Structured Predictions with Deep Learning, Part 2

James Hays

Outline – More complex outputs from deep networks

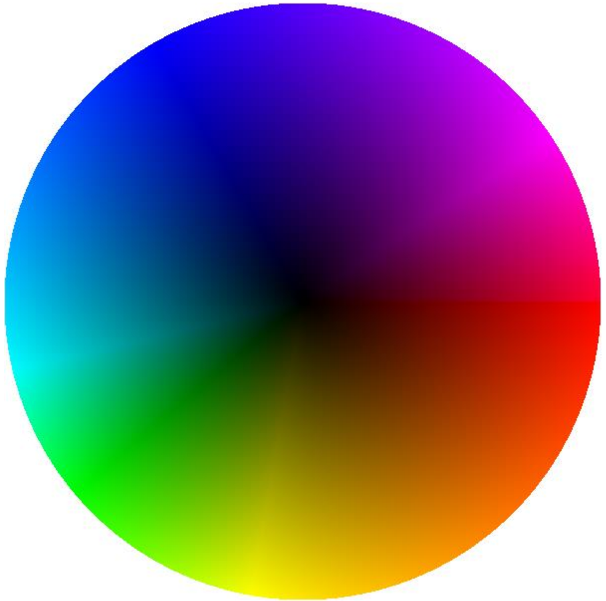
- Image Output (e.g. colorization, semantic segmentation, super-resolution, stylization, depth estimation…)
- Attributes
- Text Captions
- Semantic Keypoints
- Object Detection

Multi-Person Pose Estimation

Multi-Person Pose Estimation

Major Challenge: Part-to-Person Association

Novelty: Part Affinity Fields for Parts Association



Part Affinity Field between right elbow and wrist

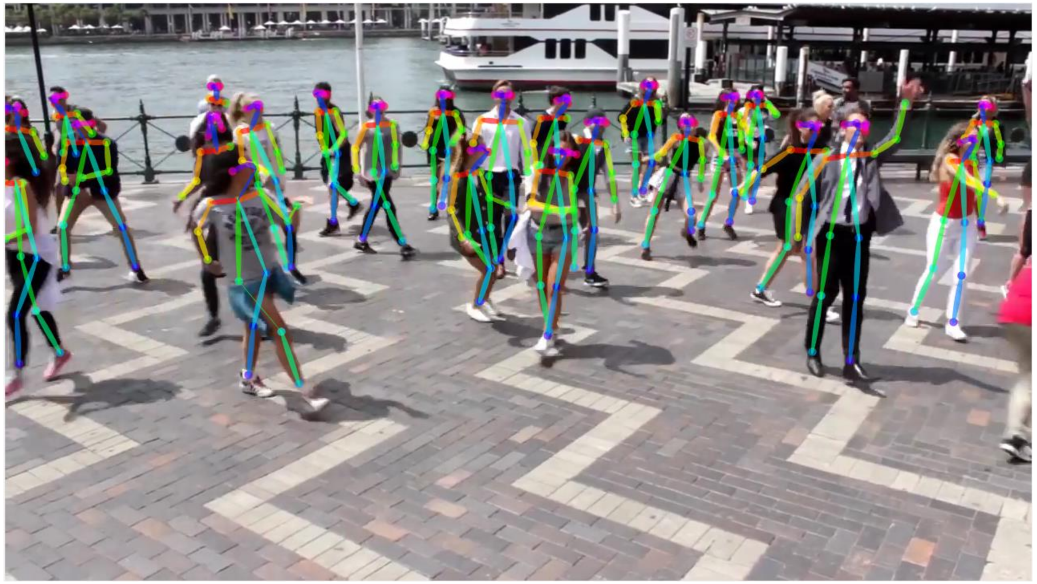
Novelty: Part Affinity Fields for Parts Association

Part Affinity Field between right elbow and wrist

Realtime Multi-Person Pose Estimation using Part Affinity Fields

Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh. CVPR 2017

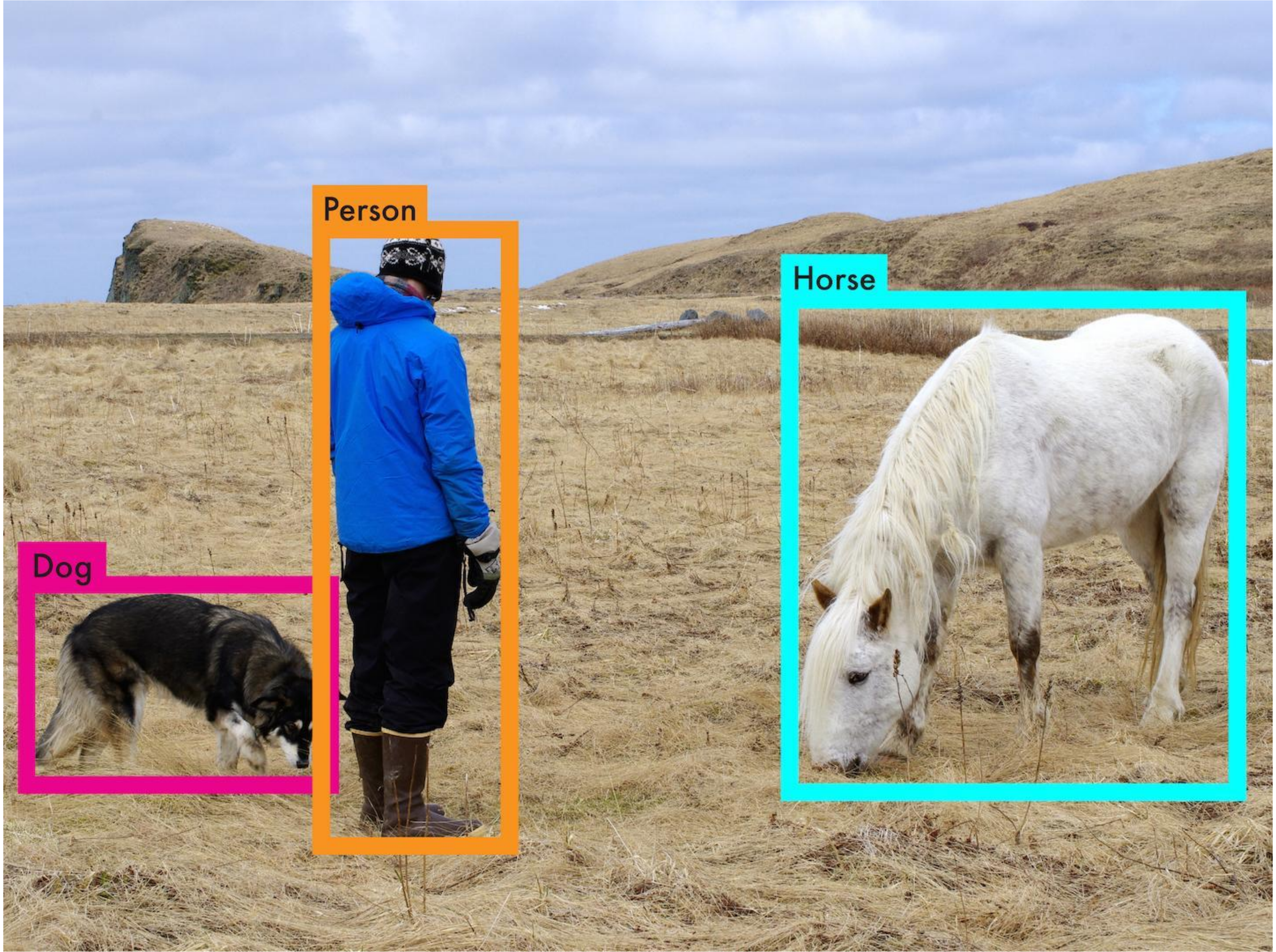
- “Bottom Up” method of detecting one category of objects (people)
- Instead of detecting person instances and then their keypoints, it detects keypoints and then assembles them into person instances.
- The method is not “end to end”. The network is supervised to produce an intermediate representation that is easy to post-process into the desired output. The network still does the heavy lifting.



Outline – More complex outputs from deep networks

- Image Output (e.g. colorization, semantic segmentation, super-resolution, stylization, depth estimation…)
- Attributes
- Text Captions
- Semantic Keypoints
- Object Detection
  - “Single shot” or “one stage” detectors like YOLO or SSD. The network runs once per image.
  - “Two stage” detectors like Mask RCNN. A feature extractor network runs once per image, various “head” networks run an arbitrary amount of times.

Joseph
Redmon
Ross
Girshick
Santosh
Divvala
Ali
Farhadi
Dog
Most
Accurate
Real-Time Detector
2016
Fastest
object detector
in the literature
2016
“You Only Look Once”
Real-Time
Detection


Person
Horse
Dog

Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img

Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img

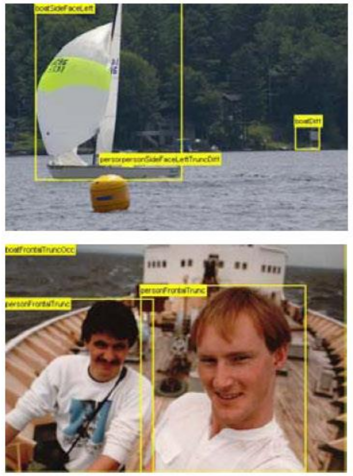
Examples

Aeroplane

Bicycle

Bird

Boat



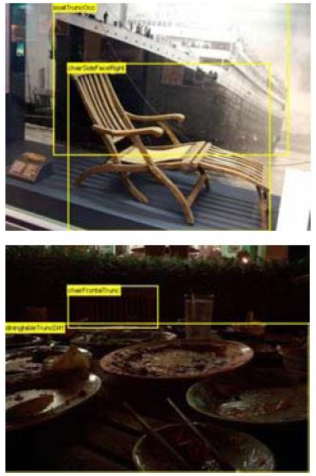
Bottle

Bus

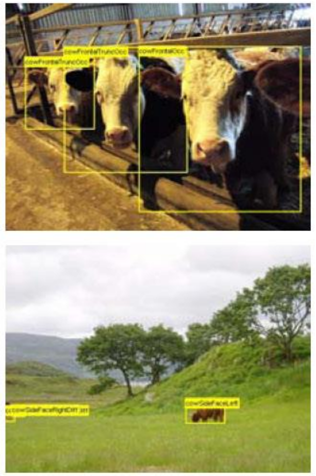
Car

Cat

Chair

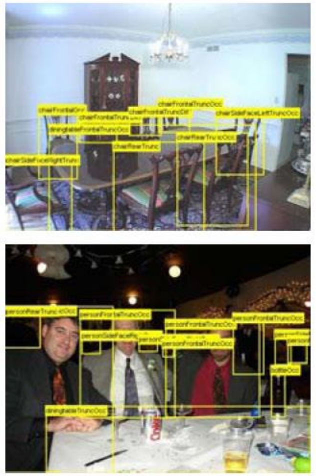


Cow



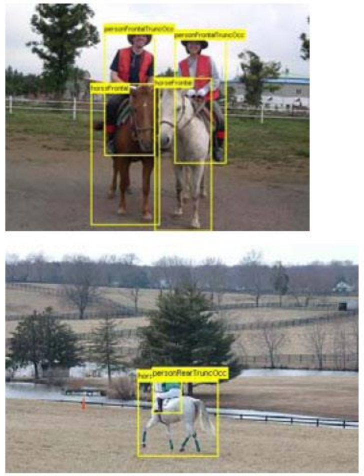
Examples

Dining Table



Dog

Horse

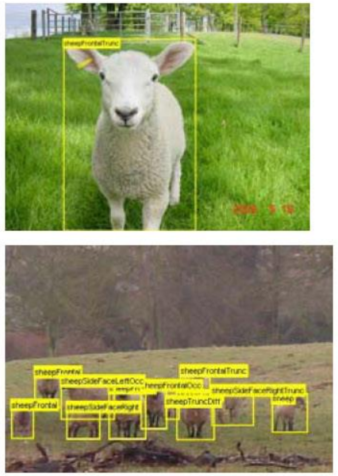


Motorbike

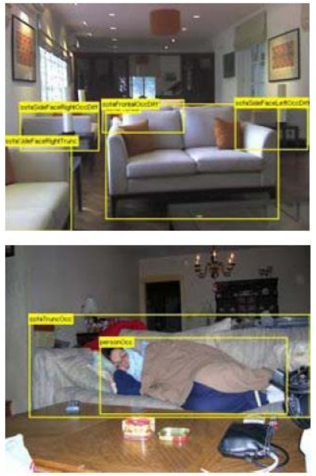
Person

Potted Plant

Sheep

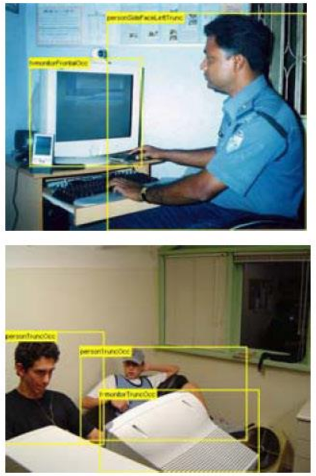


Sofa



Train

TV/Monitor



Accurate object detection is slow!

	Pascal 2007 mAP	**Speed**	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img

⅓ Mile, 1760 feet

Accurate object detection is slow!

	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

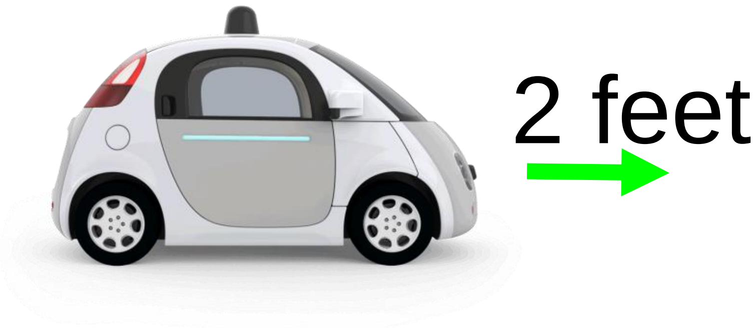
176 feet

Accurate object detection is slow!

	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

Accurate object detection is slow!

	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	63.4	45 FPS	22 ms/img

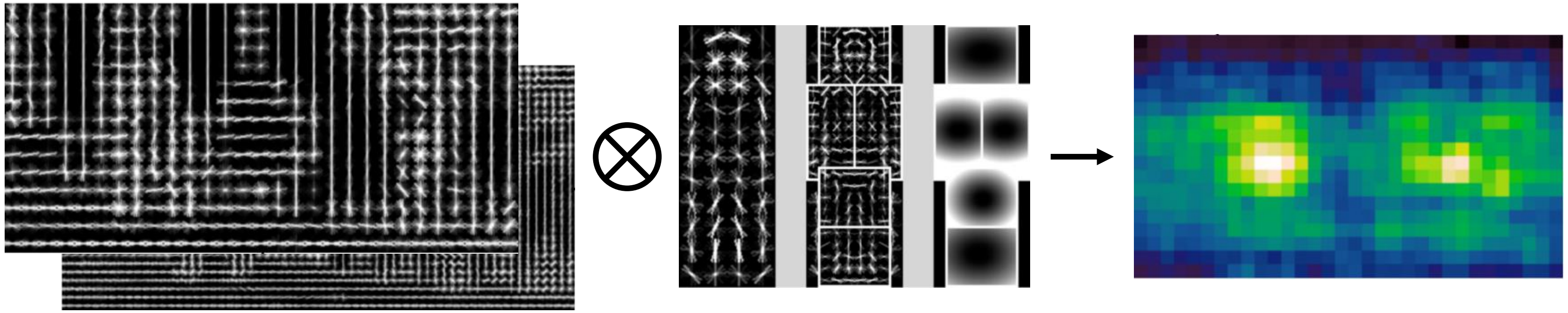


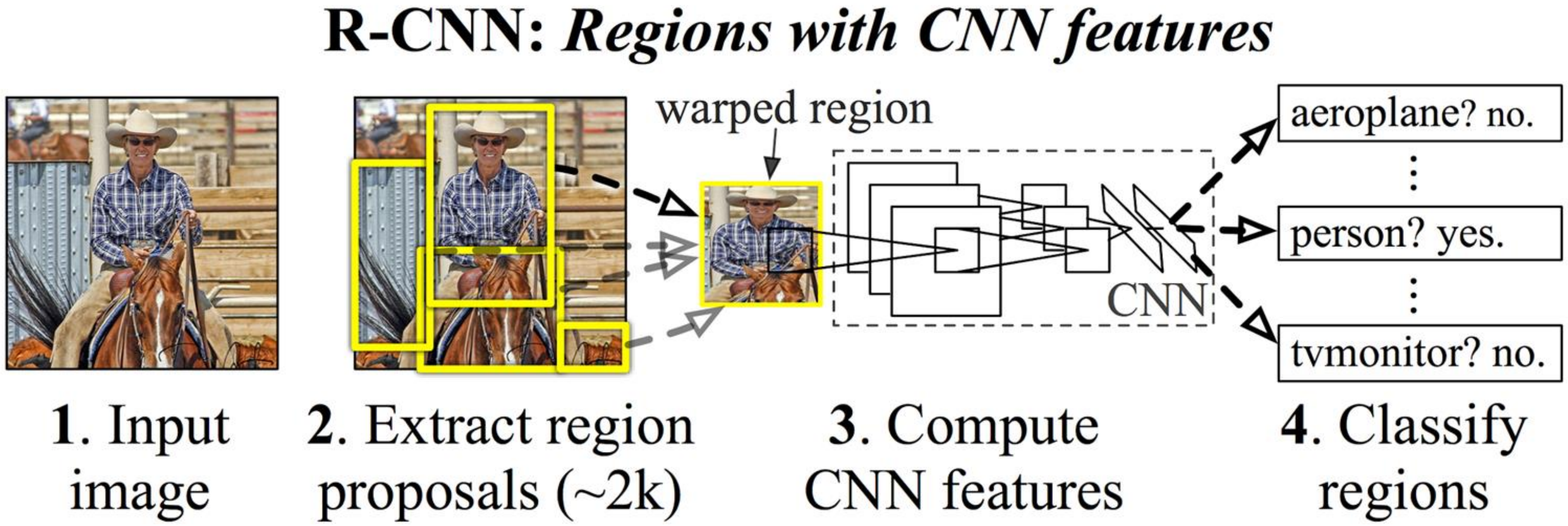
Accurate object detection is slow!

	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

2 feet

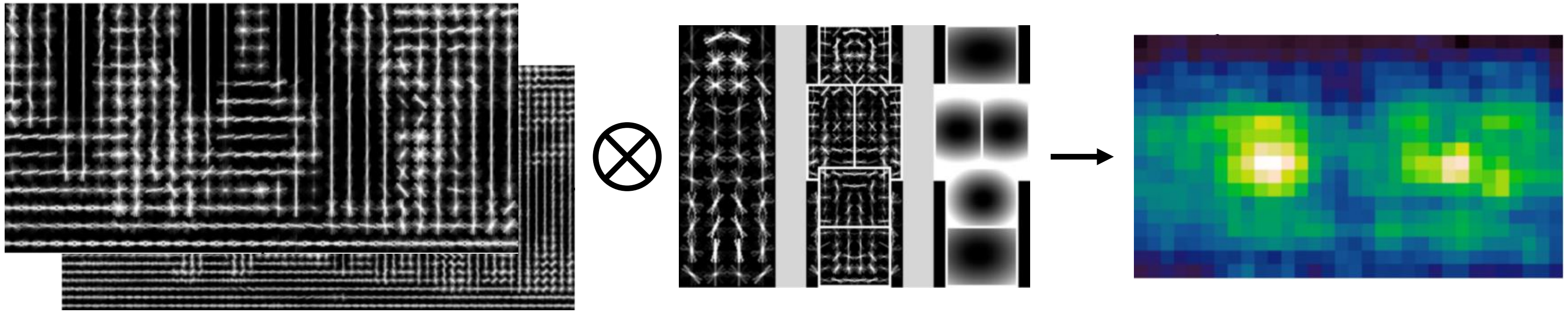
DPM: ***Deformable Part Models***



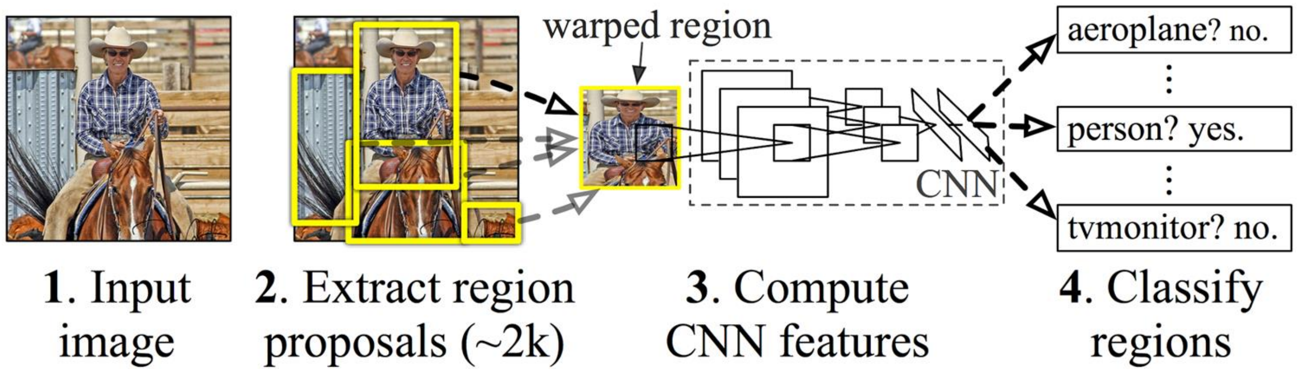

R-CNN: *Regions with CNN features*
warped region
CNN
aeroplane? no.
person? yes.
tvmonitor? no.
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

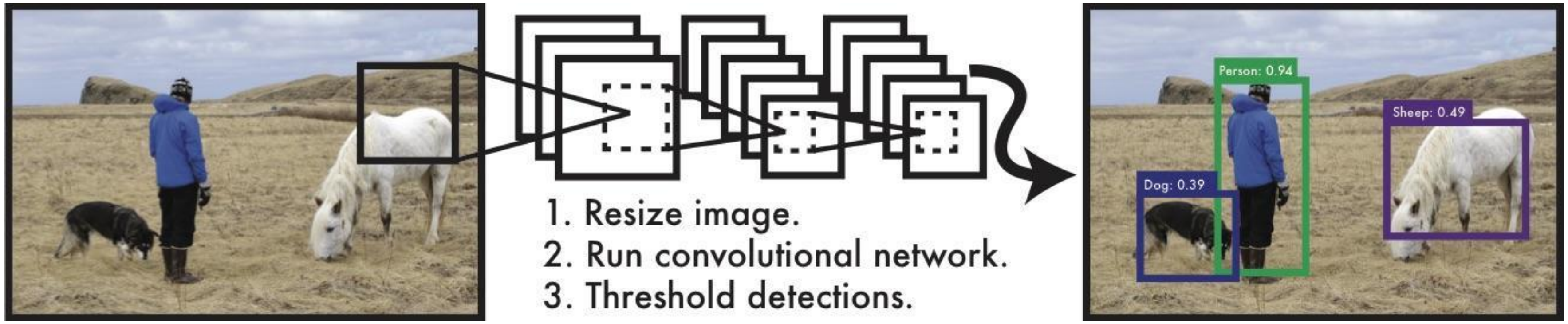
Sliding window, DPM, R-CNN all train region-based classifiers to perform detection

DPM: ***Deformable Part Models***



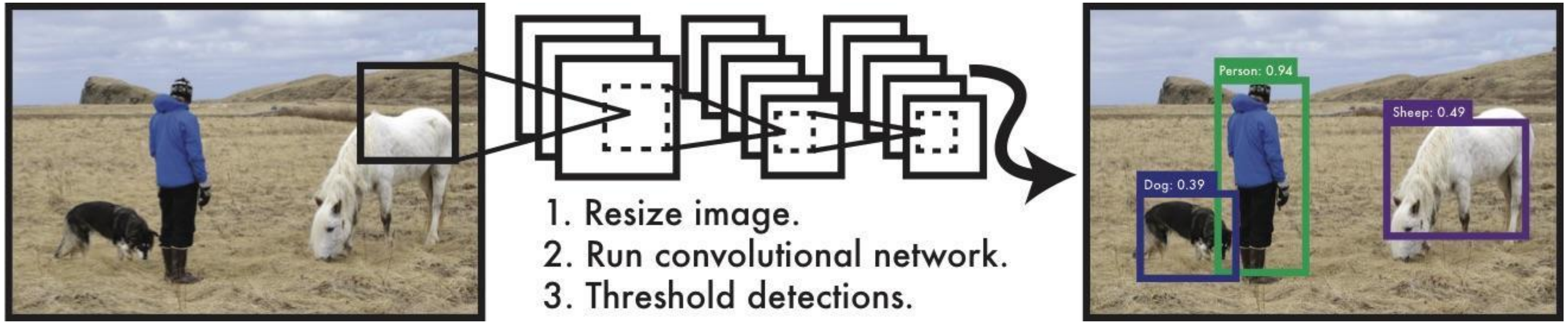
R-CNN: ***Regions with CNN features***



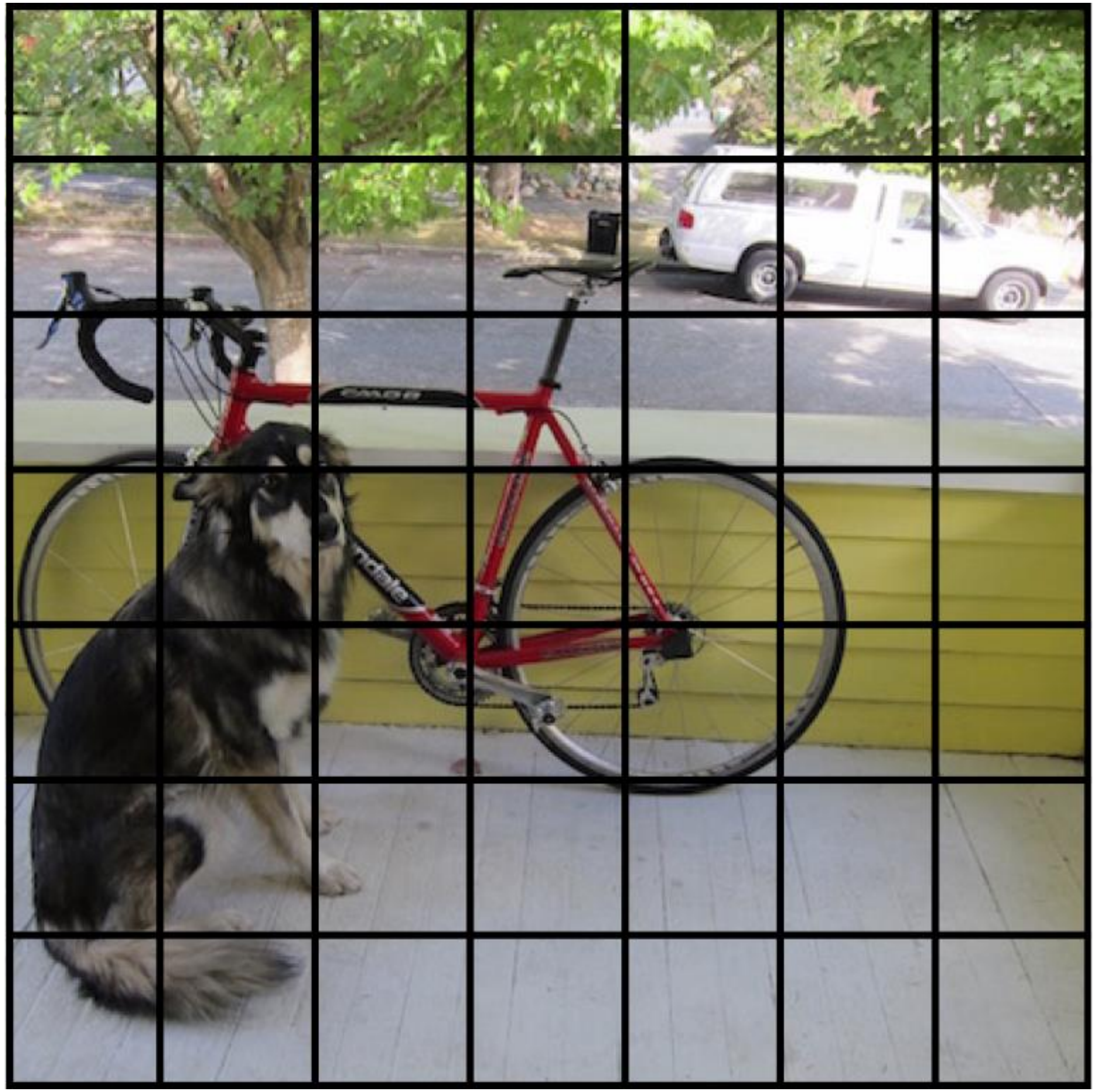

1. Resize image.
2. Run convolutional network.
3. Threshold detections.
Person: 0.94
Sheep: 0.49
Dog: 0.39

With YOLO, you only look once at an image to perform detection

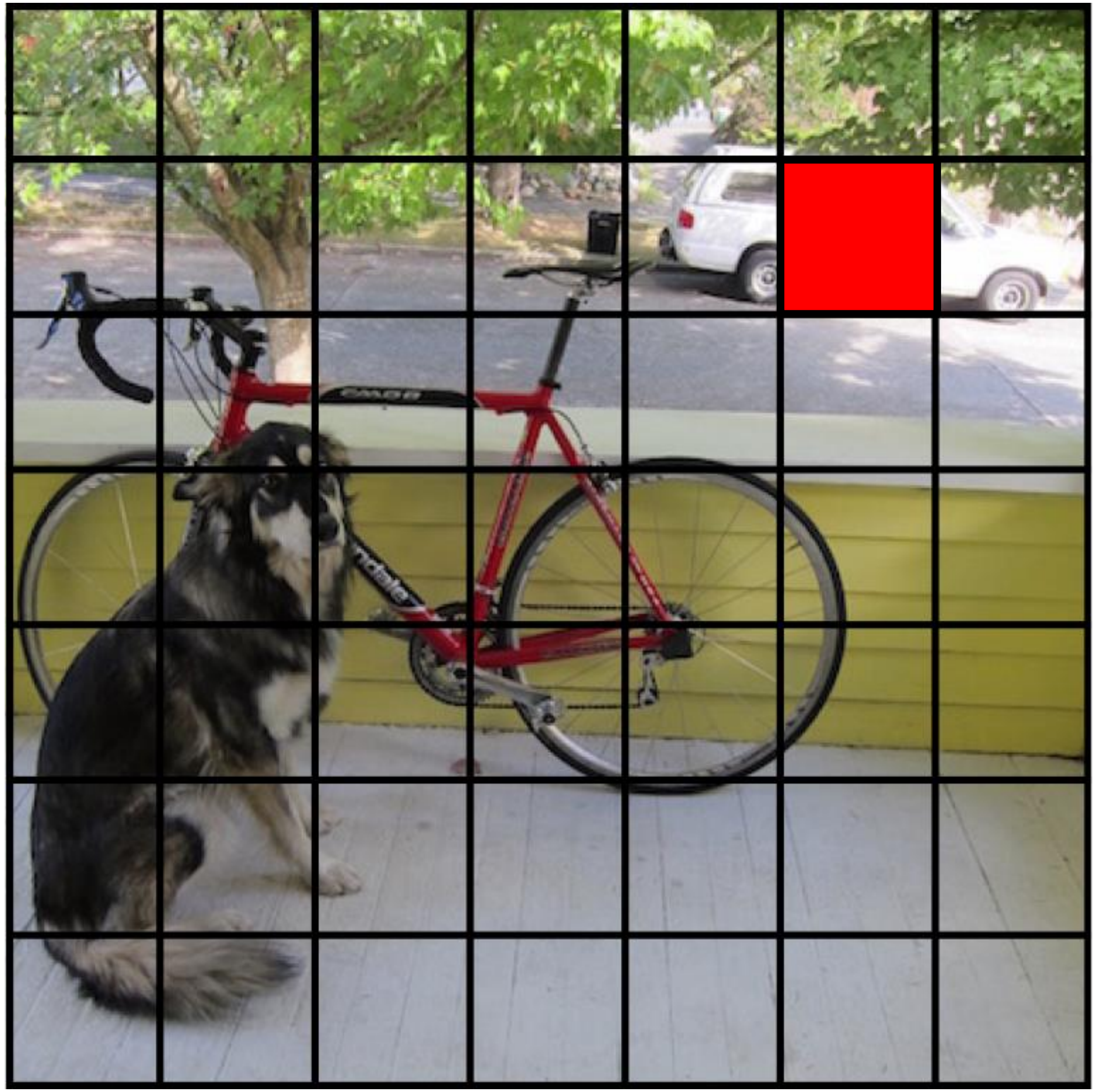
YOLO: ***You Only Look Once***



We split the image into a grid

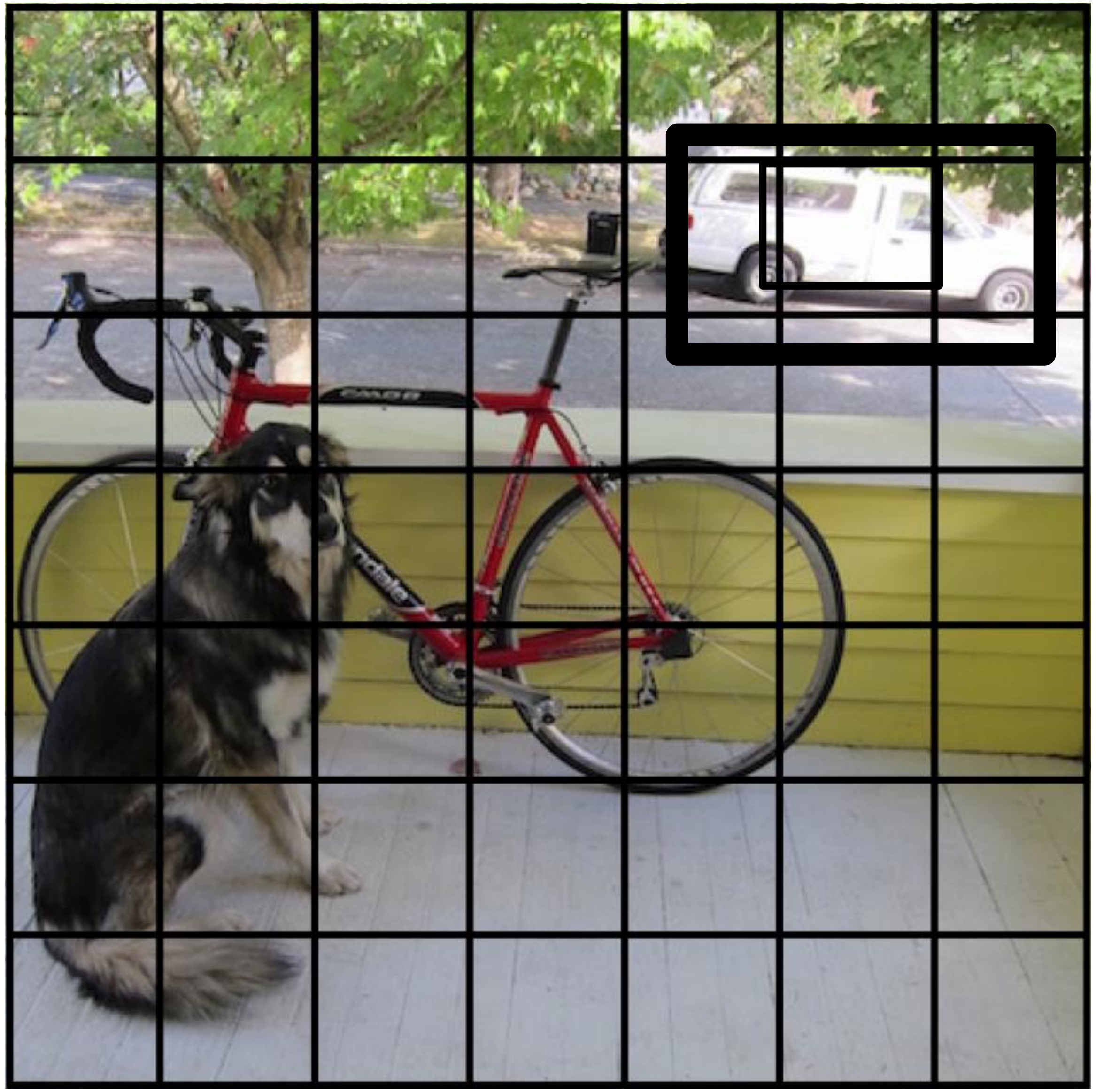


Each cell predicts boxes and confidences: P(Object)

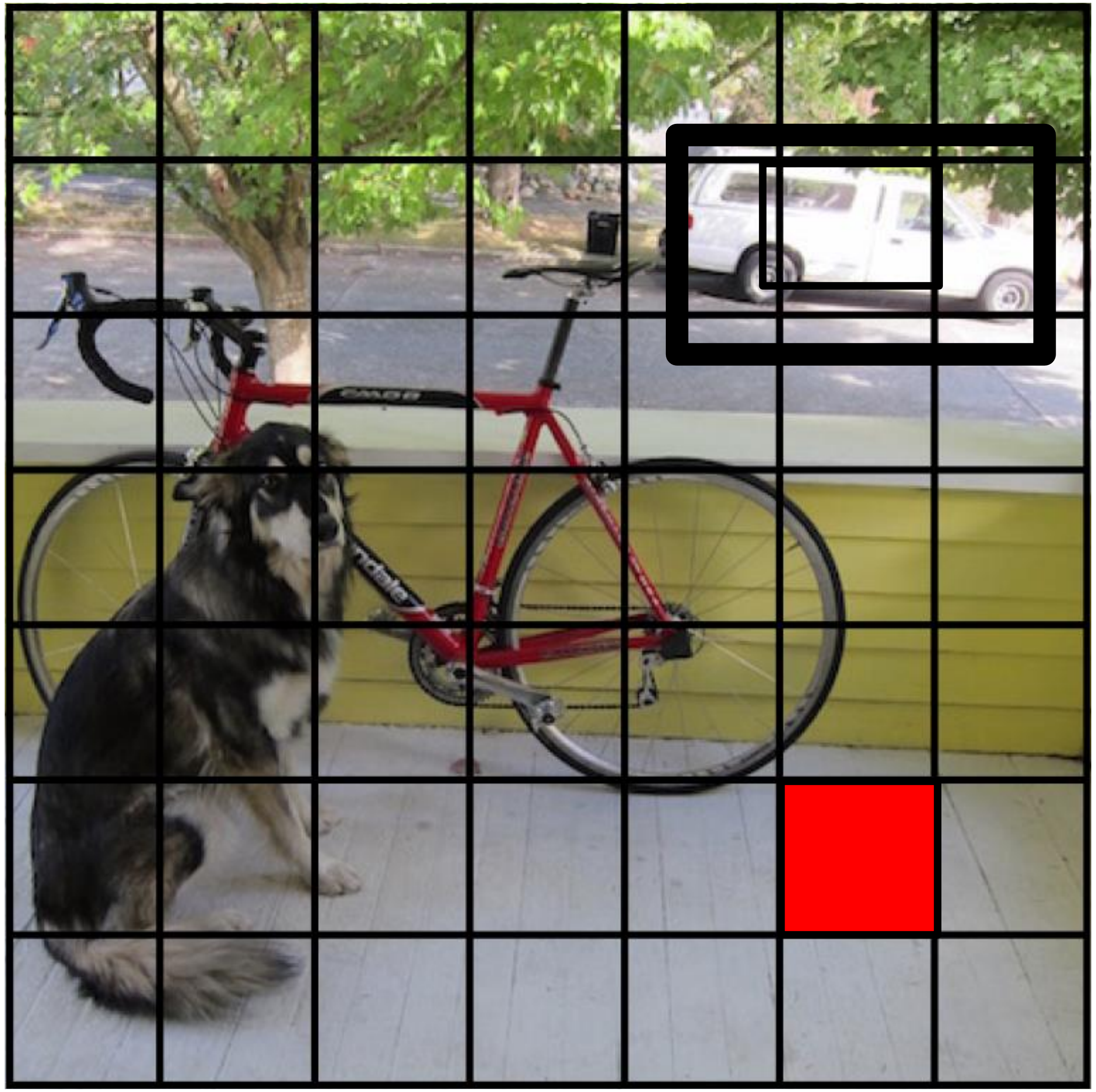


Each cell predicts boxes and confidences: P(Object)

