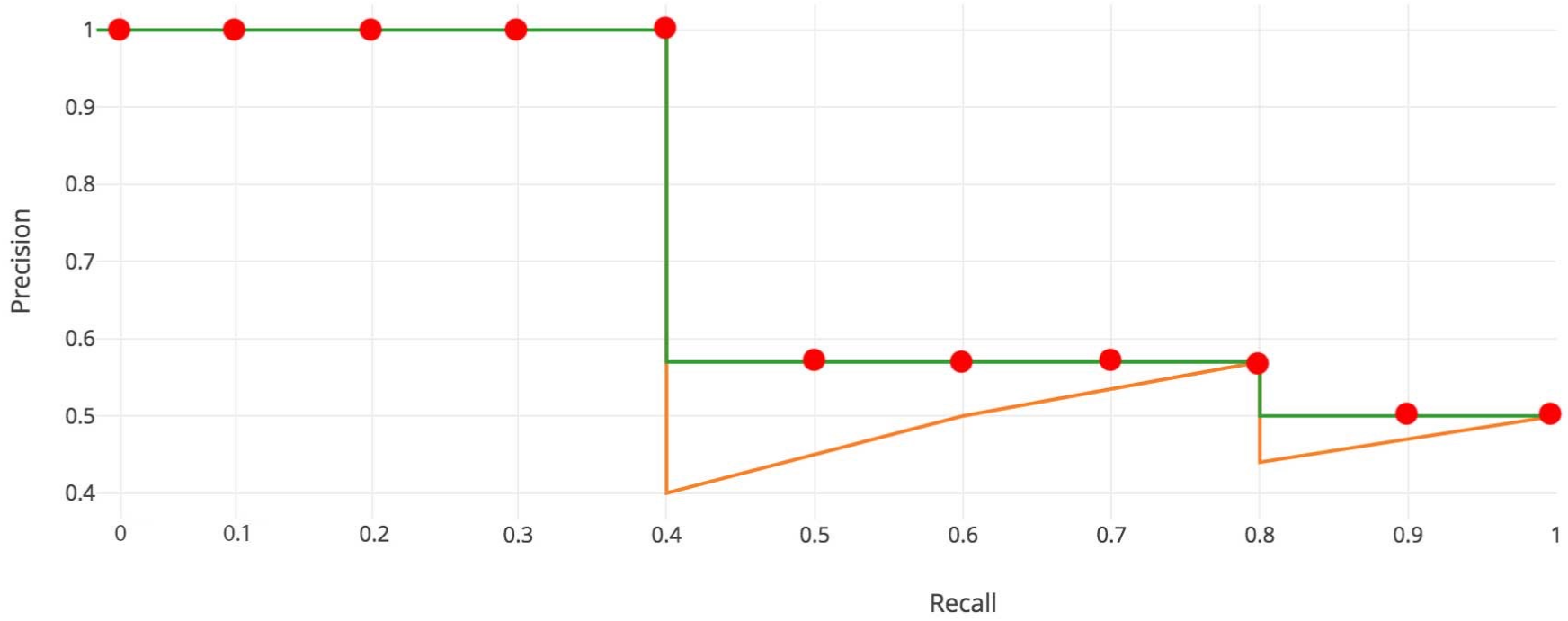
Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

$$AP = \frac{1}{11} \times \left(AP_r(0) + AP_r(0.1) + \ldots + AP_r(1.0) \right)$$

mean Average Precision (mAP)

mAP is mean of AP over all classes

$$AP = \frac{1}{11} \times \left(AP_r(0) + AP_r(0.1) + \ldots + AP_r(1.0) \right)$$



COCO mAP

Average Precision (AP):

AP	% AP at IoU=.50:.05:.95 **(primary challenge metric)**
$AP^{IoU=.50}$	% AP at IoU=.50 (PASCAL VOC metric)
$AP^{IoU=.75}$	% AP at IoU=.75 (strict metric)

AP Across Scales:

AP^{small}	% AP for small objects: area < 32^2
AP^{medium}	% AP for medium objects: 32^2 < area < 96^2
AP^{large}	% AP for large objects: area > 96^2

Average Recall (AR):

$AR^{max=1}$	% AR given 1 detection per image
$AR^{max=10}$	% AR given 10 detections per image
$AR^{max=100}$	% AR given 100 detections per image

AR Across Scales:

AR^{small}	% AR for small objects: area < 32^2
AR^{medium}	% AR for medium objects: 32^2 < area < 96^2
AR^{large}	% AR for large objects: area > 96^2

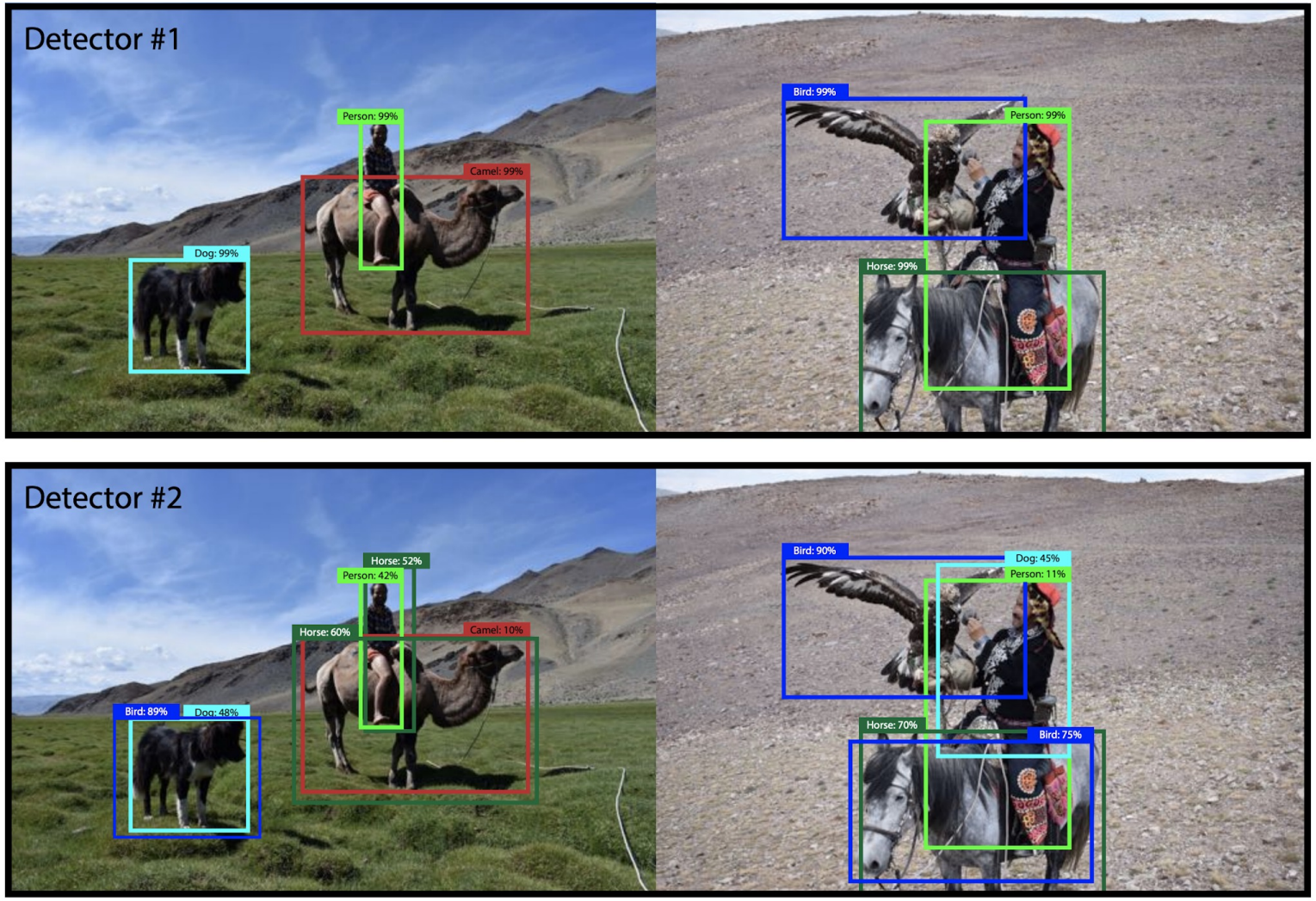
See https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

mean Average Precision (mAP)

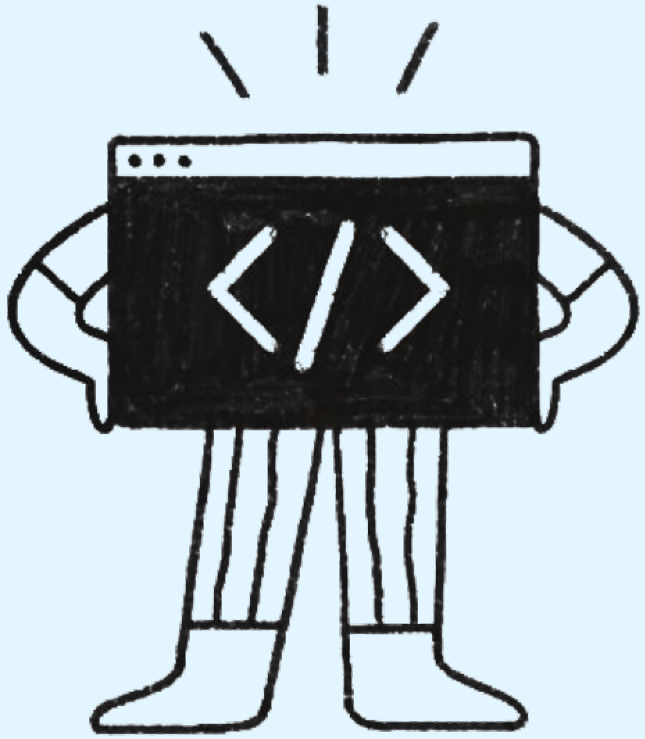
Problems

These two hypothetical detectors are perfect according to mAP over these two images. They are both perfect. Totally equal

(from YOLOv3 article)



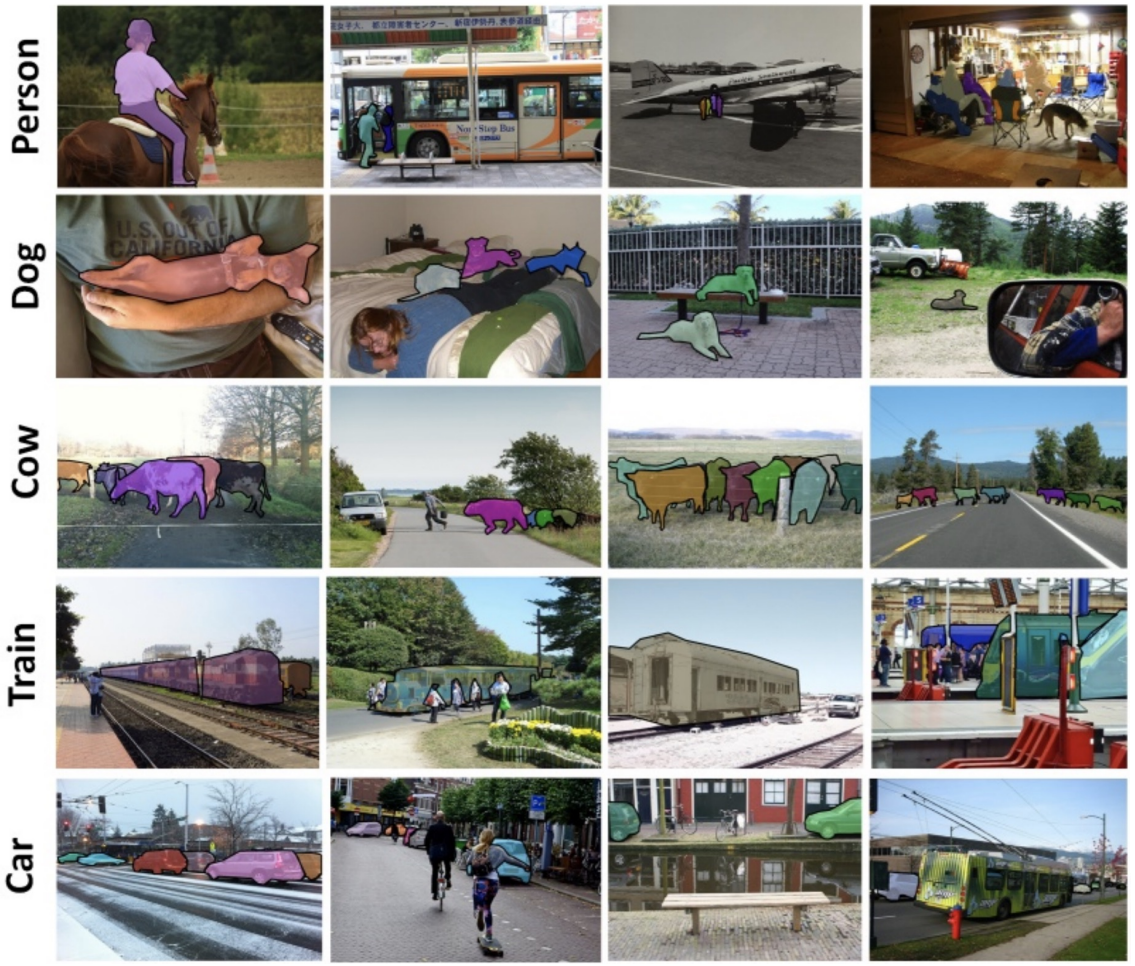
Datasets

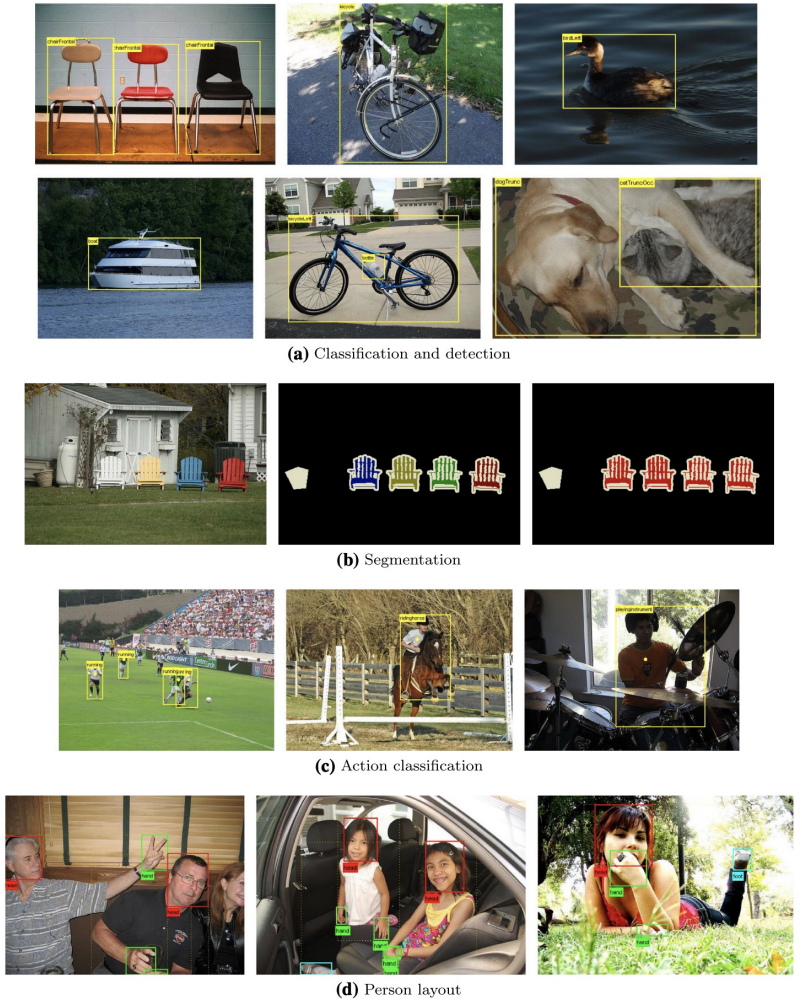


03

Detection datasets

1. PASCAL VOC 2012
2. ImageNet
3. COCO 2017
4. Open Images




(a) Classification and detection

(b) Segmentation

(c) Action classification

(d) Person layout

Thanks to Ilya Zakharkin for materials

PASCAL VOC 2012

- PASCAL Visual Object Classes
- developed since 2005 and 4 classes
- 11,5k images
- 20 classes
- 31,5k detections
- 7k segmentations
- Images came from flickr.com

Table 1 The VOC classes

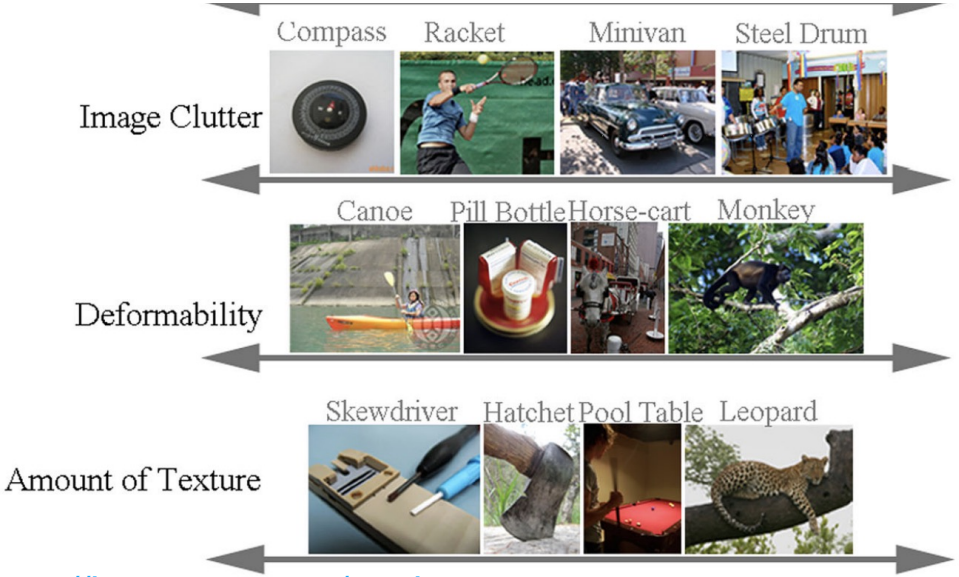
Vehicles	Household	Animals	Other
Aeroplane	Bottle	Bird	Person
Bicycle	Chair	Cat	
Boat	Dining table	Cow	
Bus	Potted plant	Dog	
Car	Sofa	Horse	
Motorbike	TV/Monitor	Sheep	
Train			

The classes can be considered in a notional taxonomy

Relevant dataset overview: http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham15.pdf

ImageNet

- organized according to the WordNet hierarchy
- 14.2m images
- 1000 classes
- 1m images with BBs
- more fine graded classes than PASCAL
- more diverse objects properties



source: http://image-net.org/explore

ImageNet

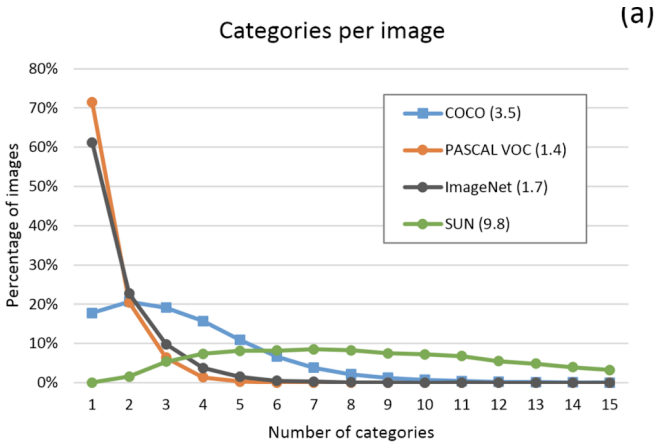
Imagenet is actually dataset of dogs species.

And the hardest task is to split pancakes and chihuahua

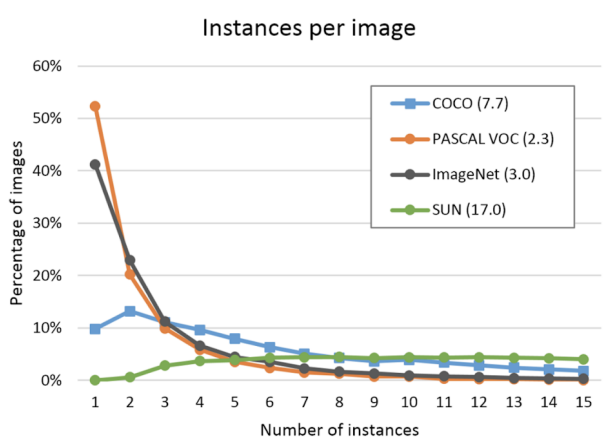
MS COCO 2017

- MicroSoft Common Objects in COntext
- 328k images
- 91 object categories
- 82 having more than 5,000 labeled instances
- 2,5m labeled instances in total
- 250k people with keypoints
- online exploration tool http://cocodataset.org/#explore

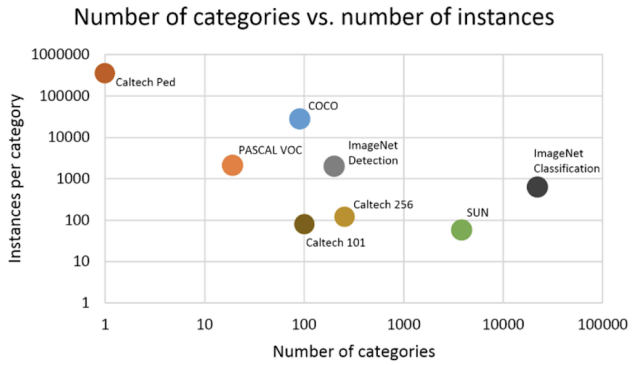
- overview https://arxiv.org/pdf/1405.0312.pdf



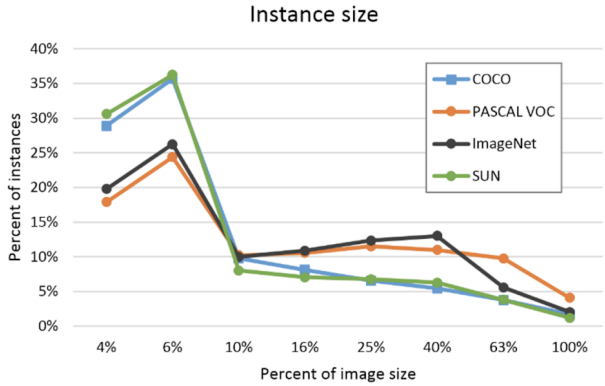
(b)



(c)



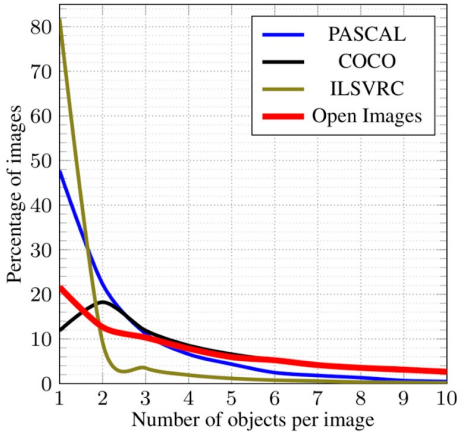
(d)

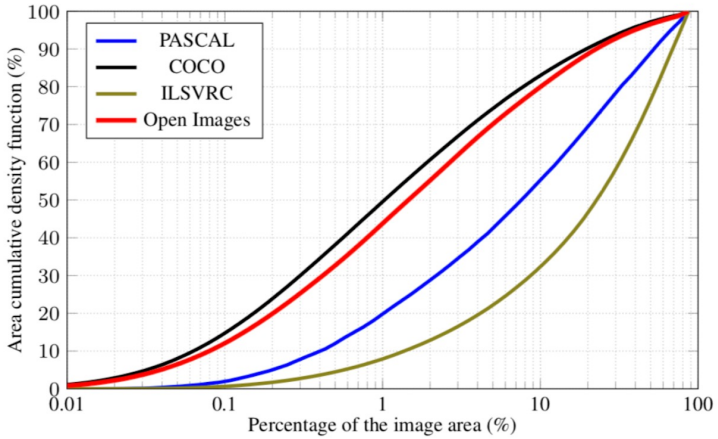


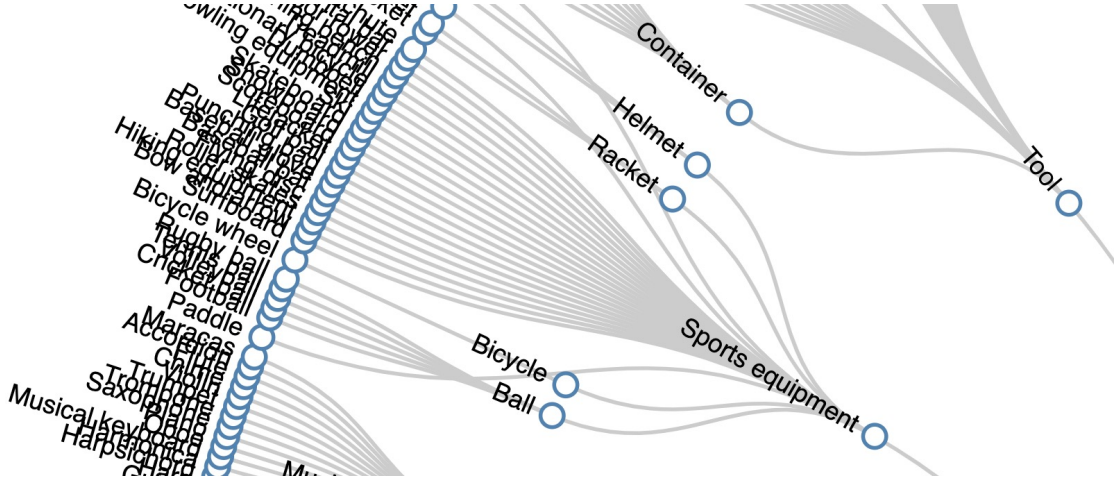
(e)

