The Generalized R-CNN Framework for Object Detection

CVPR 2019 Tutorial
Visual Recognition and Beyond

Ross Girshick

facebook Artificial Intelligence Research

Overview of this Tutorial

Topics to cover

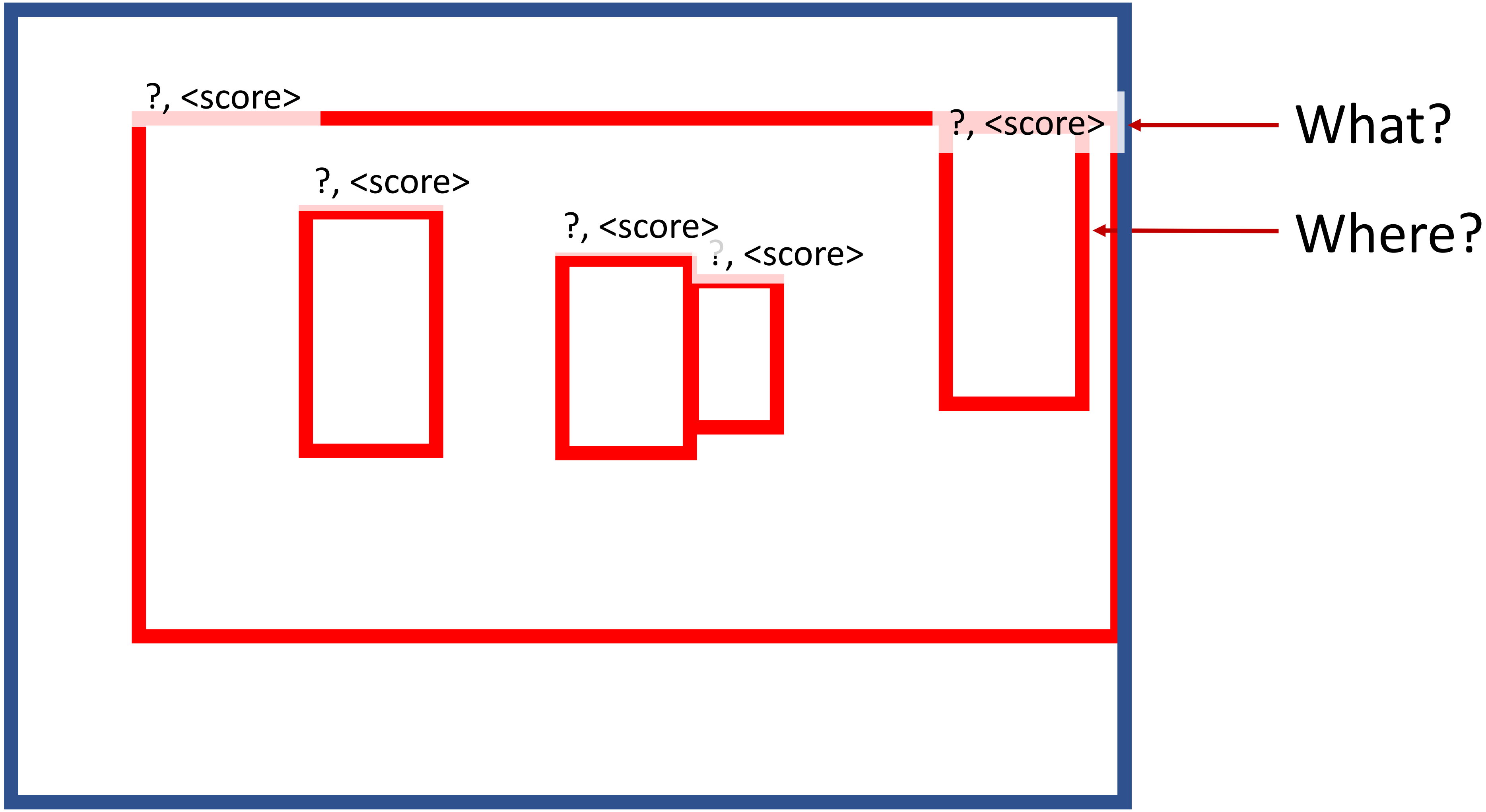
- Object detection intro (very brief)
- The Generalized R-CNN framework
- Open challenges in object detection

Overview of this Tutorial

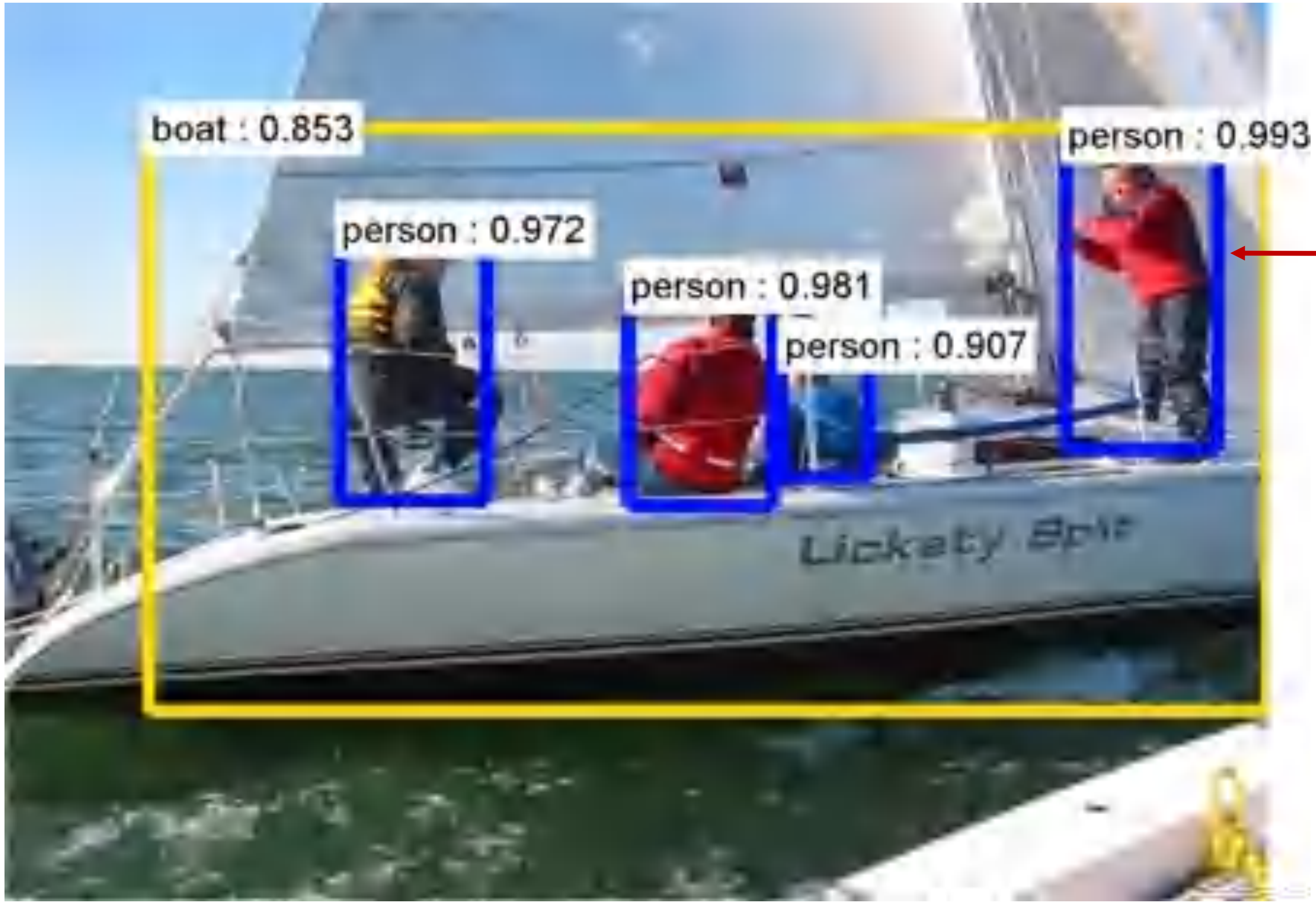
Topics to cover

- Object detection intro (very brief)
- The Generalized R-CNN framework
- Open challenges in object detection

Object Detection



Object Detection with Bounding Boxes

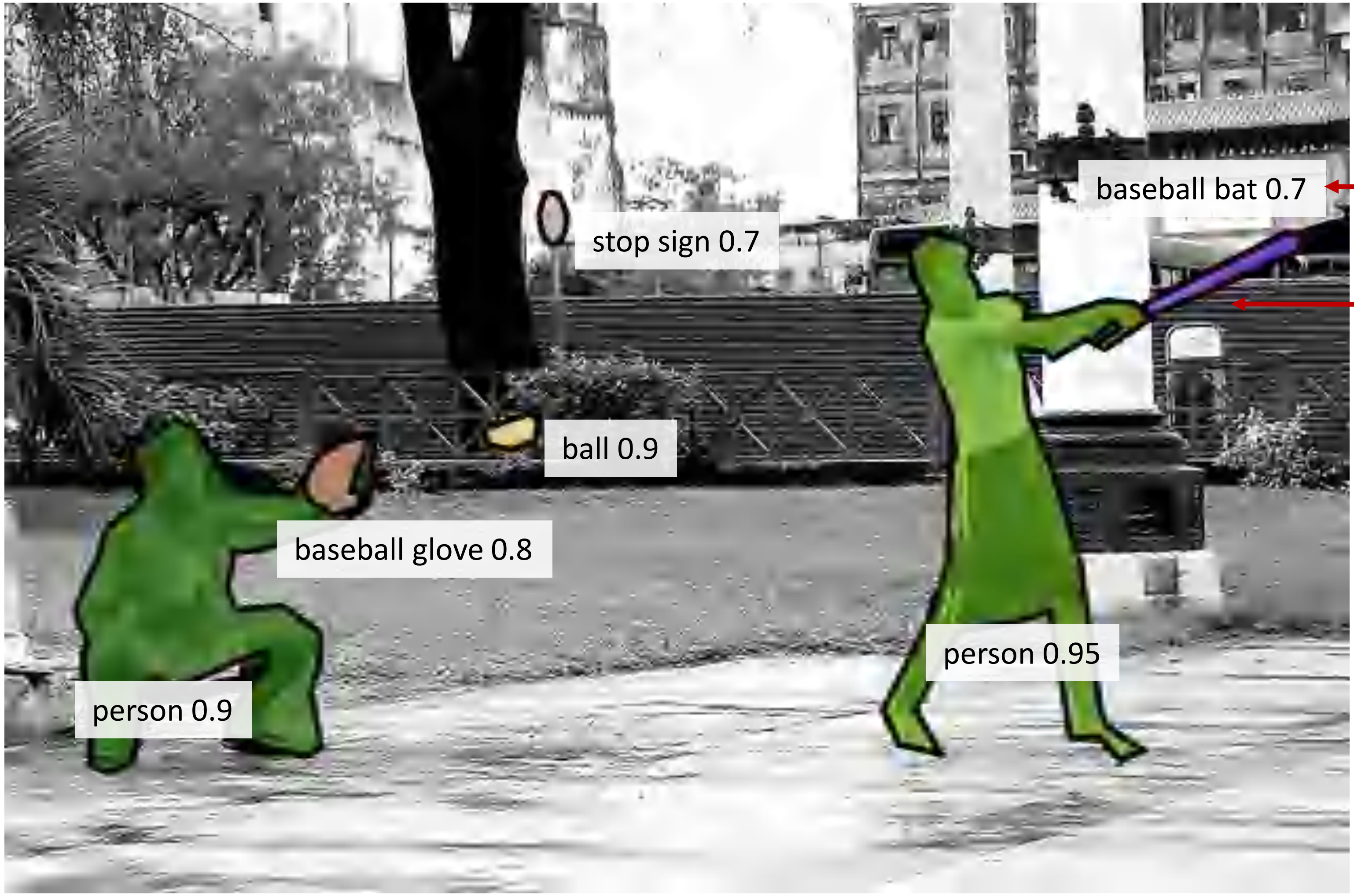


What?

Where?

"Object detection"

Object Detection with Segmentation Masks



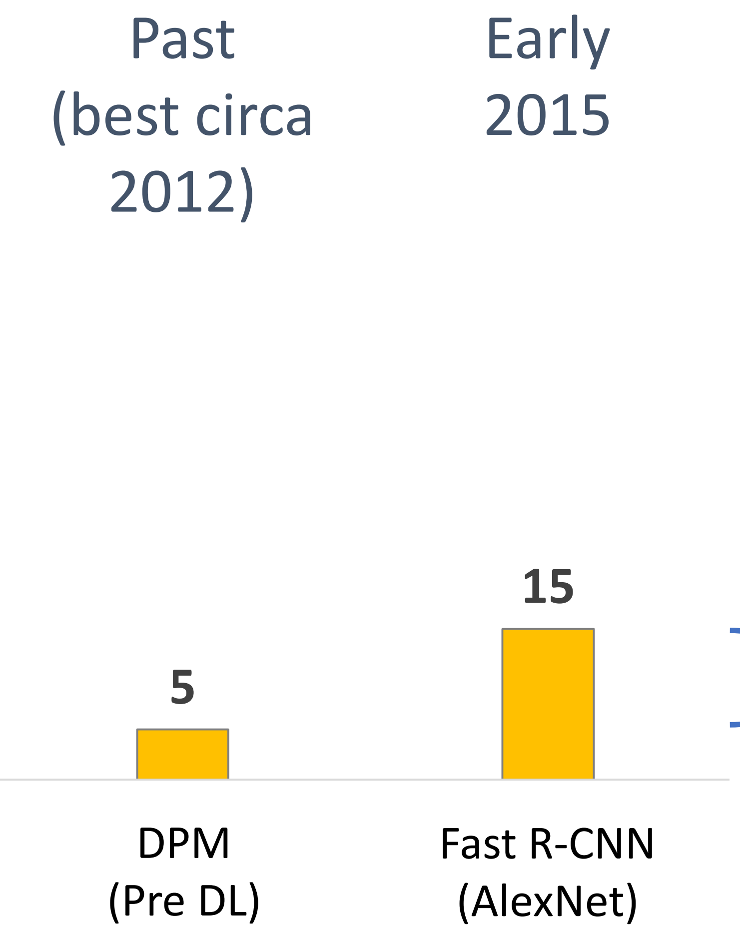
What?

Where?

"Instance segmentation"

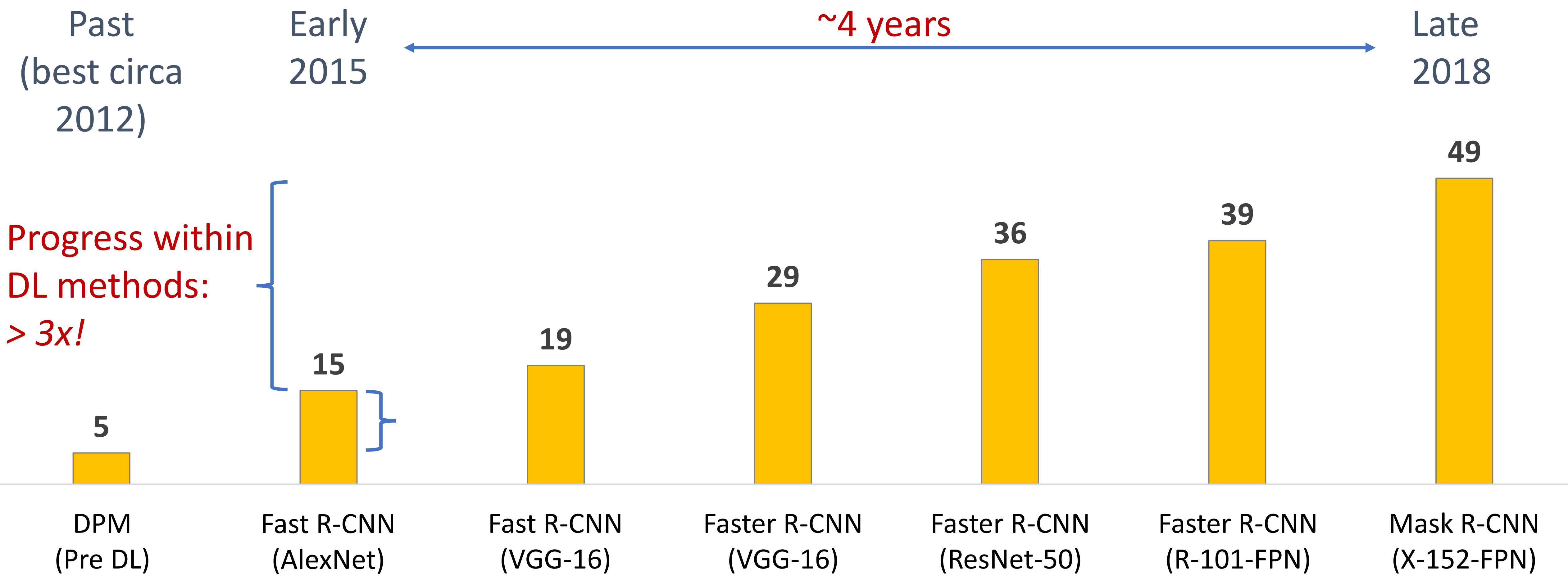
Modern Object Detection: Is this Picture Correct?

COCO Object Detection Average Precision (%)

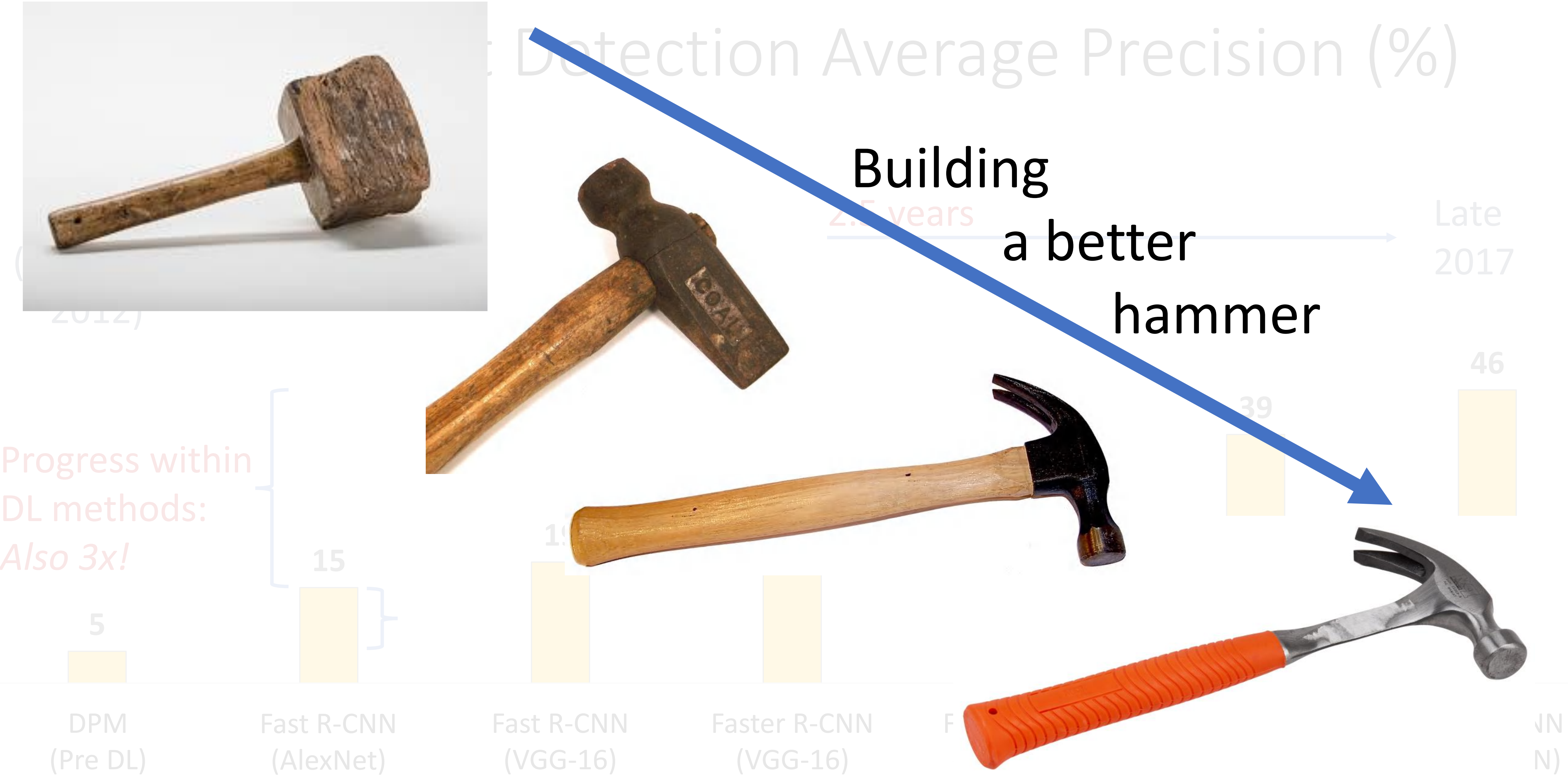


Movement to
Deep Learning methods:
3x improvement in AP

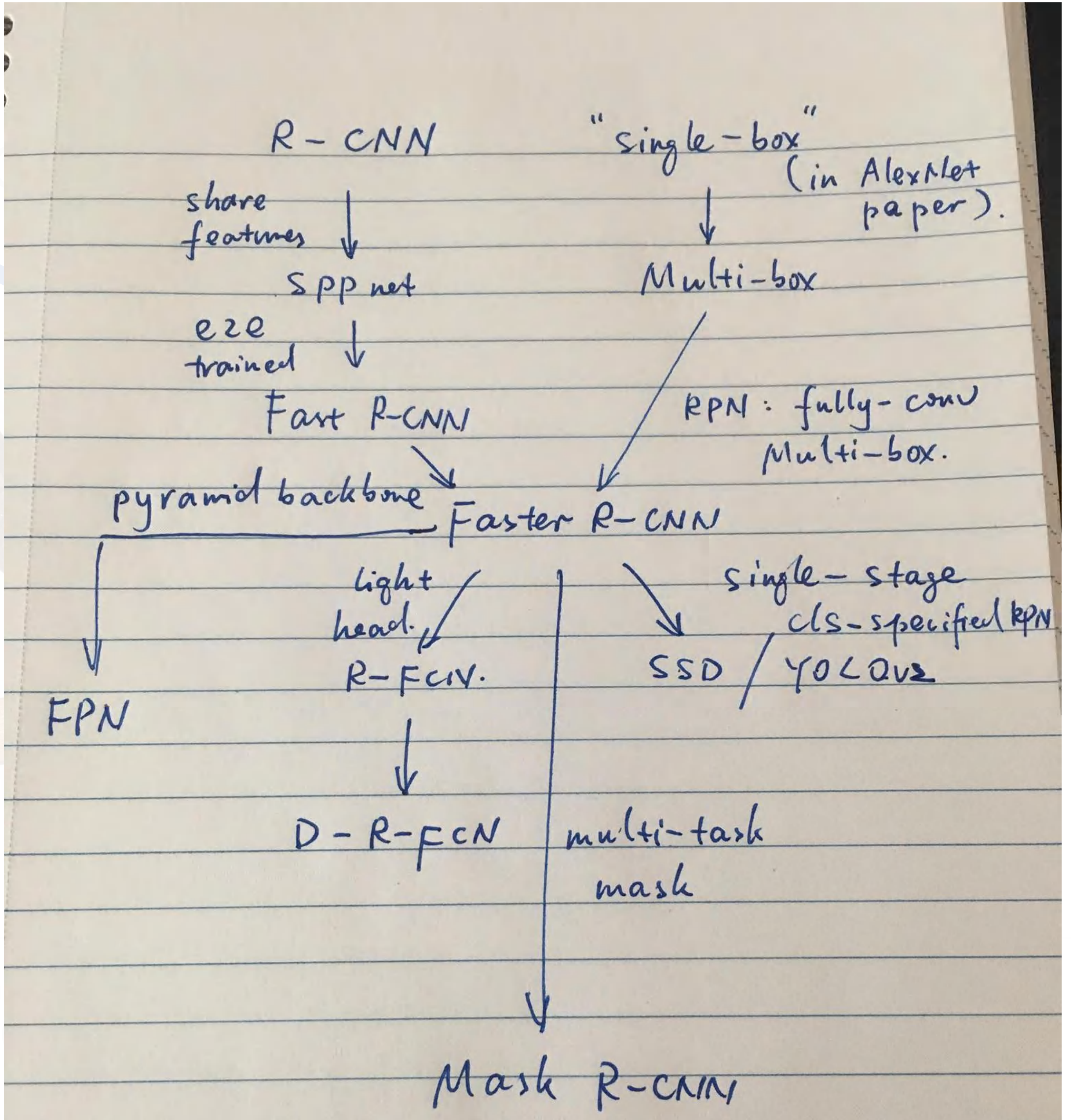
COCO Object Detection Average Precision (%)



Modern Object Detection: Is this Picture Correct?

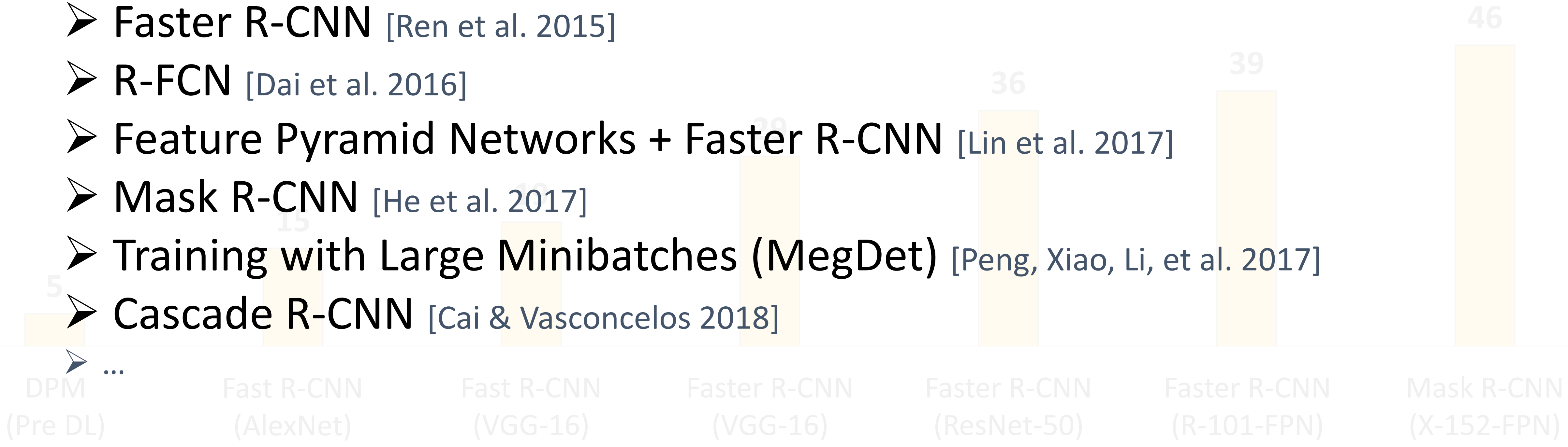

Building
a better
hammer
Detection Average Precision (%)
Late
2017
Progress within
DL methods:
Also 3x!
5
15
39
46
DPM
(Pre DL)
Fast R-CNN
(AlexNet)
Fast R-CNN
(VGG-16)
Faster R-CNN
(VGG-16)

Modern object detection is a complex web of related methods



Steady Progress on Boxes and Masks

- R-CNN [Girshick et al. 2014]
- SPP-net [He et al. 2014]
- Fast R-CNN [Girshick. 2015]
- Faster R-CNN [Ren et al. 2015]
- R-FCN [Dai et al. 2016]
- Feature Pyramid Networks + Faster R-CNN [Lin et al. 2017]
- Mask R-CNN [He et al. 2017]
- Training with Large Minibatches (MegDet) [Peng, Xiao, Li, et al. 2017]
- Cascade R-CNN [Cai & Vasconcelos 2018]
- …



Beyond Boxes and Masks: Human Keypoints

COCO Keypoint Detection Task
[COCO team @ cocodataset.org 2016 - present]

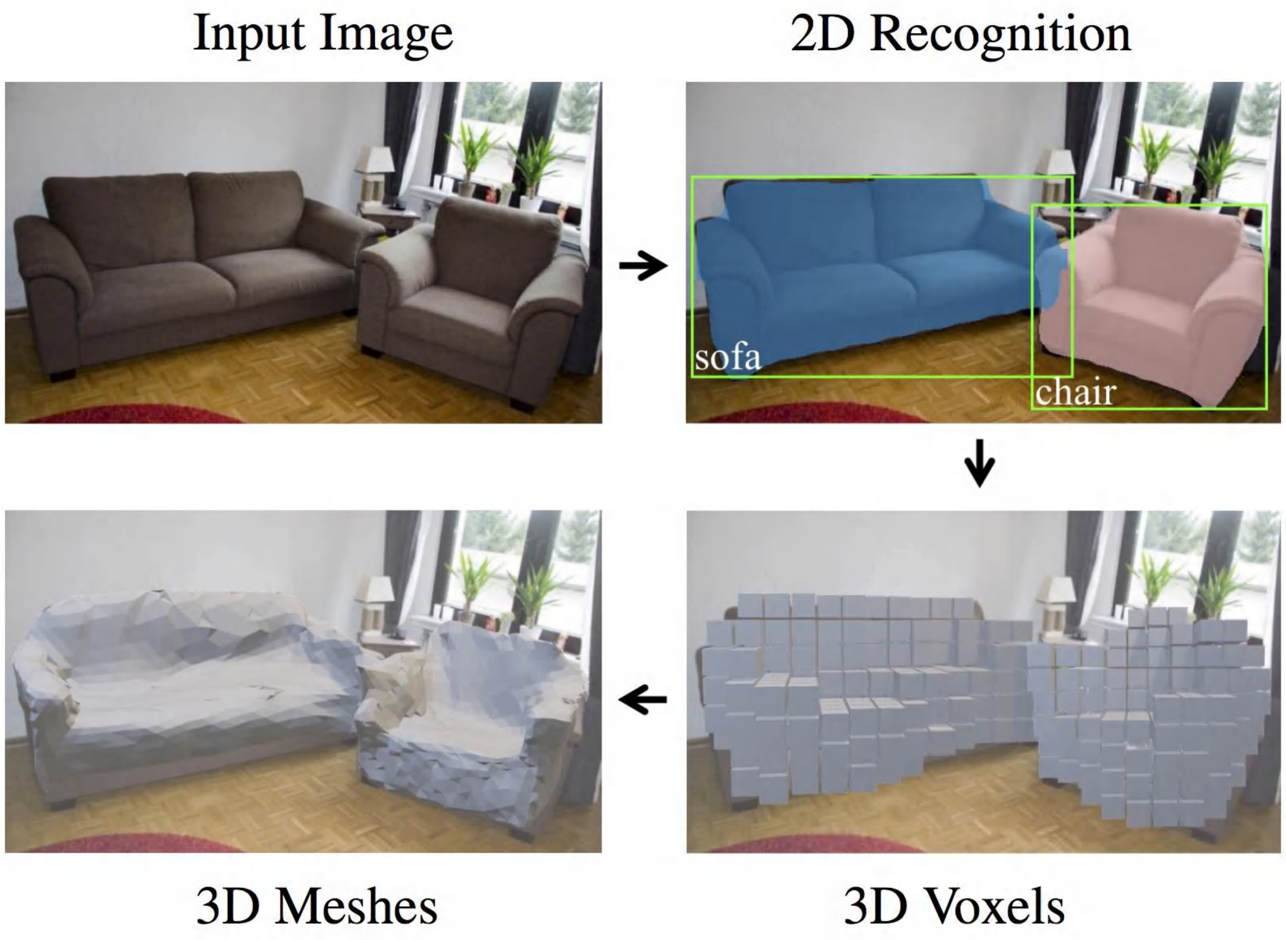
Beyond Boxes and Masks: Human Surfaces

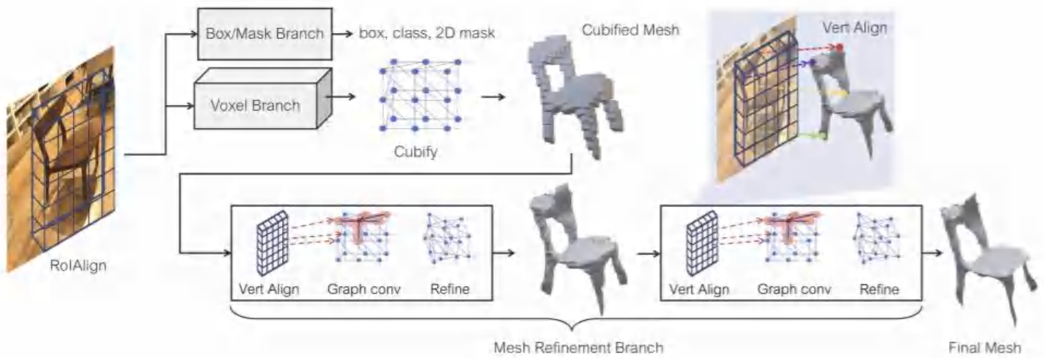
DensePose: Dense Human Pose Estimation In The Wild
[Güler, Neverova, Kokkinos CVPR 2018]

Beyond Boxes and Masks: Panoptic Segmentation

Panoptic Segmentation: task and methods

Beyond Boxes and Masks: 3D Shape





Mesh R-CNN
[Gkioxari, Malik, Johnson arXiv 2019]

Overview of this Tutorial

Topics to cover

- ➢ Object detection intro (very brief)
- ➢ The Generalized R-CNN framework (presented as a sequence of “hammers”)
- ➢ Open challenges in object detection

Not in This Tutorial – *A Whole Lot!*

- One-stage detection methods
  - Anchor based (e.g., SSD, YOLOv2/3, RetinaNet)
  - Point based (e.g., CornerNet, CenterNet)

- Numerous extensions in Generalized R-CNN family
  - e.g., Cascade R-CNN, Mesh R-CNN

R-CNN

References

- B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. TPAMI, 2012.
- I. Endres and D. Hoiem. Category independent object proposals. In ECCV, 2010.
- J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. IJCV, 2013.
- A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012.
- X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In ICCV, 2013.
- R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

R-CNN Today

Conceptual basis for modern proposal-based detectors

Methodology for state-of-the-art human pose estimation

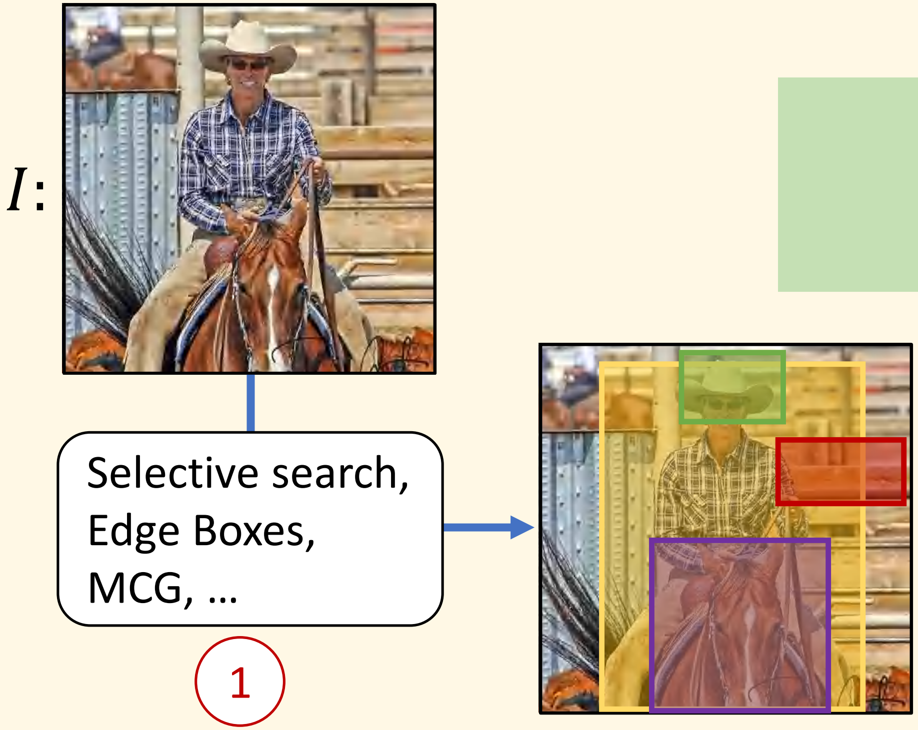
- E.g.,: B. Xiao, H. Wu, Y. Wei. Simple Baselines for Human Pose Estimation and Tracking. ECCV 2018.

Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN (Region-based Convolutional Neural Net)

Per-image computation

Per-region computation



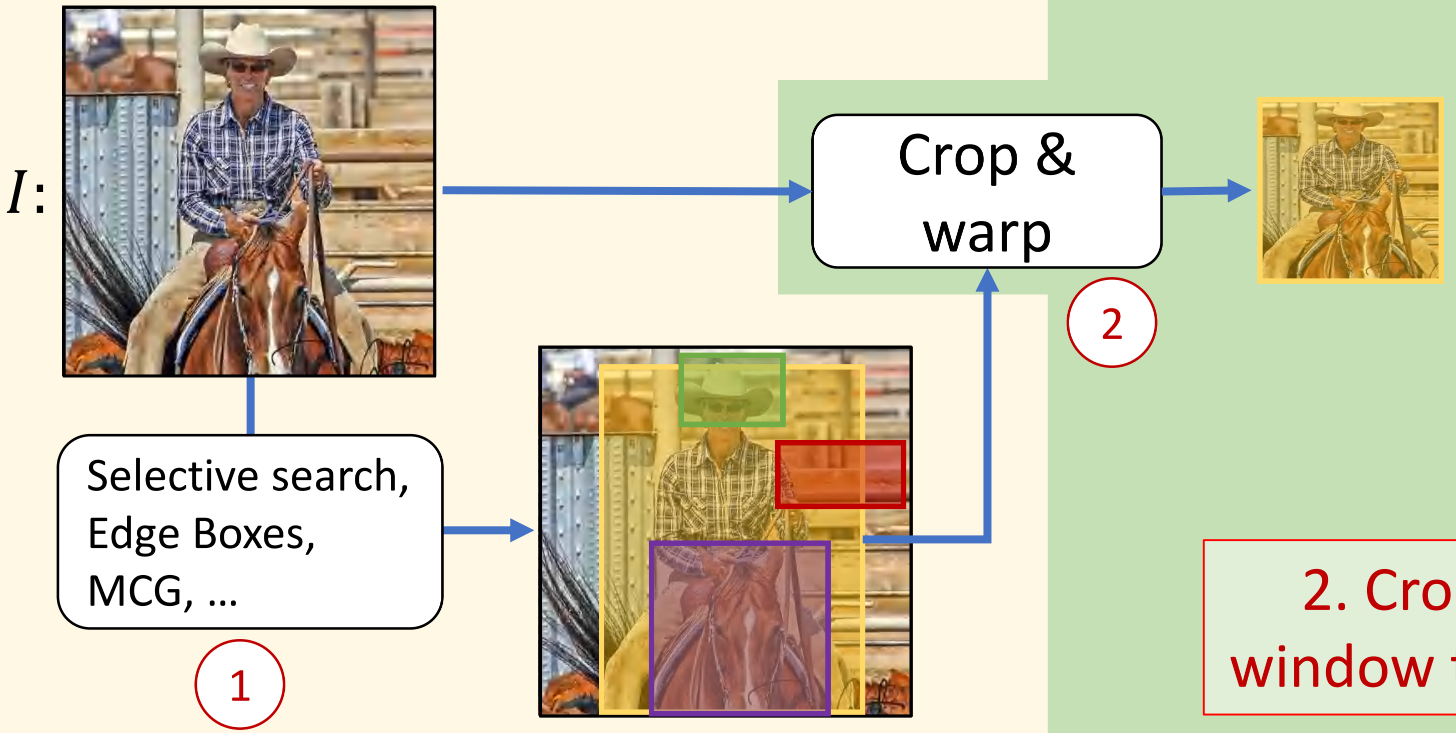
1. Use an off-the-shelf *Region of Interest* (RoI) proposal algorithm (~2k proposals per image)

Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN

Per-image computation

Per-region computation for each $r_i \in r(I)$

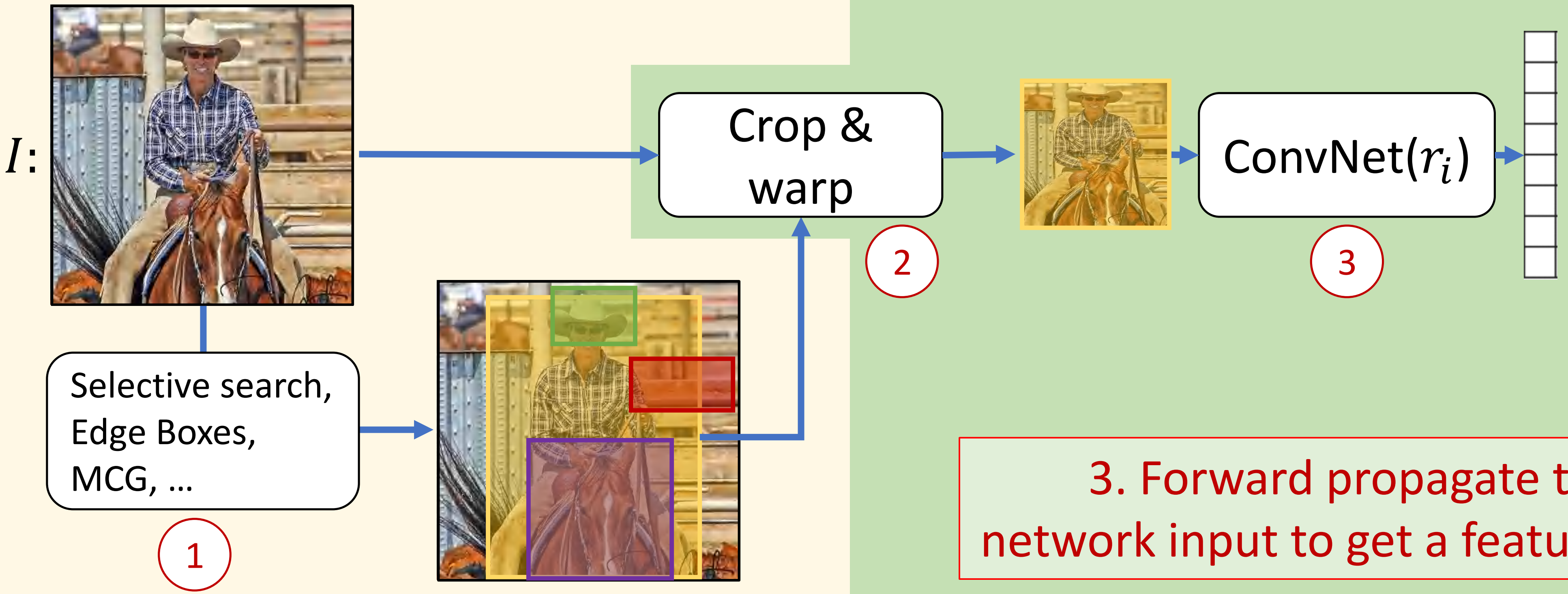


2. Crop and warp each proposal image window to obtain a fixed-size network input

R-CNN

Per-image computation

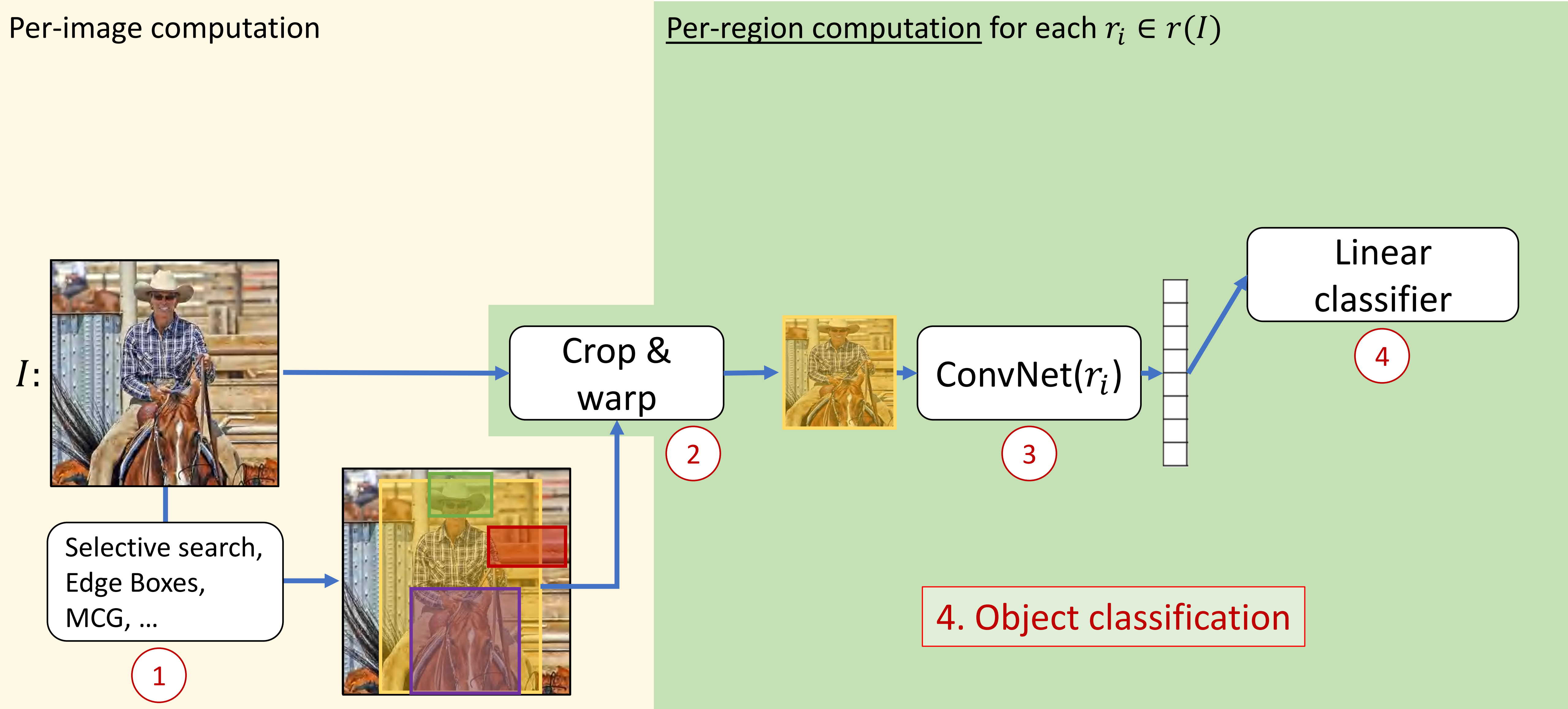
Per-region computation for each $r_i \in r(I)$



3. Forward propagate the fixed-size network input to get a feature representation

Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN

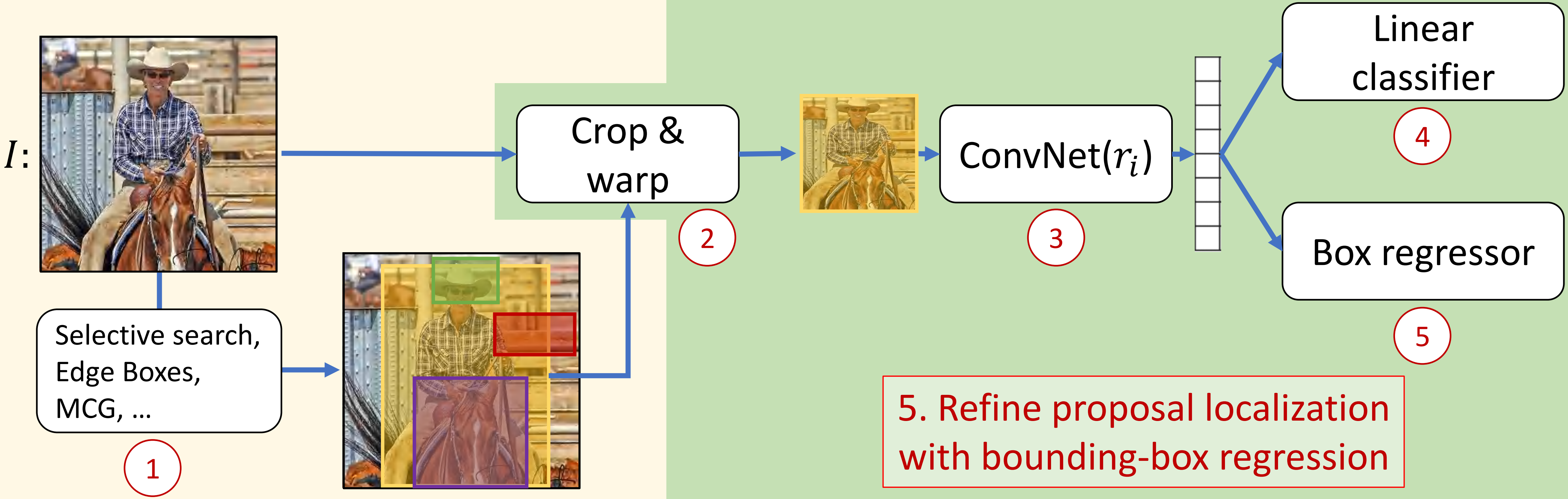


Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN

Per-image computation

Per-region computation for each $r_i \in r(I)$



Generalized R-CNN Framework

A common framework for understanding

- R-CNN
- Fast R-CNN
- Faster R-CNN
- Feature Pyramid Networks (FPN) + Faster R-CNN
- Mask R-CNN
- … and more (e.g., Cascade R-CNN, DensePose, Mesh R-CNN)

https://github.com/facebookresearch/detectron

detectron2 – in PyTorch – is coming later this year

Generalized R-CNN Framework

Per-image computation

Per-region computation for each $r_i \in r(I)$

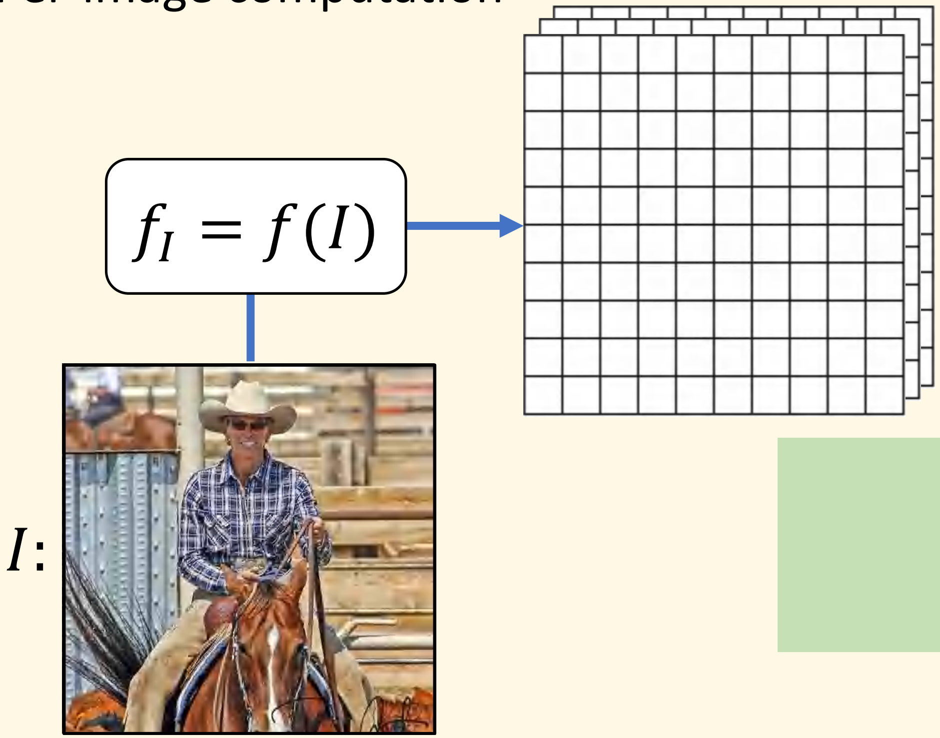
I:

Input image

per-image operations | per-region operations

Generalized R-CNN Framework

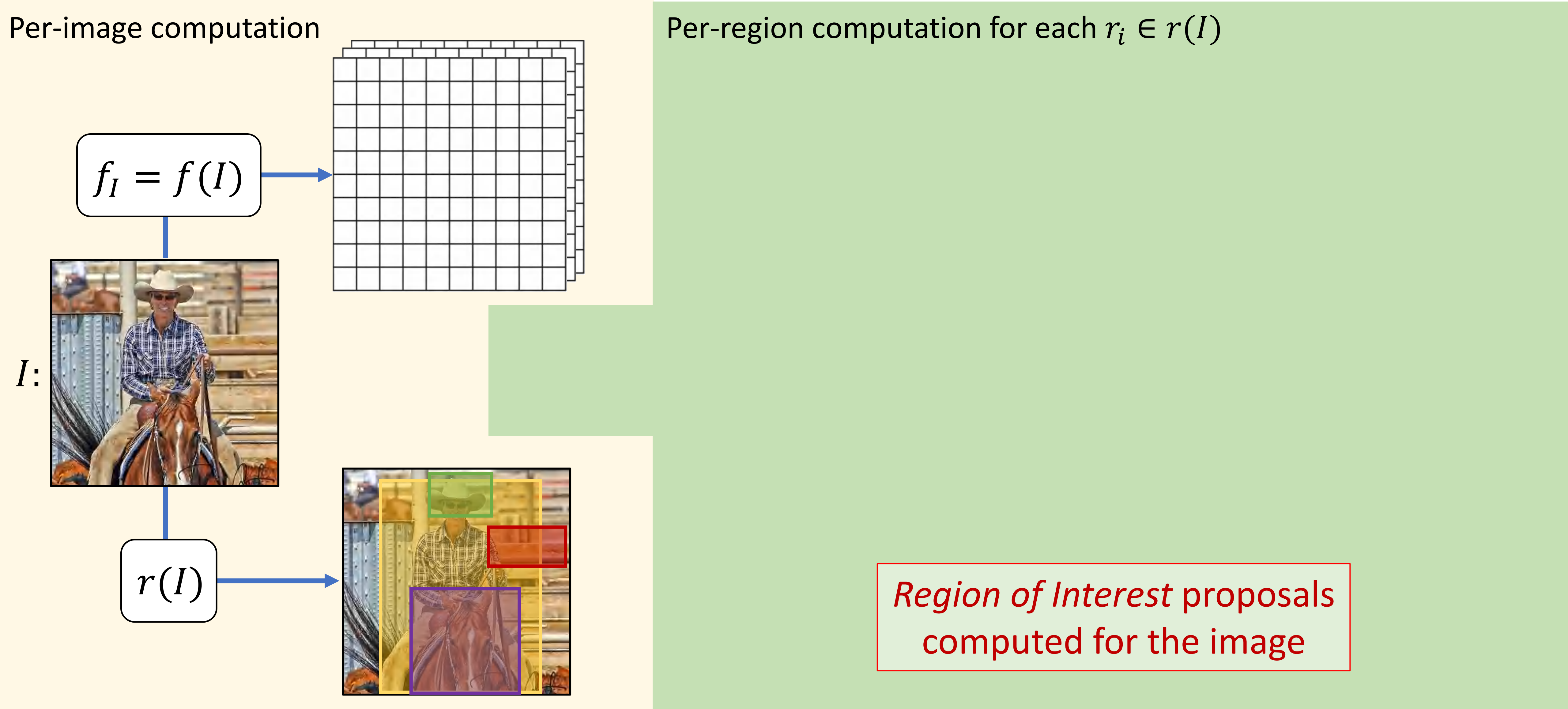
Per-image computation



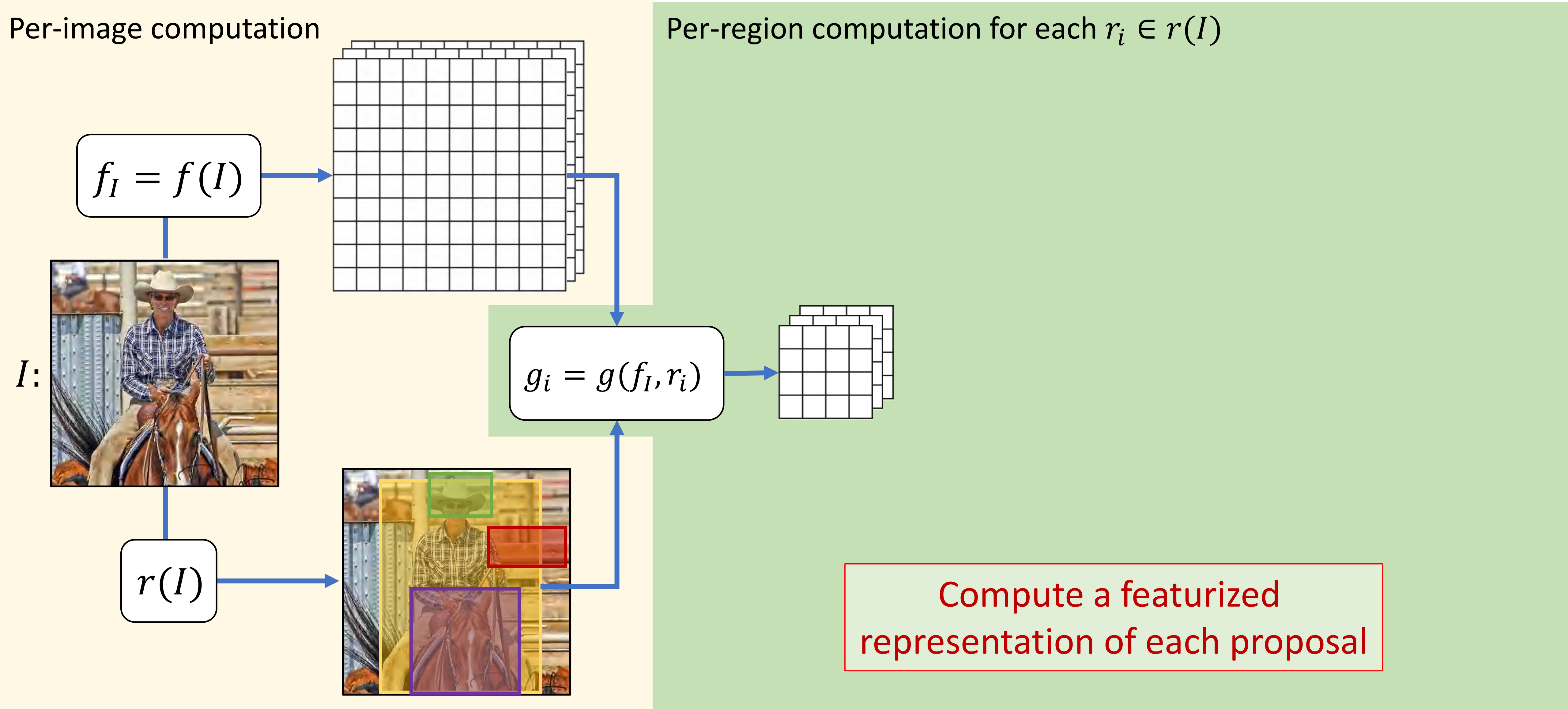
Per-region computation for each $r_i \in r(I)$

Transformation of the input image into a featurized representation

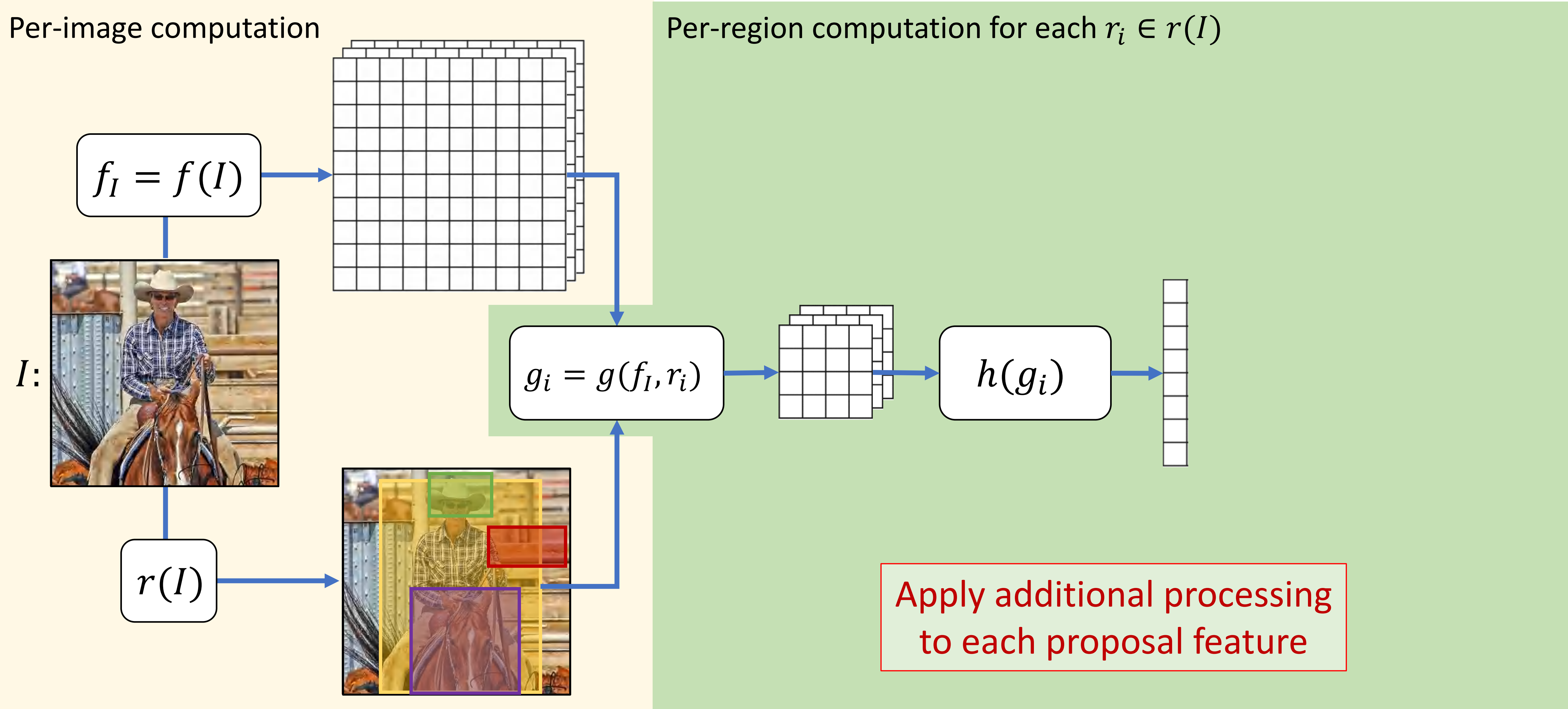
Generalized R-CNN Framework



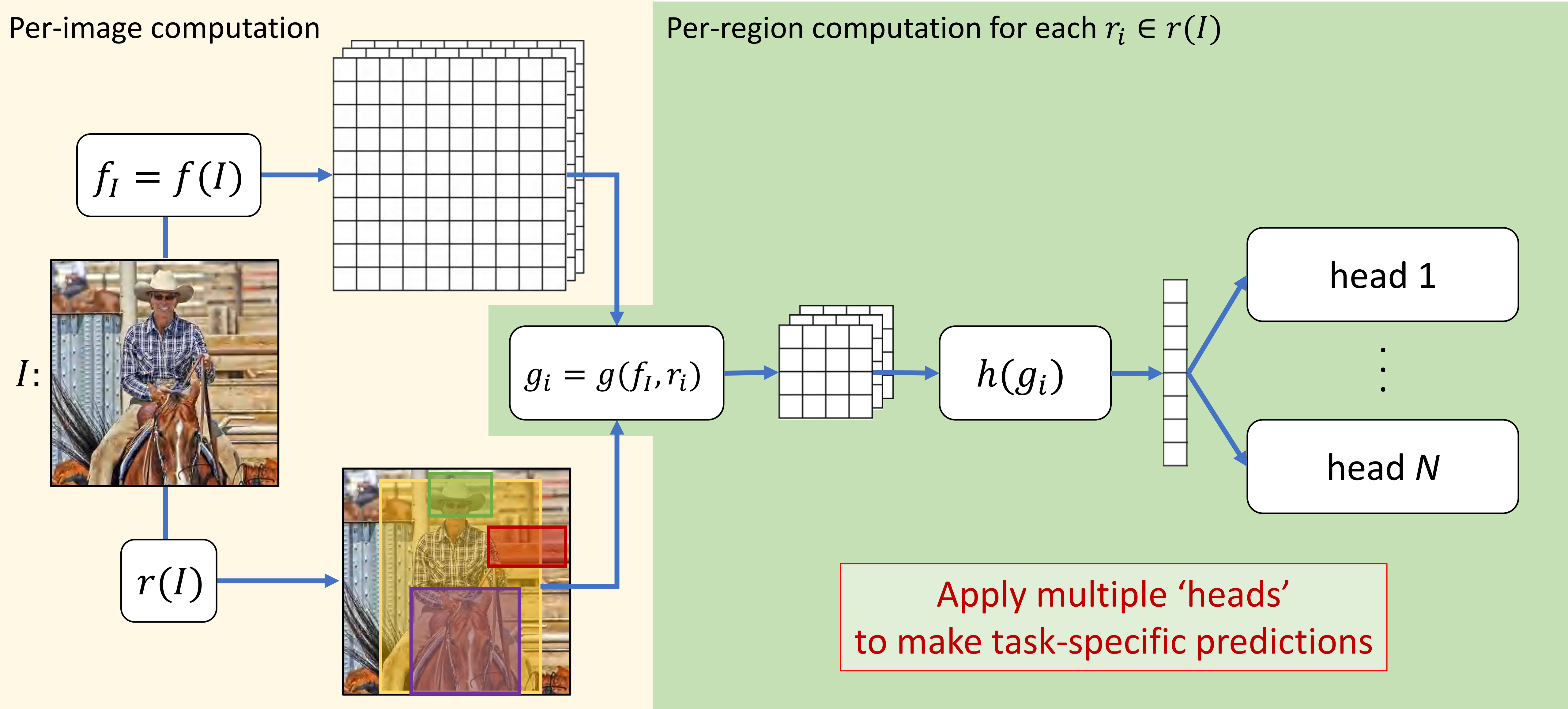
Generalized R-CNN Framework



Generalized R-CNN Framework

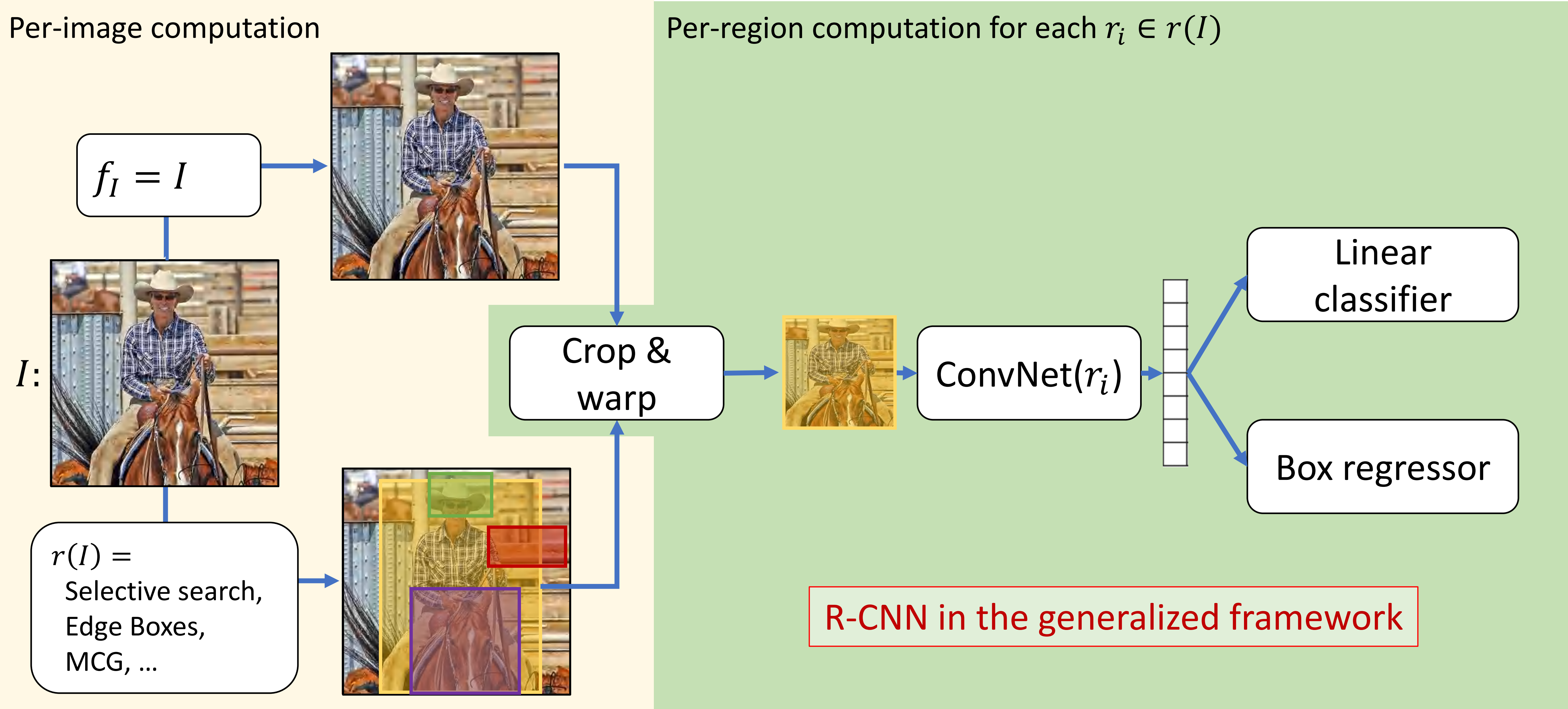


Generalized R-CNN Framework



What Does R-CNN Look Like in the Generalized Framework?

R-CNN in the Generalized Framework



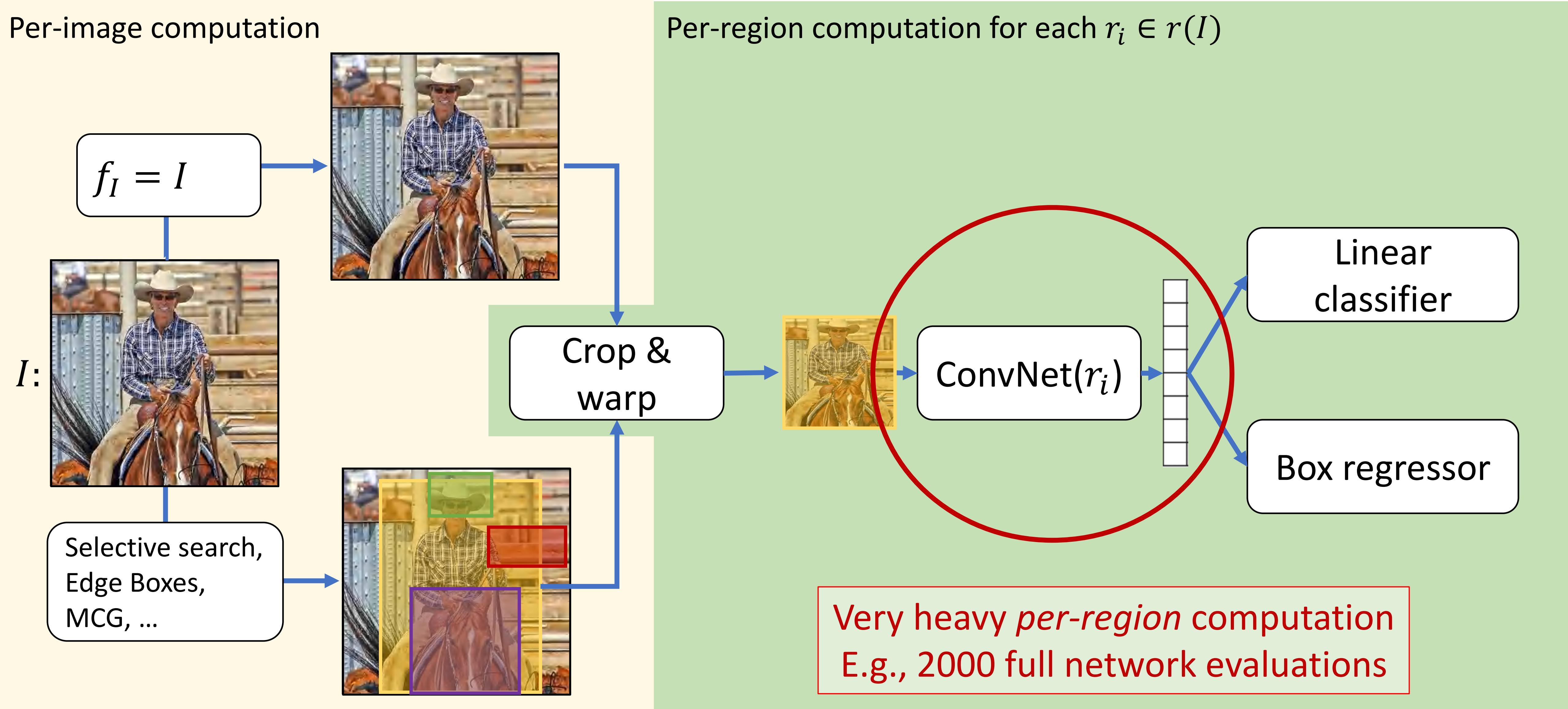
The Problem with R-CNN

Building
a better
hammer

R-CNN

?

"Slow" R-CNN



Very heavy *per-region* computation
E.g., 2000 full network evaluations

Fast R-CNN

References

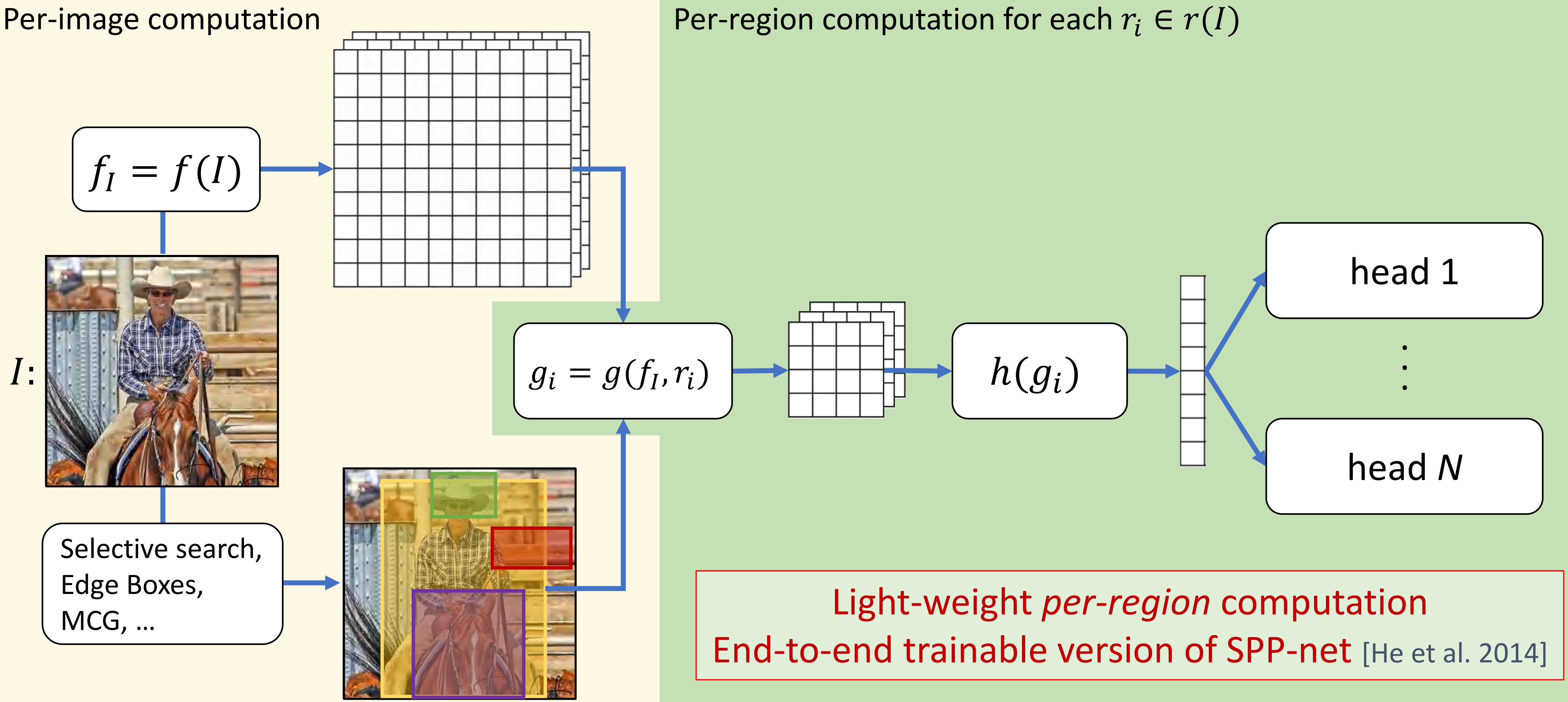
- K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014.
- R. Girshick. Fast R-CNN. In ICCV, 2015.

Building
a better
hammer

R-CNN

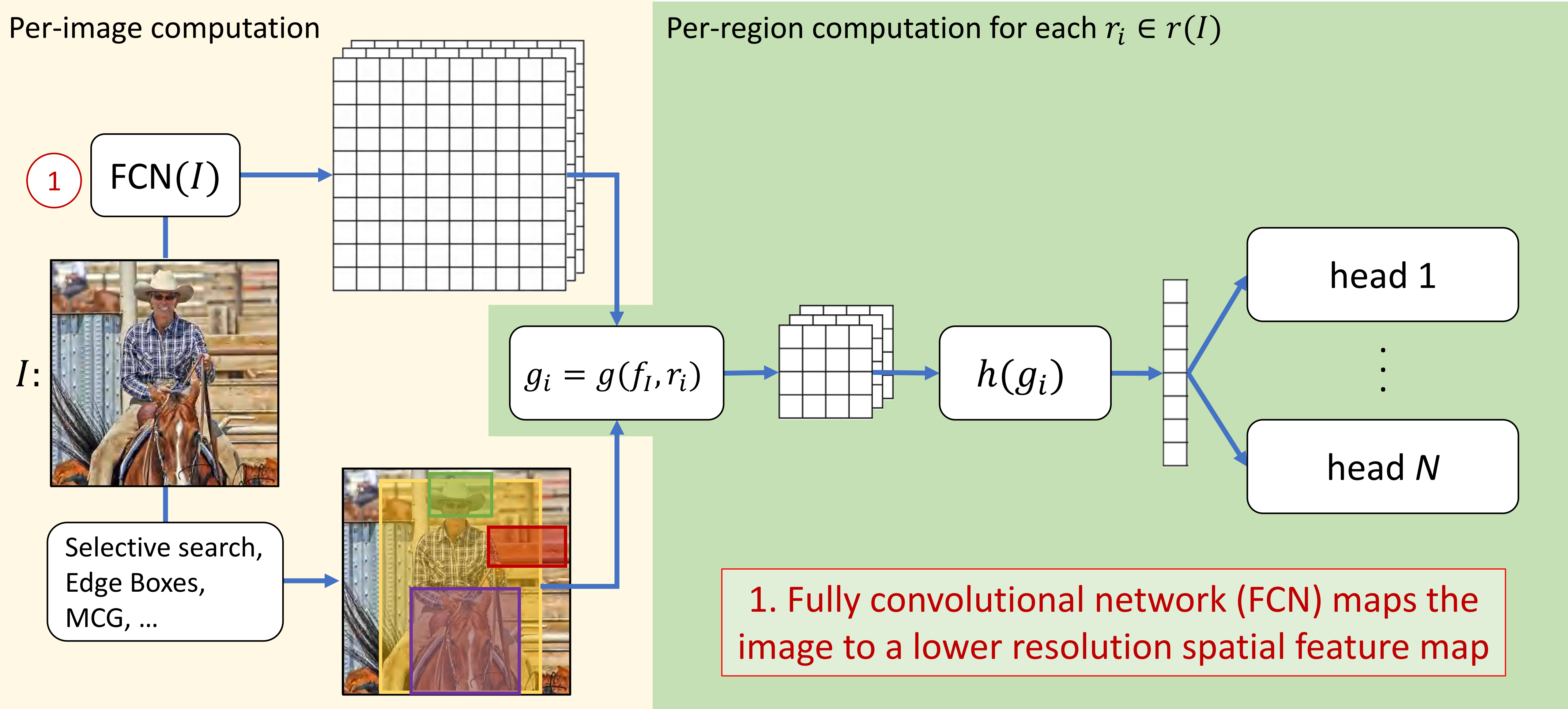
Fast R-CNN

Generalized R-CNN → Fast R-CNN



Girshick. Fast R-CNN. ICCV 2015.

