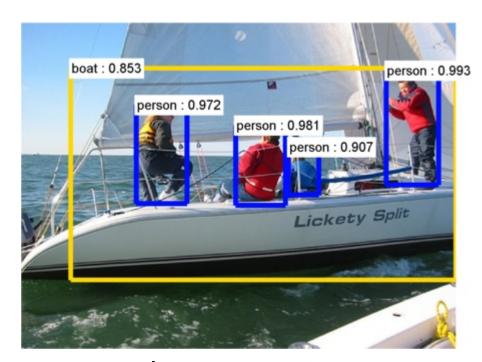
Object Detection

Lecture 5

Object Detection

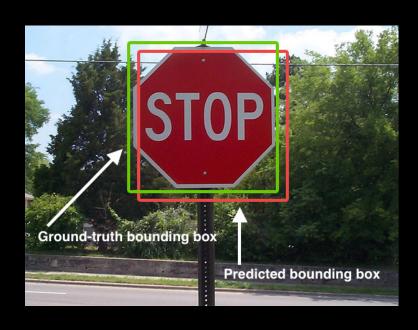


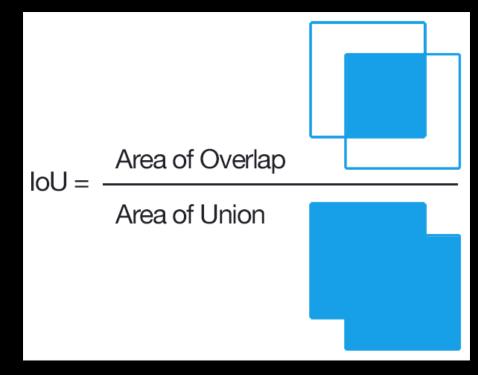
Image Classification (what?)

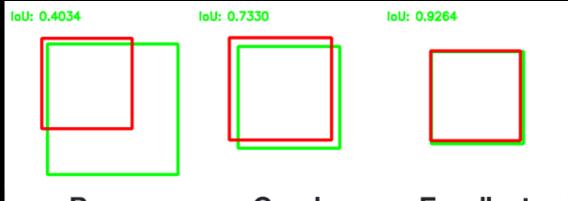


Object Detection (what + where?)

Intersection over Union (IoU) metric





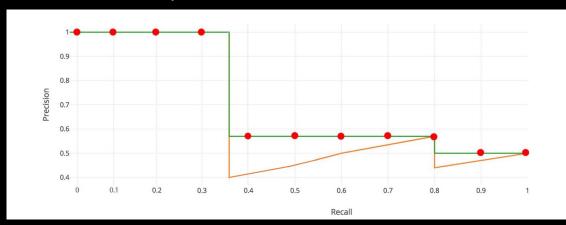


https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Mean Average Precision (mAP) metric

$$Precision = \frac{TP}{TP + FP}$$
 $TP = True positive$ $TN = True negative$ $TP = True negative$ $TN = True negative$ $TP = True positive$ $TP = True positive$

Recall-precision curve



List of bounding boxes across entire dataset, ranked by detector confidence:

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

Correct = IoU > 0.5

Orange = recall-precision Green = interpolated Red dots used to compute average

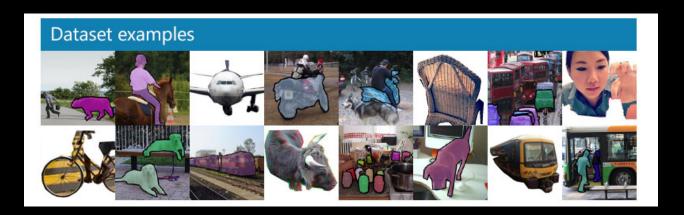
COCO metrics

Common Objects in Context https://cocodataset.org/#home

Leading detection benchmark

COCO metrics

```
Average Precision (AP):
                          % AP at IoU=.50:.05:.95 (primary challenge metric)
   ΔPIoU=.50
                          % AP at IoU=.50 (PASCAL VOC metric)
   ΔP<sup>IoU=.75</sup>
                          % AP at IoU=.75 (strict metric)
AP Across Scales:
   APsmall
                          % AP for small objects: area < 32<sup>2</sup>
   Δpmedium
                          % AP for medium objects: 32^2 < area < 96^2
   Aplarge
                          % AP for large objects: area > 96<sup>2</sup>
Average Recall (AR):
   AR<sup>max=1</sup>
                          % AR given 1 detection per image
   ARmax=10
                          % AR given 10 detections per image
   ARmax=100
                          % AR given 100 detections per image
AR Across Scales:
   AR<sup>small</sup>
                          % AR for small objects: area < 32<sup>2</sup>
   ARmedium
                          % AR for medium objects: 32^2 < area < 96^2
  ARlarge
                          % AR for large objects: area > 96<sup>2</sup>
```



Typical results table in paper (e.g. YOLOv3)

	backbone	AP	AP_{50}	AP ₇₅	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [4]	Inception-ResNet-v2 [19]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [18]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
One-stage methods							
YOLOv2 [13]	DarkNet-19 [13]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9, 2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [2]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [7]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [7]	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9
COCO for YOLOv3							

Viola and Jones (2001)

First effective real-time face detector

Integral Image trick for fast computation of features

Boosting-based learning

Deployed in numerous real-world applications

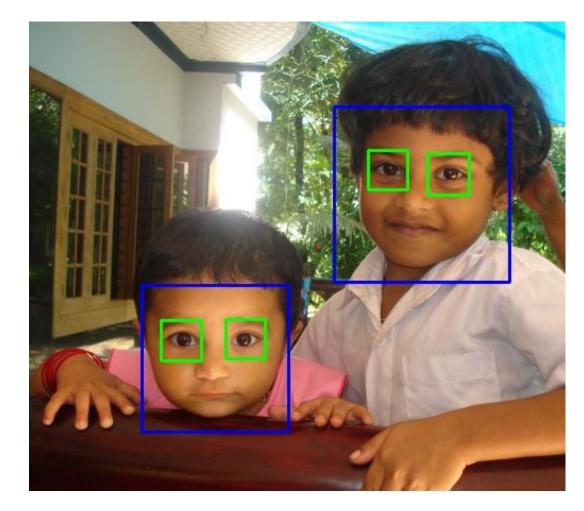
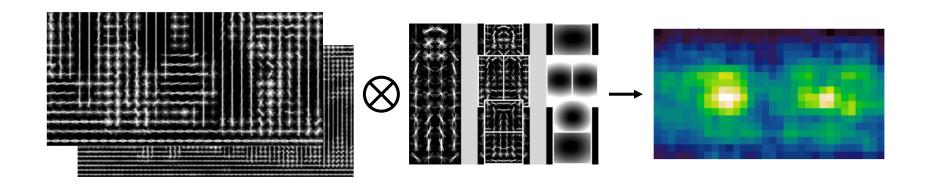


Image from OpenCV 3.3 website

DPM: Deformable Part Models

Object Detection with Discriminatively Trained Part Based Models Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester and Deva Ramanan, PAMI 2009



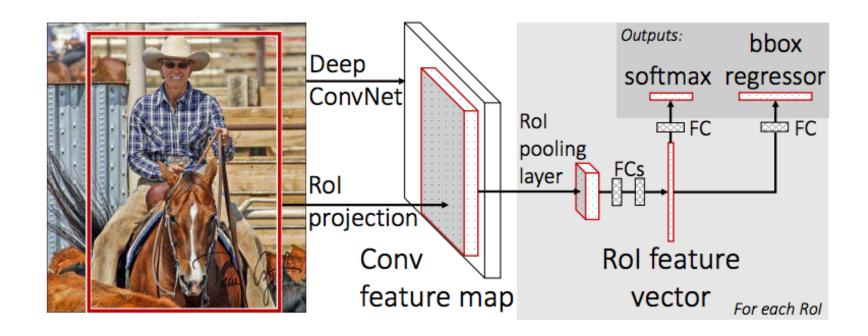
Leading detection approach pre-deep learning

HOG-based features

Small number of deformable part detectors

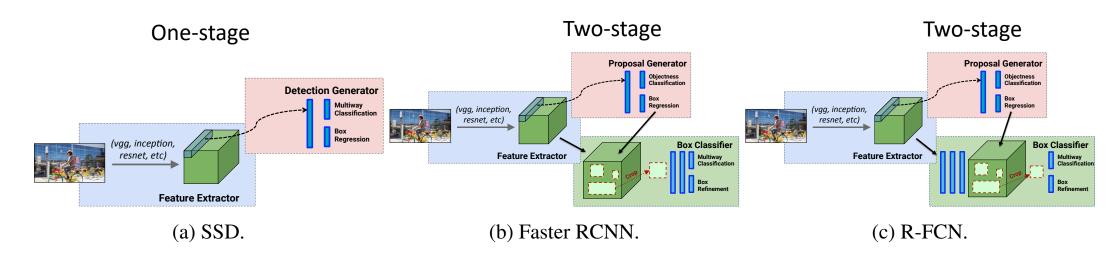
Fast R-CNN

- Deep learning-based approach
- Part of current leading direction in detection

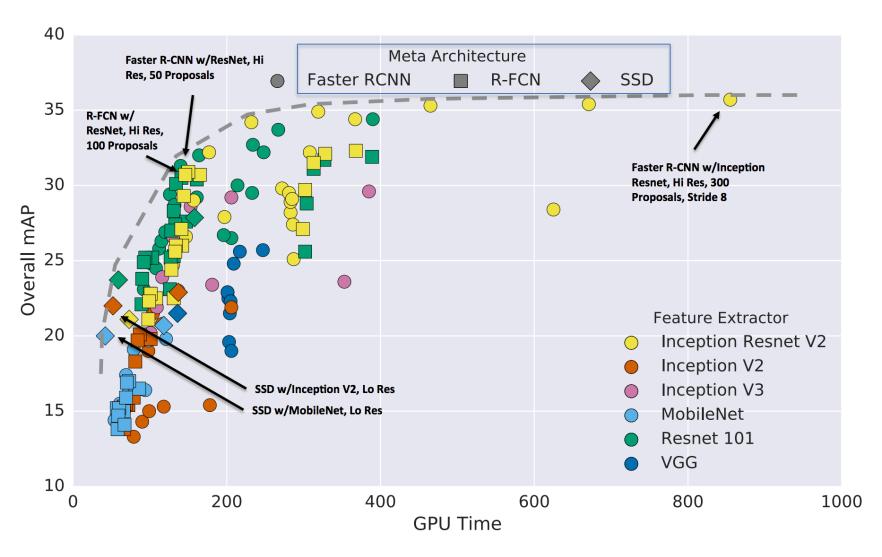


One-stage vs. Two-stage

- One-stage
 - Fast
 - Simple
- Two-stage
 - 10 40% better accuracy



One-stage vs. Two-stage



Speed/accuracy trade-offs for modern convolutional object detectors, Huang et al., CVPR 2017

One-Stage Detectors

JOSEPH REDMON

Ross

GIRSHICK

SANTOSH

ALI

DIVVALA

FARHADI









"You ONLY LOOK ONCE"

REAL-TIME DETECTION

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

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

	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

	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

	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



	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	

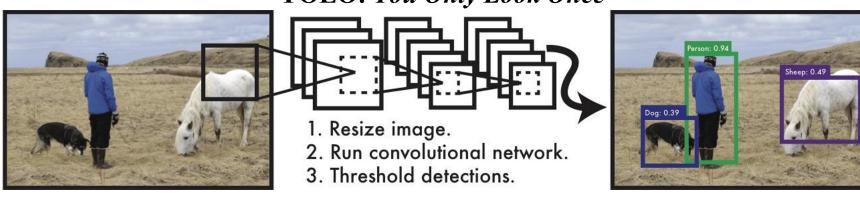


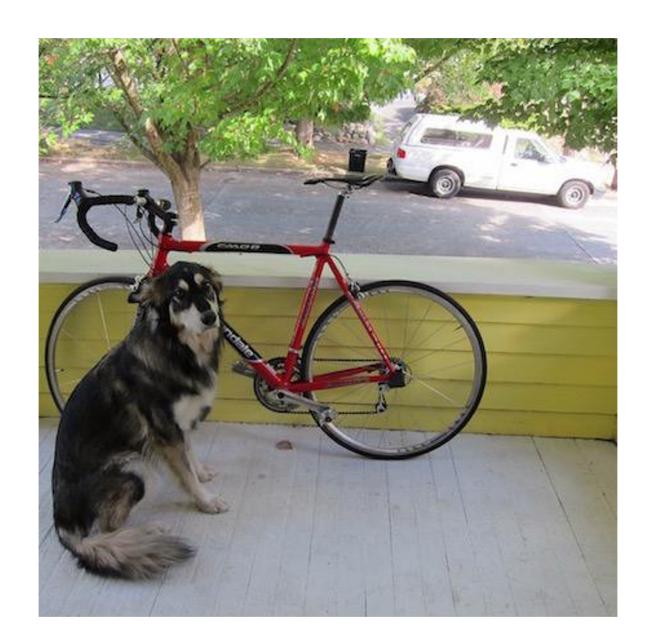
	Pascal 2007 mAP	Speed		
DPM v5	33.7	.07 FPS	14 s/img	
R-CNN	66.0	.05 FPS	20 s/img	
Fast R-CNN	70.0	.5 FPS	2 s/img	
Faster R-CNN	73.2	7 FPS	140 ms/img	
YOLO	69.0	45 FPS	22 ms/img	



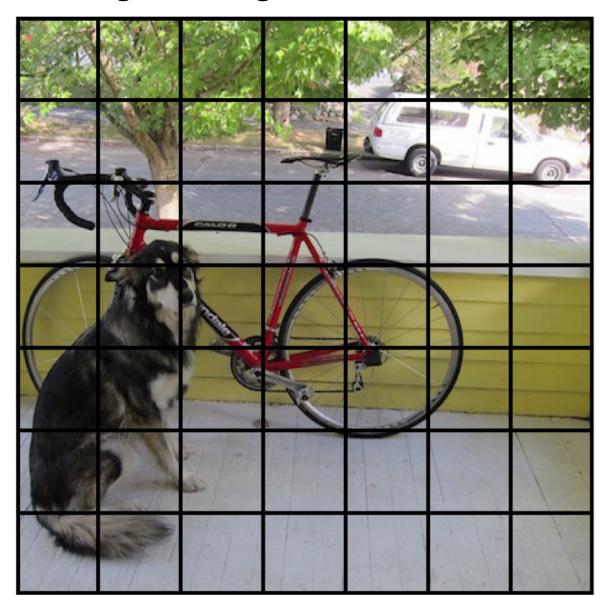
With YOLO, you only look once at an image to perform detection