Open Images

- Provided up to 2020 (v6)
- 15m boxes on 600 categories
- 2,8m instance segmentations on 350 categories
- 36,5m image-level labels on 20k categories
- 391k relationship annotations of 329 relationships
- Extension - 478k crowdsourced images with 6k+ categories





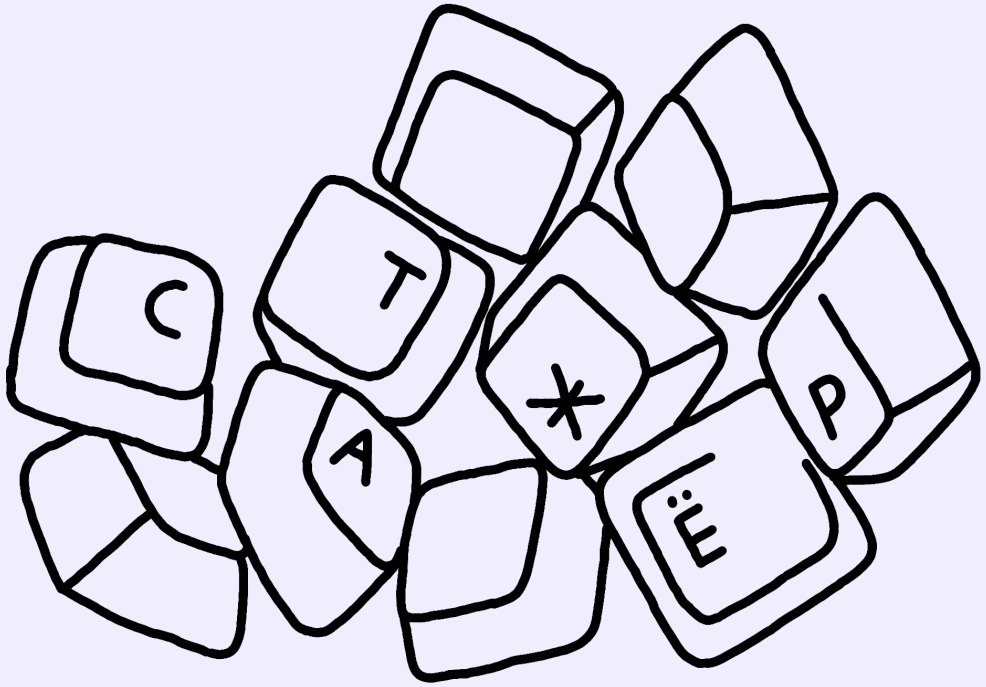


source: https://storage.googleapis.com/openimages/web/factsfigures.html

Practical notice

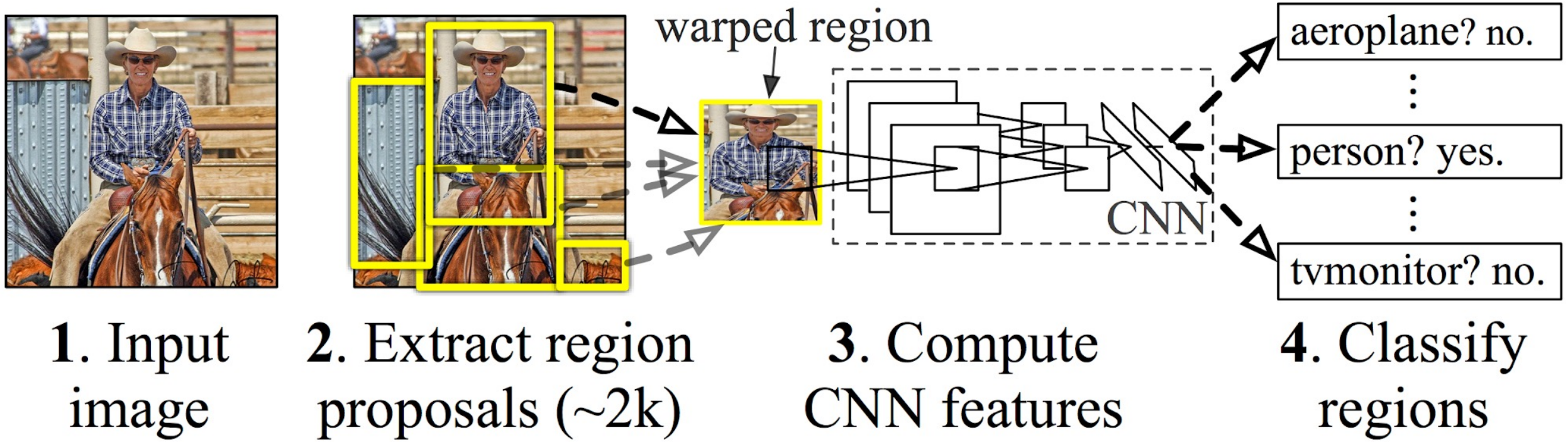
Mixup your small datasets with opensourse dataset to prevent overfitting

Detection

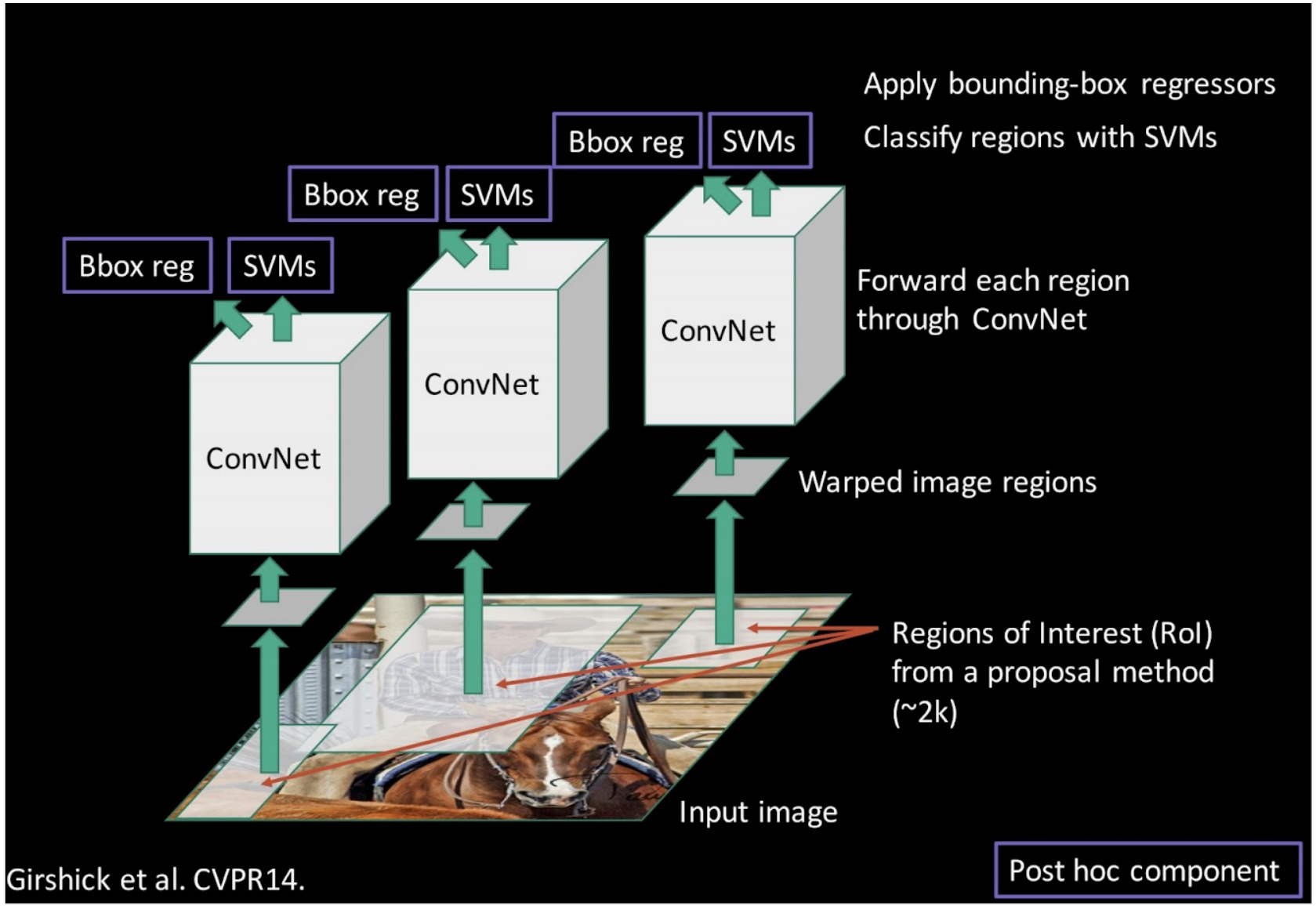


04

R-CNN: *Regions with CNN features*



Slide source: YOLO CVPR 2016 presentation

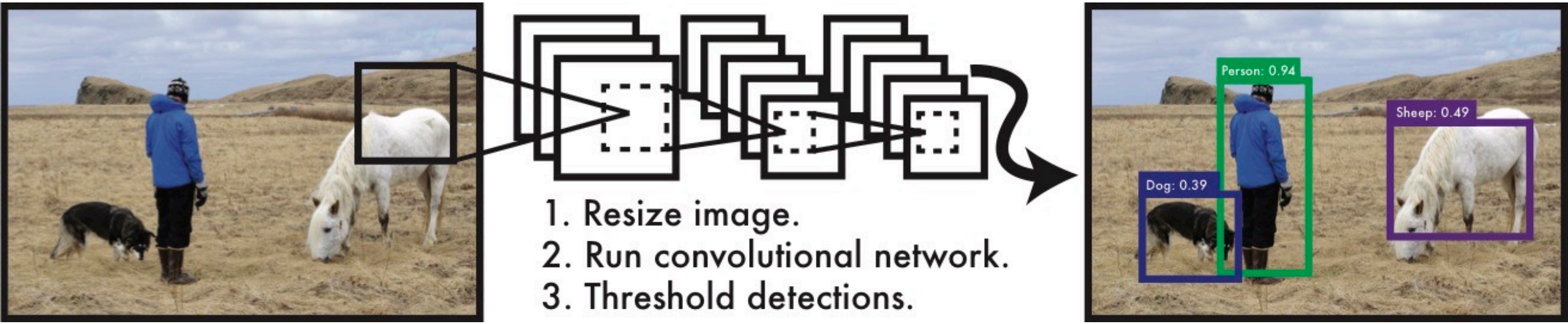


Girschick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

Slide credit: Ross Girschick

Slide source: YOLO CVPR 2016 presentation

YOLO: You Only Look Once

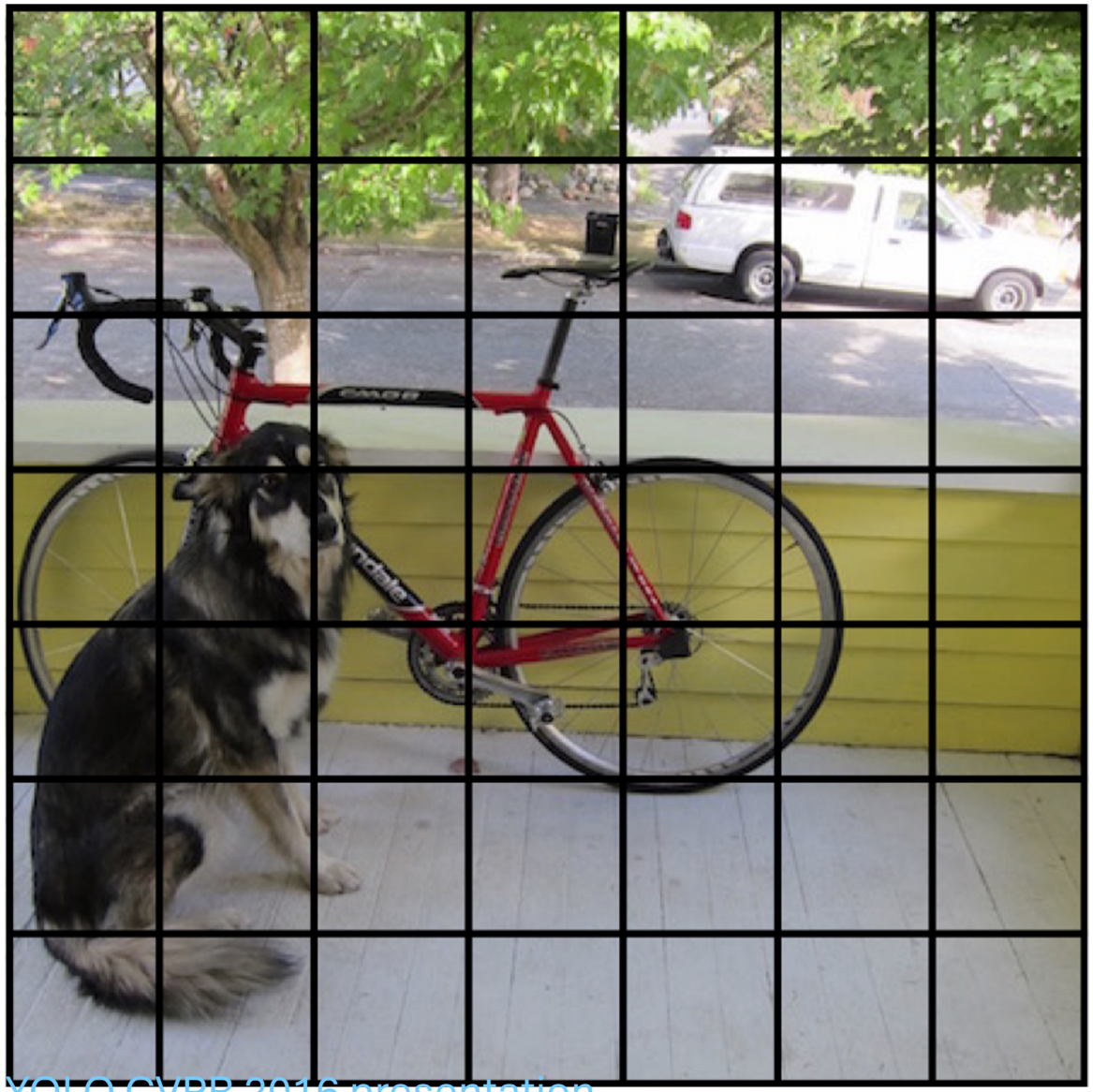


Slide source: [YOLO CVPR 2016 presentation]

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	69.0	45 FPS	22 ms/img

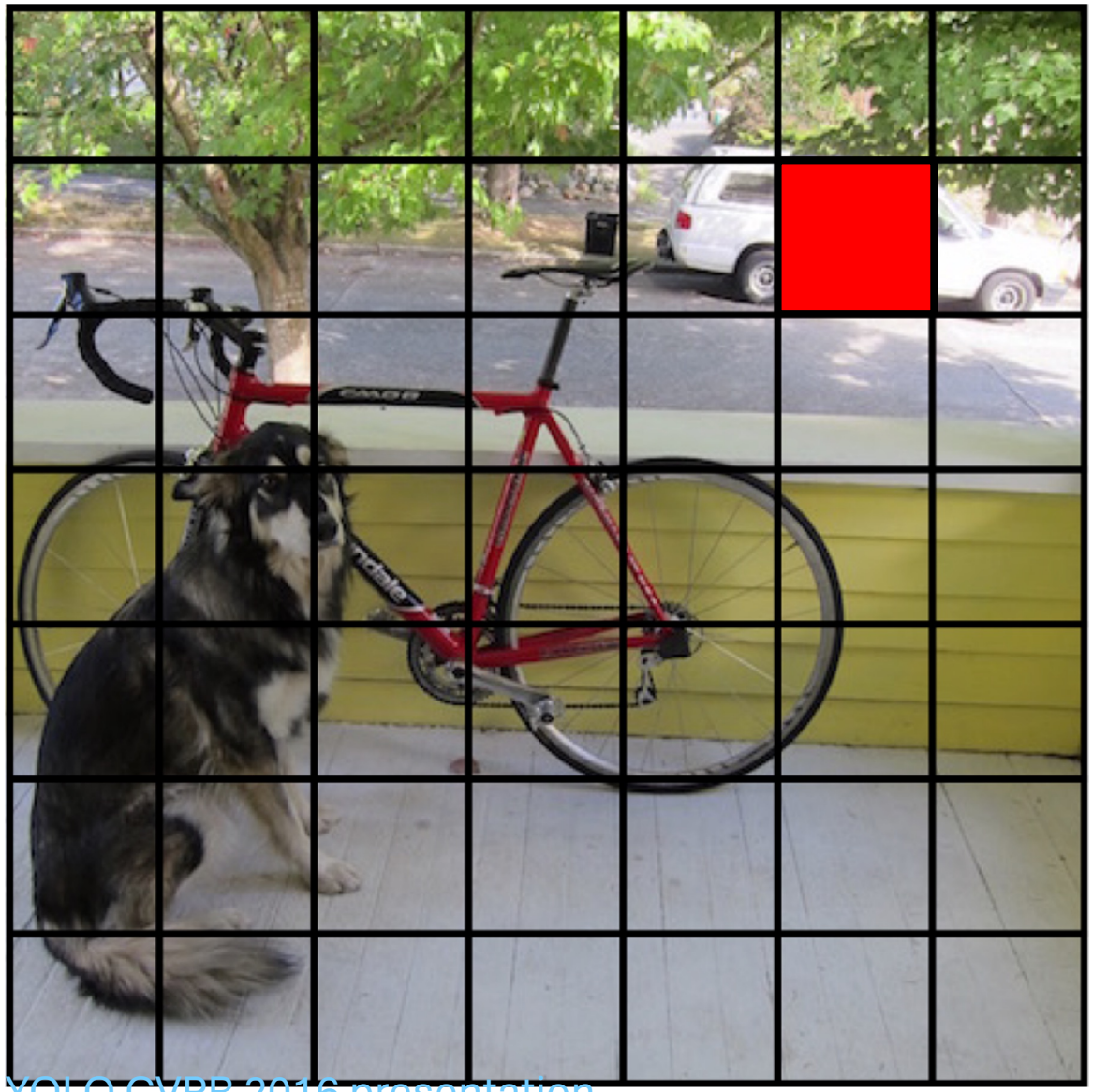
Slide source: YOLO CVPR 2016 presentation

We split the image into a grid



Slide source: YOLO CVPR 2016 presentation

Each cell predicts boxes and confidences: P(Object)

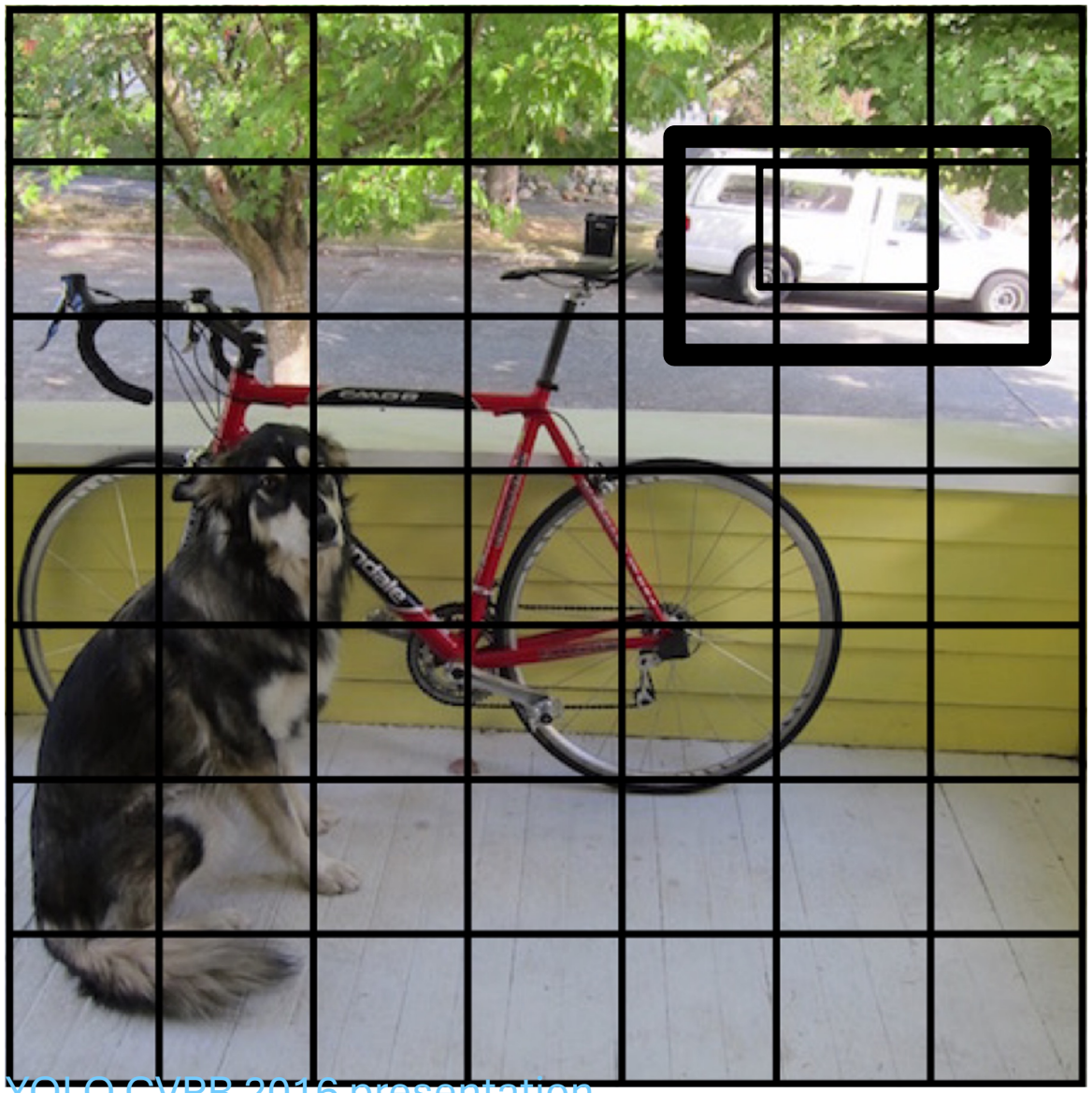


Slide source: YOLO CVPR 2016 presentation

Each cell predicts boxes and confidences: P(Object)

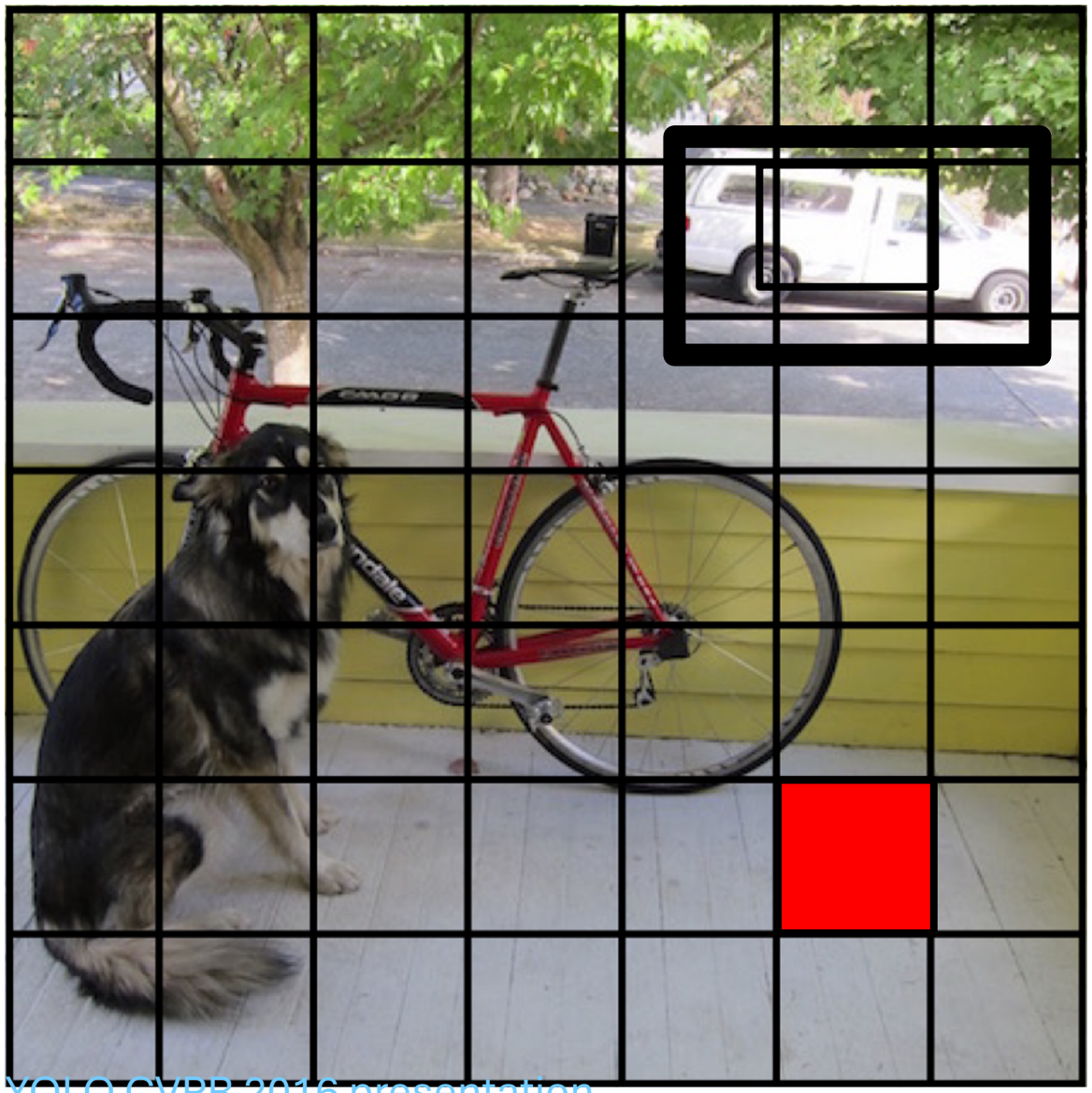
Slide source: YOLO CVPR 2016 presentation

