Today’s agenda

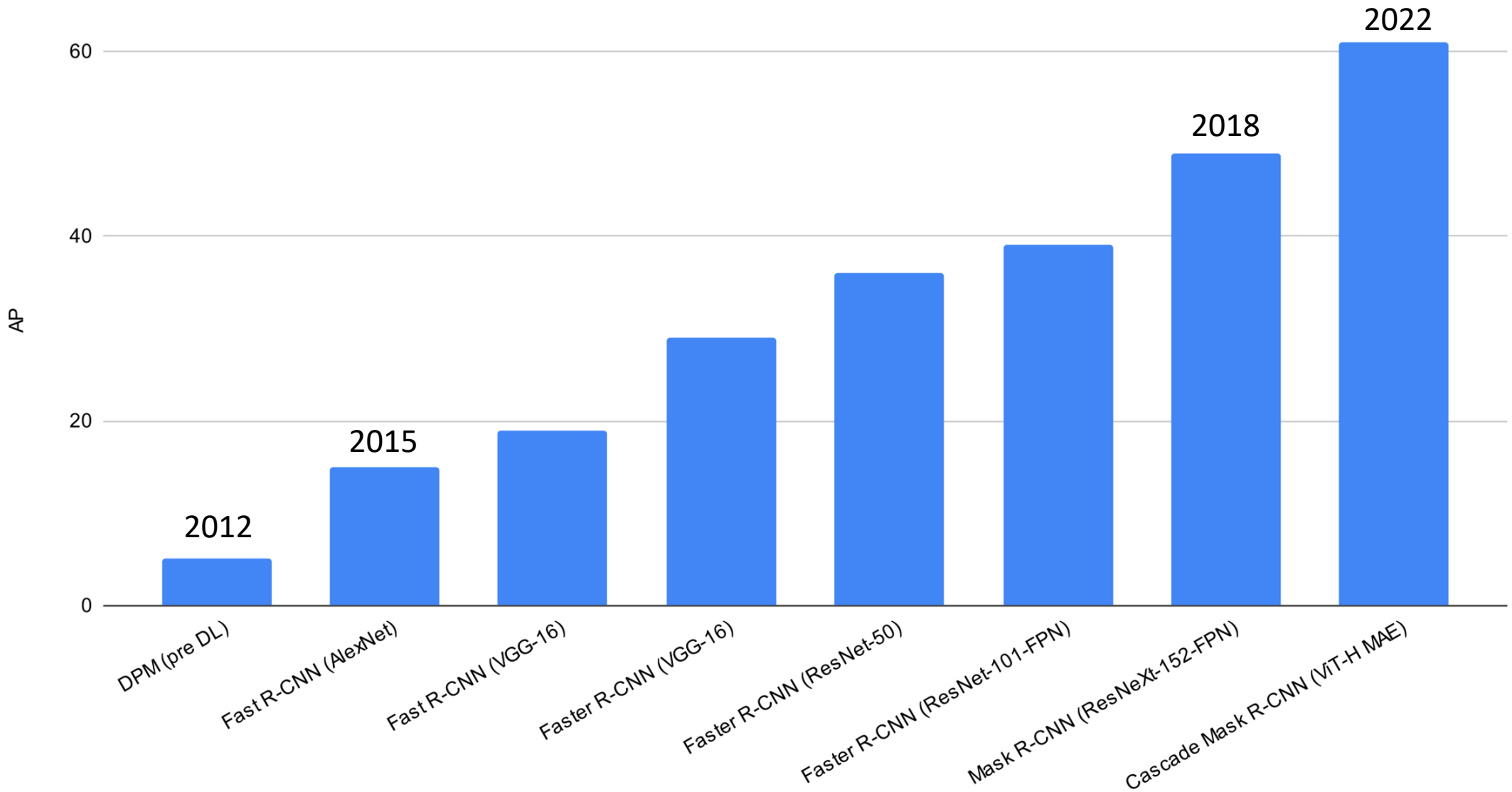
- Q&A
- 10 minute break
- Lecture 3
- Slides on tips for reading (and writing) detection research papers

Lecture 3:
Slow, Fast, Faster & Mask R-CNN

Ross Girshick (FAIR)

AMMI 2022, Computer Vision, Week 2

COCO average precision (%) over time

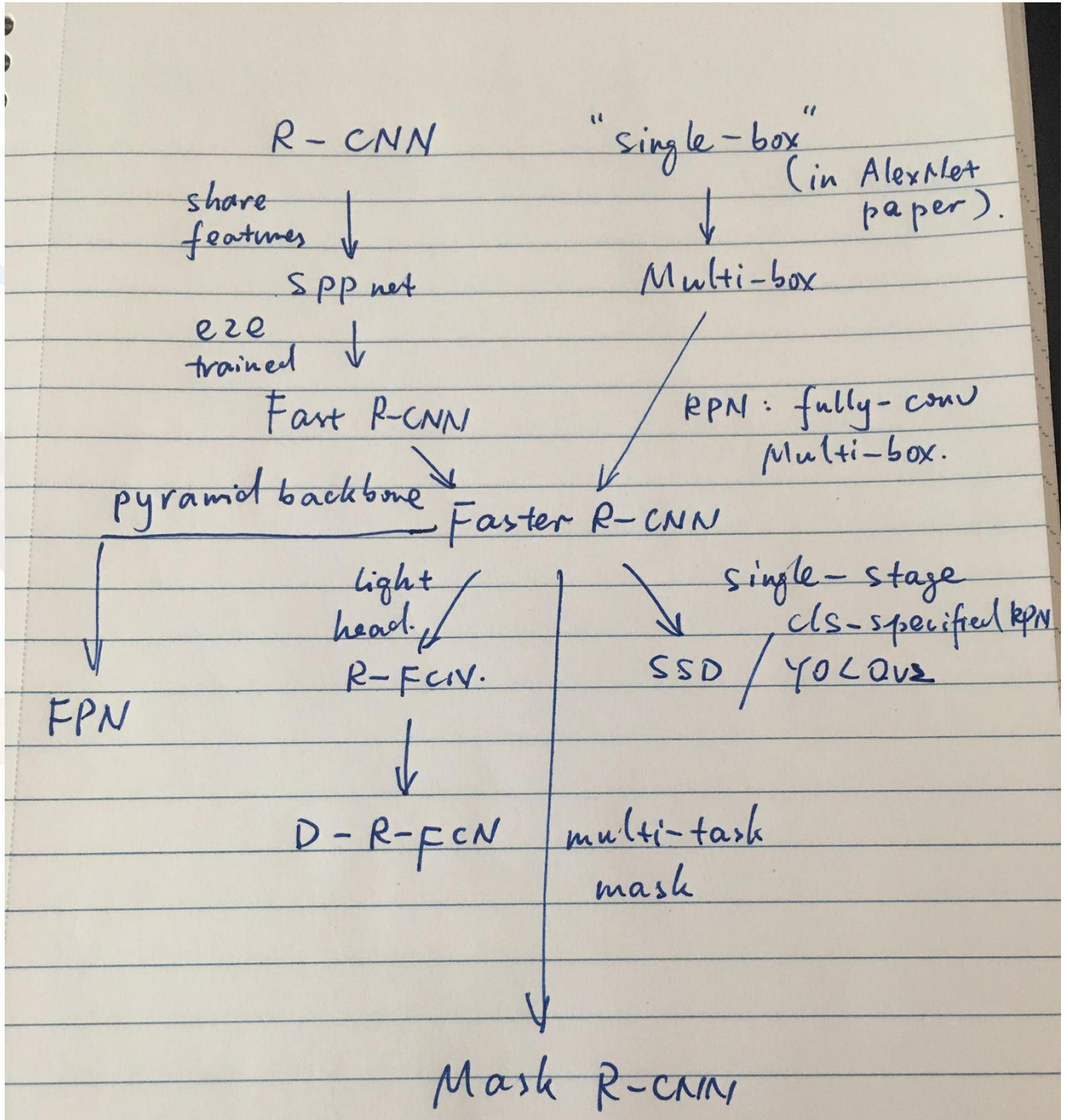


Where do these improvements come from?

- Better training recipes
  - Learning rate, weight decay, # epochs, regularization methods, etc.
- Better data augmentation
  - E.g., large scale jitter
- Better pre-training
  - E.g., Masked autoencoders
- Better “backbone” neural networks
  - AlexNet → VGG → ResNet → ResNeXt → ViT
- Better detector design (“task-specific engineering” from lecture 1)
  - R-CNN → Fast R-CNN → Faster R-CNN → Mask R-CNN → …

Building
a better
hammer

Modern object detection is a complex web of related methods



R-CNN Today

- Conceptual basis for modern proposal-based detectors
- Extensions of R-CNN are state-of-the-art on COCO, LVIS, Open Images, etc.

Learning objectives

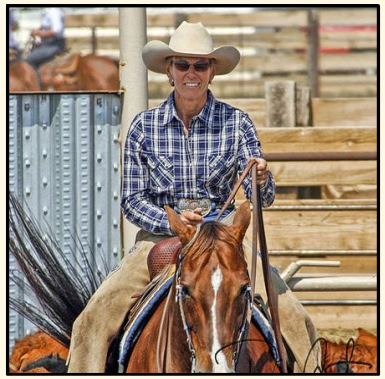
- We've learned fundamentals so far
  - Task formulation, how it's evaluated, ML problem formulation, computational formulation
- Now let's learn about specific models (i.e., computational solutions)
  - Specifically the R-CNN family of detectors
- We'll integrate knowledge from the prior lectures
  - E.g., bounding box regression, cascades, box classification, region proposals, NMS, etc.

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

I:



Per-region computation

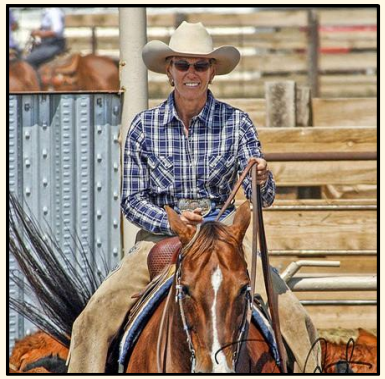
Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN (Region-based Convolutional Neural Net)

Per-image computation

Per-region computation

I:

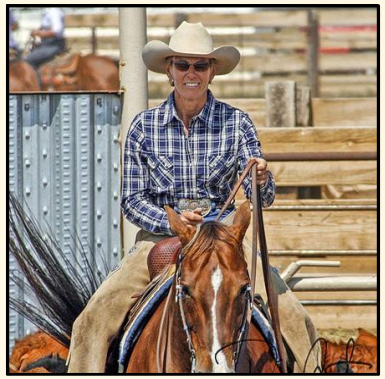


Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN (Region-based Convolutional Neural Net)

Per-image computation

I:



Per-region computation

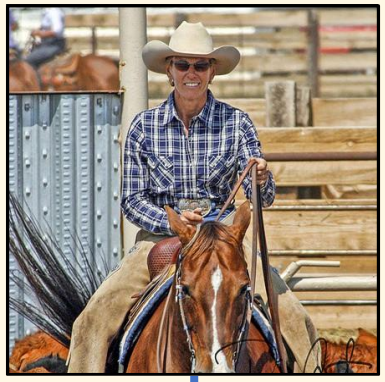
Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN (Region-based Convolutional Neural Net)

Per-image computation

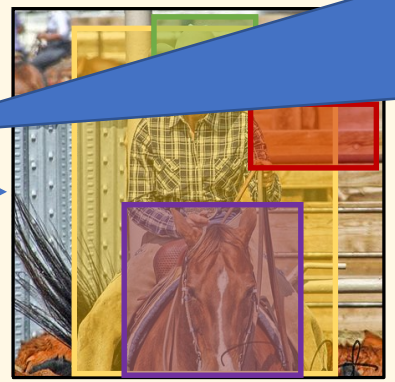
Per-region computation

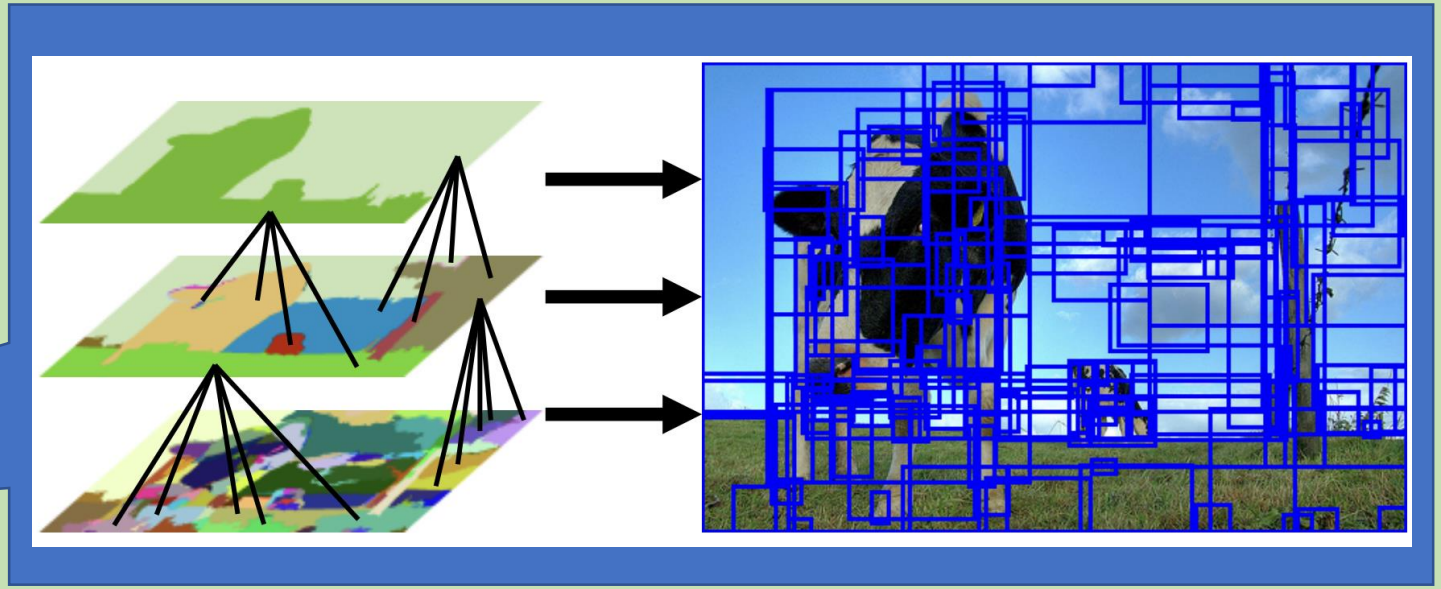
I:



Selective search,
Edge Boxes,
MCG, …

1





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

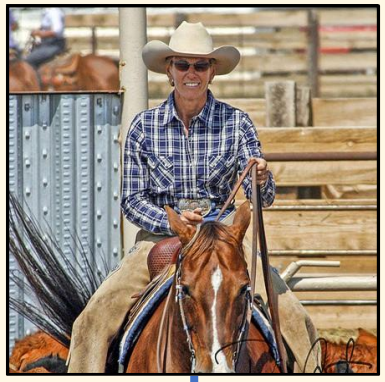
Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN

Per-image computation

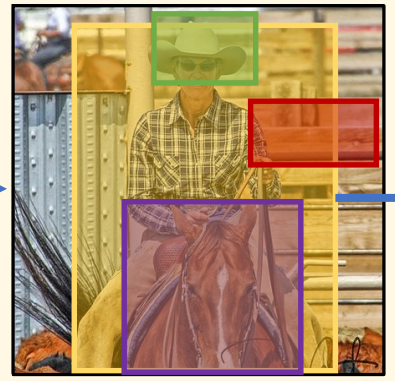
Per-region computation for each $r_i \in r(I)$

I:



Selective search, Edge Boxes, MCG, …

1



Crop & warp

2

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

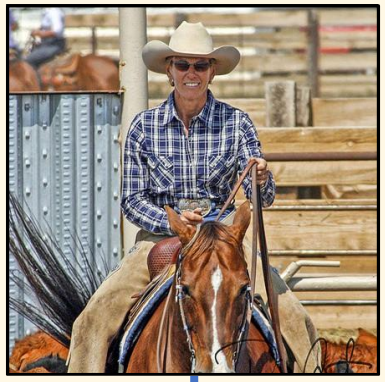
Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN

Per-image computation

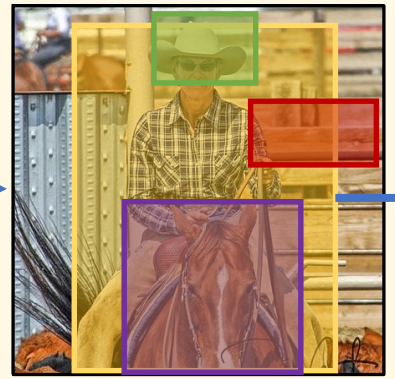
Per-region computation for each $r_i \in r(I)$

I:



Selective search, Edge Boxes, MCG, …

1



Crop & warp

2

ConvNet(r_i)

3

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

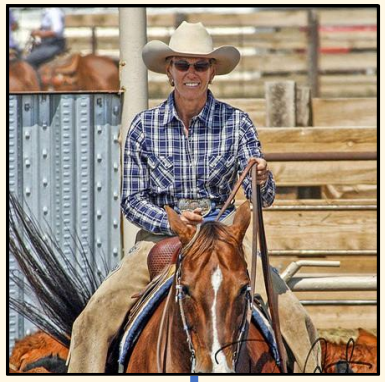
Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN

Per-image computation

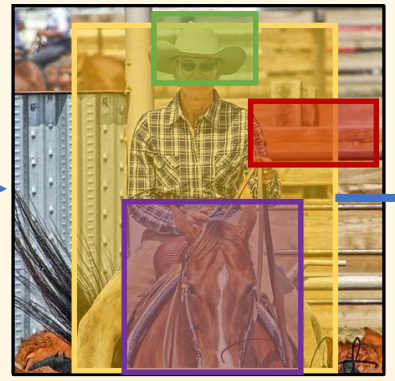
Per-region computation for each $r_i \in r(I)$

I:



Selective search, Edge Boxes, MCG, …

1



Crop & warp

2

ConvNet(r_i)

3

Linear classifier

4

4. Object classification

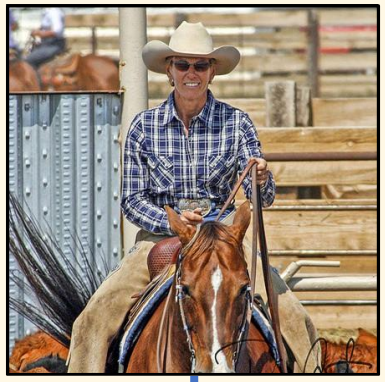
Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.

R-CNN

Per-image computation

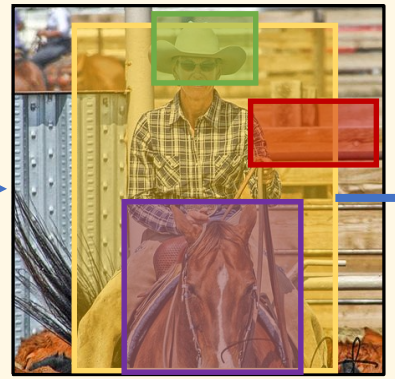
Per-region computation for each $r_i \in r(I)$

I:



Selective search, Edge Boxes, MCG, …

1



Crop & warp

2

ConvNet(r_i)

3

Linear classifier

4

Box regressor

5

5. Refine proposal localization with bounding-box regression

The 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, …)

Detectron2

PyTorch

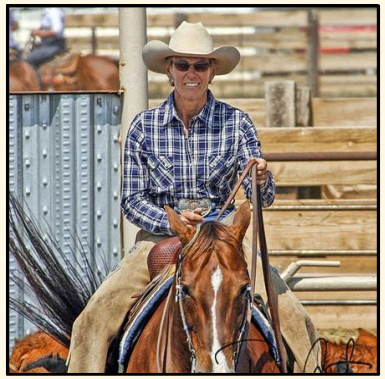
https://github.com/facebookresearch/detectron2

Generalized R-CNN Framework

Per-image computation

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

I:



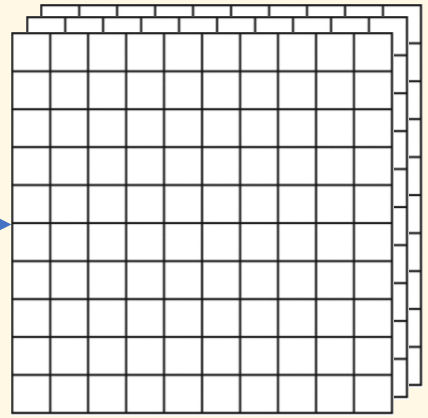
Input image

per-image operations | per-region operations

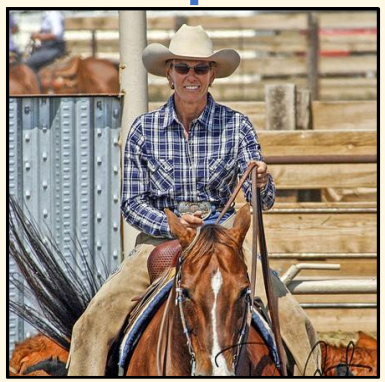
Generalized R-CNN Framework

Per-image computation

$f_I = f(I)$



I:



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

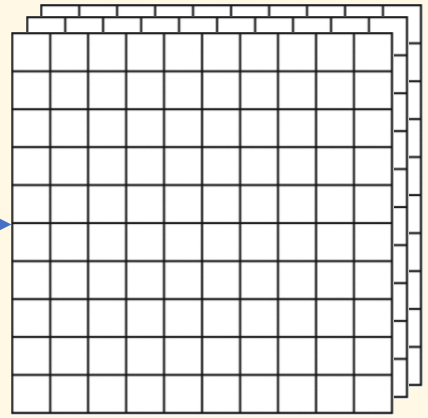
Transformation of the input image into a featurized representation

Generalized R-CNN Framework

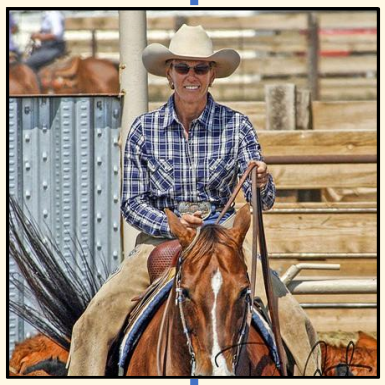
Per-image computation

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

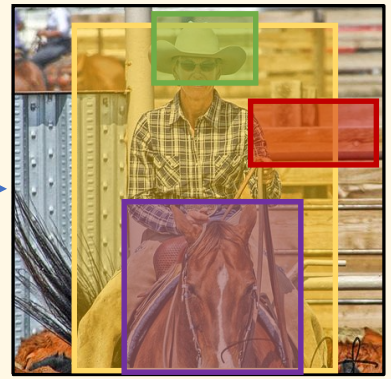
$f_I = f(I)$



I:



$r(I)$



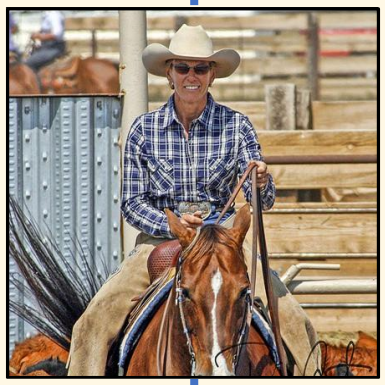
Region of Interest proposals computed for the image

Generalized R-CNN Framework

Per-image computation

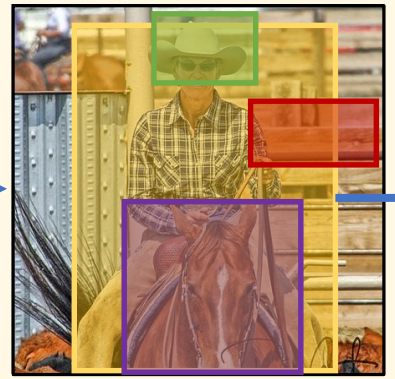
Per-region computation for each $r_i \in r(I)$

$f_I = f(I)$



I:

$r(I)$



$g_i = g(f_I, r_i)$

Compute a featurized representation of each proposal

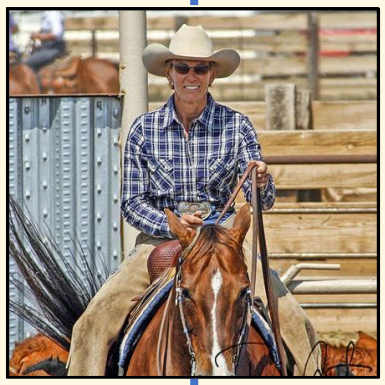
Generalized R-CNN Framework

Per-image computation

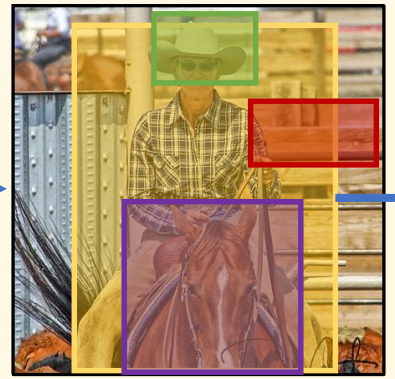
Per-region computation for each $r_i \in r(I)$

$f_I = f(I)$

I:



$r(I)$



$g_i = g(f_I, r_i)$

$h(g_i)$

Apply additional processing to each proposal feature

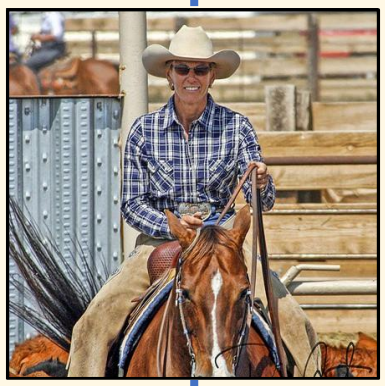
Generalized R-CNN Framework

Per-image computation

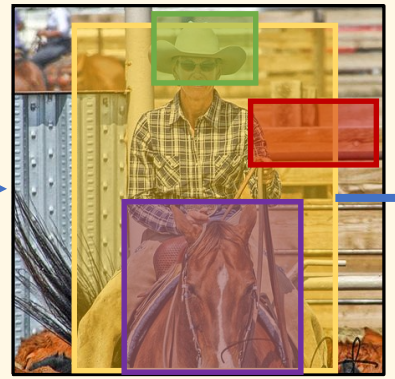
Per-region computation for each $r_i \in r(I)$