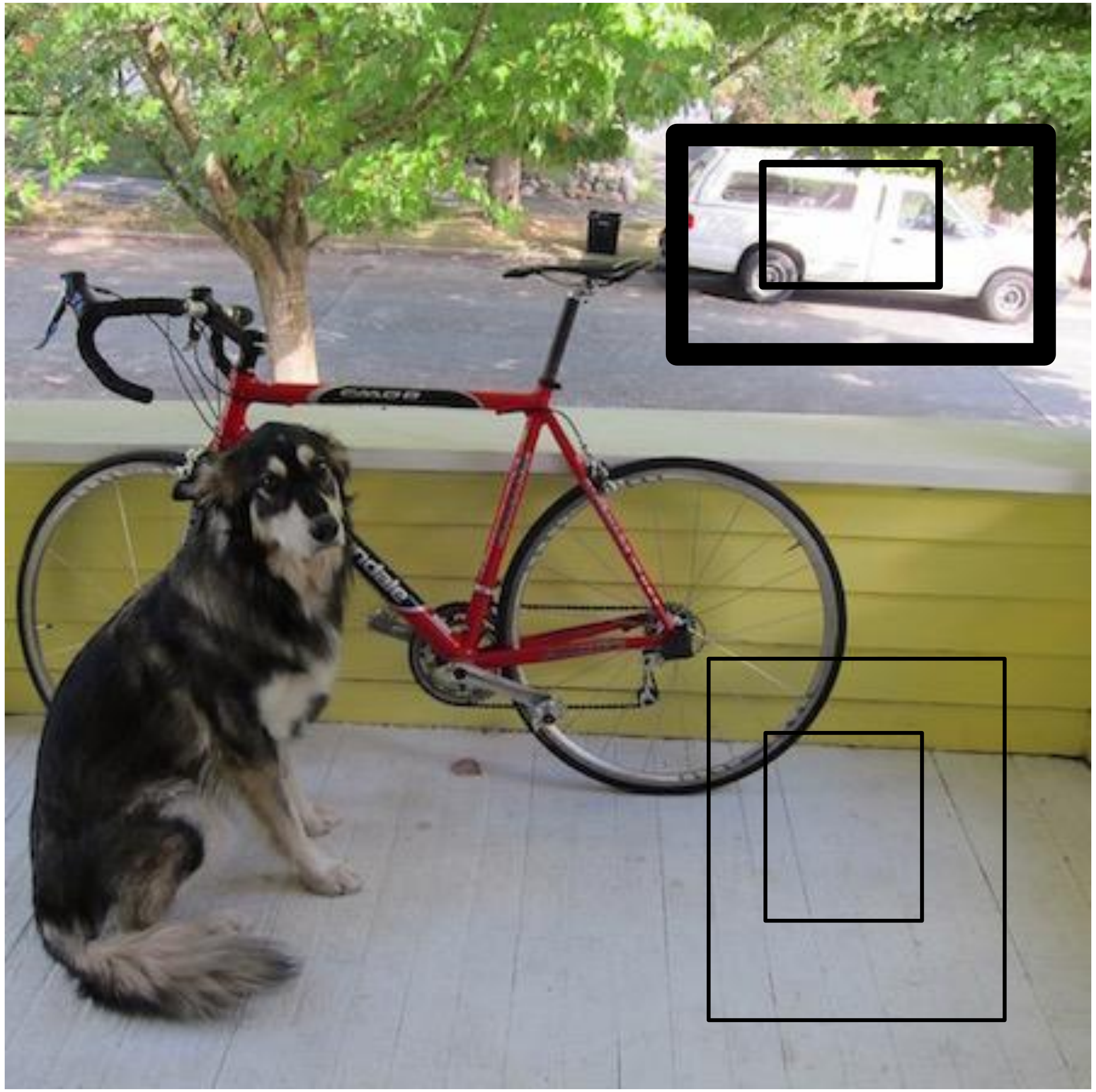
Each cell predicts boxes and confidences: P(Object)



Each cell predicts boxes and confidences: P(Object)

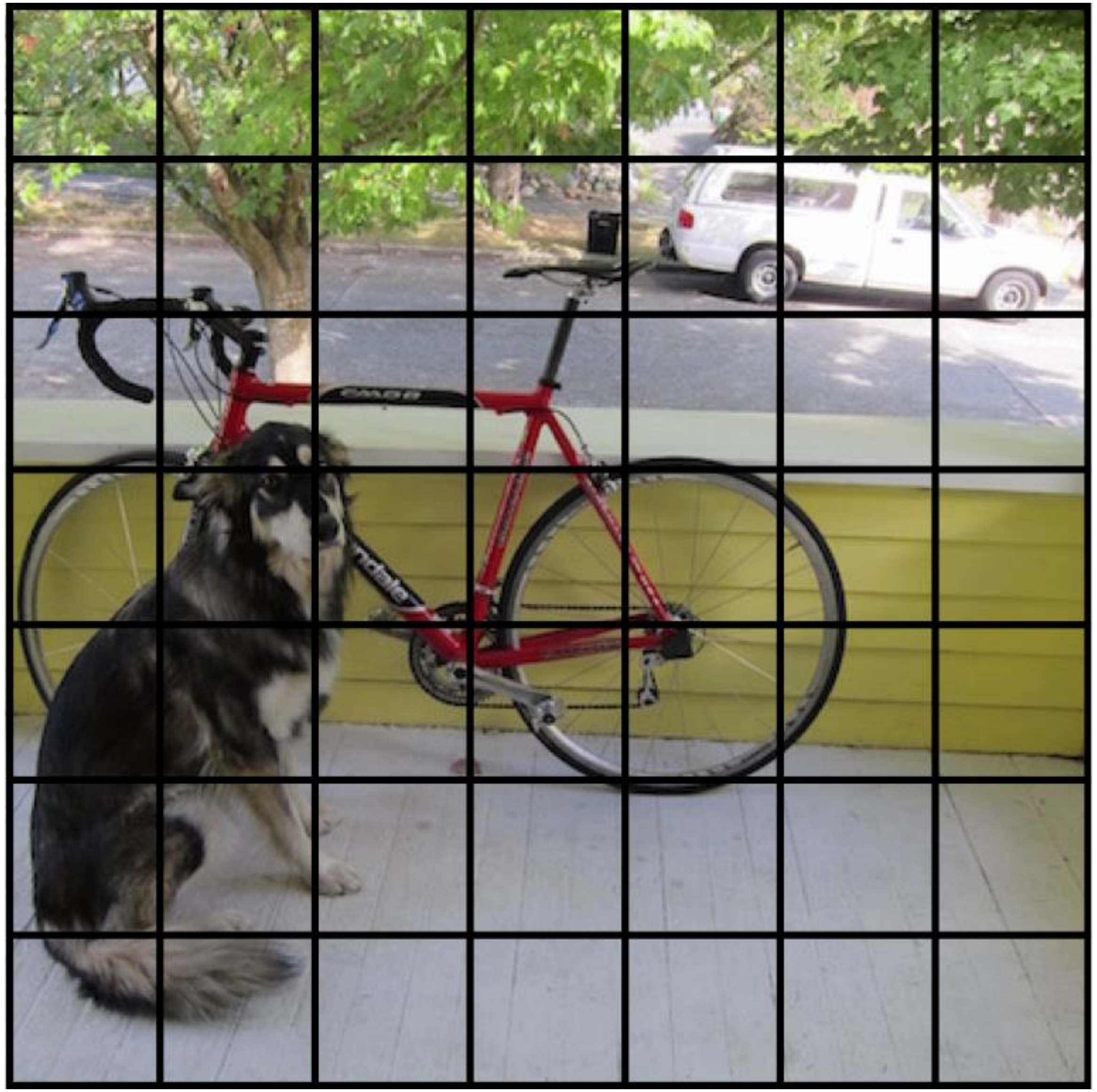


Each cell predicts boxes and confidences: P(Object)

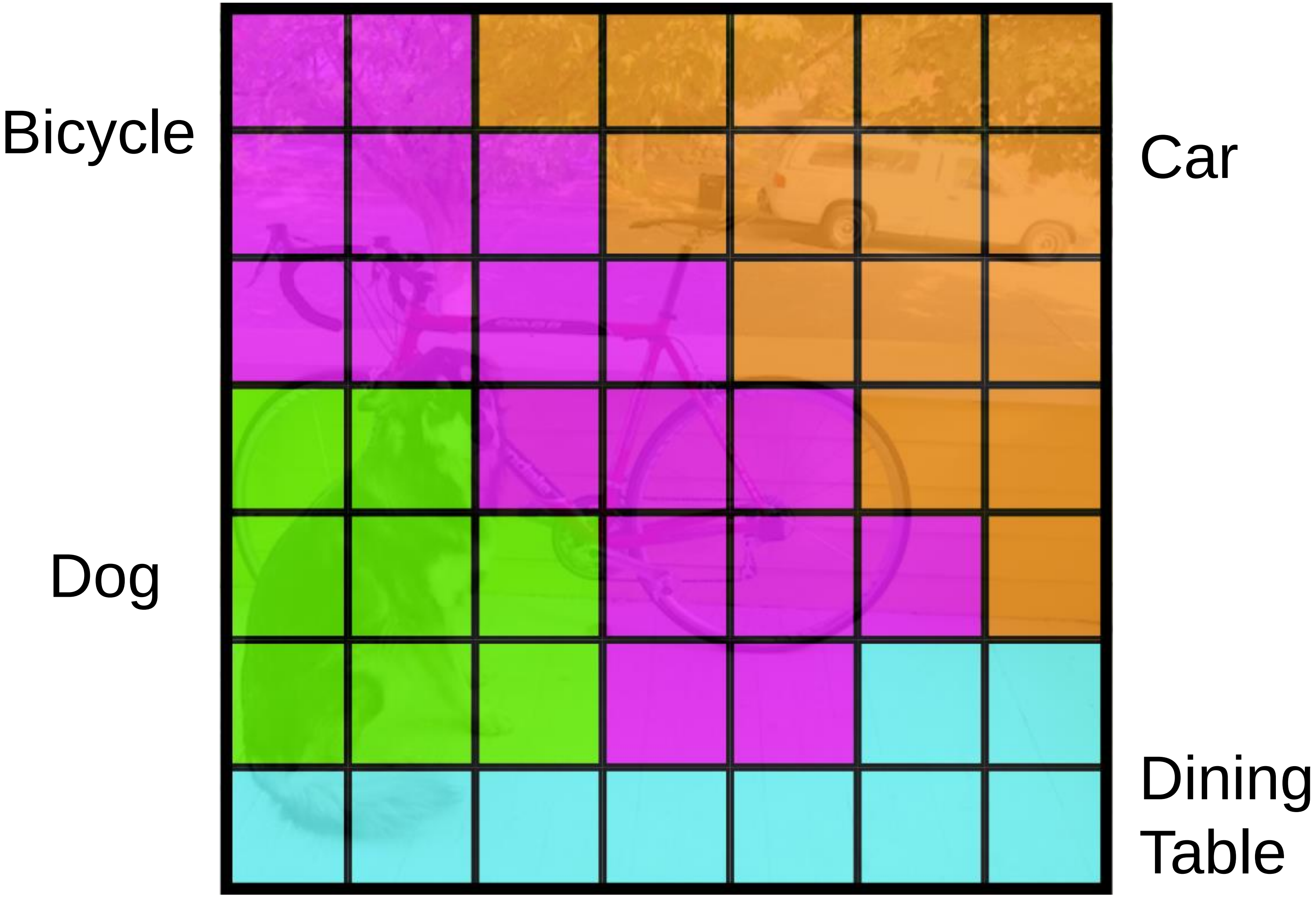


Each cell predicts boxes and confidences: P(Object)

Each cell also predicts a class probability.



Each cell also predicts a class probability.



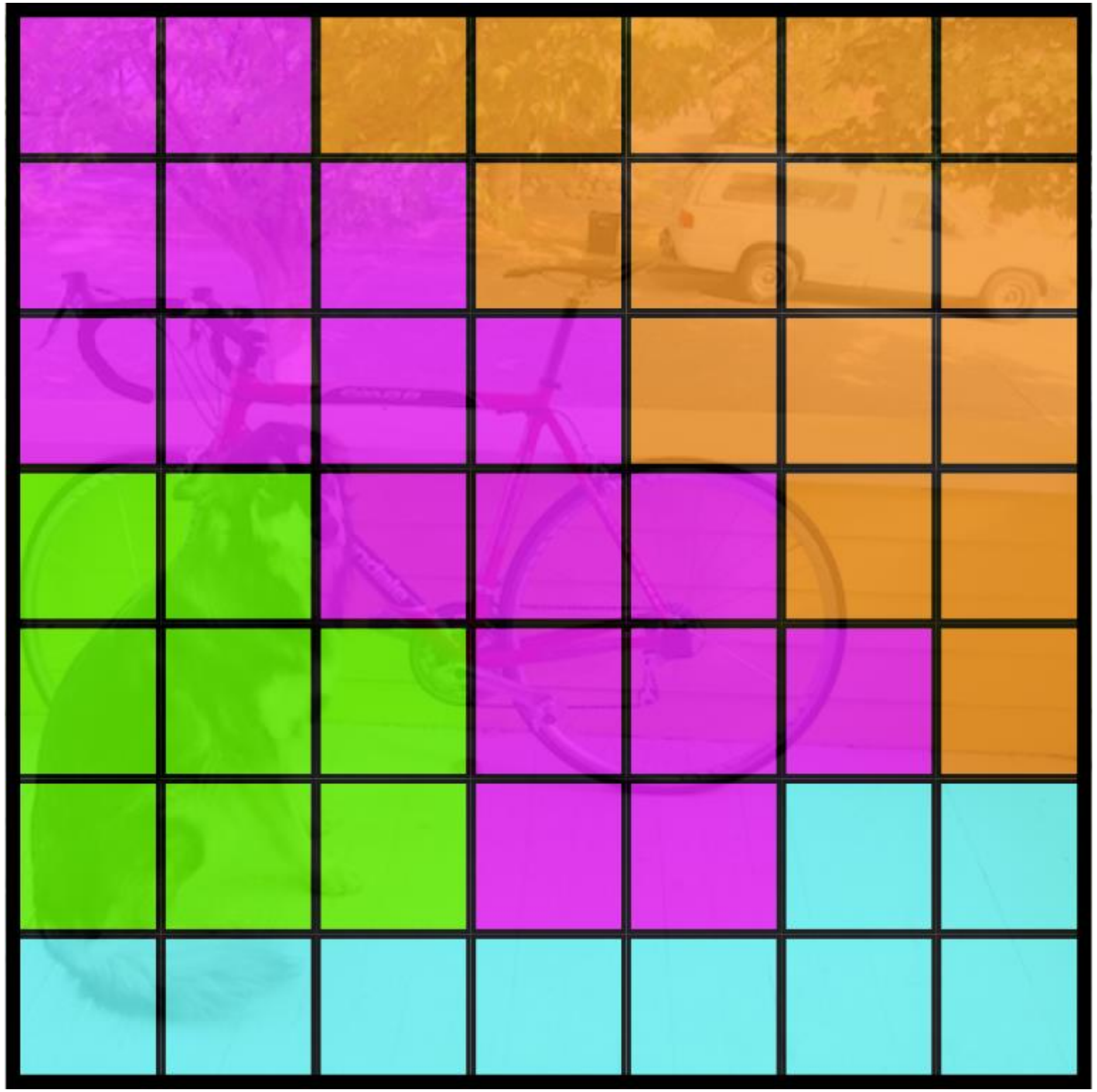
Conditioned on object: P(Car | Object)

Bicycle

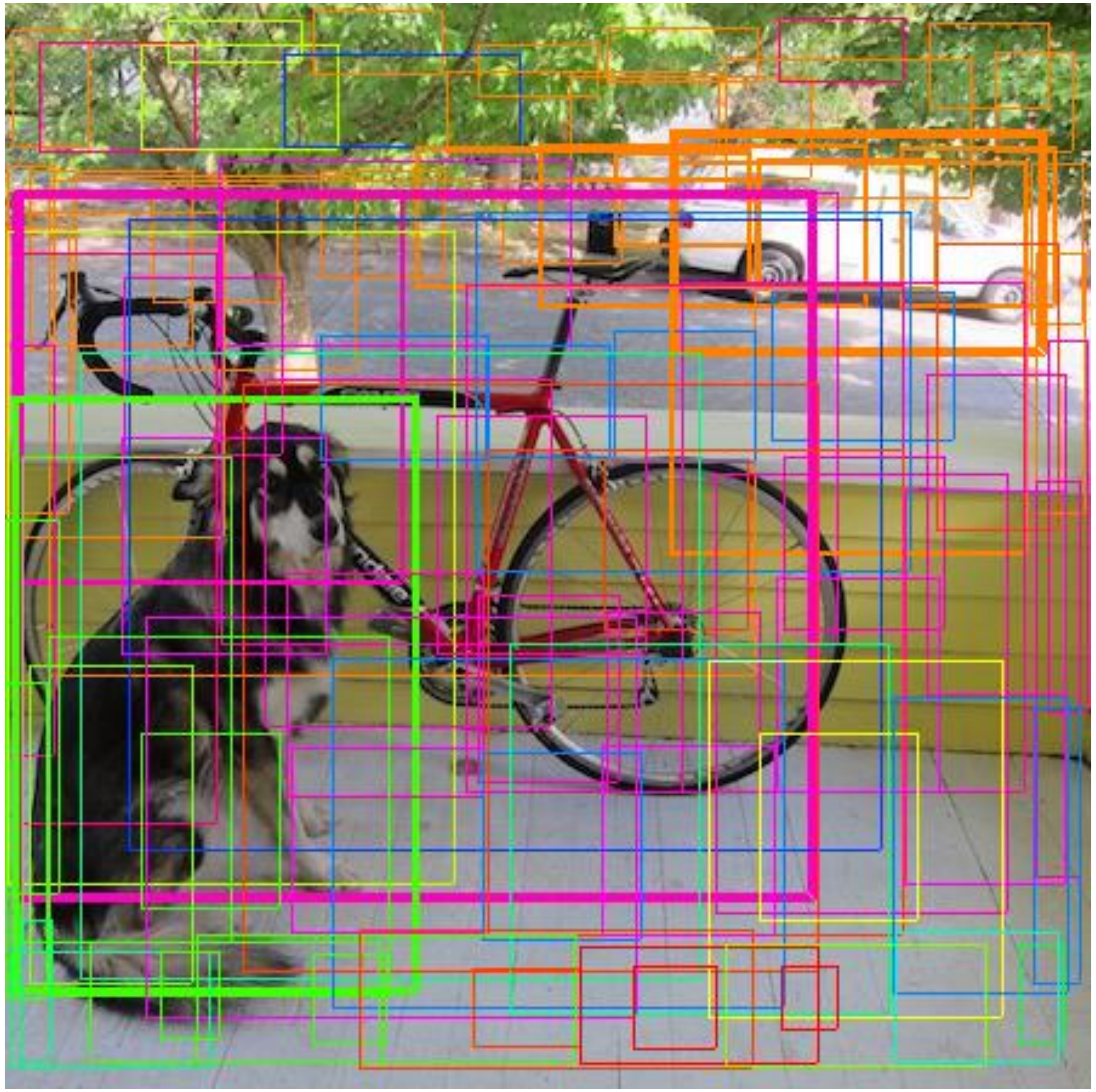
Car

Dog

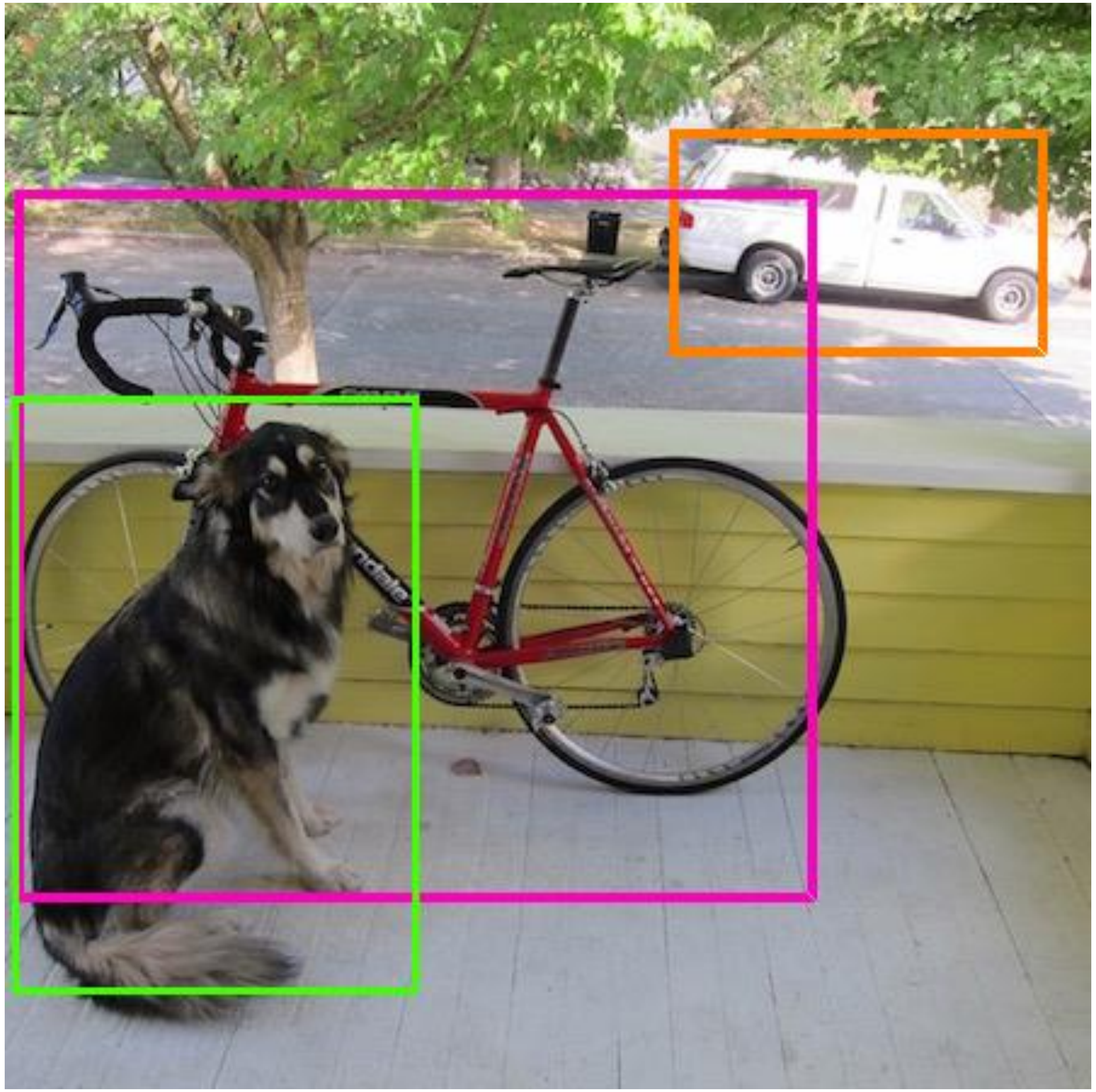
Dining Table



Then we combine the box and class predictions.



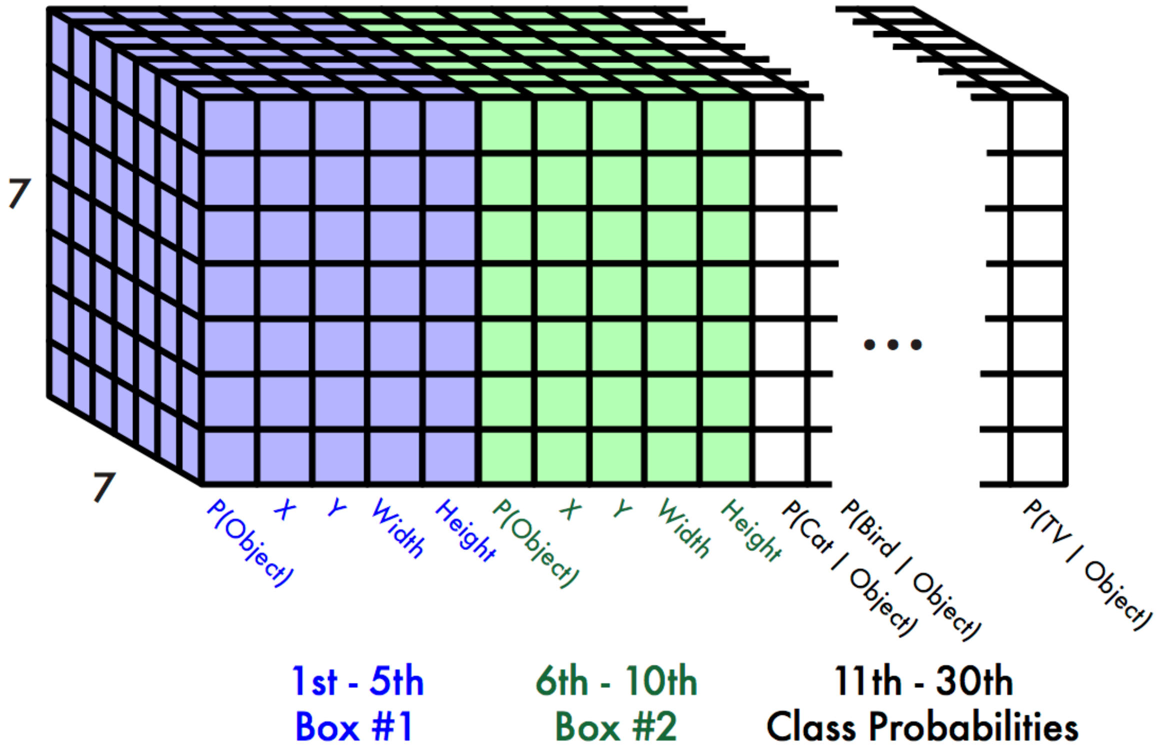
Finally we do NMS and threshold detections



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

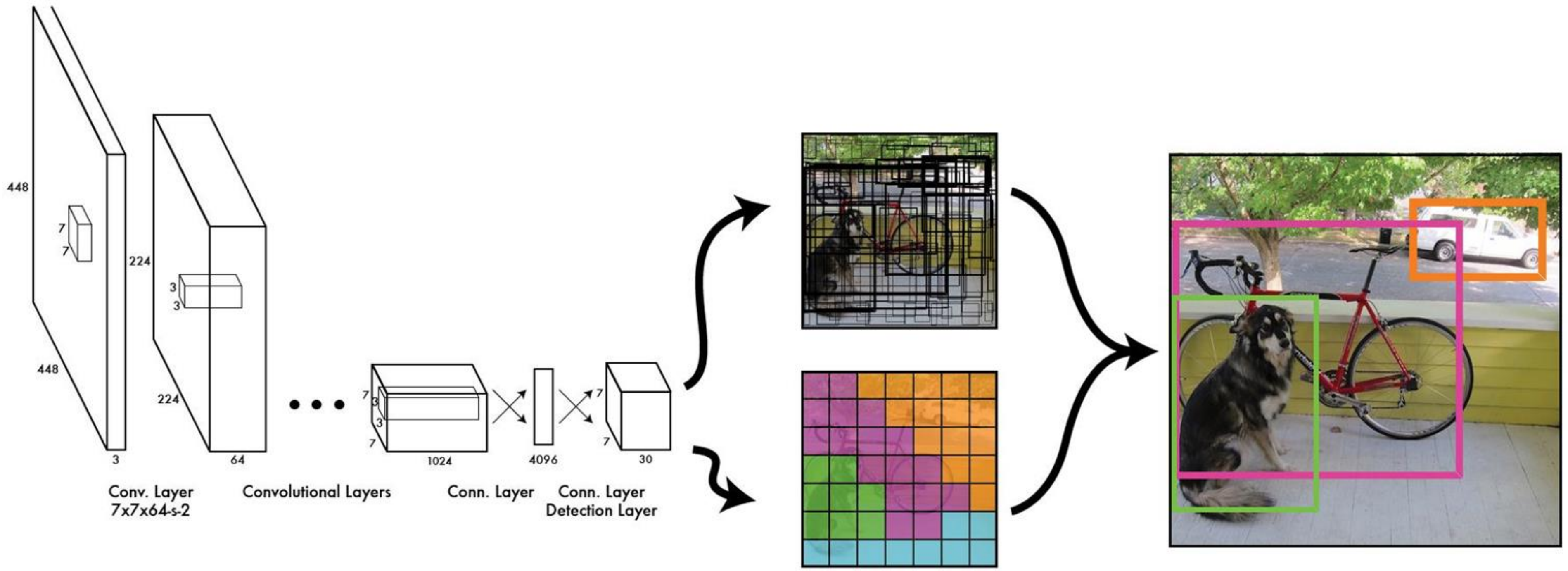


For Pascal VOC:

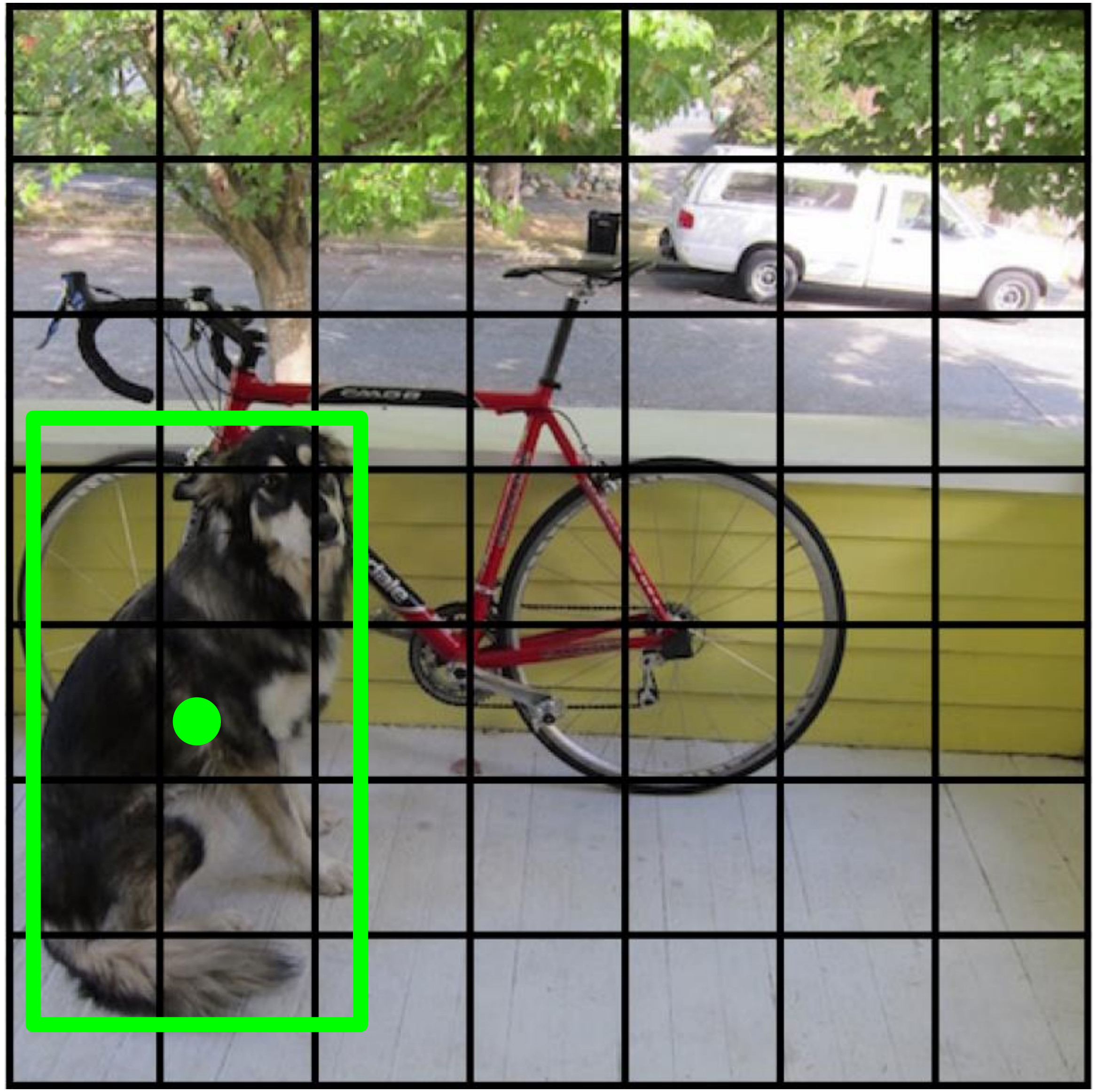
- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

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

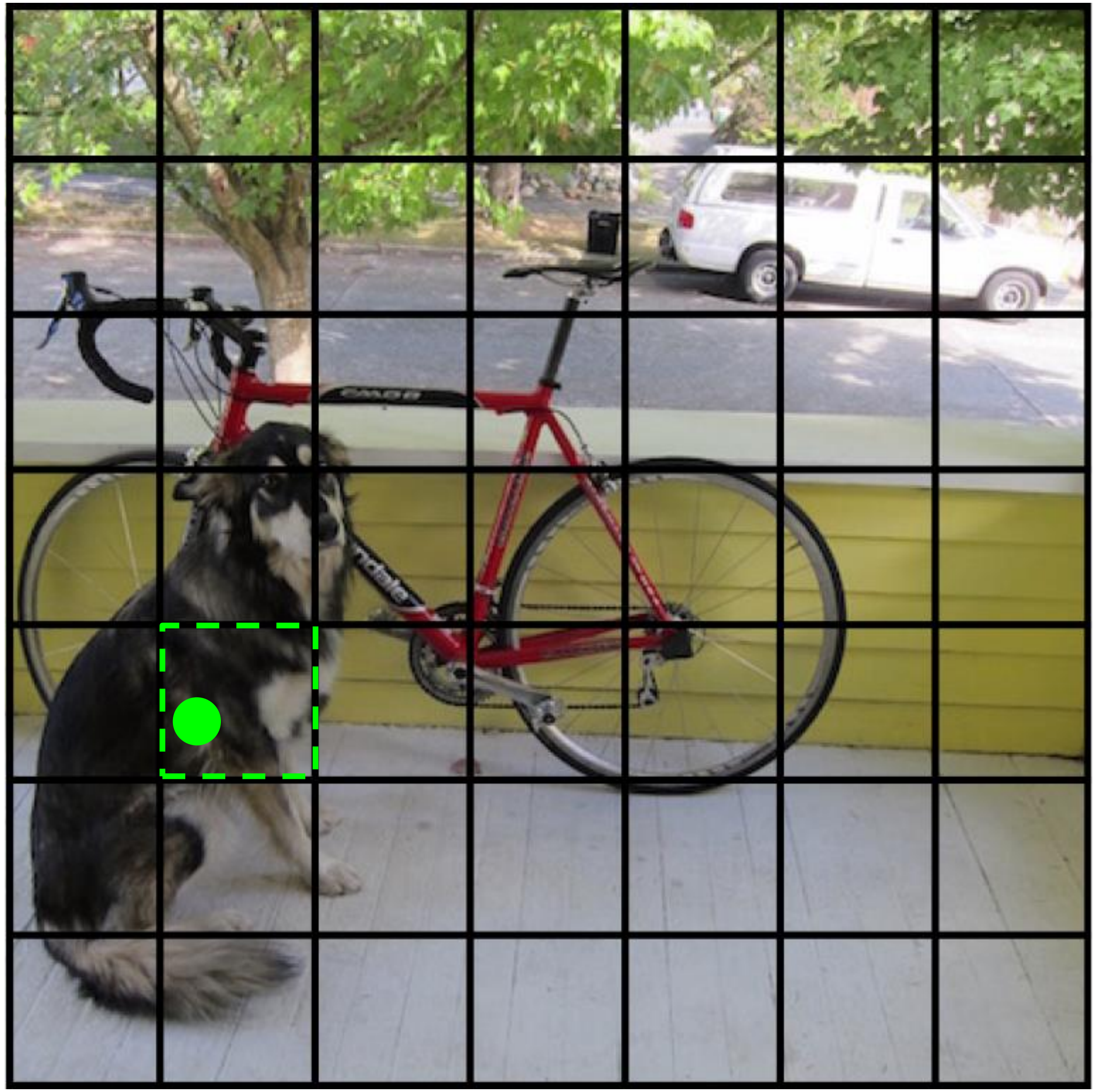
Thus we can train one neural network to be a whole detection