Each cell predicts boxes and confidences: P(Object)



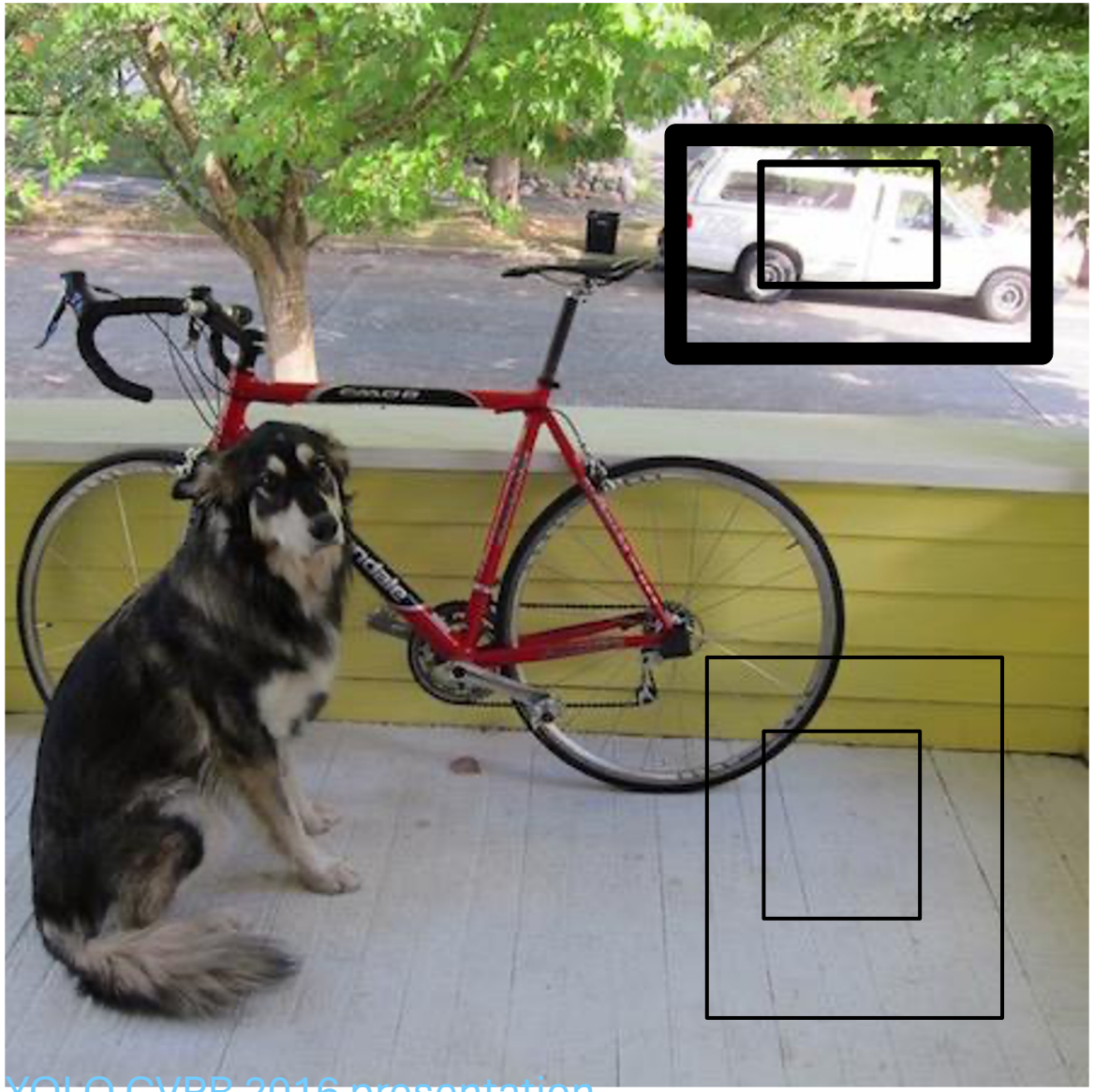
Slide source: YOLO CVPR 2016 presentation

Each cell predicts boxes and confidences: P(Object)



Slide source: YOLO CVPR 2016 presentation

Each cell predicts boxes and confidences: P(Object)

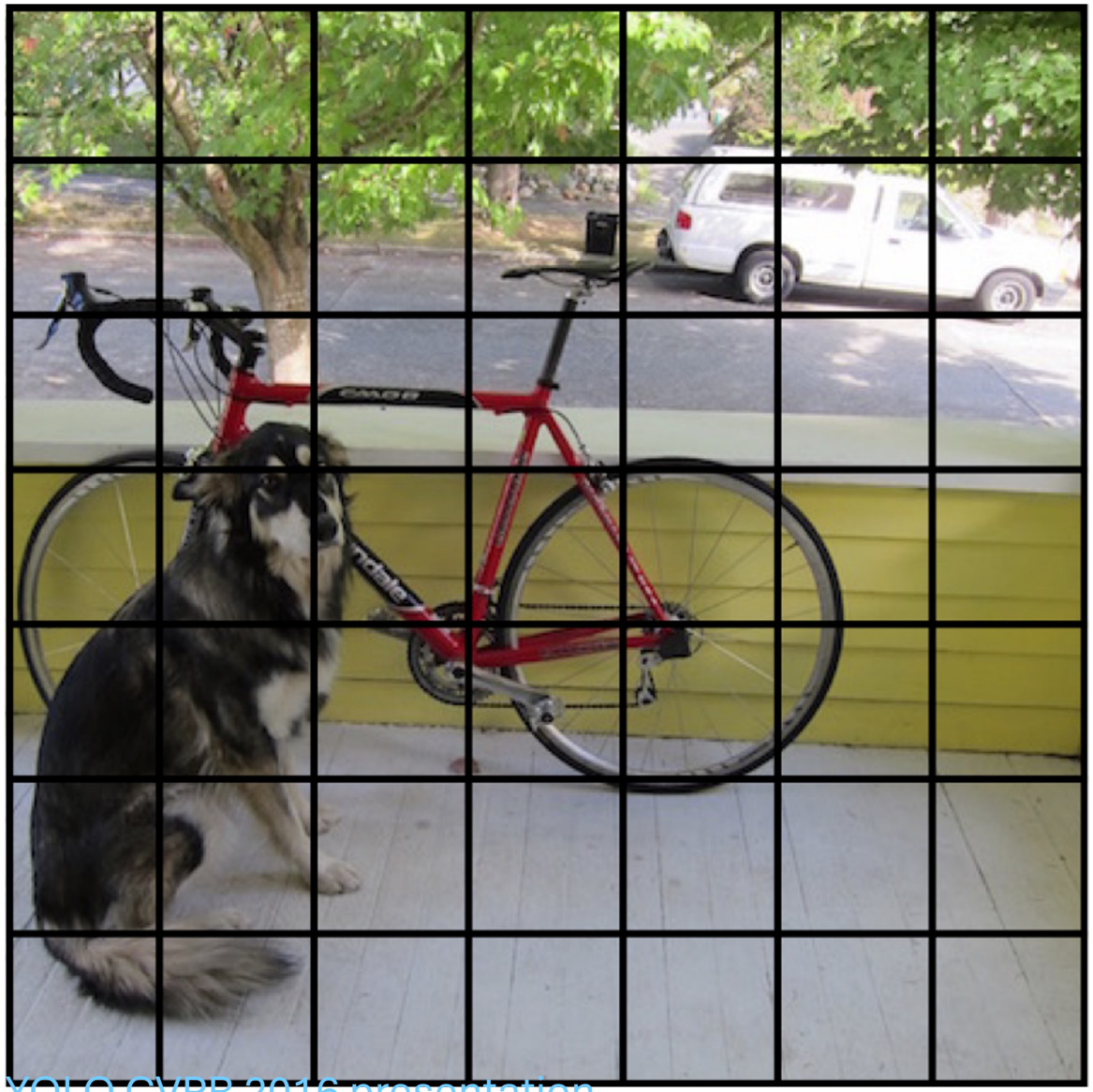


Slide source: YOLO CVPR 2016 presentation

Each cell predicts boxes and confidences: P(Object)

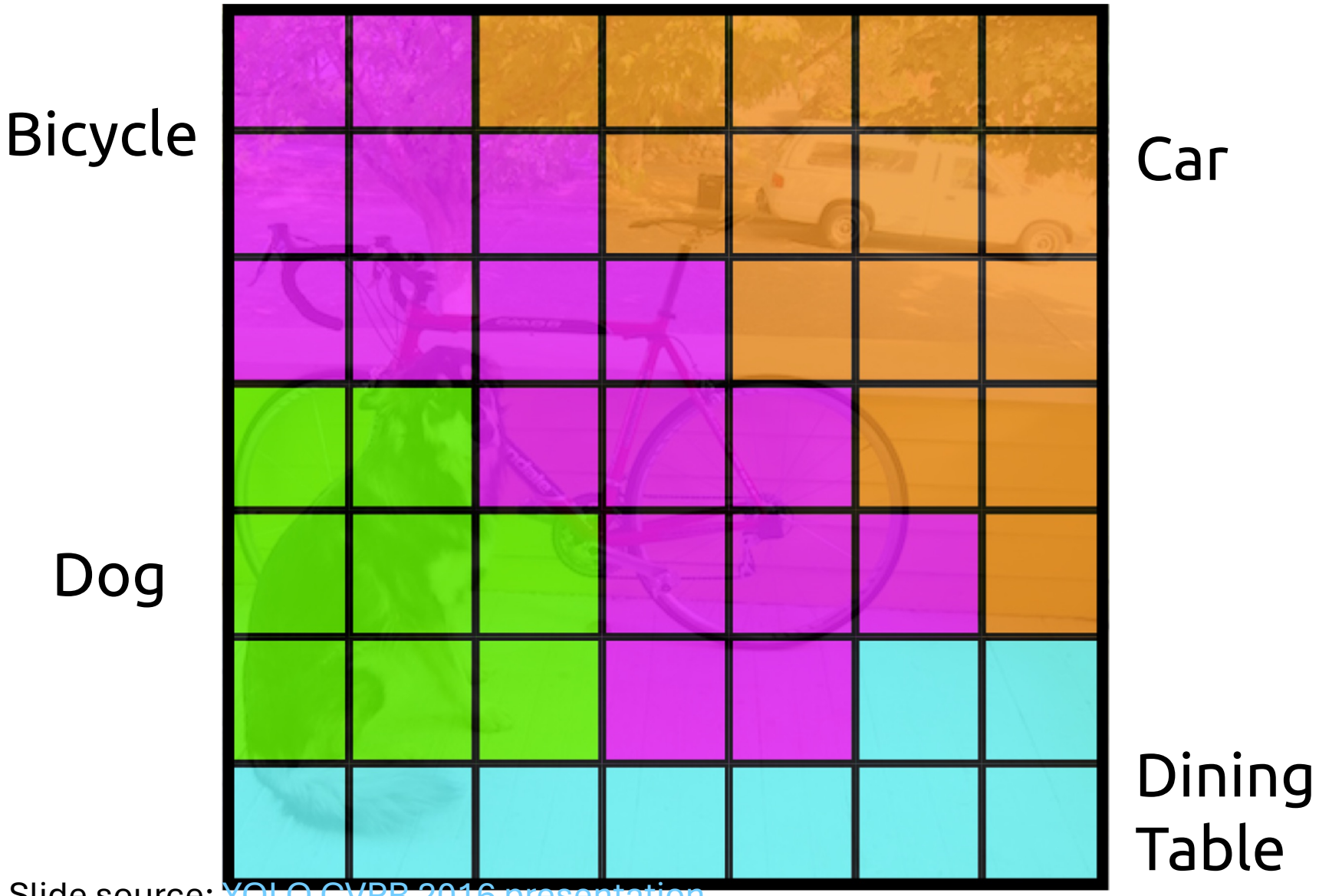
Slide source: YOLO CVPR 2016 presentation

Each cell also predicts a class probability.



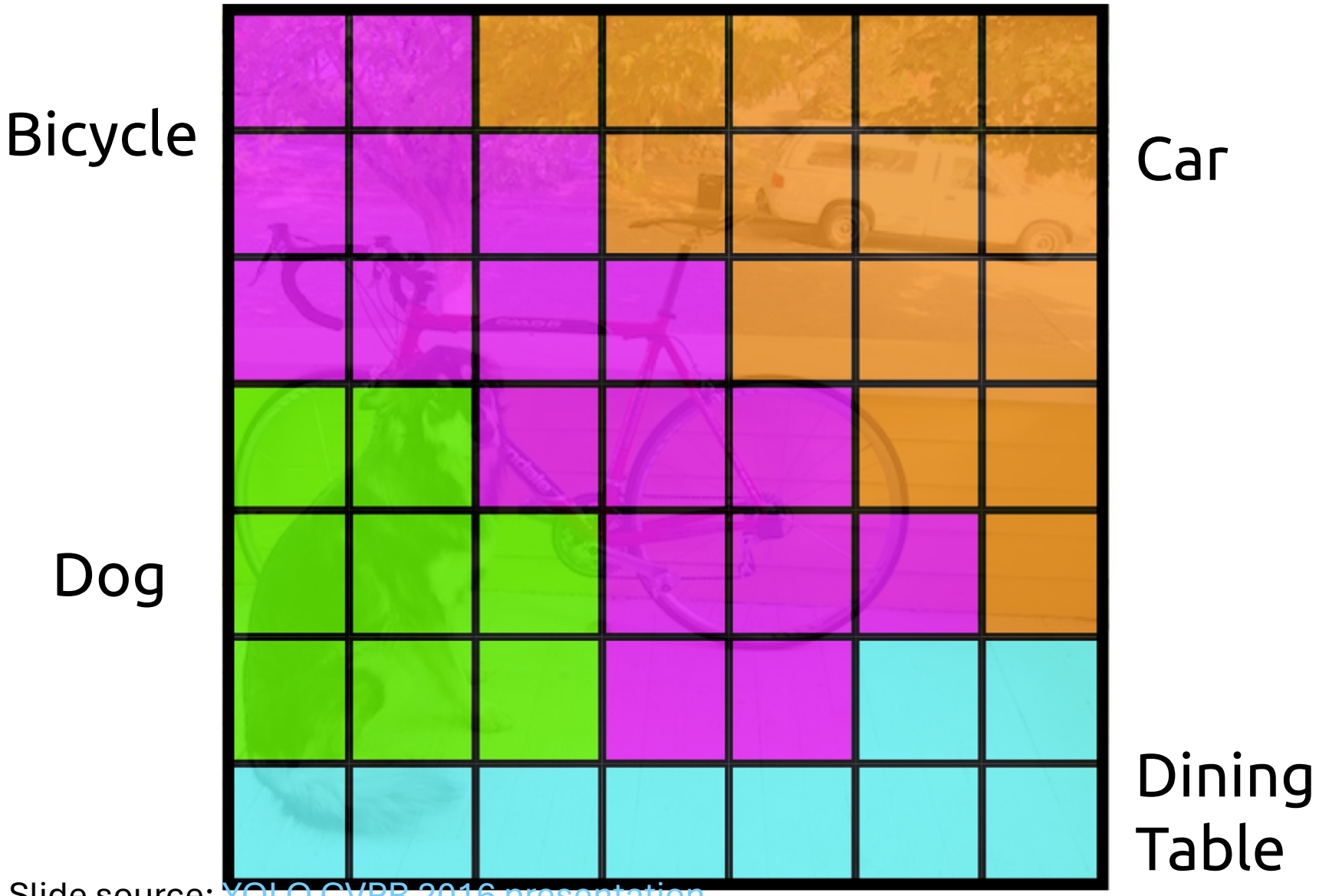
Slide source: YOLO CVPR 2016 presentation

Each cell also predicts a class probability.



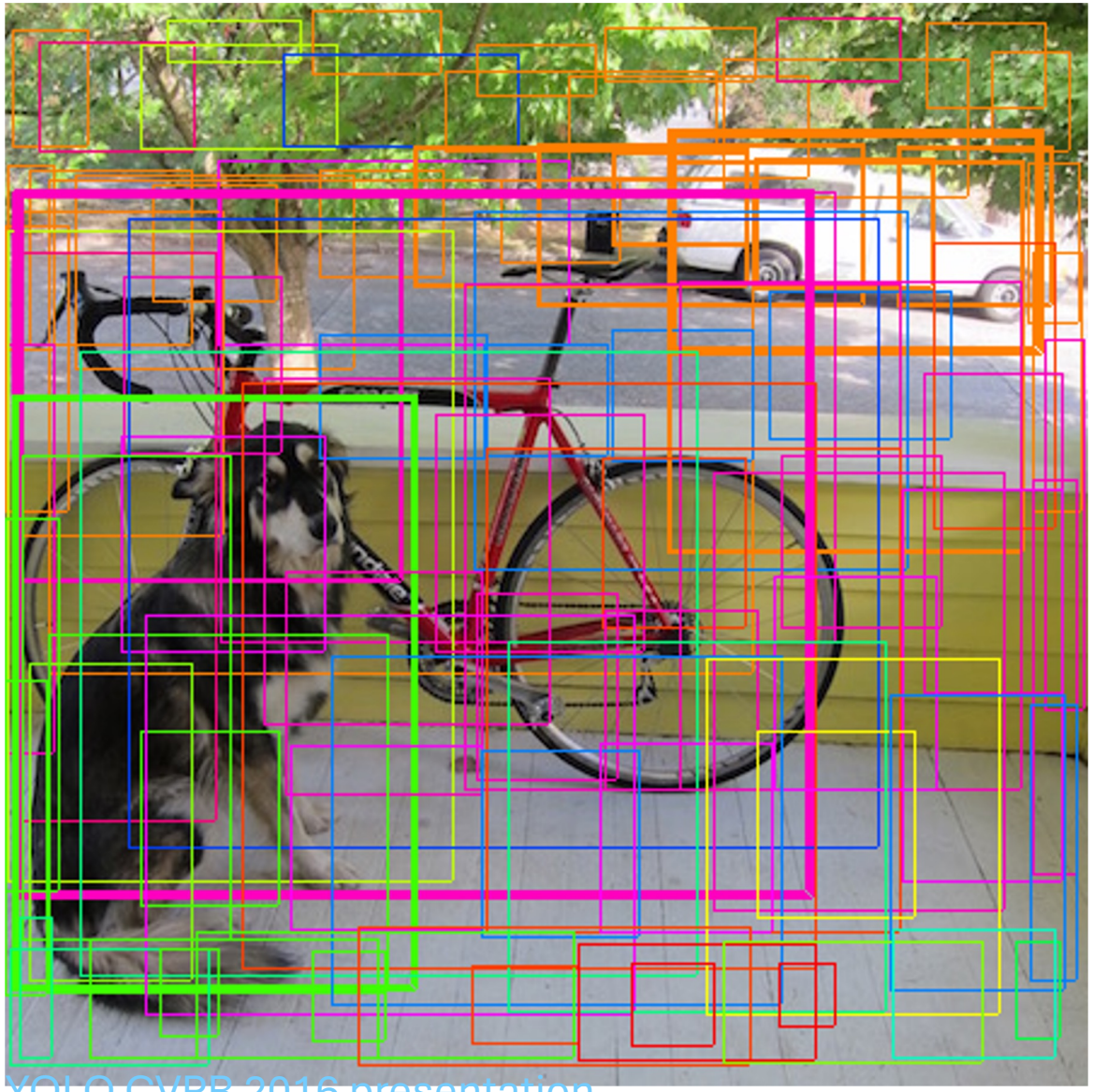
Slide source: YOLO CVPR 2016 presentation

Conditioned on object: P(Car | Object)



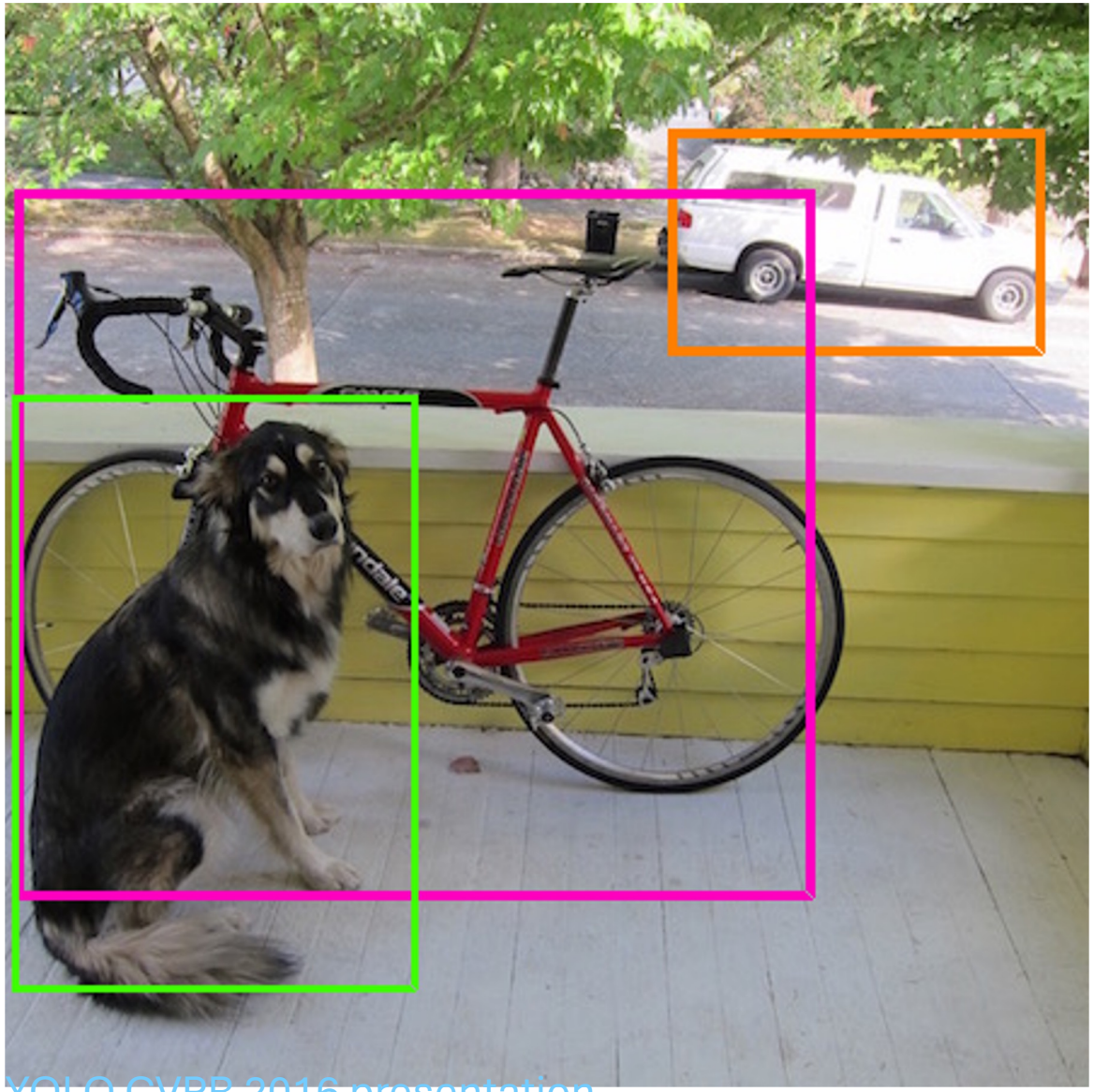
Slide source: YOLO CVPR 2016 presentation

Then we combine the box and class predictions.