$f_I = f(I)$

I:



$r(I)$



$g_i = g(f_I, r_i)$

$h(g_i)$

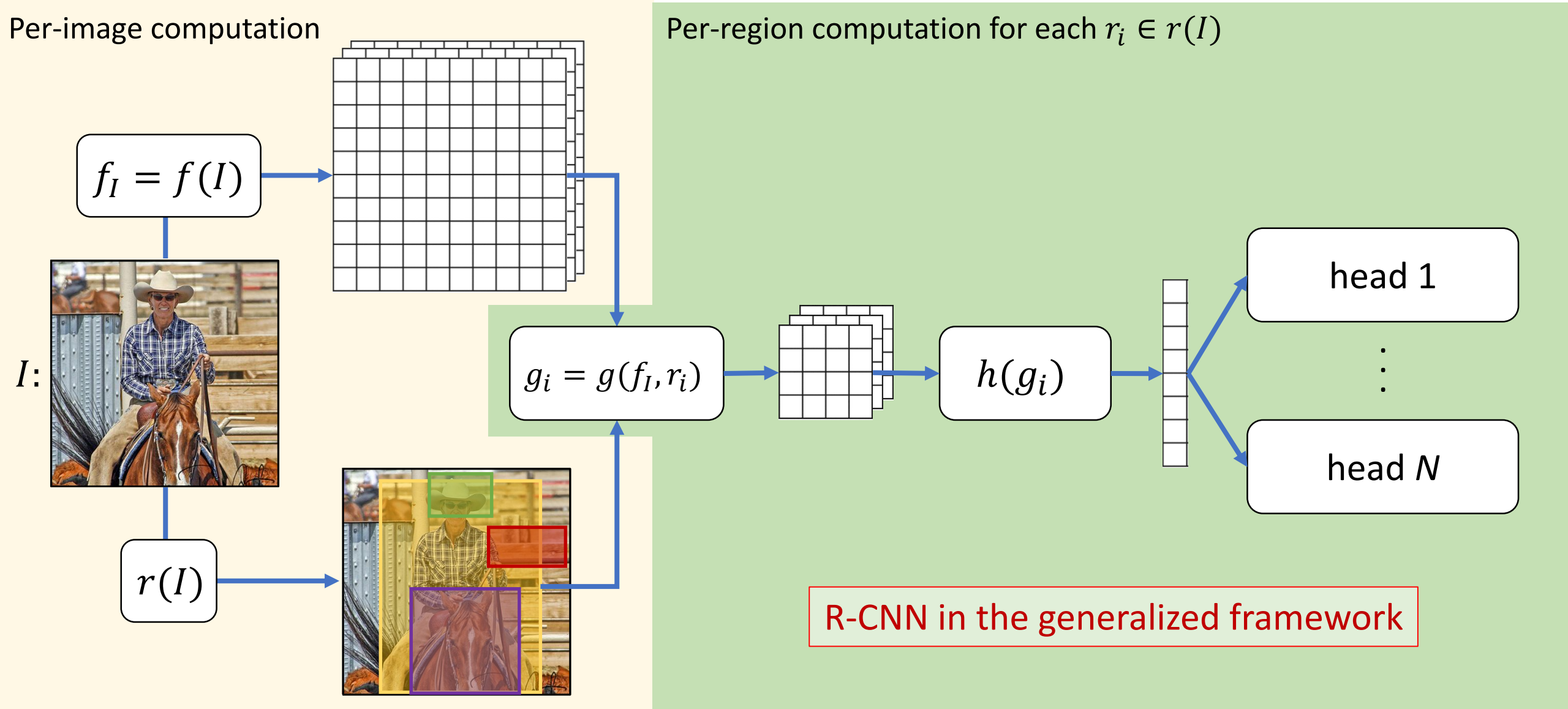
head 1

head *N*

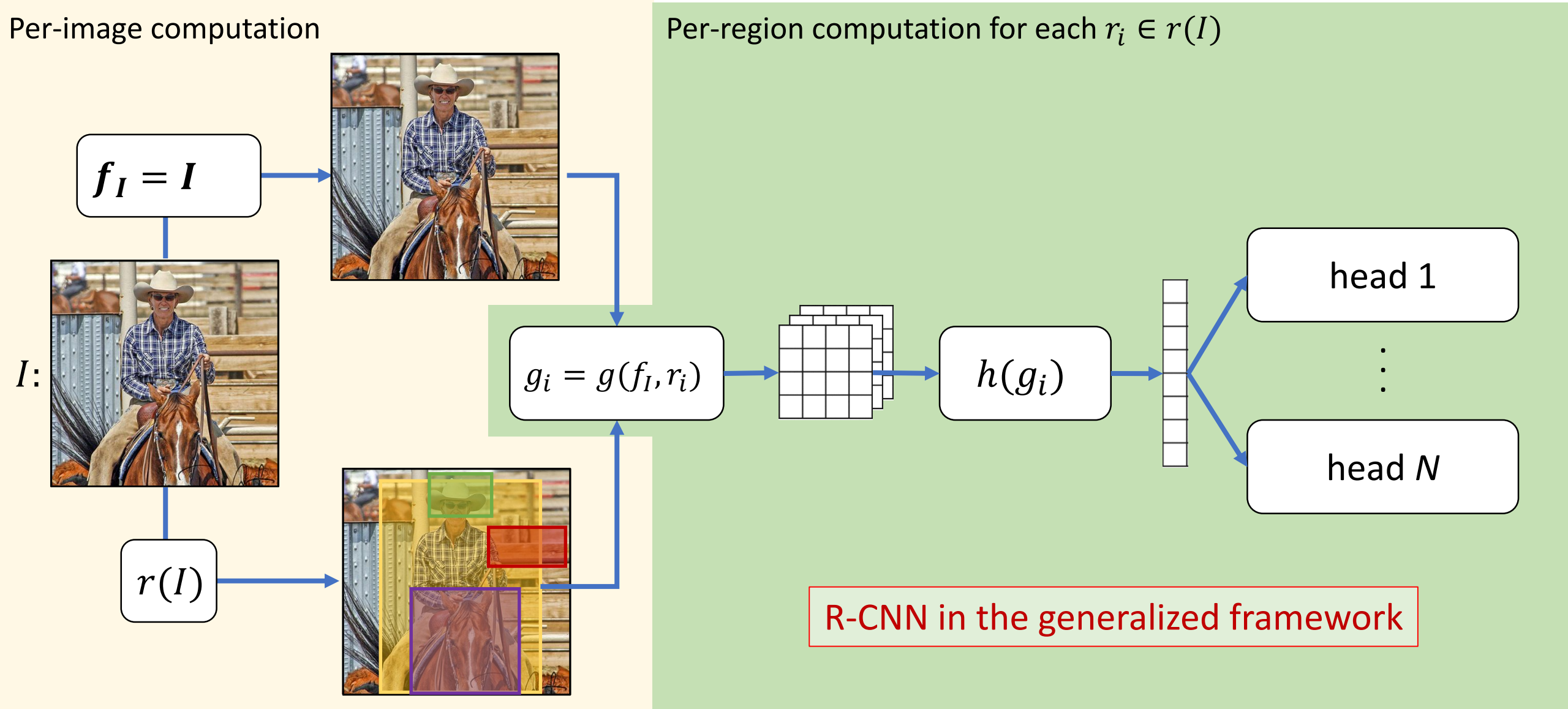
Apply multiple ‘heads’
to make task-specific predictions

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

R-CNN in the Generalized Framework



R-CNN in the Generalized Framework

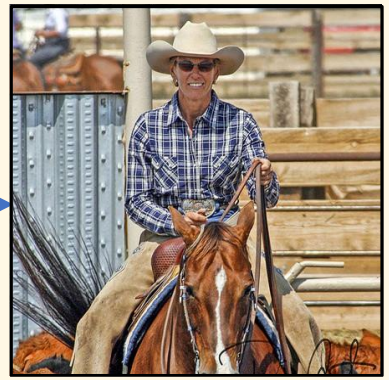


R-CNN in the Generalized Framework

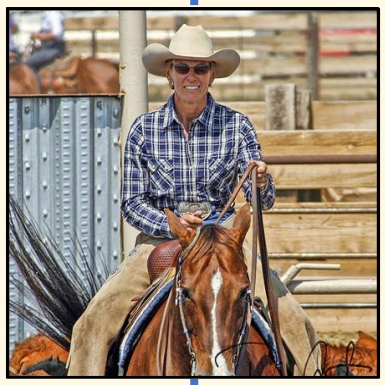
Per-image computation

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

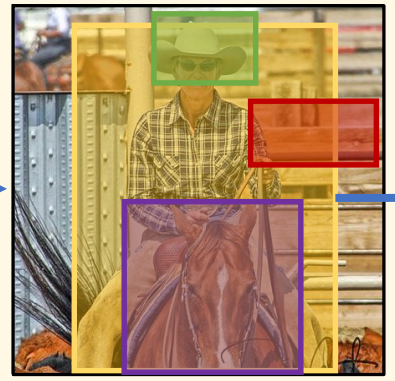
$f_I = I$



I:



$\boldsymbol{r(I)} =$
Selective search,
Edge Boxes,
MCG, …



Crop & warp

$h(g_i)$

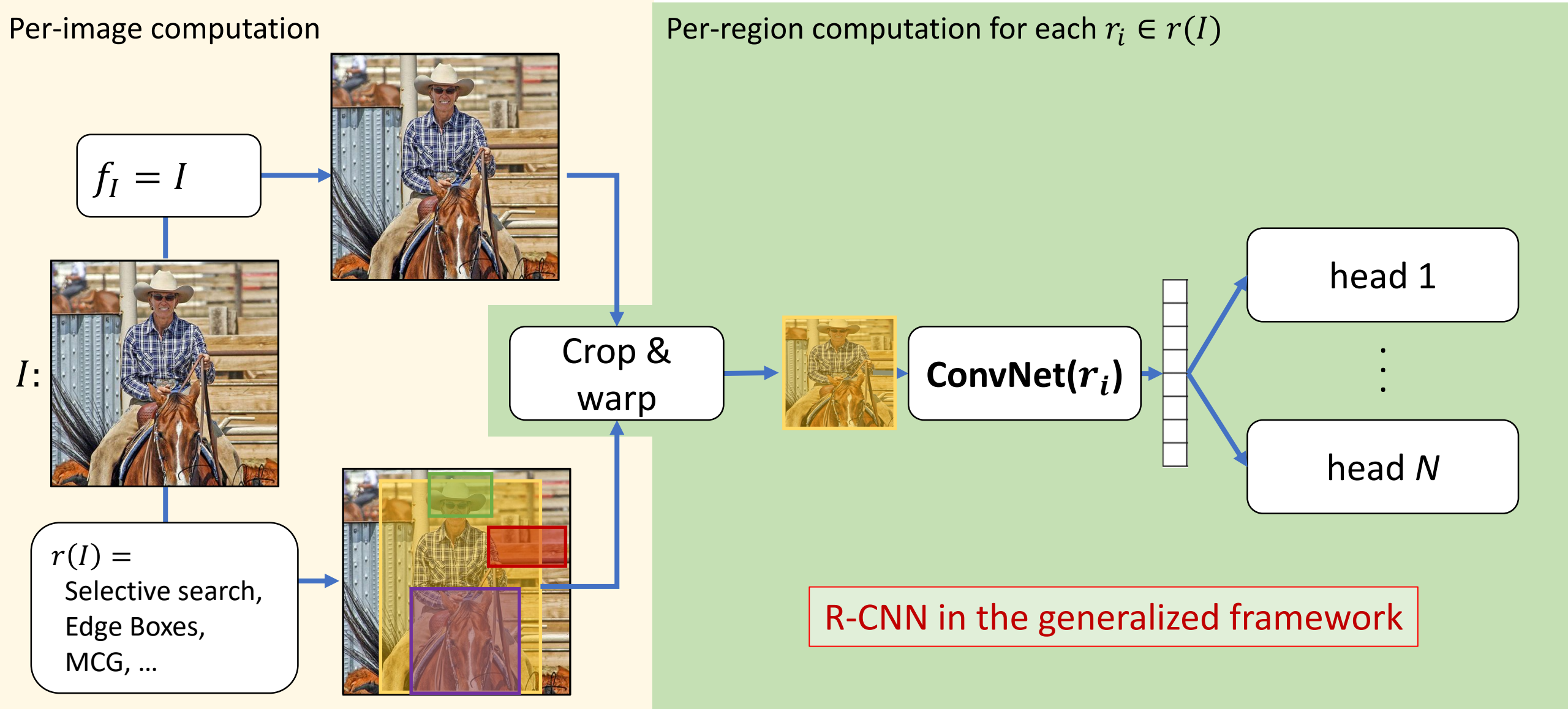
head 1

⋮

head *N*

R-CNN in the generalized framework

R-CNN in the Generalized Framework

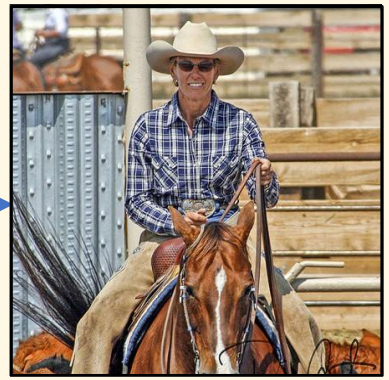


R-CNN in the Generalized Framework

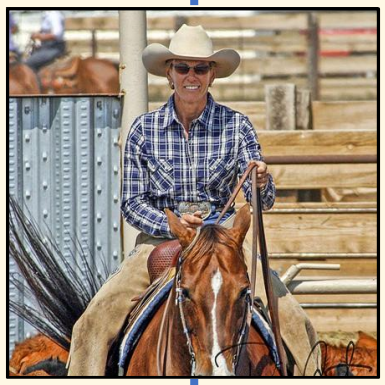
Per-image computation

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

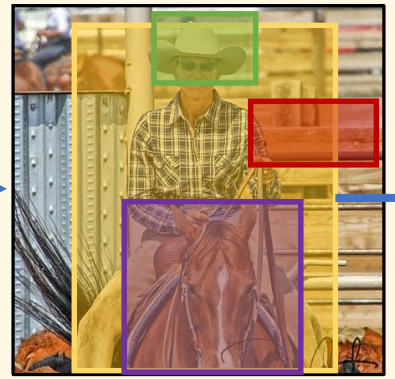
$f_I = I$



I:



$r(I) =$
Selective search,
Edge Boxes,
MCG, …



Crop & warp

ConvNet(r_i)

Linear classifier

Box regressor

R-CNN in the generalized framework

The Problem with R-CNN

R-CNN

Building a better hammer

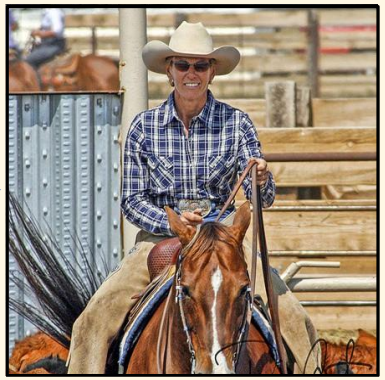
?

"Slow" R-CNN

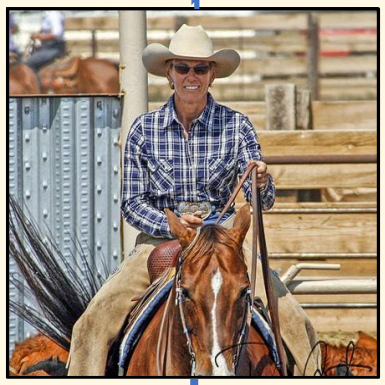
Per-image computation

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

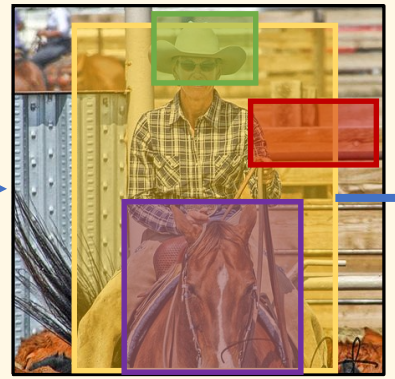
$f_I = I$



I:



Selective search, Edge Boxes, MCG, …



Crop & warp

ConvNet(r_i)

Linear classifier

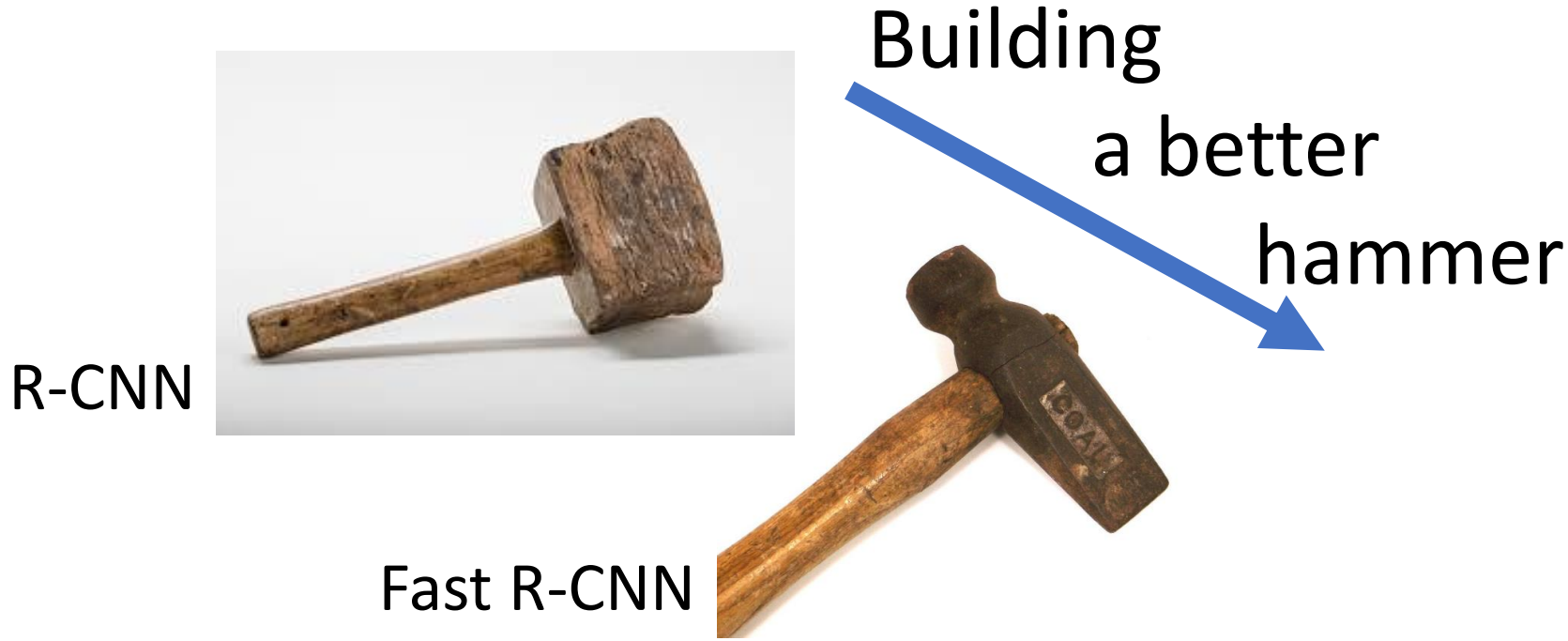
Box regressor

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.



Girshick. Fast R-CNN. ICCV 2015.

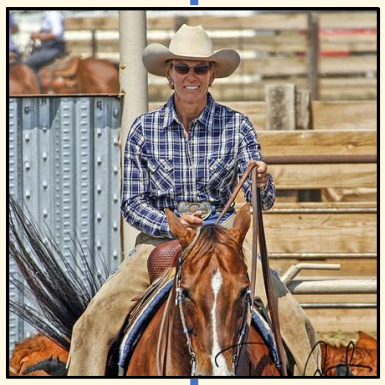
Generalized R-CNN → Fast R-CNN

Per-image computation

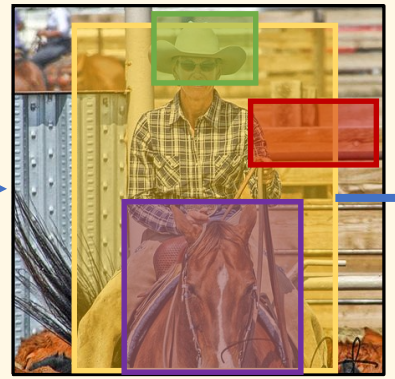
Per-region computation for each $r_i \in r(I)$

$f_I = f(I)$

I:



Selective search,
Edge Boxes,
MCG, …



$g_i = g(f_I, r_i)$

$h(g_i)$

head 1

head *N*

Light-weight *per-region* computation
End-to-end trainable version of SPP-net [He et al. 2014]

Generalized 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, …

$g_i = g(f_I, r_i)$

$h(g_i)$

head 1

head *N*

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

Girshick. Fast R-CNN. ICCV 2015.

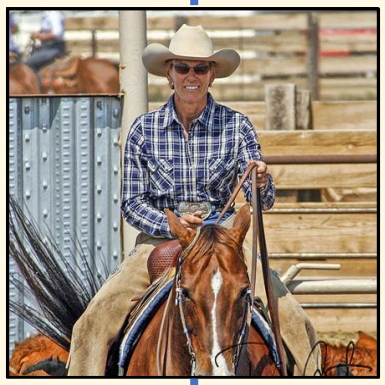
Generalized R-CNN → Fast R-CNN

Per-image computation

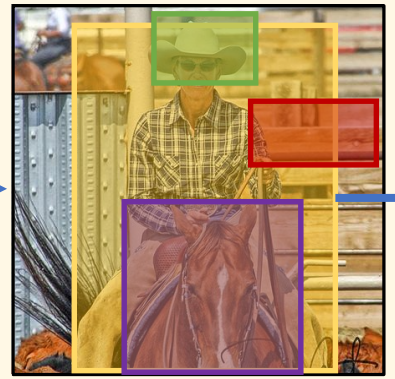
Per-region computation for each $r_i \in r(I)$

1 FCN(I)

I:



Selective search, Edge Boxes, MCG, …



2 RoIPool

$h(g_i)$

head 1

head *N*

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

Girshick. Fast R-CNN. ICCV 2015.

Generalized R-CNN → Fast R-CNN

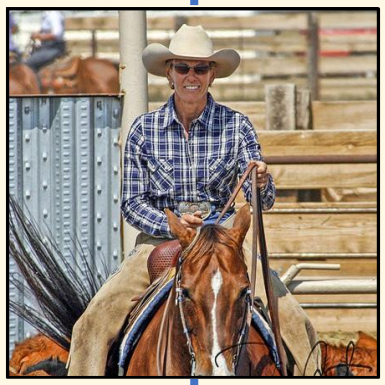
Per-image computation

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

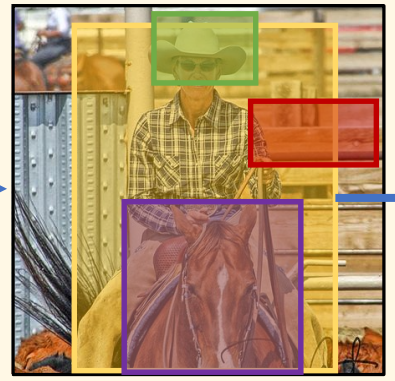
1

FCN(I)

I:



Selective search,
Edge Boxes,
MCG, …



2

RoIPool

3

MLP

head 1

head *N*

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

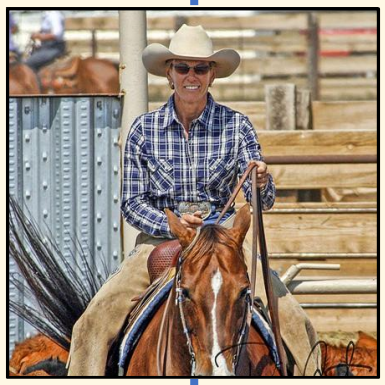
Per-image computation

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

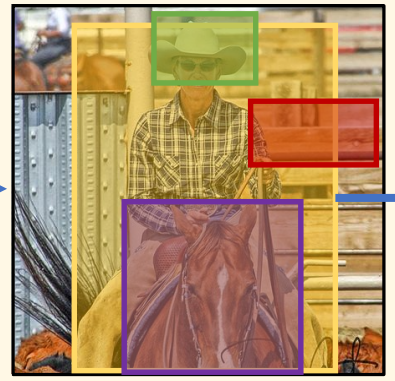
1

FCN(I)

I:



Selective search,
Edge Boxes,
MCG, …



2

RoIPool

3

MLP

Softmax clf.

4

Box regressor

4. Classification and box regression heads are applied

Fast R-CNN

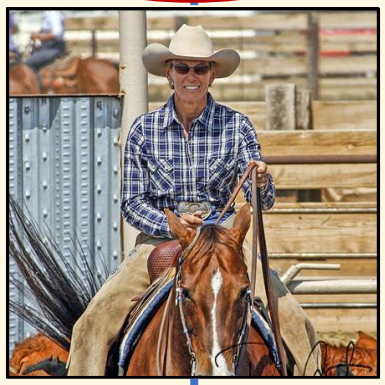
Per-image computation

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

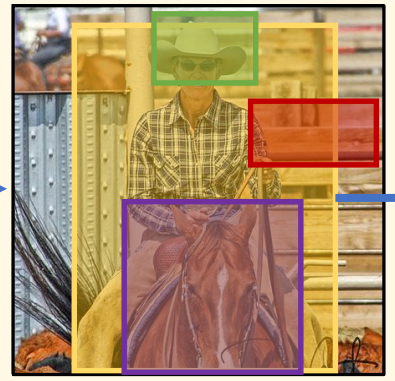
1

FCN(I)

I:



Selective search,
Edge Boxes,
MCG, …



2

RoIPool

MLP

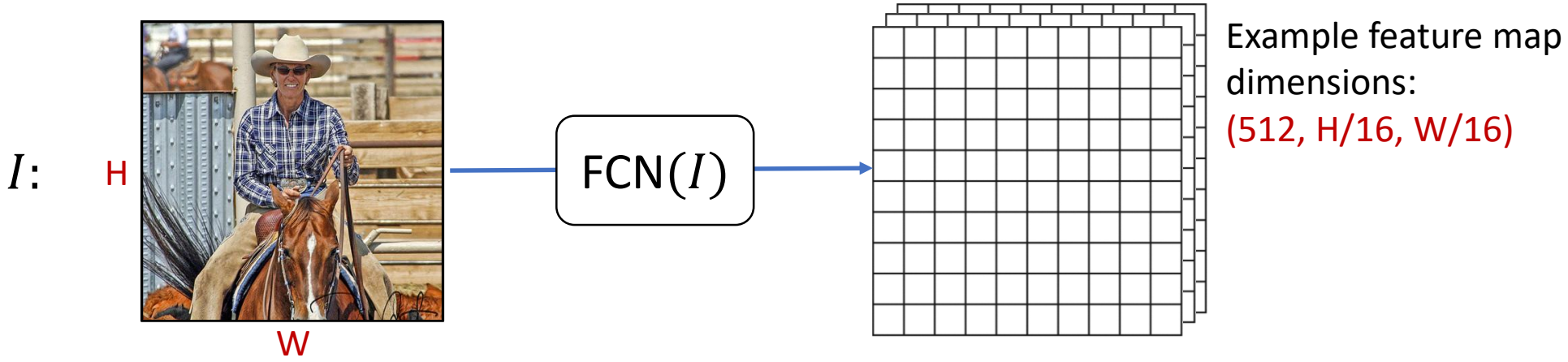
Softmax clf.

