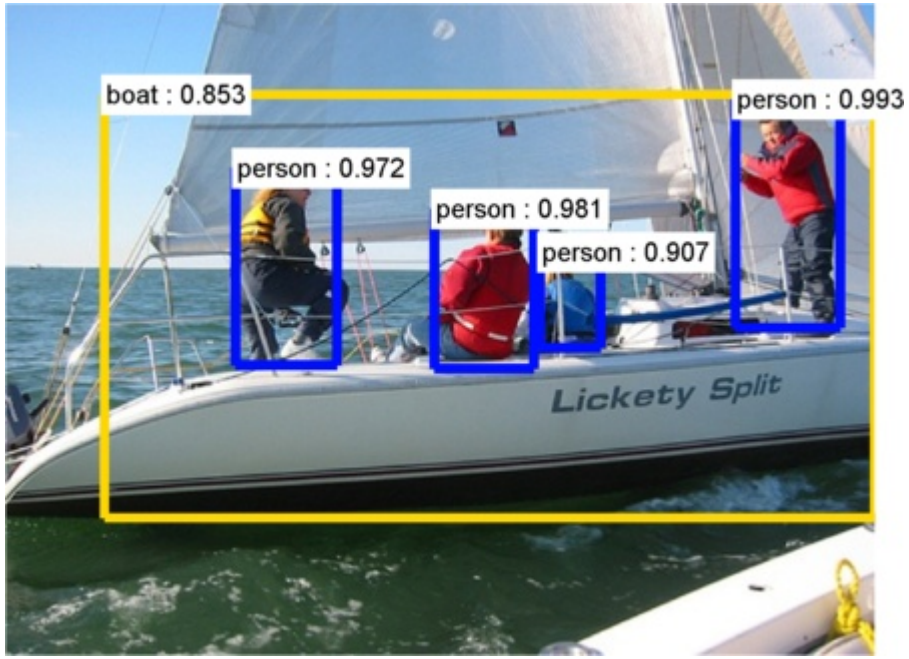
Object Detection

Lecture 5

Object Detection

Image Classification
(what?)



Object Detection
(what + where?)

Detection with ConvNets

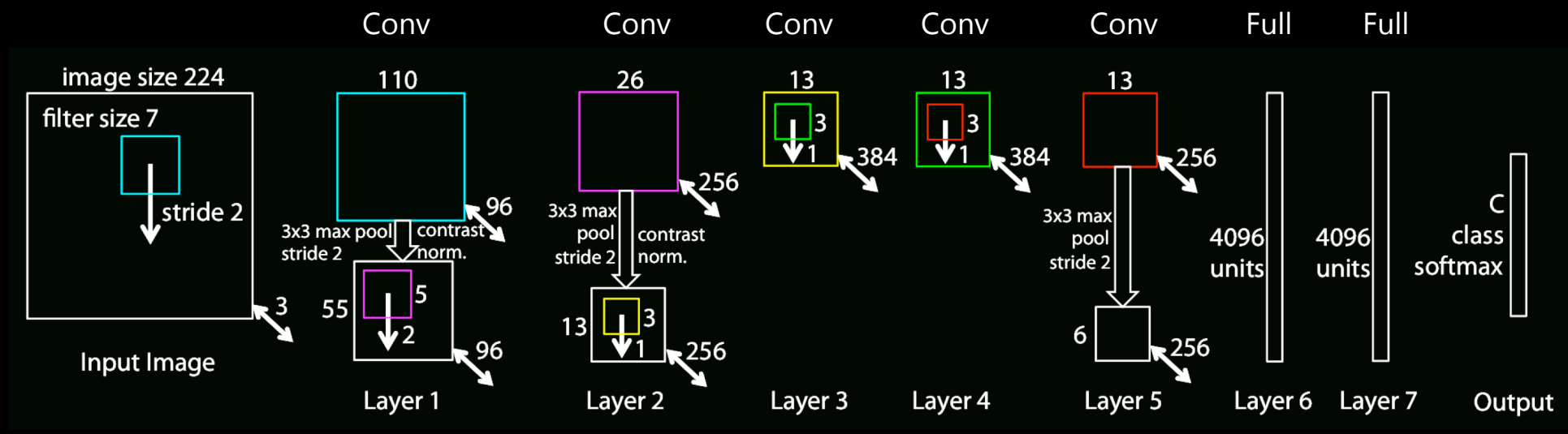
- So far, all about classification
- What about localizing objects within the scene?

Two General Approaches

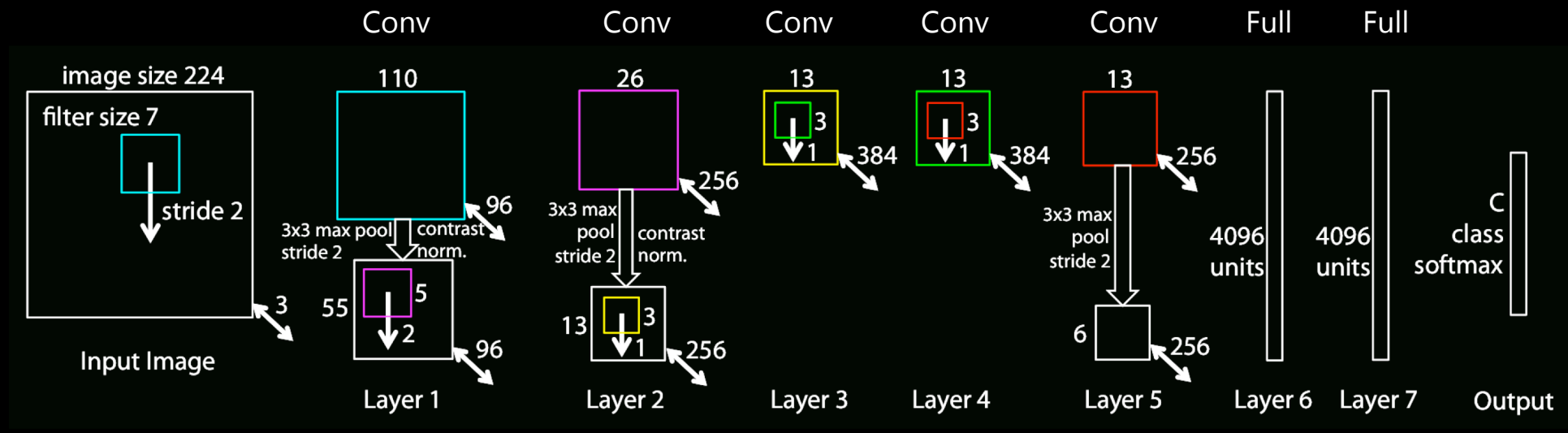
1. Examine very position / scale
   - E.g. Overfeat: Integrated recognition, localization and detection using convolutional networks, Sermanet et al., ICLR 2014

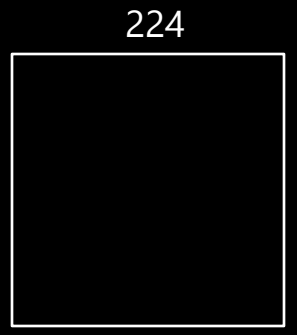
2. Use some kind of proposal mechanism to attend to a set of possible regions
   - E.g. Region-CNN [Rich feature hierarchies for accurate object detection and semantic segmentation, Girshick et al., CVPR 2014]

Sliding Window with ConvNet



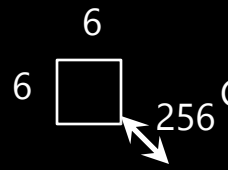
Sliding Window with ConvNet





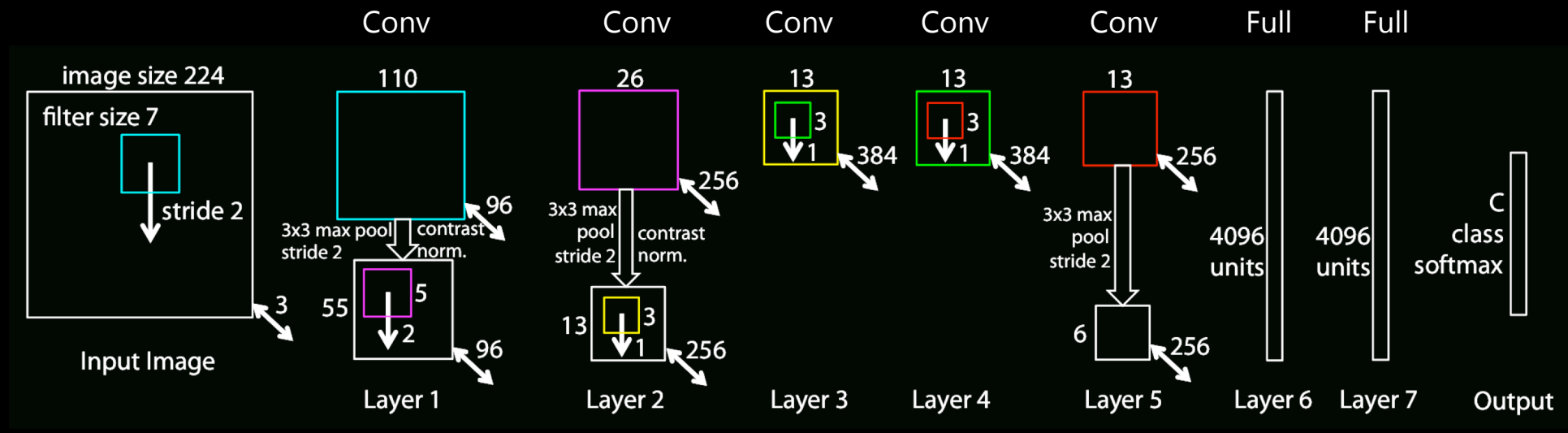
Input Window

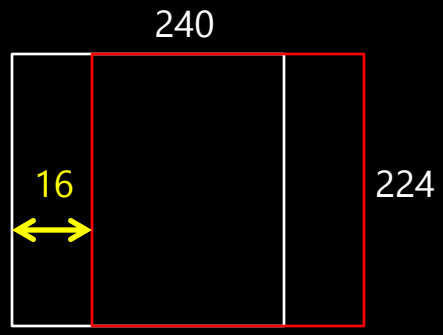
Feature Extractor



Classifier

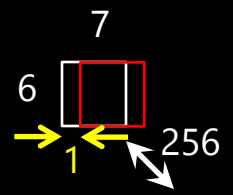
Sliding Window with ConvNet





Input Window

Feature Extractor



No need to compute two separate windows --- Just one big input window

Multi-Scale Sliding Window ConvNet

Feature Extractor

Feature Maps

256

256

256

256

Classifier

Class Maps

C=1000

C=1000

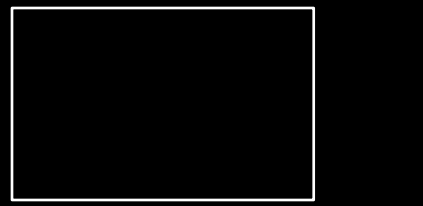
C=1000

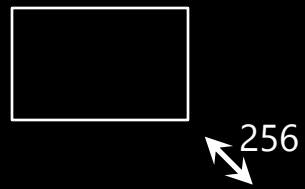
C=1000

Multi-Scale Sliding Window ConvNet

Feature Extractor

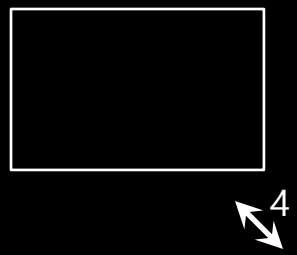
Feature Maps

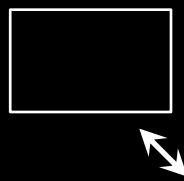


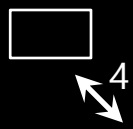


Regression Network

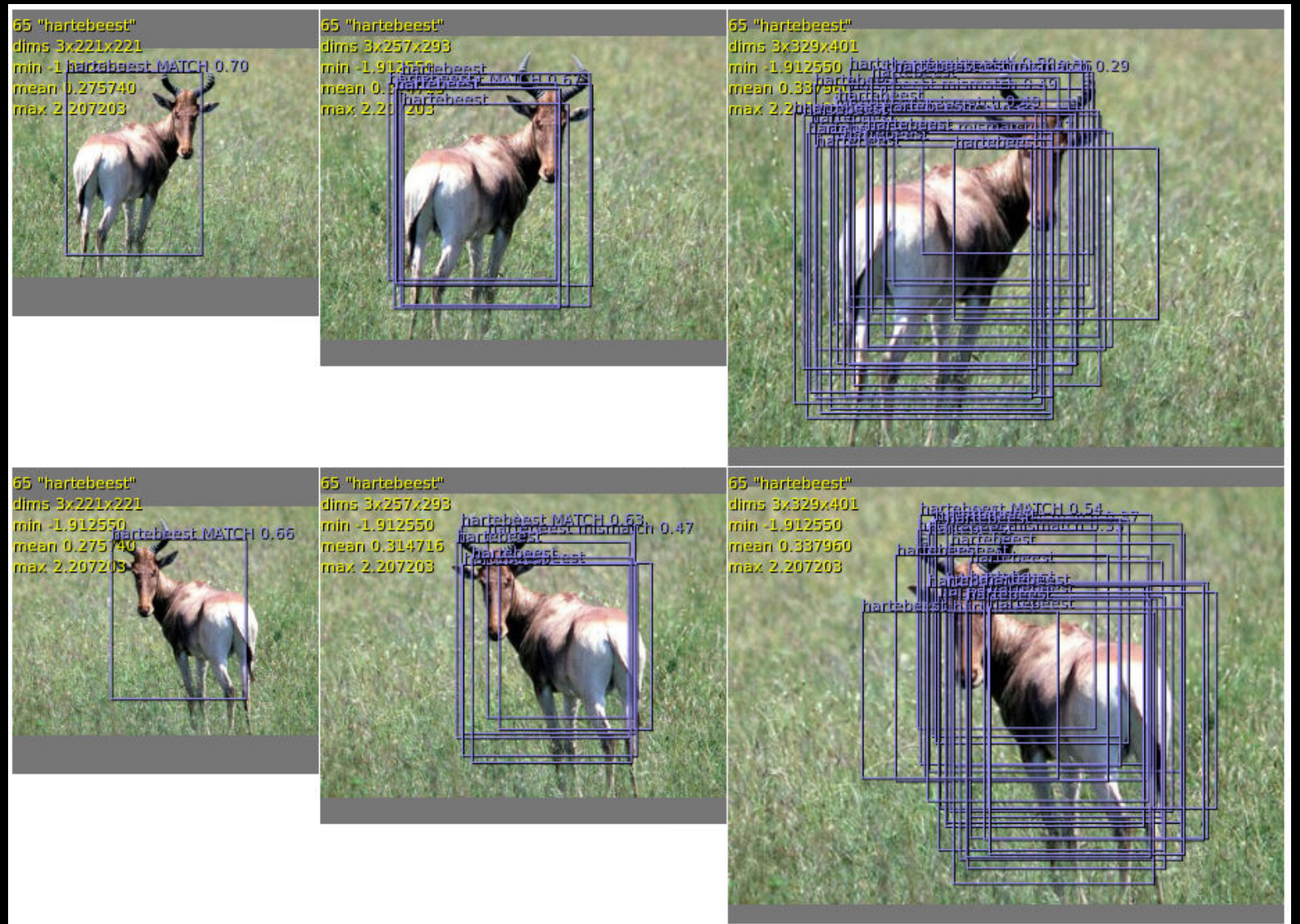
Bounding Box Maps





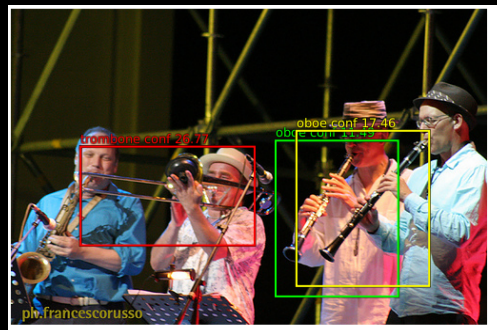


OverFeat – Output before NMS



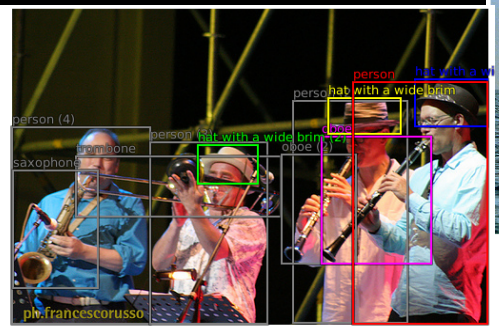
Overfeat Detection Results

[Sermanet et al. ICLR 2014]

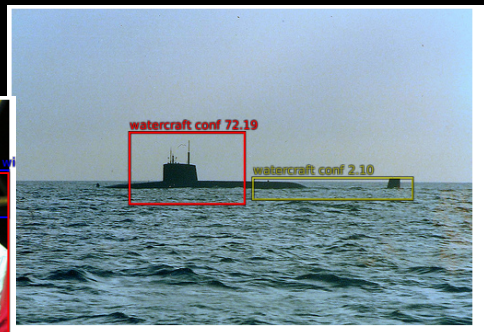


Top predictions:
trombone (confidence 26.8)
oboe (confidence 17.5)
oboe (confidence 11.5)

ILSVRC2012_val_00000614.JPEG

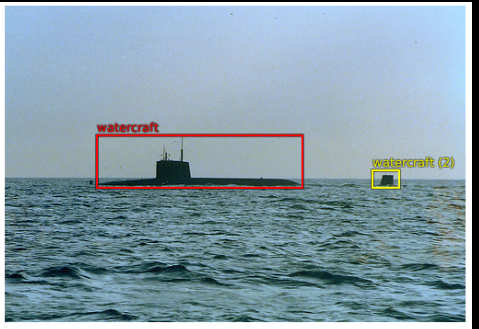


Groundtruth:
person
hat with a wide brim
hat with a wide brim (2)
hat with a wide brim (3)
oboe
oboe (2)
saxophone
trombone
person (2)
person (3)
person (4)



Top predictions:
watercraft (confidence 72.2)
watercraft (confidence 2.1)

ILSVRC2012_val_00000623.JPEG

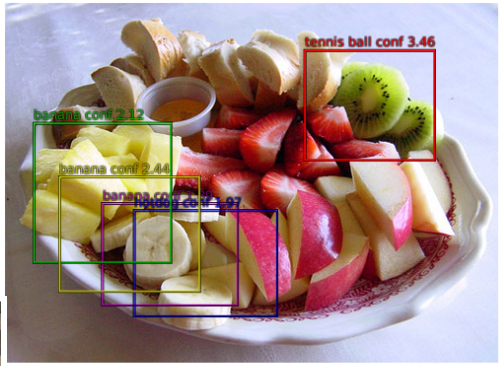


Groundtruth:
watercraft
watercraft (2)

Top predictions:
microwave (confidence 5.6)
refrigerator (confidence 2.5)

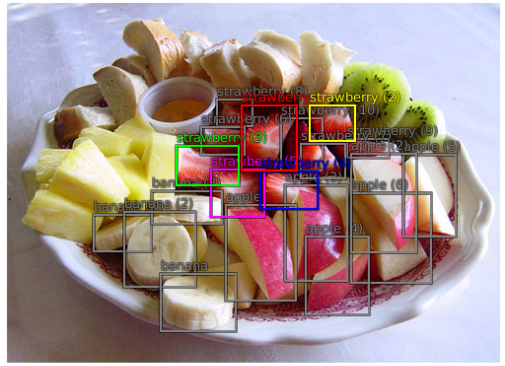
ILSVRC2012_val_00000519.JPEG

Groundtruth:
bowl
microwave



Top predictions:
tennis ball (confidence 3.5)
banana (confidence 2.4)
banana (confidence 2.1)
hotdog (confidence 2.0)
banana (confidence 1.9)

ILSVRC2012_val_00000320.JPEG



Groundtruth:
strawberry
strawberry (2)
strawberry (3)
strawberry (4)
strawberry (5)
strawberry (6)
strawberry (7)
strawberry (8)
strawberry (9)
strawberry (10)
apple
apple (2)
apple (3)

Two General Approaches

1. Examine very position / scale
   - E.g. Overfeat: Integrated recognition, localization and detection using convolutional networks, Sermanet et al., ICLR 2014

2. Use some kind of proposal mechanism to attend to a set of possible regions
   - E.g. Region-CNN [Rich feature hierarchies for accurate object detection and semantic segmentation, Girshick et al., CVPR 2014]

Reproducible research – get the code!

http://git.io/vBqm5

Fast R-CNN

Ross Girshick

Facebook AI Research (FAIR)

Work done at Microsoft Research

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

Localization
Where?

Recognition
What?

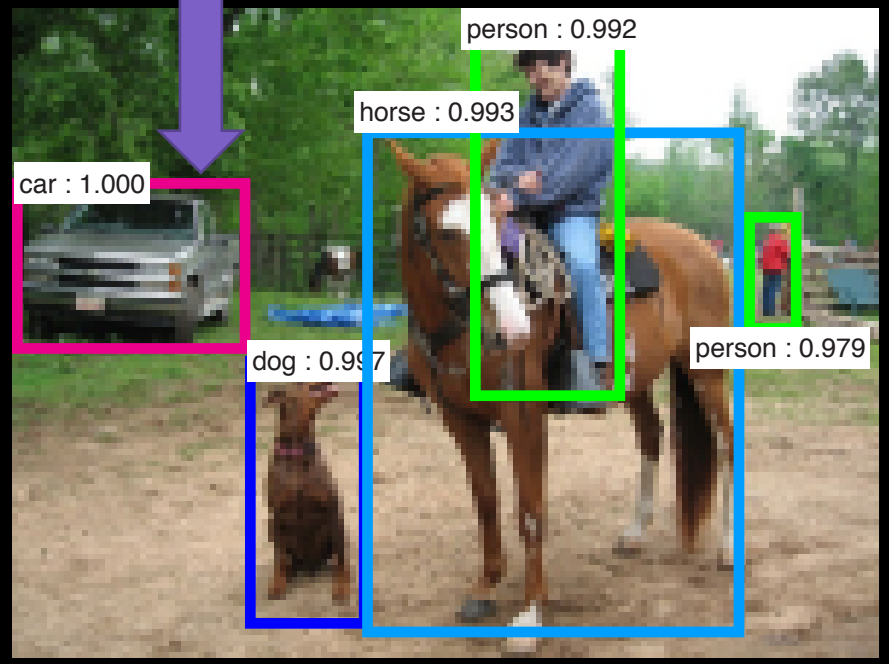
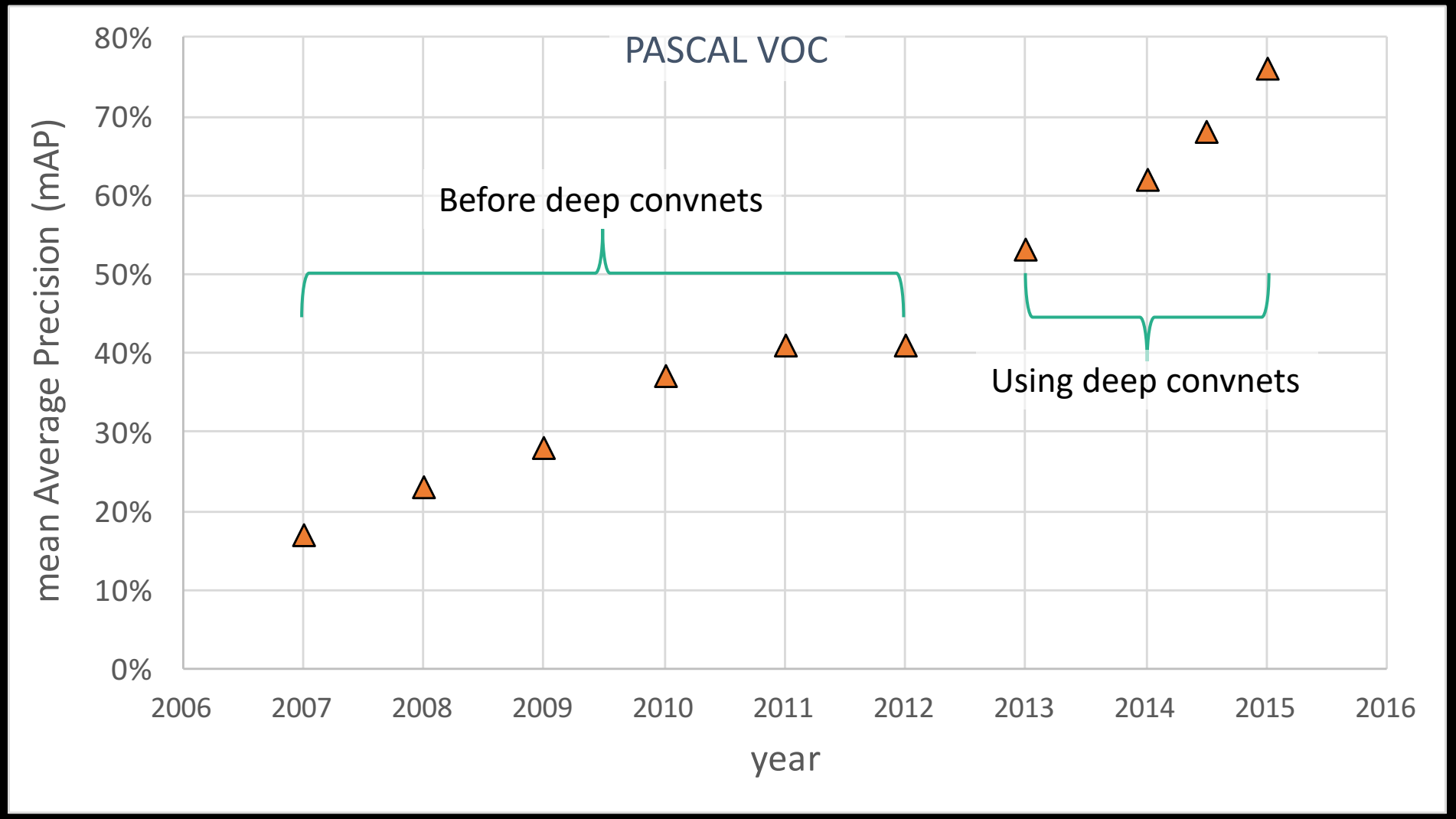
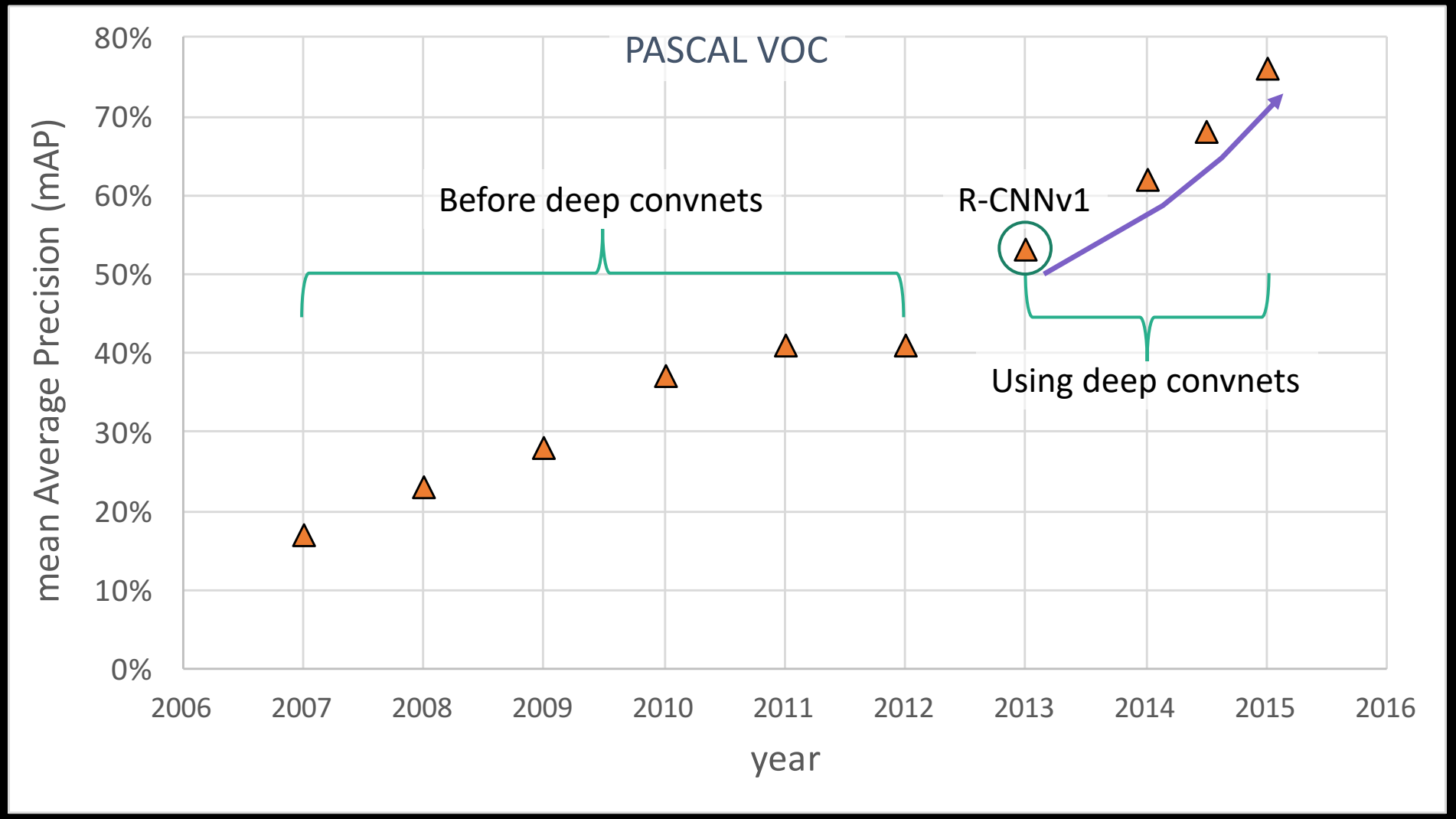


Figure adapted from Kaiming He

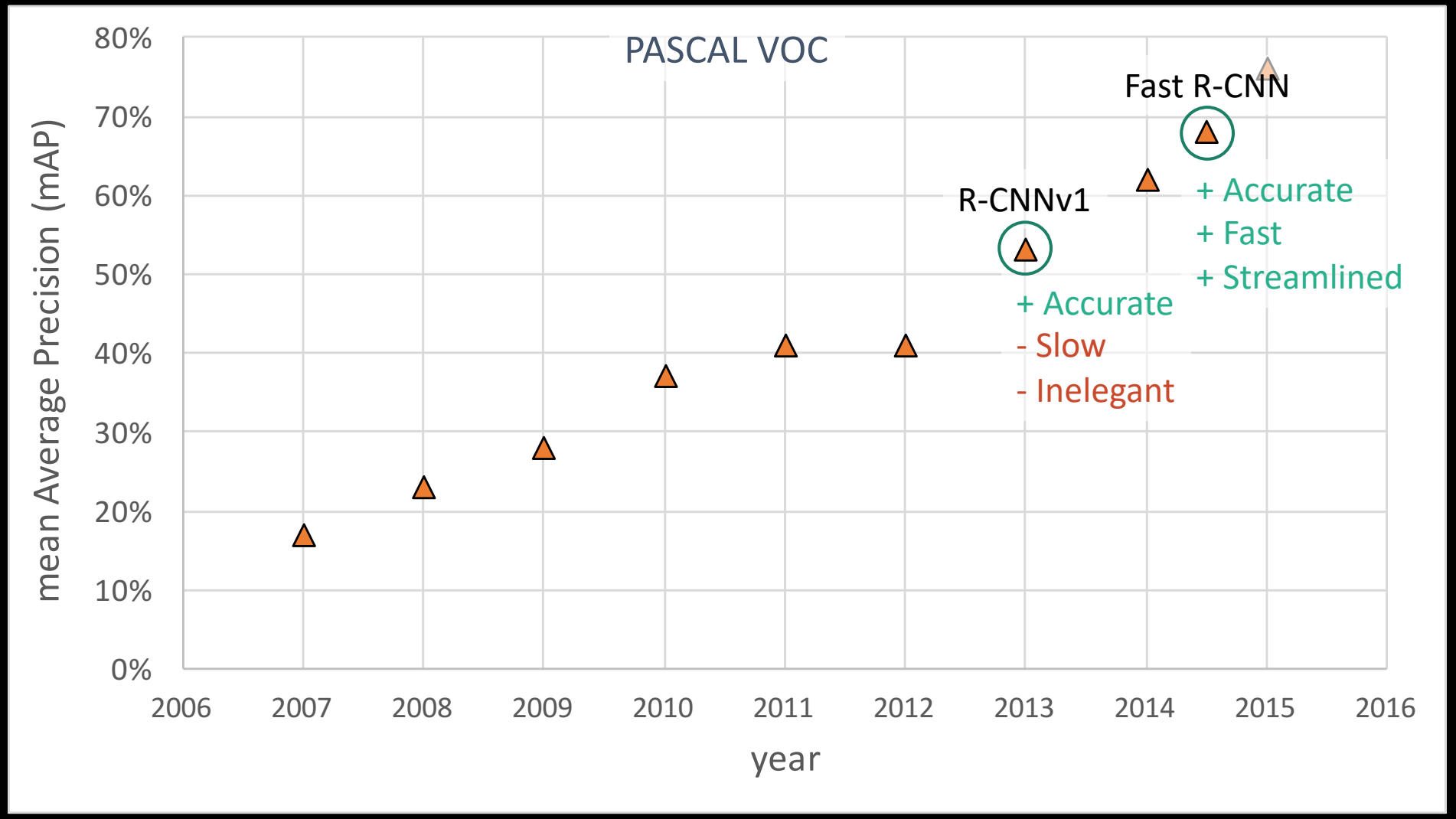
Object detection renaissance (2013-present)



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]

Slow R-CNN

Input image

Girshick et al. CVPR14.