Slide source: YOLO CVPR 2016 presentation

Finally we do NMS and threshold detections

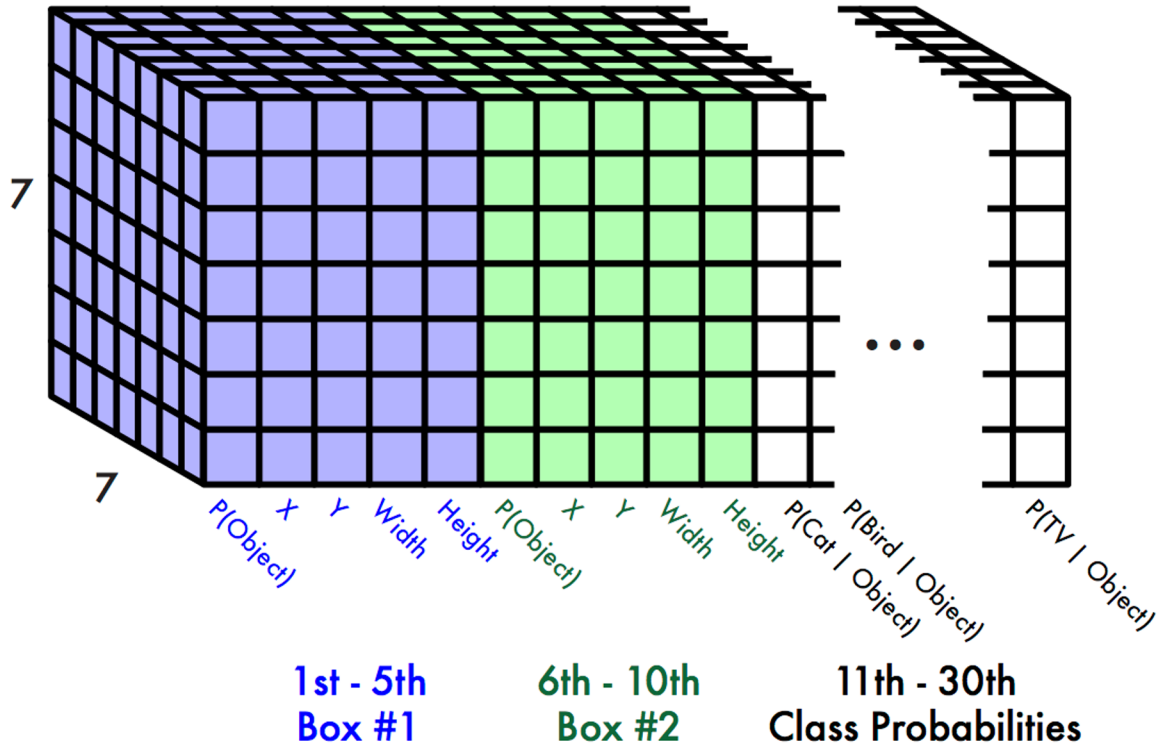


Slide source: YOLO CVPR 2016 presentation

This parameterization fixes the output size

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities



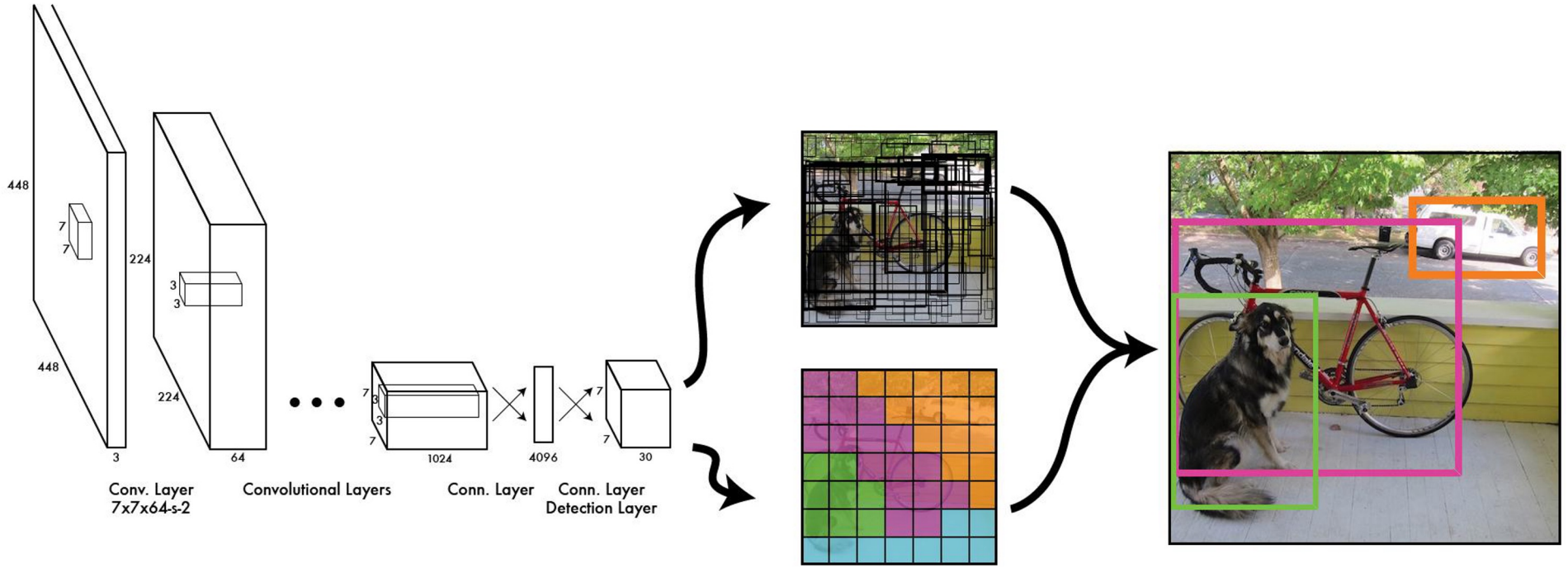
For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

Slide source: YOLO CVPR 2016 presentation

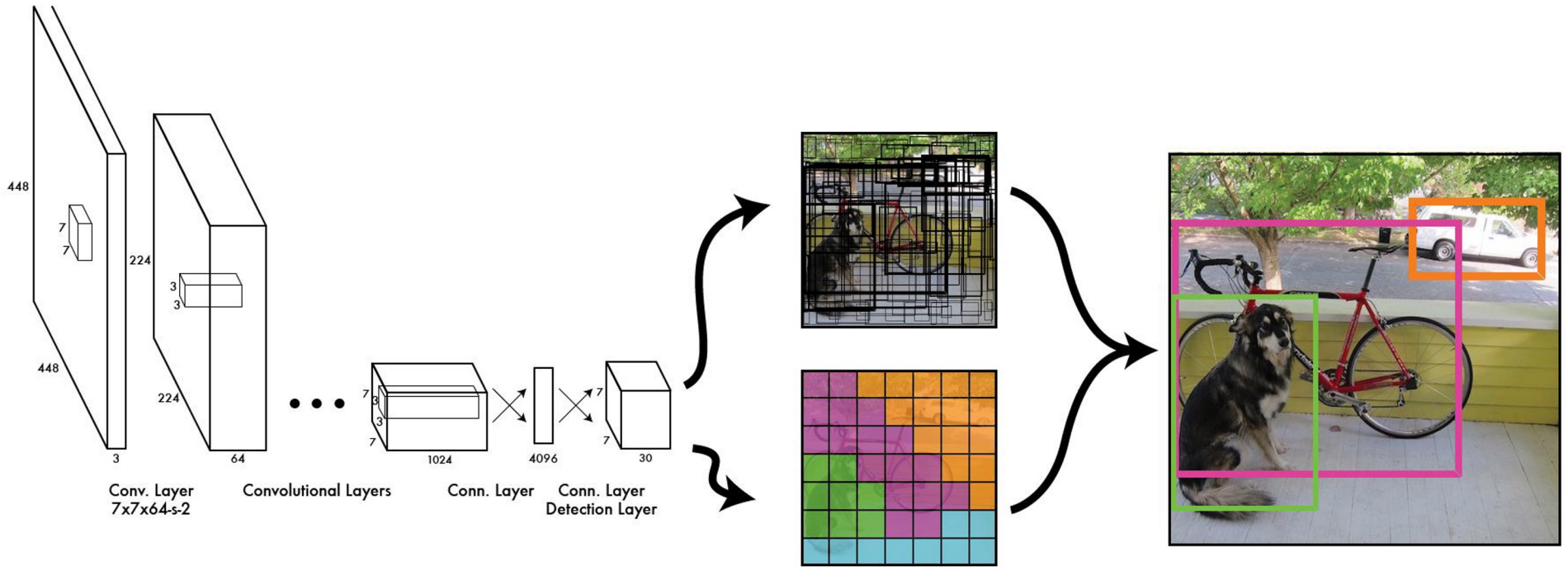
Thus we can train one neural network to be a whole detection pipeline



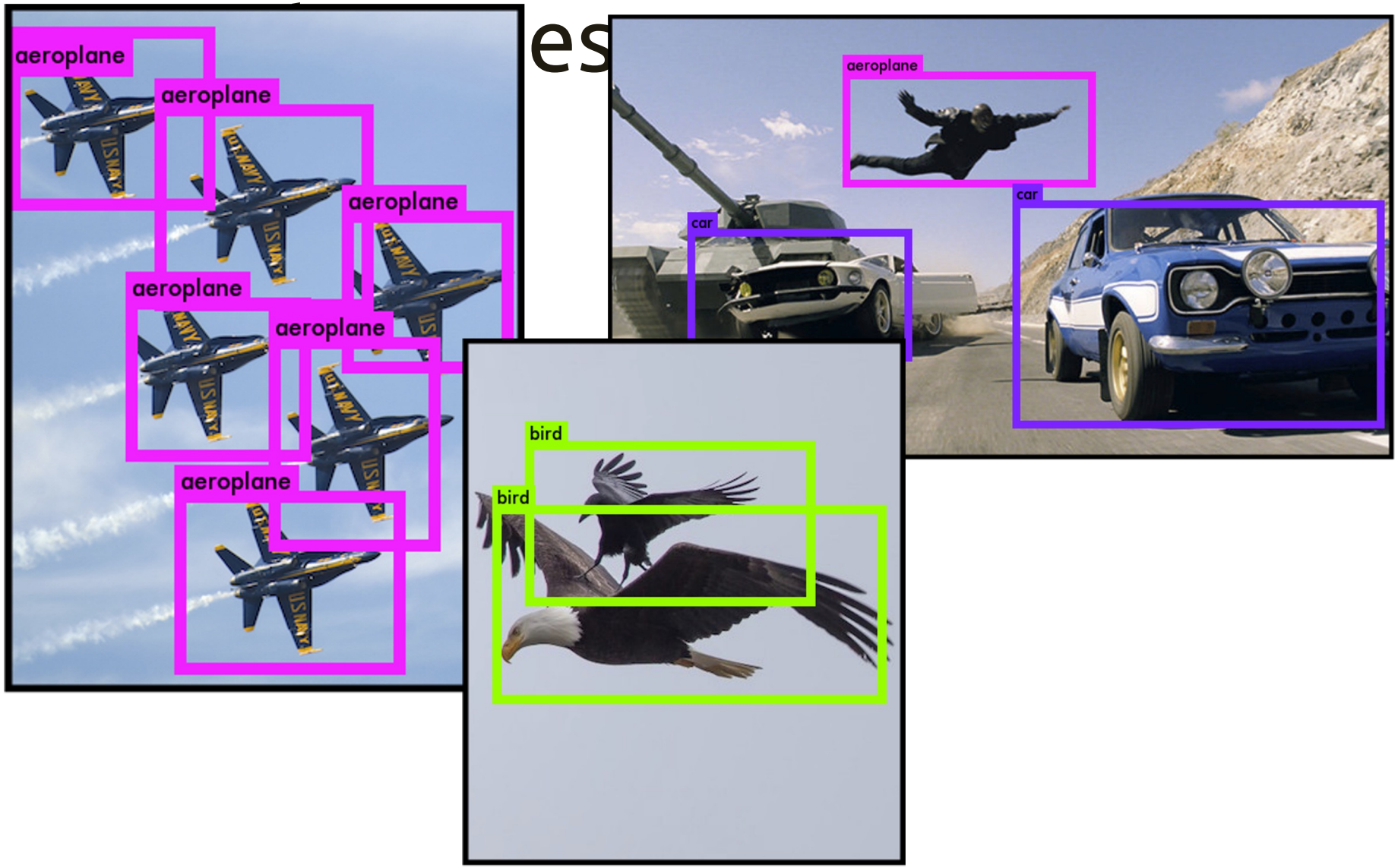
Slide source: YOLO CVPR 2016 presentation

We train with standard tricks:

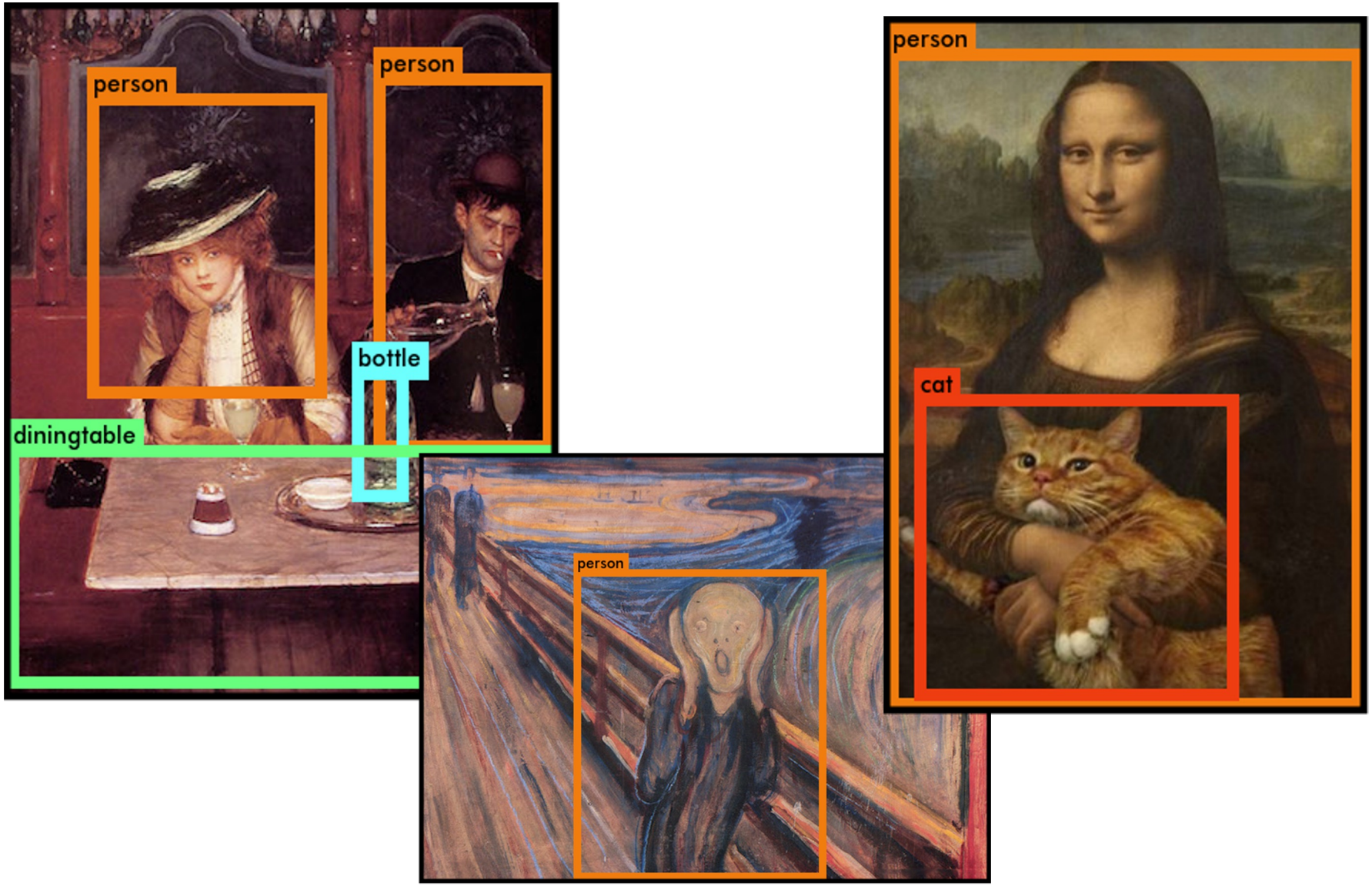
- Pretraining on Imagenet
- SGD with decreasing learning rate
- Extensive data augmentation
- For details, see the paper



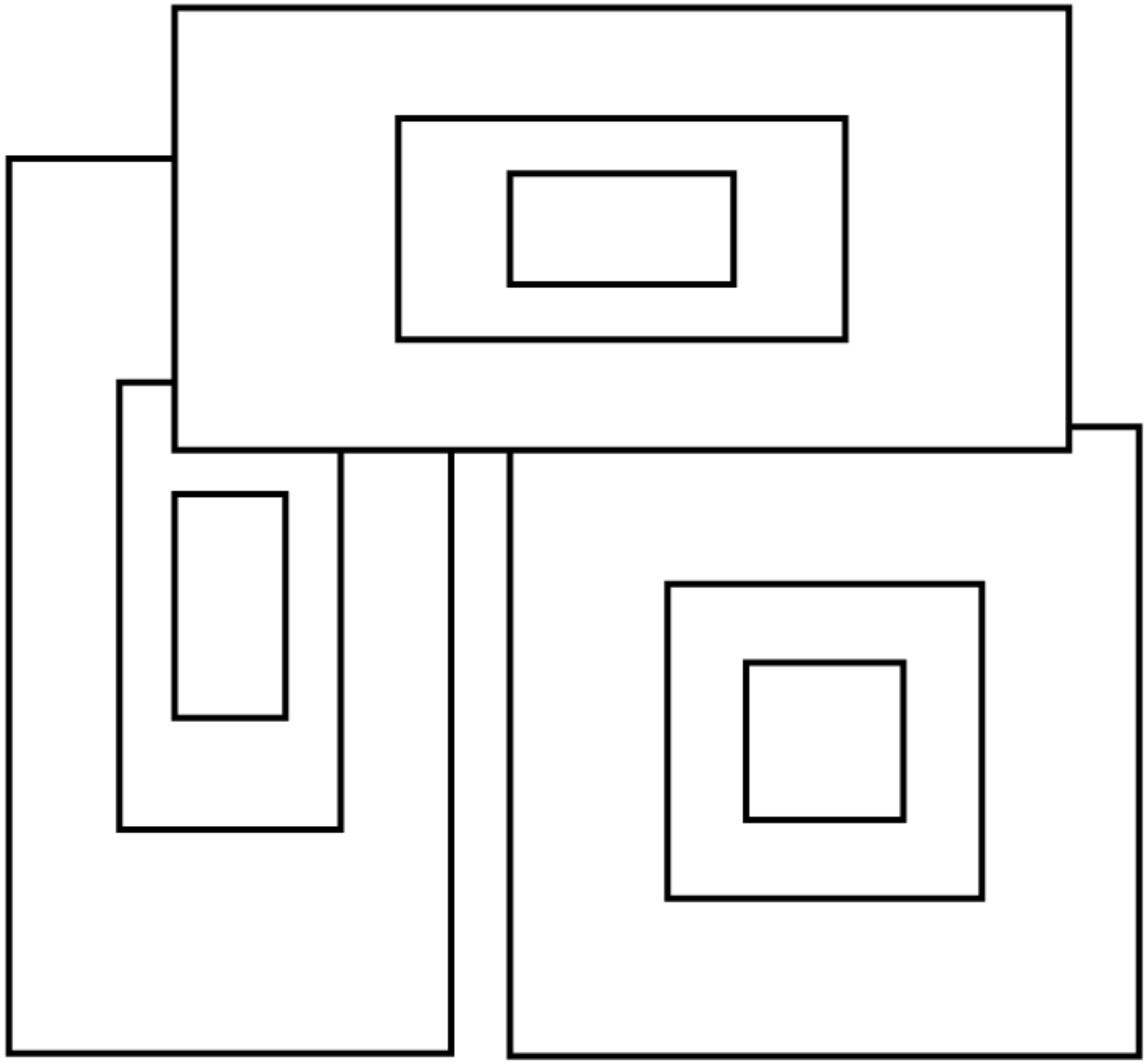
YOLO works across a variety of es



It also generalizes well to new domains (like art)



YOLOv2 – add ”Anchor boxes”



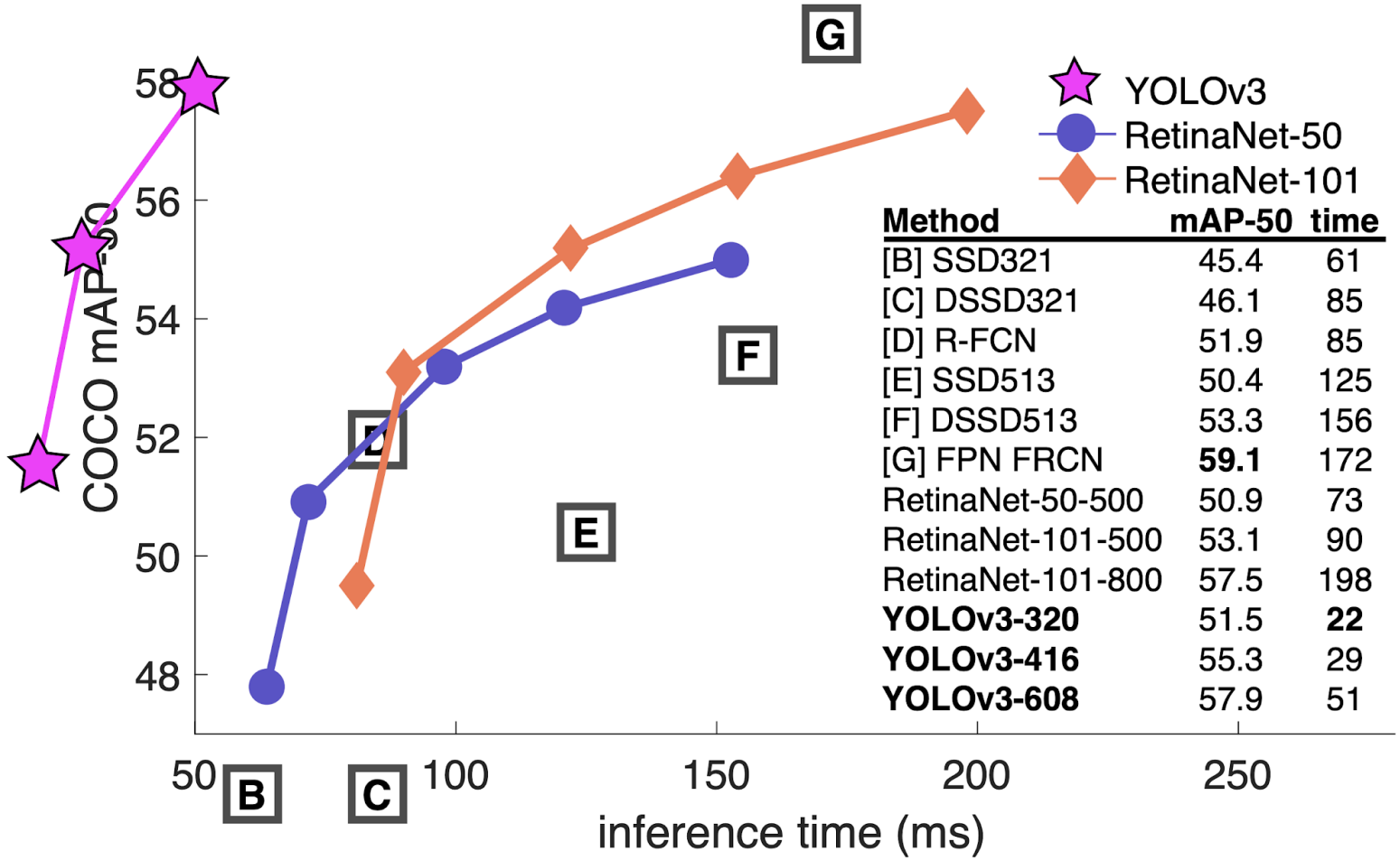
Sizes:
- 128
- 256
- 512

Ratios:
- 1x1
- 1x2
- 2x1

YOLOv3

- Bounding Box Prediction
- Class Prediction
- Predictions Across Scales
- Feature Extractor

All articles published on [Joseph's website]