During training, match example to the right cell

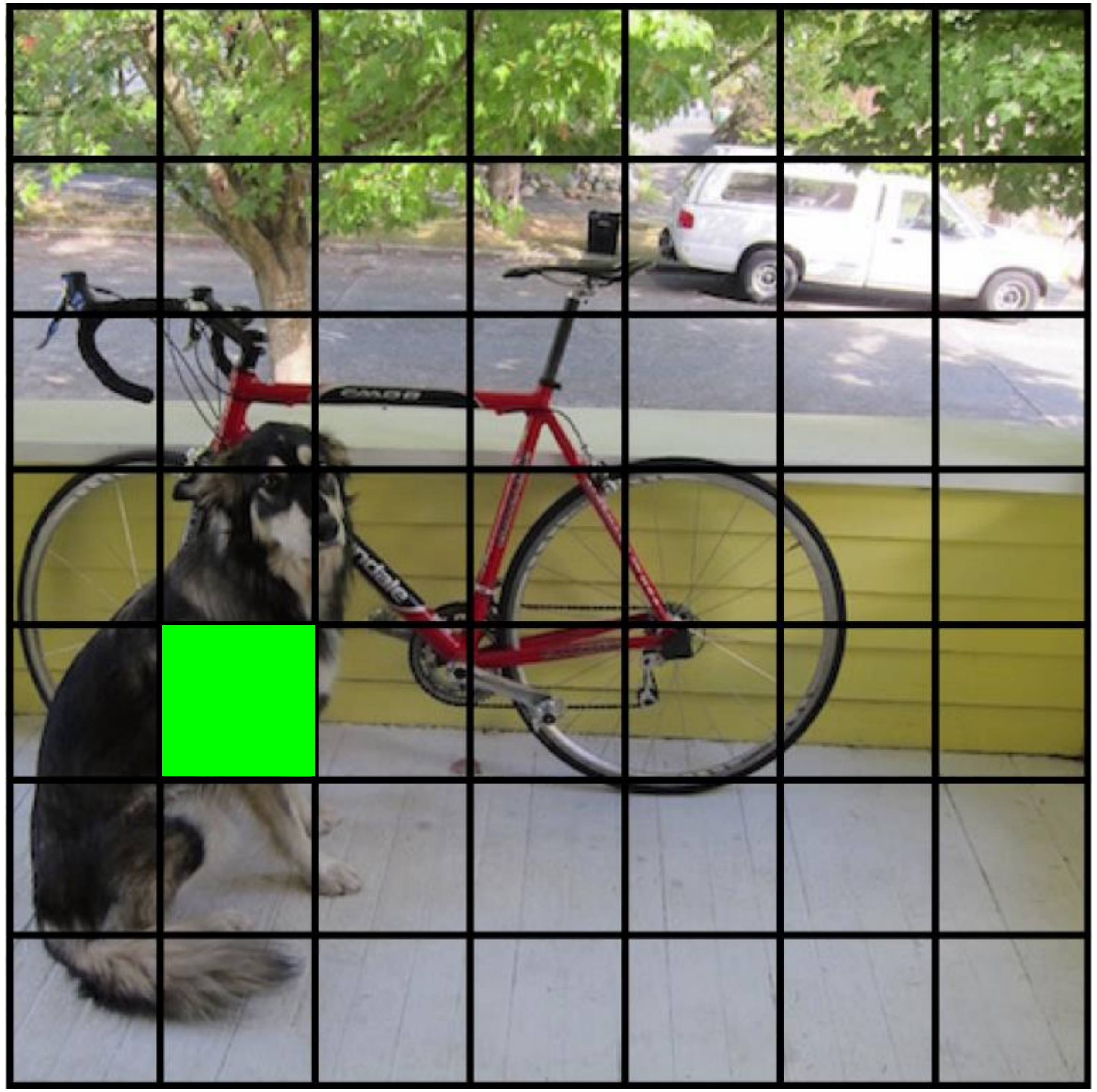


During training, match example to the right cell

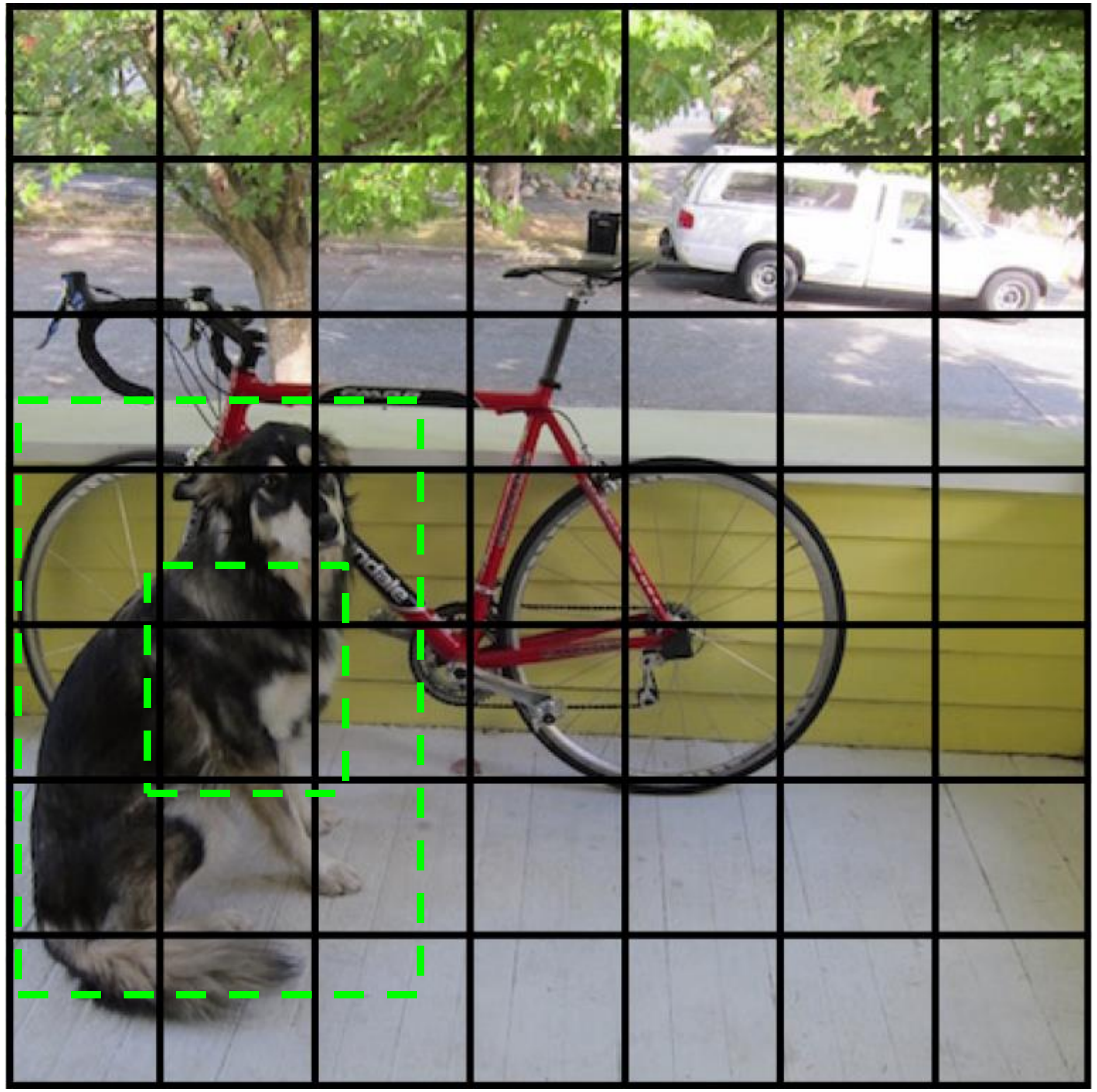


Adjust that cell's class prediction

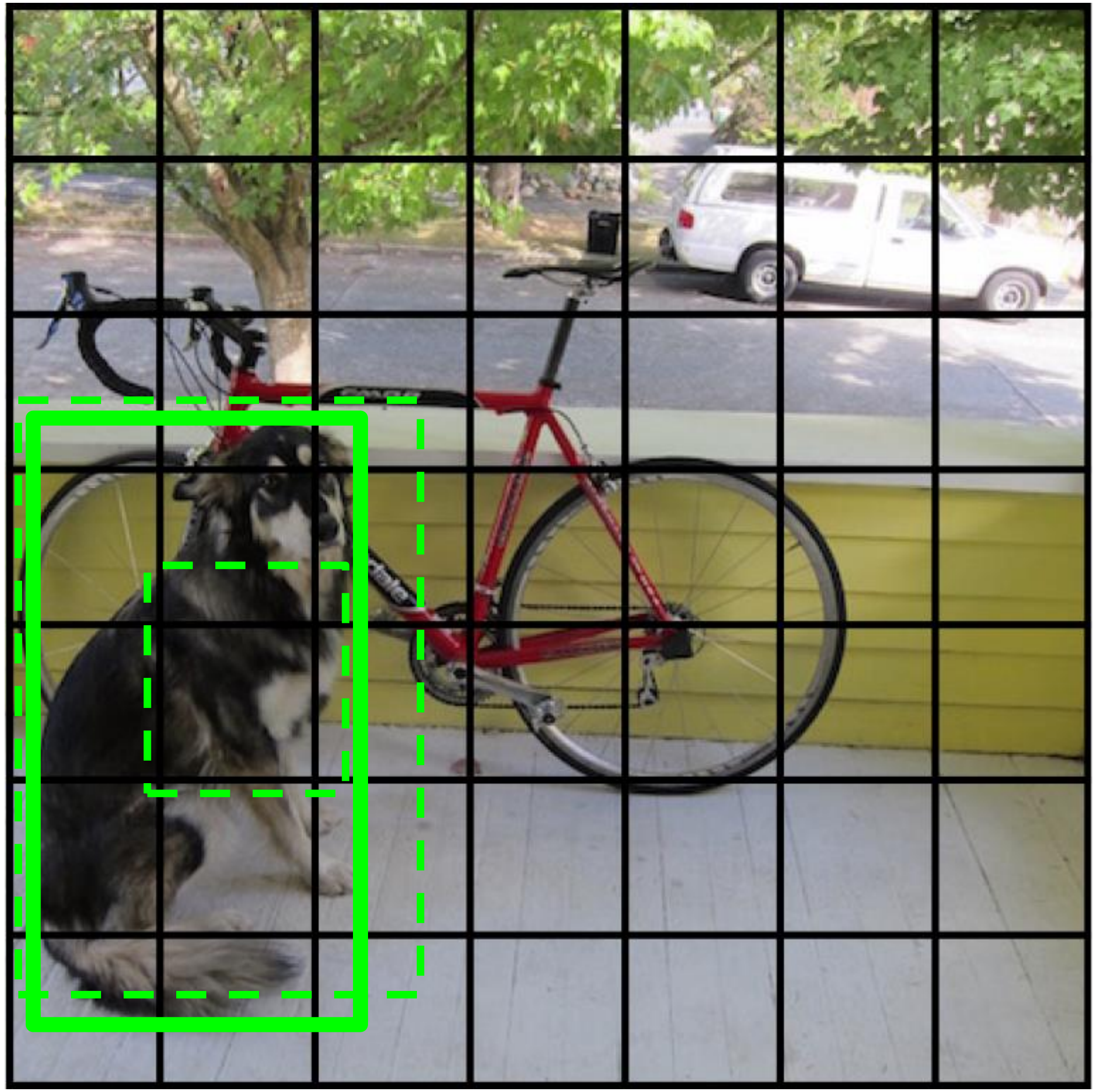
Dog = 1
Cat = 0
Bike = 0
...



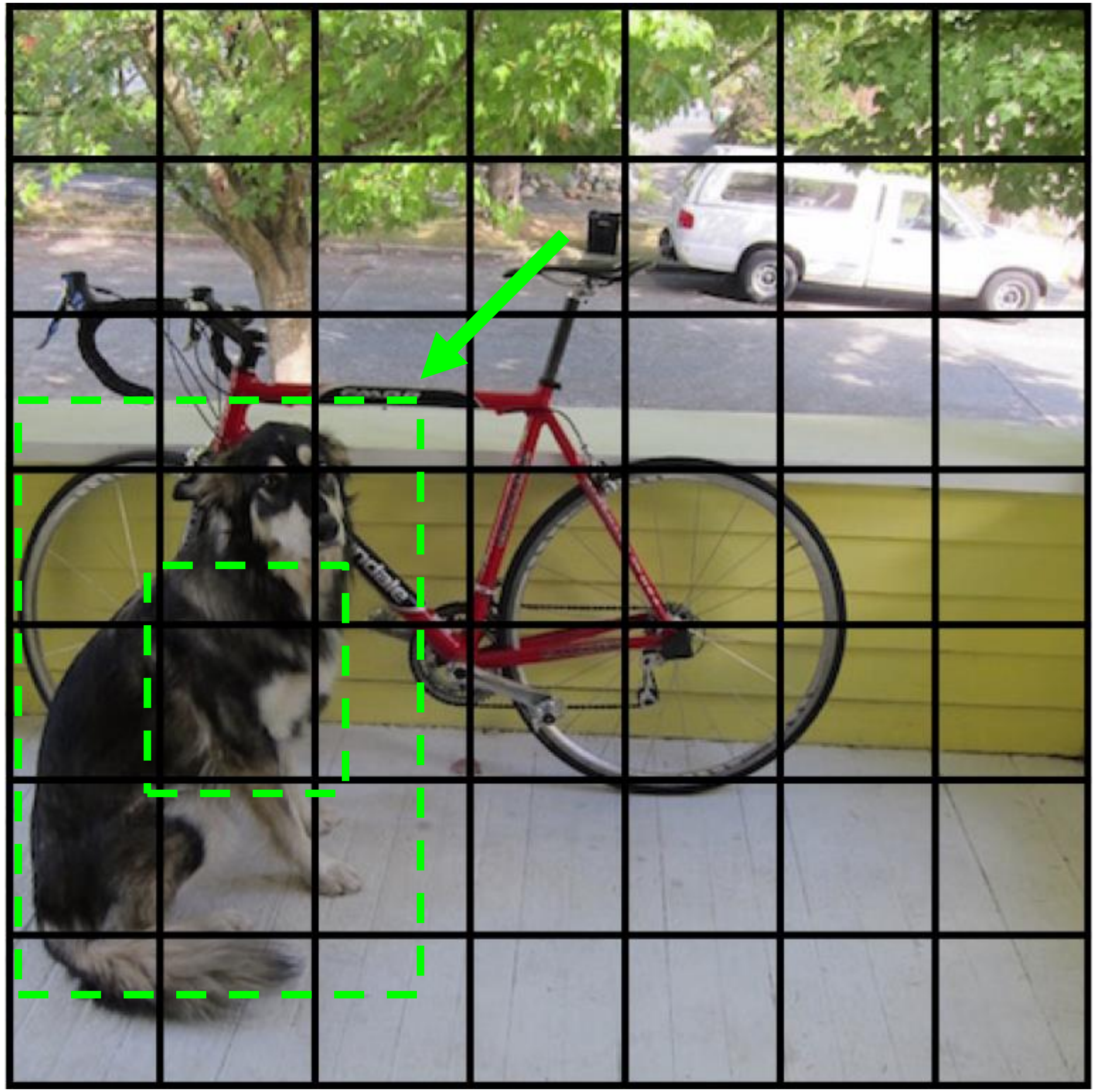
Look at that cell's predicted boxes



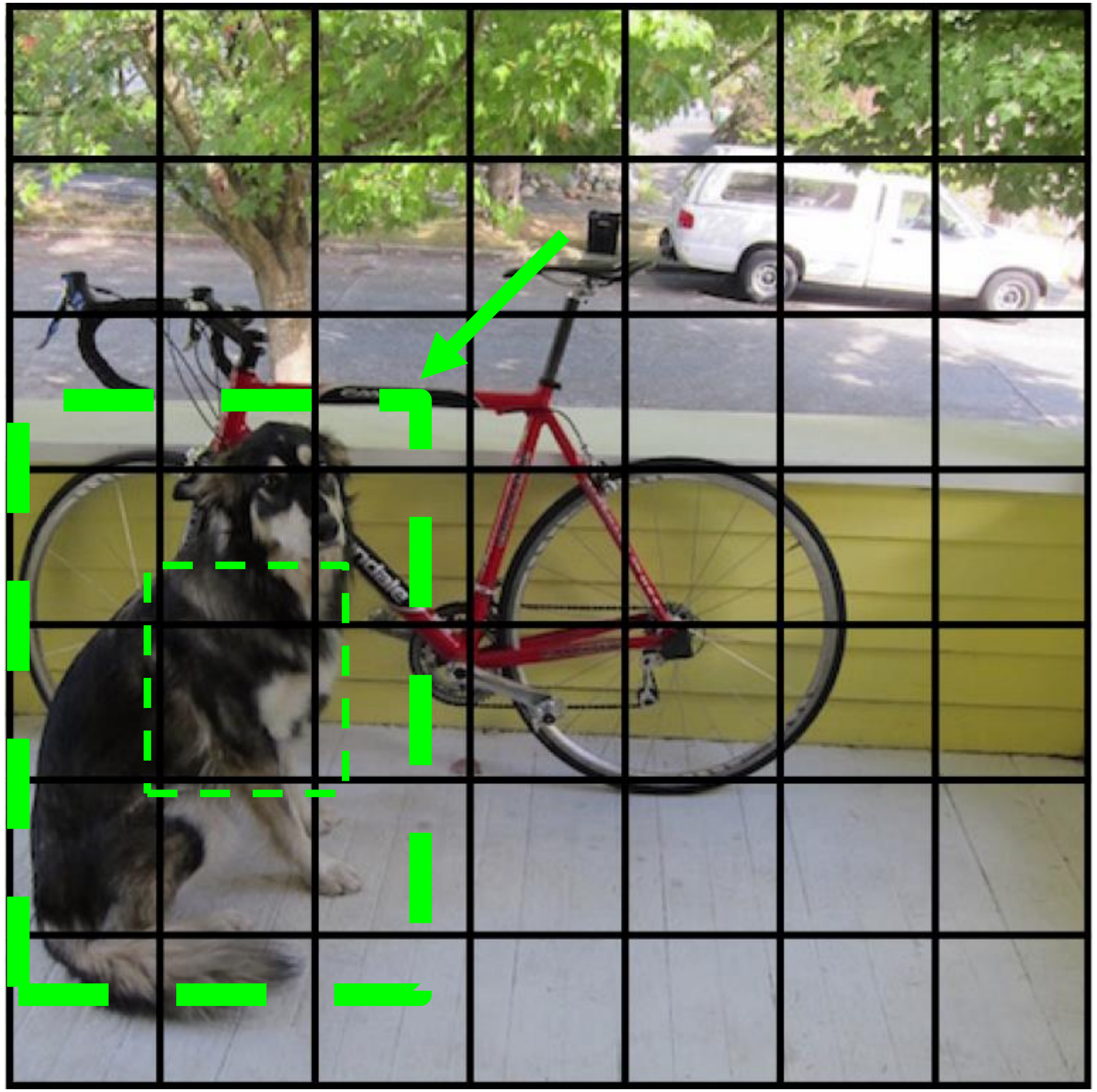
Find the best one, adjust it, increase the confidence



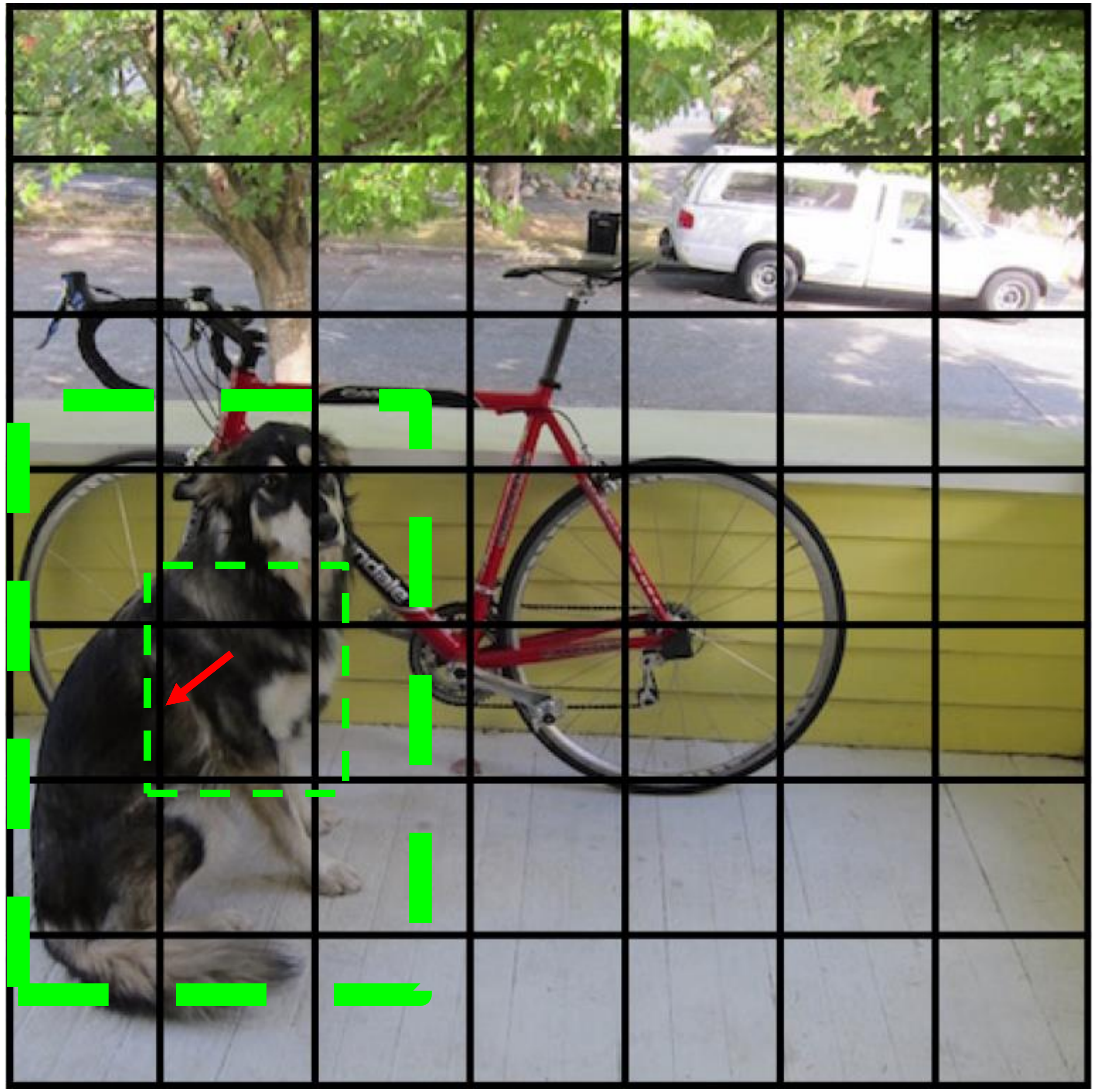
Find the best one, adjust it, increase the confidence



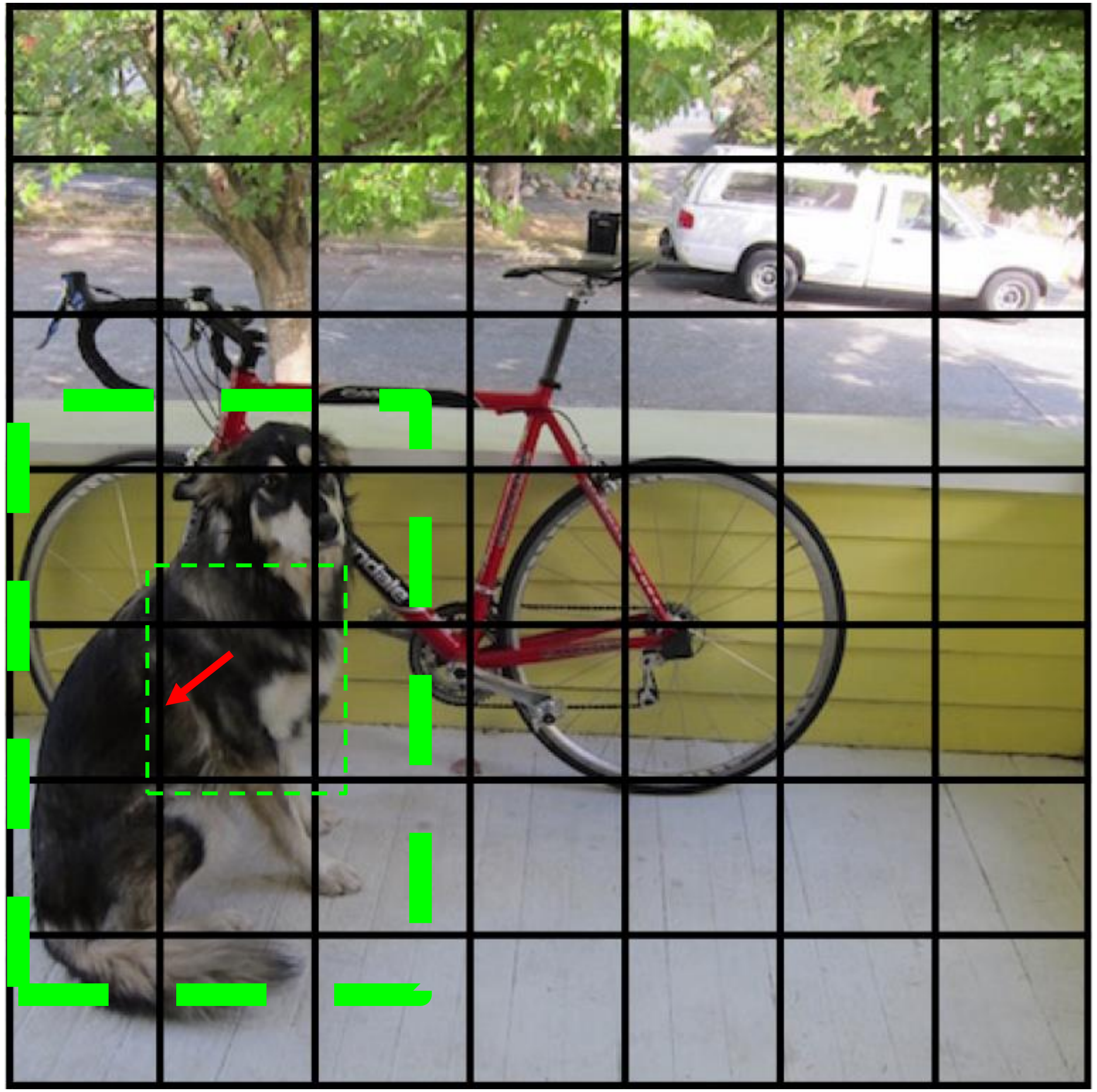
Find the best one, adjust it, increase the confidence



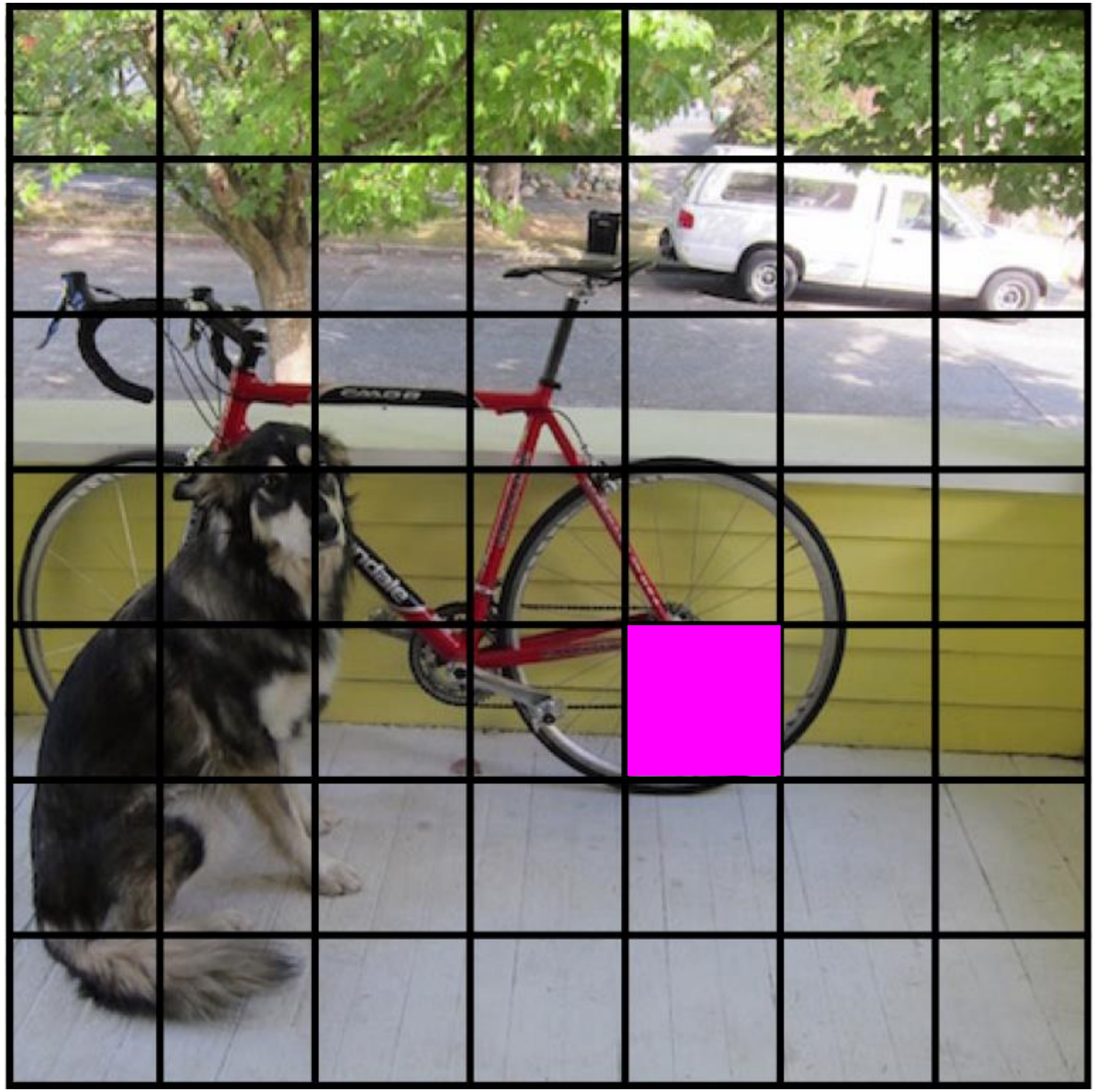
Decrease the confidence of other boxes



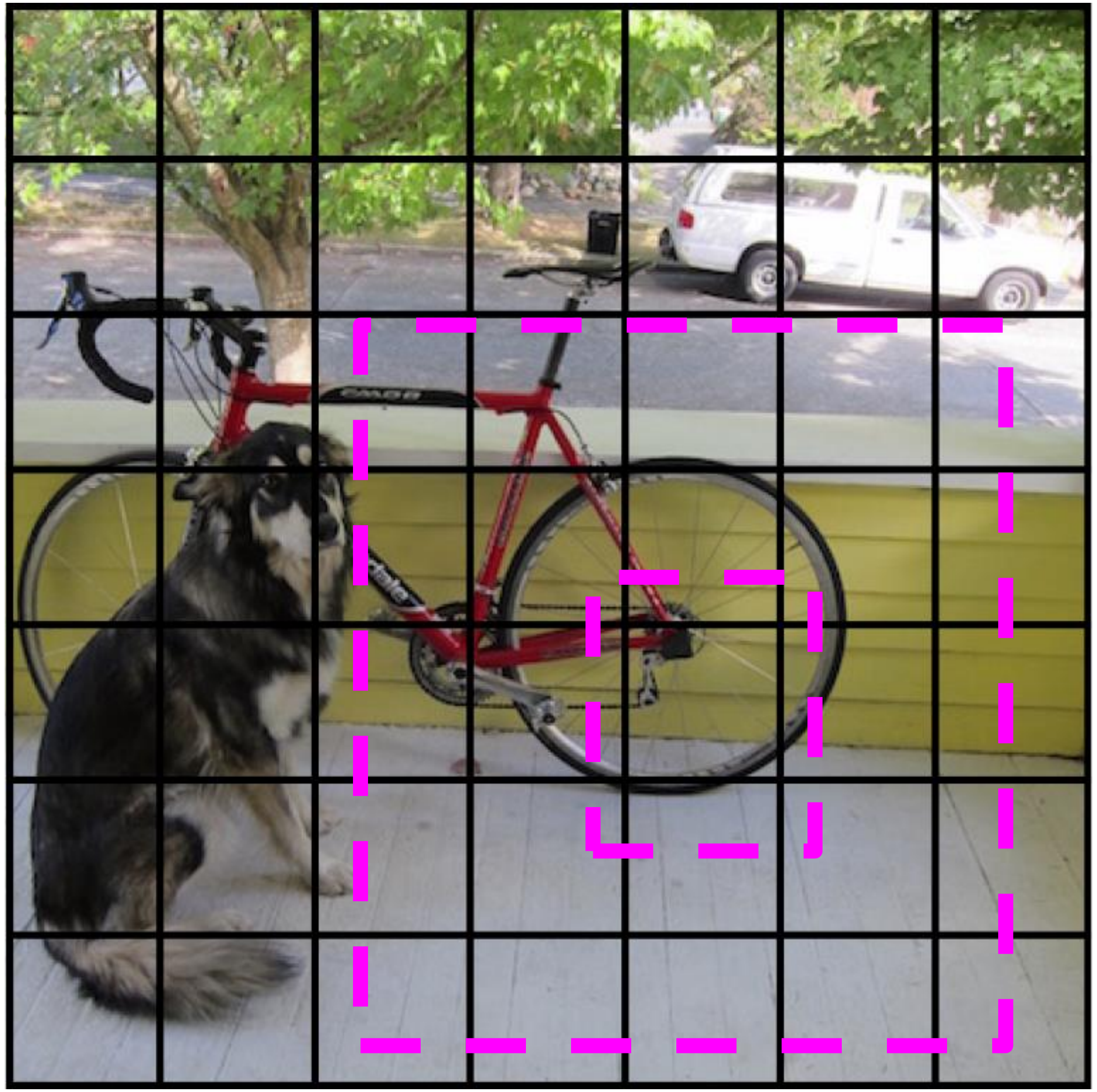
Decrease the confidence of other boxes



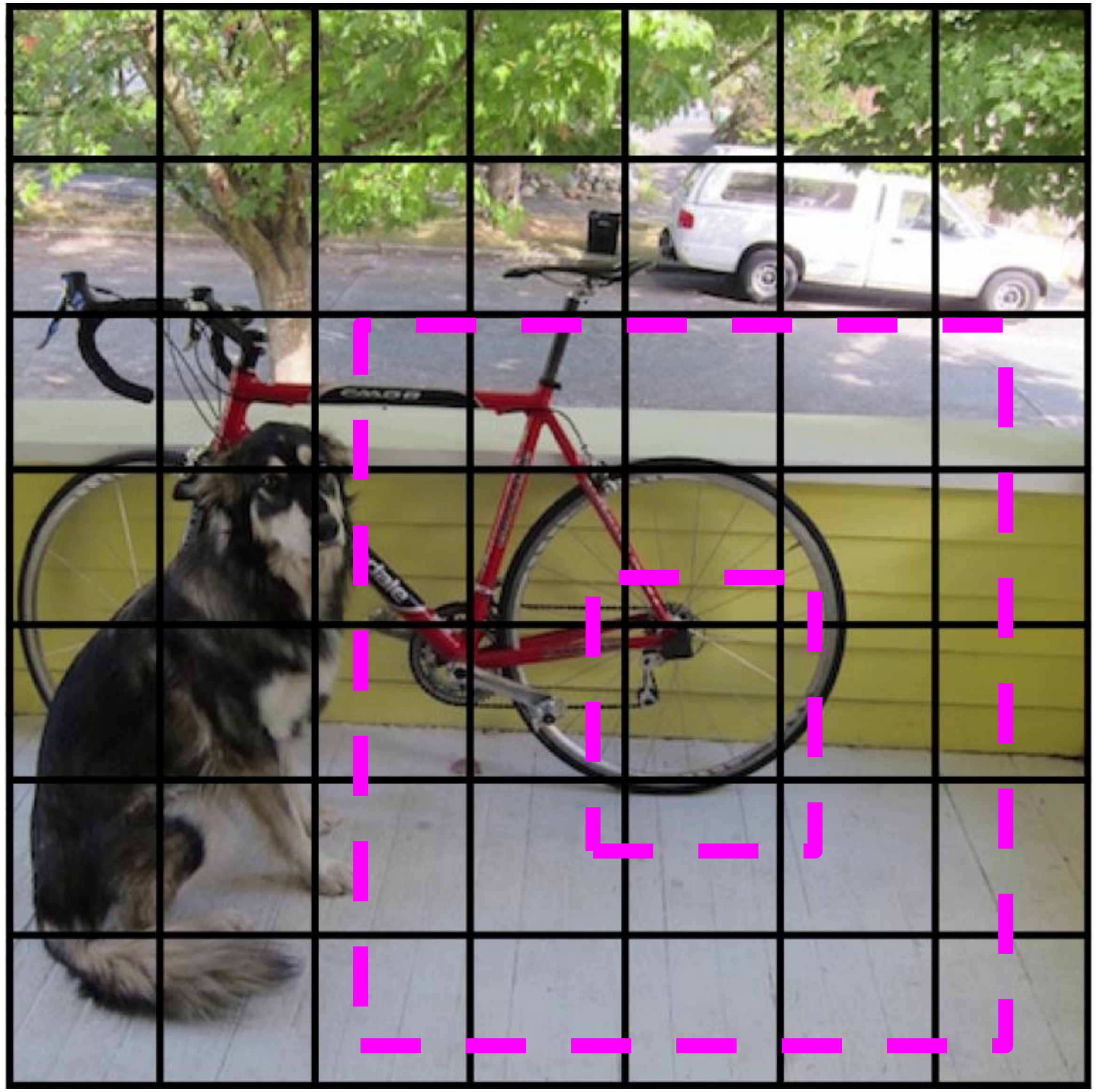
Some cells don’t have any ground truth detections!