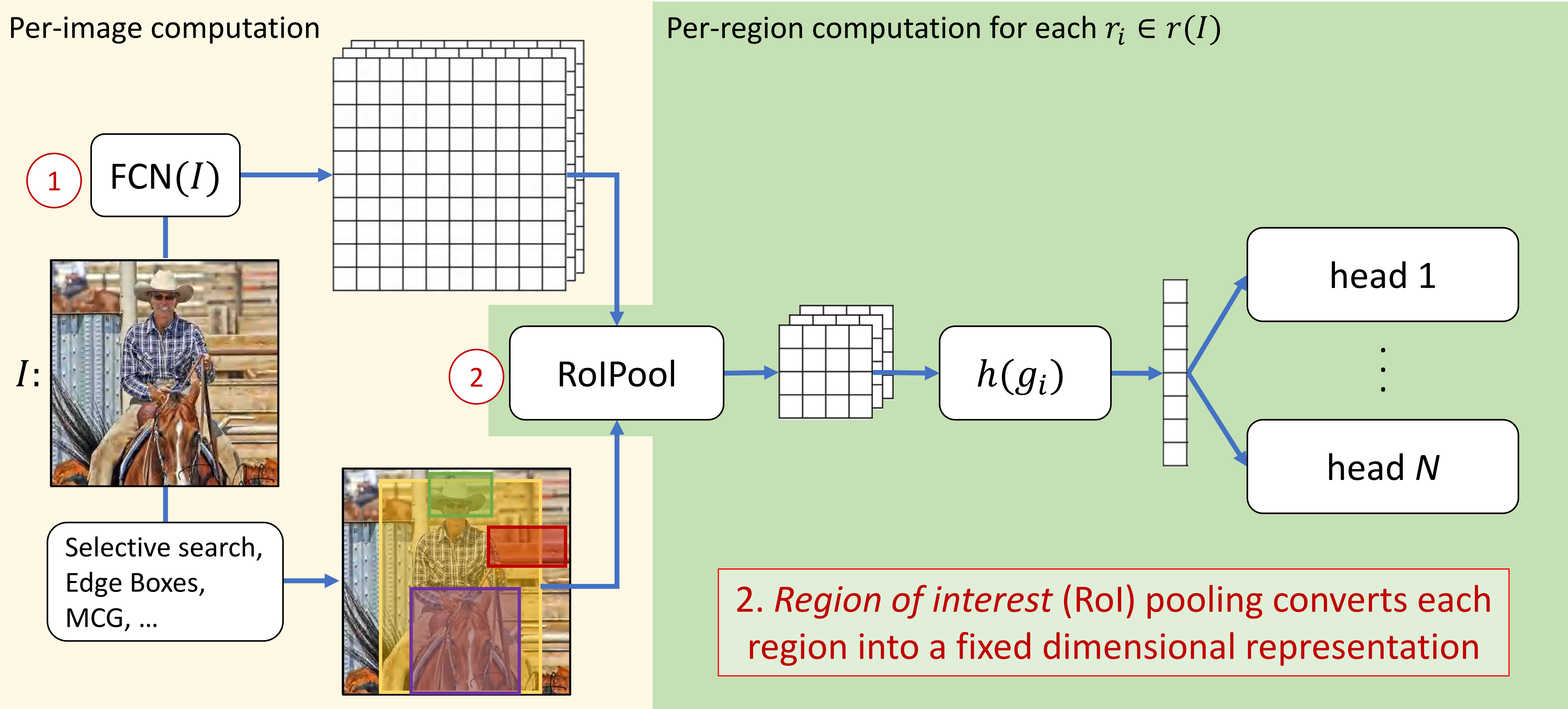
Generalized R-CNN → Fast R-CNN



1. Fully convolutional network (FCN) maps the image to a lower resolution spatial feature map

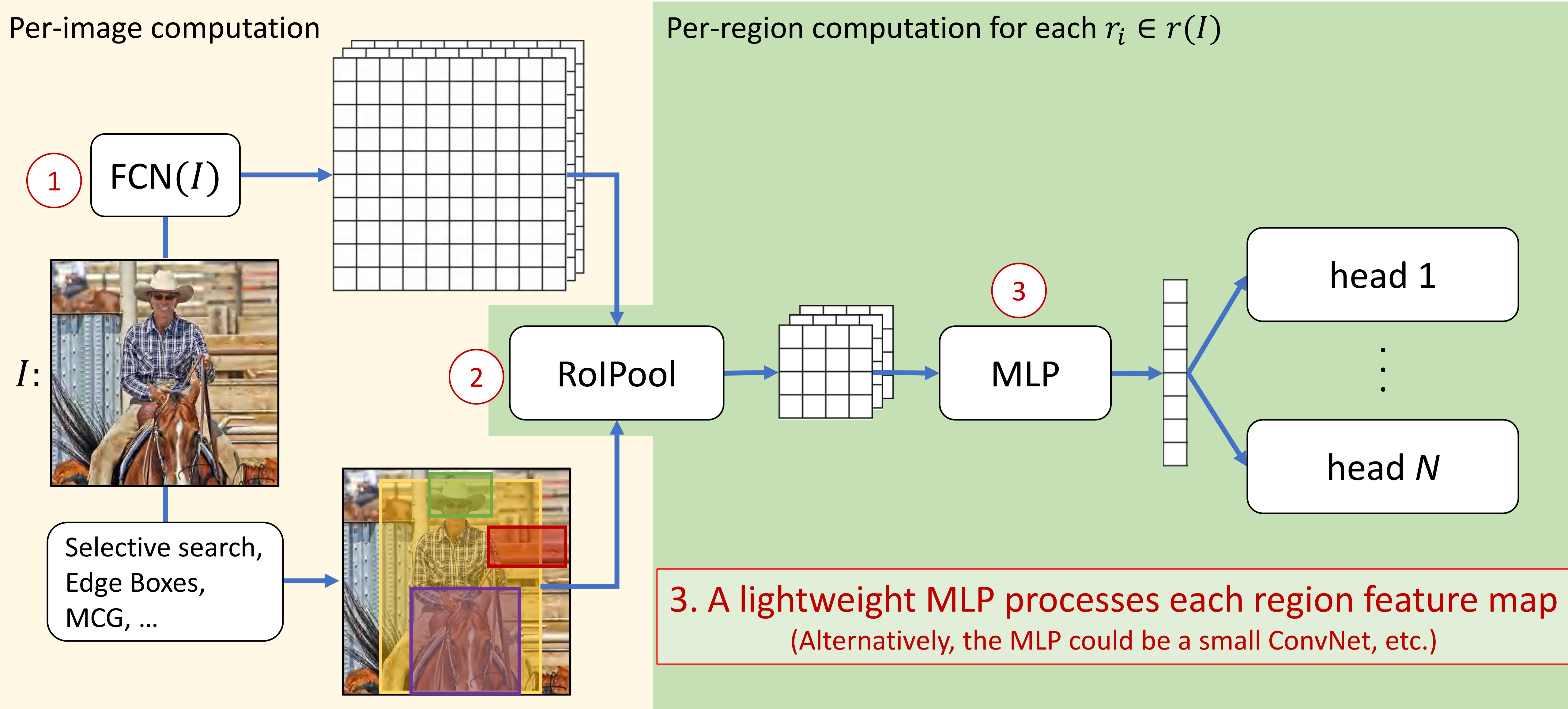
Girshick. Fast R-CNN. ICCV 2015.

Generalized R-CNN → Fast R-CNN



2. *Region of interest* (RoI) pooling converts each region into a fixed dimensional representation

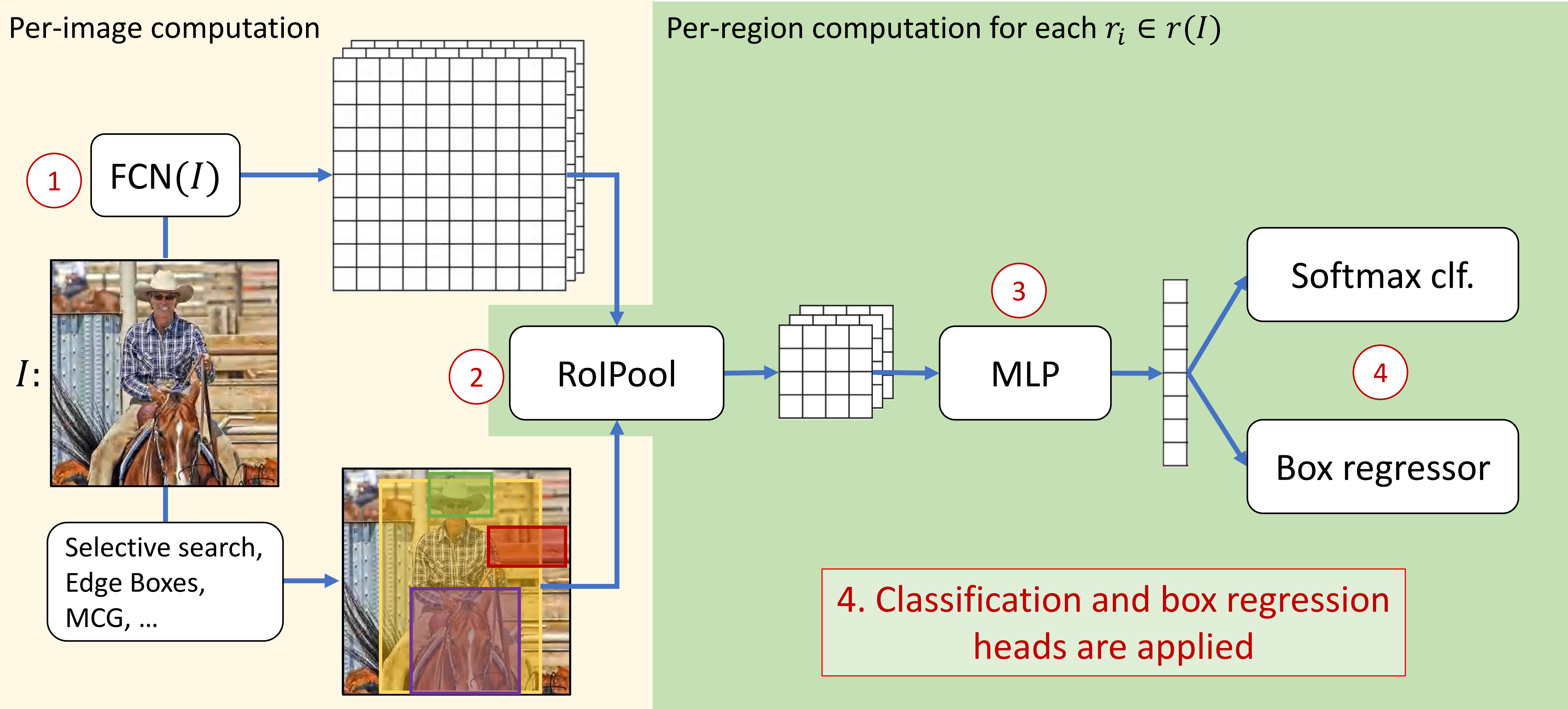
Generalized R-CNN → Fast R-CNN



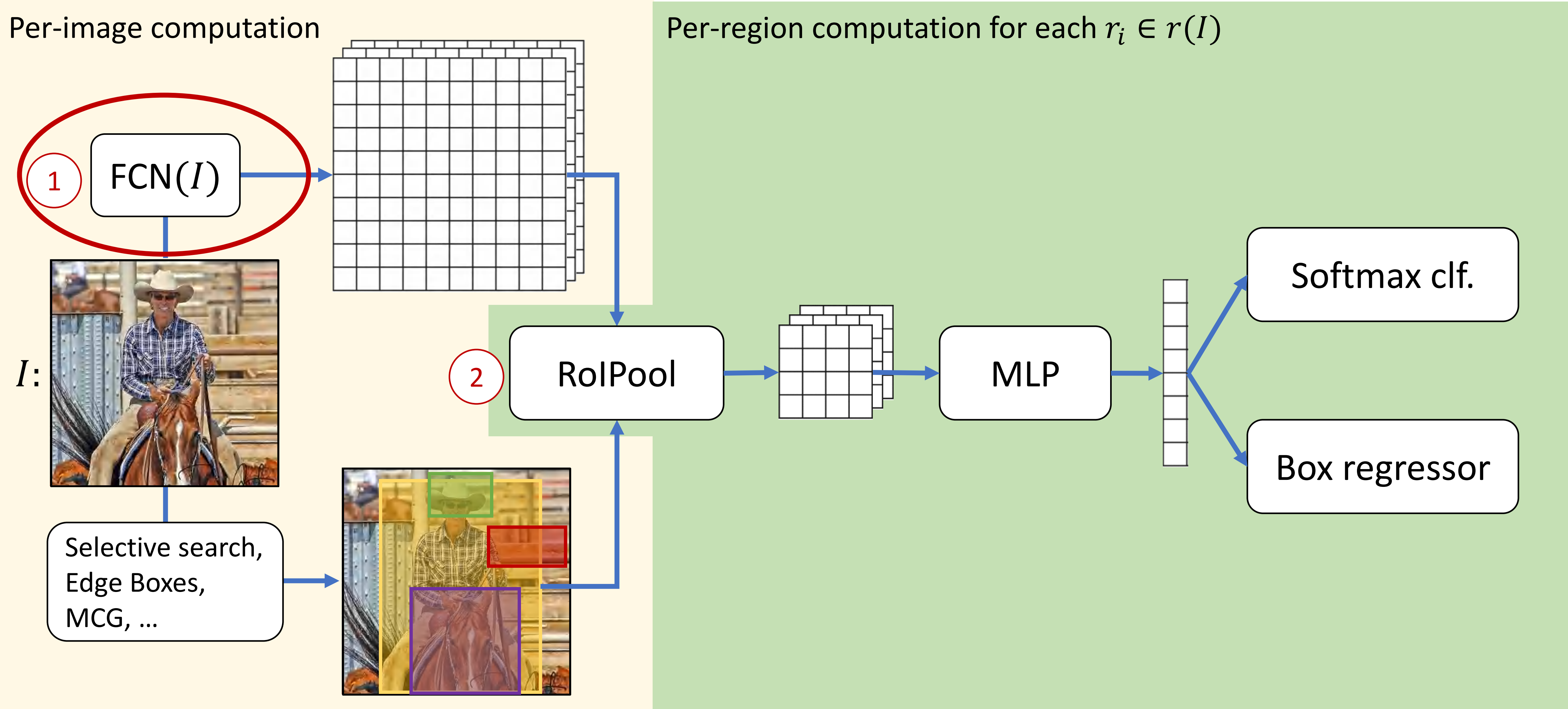
3. A lightweight MLP processes each region feature map
(Alternatively, the MLP could be a small ConvNet, etc.)

Girshick. Fast R-CNN. ICCV 2015.

Generalized R-CNN → Fast R-CNN

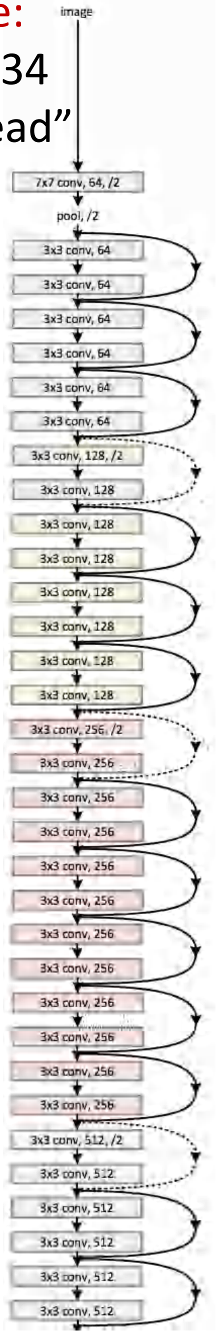


Fast R-CNN



Per-image computation
Per-region computation for each $r_i \in r(I)$
1
FCN(I)
I:
Selective search,
Edge Boxes,
MCG, …
2
RoIPool
MLP
Softmax clf.
Box regressor

Example:
ResNet-34
w/o “head”



Whole-image, Fully Conv. Network (FCN)

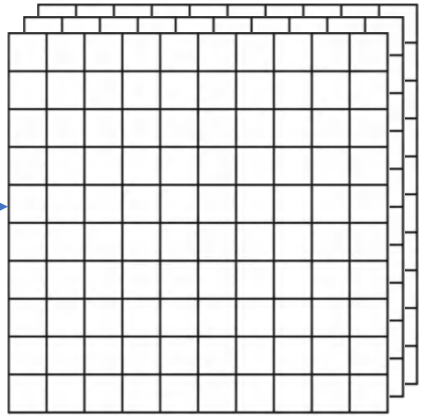
Use any standard ConvNet as the “backbone architecture”

- AlexNet, VGG, ResNet, Inception, Inception-ResNet, ResNeXt, DenseNet, NAS*, …
- Remove global pooling / FC
- Output spatial dims are proportional to input spatial dims

I: H

W

$\longrightarrow$ $\boxed{\mathrm{FCN}(I)}$ $\longrightarrow$



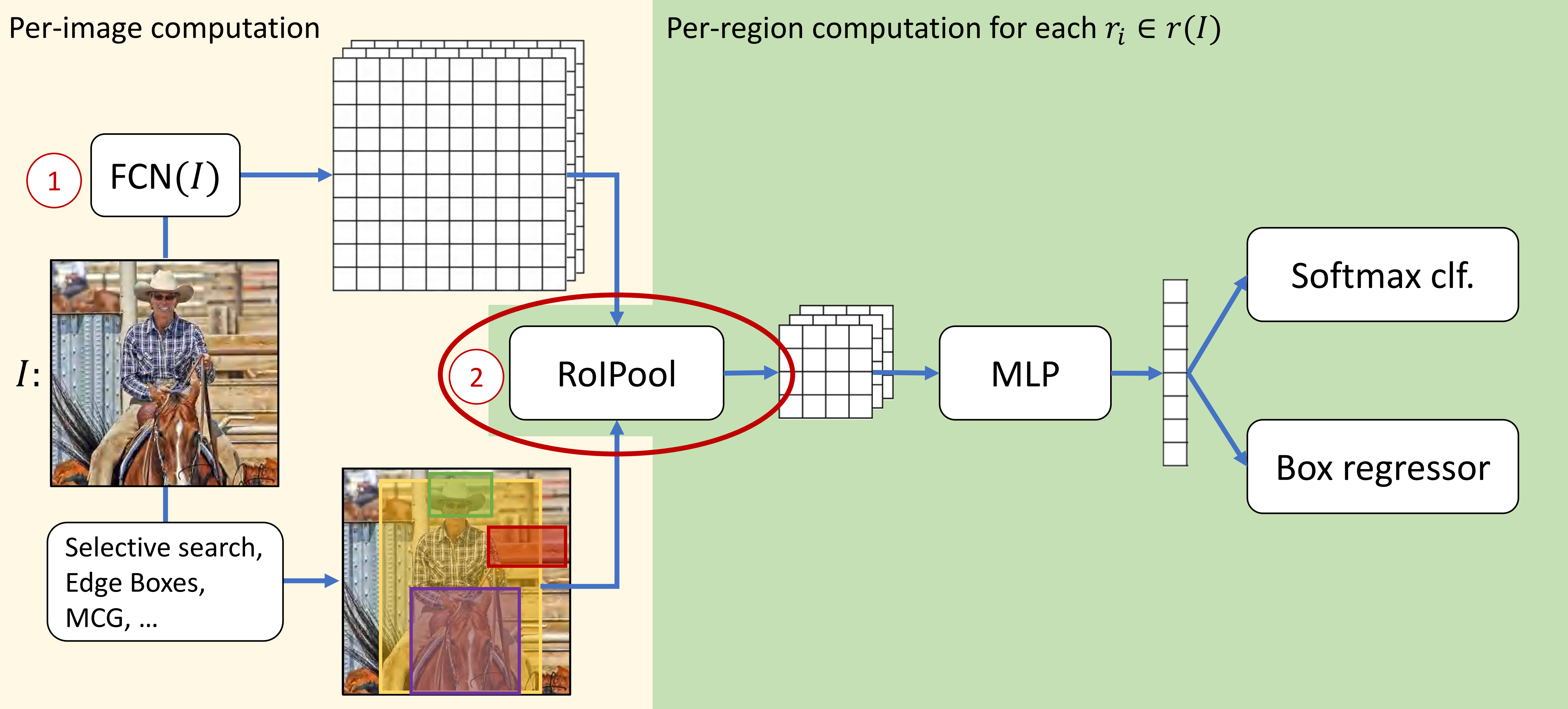
Example feature map dimensions:
(512, H/16, W/16)

The Engine of Recognition

A good network lift all boats (“features matter”)

- AlexNet
- VGG
- GoogleNet / Inception
- ResNet
- ResNeXt
- NAS*
- …

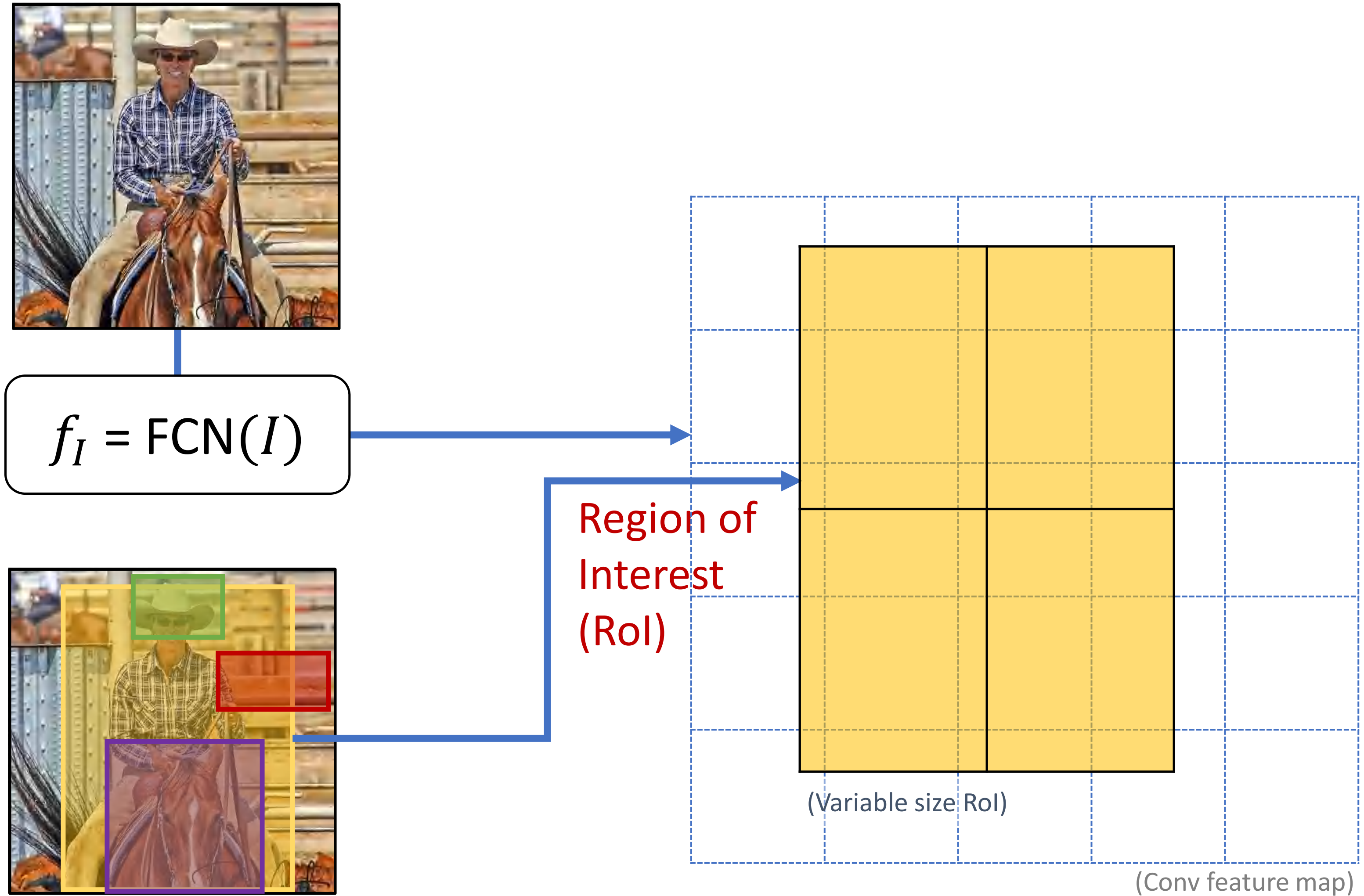
Fast R-CNN



Per-image computation
Per-region computation for each $r_i \in r(I)$
1
FCN(I)
I:
Selective search,
Edge Boxes,
MCG, …
2
RoIPool
MLP
Softmax clf.
Box regressor

He, Zhang, Ren, Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. ECCV 2014.

RoIPool Operation (on each Proposal)

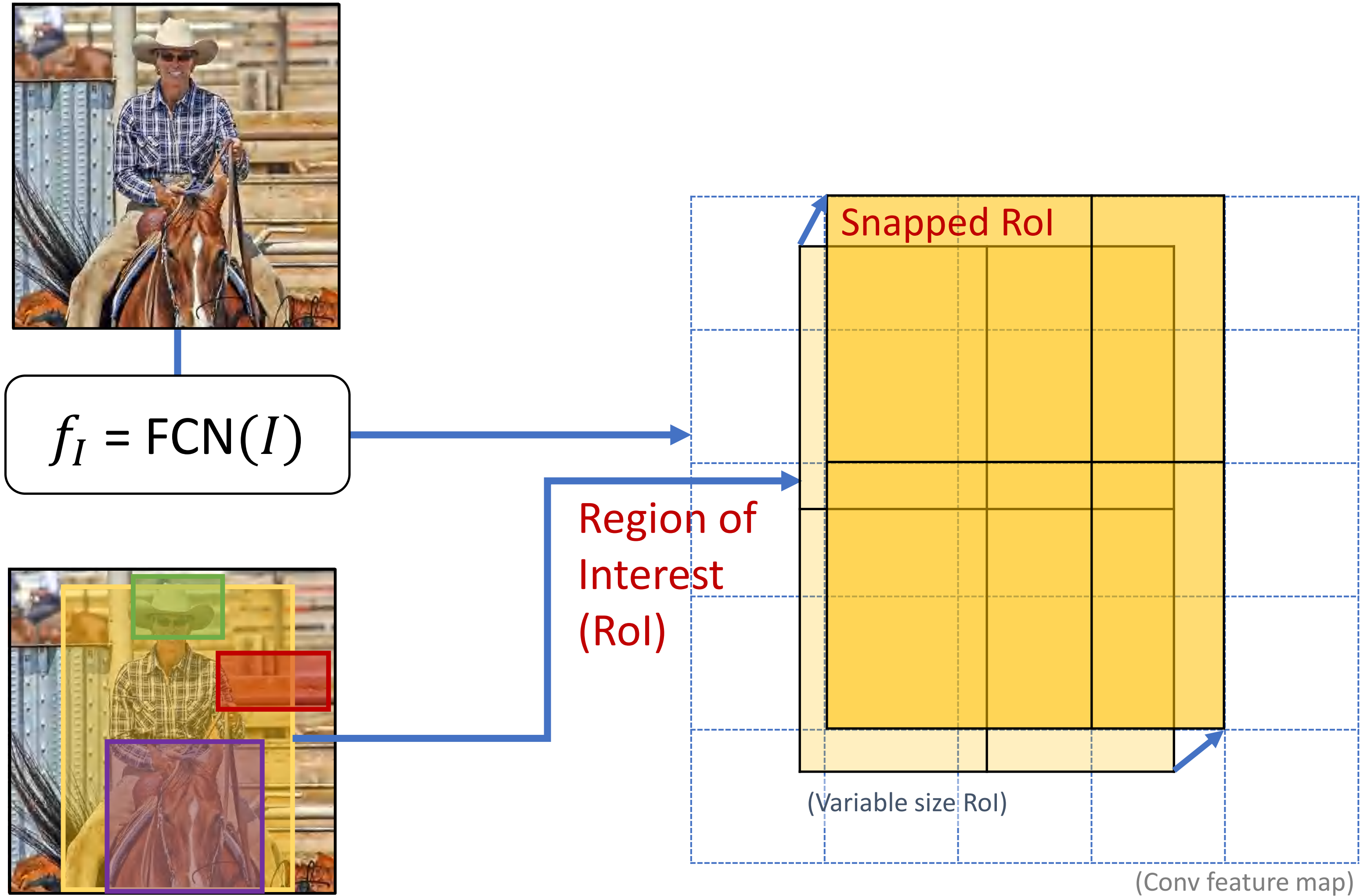


Key innovation in SPP-net

[He et al. 2014]

He, Zhang, Ren, Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. ECCV 2014.

RoIPool Operation (on each Proposal)



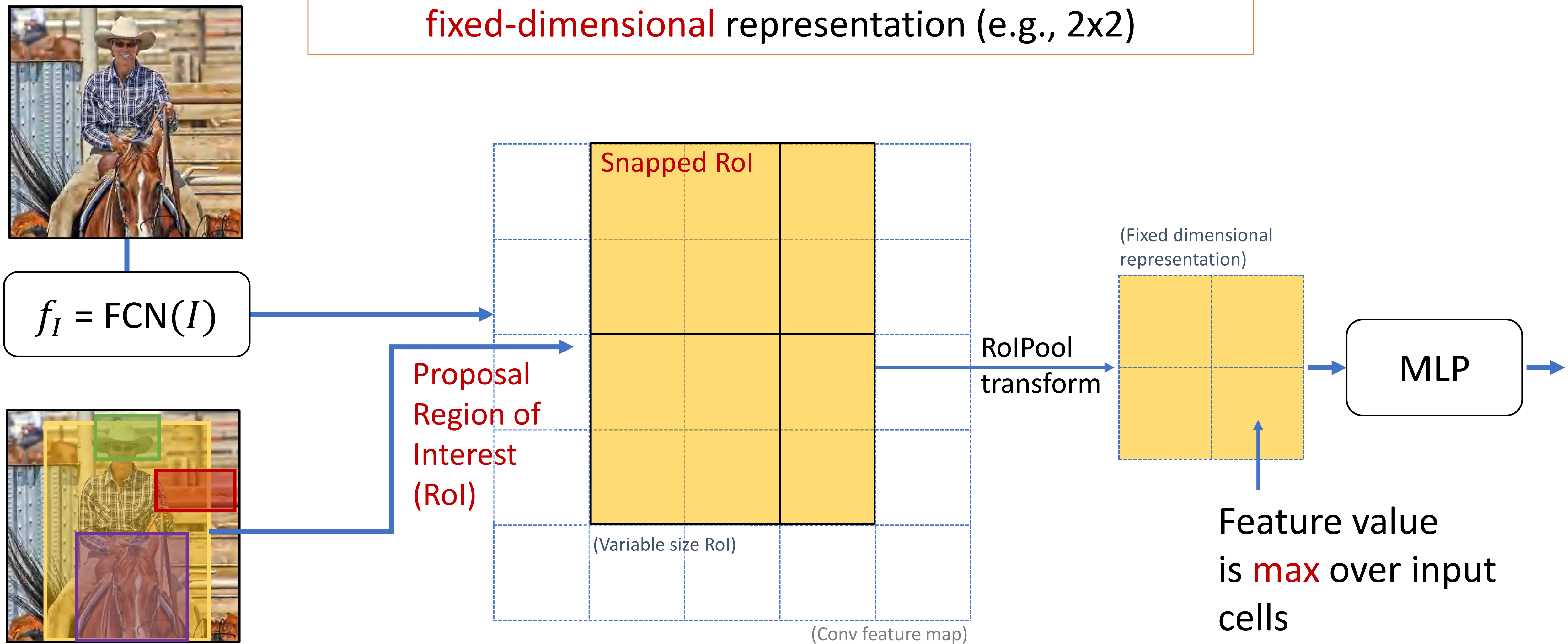
Key innovation in SPP-net

[He et al. 2014]

He, Zhang, Ren, Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. ECCV 2014.

RoIPool Operation (on each Proposal)

Transform arbitrary size proposal into a fixed-dimensional representation (e.g., 2x2)



The Problem with Fast R-CNN

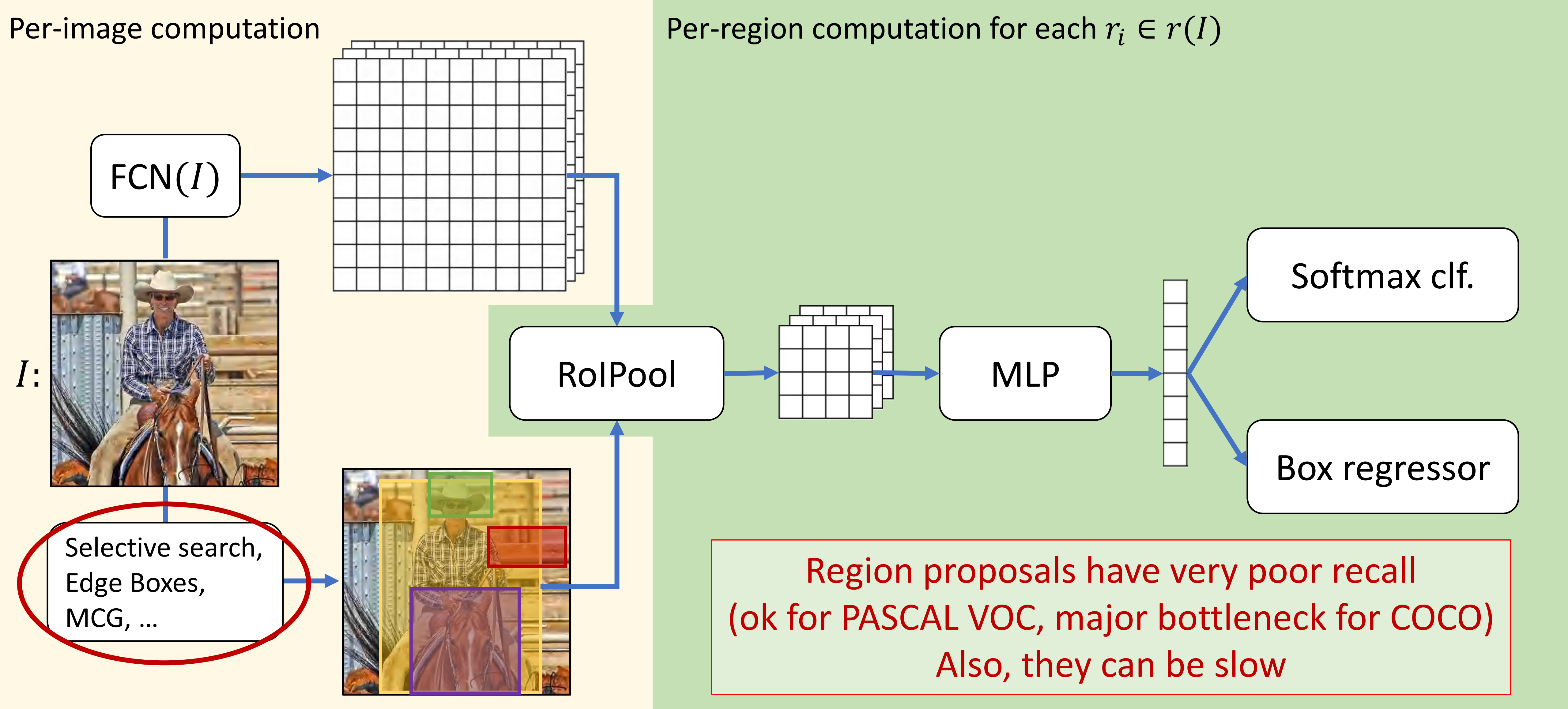
R-CNN

Fast R-CNN

Building
a better
hammer

?

The Problem with Fast R-CNN



Region proposals have very poor recall
(ok for PASCAL VOC, major bottleneck for COCO)
Also, they can be slow

Faster R-CNN

References

- D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In CVPR, 2014.
- P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In NIPS, 2015.
- S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

Building a better hammer

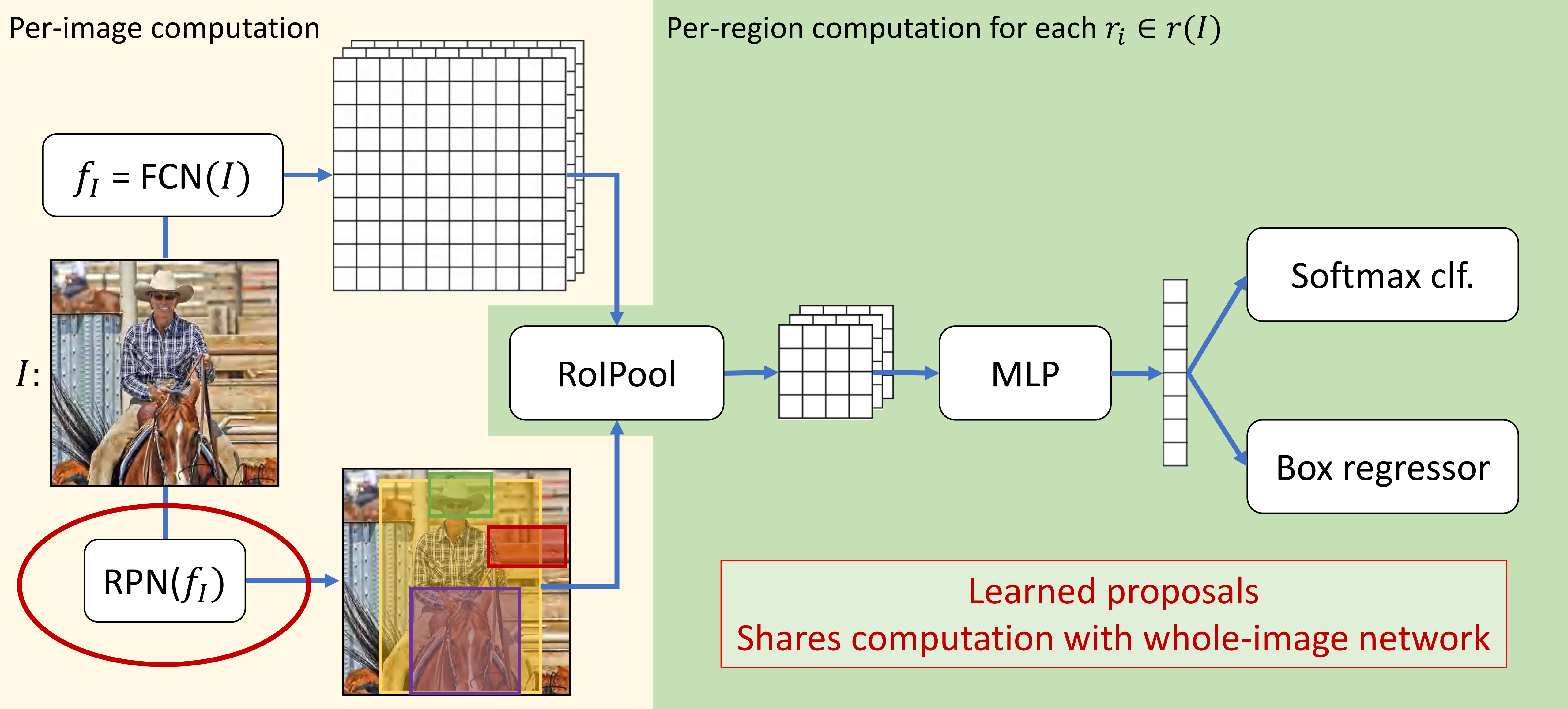
R-CNN

Fast R-CNN

Faster R-CNN

Ren, He, Girshick, Sun. Faster R-CNN: Towards Real-Time Object Detection. NIPS 2015.