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

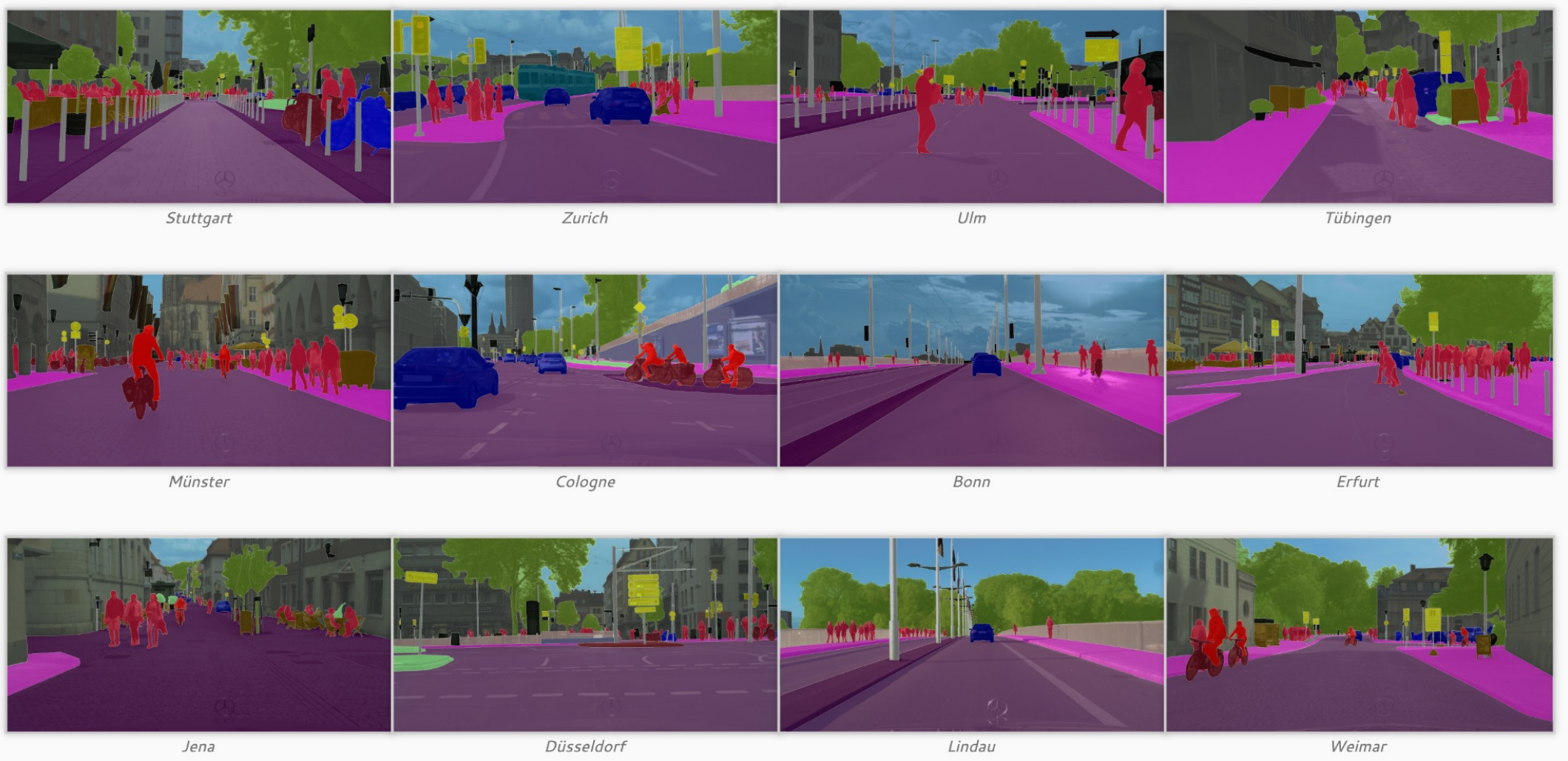
Segmentation

05

COCO Panoptic Segmentation

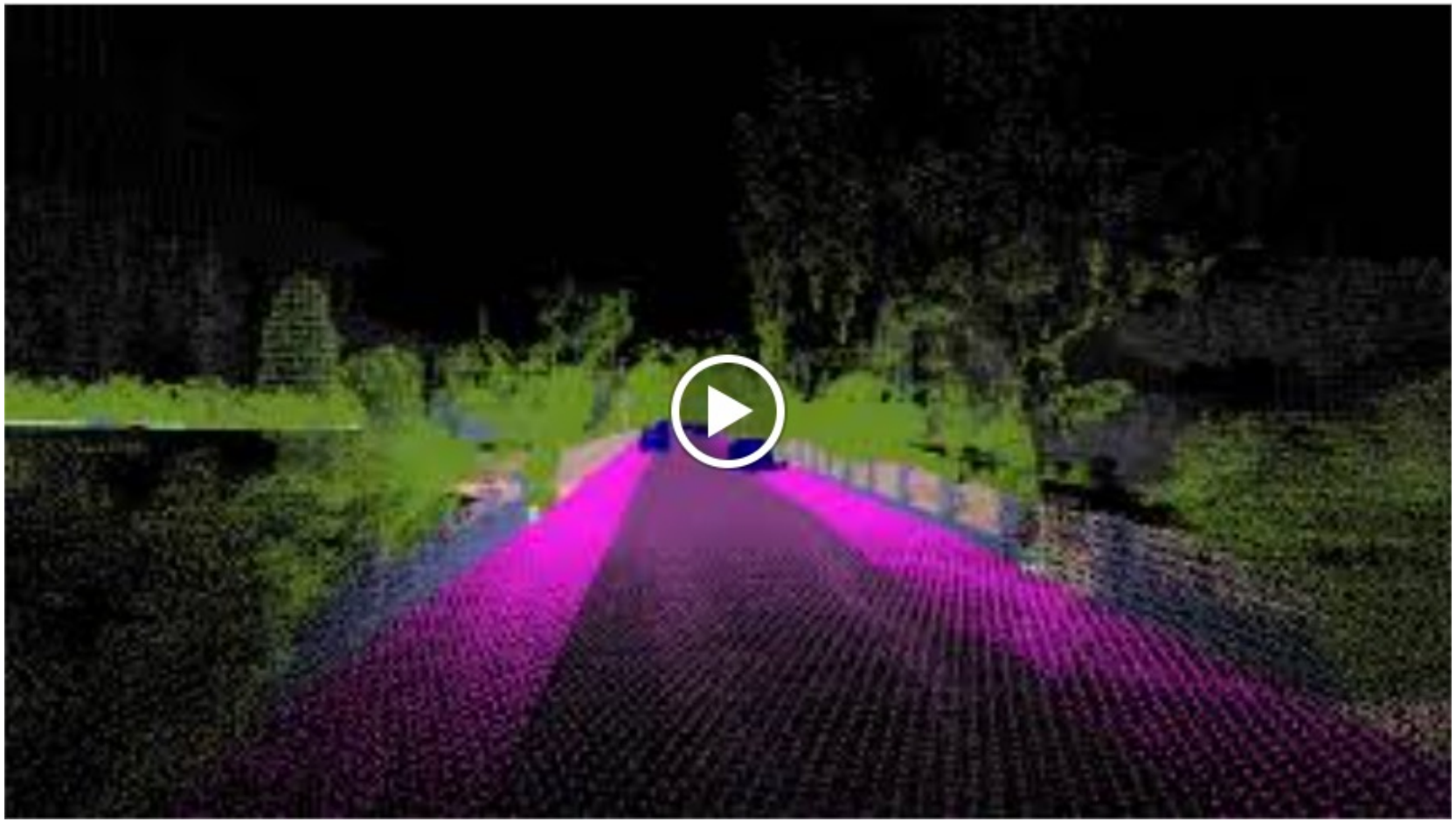
https://cocodataset.org/#panoptic-2018

Cityscapes



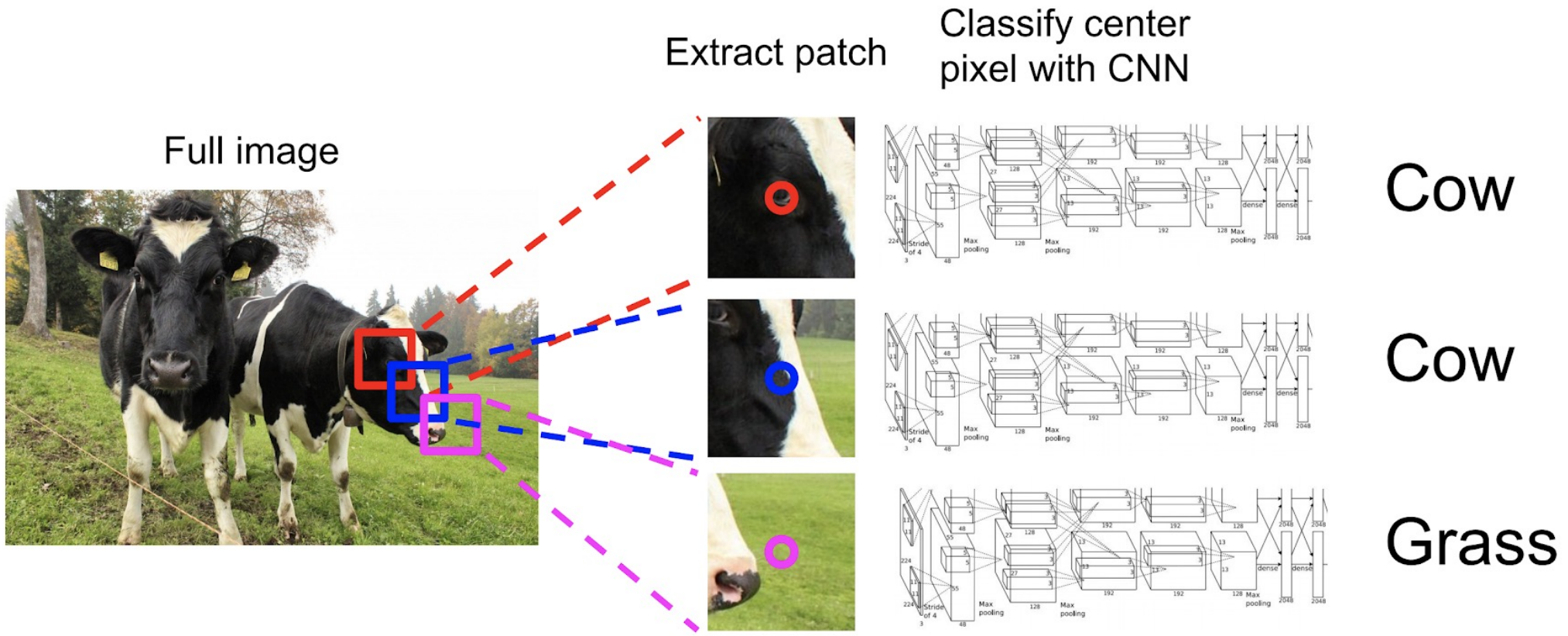
https://www.cityscapes-dataset.com/

KITTI



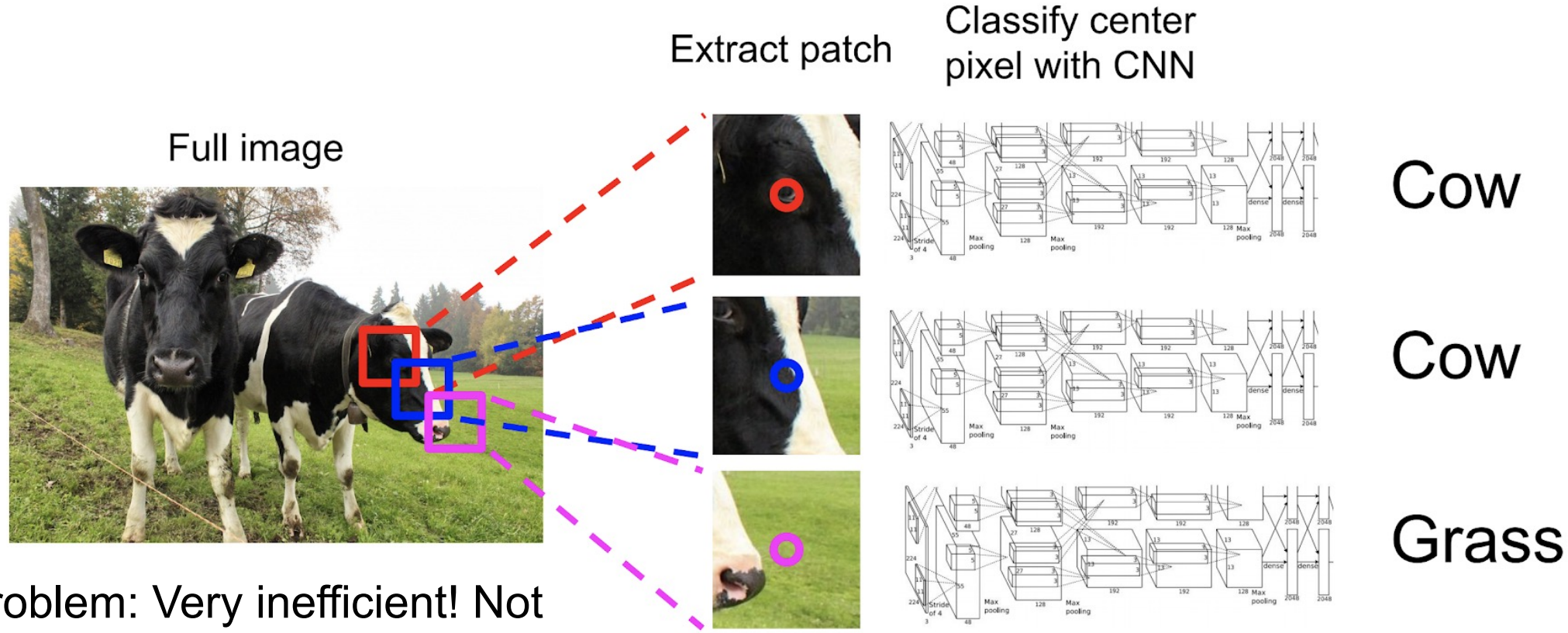
https://www.cvlibs.net/datasets/kitti-360/

Naive solution



Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

Naive solution

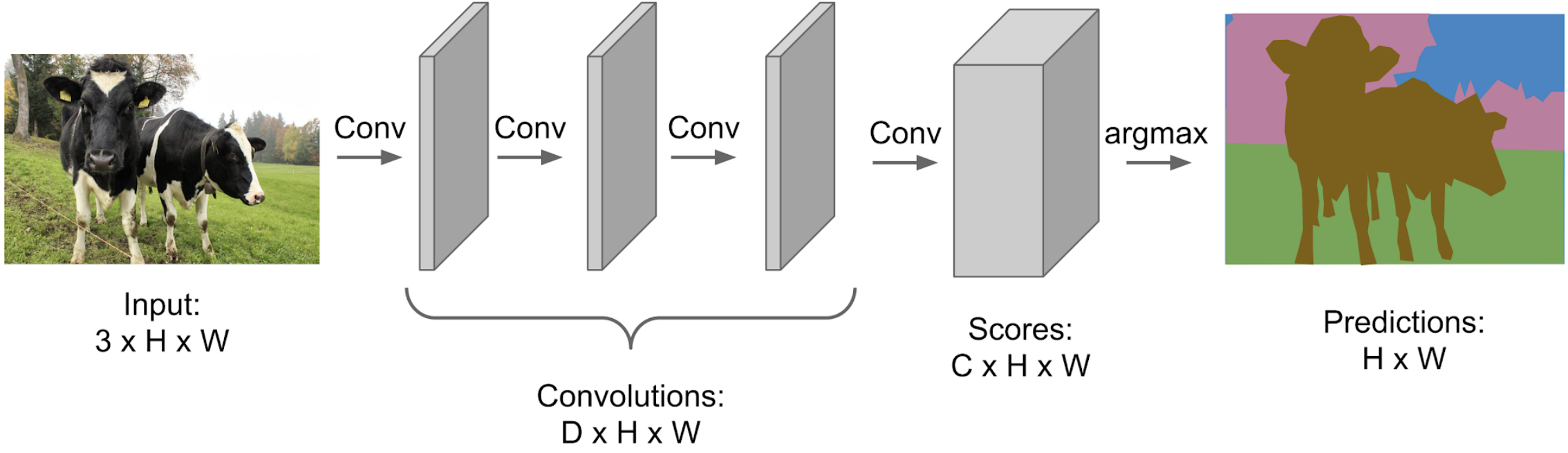


Problem: Very inefficient! Not reusing shared features between overlapping patches (same as R-CNN)

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

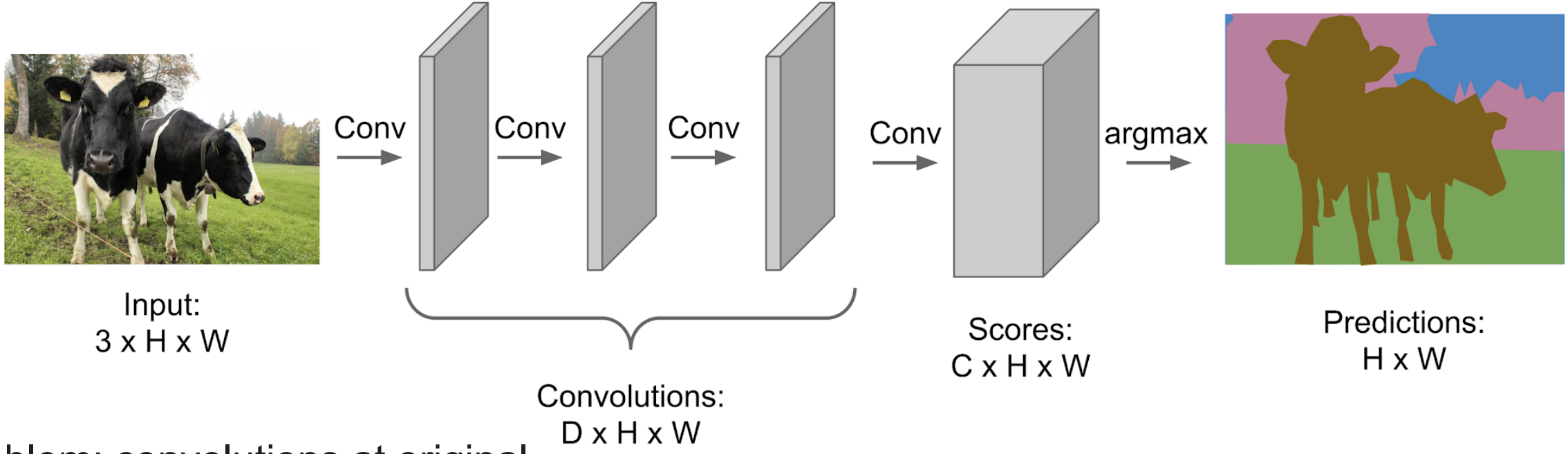
Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

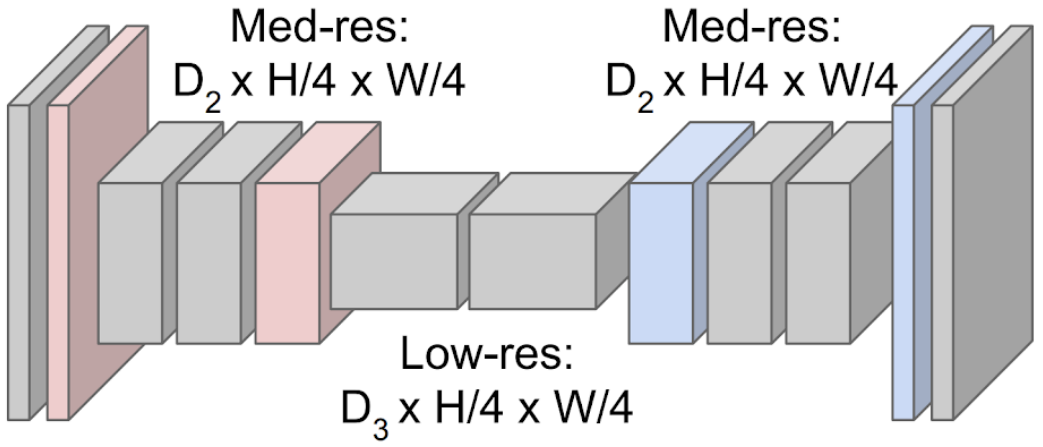


Problem: convolutions at original image resolution will be very expensive (remember VGG architecture)

Downsampling and Upsampling

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



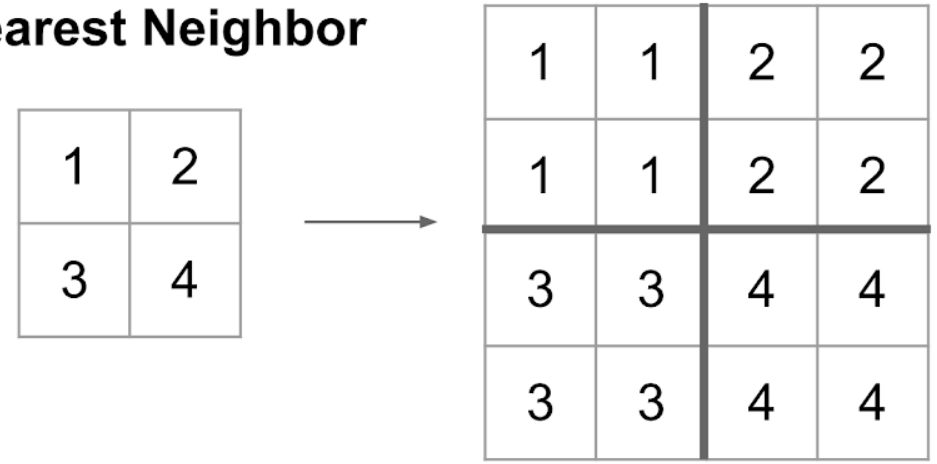
High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Simple unpooling

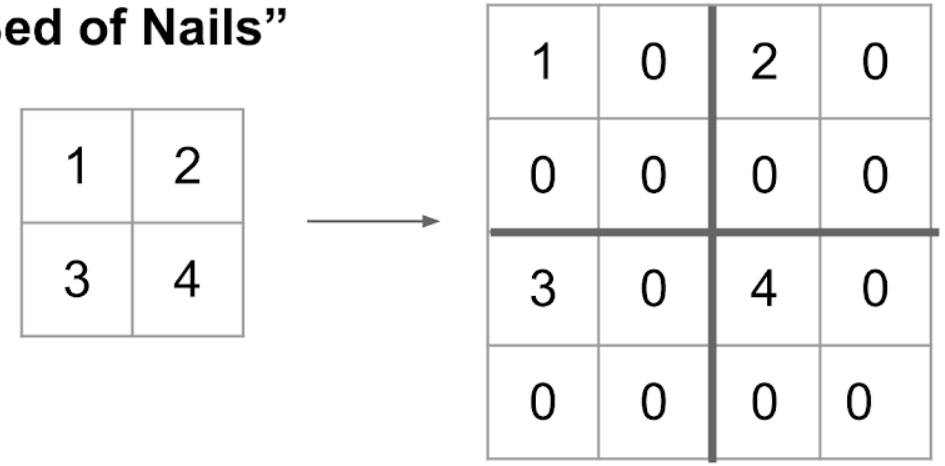
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

“Bed of Nails”



Input: 2 x 2

Output: 4 x 4

Max Unpooling

Max Pooling
Remember which element was max!