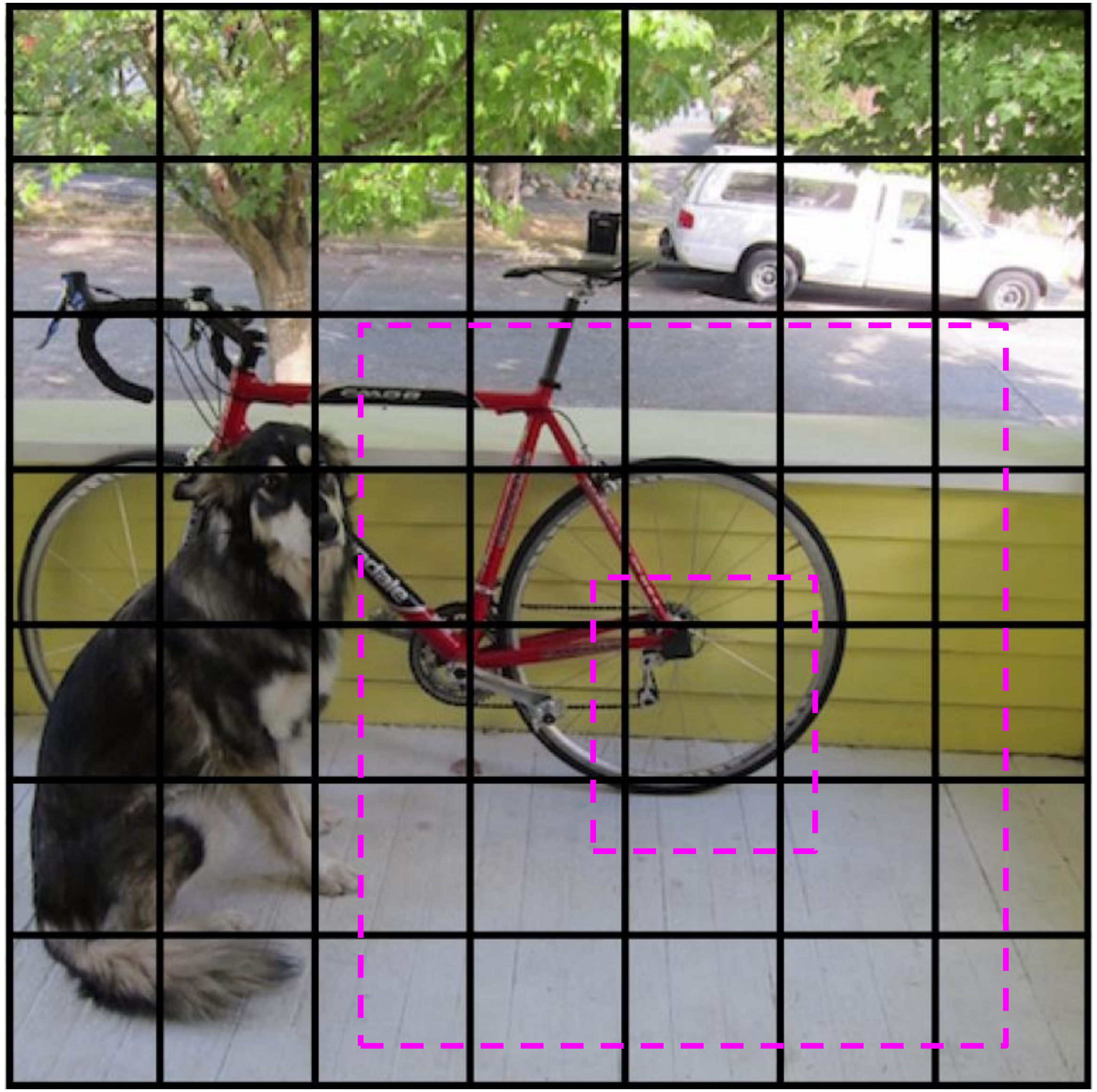
Some cells don’t have any ground truth detections!



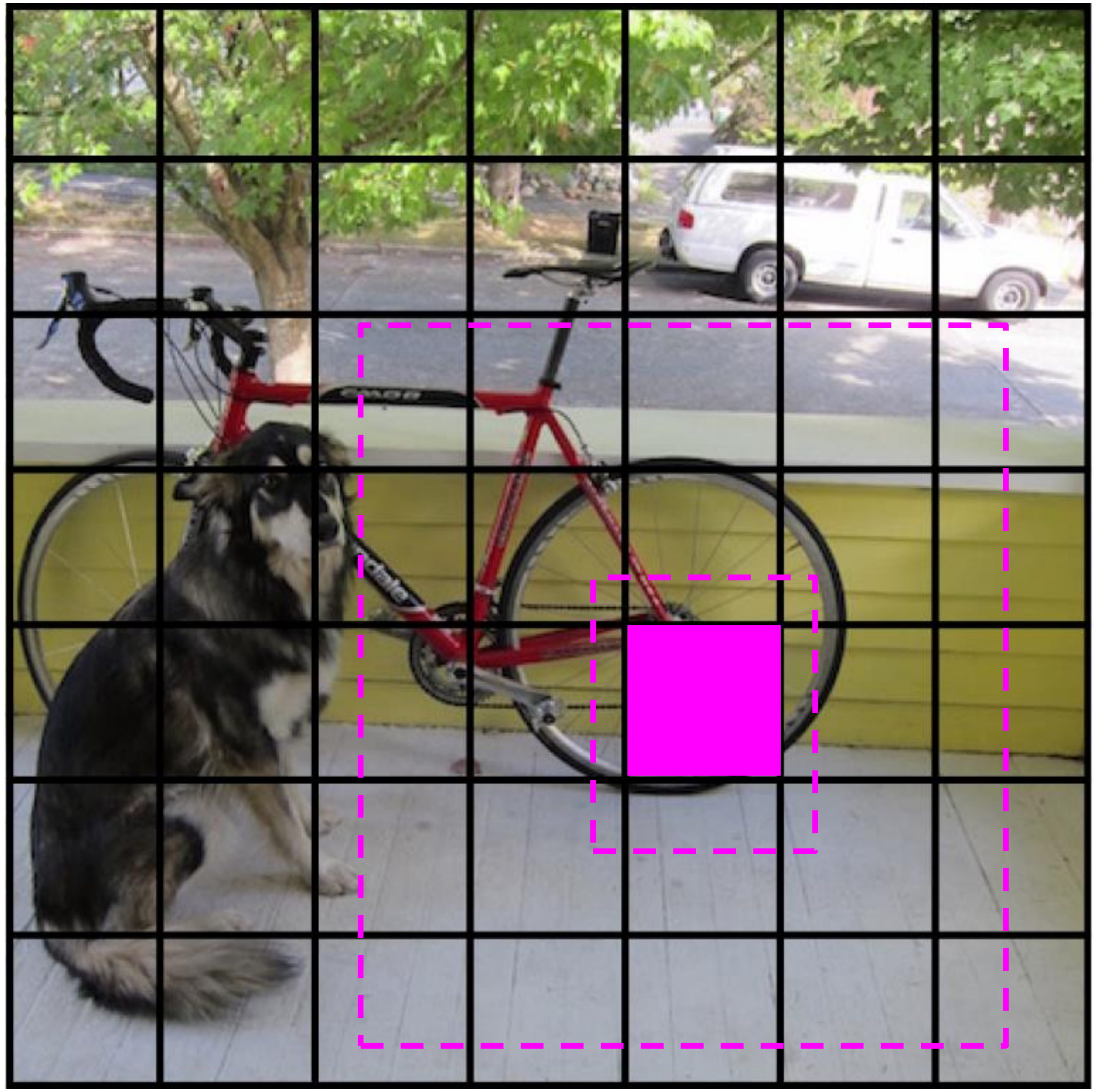
Decrease the confidence of these boxes



Decrease the confidence of these boxes

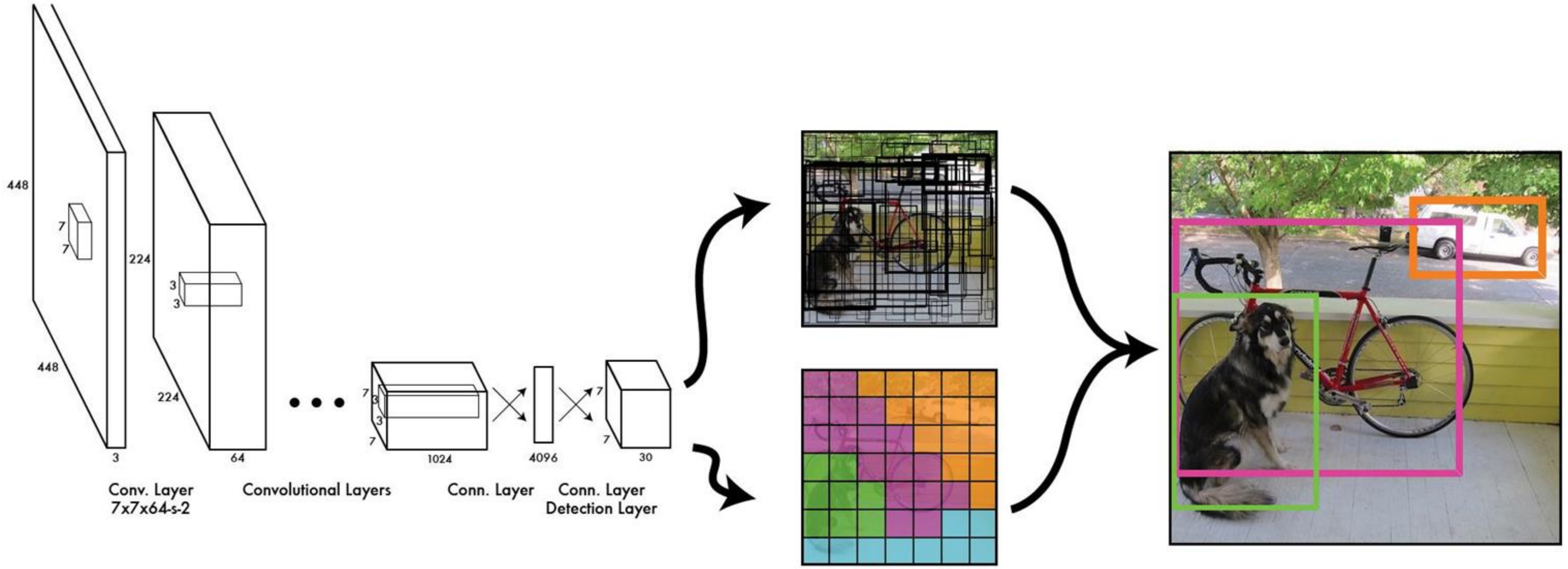


Don’t adjust the class probabilities or coordinates

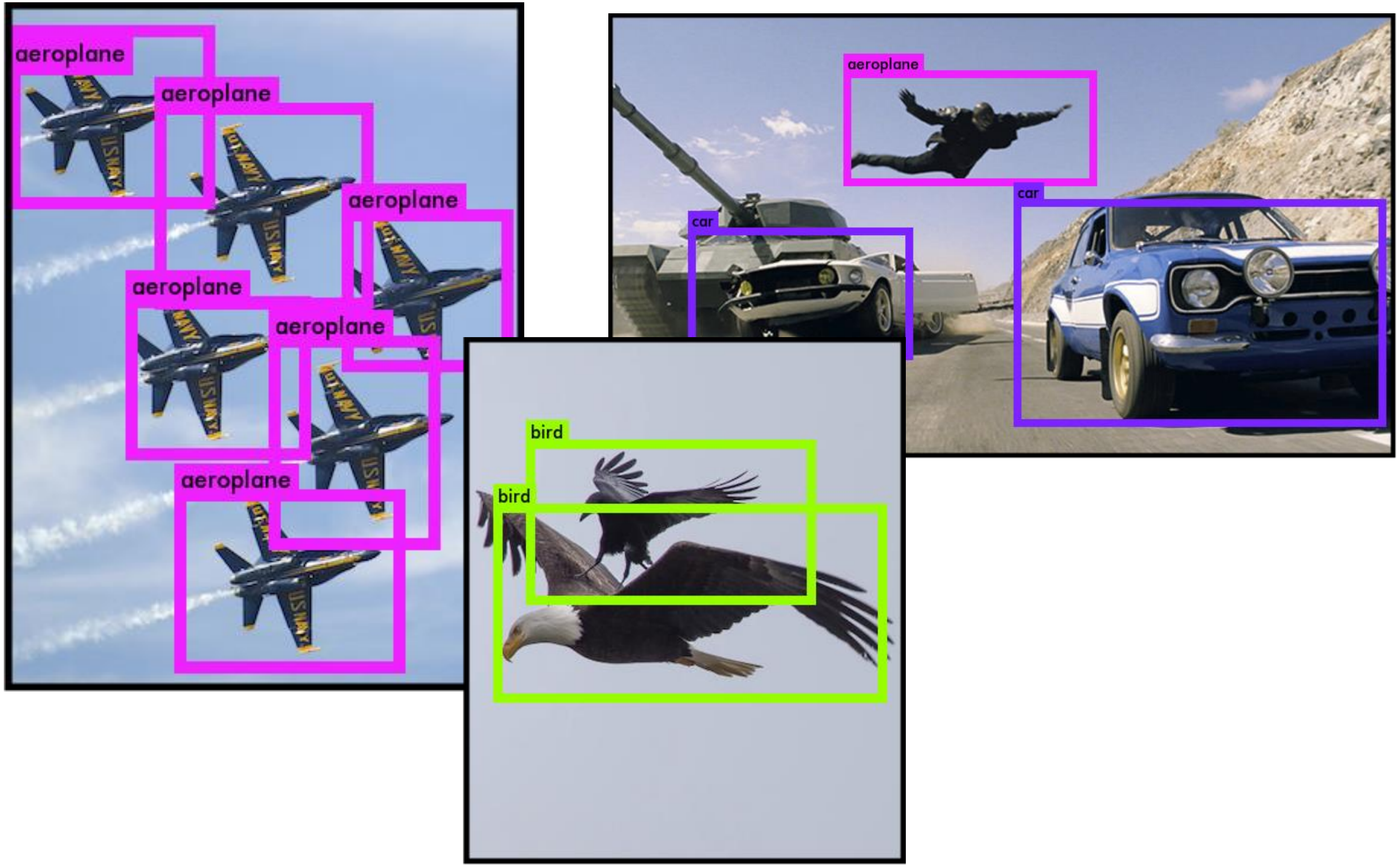


We train with standard tricks:

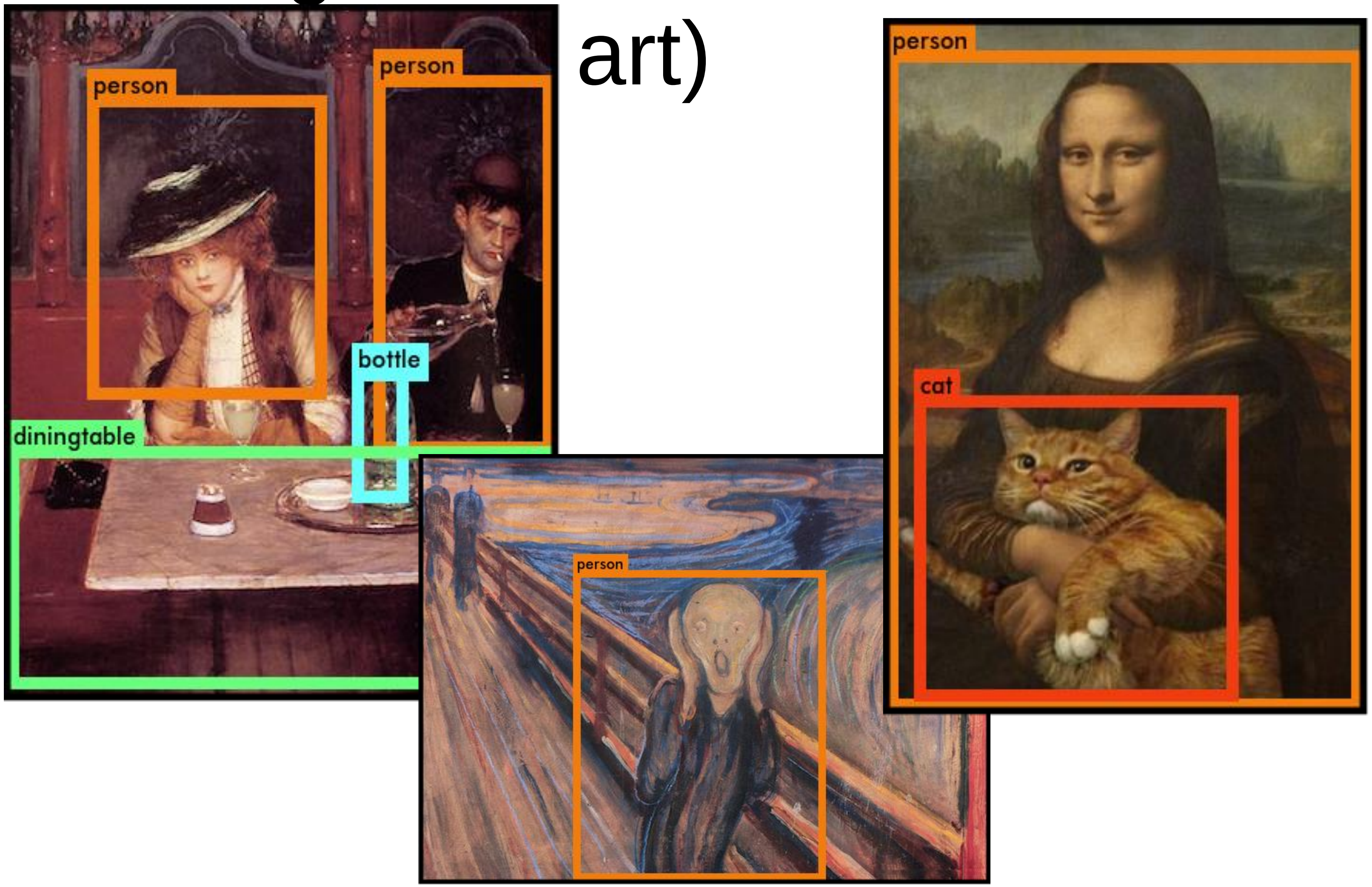
- Pretraining on Imagenet
- SGD with decreasing learning rate
- Extensive data augmentation
- For details, see the paper



YOLO works across a variety of natural images

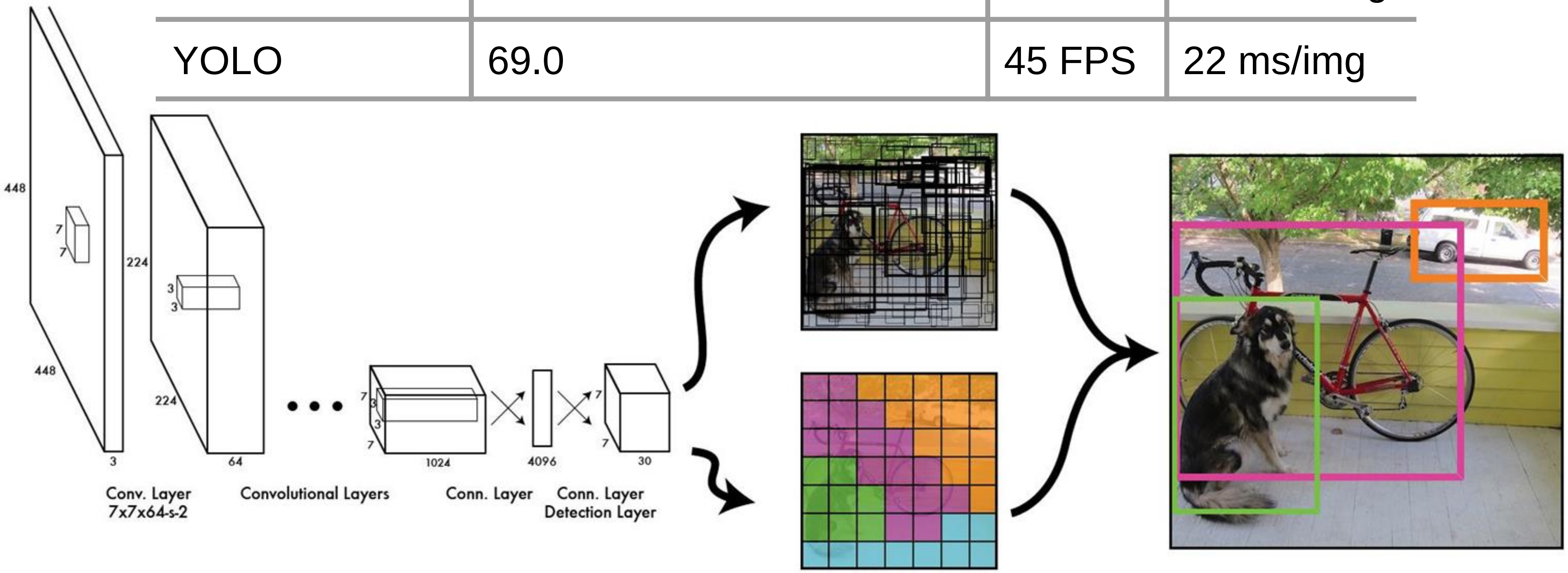


It also generalizes well to new
art)



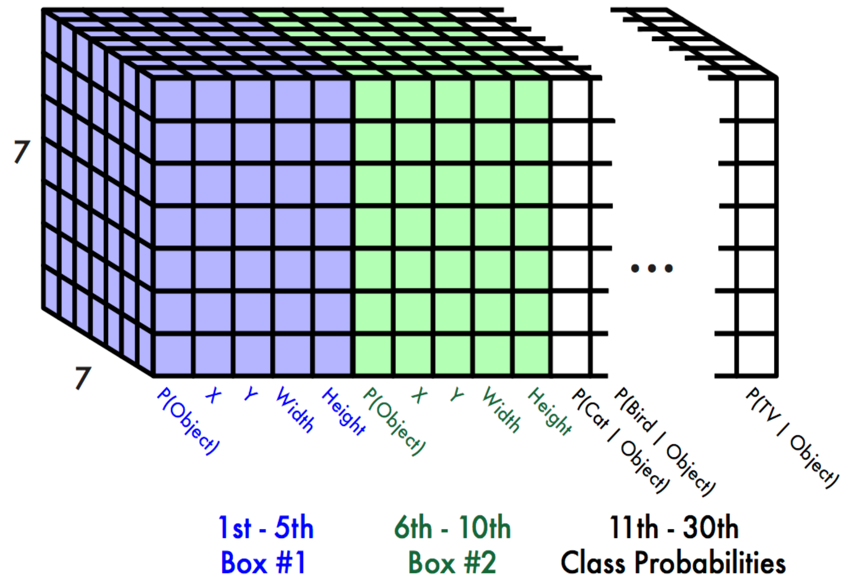
Code available! pjreddie.com/yolo

	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



YOLO (V1) Summary

- Simple way to detect objects. The problem of arbitrary cardinality output with continuous positioning is solved by having a bounded, fixed, output (7x7 grid of outputs, with at most 2 boxes per grid cell)
- Works well on PASCAL VOC. Doesn’t work well on MS COCO or crowded scenes.
- There are a string of works following the original YOLO with various improvements.




JOSEPH REDMON
ALI FARHADI
RETURN IN.....
YOLO9000
Better, Faster, Stronger
musher
malamute
NOW PLAYING IN A DEMO NEAR YOU
IN ASSOCIATION WITH XNOR.AI AND THE ALLEN INSTITUTE FOR ARTIFICIAL INTELLIGENCE
MODELS BY DARKNET: OPEN SOURCE NEURAL NETWORKS
@DARKNETFOREVER #YOLO9000
pjreddie.com/yolo
AI2
XNOR.AI
PAUL G. ALLEN SCHOOL
W

	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	63.4	45 FPS	22 ms/img

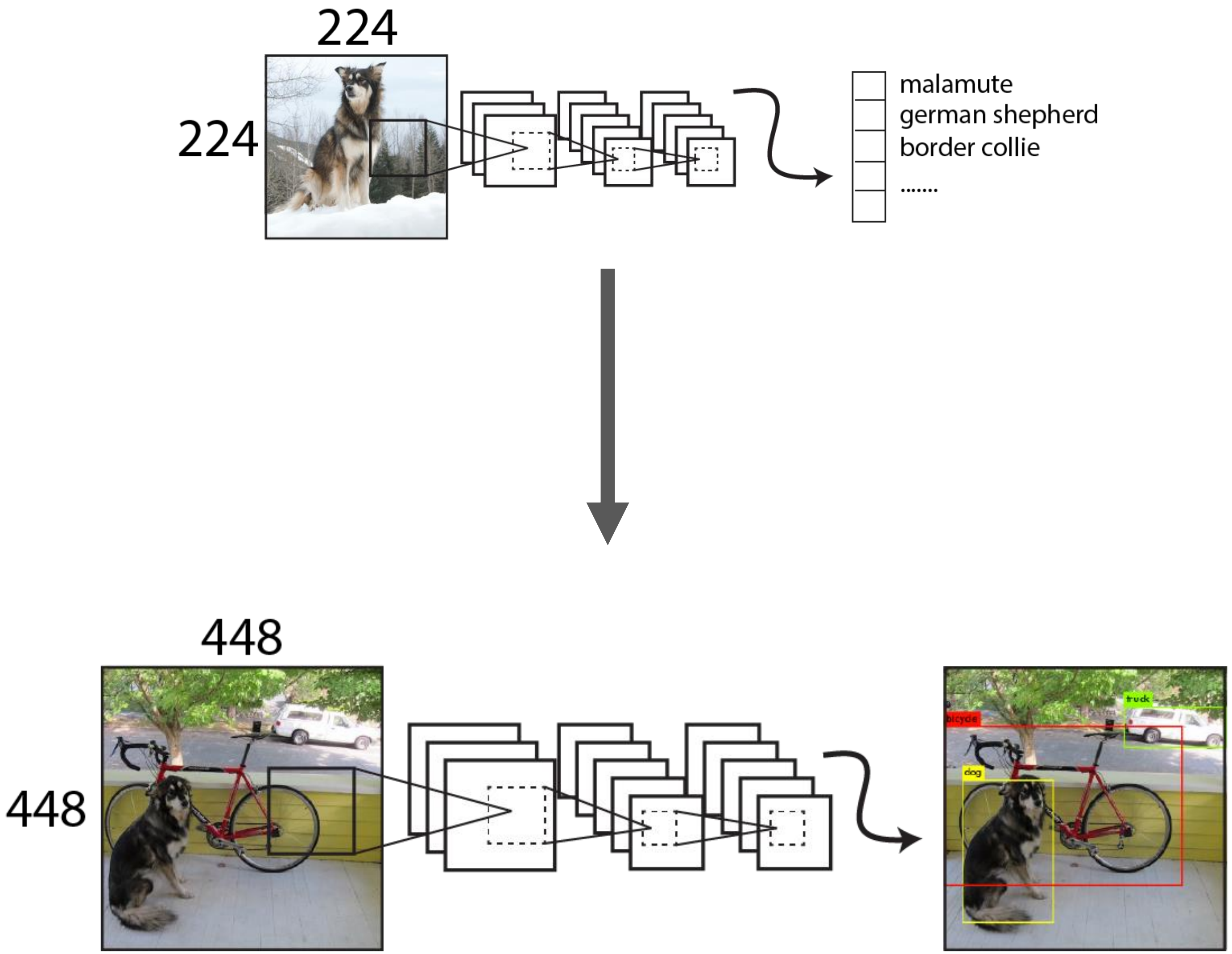
	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img