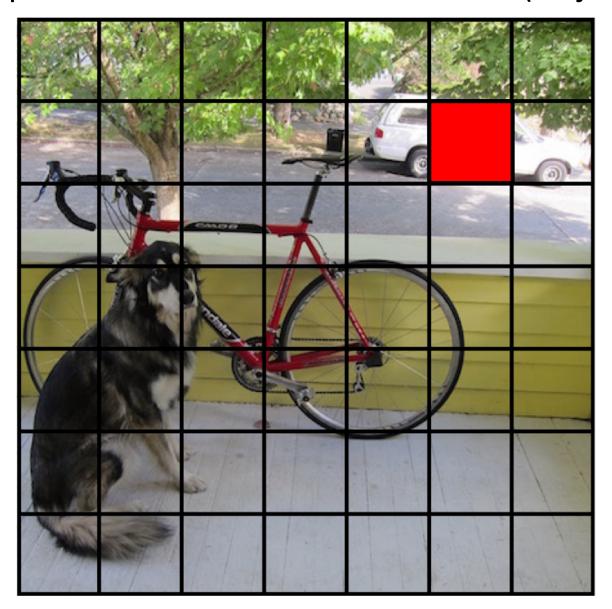




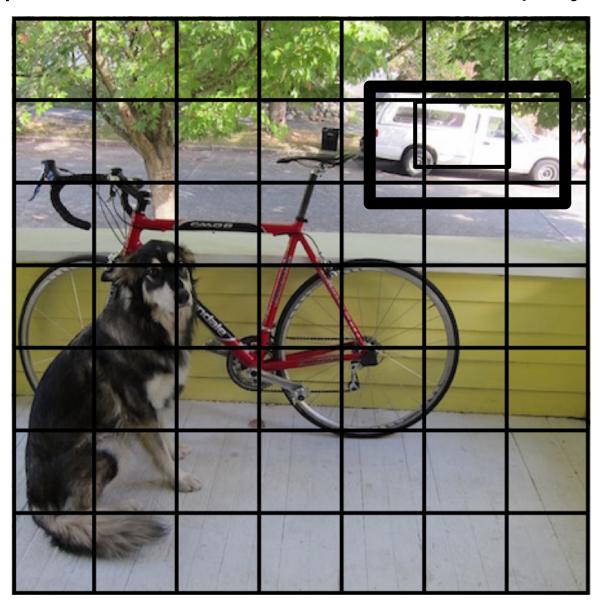


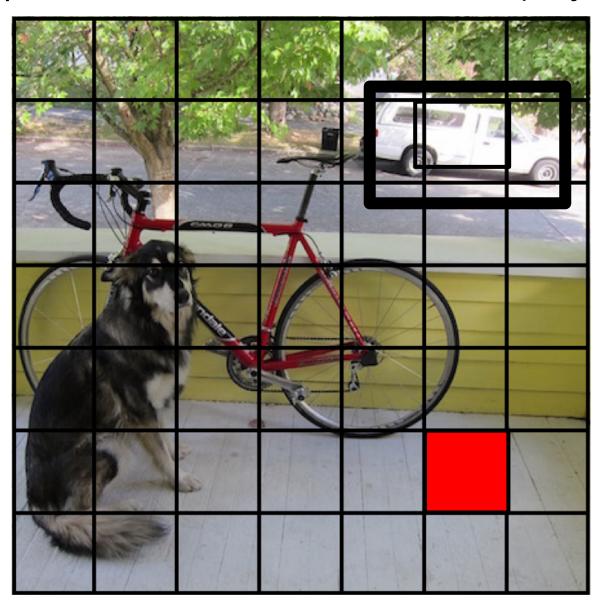
We split the image into a grid

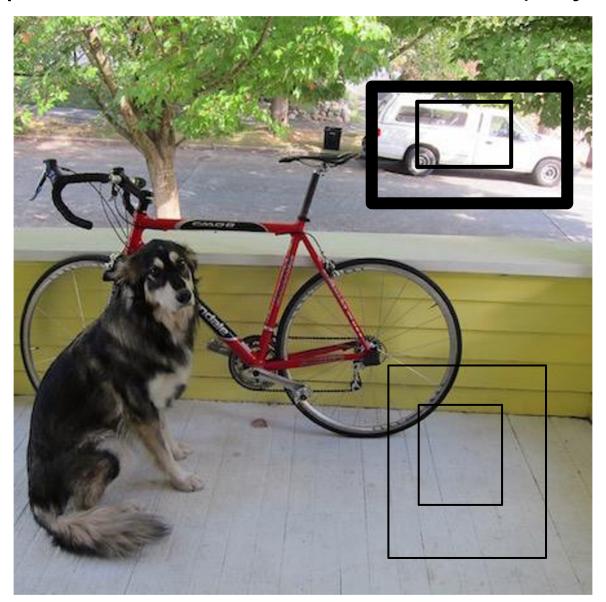


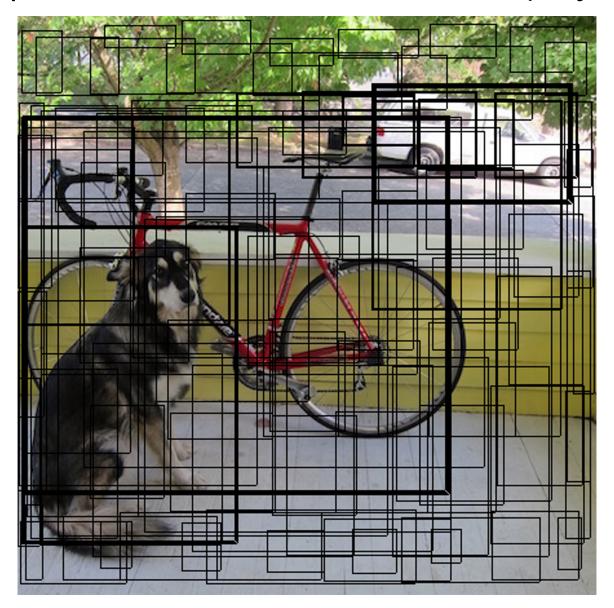




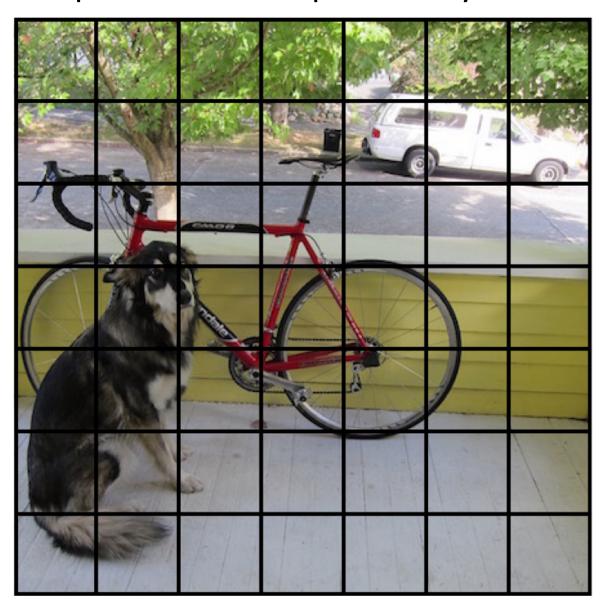




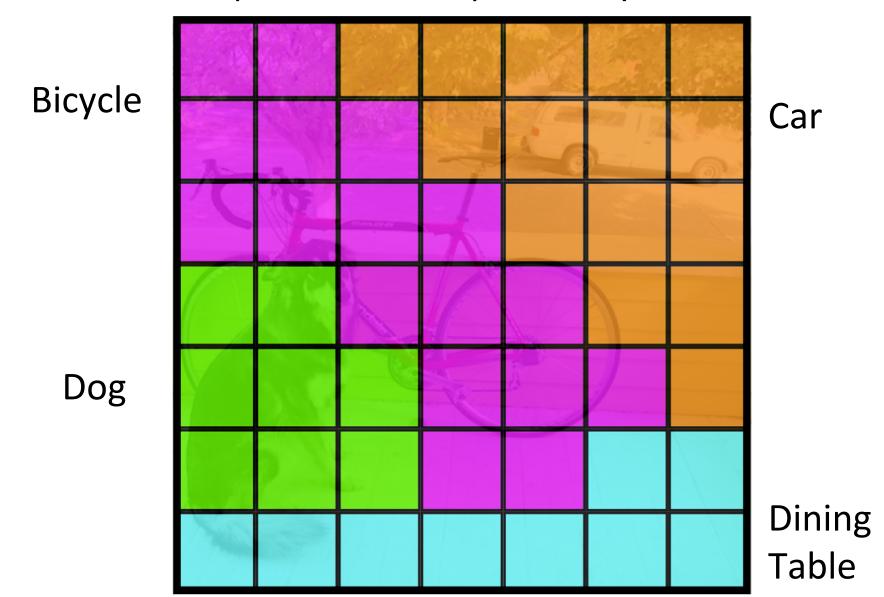




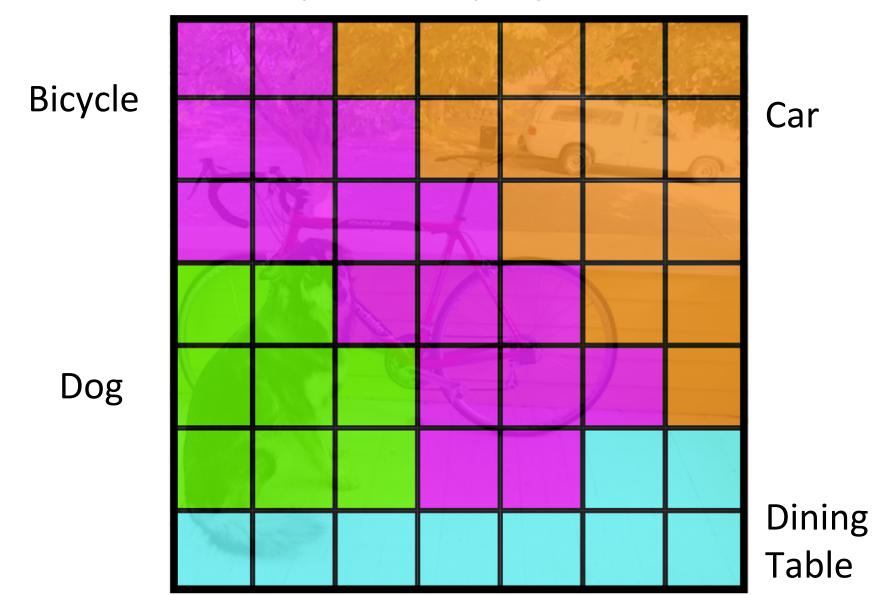
Each cell also predicts a class probability.



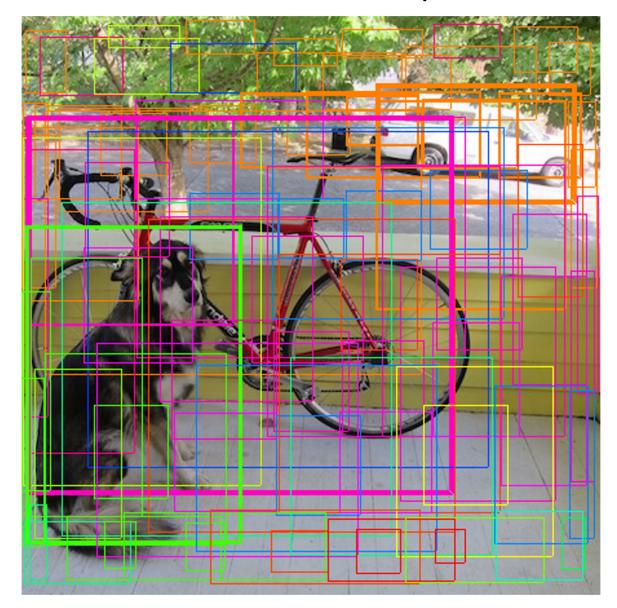
Each cell also predicts a class probability.



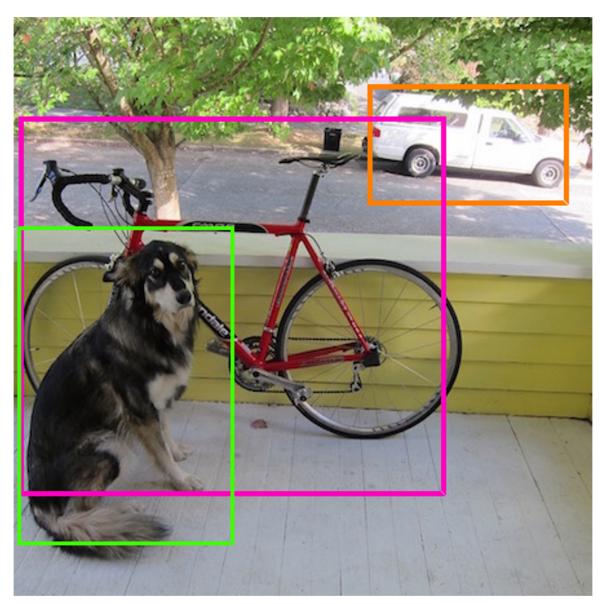
Conditioned on object: P(Car | Object)



Then we combine the box and class predictions.



Finally we do NMS and threshold detections



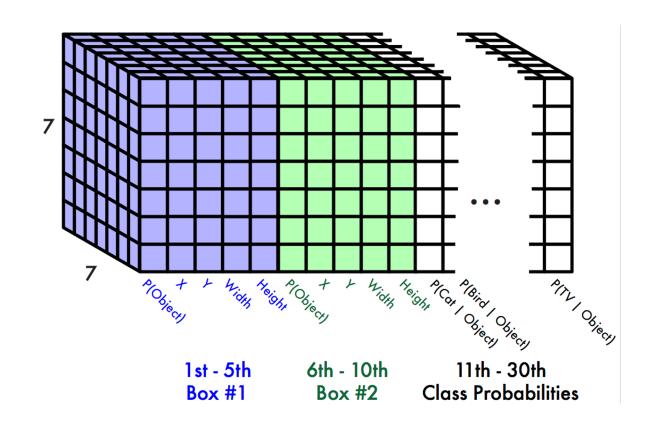
This parameterization fixes the output size

Each cell predicts:

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

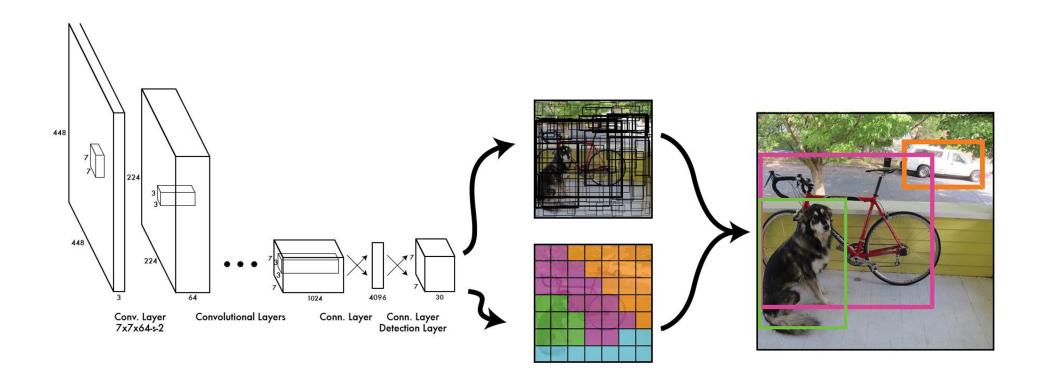
For Pascal VOC:

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

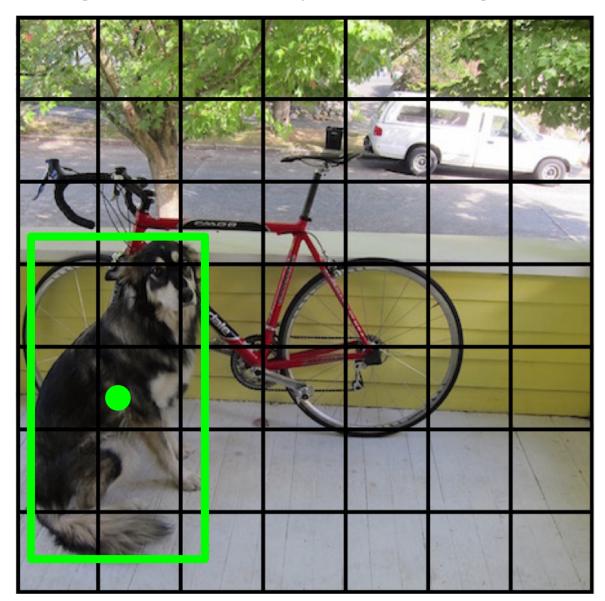


 $7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30 \text{ tensor} = 1470 \text{ outputs}$

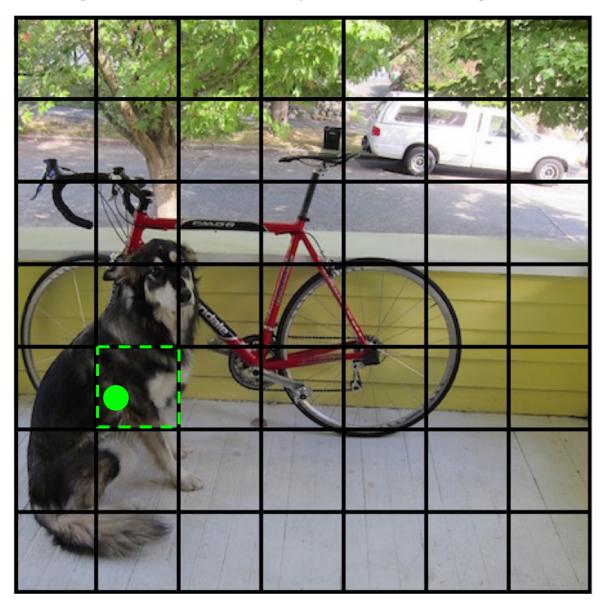
Thus we can train one neural network to be a whole detection pipeline



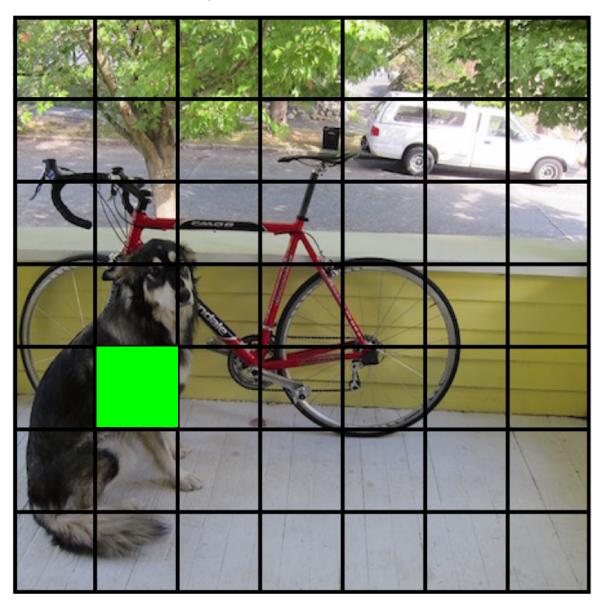
During training, match example to the right cell



During training, match example to the right cell



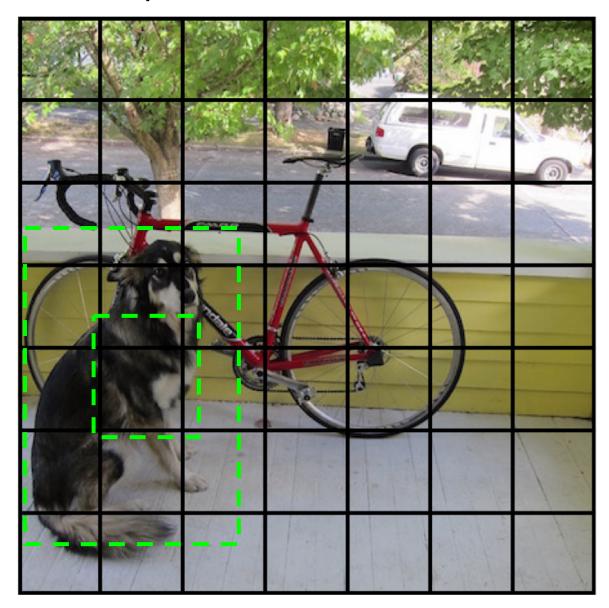
Adjust that cell's class prediction



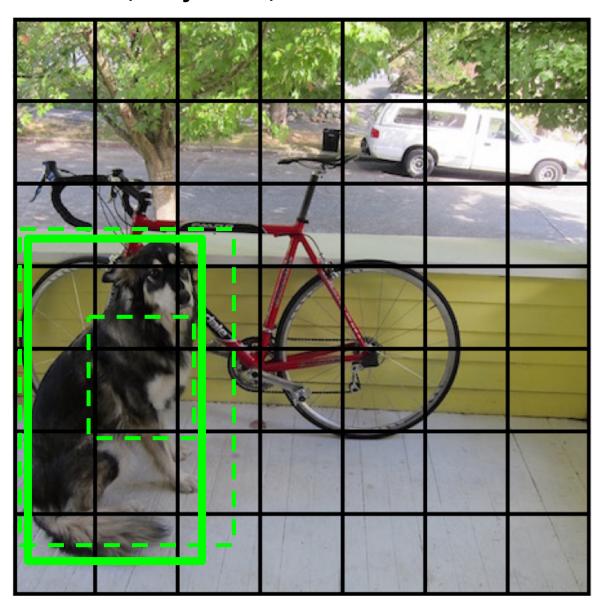
Dog = 1 Cat = 0 Bike = 0

• • •

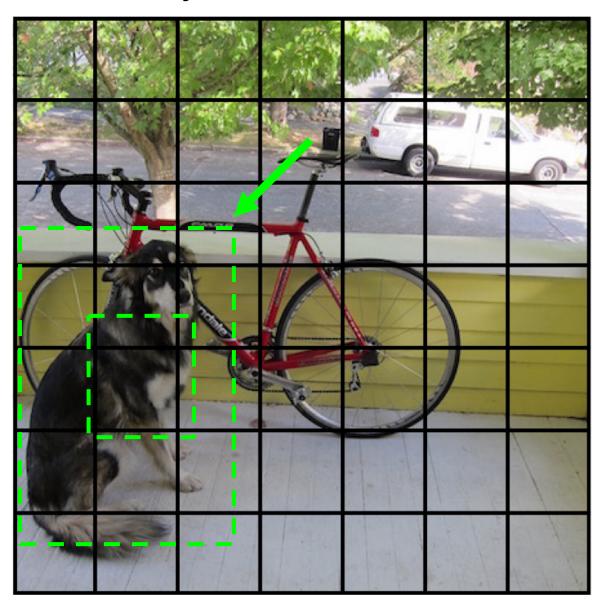
Look at that cell's predicted boxes



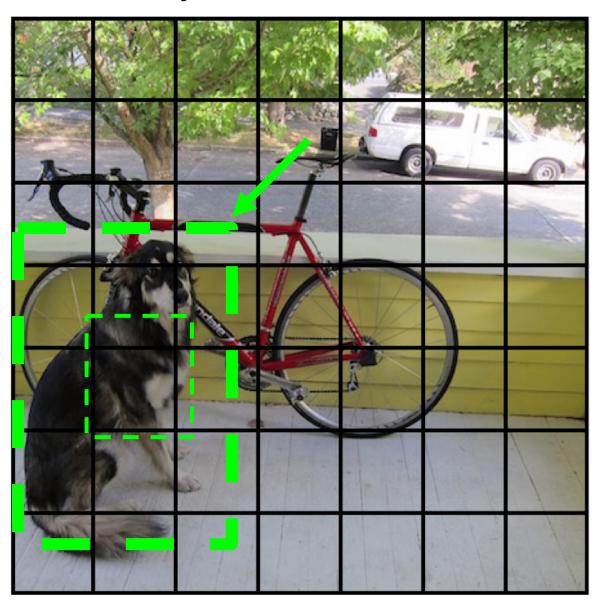
Find the best one, adjust it, increase the confidence



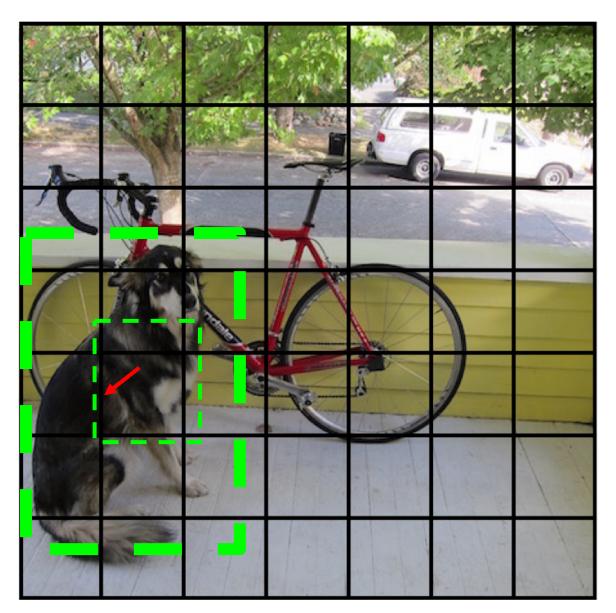
Find the best one, adjust it, increase the confidence



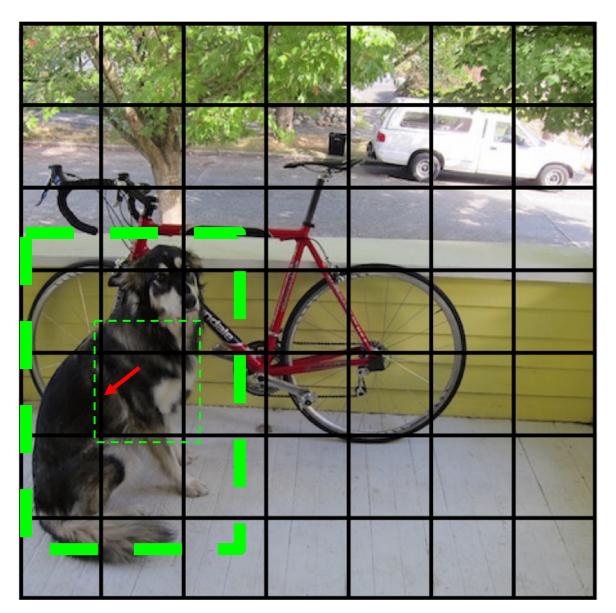
Find the best one, adjust it, increase the confidence