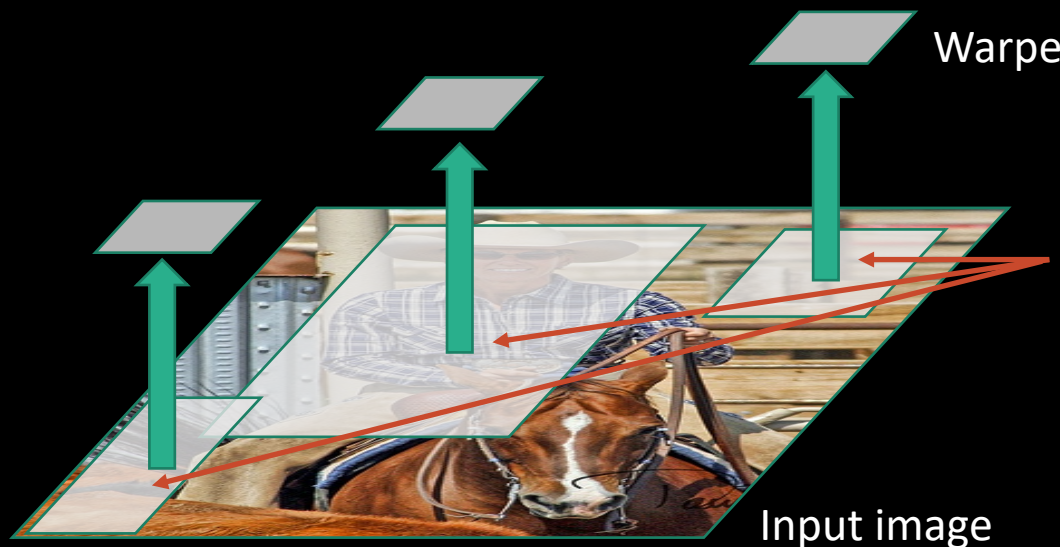
Slow R-CNN

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

Input image

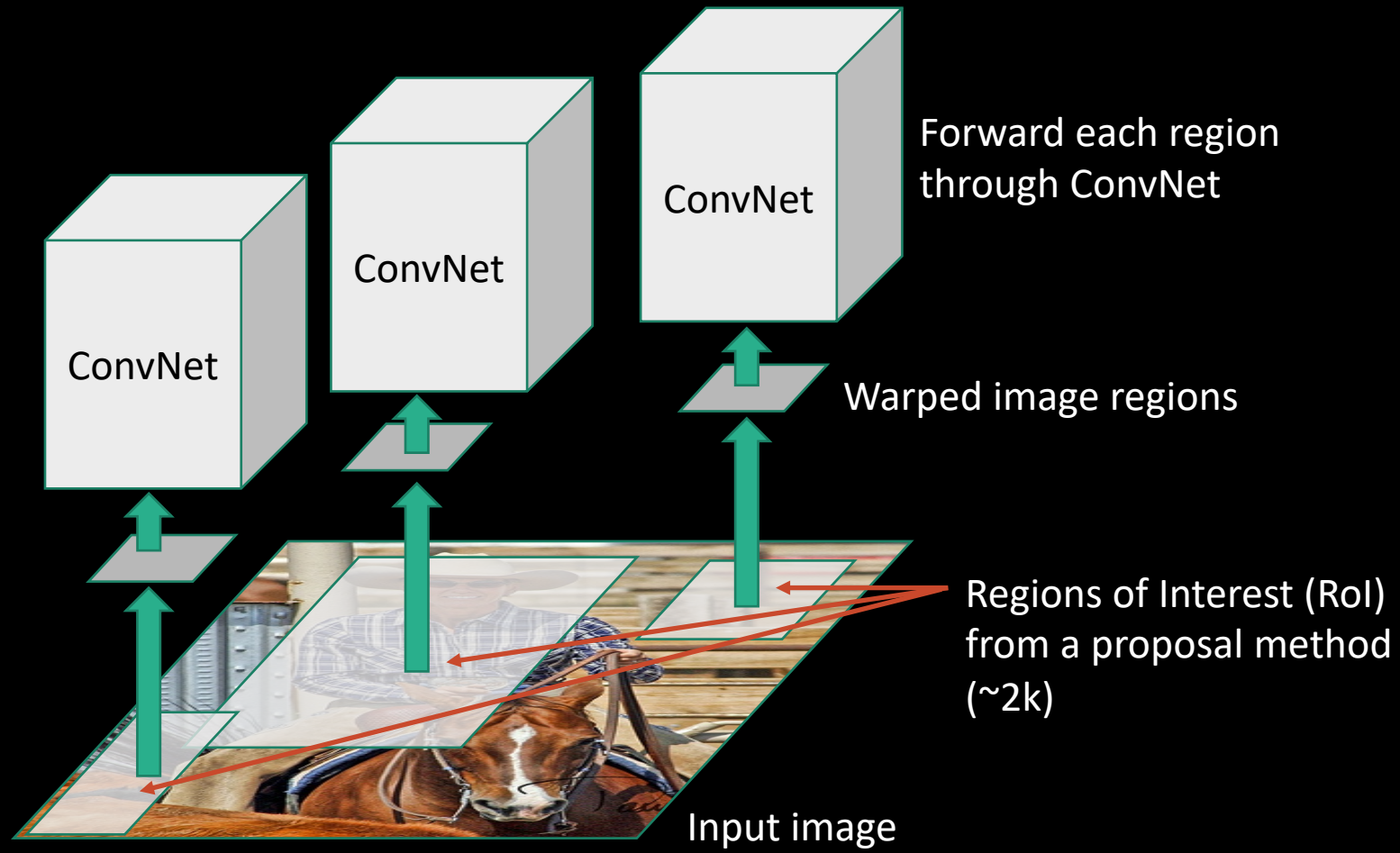
Girshick et al. CVPR14.

Slow R-CNN



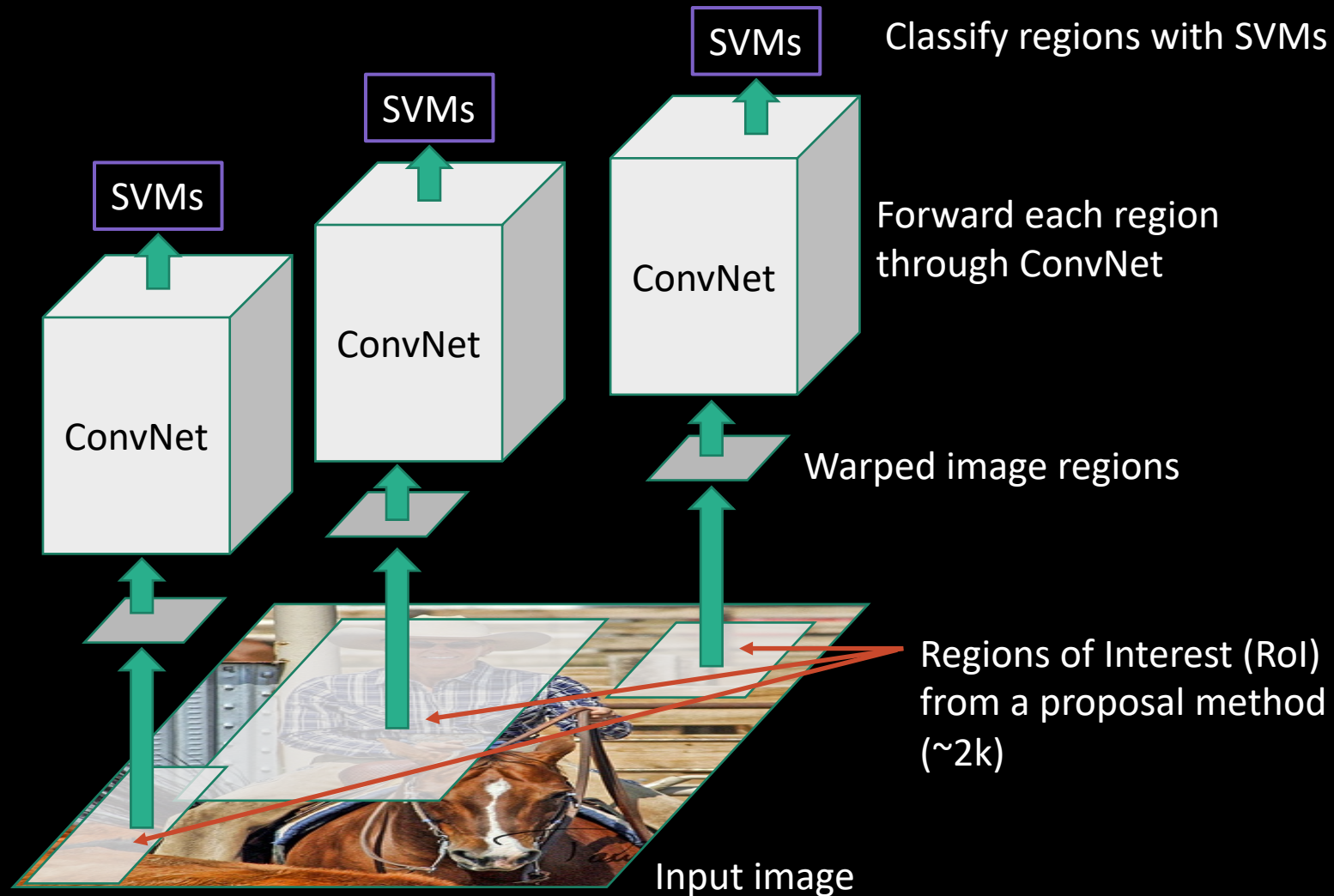
Girshick et al. CVPR14.

Slow R-CNN



Girshick et al. CVPR14.

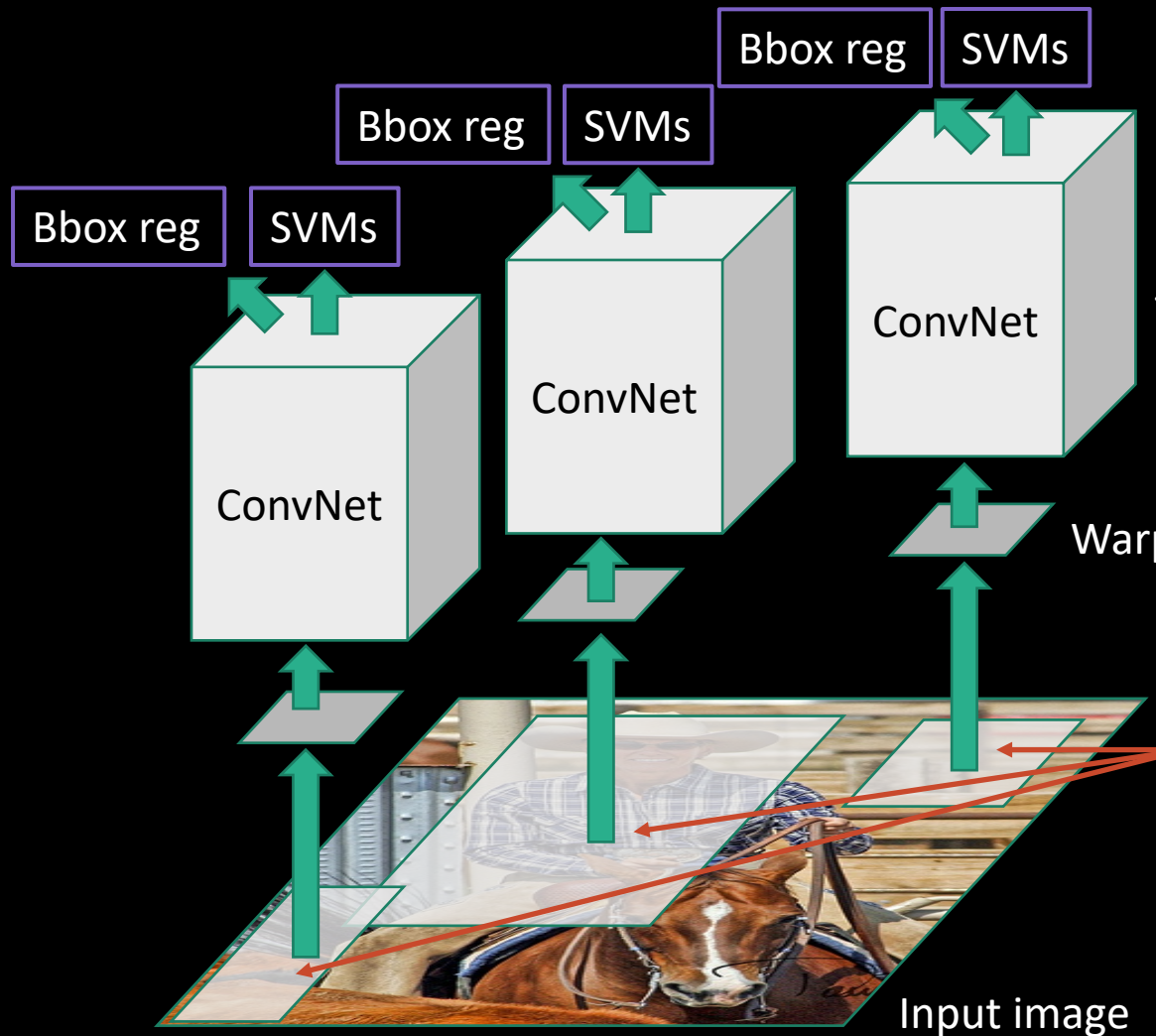
Slow R-CNN



Girshick et al. CVPR14.

Post hoc component

Slow R-CNN



Apply bounding-box regressors

Classify regions with SVMs

Forward each region through ConvNet

Warped image regions

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

Input image

Girshick et al. CVPR14.

Post hoc component

What’s wrong with slow R-CNN?

What's wrong with slow R-CNN?

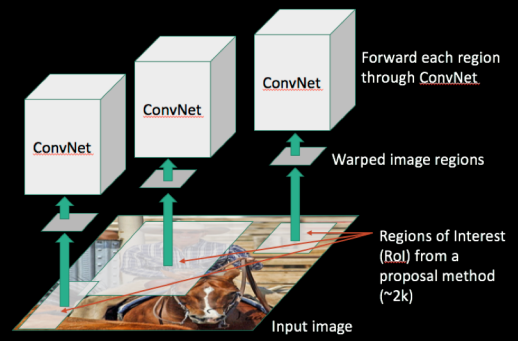
- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressors (squared loss)

What’s wrong with slow R-CNN?

- 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

What's wrong with slow R-CNN?

- 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]



~2000 ConvNet forward passes per image

SPP-net

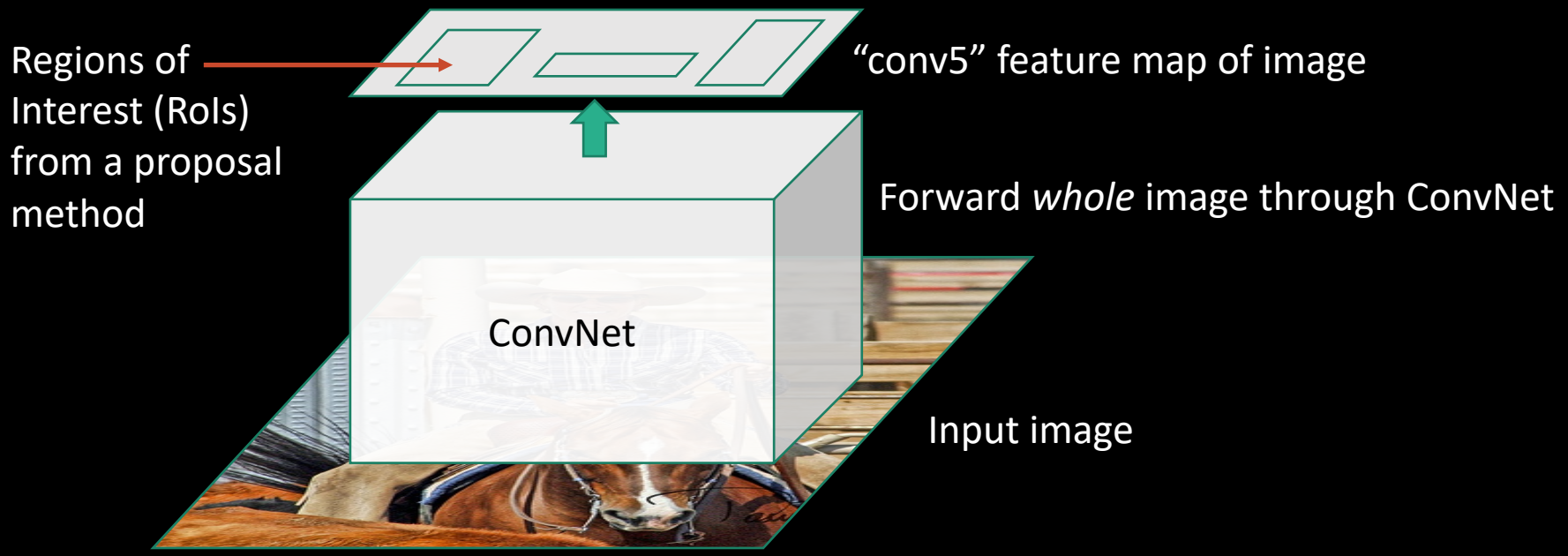
Input image

He et al. ECCV14.

SPP-net

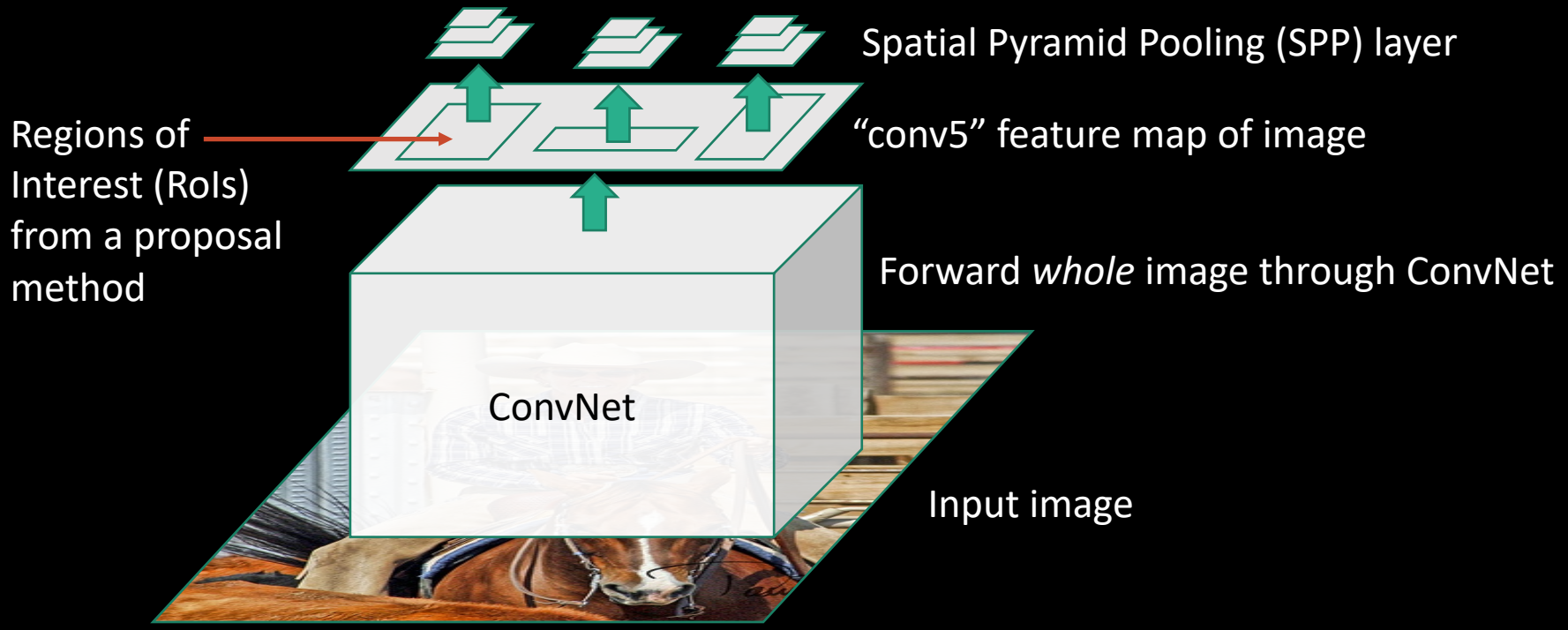
He et al. ECCV14.

SPP-net



He et al. ECCV14.

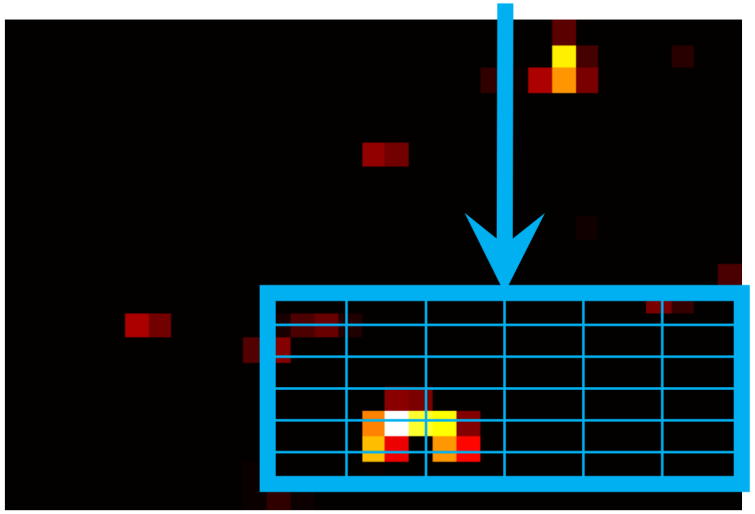
SPP-net

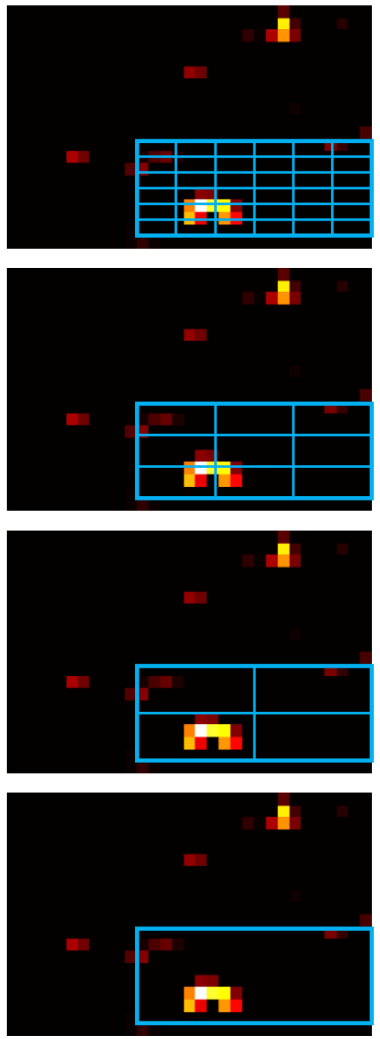


He et al. ECCV14.

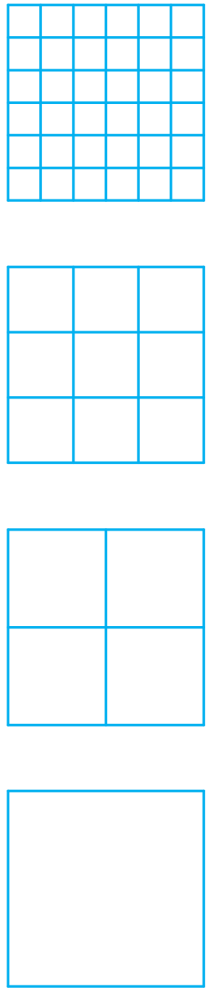
Spatial Pyramid Pooling (SPP) Layer

- fix the number of bins (instead of filter sizes)
- adaptively-sized bins





pooling



a finer level maintains explicit spatial information

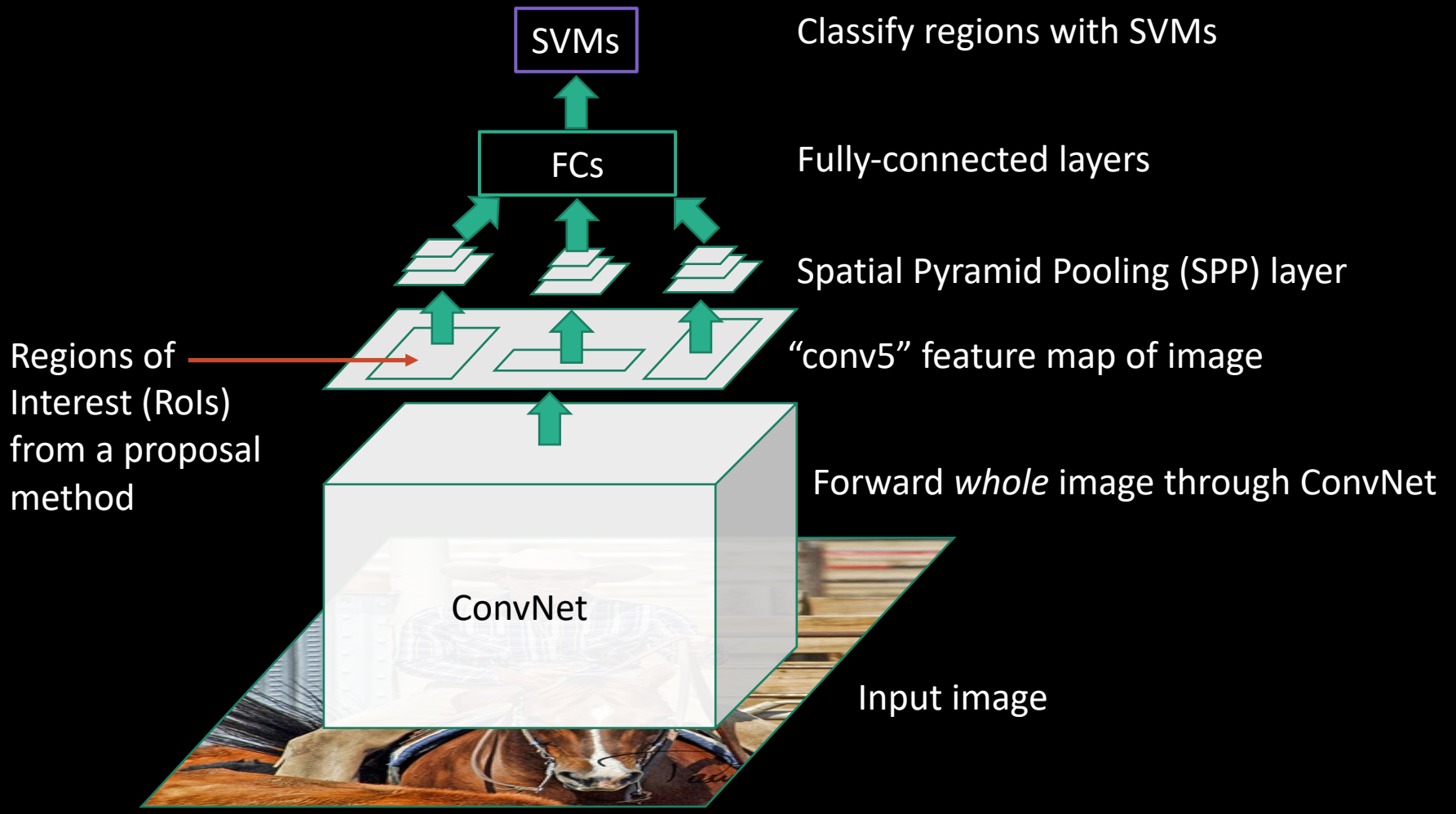
concatenate, fc layers…

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

ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”. ECCV 2014.

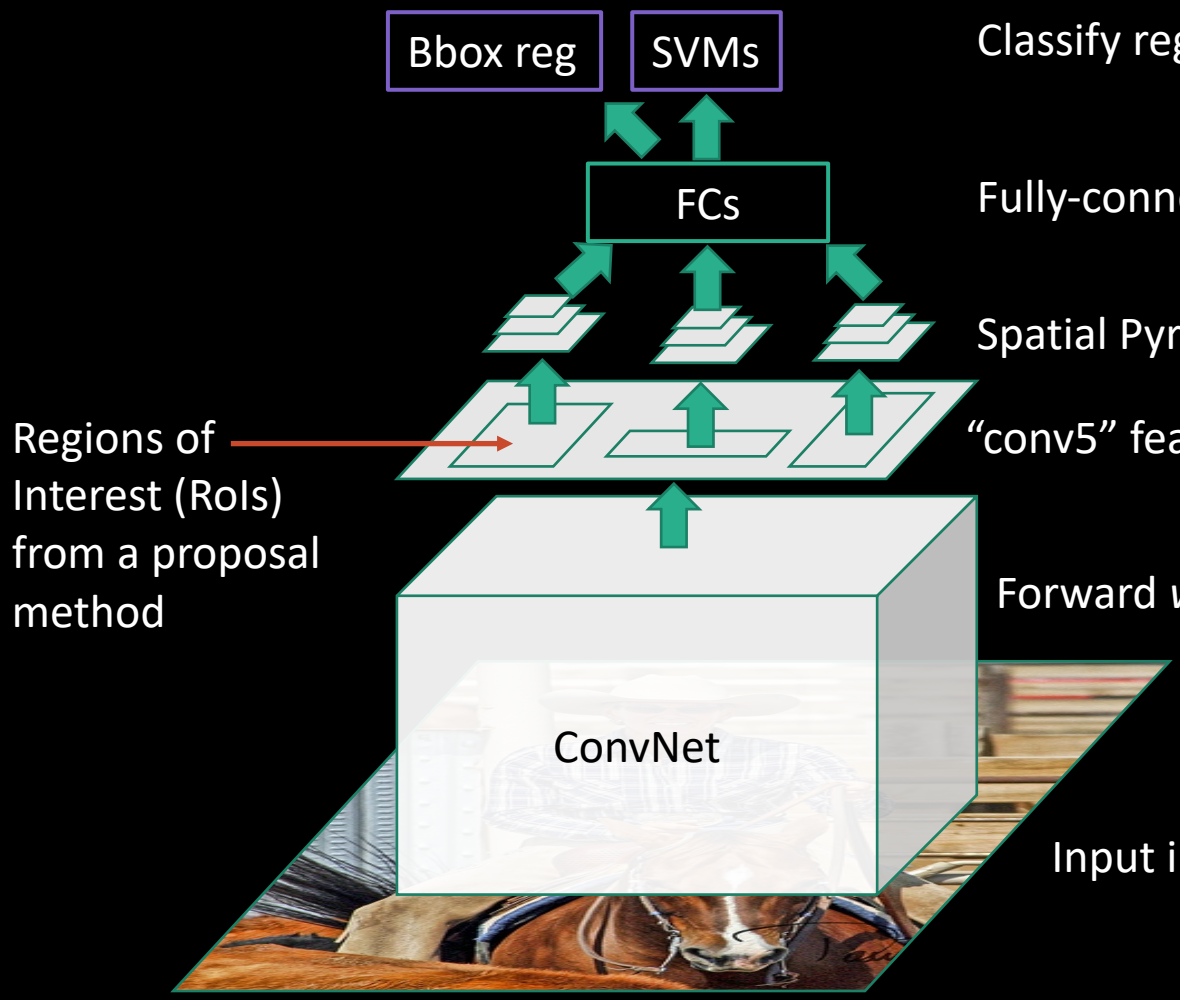
SPP-net



He et al. ECCV14.

Post hoc component

SPP-net



Apply bounding-box regressors

Classify regions with SVMs

Fully-connected layers

Spatial Pyramid Pooling (SPP) layer

"conv5" feature map of image

Forward *whole* image through ConvNet

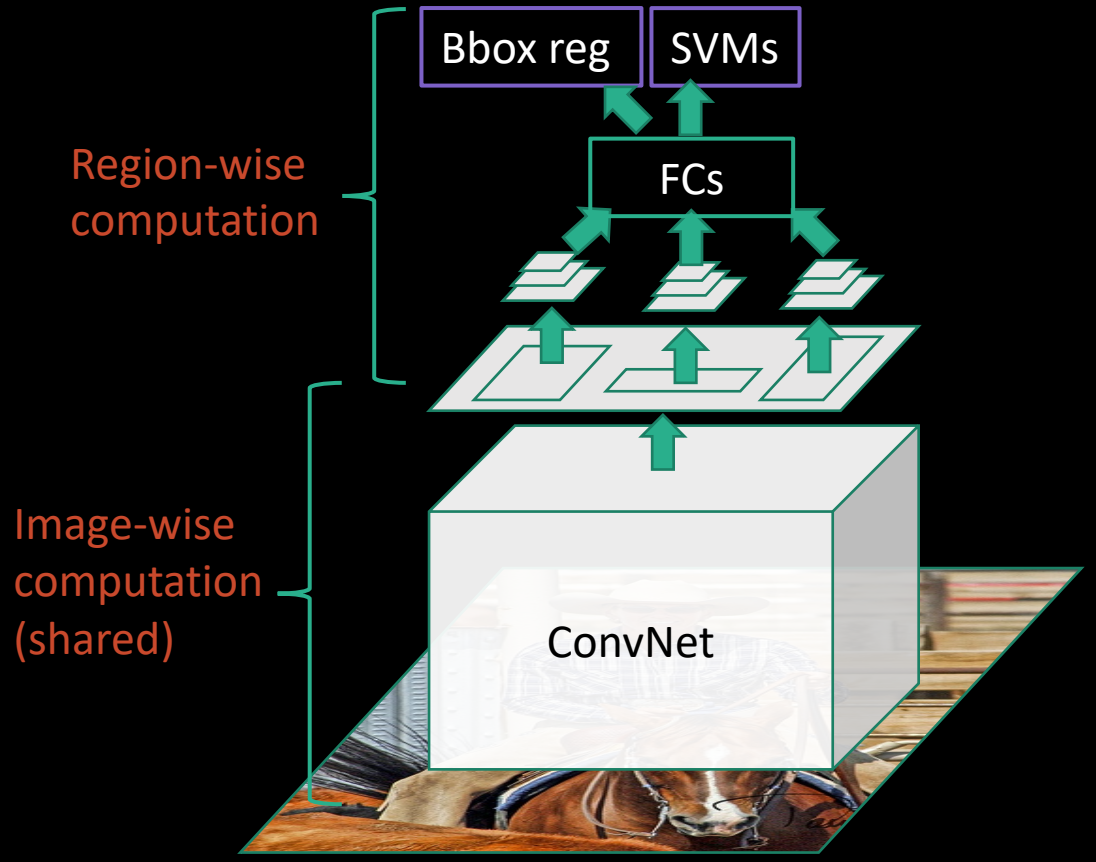
Input image

He et al. ECCV14.