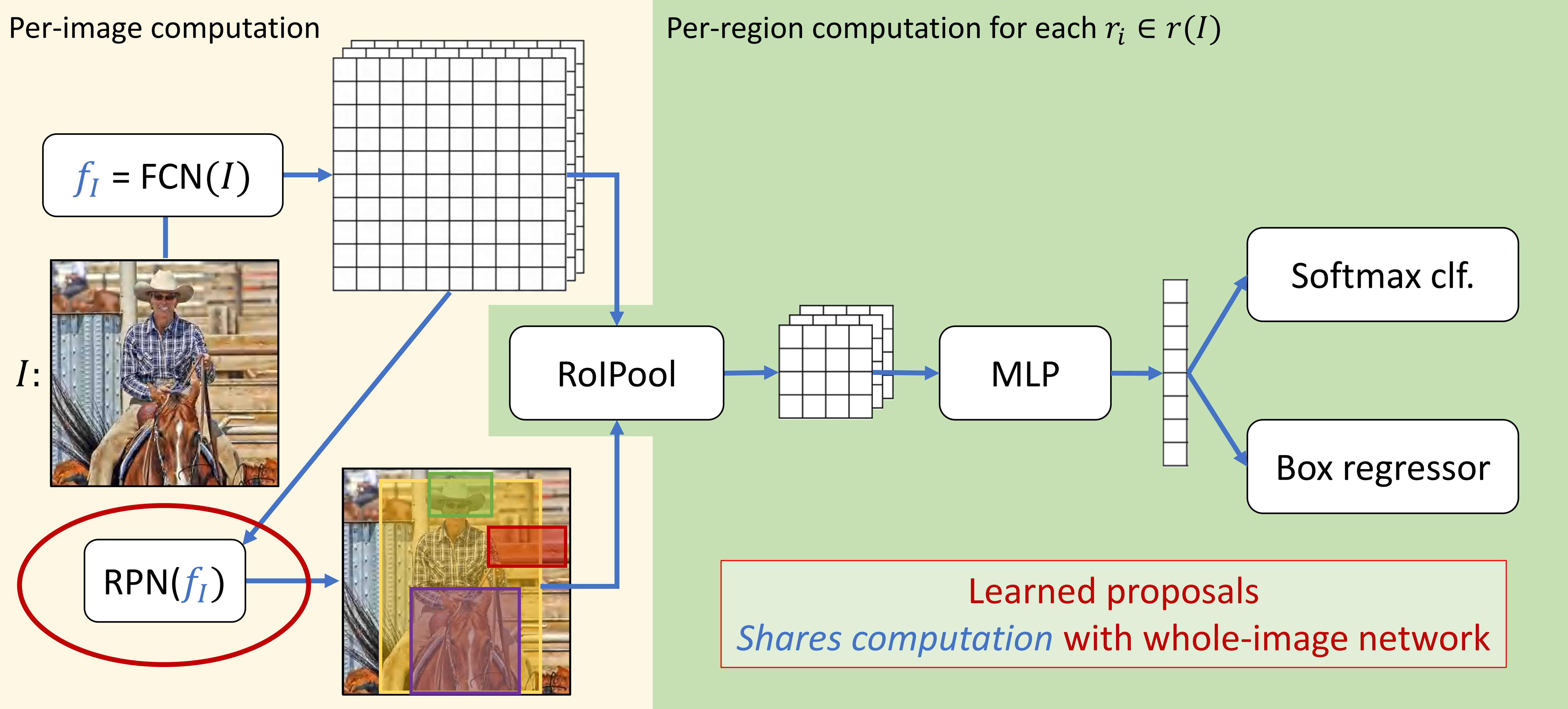
Faster R-CNN



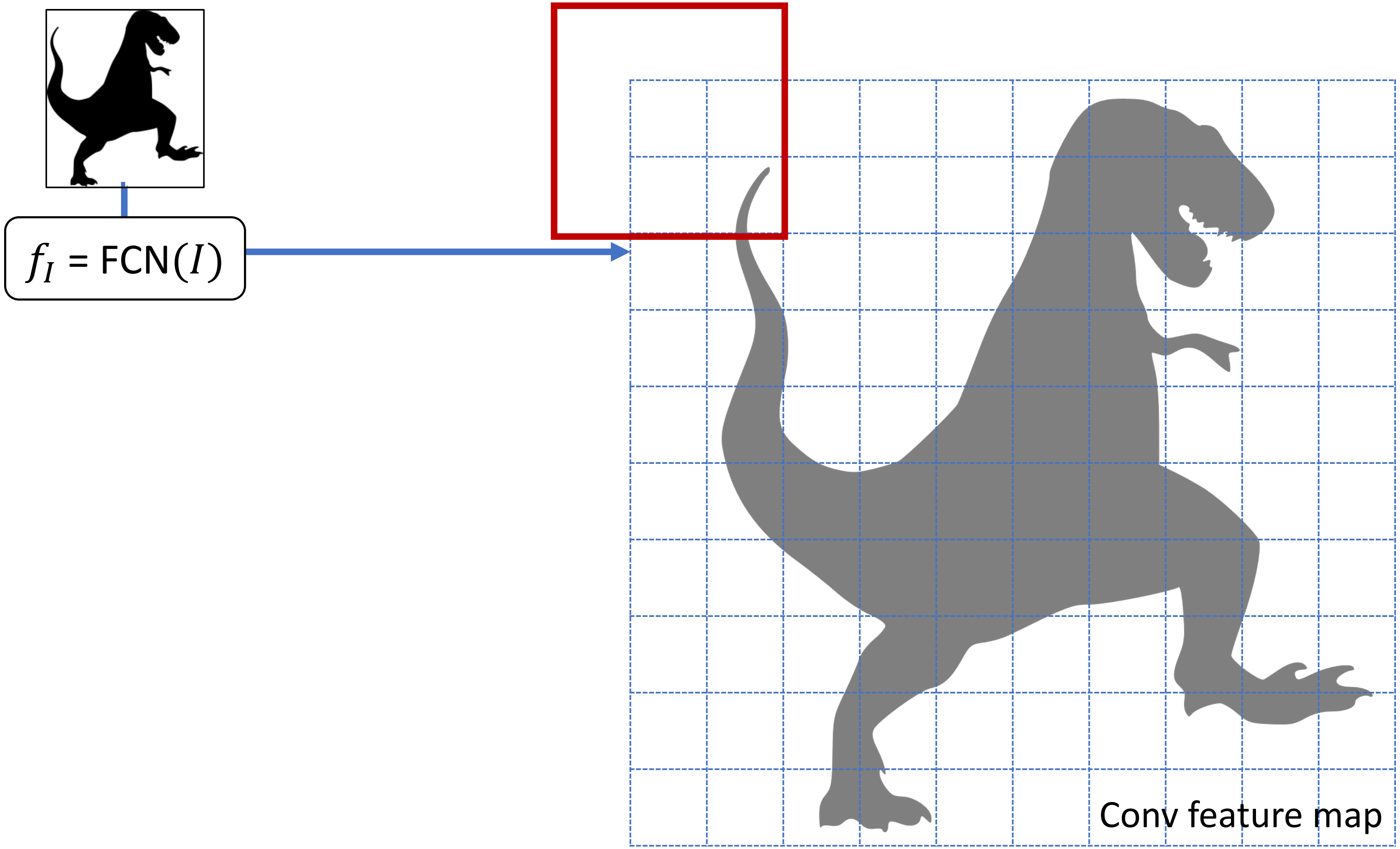
Learned proposals
Shares computation with whole-image network

Ren, He, Girshick, Sun. Faster R-CNN: Towards Real-Time Object Detection. NIPS 2015.

Faster R-CNN



RPN: Region Proposal Network

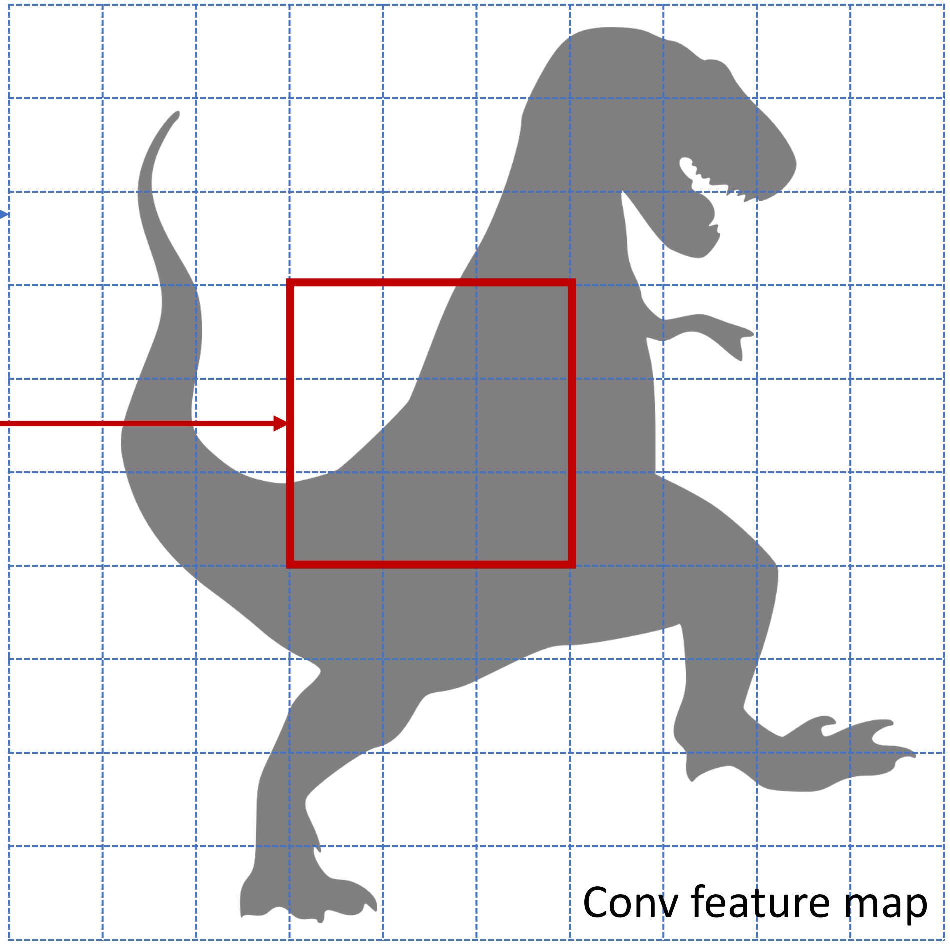


RPN: Region Proposal Network

f_I = FCN(I)

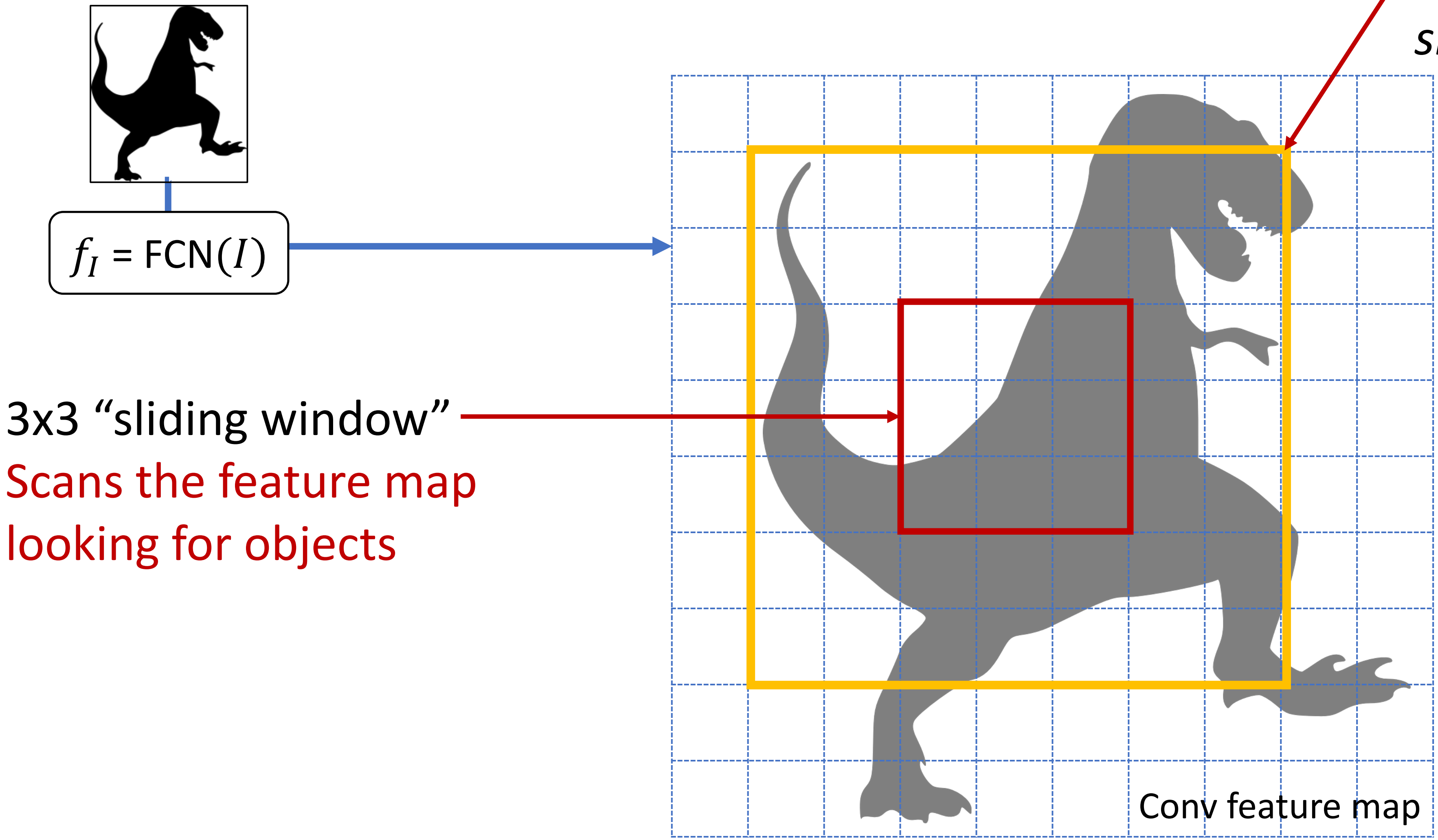
3x3 “sliding window”

Scans the feature map looking for objects



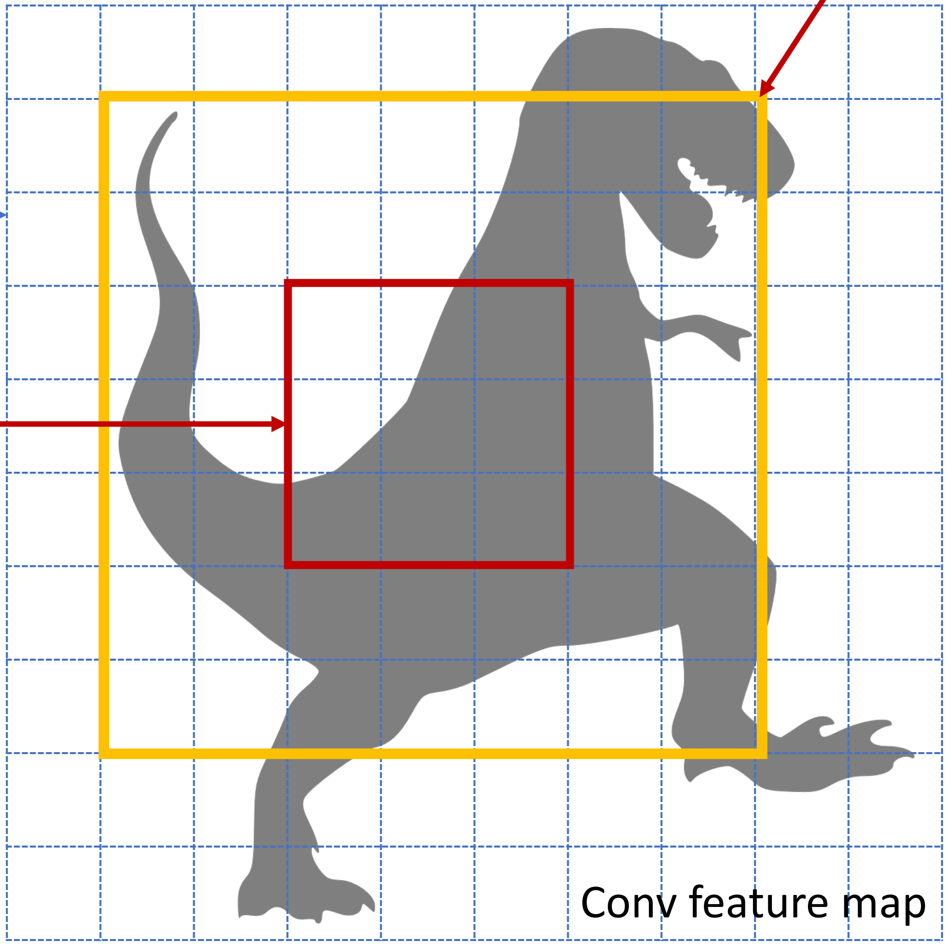
RPN: Anchor Box

Anchor box: predictions are w.r.t. this box, *not the 3x3 sliding window*



RPN: Anchor Box

f_I = FCN(I)



Anchor box: predictions are w.r.t. this box, *not the 3x3 sliding window*

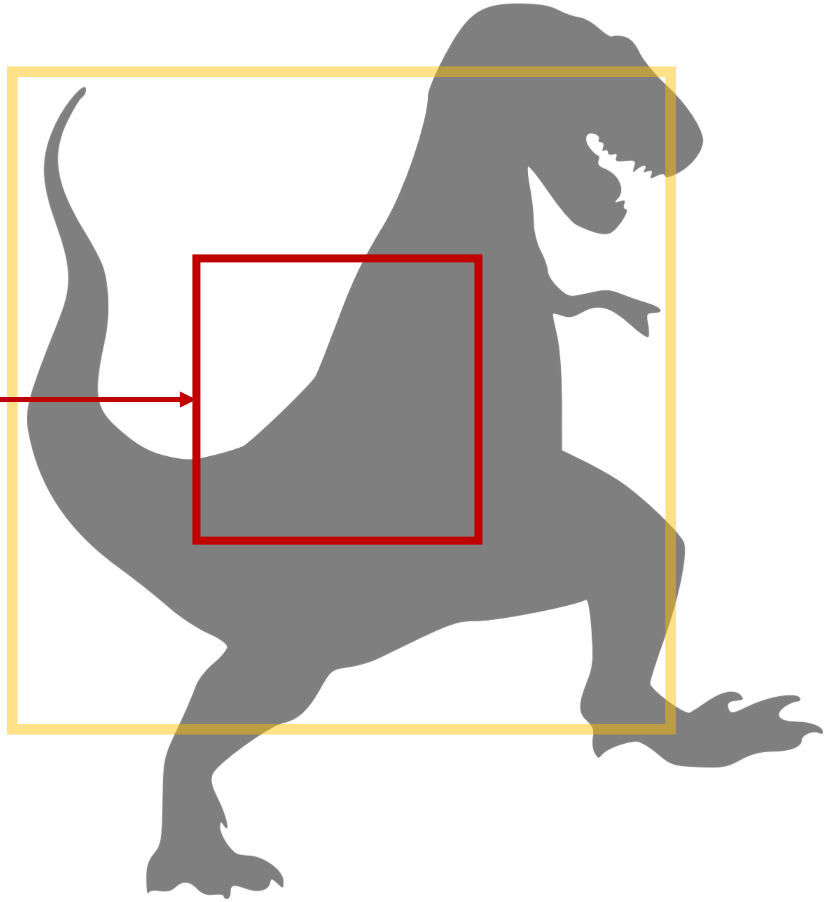
3x3 "sliding window"

- Objectness classifier [0, 1]
- Box regressor predicting (dx, dy, dh, dw)

RPN: Prediction (on object)

Objectness score

P(object) = 0.94



3x3 "sliding window"

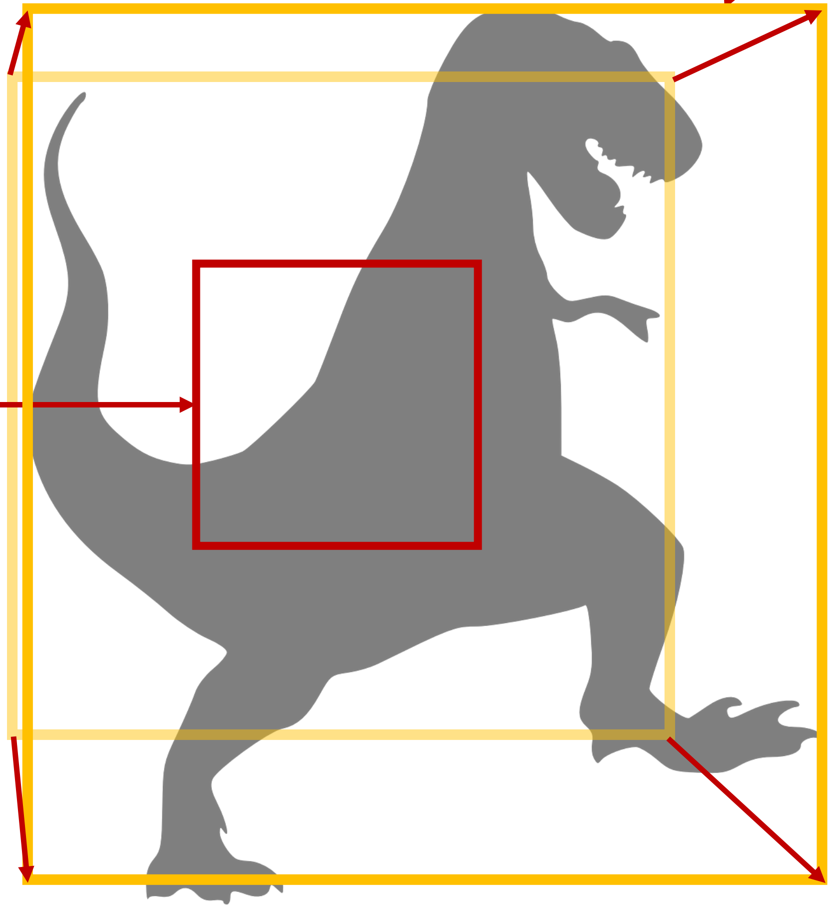
- Objectness classifier [0, 1]

- Box regressor
  predicting (dx, dy, dh, dw)

RPN: Prediction (on object)

Anchor box: transformed by box regressor

P(object) = 0.94



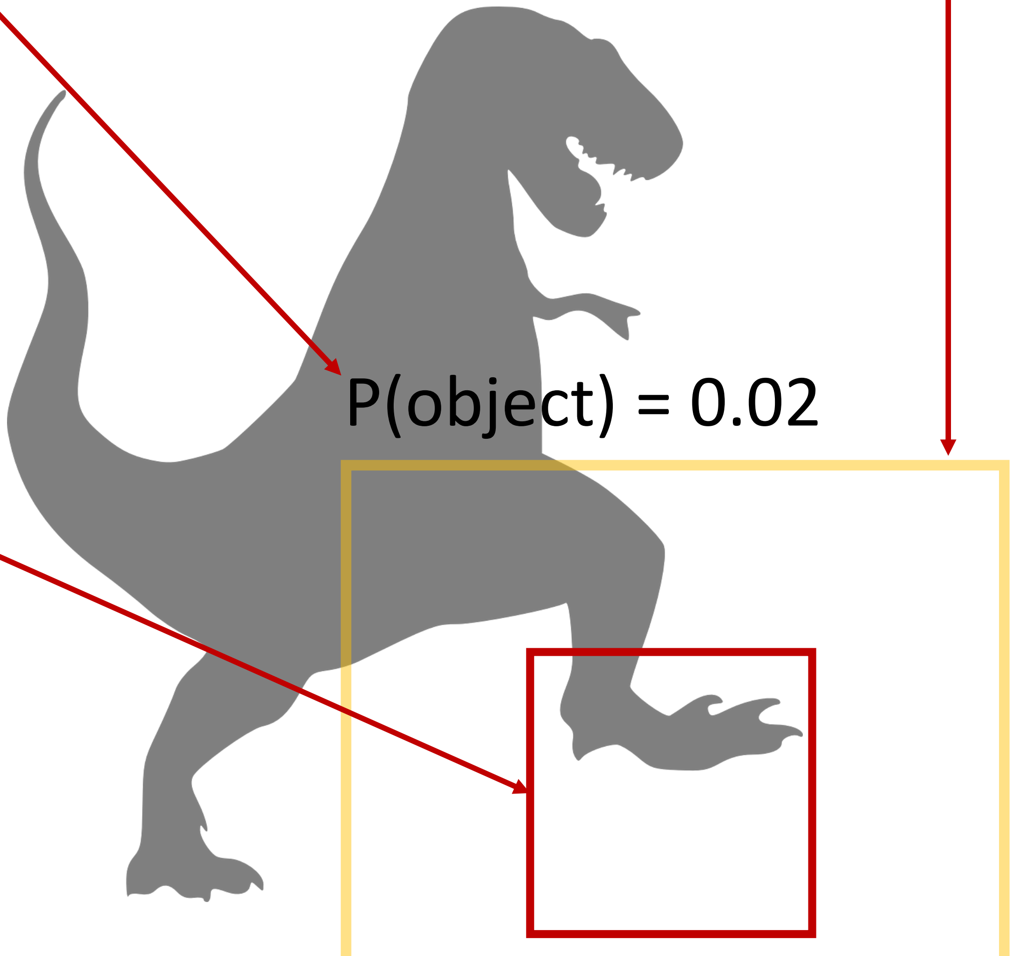
3x3 “sliding window”

- Objectness classifier [0, 1]
- Box regressor predicting (dx, dy, dh, dw)

RPN: Prediction (off object)

Objectness score

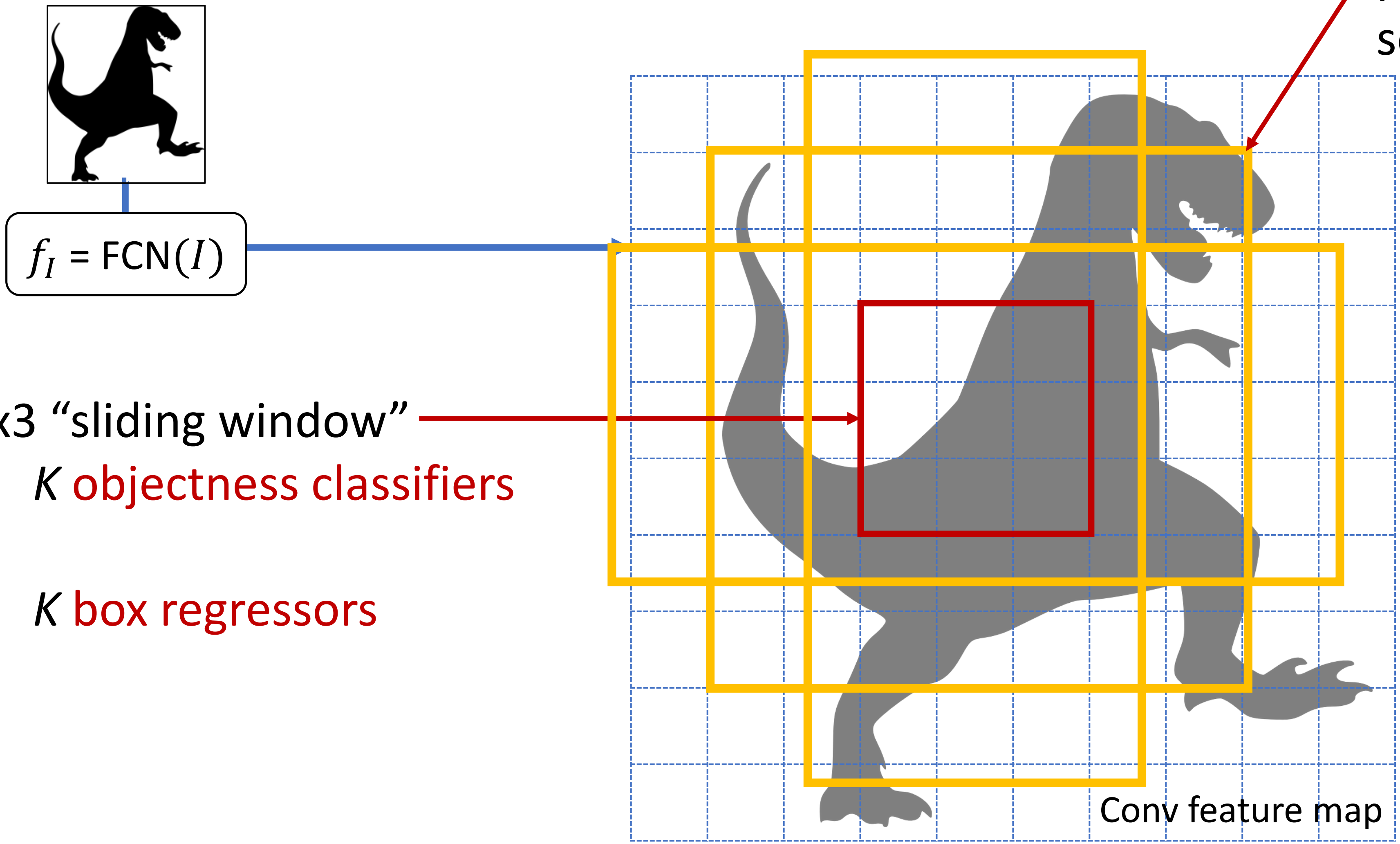
Anchor box: transformed by box regressor



3x3 "sliding window"

- Objectness classifier
- Box regressor predicting (dx, dy, dh, dw)

RPN: Multiple Anchors



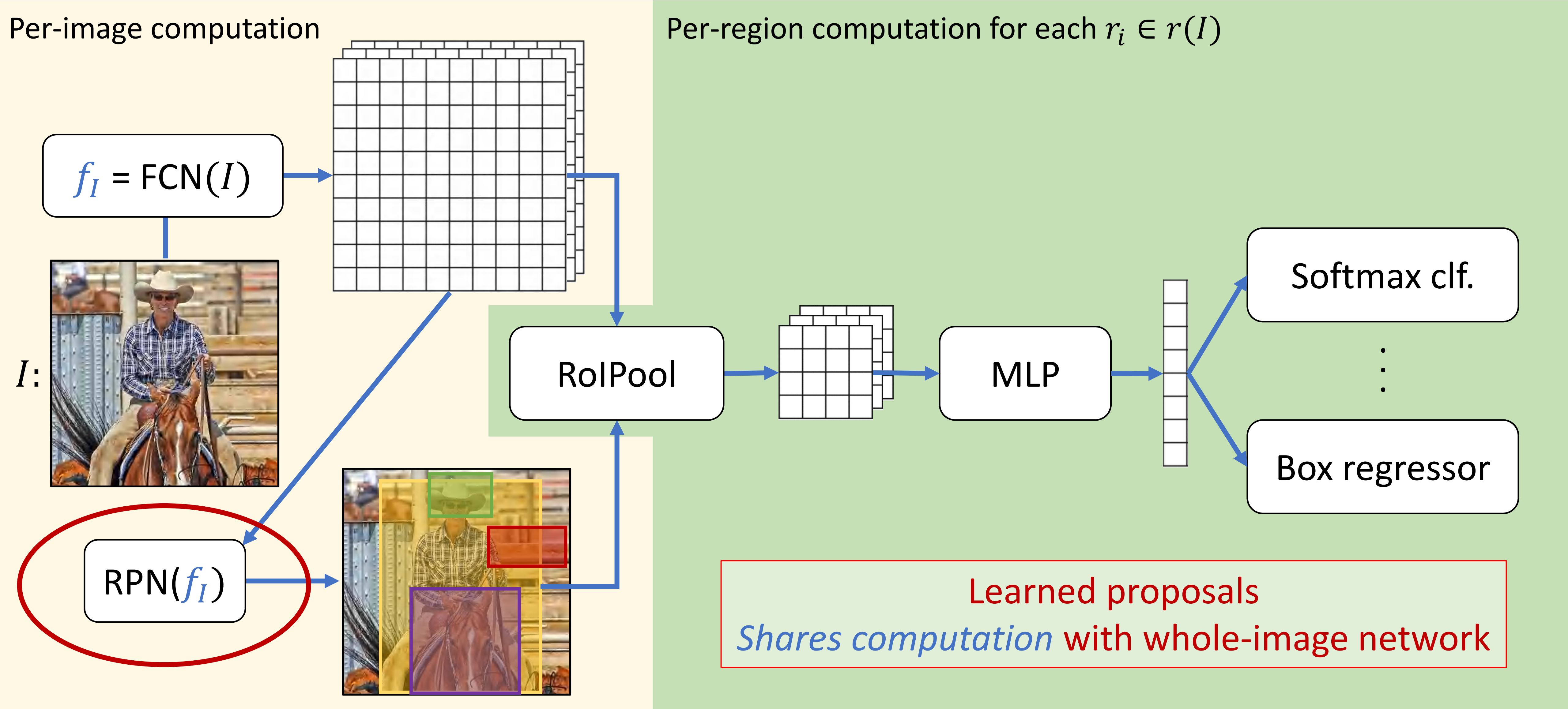
Anchor boxes: K anchors per location with different scales and aspect ratios

3x3 "sliding window"

- K objectness classifiers
- K box regressors

Ren, He, Girshick, Sun. Faster R-CNN: Towards Real-Time Object Detection. NIPS 2015.

Faster R-CNN



Improvements from “Mixin Methods”

- Largely orthogonal to the base detector
- Can be added to many different detectors as a “mixin”
  (A better backbone network is one example)

Building
a better
hammer

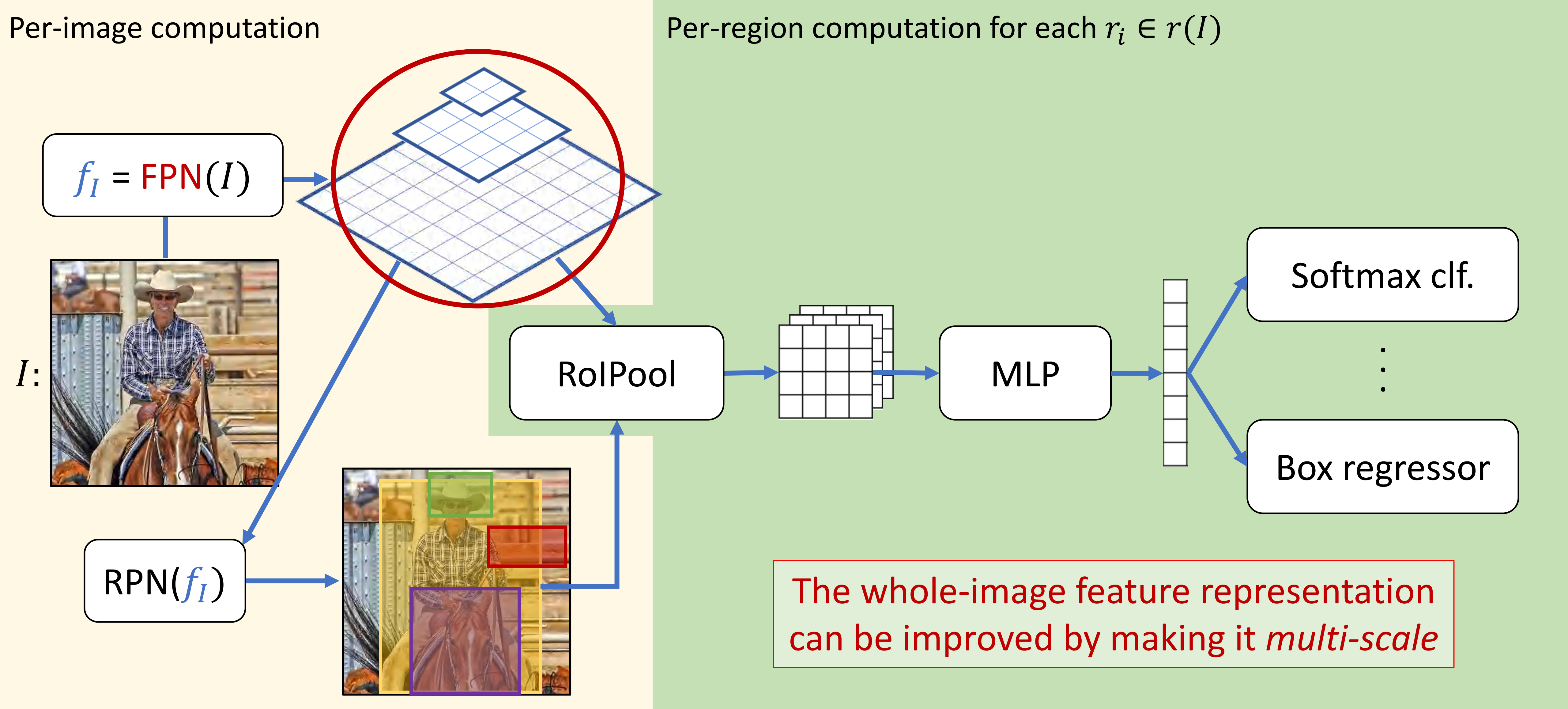
R-CNN

Fast R-CNN

Mixin method

Faster R-CNN

Faster R-CNN with a Feature Pyramid Network

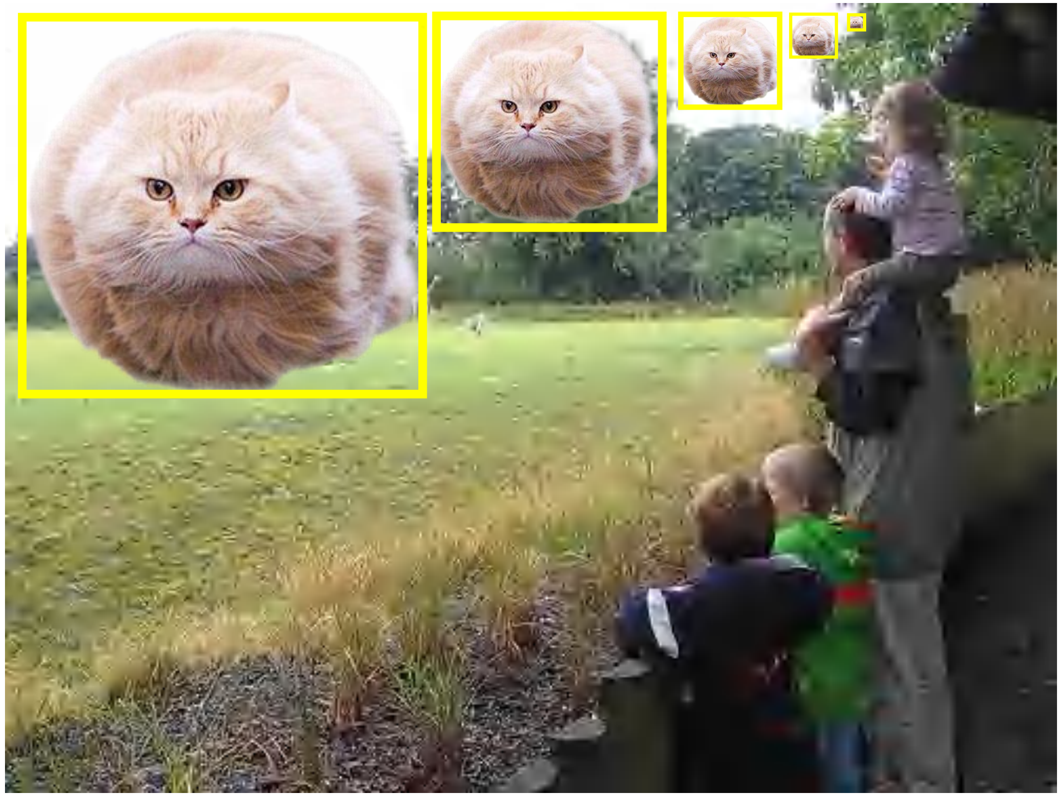


Feature Pyramid Network (FPN)

References

- O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In MIC- CAI, 2015.
- P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In ECCV, 2016.
- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. SSD: Single shot multibox detector. In ECCV, 2016.
- A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. arXiv:1612.06851, 2016.
- T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie. Feature Pyramid Networks for Object Detection. In CVPR, 2017.