	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	63.4	45 FPS	22 ms/img

Train on ImageNet

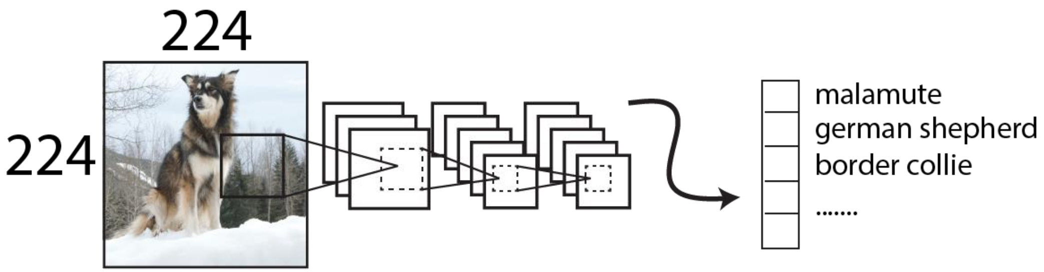
Resize network

Fine-tune on detection

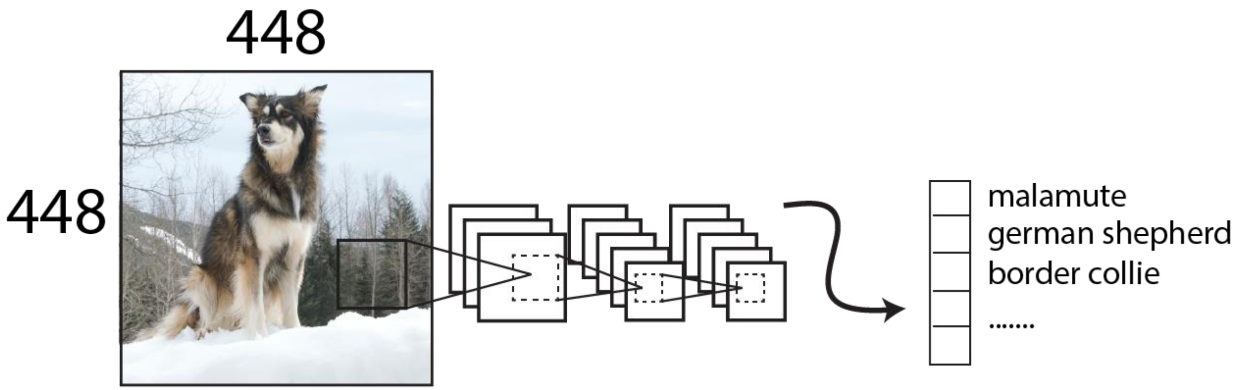


Fine-tune 448x448 Classifier: +3.5% mAP

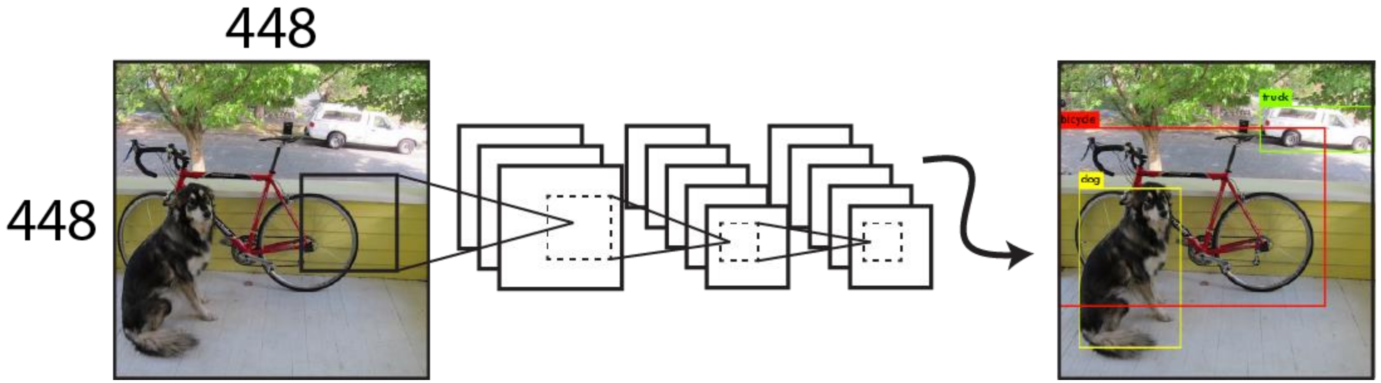
Train on ImageNet



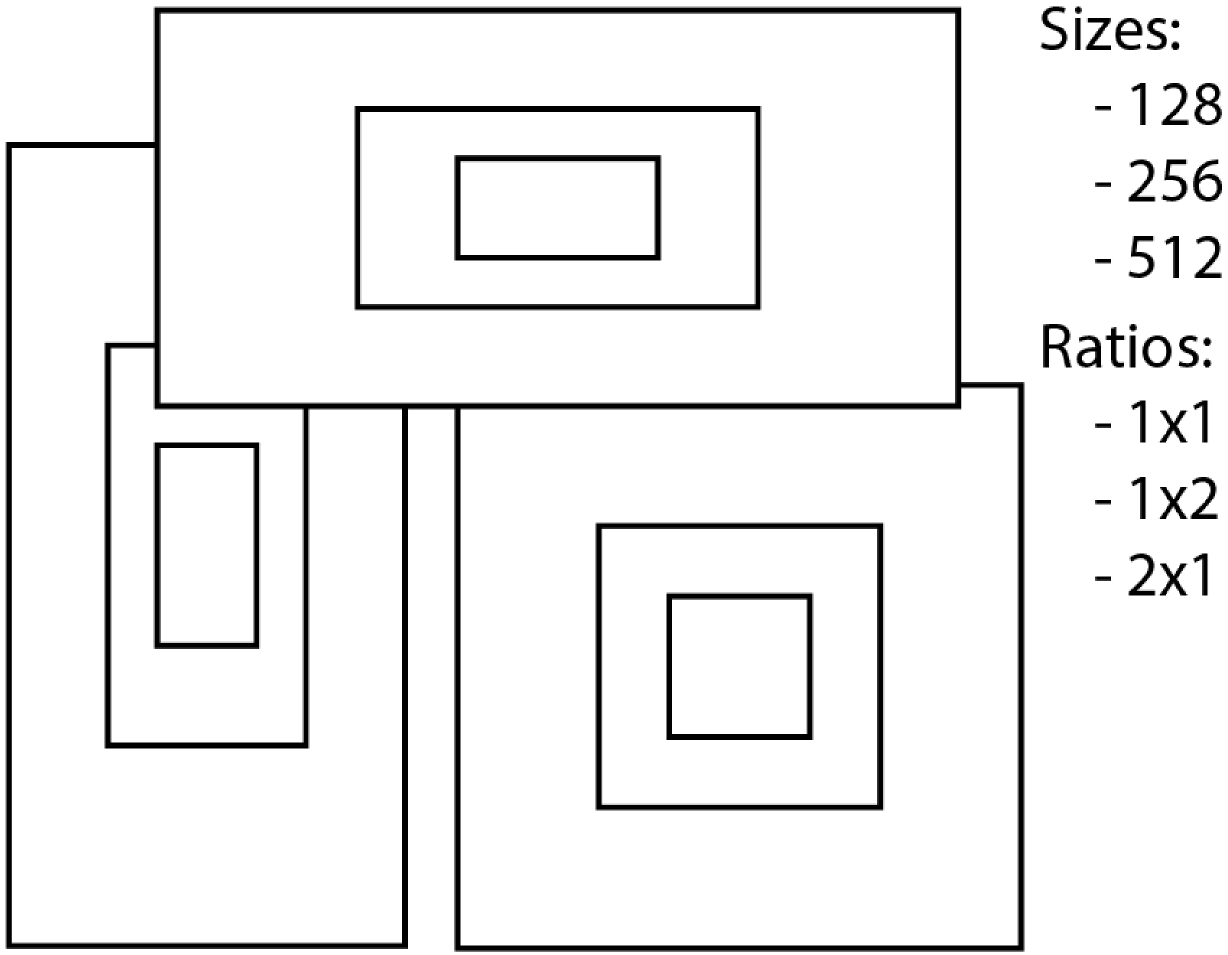
Resize, fine-tune
on ImageNet



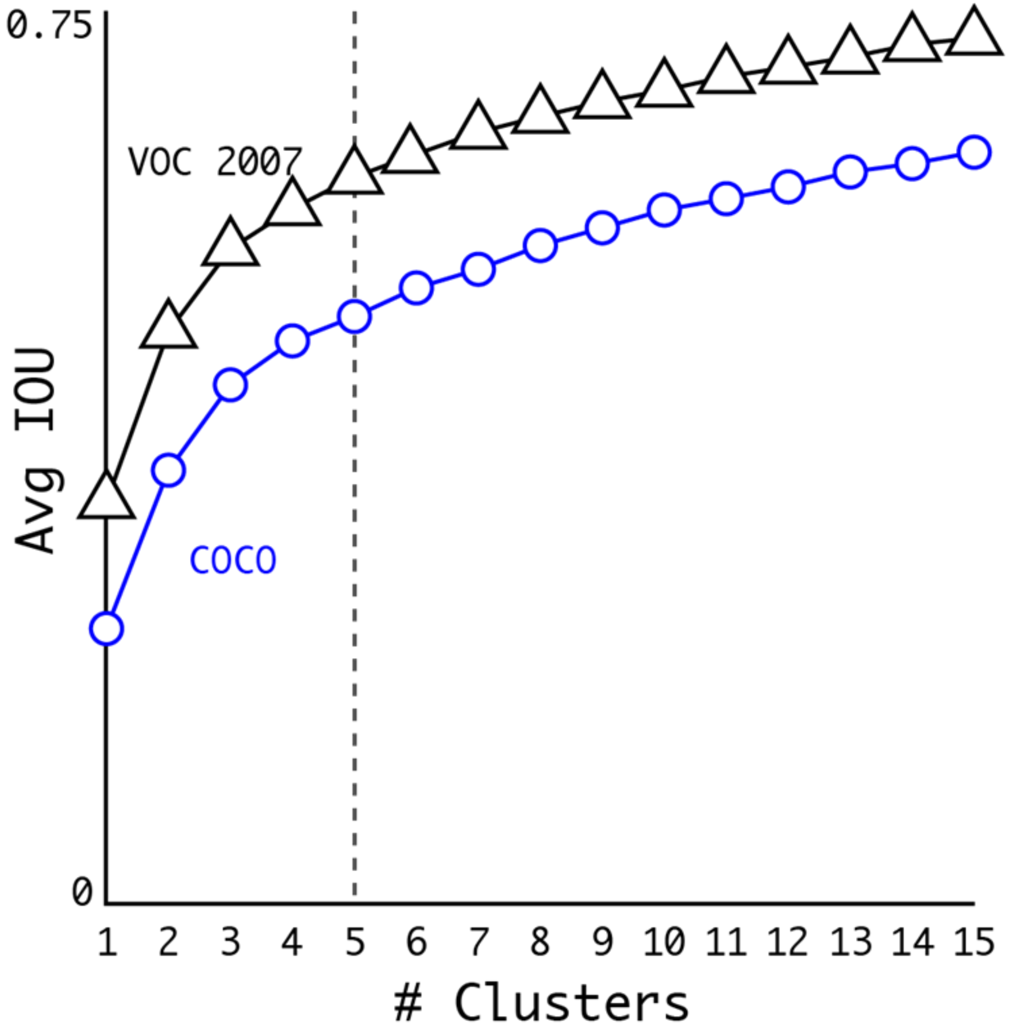
Fine-tune on detection

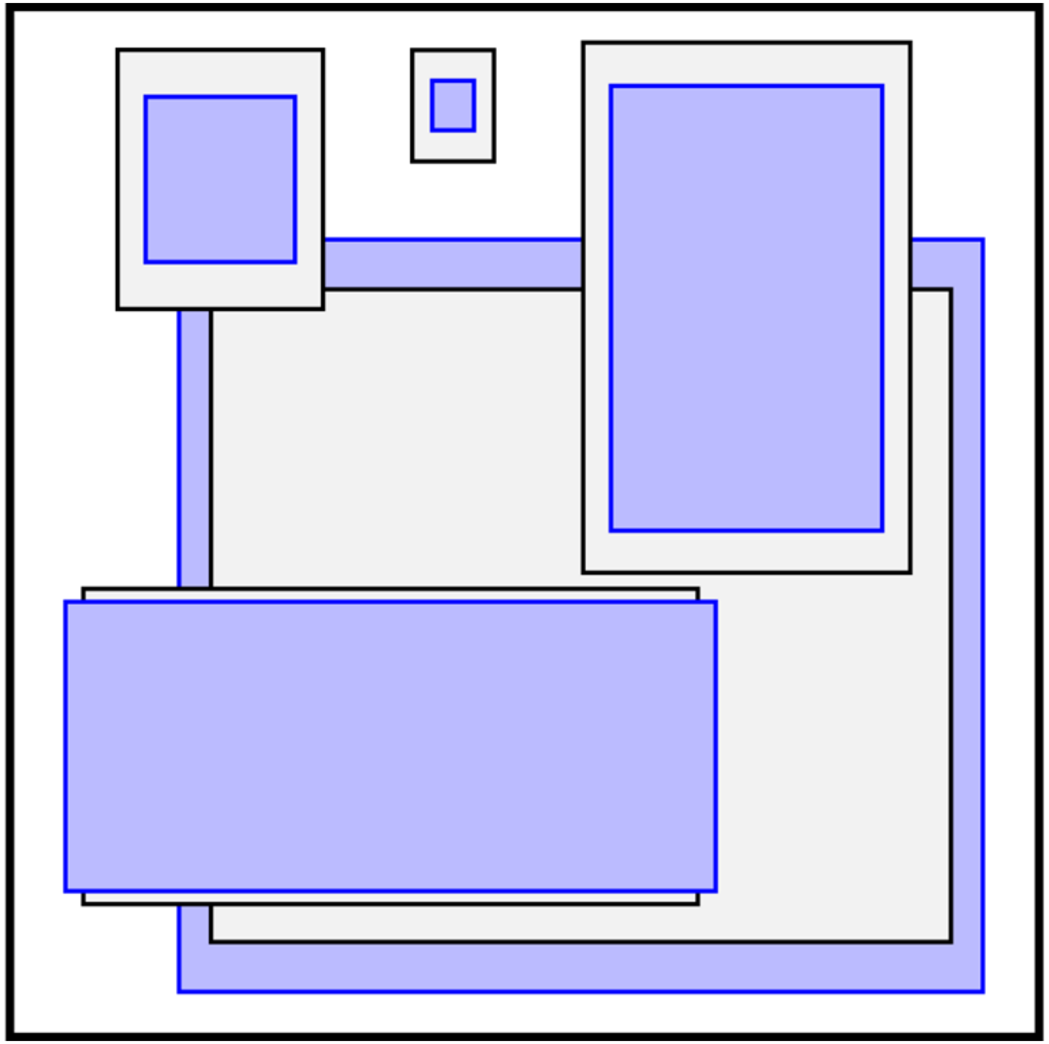


Anchor boxes use static initialization

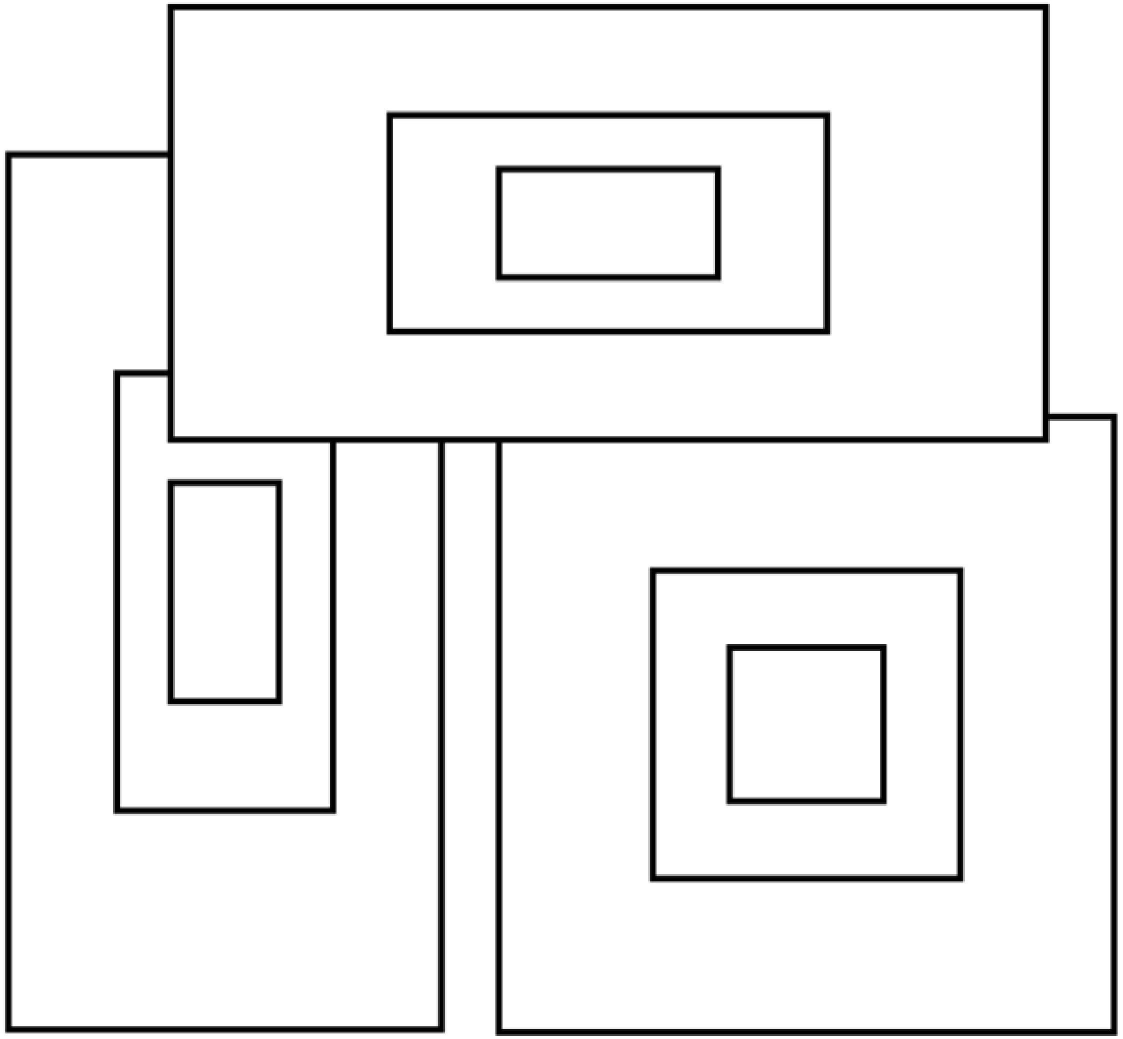


We use k-means to find better initializations

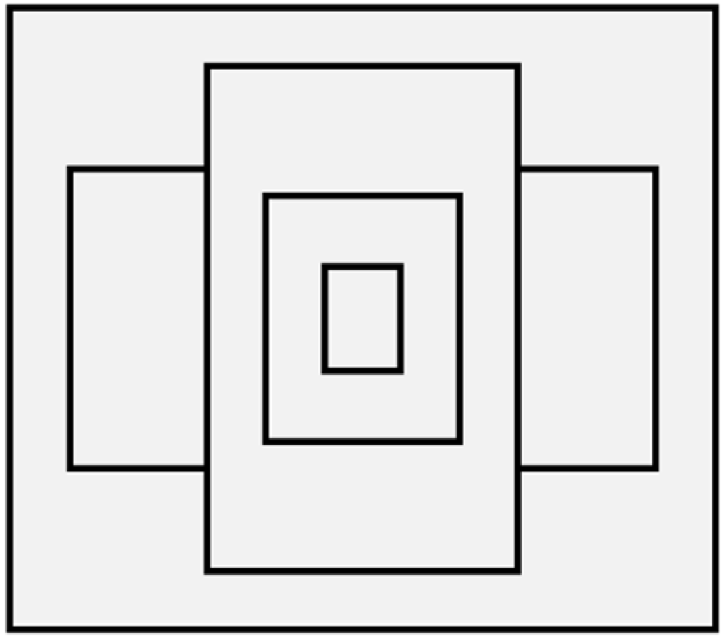




Anchor Boxes



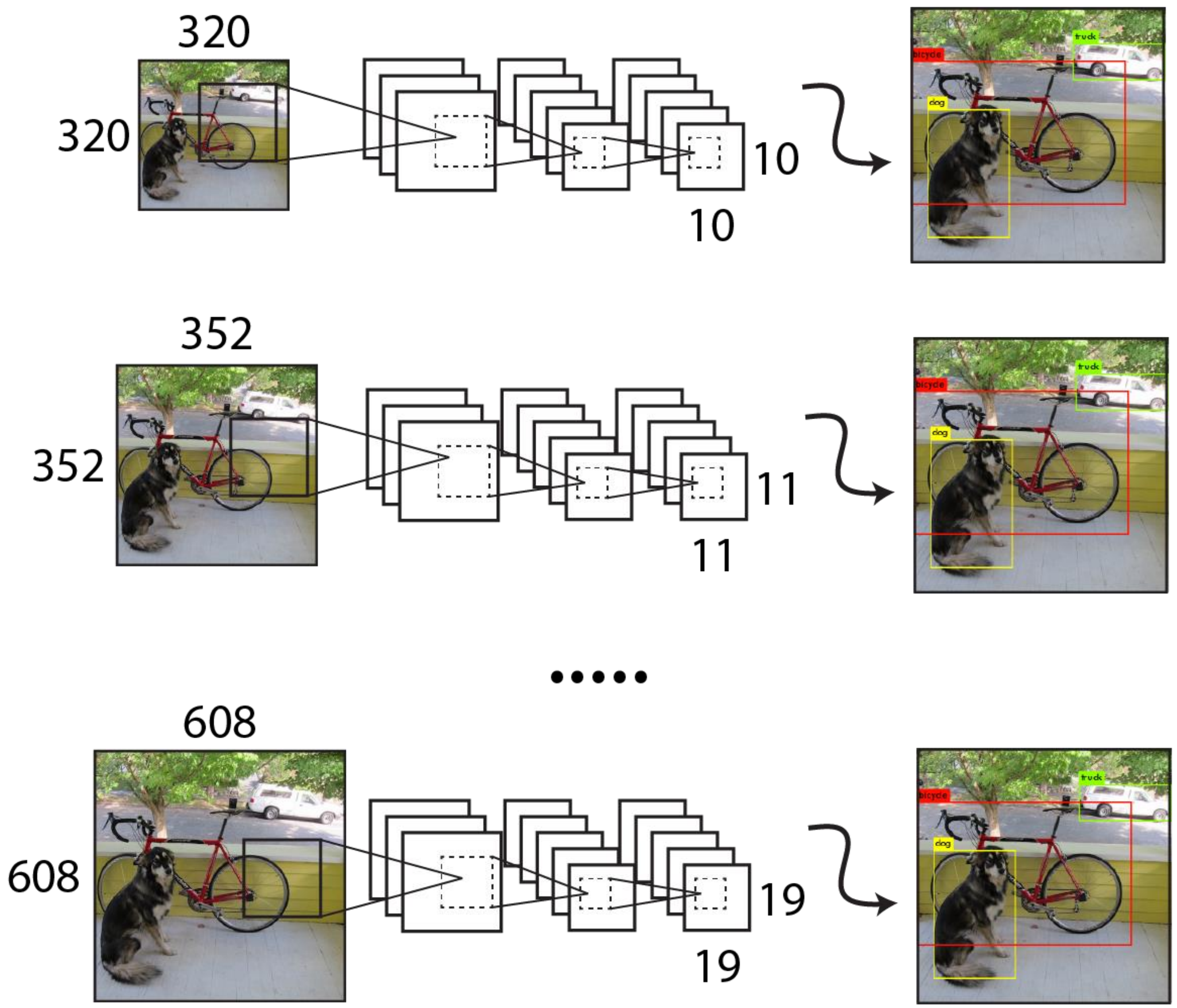
Dimension Clusters



Dimension Clusters: +5% mAP

Box Generation	#	Avg IOU
Cluster SSE	5	58.7
Cluster IOU	5	61.0
Anchor Boxes [15]	9	60.9

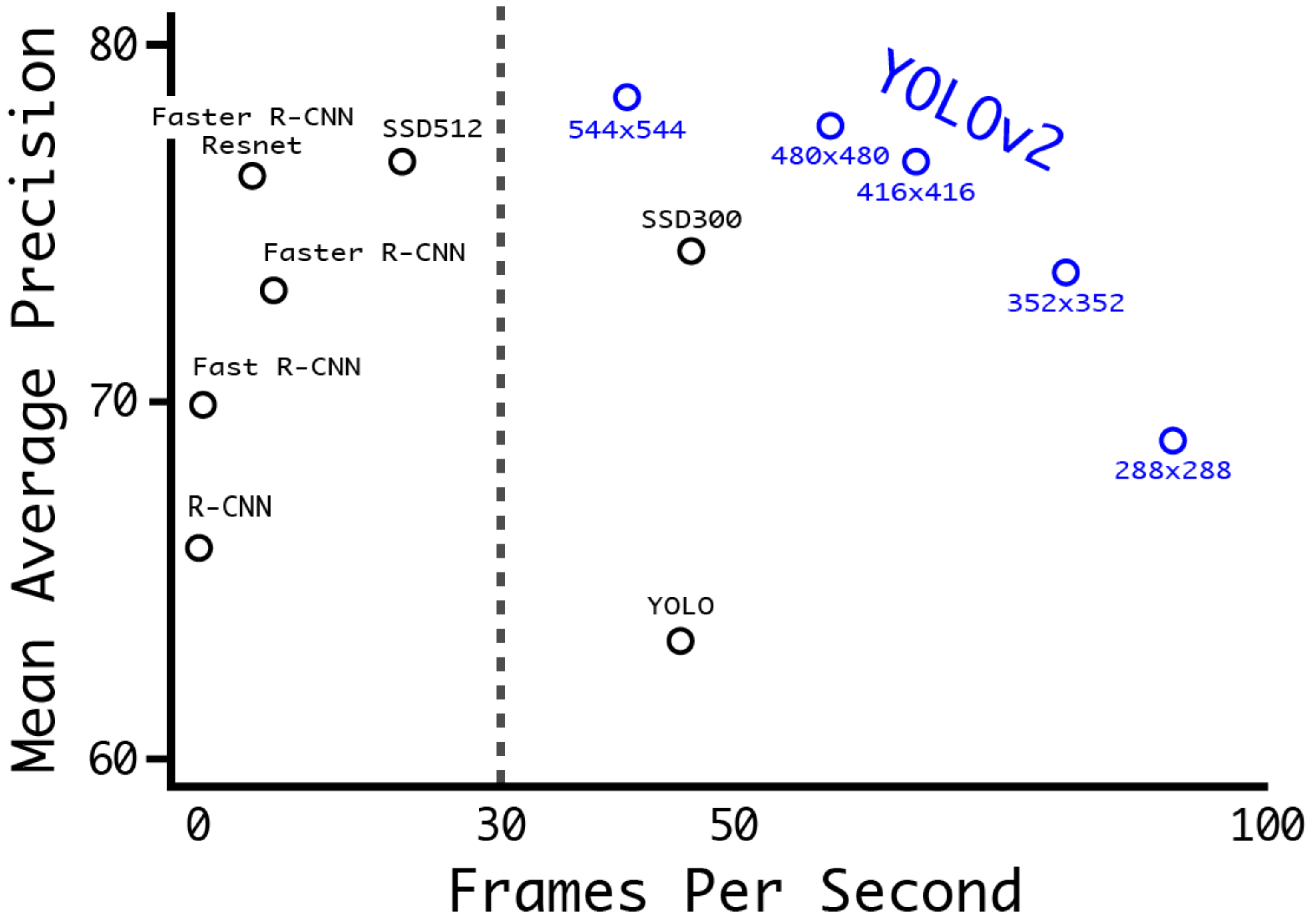
Multi-scale training: +1.5% mAP



Ablations

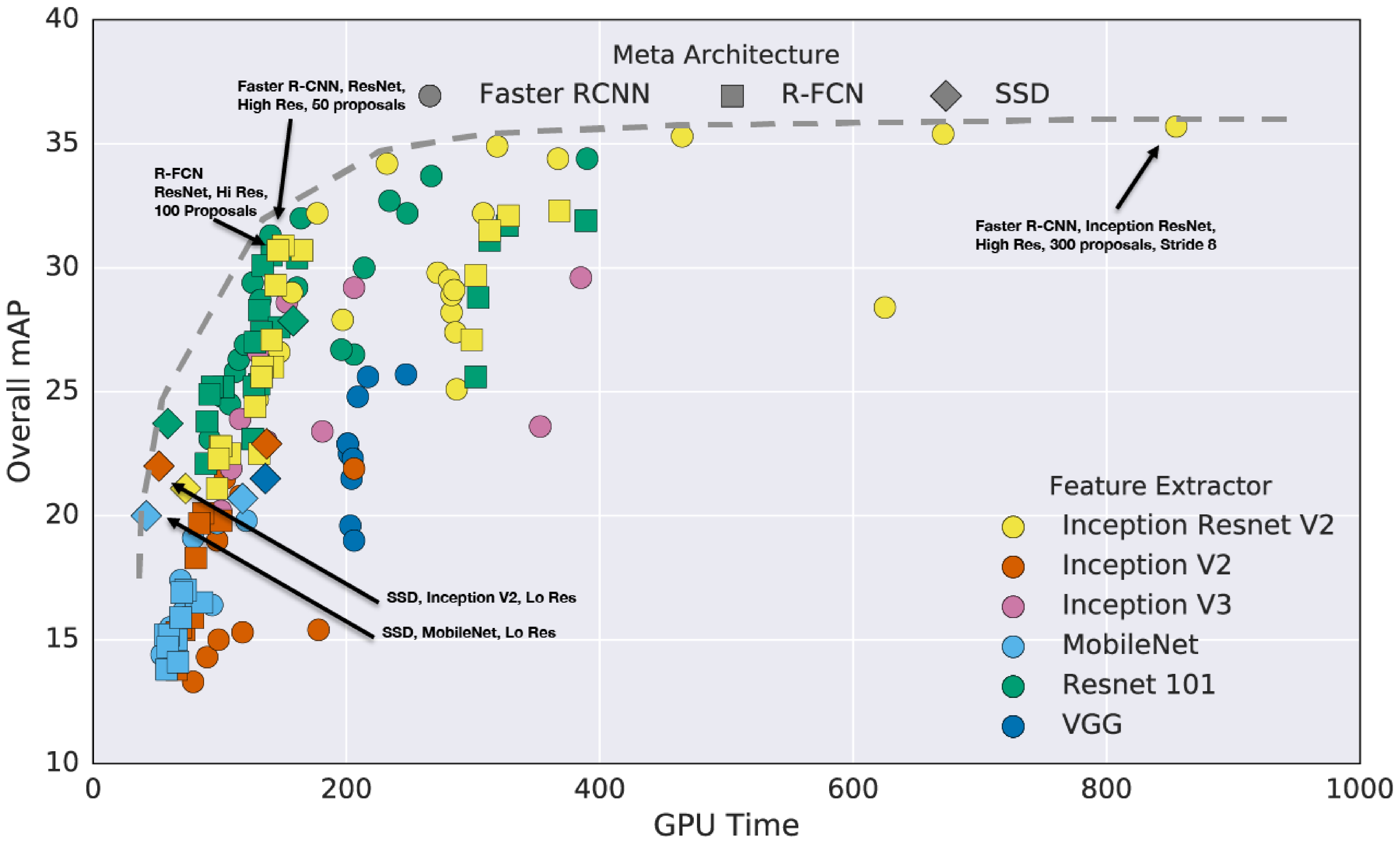
	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

YOLOv2: Fast, Accurate Detection

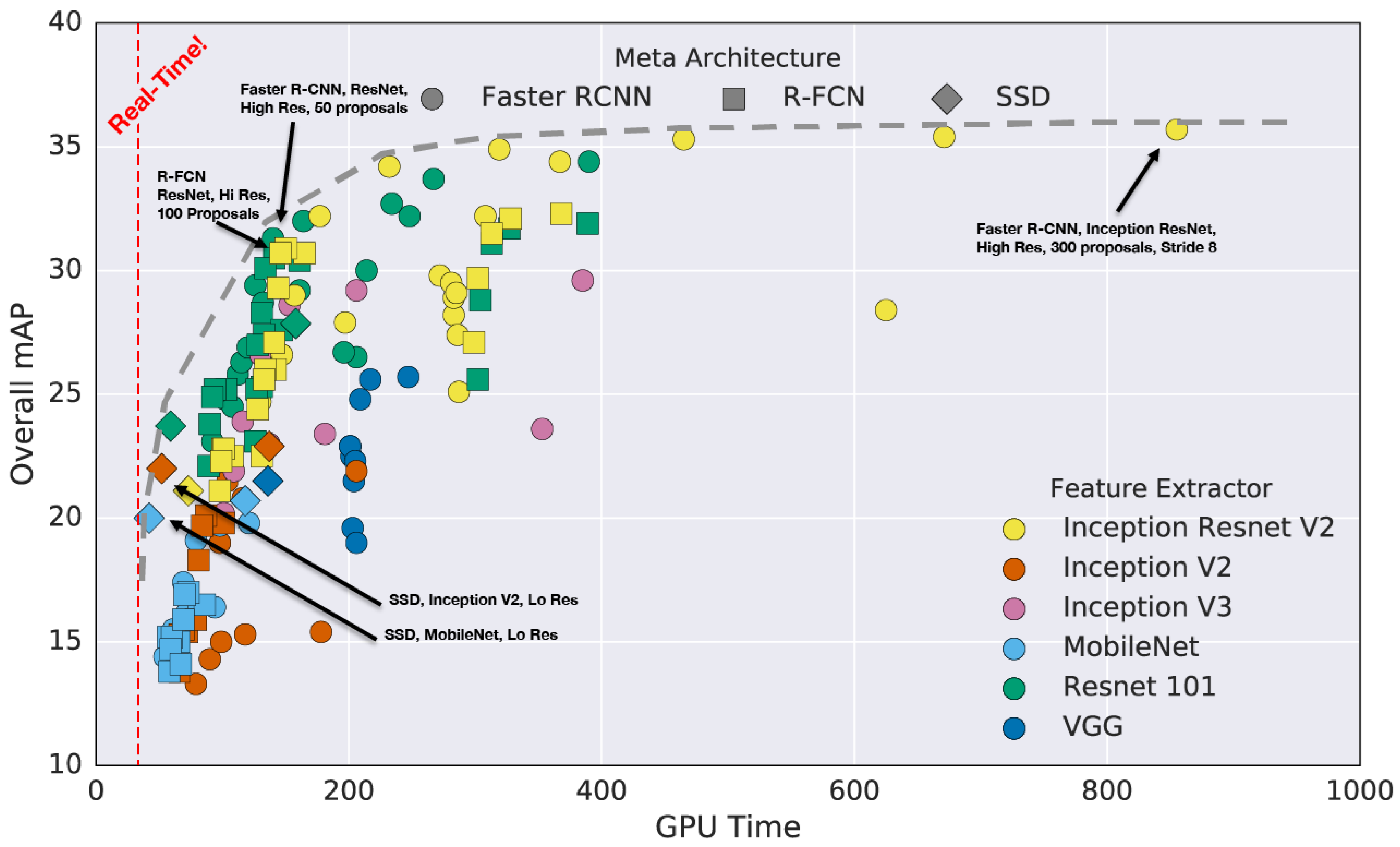


COCO dataset performance

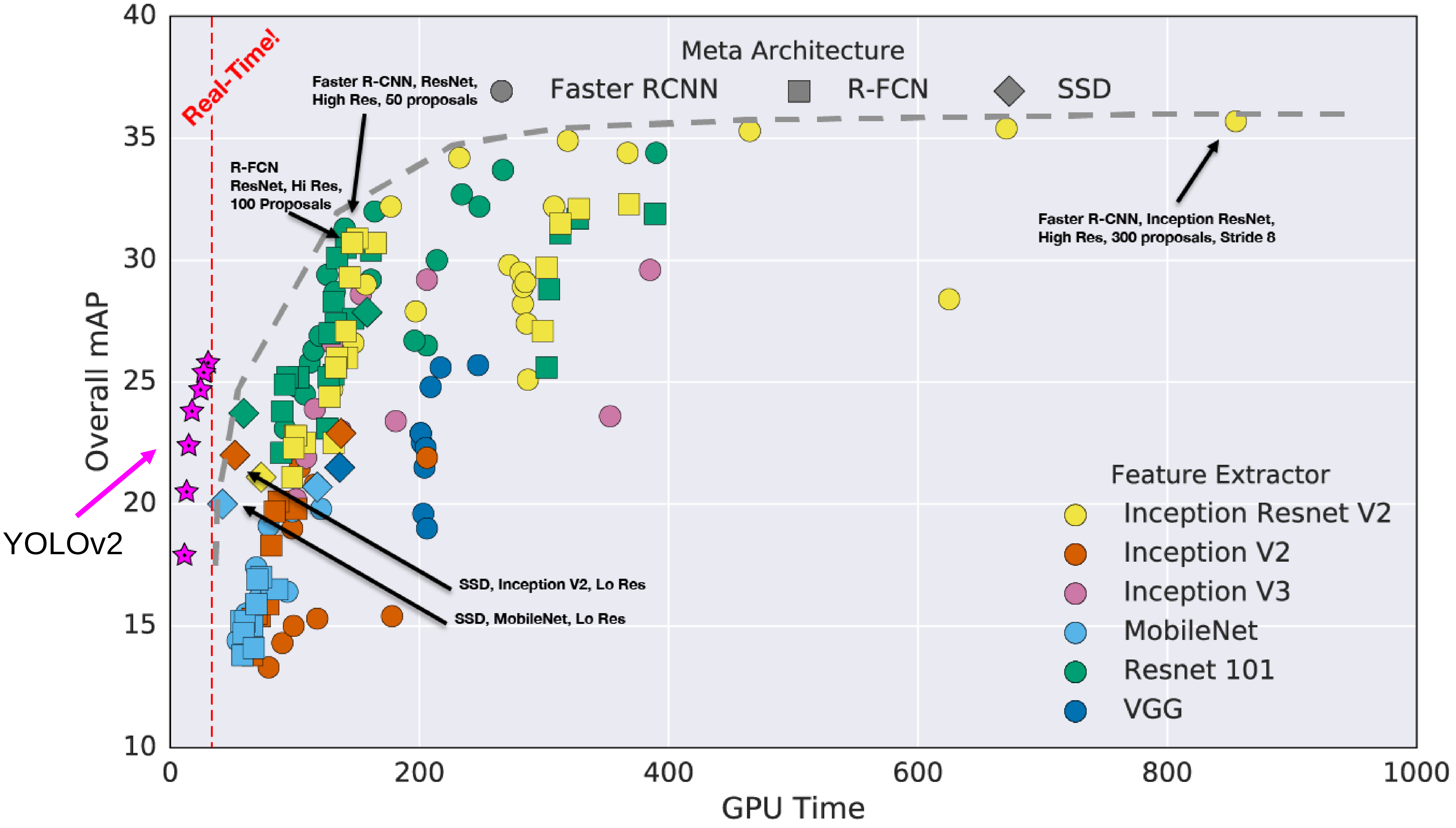
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast R-CNN [5]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast R-CNN[1]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster R-CNN[15]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [1]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster R-CNN[10]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300 [11]	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512 [11]	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**
YOLOv2 [11]	trainval35k	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4



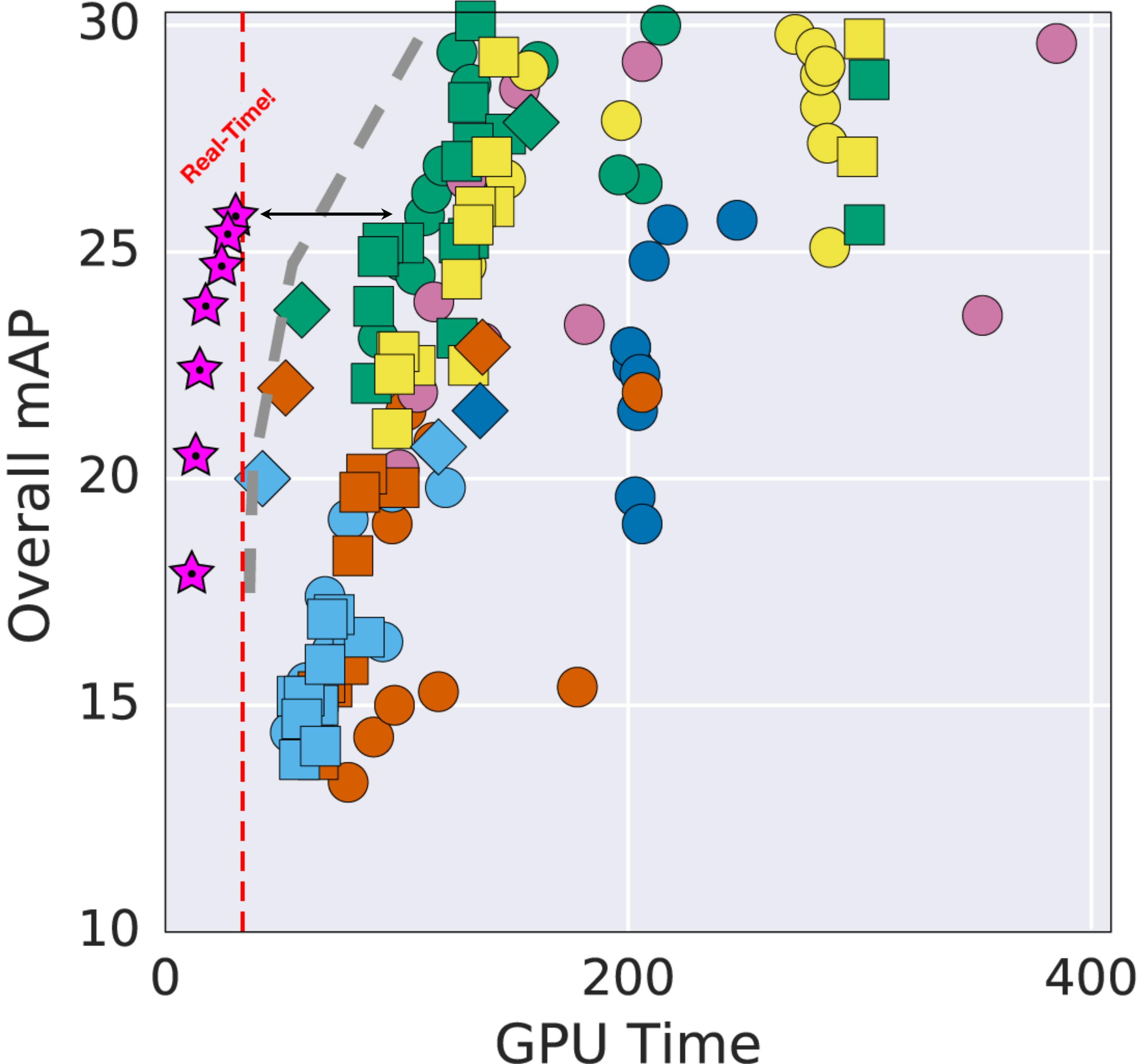
Huang, Jonathan, et al. "Speed/accuracy trade-offs for modern convolutional object detectors." *arXiv preprint arXiv:1611.10012* (2016).



Huang, Jonathan, et al. "Speed/accuracy trade-offs for modern convolutional object detectors." *arXiv preprint arXiv:1611.10012* (2016).



Huang, Jonathan, et al. "Speed/accuracy trade-offs for modern convolutional object detectors." *arXiv preprint arXiv:1611.10012* (2016).



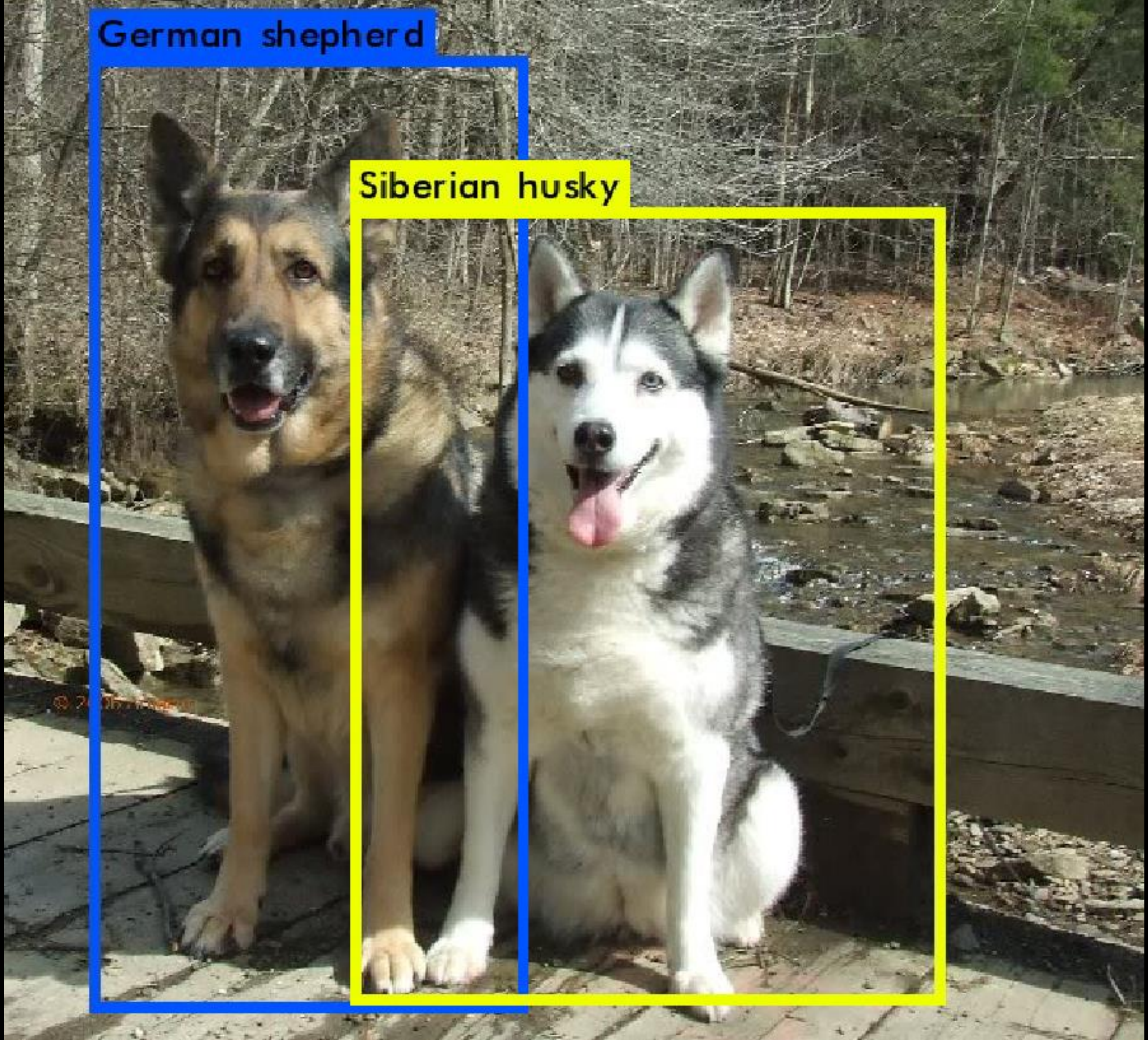
Real-Time!
Overall mAP
GPU Time
10
15
20
25
30
0
200
400

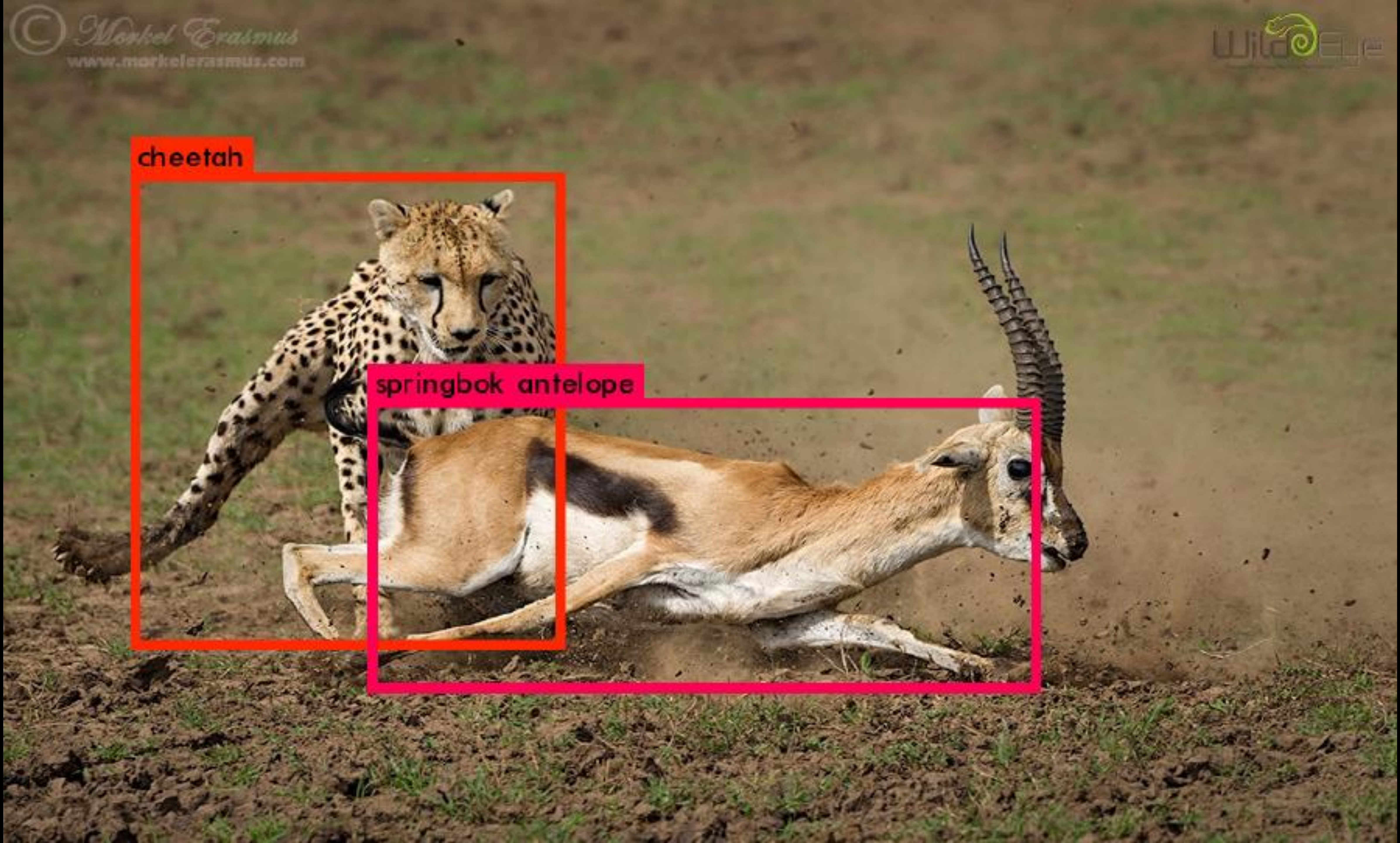
Speed is not just parameter counts or FLOPs