Post hoc component

What's good about SPP-net?

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



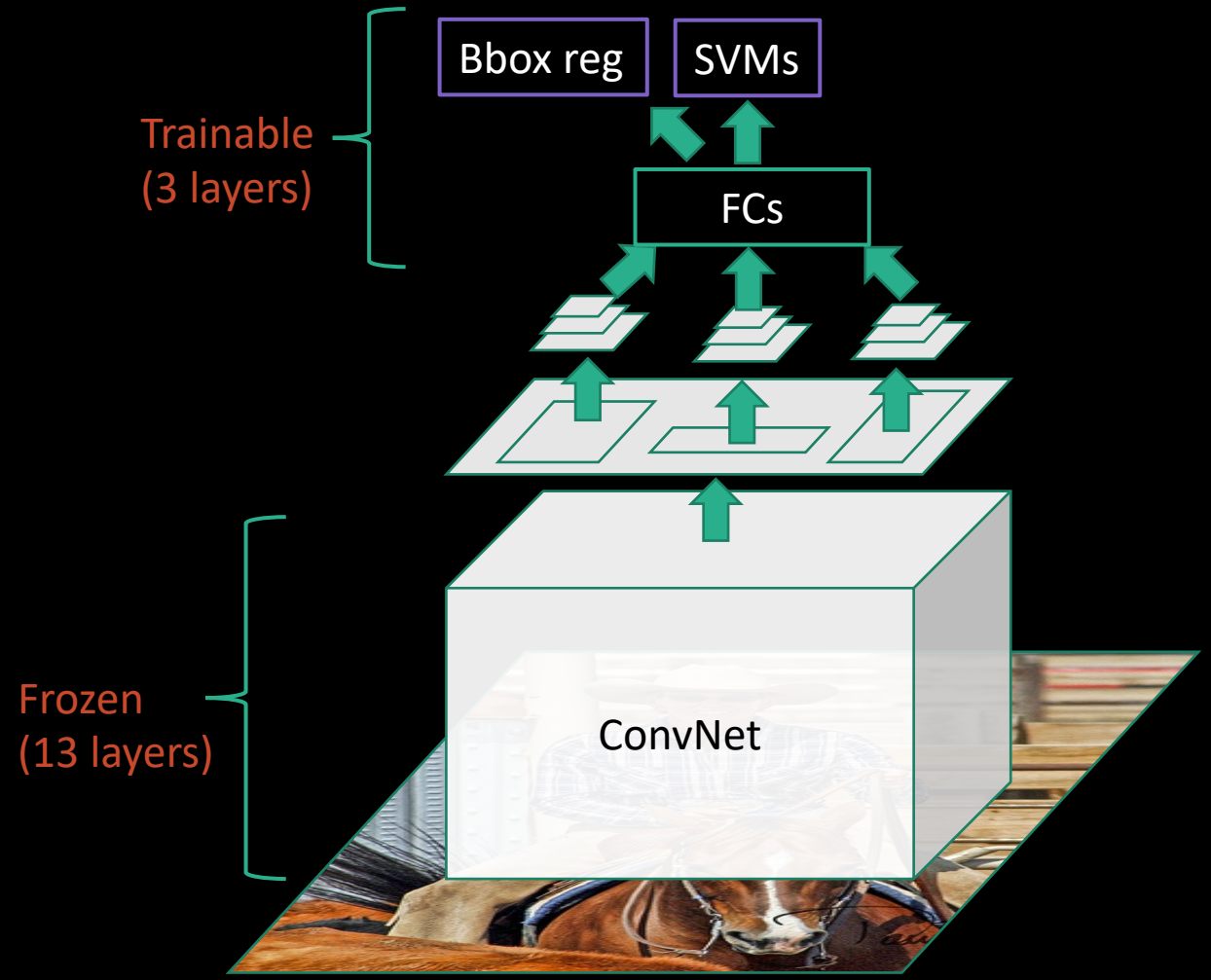
What's wrong with SPP-net?

- 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



He et al. ECCV14.

Post hoc component

Fast R-CNN

- Fast test-time, like SPP-net

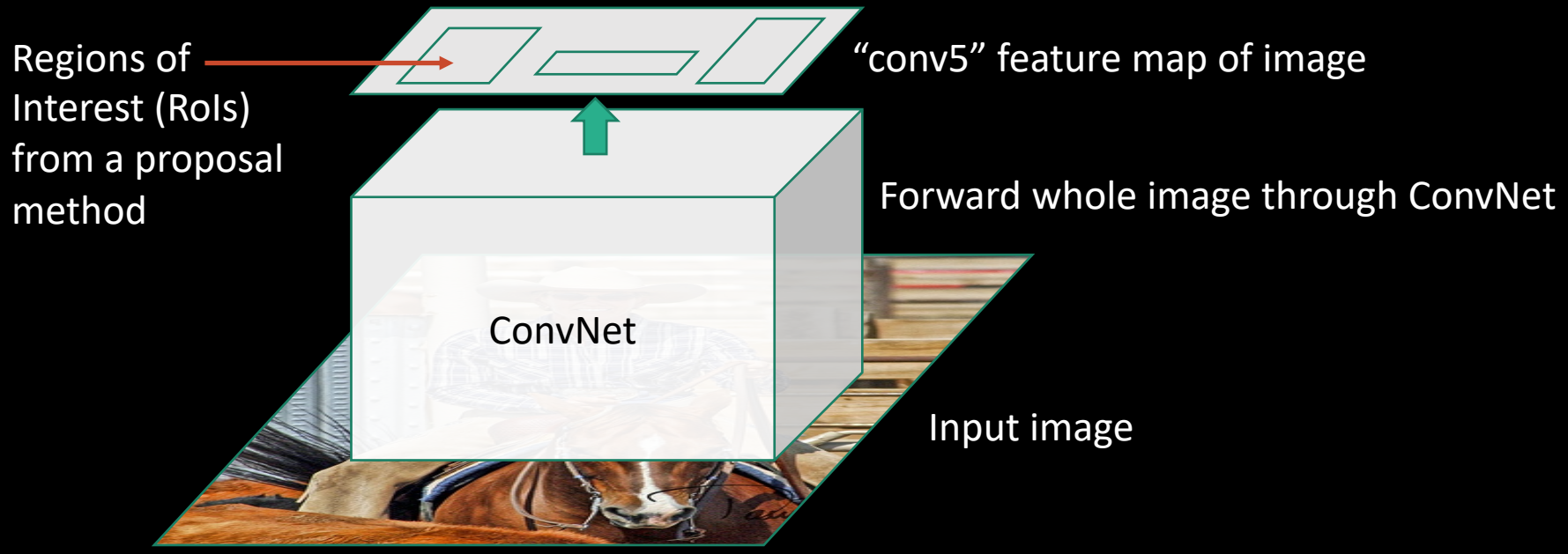
Fast R-CNN

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

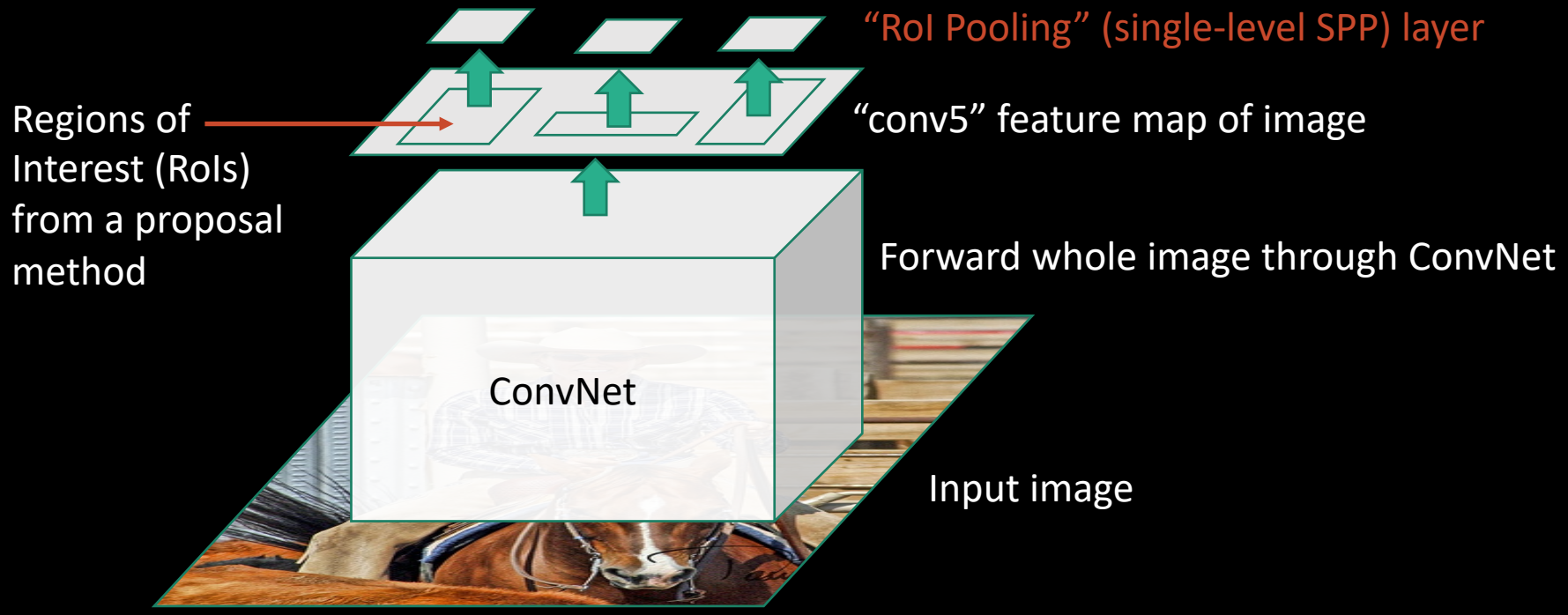
Fast R-CNN

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

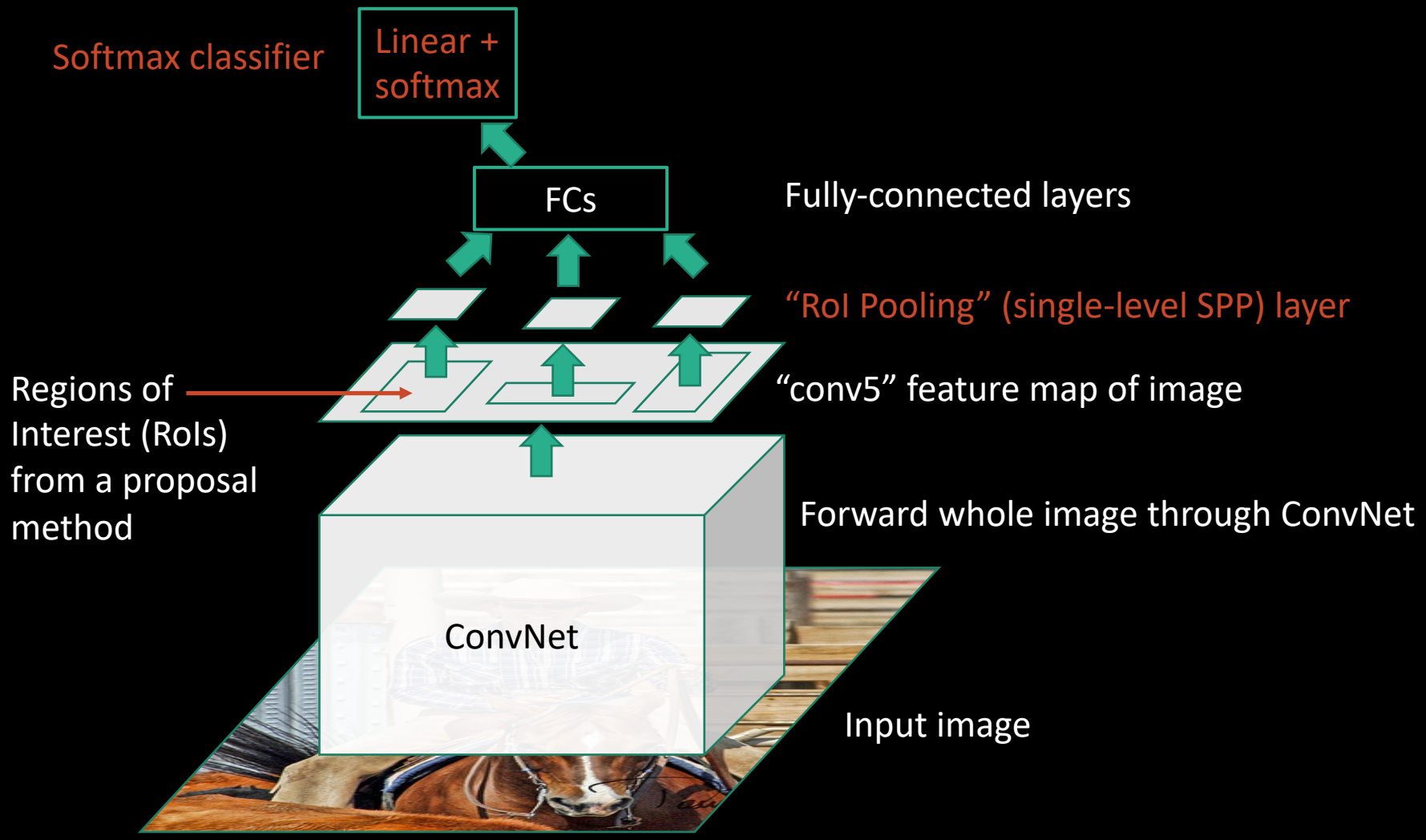
Fast R-CNN (test time)



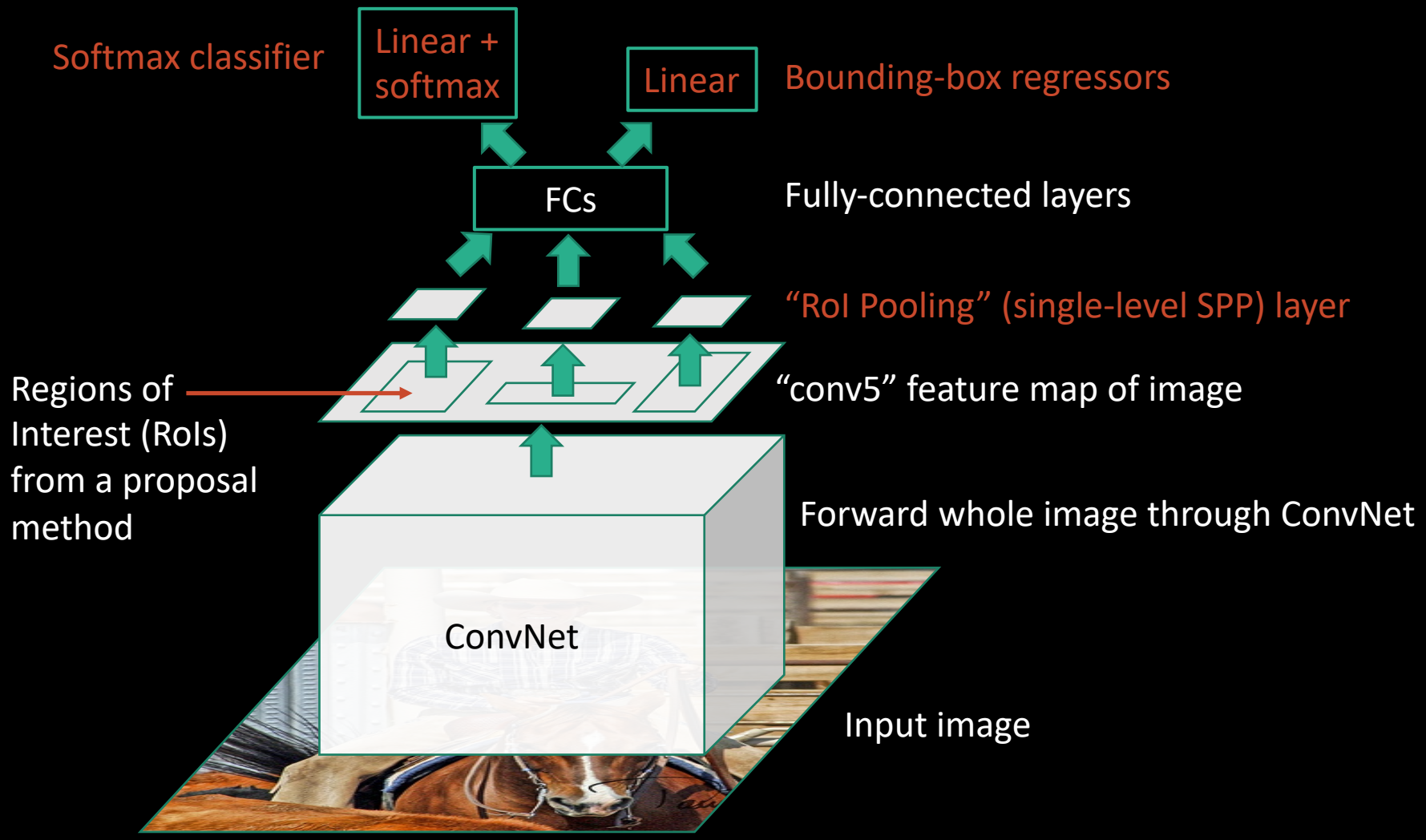
Fast R-CNN (test time)



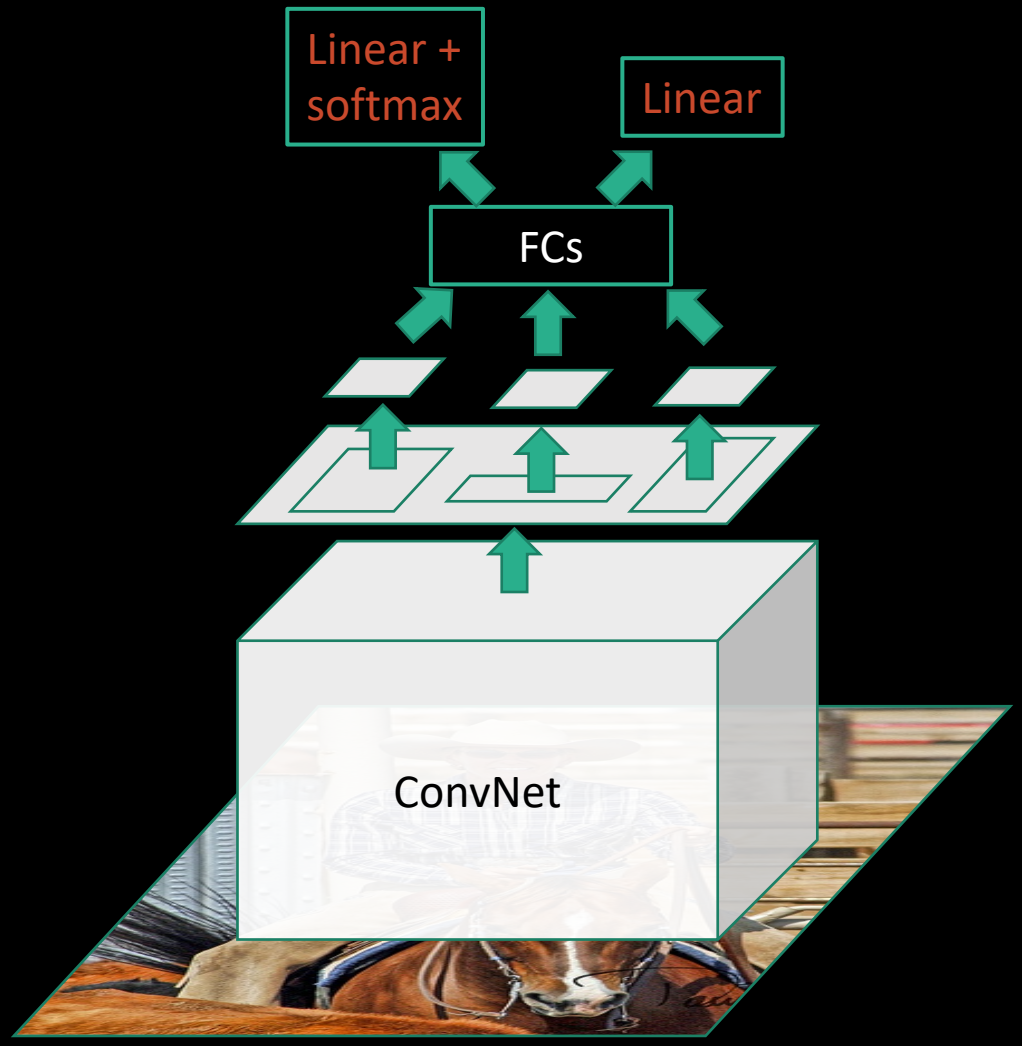
Fast R-CNN (test time)



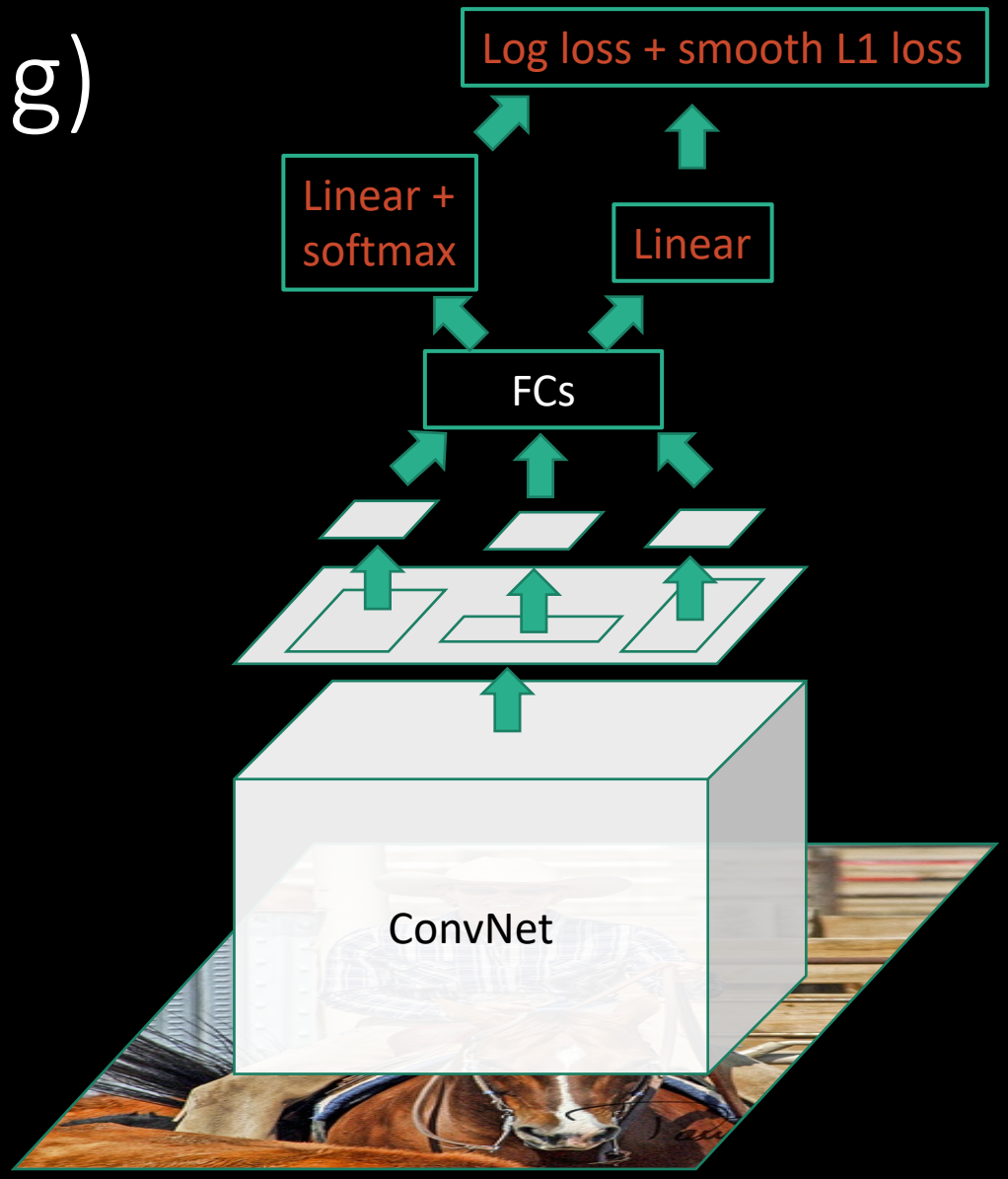
Fast R-CNN (test time)



Fast R-CNN (training)

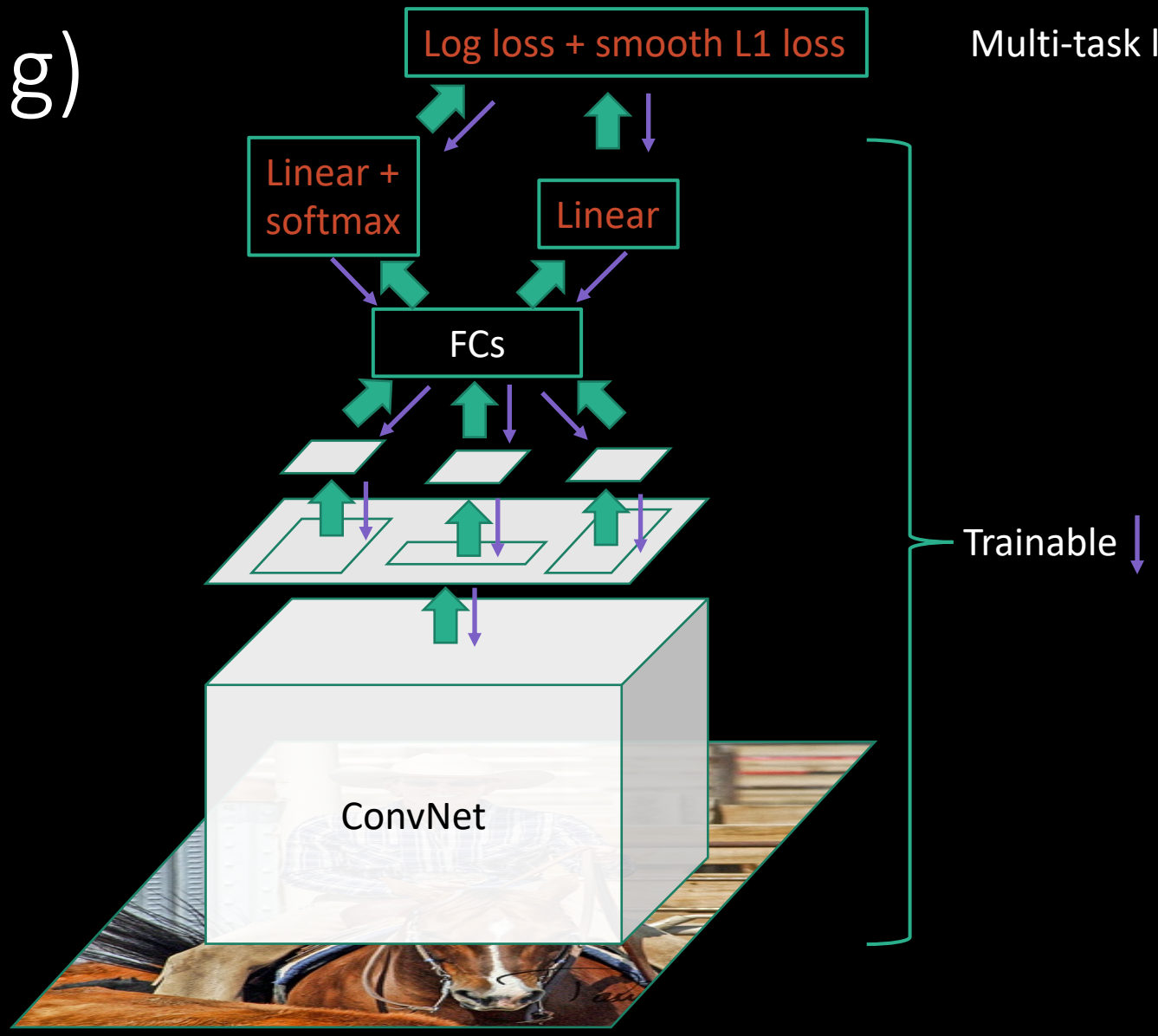


Fast R-CNN (training)



Multi-task loss

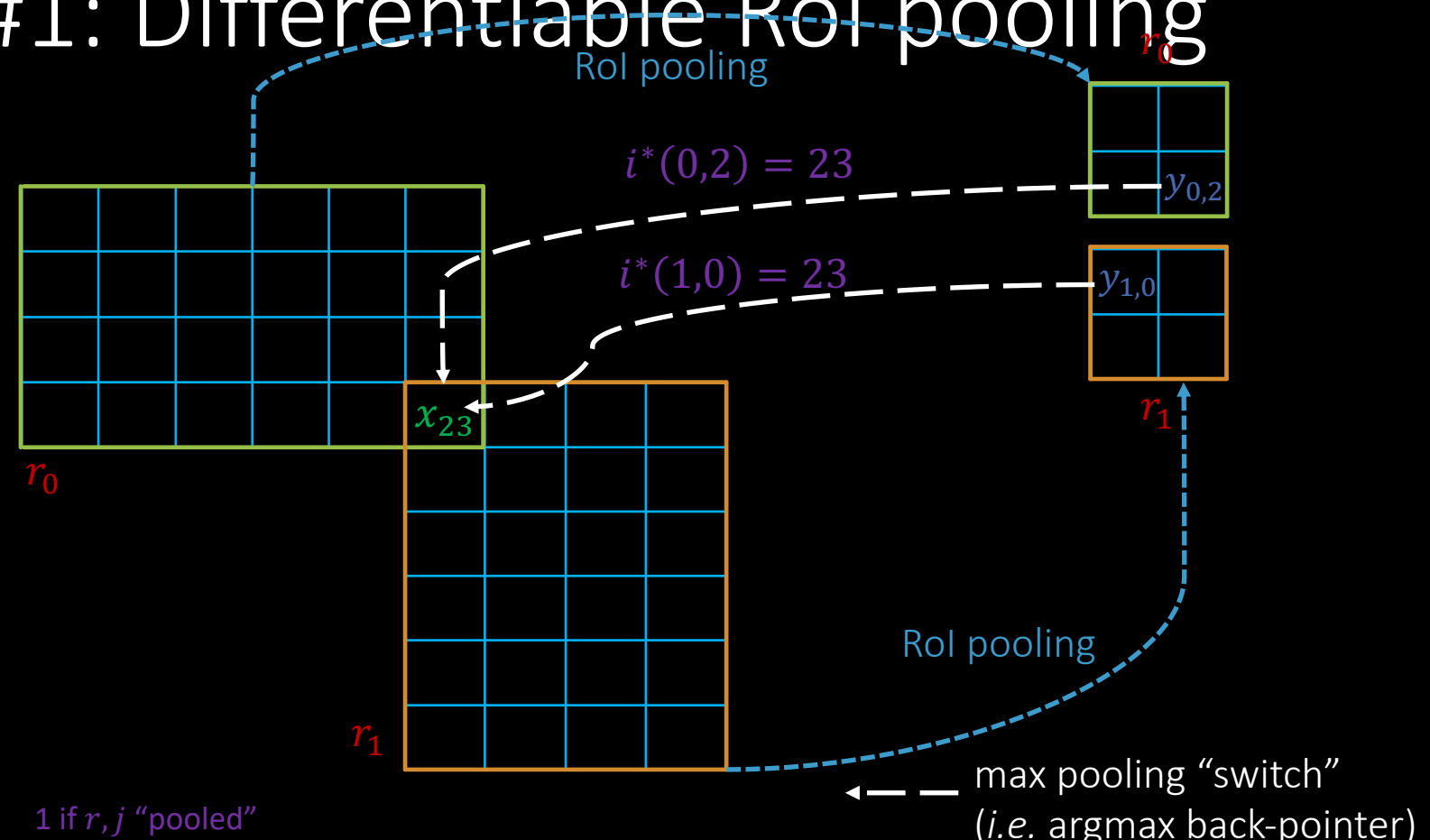
Fast R-CNN (training)