Box regressor

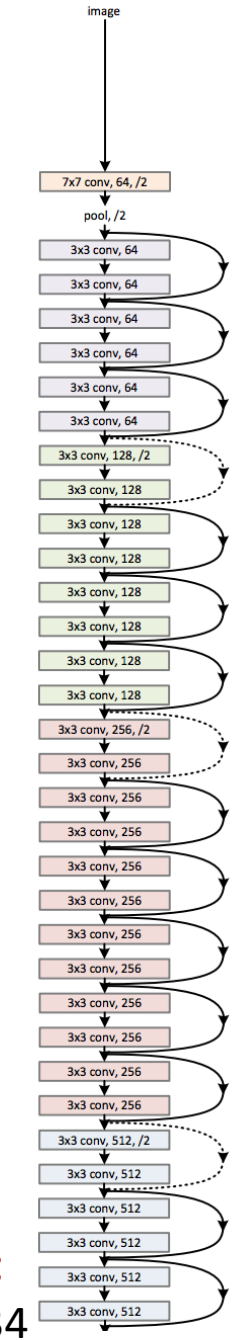
Whole-image, Fully Conv. Network (FCN)

Use any standard ConvNet (or Transformer) as the "backbone architecture"

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



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



Example:
ResNet-34
w/o "head"

The Engine of Recognition

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

- AlexNet
- VGG
- GoogleNet / Inception
- ResNet
- ResNeXt
- NAS-derived networks
- ViT
- …

Foundational Concept: Pre-training

- Neural networks are typically initialized randomly

- Pre-training
  - An alternative to random initialization
  - First, train a neural network for some task
  - Then, use this “pre-trained” network as an initialization for your task

- This is often called “transfer learning”
  - Information from the pre-training task is *transferred* to your downstream task

Pre-training Example

- Pre-training architecture: ResNet-50
- Pre-training task: ImageNet classification

- Downstream architecture: Fast R-CNN
- Downstream task: object detection

- Network “surgery” required:
  - Remove ResNet-50’s global pooling and FC layers (the classification “head”)
  - Use this “headless” ResNet-50 as the FCN component in Fast R-CNN

Fast R-CNN

Pre-training: Initialize this FCN with the headless ResNet-50

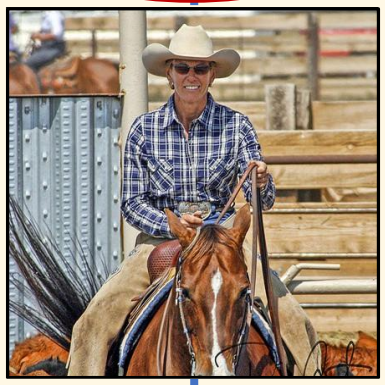
Per-image computation

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

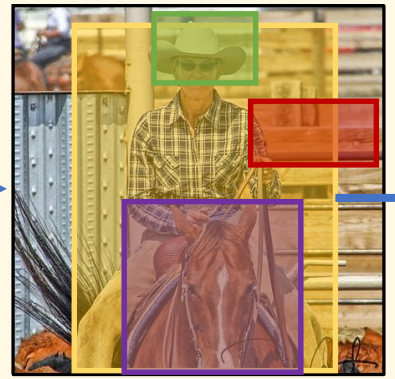
1

FCN(I)

I:



Selective search, Edge Boxes, MCG, …



2

RoIPool

MLP

Softmax clf.

Box regressor

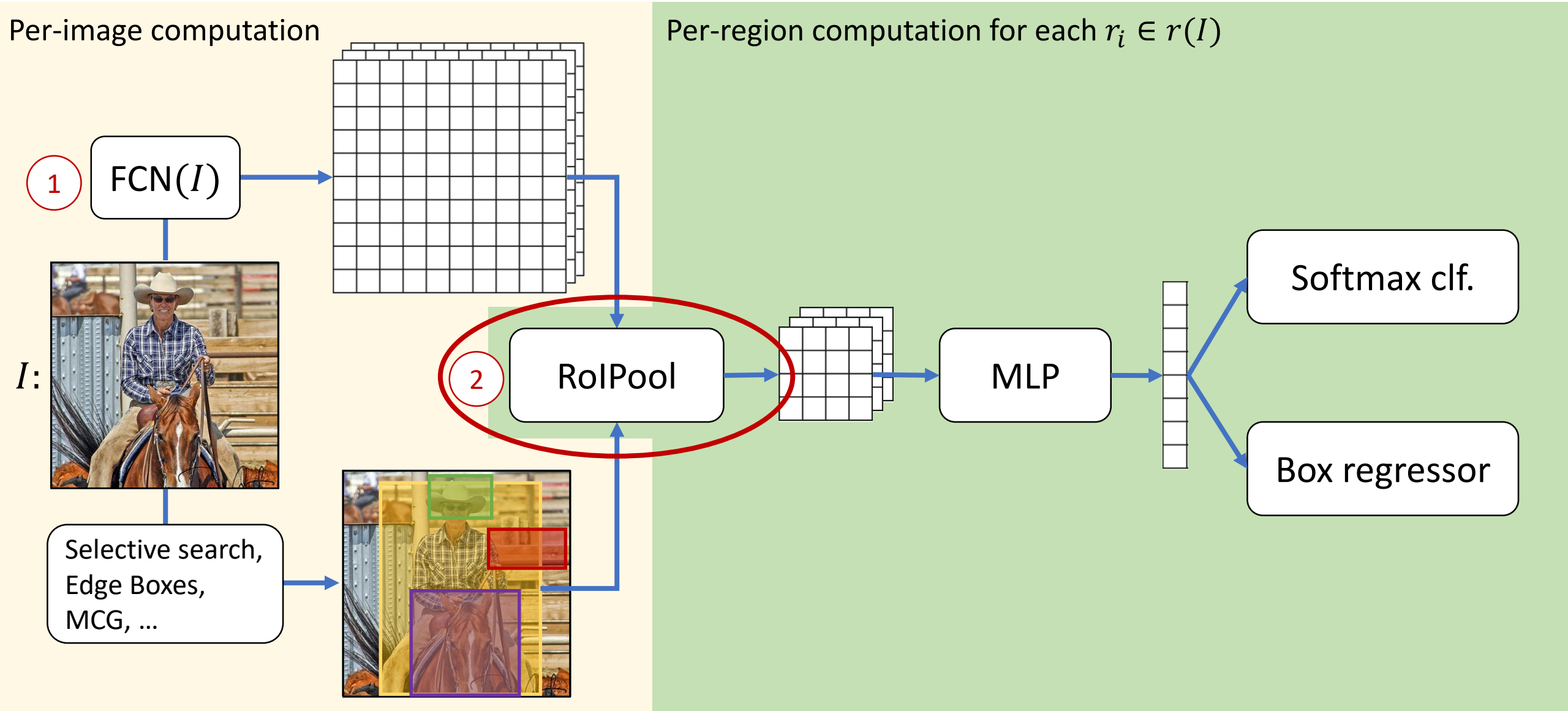
Foundational Concept: Fine-tuning a Pre-trained Network

- "Training from scratch": Training a network from random initialization

- "Fine-tuning": Training a network from a pre-trained initialization
  - In practice, the network may contain *some* randomly initialized parameters
    - E.g., in Fast R-CNN, the MLP, softmax classifier, and box regressor

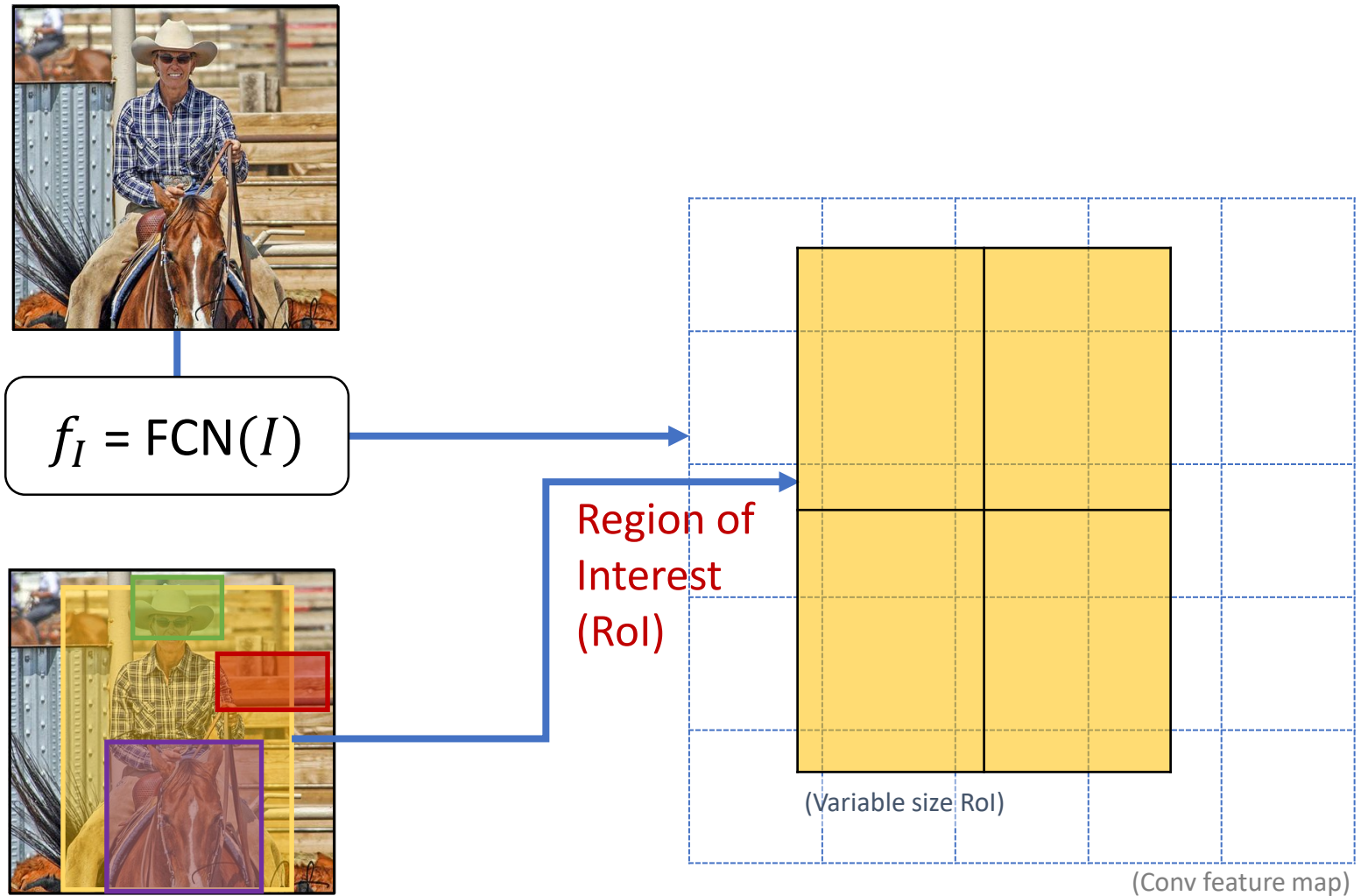
- Fine-tuning sometimes involves "freezing" (i.e., not training) some portions of the pre-trained network
  - Always an empirical question (i.e., "does it improve AP?")
  - Freezing the first few layers of a pre-trained backbone is common

Fast R-CNN


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

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

RoIPool Operation (on each Proposal)

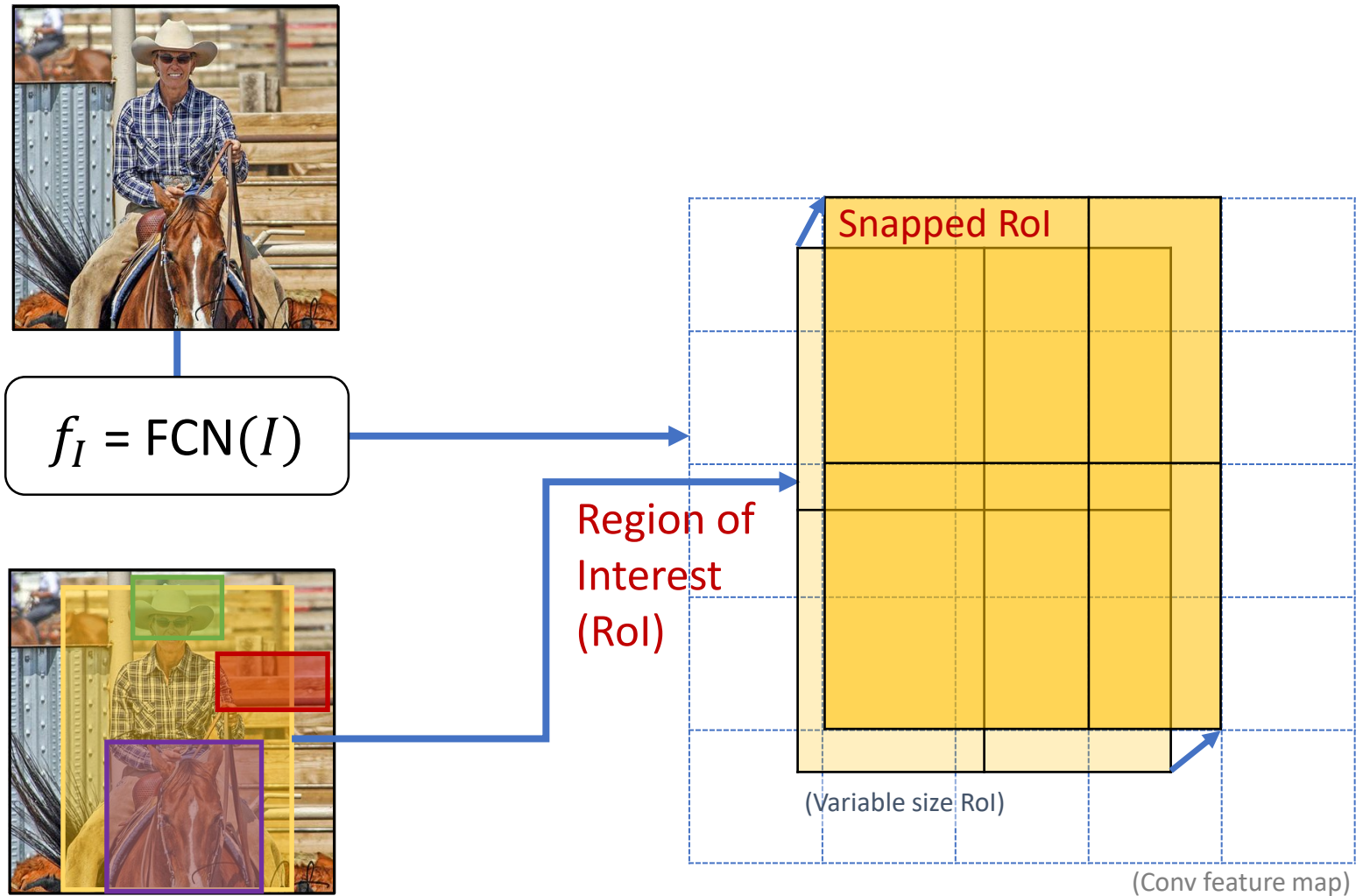


Key innovation in SPP-net

[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)



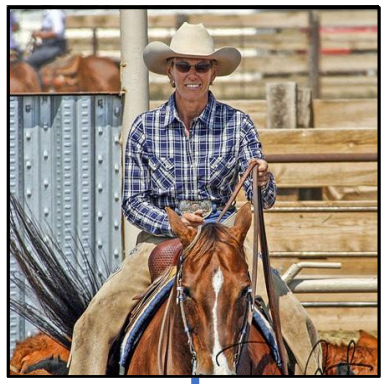
Key innovation in SPP-net

[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., 7x7)



f_I = FCN(I)

Proposal Region of Interest (RoI)

Snapped RoI

(Variable size RoI)

(Conv feature map)

RoIPool transform

(Fixed dimensional representation)

MLP

Feature value is max over input cells

The Problem with Fast R-CNN

R-CNN

Building a better hammer

Fast R-CNN

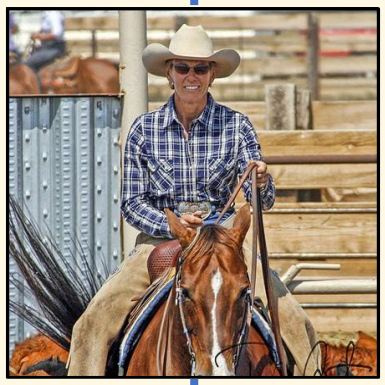
?

The Problem with Fast R-CNN

Per-image computation

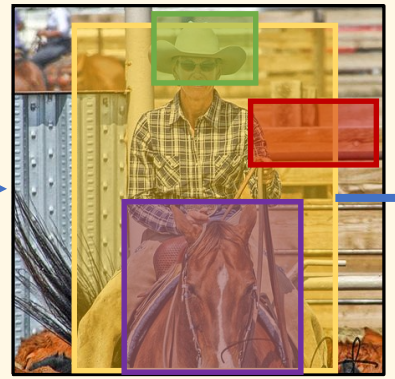
Per-region computation for each $r_i \in r(I)$

FCN(I)



I:

Selective search,
Edge Boxes,
MCG, …



RoIPool

MLP

Softmax clf.

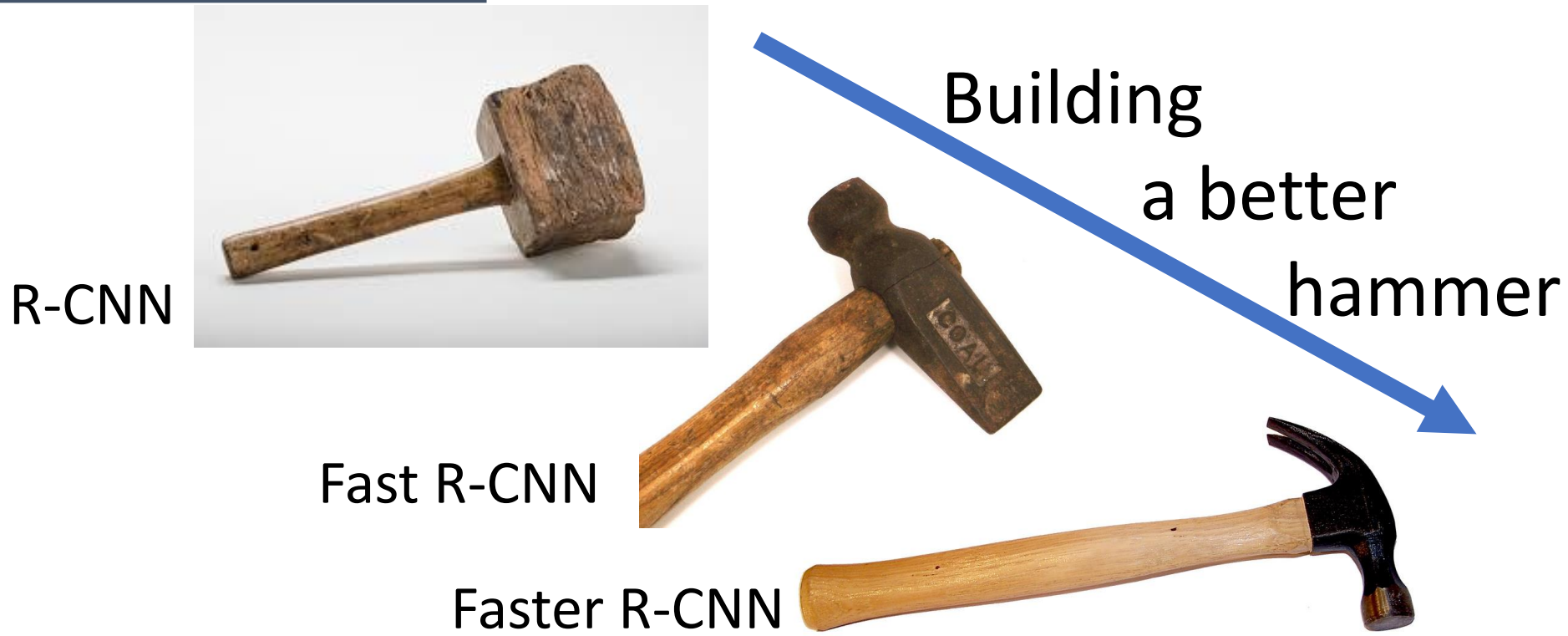
Box regressor

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.



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

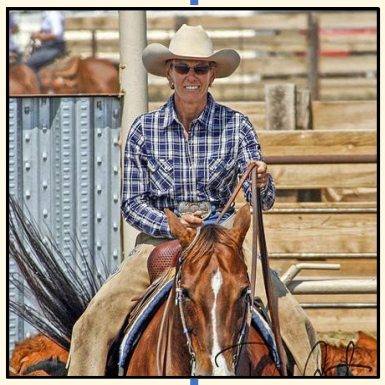
Faster R-CNN

Per-image computation

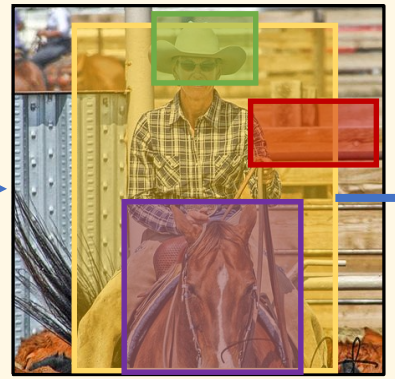
Per-region computation for each $r_i \in r(I)$

f_I = FCN(I)

I:



RPN(f_I)



RoIPool

MLP

Softmax clf.

Box regressor

Learned proposals
Shares computation with whole-image network

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

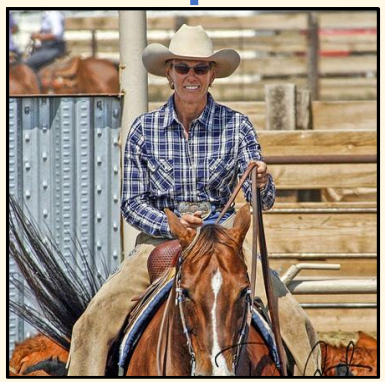
Faster R-CNN

Per-image computation

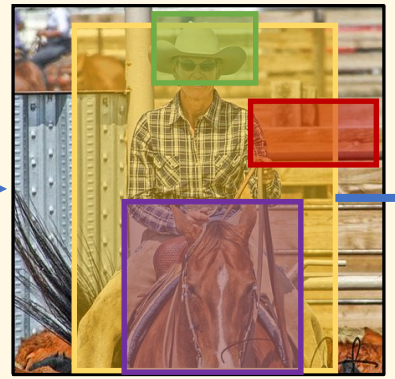
Per-region computation for each $r_i \in r(I)$

f_I = FCN(I)

I:



RPN(f_I)



RoIPool

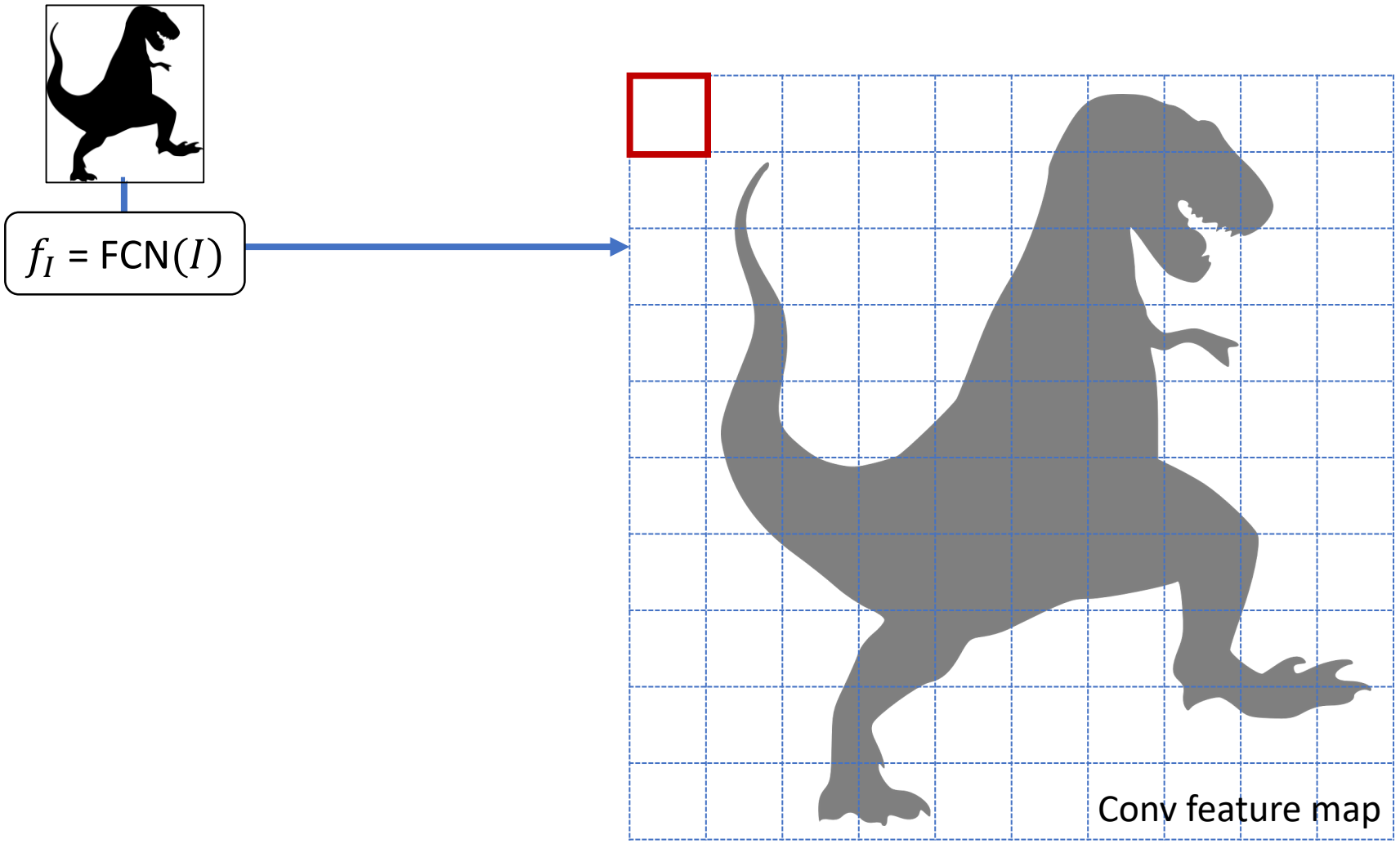
MLP

Softmax clf.