FPN: Improving Scale Invariance and Equivariance



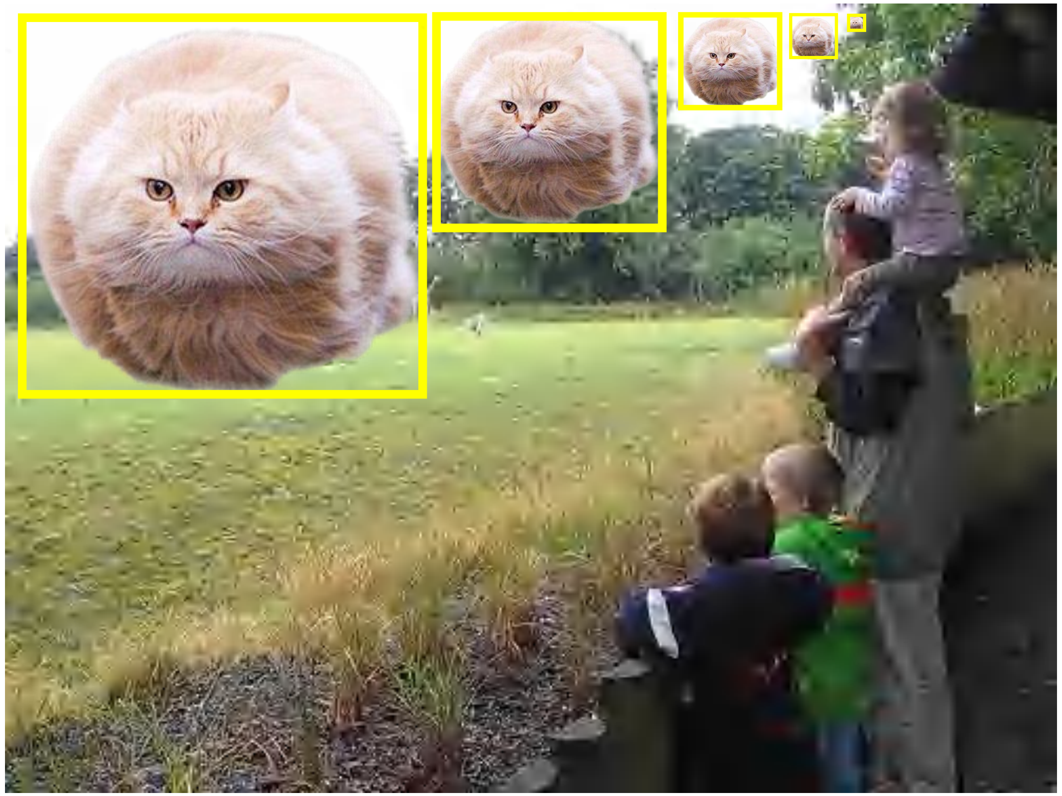
Detectors need to

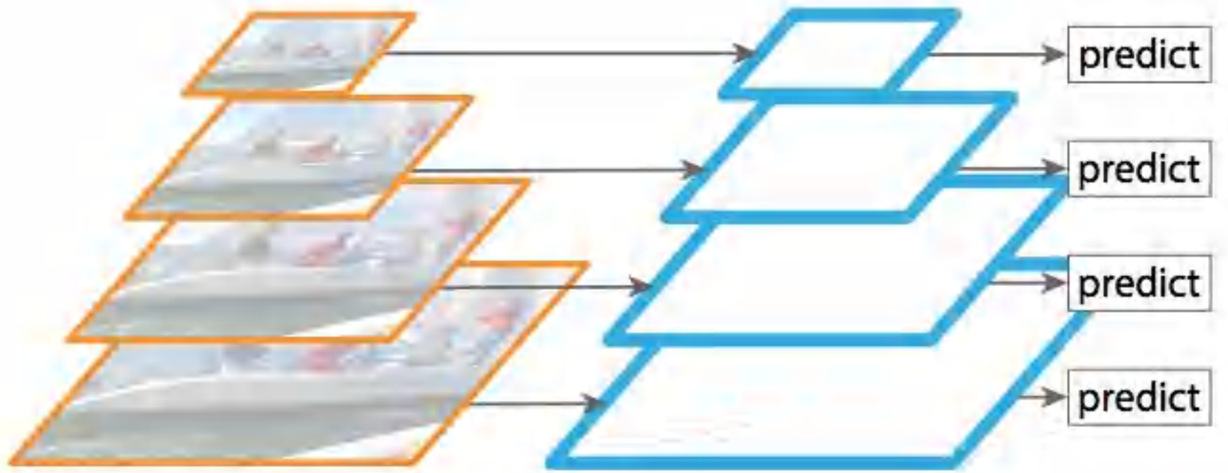
1. classify and
2. localize

objects over a wide range of scales

FPN improves this ability

Strategy 1: Image Pyramid



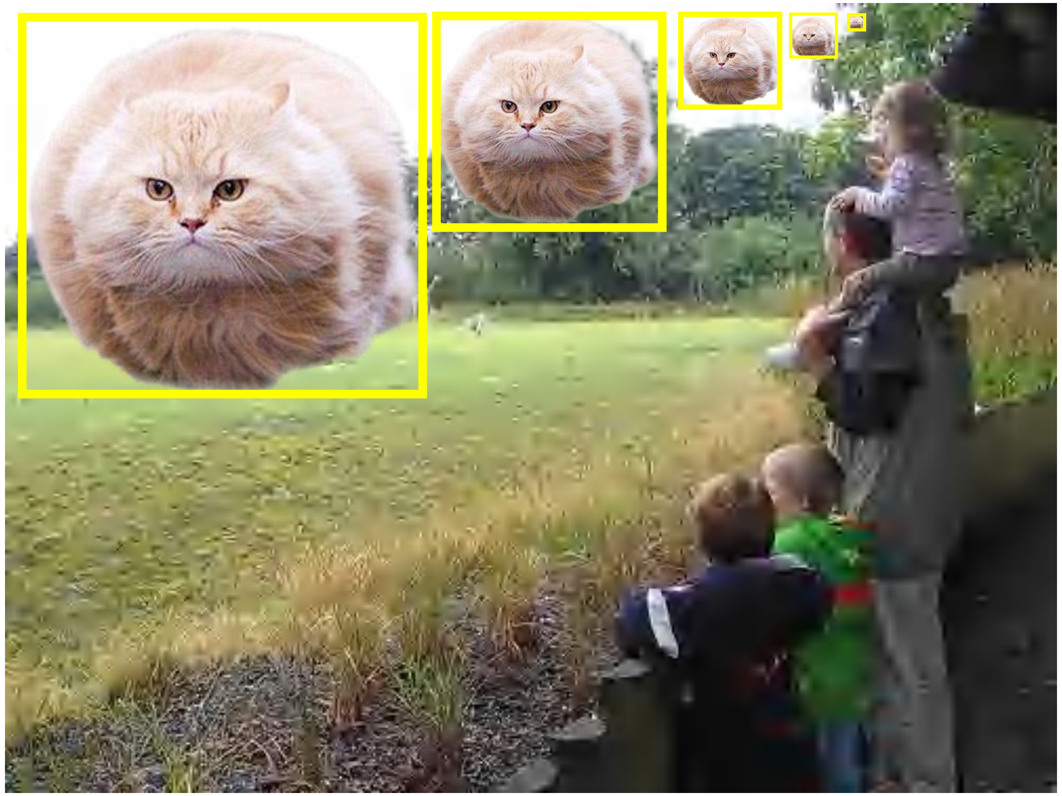


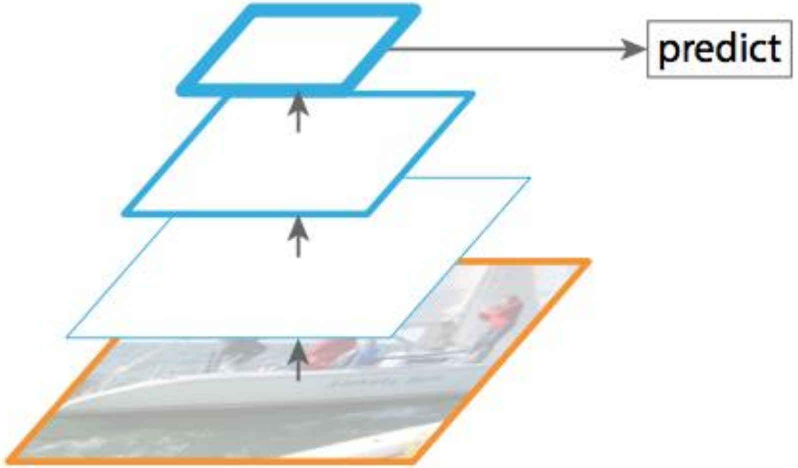
(a) Featurized image pyramid

Standard solution – *slow!*

(E.g., Viola & Jones, HOG, DPM, SPP-net, multi-scale Fast R-CNN, …)

Strategy 2: Multi-scale Features (Single-scale Map)



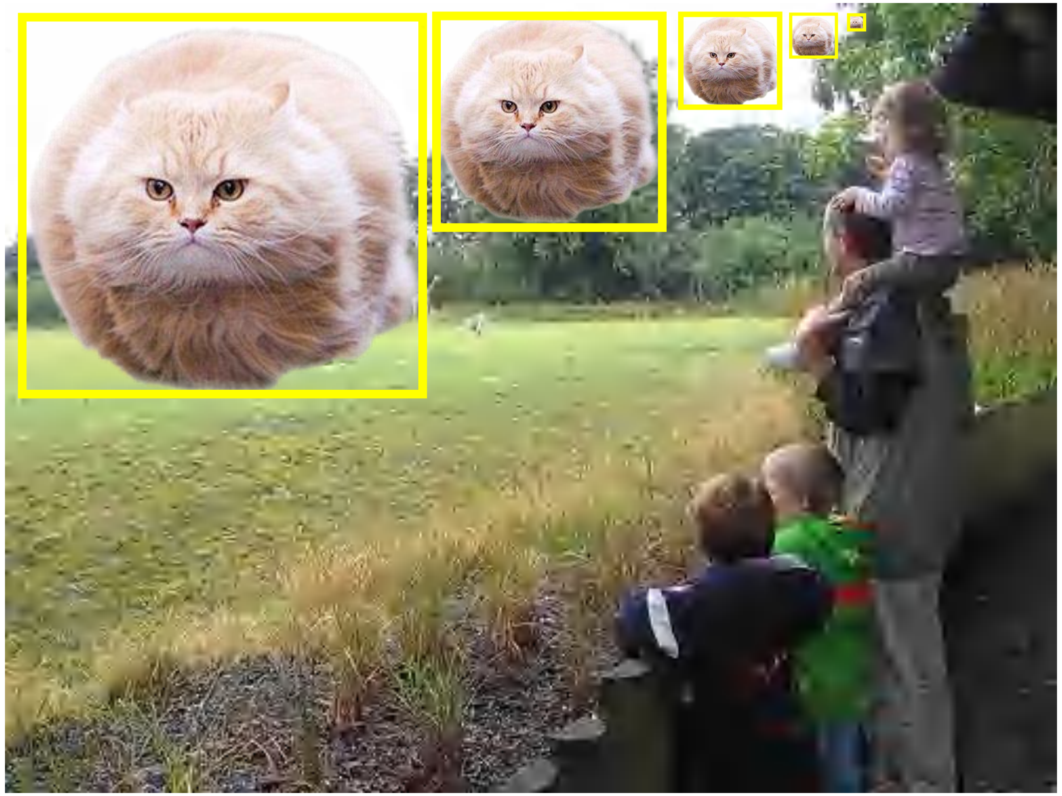


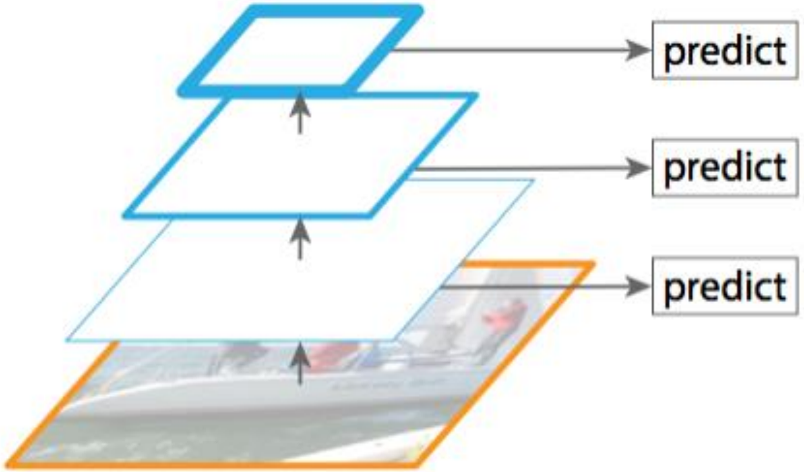
(b) Single feature map

Leave it all to the features – *fast, suboptimal*

(E.g., Fast/er R-CNN, YOLO, …)

Strategy 3: Naïve In-network Pyramid



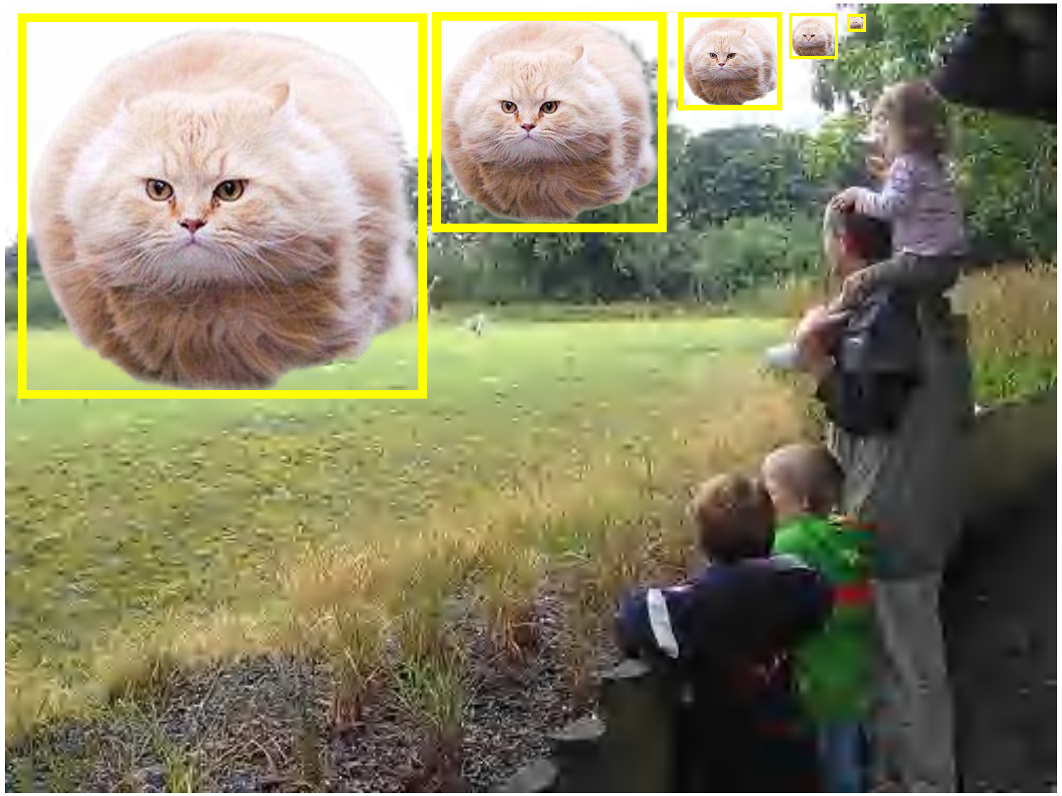


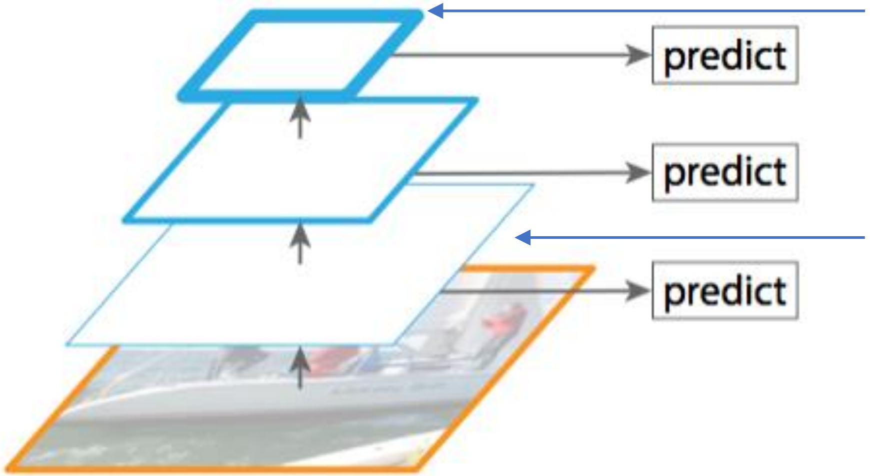
(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Strategy 3: Naïve In-network Pyramid





Low resolution,
Strong features

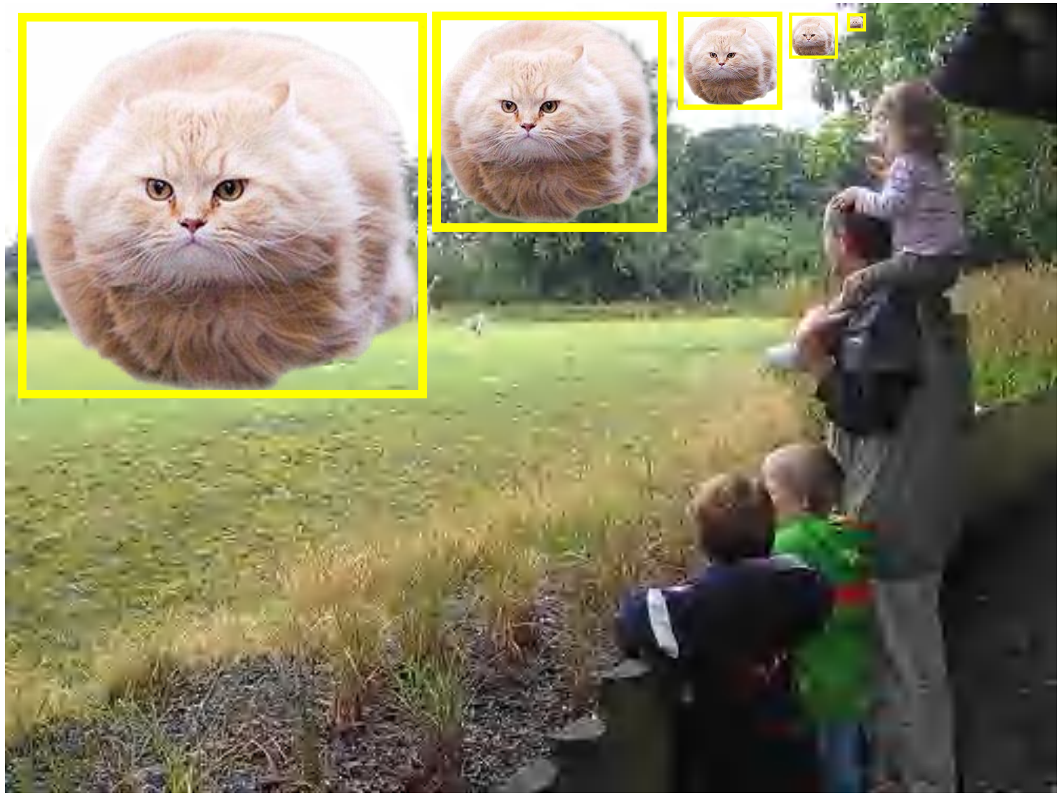
High resolution,
Weak features

(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Strategy 4: Feature Pyramid Network

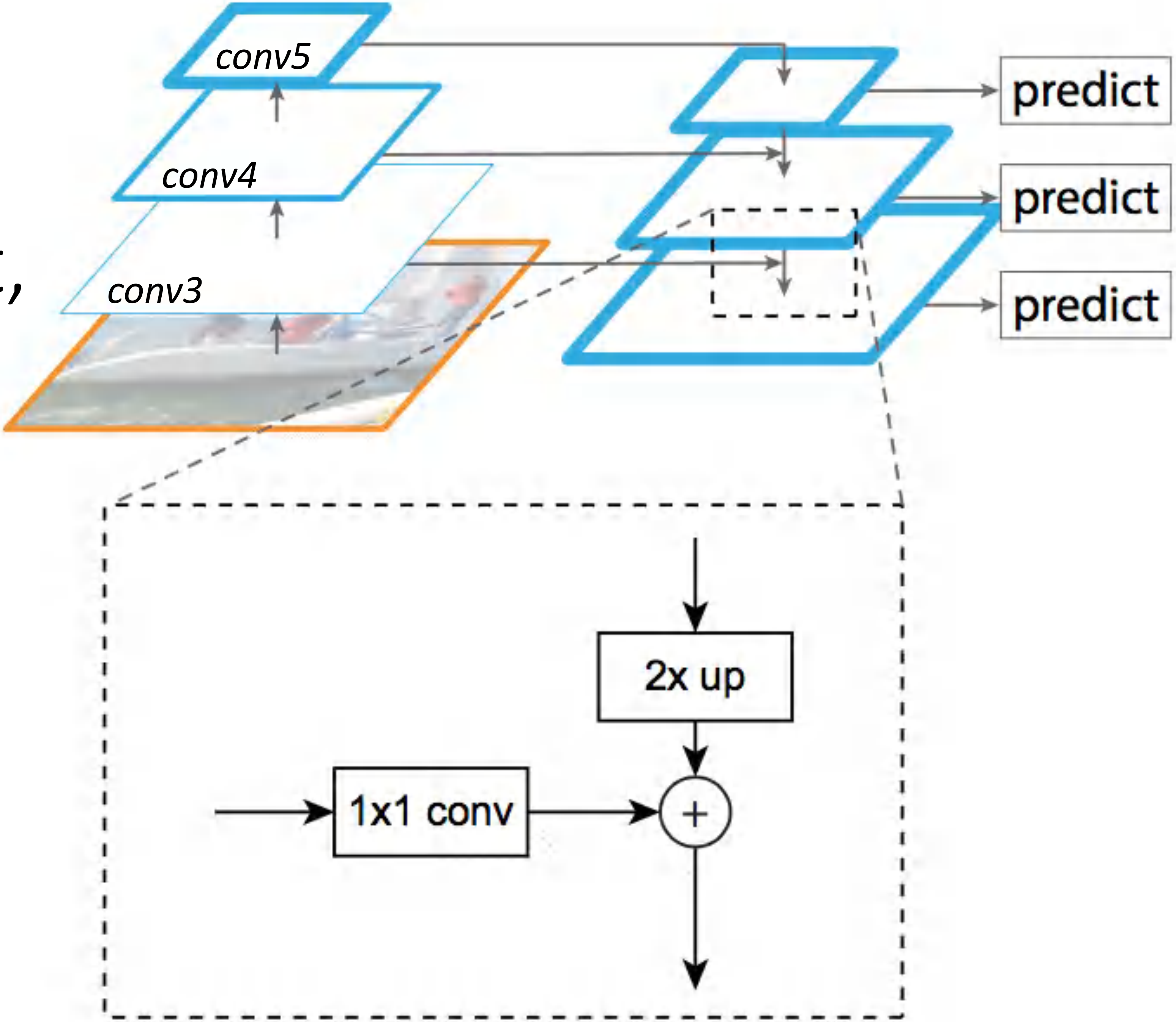


(d) Feature Pyramid Network

Top-down enrichment of high-res features – *fast, less suboptimal*

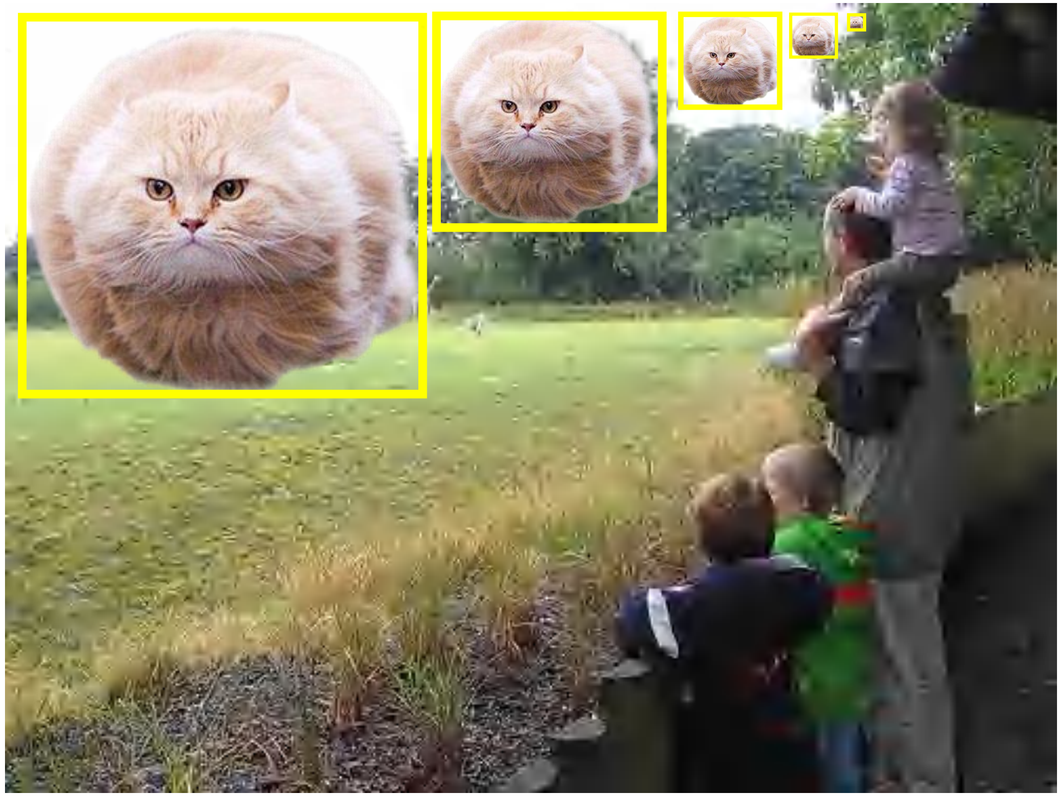
Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017.

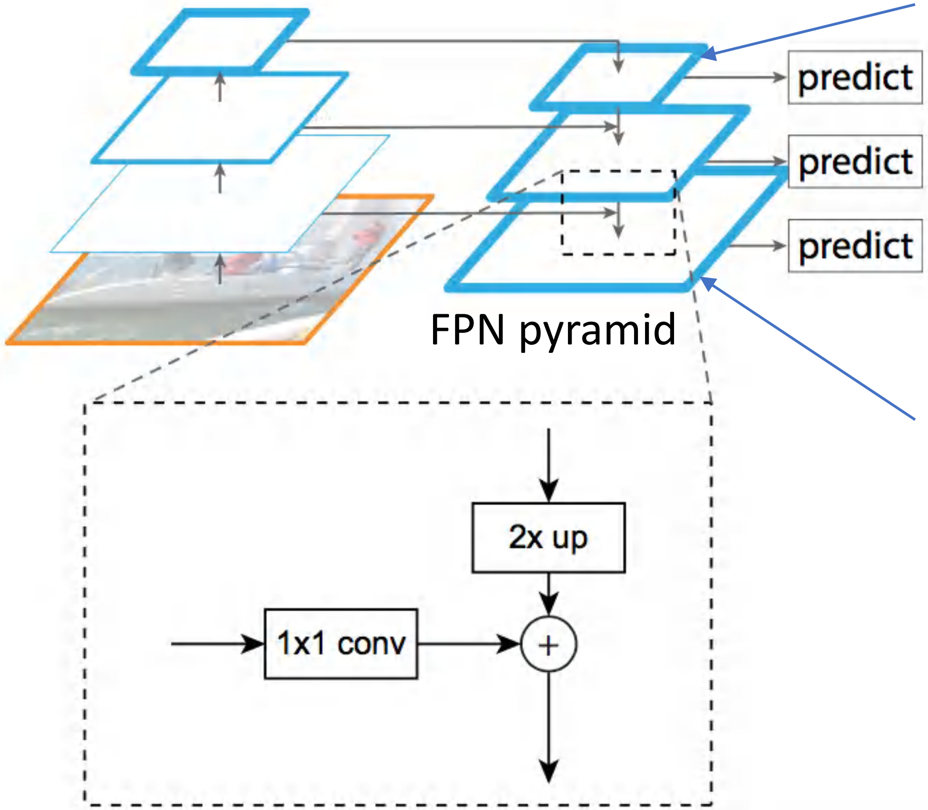
FPN: Light-weight, Top-down Refinement Module



Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017.

No Compromise on Feature Quality, Still Fast





Low resolution,
Strong features

High resolution,
Strong features

Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017. See also: Shrivastava's TDM.

FPN – A Generic Backbone Modification

Generates a feature pyramid—*useful in many applications!*

- RPN
- Fast/er R-CNN
- Mask R-CNN
- RetinaNet
- **TensorMask**
- **Panoptic FPN – *more details in the next talk!***

Mask R-CNN: The Final Hammer of this Tutorial

R-CNN

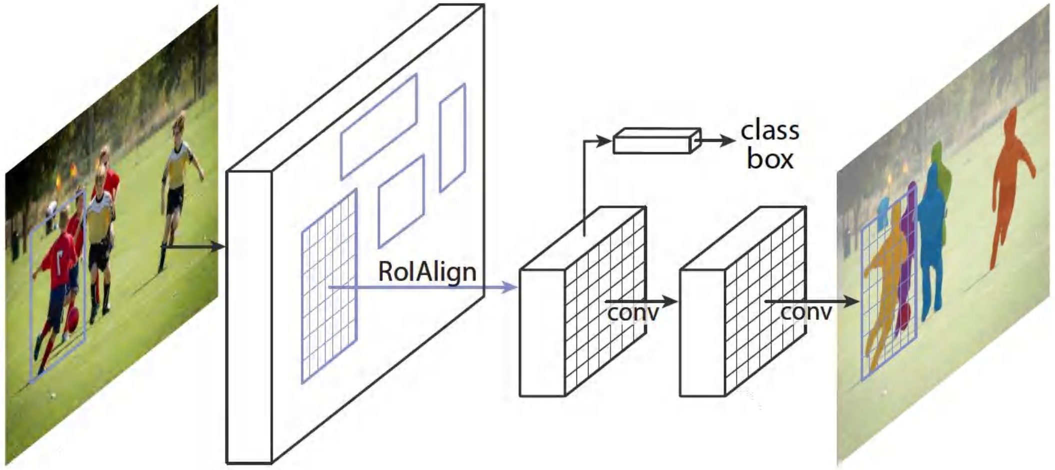
Fast R-CNN

Building
a better
hammer

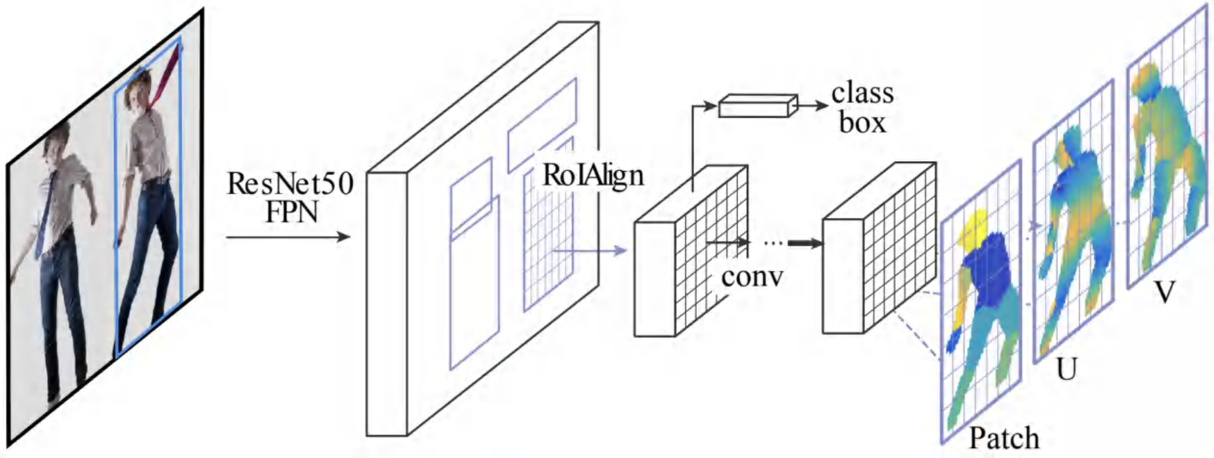
Faster R-CNN

Mask R-CNN

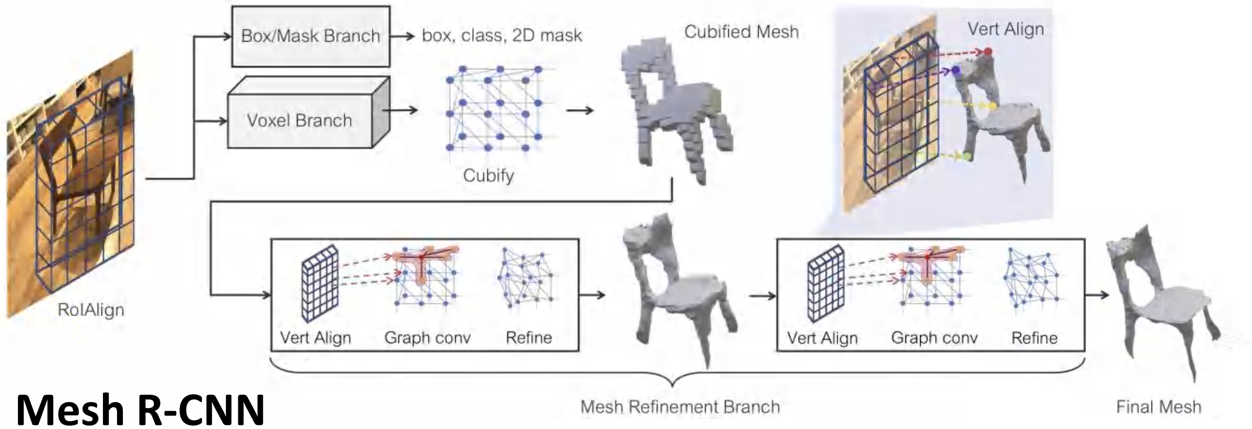
Generalized R-CNN: Adding More Heads



Mask R-CNN
[He, Gkioxari, Dollár, Girshick]

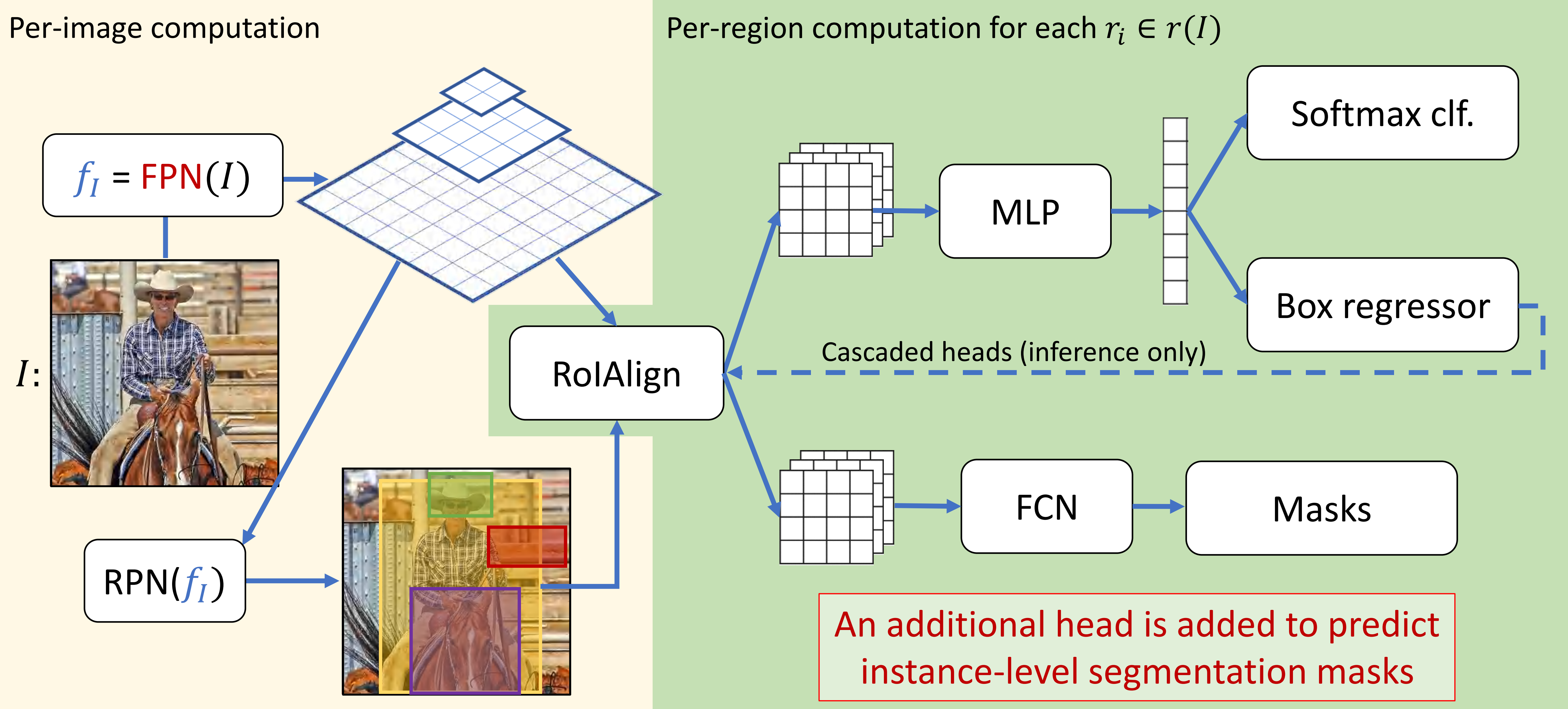


DensePose
[Güler, Neverova, Kokkinos]

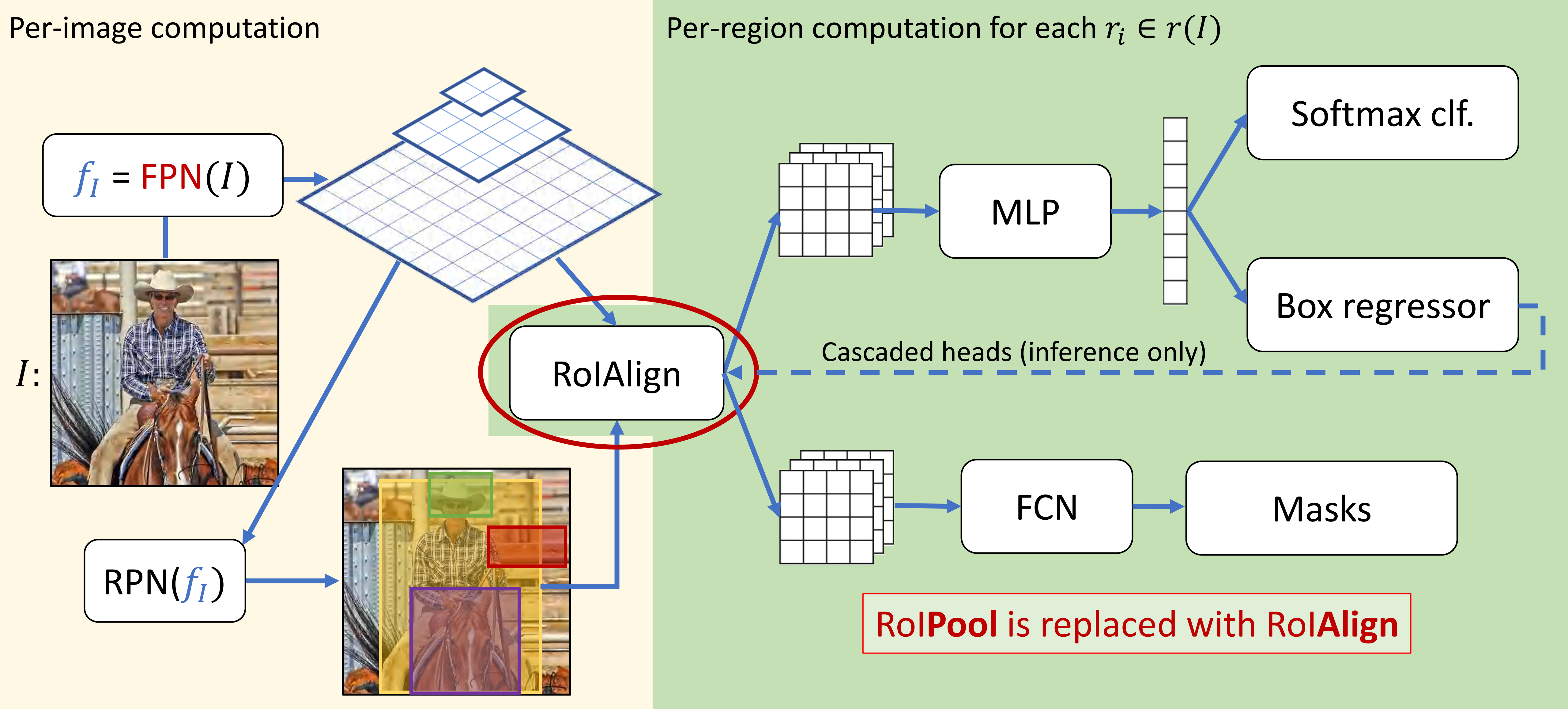


Mesh R-CNN
[Gkioxari, Malik, Johnson arXiv 2019]

Mask R-CNN


Per-image computation
Per-region computation for each $r_i \in r(I)$
f_I = FPN(I)
I:
RPN(f_I)
RoIAlign
MLP
Softmax clf.
Box regressor
Cascaded heads (inference only)
FCN
Masks
An additional head is added to predict instance-level segmentation masks

Mask R-CNN

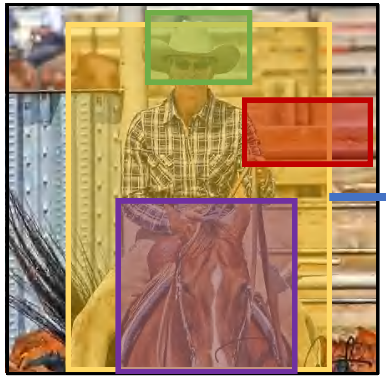

Per-image computation
Per-region computation for each $r_i \in r(I)$
f_I = FPN(I)
I:
RPN(f_I)
RoIAlign
MLP
Softmax clf.
Box regressor
Cascaded heads (inference only)
FCN
Masks
RoIPool is replaced with RoIAlign

He et al. Mask R-CNN. ICCV 2017.