Obstacle #1: Differentiable RoI pooling

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

Obstacle #1: Differentiable RoI pooling



$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{rj}}$$

1 if r, j "pooled" input i; 0 o/w

Partial for x_i

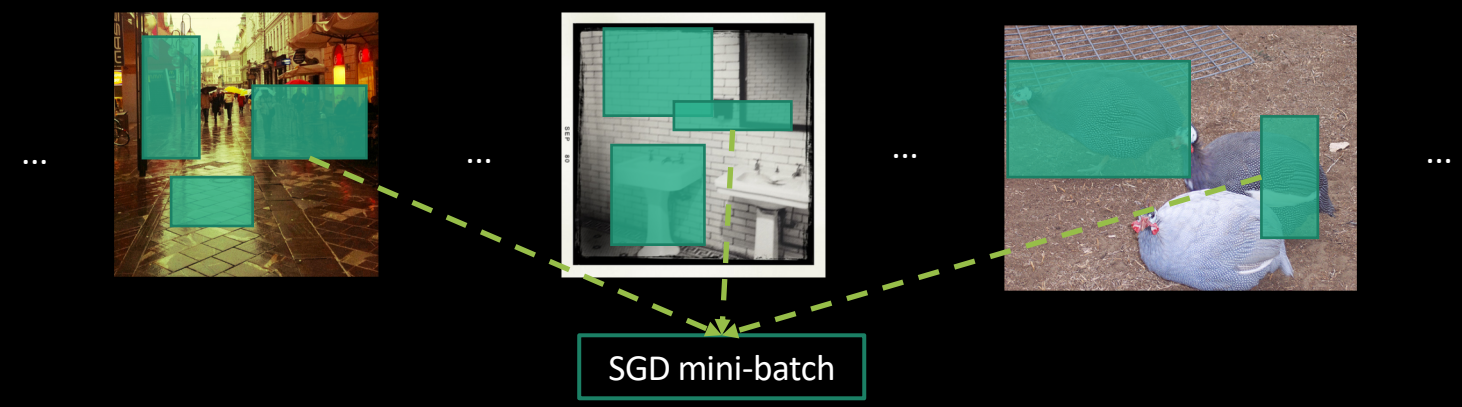
Over regions r, locations j

Partial from next layer

Obstacle #2: efficient SGD steps

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

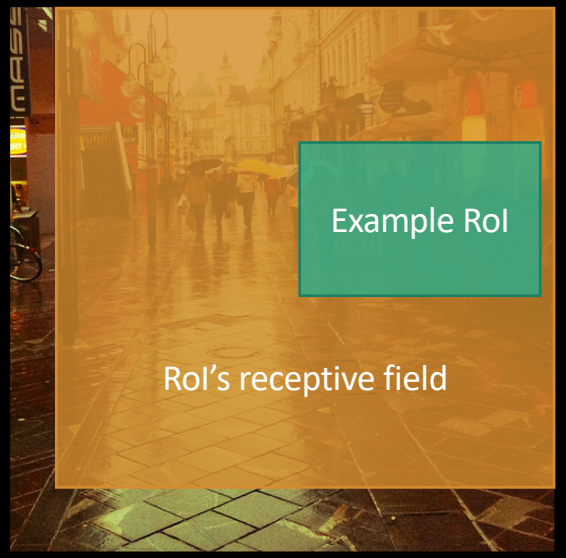
- Sample 128 example RoIs uniformly at random
- Examples will come from different images with high probability



Obstacle #2: efficient SGD steps

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

- Worst case: the receptive field is the entire image

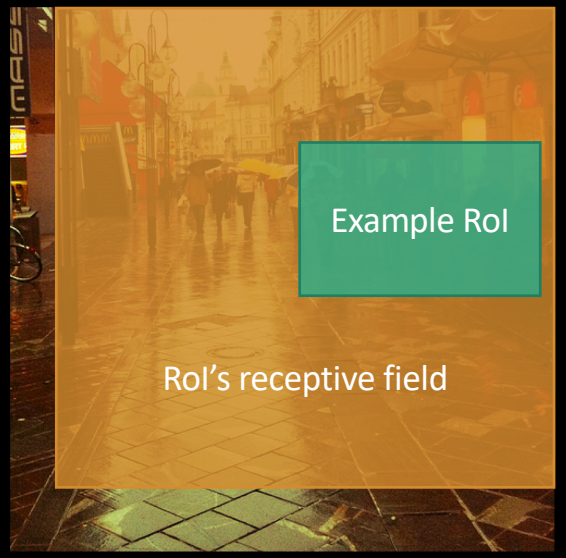


Obstacle #2: efficient SGD steps

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

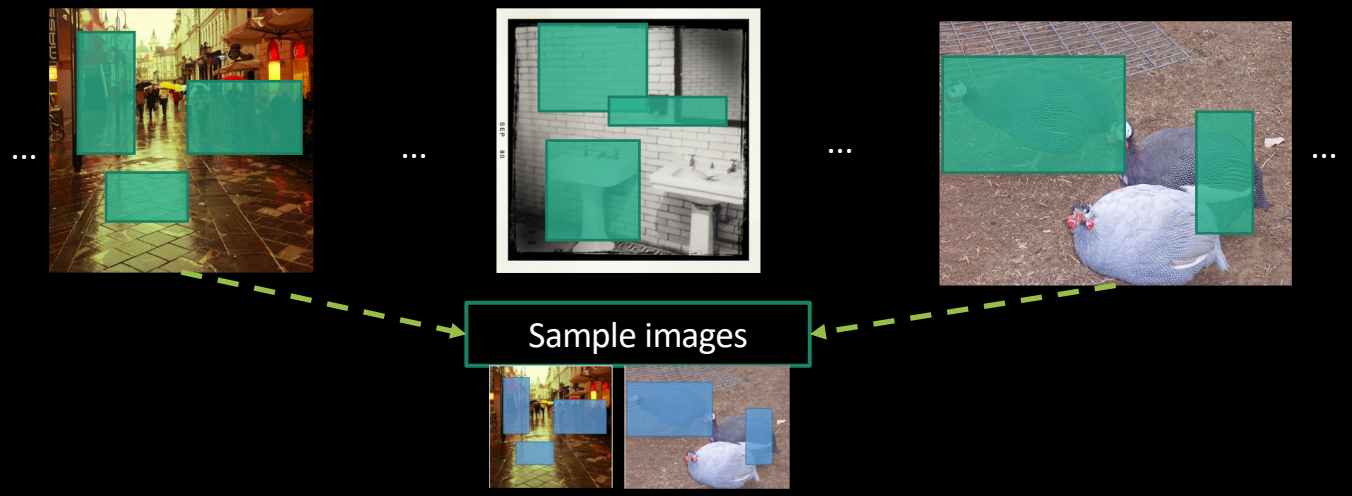


Obstacle #2: efficient SGD steps

Solution: use hierarchical sampling to build mini-batches

Obstacle #2: efficient SGD steps

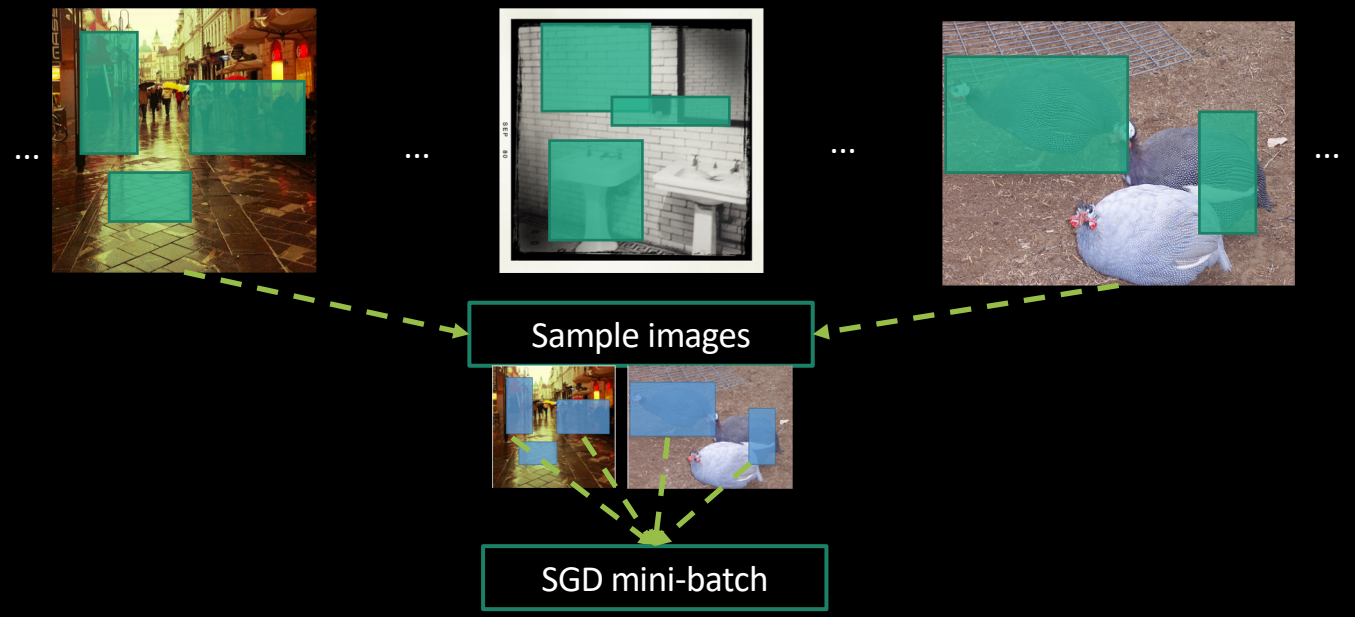
Solution: use hierarchical sampling to build mini-batches



- Sample a small number of images (2)

Obstacle #2: efficient SGD steps

Solution: use hierarchical sampling to build mini-batches

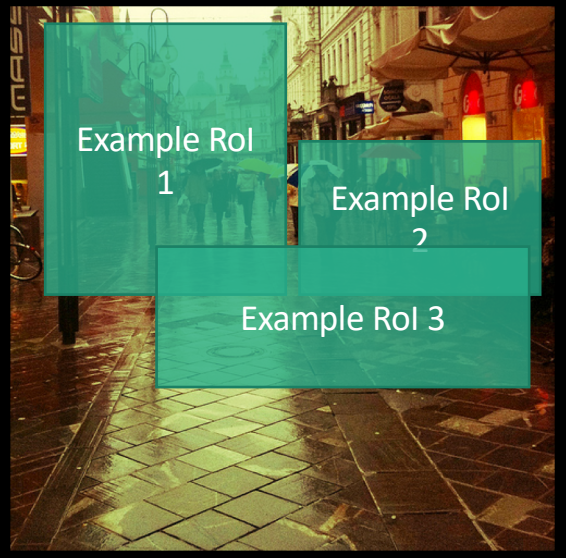


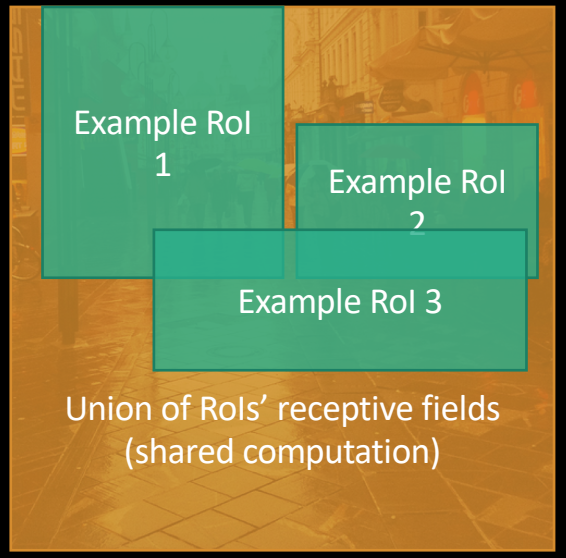
- Sample a small number of images (2)
- Sample many examples from each image (64)

Obstacle #2: efficient SGD steps

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

- Share computation between overlapping examples from the same image



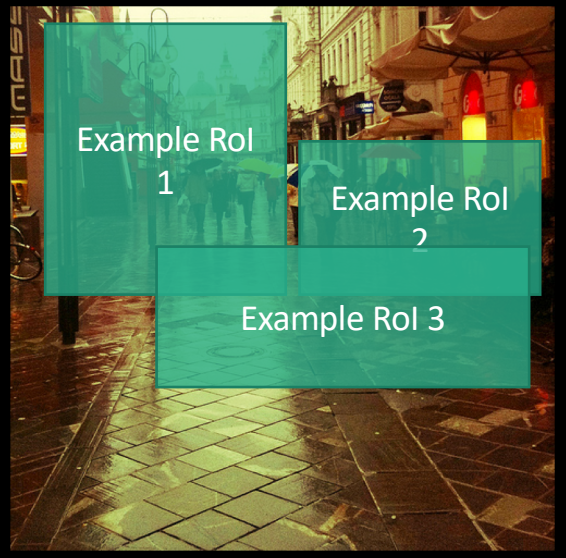


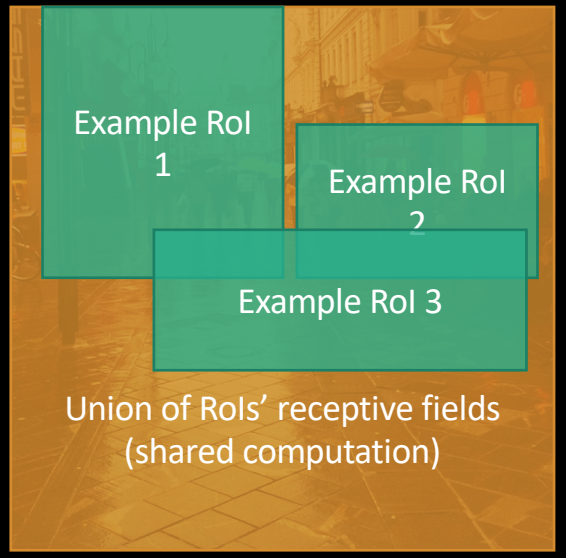
Obstacle #2: efficient SGD steps

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

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

(input size for Fast R-CNN / input size for 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.

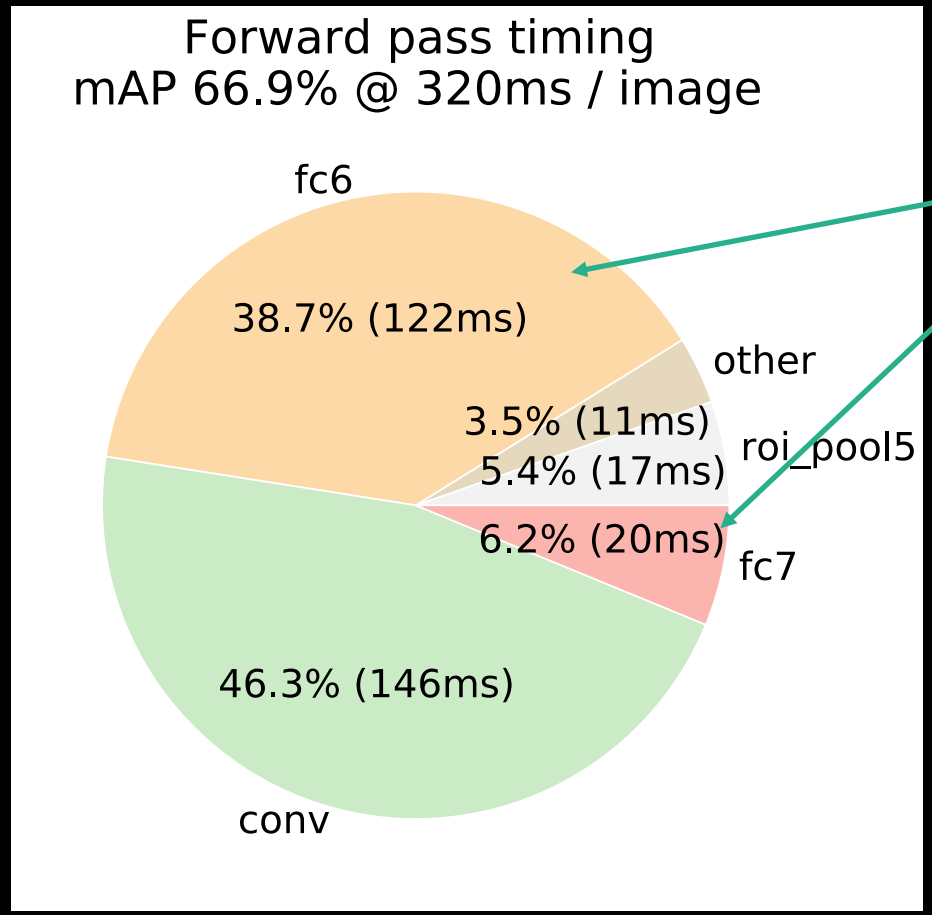
Main results

	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.

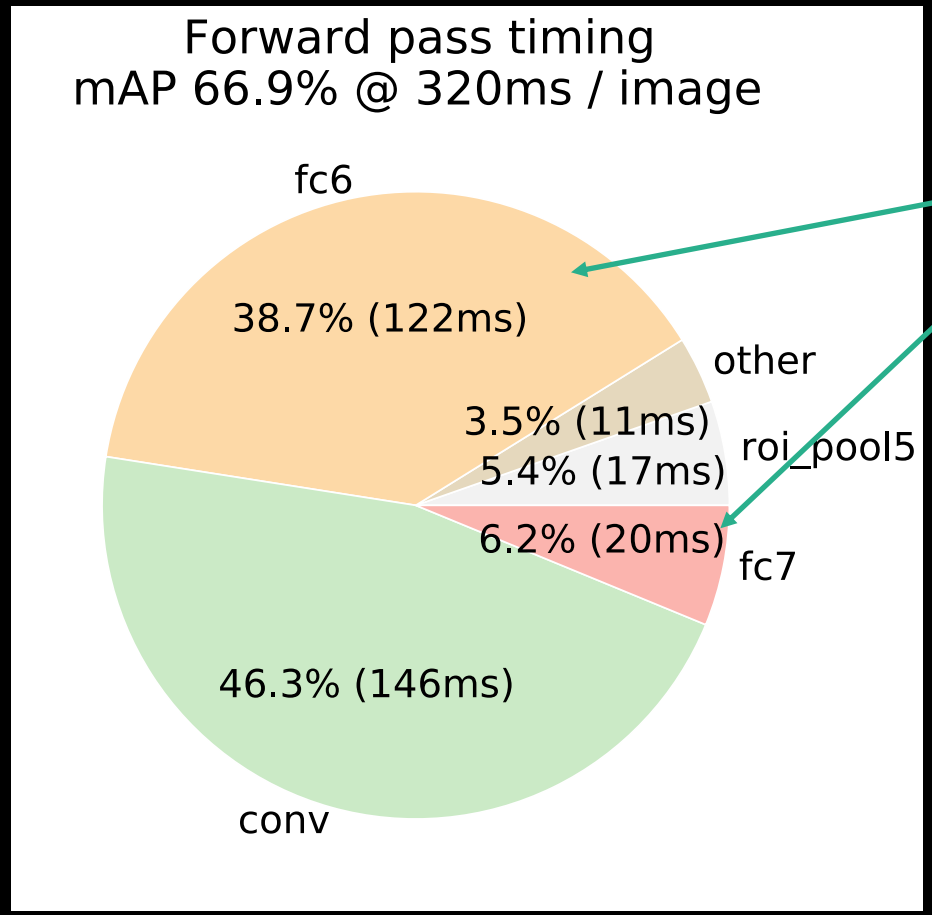
[1] Girshick et al. CVPR14.
[2] He et al. ECCV14.

Further test-time speedups



Fully connected layers take 45% of the forward pass time

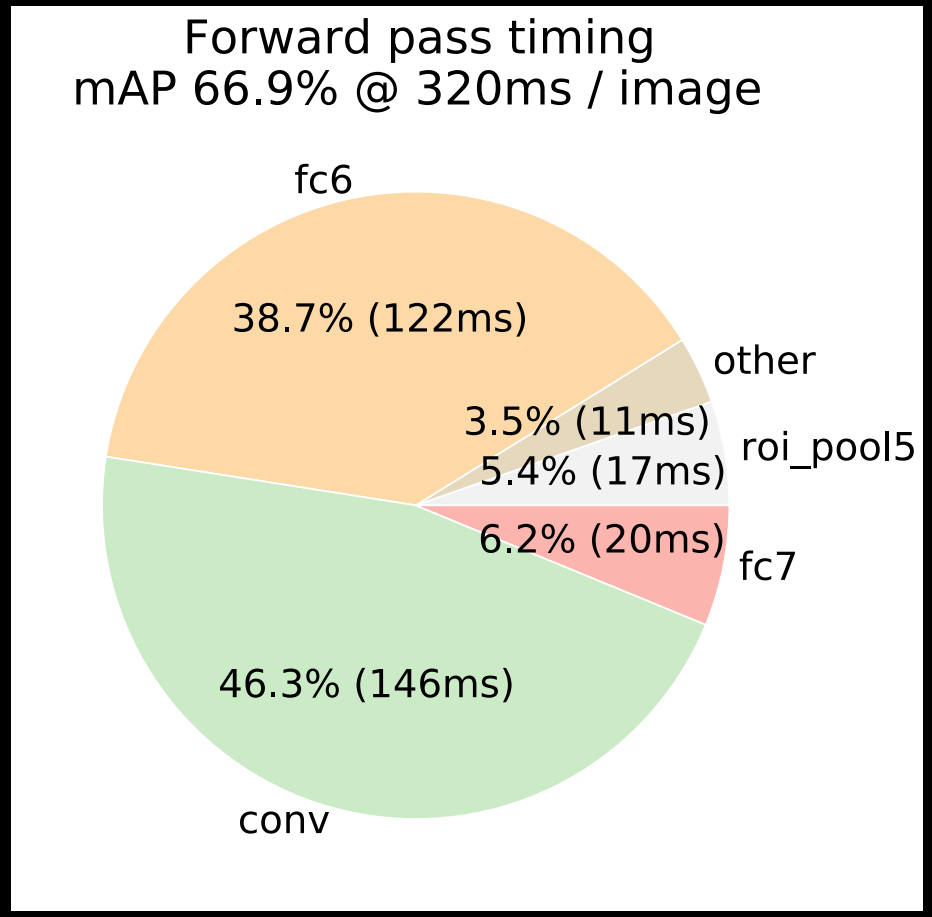
Further test-time speedups



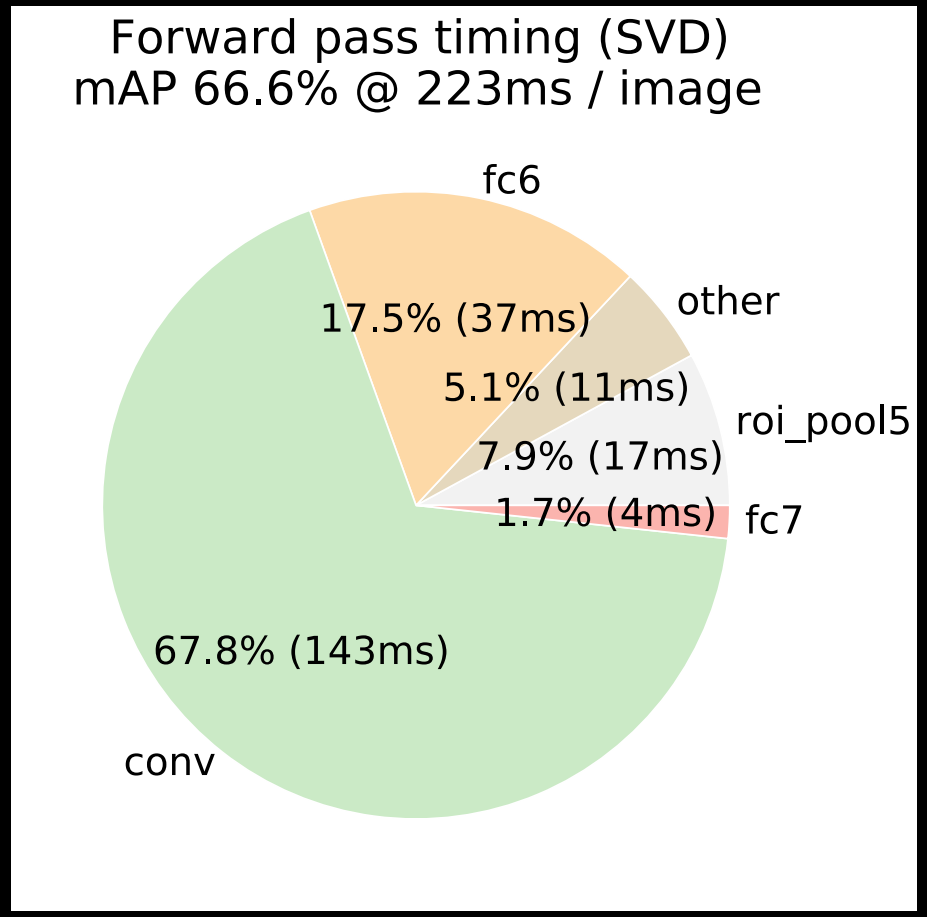
Compress these layers with truncated SVD

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

Multi-task training helps

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

Multi-task training helps

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

Trained without
a bbox regressor

Multi-task training helps

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

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

Multi-task training helps

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

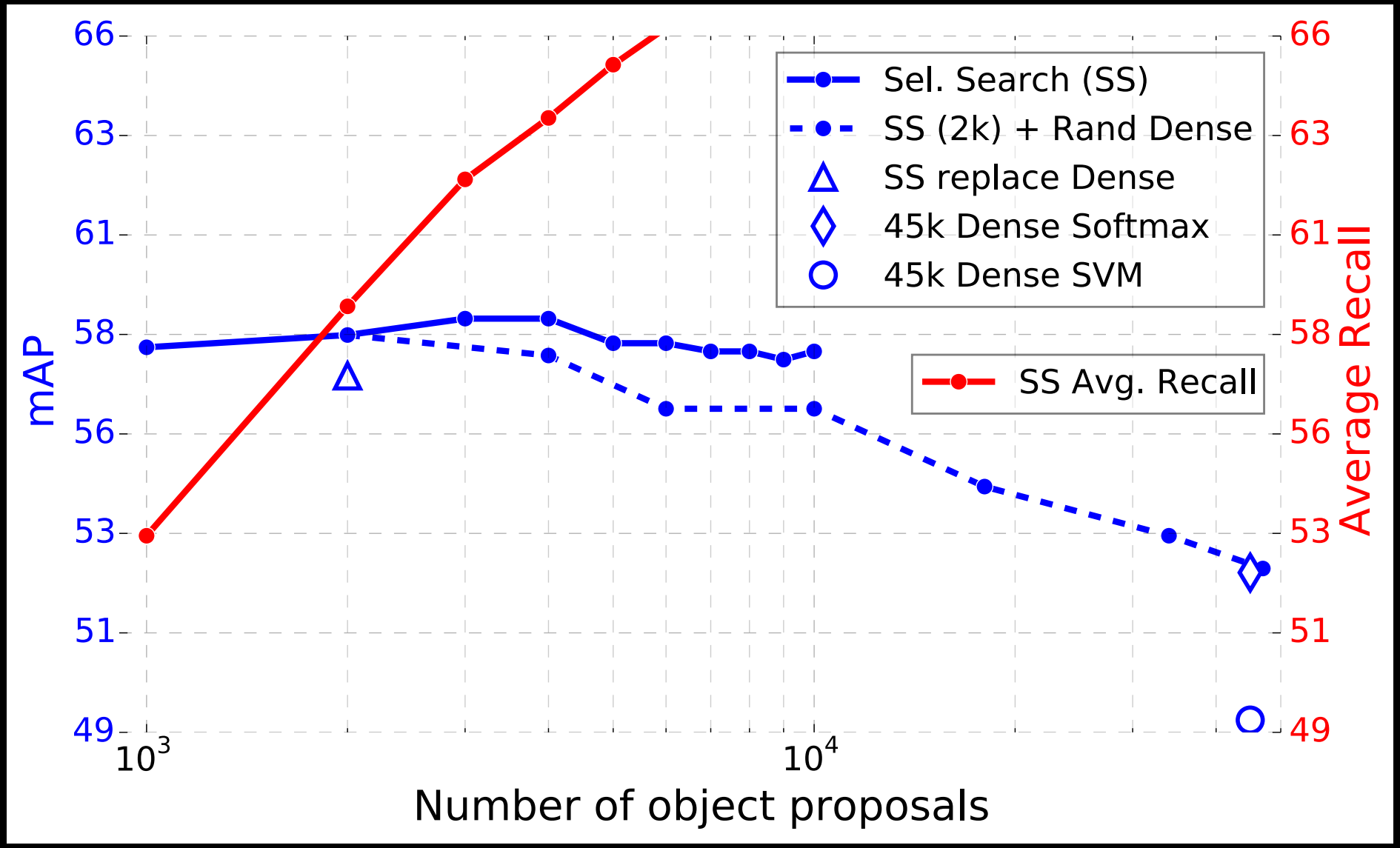
Post hoc bbox regressor, used at test time

Multi-task training helps

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

Multi-task objective, using bbox regressors at test time

More proposals is harmful



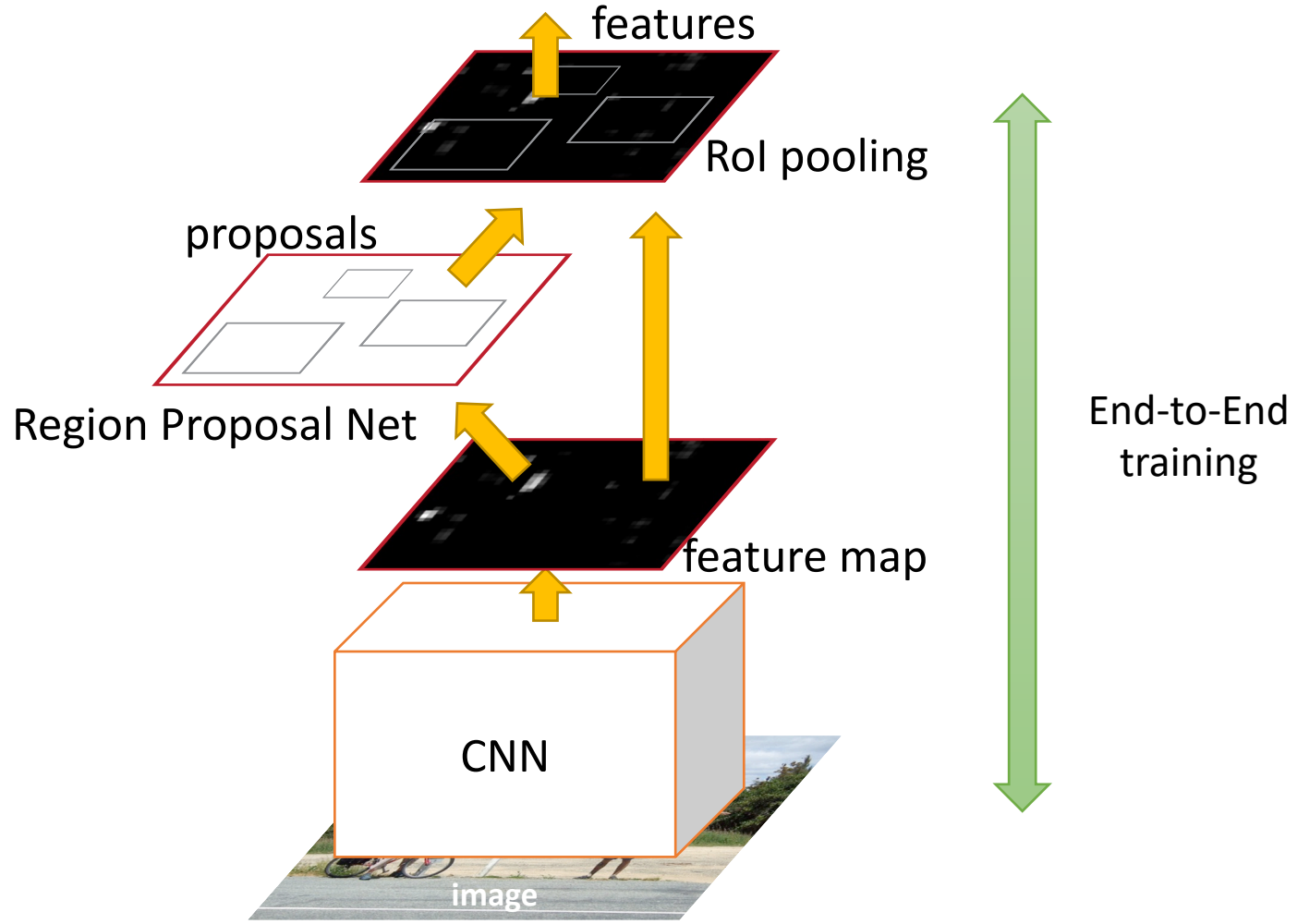
What's still wrong?

- 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

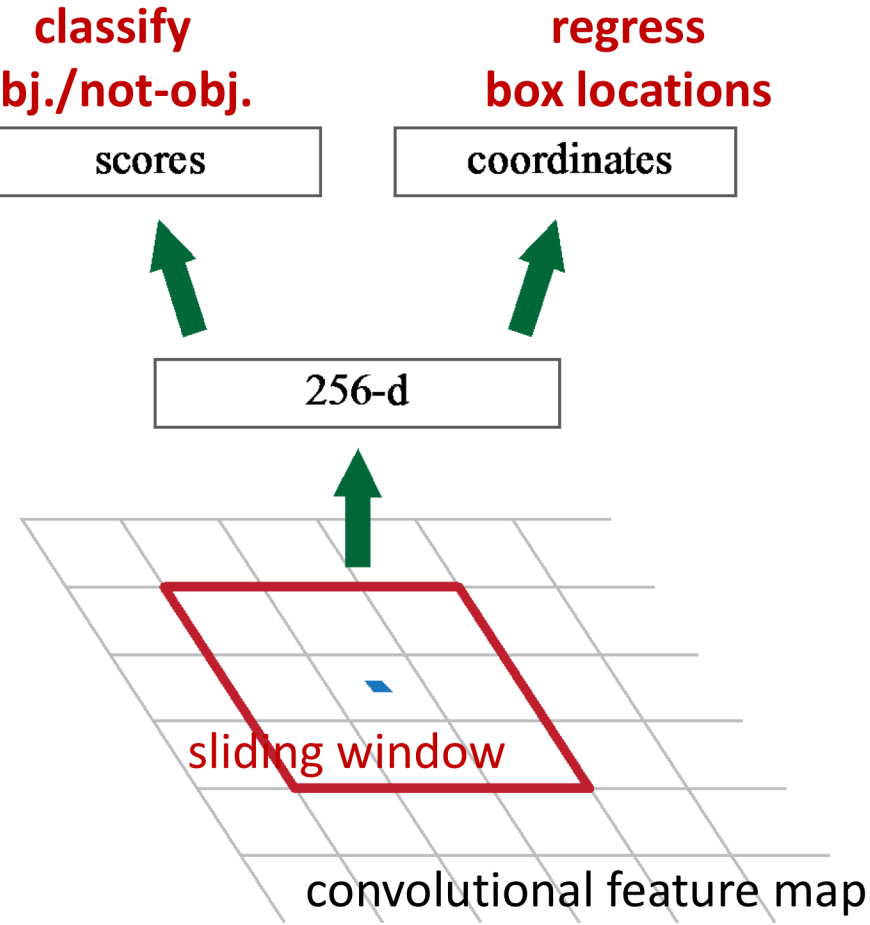


Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. NIPS 2015.