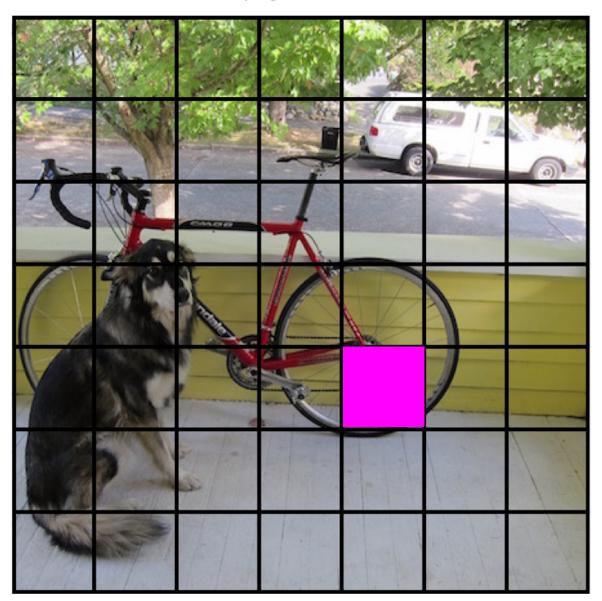
Decrease the confidence of other boxes



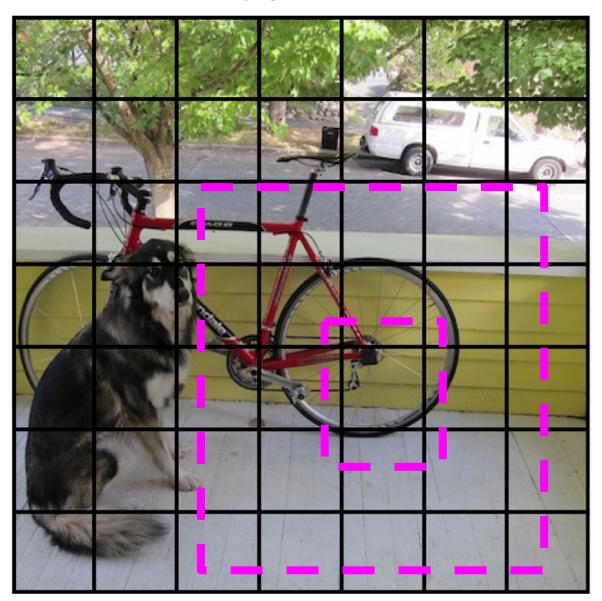
Decrease the confidence of other boxes



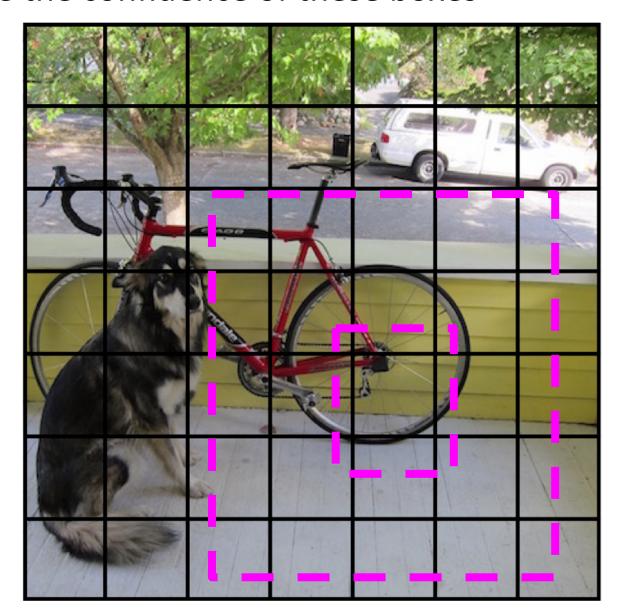
Some cells don't have any ground truth detections!



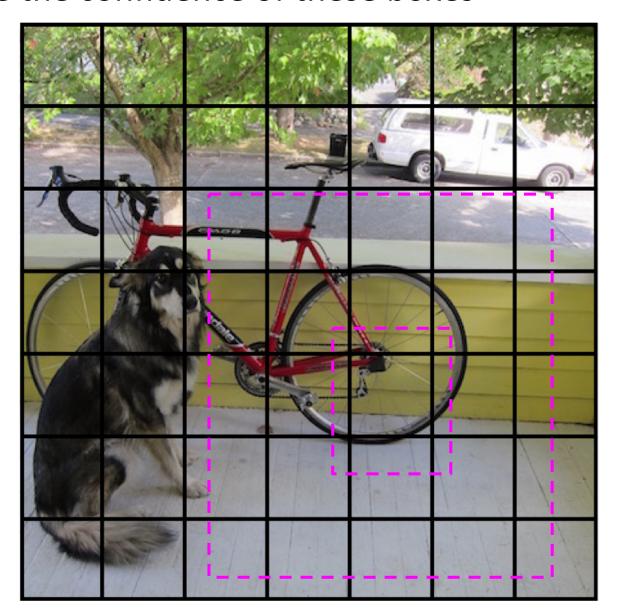
Some cells don't have any ground truth detections!



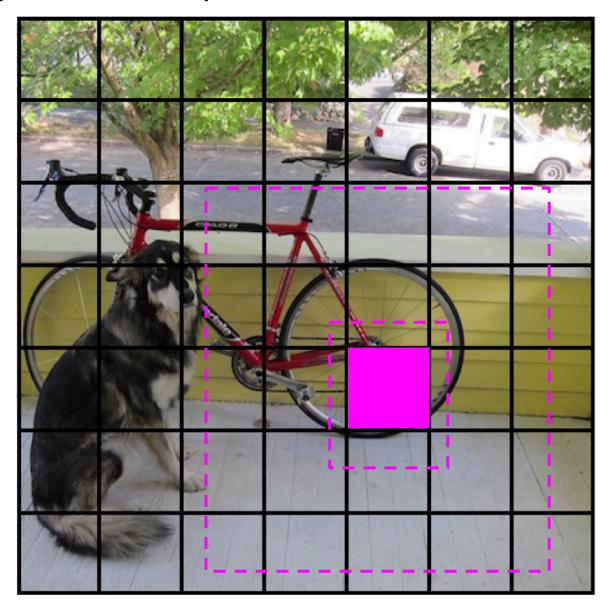
Decrease the confidence of these boxes



Decrease the confidence of these boxes

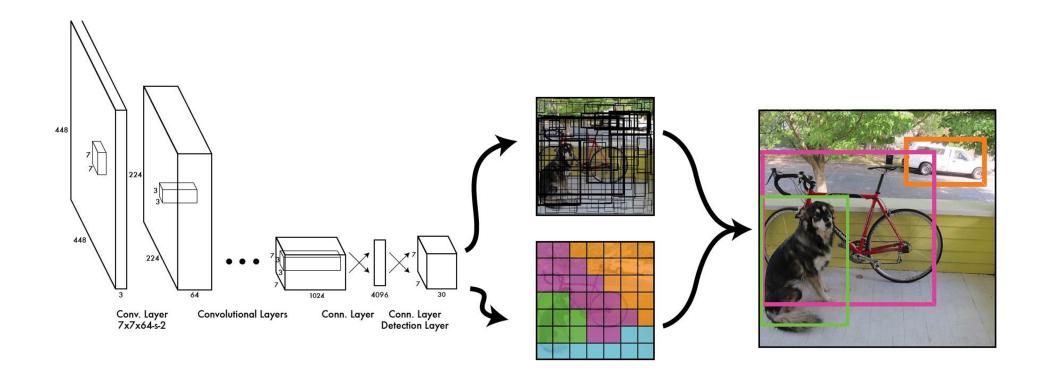


Don't adjust the class probabilities or coordinates

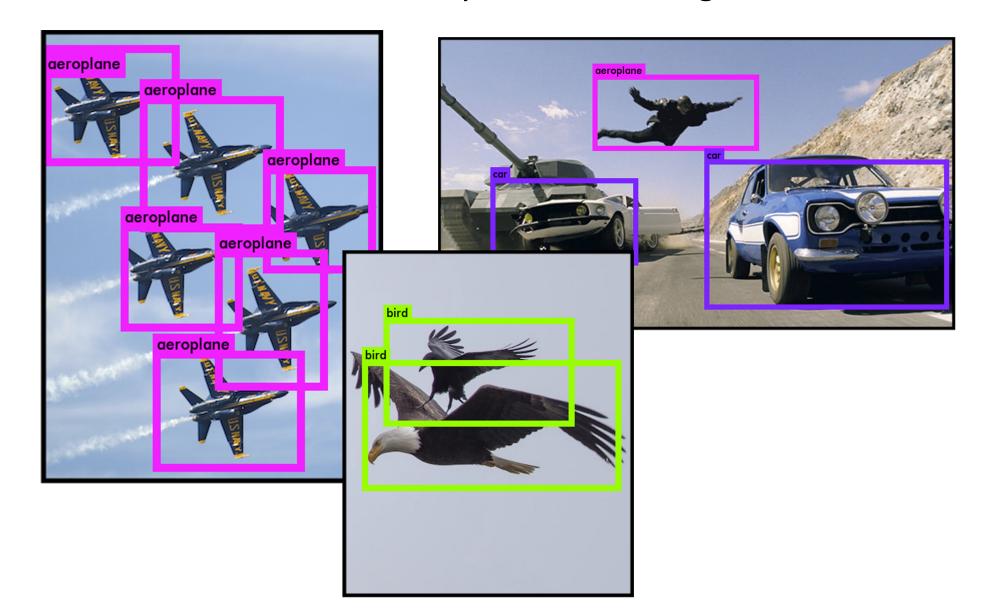


We train with standard tricks:

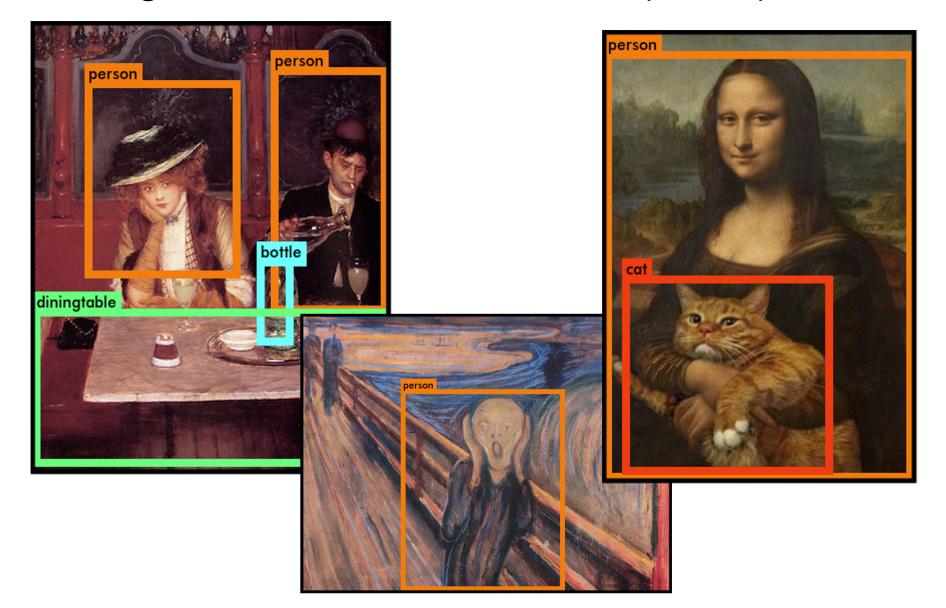
- Pretraining on Imagenet
- SGD with decreasing learning rate
- Extensive data augmentation
- For details, see the paper



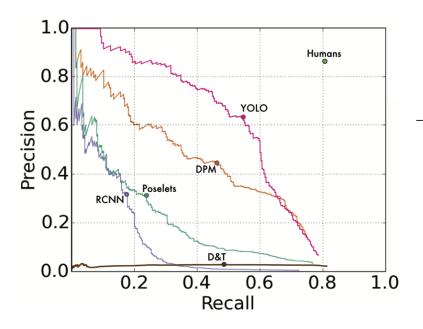
YOLO works across a variety of natural images



It also generalizes well to new domains (like art)



YOLO outperforms methods like DPM and R-CNN when generalizing to person detection in artwork



	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	59.2	53.3	0.590	45
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32

H. Cai, Q. Wu, T. Corradi, and P. Hall. The cross-depiction problem: Computer vision algorithms for recognising objects in artwork and in photographs.

S. Ginosar, D. Haas, T. Brown, and J. Malik. Detecting people in cubist art. In Computer Vision-ECCV 2014 Workshops, pages 101–116. Springer, 2014.

Code available! <u>pjreddie.com/yolo</u>







		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
N N	Faster R-CNN	73.2	7 FPS	140 ms/img
	YOLO	69.0	45 FPS	22 ms/img
224 3 3 3 224 448 Conv. Layer 7x7x64-s-2 Convolutional Layers Conn. Layer		nn. Layer Conn. Layer Detection Layer		

JOSEPH

ALI

REDMON

FARHADI

RETURN IN....

