Instance and panoptic segmentation

Juan David García and Jessica Castillo Rojas.

Advisors: Stephannie Jiménez and Laura Bravo.

Outline.

- Introduction.
- Brief review of semantic segmentation.
- Instance Segmentation.

  - Mask R-CNN.
- Panoptic Segmentation.
- Papers.

  - Panoptic Feature Pyramid Networks.

  - UPSNet.
- Tutorial.
- Homework.

Introduction

Classic computer vision problems

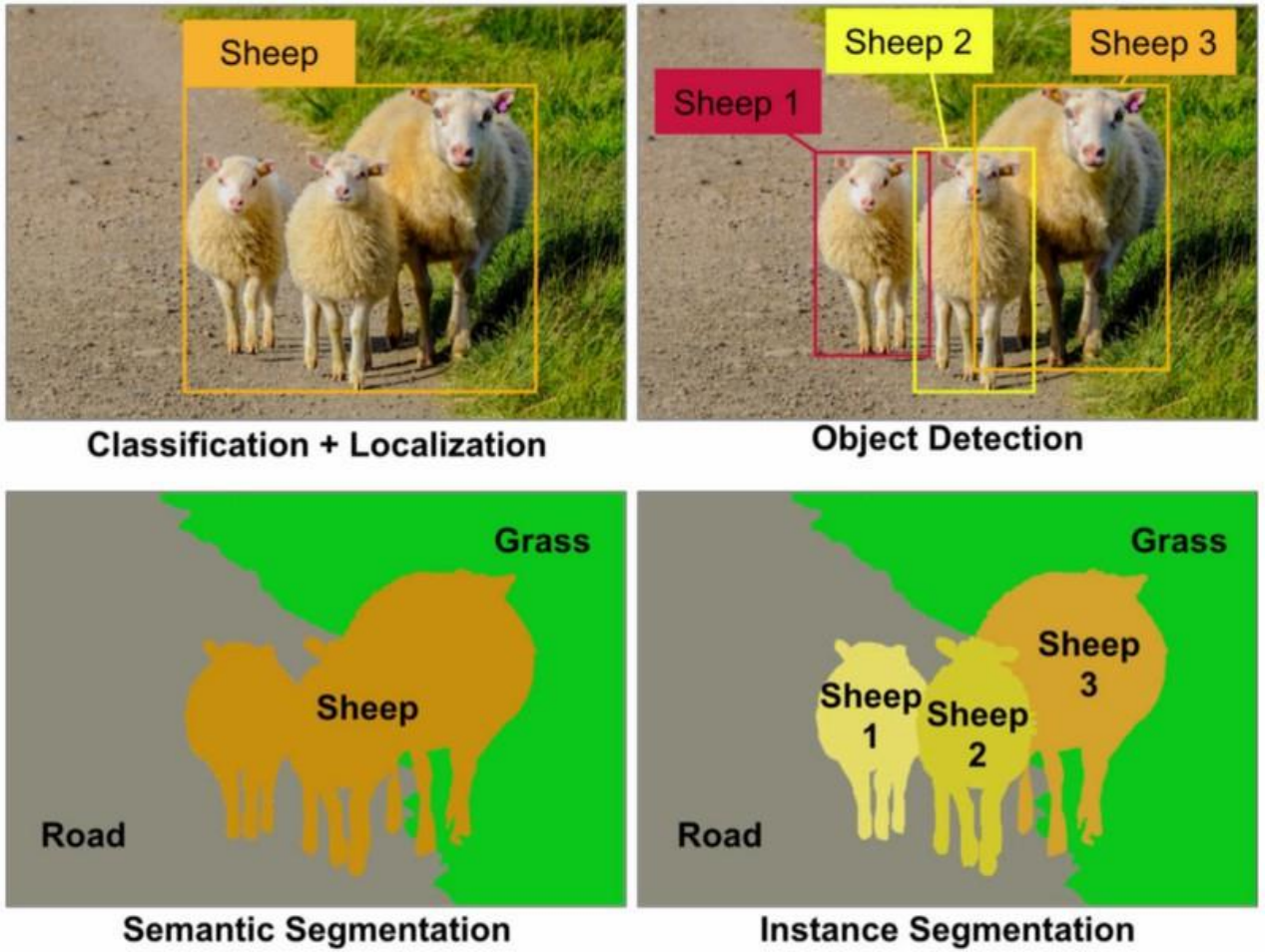
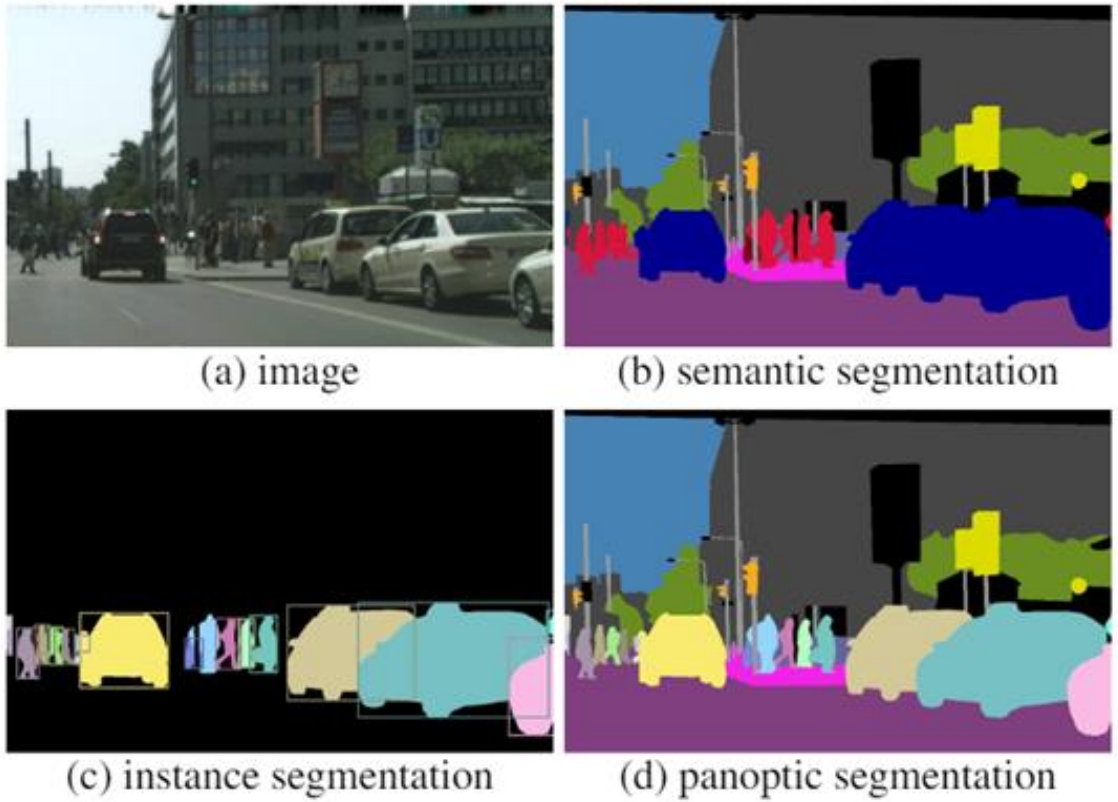


Image taken from: https://towardsdatascience.com/detection-and-segmentation-through-convnets-47aa42de27ea

Panoptic segmentation



The Task

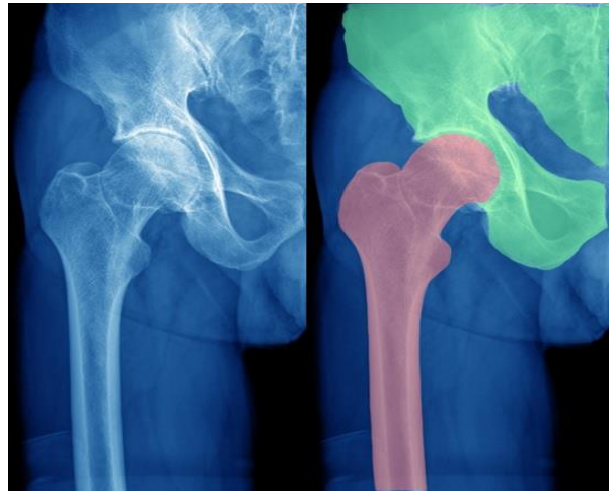
Scene parsing.
Also known as image parsing or holistic scene understanding.
Join instance segmentation of thing classes with semantic segmentation of stuff classes.

Kirillov, A., He, K., Girshick, R., Rother, C. y Dollár, P. (2018). Panoptic Segmentation. CoRR, abs/1801.00868.

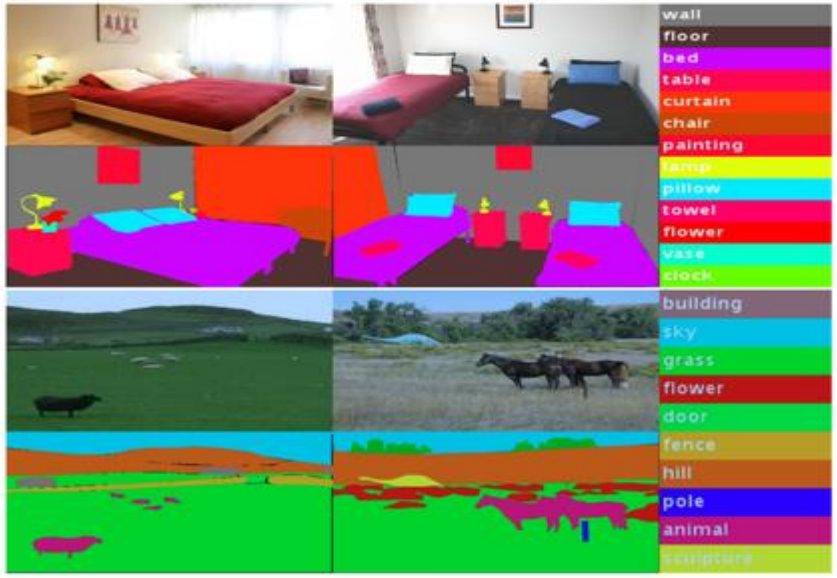
Some applications of image segmentation

Autonomous driving.

Medical image segmentation.



Scene understanding.



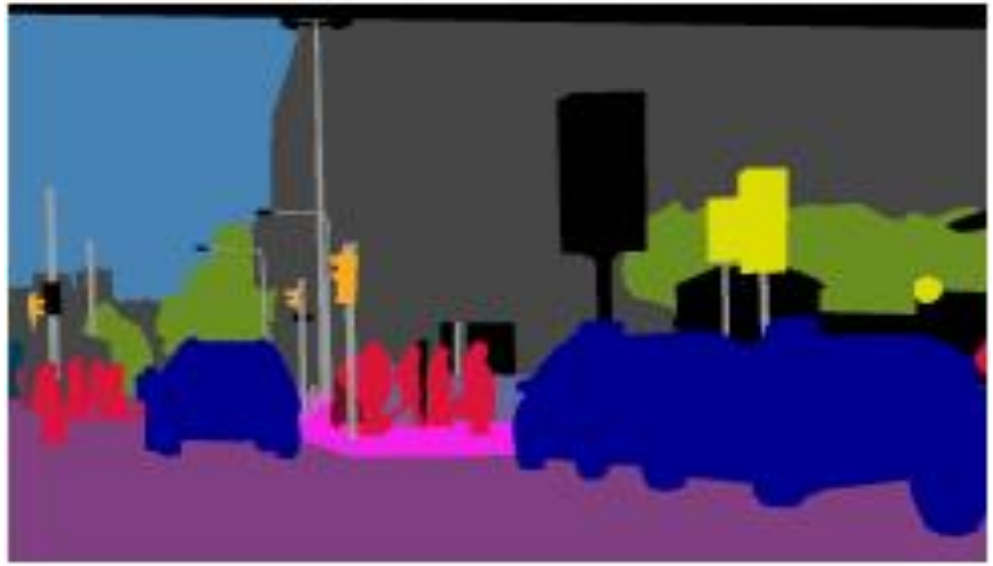
Brief review of Semantic segmentation

Semantic segmentation

What is semantic segmentation?

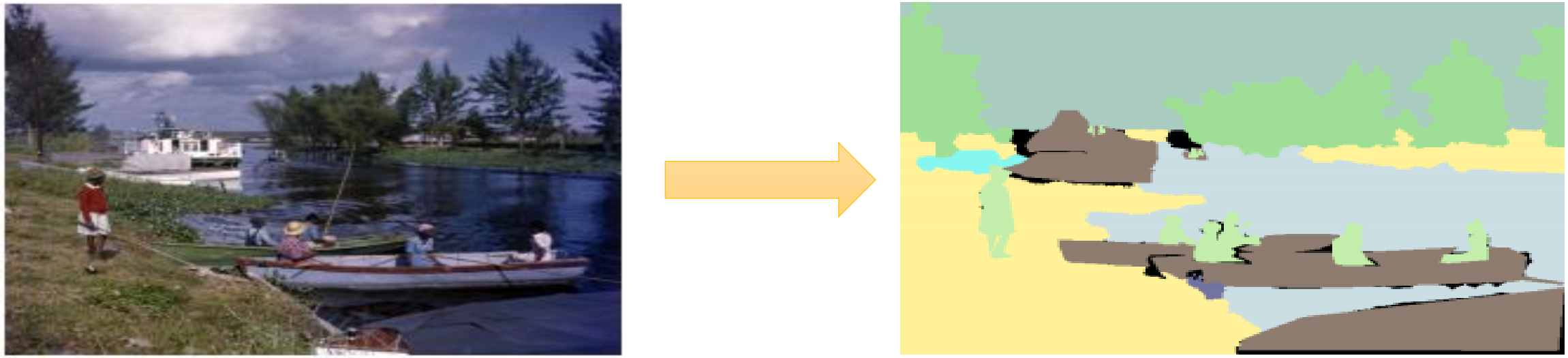
Given an image, the goal is to assign a class-label or a category to each one of the pixels of the image.

Image



Prediction (each pixel with a class-label)

Semantic Segmentation

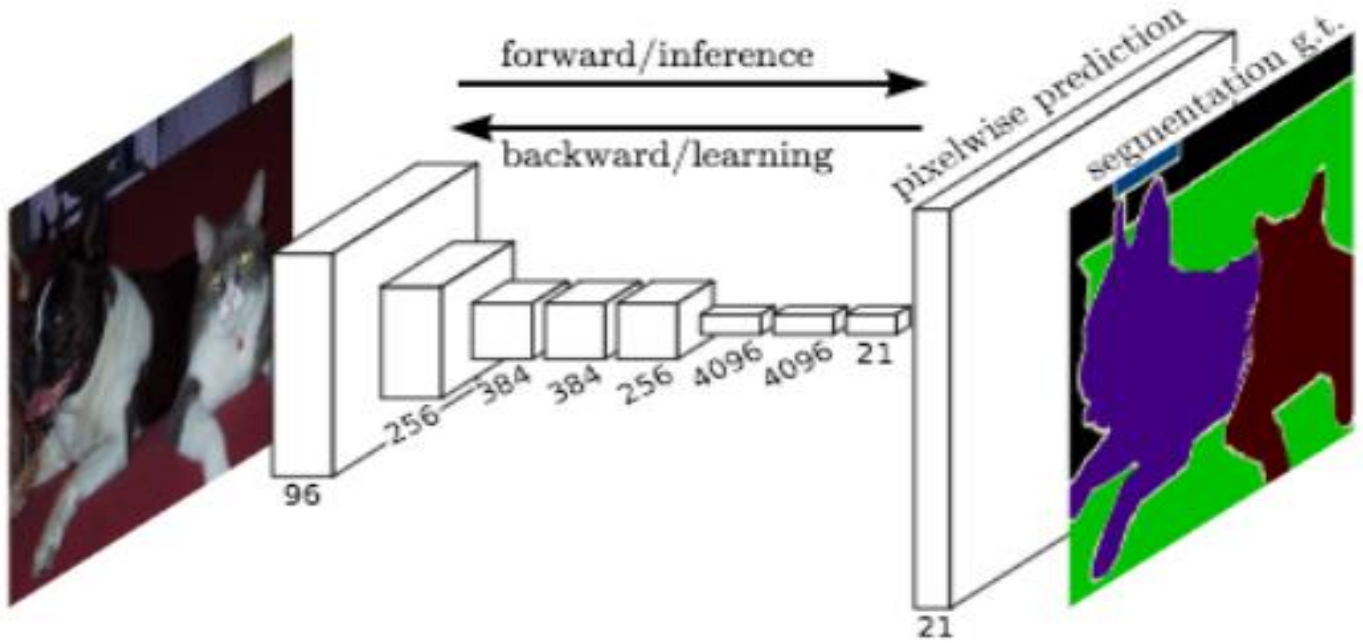


- Simple accuracy measure (IoU).
- Do not differentiate instances only care about pixels.
- “Thing classes are treated as stuff”.

Brief review of two **important** semantic segmentation methods

Fully convolutional Networks (FCN)

Baseline for other semantic segmentation methods.

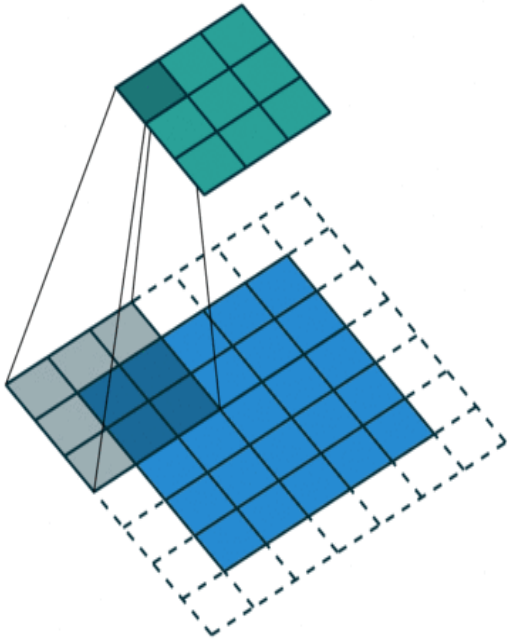


Architecture of the FCN

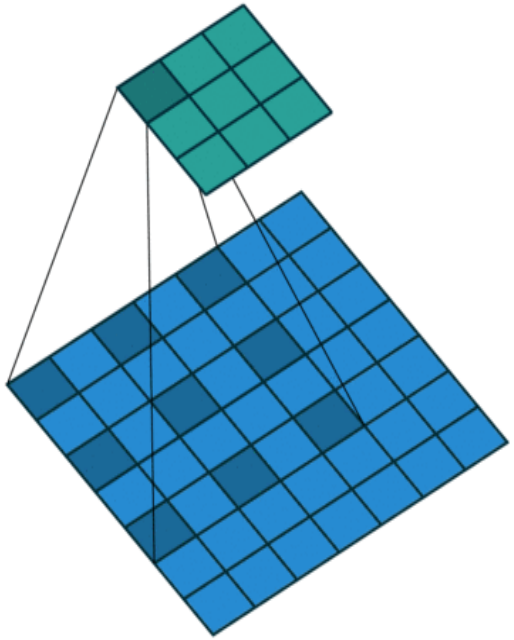
Long, J. et.al (2015). Fully convolutional networks for semantic segmentation

DeepLab

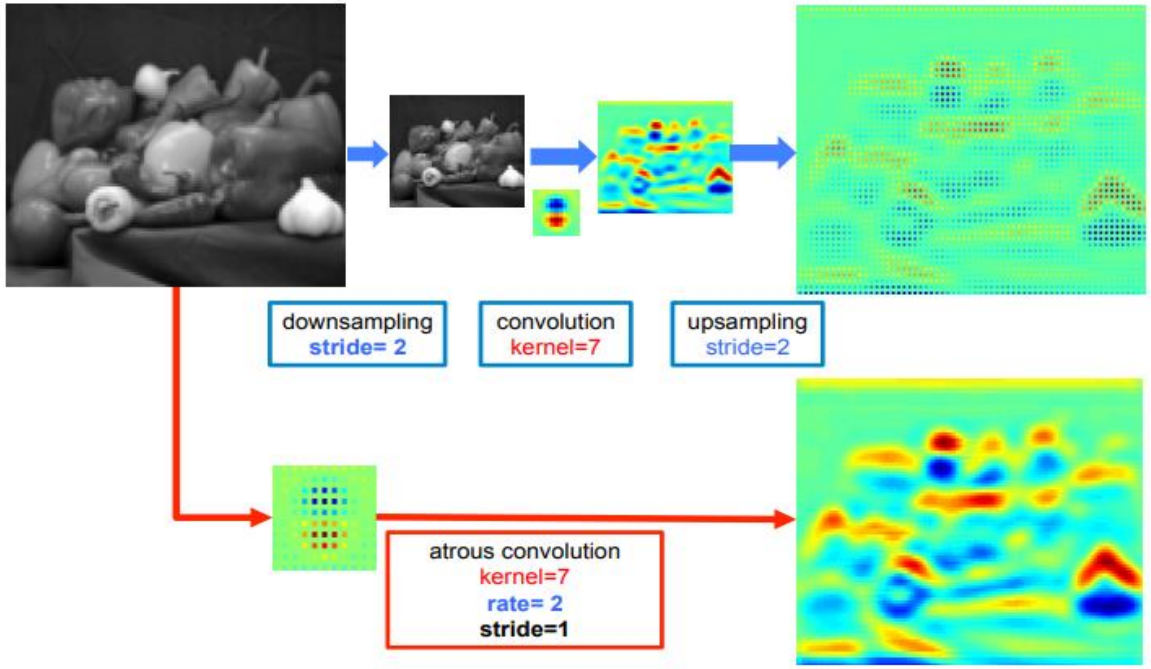
One of the main contribution is the **dilated** or **atrous** convolution