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

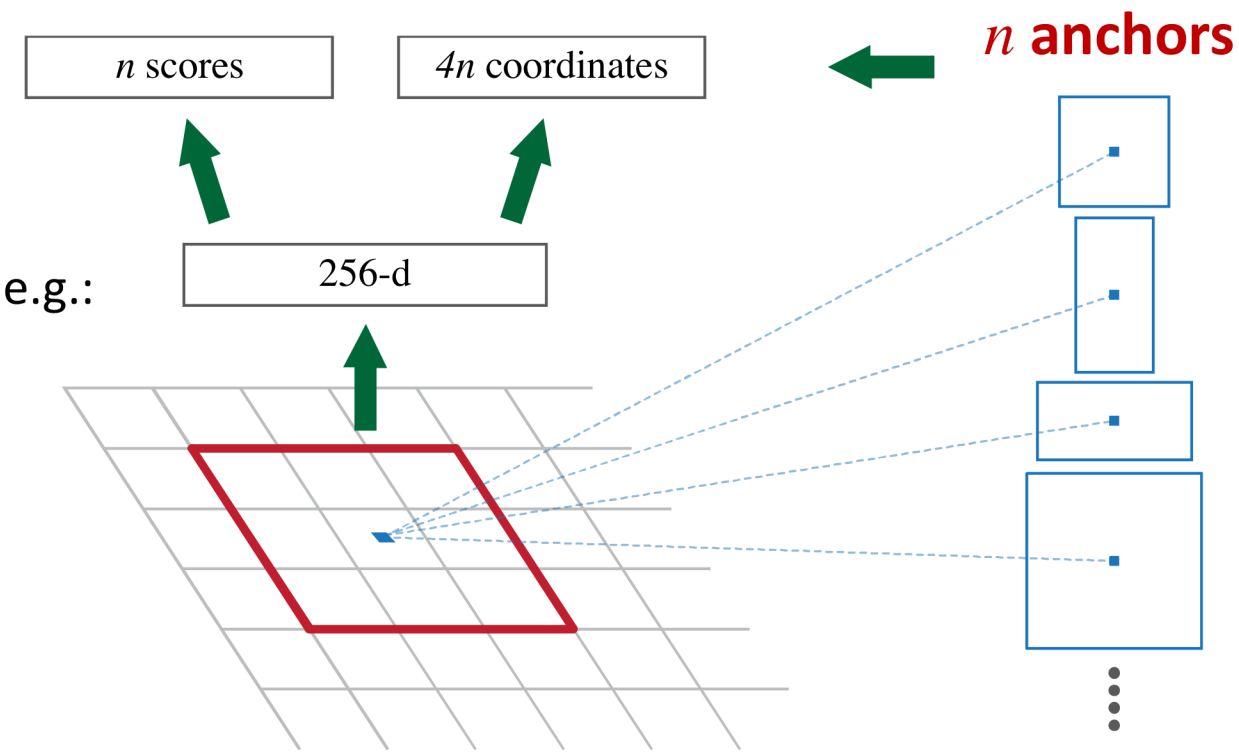
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window



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

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

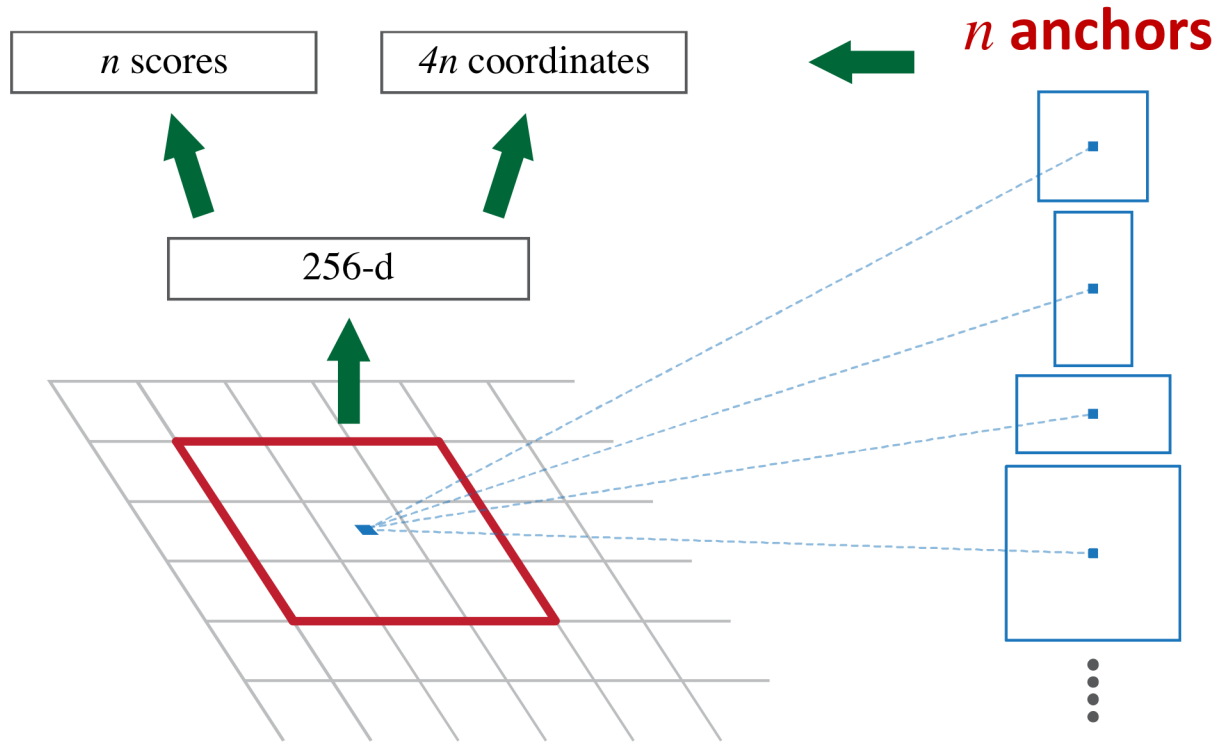


Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. NIPS 2015.

Anchors as references

- **Anchors**: pre-defined reference boxes

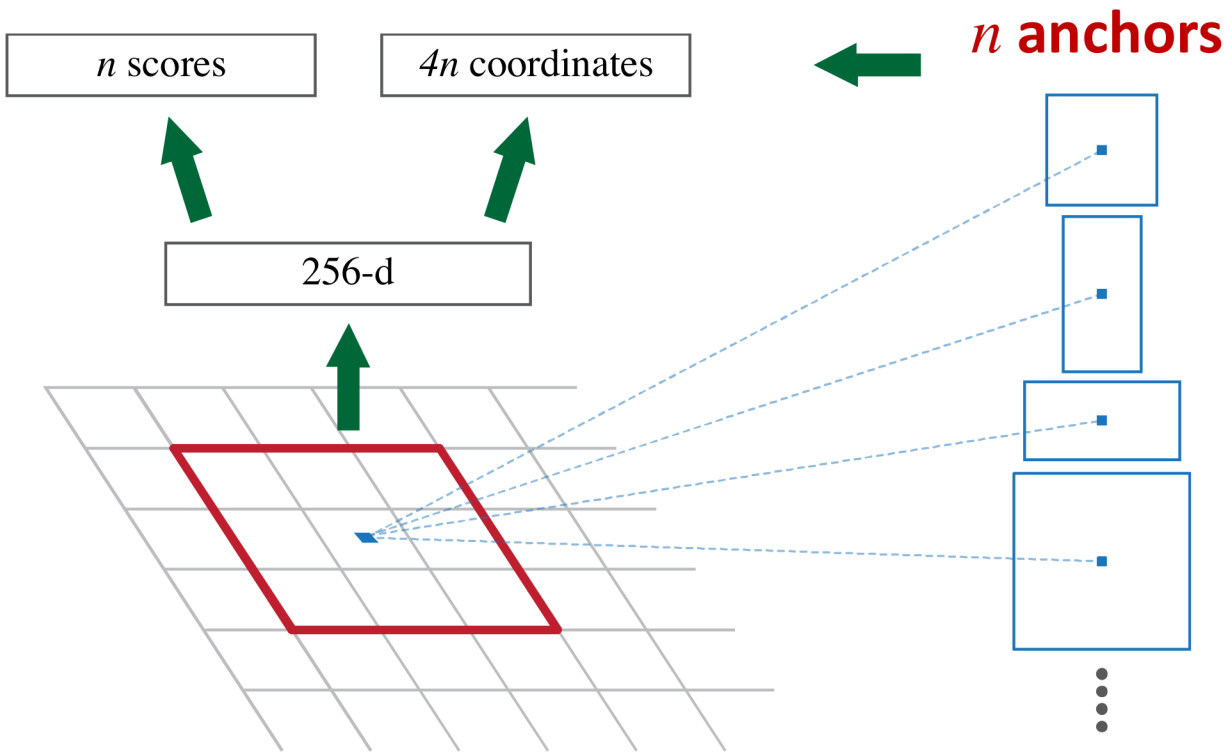
- Multi-scale/size anchors:
  - multiple anchors are used at each position:
    e.g., 3 scales (128^2, 256^2, 512^2) and 3 aspect ratios (2:1, 1:1, 1:2) yield 9 anchors
  - each anchor has its own prediction function
  - single-scale features, multi-scale predictions



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

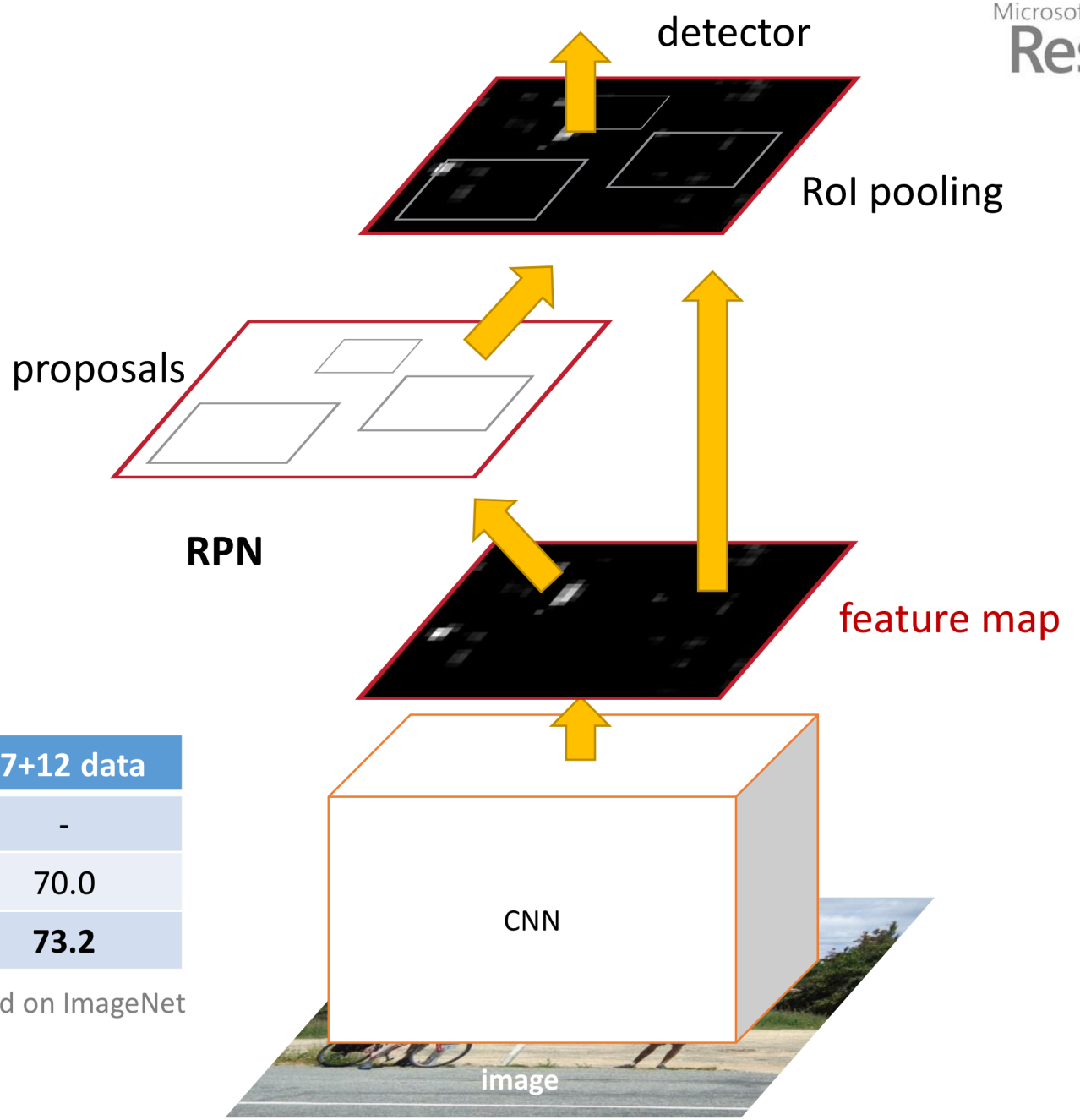
Region Proposal Network

- 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)



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

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

ICCV15 International Conference on Computer Vision

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

Fast R-CNN take-aways

- End-to-end training of deep ConvNets for detection
- Fast training times
- Open source for easy experimentation
  "I think [the Fast R-CNN] code is average-somewhat above average for what it is."
  – sporkles on r/MachineLearning
- A large number of ImageNet detection and COCO detection methods are built on Fast R-CNN
  Checkout the ImageNet / COCO Challenge workshop on Thursday!

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

Viola and Jones (2001)

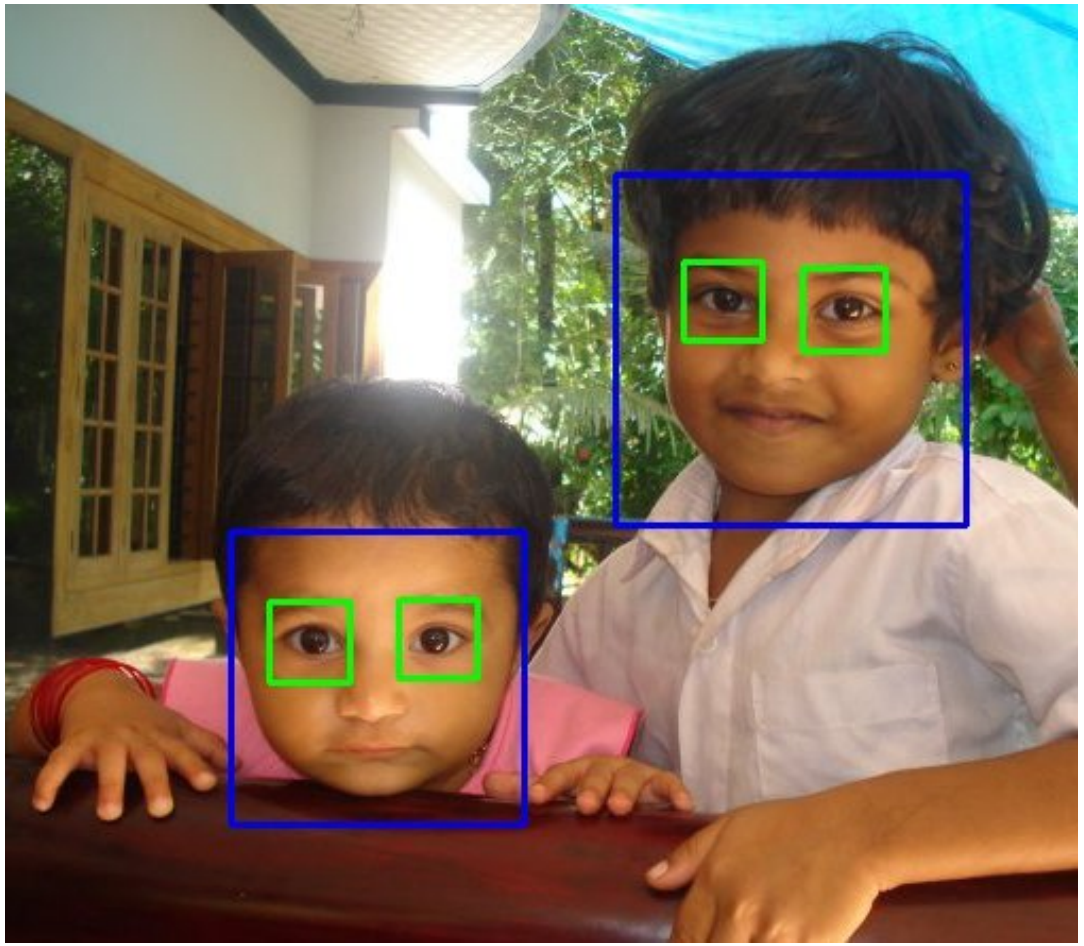
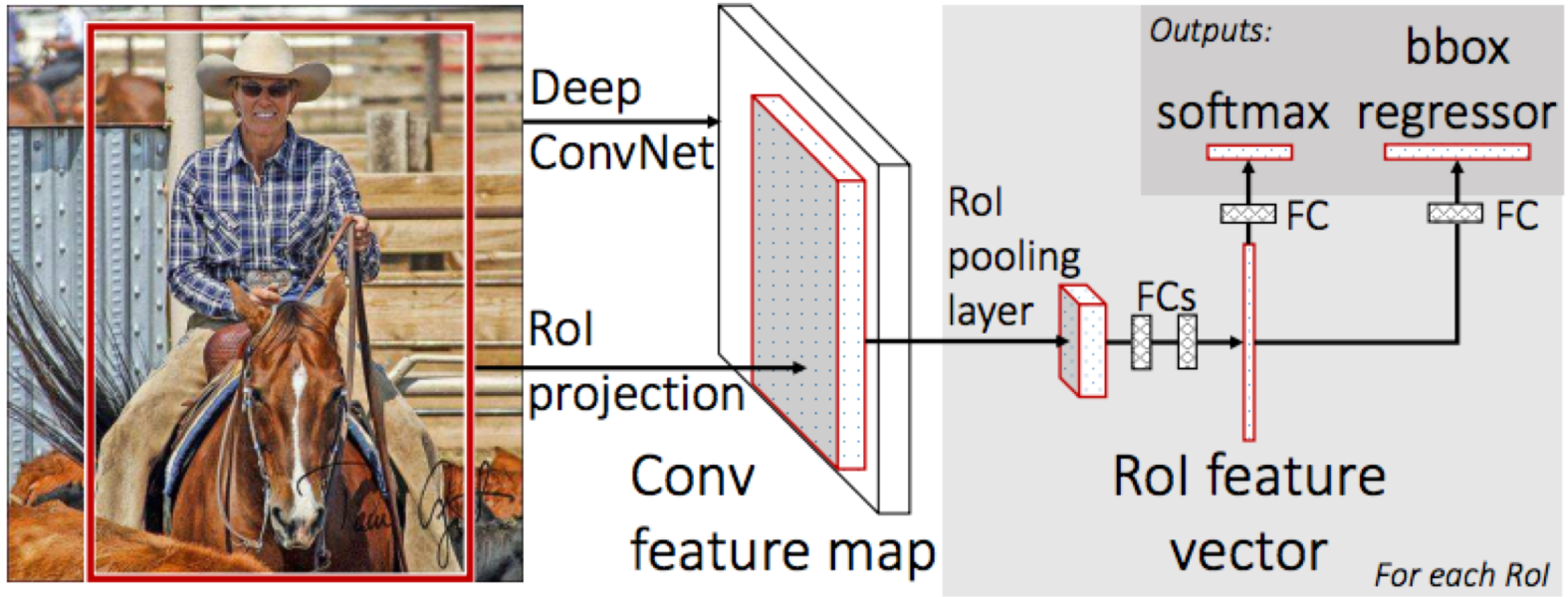


Image from OpenCV 3.3 website

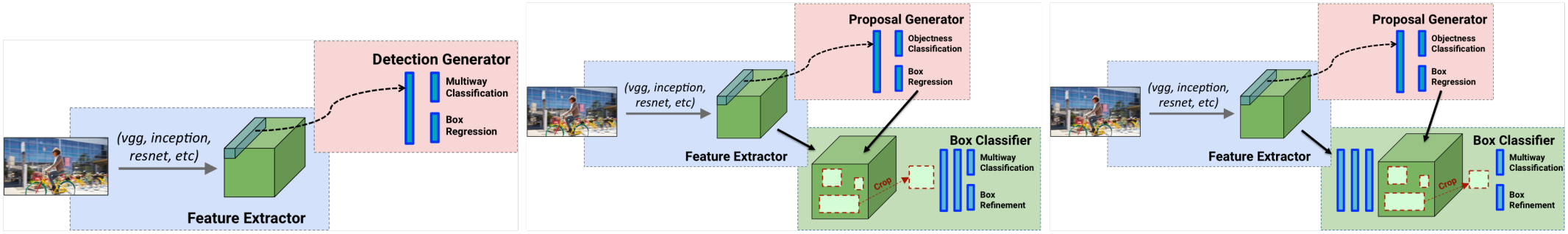
Fast R-CNN



Fast R-CNN, Girshick

One-stage vs. Two-stage

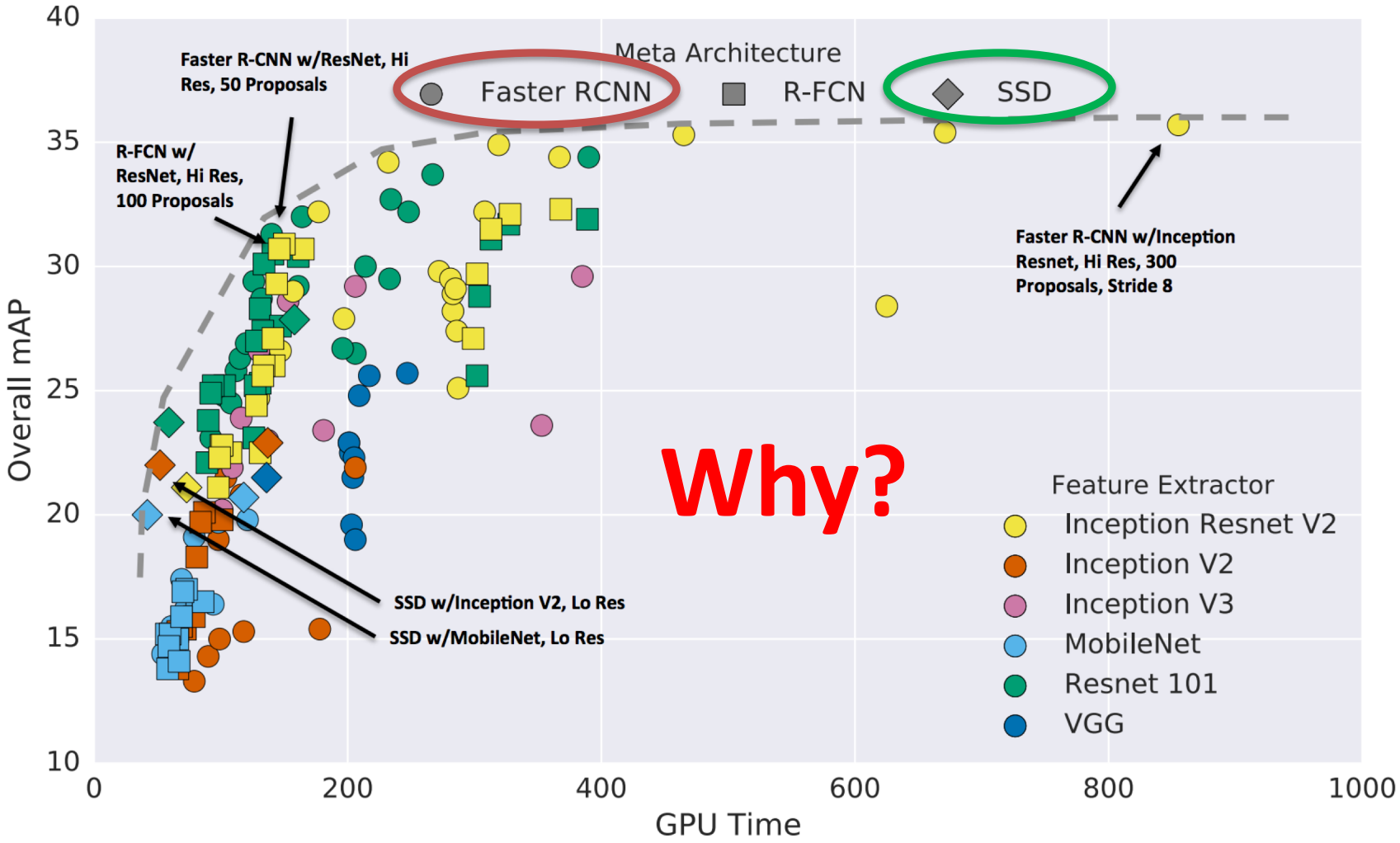
- One-stage
  - Fast
  - Simple
- Two-stage
  - 10 - 40% better accuracy



(a) SSD. (b) Faster RCNN. (c) R-FCN.

https://arxiv.org/pdf/1611.10012.pdf

One-stage vs. Two-stage



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

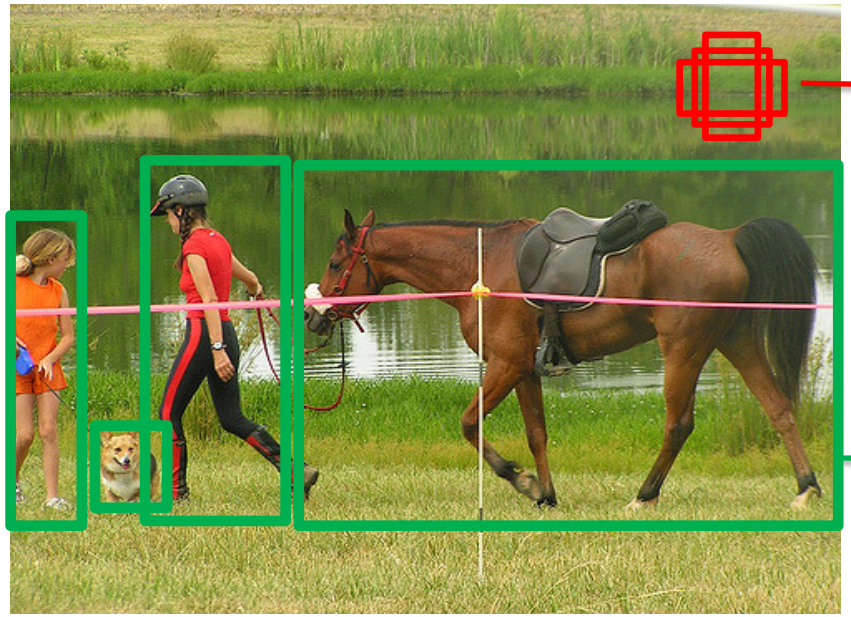
Toward dense detection

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

- This work – ~100k

Class Imbalance

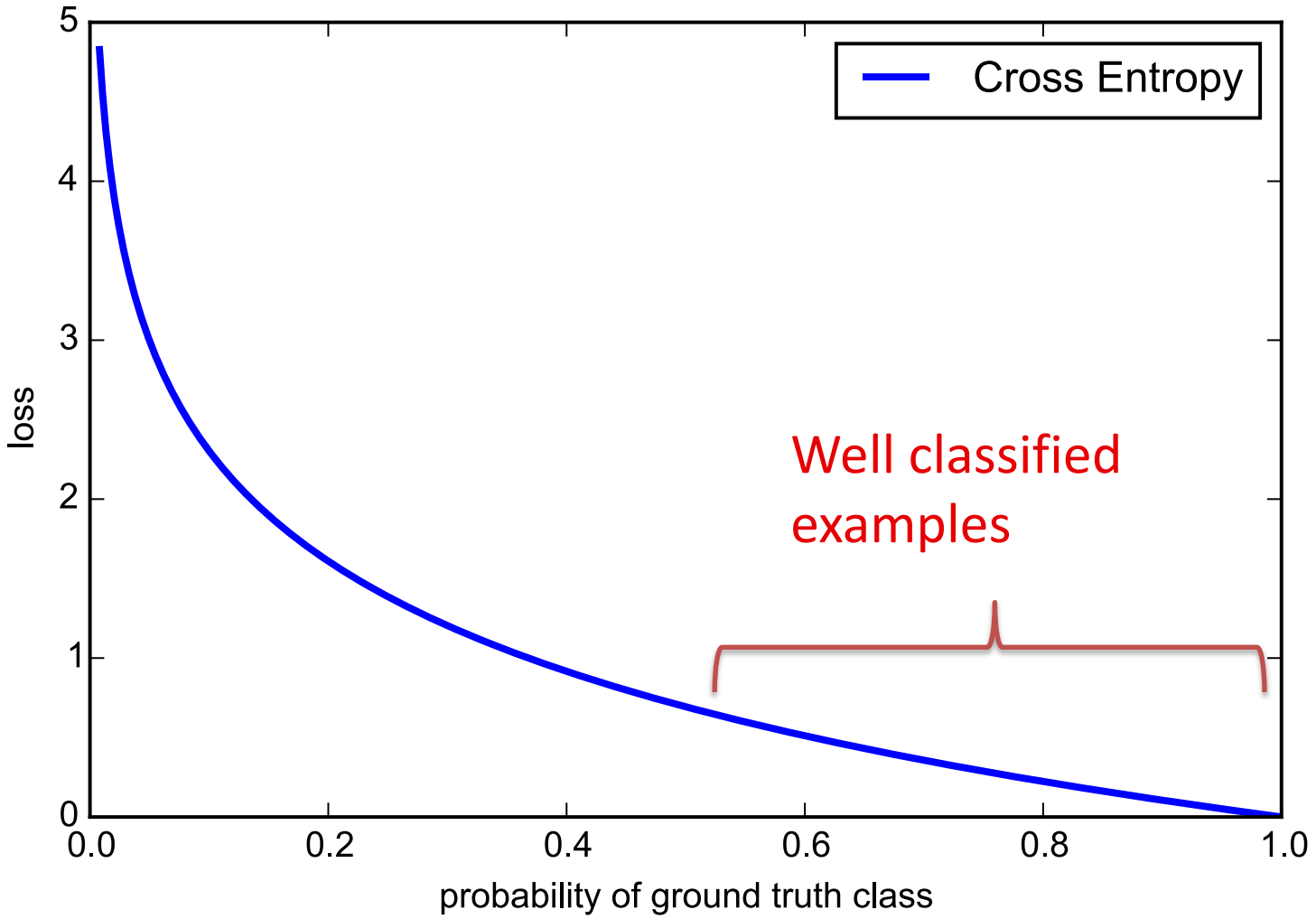
- Few training examples from foreground
- Most examples from background
  - Easy and uninformative
  - Distracting



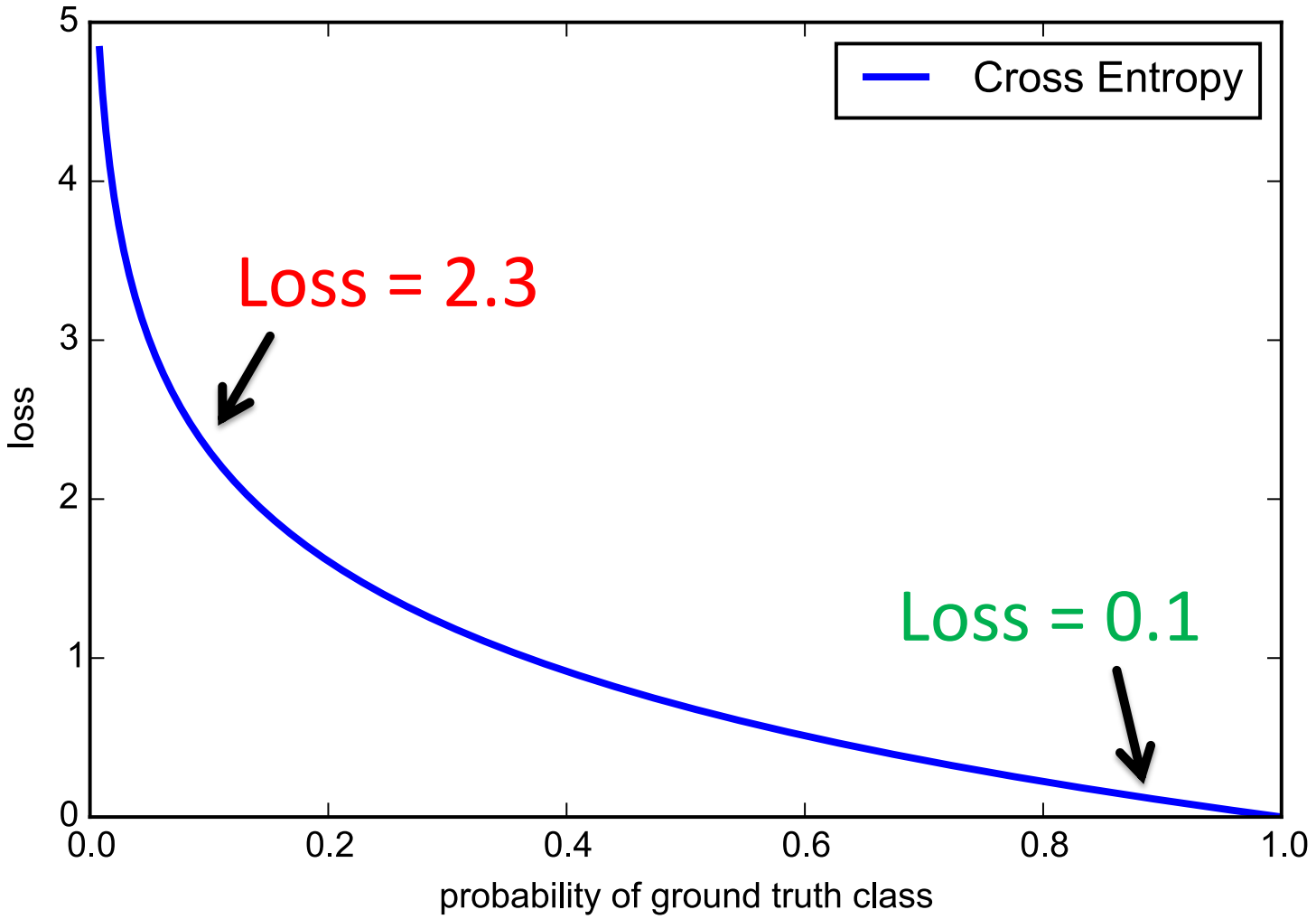
Many negative examples, no useful signal