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

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling
Use positions from pooling layer

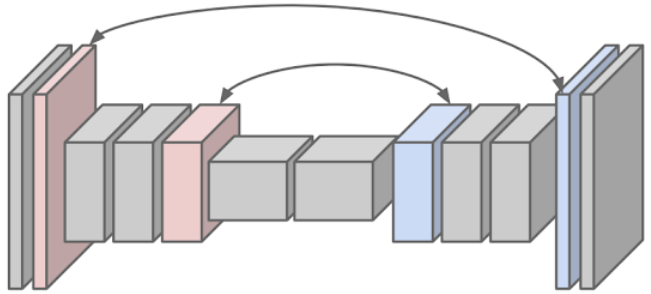
1	2
3	4

Input: 2 x 2

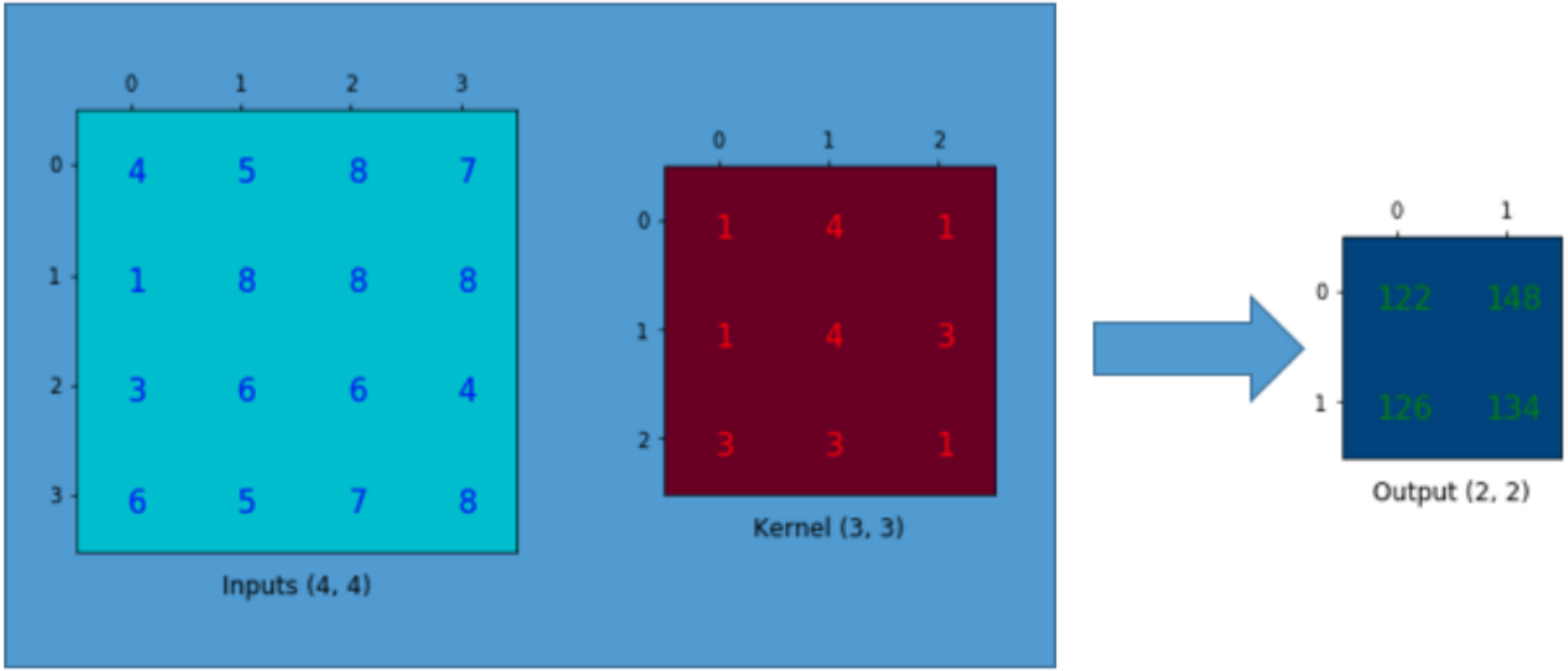
0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

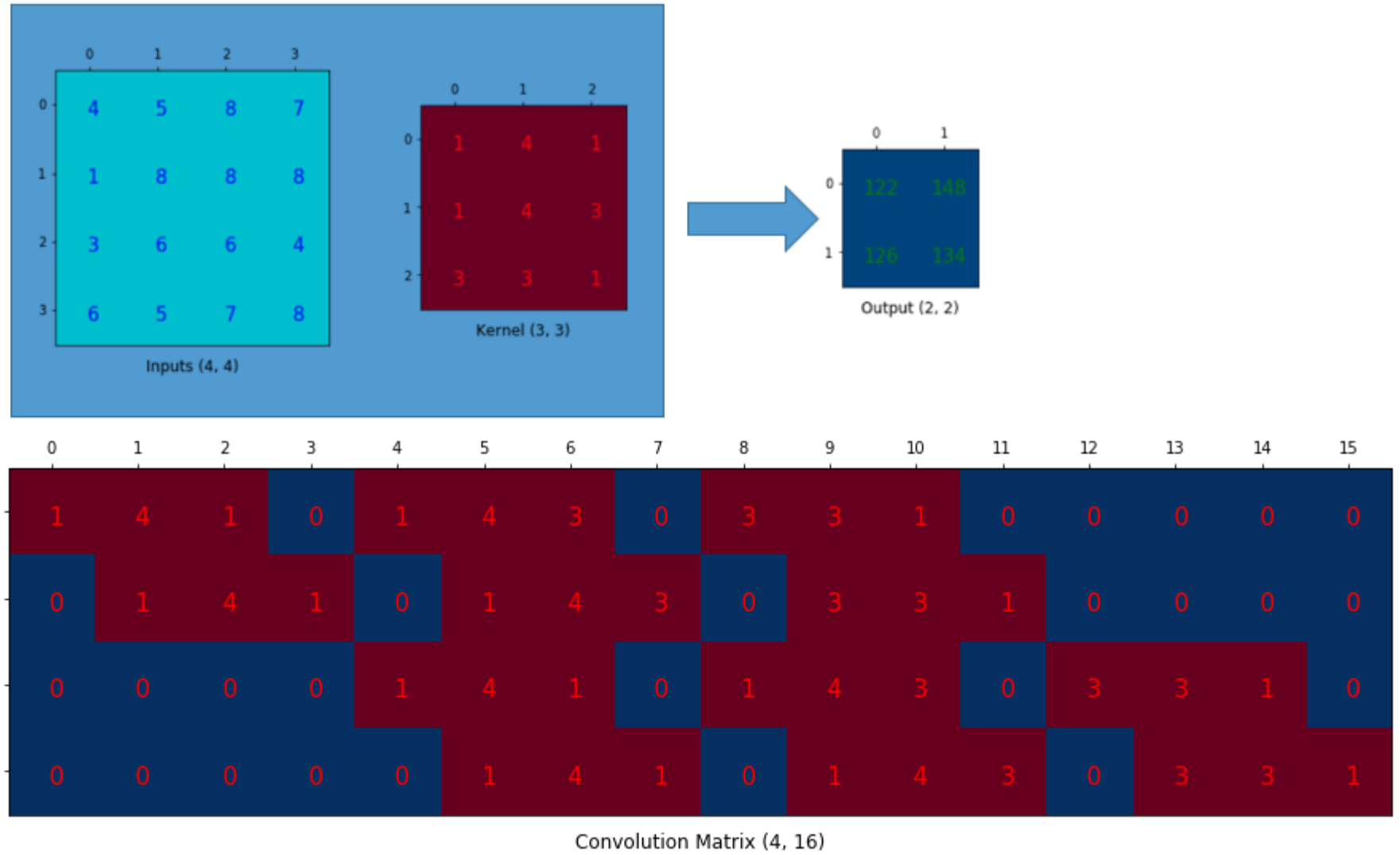


Regular convolution

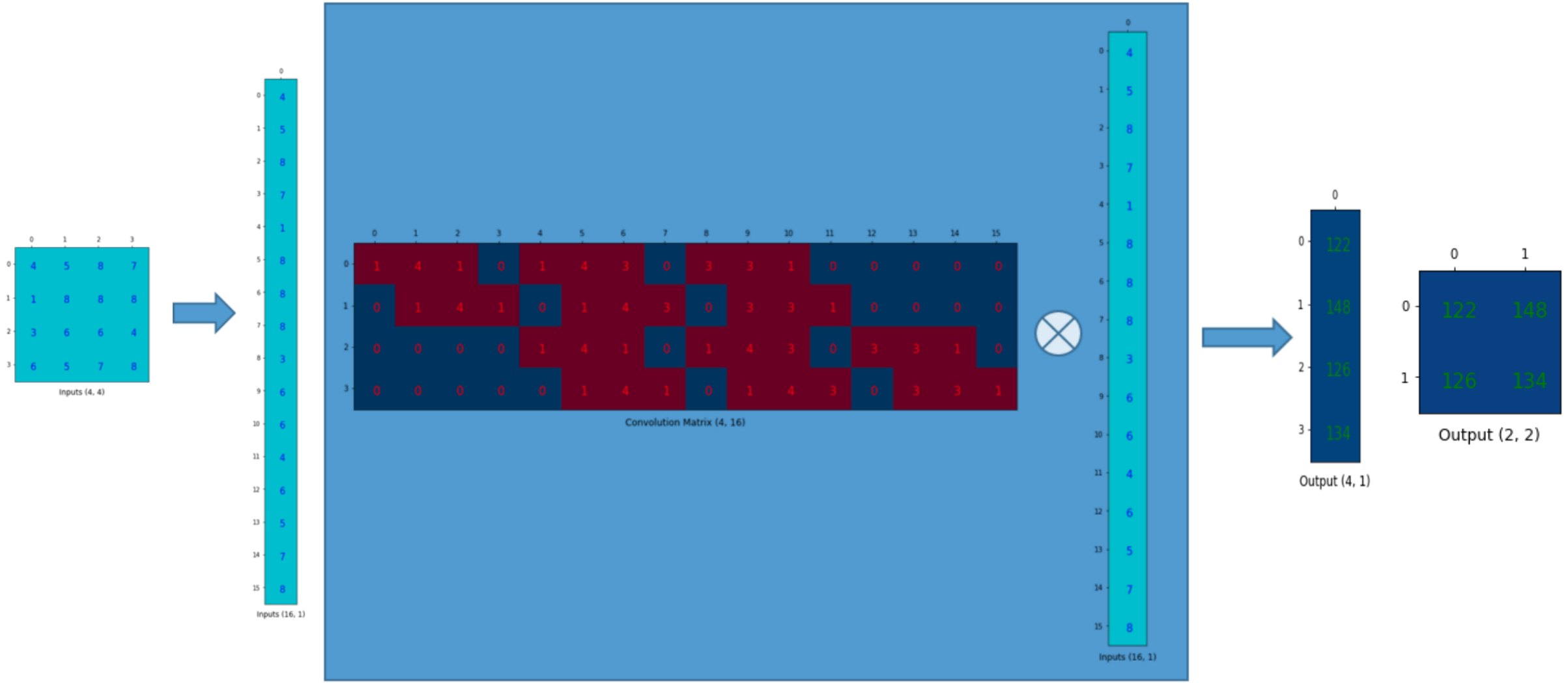


Images credentials

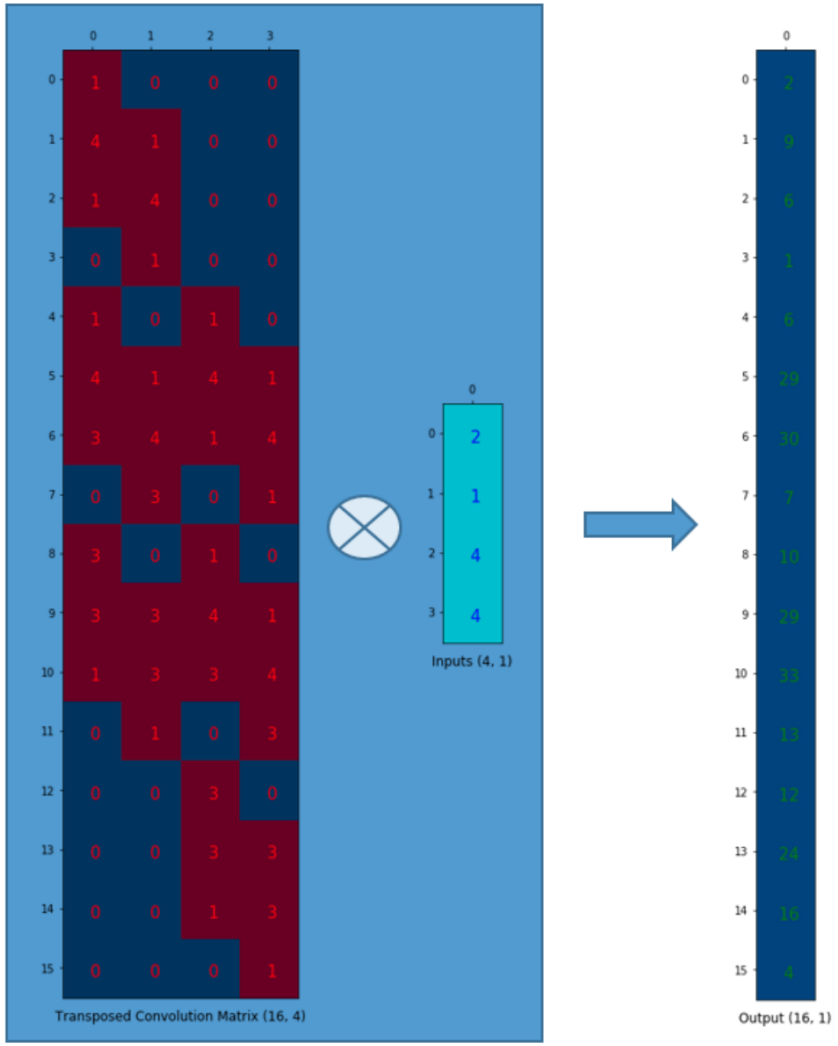
Regular convolution: matrix form

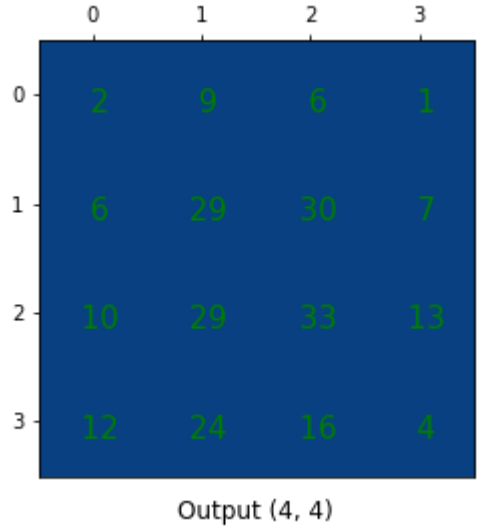


Regular convolution: matrix form



Transposed convolution matrix





Convolution as Matrix Multiplication (1D)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

Terms

Some researchers denote Transposed convolution with term “Deconvolution”. However:

deconvolution is an algorithm-based process used to reverse the effects of convolution on recorded data [Wikipedia]

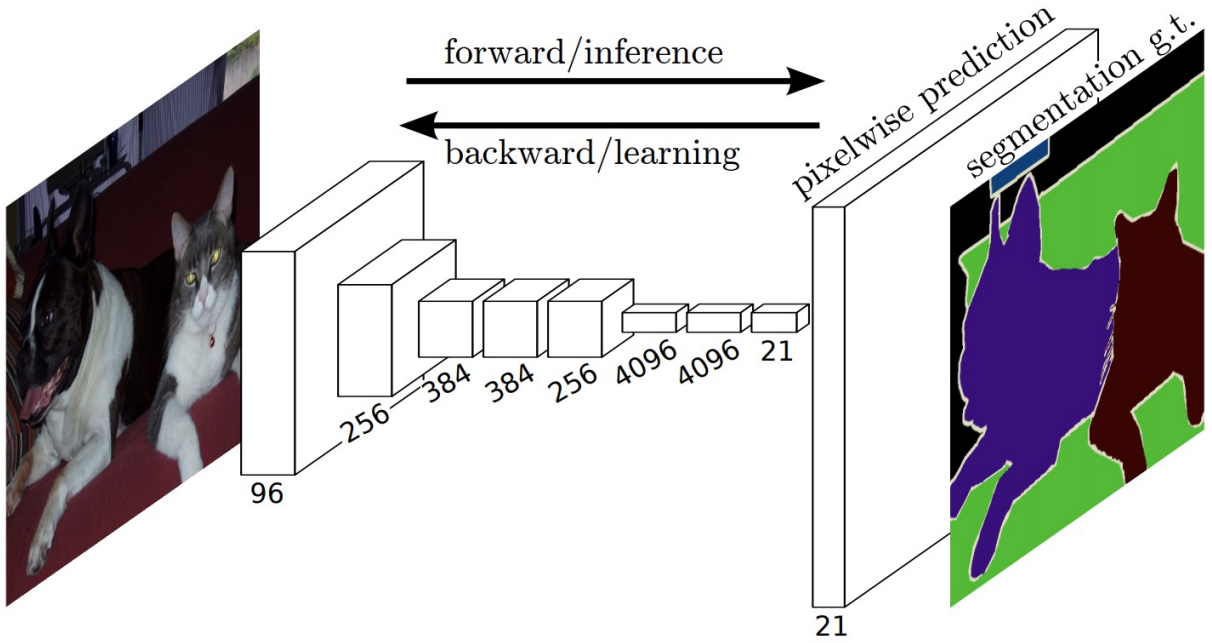
What we do is not a deconvolution.

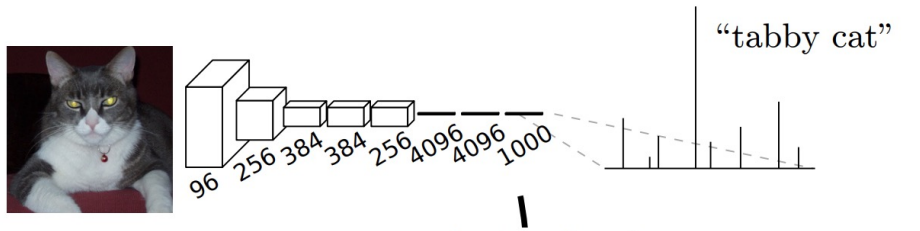
Please be precise in terms

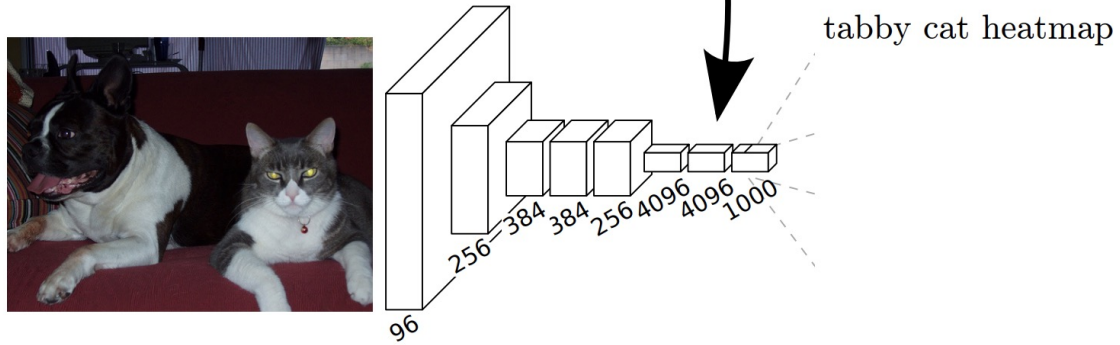
Right terms:

- Transposed convolution
- Upconvolution

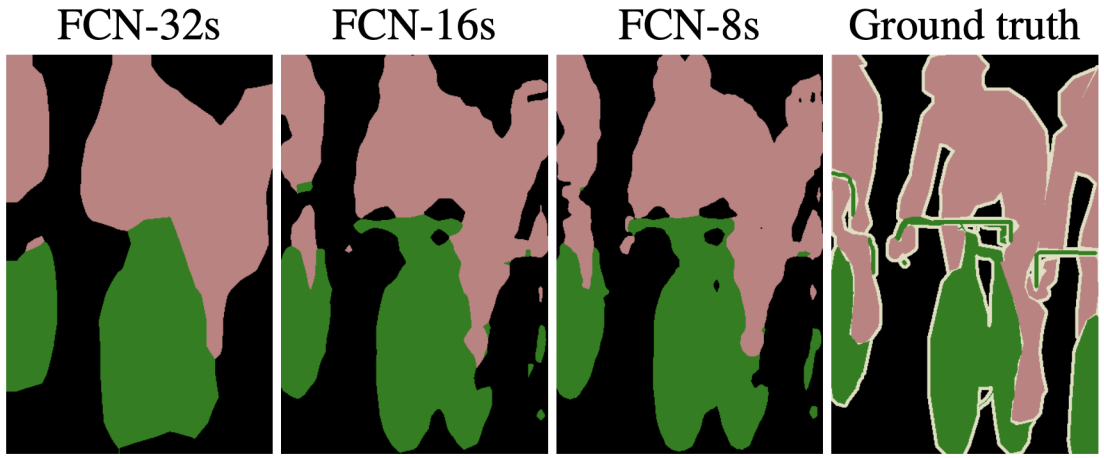
FCN – Fully Convolutional Network





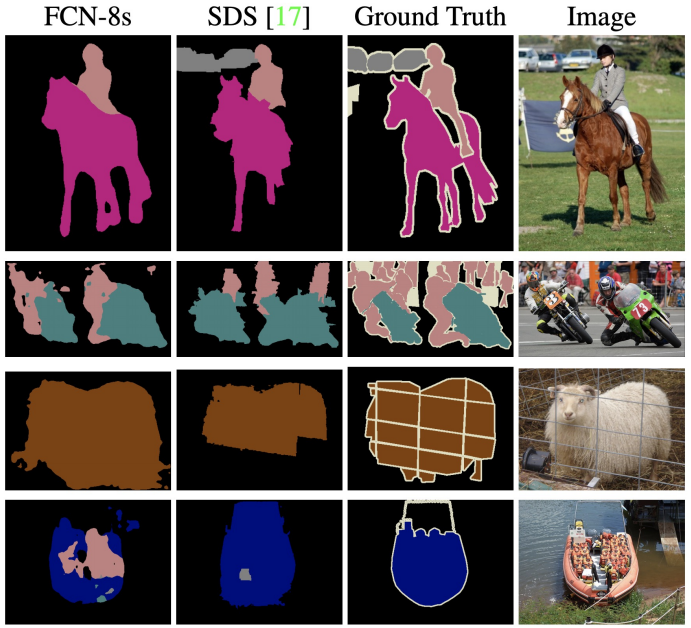


FCN: Results

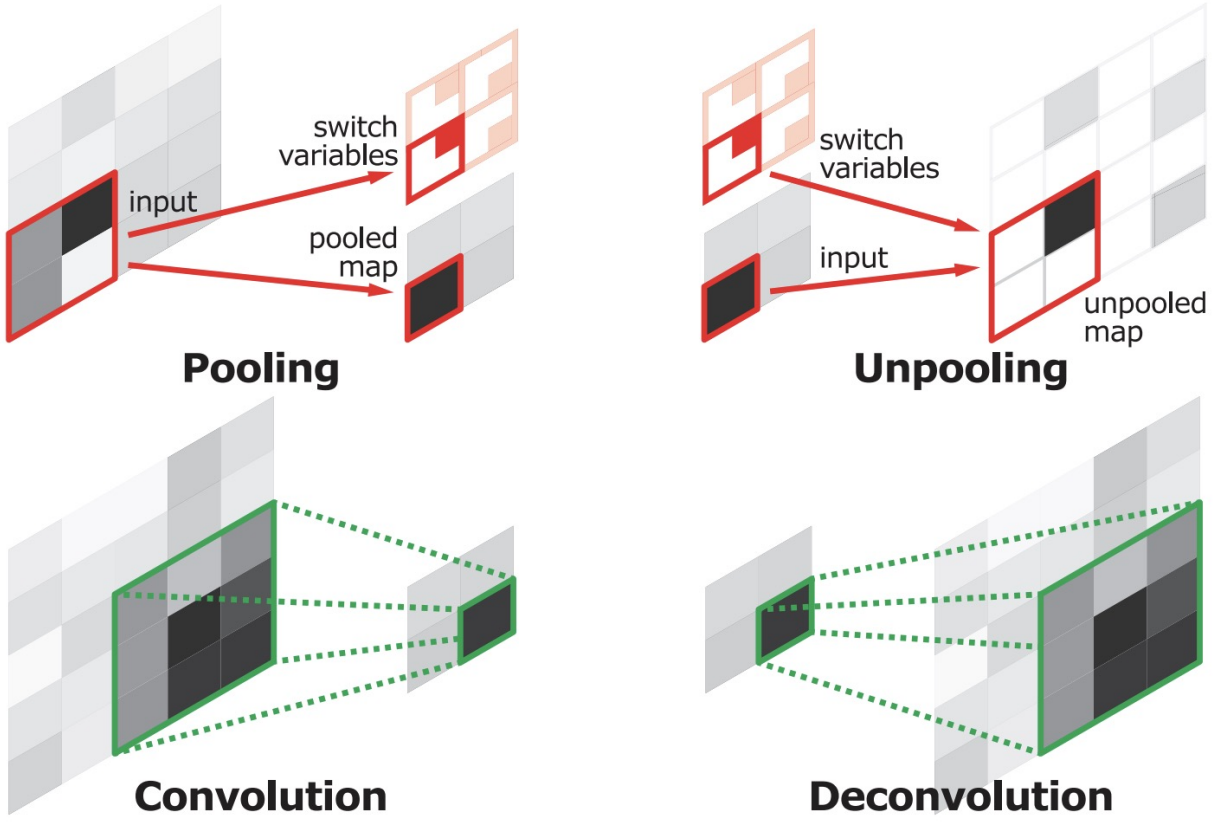


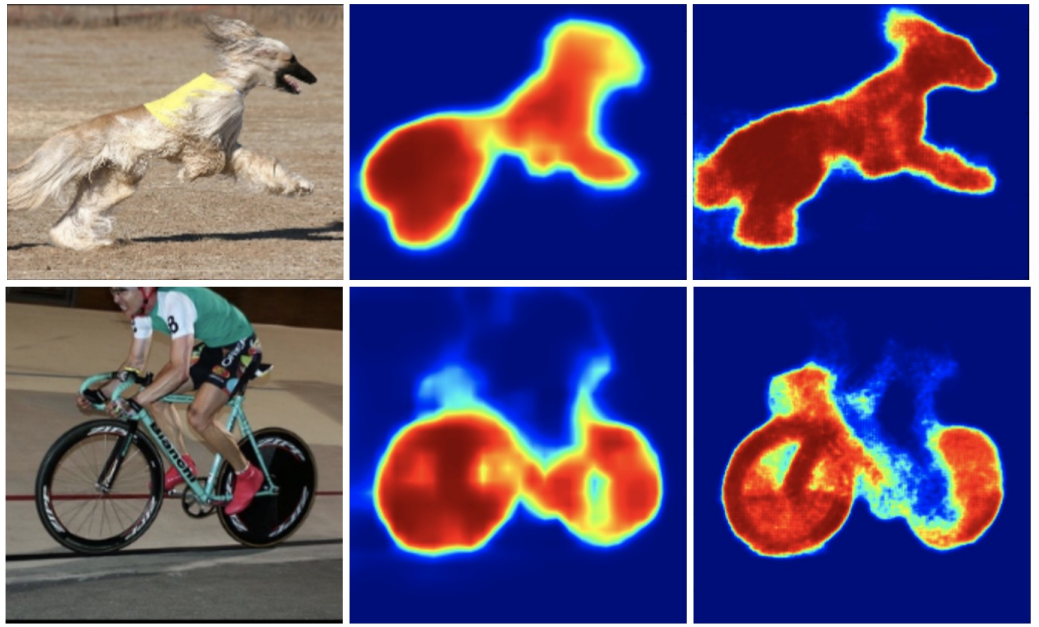
	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**



DeconvNet: use Max Unpooling





(a) Input image (b) FCN-8s (c) Ours

Figure 5. Comparison of class conditional probability maps from FCN and our network (top: dog, bottom: bicycle).