Normal convolution r = 1



Dilated convolution r = 2

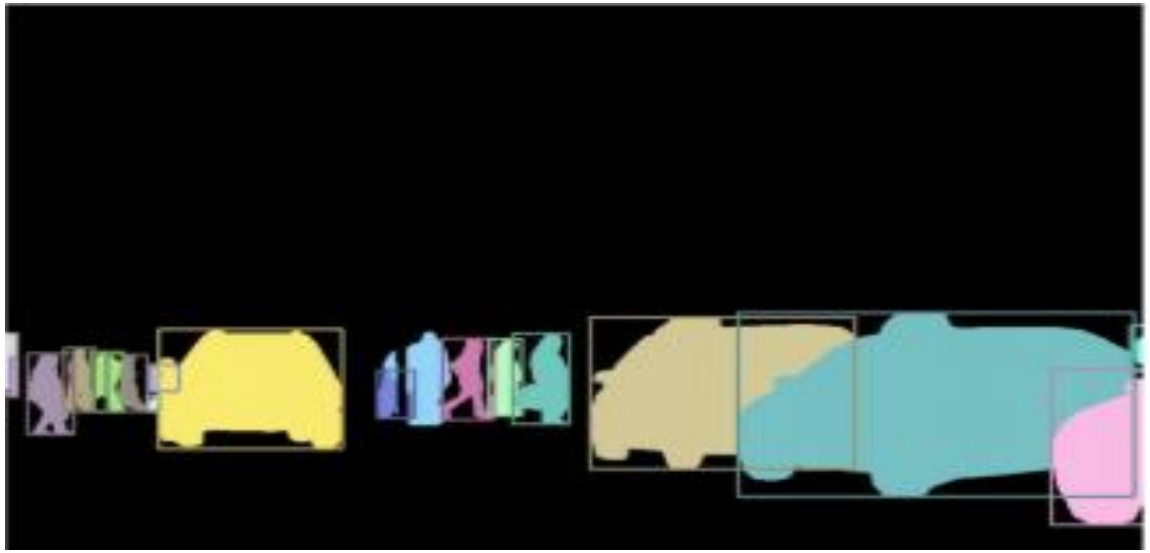


Atrous convolution in 2D

$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k] w[k].$$

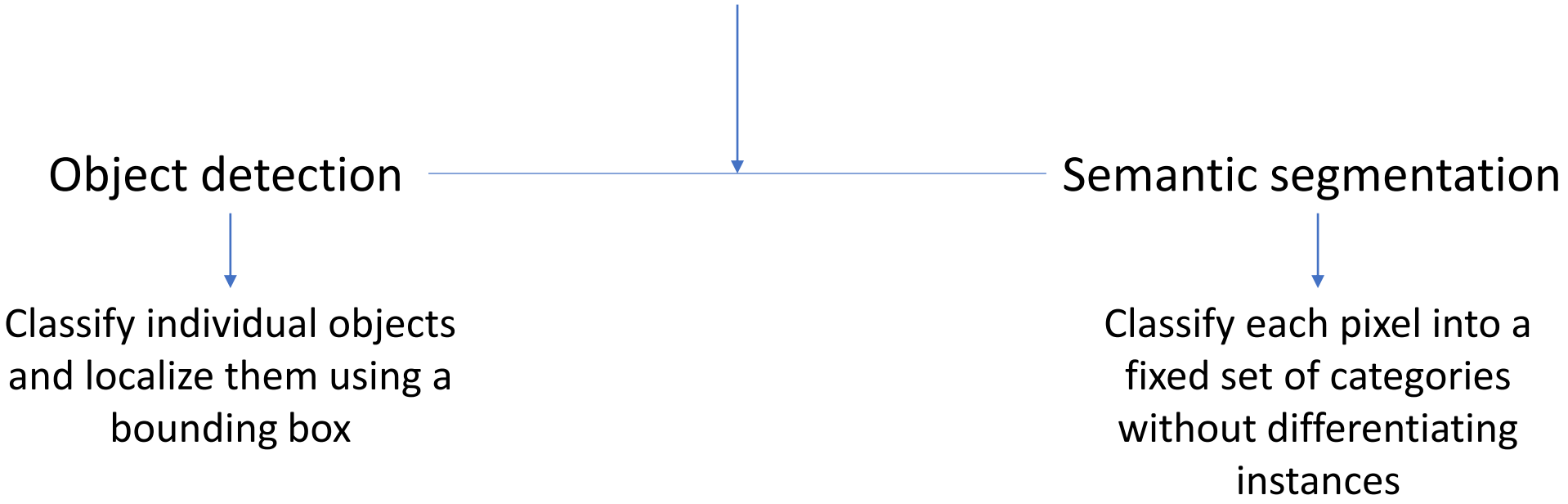
Chen et.al(2017) DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs .

Instance segmentation



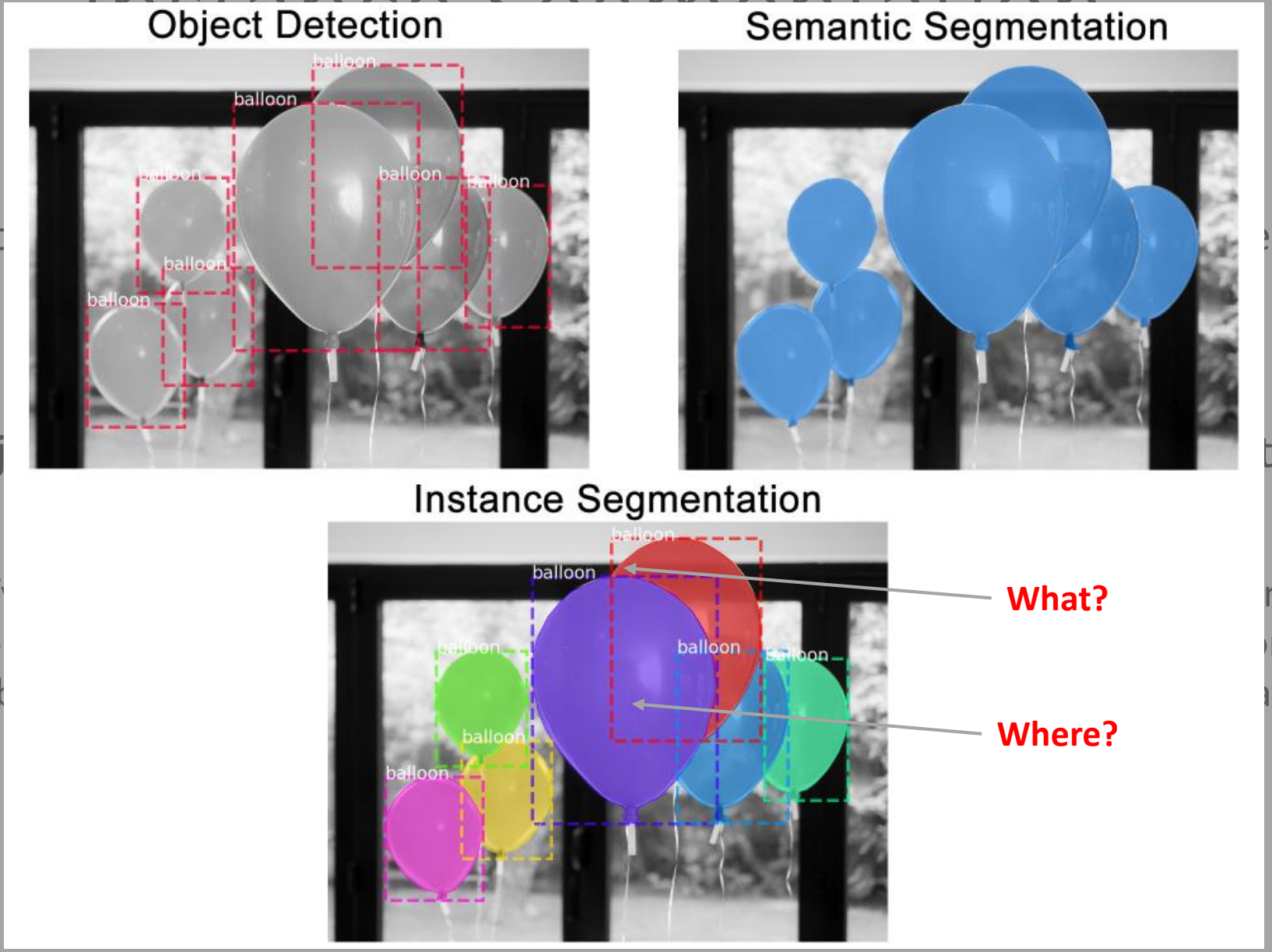
Instance Segmentation

Detection of all objects in the image **and** precisely segment each instance.



He, K., Gkioxari, G., Dollár, P., y Girshick, R. (2017). Mask R-CNN.

Instance Segmentation



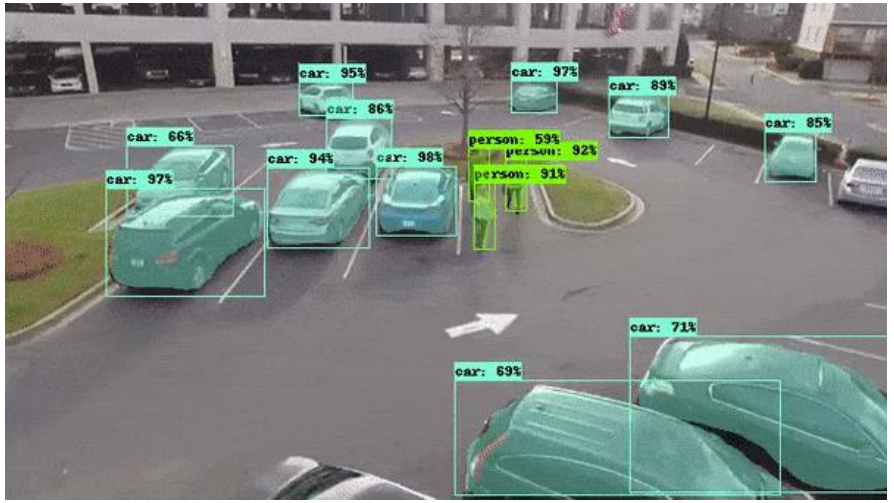
He, K., Gkioxari, G., Dollár, P., y Girshick, R. (2017). Mask R-CNN.

Instance Segmentation

- If we compare **instance** vs **semantic** segmentation, the first task is more challenging because it requires **the correct detection** of all objects while also precisely **segmenting each instance**.

- In **semantic segmentation** we only care about assign a class-label to each pixel which is **easier**.

Semantic segmentation

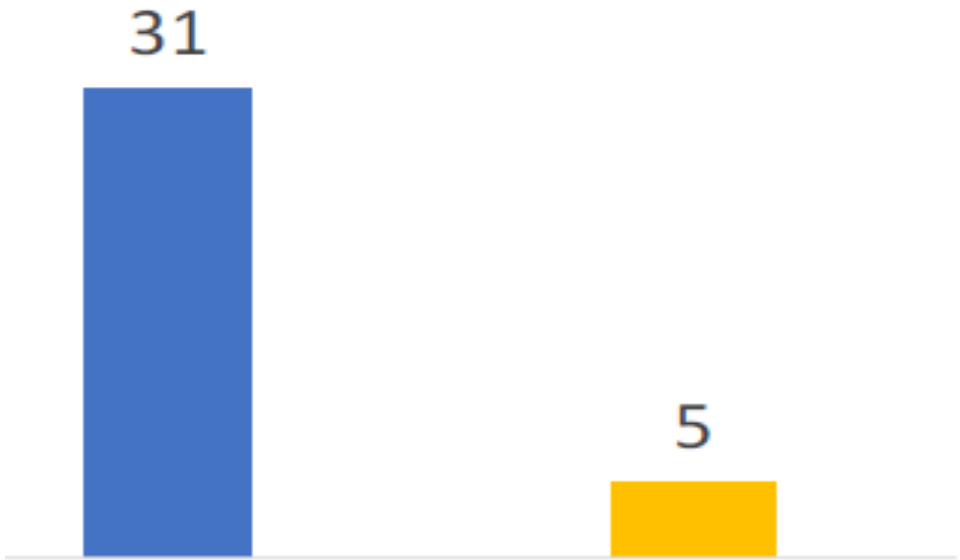


Instance segmentation

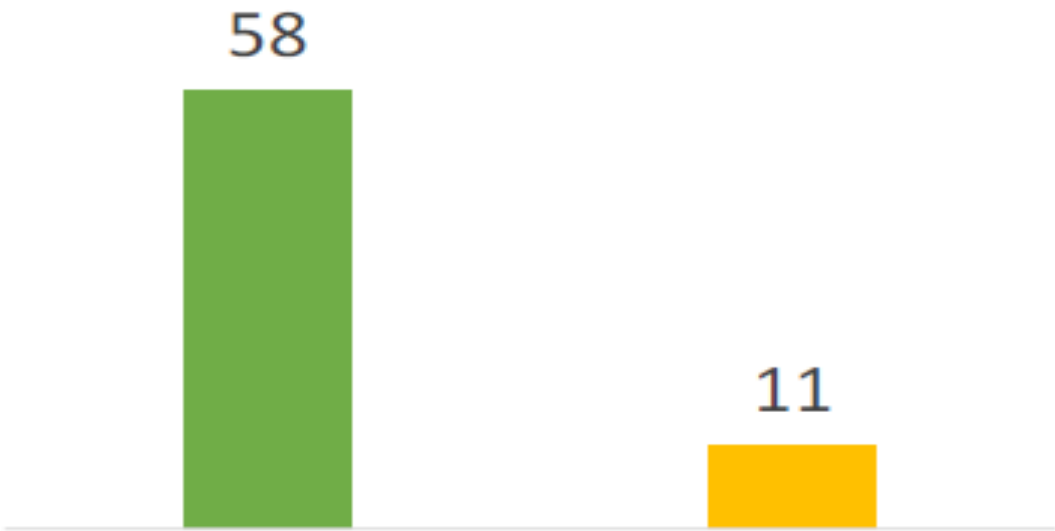
Instance Segmentation

A **very** challenging problem

Number of entries on COCO leaderboard



Number of entries on Cityscapes leaderboard

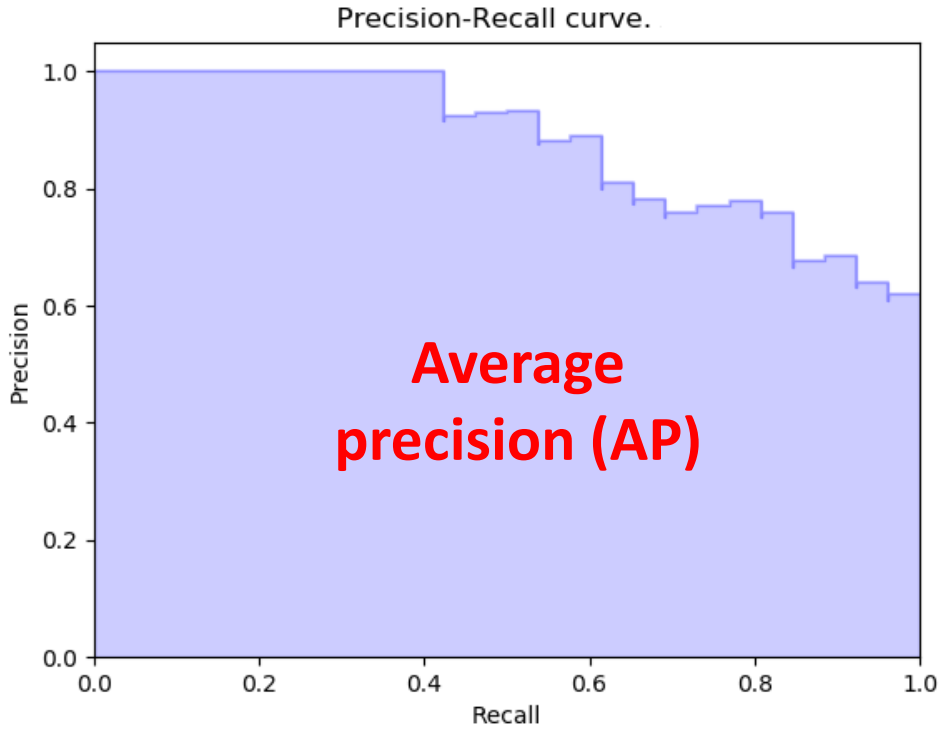


Metrics

$$Precision = \frac{TP}{TP + FP}$$

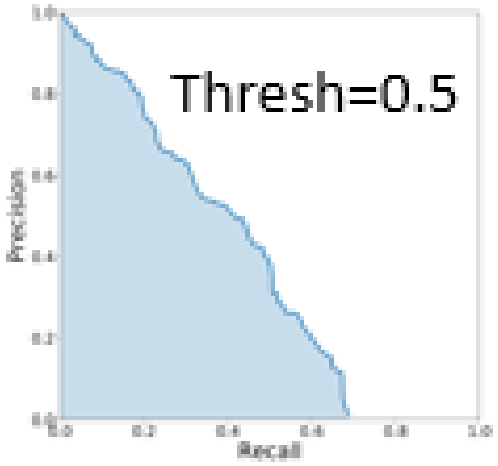
$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2\,\frac{Precision \cdot Recall}{Precision + Recall}$$

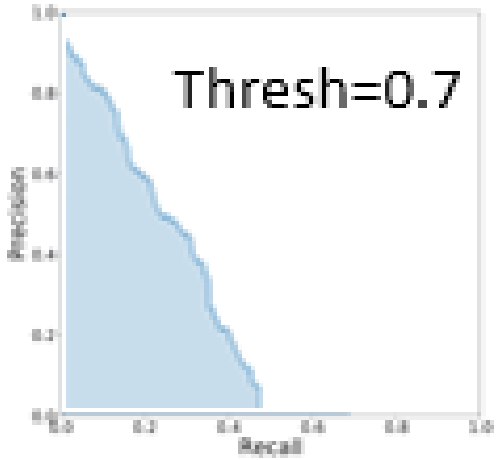


Scikit-learn. Precision-Recall. https://bit.ly/2JzHAGs

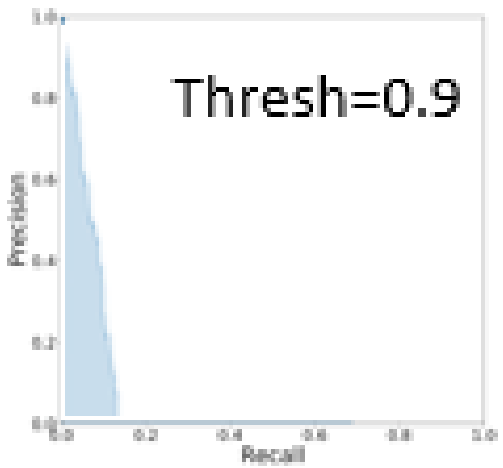
Metrics



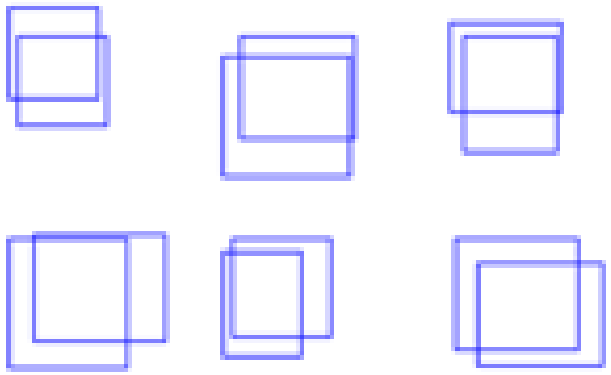
...



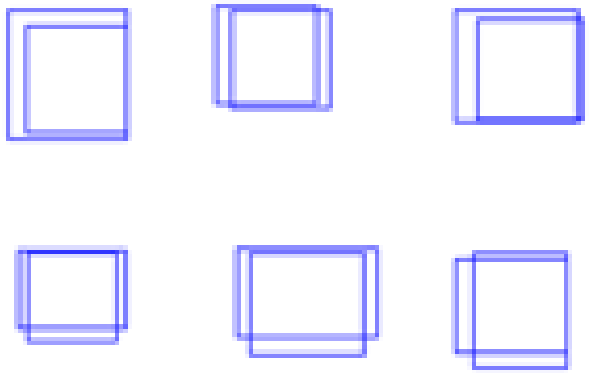
...