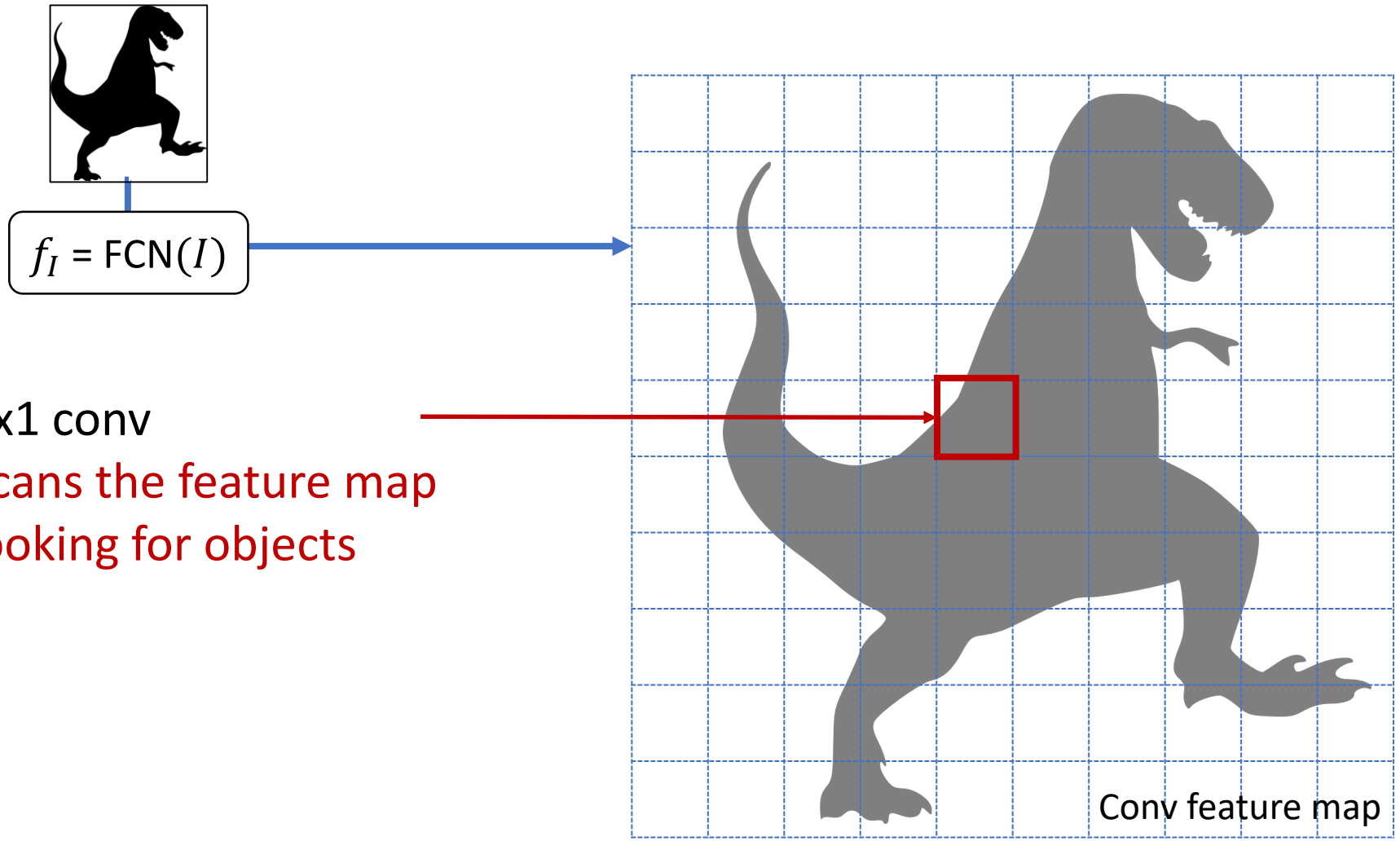
Box regressor

Learned proposals
Shares computation with whole-image network

RPN: Region Proposal Network



RPN: Region Proposal Network

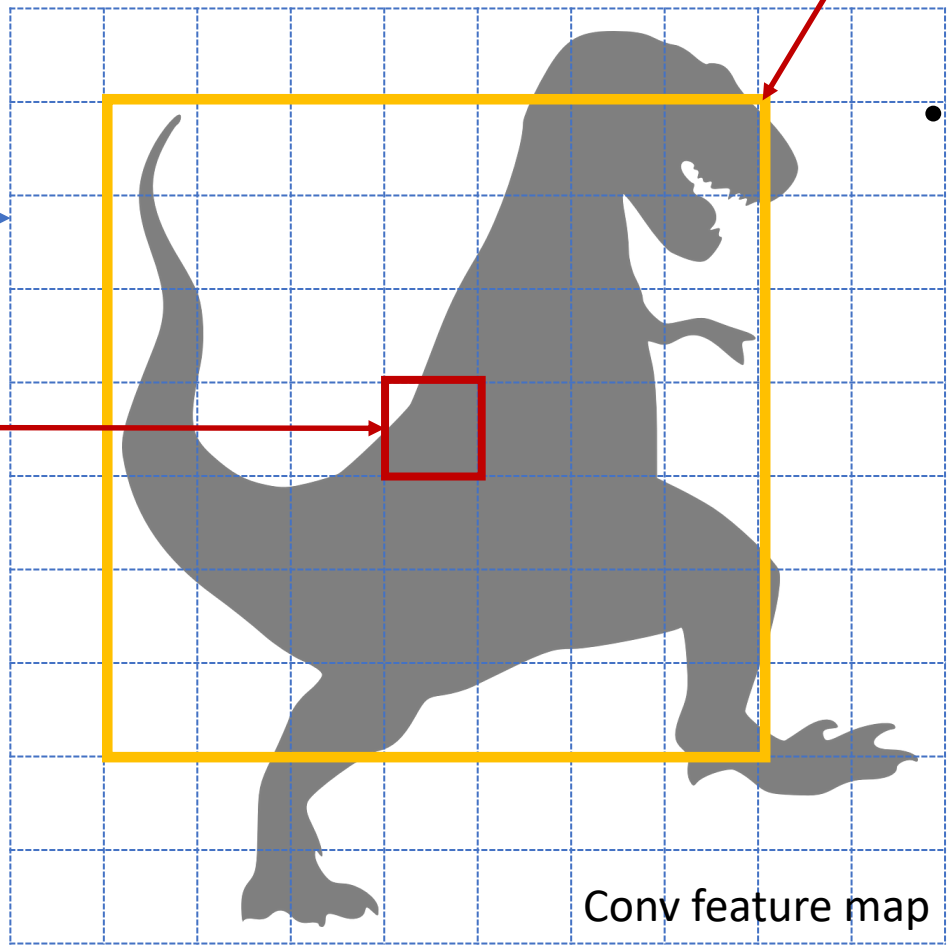


RPN: Anchor Box

f_I = FCN(I)

1x1 conv

Scans the feature map looking for objects

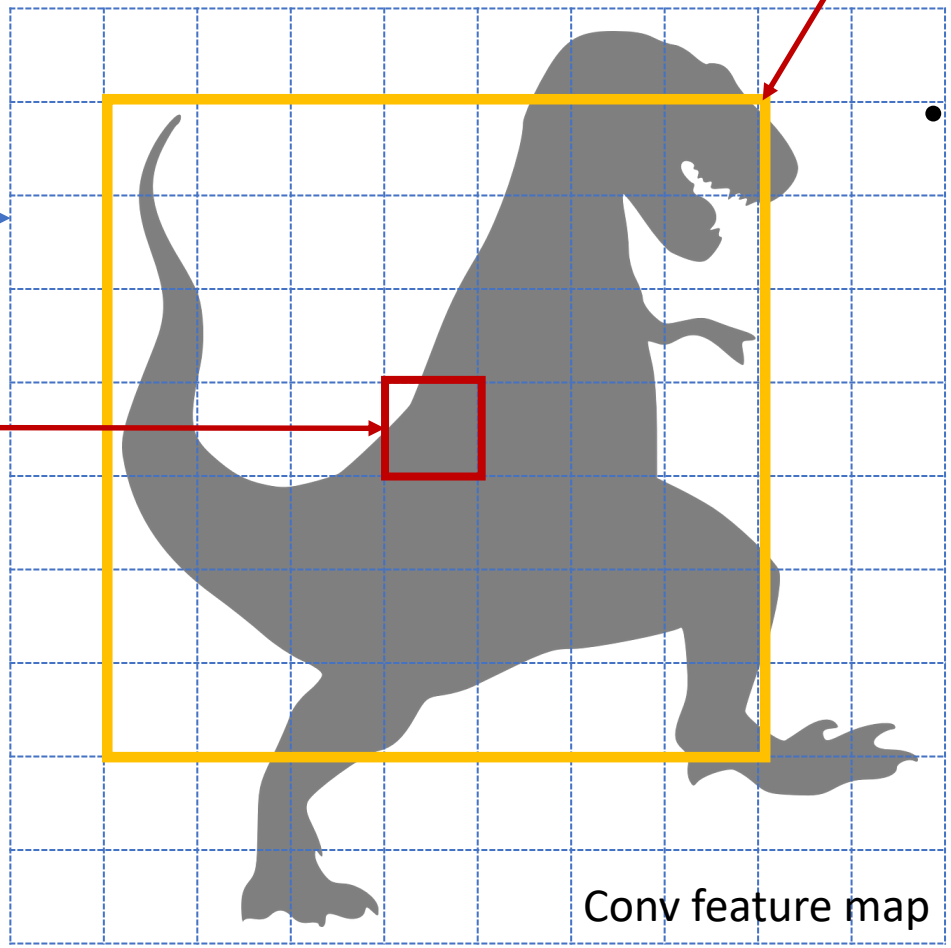


Anchor box

- Predictions are w.r.t. this anchor box, *not* the 1x1 conv window
- This decoupling will be important later

RPN: Anchor Box

f_I = FCN(I)



Anchor box

- Predictions are w.r.t. this anchor box, *not* the 1x1 conv window
- This decoupling will be important later

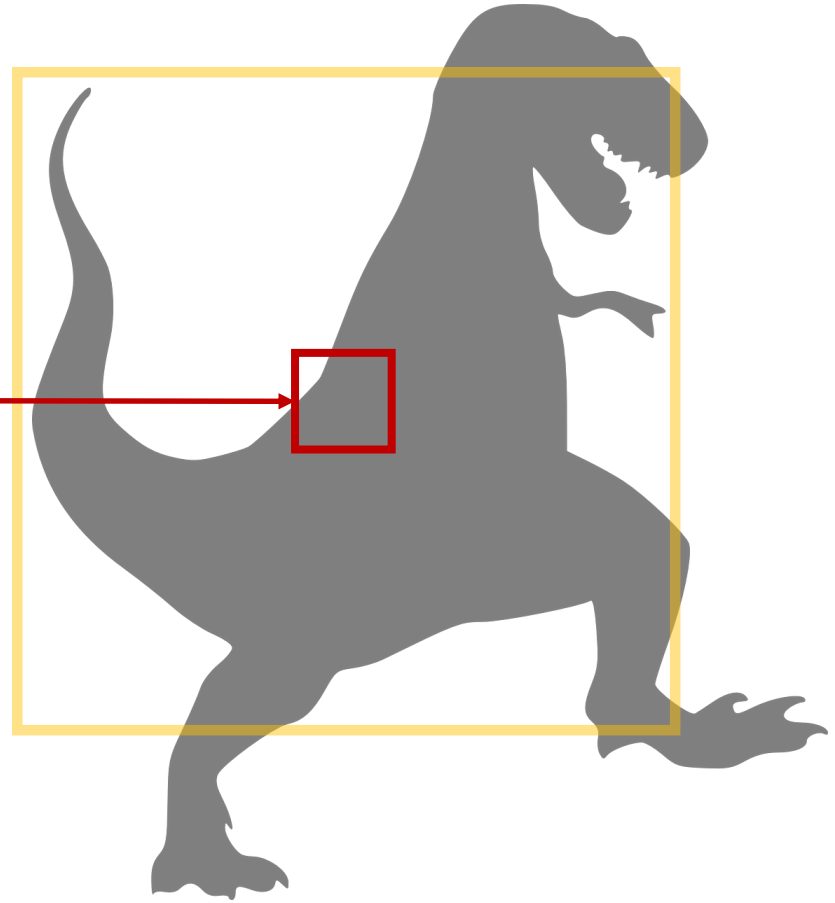
1x1 conv
Five output channels:

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

RPN: Prediction (on object case)

Objectness score

P(object) = 0.94



1x1 conv

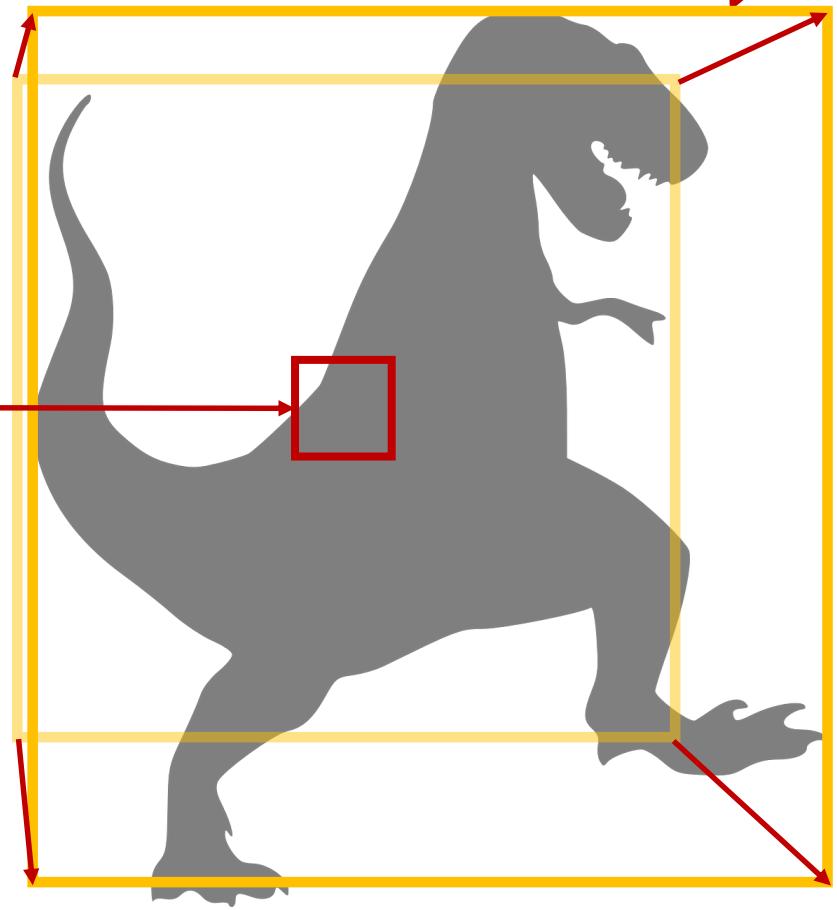
Five output channels:

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

RPN: Prediction (on object case)

Transformed anchor box after box regressor

P(object) = 0.94



1x1 conv

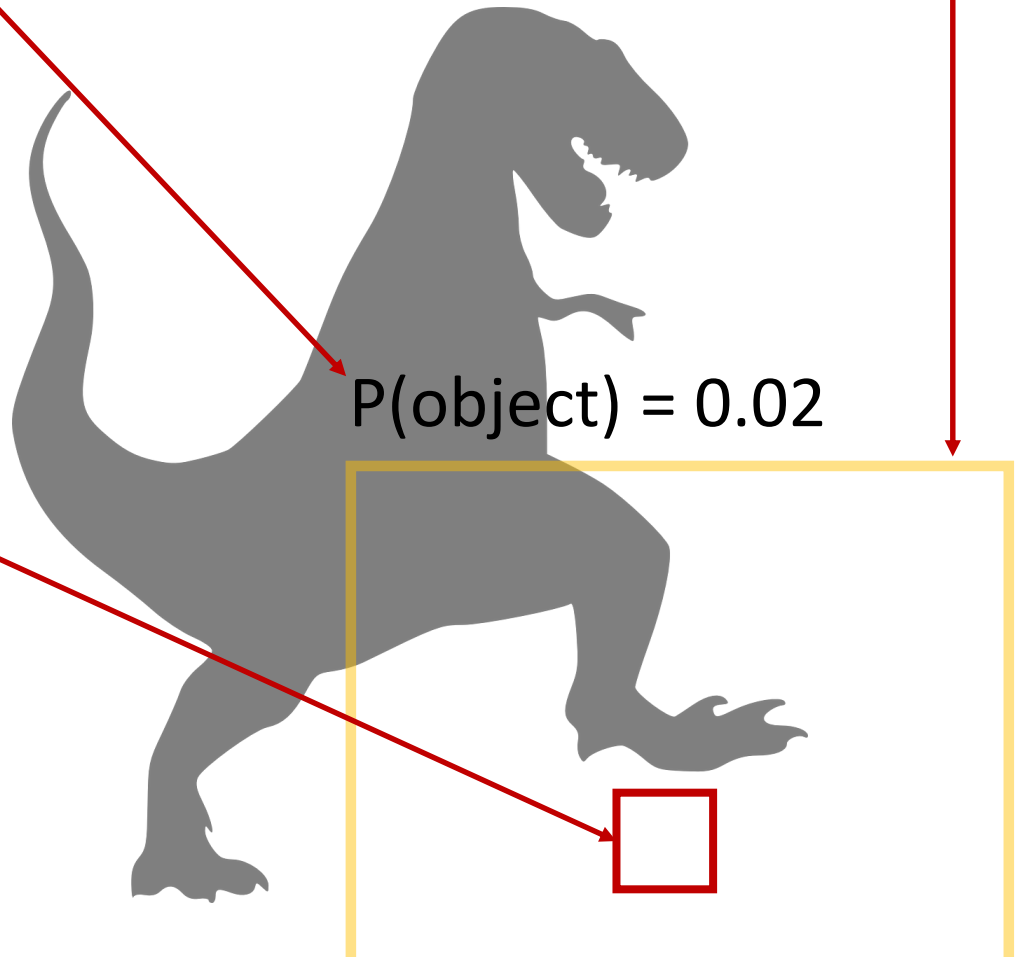
Five output channels:

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

RPN: Prediction (off object case)

Transformed anchor box after box regressor

Objectness score

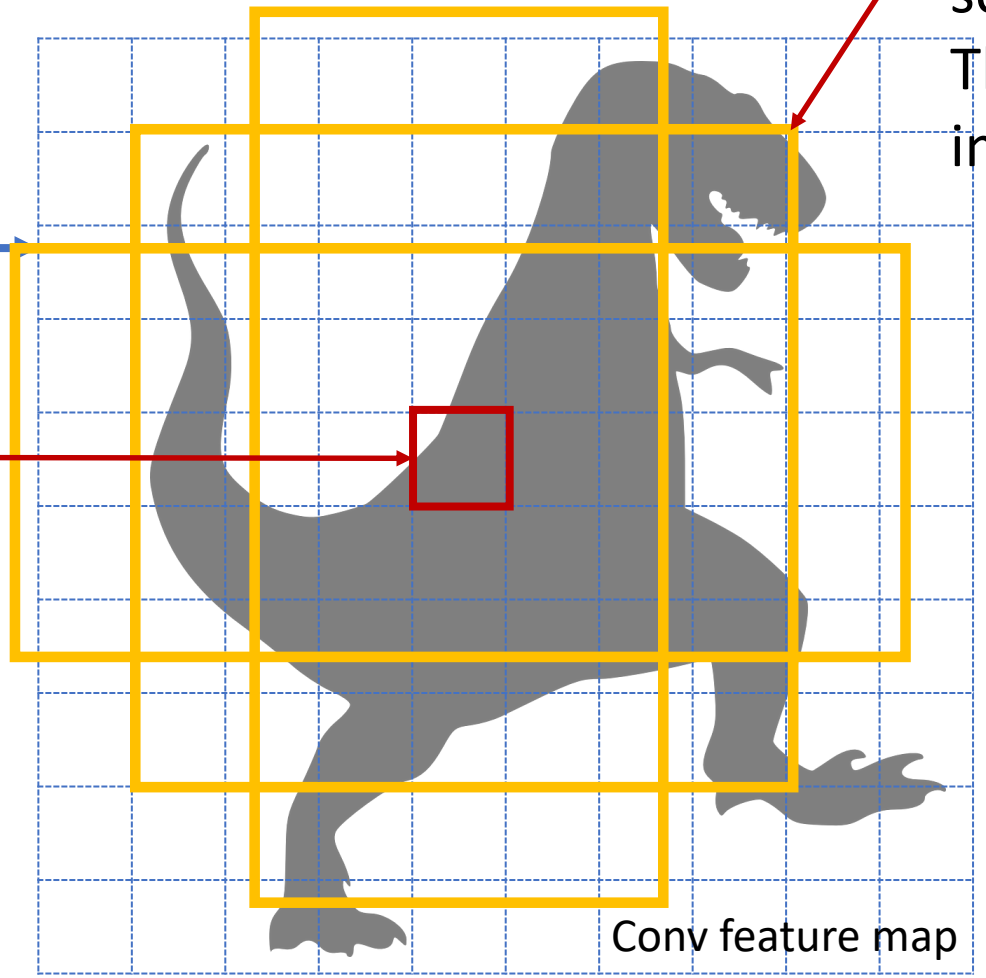


1x1 conv
Five output channels:

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

RPN: Multiple Anchors

f_I = FCN(I)



Anchor boxes: *K* anchors per location with different scales and aspect ratios
This is why decoupling is important

1x1 conv
5*K* output channels:

- *K* objectness classifiers [0, 1]
- *K* box regressors predicting (dx, dy, dh, dw)

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

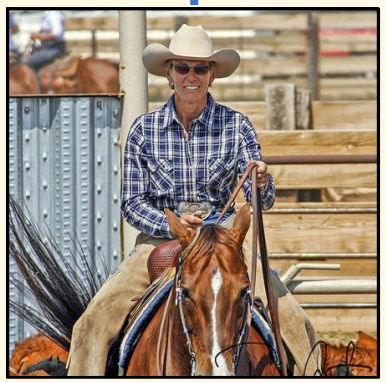
Faster R-CNN

Per-image computation

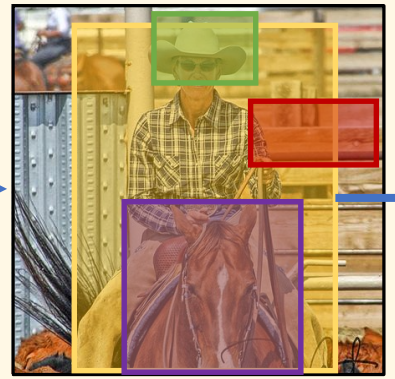
Per-region computation for each $r_i \in r(I)$

f_I = FCN(I)

I:



RPN(f_I)



RoIPool

MLP

Softmax clf.