Few positive examples, rich information

Cross Entropy

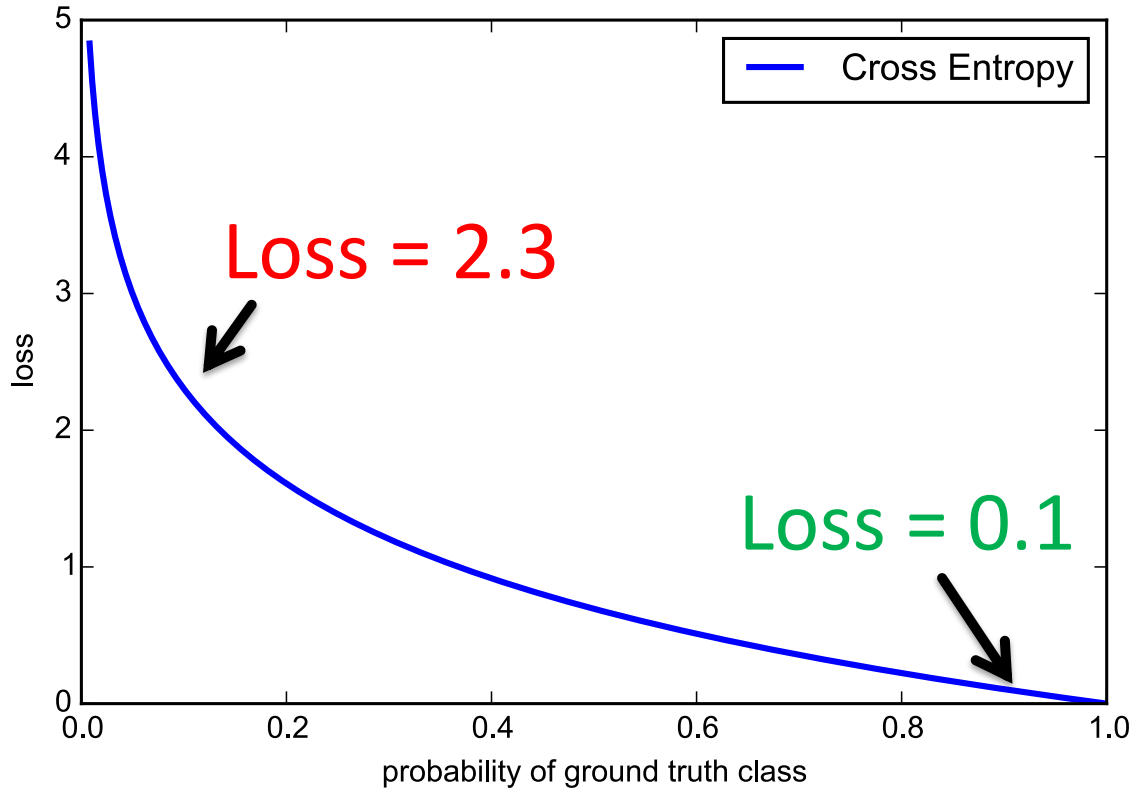


Cross Entropy



Cross Entropy with Imbalance Data

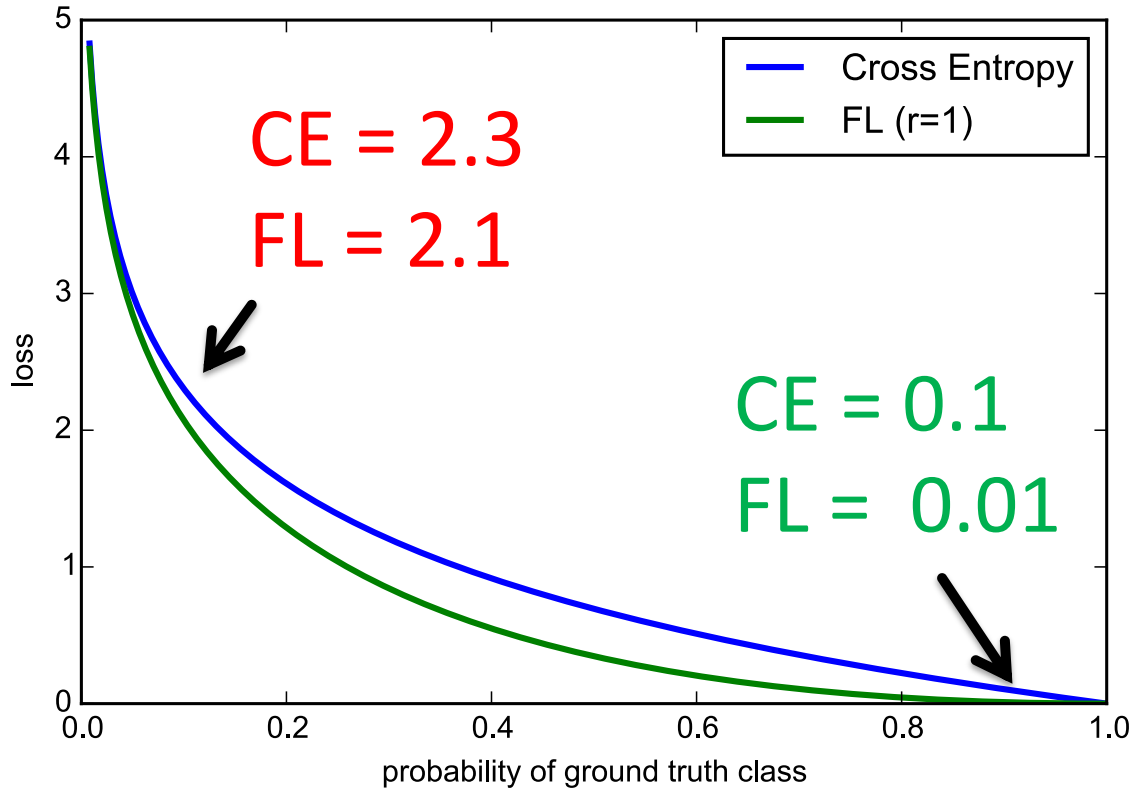
- 100000 easy : 100 hard examples
- **40x bigger loss** from easy examples



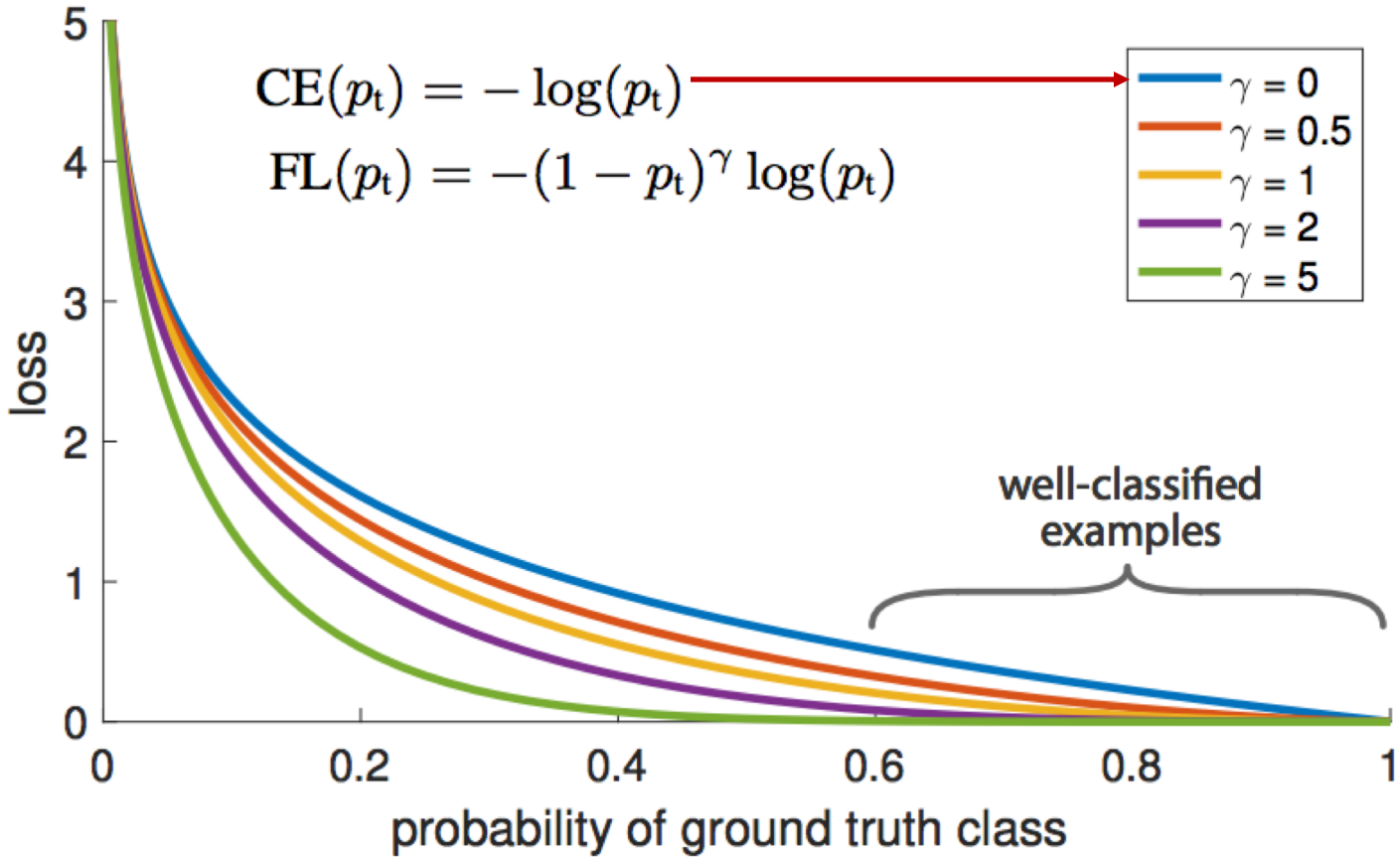
Focal Loss

$$\mathrm{CE}(p_t) = -\log(p_t)$$

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



Focal Loss


$\mathrm{CE}(p_t) = -\log(p_t)$
$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t)$
$\gamma = 0$
$\gamma = 0.5$
$\gamma = 1$
$\gamma = 2$
$\gamma = 5$
well-classified examples
loss
probability of ground truth class
0
1
2
3
4
5
0
0.2
0.4
0.6
0.8
1

Prior

- α-balanced Cross entropy

$$\mathrm{CE}(p_\mathrm{t}) = -\alpha_t \log(p_\mathrm{t})$$

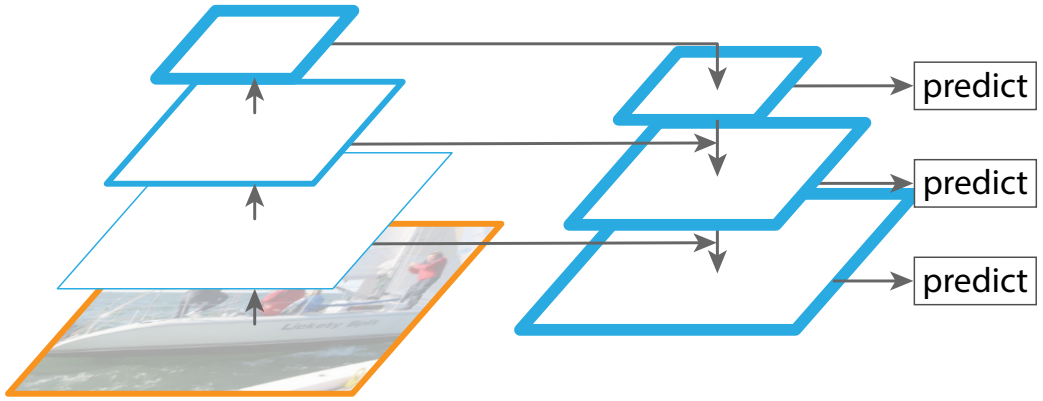
- α-balanced Focal Loss

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

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

Feature Pyramid Network

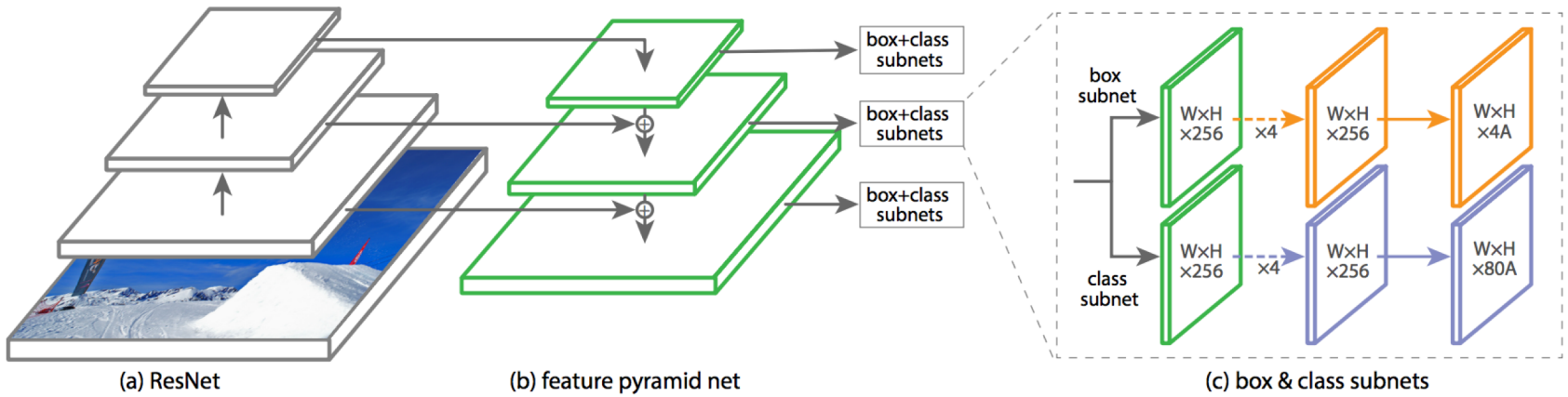
- Multiscale
- Semantically strong at all scales
- Fast to compute



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

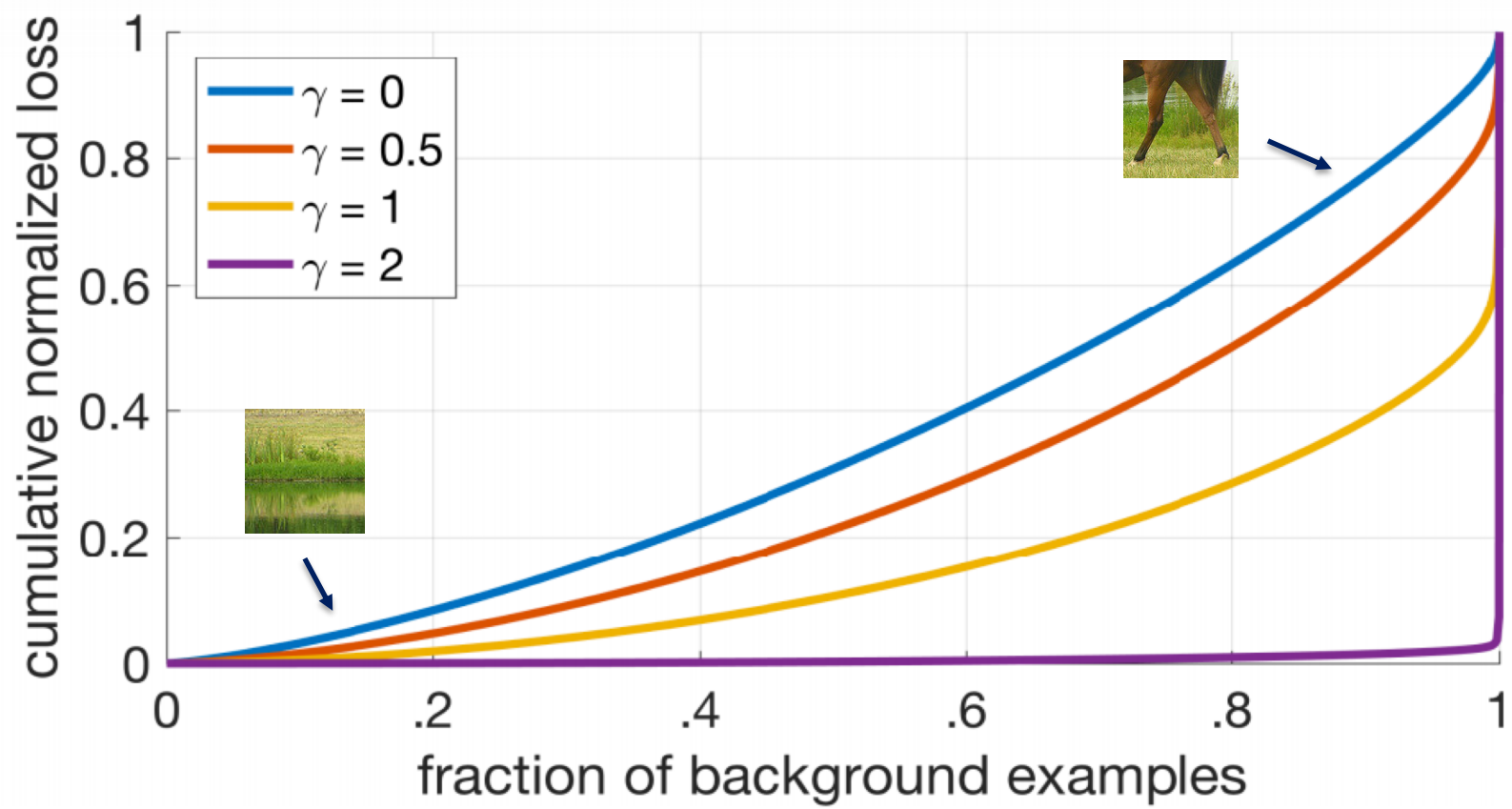
Architecture

- RetinaNet
  - FPN + 100k boxes
  - Focal loss

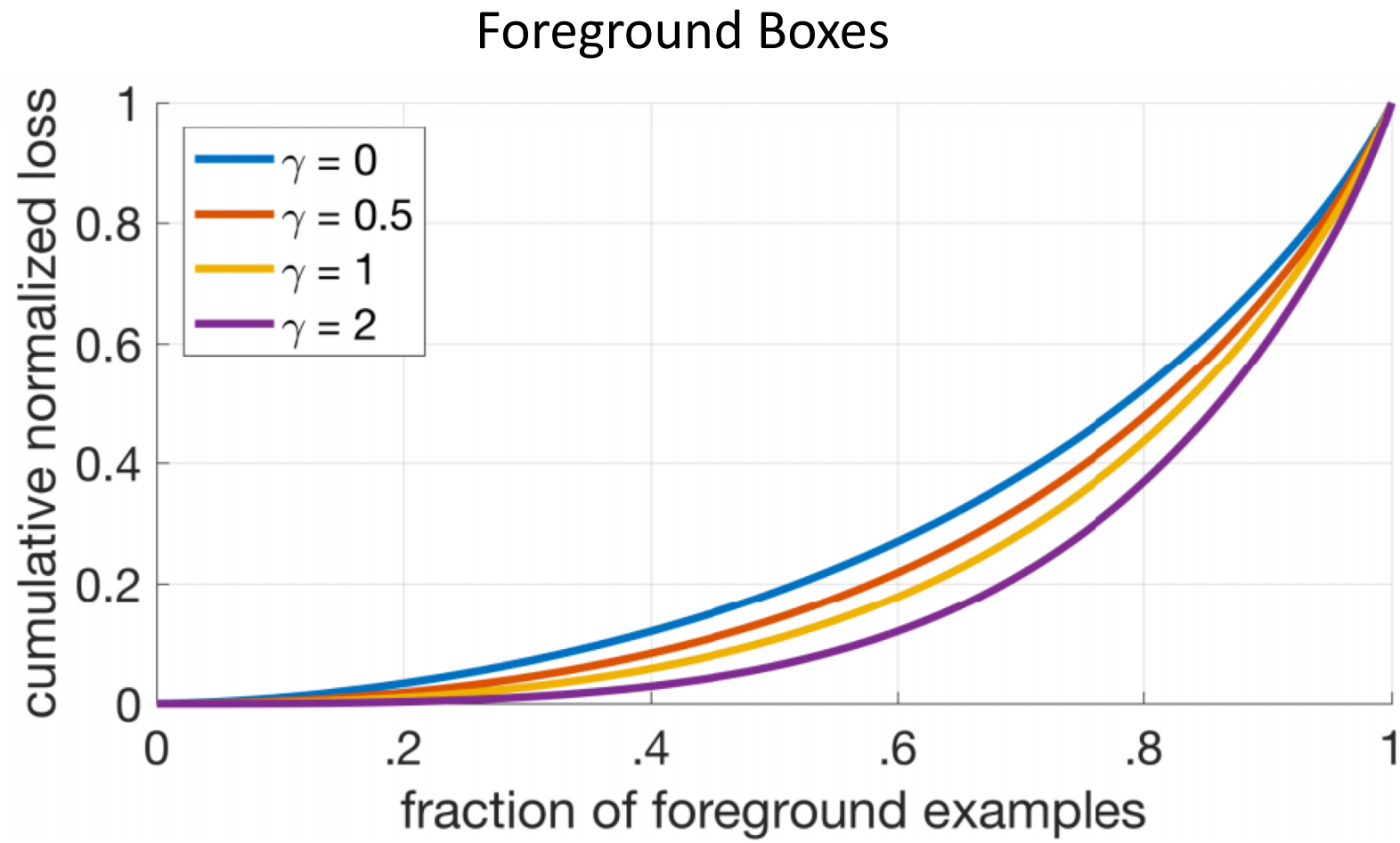


Loss Distribution under Focal Loss

Background Boxes



Loss Distribution under Focal Loss



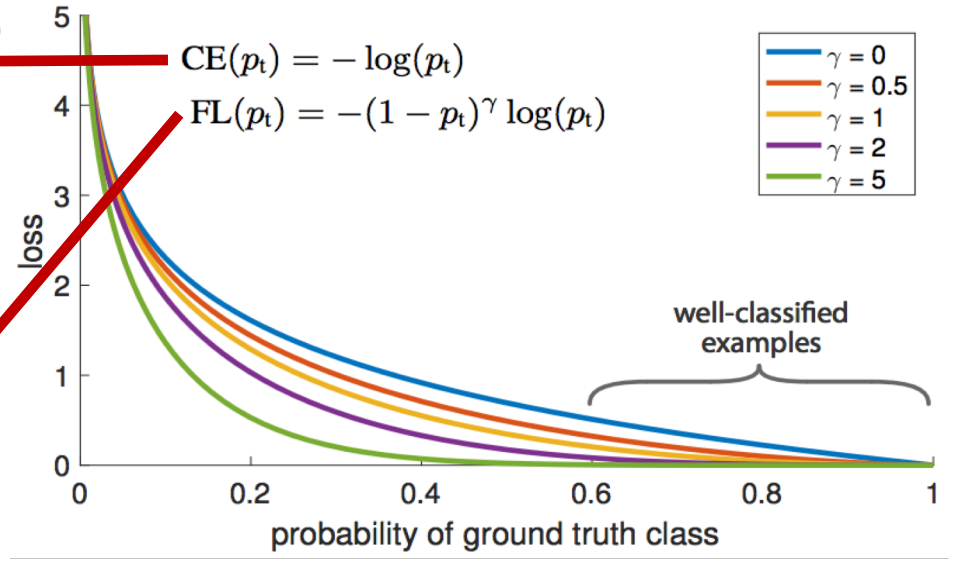
vs. Cross Entropy

- + 2.9 AP to α-balanced cross entropy

γ	α	AP	AP_{50}	AP_{75}
0	.75	31.1	49.4	33.0
0.1	.75	31.4	49.9	33.1
0.2	.75	31.9	50.7	33.4
0.5	.50	32.9	51.7	35.2
1.0	.25	33.7	52.0	36.2
2.0	.25	**34.0**	**52.5**	**36.5**
5.0	.25	32.2	49.6	34.8

(b) **Varying γ for FL** (w. optimal α)

(ResNet-50-FPN 600px input image)



vs. OHEM

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

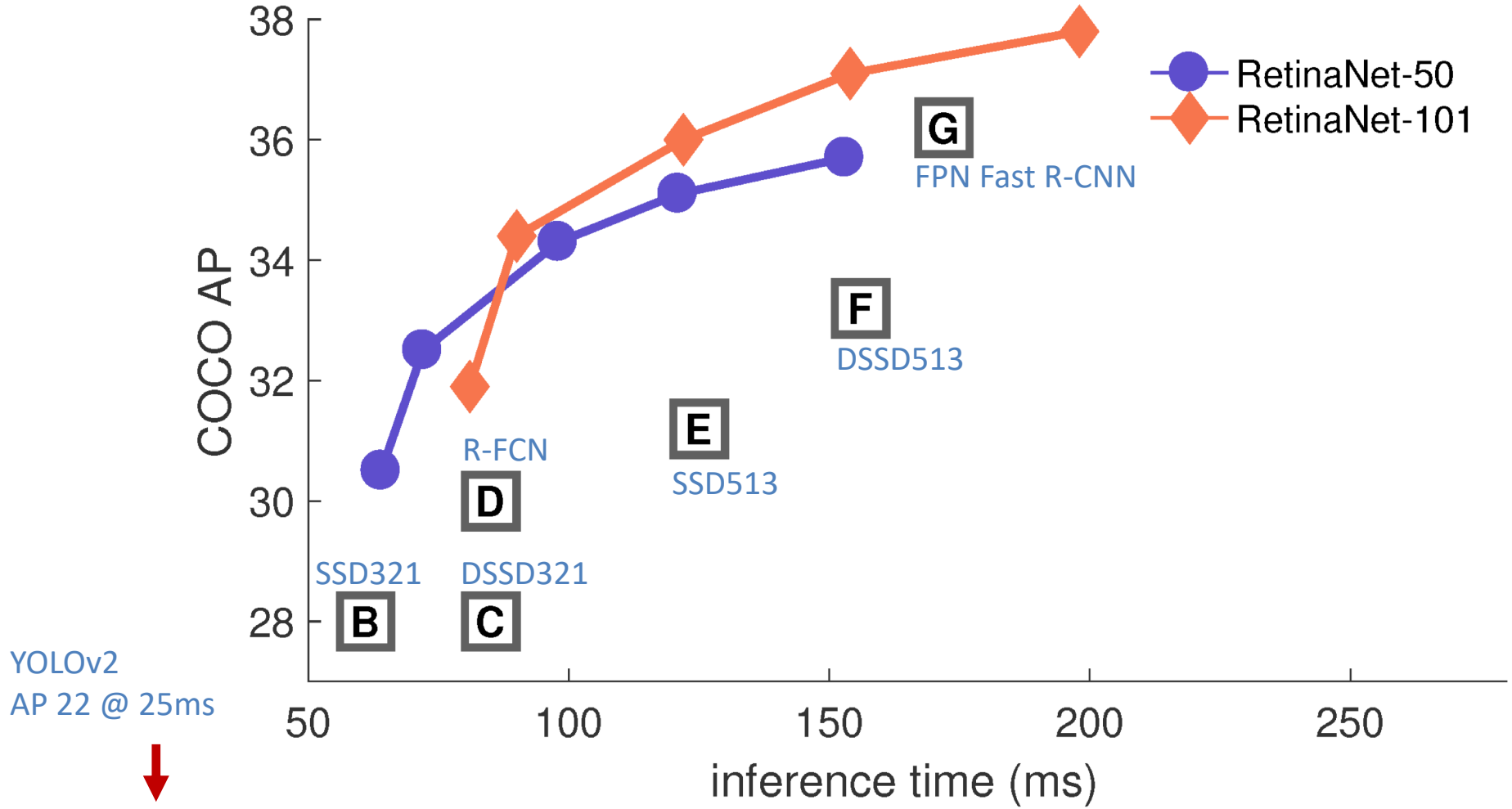
method	batch size	nms thr	AP
OHEM	128	.7	31.1
OHEM	256	.7	31.8
OHEM	512	.7	30.6
OHEM	128	.5	32.8
OHEM	256	.5	31.0
OHEM	512	.5	27.6
OHEM 1:3	128	.5	31.1
OHEM 1:3	256	.5	28.3
OHEM 1:3	512	.5	24.0
FL	n/a	n/a	**36.0**

→ **Best OHEM**

→ **Best Focal Loss**

Online Hard Example Mining, *Shrivastava et al., 2016*

RetinaNet performance



Summary

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

Mask R-CNN:
A Perspective on Equivariance

ICCV 2017 Tutorial, Venice, Italy

Kaiming He

in collaboration with: Georgia Gkioxari, Piotr Dollár, and Ross Girshick

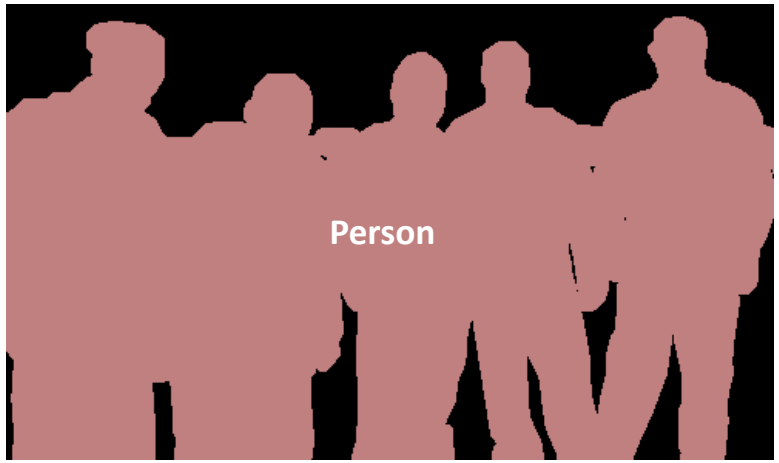
Facebook AI Research (FAIR)

Introduction

Visual Perception Problems

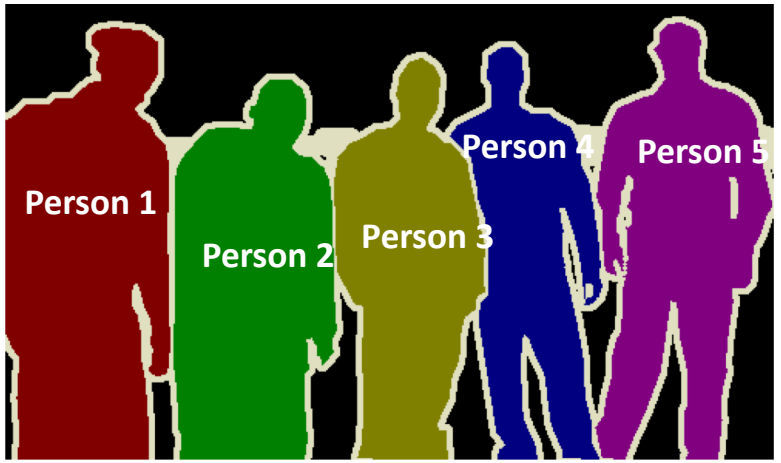
Object Detection

✓



Semantic Segmentation

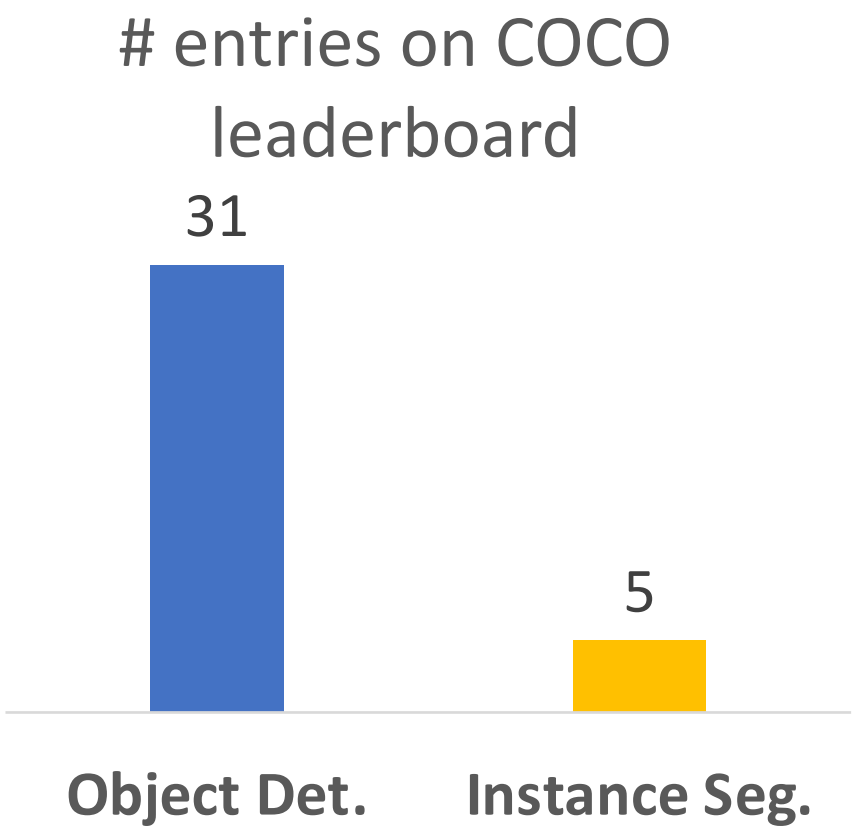
✓

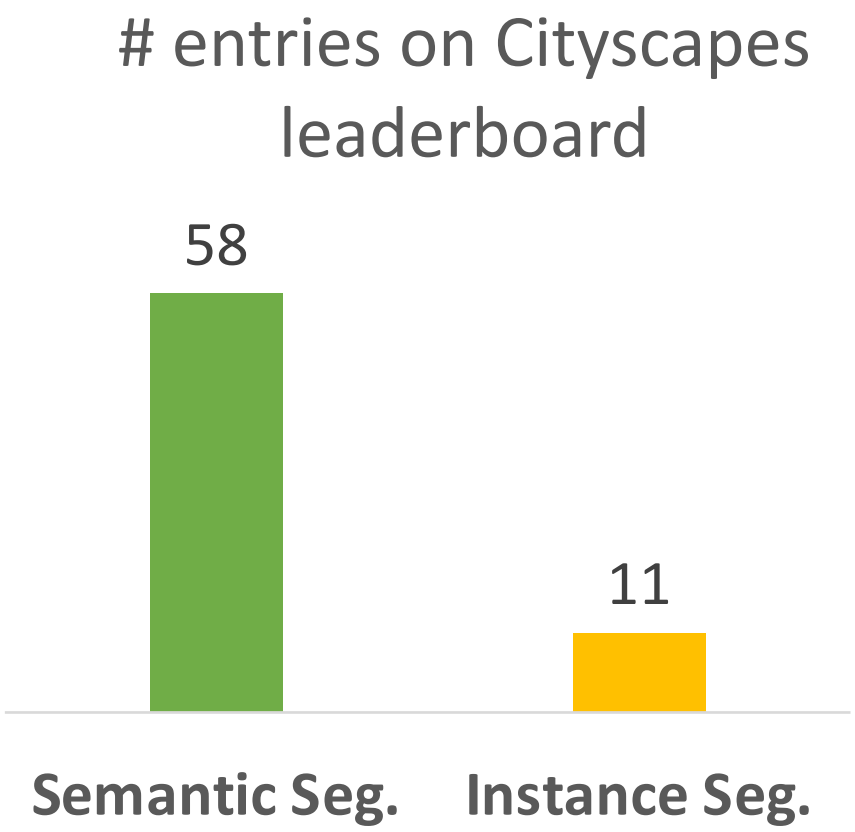


Instance Segmentation

?