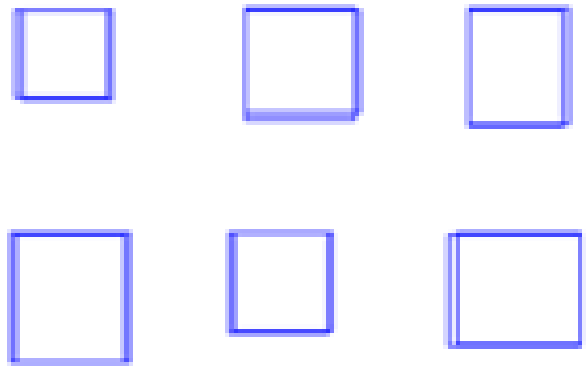
...



IoU = 0.5



IoU = 0.7



IoU = 0.9

Loose

Tight

Scikit-learn. Precision-Recall. https://bit.ly/2JzHAGs

COCO Metrics

Average Precision (AP):

- $AP \rightarrow AP\ at\ IoU = 0.50 : 0.05 : 0.95$ **(primary challenge metric)**
- $AP^{IoU=0.5}\ (AP_{50}) \rightarrow AP\ at\ IoU = 0.50$ (PASCAL VOC metric)
- $AP^{IoU=0.75}\ (AP_{75}) \rightarrow AP\ at\ IoU = 0.75$ (strict metric)

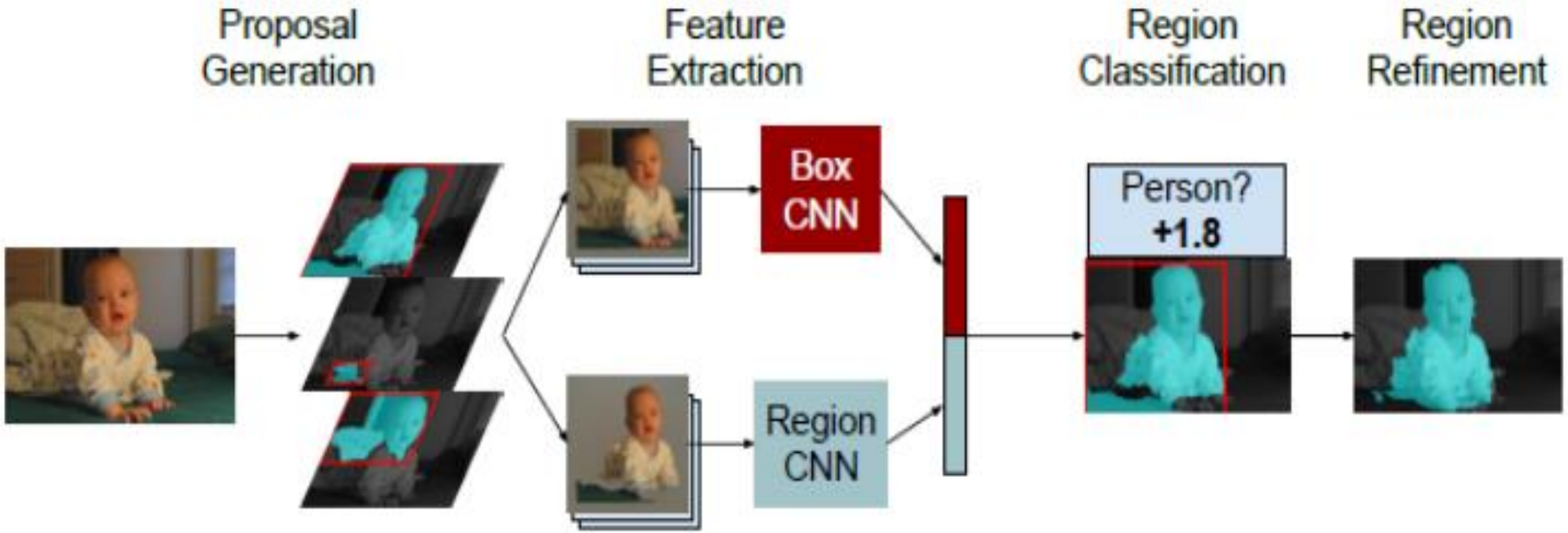
AP Across Scales:

- $AP^{small}\ (AP_S) \rightarrow AP\ for\ small\ objects$
- $AP^{medium}\ (AP_M) \rightarrow AP\ for\ medium\ objects$
- $AP^{large}\ (AP_L) \rightarrow AP\ for\ large\ objects$

COCO. *Detection Evaluation.* http://cocodataset.org/#detection-eval

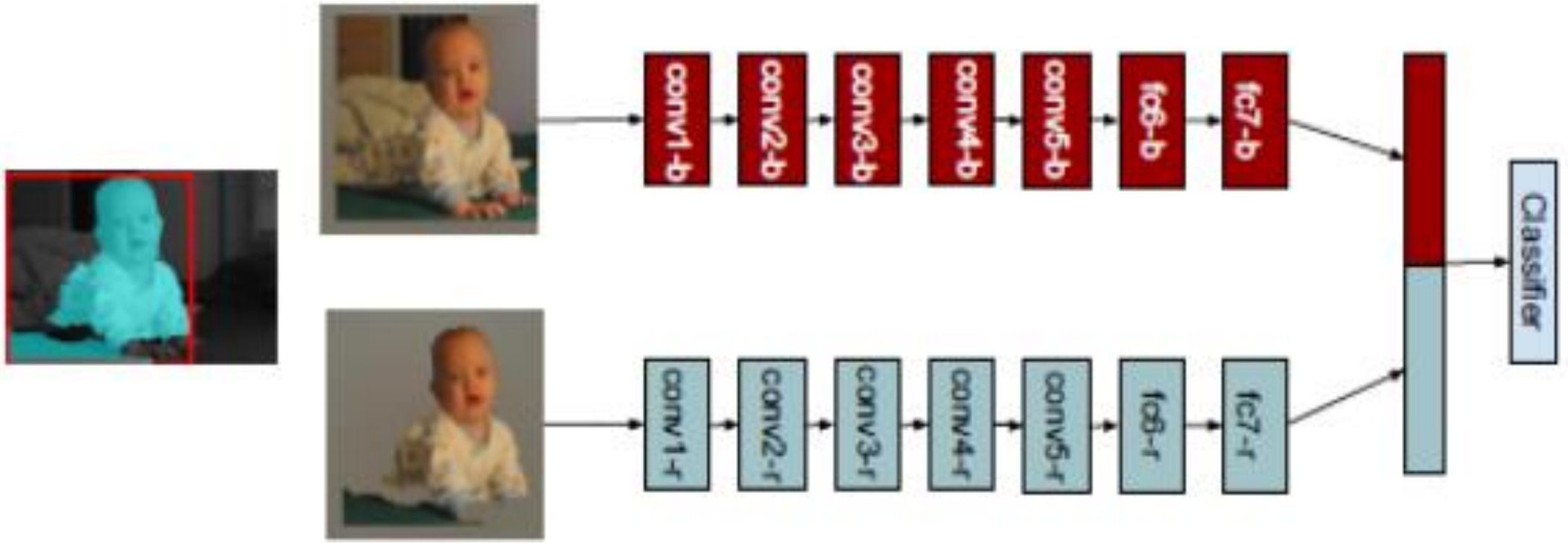
First approaches of instance segmentation using CNN

Instance Segmentation Methods- Simultaneous Detection and Segmentation (2014).



Hariharan et.al (2014) Simultaneous detection and segmentation.

Instance Segmentation Methods - Simultaneous Detection and Segmentation (2014).



Hariharan et.al (2014) Simultaneous detection and segmentation.

Instance Segmentation Methods- Simultaneous Detection and Segmentation (2014).

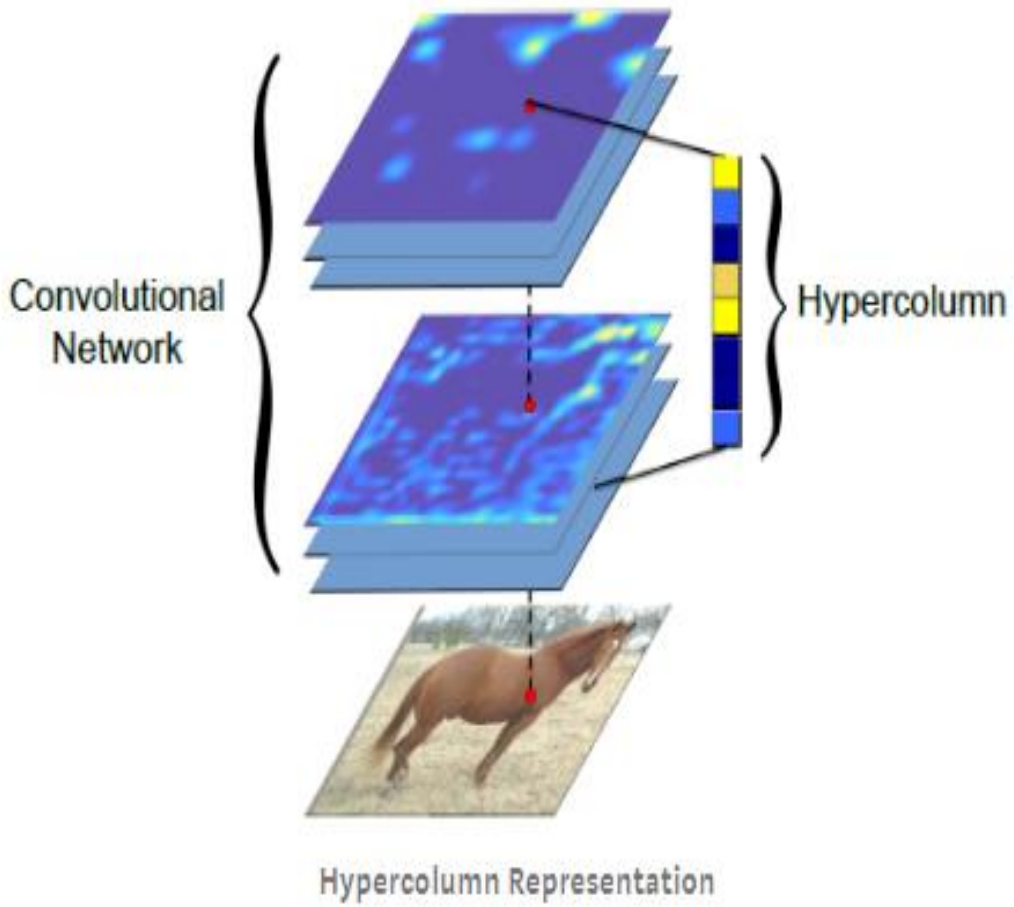
Hariharan et.al (2014) Simultaneous detection and segmentation.

Instance Segmentation Methods- Simultaneous Detection and Segmentation (2014).

	R-CNN[16]	R-CNN-MCG	A	B	C
Mean AP^b	51.0	51.7	51.9	**53.9**	53.0
Mean AP^b_{vol}	41.9	42.4	43.2	**44.6**	44.2

Hariharan et.al (2014) Simultaneous detection and segmentation.

Instance Segmentation Methods- Hypercolumn (2015)

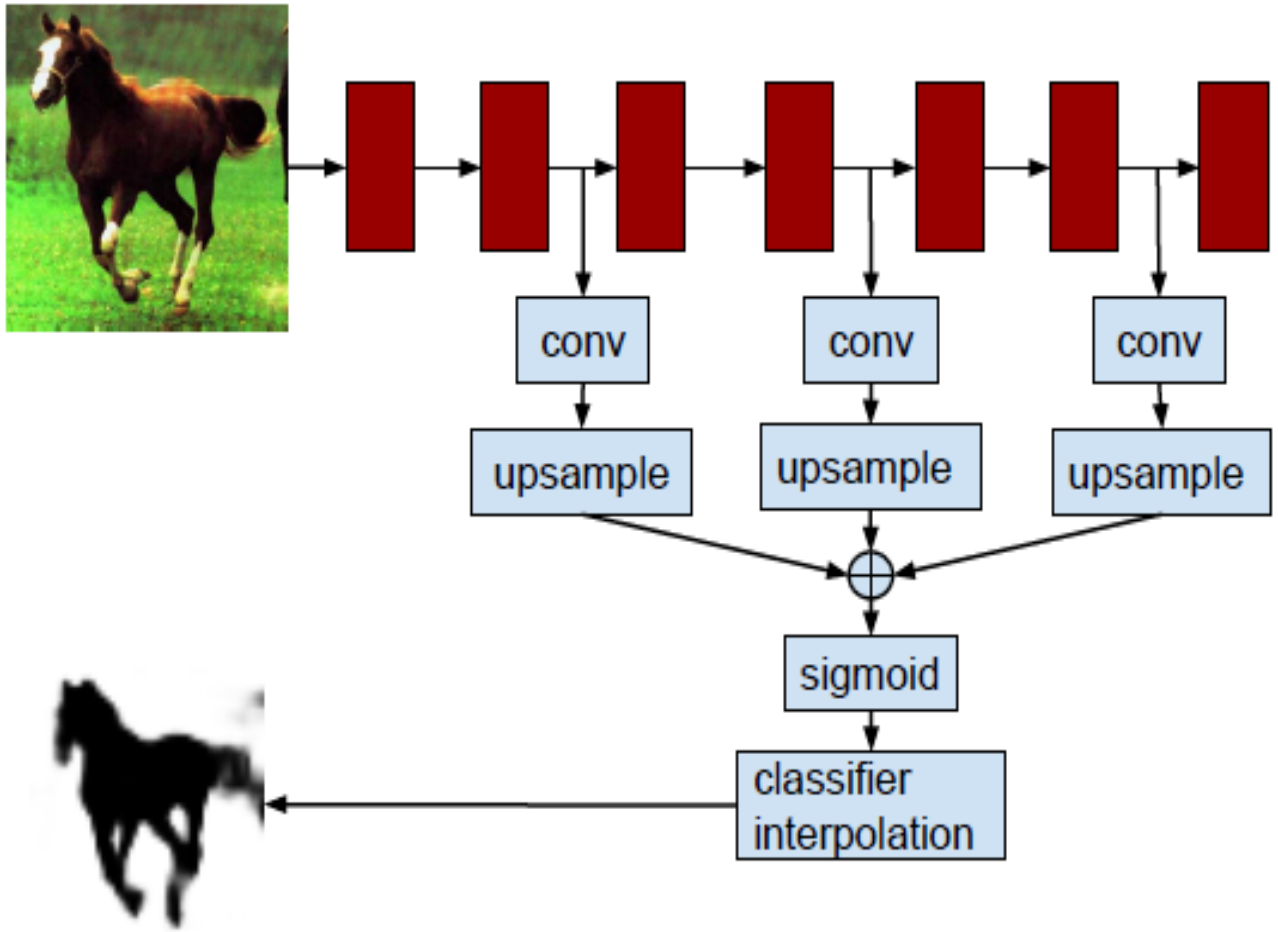


Key points

- Vector of activations
- Spatial locations
- Small objects

Hariharan et.al (2014) Hypercolumns for object segmentation and fine-grained localization

Instance Segmentation Methods- Hypercolumn (2015).



Problem setting

- Detections after non-maximum suppression
- Heat map prediction
- Adjacent pixels should be similar to each other.

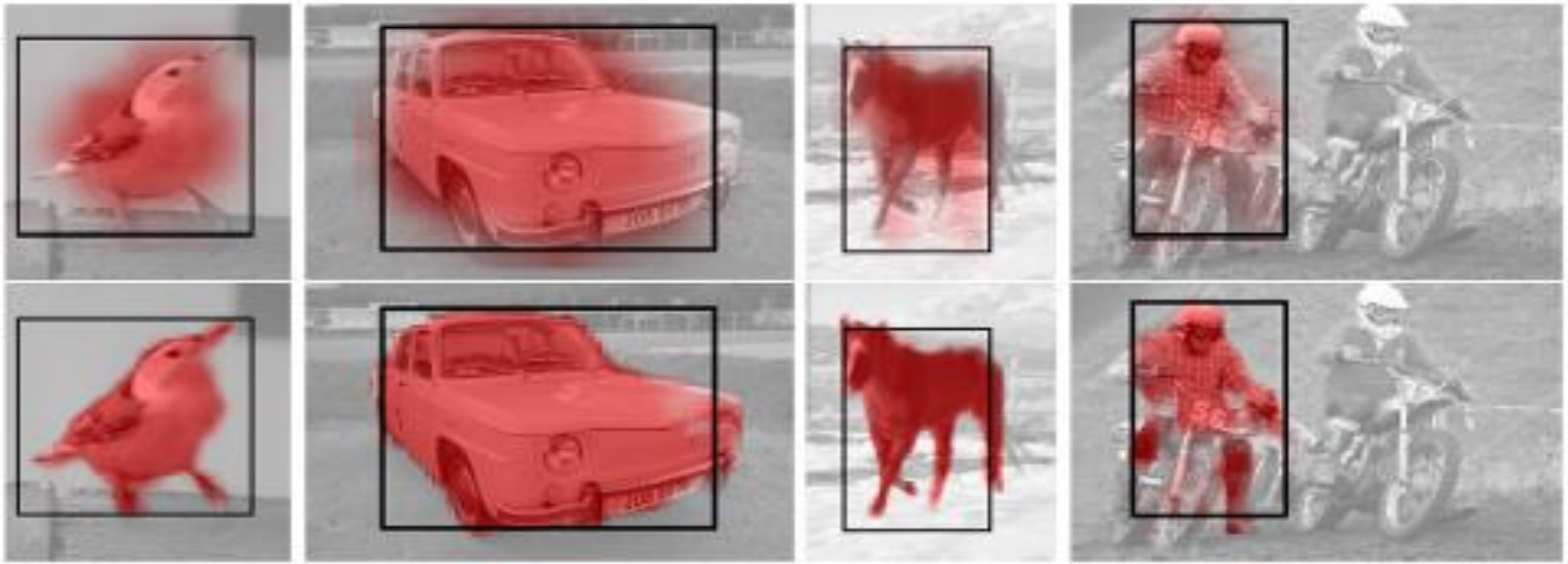
Hariharan et.al (2014) Hypercolumns for object segmentation and fine-grained localization

Instance Segmentation Methods- Hypercolumn (2015).

Metric	[22]	[22] refined	Hyp	Hyp +bbox-reg	Hyp+FT +bbox-reg	Only *fc7*	*fc7*+ *pool2*	*fc7*+ *conv4*	*pool2*+ *conv4*	1×1 grid	2×2 grid	5×5 grid
mean AP^r at 0.5	47.7	49.7	51.2	51.9	**52.8**	49.7	50.5	51.0	50.7	50.3	51.2	51.3
mean AP^r at 0.7	22.8	25.3	31.6	32.4	**33.7**	25.8	30.6	31.2	30.8	28.8	30.2	31.8

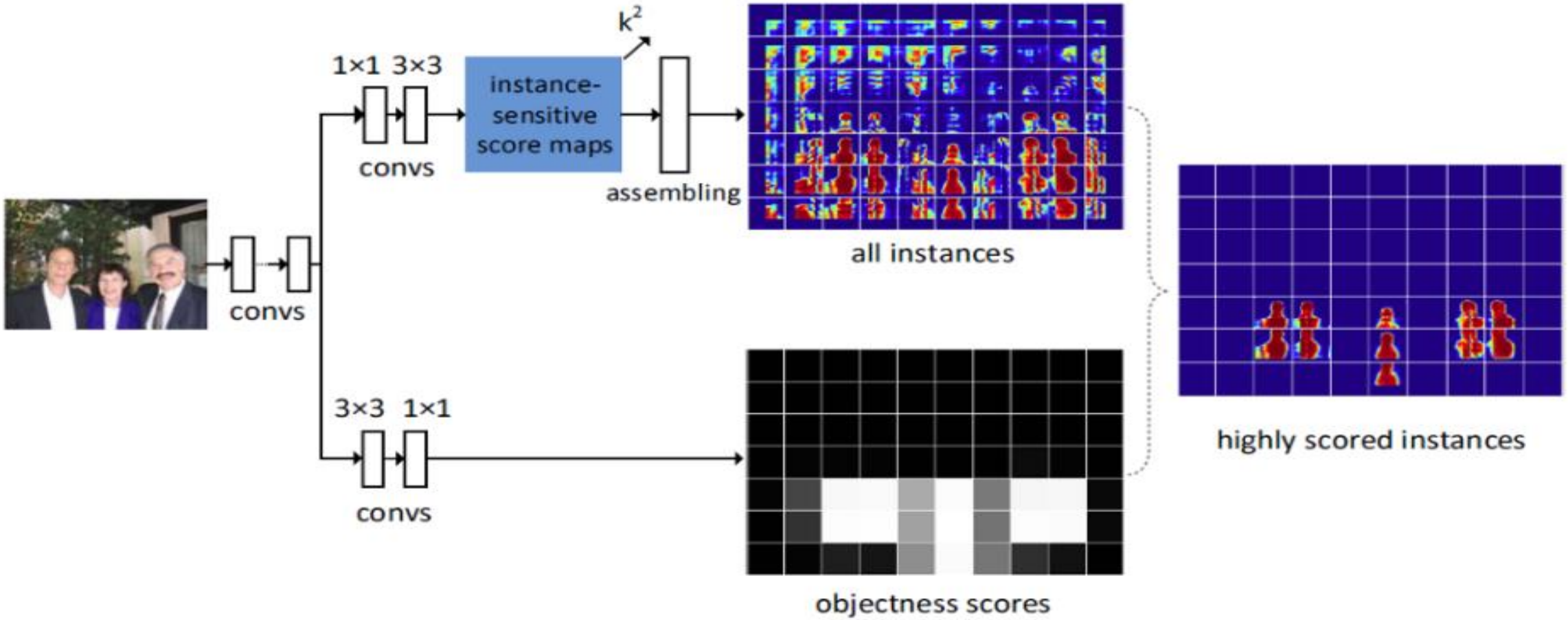
Hariharan et.al (2014) Hypercolumns for object segmentation and fine-grained localization

Instance Segmentation Methods- Hypercolumn (2015).