DeconvNet: use Max Unpooling

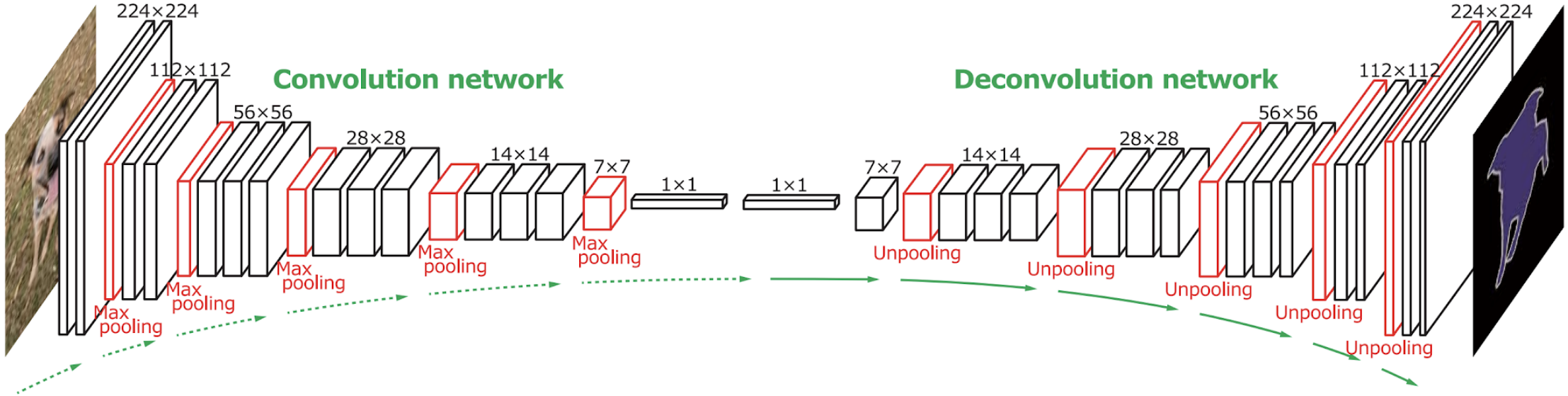
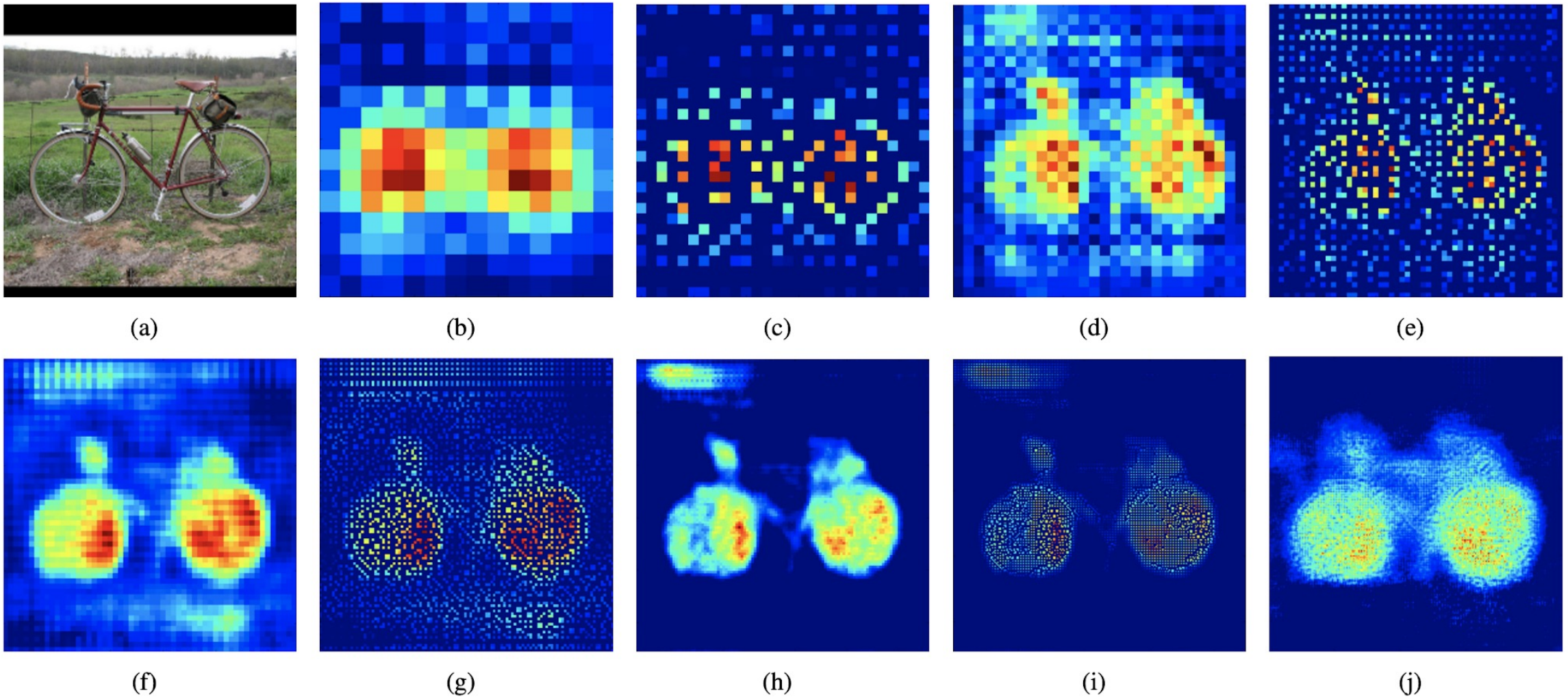


Figure 2. Overall architecture of the proposed network. On top of the convolution network based on VGG 16-layer net, we put a multi-layer deconvolution network to generate the accurate segmentation map of an input proposal. Given a feature representation obtained from the convolution network, dense pixel-wise class prediction map is constructed through multiple series of unpooling, deconvolution and rectification operations.

DeconvNet: use Max Unpooling



SegNet: Further development

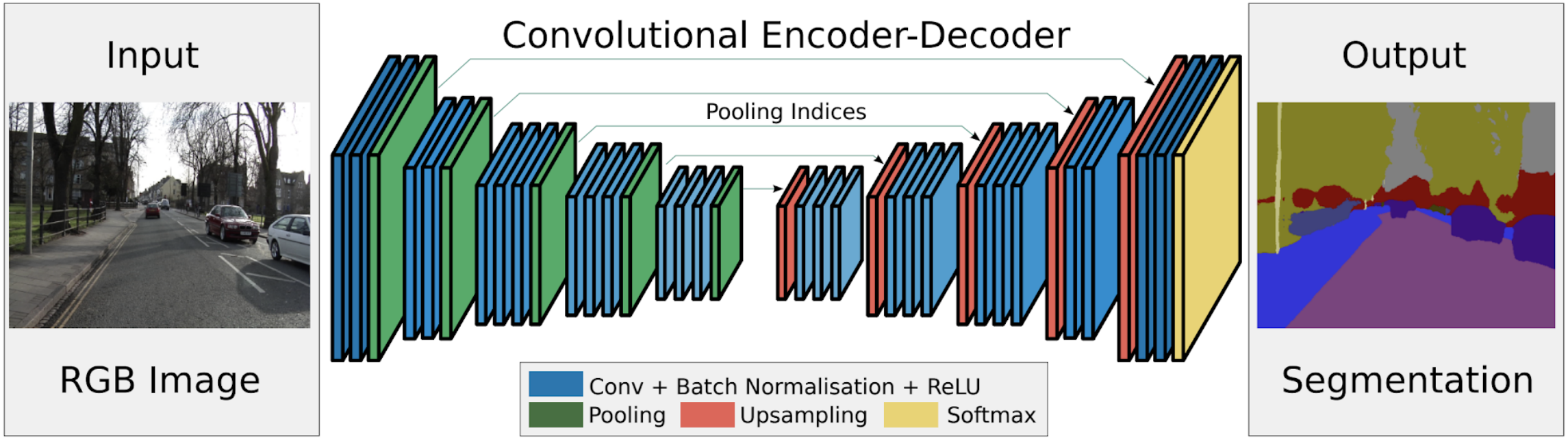
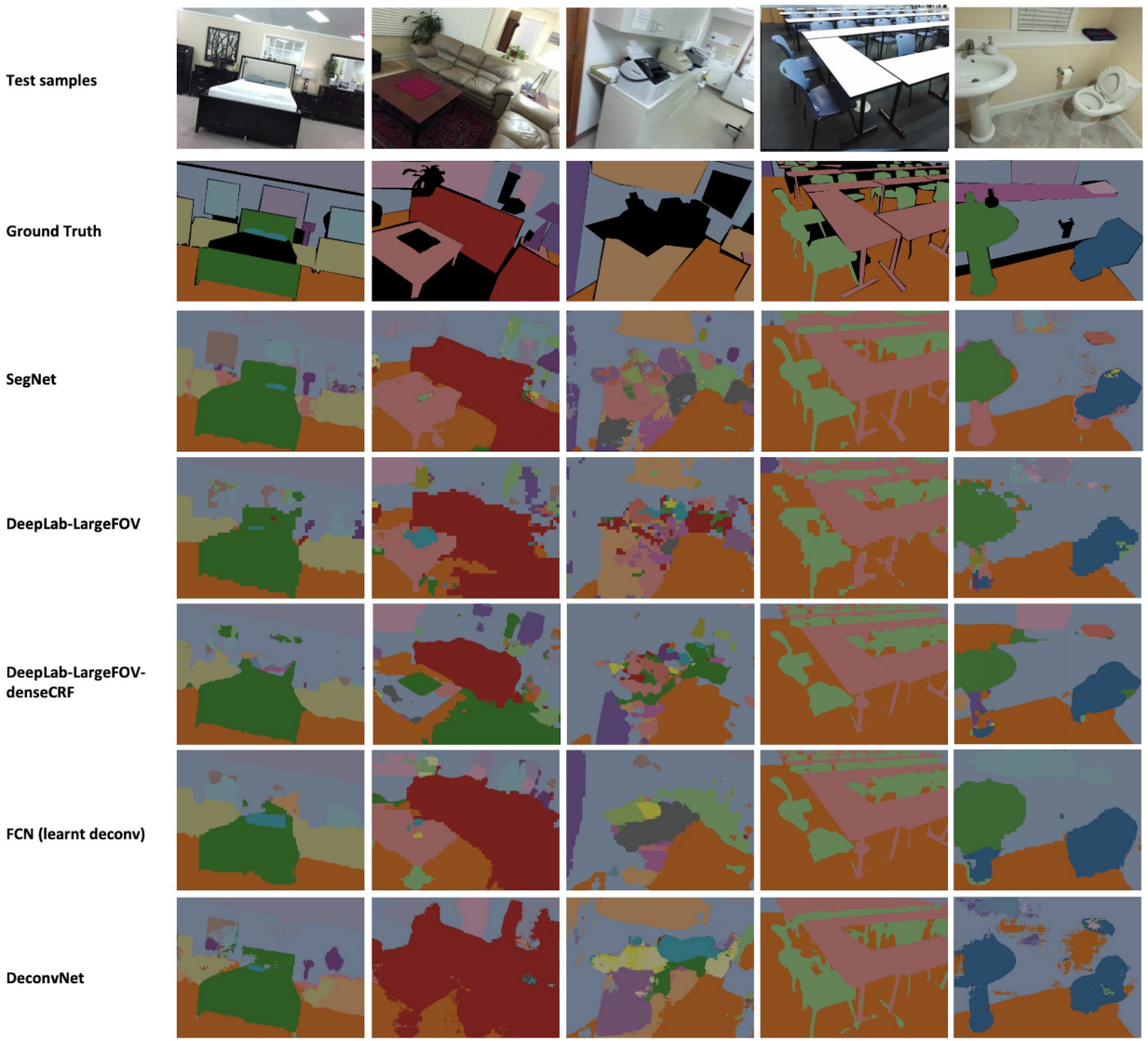
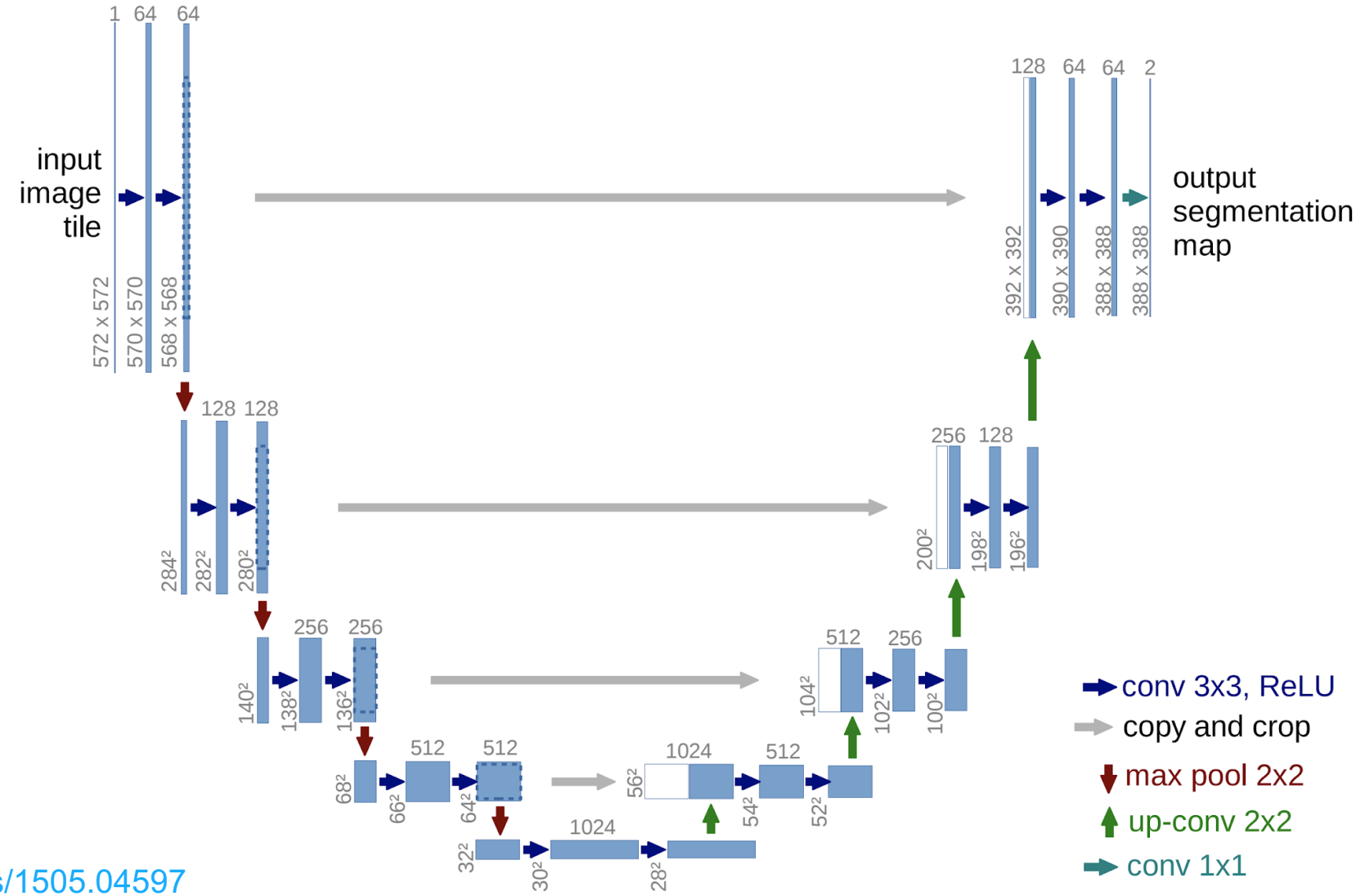


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

SegNet: Further development



U-Net



https://arxiv.org/abs/1505.04597

U-Net

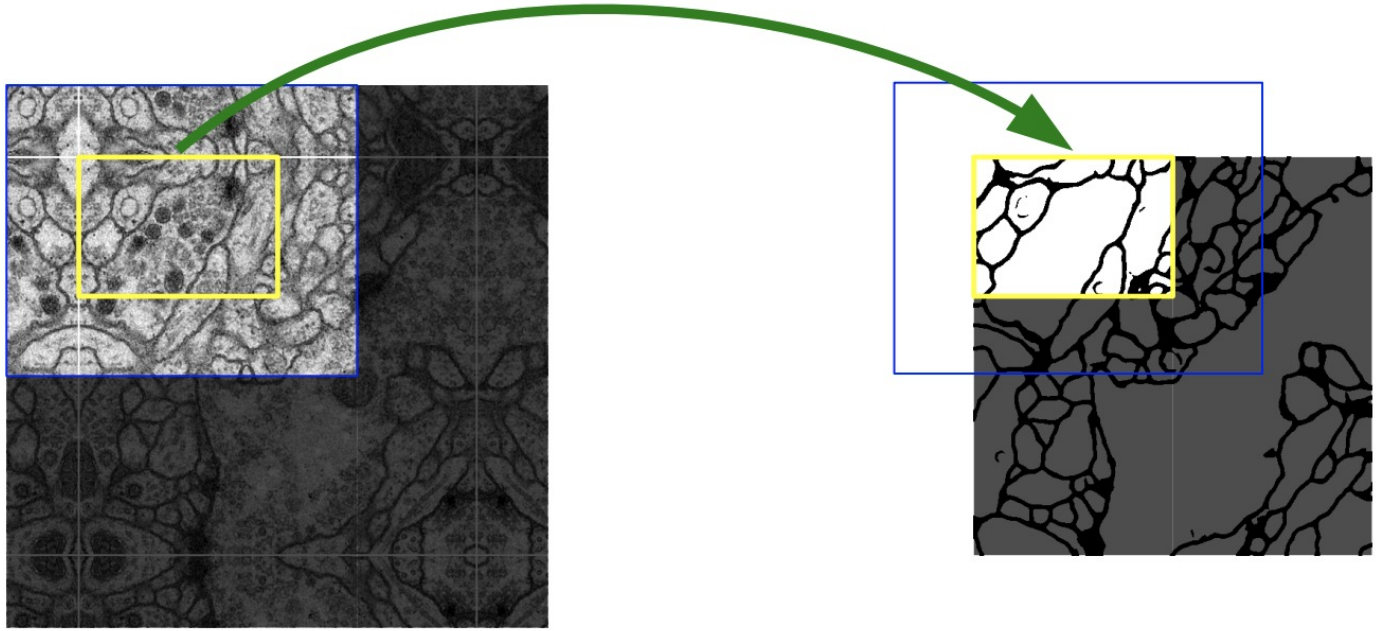
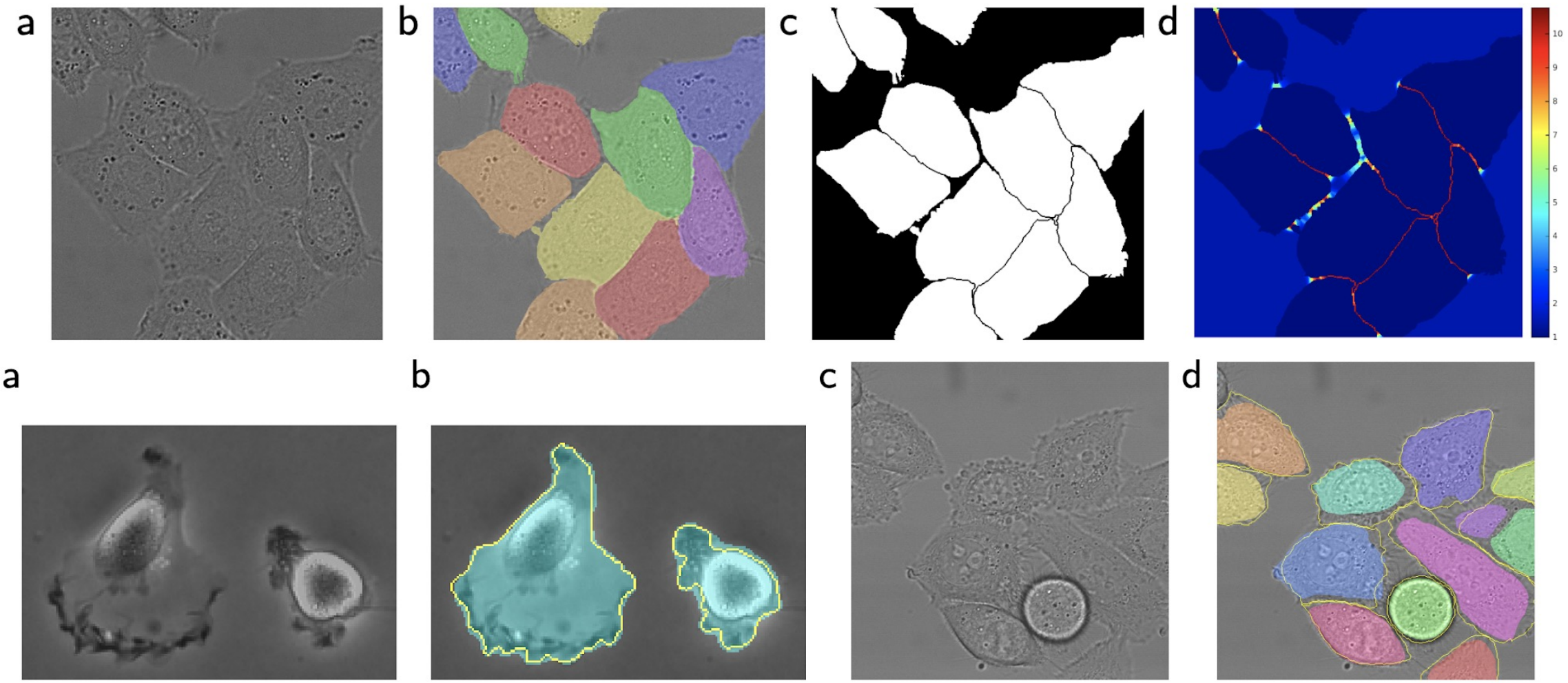
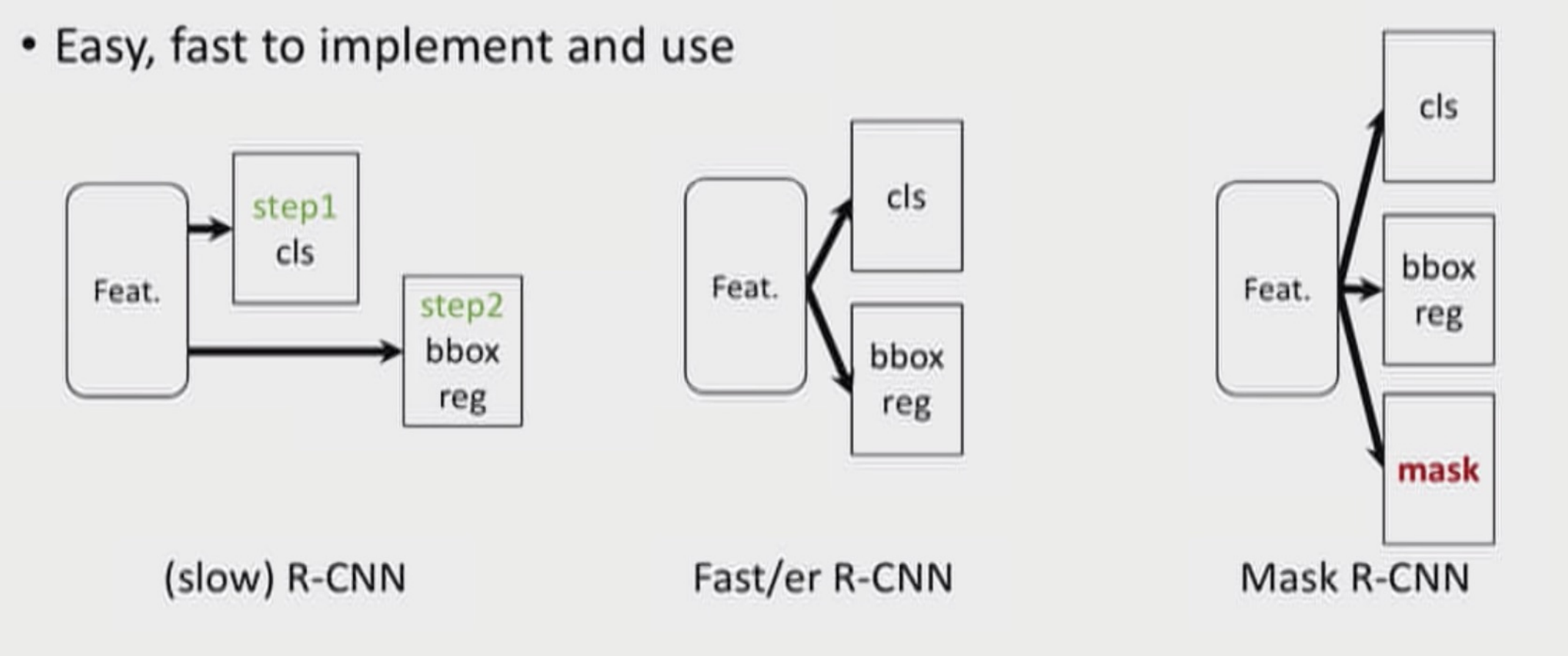