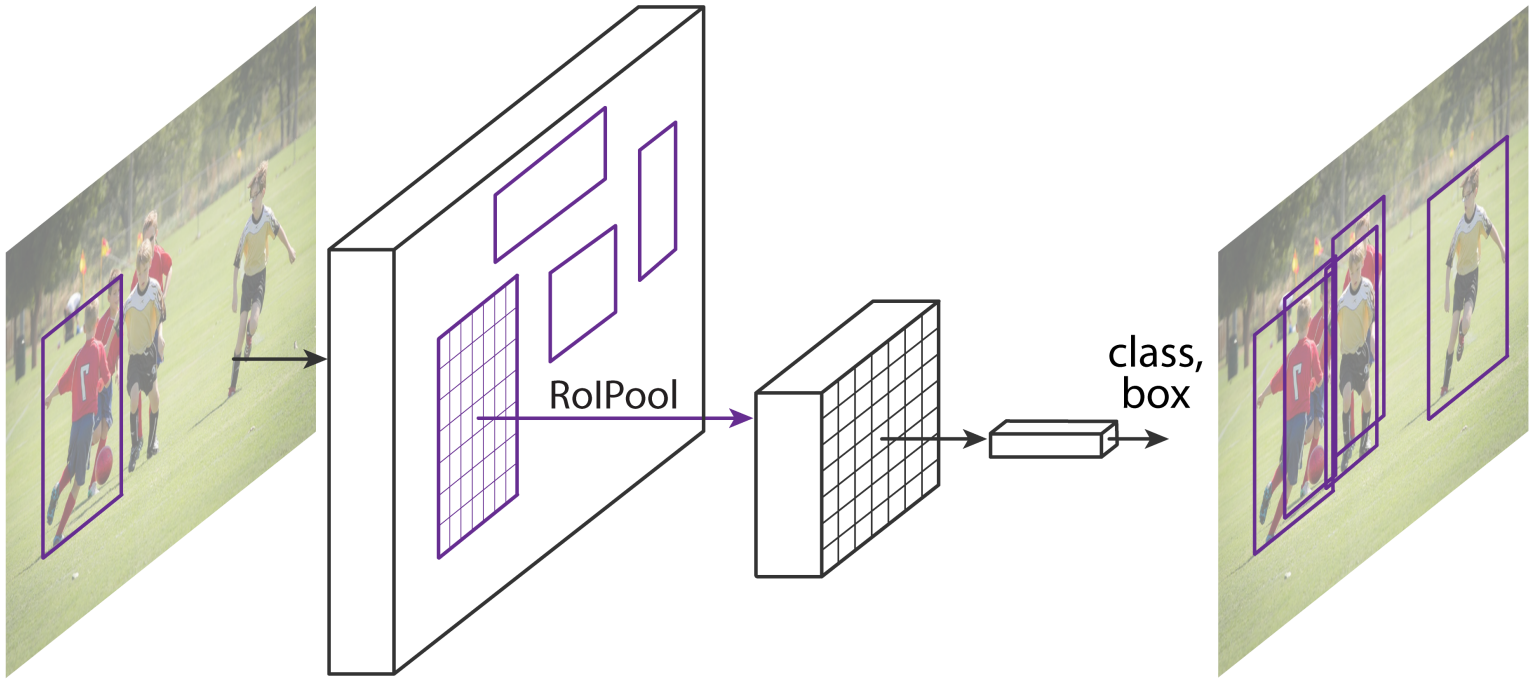
A Challenging Problem…





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

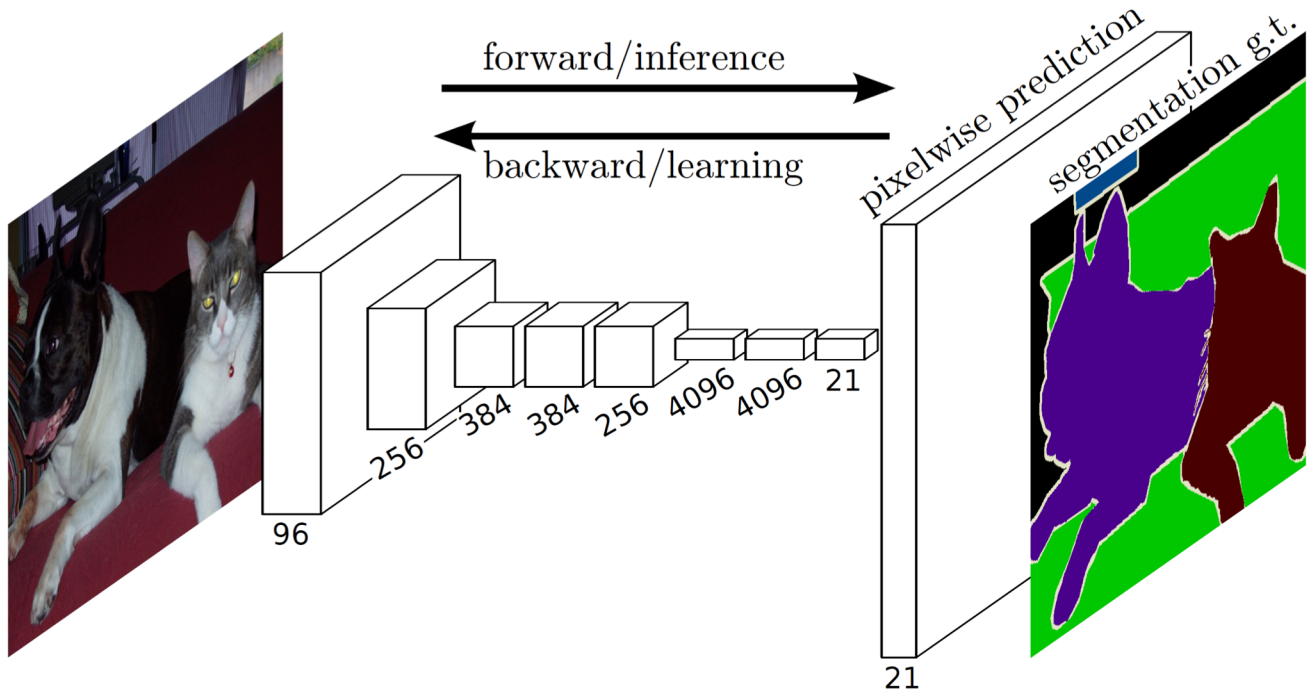
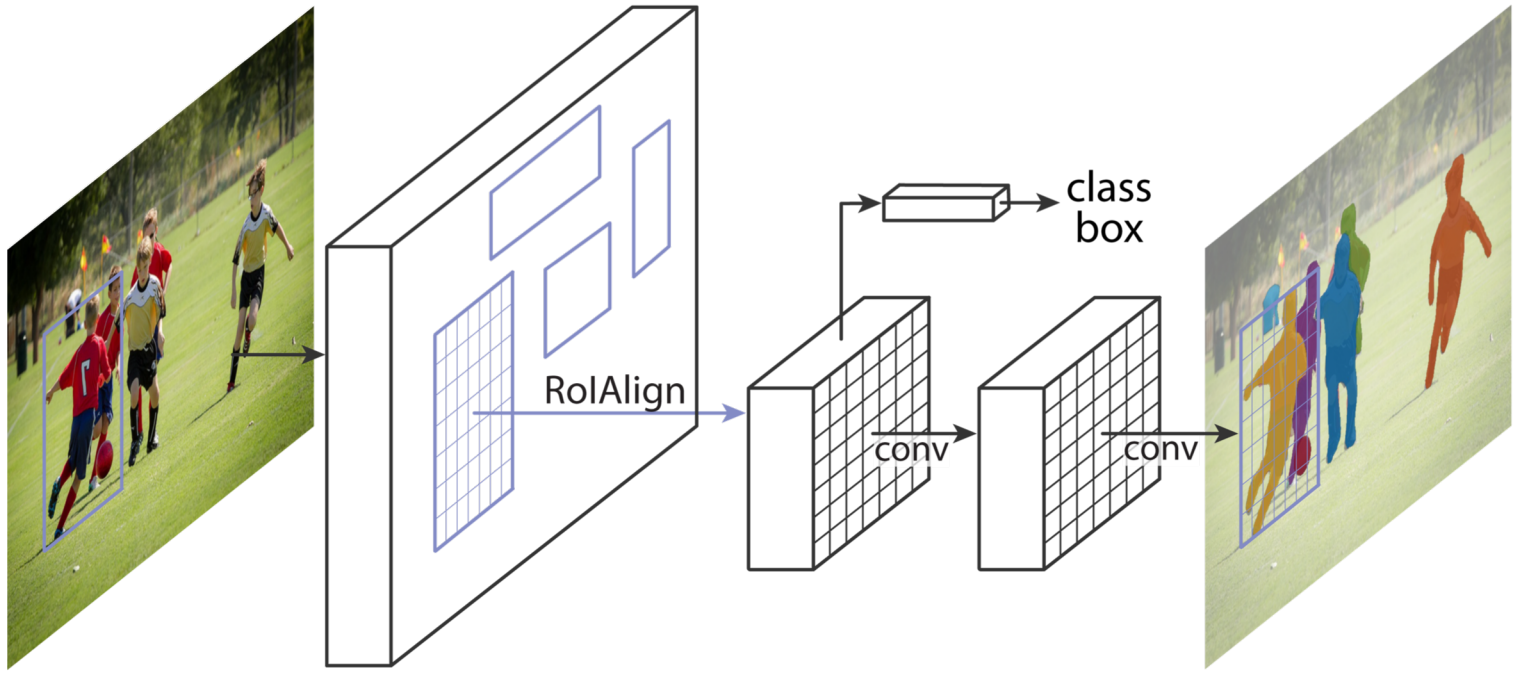


Figure credit: Long et al

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

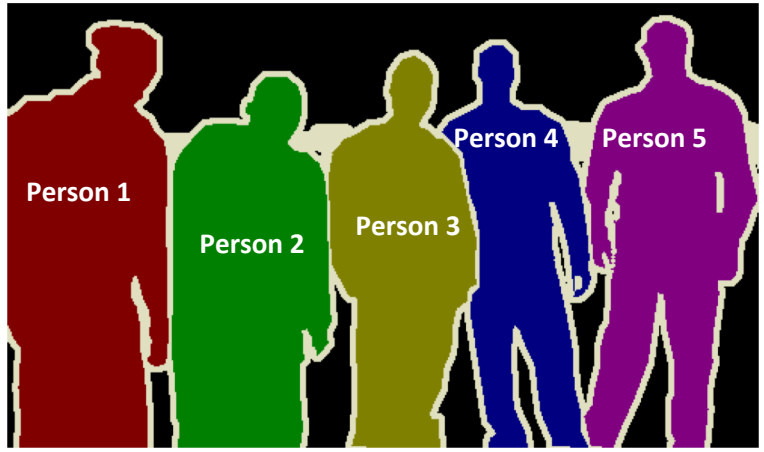
Instance Segmentation

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

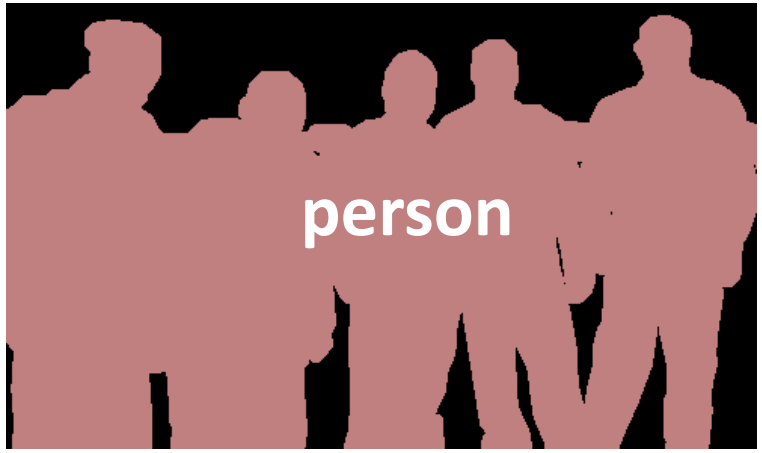


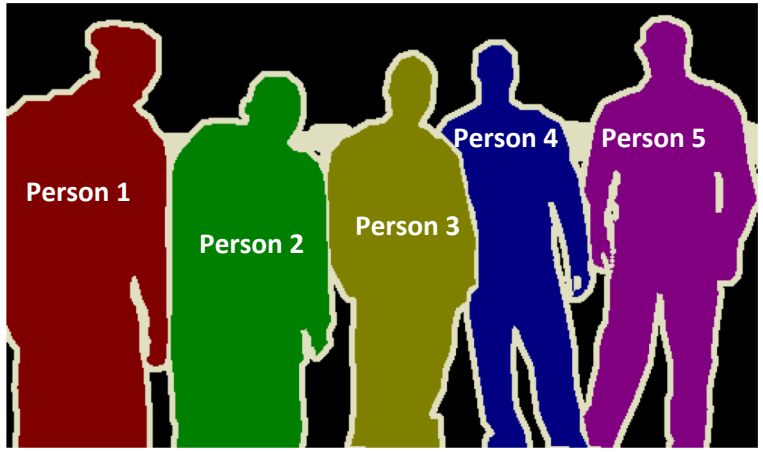
Instance Segmentation Methods

R-CNN driven

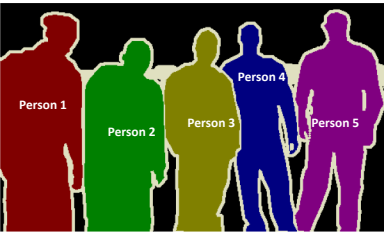


FCN driven



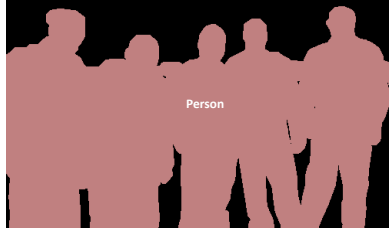


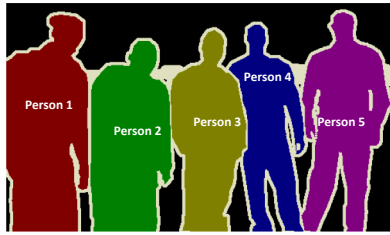
Instance Segmentation Methods



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]



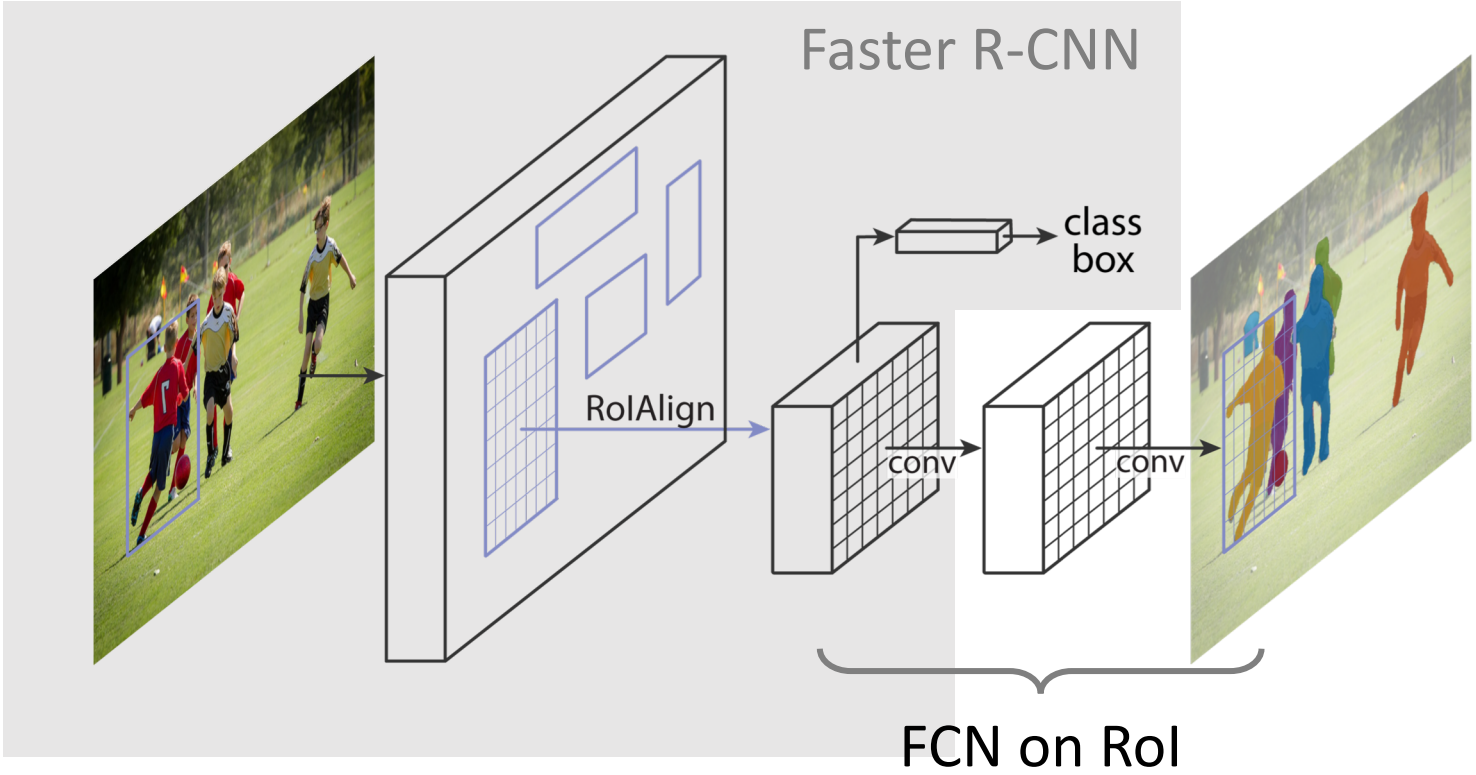


FCN-driven

- 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]

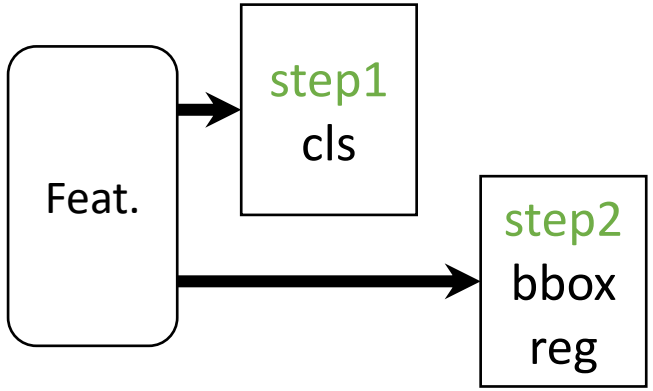
Mask R-CNN

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

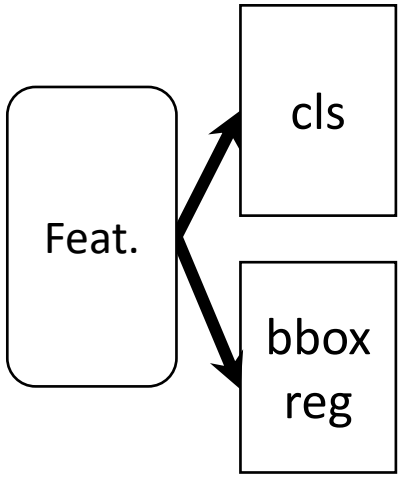


Parallel Heads

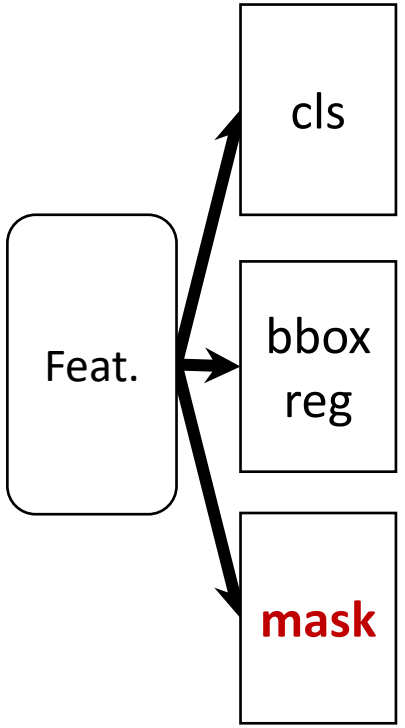
- Easy, fast to implement and train



(slow) R-CNN



Fast/er R-CNN

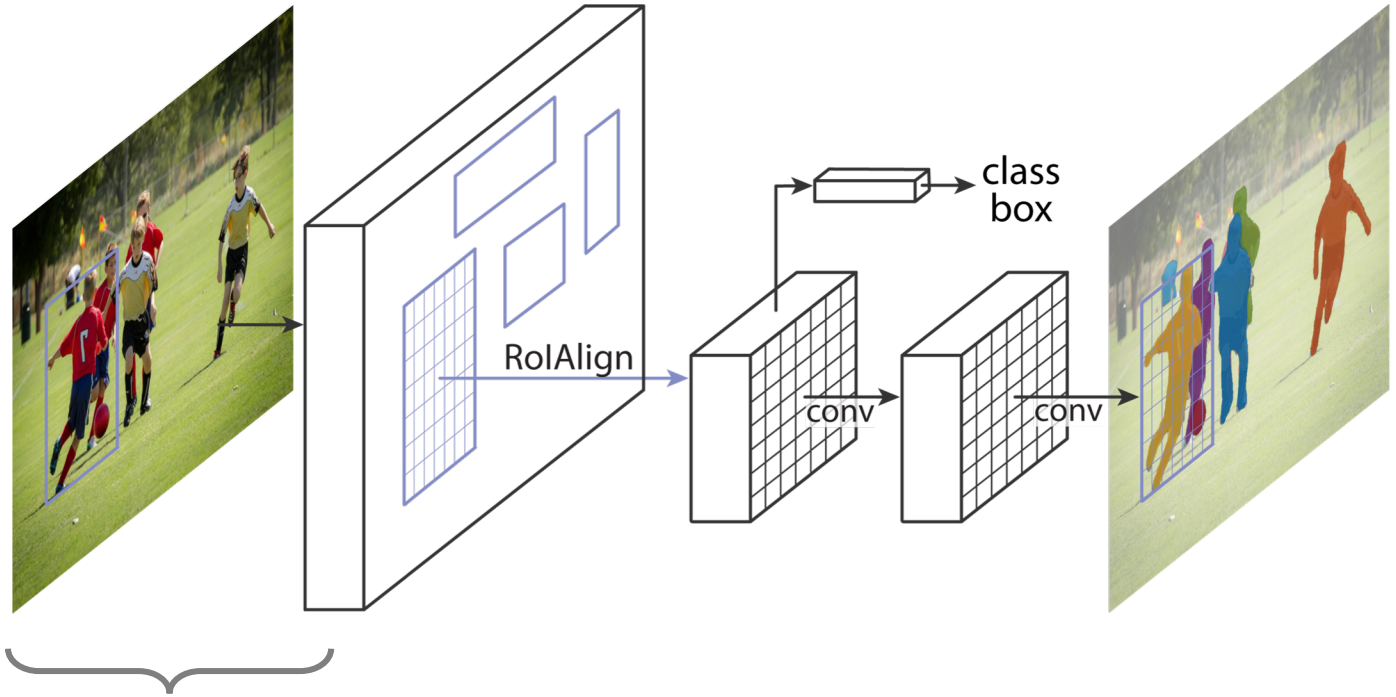


Mask R-CNN

Invariance vs. Equivariance

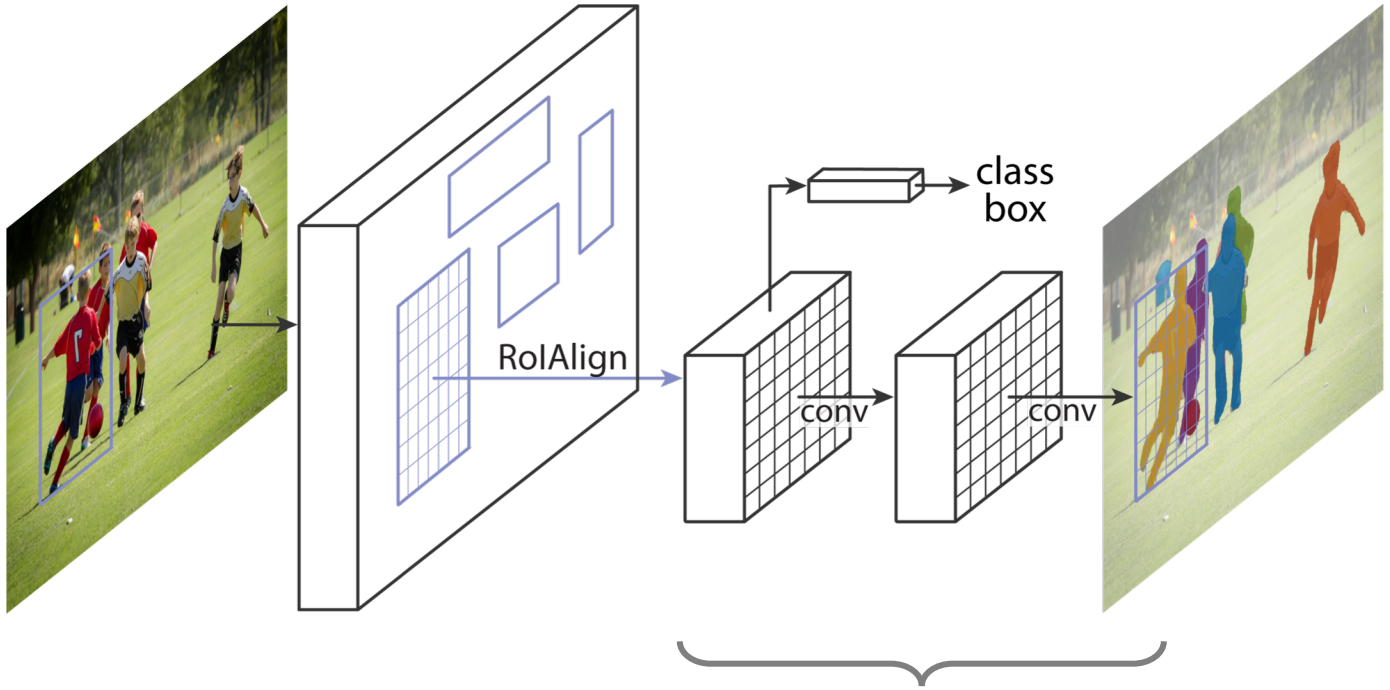
- Convolutions are translation-equivariant
- *Fully*-ConvNet (FCN) is translation-equivariant
- ConvNet becomes translation-invariant due to fully-connected or global pool layers

Equivariance in Mask R-CNN



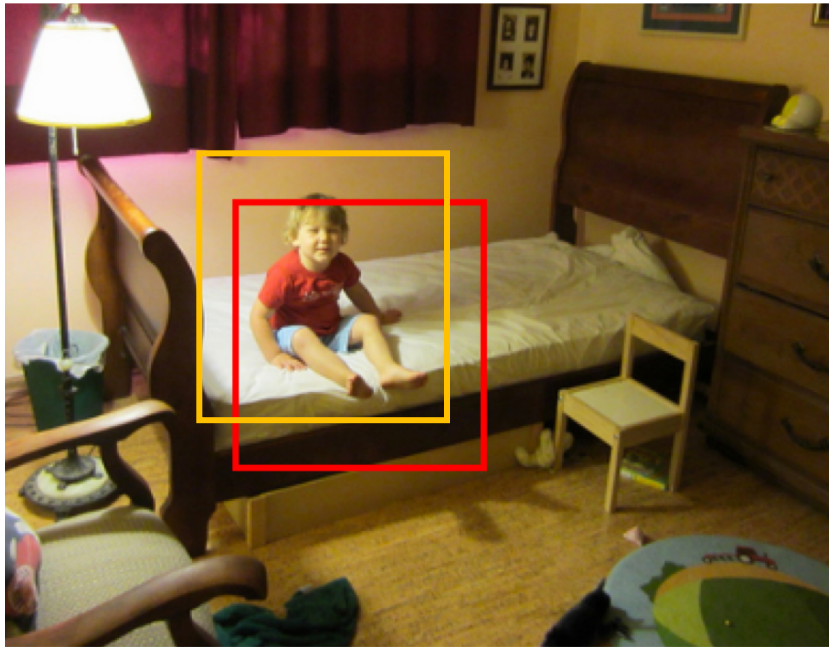
1. Fully-Conv Features:
equivariant to global (image) translation