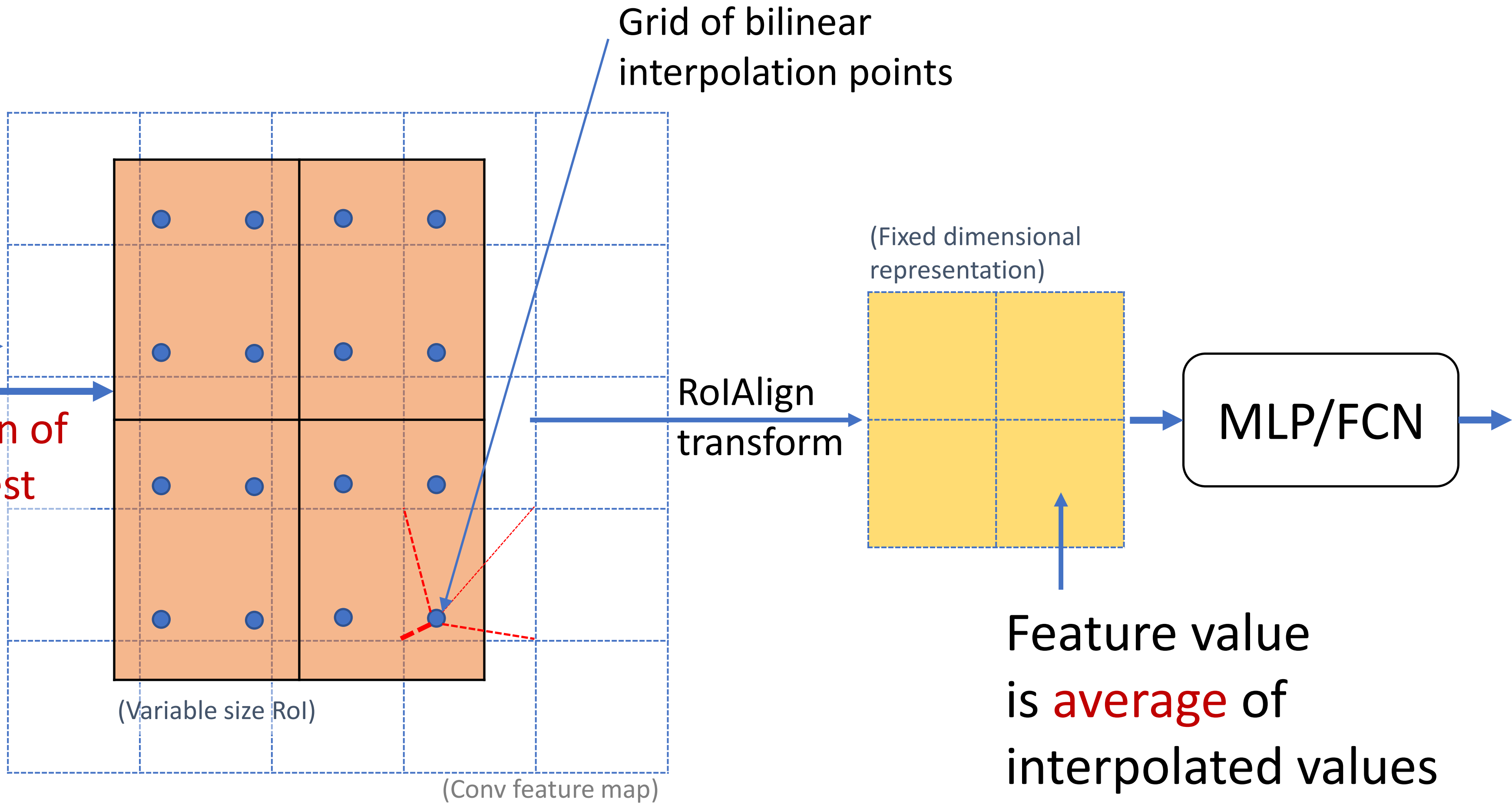
RoIAlign Operation (on each Proposal)

Transform arbitrary size proposal into a fixed-dimensional representation (e.g., 2x2)

f_I = FCN(I)

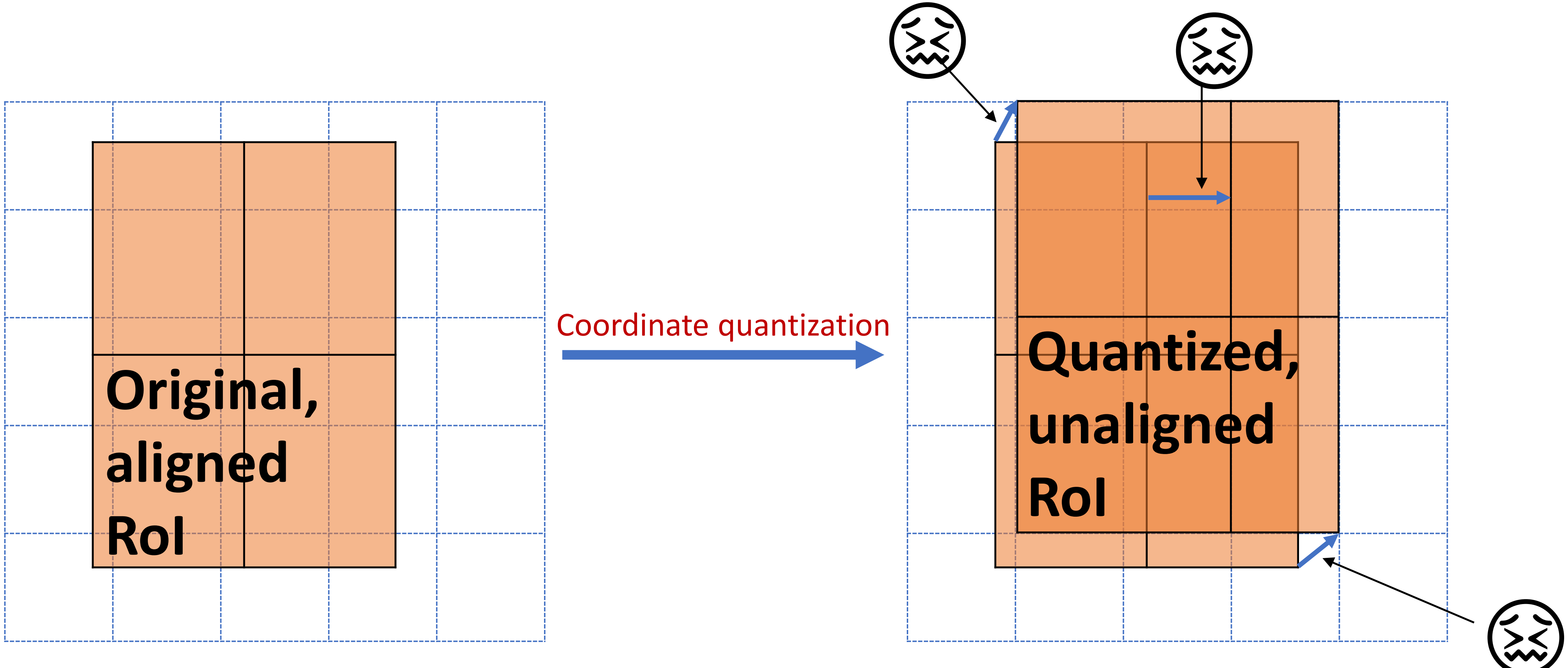


Region of Interest (RoI)



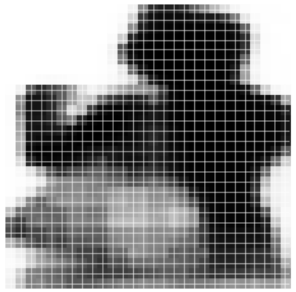
Compare to RoIPool and RoIWarp

Quantization breaks pixel-to-pixel **alignment** between input and output



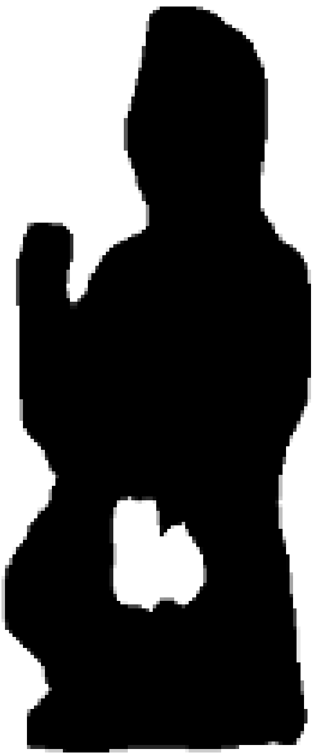
Mask Prediction

28x28 soft prediction



Resized soft prediction

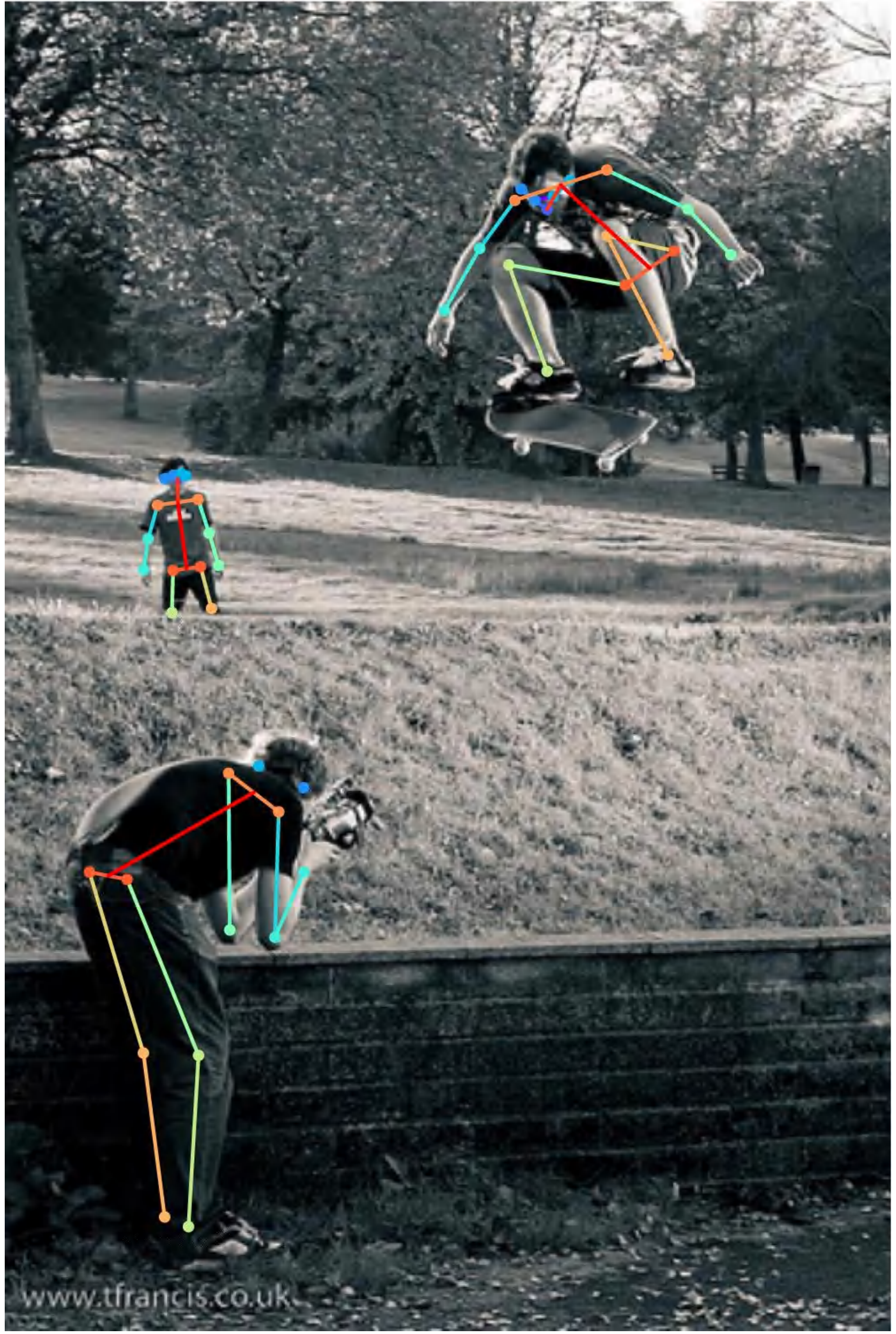
Final mask



Validation image with box detection shown in red


person1.00
person1.00
person1.00
person.98
surfboard1.00
person1.00
person.91
surfboard1.00
surfboard.98
surfboard1.00
person.74
surfboard1.00


person.88
person1.00
tv.98
tv.84
person1.00
person1.00
bottle.97
wine glass.99
dining table.95
wine glass1.00
wine glass1.00

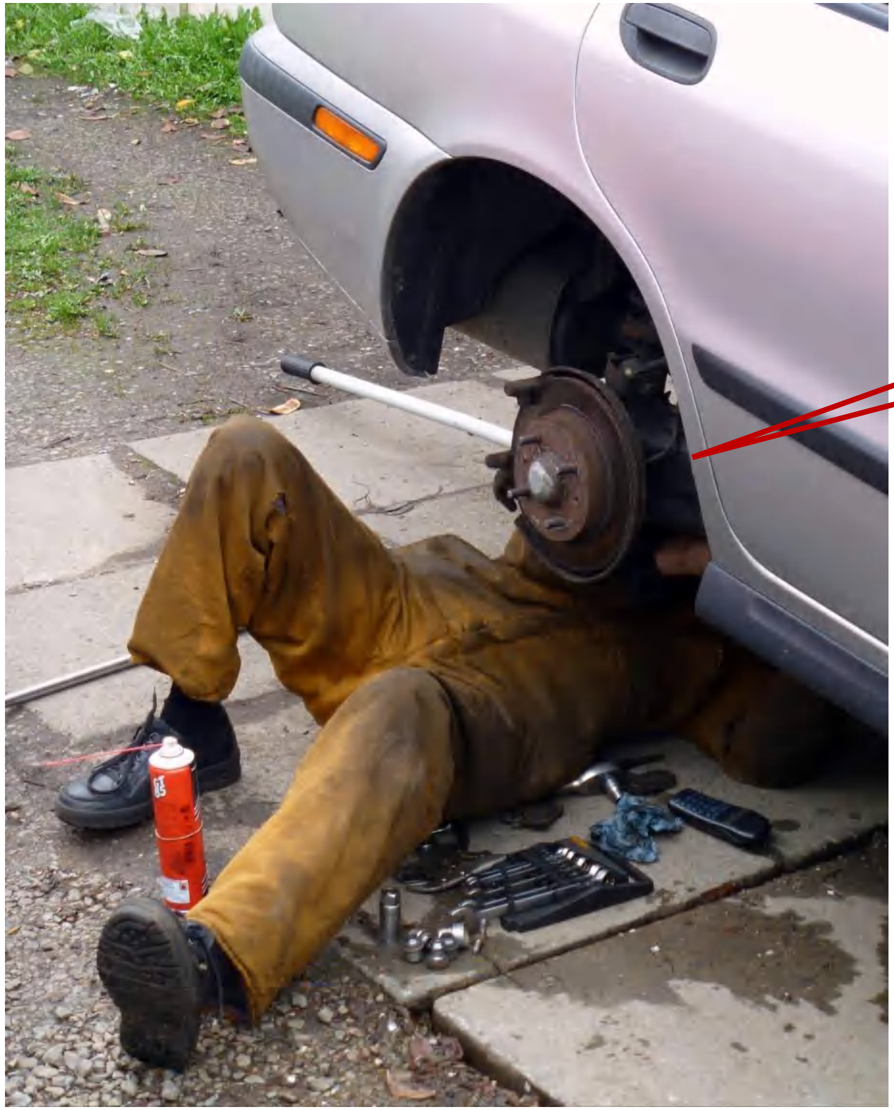

www.tfrancis.co.uk

Overview of this Tutorial

Topics to cover

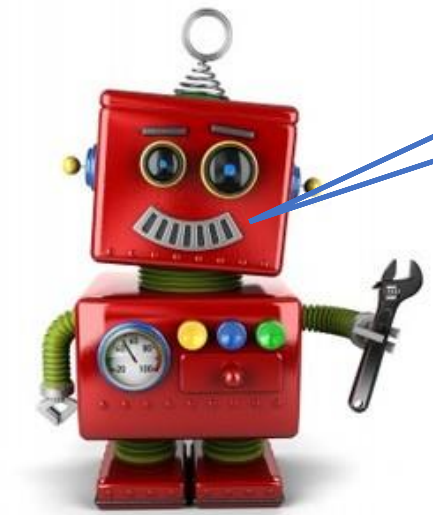
- Object detection intro (very brief)
- The Generalized R-CNN framework
- Open challenges in object detection

Ubiquitous Human <-> Machine Communication



https://www.flickr.com/photos/comedynose/5596445582 (public domain)

Please hand me the *wrench*



I can help!
Here’s your wrench!

The Future: Detect Every Thing!

- Ubiquitous, visually grounded human <-> machine communication requires machines with large vocabularies

From COCO…

Detect Every Thing!

- Ubiquitous, visually grounded human <-> machine communication requires machines with large vocabularies

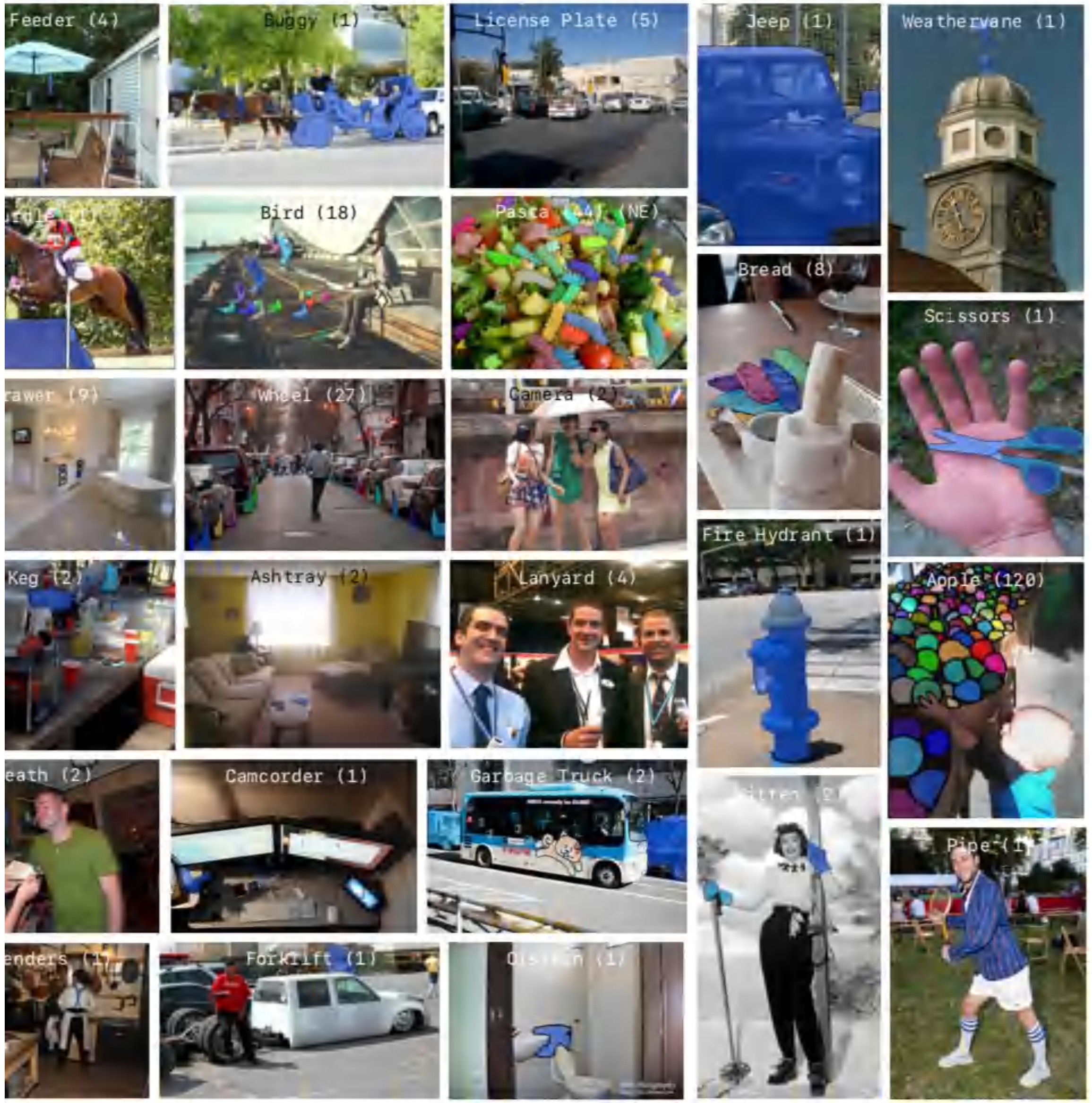
From COCO (80 categories)…

…to the future! (1000's)

Introducing LVIS (*"el-vis"*)

Large Vocabulary Instance Segmentation

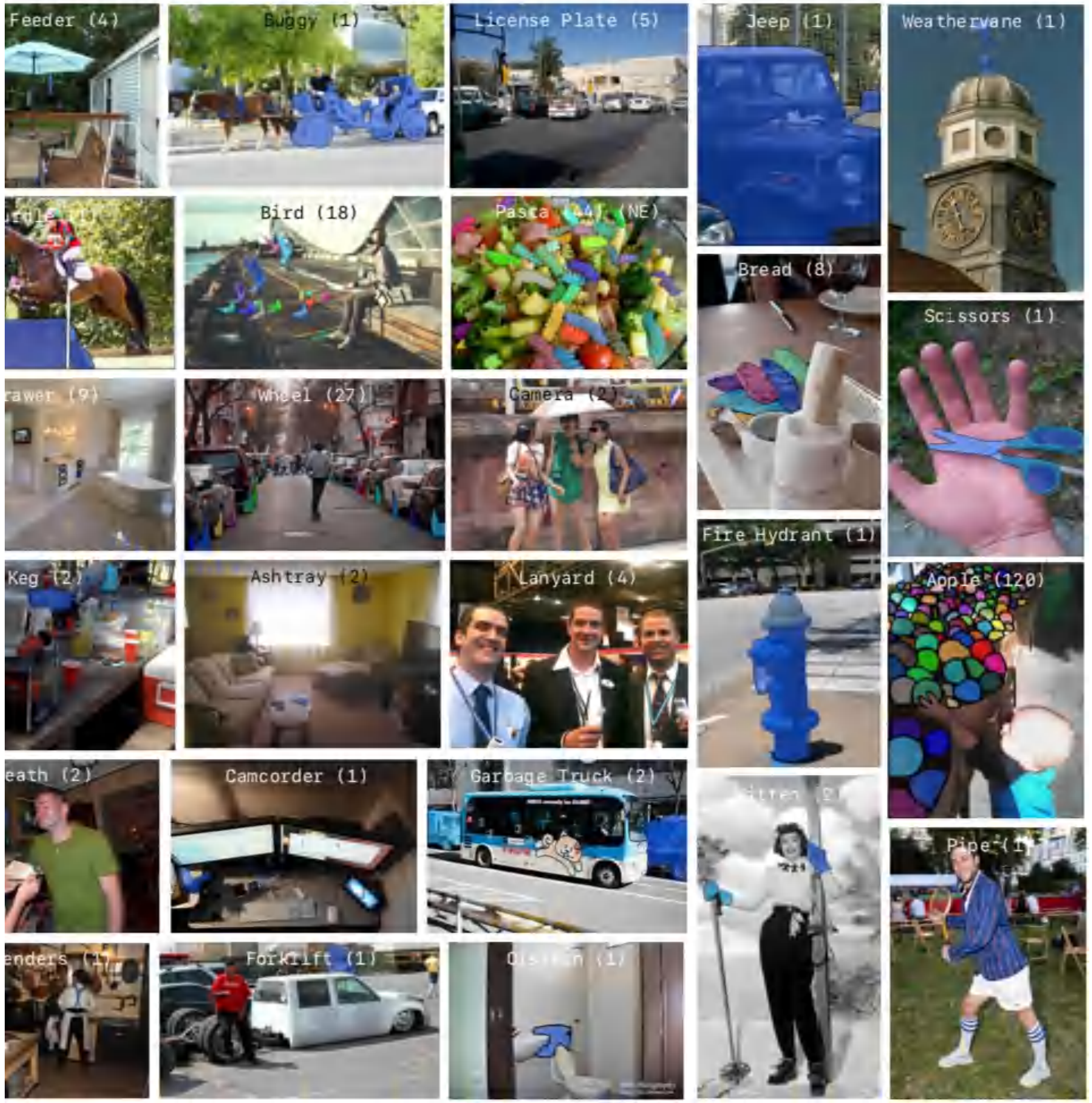
- 164k images (from COCO)



Introducing LVIS *("el-vis")*

Large Vocabulary Instance Segmentation

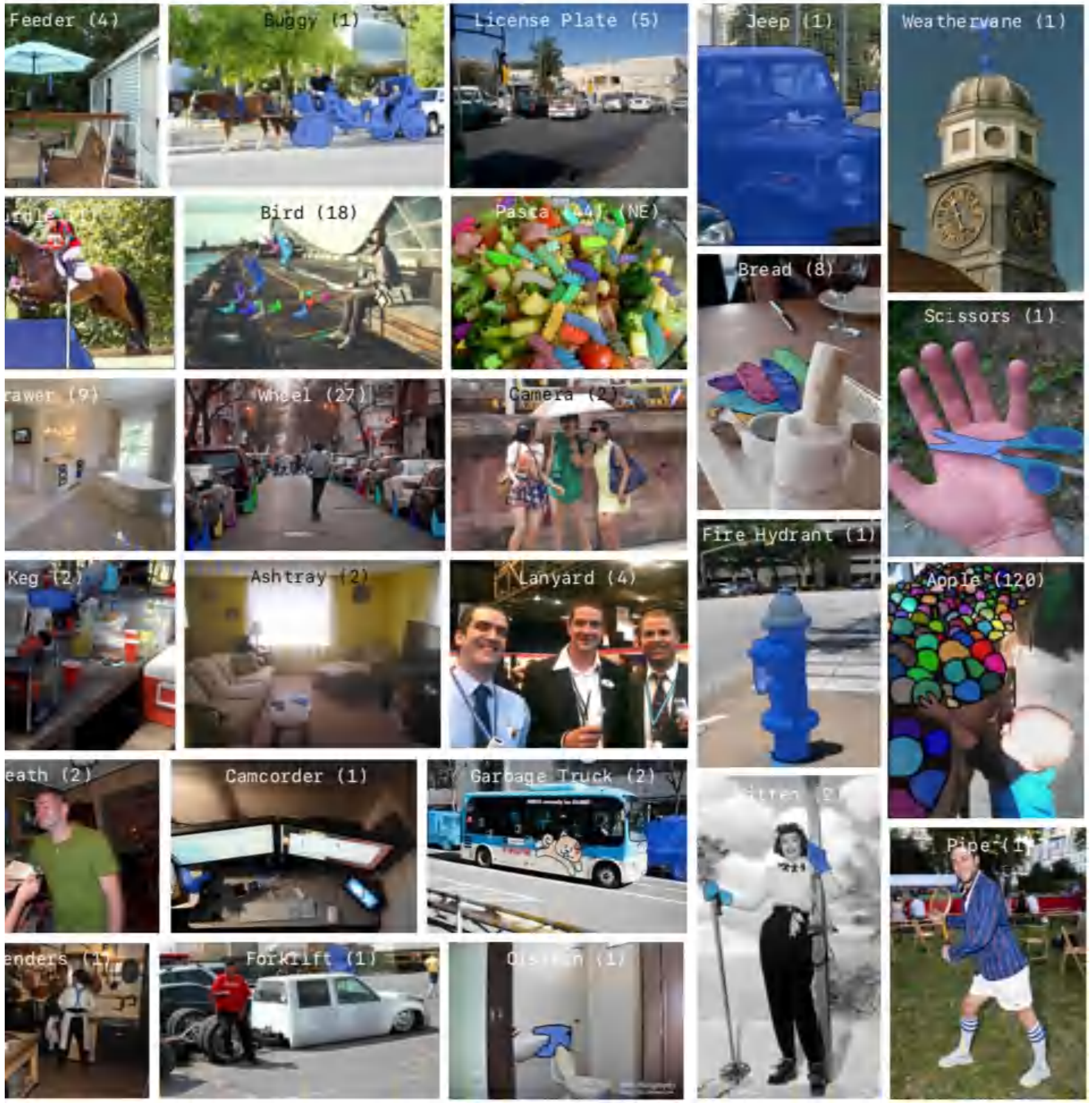
- 164k images (from COCO)
- ~1200 *discovered* categories



Introducing LVIS *("el-vis")*

Large Vocabulary Instance Segmentation

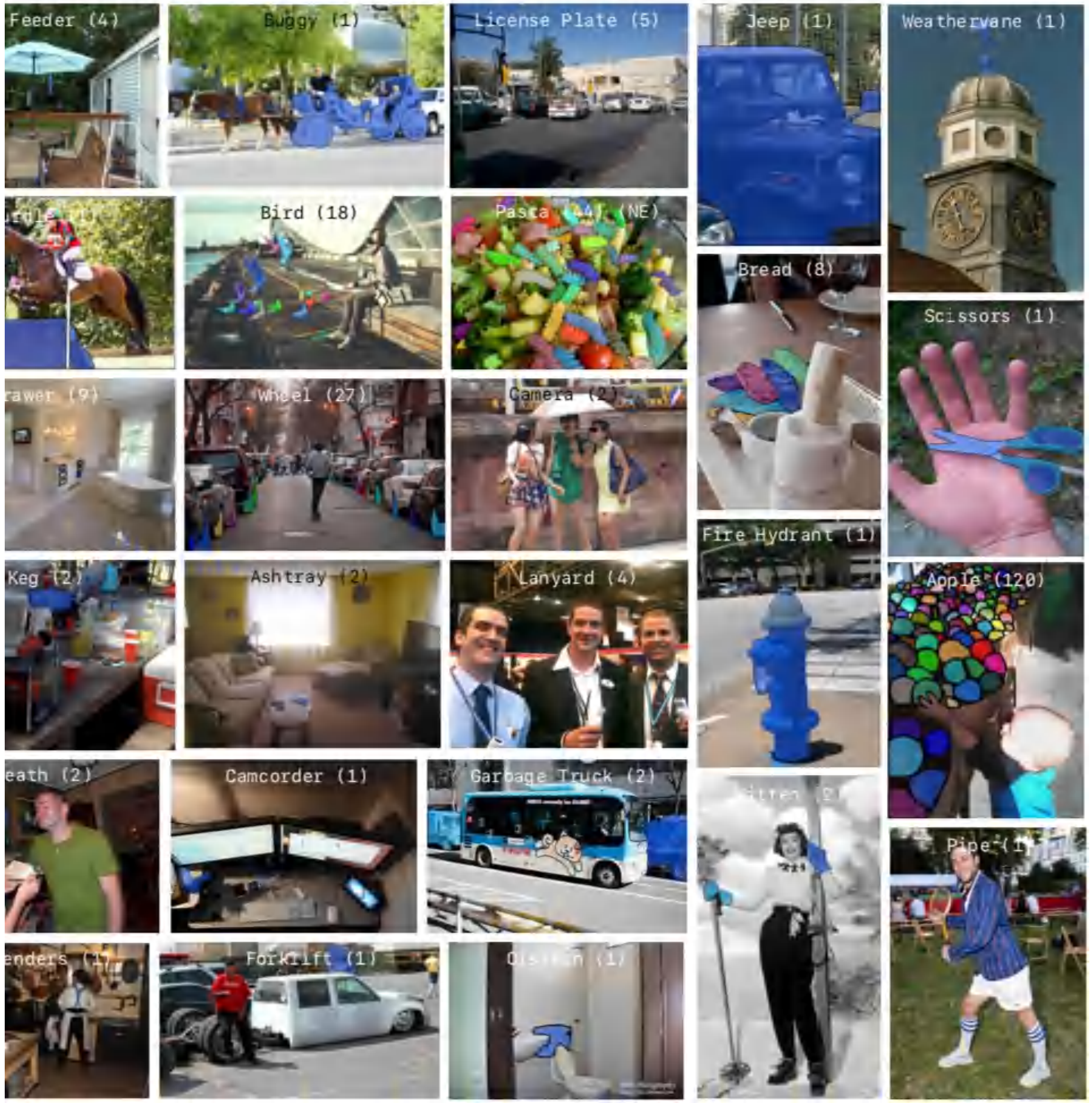
- 164k images (from COCO)
- ~1200 *discovered* categories
- ~2.2M instance segmentations



Introducing LVIS *("el-vis")*

Large Vocabulary Instance Segmentation

- 164k images (from COCO)
- ~1200 *discovered* categories
- ~2.2M instance segmentations
- Long-tailed frequency distribution

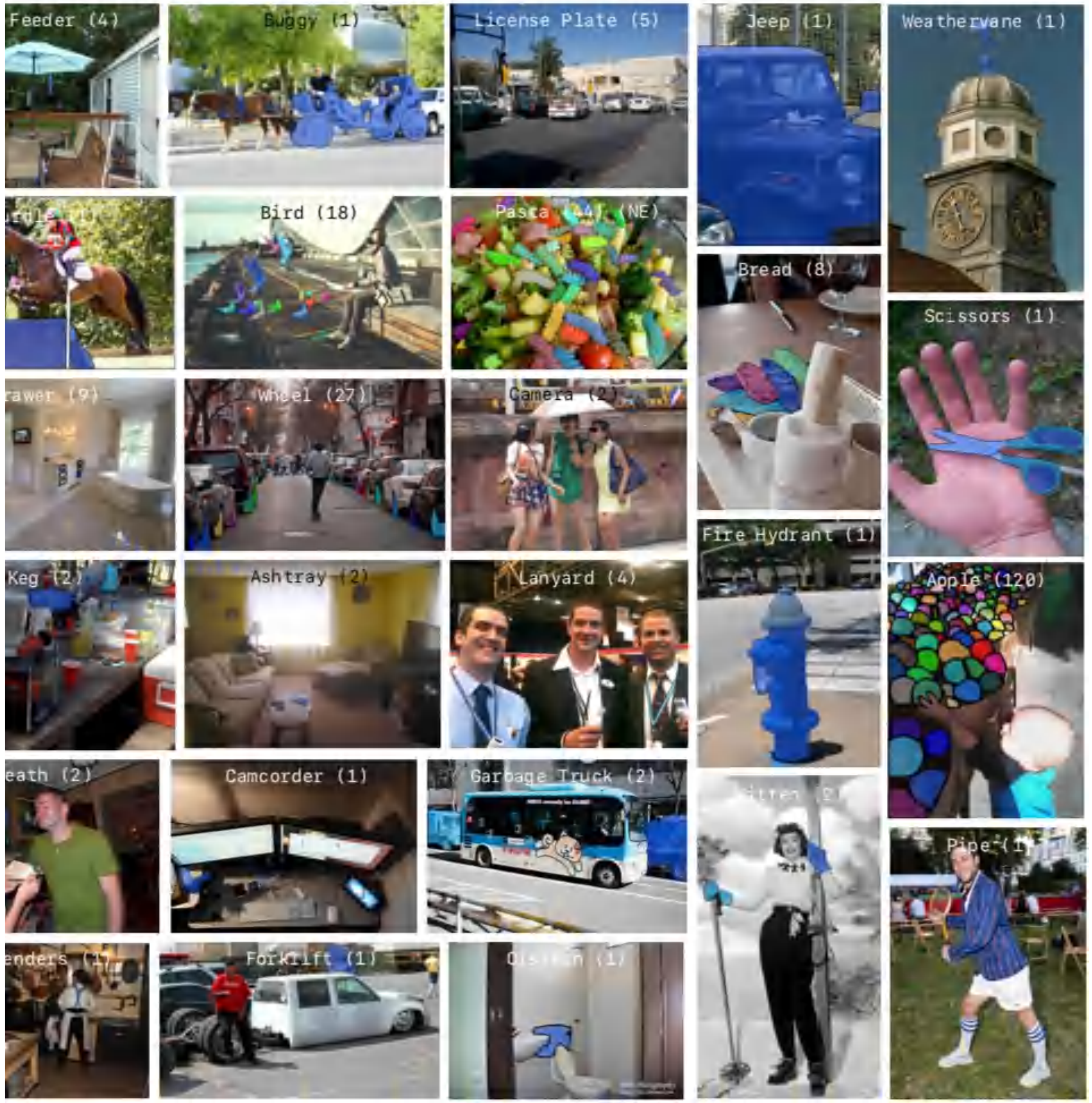


Introducing LVIS *("el-vis")*

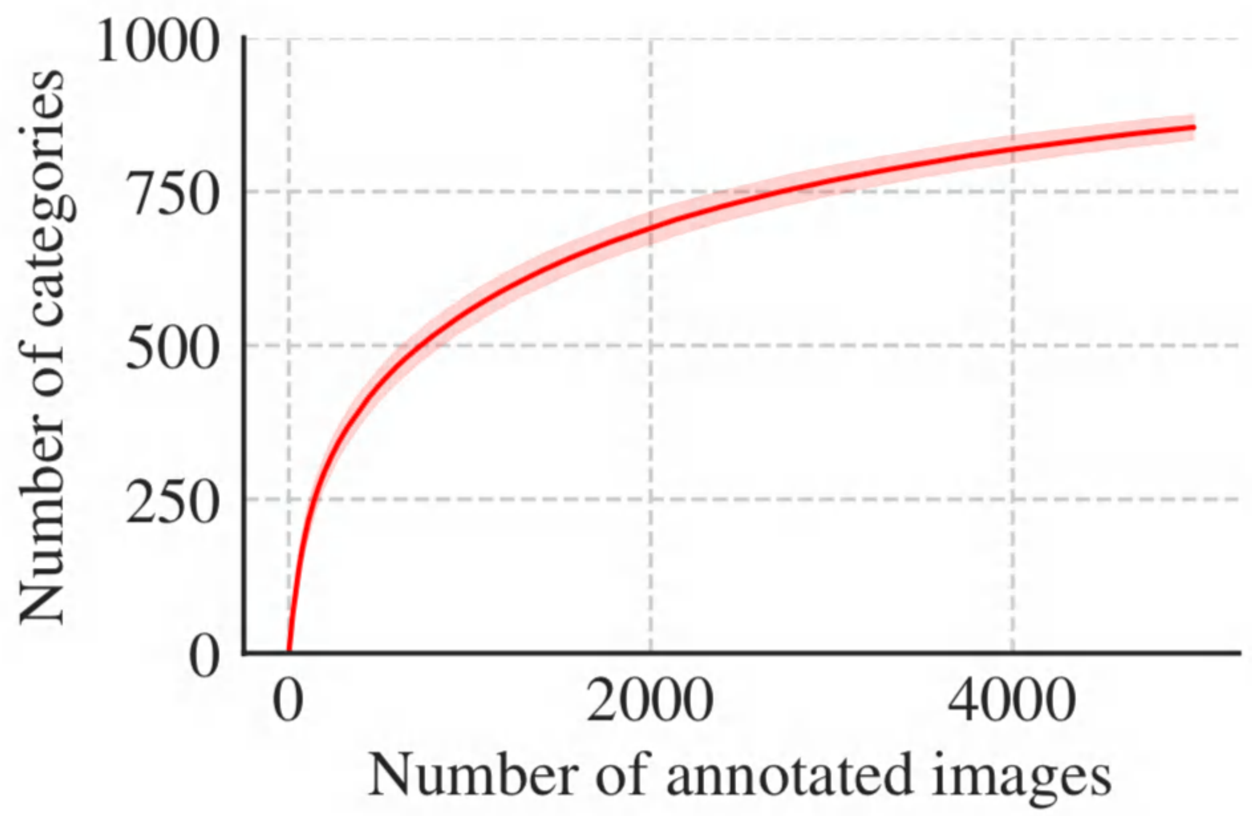
Large Vocabulary Instance Segmentation

- 164k images (from COCO)
- ~1200 *discovered* categories
- ~2.2M instance segmentations
- Long-tailed frequency distribution

Big Little Data

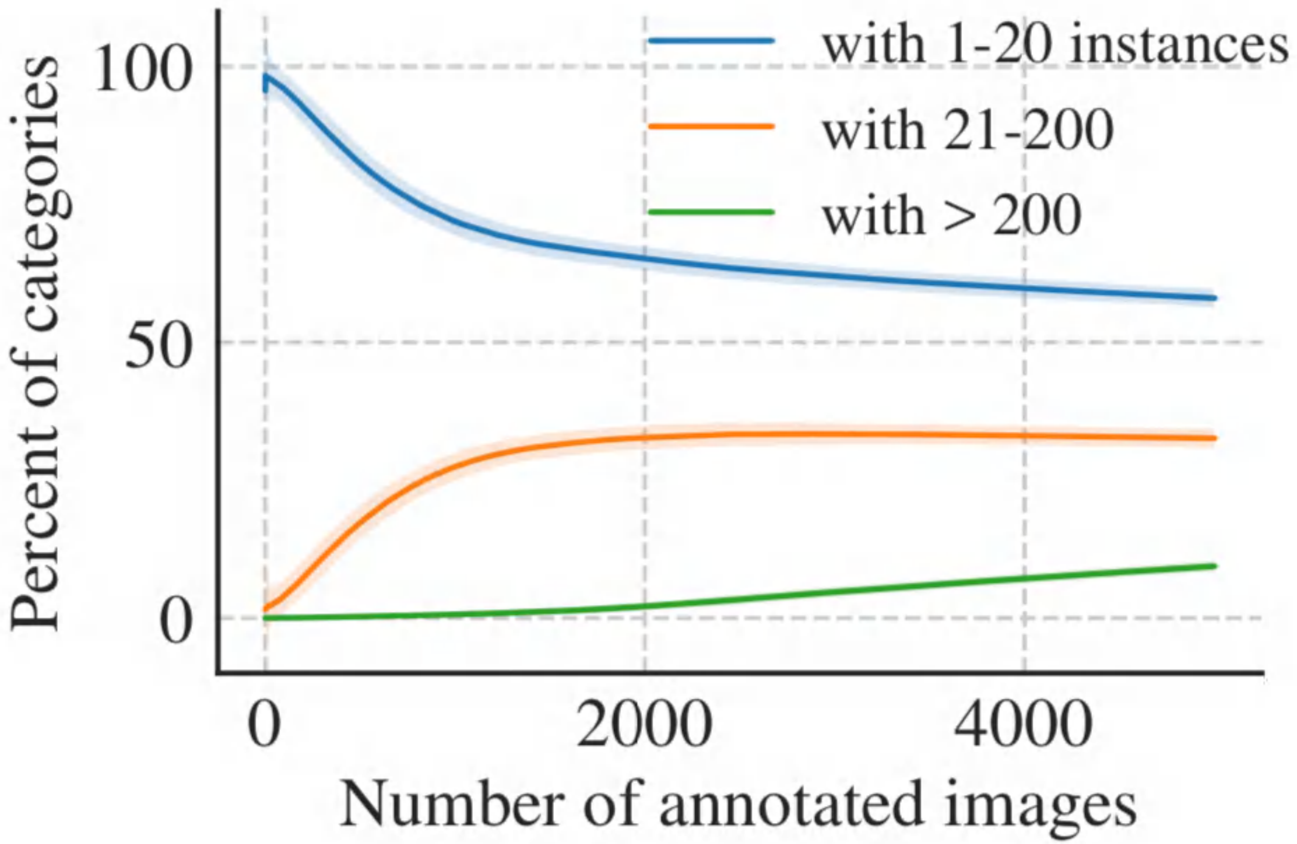


Data Driven Category Discovery



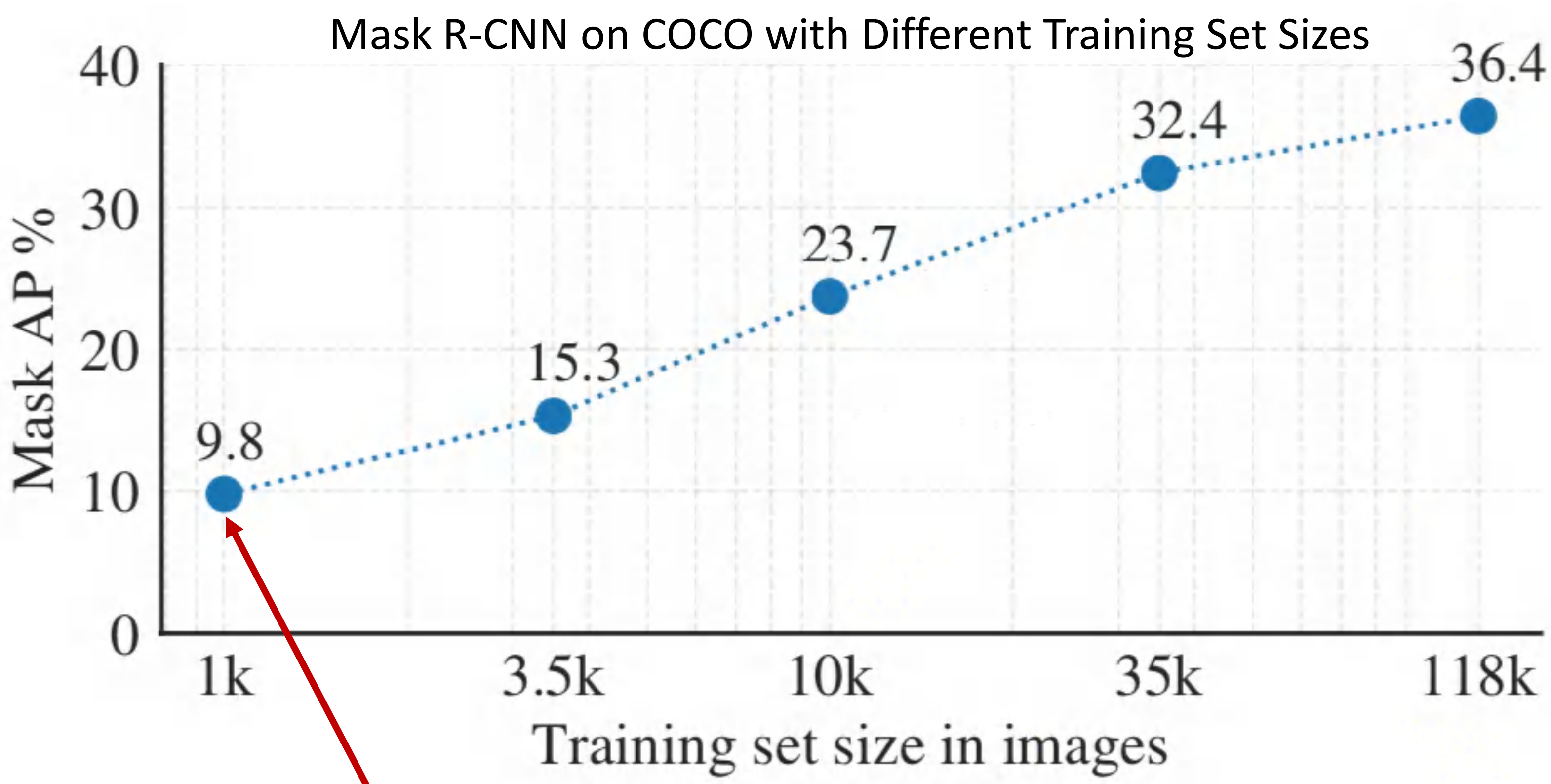
More images, more categories

The Long Tail is Inescapable!