YOLO9000

Better, Faster,

Stronger

NOW PLAYING IN A DEMO NEAR YOU

SIY O WASHINGTON MULTIPLICATION IN ASSOCIATION WITH XNORAL AND THE ALLEN INSTITUTE FOR ARTIFICIAL INTELLIGENCE

MODELS BY DARWINET: OPEN SOURCE NEURAL NETWORKS

@DARKNETFOREVER #YOLO9000

pjreddie.com/yolo

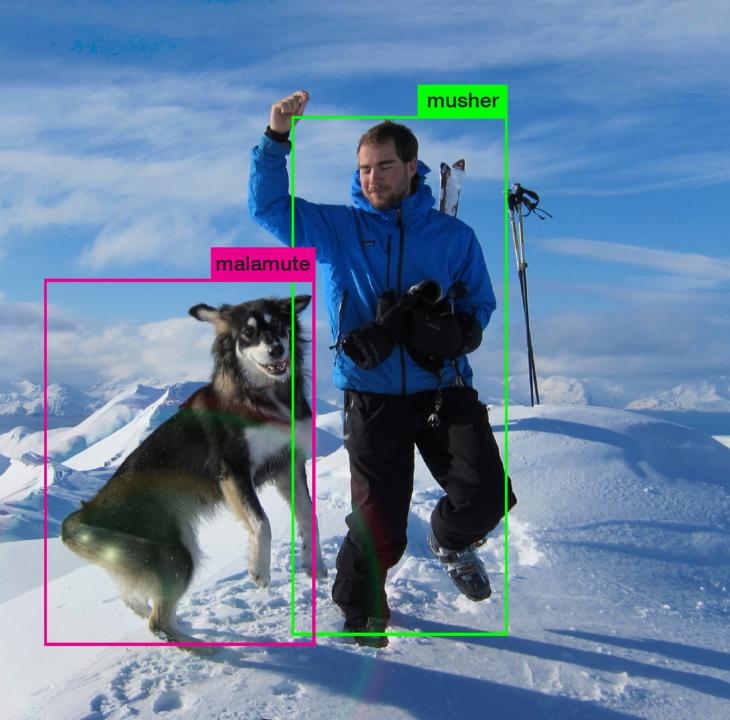




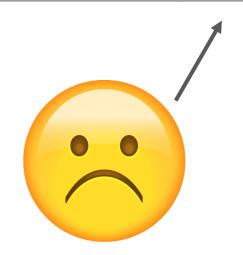


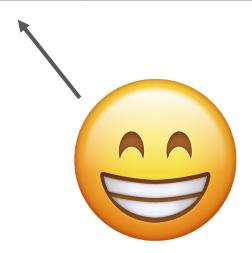






	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img



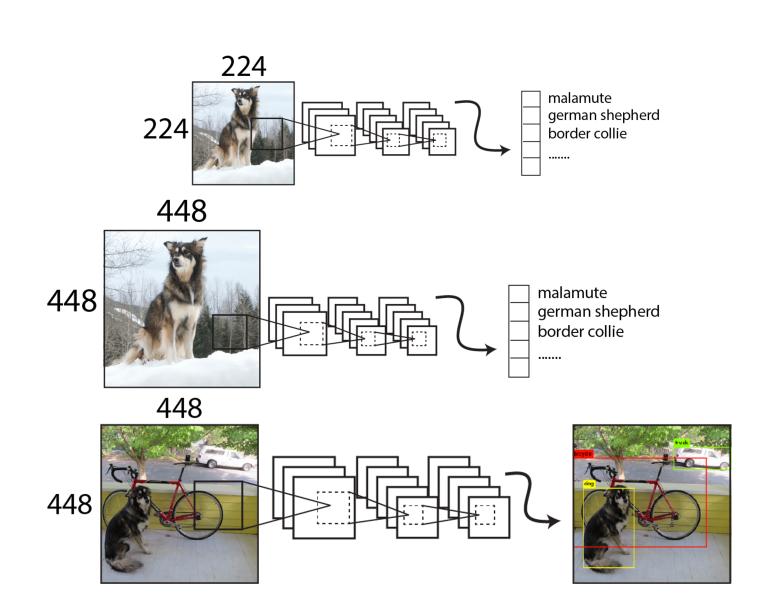


Fine-tune 448x448 Classifier: +3.5% mAP

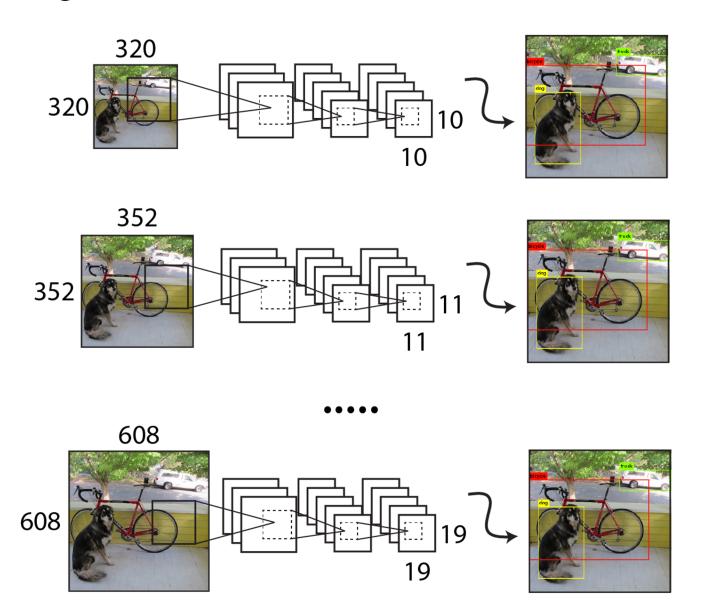
Train on ImageNet

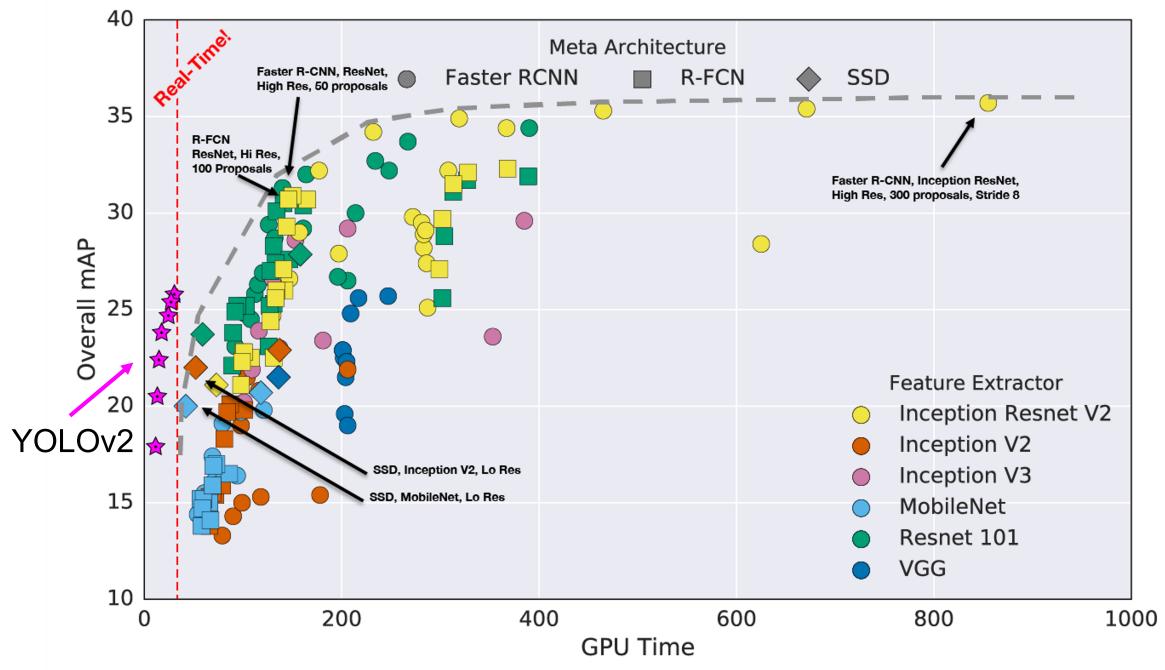
Resize, fine-tune on ImageNet

Fine-tune on detection



Multi-scale training: +1.5% mAP





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

Two-stage Detectors





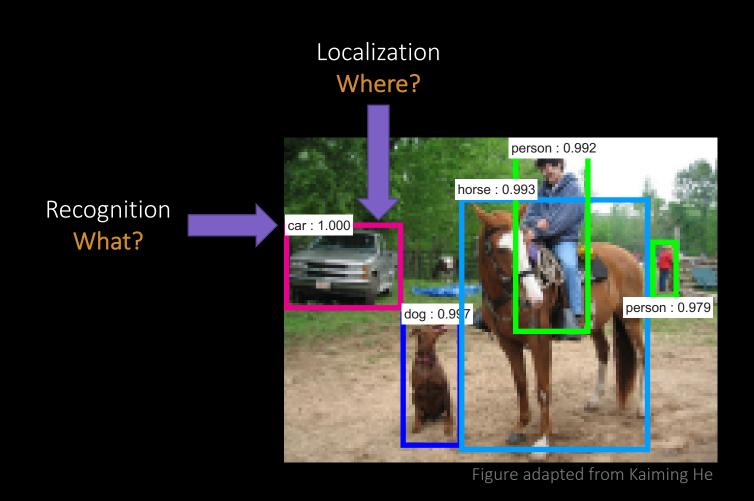
Fast R-CNN

Ross Girshick

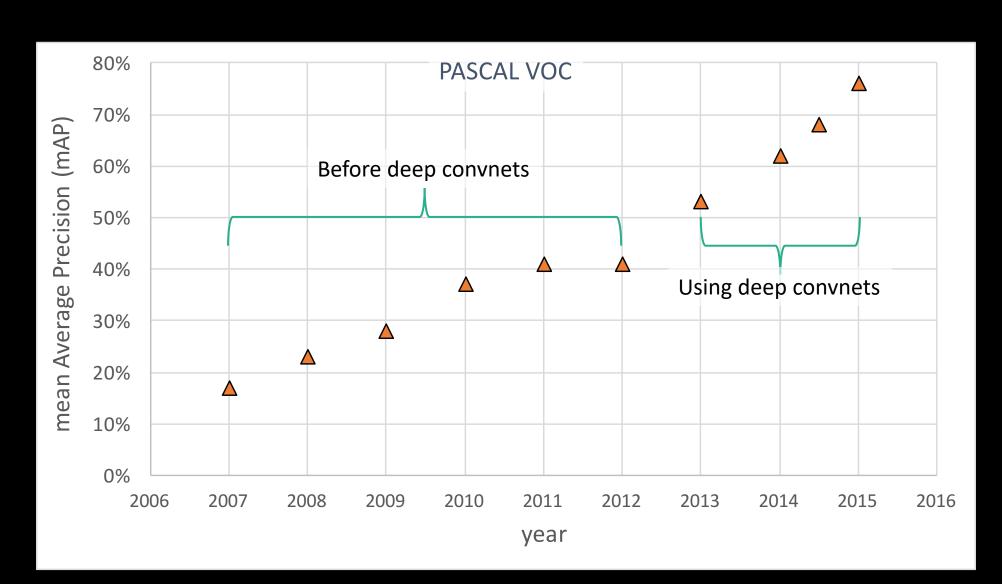
Facebook AI Research (FAIR)

Work done at Microsoft Research

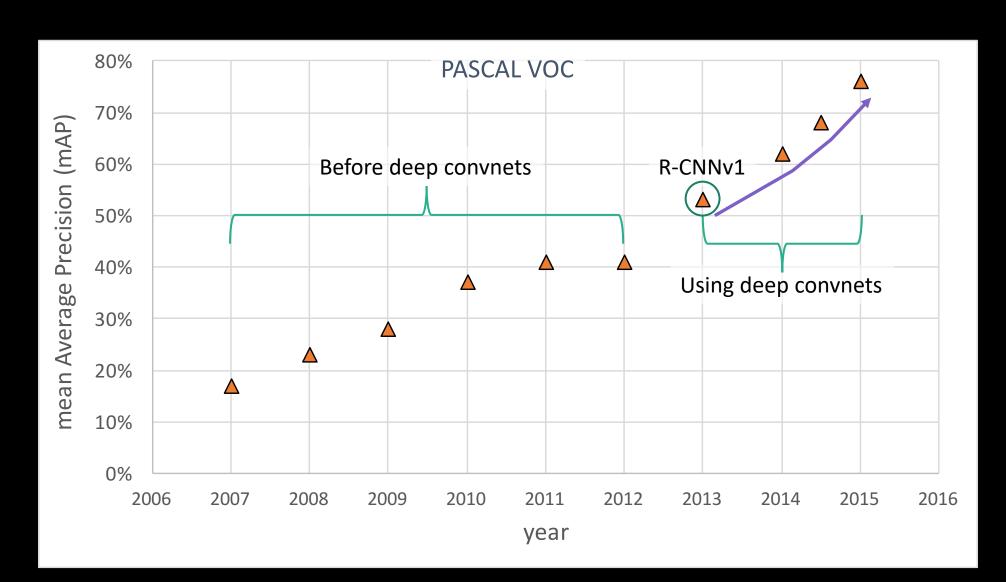
Fast Region-based ConvNets (R-CNNs) for Object Detection



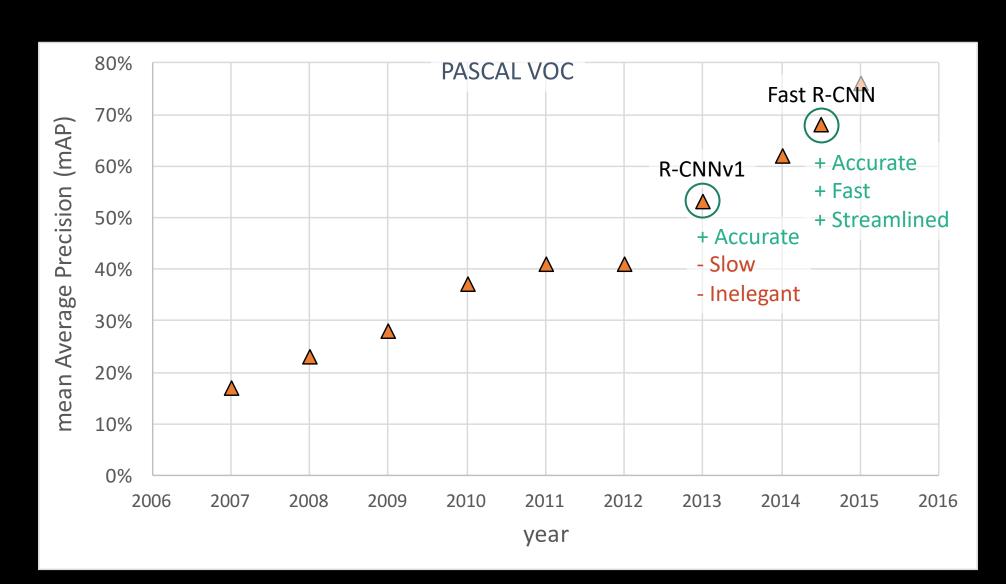
Object detection renaissance (2013-present)



Object detection renaissance (2013-present)



Object detection renaissance (2013-present)



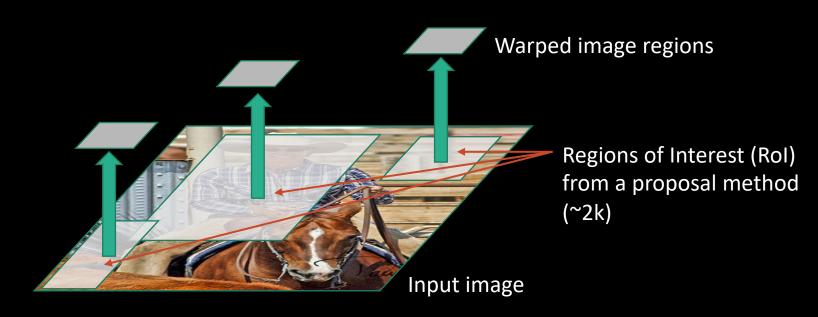
Region-based convnets (R-CNNs)

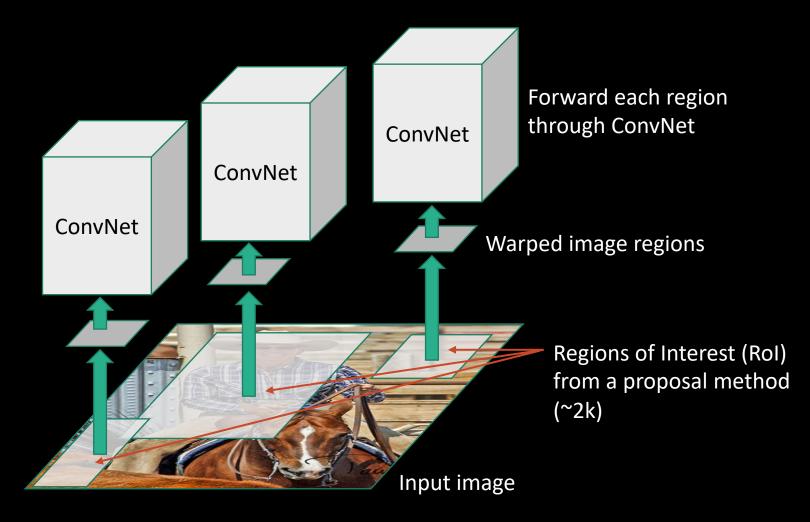
- R-CNN (aka "slow R-CNN") [Girshick et al. CVPR14]
- SPP-net [He et al. ECCV14]

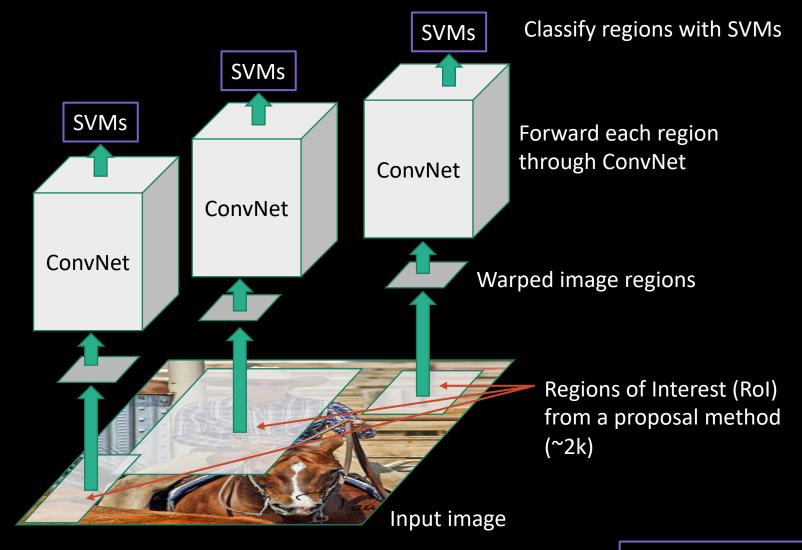




Regions of Interest (RoI) from a proposal method (~2k)

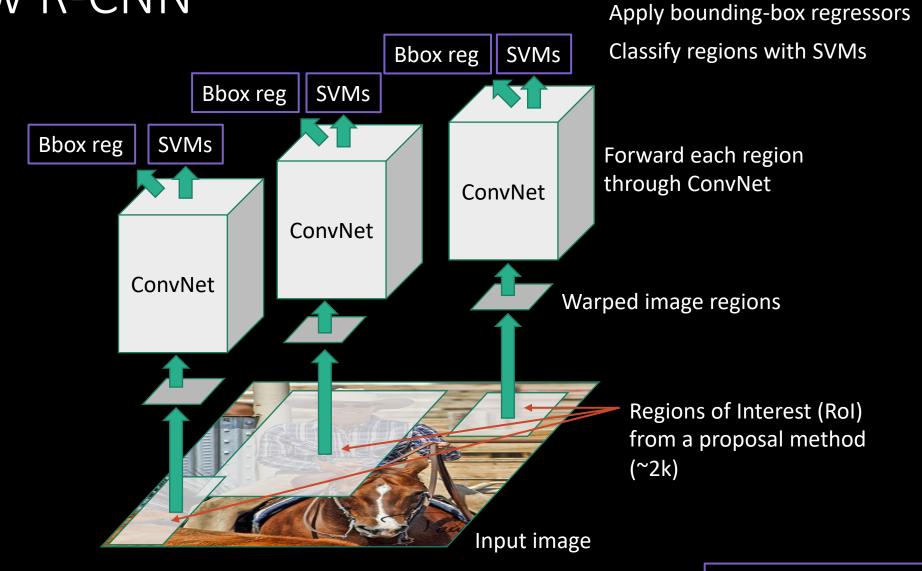






Girshick et al. CVPR14.

Post hoc component



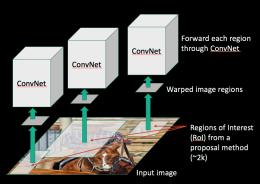
Post hoc component

Ad hoc training objectives

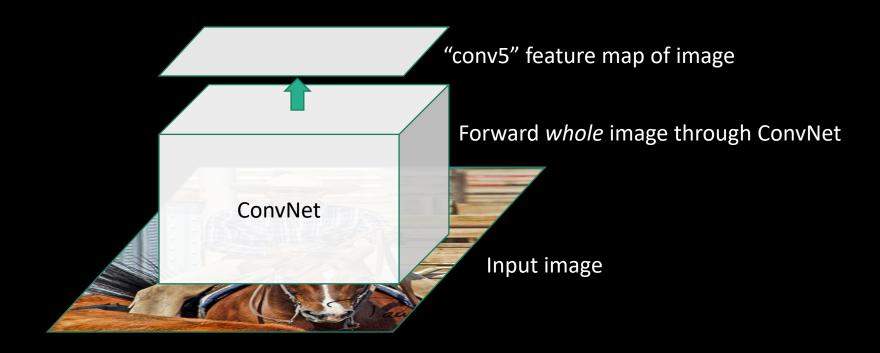
- Fine-tune network with softmax classifier (log loss)
- Train post-hoc linear SVMs (hinge loss)
- Train post-hoc bounding-box regressors (squared loss)

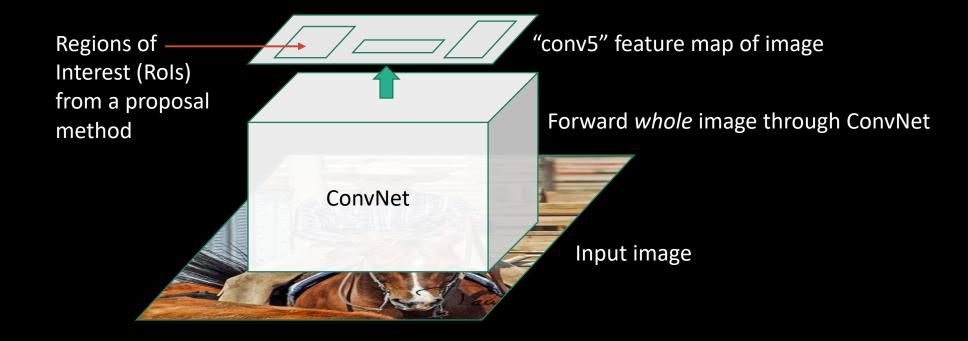
- Ad hoc training objectives
 - Fine-tune network with softmax classifier (log loss)
 - Train post-hoc linear SVMs (hinge loss)
 - Train post-hoc bounding-box regressors (squared loss)
- Training is slow (84h), takes a lot of disk space

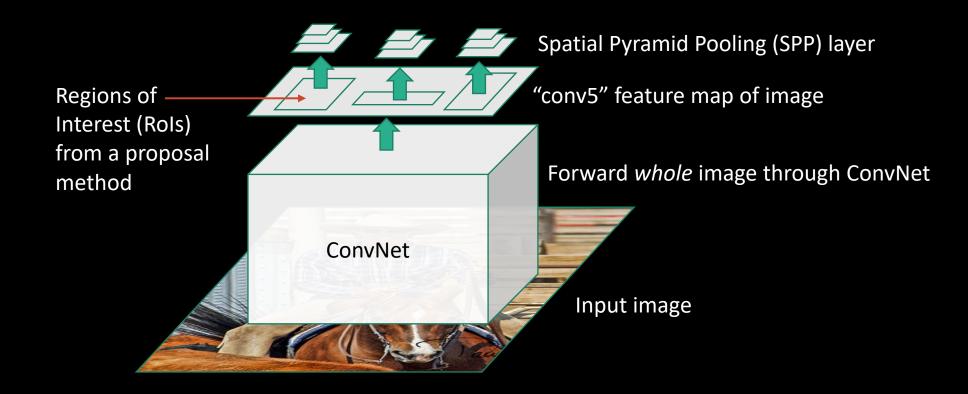
- Ad hoc training objectives
 - Fine-tune network with softmax classifier (log loss)
 - Train post-hoc linear SVMs (hinge loss)
 - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
 - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
 - Fixed by SPP-net [He et al. ECCV14]







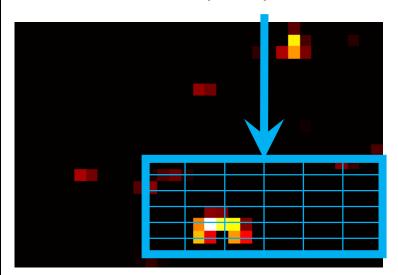


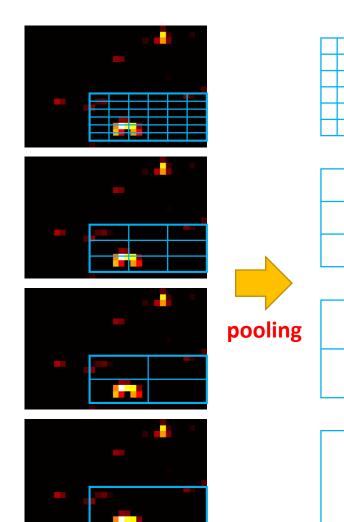


Spatial Pyramid Pooling (SPP) Layer

 fix the number of bins (instead of filter sizes)

adaptively-sized bins



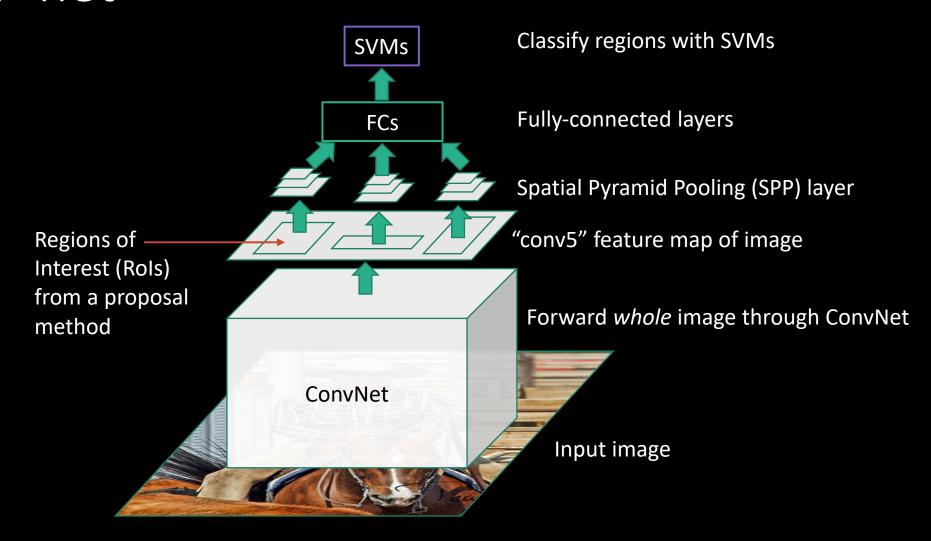


a finer level maintains explicit spatial information

concatenate, fc layers...

a coarser level removes explicit spatial information (bag-of-features)



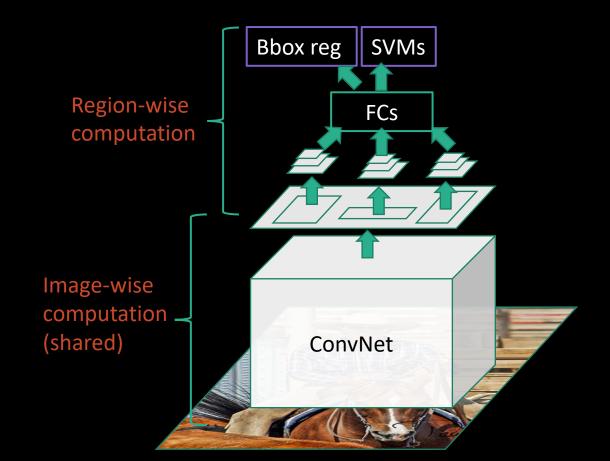


Classify regions with SVMs SVMs Bbox reg Fully-connected layers **FCs** Spatial Pyramid Pooling (SPP) layer "conv5" feature map of image Regions of Interest (Rols) from a proposal Forward *whole* image through ConvNet method ConvNet Input image

Apply bounding-box regressors

What's good about SPP-net?