Equivariance in Mask R-CNN

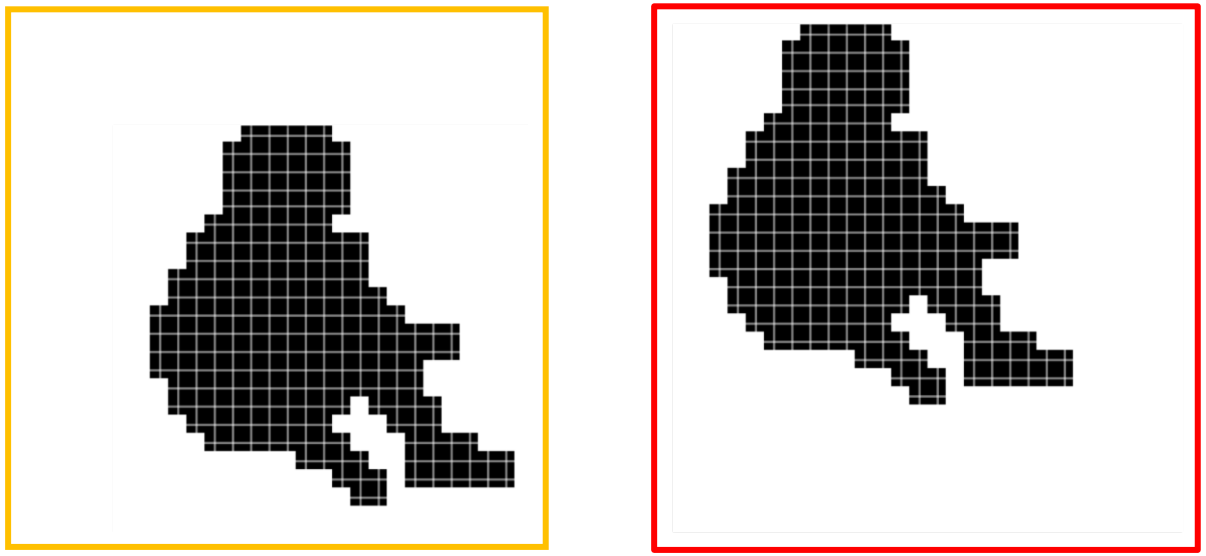


2. Fully-Conv on RoI:
equivariant to translation within RoI

Fully-Conv on RoI



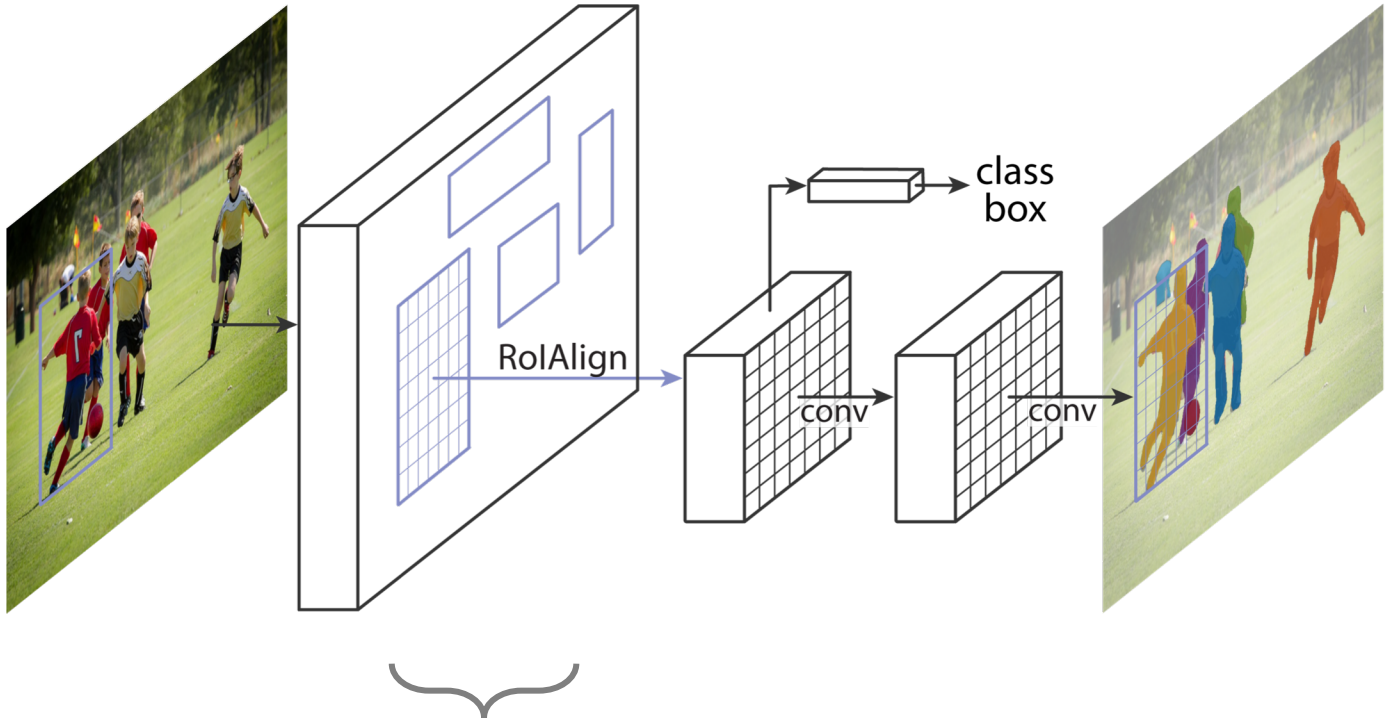
target masks on RoIs



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



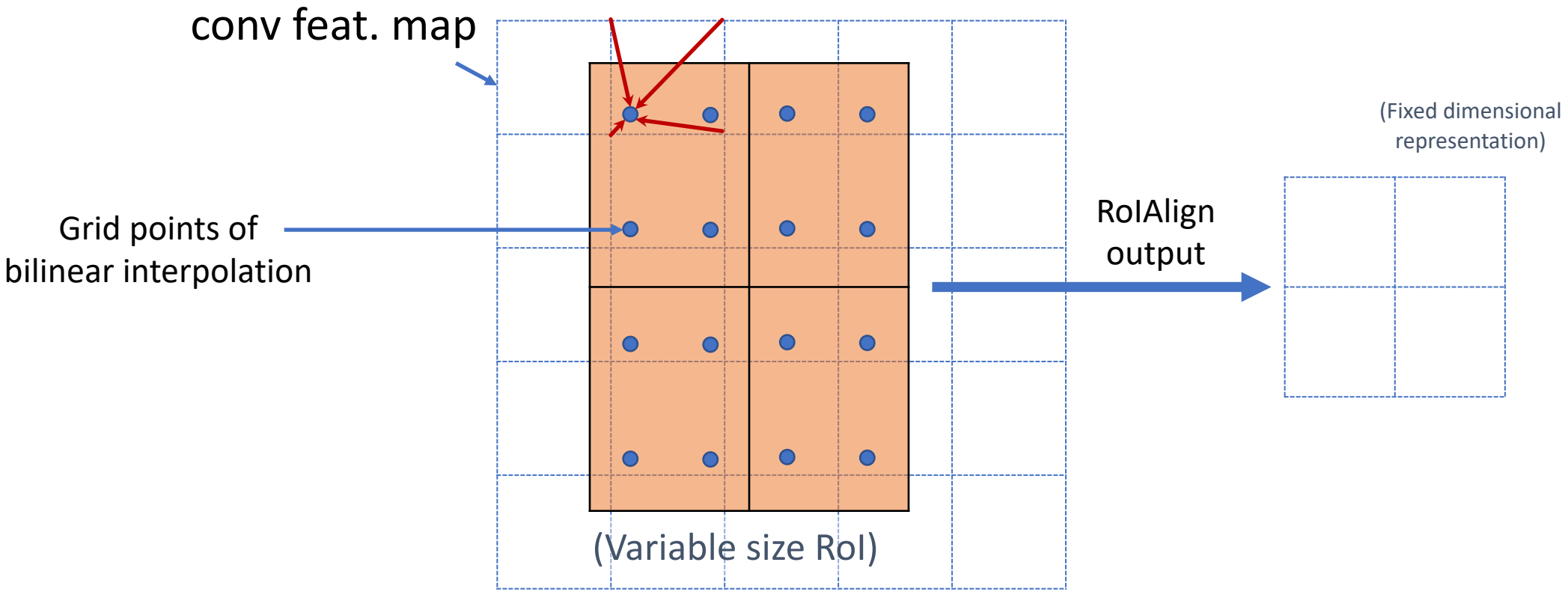
3. RoIAlign:

3a. maintain translation-equivariance before/after RoI

RoIAlign

FAQs: how to sample grid points within a cell?

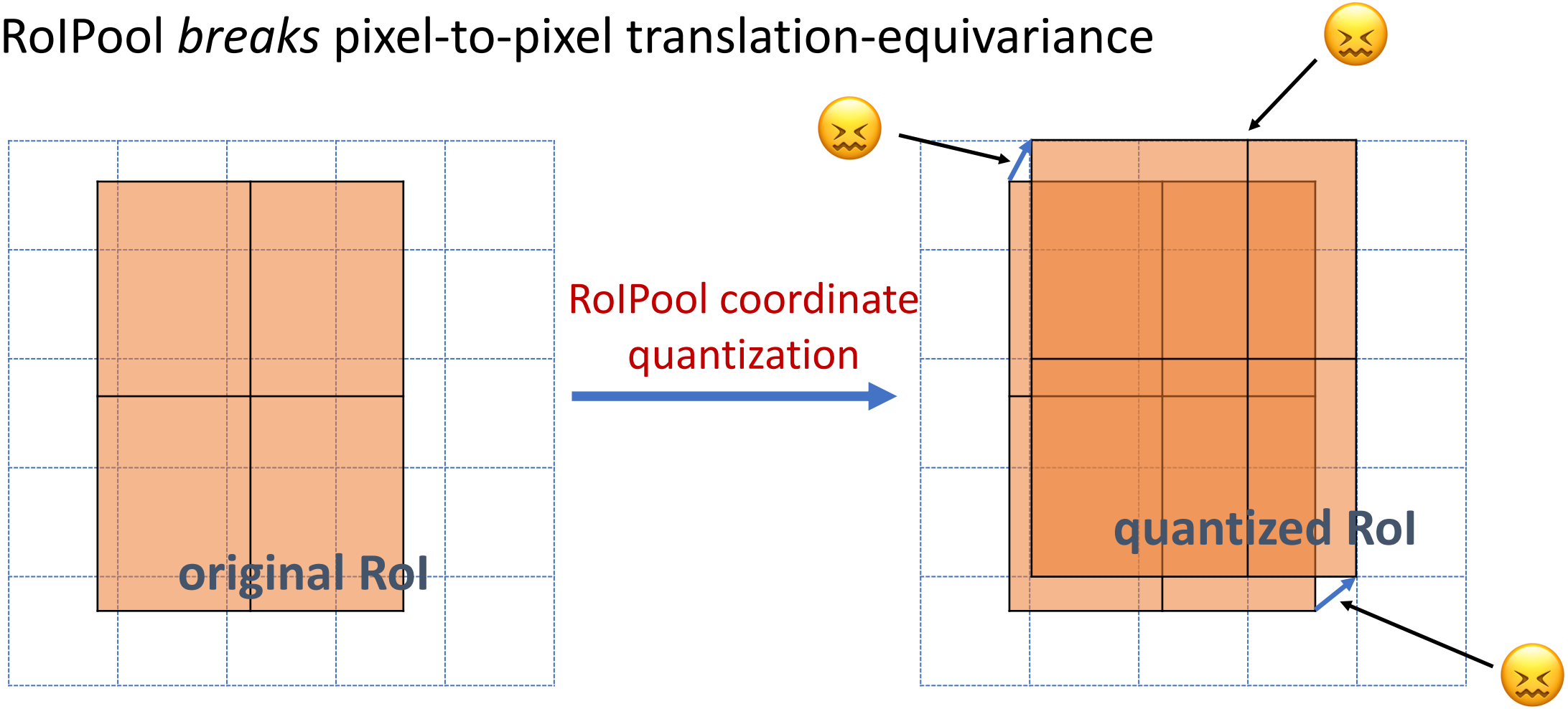
- 4 regular points in 2x2 sub-cells
- other implementation could work



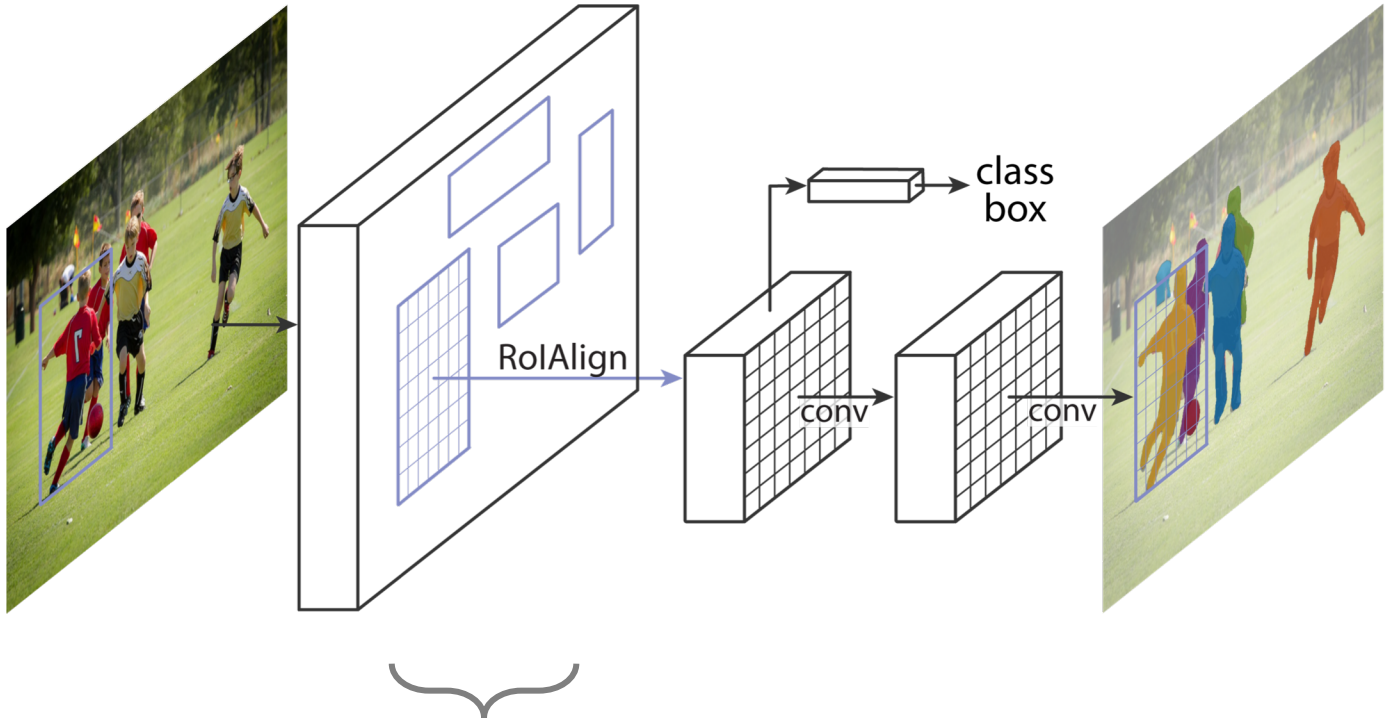
see also “What is wrong with convolutional neural nets?”, Geoffrey Hinton, 2017

RoIAlign vs. RoIPool

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



Equivariance in Mask R-CNN



3. RoIAlign:

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

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

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**

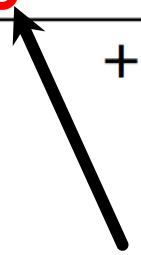
	mask AP			box AP		
	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

- 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

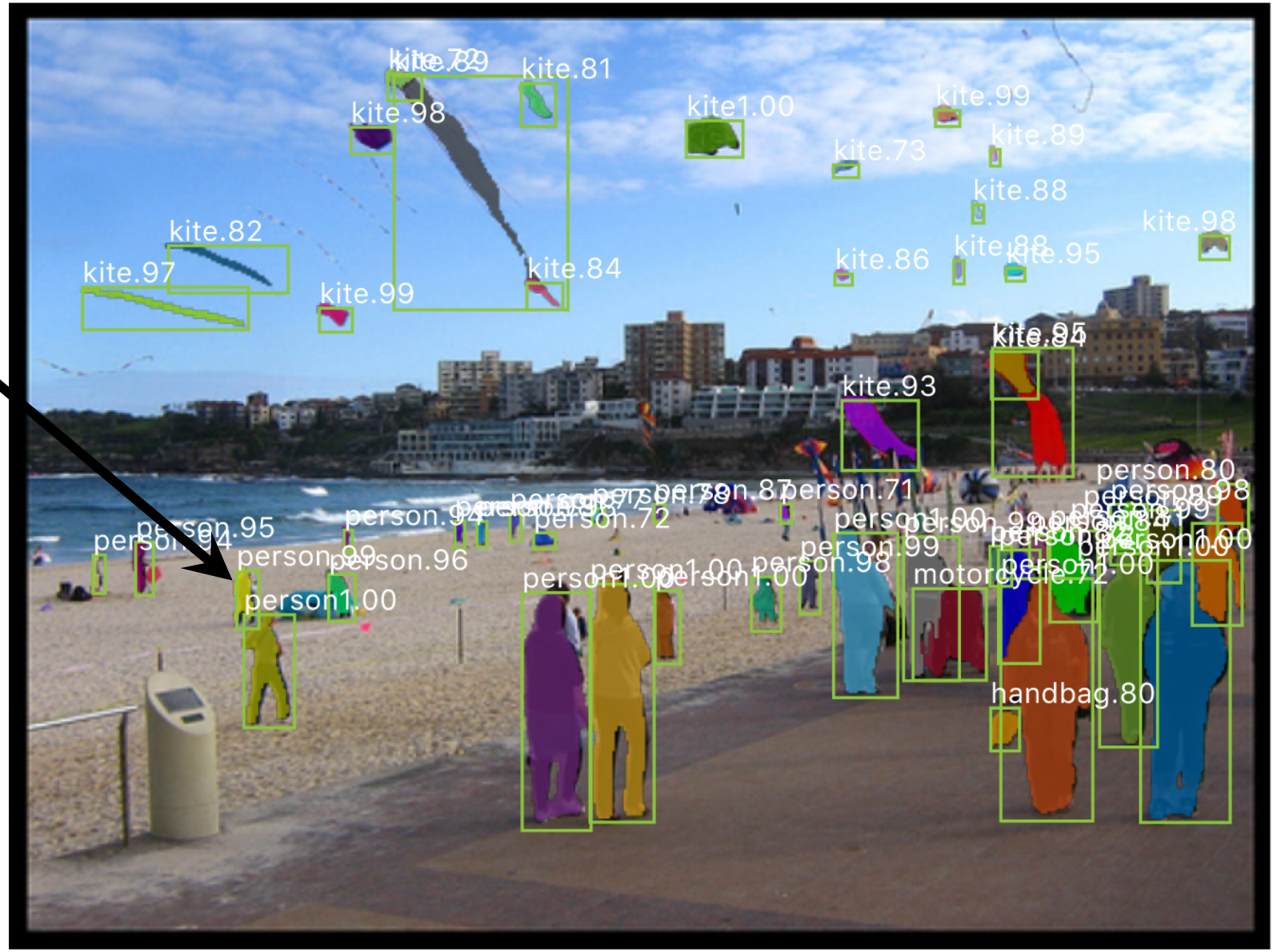
bbox detection improved by:

- RoIAlign
- Multi-task training w/ mask

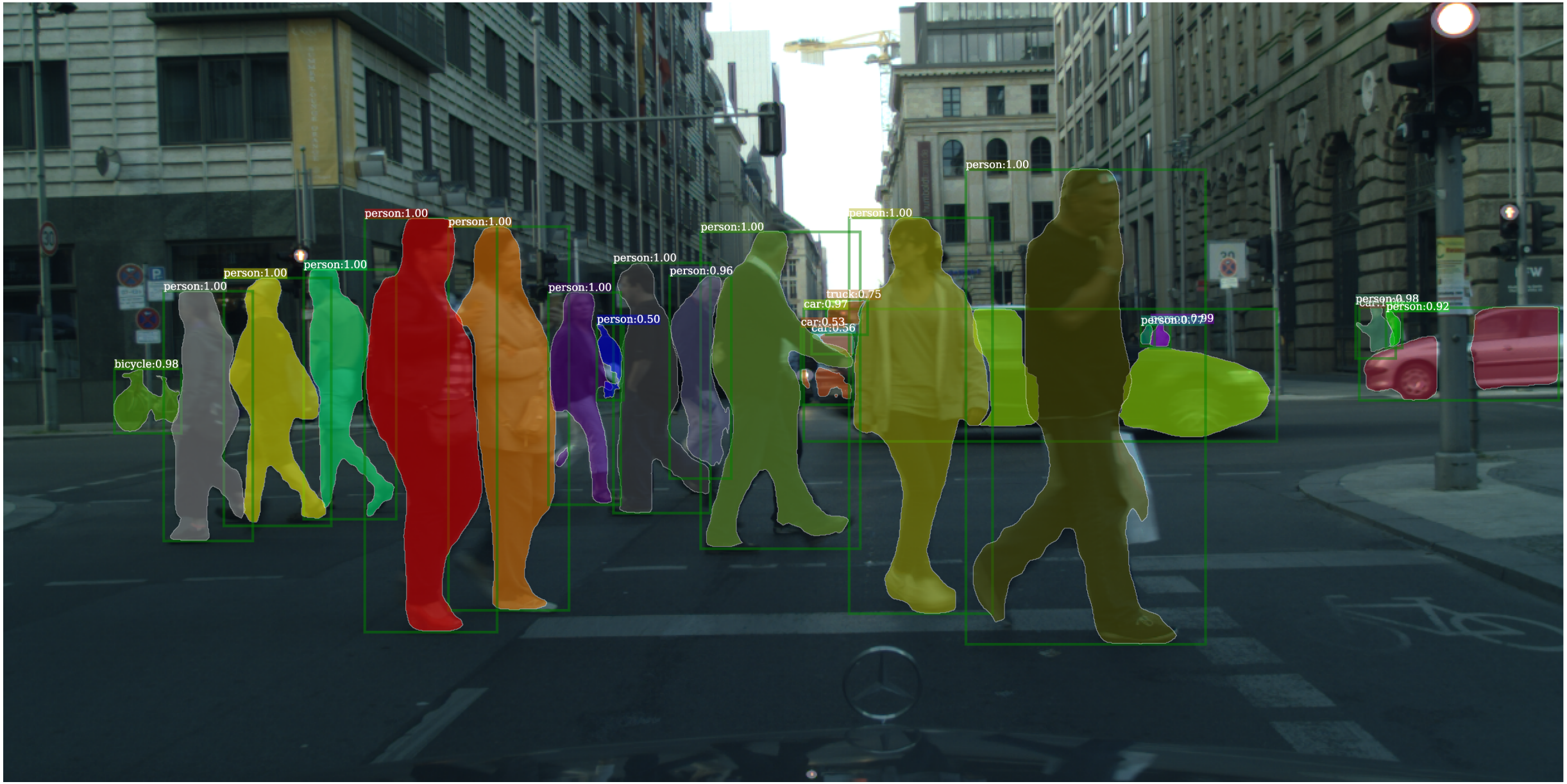
disconnected object

Mask R-CNN results on COCO

small objects



Mask R-CNN results on COCO

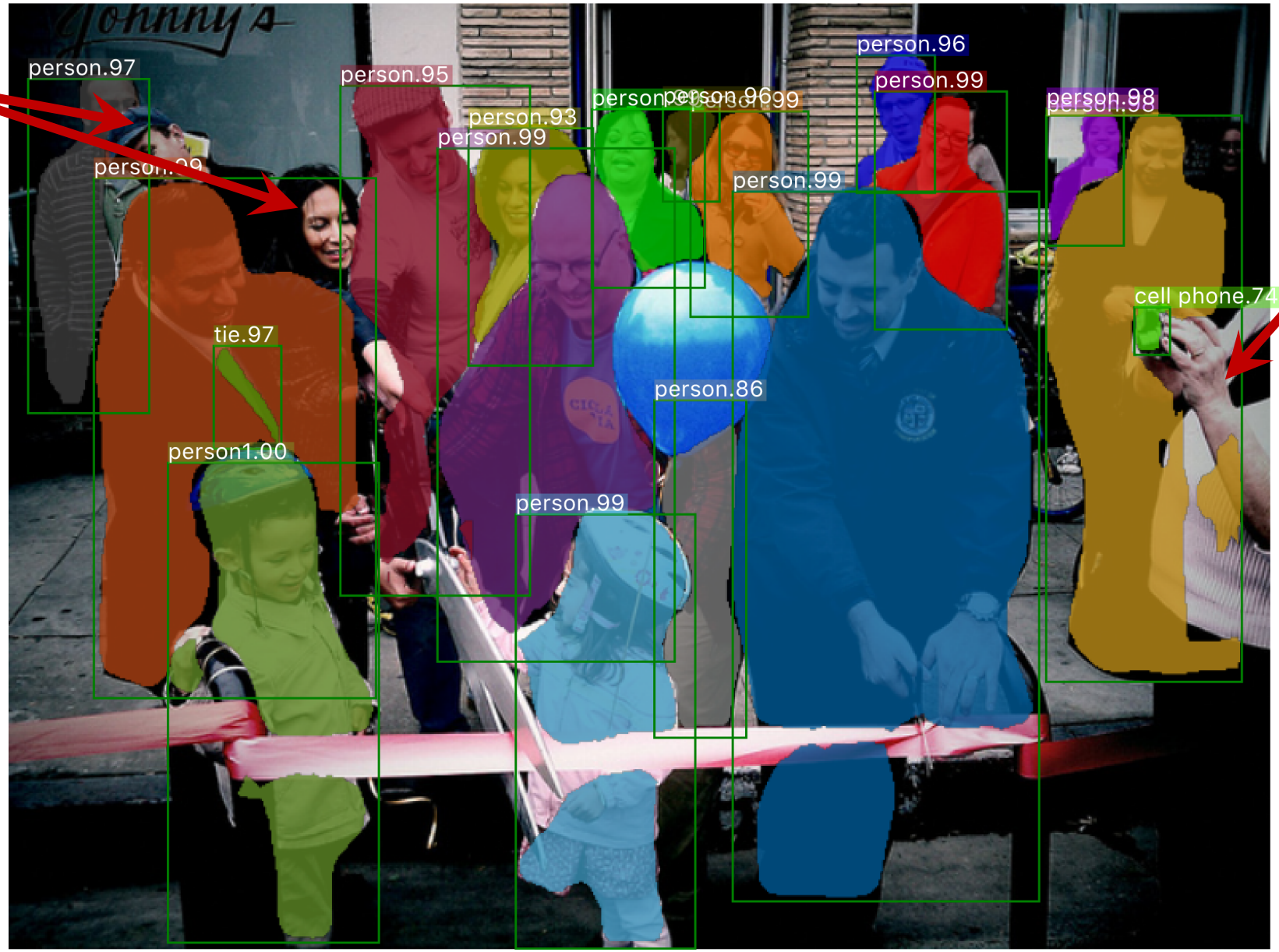


Mask R-CNN results on CityScapes

Failure case: detection/segmentation

missing

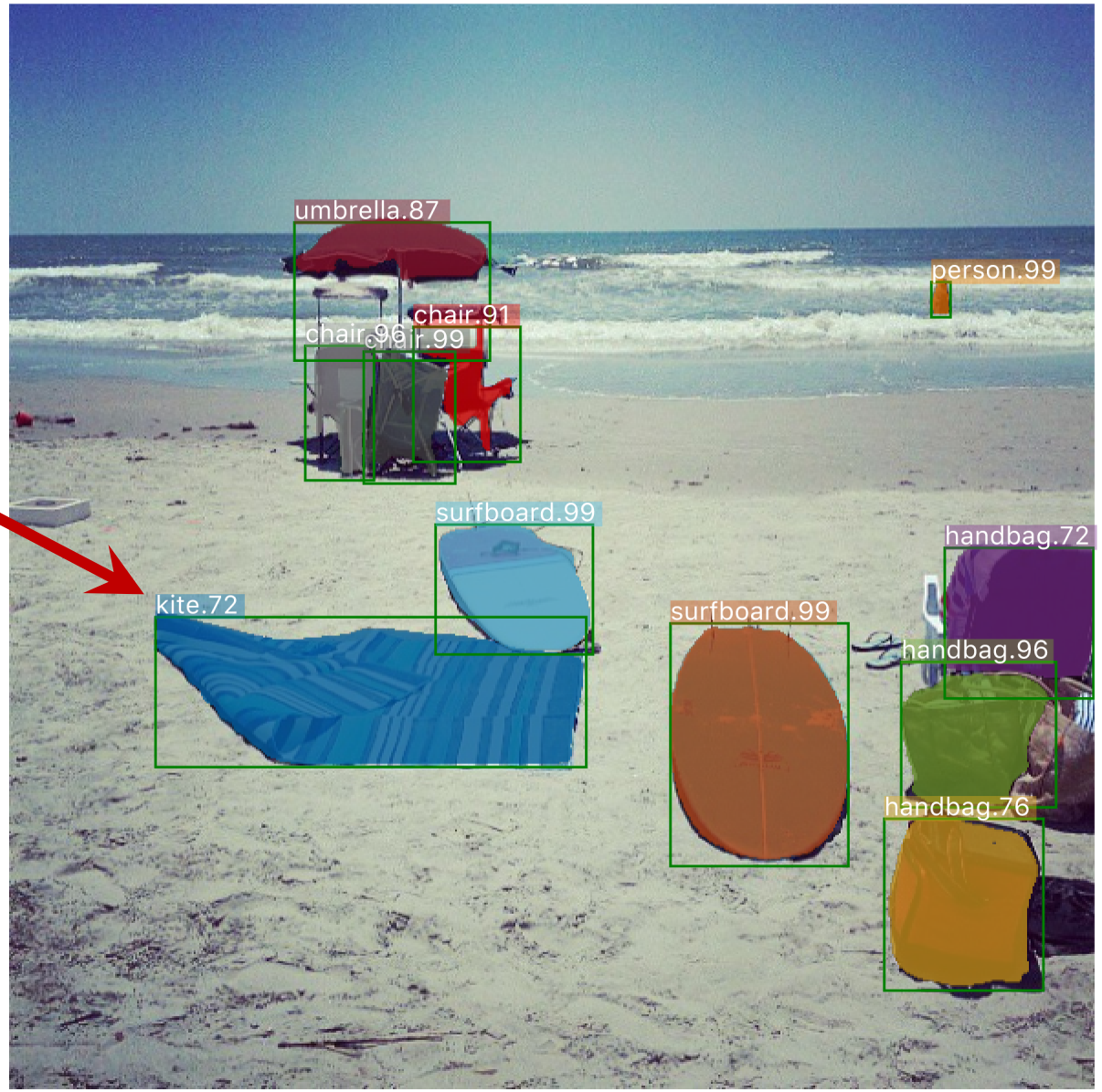
missing, false mask



Mask R-CNN results on COCO

Failure case: recognition

not a kite

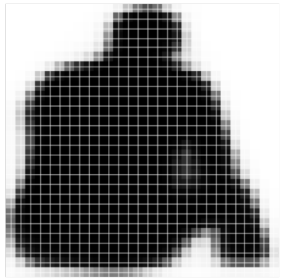


Mask R-CNN results on COCO

Validation image with box detection shown in red

28x28 soft prediction from Mask R-CNN

(enlarged)



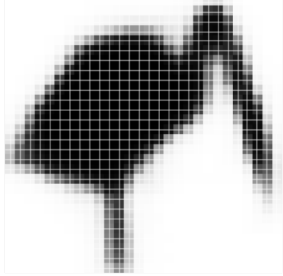
Soft prediction resampled to image coordinates

(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

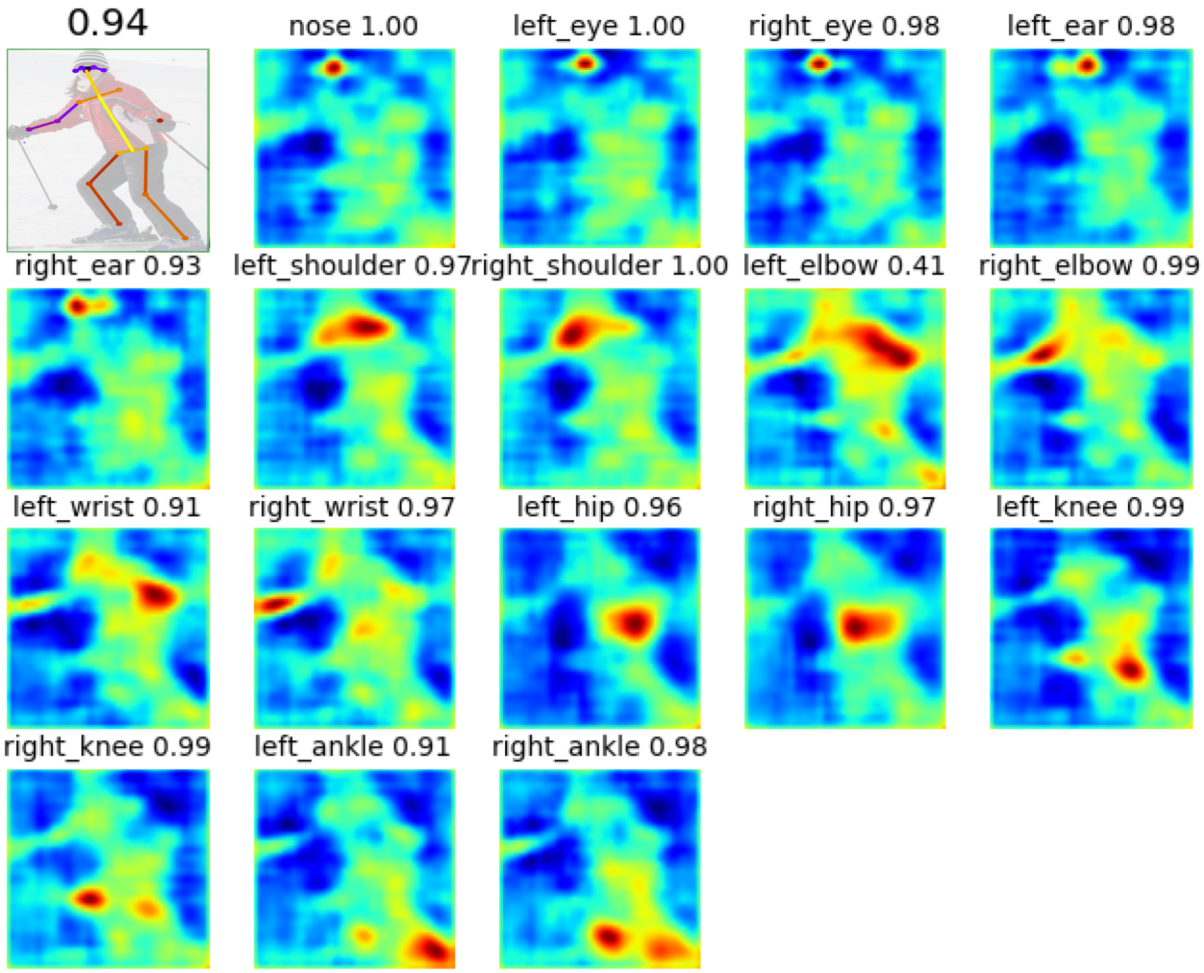
Final mask

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

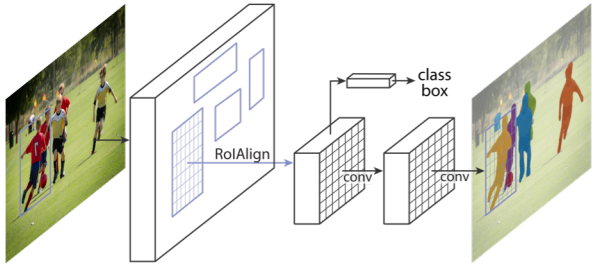
- Desire the same equivariances
  - translation, scale, aspect ratio




www.tfrancis.co.uk

Mask R-CNN frame-by-frame

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