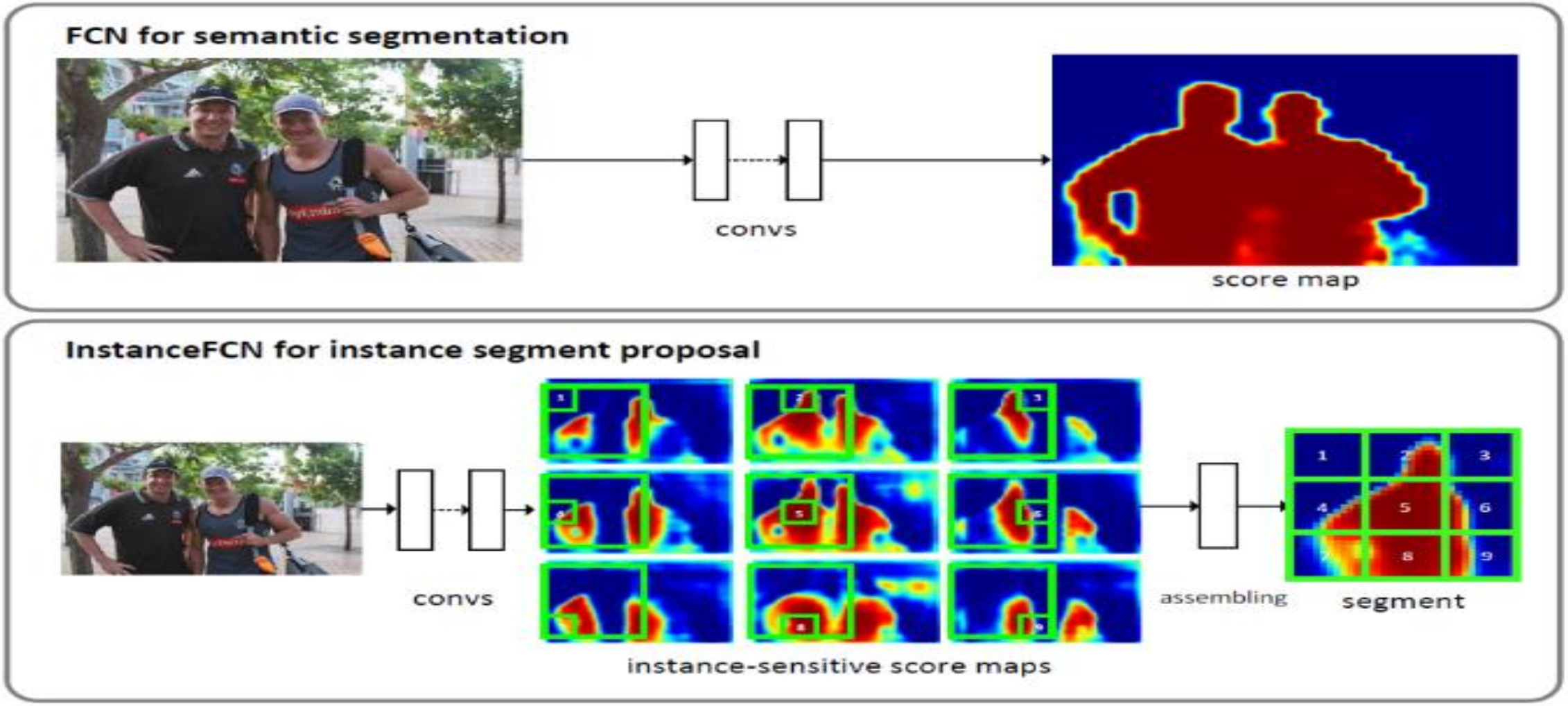
Hariharan et.al (2014) Hypercolumns for object segmentation and fine-grained localization

Instance Segmentation Methods - InstanceFCN (2016)



Dai et.al (2016) *Instance-sensitive Fully Convolutional Networks*

Instance Segmentation Methods - InstanceFCN (2016)

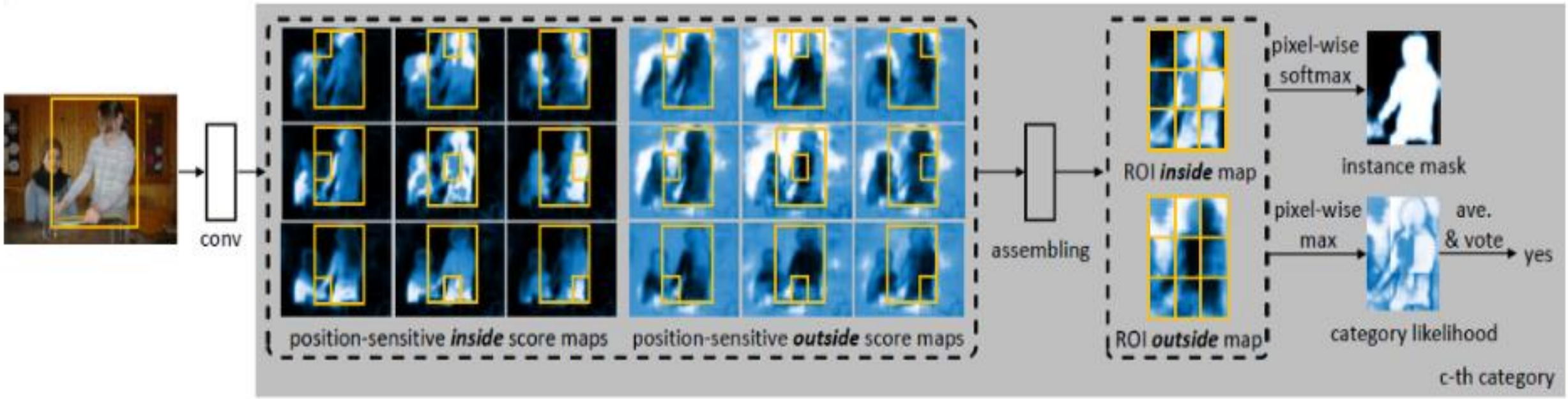


Dai et.al (2016) *Instance-sensitive Fully Convolutional Networks*

Instance Segmentation Methods - InstanceFCN (2016)

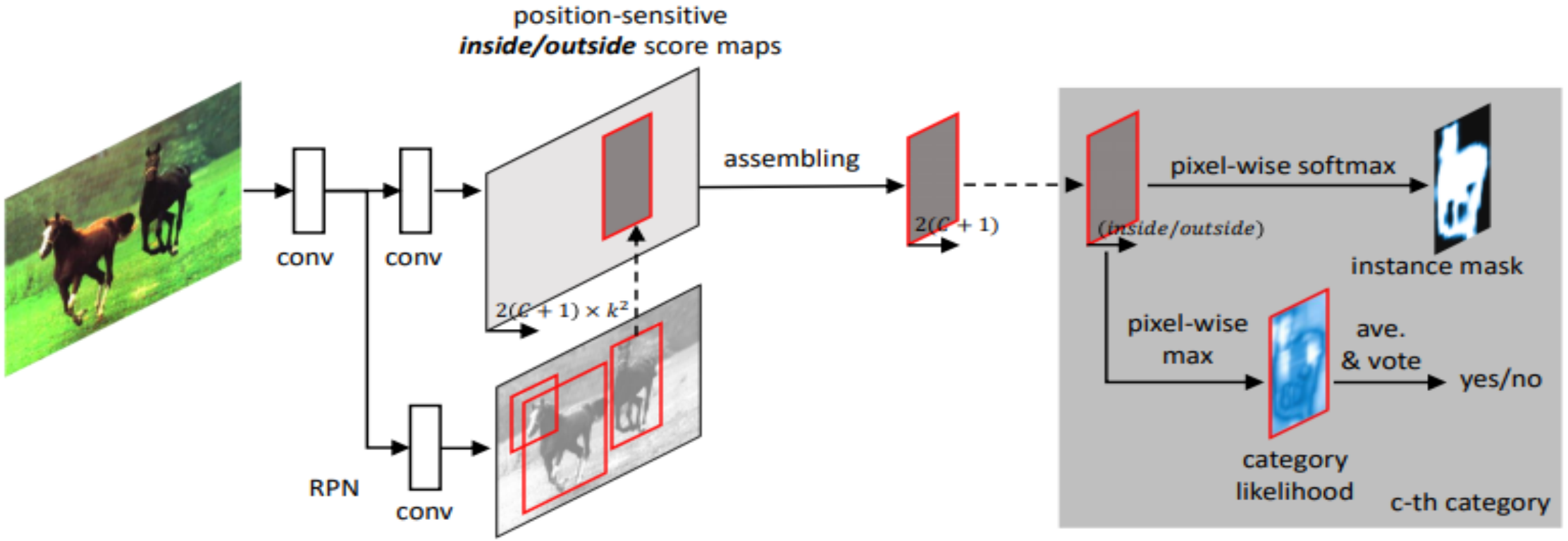
Dai et.al (2016) *Instance-sensitive Fully Convolutional Networks*

Instance Segmentation Methods - Fully Convolutional Instance-aware Semantic Segmentation (FCIS) (2017)



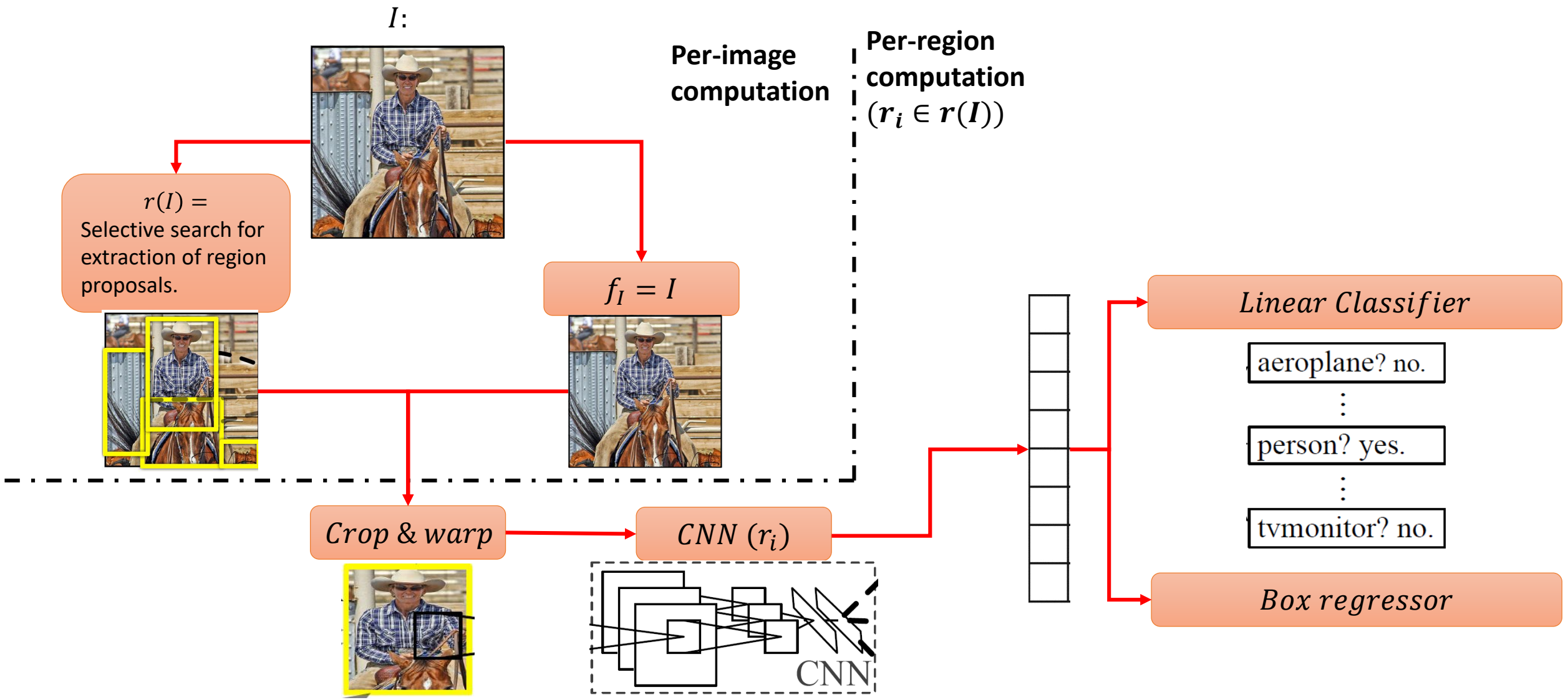
Li.Y et.al (2017) Fully Convolutional Instance-aware Semantic Segmentation (FCIS) .

Instance Segmentation Methods - Fully Convolutional Instance-aware Semantic Segmentation (FCIS) (2017)



Li.Y et.al (2017) Fully Convolutional Instance-aware Semantic Segmentation (FCIS) .

R-CNN : Region-based Convolutional Neuronal Networks for **object detection** (2014)

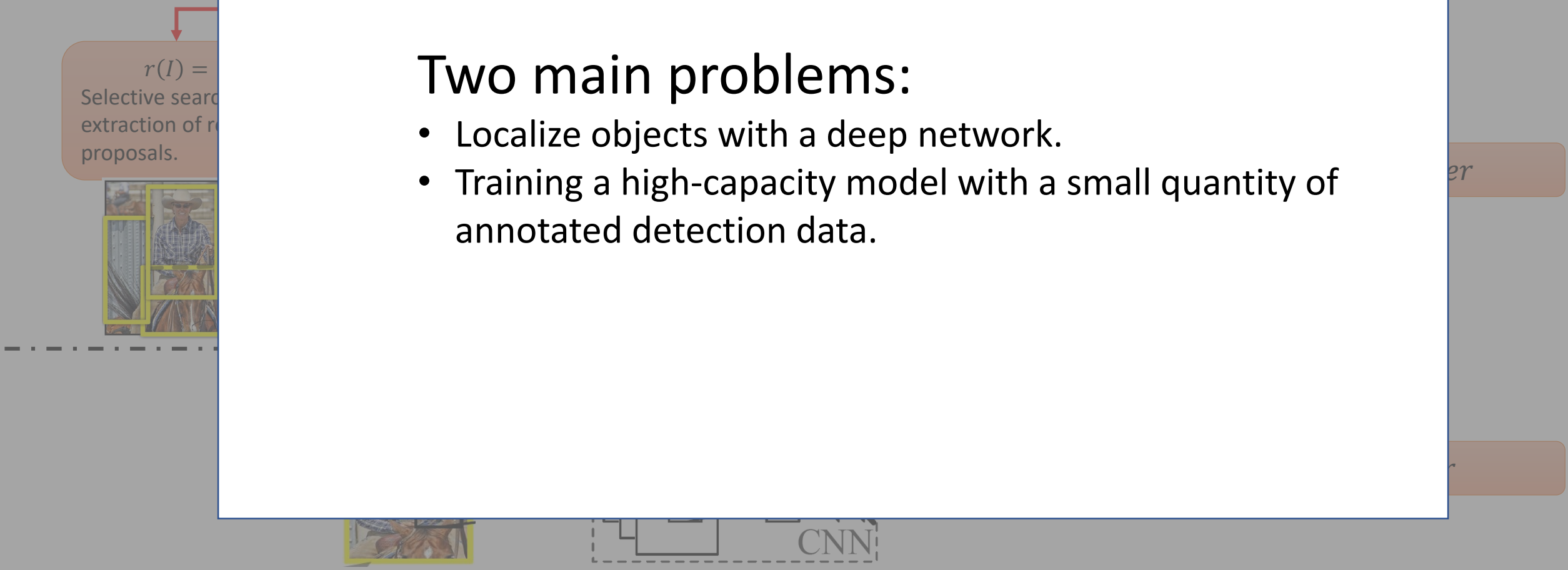


Girshick, R. et.al (2014) Rich feature hierarchies for accurate object detection and semantic segmentation.

R-CNN : Region-based Convolutional Neuronal Networks for **object detection** (2014)

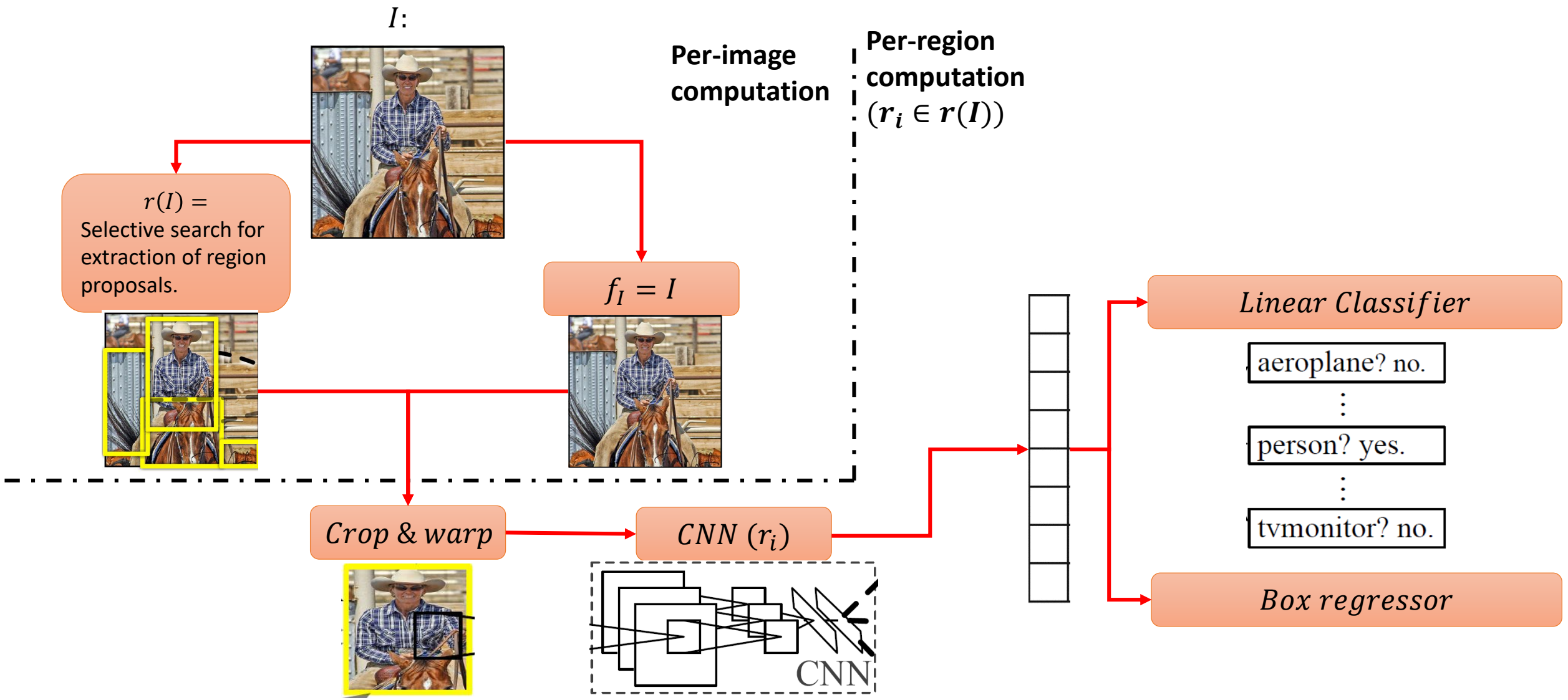
Two main problems:

- Localize objects with a deep network.
- Training a high-capacity model with a small quantity of annotated detection data.



Girshick, R. et.al (2014) Rich feature hierarchies for accurate object detection and semantic segmentation.

R-CNN : Region-based Convolutional Neuronal Networks for **object detection** (2014)



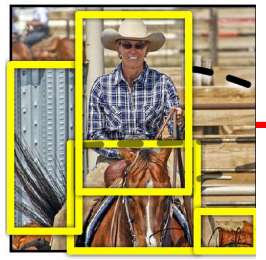
Girshick, R. et.al (2014) Rich feature hierarchies for accurate object detection and semantic segmentation.

R-CNN : Region-based Convolutional Neuronal Networks for **object dtection** (2014)

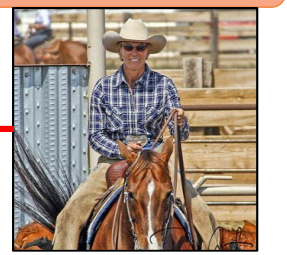
Per-image computation

I:

$r(I) =$
Selective search for extraction of region proposals.