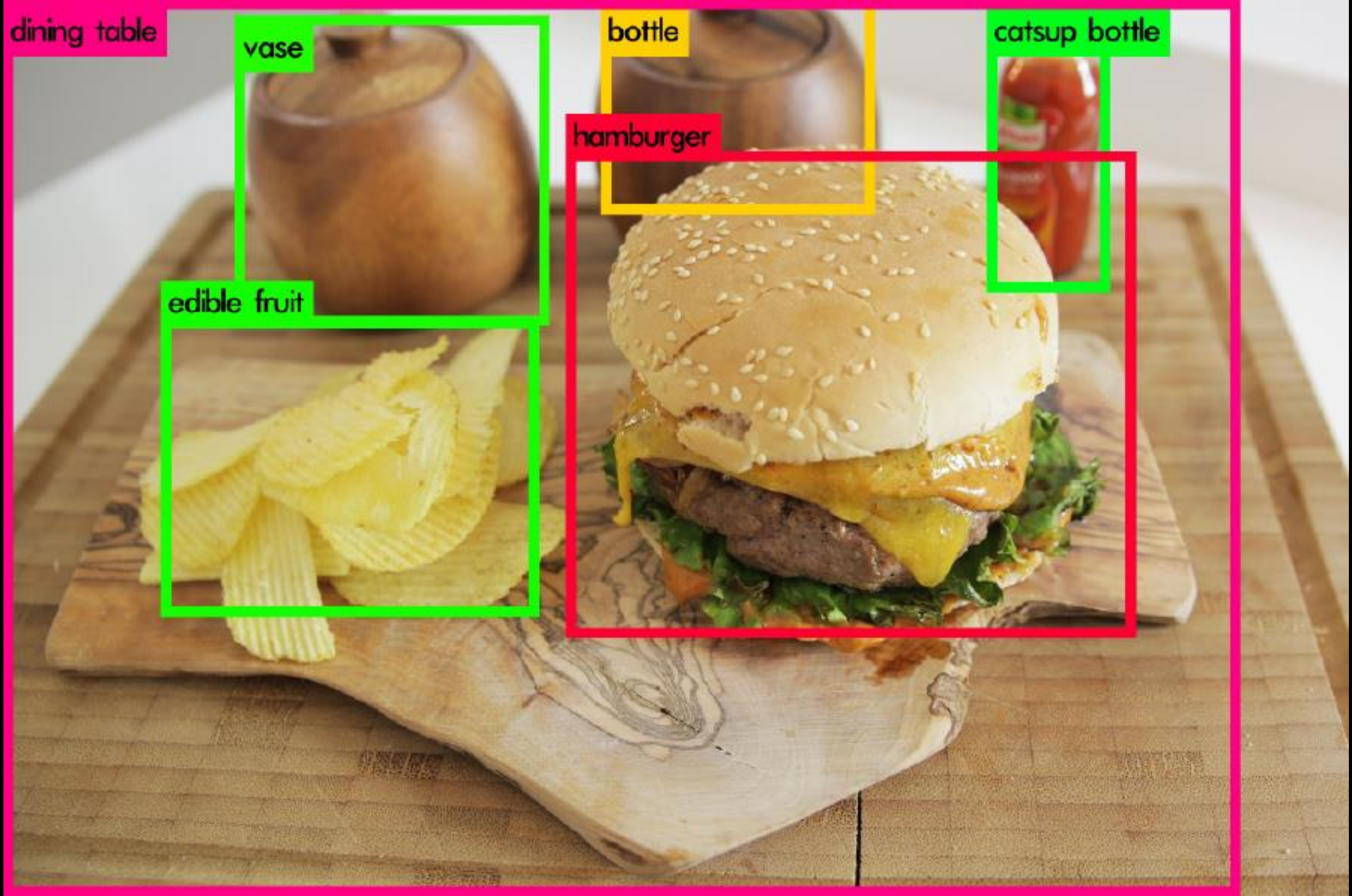
	Top 1	**Top 5**	**FLOPs**	**GPU Speed**
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	**180 FPS**
Resnet50	**75.3**	**92.2**	7.66 Bn	90 FPS

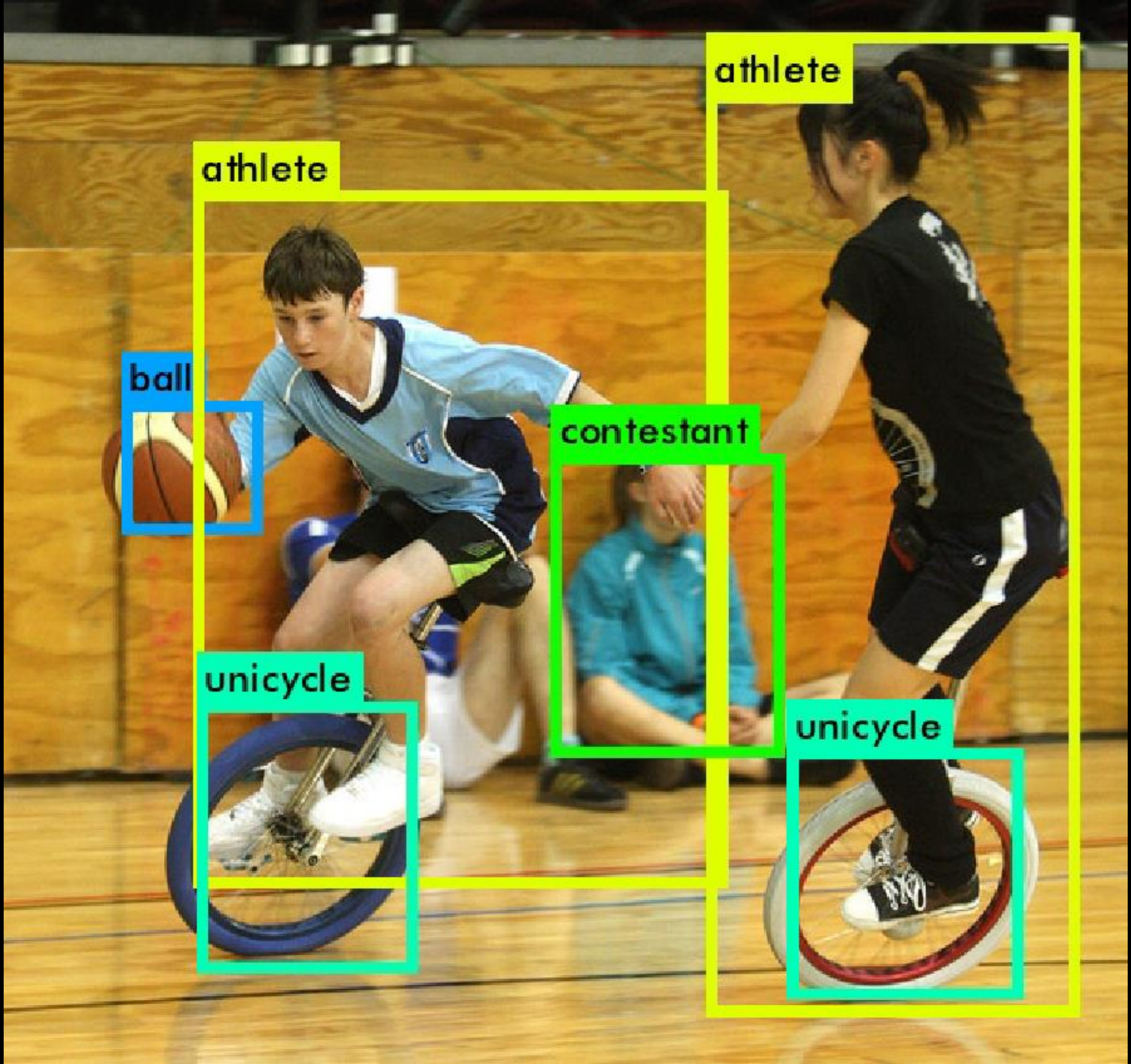
Darknet19: A good balance of speed and accuracy

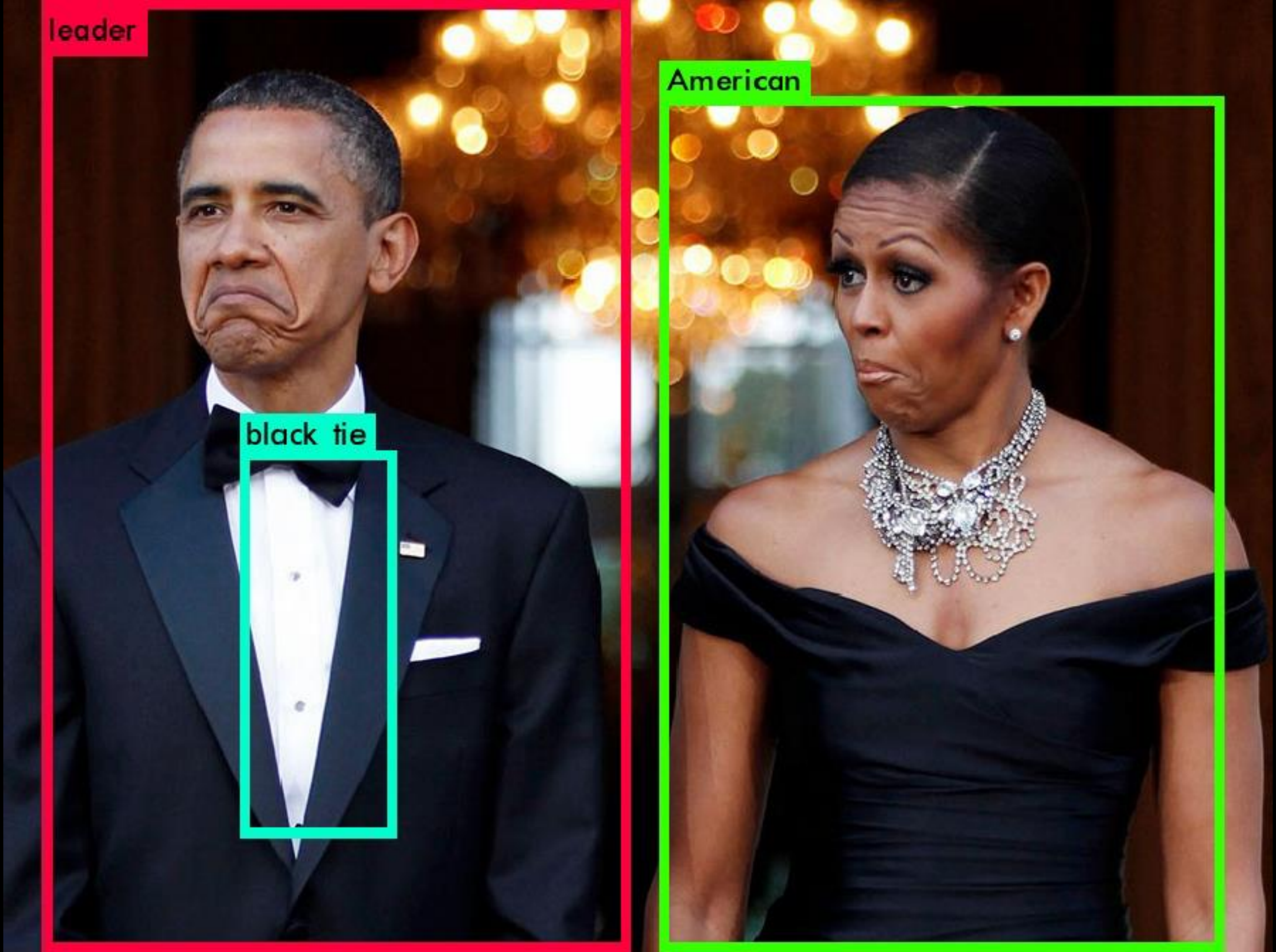
	Top 1	**Top 5**	**FLOPs**	**GPU Speed**
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	180 FPS
Resnet50	**75.3**	**92.2**	7.66 Bn	90 FPS
Darknet19	74.0	91.8	**5.58 Bn**	**200 FPS**

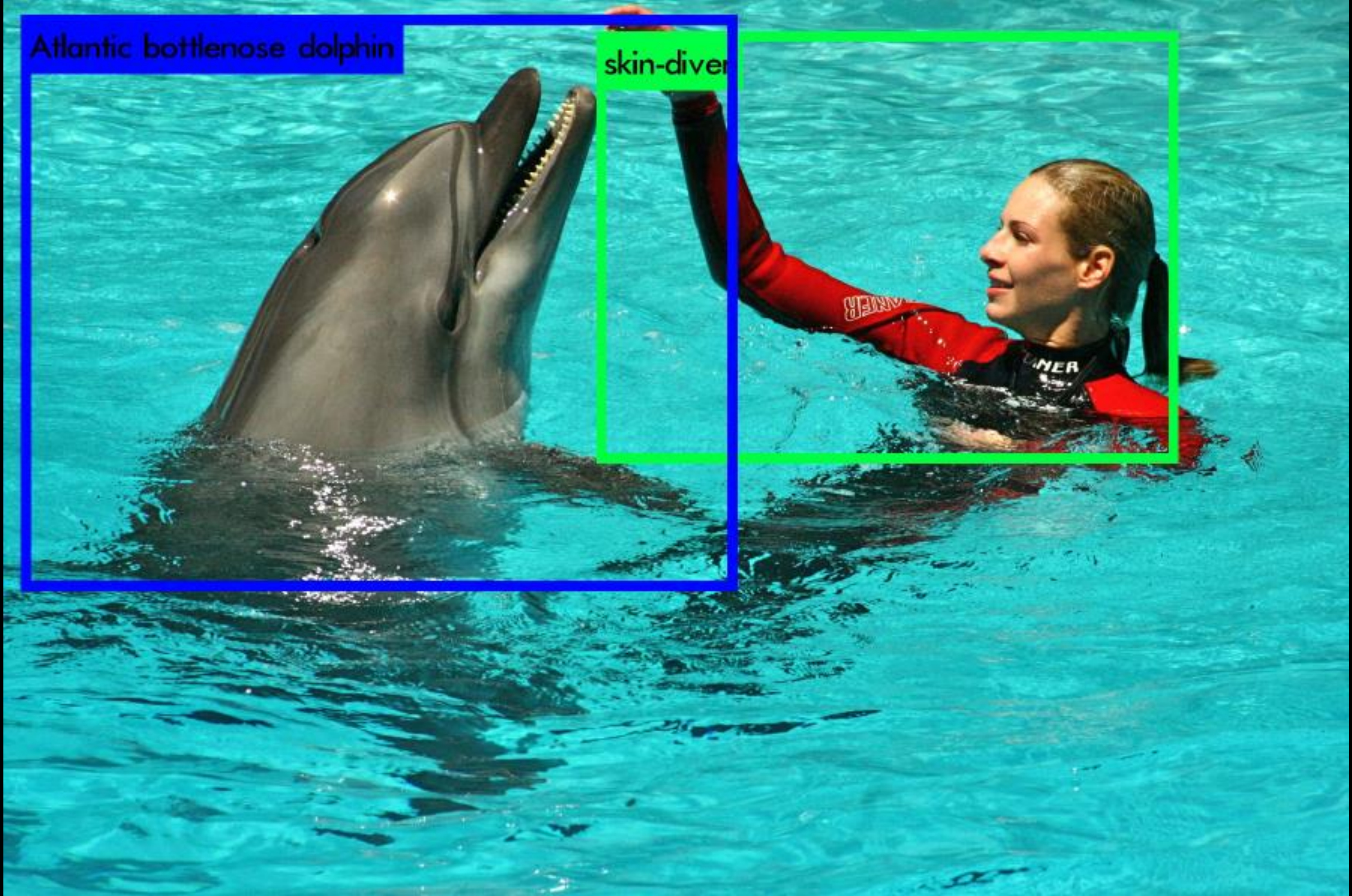

German shepherd
Siberian husky

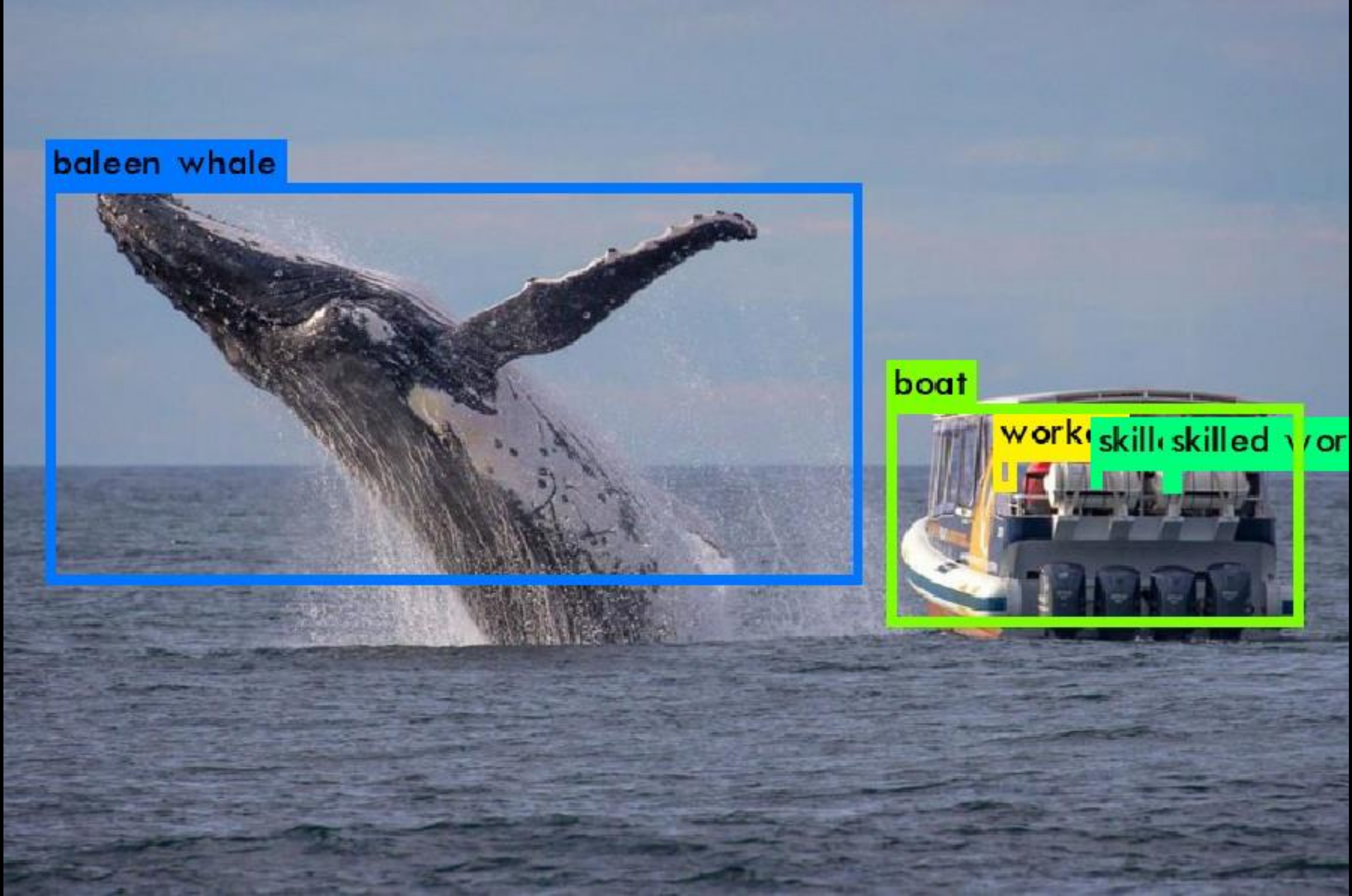

© Morkel Erasmus
www.morkelerasmus.com
cheetah
springbok antelope


dining table
vase
bottle
catsup bottle
hamburger
edible fruit


athlete
athlete
ball
contestant
unicycle
unicycle


leader
black tie
American


Atlantic bottlenose dolphin
skin-diver


baleen whale
boat
work
skill
skilled wor


chanterelle
chanterelle
carrot

Outline – More complex outputs from deep networks

- Image Output (e.g. colorization, semantic segmentation, super-resolution, stylization, depth estimation…)
- Attributes
- Text Captions
- Semantic Keypoints
- Object Detection
  - “Single shot” or “one stage” detectors like YOLO or SSD. The network runs once per image.
  - “Two stage” detectors like Mask RCNN. A feature extractor network runs once per image, various “head” networks run an arbitrary amount of times.

Mask R-CNN

ICCV 2017

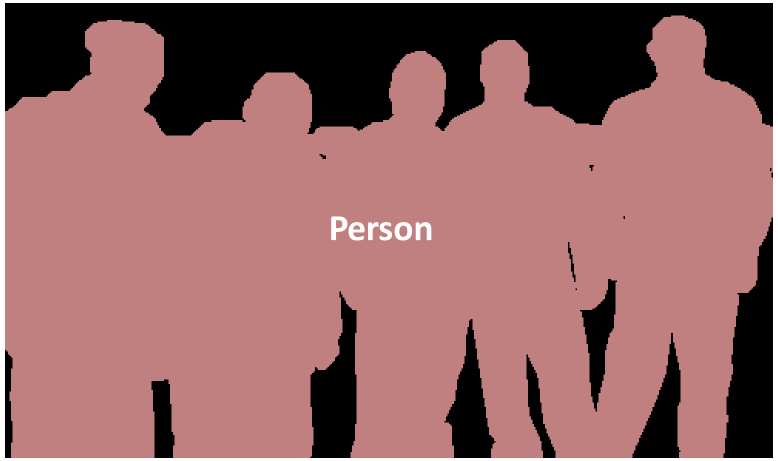
Kaiming He

Georgia Gkioxari, Piotr Dollár, and Ross Girshick Facebook AI Research (FAIR)

Visual Perception Problems

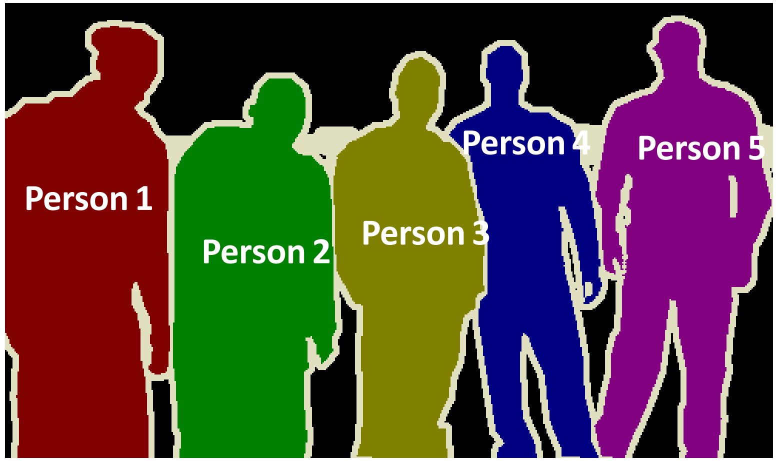
Object Detection

✓



Semantic Segmentation

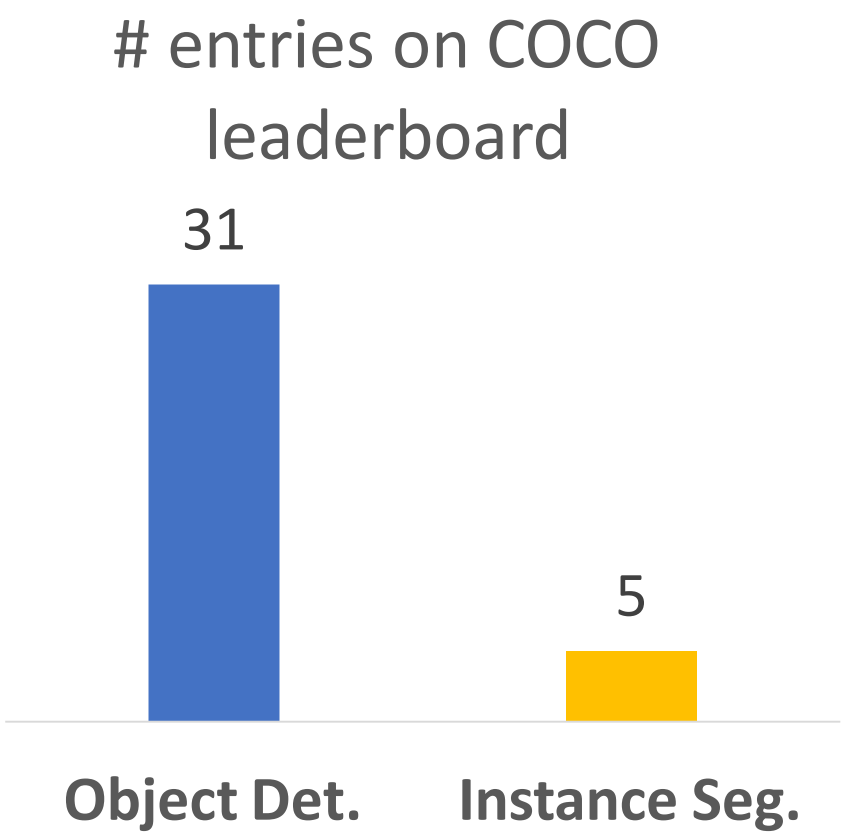
✓

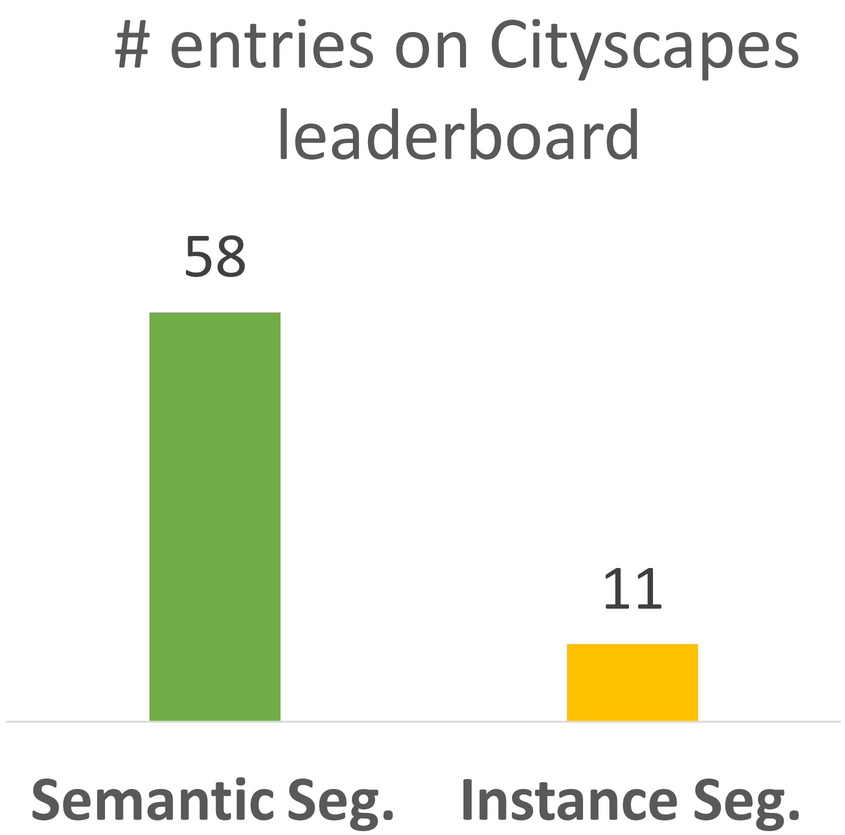


Instance Segmentation

?

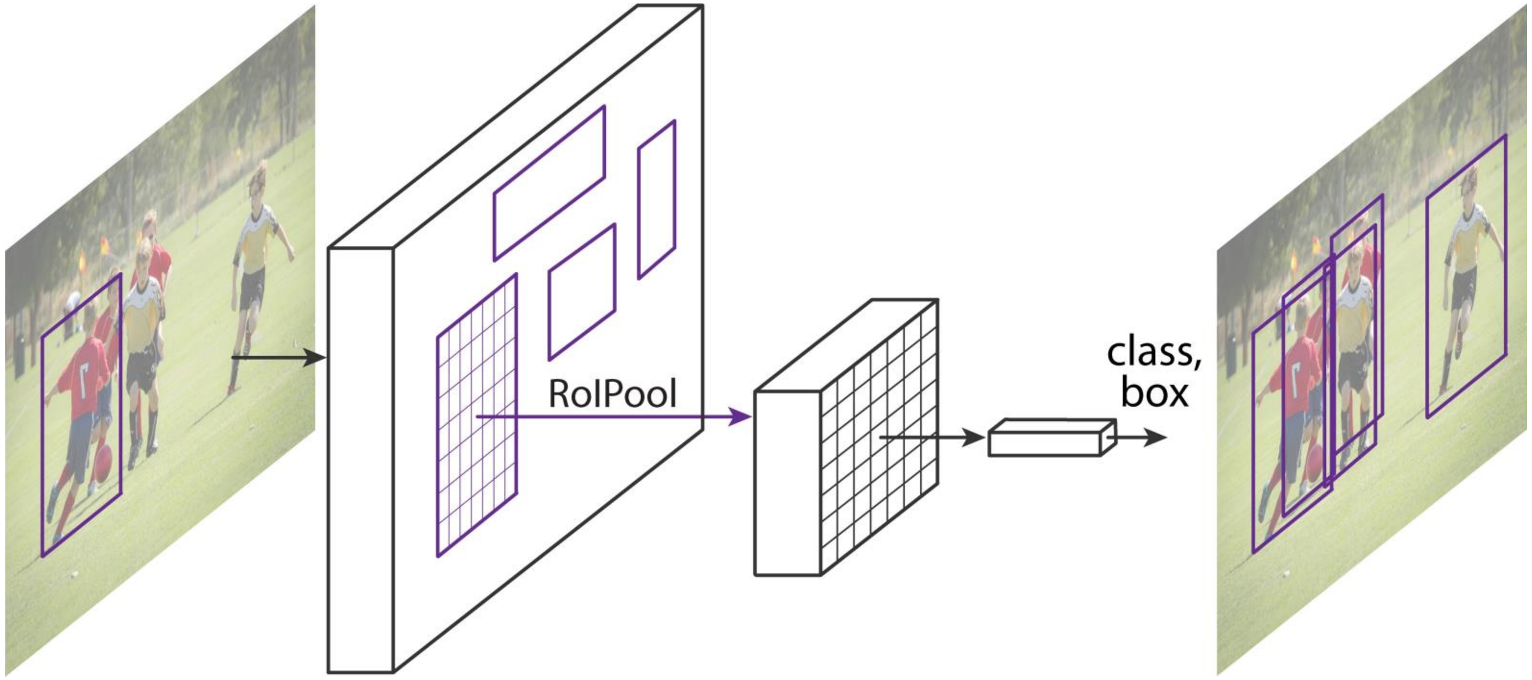
A Challenging Problem...





Object Detection

- Fast/Faster R-CNN
  - ✓Good speed
  - ✓Good accuracy
  - ✓Intuitive
  - ✓Easy to use



Ross Girshick. "Fast R-CNN". ICCV 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.

Semantic Segmentation

- Fully Convolutional Net (FCN)
  - ✓Good speed
  - ✓Good accuracy
  - ✓Intuitive
  - ✓Easy to use

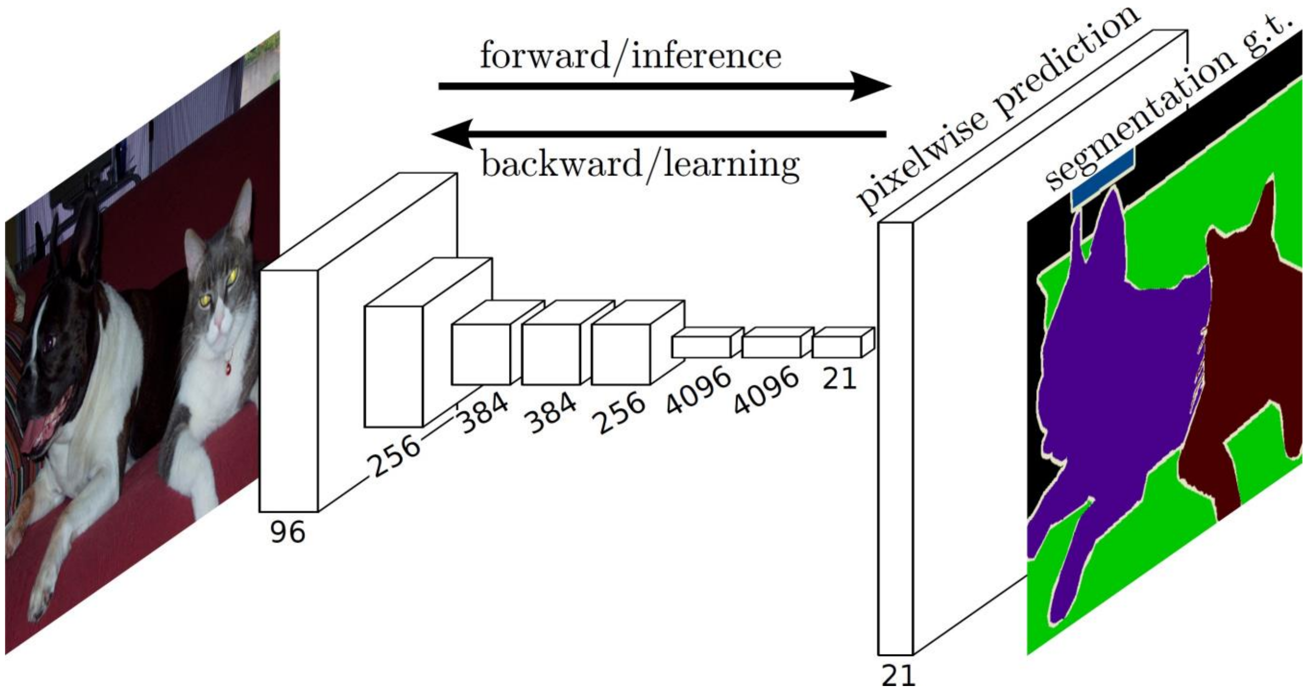
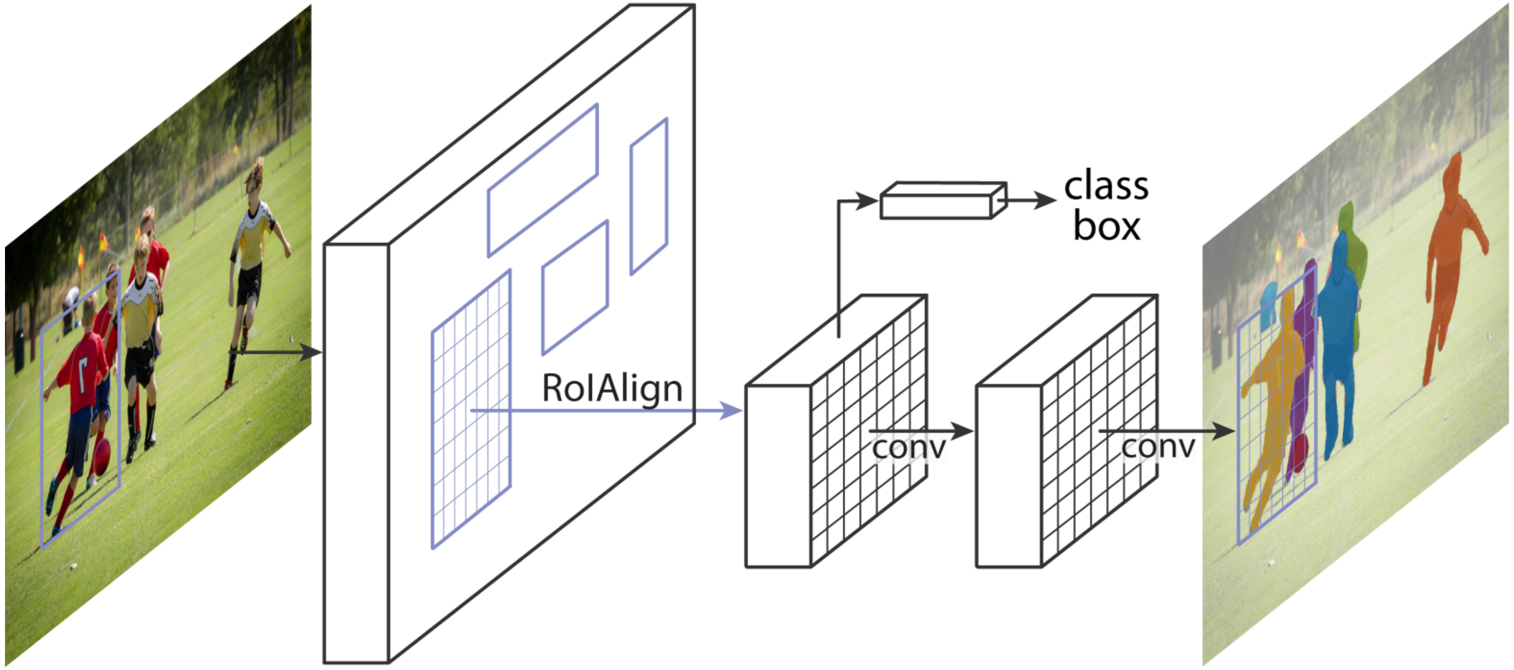


Figure credit: Long et al

Jonathan Long, Evan Shelhamer, & Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". CVPR 2015.