• Fixes one issue with R-CNN: makes testing fast



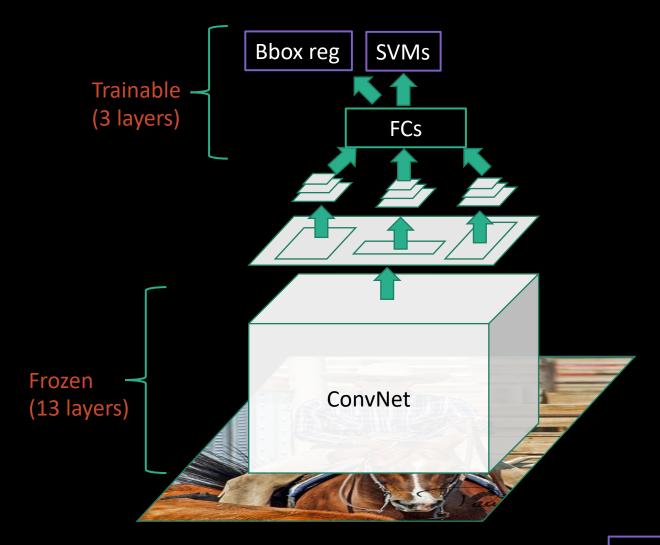
What's wrong with SPP-net?

- Inherits the rest of R-CNN's problems
 - Ad hoc training objectives
 - Training is slow (25h), takes a lot of disk space

What's wrong with SPP-net?

- Inherits the rest of R-CNN's problems
 - Ad hoc training objectives
 - Training is slow (though faster), takes a lot of disk space
- Introduces a new problem: cannot update parameters below SPP layer during training

SPP-net: the main limitation



Fast R-CNN

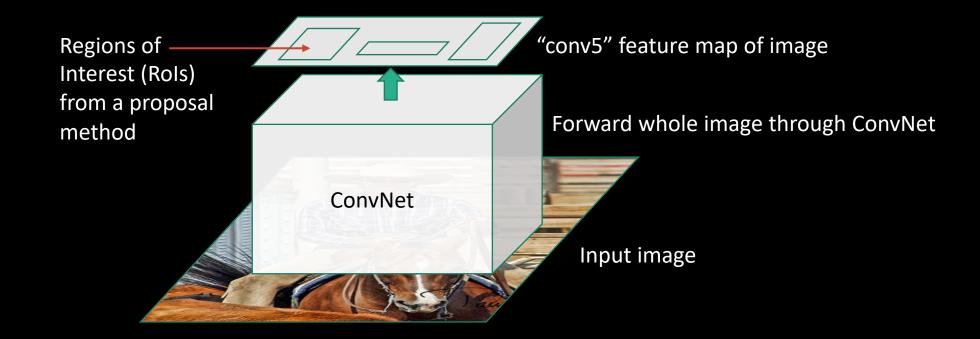
• Fast test-time, like SPP-net

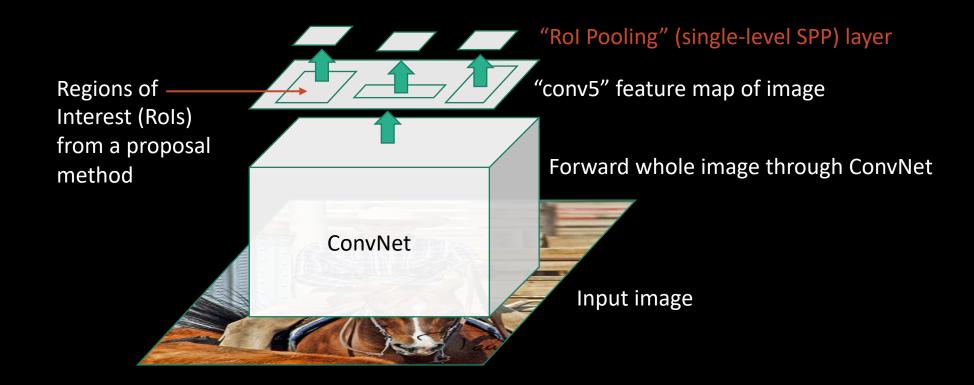
Fast R-CNN

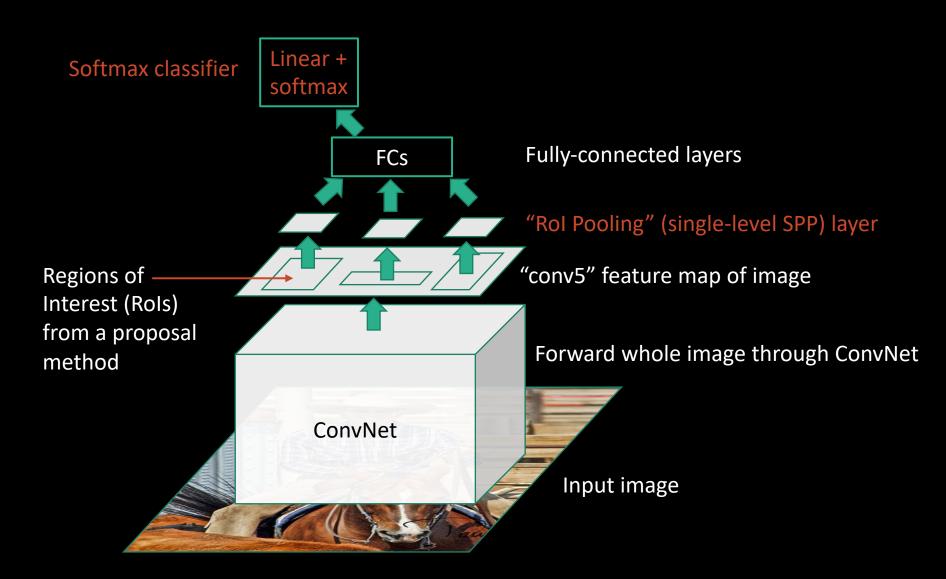
- Fast test-time, like SPP-net
- One network, trained in one stage

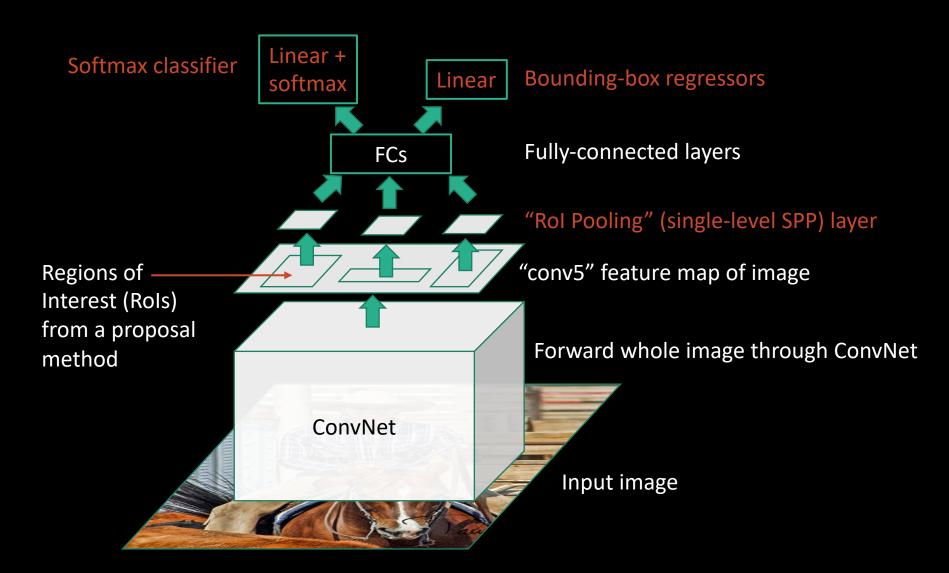
Fast R-CNN

- Fast test-time, like SPP-net
- One network, trained in one stage
- Higher mean average precision than slow R-CNN and SPP-net

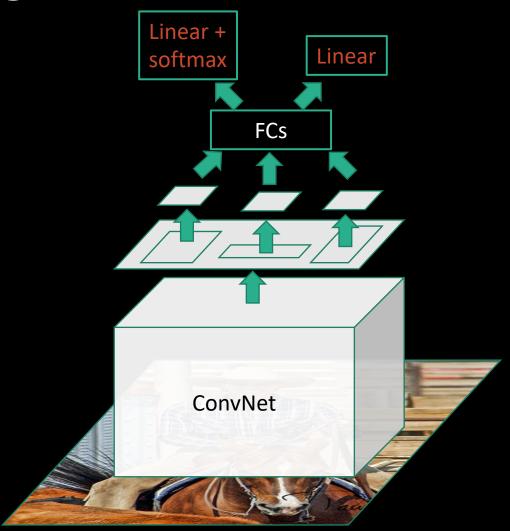




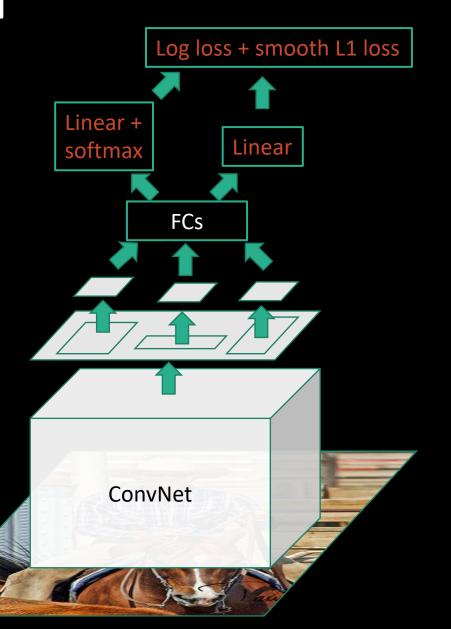




Fast R-CNN (training)

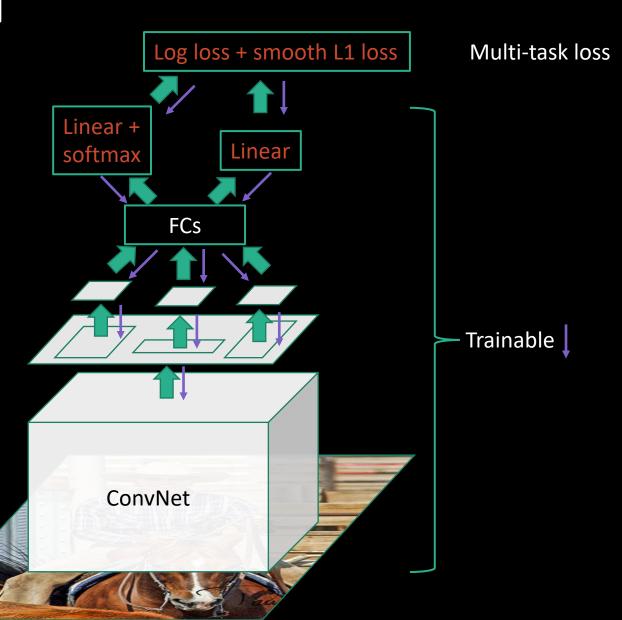


Fast R-CNN (training)



Multi-task loss

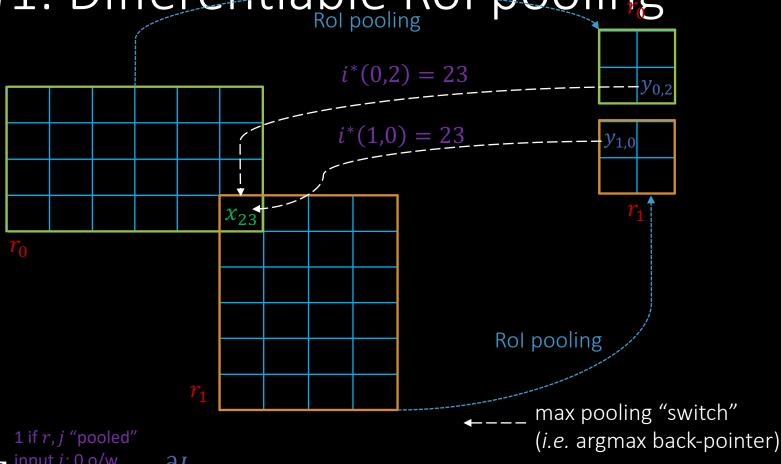
Fast R-CNN (training)



Obstacle #1: Differentiable Rol pooling

Region of Interest (RoI) pooling must be (sub-) differentiable to train conv layers

Obstacle #1: Differentiable Rol pooling



$$\frac{\partial L}{\partial x_i} = \sum_{\substack{r \\ j}} \sum_{\substack{i \text{input } i; \text{ 0 o/w} \\ [i = i^*(r, j)]}} \frac{\partial L}{\partial y_{rj}}$$
Partial Over regions r , Partial from for x_i locations j next layer

Slow R-CNN and SPP-net use region-wise sampling to make mini-batches

- Sample 128 example Rols uniformly at random
- Examples will come from different images with high probability



Note the receptive field for one example Rol is often very large

• Worst case: the receptive field is the entire image





Worst case cost per mini-batch (crude model of computational complexity)

input size for Fast R-CNN

input size for slow R-CNN

128*600*1000 / (128*224 *224) = 12x more computation than slow R-CNN

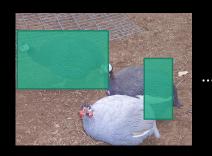




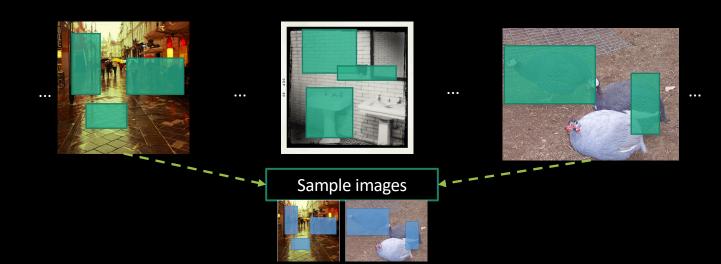
Solution: use hierarchical sampling to build minibatches





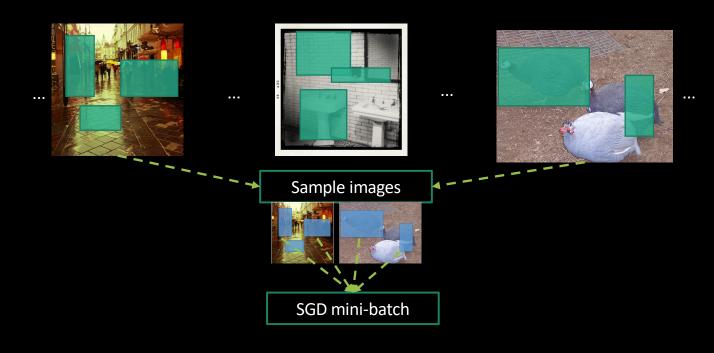


Solution: use hierarchical sampling to build minibatches



 Sample a small number of images
 (2)

Solution: use hierarchical sampling to build minibatches

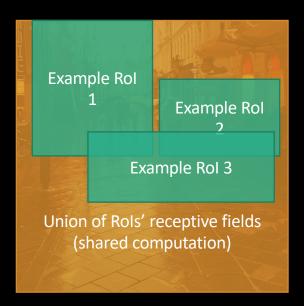


- Sample a small number of images
 (2)
- Sample many examples from each image (64)

Use the test-time trick from SPP-net during training

 Share computation between overlapping examples from the same image

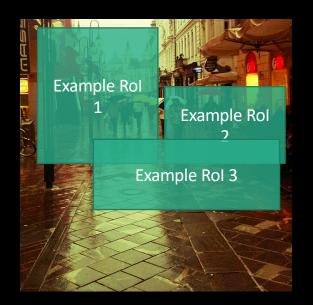


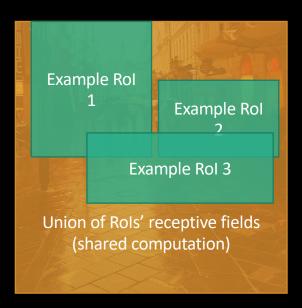


Cost per mini-batch compared to slow R-CNN (same crude cost model)

input size for Fast R-CNN input size for slow R-CNN

• 2*600*1000 / (128*224*224) = 0.19x less computation than slow R-CNN





Main results

	Fast R-CNN	R-CNN [1]	SPP-net [2]
Train time (h)	9.5	84	25
- Speedup	8.8x	1x	3.4x
Test time / image	0.32s	47.0s	2.3s
Test speedup	146x	1x	20x
mAP	66.9%	66.0%	63.1%

Timings exclude object proposal time, which is equal for all methods. All methods use VGG16 from Simonyan and Zisserman.

- [1] Girshick et al. CVPR14.
- [2] He et al. ECCV14.

Main results

	Fast R-CNN	R-CNN [1]	SPP-net [2]
Train time (h)	9.5	84	25
- Speedup	8.8x	1x	3.4x
Test time / image	0.32s	47.0s	2.3s
Test speedup	146x	1x	20x
mAP	66.9%	66.0%	63.1%

Timings exclude object proposal time, which is equal for all methods. All methods use VGG16 from Simonyan and Zisserman.

- [1] Girshick et al. CVPR14.
- [2] He et al. ECCV14.

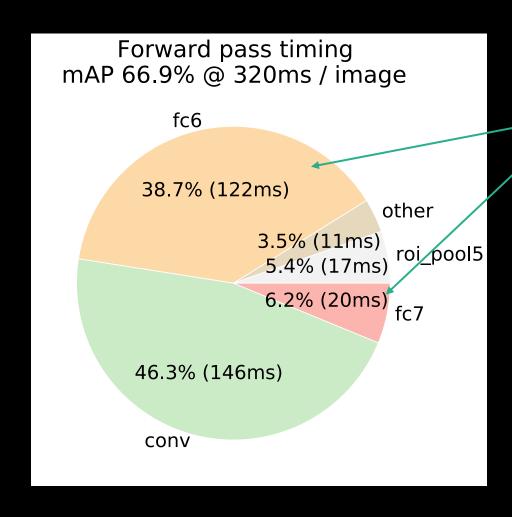
Main results

	Fast R-CNN	R-CNN [1]	SPP-net [2]
Train time (h)	9.5	84	25
- Speedup	8.8x	1x	3.4x
Test time / image	0.32s	47.0s	2.3s
Test speedup	146x	1x	20x
mAP	66.9%	66.0%	63.1%

Timings exclude object proposal time, which is equal for all methods. All methods use VGG16 from Simonyan and Zisserman.

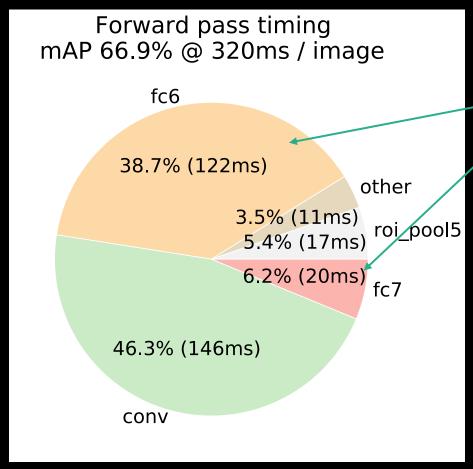
- [1] Girshick et al. CVPR14.
- [2] He et al. ECCV14.