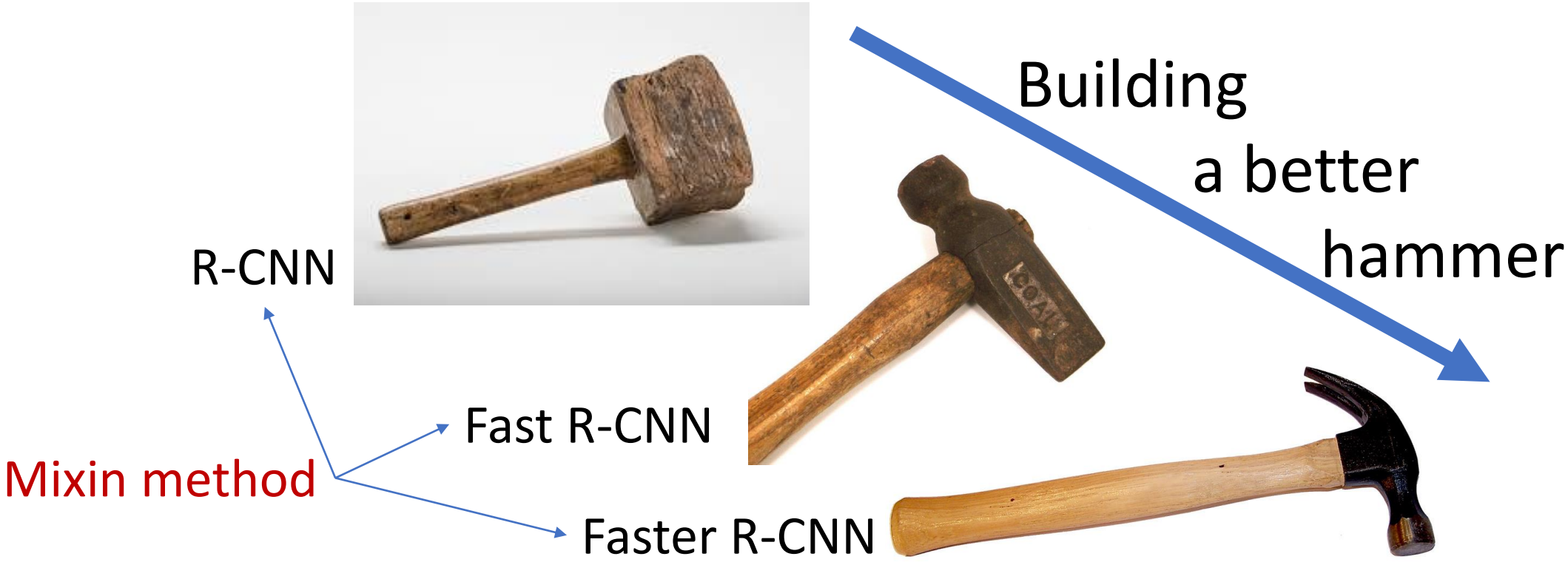
Box regressor

Learned proposals

Shares computation with whole-image network

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)



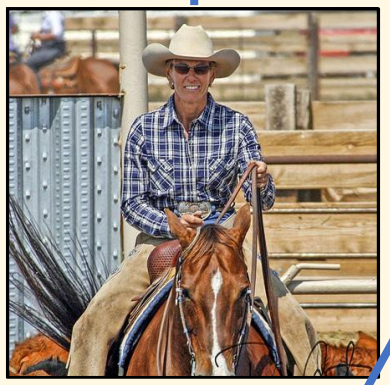
Faster R-CNN with a Feature Pyramid Network

Per-image computation

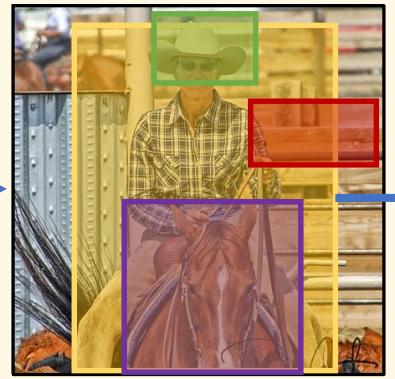
Per-region computation for each $r_i \in r(I)$

f_I = FPN(I)

I:



RPN(f_I)



RoIPool

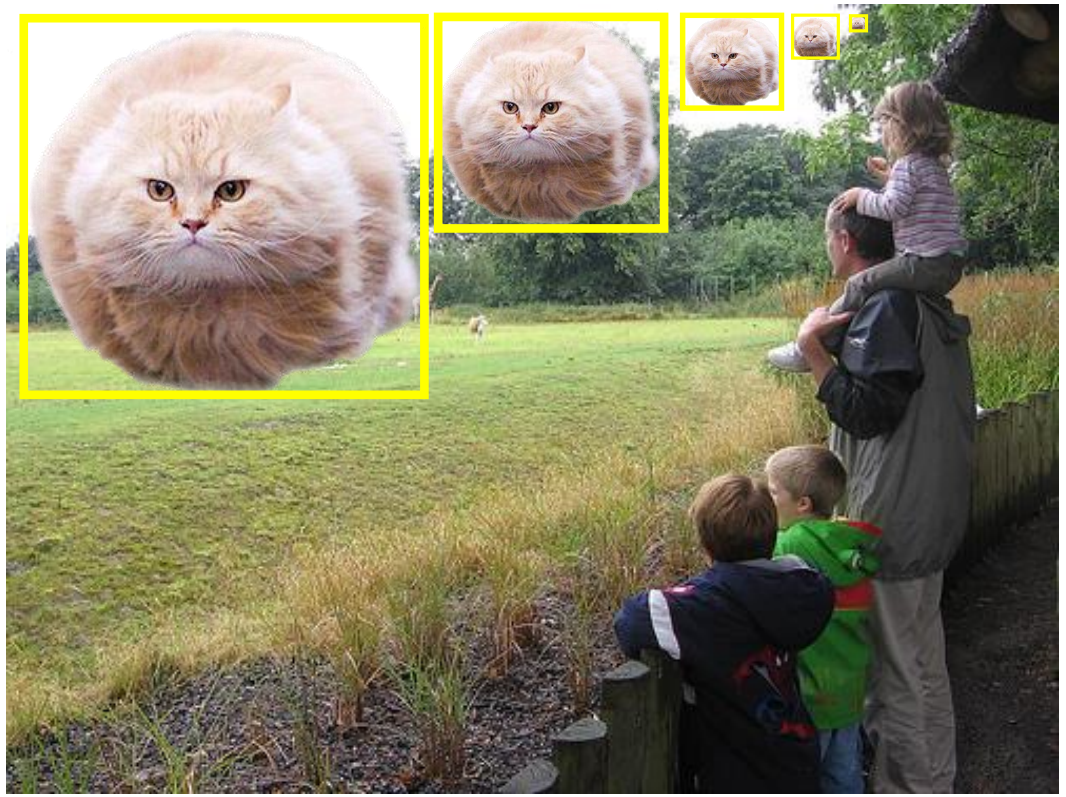
MLP

Softmax clf.

Box regressor

The whole-image feature representation can be improved by making it *multi-scale*

FPN: Improving Scale Invariance and Equivariance



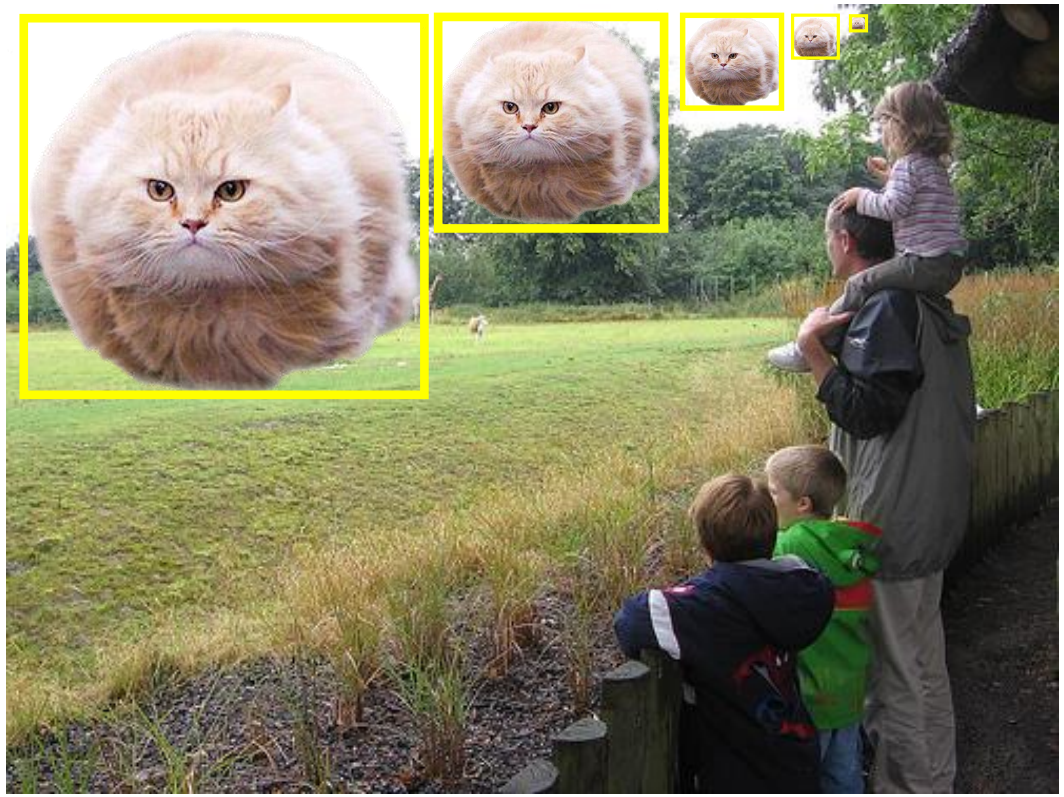
Detectors need to

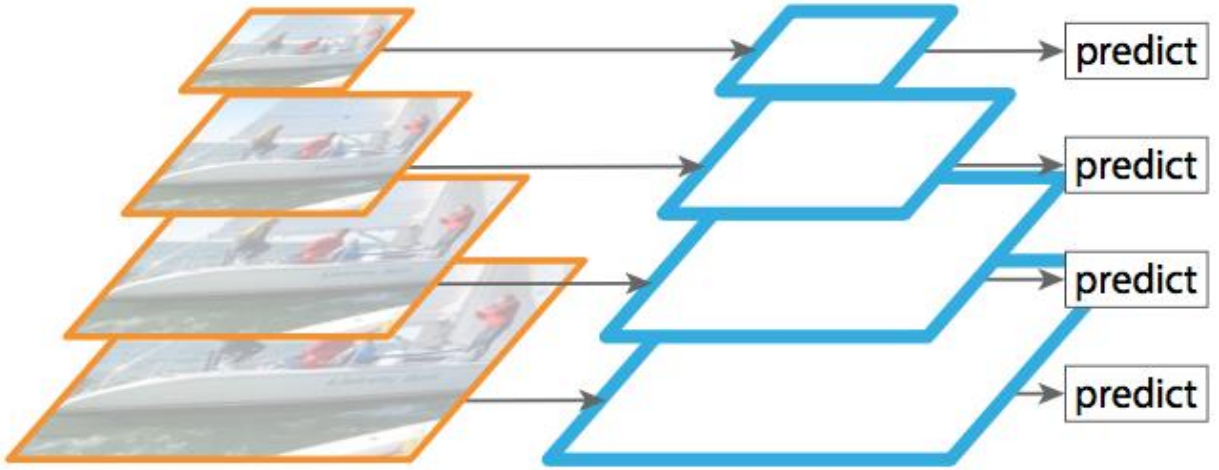
1. Classify (invariance) and
2. Localize (equivariance)

objects over a wide range of scales

FPN improves this ability

Strategy 1: Image Pyramid



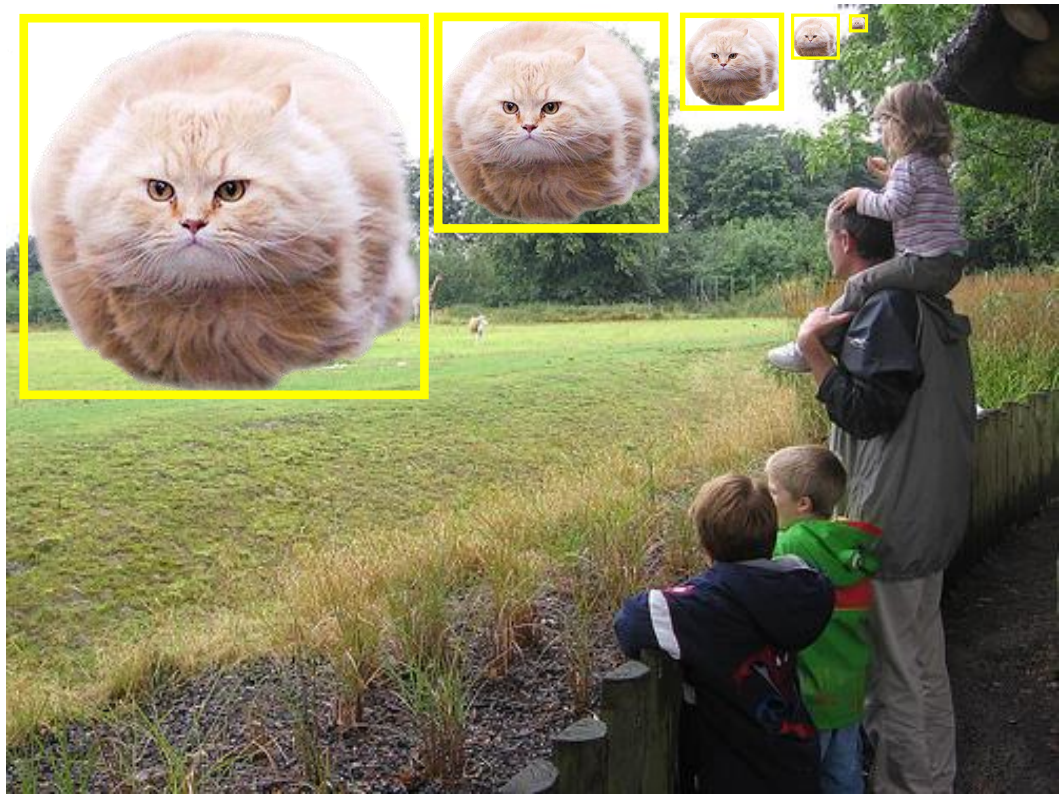


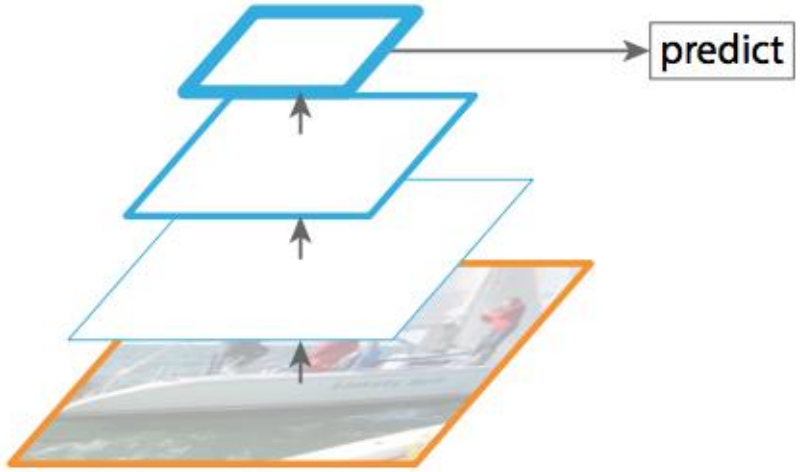
(a) Featurized image pyramid

Standard solution – *slow!*

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

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



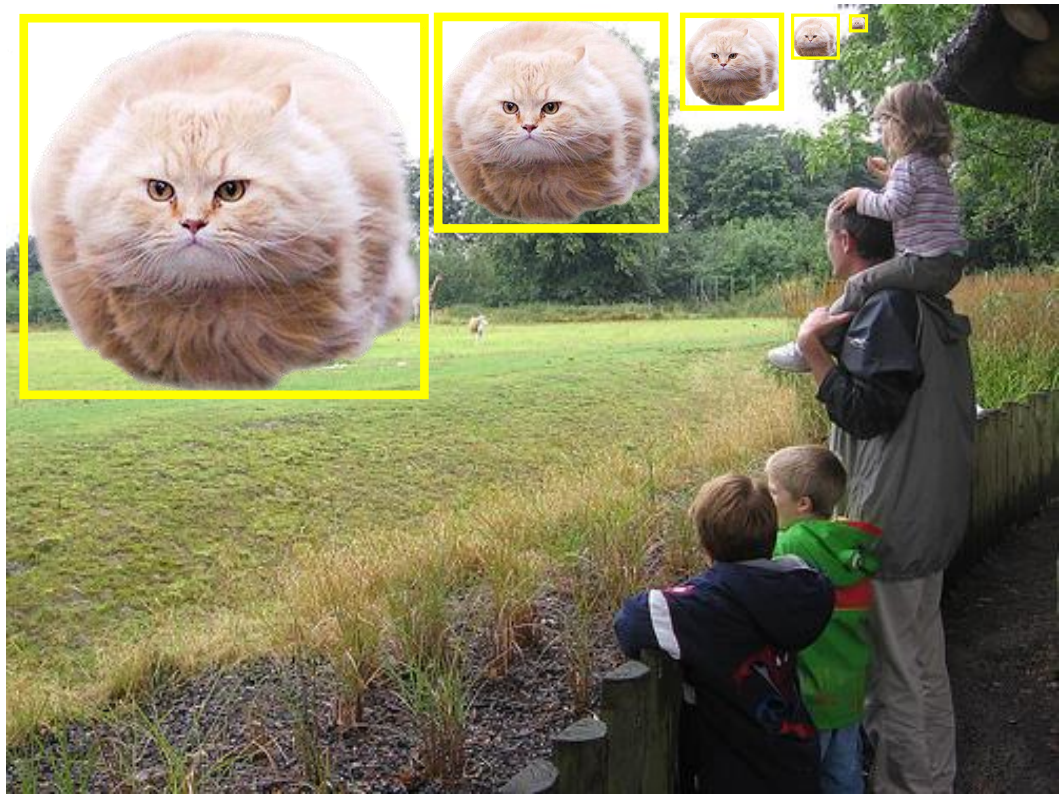


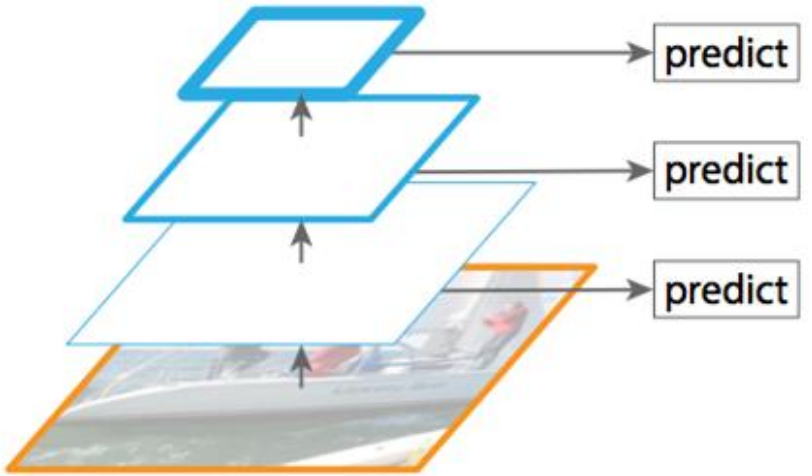
(b) Single feature map

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

(E.g., Fast/er R-CNN, YOLO, …)

Strategy 3: Naïve In-network Pyramid



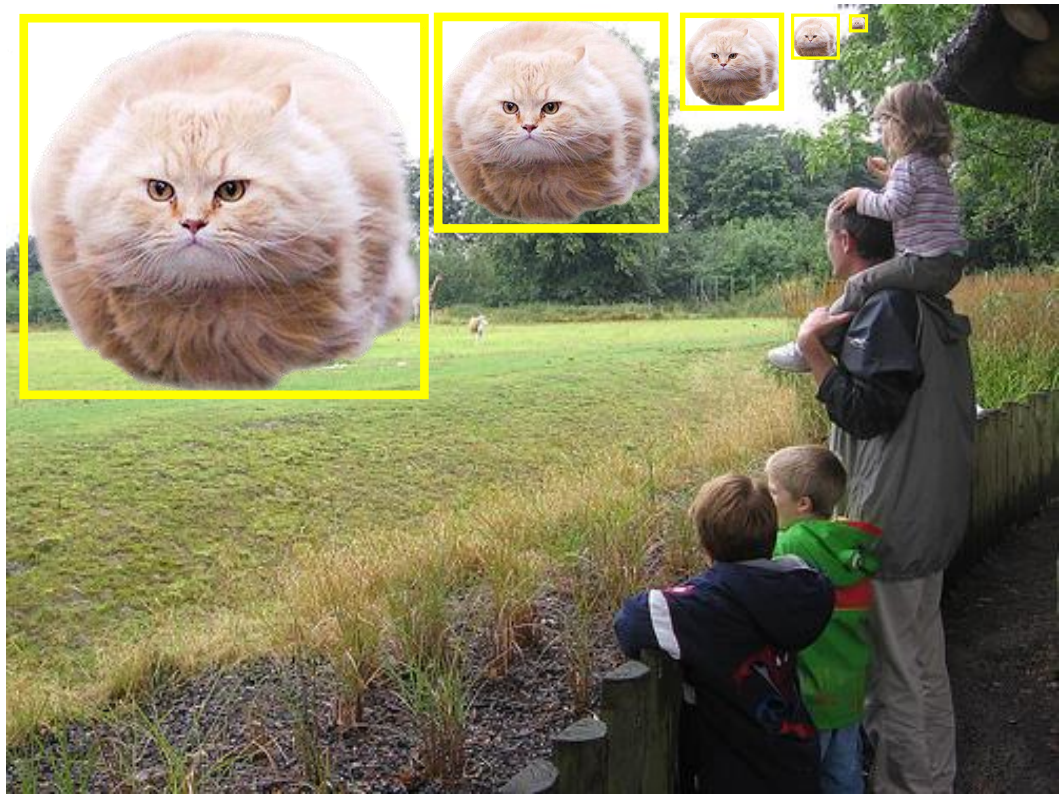


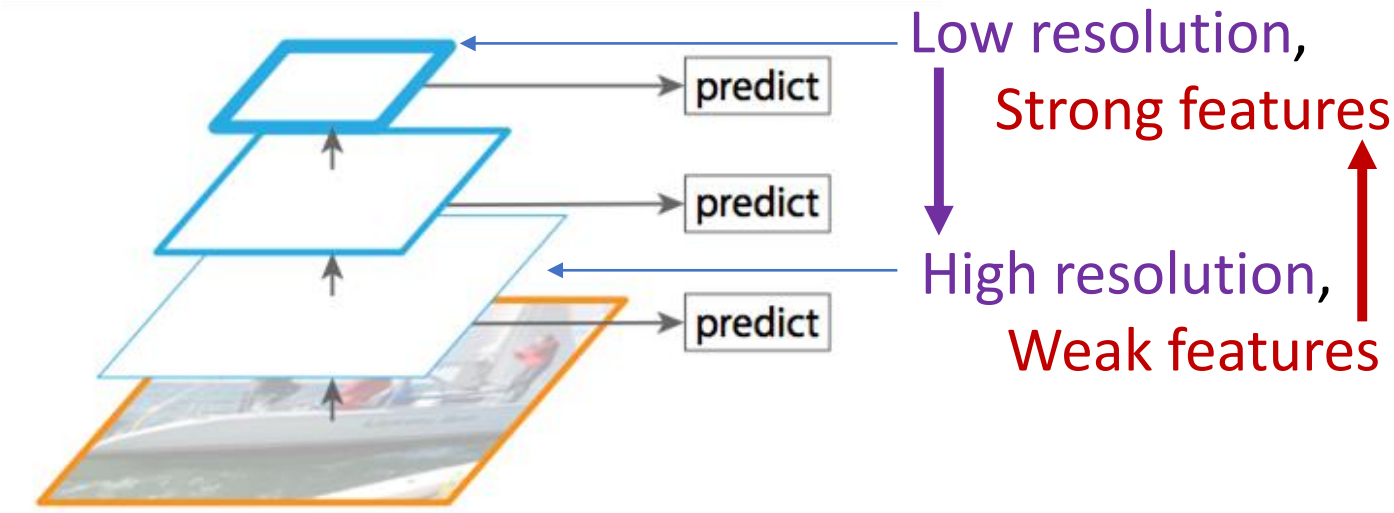
(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Strategy 3: Naïve In-network Pyramid



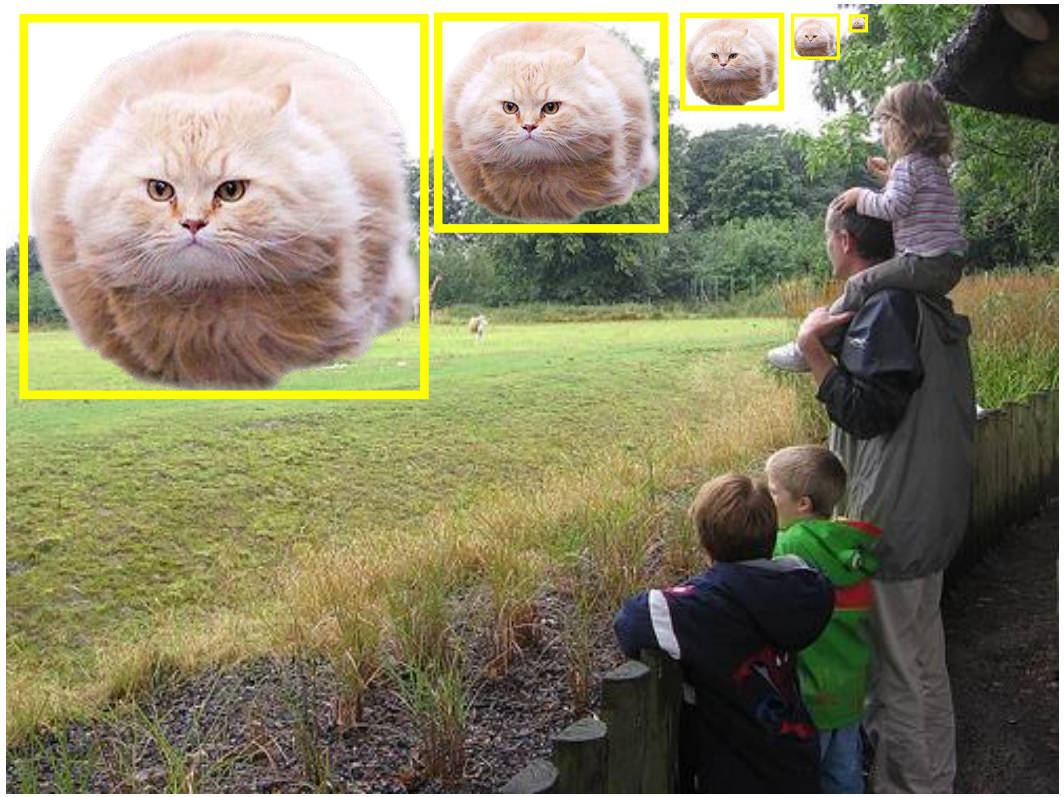


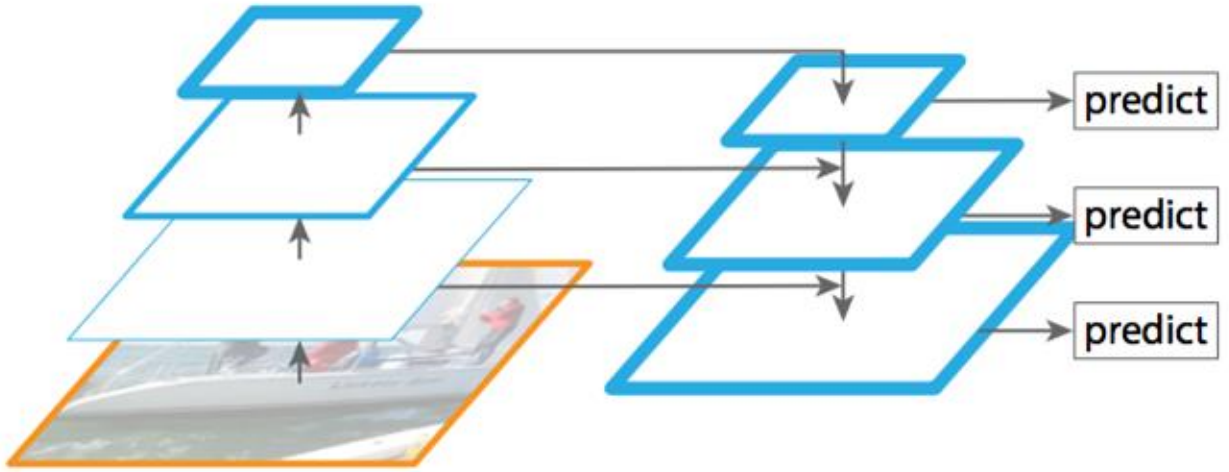
(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Strategy 4: Feature Pyramid Network