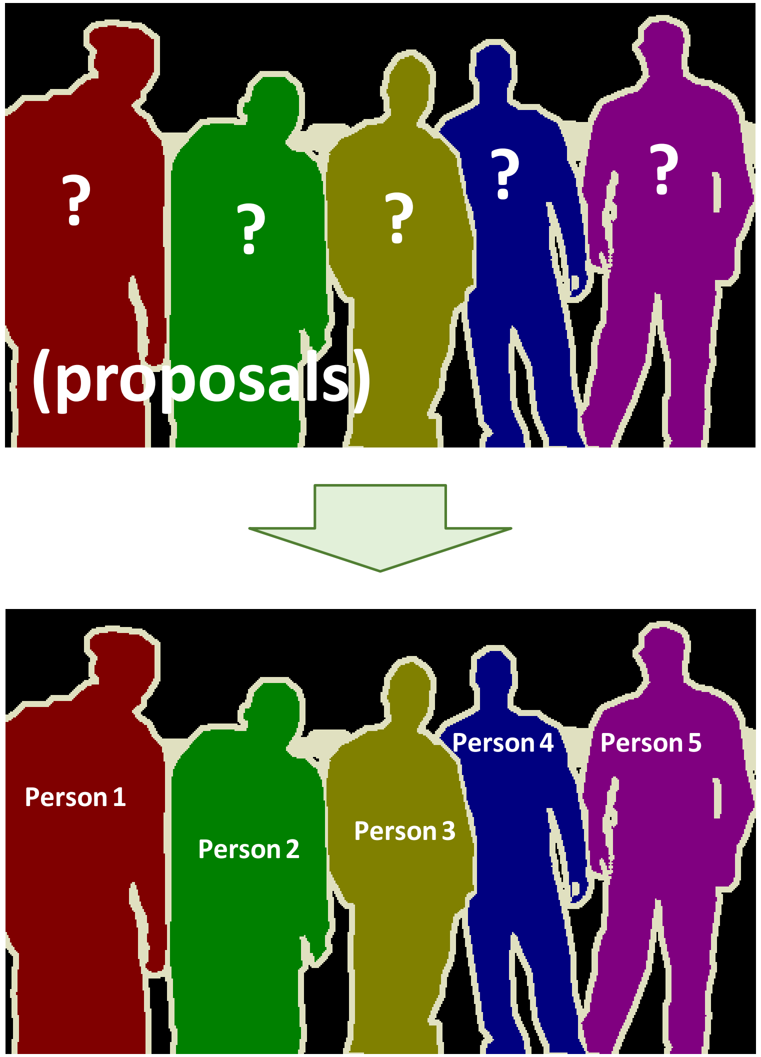
Instance Segmentation

- **Goals** of Mask R-CNN
  - ✓Good speed
  - ✓Good accuracy
  - ✓Intuitive
  - ✓Easy to use

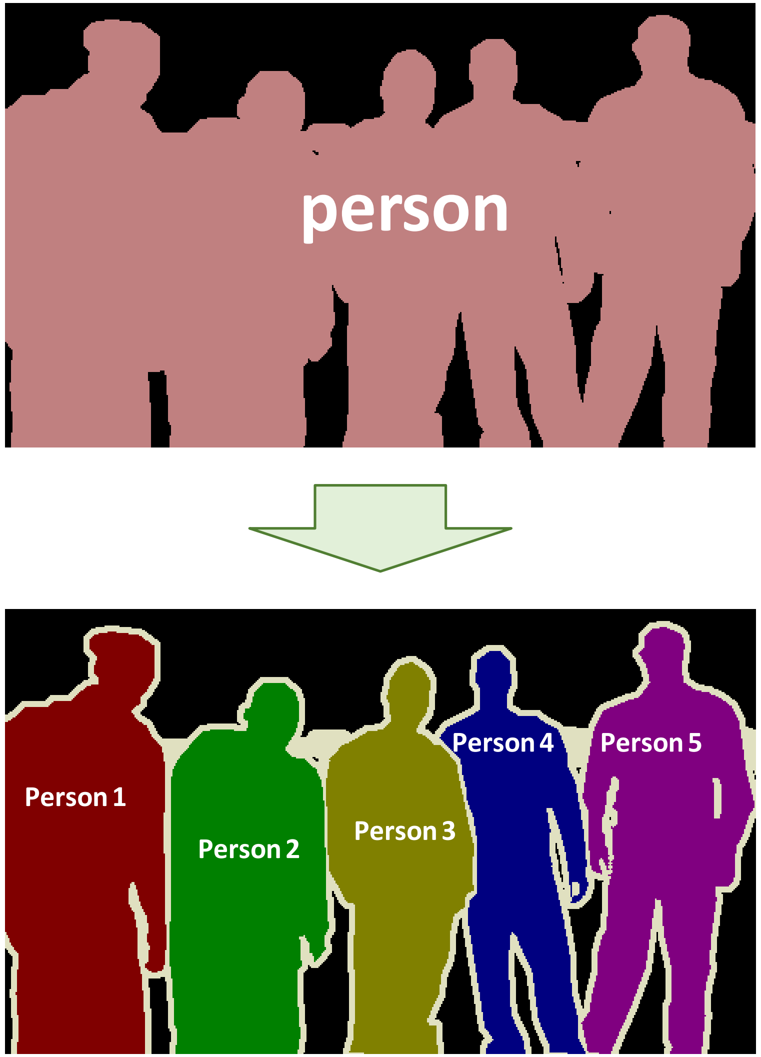


Instance Segmentation Methods

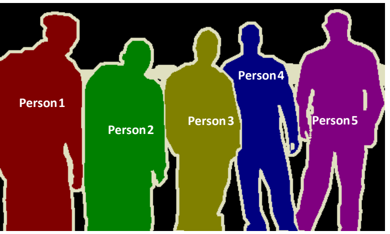
R-CNN driven



FCN driven

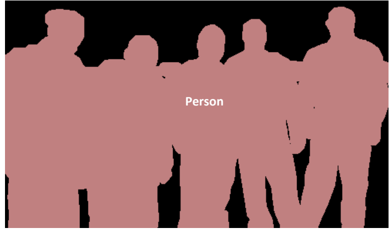


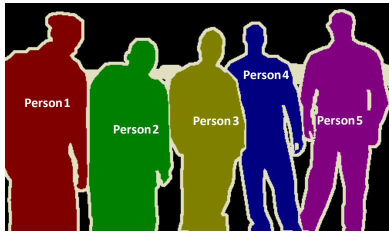
Instance Segmentation Methods



RCNN-driven

- SDS [Hariharan et al, ECCV'14]
- HyperCol [Hariharan et al, CVPR'15]
- CFM [Dai et al, CVPR'15]
- MNC [Dai et al, CVPR'16]





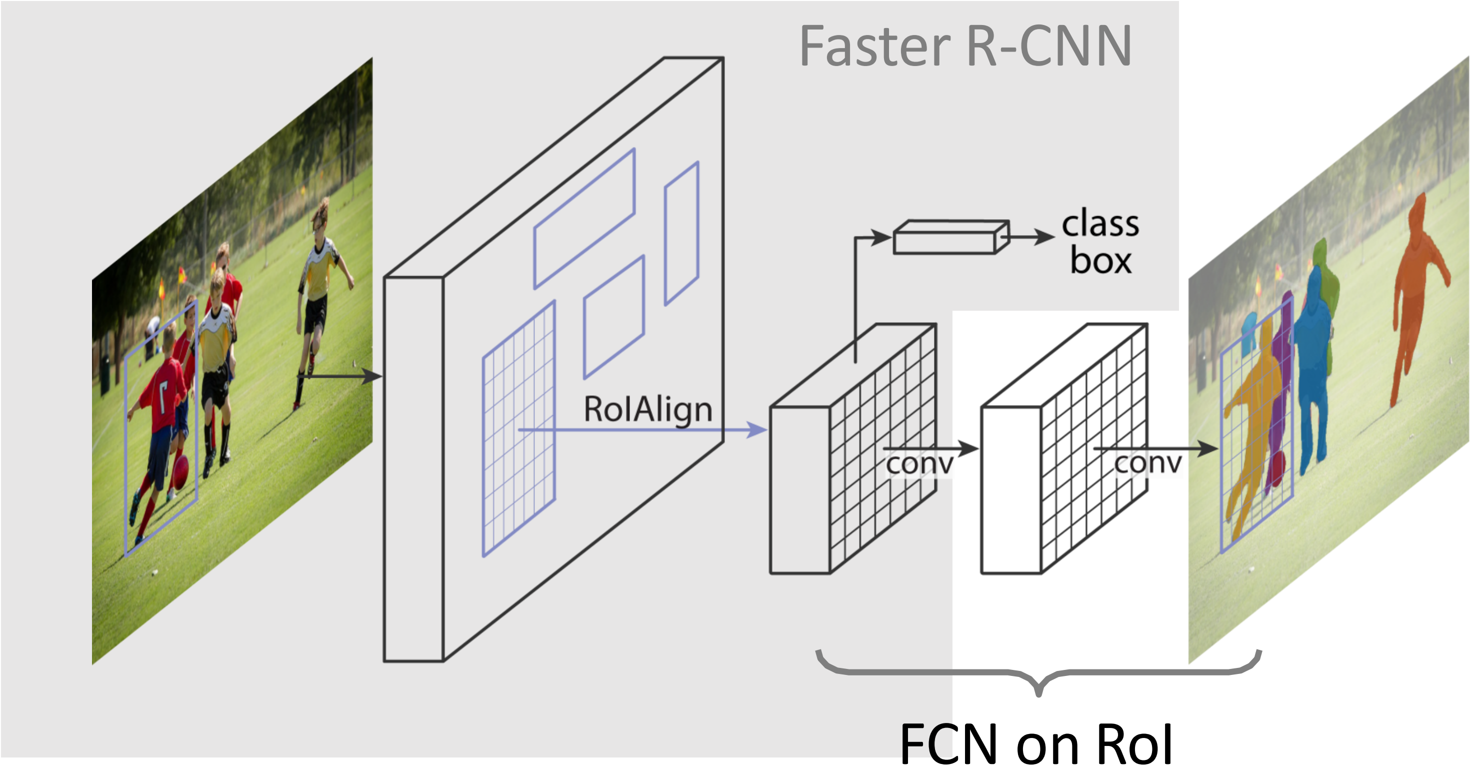
FCN-driven

- PFN [Liang et al, arXiv'15]
- InstanceCut [Kirillov et al, CVPR'17]
- Watershed [Bai & Urtasun, CVPR'17]

- FCIS [Li et al, CVPR'17]
- DIN [Arnab & Torr, CVPR'17]

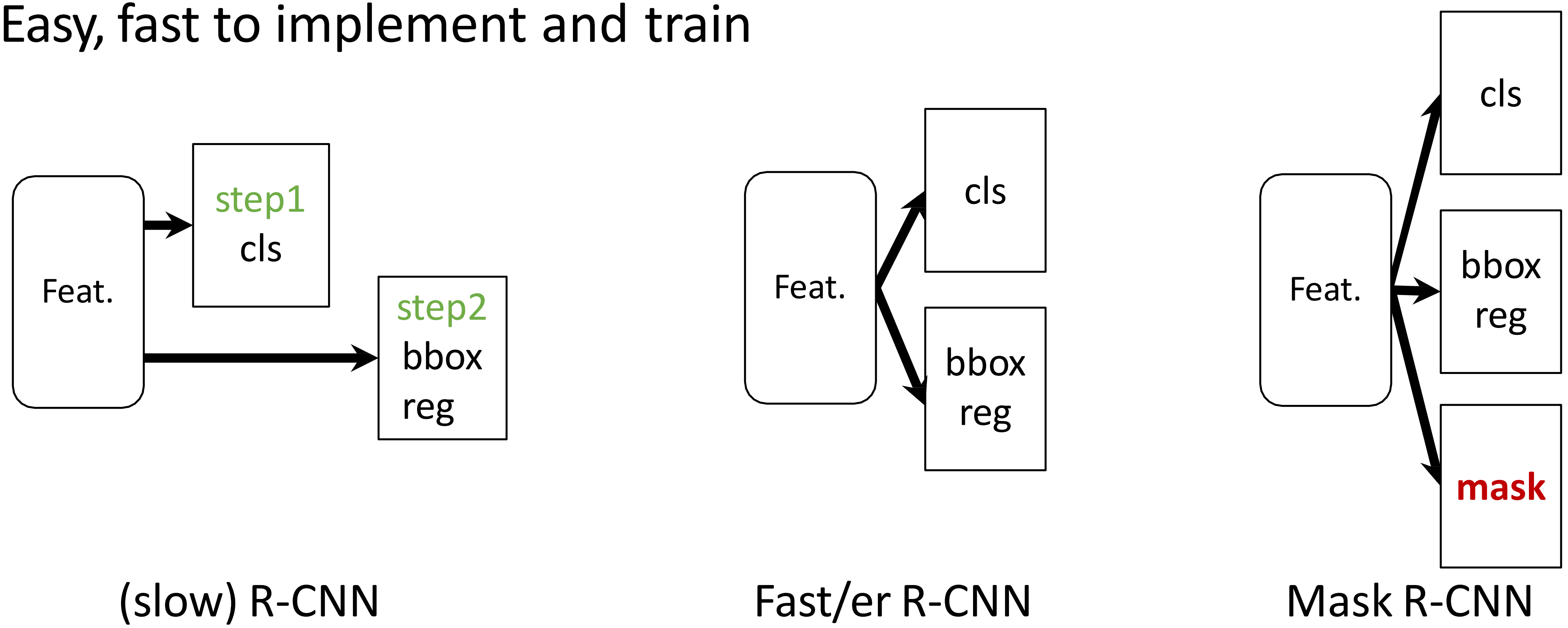
Mask R-CNN

- Mask R-CNN = **Faster R-CNN** with **FCN** on RoIs



Parallel Heads

- Easy, fast to implement and train

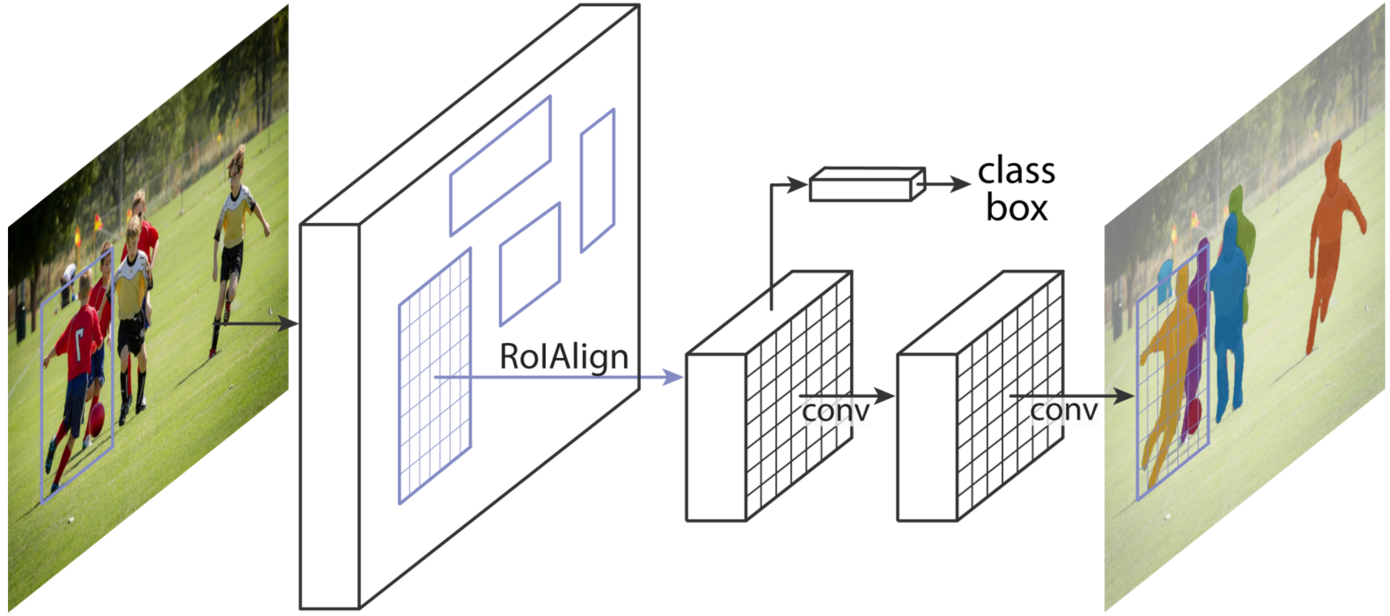


Invariance vs. Equivariance

- **Equivariance**: changes in input lead to corresponding changes in output

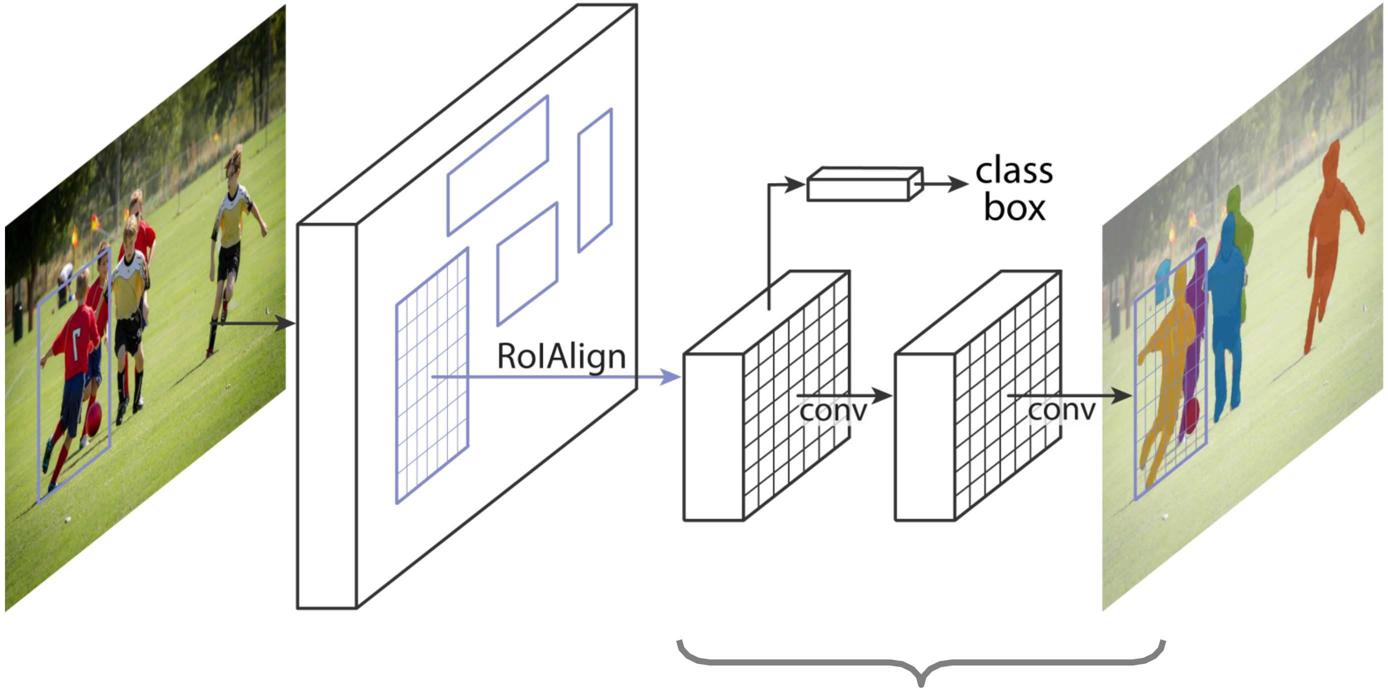
- *Classification* desires *invariant* representations: output a label
- *Instance Seg.* desires *equivariant* representations:
  - Translated object => translated mask
  - Scaled object => scaled mask
  - *Big and small* objects are equally important (due to AP metric)
    - unlike semantic seg. (counting pixels)

Equivariance in Mask R-CNN



1. Fully-Conv Features:
equivariant to global (image) translation

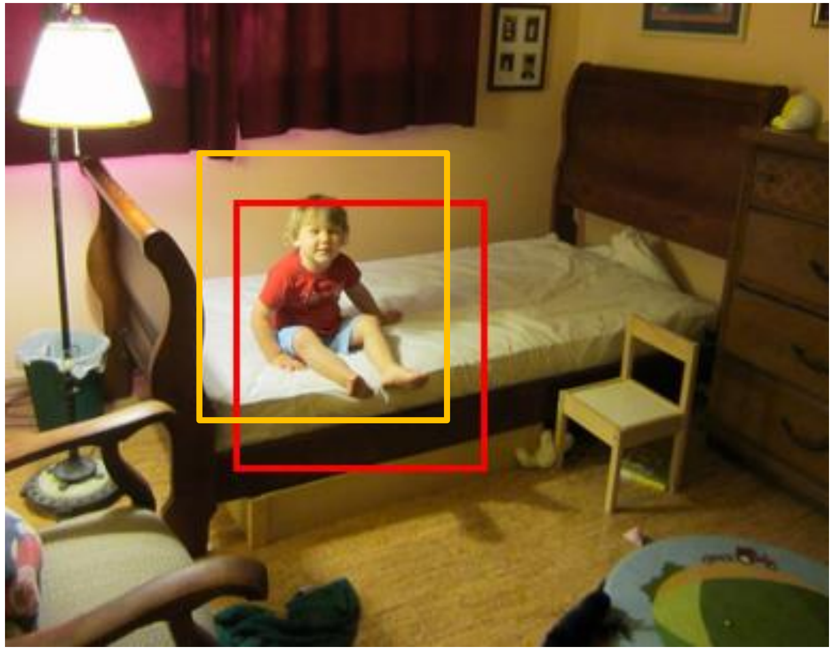
Equivariance in Mask R-CNN

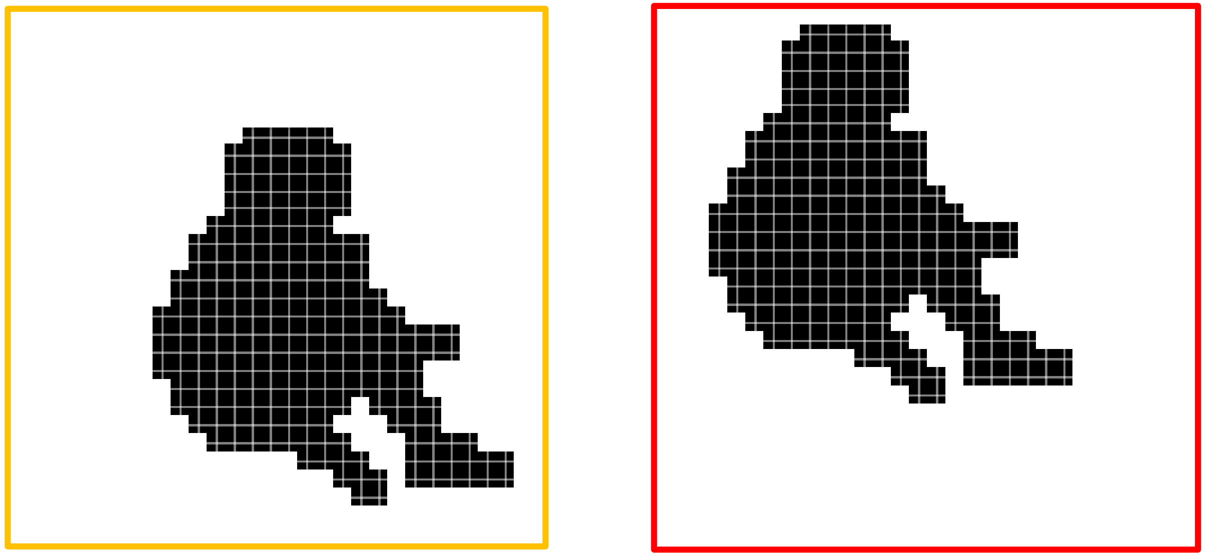


2. Fully-Conv on RoI:
equivariant to translation within RoI

Fully-Conv on RoI

target masks on RoIs

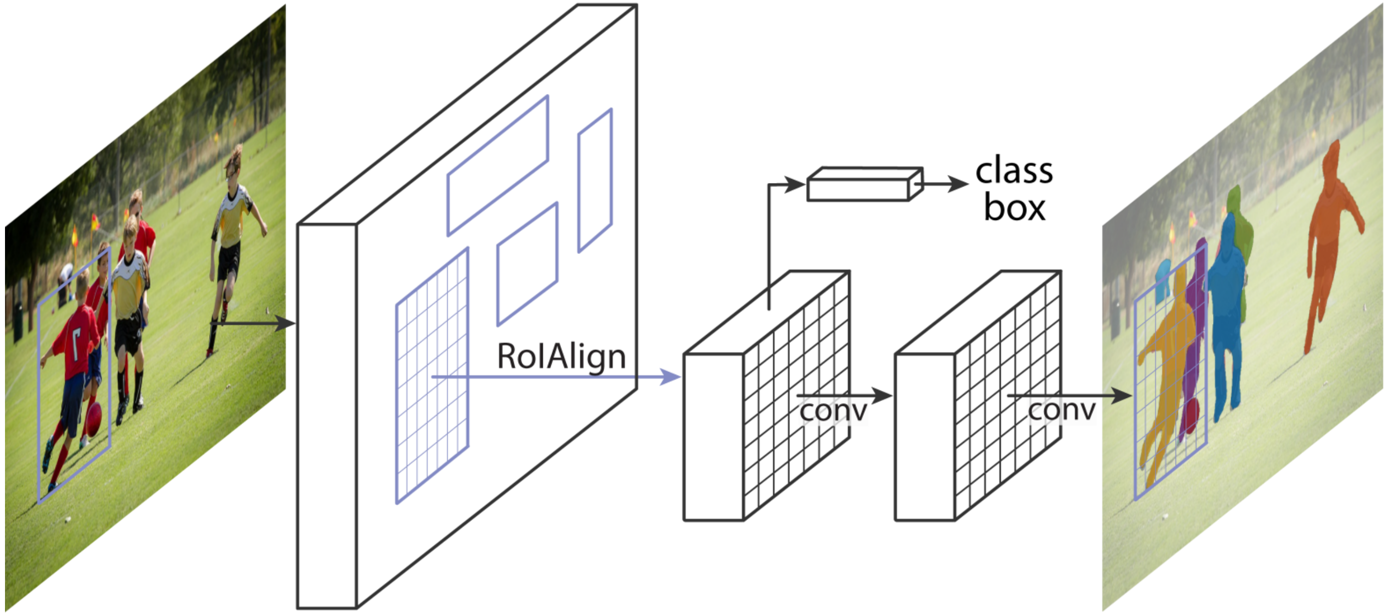




Translation of object in RoI => Same translation of mask in RoI

- Equivariant to small translation of RoIs
- More robust to RoI's localization imperfection

Equivariance in Mask R-CNN



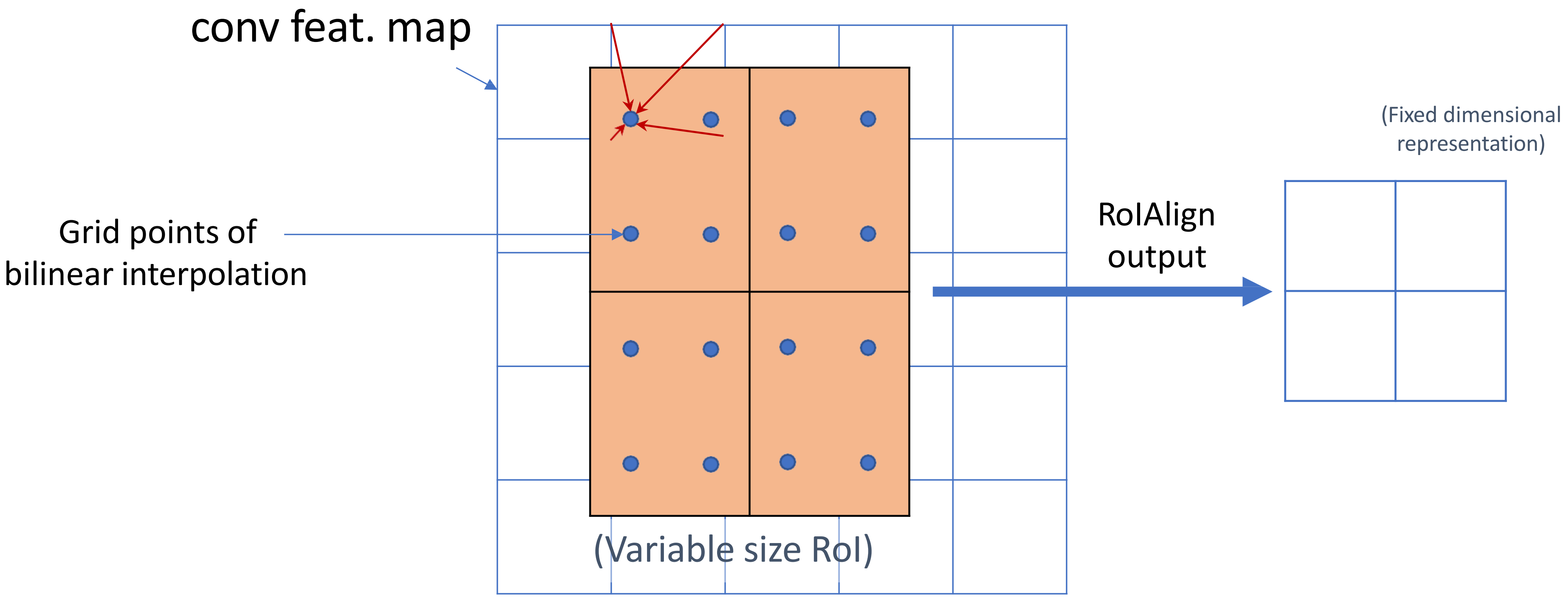
3. RoIAlign:

3a. maintain translation-equivariance before/after RoI

RoIAlign

FAQs: how to sample grid points within a cell?

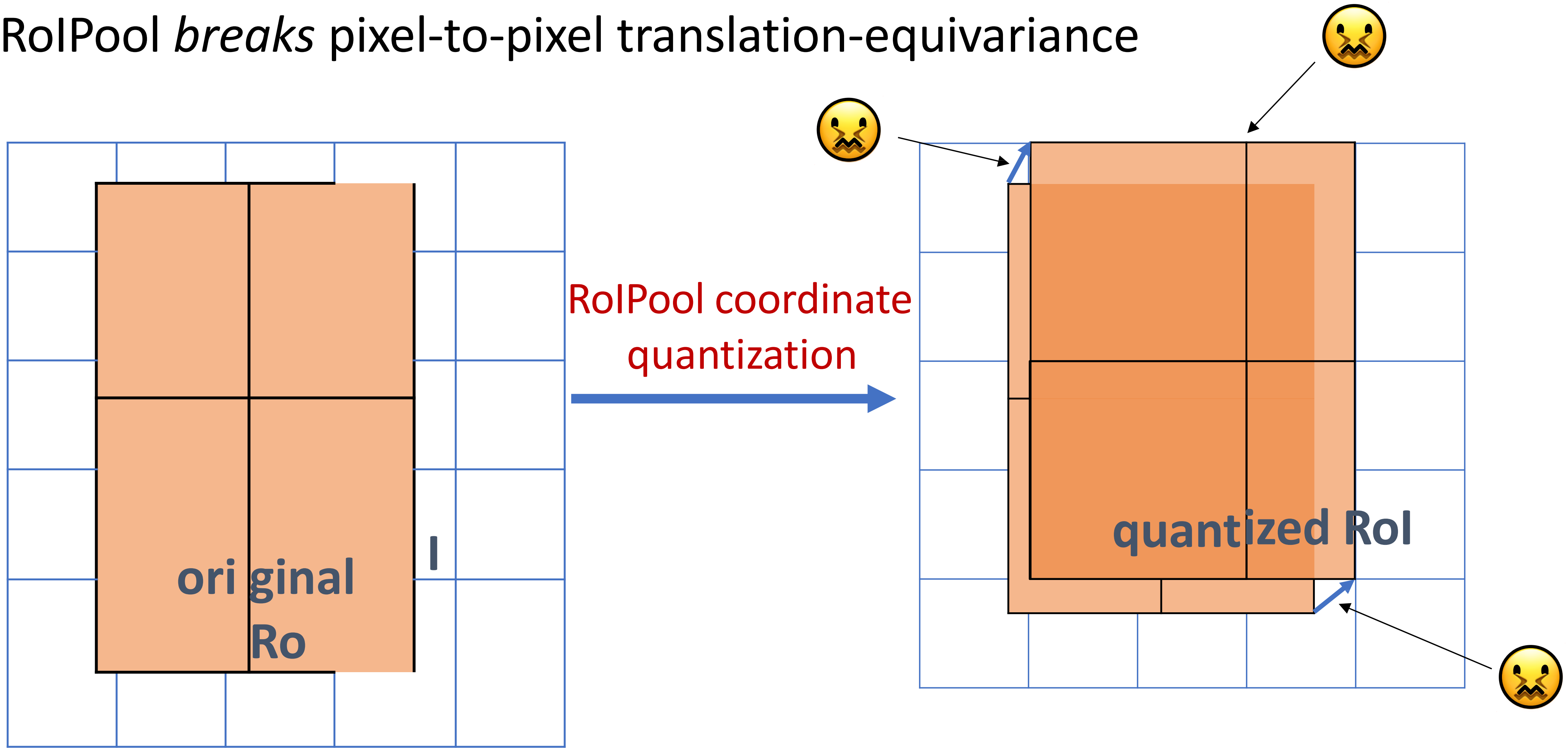
- 4 regular points in 2x2 sub-cells
- other implementation could work



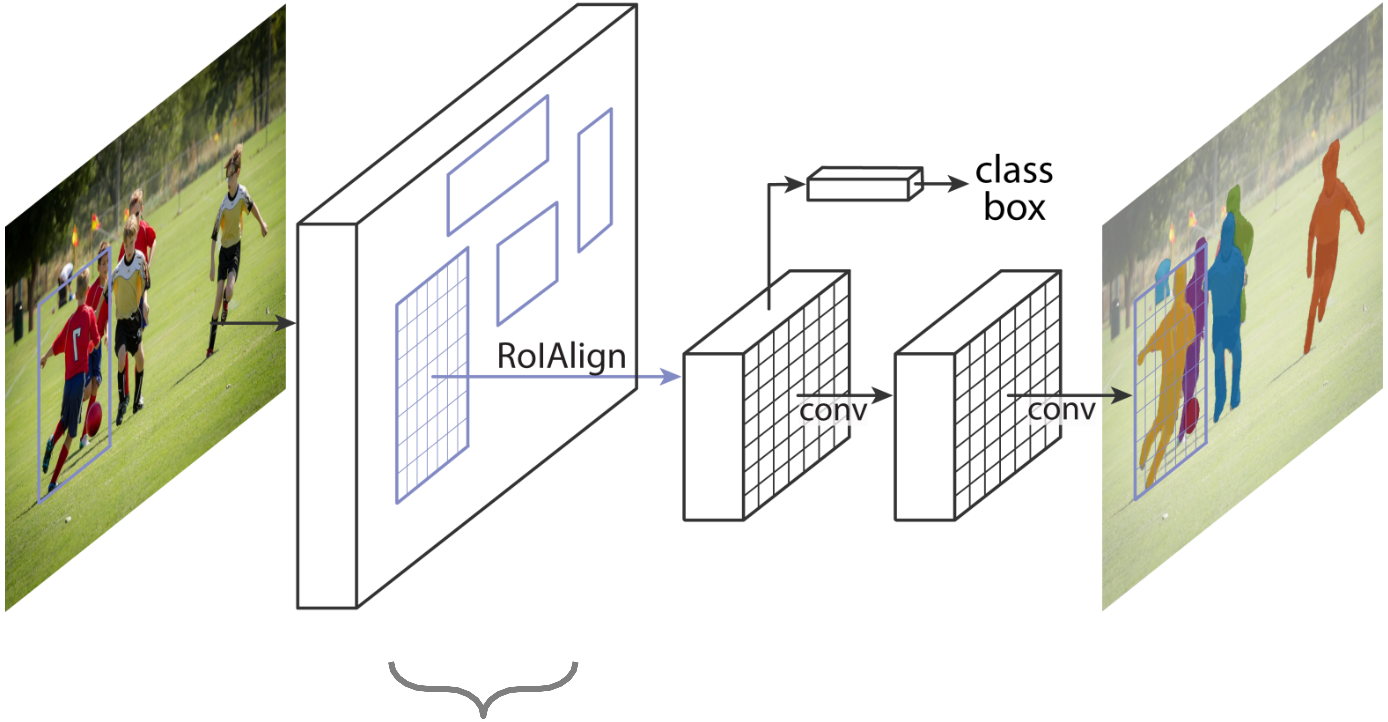
see also “What is wrong with convolutional neural nets?”, Geoffrey Hinton, 2017

RoIAlign vs. RoIPool

- RoIPool *breaks* pixel-to-pixel translation-equivariance



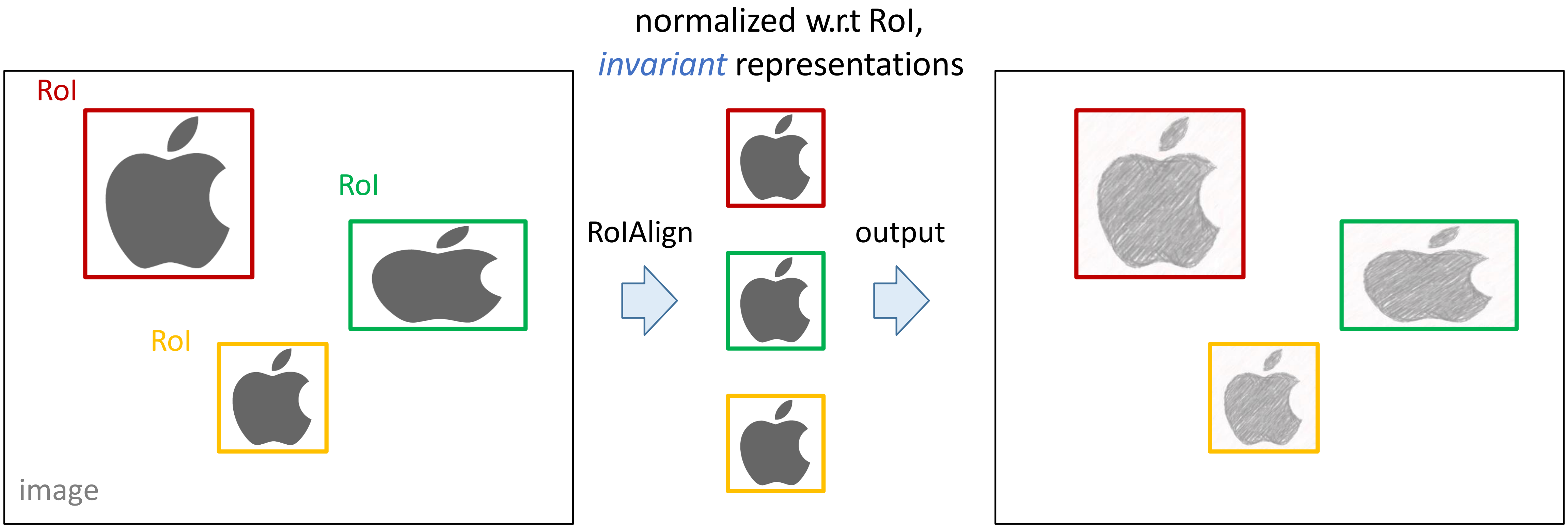
Equivariance in Mask R-CNN



3. RoIAlign:

3b. Scale-equivariant (and aspect-ratio-equivariant)