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

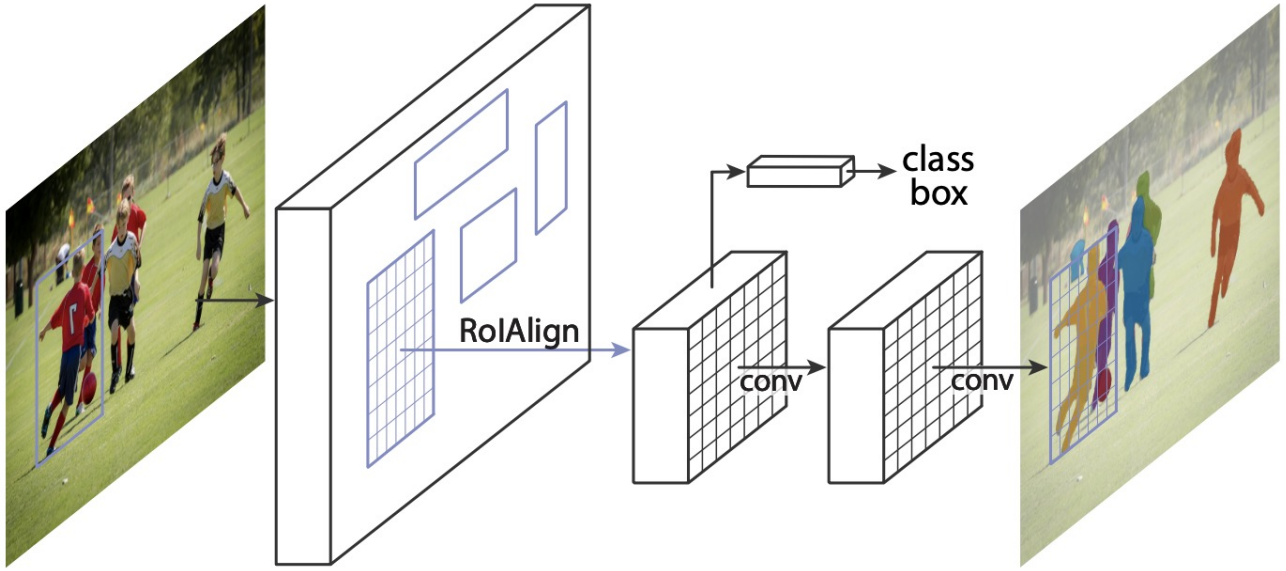
U-Net

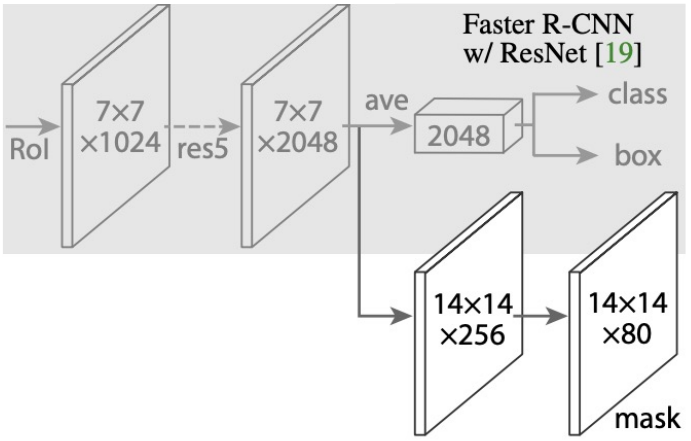


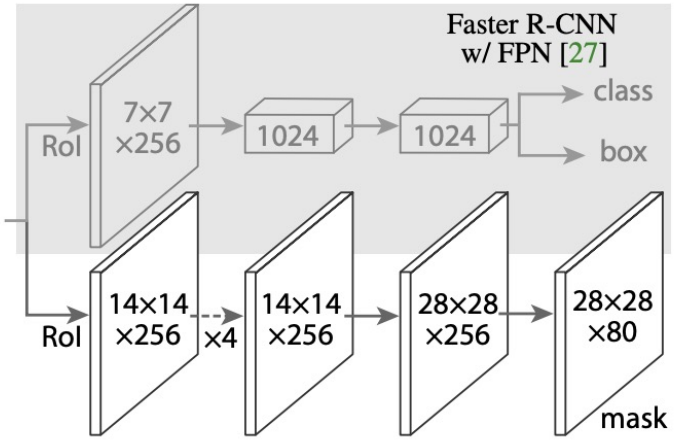
Mask R-CNN: Instance segmentation



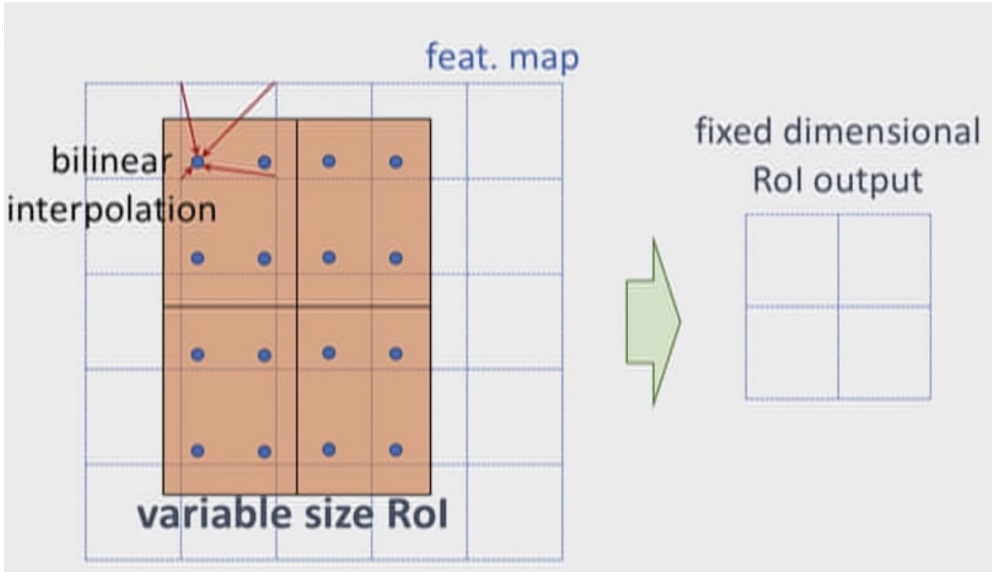
Mask R-CNN

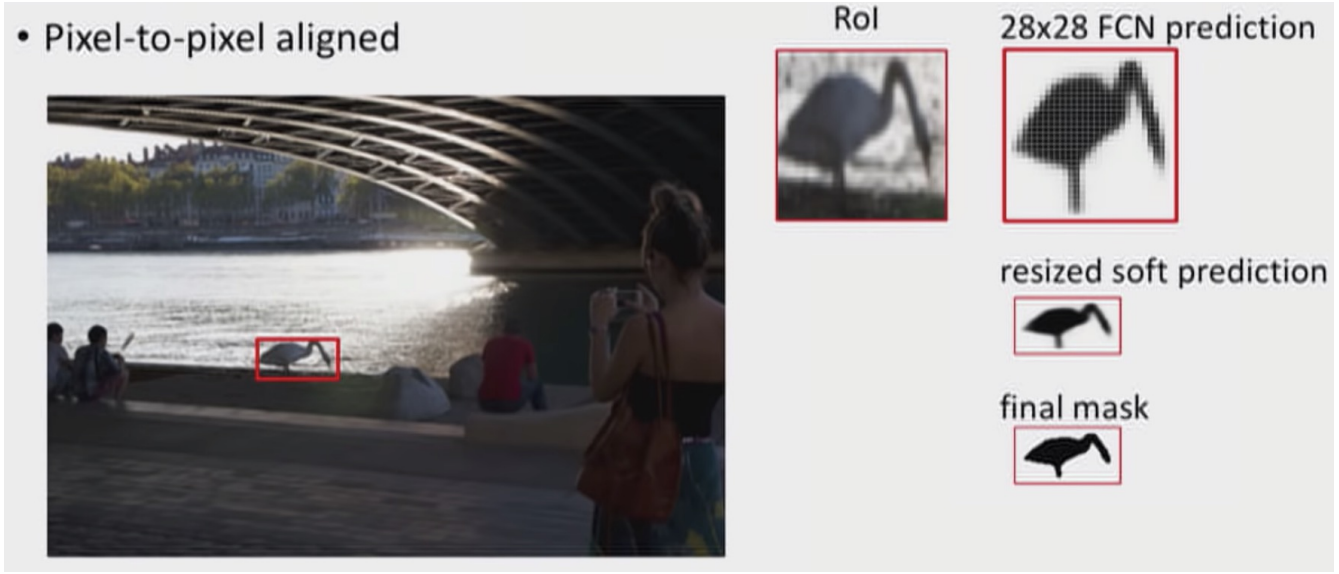

class
box
RoIAlign
conv
conv


Faster R-CNN
w/ ResNet [19]
RoI
7×7
×1024
res5
7×7
×2048
ave
2048
class
box
14×14
×256
14×14
×80
mask


Faster R-CNN
w/ FPN [27]
RoI
7×7
×256
1024
1024
class
box
RoI
14×14
×256
×4
14×14
×256
28×28
×256
28×28
×80
mask

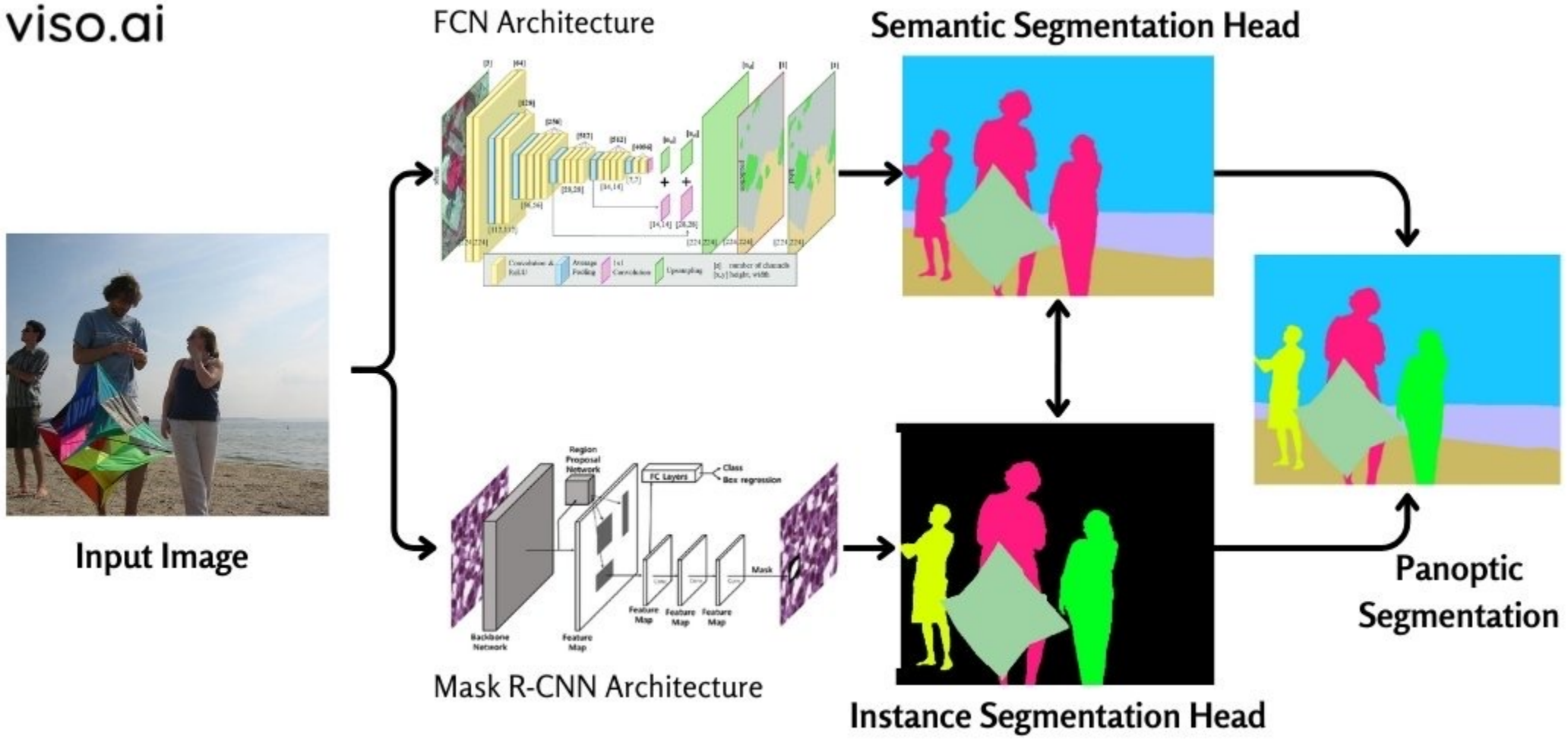
Mask R-CNN: RoI Align





Mask R-CNN: Results

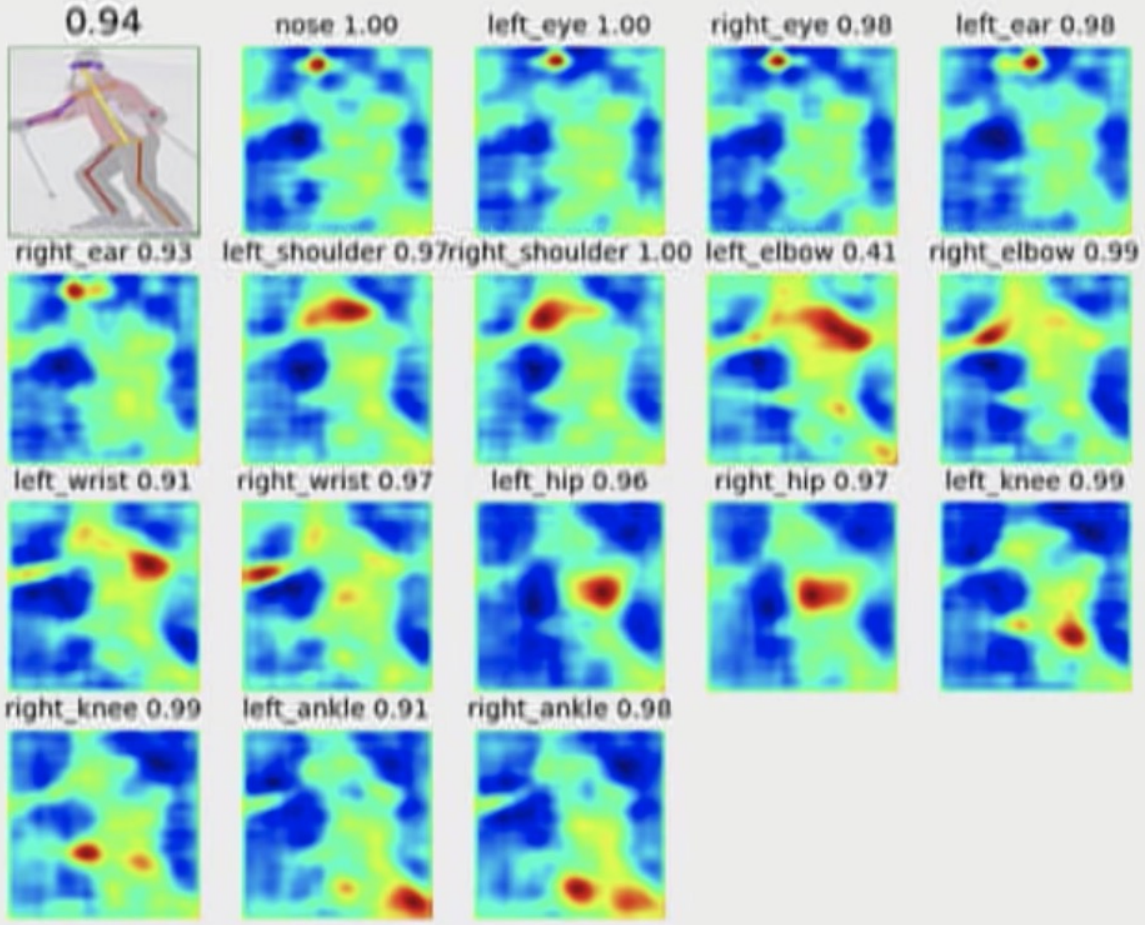
Panoptic segmentation with Mask R-CNN



Bonus: Pose estimation

- keypoint = 1-hot mask
- human pose = 17 masks

- One framework for
  - ✓bbox
  - ✓mask
  - ✓keypoint

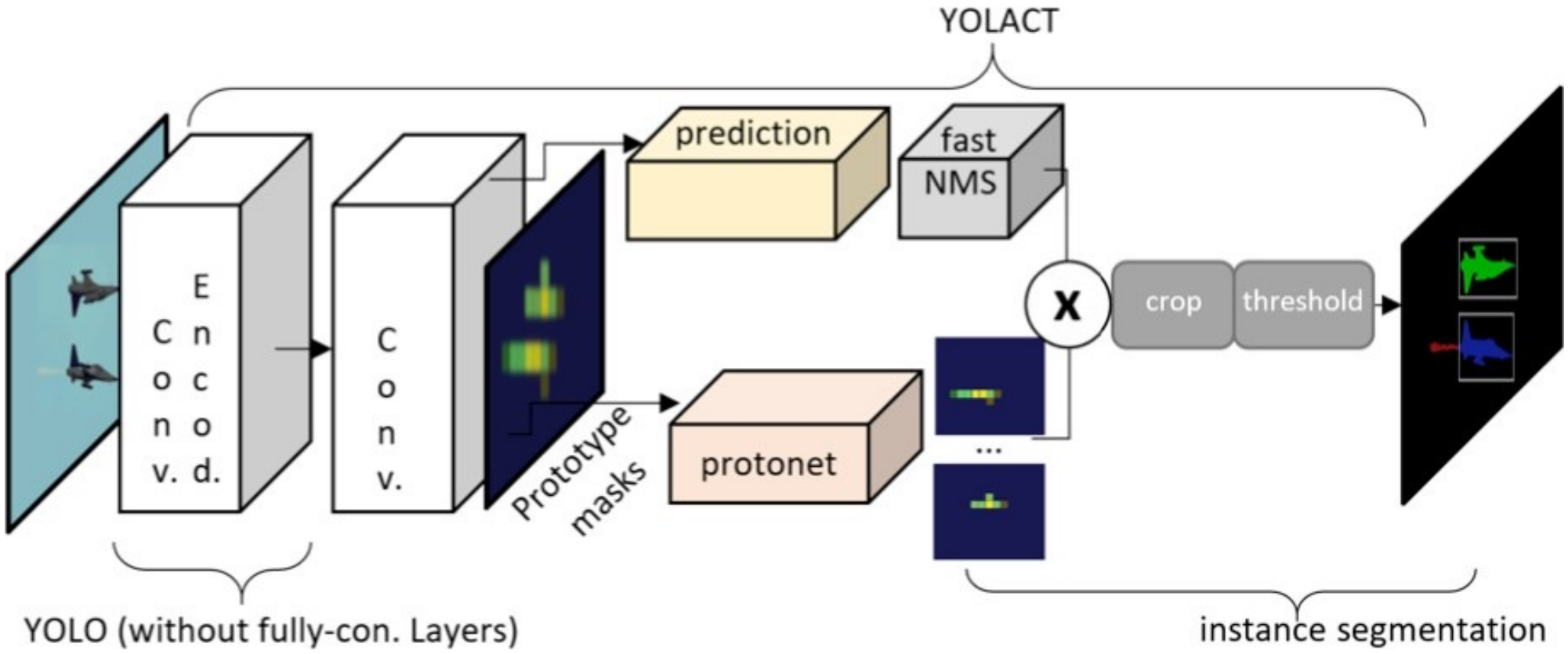


Bonus: Pose estimation

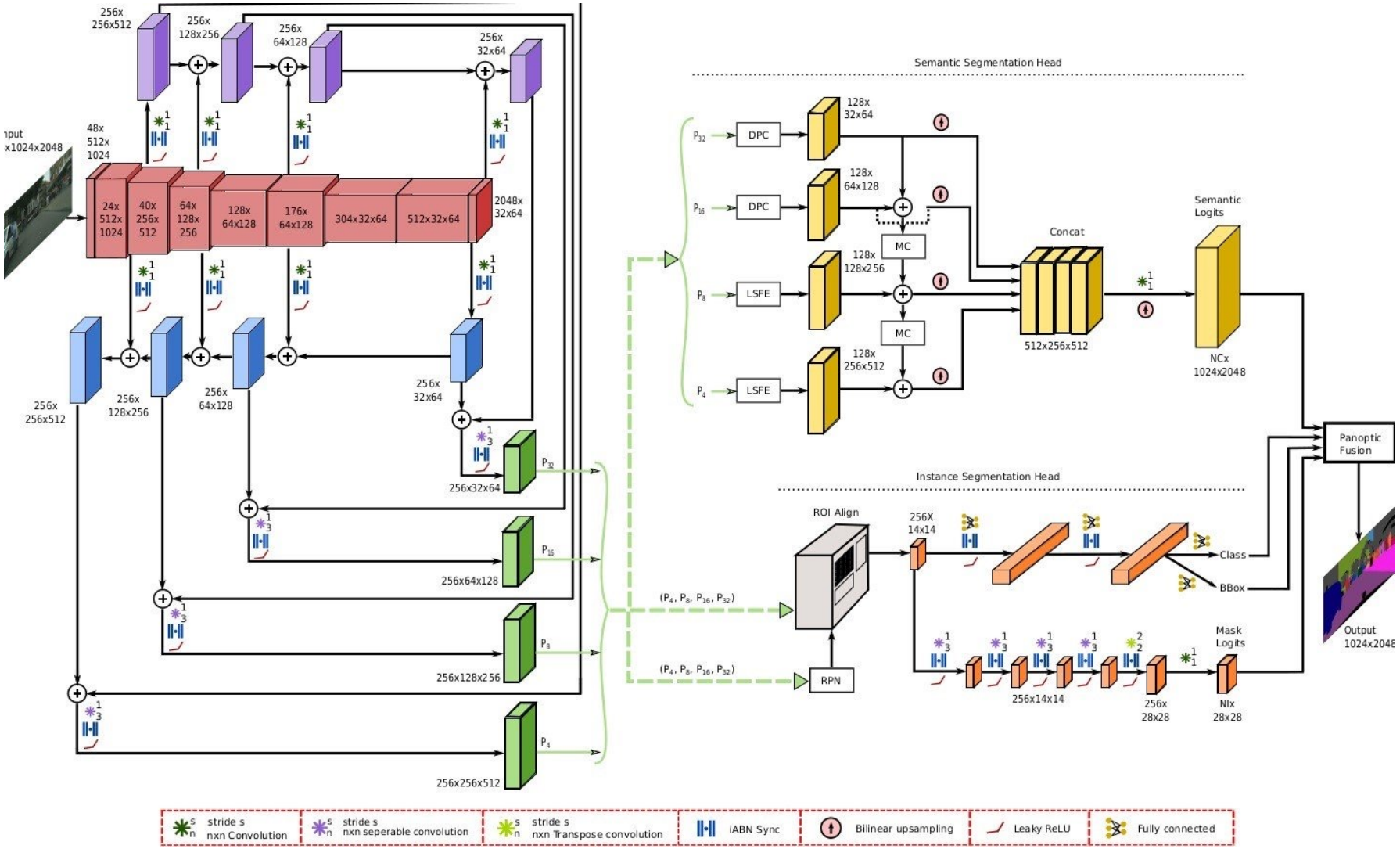
YOLACT

31.2% mAP on COCO dataset

~ 30 fps



EfficientPS



SAM

Segment Anything Model is universal solution for such problems

Further development https://github.com/yangchris11/samurai

More nice works

- Total Segmentator - segments all human bones and organs on MRI
  - https://github.com/wasserth/TotalSegmentator

Links

- https://arxiv.org/pdf/2006.14822.pdf
- https://github.com/JunMa11/SegLossOdyssey
- https://github.com/shruti-jadon/Semantic-Segmentation-Loss-Functions
- https://www.sciencedirect.com/science/article/abs/pii/S1361841521000815
- https://github.com/LIVIAETS/boundary-loss
- https://arxiv.org/pdf/1812.07032.pdf
- Transposed convolution visualization
- https://www.cvat.ai/
- https://github.com/hustvl/YOLOP/tree/main

Канал стажировок Яндекса

Спасибо за внимание

Доп. материалы

Y&Y