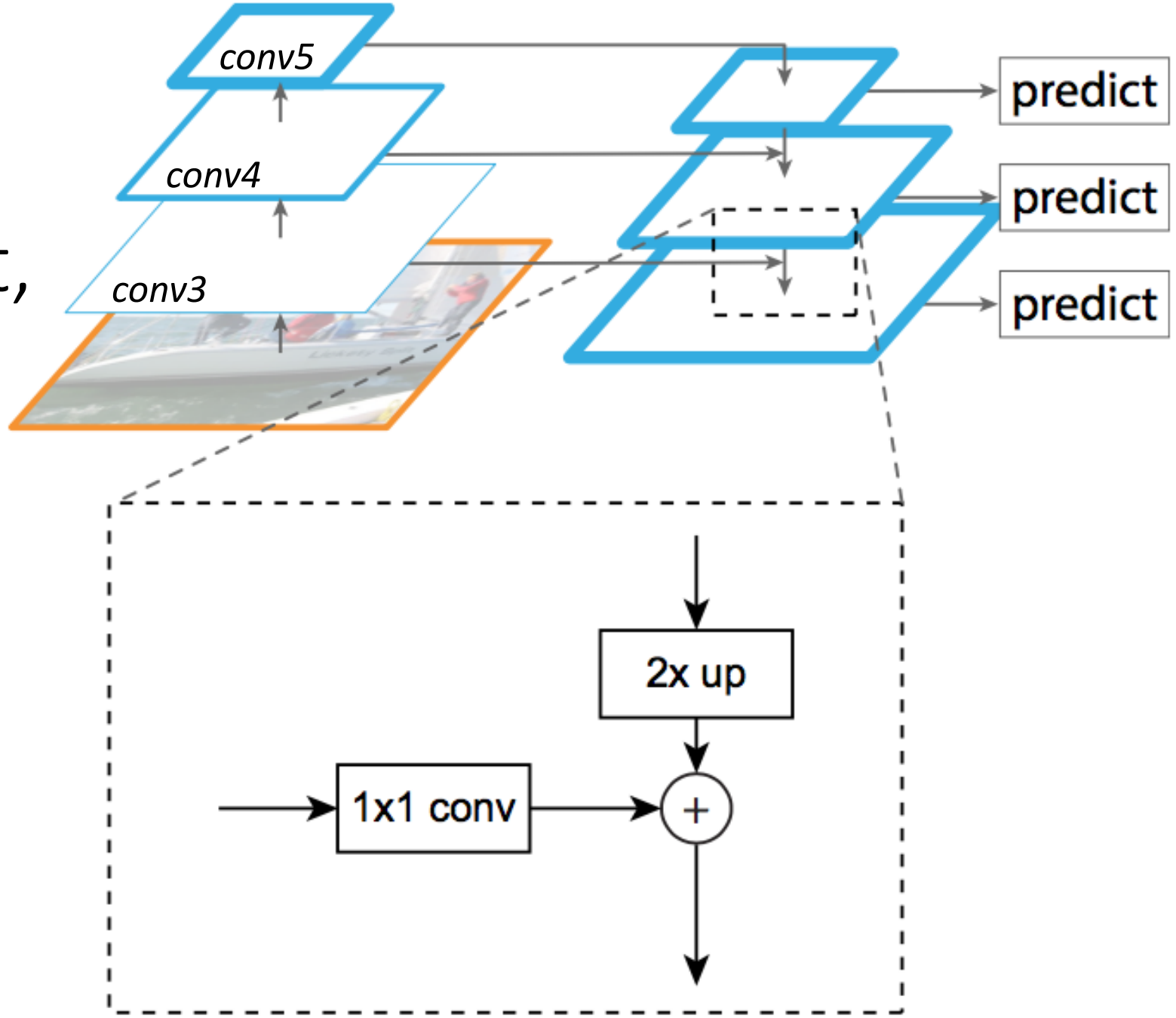


(d) Feature Pyramid Network

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

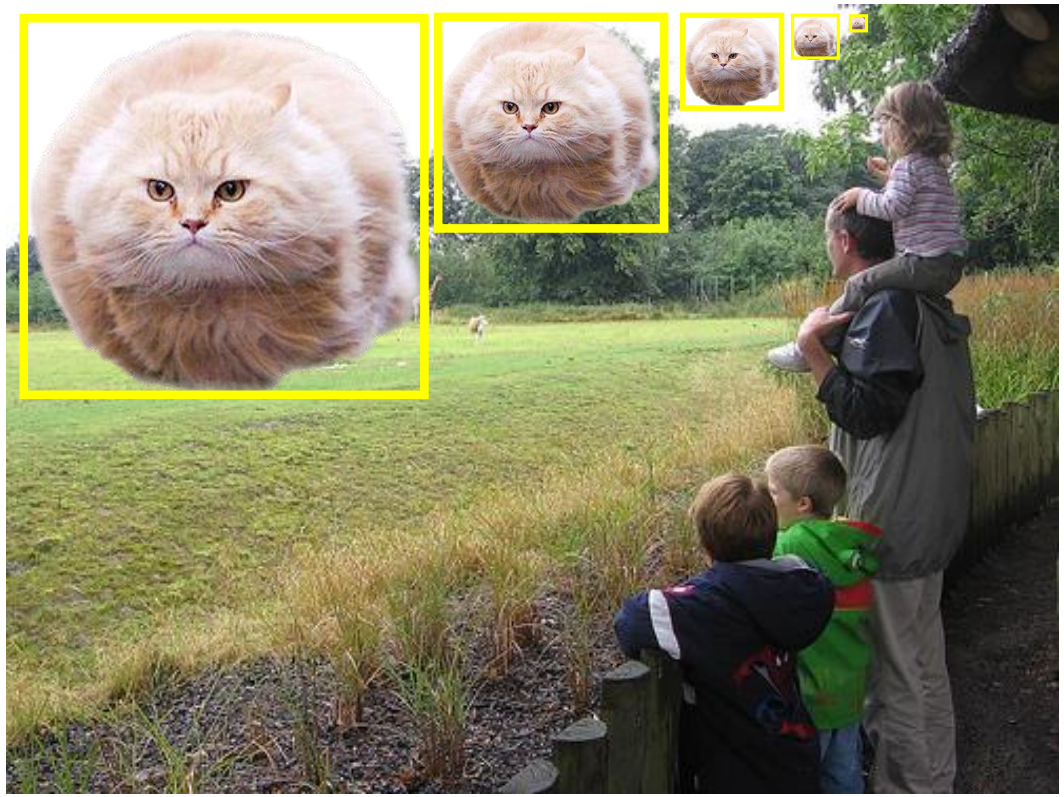
Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017.

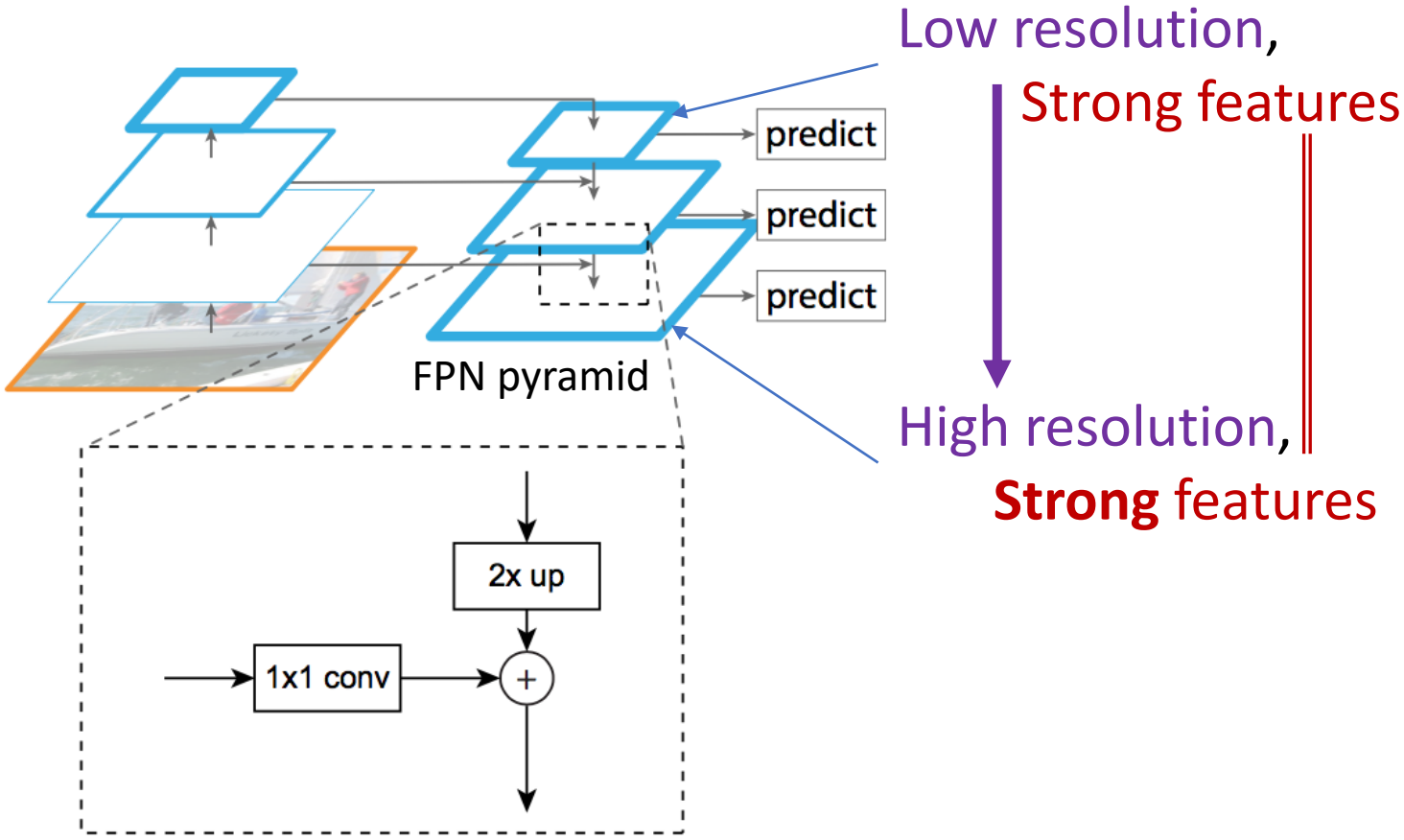
FPN: Light-weight, Top-down Refinement Module



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

No Compromise on Feature Quality, Still Fast





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
- Panoptic FPN
- …

Mask R-CNN: The Final Hammer of this Tutorial

R-CNN

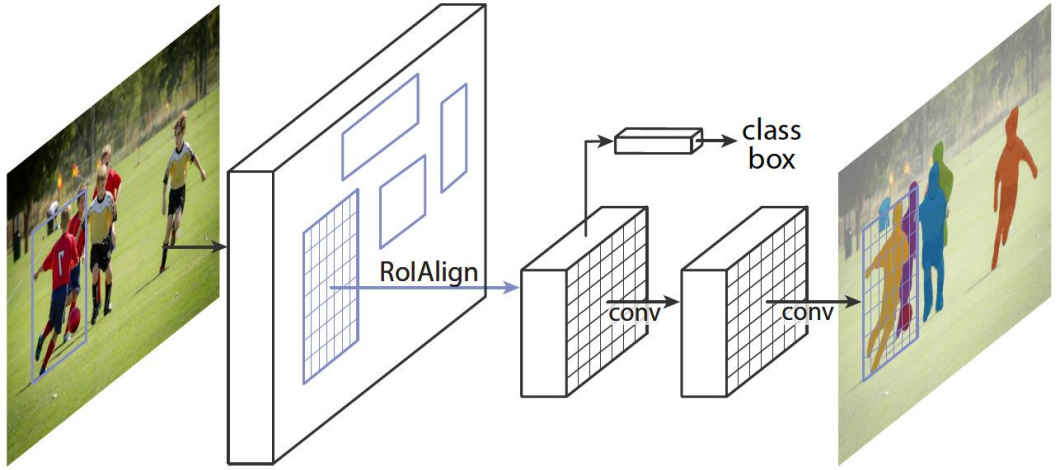
Building a better hammer

Fast R-CNN

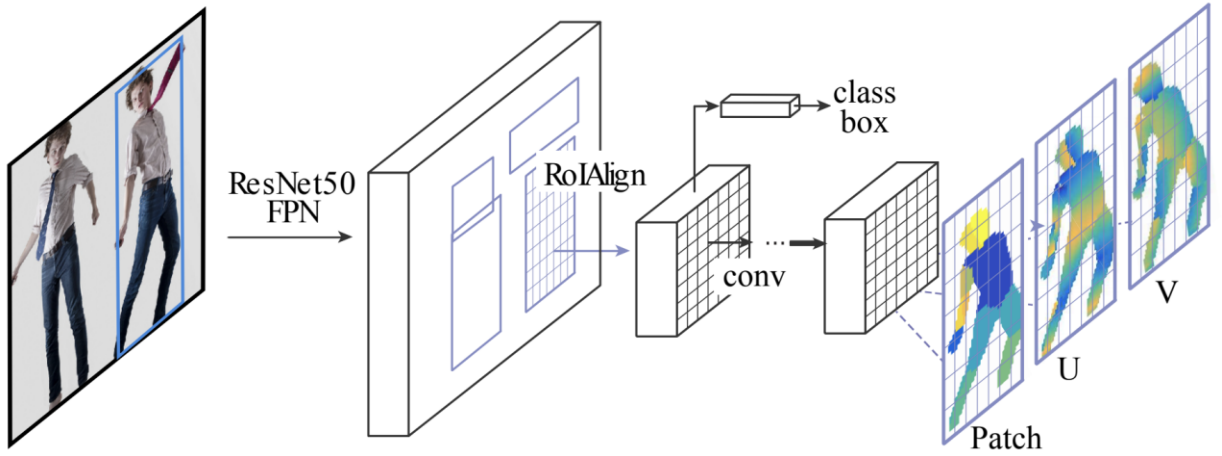
Faster R-CNN

Mask R-CNN

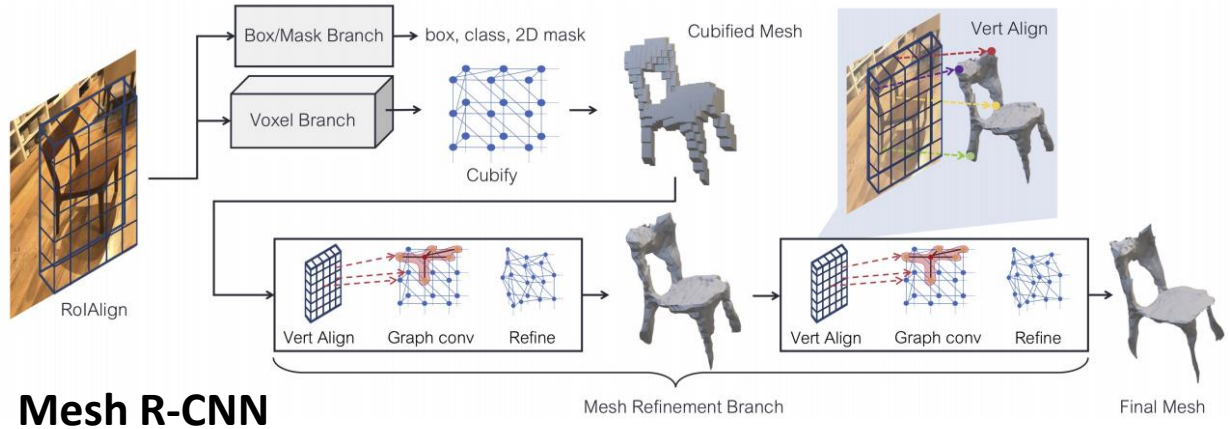
Generalized R-CNN: Adding More Heads



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



DensePose
[Güler, Neverova, Kokkinos]



Mesh R-CNN
[Gkioxari, Malik, Johnson. ICCV 2019]

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

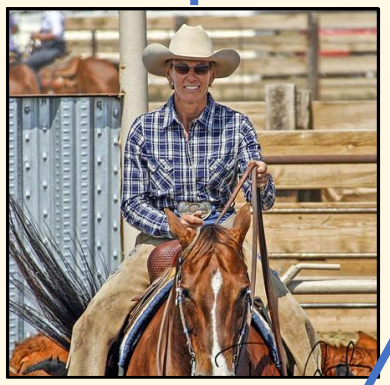
Mask R-CNN

Per-image computation

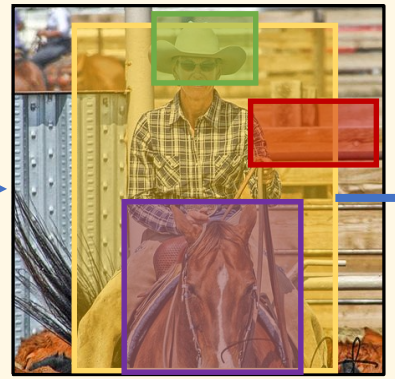
Per-region computation for each $r_i \in r(I)$

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

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

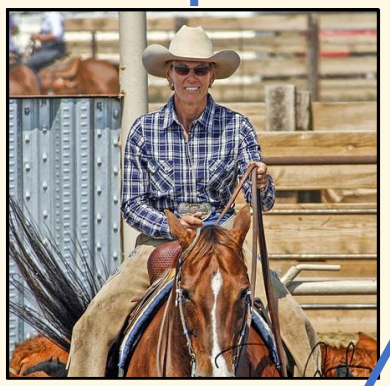
Mask R-CNN

Per-image computation

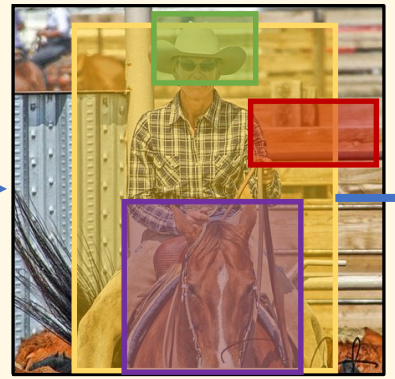
Per-region computation for each $r_i \in r(I)$

f_I = FPN(I)

I:



RPN(f_I)



RoIAlign

MLP

Softmax clf.

Box regressor

Cascaded heads (inference only)

FCN

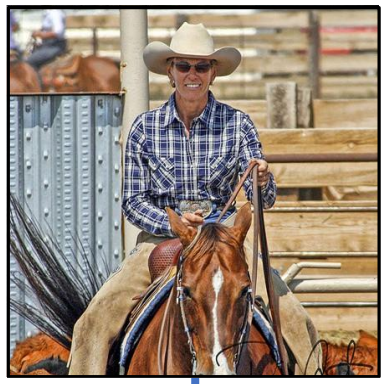
Masks

RoI**Pool** is replaced with RoI**Align**

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

RoIAlign Operation (on each Proposal)

Transform arbitrary size proposal into a fixed-dimensional representation (e.g., 7x7)



f_I = FCN(I)

Region of Interest (RoI)

Grid of bilinear interpolation points

(Variable size RoI)

(Conv feature map)

RoIAlign transform

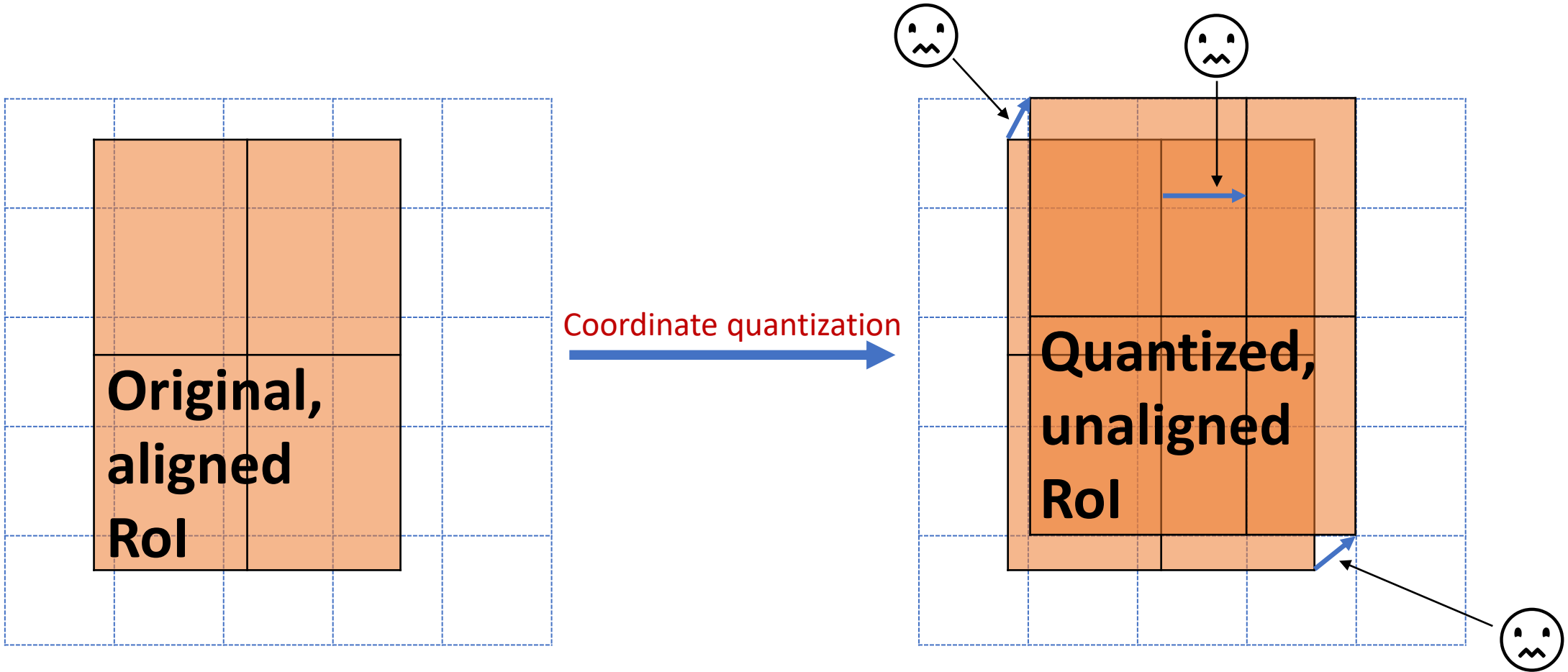
(Fixed dimensional representation)

MLP/FCN

Feature value is average of interpolated values

Compare to RoIPool and RoIPool

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



Inference (Faster and Mask R-CNN)

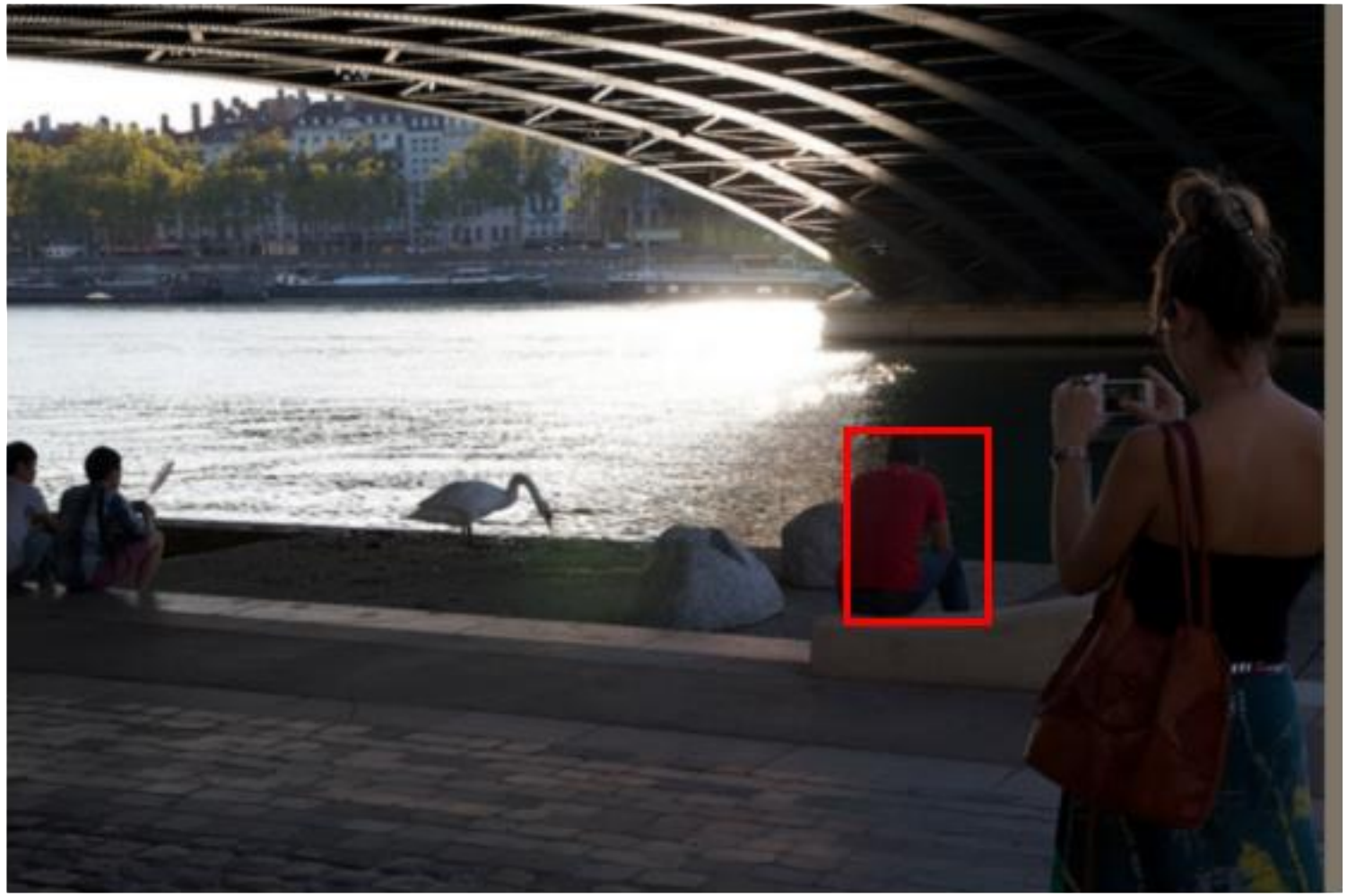
1. Perform Faster R-CNN inference

- Generate proposals (RPN)
- Score the proposals (classifier head)
- Regress from proposals to refined detection boxes (box regressor)
- Apply NMS and take the top *K* (= 100, e.g.)