RoIAlign: Scale-Equivariance

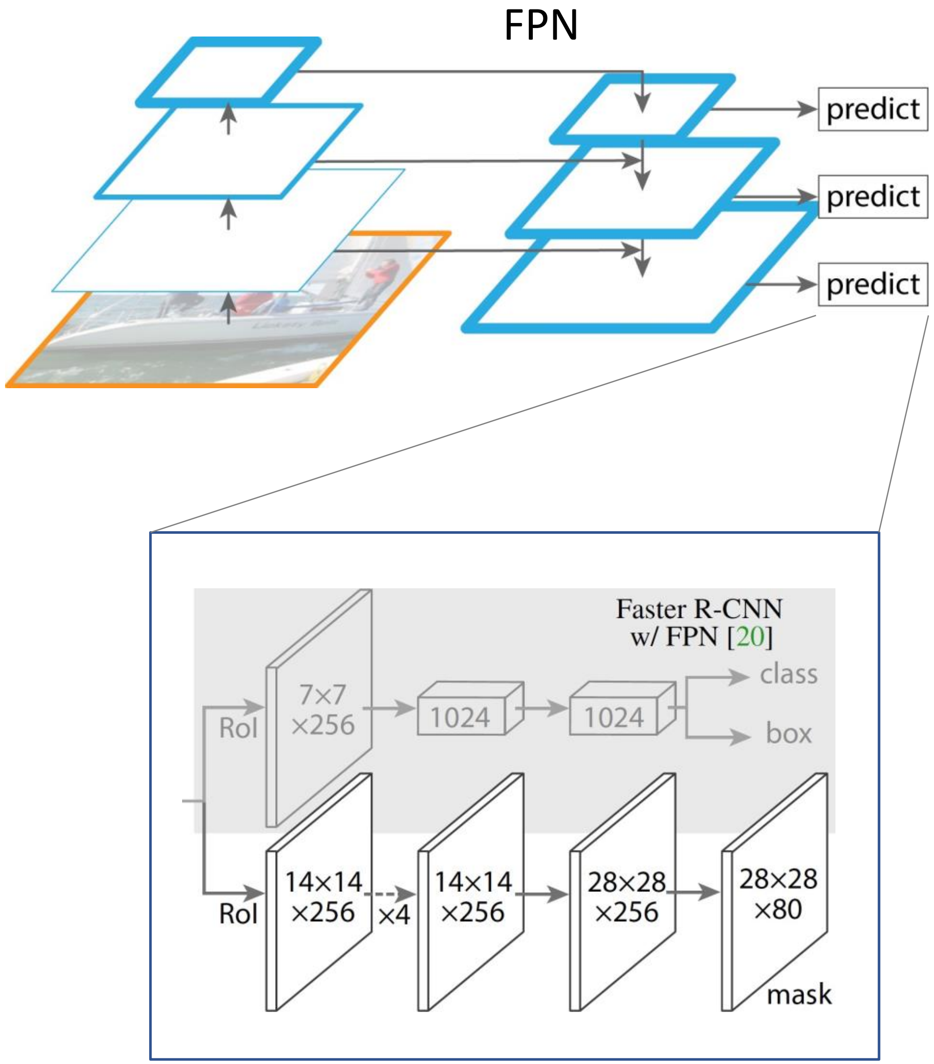


- RoIAlign creates *scale-invariant* representations
- RoIAlign + "output pasted back" provides *scale-equivariance*

More about Scale-Equivariance: FPN

- RoIAlign is scale-invariant if on raw pixels:
  - = (slow) R-CNN: crops and warps RoIs
- RoIAlign is scale-invariant if on scale-invariant feature maps

- Feature Pyramid Network (FPN) [Lin et al. CVPR'17] creates approx. scale-invariant features



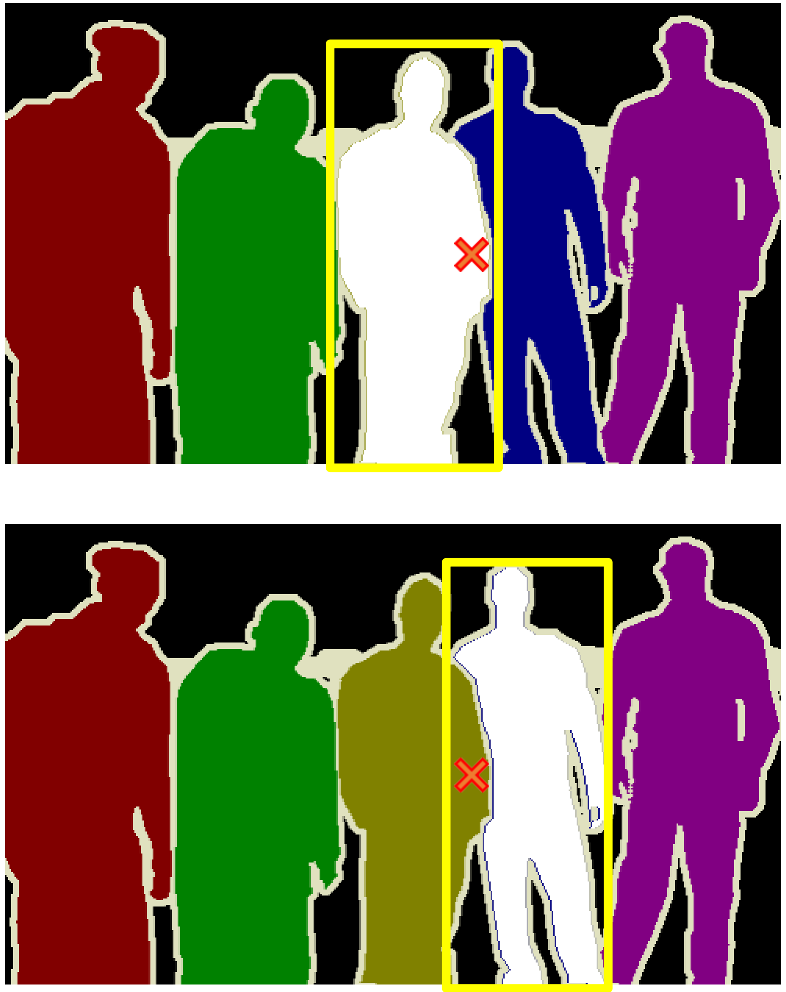
Equivariance in Mask R-CNN: Summary

- Translation-equivariant
  - FCN features
  - FCN mask head
  - RoIAlign (pixel-to-pixel behavior)

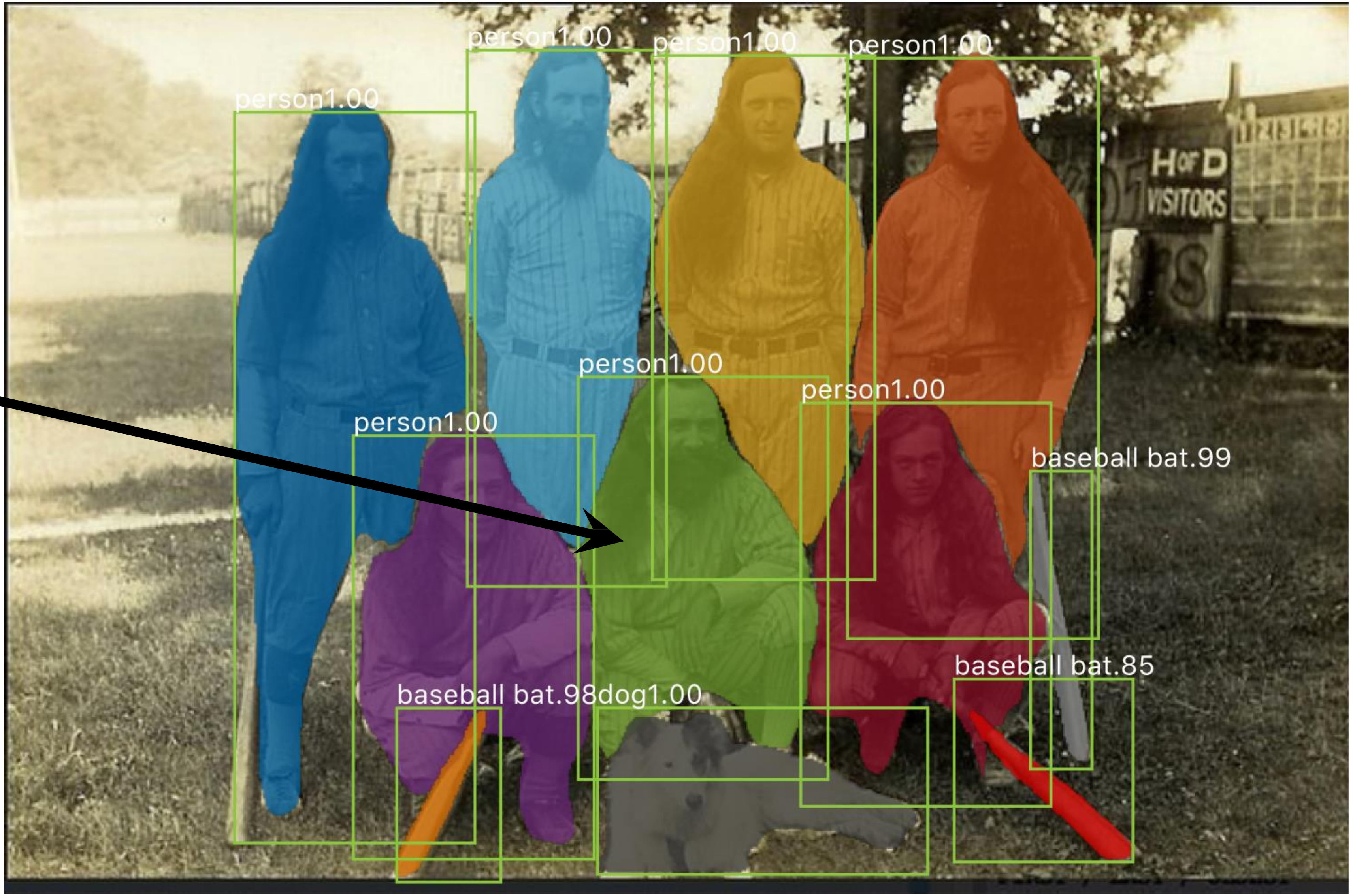
- Scale-equivariant (and aspect-ratio-equivariant)
  - RoIAlign (warping and normalization behavior) + paste-back
  - FPN features

Instance Seg: When we don’t want equivariance?

- A pixel x could have a different label w.r.t. different RoIs
  - zero-padding in RoI boundary breaks equivariance
  - outside objects are suppressed
  - only equivariant to small changes of RoIs (which is desired)



object surrounded by same-category objects



Mask R-CNN results on COCO

Result Analysis

Ablation: RoIPool vs. RoIAlign

baseline: ResNet-50-Conv5 backbone, **stride=32**

	mask AP			box AP		
	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	$\boxed{+10.5}$	*+5.8*	*+2.6*	*+9.5*

- huge gain at high IoU, in case of big stride (32)

Ablation: RoIPool vs. RoIAlign

baseline: ResNet-50-Conv5 backbone, **stride=32**

	mask AP			box AP		
	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

- nice box AP without dilation/upsampling

Instance Segmentation Results on COCO

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [7]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [20] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [20] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

- **2 AP better** than SOTA w/ R101, without bells and whistles
- **200ms / img**

Instance Segmentation Results on COCO

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [7]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [20] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [20] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

- benefit from better features (ResNeXt [Xie et al. CVPR'17])

Object Detection Results on COCO

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{S}	AP^{bb}_{M}	AP^{bb}_{L}
Faster R-CNN+++ [15]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [22]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [32]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [31]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

bbox detection improved by:

- RoIAlign

Object Detection Results on COCO

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{S}	AP^{bb}_{M}	AP^{bb}_{L}
Faster R-CNN+++ [15]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [22]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [32]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [31]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

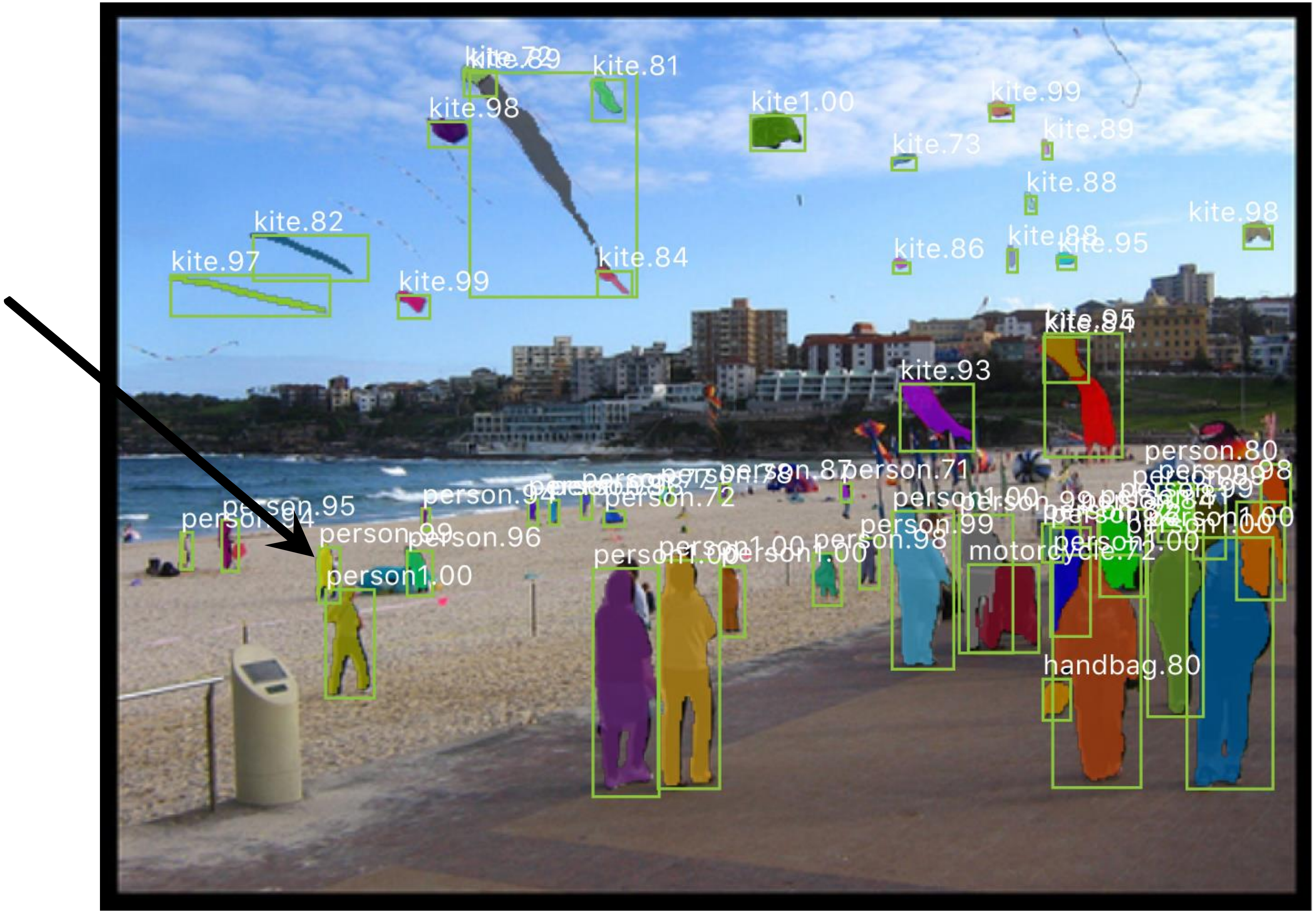
bbox detection improved by:

- RoIAlign
- Multi-task training w/ mask

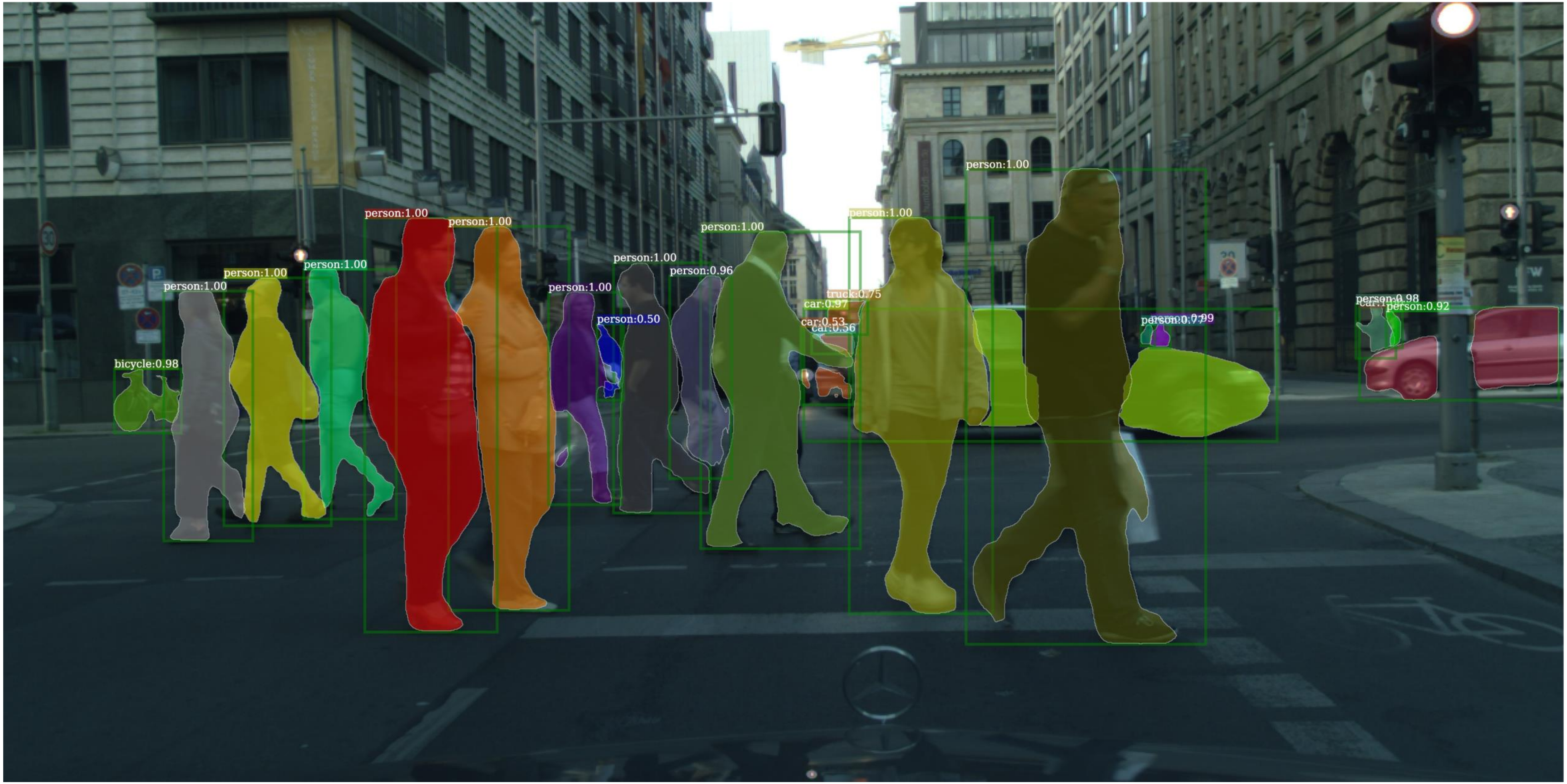
disconnected object

Mask R-CNN results on COCO

small
objects



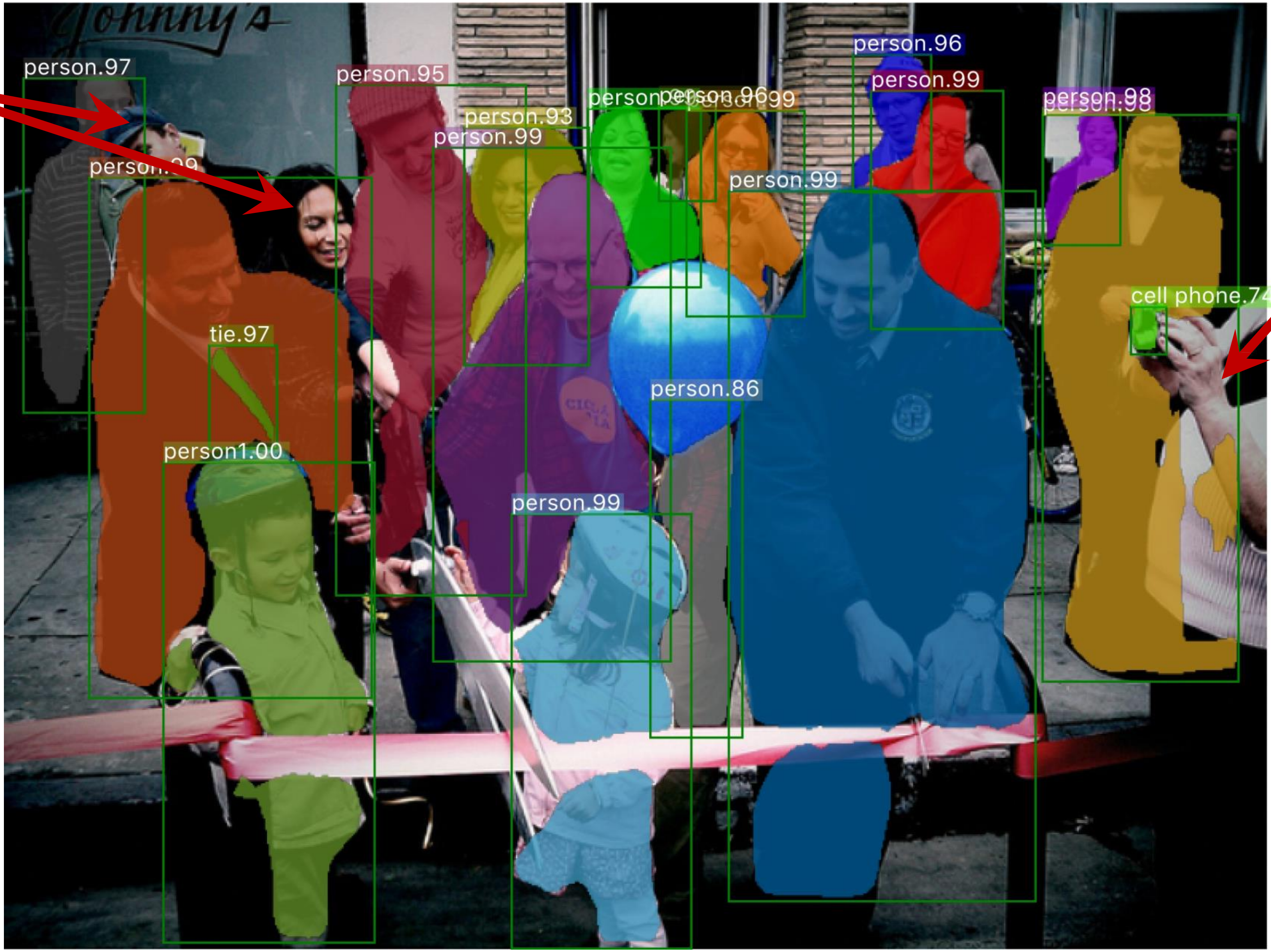
Mask R-CNN results on COCO



Mask R-CNN results on CityScapes

Failure case: detection/segmentation

missing

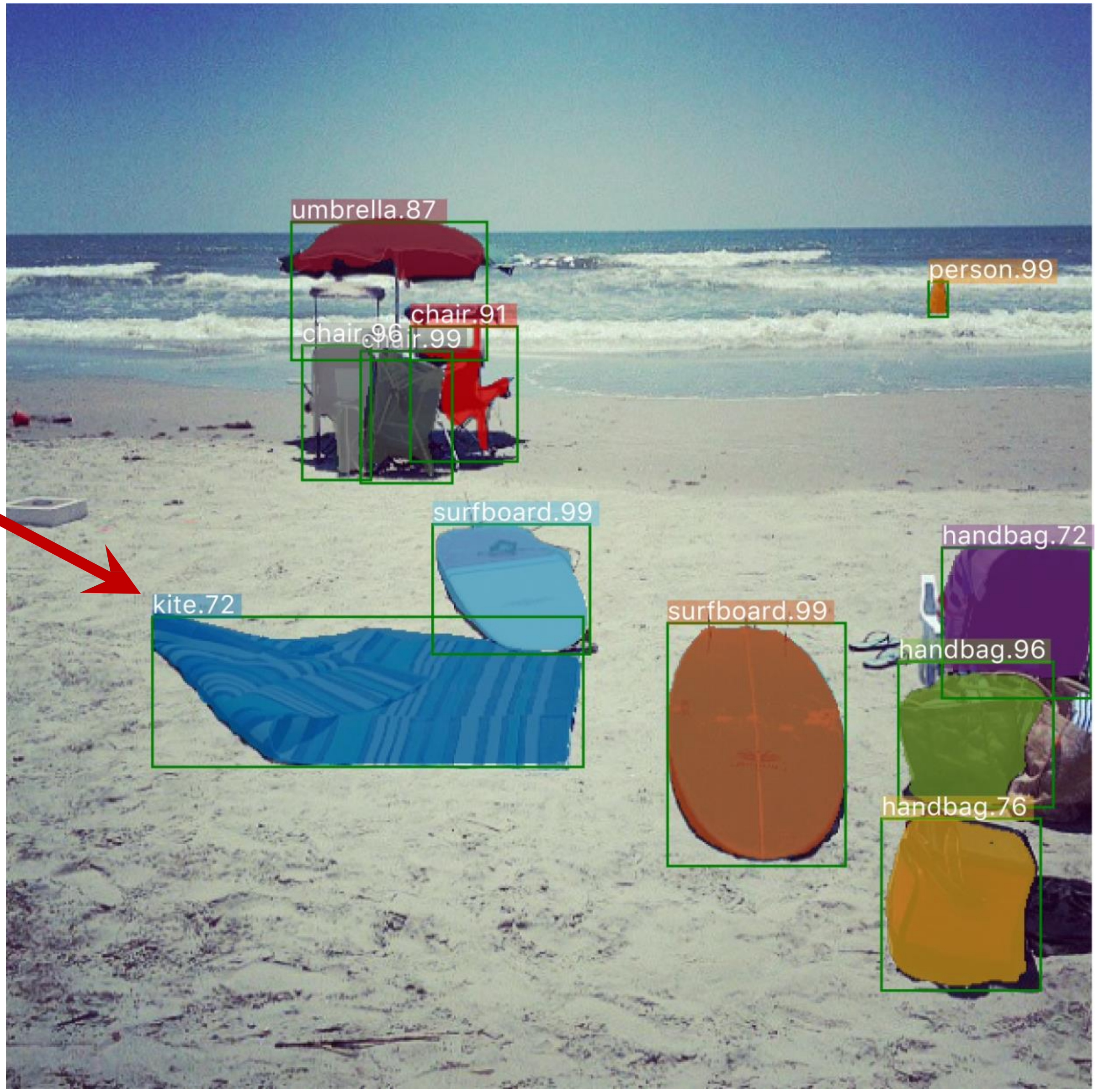


missing,
false mask

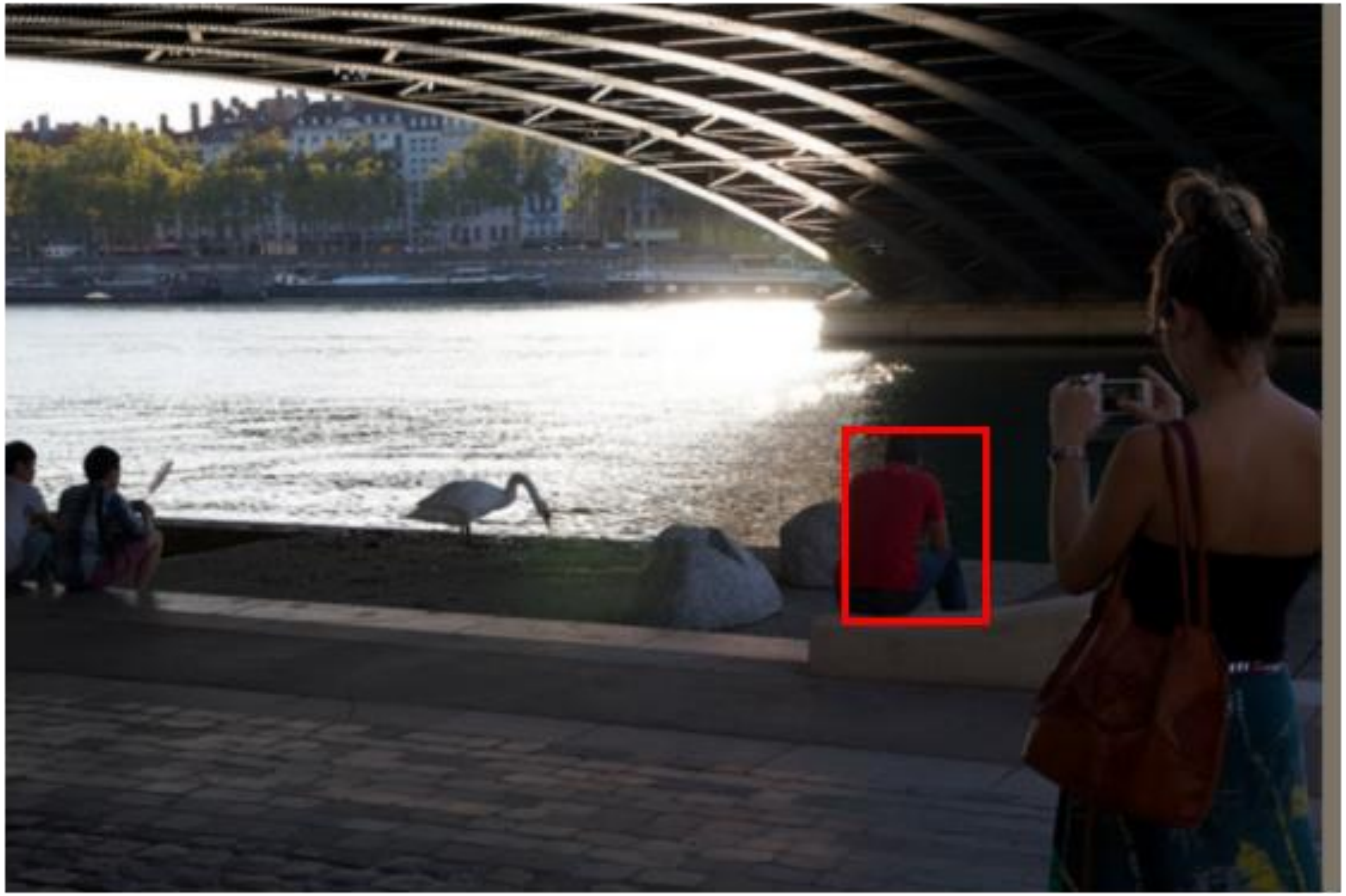
Mask R-CNN results on COCO

Failure case: recognition

not a kite



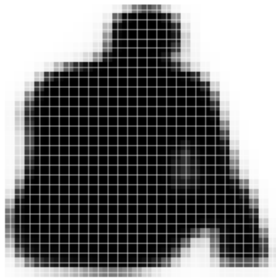
Mask R-CNN results on COCO



Validation image with box detection shown in red

28x28 soft prediction from Mask R-CNN

(enlarged)



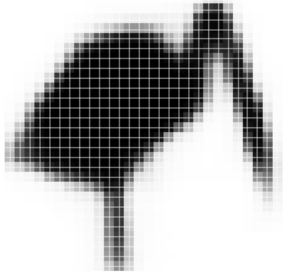
Soft prediction resampled to image coordinates

(bilinear and bicubic interpolation work equally well)

Final prediction (threshold at 0.5)

Validation image with box detection shown in red

28x28 soft prediction



Resized Soft prediction

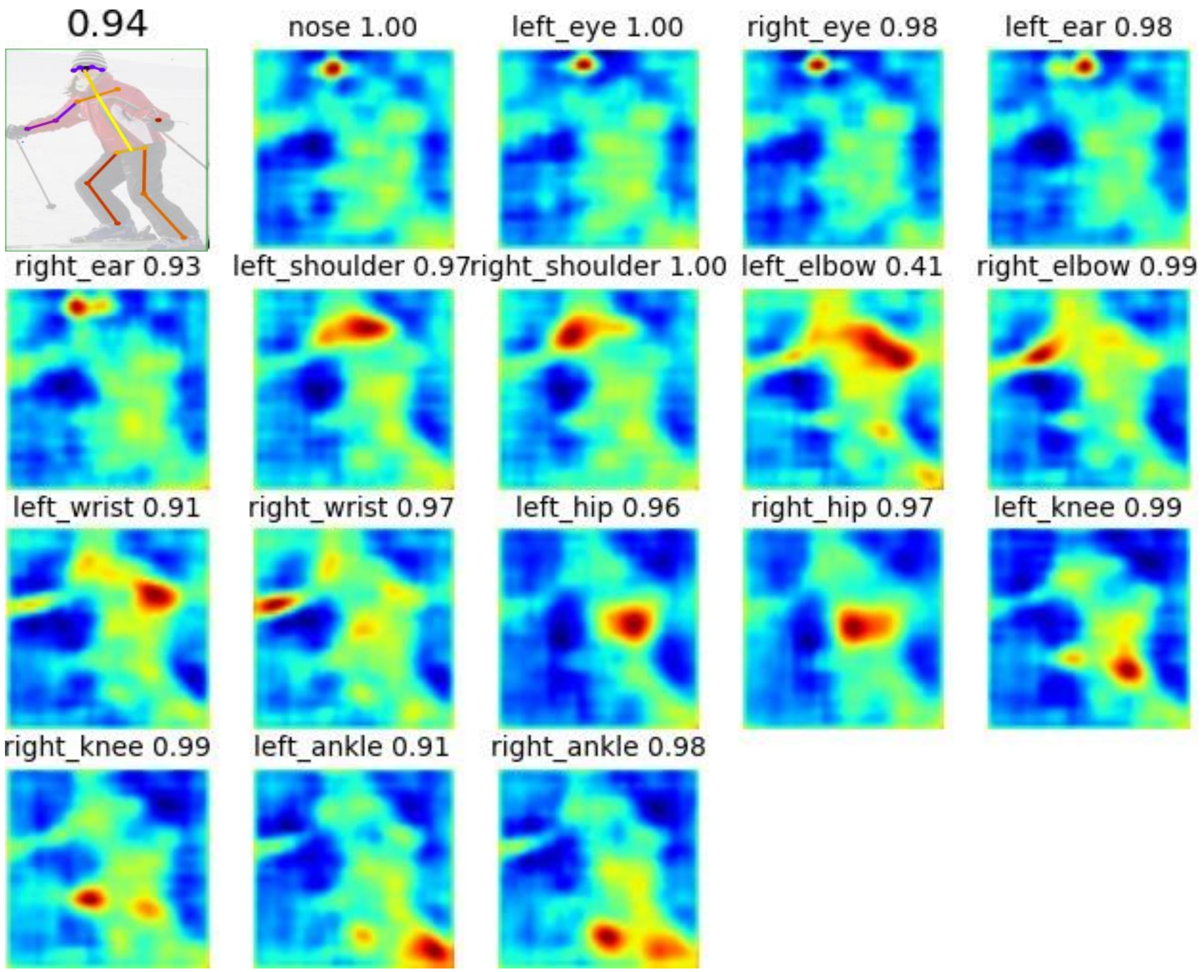
Final mask

Mask R-CNN: for Human Keypoint Detection

- 1 keypoint = 1-hot "mask"
- Human pose = 17 masks

- Softmax over spatial locations
    - e.g. 56^2-way softmax on 56x56

- Desire the same equivariances
    - translation, scale, aspect ratio

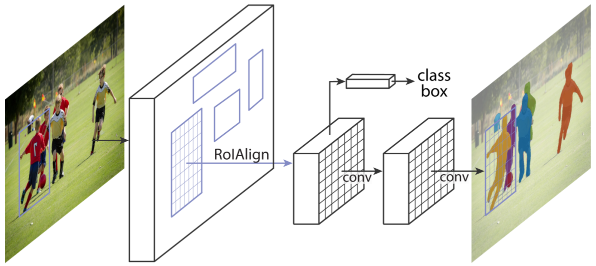



www.tfrancis.co.uk

Conclusion

Mask R-CNN

✓Good speed
✓Good accuracy
✓Intuitive
✓Easy to use
✓Equivariance matters

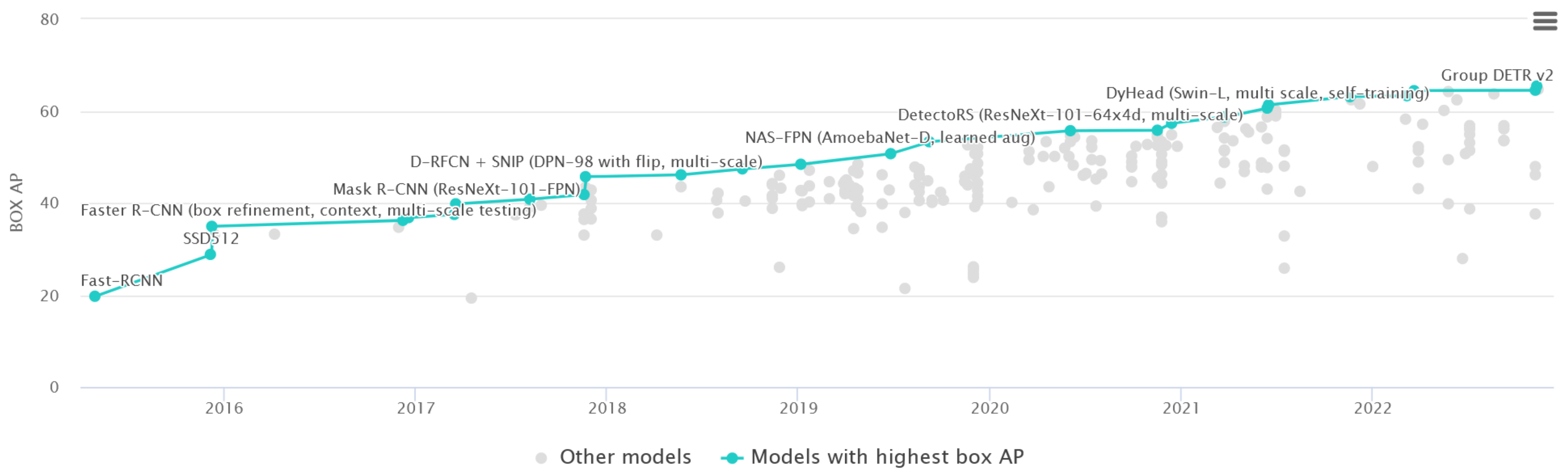


Code will be open-sourced as
Facebook AI Research's **Detectron** platform

Object Detection on COCO test-dev

Leaderboard Dataset

View box AP by Date for All models



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

Summary – More complex outputs from deep networks

- Image Output (e.g. colorization, semantic segmentation, super-resolution, stylization, depth estimation…)
- Attributes
- Text Captions
- Semantic Keypoints
- Object Detection