Further test-time speedups



Fully connected layers take 45% of the forward pass time

Further test-time speedups

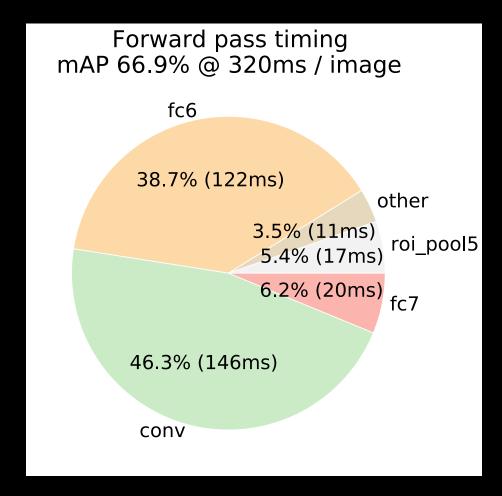


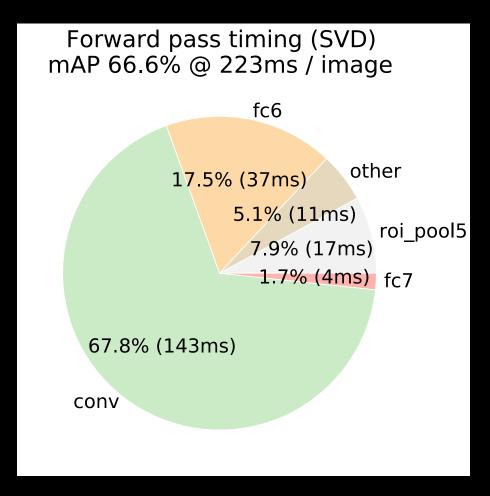
Compress these layers with truncated SVD

J. Xue, J. Li, and Y. Gong.
Restructuring of deep neural network acoustic models with singular value decomposition.

Interspeech, 2013.

Further test-time speedups





Without SVD With SVD

Other findings

End-to-end training matters

	Fast R-CNN (VGG16)			
Fine-tune layers	≥ fc6	≥ conv3_1	≥ conv2_1	
VOC07 mAP	61.4%	66.9%	67.2%	
Test time per image	0.32s	0.32s	0.32s	

1.4x slower training

	Fast R-CNN (VGG16)			
Multi-task training?		Υ		Υ
Stage-wise training?			Υ	
Test-time bbox reg.			Υ	Υ
VOC07 mAP	62.6%	63.4%	64.0%	66.9%

	Fast R-CNN (VGG16)			
Multi-task training?		Υ		Υ
Stage-wise training?			Υ	
Test-time bbox reg.			Υ	Υ
VOC07 mAP	62.6%	63.4%	64.0%	66.9%

Trained without a bbox regressor

	Fast R-CNN (VGG16)			
Multi-task training?		Υ		Υ
Stage-wise training?			Υ	
Test-time bbox reg.			Υ	Υ
VOC07 mAP	62.6%	63.4%	64.0%	66.9%

Trained with a bbox regressor, but it's disabled at test time

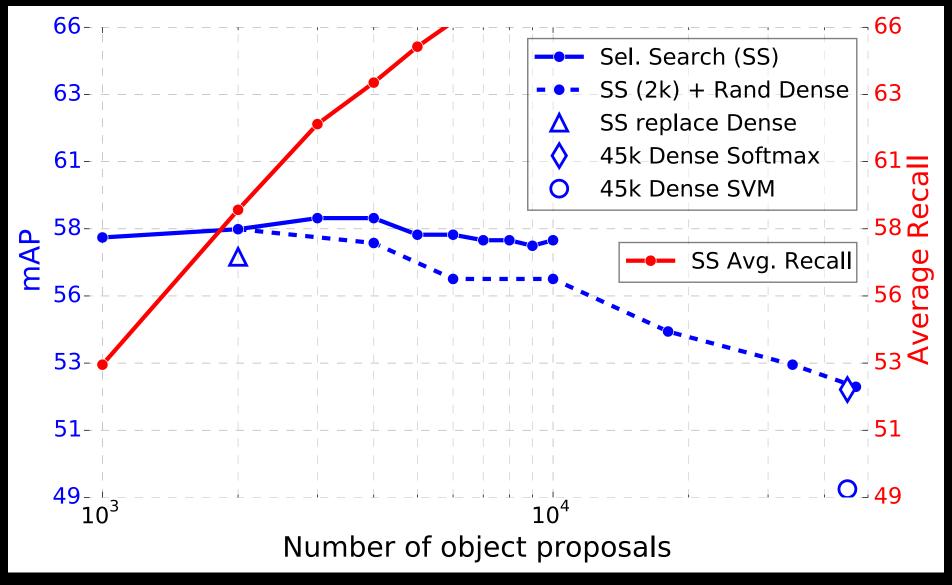
	Fast R-CNN (VGG16)			
Multi-task training?		Υ		Υ
Stage-wise training?			Υ	
Test-time bbox reg.			Υ	Υ
VOC07 mAP	62.6%	63.4%	64.0%	66.9%

Post hoc bbox regressor, used at test time

	Fast R-CNN (VGG16)			
Multi-task training?		Υ		Υ
Stage-wise training?			Υ	
Test-time bbox reg.			Υ	Υ
VOC07 mAP	62.6%	63.4%	64.0%	66.9%

Multi-task objective, using bbox regressors at test time

More proposals is harmful



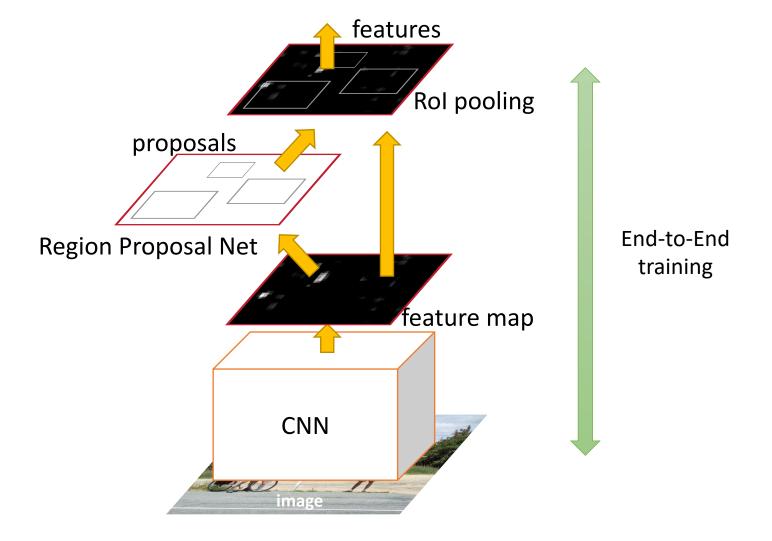
What's still wrong?

- Out-of-network region proposals
 - Selective search: 2s / im; EdgeBoxes: 0.2s / im
- Fortunately, we have a solution
 - Our follow-up work was presented last week at NIPS

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

Object Detection: Faster R-CNN

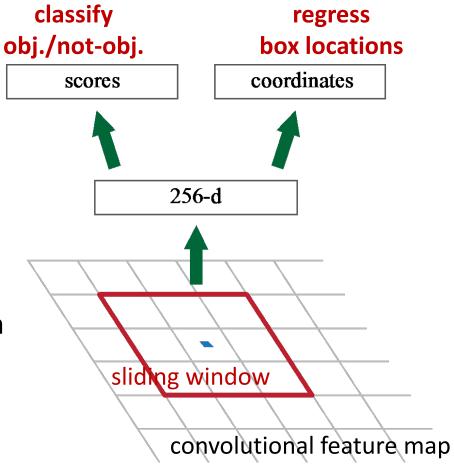
- Faster R-CNN
 - Solely based on CNN
 - No external modules
 - Each step is end-to-end





Region Proposal Network

- Slide a small window on the feature map
- Build a small network for:
 - classifying object or not-object, and
 - regressing bbox locations
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window

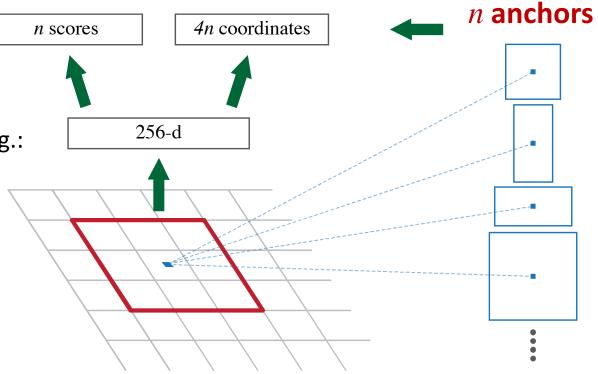






Anchors as references

- Anchors: pre-defined reference boxes
 - Box regression is with reference to anchors: regressing an anchor box to a ground-truth box
 - Object probability is with reference to anchors, e.g.:
 - anchors as positive samples: if IoU > 0.7 or IoU is max
 - anchors as negative samples: if IoU < 0.3

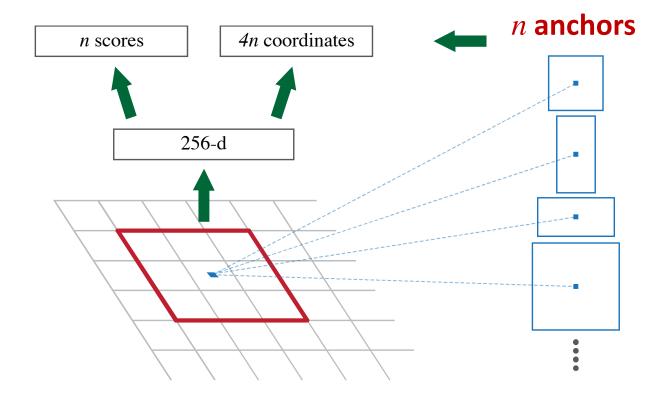






Anchors as references

- Anchors: pre-defined reference boxes
- Multi-scale/size anchors:
 - multiple anchors are used at each position: e.g., 3 scales (128², 256², 512²) and 3 aspect ratios (2:1, 1:1, 1:2) yield 9 anchors
 - each anchor has its own prediction function
 - single-scale features, multi-scale predictions

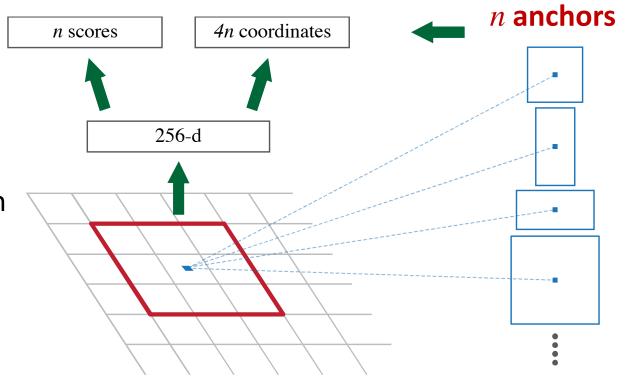






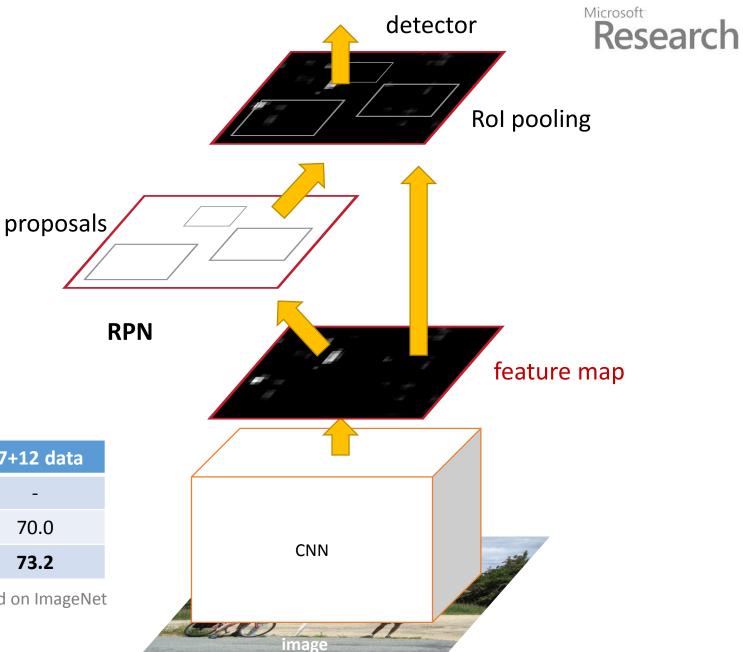
Region Proposal Network

- RPN is fully convolutional [Long et al. 2015]
- RPN is trained end-to-end
- RPN shares convolutional feature maps with the detection network (covered in Ross's section)





Faster R-CNN



system	time	07 data	07+12 data
R-CNN	~50s	66.0	-
Fast R-CNN	~2s	66.9	70.0
Faster R-CNN	198ms	69.9	73.2

detection mAP on PASCAL VOC 2007, with VGG-16 pre-trained on ImageNet



Focal Loss for Dense Object Detection

Tsung-Yi Lin, Google Brain

Work done at Facebook AI Research with Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár

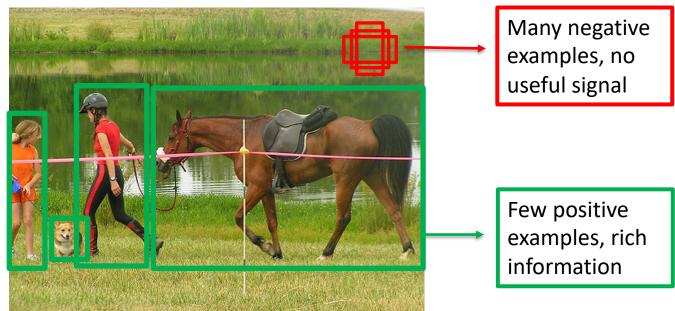
Toward dense detection

- YOLOv1 98 boxes
- YOLOv2 ~1k
- OverFeat ^1-2k
- SSD ~8-26k

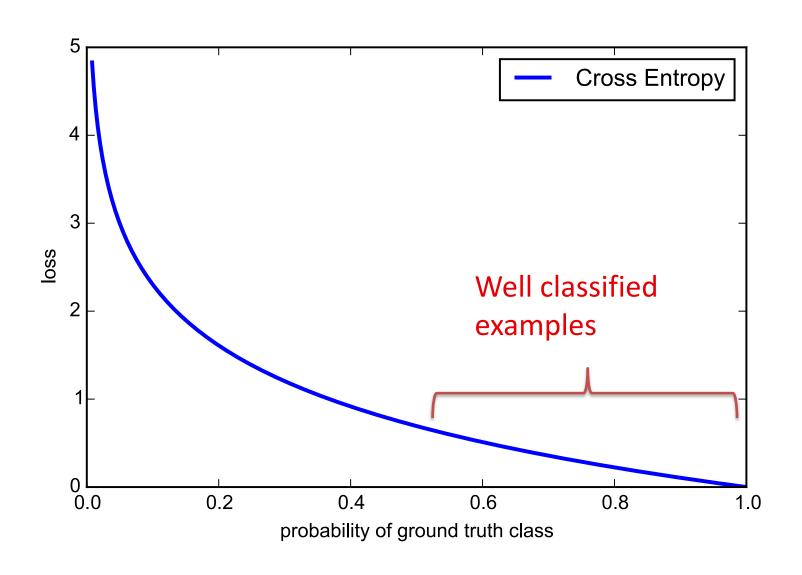
• This work – ~100k

Class Imbalance

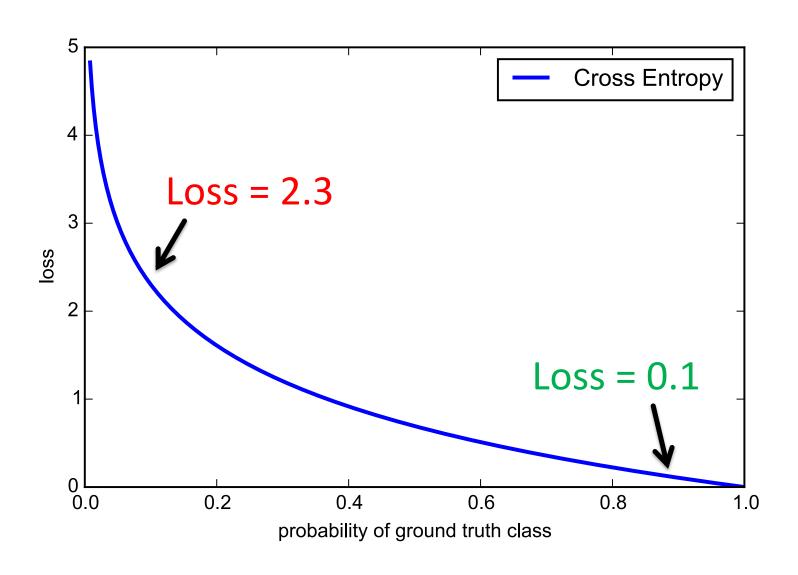
- Few training examples from foreground
- Most examples from background
 - Easy and uninformative
 - Distracting



Cross Entropy

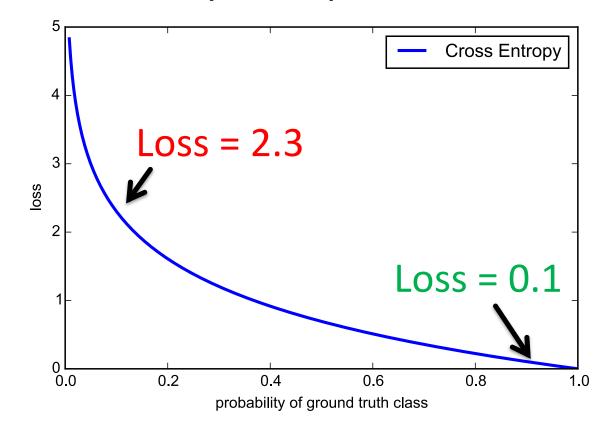


Cross Entropy



Cross Entropy with Imbalance Data

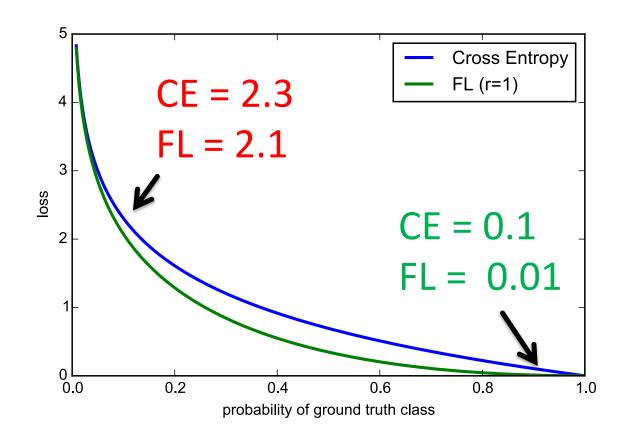
- 100000 easy: 100 hard examples
- 40x bigger loss from easy examples



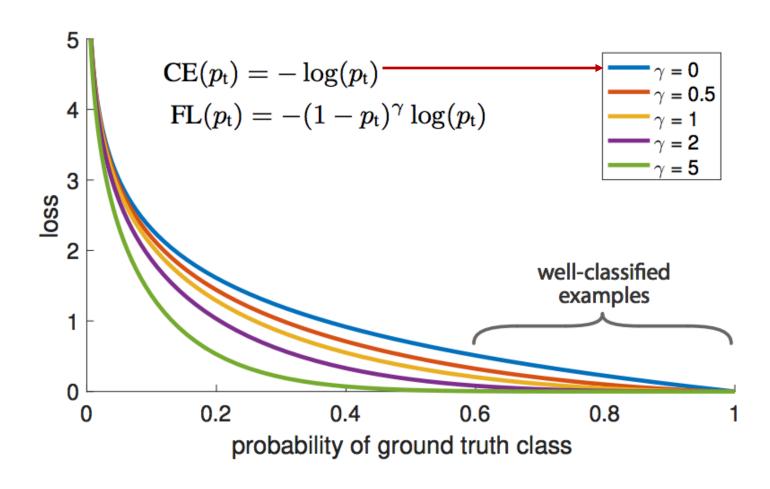
Focal Loss

$$CE(p_t) = -\log(p_t)$$

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$



Focal Loss



Prior

α-balanced Cross entropy

$$CE(p_t) = -\alpha_t \log(p_t)$$

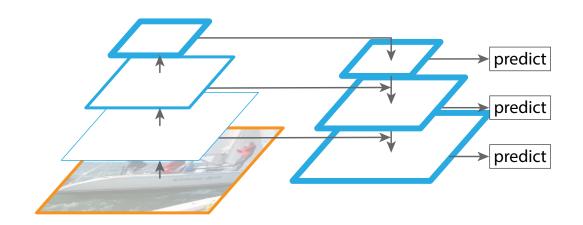
α-balanced Focal Loss

$$FL(p_{t}) = -\alpha_{t}(1 - p_{t})^{\gamma} \log(p_{t})$$

- γ: focus more on hard examples
- α : offset class imbalance of number of examples