2. Run RoIAlign and mask head on top-*K* refined, post-NMS boxes

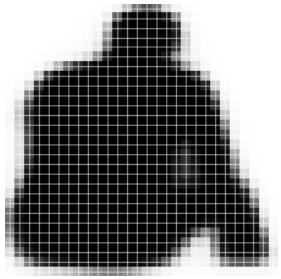
- Fast (only compute masks for top-*K* detections)
- Improves accuracy (uses *refined* detection boxes, not proposals)

Mask Prediction



Validation image with box detection shown in red

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



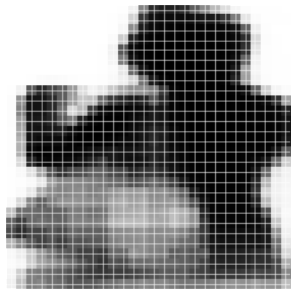
Soft prediction resampled to image coordinates
(bilinear and bicubic interpolation work equally well)

Final prediction (threshold at 0.5)

Mask Prediction

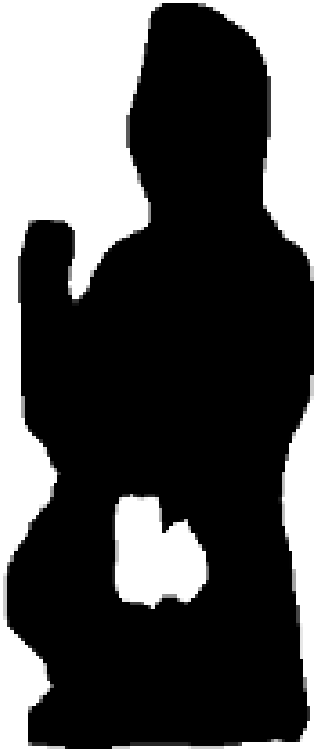
Validation image with box detection shown in red

28x28 soft prediction



Resized soft prediction

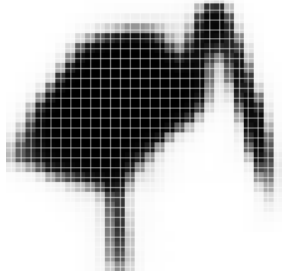
Final mask



Mask Prediction

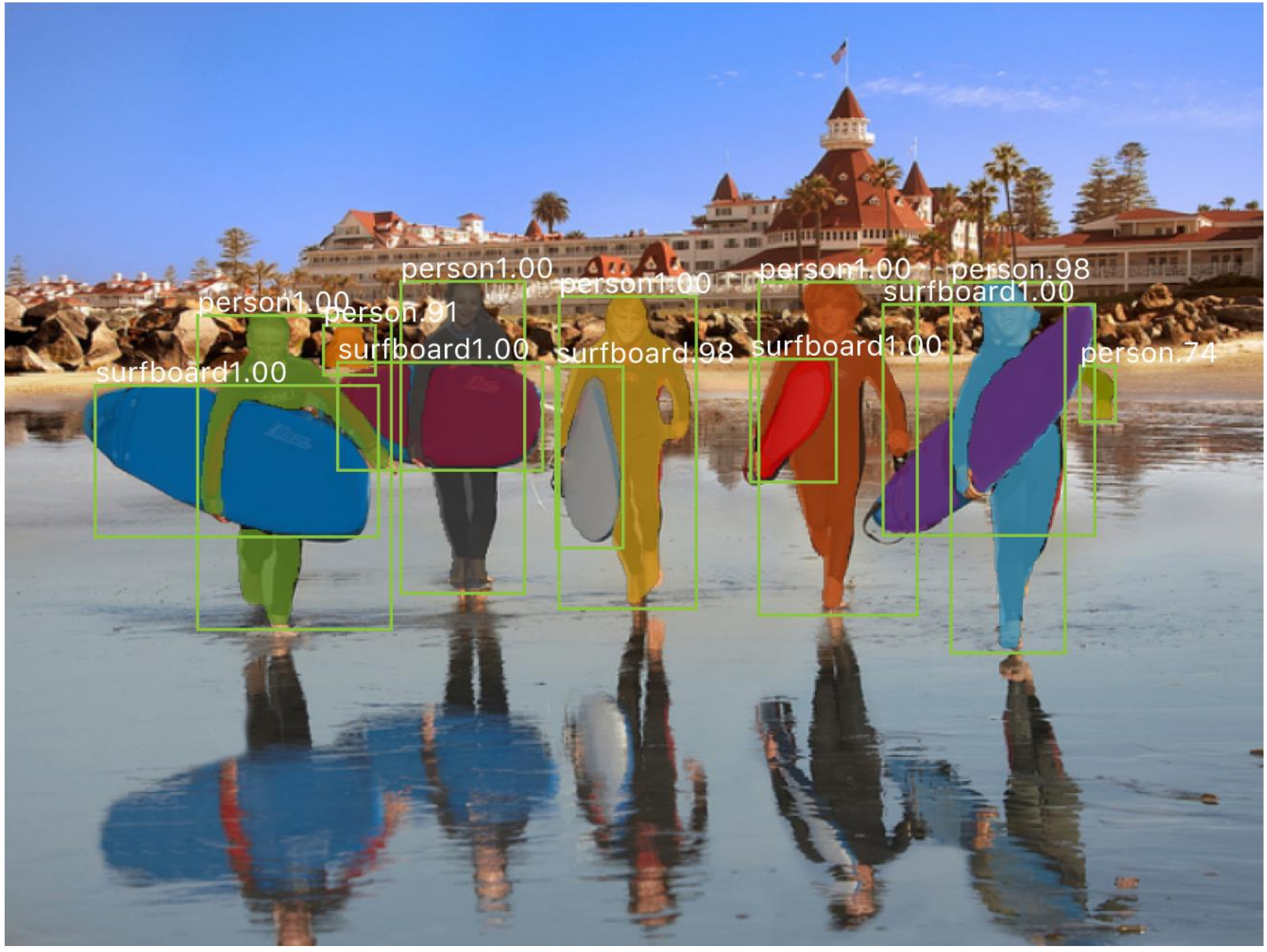
Validation image with box detection shown in red

28x28 soft prediction



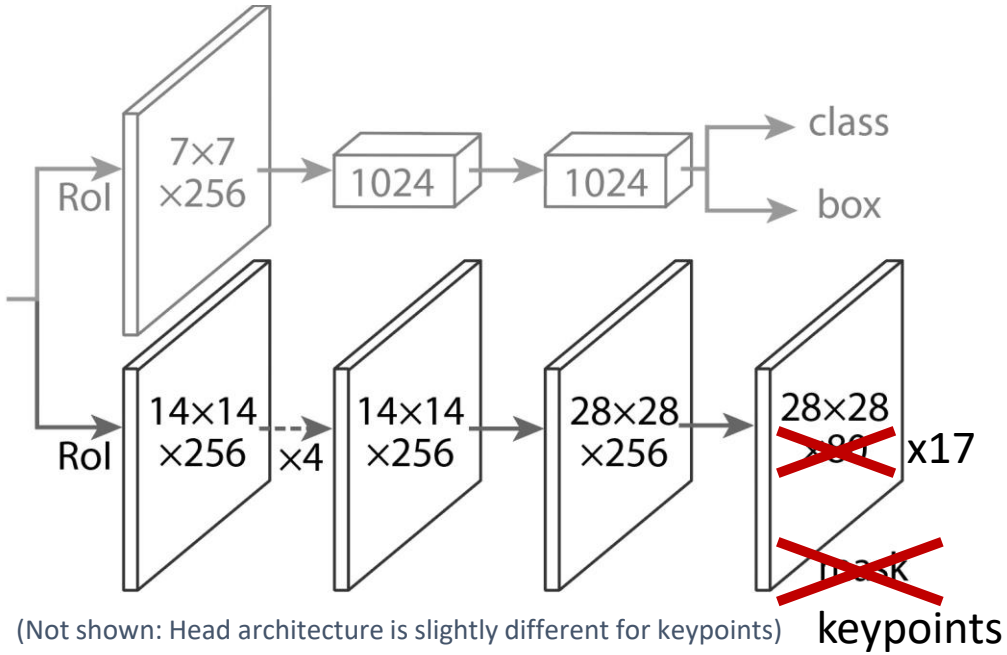
Resized Soft prediction

Final mask

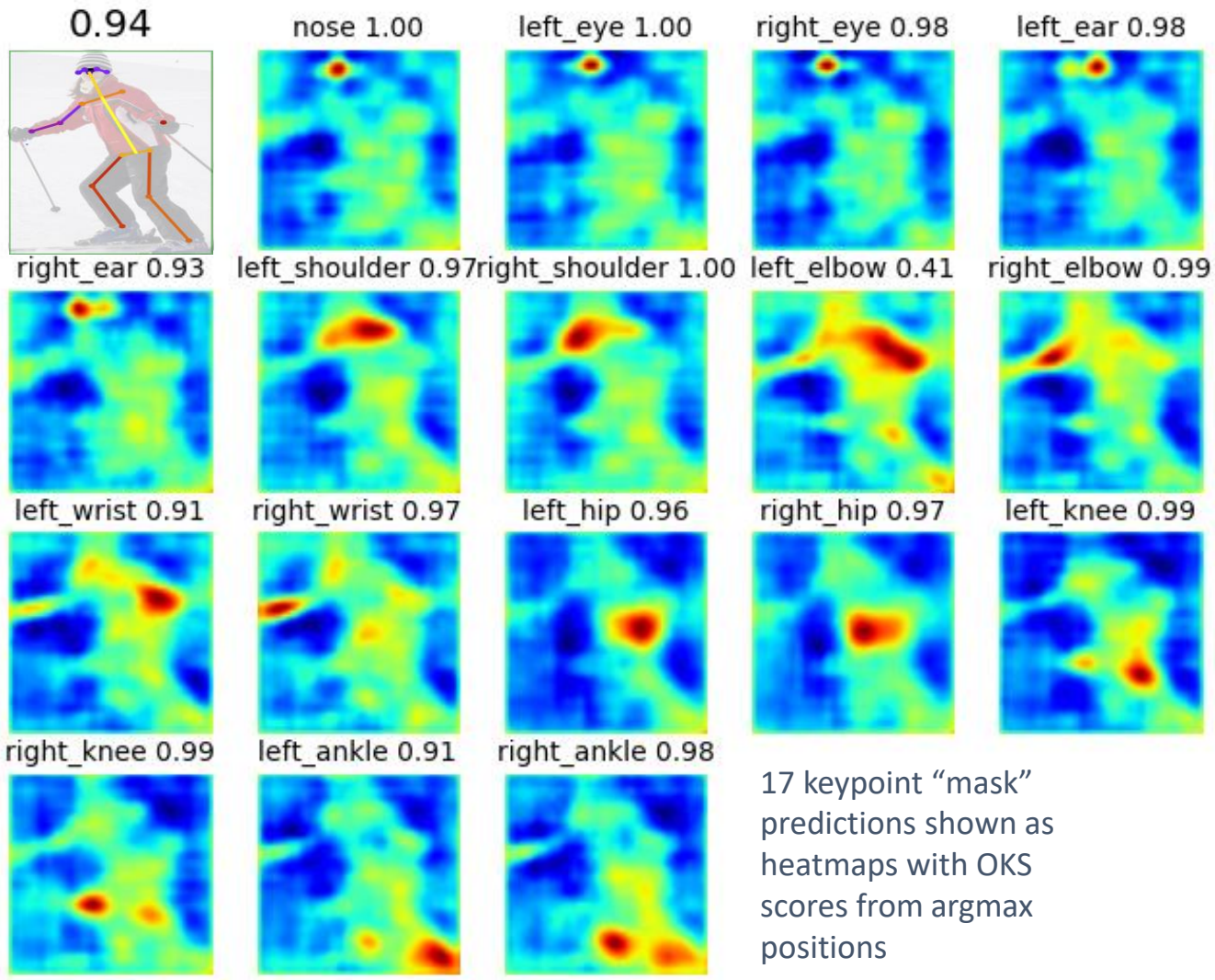

person1.00
person1.00
person1.00
person.98
surfboard1.00
person1.00
person.91
surfboard1.00
surfboard.98
surfboard1.00
person.74
surfboard1.00


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

Human Pose

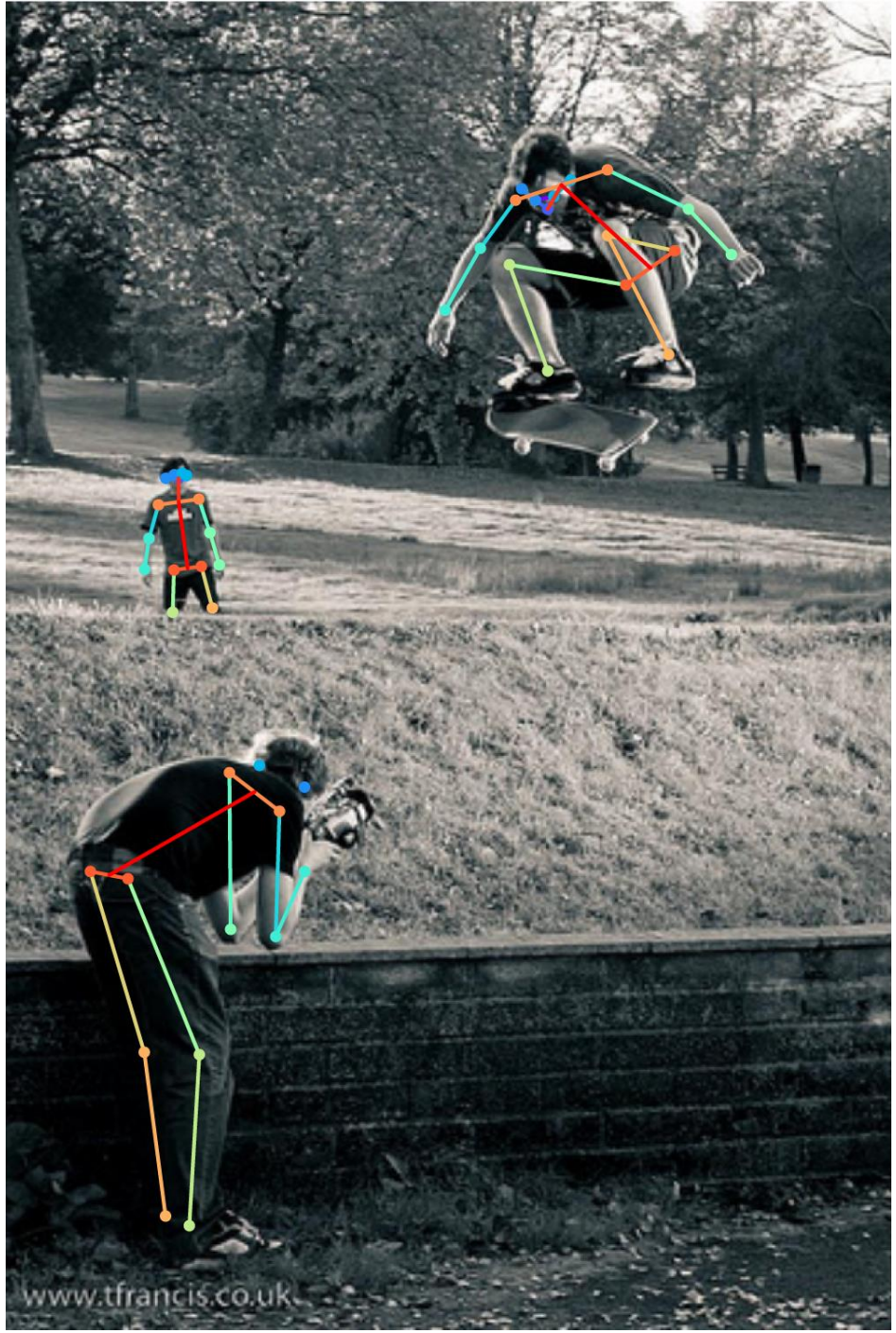


(Not shown: Head architecture is slightly different for keypoints)

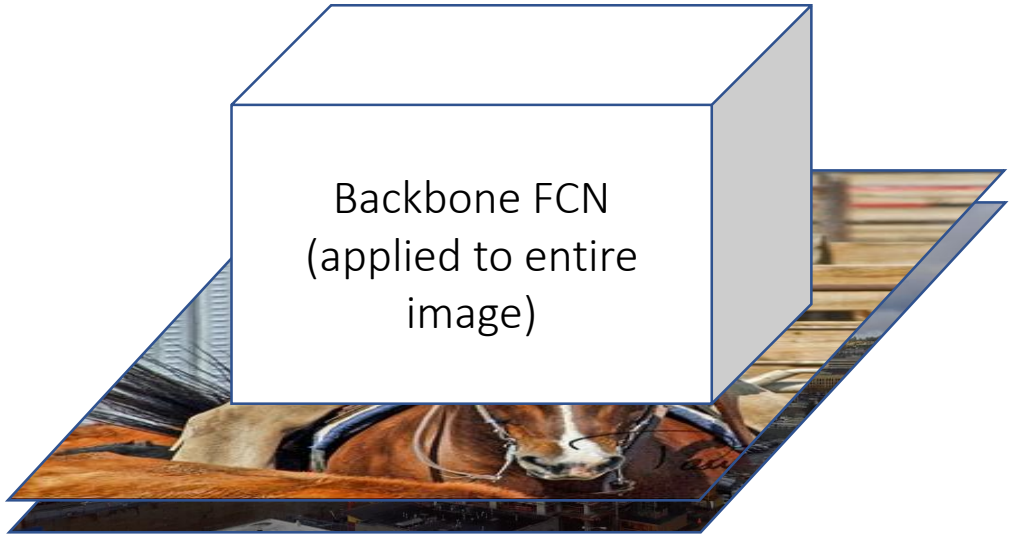


17 keypoint "mask" predictions shown as heatmaps with OKS scores from argmax positions

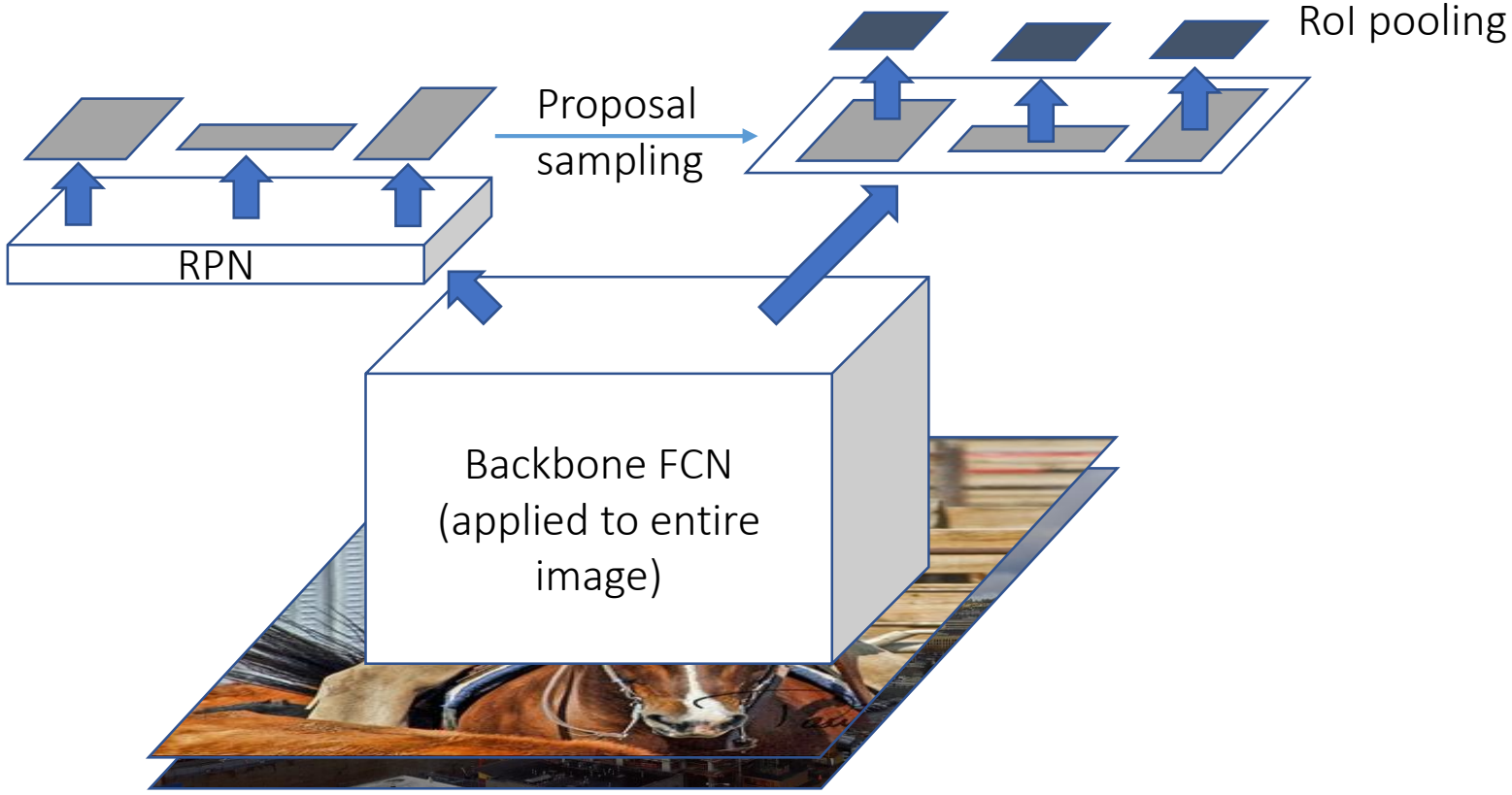
- Add keypoint head (28x28x17)
- Predict one "mask" for each keypoint
- Softmax over spatial locations (encodes one keypoint per mask "prior")