$f_I = I$



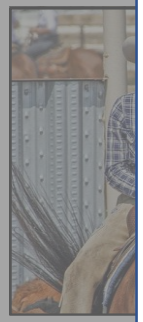
$Crop\ \&\ warp$

CNN

Girshick et.al (2014) Rich feature hierarchies for accurate object detection and semantic segmentation

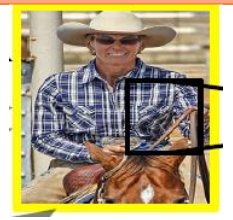
R-CNN : Region-based Convolutional Neuronal Networks for **object dtection** (2014)

Per-region computation
$(\boldsymbol{r_i} \in \boldsymbol{r}(\boldsymbol{I}))$

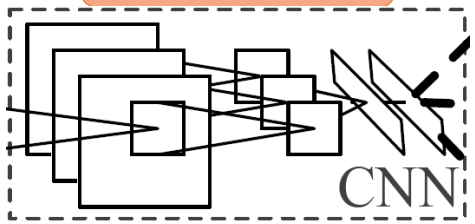


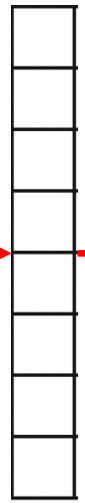
Selective search fo
proposals.

Crop & warp



CNN (r_i)





Linear Classifier

aeroplane? no.

⋮

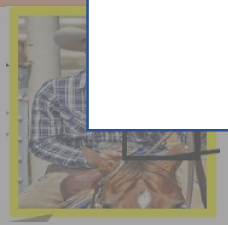
person? yes.

⋮

tvmonitor? no.

Box regressor

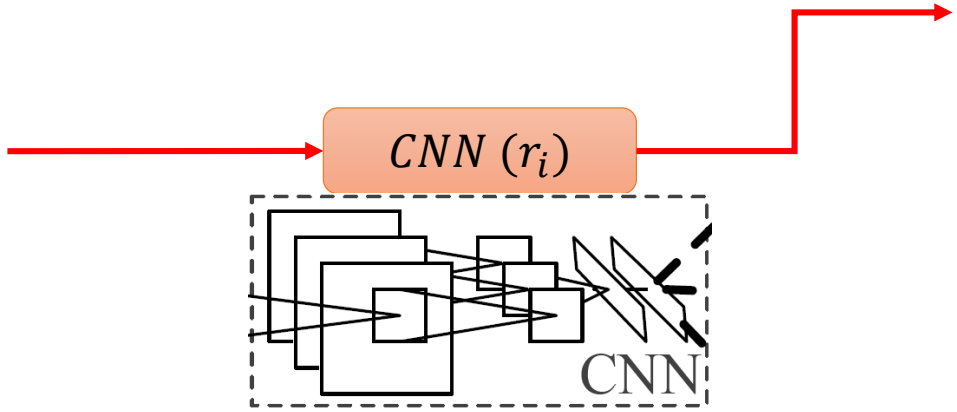
Crop



Girshick et.al (2014) Rich feature hierarchies for accurate object detection and semantic segmentation

R-CNN : Region-based Convolutional Neuronal Networks for **object detection** (2014)

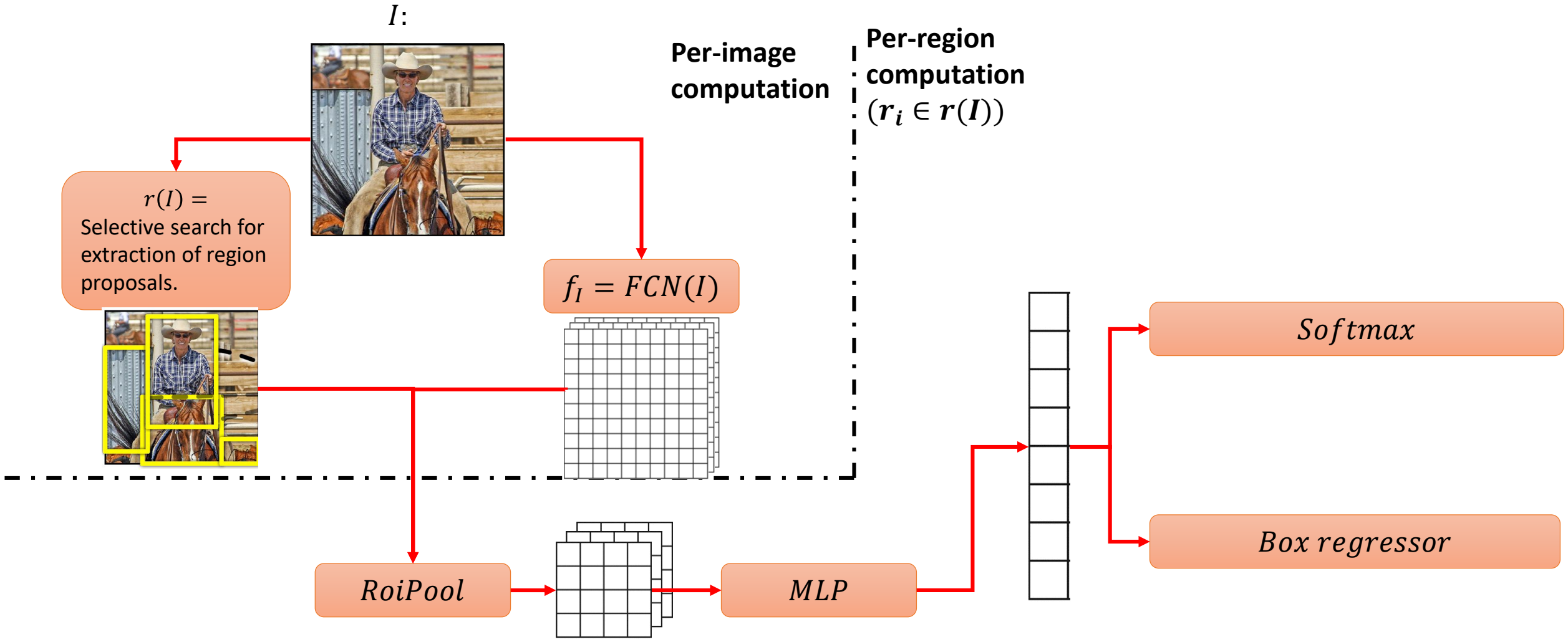
But is it efficient? → ***"Slow R-CNN"***



The CNN computation is very heavy per-region!

Girshick, R. et.al (2014) Rich feature hierarchies for accurate object detection and semantic segmentation.

Fast R-CNN



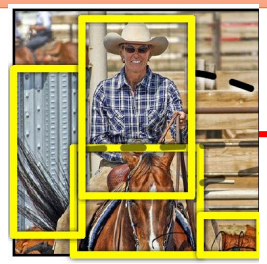
Girshick, R. (2015). Fast R-CNN.

Fast R-CNN

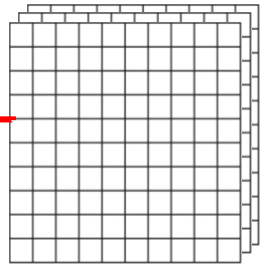
Per-image computation

I:

$r(I) =$
Selective search for extraction of region proposals.



$f_I = FCN(I)$

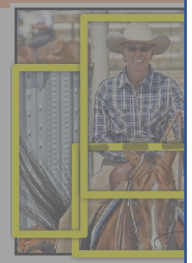


Girshick, R. (2015). Fast R-CNN.

Fast R-CNN

Per-region computation
$(\boldsymbol{r_i} \in \boldsymbol{r}(\boldsymbol{I}))$

$r(I) =$
Selective sear
extraction of
proposals.



RoiPool

MLP

Softmax

Box regressor

er

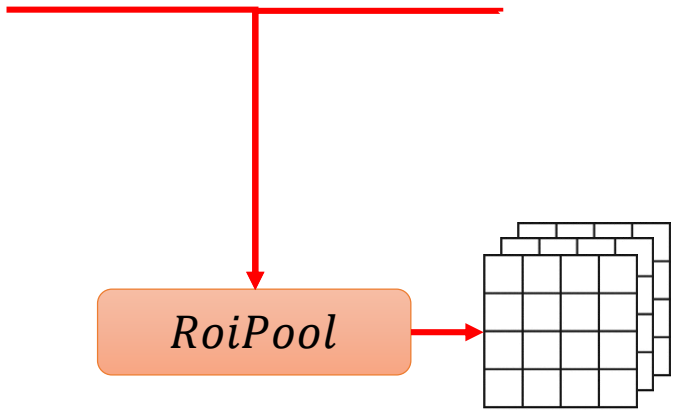
r

Girshick, R. (2015). Fast R-CNN.

History of instance segmentation

R-CNN (014)

What changes?

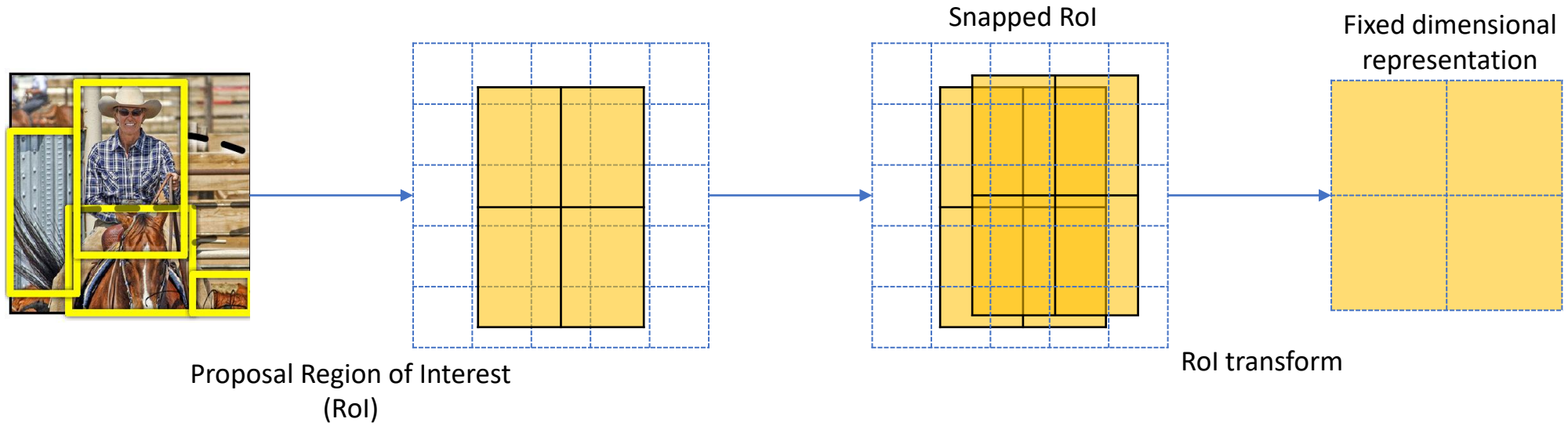


Converts each region into a fixed dimensional representation.

Girshick, R. (2015). Fast R-CNN.

RoIPool

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



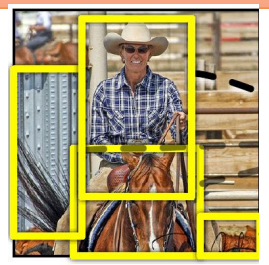
Girshick, R. (2015). Fast R-CNN.

History of instance segmentation

R-CNN (2014)

Can it be enhanced?

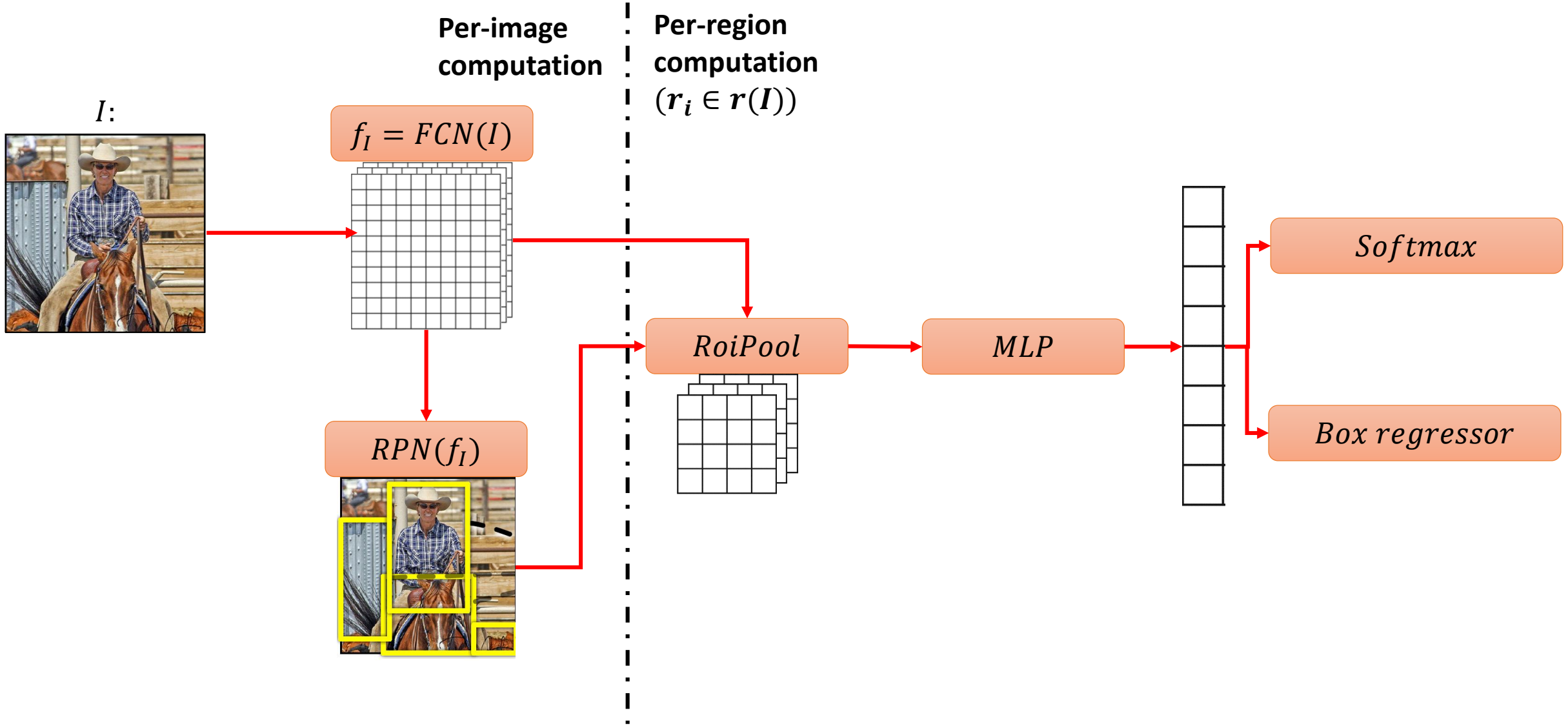
$r(I) =$
Selective search for extraction of region proposals.



The region proposals have very poor recall and they are slow!

Girshick, R. (2015). Fast R-CNN.

Faster R-CNN

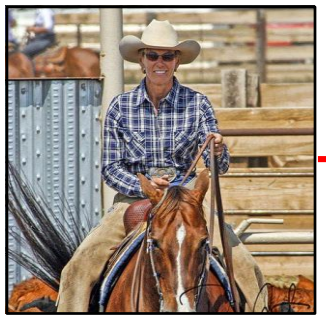


Girshick, R., Ren, S., He, K., and Sun, J. *(2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.*

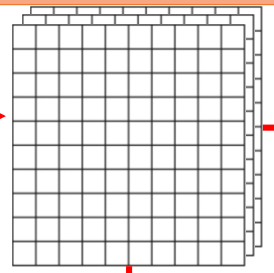
Faster R-CNN

Per-image computation

I:



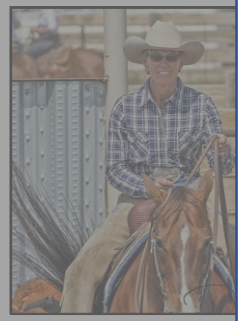
$f_I = FCN(I)$



$RPN(f_I)$

$RoiPool$

I:



max

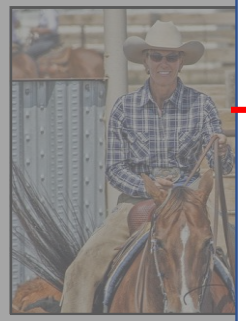
$ressor$

Girshick, R., Ren, S., He, K., and Sun, J. *(2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.*

Faster R-CNN

Per-region computation
$(\boldsymbol{r_i} \in \boldsymbol{r}(\boldsymbol{I}))$

I:



RoiPool

MLP

Softmax

Box regressor

max

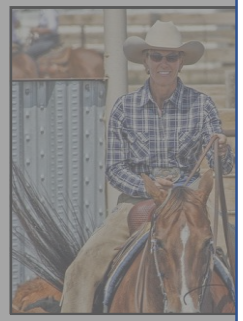
ressor

Girshick, R., Ren, S., He, K., and Sun, J. *(2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.*

Faster R-CNN

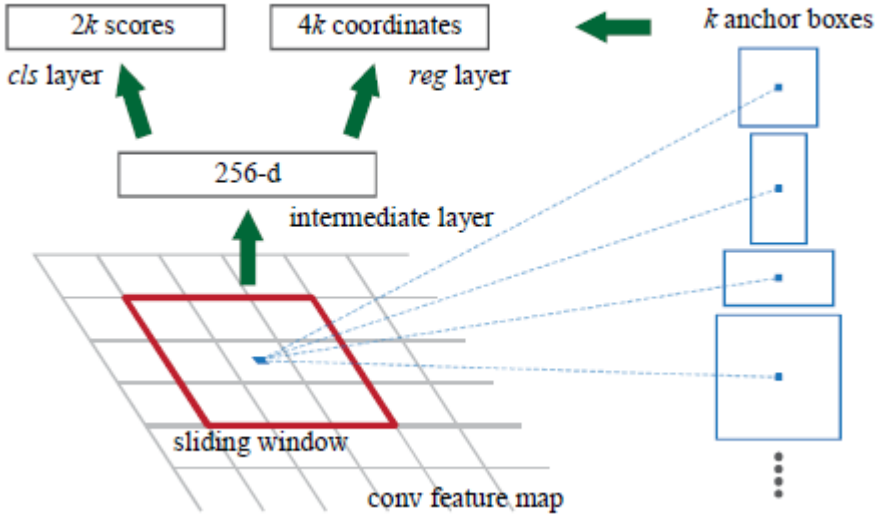
What changes?