Feature Pyramid Network

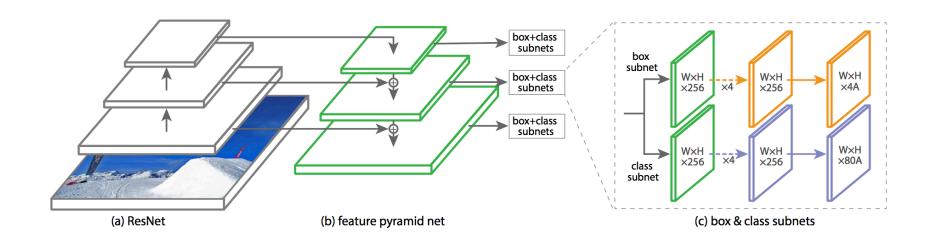
- Multiscale
- Semantically strong at all scales
- Fast to compute



Feature Pyramid Network for Object Detection, Lin et al., CVPR 2017

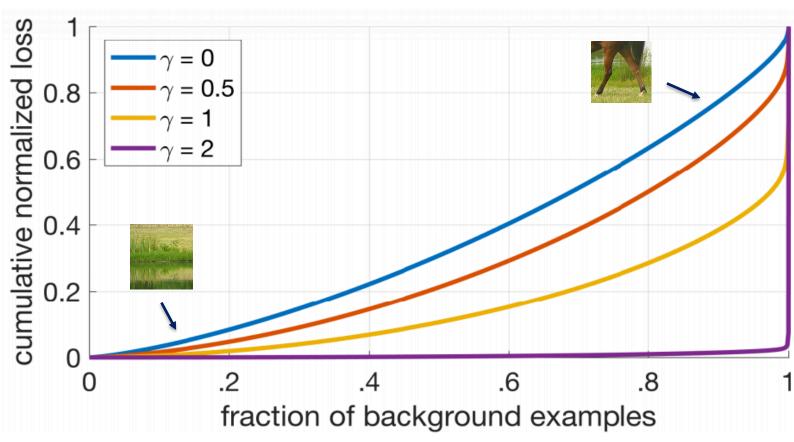
Architecture

- RetinaNet
 - -FPN + 100k boxes
 - Focal loss



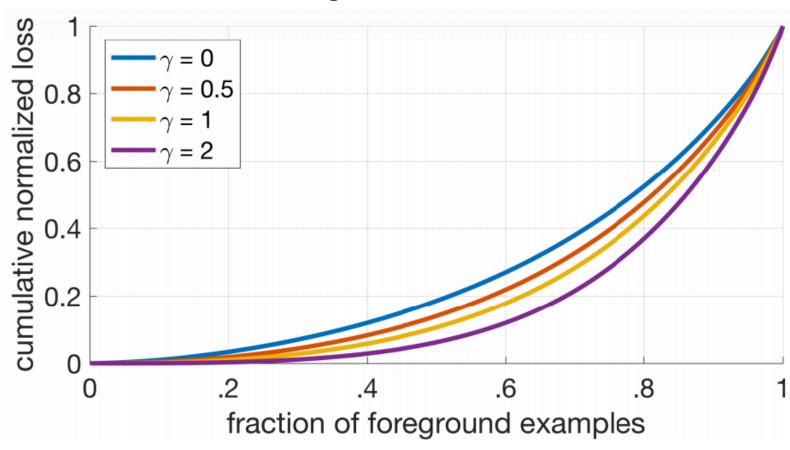
Loss Distribution under Focal Loss





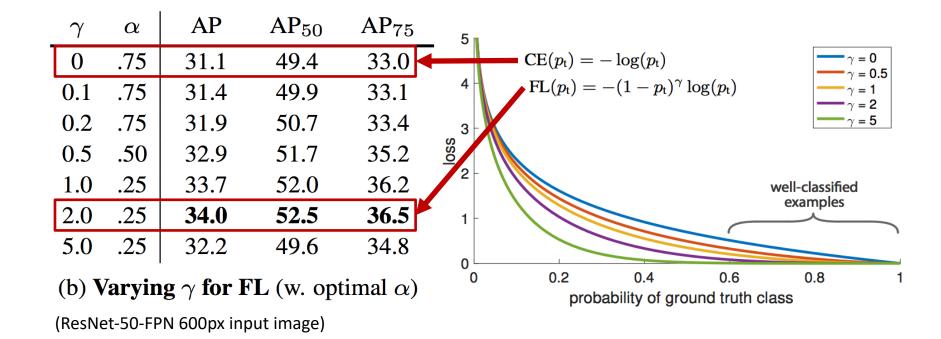
Loss Distribution under Focal Loss





vs. Cross Entropy

• + 2.9 AP to α-balanced cross entropy



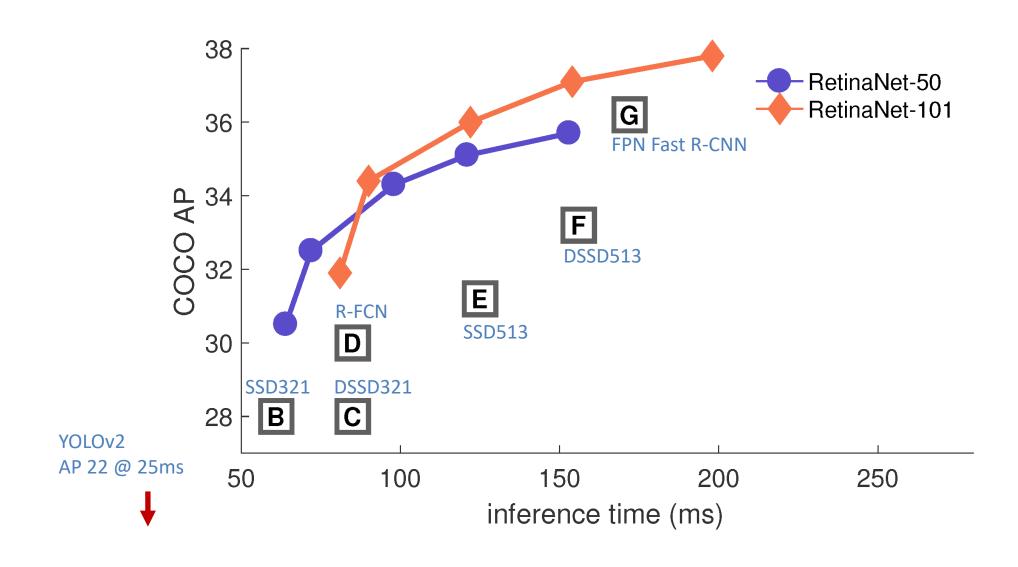
vs. OHEM

• +3.2 AP to best OHEM (ResNet-101 FPN)

method	batch size	nms thr	AP	
OHEM	128	.7	31.1	
OHEM	256	.7	31.8	
OHEM	512	.7	30.6	
OHEM	128	.5	32.8] → Best OHEM
OHEM	256	.5	31.0	
OHEM	512	.5	27.6	
OHEM 1:3	128	.5	31.1	-
OHEM 1:3	256	.5	28.3	
OHEM 1:3	512	.5	24.0	
FL	n/a	n/a	36.0	→ Best Focal Loss

Online Hard Example Mining, Shrivastava et al., 2016

RetinaNet performance



Summary

- Identify class imbalance is the major issue for training onestage dense detector
- Propose Focal Loss to address class imbalance
- Achieve state-of-the-art accuracy and speed



ICCV 2017 Tutorial, Venice, Italy

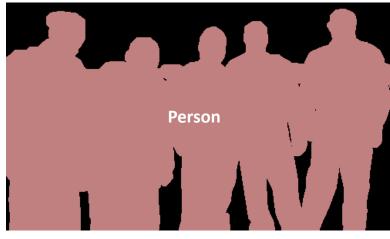
Kaiming He

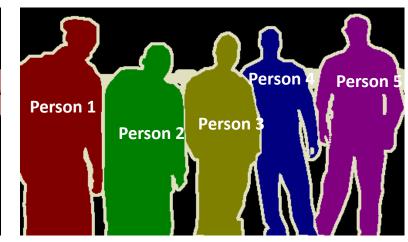
in collaboration with: Georgia Gkioxari, Piotr Dollár, and Ross Girshick Facebook AI Research (FAIR)

Introduction

Visual Perception Problems







Object Detection



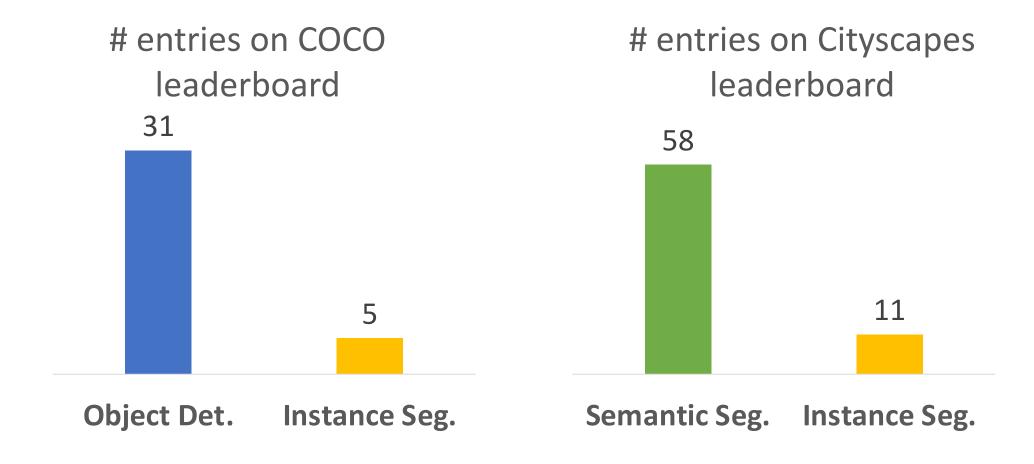
Instance Segmentation





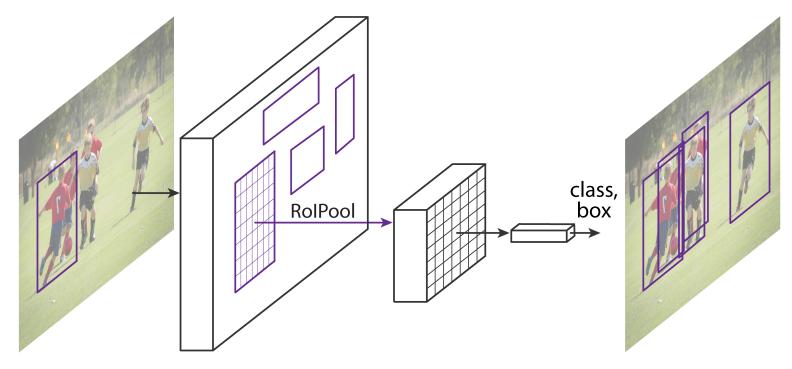


A Challenging Problem...



Object Detection

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



Semantic Segmentation

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

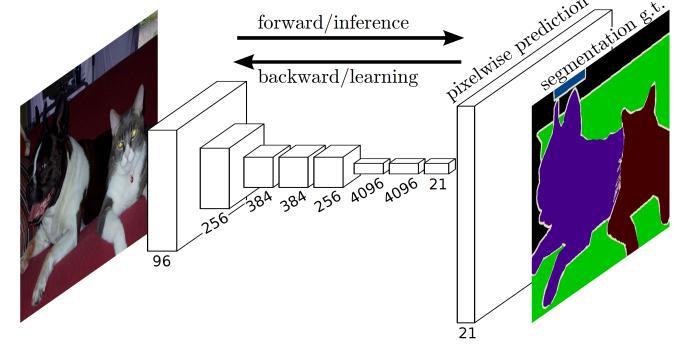
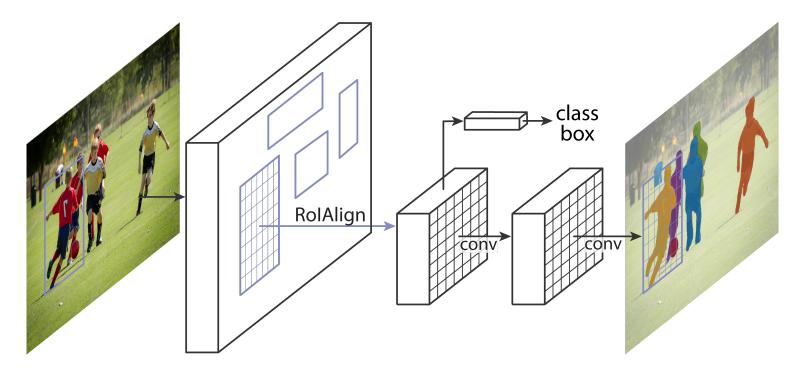


Figure credit: Long et al

Instance Segmentation

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

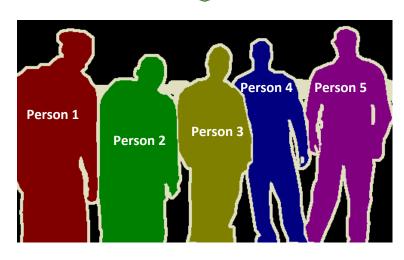


Instance Segmentation Methods

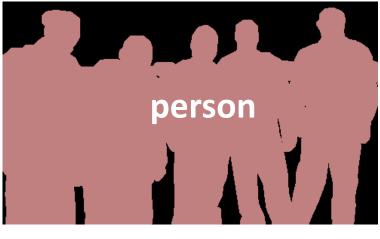
R-CNN driven



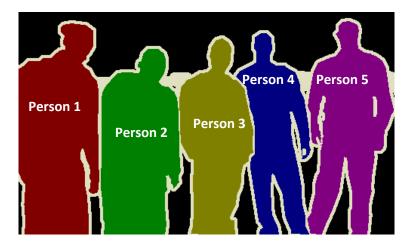




FCN driven

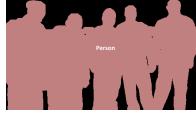




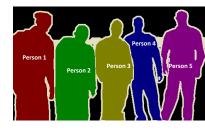




Instance Segmentation Methods



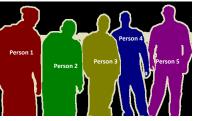




FCN-driven

- 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]



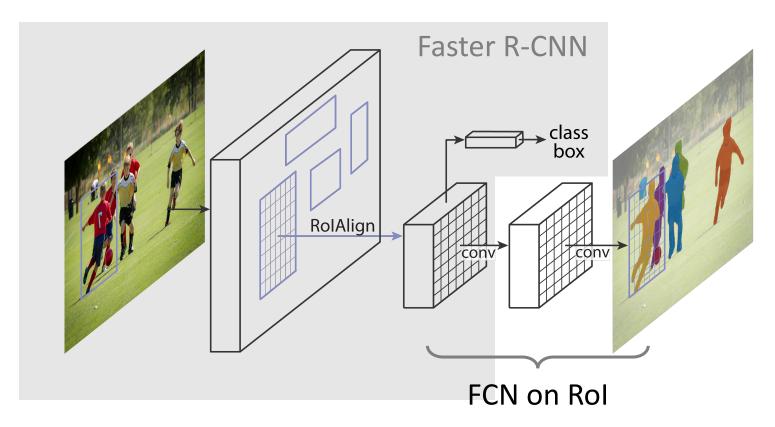


RCNN-driven

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

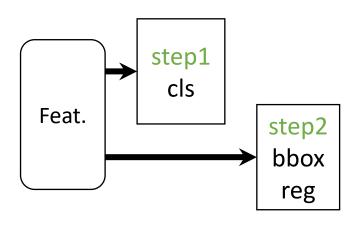
Mask R-CNN

• Mask R-CNN = Faster R-CNN with FCN on Rols

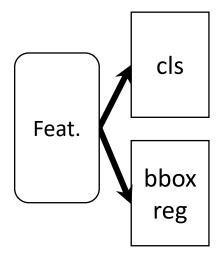


Parallel Heads

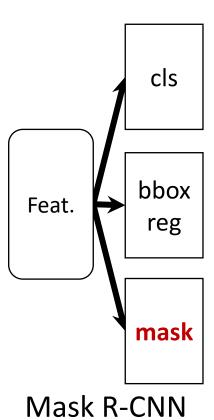
• Easy, fast to implement and train



(slow) R-CNN



Fast/er R-CNN



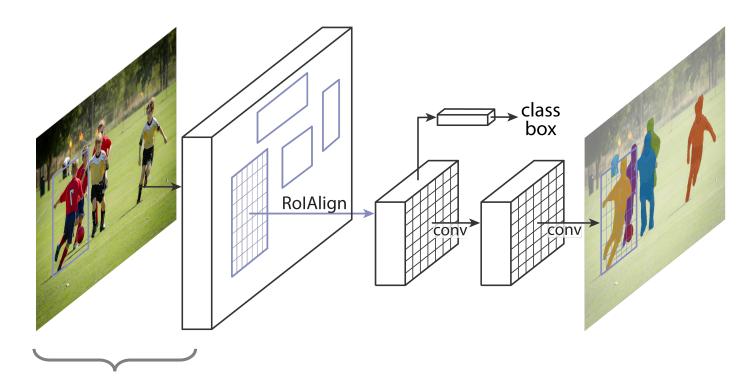
Invariance vs. Equivariance

Convolutions are translation-equivariant

• Fully-ConvNet (FCN) is translation-equivariant

 ConvNet becomes translation-invariant due to fully-connected or global pool layers

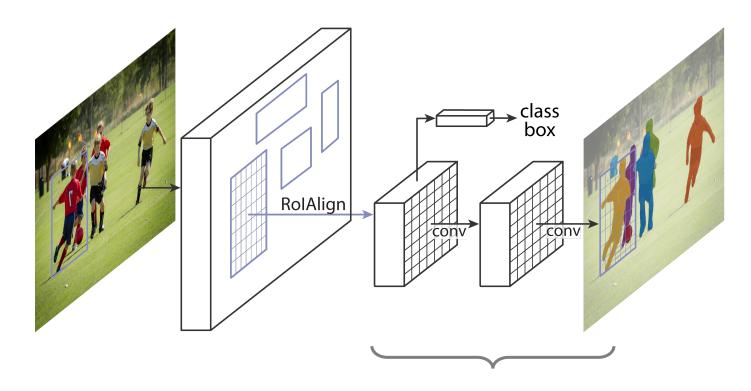
Equivariance in Mask R-CNN



1. Fully-Conv Features:

equivariant to global (image) translation

Equivariance in Mask R-CNN



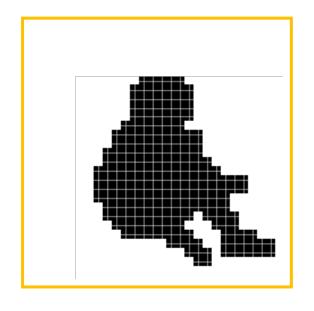
2. Fully-Conv on Rol:

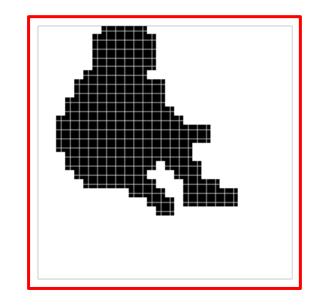
equivariant to translation within Rol

Fully-Conv on Rol



target masks on Rols

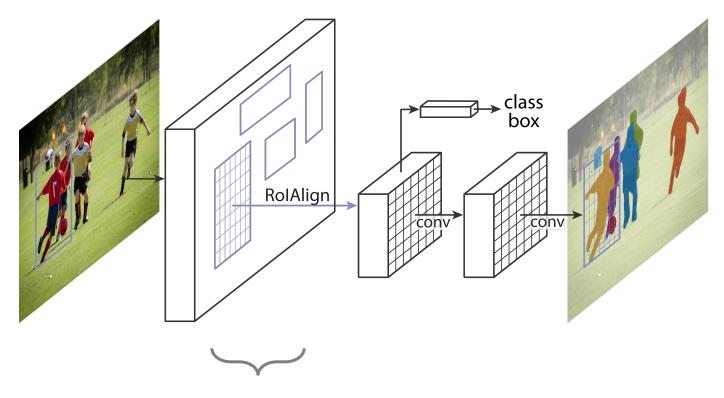




Translation of object in RoI => Same translation of mask in RoI

- Equivariant to small translation of Rols
- More robust to Rol's localization imperfection