More images, more categories → new rare categories

Data Efficient Learning



90% of categories have > 20 examples

Not even "few shot"

LVIS

(showing one category for clarity)

LVIS

(showing one category for clarity)

LVIS

(showing one category for clarity)

Preliminary Results on LVIS (40k image subset)

model	AP	AP rare	AP common	AP frequent
Mask R-CNN	16.2	<u>5.2</u>	20.0	22.2

(ResNet-101-FPN backbone)

Few-shot Learning Does Not Work at All Today

LVIS: Large Vocabulary Instance Segmentation

We’ve been designing this dataset for more than a year

Poster: Wednesday morning #81

Agrim Gupta

Piotr Dollár

Ross Girshick

Data collection should finish around November

High quality annotations, many challenges along the way

http://www.lvisdataset.org/

The LVIS Challenge at ICCV 2019!

- At the next COCO workshop (on a “teaser” beta version of LVIS)
- Doing well requires data efficient models in the Big Little Data regime

See you there!

Takeaway

1. Detection is not solved – *look to LVIS for new challenges!*
2. Features matter – *the backbone net is the engine of recognition*
3. Detector design matters – *progress from domain knowledge*
4. Generalized R-CNN – *a flexible framework with OSS code*

https://github.com/facebookresearch/detectron

detectron2 – in PyTorch – is coming later this year

FAIR Research Engineer

Menlo Park, CA
Seattle, WA

ACCELERATE AND SCALE CV RESEARCH

Familiarity with CV and ML
Ability to write high-quality and performance-critical code

wlo@fb.com

End of Slides

Additional Material not Covered at CVPR 2019

- Mask R-CNN training
- One-stage vs. multi-stage detectors & speed/accuracy tradeoffs
- AP metrics
- Other related detectors (R-FCN, …)

Mask R-CNN: Training

Same as “image centric” Fast/er R-CNN training

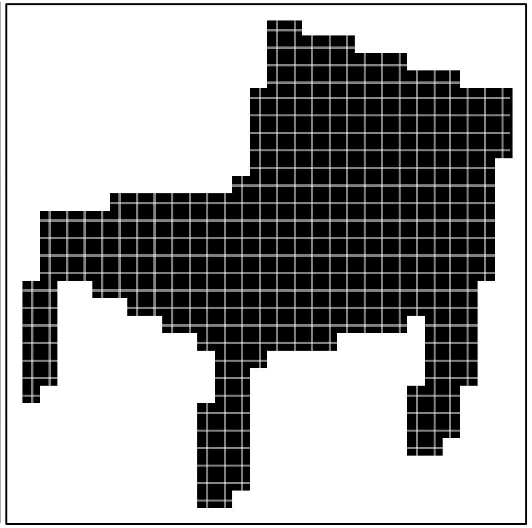
- Use precomputed proposals for faster experimentation
- Use joint / end-to-end training for sharing features & higher AP

But, with training targets for masks

Example Mask Training Targets

Image with training proposal

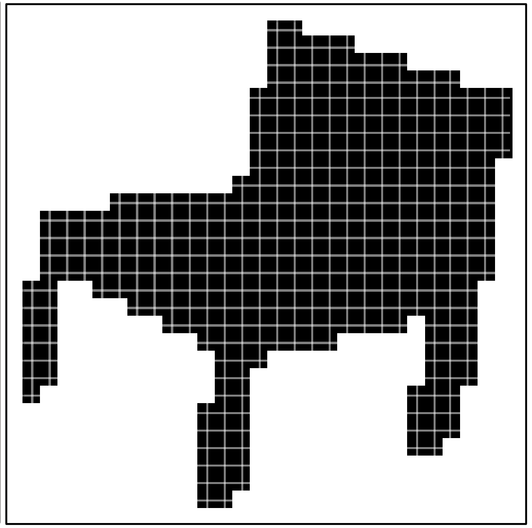
28x28 mask target

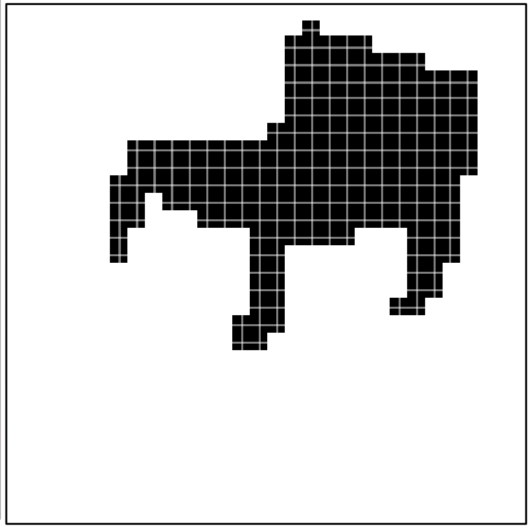


Example Mask Training Targets

Image with training proposal

28x28 mask target





Example Mask Training Targets

Image with training proposal

28x28 mask target

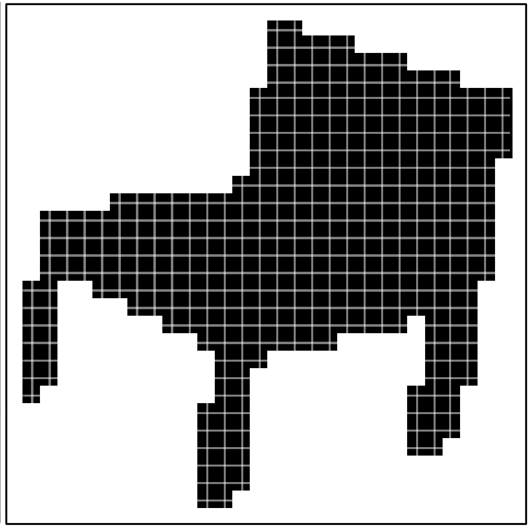
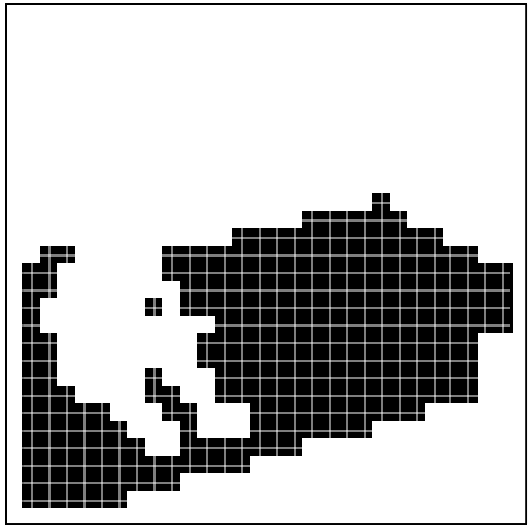
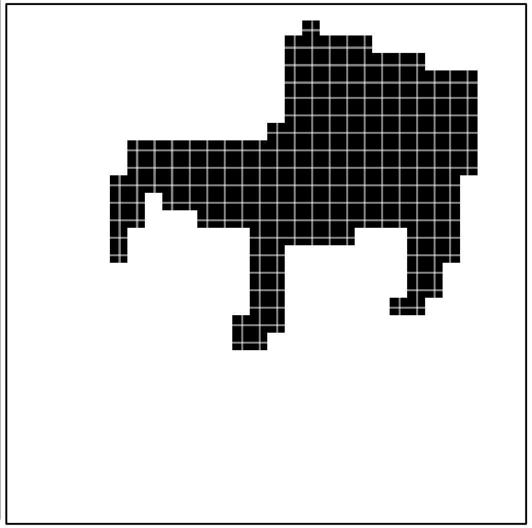


Image with training proposal

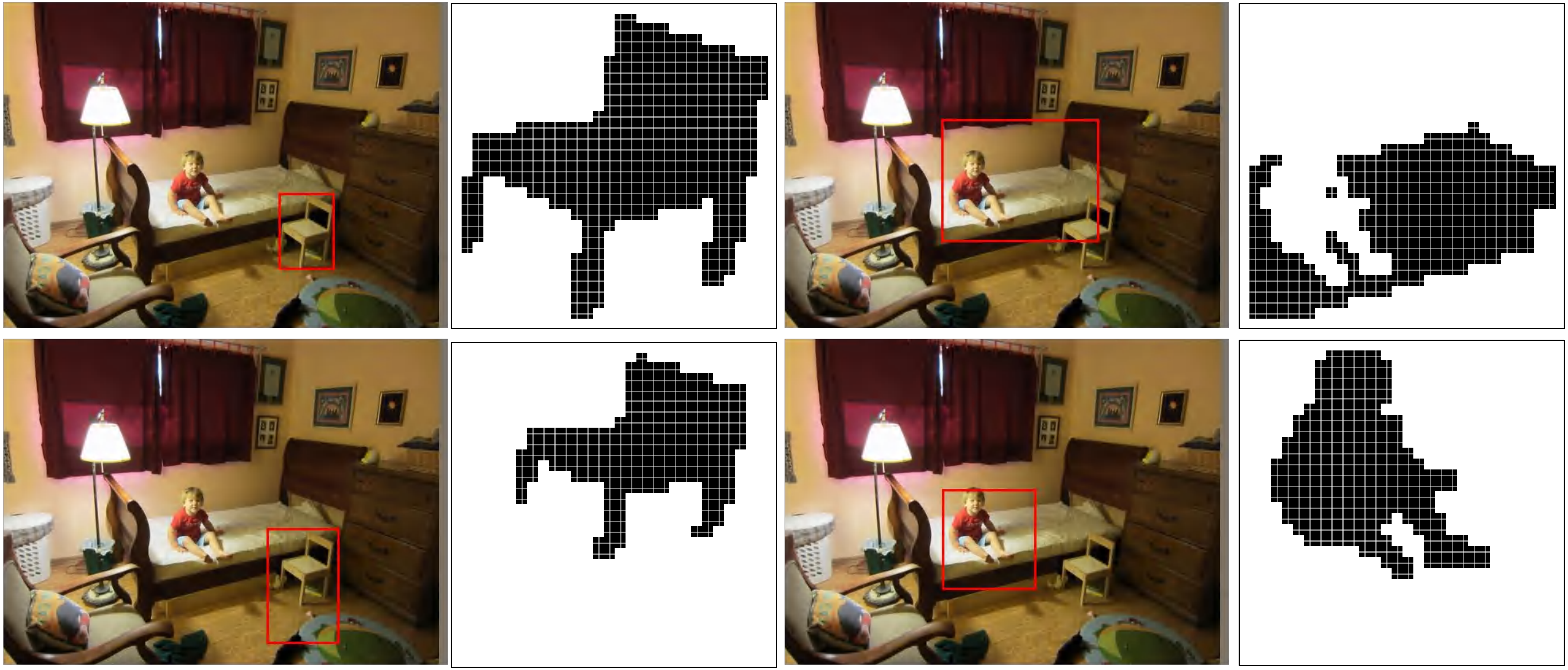
28x28 mask target





Example Mask Training Targets

Image with training proposal | 28x28 mask target | Image with training proposal | 28x28 mask target



Stages: What and Why?

Detection output space: $N = H \times W$ pixel image has $O(N^2)$ boxes

Output space is HUGE, even for small images it's *billions of boxes*

Number of target objects is small, say 10

Massive foreground / background class imbalance

How do we Deal with Class Imbalance?

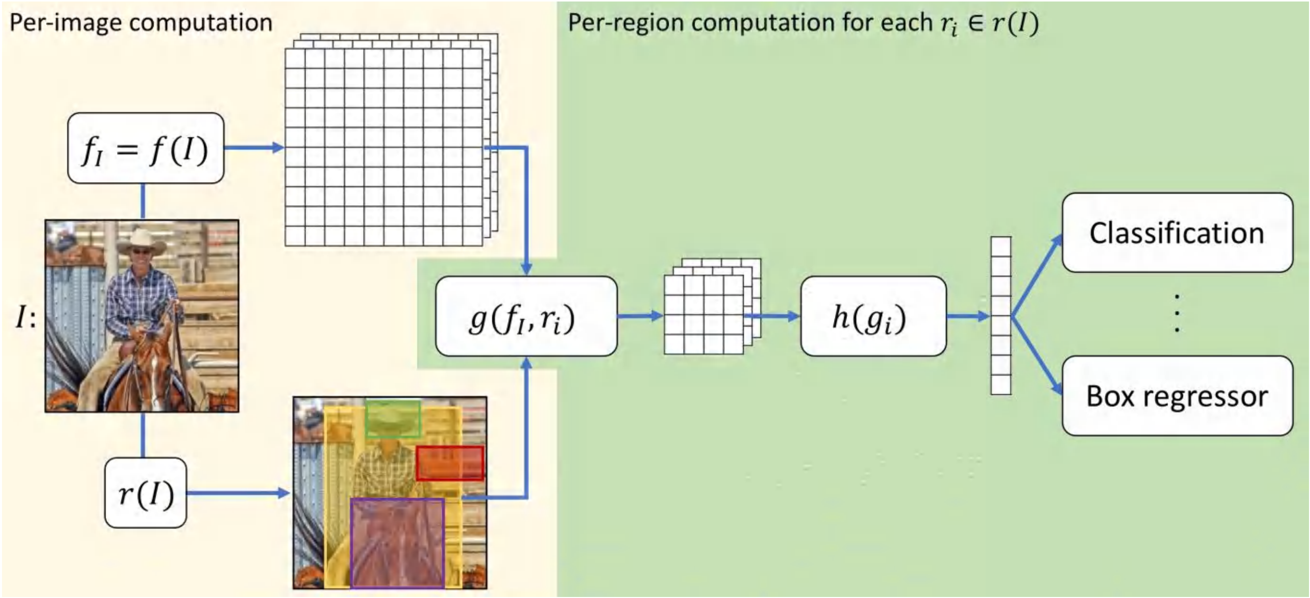
1. Subsample the output space while maintaining *high recall*

- ➢ Sliding window
- ➢ Object proposals

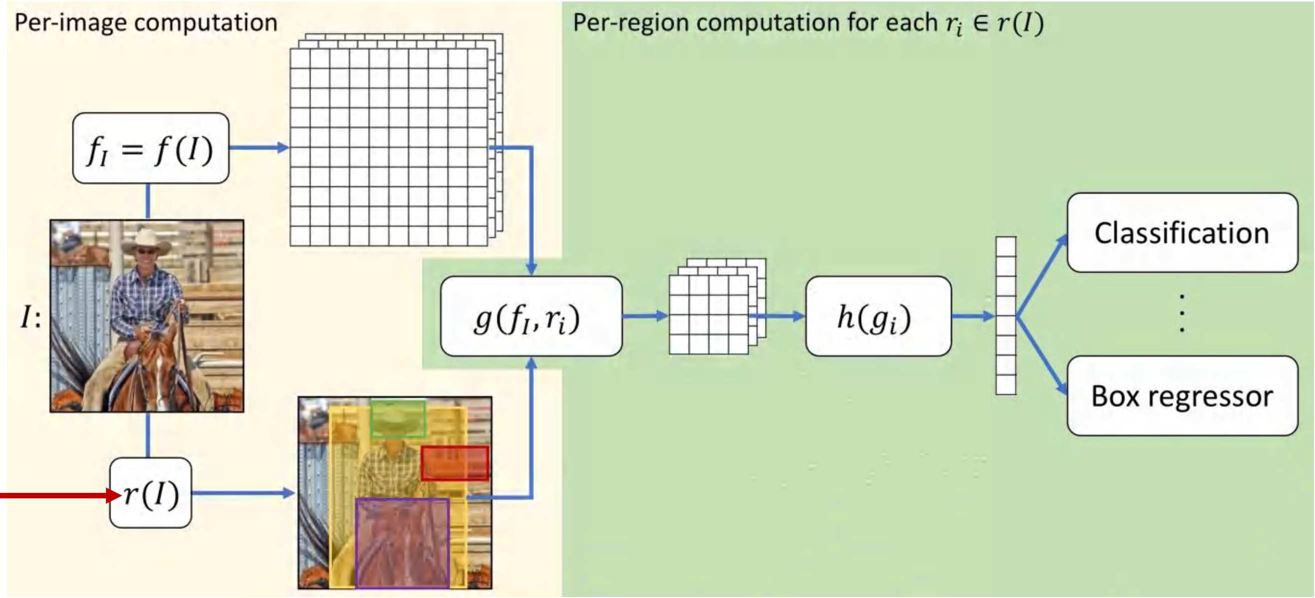
2. Classify boxes in a *cascade of stages*

- ➢ Most famous example: Viola-Jones detector
- ➢ Typically combined with subsampling
- ➢ *Note that subsampling is already an implicit stage*

Example of two stages using proposals: Generalized R-CNN Framework



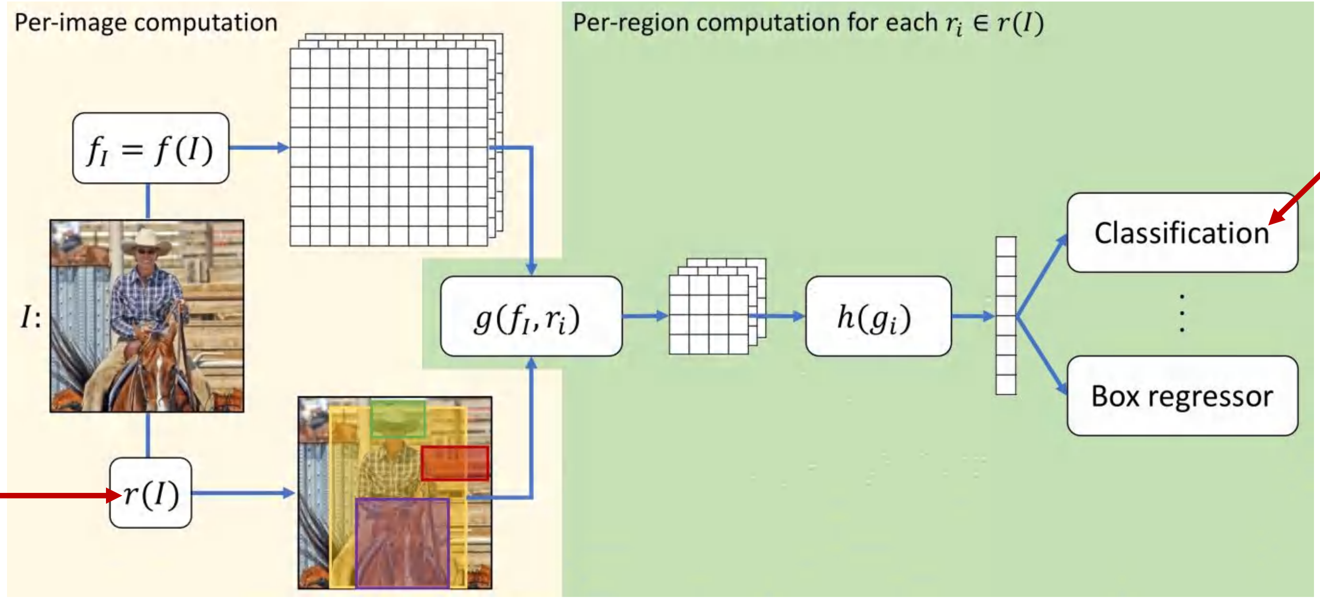
Example of two stages using proposals: Generalized R-CNN Framework



1. Proposals eliminate most of the output space

Example of two stages using proposals: Generalized R-CNN Framework

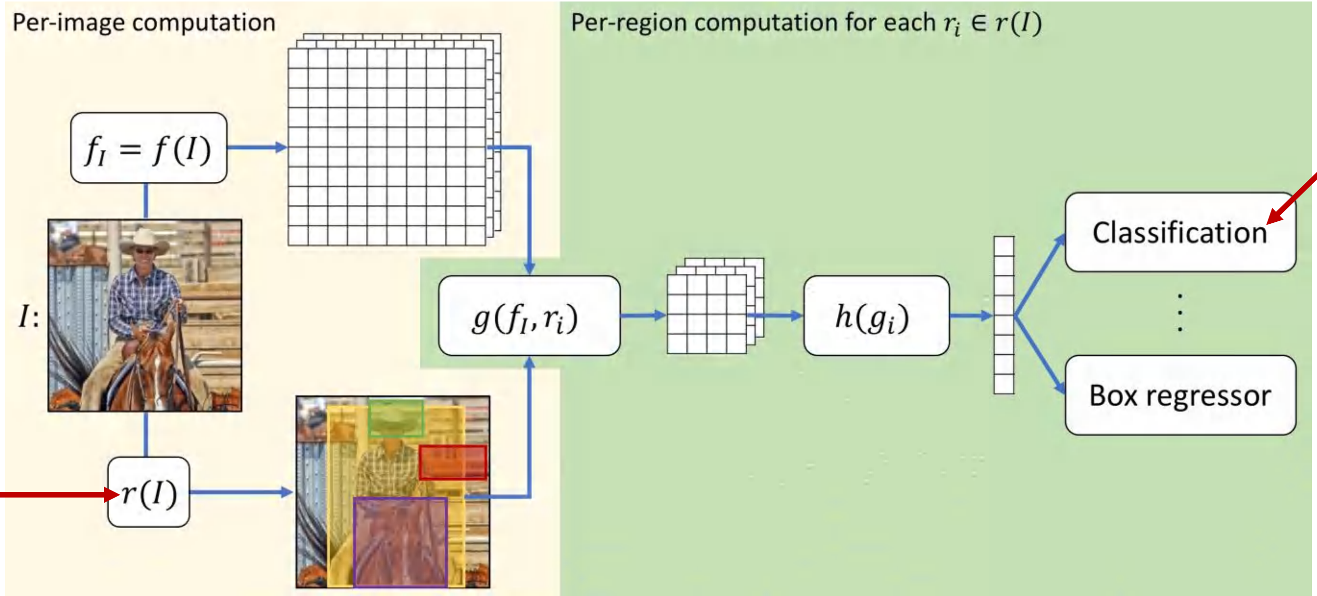
2. Classification on small subset of output space



1. Proposals eliminate most of the output space

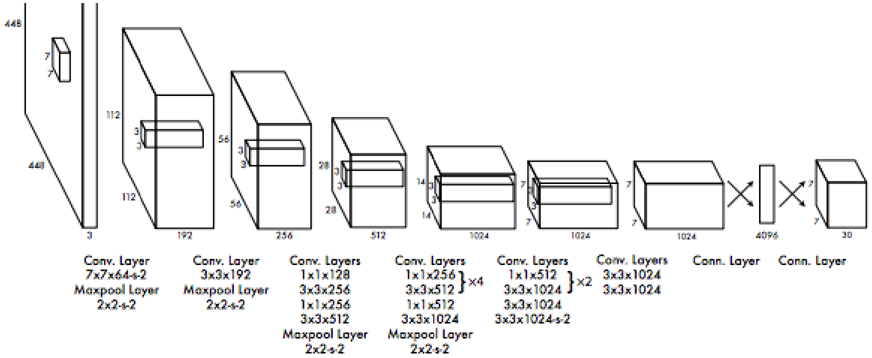
Example of two stages using proposals: Generalized R-CNN Framework

2. Classification on small subset of output space



1. Proposals eliminate most of the output space

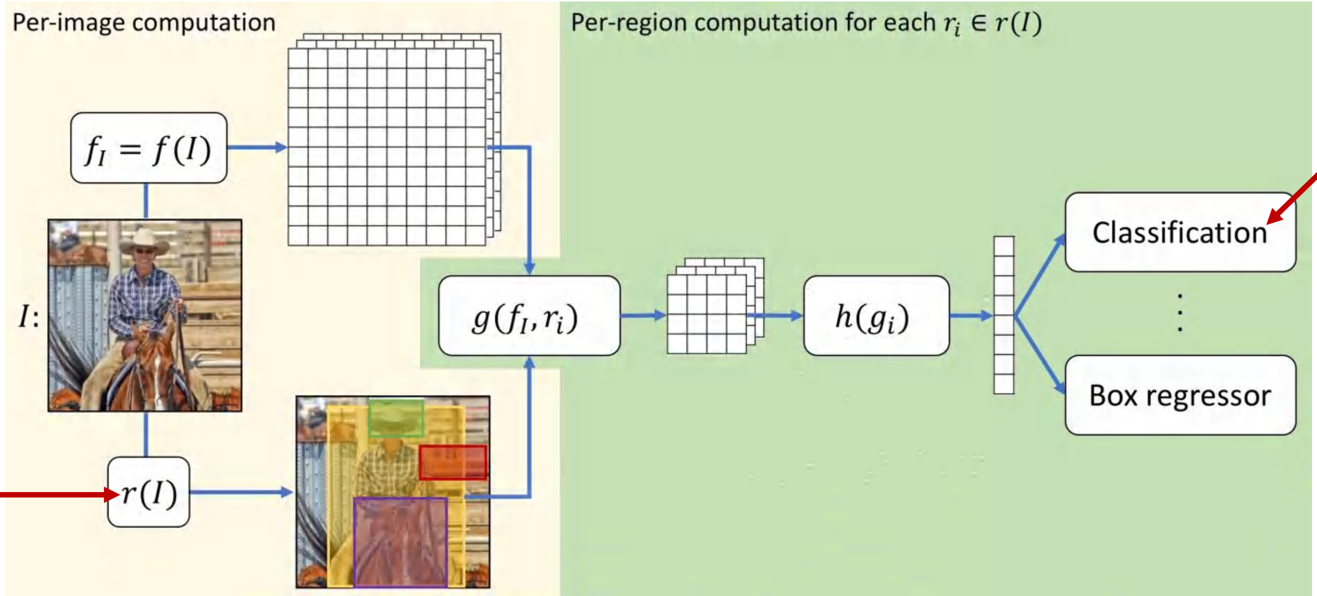
Example one stage using *dramatic* subsampling



Redmond et al. You Only Look Once:
Unified Real-time Object Detection. CVPR 2016.

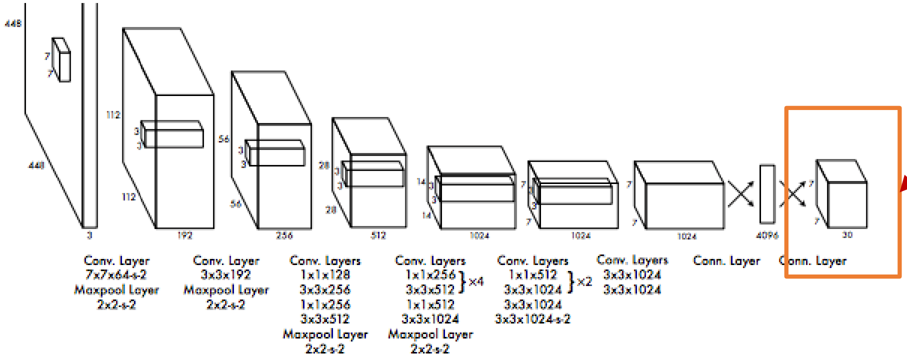
Example of two stages using proposals: Generalized R-CNN Framework

2. Classification on small subset of output space



1. Proposals eliminate most of the output space

Example one stage using *dramatic* subsampling



Redmond et al. You Only Look Once: Unified Real-time Object Detection. CVPR 2016.

1. Consider a *tiny* subset of the output space by design; directly classify this small set of boxes

"You only look once" = "Single shot" = "One stage"

Subsampled Output Space [+ Hard Example Mining]

Central challenge for one-stage detectors: class imbalance

- YOLOv1 – 98 boxes
- YOLOv2 – ~ 1k
- OverFeat – ~ 1-2k
- SSD – ~ 8–26k (hard-example mining)

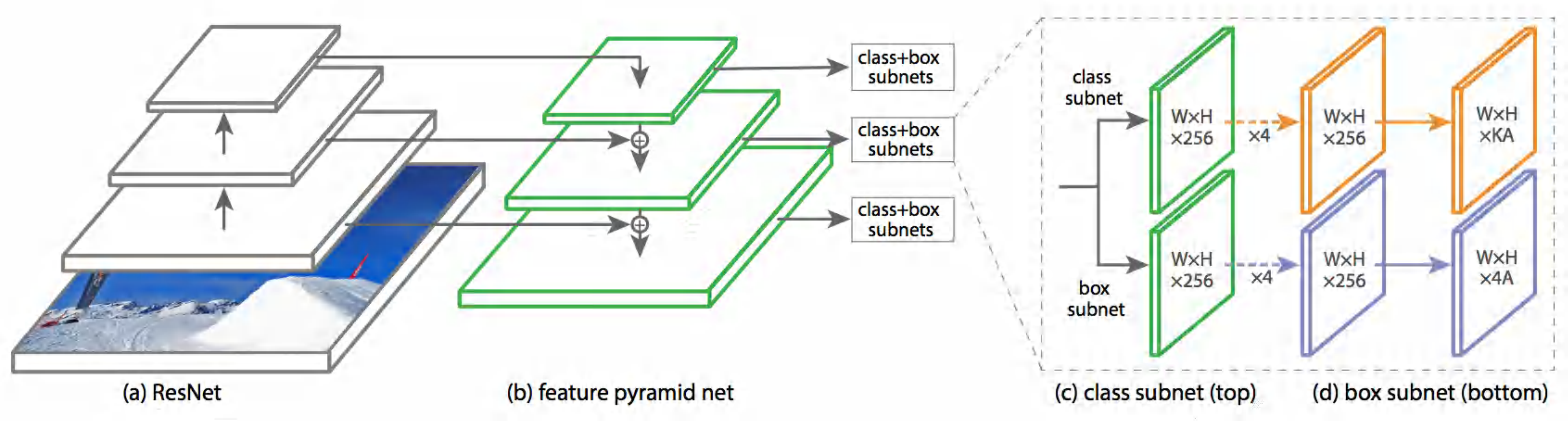
- RetinaNet – ~ 100-200k ("soft" hard-example mining)

RetinaNet

References

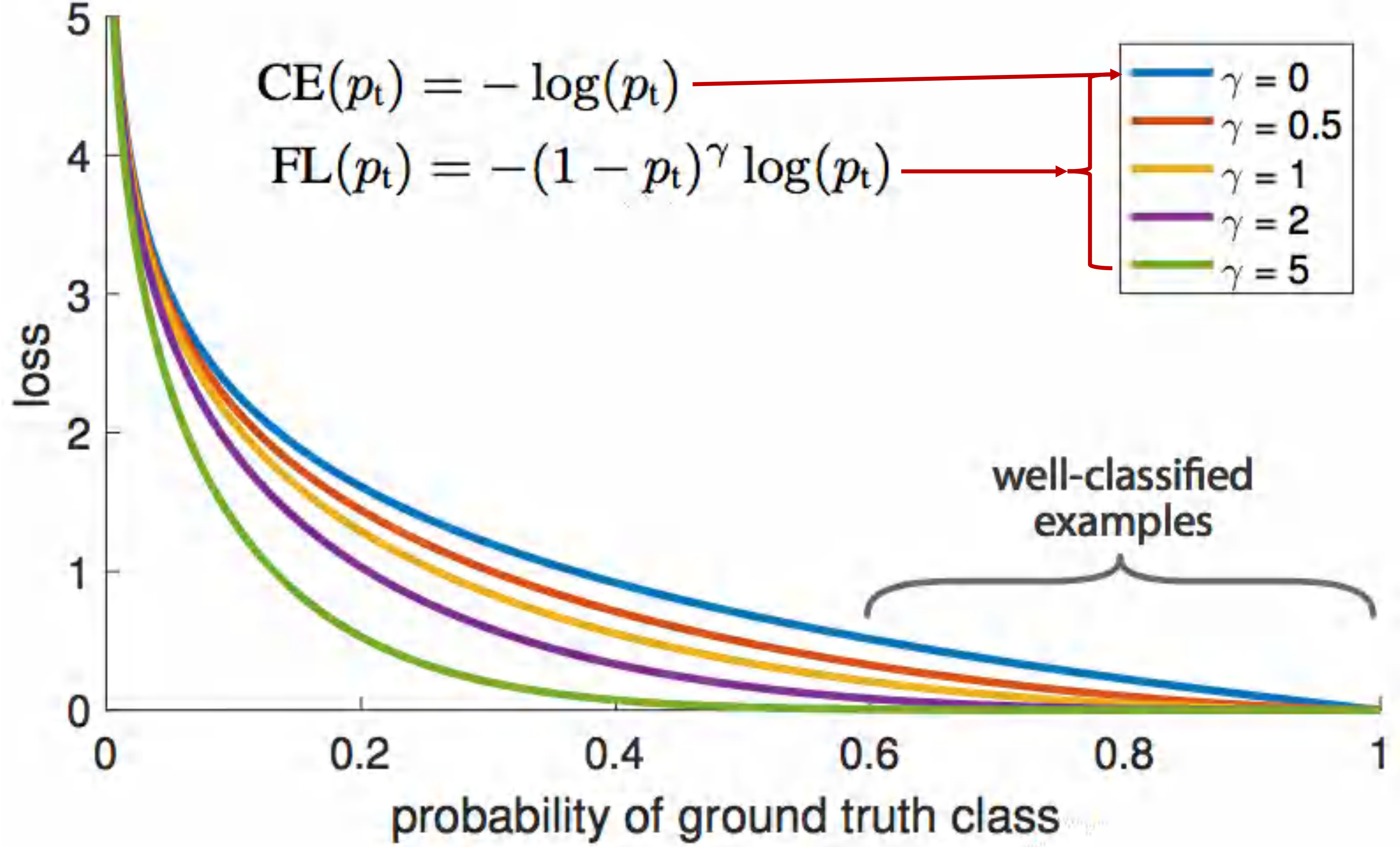
- P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In ICLR, 2014.
- A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In CVPR, 2016.
- P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In NIPS, 2015.