I:



$RPN(f_I)$

RPN: Region Proposal Network

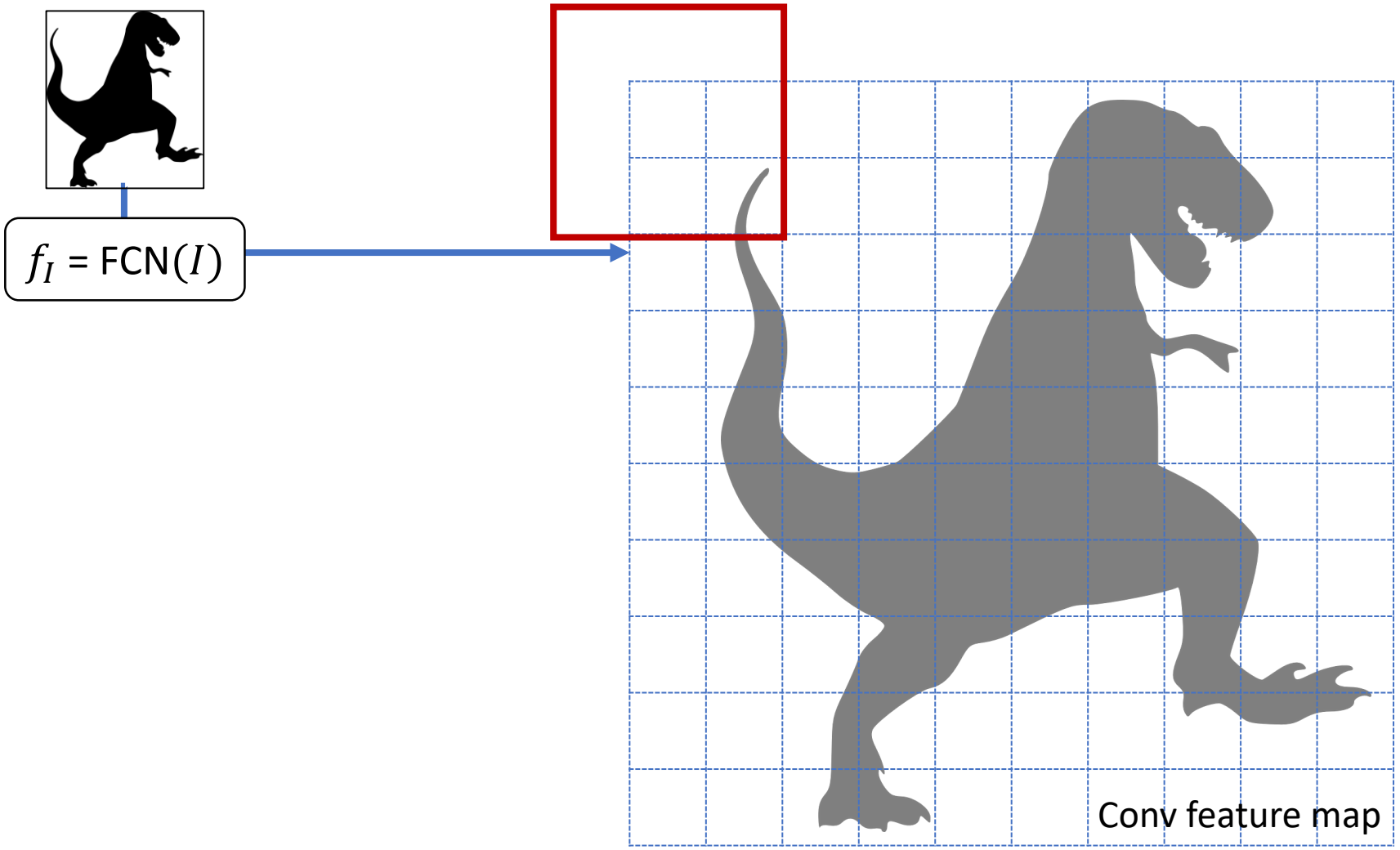


max

ressor

Girshick, R., Ren, S., He, K., and Sun, J. *(2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.*

RPN: Region Proposal Network

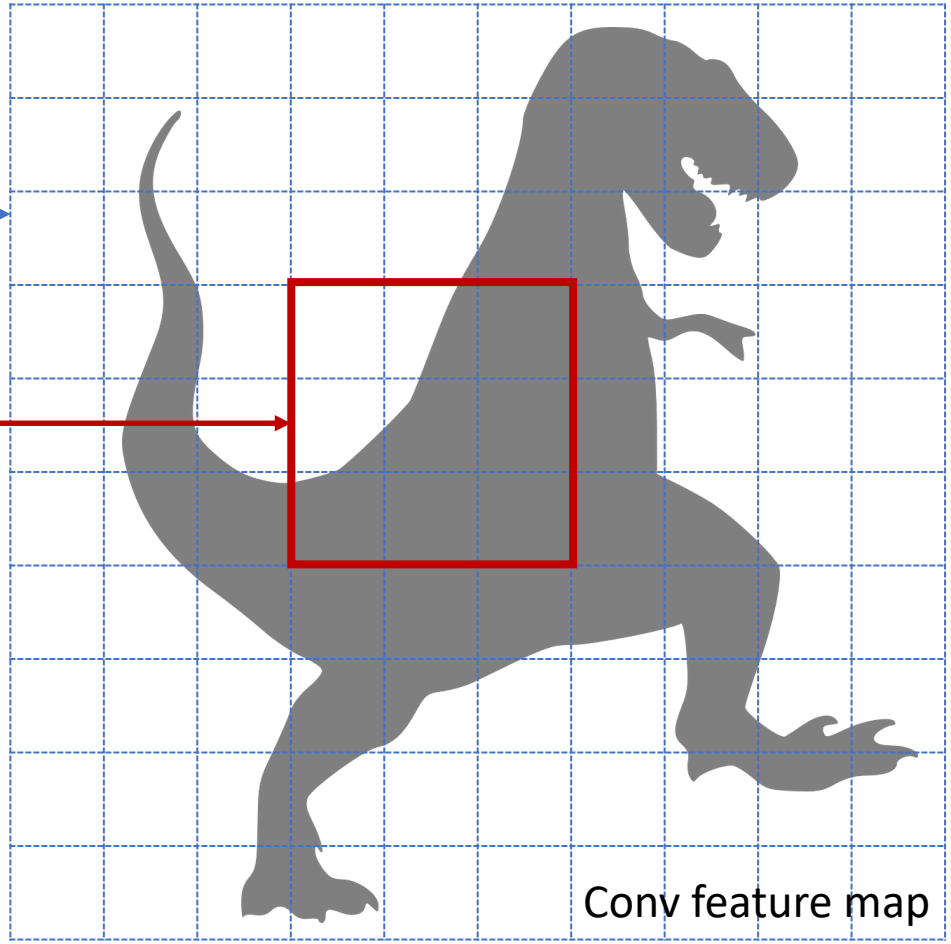


RPN: Region Proposal Network

f_I = FCN(I)

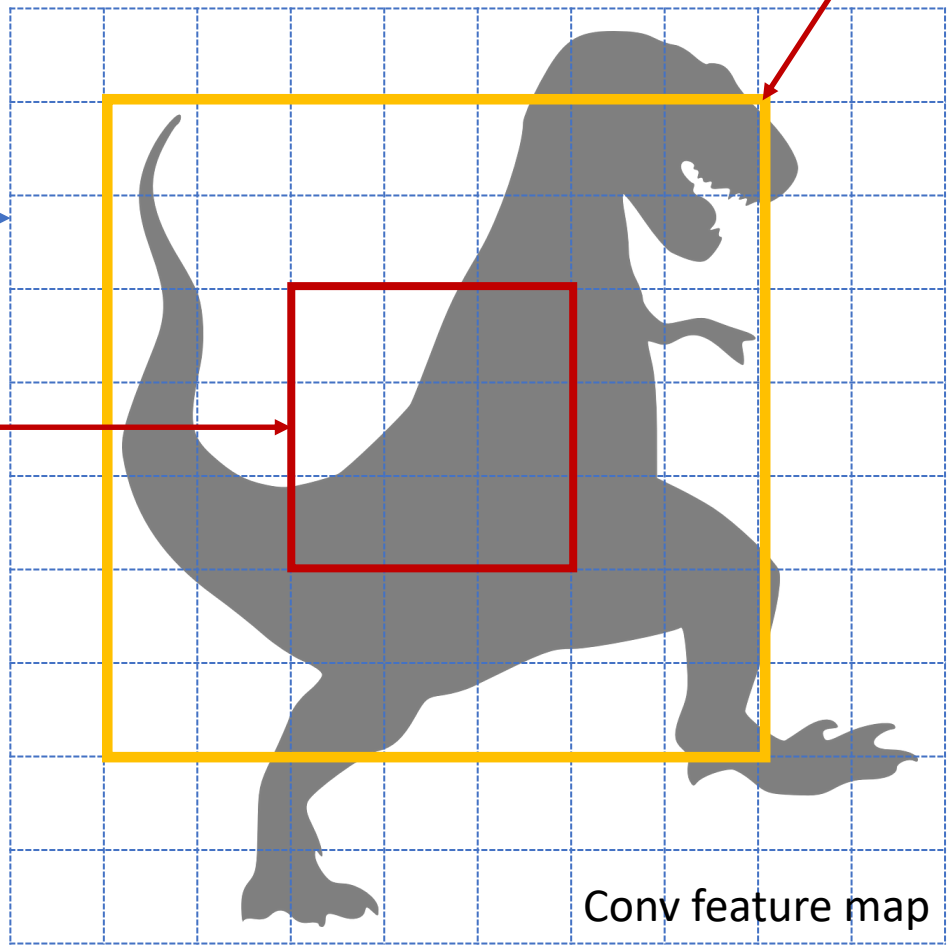
3x3 “sliding window”

Scans the feature map looking for objects



RPN: Anchor Box

f_I = FCN(I)



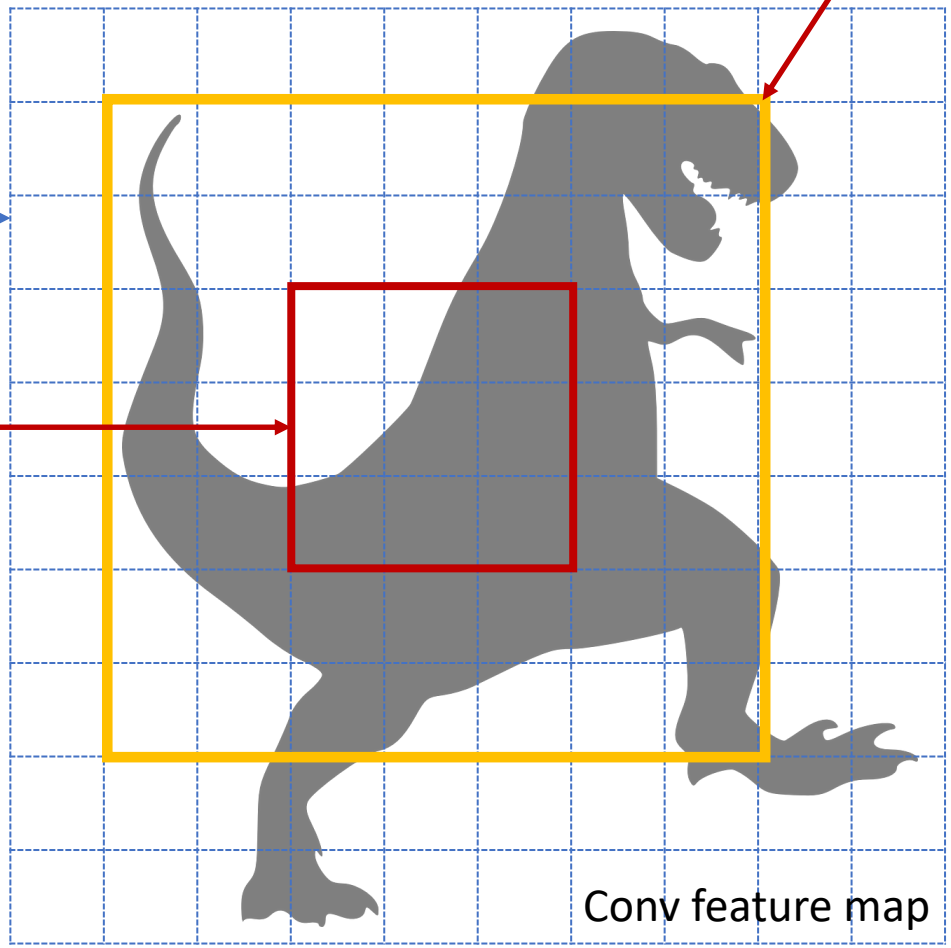
Anchor box: predictions are for this box, *not the 3x3 sliding window.*

3x3 “sliding window”

Scans the feature map looking for objects

RPN: Anchor Box

f_I = FCN(I)



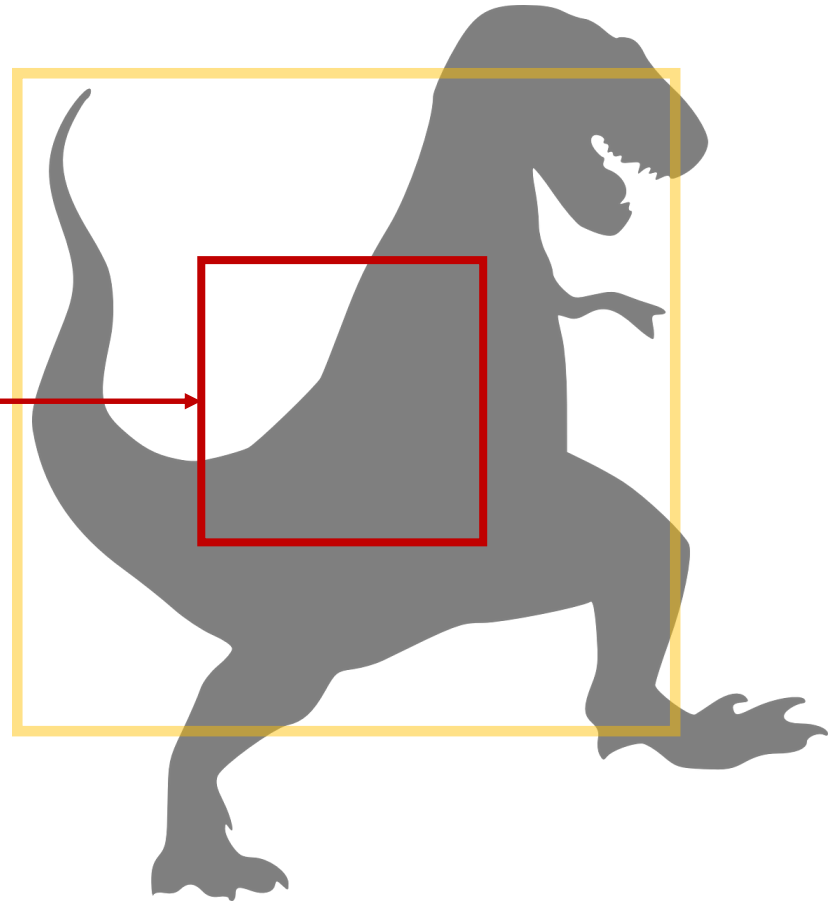
Anchor box: predictions are for this box, *not the 3x3 sliding window*

3x3 “sliding window”

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

RPN: Prediction (on object)

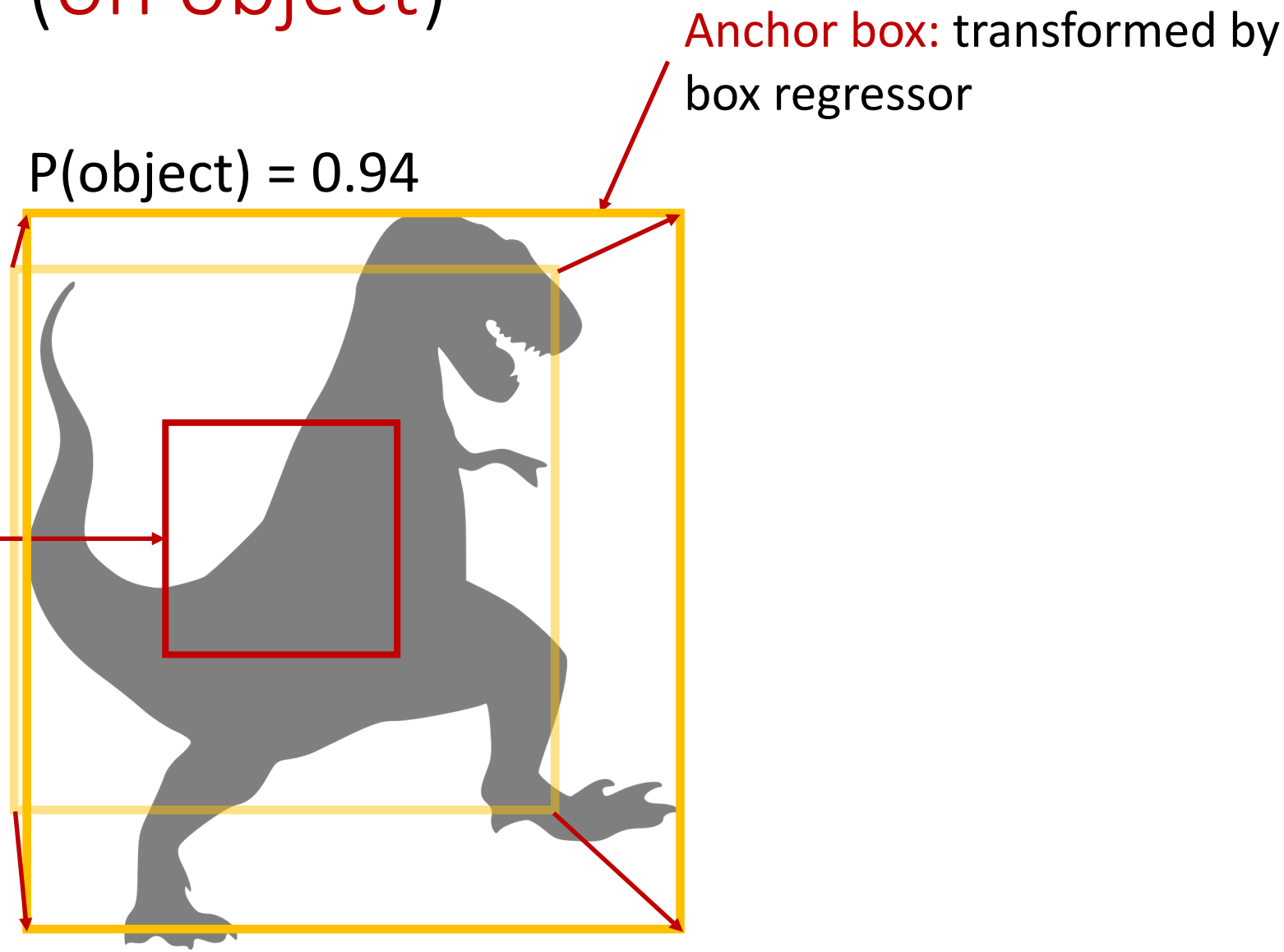
Objectness score → P(object) = 0.94



3x3 “sliding window” →

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

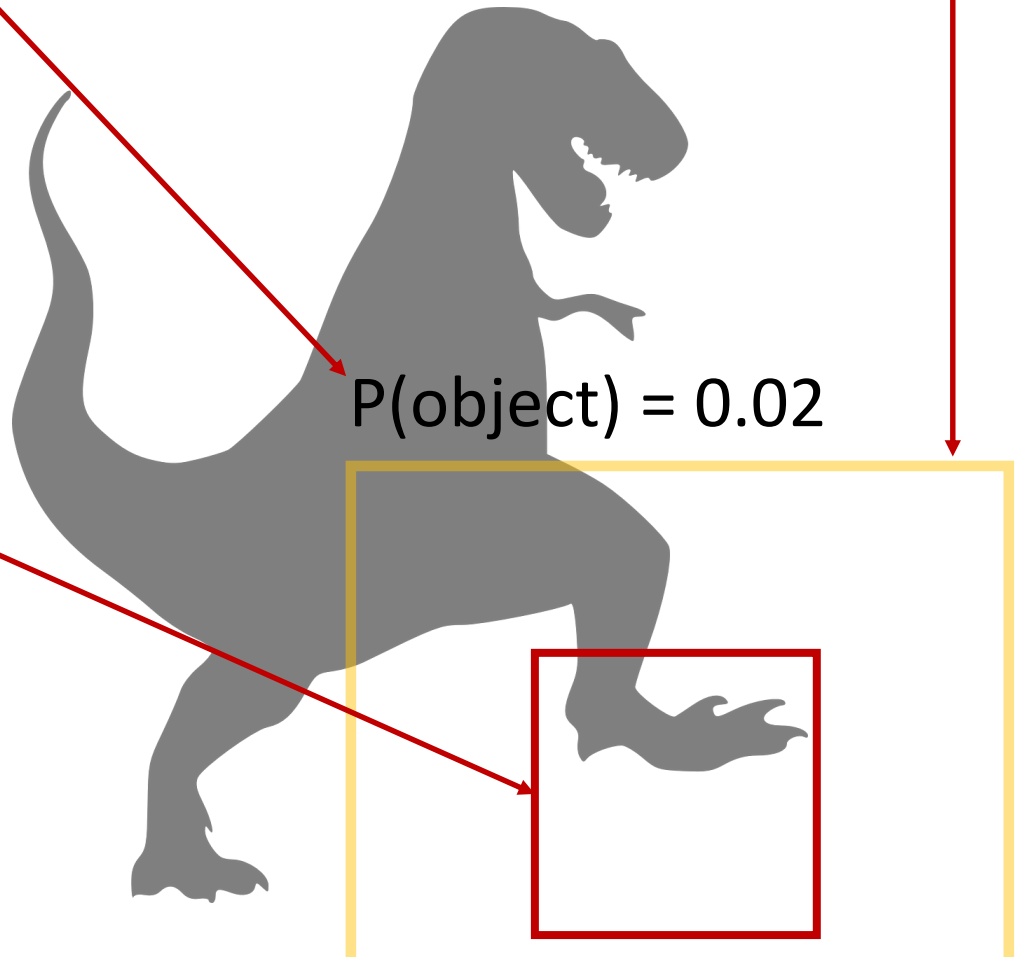
RPN: Prediction (on object)



RPN: Prediction (off object)