www.tfrancis.co.uk

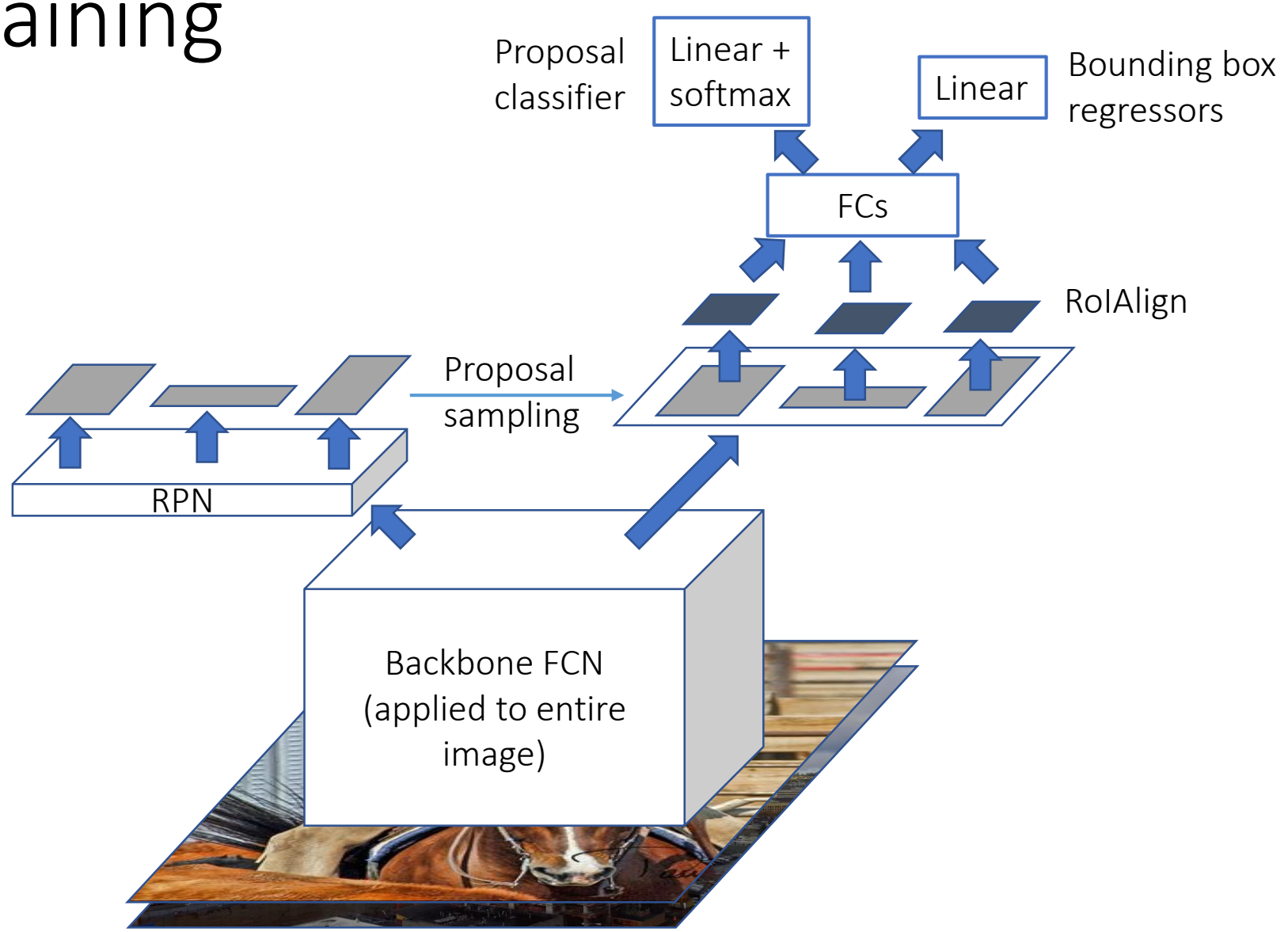
Faster R-CNN Training



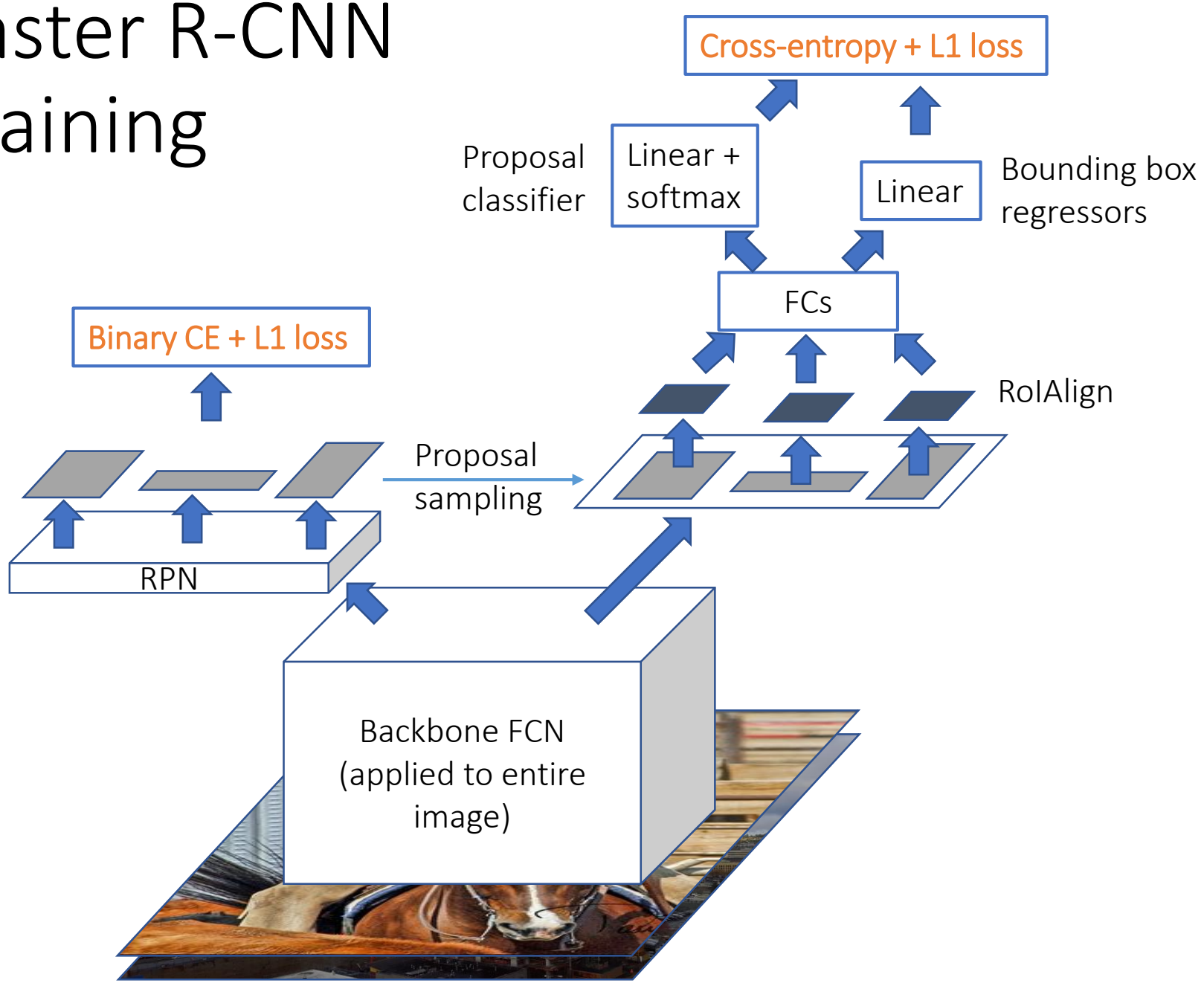
Faster R-CNN Training



Faster R-CNN Training

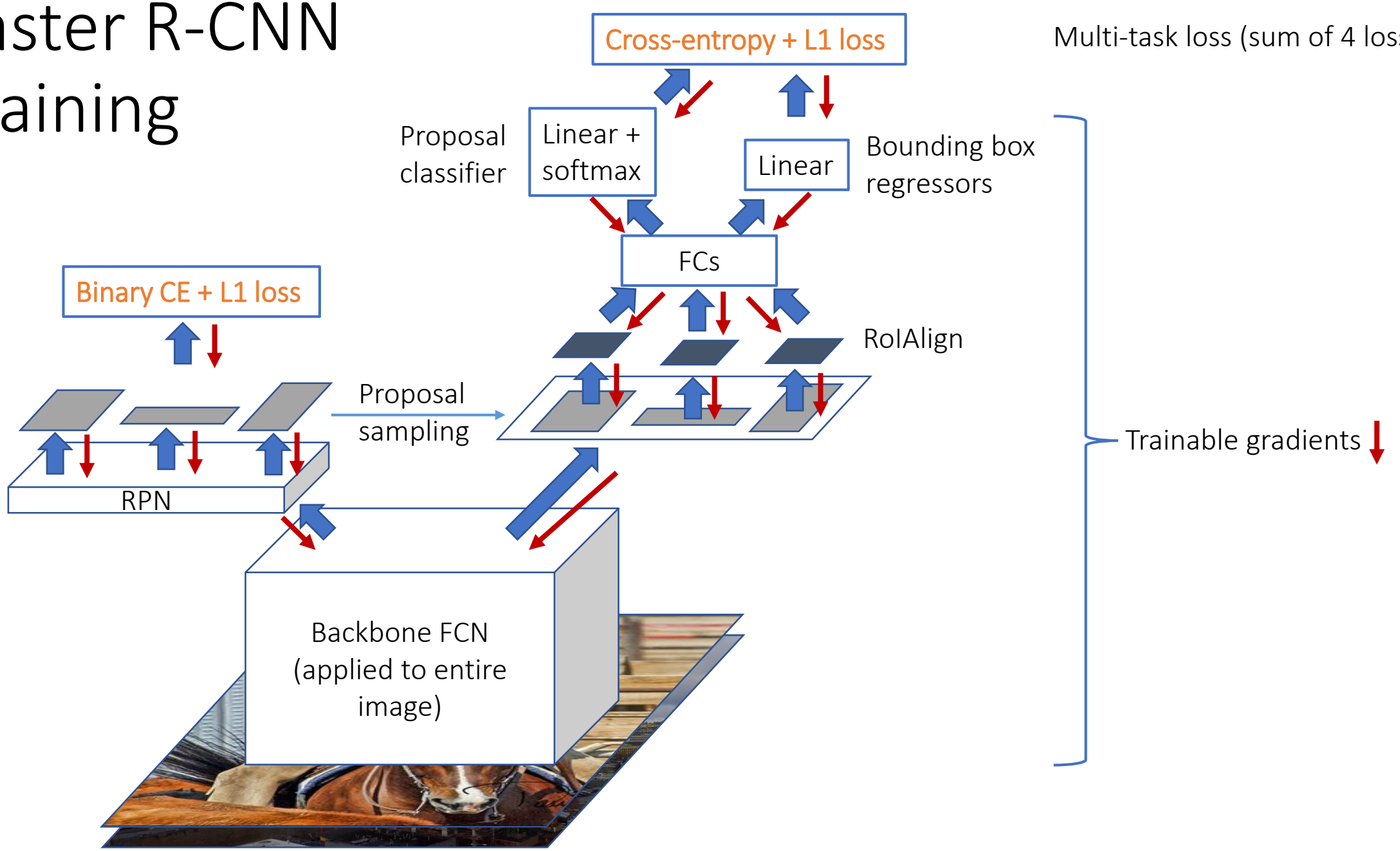


Faster R-CNN Training



Multi-task loss (sum of 4 losses)

Faster R-CNN Training



What We Didn't Talk About

- A lot! Object detection is a huge and very active field
- One stage vs. two-stage detectors (e.g., YOLO, SSD, RetinaNet, etc. vs. R-CNN, Cascade R-CNN, etc.)
- Alternative anchor formulations (e.g., FCOS, CenterNet)
- Set matching approaches (e.g., DETR)
- And more…

Tips on reading papers

- For a newcomer, reading object detection papers is really, really hard!
  - A huge volume of background knowledge is assumed
  - Many implicit community norms
  - Jargon, confusing terminology
  - Misleading comparisons
  - Unscientific statements
  - …
- Many papers are bad, here's what to look for to identify good papers

What did I learn from a paper?

- A paper should be about a focused idea or question
- It should teach you something you didn't know before
- Examples:
  - A new model and training formulation, and what are it's pros and cons?
  - An interesting or surprising empirical finding
  - A simple new baseline that future papers should compare to
  - A new neural network architecture with good scaling properties
  - Showing that a previous result can be reproduced (this is not trivial!)
  - …
- "Novelty" comes in many forms, including new empirical data from existing models

What did I learn from a paper? continued

- If the paper proposes a new model:
  - Under what conditions does it work?
  - When does it *not* work?
  - If the idea has multiple components, which are the most important?
  - Which implementation details are important?
  - What are the empirical runtime and memory requirements?

How papers should show some of these

- Start from a solid baseline
- Apply the main idea to it
- Perform ablations under simple settings
- Show the simplest version that can work well
  - Avoid adding complexity for an additional small gain (e.g., +0.3 AP)

Example ablations: “One table, one message”

net-depth-features	AP	AP_{50}	AP_{75}
ResNet-50-C4	30.3	51.2	31.5
ResNet-101-C4	32.7	54.2	34.3
ResNet-50-FPN	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	**36.7**	**59.5**	**38.9**

(a) **Backbone Architecture**: Better backbones bring expected gains: deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet.

	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

(b) **Multinomial *vs.* Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

	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*

(d) **RoIAlign** (ResNet-50-**C5**, *stride 32*): Mask-level and box-level AP using *large-stride* features. Misalignments are more severe than with stride-16 features (Table 2c), resulting in massive accuracy gaps.

	mask branch	AP	AP_{50}	AP_{75}
MLP	fc: 1024→1024→80·28^2	31.5	53.7	32.8
MLP	fc: 1024→1024→1024→80·28^2	31.5	54.0	32.6
FCN	conv: 256→256→256→256→256→80	**33.6**	**55.2**	**35.3**

(e) **Mask Branch** (ResNet-50-FPN): Fully convolutional networks (FCN) *vs.* multi-layer perceptrons (MLP, fully-connected) for mask prediction. FCNs improve results as they take advantage of explicitly encoding spatial layout.

Table 2. **Ablations** for Mask R-CNN. We train on `trainval35k`, test on `minival`, and report *mask* AP unless otherwise noted.

Example: Mask R-CNN paper

Example ablations: “One table, one message”

	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*

(d) **RoIAlign** (ResNet-50-**C5**, *stride 32*): Mask-level and box-level AP using *large-stride* features. Misalignments are more severe than with stride-16 features (Table 2c), resulting in massive accuracy gaps.

Example: *Where does RoIAlign improve over RoIPool?*
(When feature stride is large; for high IoU thresholds)

Example ablations: “One table, one message”

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

Example: *What makes RoIAlign work?*

(It’s not bilinear vs. nearest; it’s not the aggregation function; it’s avoiding quantization error due to rounding coords)

Supporting claims

- All claims should be supported
  - By an appropriate citation, or
  - By experiment

- Otherwise, statements should be *qualified;* examples:
  - "Increasing X is important for Y" vs.
    "*Intuitively*, increasing X is important for Y…"
    This statement is your intuition (not fact), others may disagree!

  - "Increasing X leads to improved Y…" vs.
    "Increasing X *may* lead to improved Y…"
    Expresses uncertainty or that some conditions may apply

Support all of Your Claims

In principle Mask R-CNN is an intuitive extension of Faster R-CNN, yet constructing the mask branch properly is critical for good results. Most importantly, Faster R-CNN was not designed for pixel-to-pixel alignment between network inputs and outputs. This is most evident in how *RoIPool* [18, 12], the *de facto* core operation for attending to instances, performs coarse spatial quantization for feature extraction. To fix the misalignment, we propose a simple, quantization-free layer, called *RoIAlign*, that faithfully preserves exact spatial locations. Despite being

Evidence

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

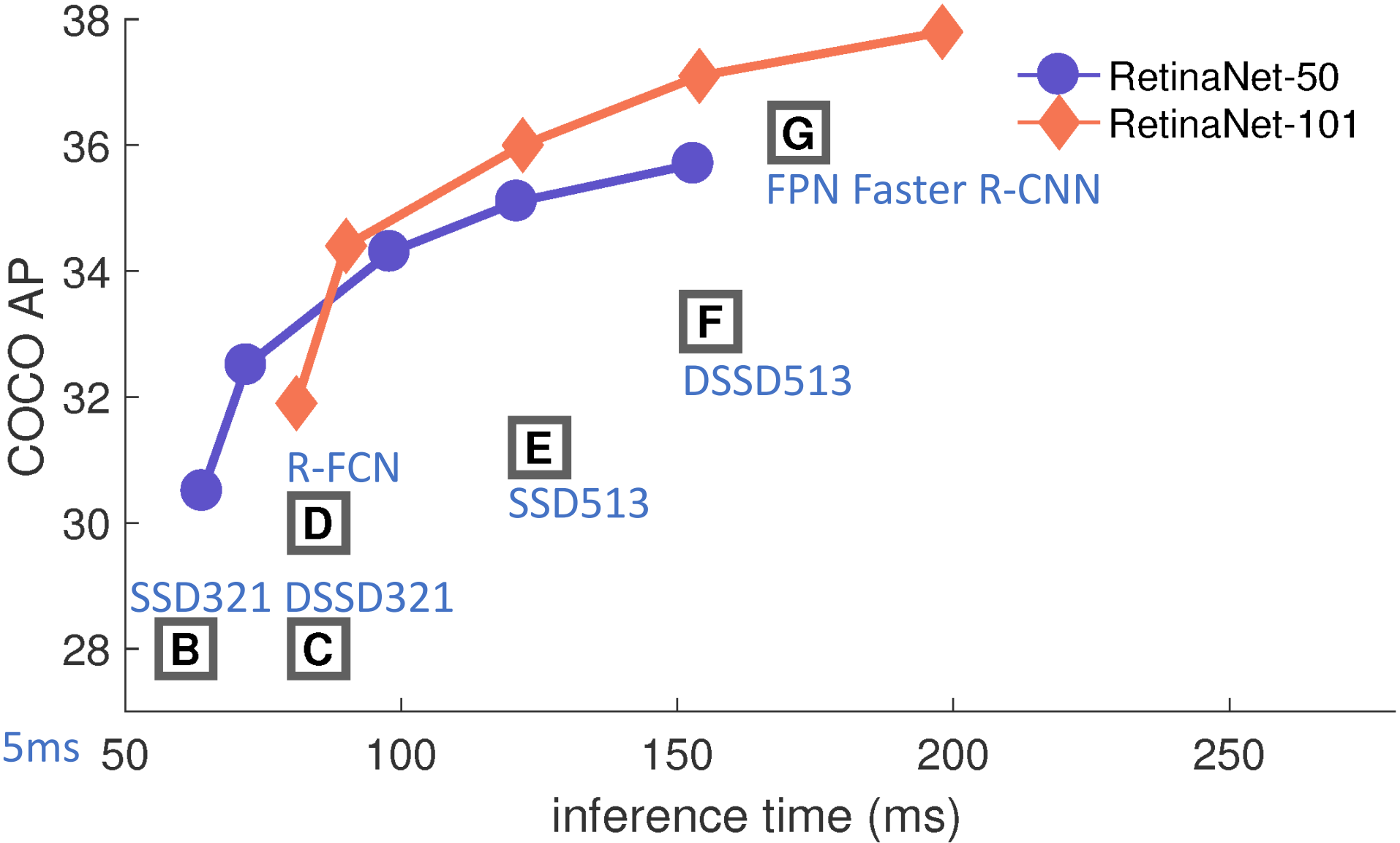
(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

Example: Mask R-CNN paper

Claim: alignment (defined as not rounding coords) is the key

Table: experimental evidence (bilinear is not key; max vs. ave is not key)

Beware of speed vs. accuracy claims

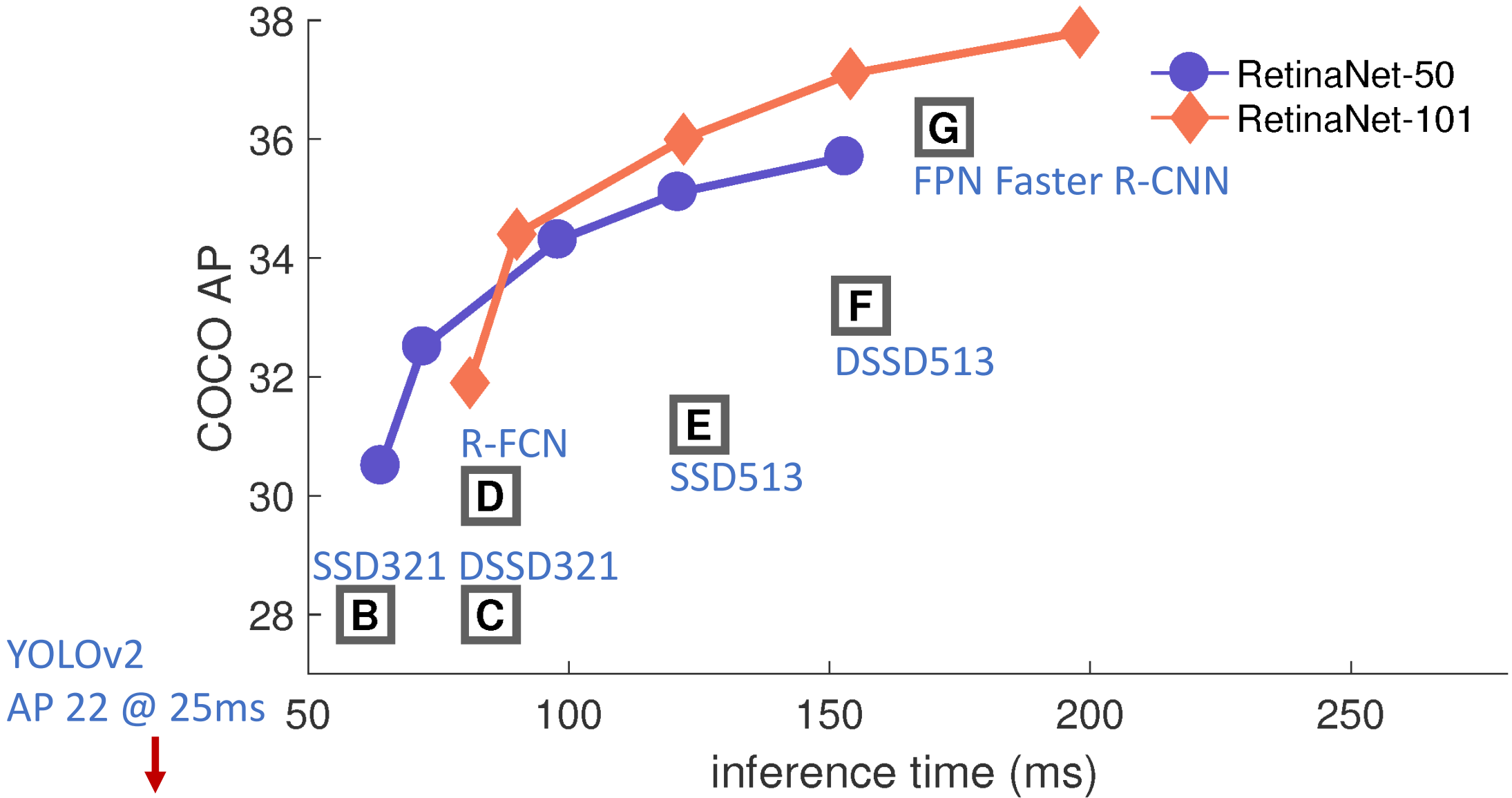


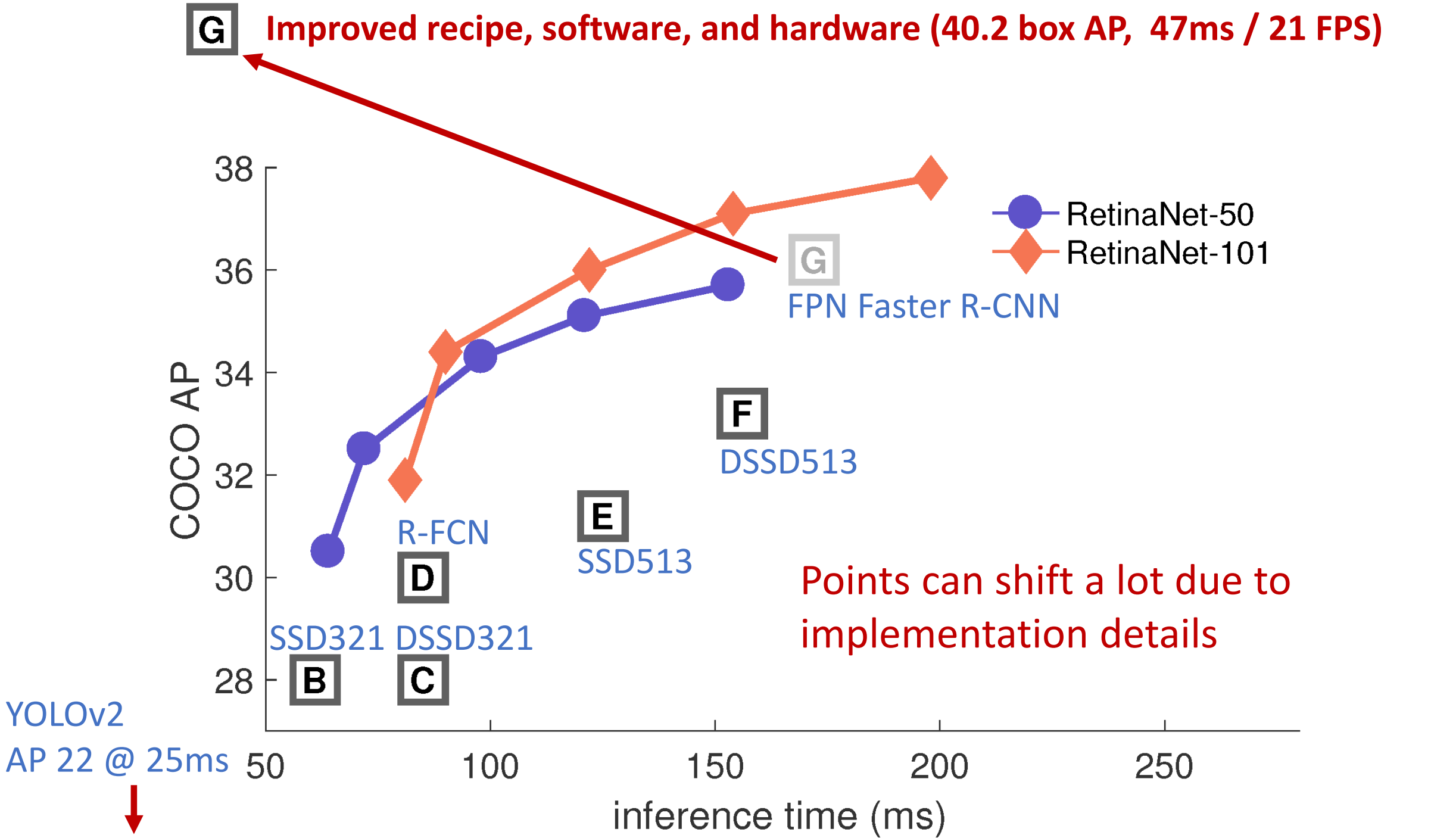
YOLOv2
AP 22 @ 25ms

Beware of speed vs. accuracy claims

Comparisons across publications are often *uncontrolled*

- Accuracy varies with hyper-parameters (often called the “recipe”)
- Speed varies with low-level optimization (perf tuning) and hardware
- Speed varies with inference details (e.g., batching during inference)
- Someone else writes fast code for a living (10 - 100x speedup)
- Therefore, speed/acc. results should be taken with a large grain of salt


38
36
34
32
30
28
COCO AP
50
100
150
200
250
inference time (ms)
RetinaNet-50
RetinaNet-101
G
FPN Faster R-CNN
F
DSSD513
E
SSD513
R-FCN
D
SSD321
DSSD321
B
C
YOLOv2
AP 22 @ 25ms


G
Improved recipe, software, and hardware (40.2 box AP, 47ms / 21 FPS)
38
36
34
32
30
28
COCO AP
50
100
150
200
250
inference time (ms)
RetinaNet-50
RetinaNet-101
G
FPN Faster R-CNN
F
DSSD513
E
SSD513
R-FCN
D
SSD321
DSSD321
B
C
Points can shift a lot due to implementation details
YOLOv2
AP 22 @ 25ms

Beware of speed vs. accuracy claims

They can be ok, if controlled by

- Making training settings as similar as possible
- Making inference settings as similar as possible
- Ensuring low-level optimization fairness
- Using the same hardware for all methods

Implement all methods in one codebase

- … if possible (not always possible)

- There are so many details that matter in detection
- E.g., COCO AP increases ~1% AP (absolute) going from detectron (v1) to detectron2 (same model)
  - A baseline in detectron (v1) is not a valid comparison to a method in detectron2
- Many good codebases now: mmdetection, detectron2
  - No excuses anymore ;)
  - Use the same codebase to the greatest extent possible

Big tables of historical comparisons: A red flag

Method	data	network	pre-train	Avg. Precision, IoU: 0.5:0.95	0.5	0.75
Faster RCNN [27]	trainval	VGGNet	✓	21.9	42.7	-
ION [1]	train	VGGNet	✓	23.6	43.2	23.6
R-FCN [19]	trainval	ResNet-101	✓	29.2	51.5	-
R-FCN$_{\text{multi-sc}}$ [19]	trainval	ResNet-101	✓	29.9	51.9	-
SSD300 (Huang et al.) [11]	< trainval35k	MobileNet	✓	18.8	-	-
SSD300 (Huang et al.) [11]	< trainval35k	Inception-v2	✓	21.6	-	-
YOLOv2 [26]	trainval35k	Darknet-19	✓	21.6	44.0	19.2
SSD300* [21]	trainval35k	VGGNet	✓	25.1	43.1	25.8
DSOD300	trainval	DS/64-192-48-1	✗	29.3	47.3	30.6

Claim: Faster R-CNN (VGG-16 backbone) has 21.9 AP on COCO

Implication: all other methods in the table outperform it

But with today's best practices…

Faster R-CNN (VGG-16 backbone): 35.6 AP

(and 35.2 AP without pre-training)

The field evolves rapidly

- “Old” papers (where old can mean, e.g., less than 10 years even) may contain seemingly weird details or bad practices
  - E.g., Why does the original R-CNN paper use SVMs?
  - E.g., Faster R-CNN was originally trained in four stages (not end-to-end)
- These are artifacts of what was *possible* at the time
  - Compute may have been limited
  - Knowledge was limited
- Never view something in a paper as “the one way” to do something
  - It may be an implementation detail that can and has changed

Recap: Tips about reading detection papers

- It’s really hard, unless you have a lot of experience!
- There are many bad papers out there that may mislead you
  - Uncontrolled comparisons to old results
  - Uncontrolled speed vs. accuracy comparisons
  - Complex methods without careful ablations
  - …
- These tips can guide you what to look for when reading a detection paper