Equivariance in Mask R-CNN



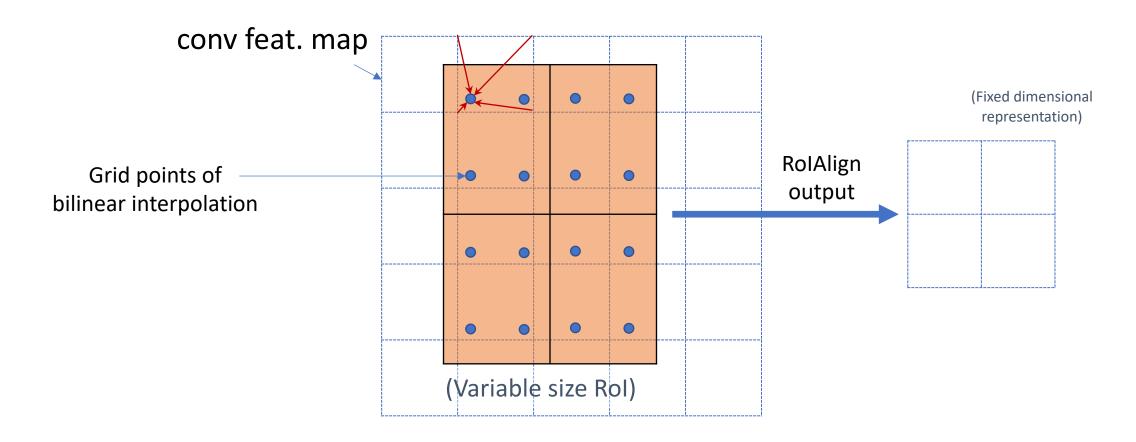
3. RolAlign:

3a. maintain translation-equivariance before/after Rol

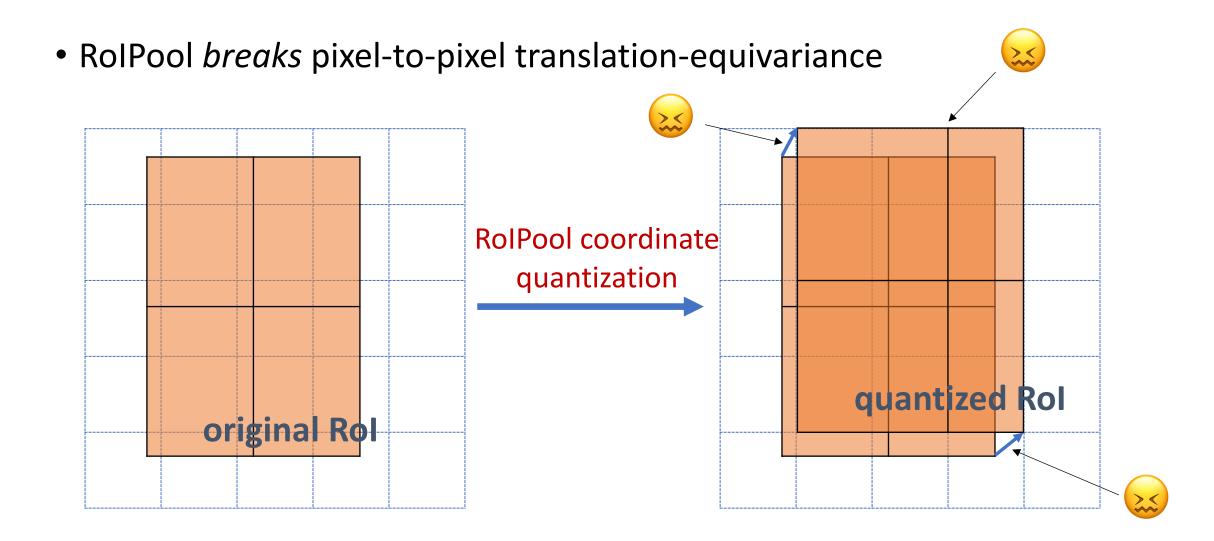
RolAlign

FAQs: how to sample grid points within a cell?

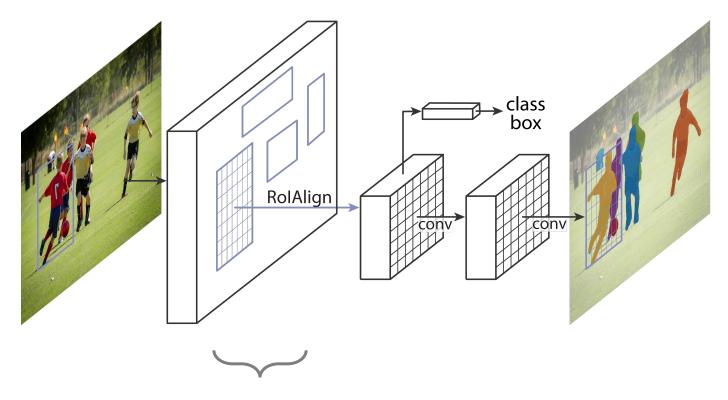
- 4 regular points in 2x2 sub-cells
- other implementation could work



RolAlign vs. RolPool



Equivariance in Mask R-CNN



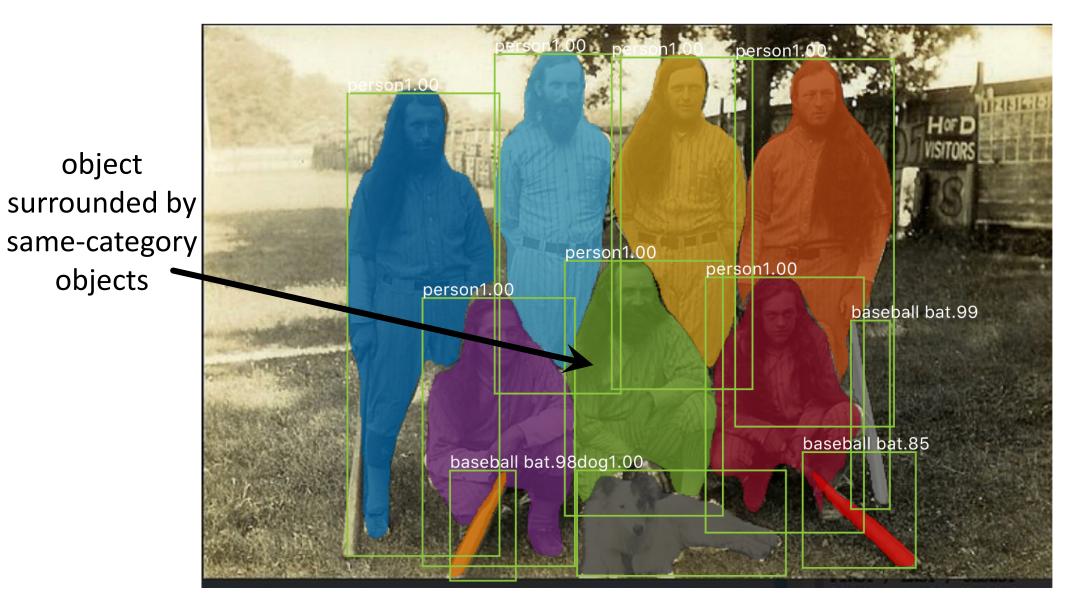
3. RolAlign:

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

Equivariance in Mask R-CNN: Summary

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

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



Mask R-CNN results on COCO

Result Analysis

Ablation: RolPool vs. RolAlign

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

	mask AP			box AP			
	AP	AP_{50}	AP ₇₅	AP^{bb}	$\mathrm{AP_{50}^{bb}}$	$\mathrm{AP^{bb}_{75}}$	
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	
•							

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

Ablation: RolPool vs. RolAlign

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

		mask AP			box AP	
	AP	AP_{50}	AP_{75}	AP ^{bb}	$\mathrm{AP_{50}^{bb}}$	$\mathrm{AP^{bb}_{75}}$
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}	$\mathrm{AP_{50}^{bb}}$	$\mathrm{AP^{bb}_{75}}$	AP^bb_S	$\mathrm{AP}^{\mathrm{bb}}_{M}$	$\mathrm{AP}^{\mathrm{bb}}_{L}$
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:

RolAlign

Object Detection Results on COCO

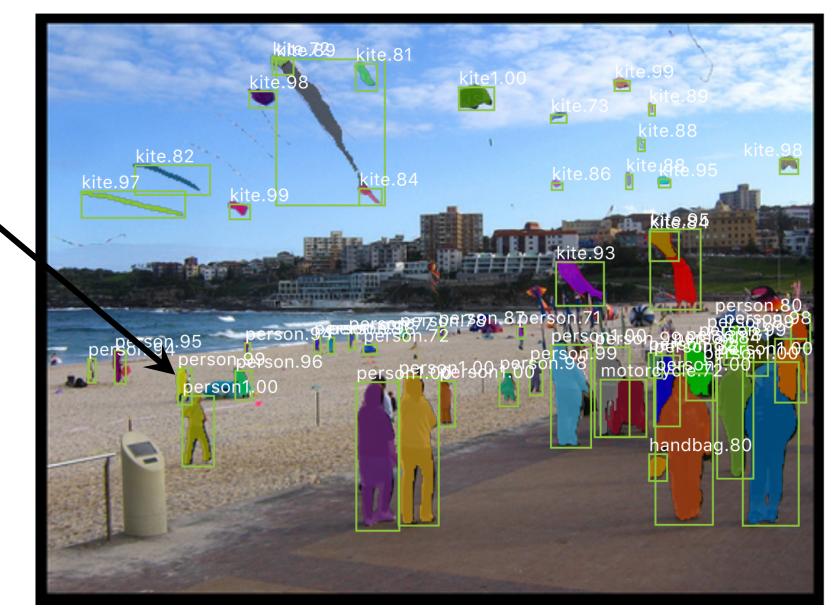
	backbone	AP ^{bb}	$\mathrm{AP_{50}^{bb}}$	$\mathrm{AP^{bb}_{75}}$	AP^bb_S	$\mathrm{AP}^{\mathrm{bb}}_{M}$	AP^bb_L
Faster R-CNN+++ [15]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [22]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [32]	34.7	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:

- RolAlign
- Multi-task training w/ mask

disconnected object o reperson1.00 person.98 surfboard1.00 surfboard1.00 surfboard

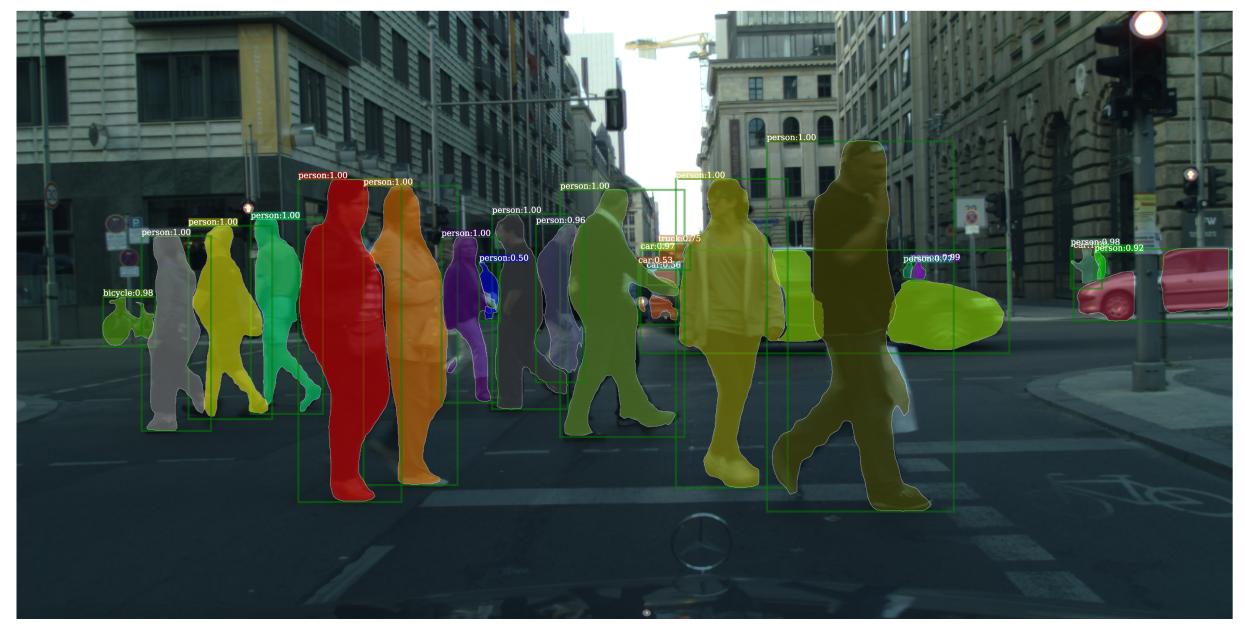
Mask R-CNN results on COCO



small

objects

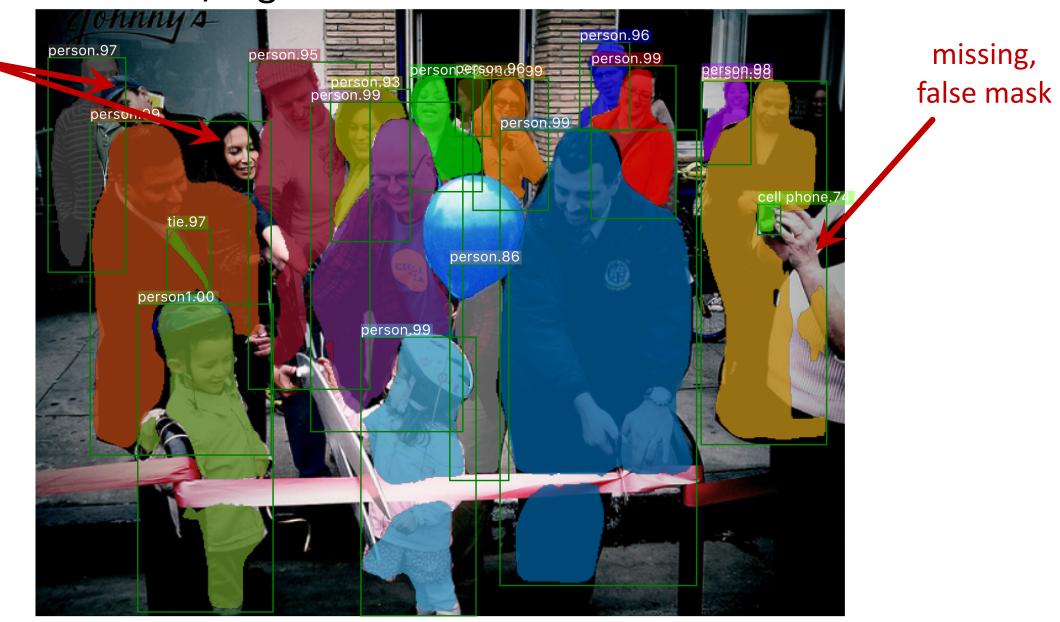
Mask R-CNN results on COCO



Mask R-CNN results on CityScapes

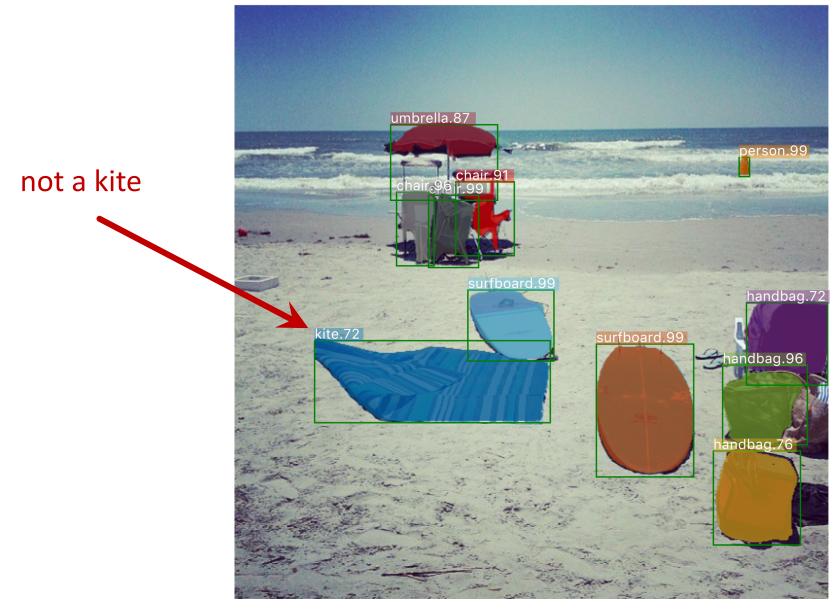
Failure case: detection/segmentation

missing

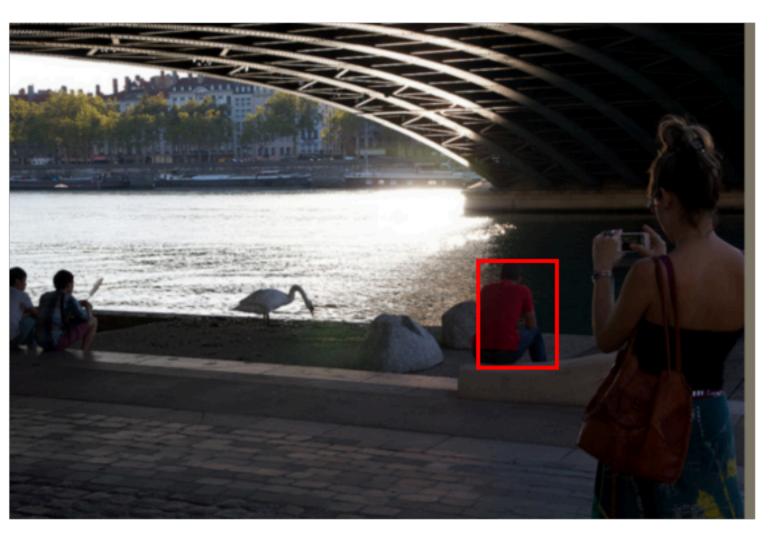


Mask R-CNN results on COCO

Failure case: recognition

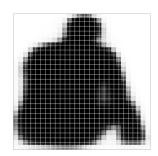


Mask R-CNN results on COCO



Validation image with box detection shown in red

28x28 soft prediction from Mask R-CNN (enlarged)

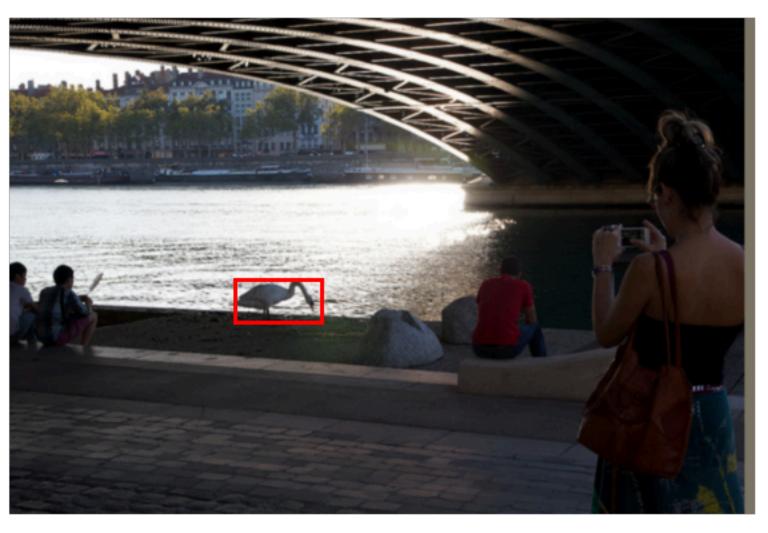


Soft prediction resampled to image coordinates (bilinear and bicubic interpolation work equally well)

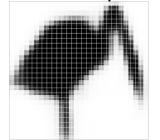


Final prediction (threshold at 0.5)





28x28 soft prediction



Resized Soft prediction



Final mask

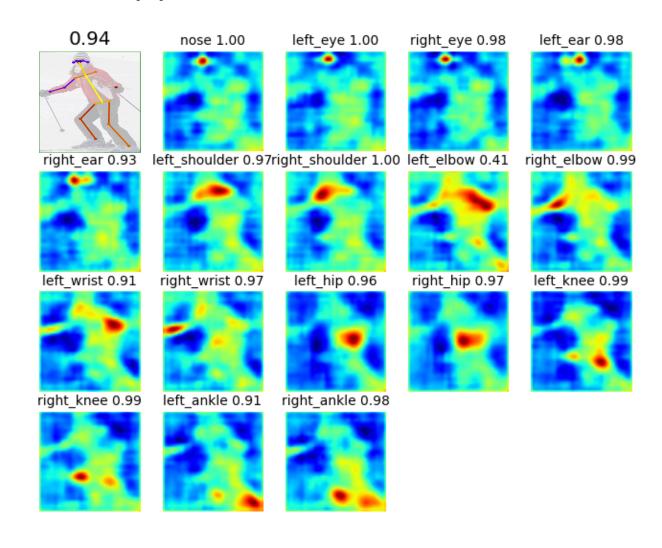


Validation image with box detection shown in red

Mask R-CNN: for Human Keypoint Detection

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

- Softmax over spatial locations
 - e.g. 56²-way softmax on 56x56
- Desire the same equivariances
 - translation, scale, aspect ratio



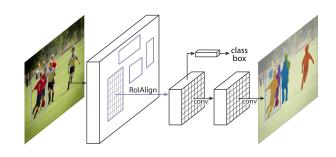




Conclusion

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