RetinaNet Model Description



- ➢ Backbone with FPN + class-specific RPN (final detections)
- ➢ 6 anchors per location (2 scales × 3 aspect ratios)
- ➢ 100 – 200k anchor boxes to classify per image → "dense" detection

Focal Loss: "Soft" Hard-Example Mining

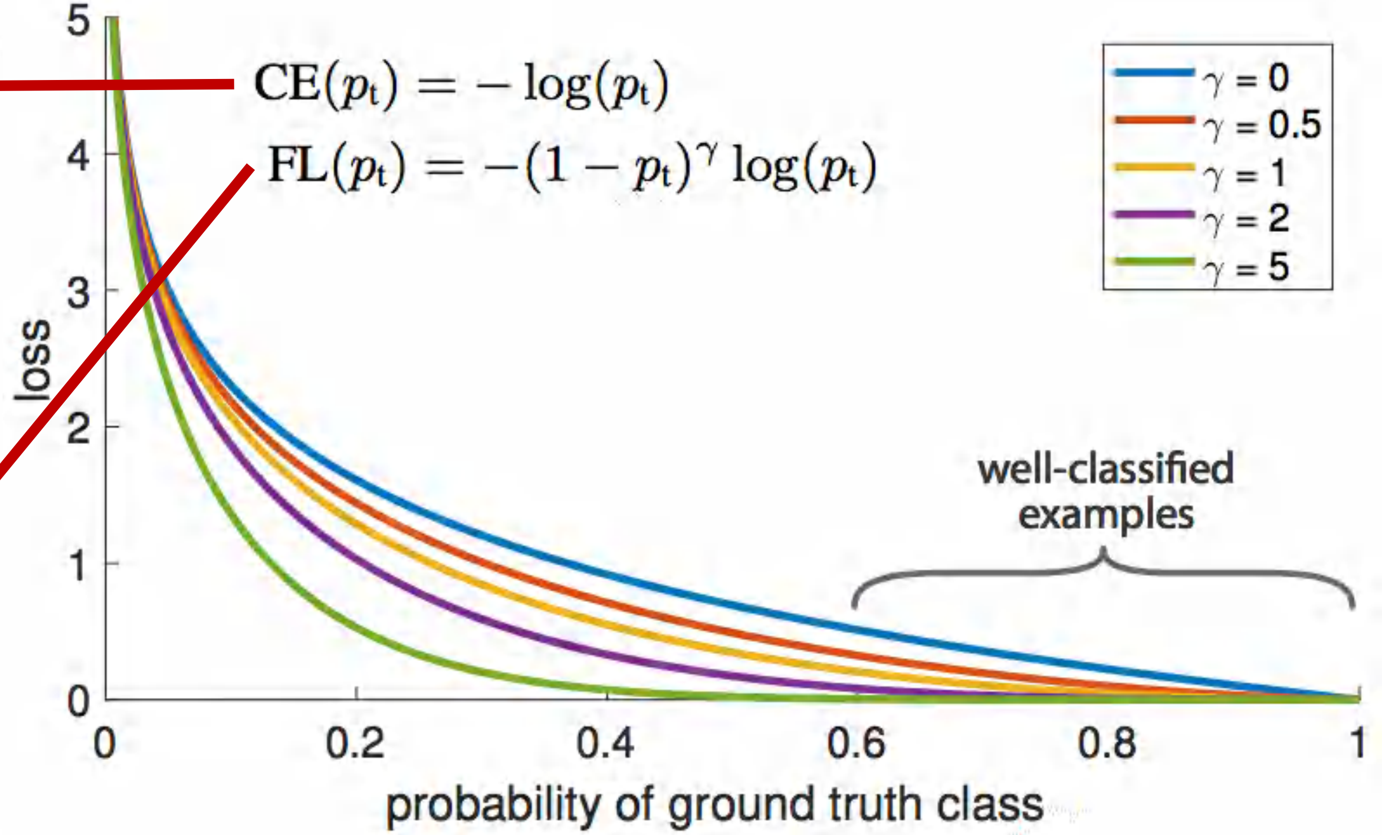


Impact of Focal Loss (FL)

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

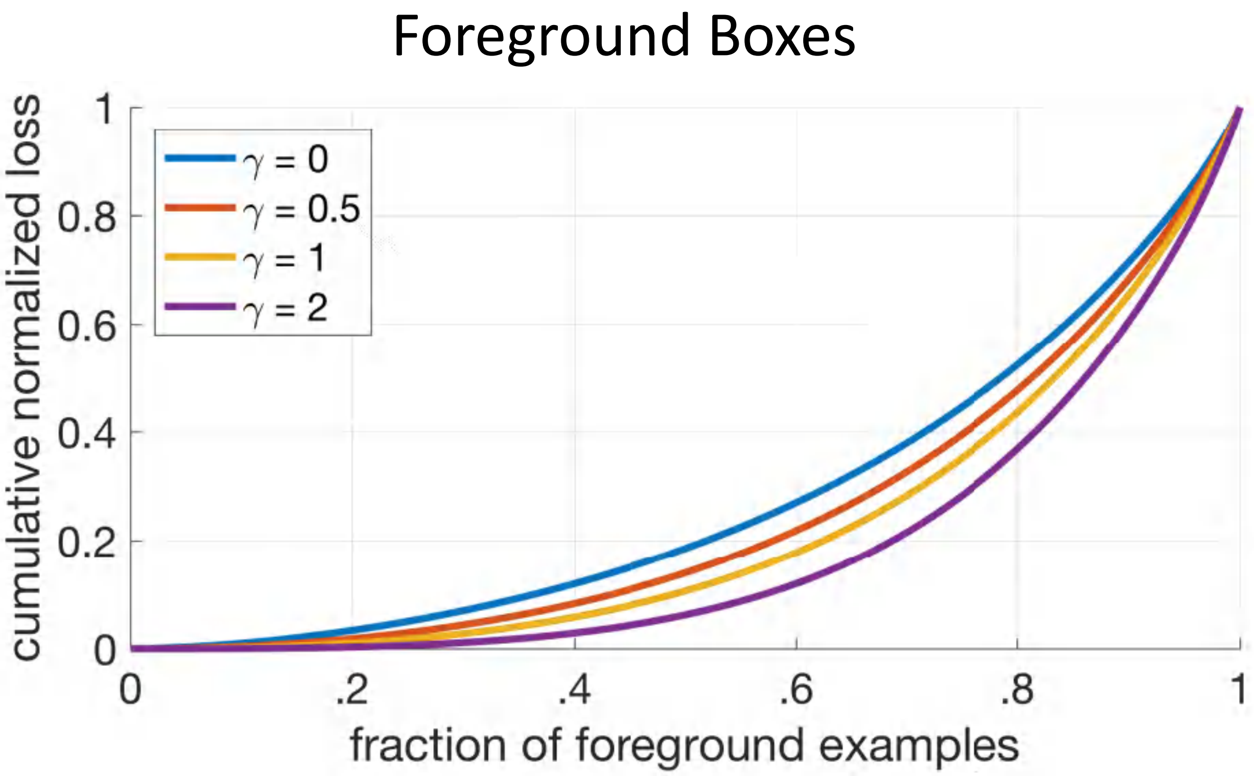
(b) **Varying γ for FL** (w. optimal α)

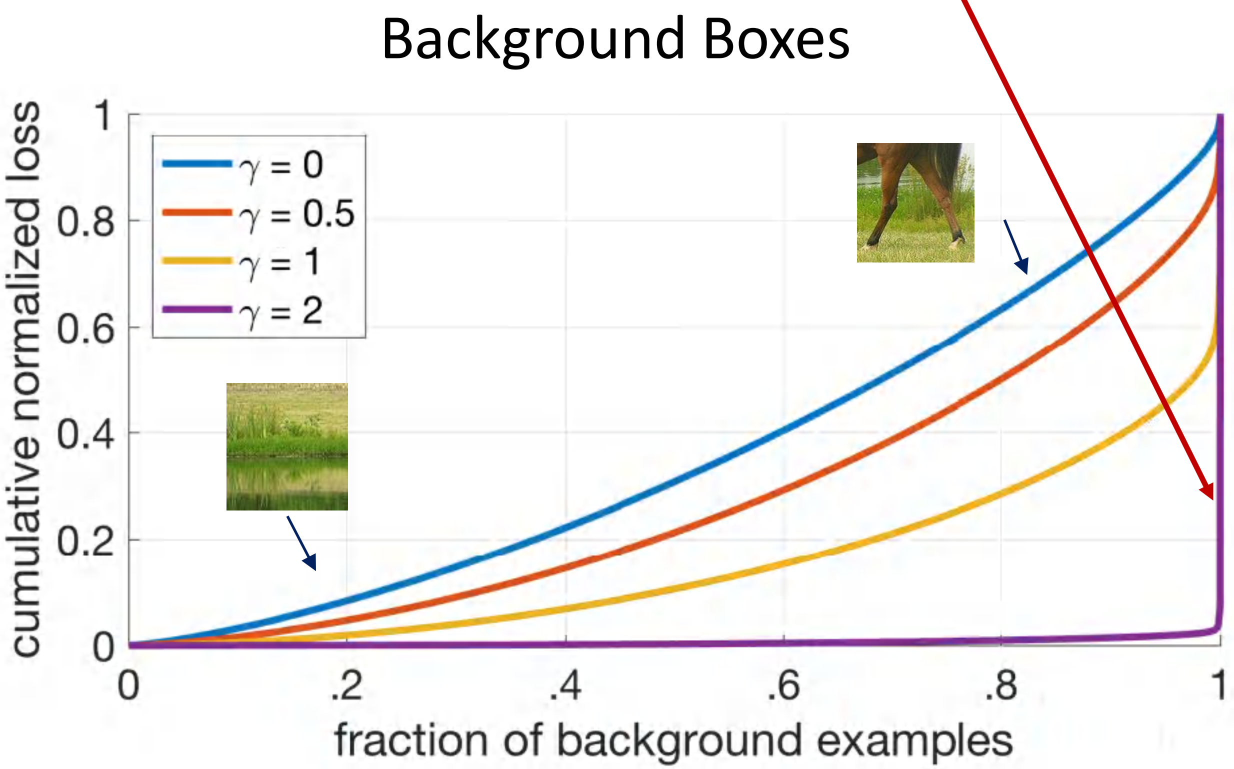
(ResNet-50-FPN 600px input image)



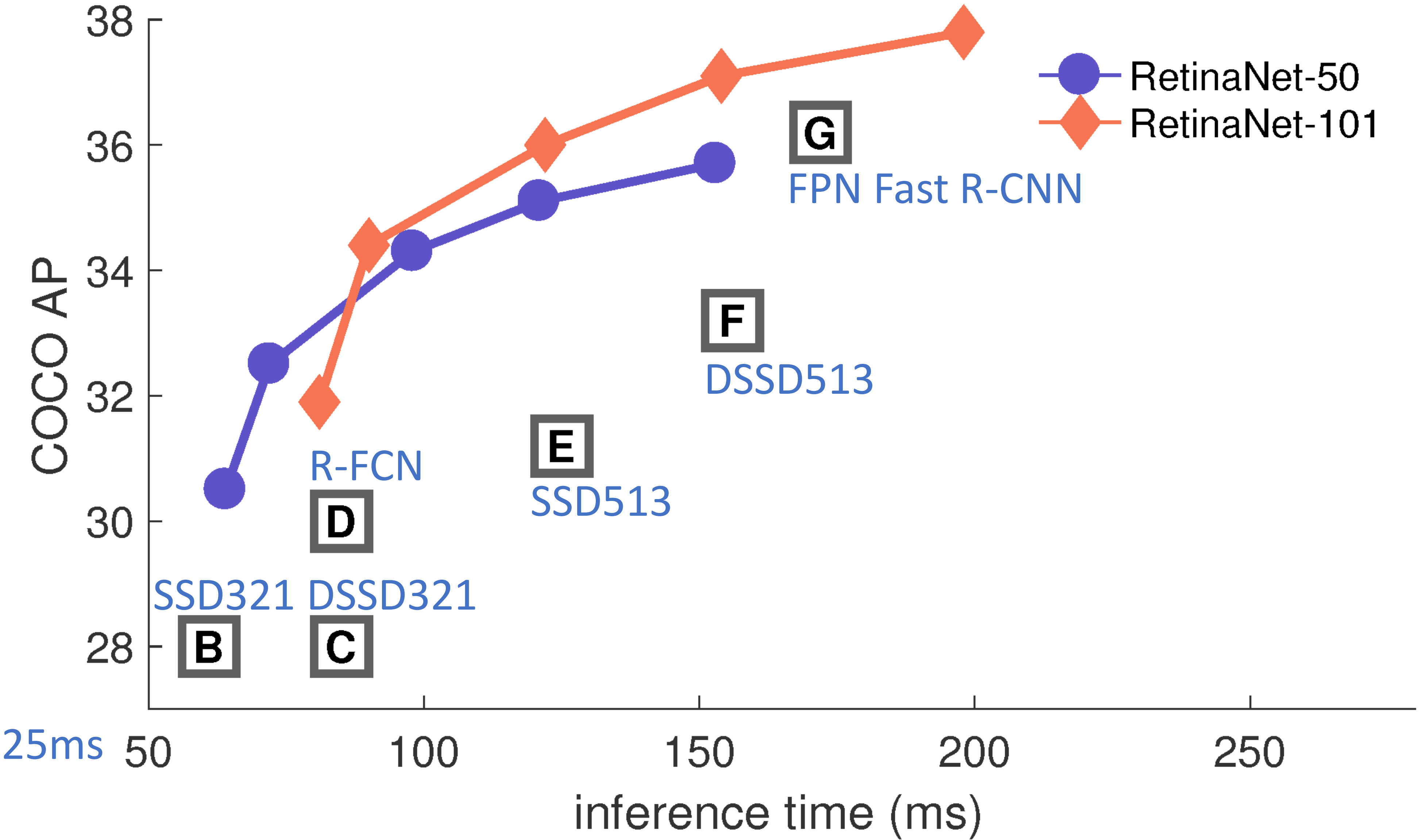
Loss Distribution under Focal Loss

Hard mining behavior



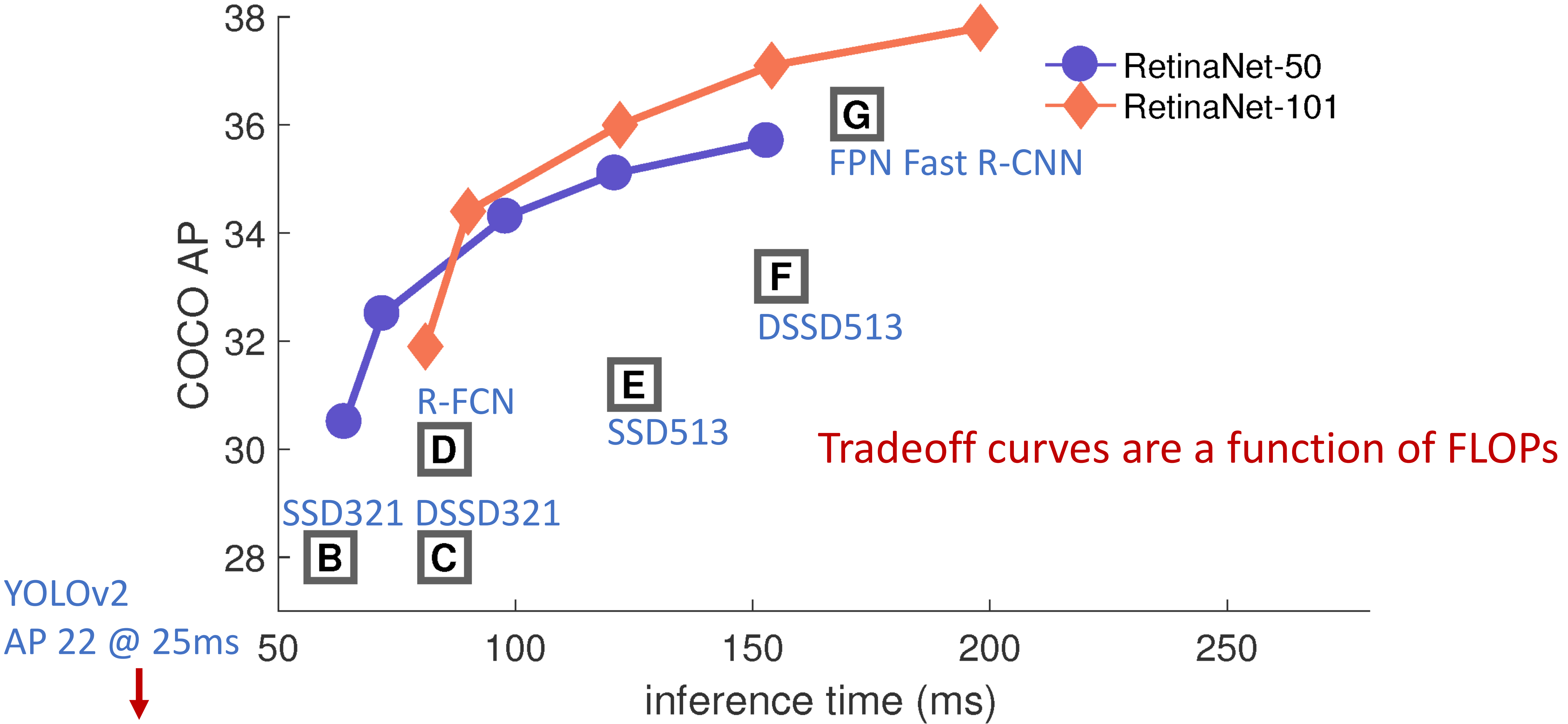


Speed/Accuracy Tradeoff



YOLOv2
AP 22 @ 25ms

Speed/Accuracy Tradeoff

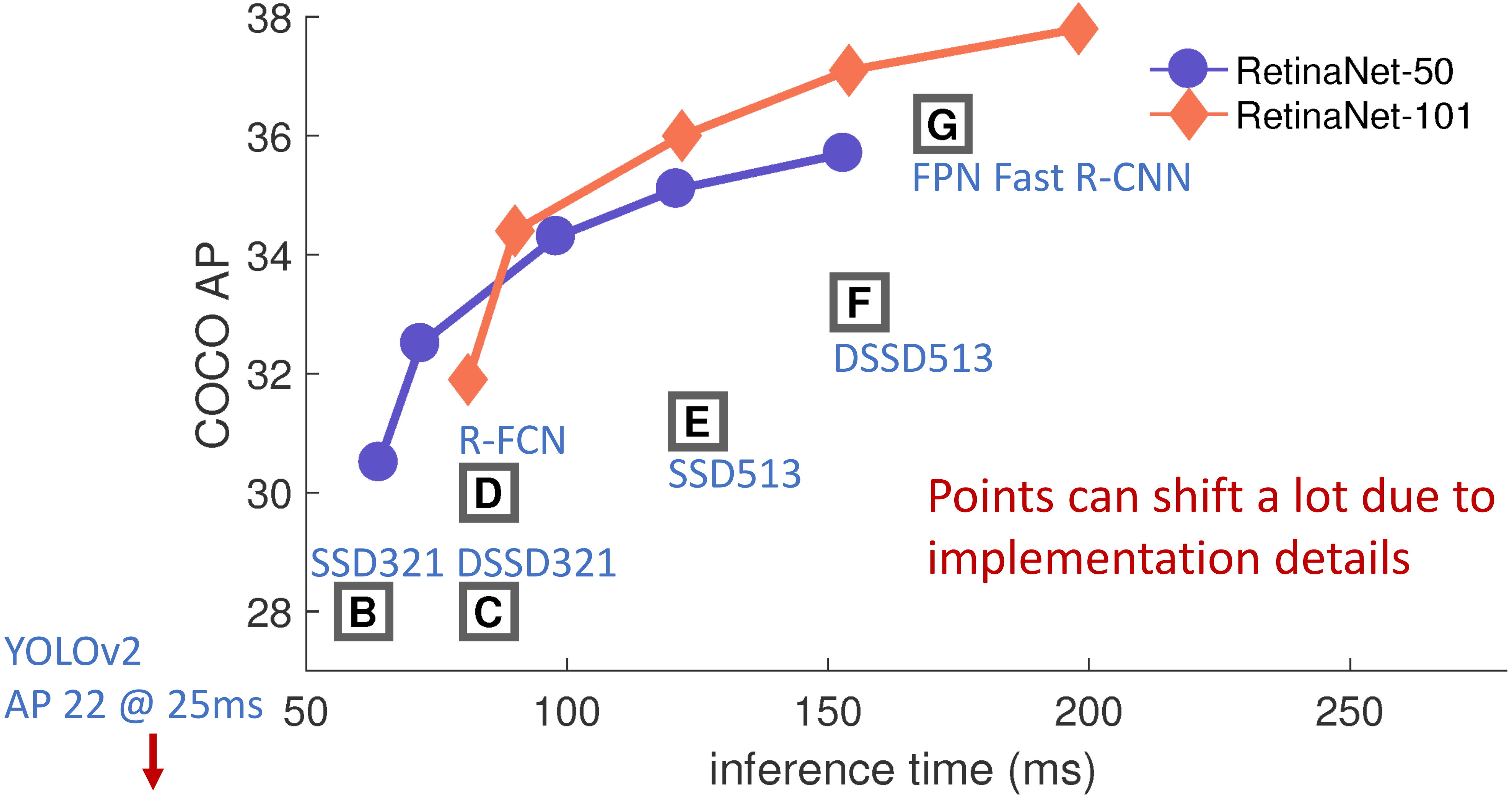


Warning about Speed/Accuracy Tradeoffs

Comparisons across publications are *completely uncontrolled*

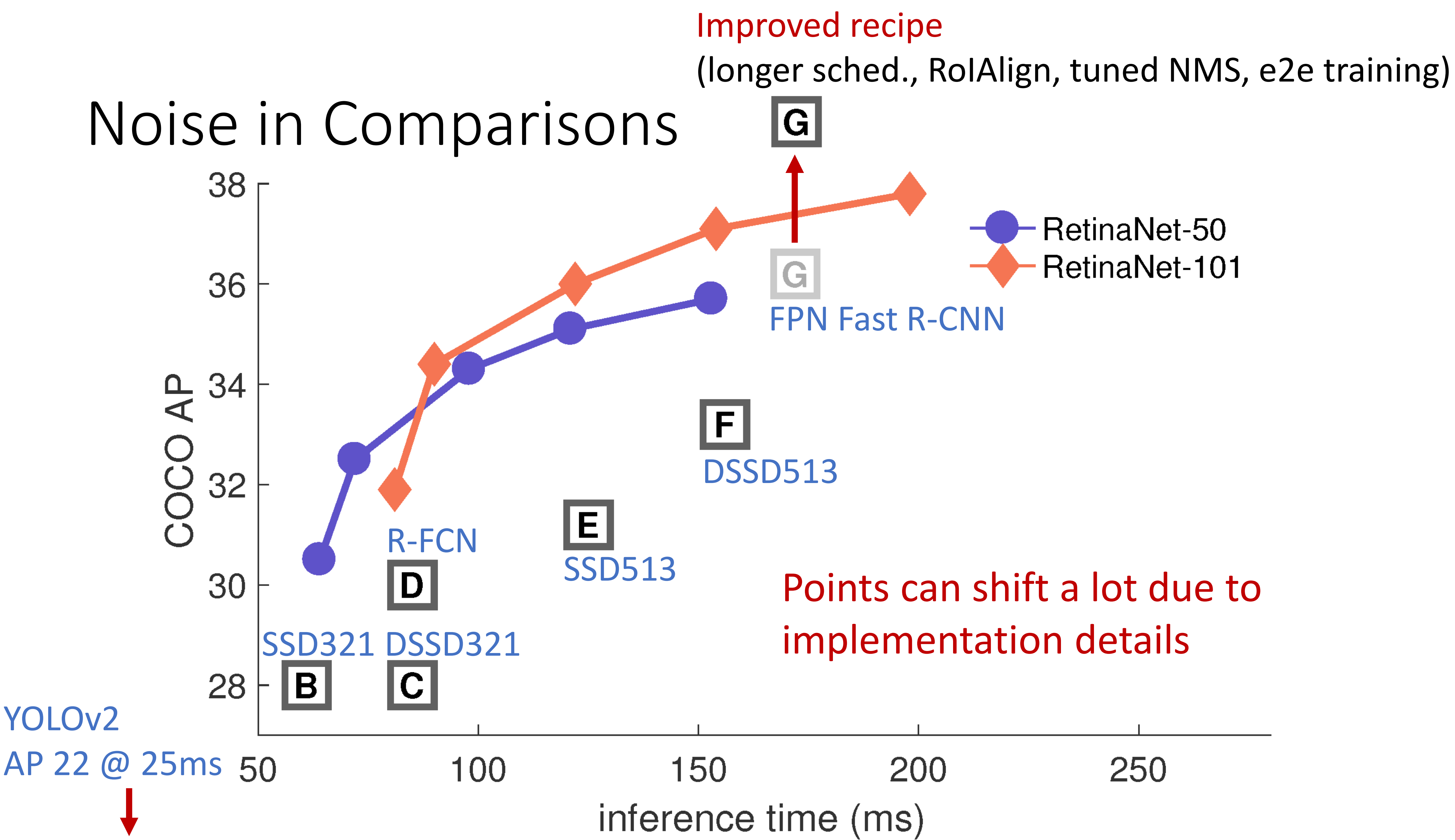
- Results should be taken with a grain of salt
- Accuracy varies with hyper-parameters ('recipe')
- Speed varies with low-level optimization (perf tuning)
- Speed varies with 'tricks' (e.g., batching during inference)

Noise in Comparisons



Points can shift a lot due to implementation details

Noise in Comparisons



Fast Detectors

Key design factors (*it's all about the FLOPs*)

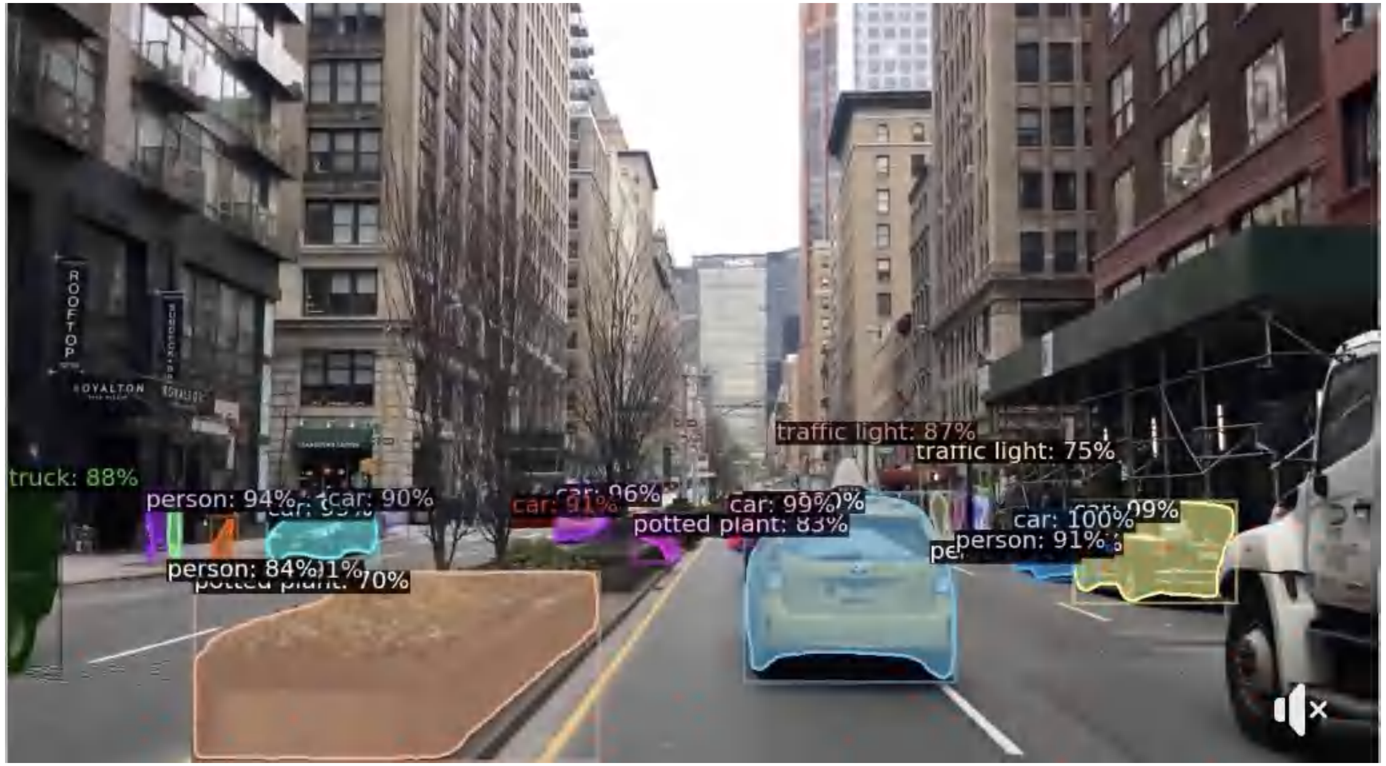
- Low resolution input
- Lightweight backbone network
- Top-*k* proposals with small *k* = 10-50 (if applicable)

"Anti-factors"

- One-stage instead of two-stage

Evaluating Visual Object Detectors

Example output of an object detection system on video



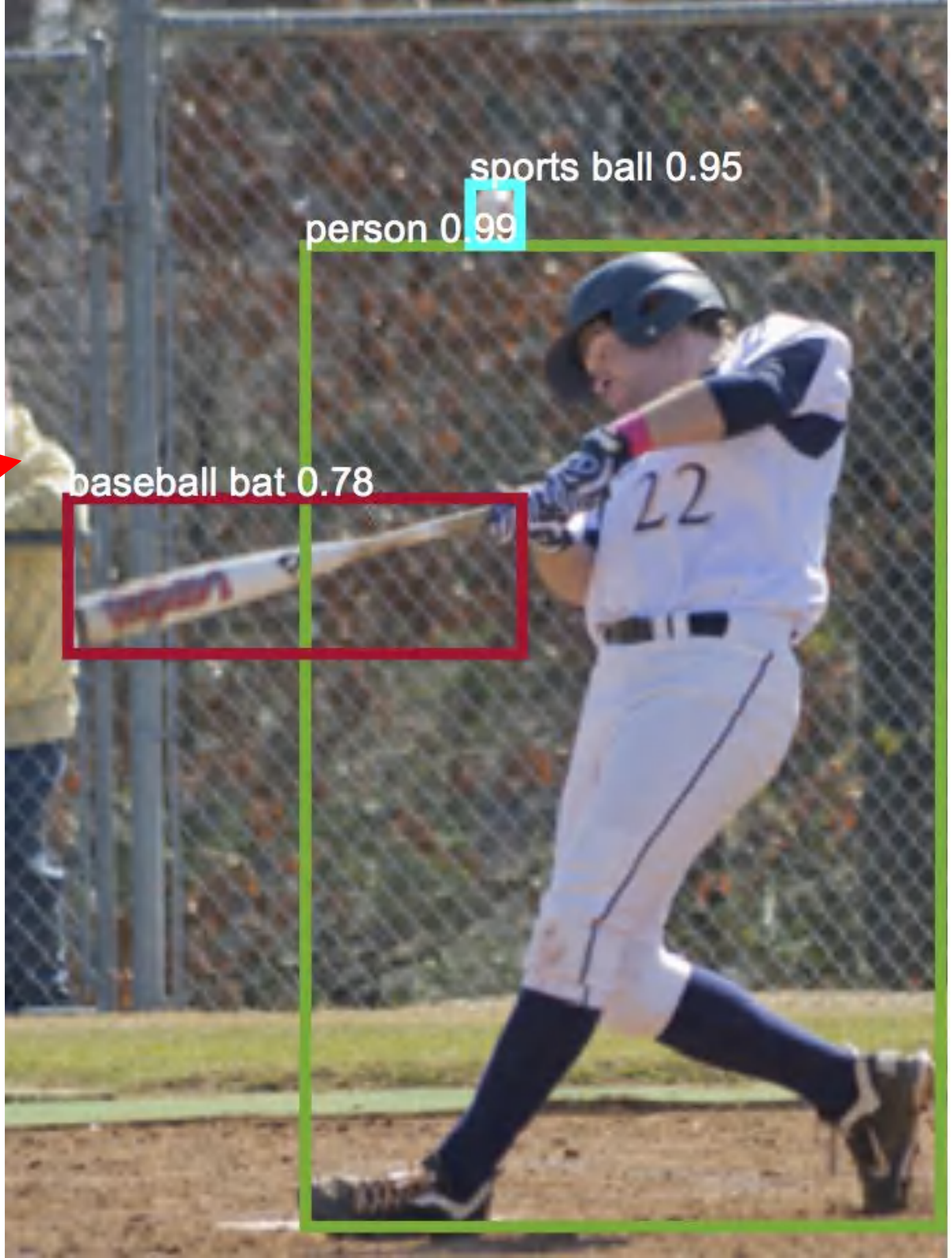
https://fb.workplace.com/100023184090772/videos/478081612974638/


kite
(ground truth)

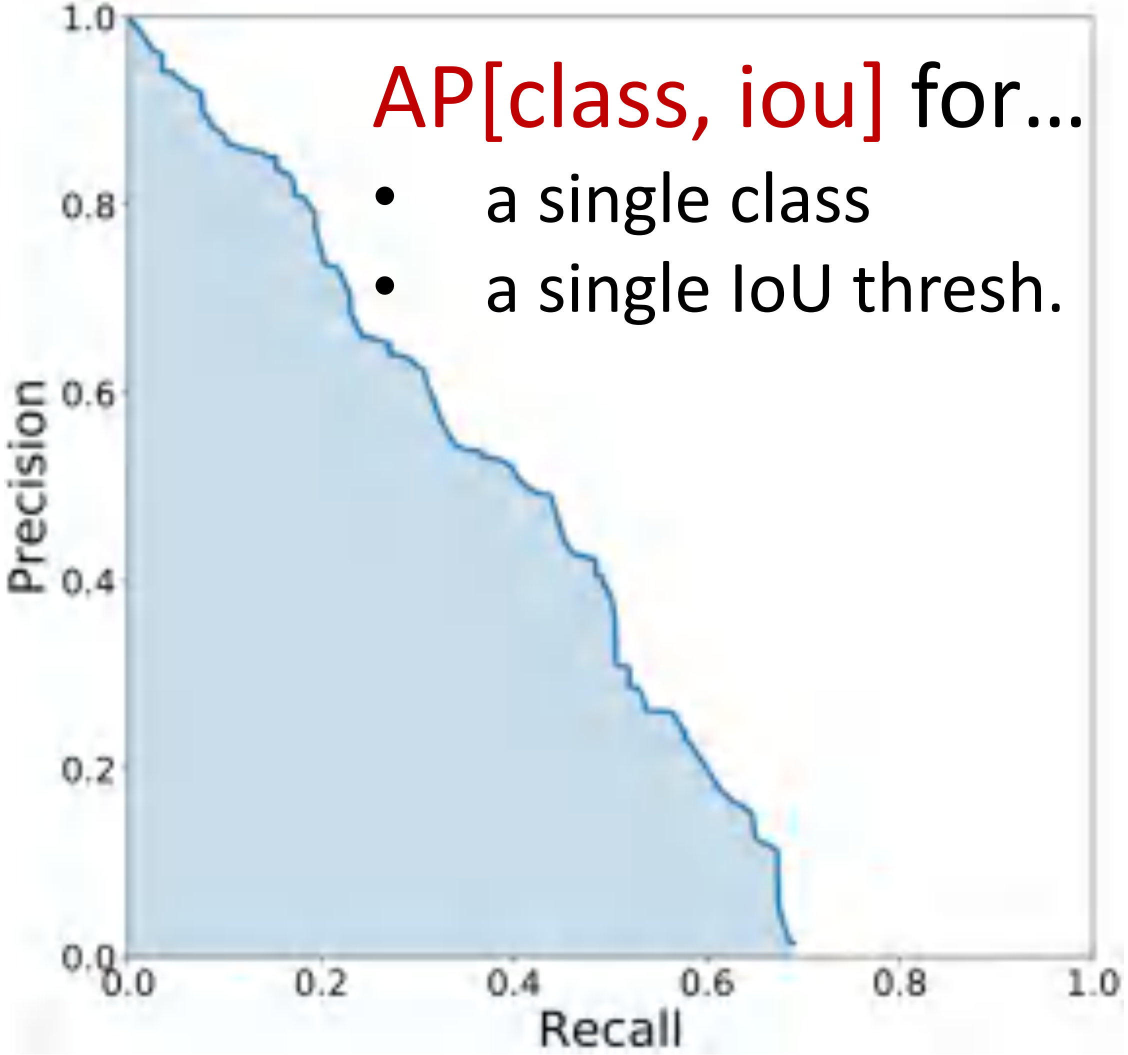

Kite, conf score = 0.97 (prediction)
kite
(ground truth)
IoU = 0.65
>= threshold?

False Positive
IoU < thresh
(or duplicate)

False Negative
(missing detection
of a person at image
edge)



Average Precision for a (class, iou threshold) pair

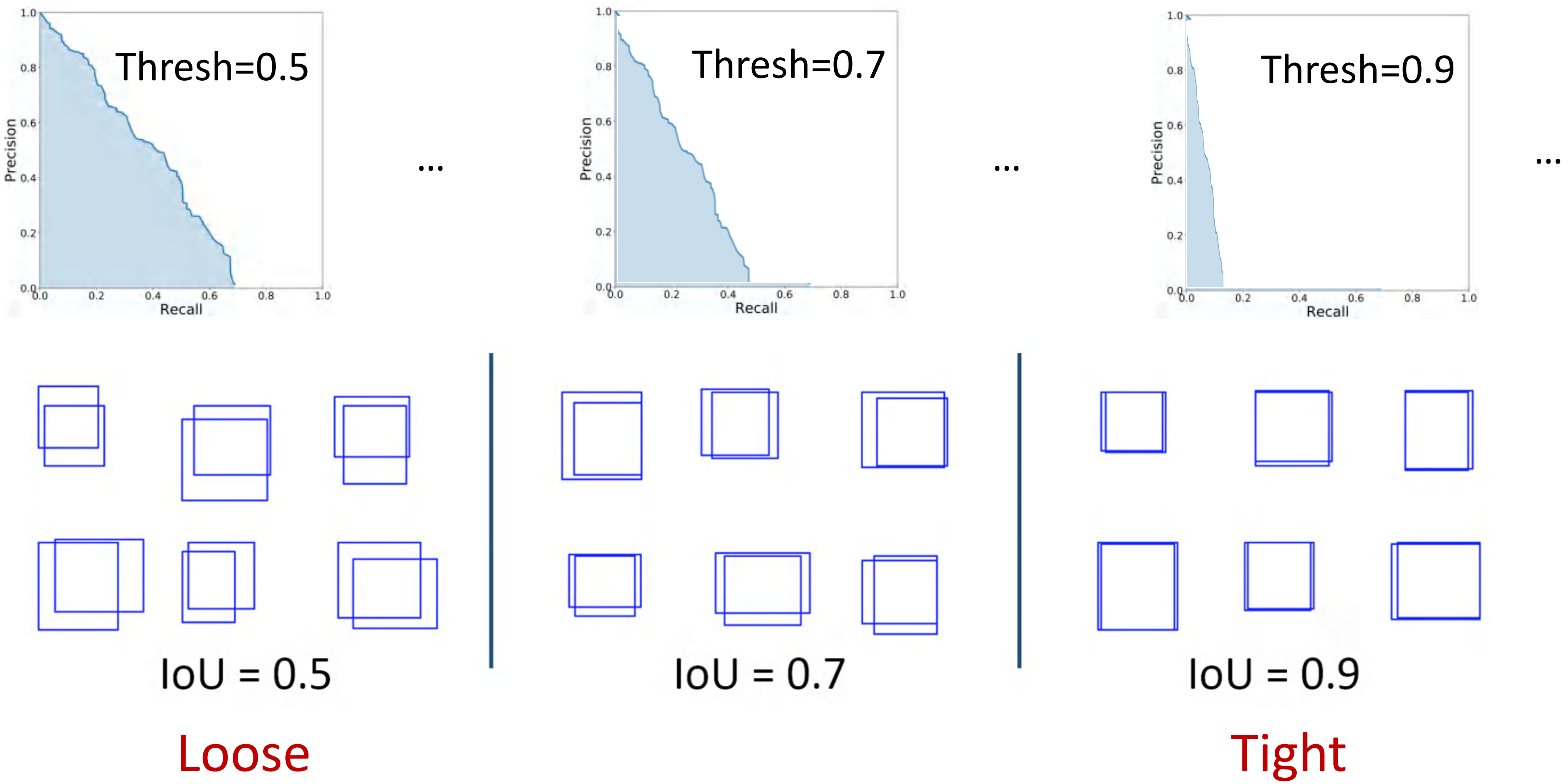


AP[class, iou] for…

- a single class
- a single IoU thresh.

Average Precision for a class

$$\mathrm{AP[class]} = \frac{1}{\#\mathrm{thresholds}} \sum_{\mathrm{iou} \in \mathrm{thresholds}} \mathrm{AP[class,iou]}$$



Overall Average Precision (%)

$$\mathrm{AP} = \frac{1}{\#\mathrm{classes}} \sum_{\mathrm{class} \in \mathrm{classes}} \mathrm{AP}[\mathrm{class}]$$

Simply average per-class AP
"AP" is really an *average, average, average* precision

classes   iou thresholds   precision @ different recall levels

Problem Dimensions and Characteristics

- # object classes (*person*, *kite*, …): ~1000
- # validation images: 5k
- # test images: 20k
- Systems to compare: 2 to 20
- Heavy class imbalance (Zipfian distribution)
  - Many classes have 1 – 10 object instances in the val / test sets
  - Others have 100 – 1000+
- Per-class AP is noisy for classes with few test instances

Goal: Compare Different Systems

- Is one object detection system better than another in terms of overall AP? E.g.,
  - Given two systems A vs B: Is A better than B?
  - Given N systems in a competition (say, N=20). What is the ranking of the systems?

Example Results of Real Systems

Class Subset	AP (mean over 5 runs)	stdev of AP (over the 5 runs)
All classes	21.3	0.24
Rare classes	9.6	0.83
Less rare classes	21.7	0.32
Common classes	25.5	0.10

System A

Class Subset	AP (mean over 5 runs)	stdev of AP (over the 5 runs)
All classes	23.2	0.21
Rare classes	13.4	0.80
Less rare classes	23.2	0.32
Common classes	27.1	0.07

System B

Class Subset	AP (mean over 5 runs)	stdev of AP (over the 5 runs)
All classes	24.4	0.06
Rare classes	14.5	0.67
Less rare classes	24.3	0.37
Common classes	28.4	0.12

System C

Results on the 5000 image validation set. Each system is trained 5 times starting from a different random initialization.

Subsets of classes that we're interested in comparing the systems on:

- **All** classes: all 840 classes in the validation set
- **Rare** classes: subset of ~200 that have low instance counts in training and test data
- **Less rare** classes: subset of ~300 that are a bit less rare (higher instance counts)
- **Common** classes: subset of ~300 that are more common (even higher instance counts)

Methods?

- Use bootstrapped CIs
  - What are the elements to bootstrap sample?
    - Could take bootstrap samples of *images*
    - Could take bootstrap samples of *classes* (there are 1000 of them)
  - What statistic to compute?
    - Per-system AP
    - Difference in AP between pairs of systems
    - Rank of each system
  - What type of boostrapping?
    - Bias-corrected (and accelerated), does it matter?
- Do something else?

Dai, Li, He, Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. NIPS 2016.

R-FCN