Objectness score

Anchor box: transformed by box regressor



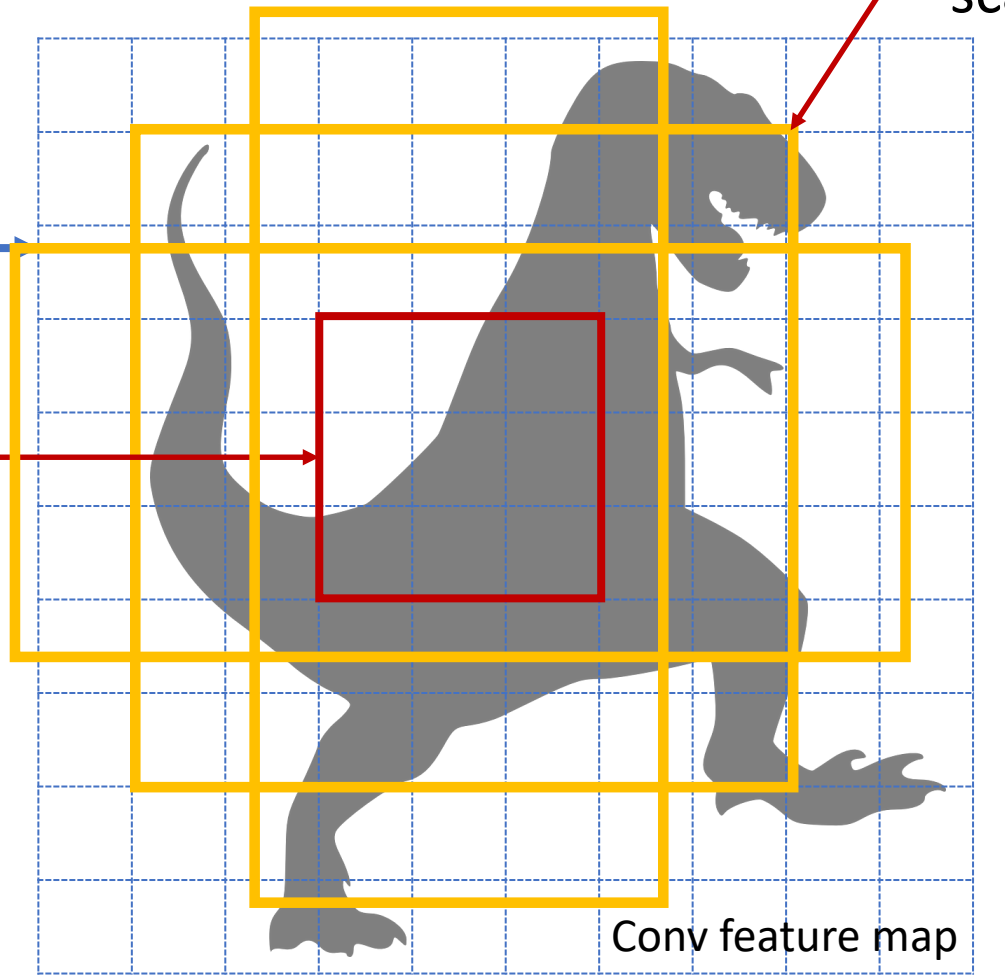
3x3 “sliding window”

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

RPN: Multiple Anchors

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

f_I = FCN(I)

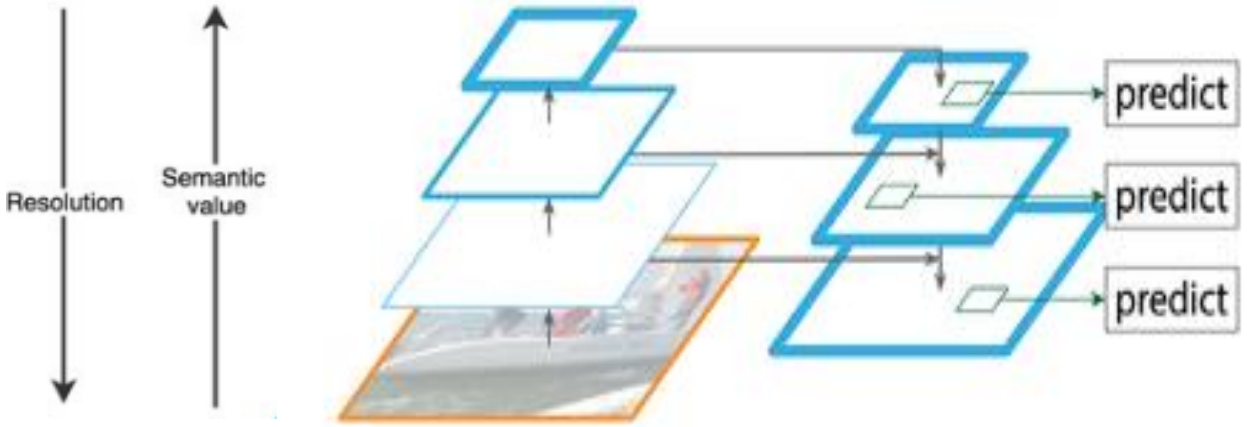


3x3 “sliding window”

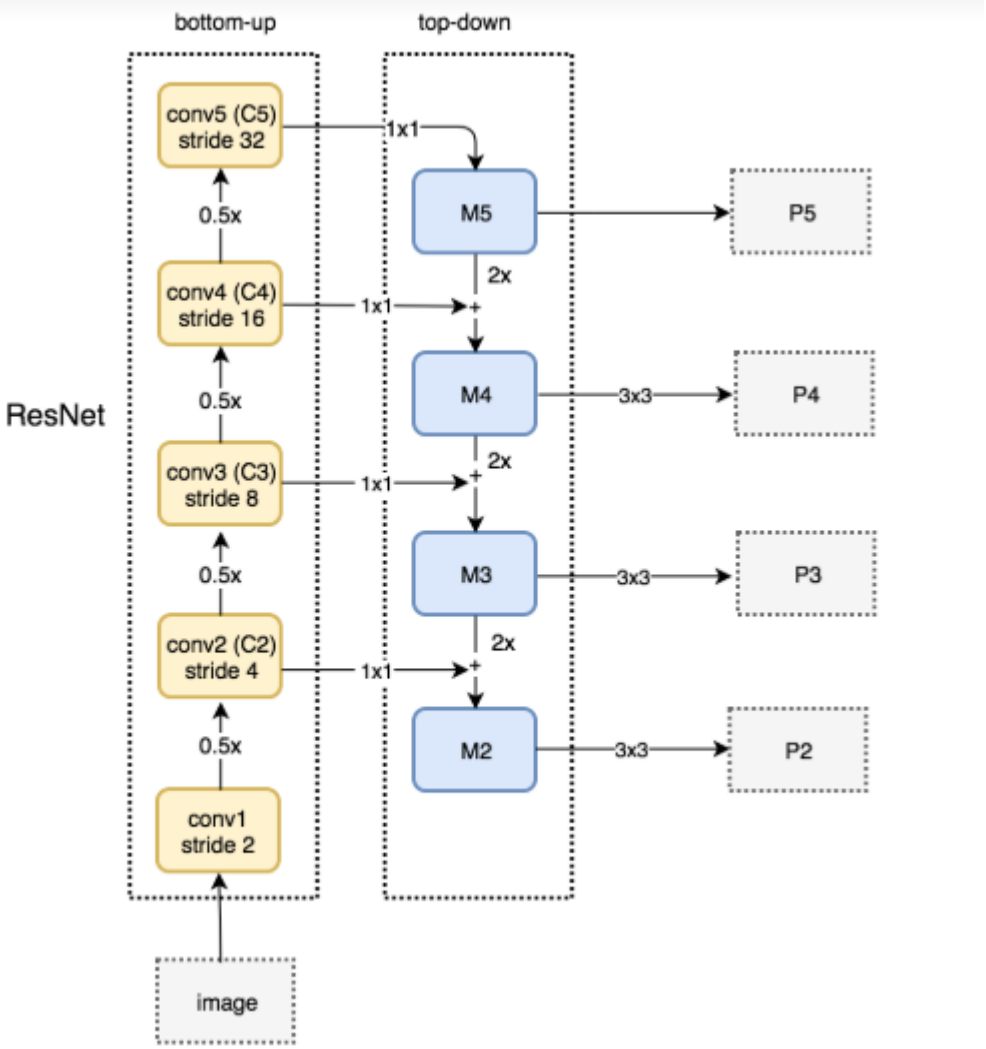
- *K* objectness classifiers
- *K* box regressors

Feature Pyramid Networks (FPN)

One of the baseline framework for object detection and instance segmentation methods.



Architecture of the FPN.



Lin T.Y et al (2017). Feature Pyramid Networks for Object Detection.

Until here….

- Some methods present **huge problems** with the **overlapping** between instances and **spurious edges**.

- Other methods use **two different networks** for object detection and the prediction of the masks for the instances.

- Finally there is a family of methods which start from semantic segmentation outputs and apply different transformations to obtain the instances.

So what if...

We combine the state of the art method for object detection , **Faster R-CNN**, and apply a **FCN** (fully convolutional network) on each **RoI** (region of interest)?

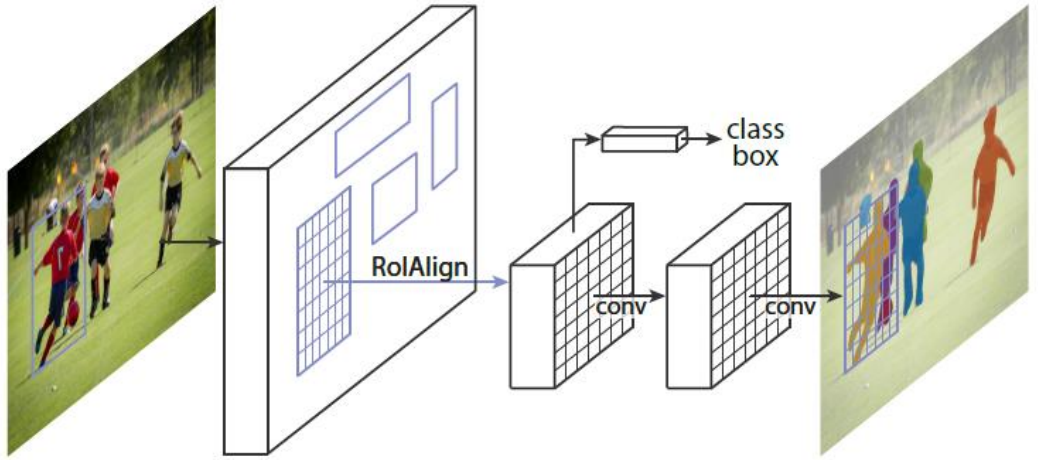
Mask R-CNN

Kaiming He Georgia Gkioxari Piotr Dollár Ross Girshick

Facebook AI Research (FAIR)

Mask R-CNN

- *What is new respect to previous methods?* → The addition of a branch for predicting an **object mask** in parallel with the existing branch for bounding box recognition.



Kaiming, H., Gkioxari, G., Dollár, P and Girshick, R. (2017). Mask R-CNN.

Mask R-CNN

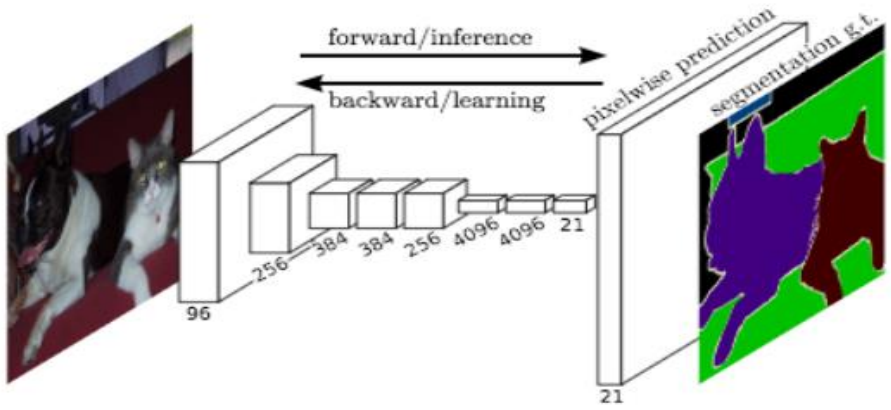
Motivation

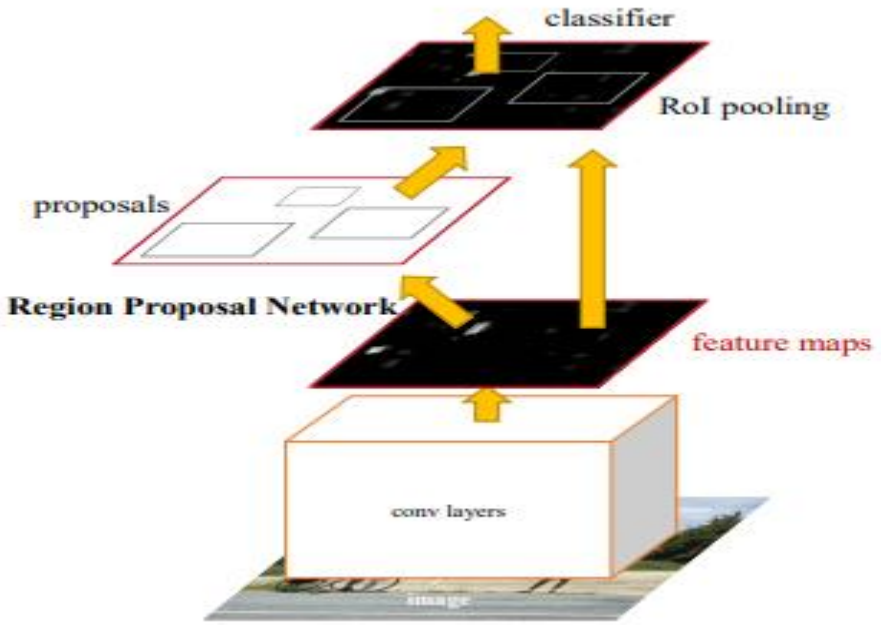
FCN and Faster R-CNN

Conceptually intuitive

Flexibility

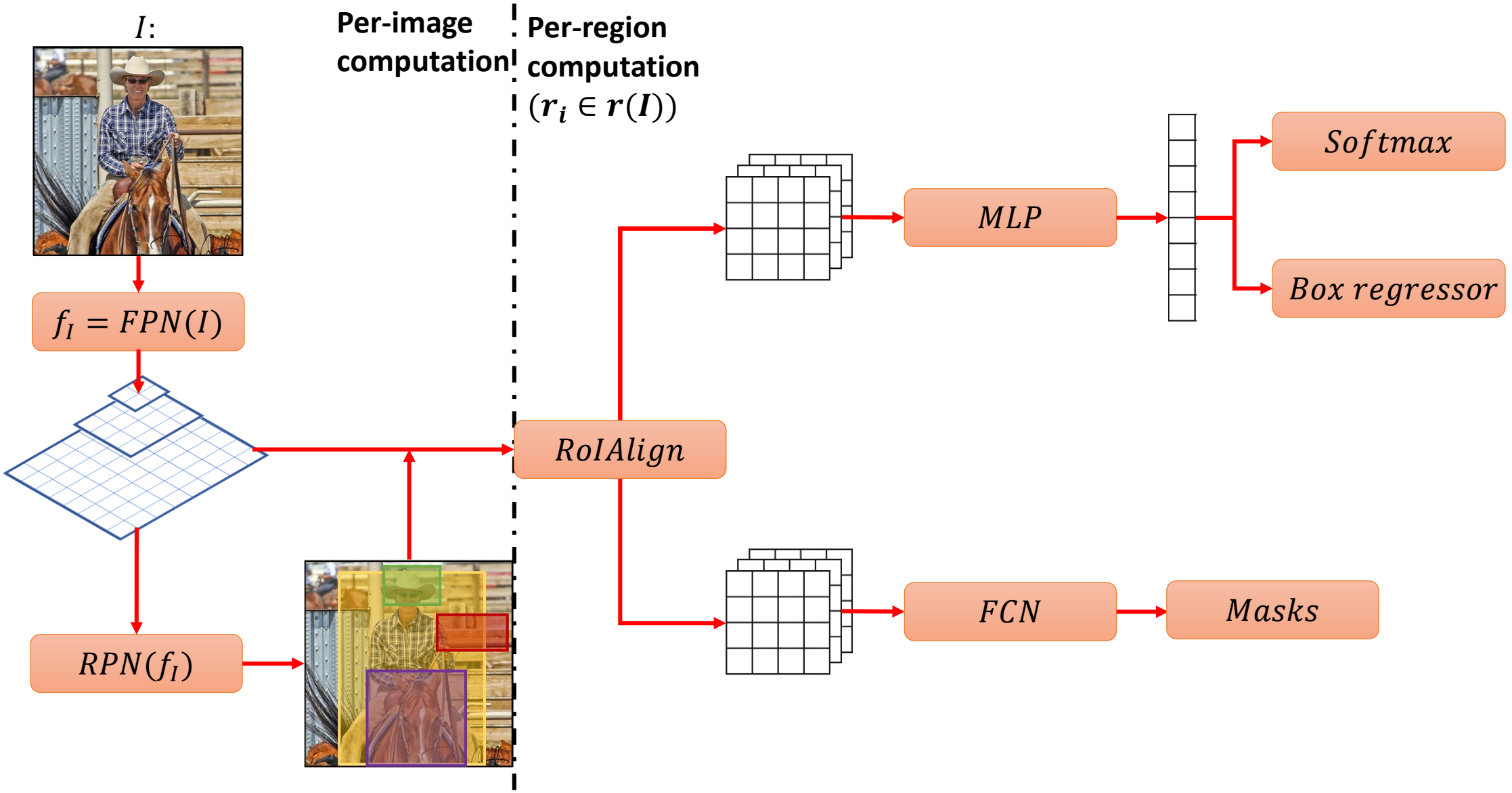
Robustness





Kaiming, H., Gkioxari, G., Dollár, P and Girshick, R. (2017). Mask R-CNN.

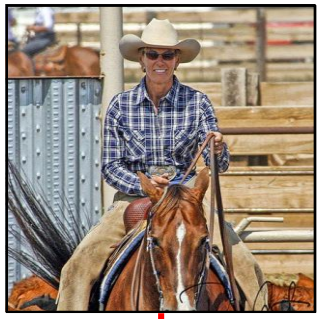
Mask R-CNN: Architecture



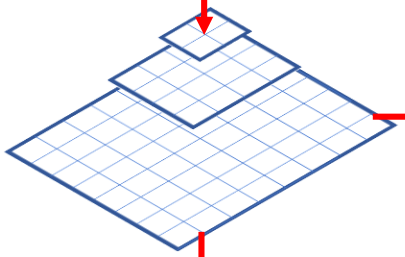
Girshick, R. (2019). The generalized R-CNN framework for Object Detection.

Per-image computation

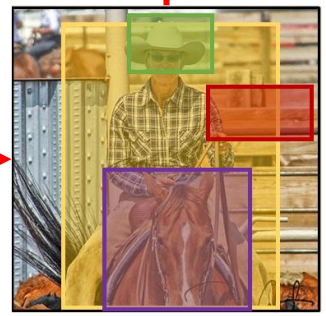
I:



$f_I = FPN(I)$



$RPN(f_I)$

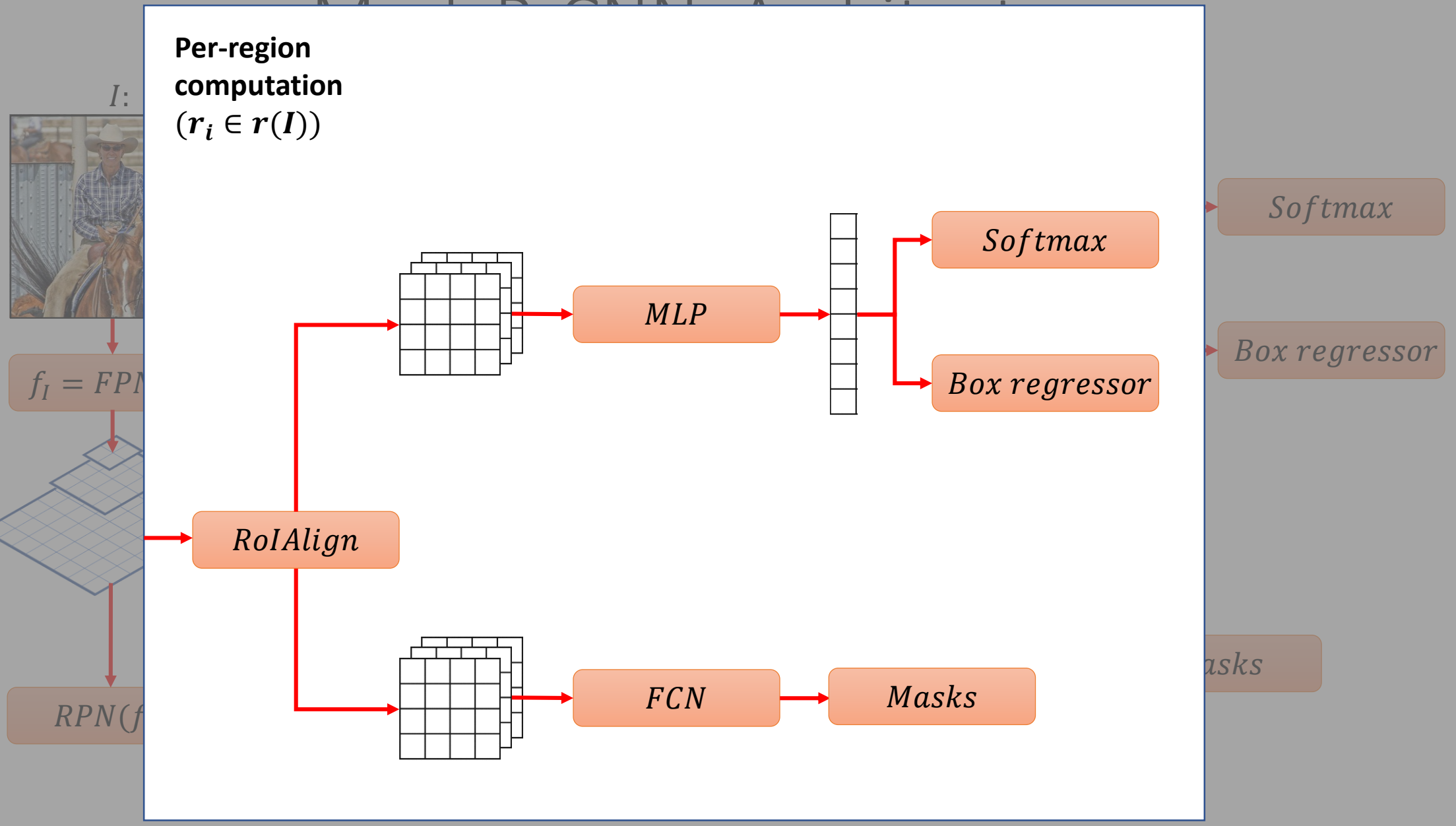


$Softmax$

$Box\ regressor$

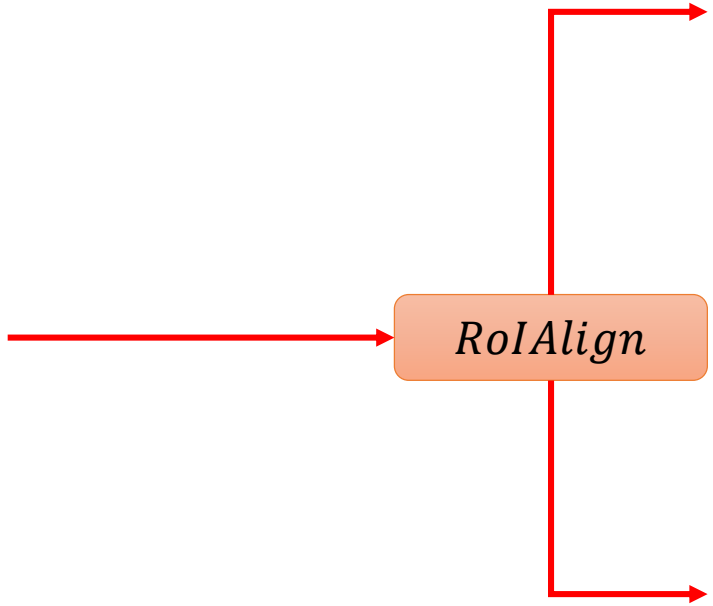
$asks$

Girshick, R. (2019). *The generalized R-CNN framework for Object Detection.*



Girshick, R. (2019). *The generalized R-CNN framework for Object Detection.*

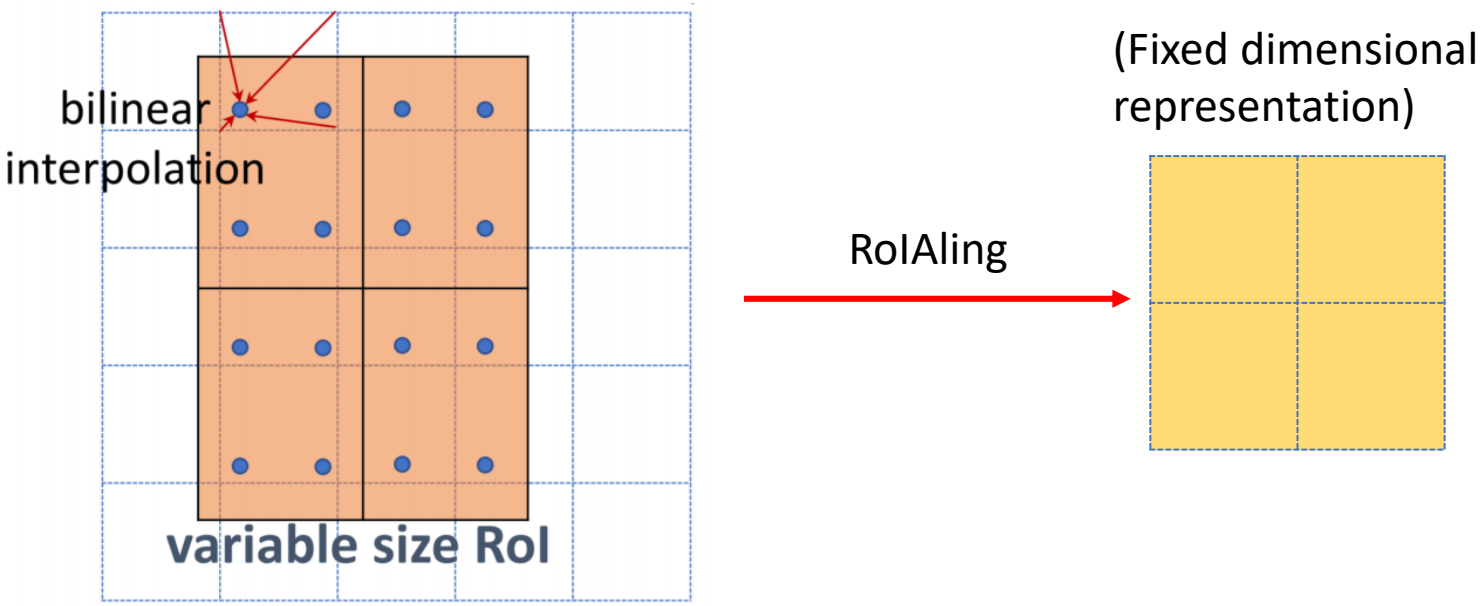
What changes?



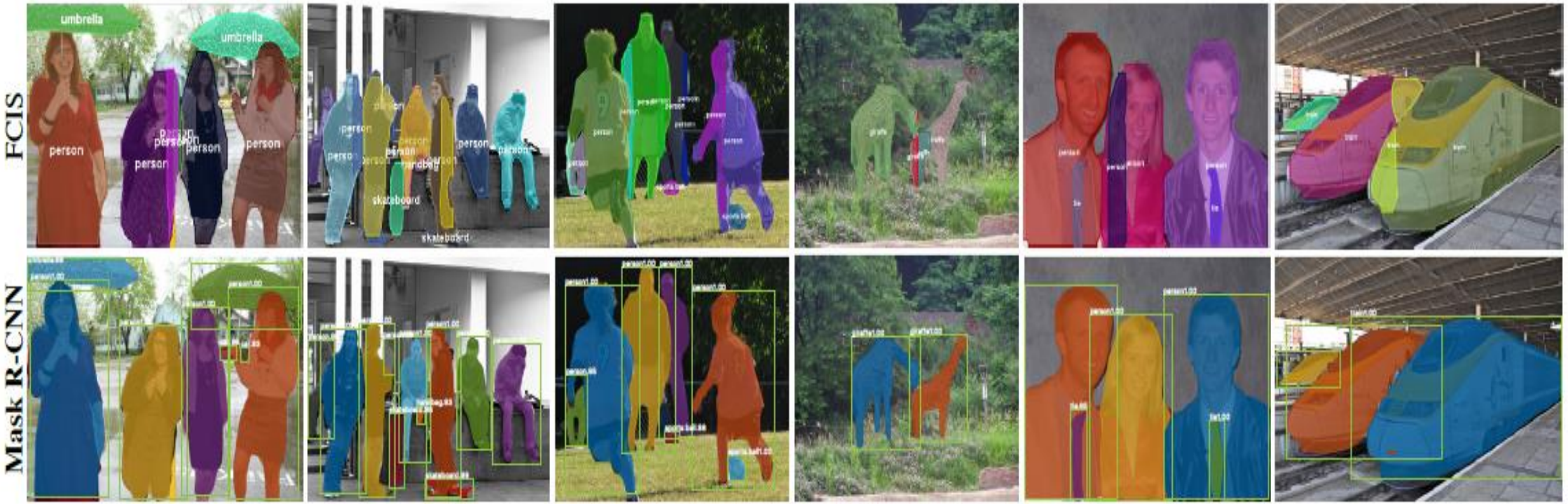
RoIPool becomes RoIAlign!

RoIAlign

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



Mask R-CNN results compared to FCIs results on COCO



BIG *difference about the overlapping.*

Mask R-CNN results on COCO.

Mask R-CNN results on instance segmentation mask AP.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Outperforms FCIS +++ by 4 points.

Mask R-CNN results – comparison between RoIPool and RoIAlign.

	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**

RoIAlign shows a big improvement in AP.

Mask R-CNN results – comparison between RoIPool and RoIAlign.

	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*

RoIAlign shows a big improvement in AP.

Other applications of Mask R-CNN

Key point-detection for human pose estimation.

So far:

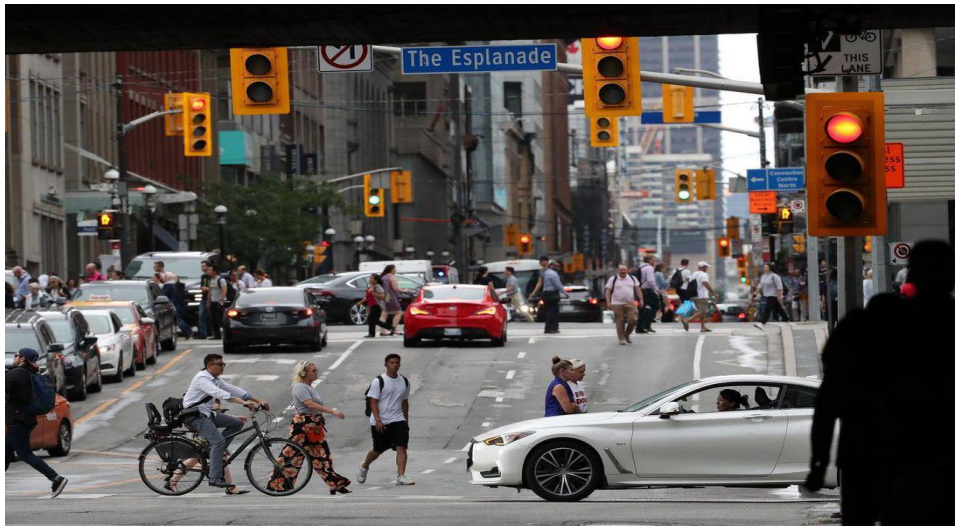
Semantic segmentation 

Instance segmentation 

Important distinctions

- **Stuff**: An **amorphous region** of an image with a similar texture that is **uncountable**. *i.e (grass, sky, water)*

- **Things** : Each one of the **objects** that appears on an image and are **countable**. *i.e (pedestrian, dog, bottle).*

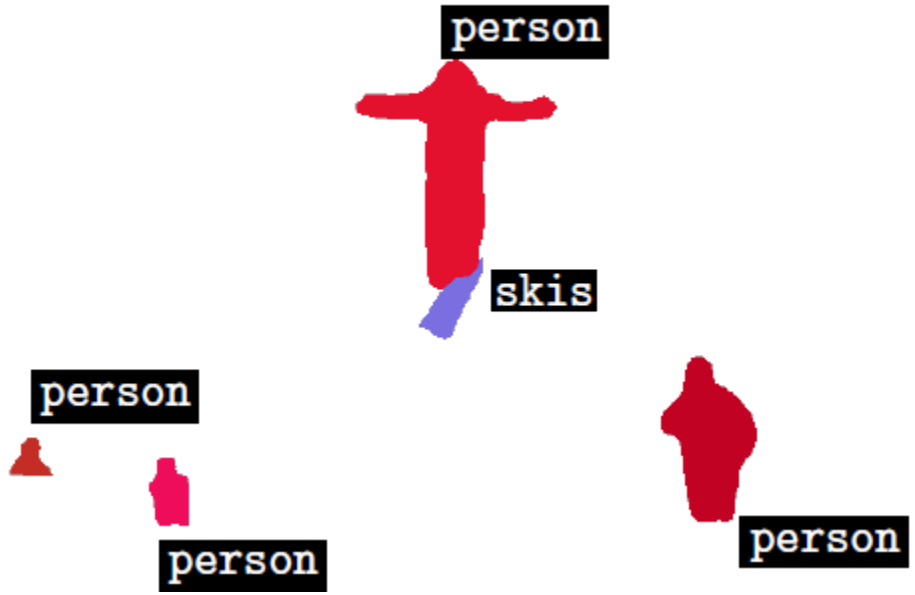


But what are the problems of solving both tasks separately?

- Real-world applications **requires** both **Semantic** and **Instance** segmentation tasks!
- Instance segmentation does not give an understanding of the general scene layout.
- In instance segmentation, “stuffs” are not segmented.
- In semantic segmentation, instances are indistinguishable.

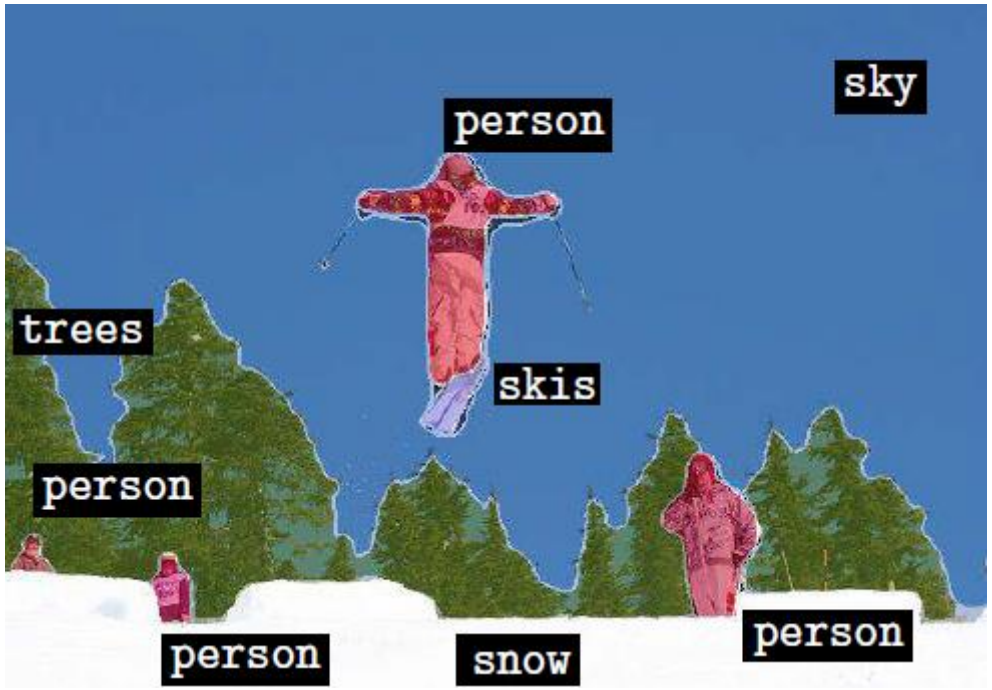
But what are the problems of solving both ta

- Instance segmentation does not give an understanding of the general scene layout.
- In instance segmentation, “stuffs” are not segmented.



But what are the problems of solving both ta

- Instance segmentation does not give an understanding of the general scene layout.
- In instance segmentation, “stuffs” are not segmented.



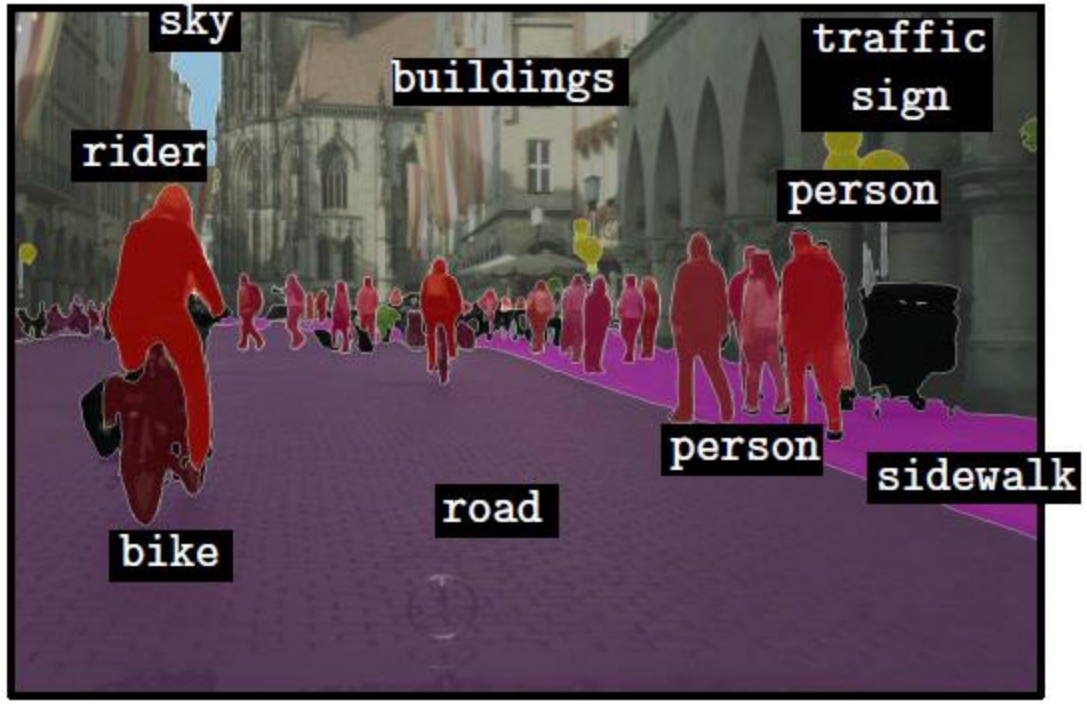
Combining with semantic segmentation predictions.

But what are the problems of solving both ta

- In semantic segmentation, instances are indistinguishable.

But what are the problems of solving both ta

- In semantic segmentation, instances are indistinguishable.

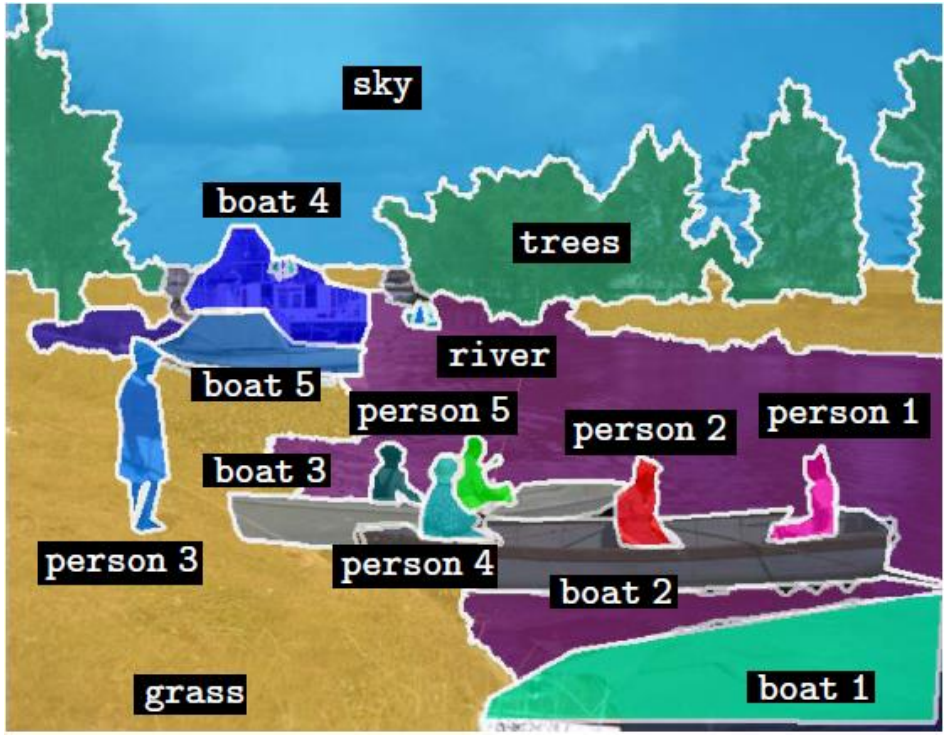


Combining with instance segmentation predictions.

Panoptic Segmentation

Panoptic Segmentation (PS)

- Unifying **Semantic** and **Instance** segmentation tasks.
- **Panoptic:** “Including everything visible in one view”



Panoptic Segmentation





Alex Kirillov Carsten Rother Kaiming He Ross Girshick Piotr Dollár

UNIVERSITÄT HEIDELBERG

FACEBOOK AI RESEARCH

Previous approaches

- ***Image parsing (2005):*** Unification of image segmentation, object detection and recognition.

- ***Holistic scene understanding (2012):*** Unification of object detection, scene classification and semantic segmentation.

- ***Scene parsing (2014):*** Assigning a semantic label at every pixel and inferring the spatial extent of individual object instances.

The task

Given a set of L semantic classes encoded by:

$$\mathcal{L} \coloneqq \{0,1,2,3, \dots, L-1\}$$

the task requires a *panoptic segmentation algorithm* to map each pixel i of an image to a pair $(l_i, z_i) \in \mathcal{L} \times \mathbb{N}$.

$l_i \rightarrow$ Semantic class of pixel i

$z_i \rightarrow$ Instance id of pixel i

Kirillov, A., He, K., Girshick, R., Rother, C. y Dollár, P. (2018). *Panoptic Segmentation. CoRR, abs/1801.00868.*

The task

Given a set of L seman

the task requires a *par* ch pixel i of an image to a pair $(l_i, z_i) \in \mathcal{L} \times \mathbb{N}$.

Kirillov, A., He, K., Girshick, R., Rother, C. y Dollár, P. (2018). *Panoptic Segmentation. CoRR, abs/1801.00868.*

The task: Things vs Stuffs

Semantic label set is composed by two subsets:

$\mathcal{L}^{St} \rightarrow$ Related to stuffs.
$\mathcal{L}^{Th} \rightarrow$ Related to things.

Such that:

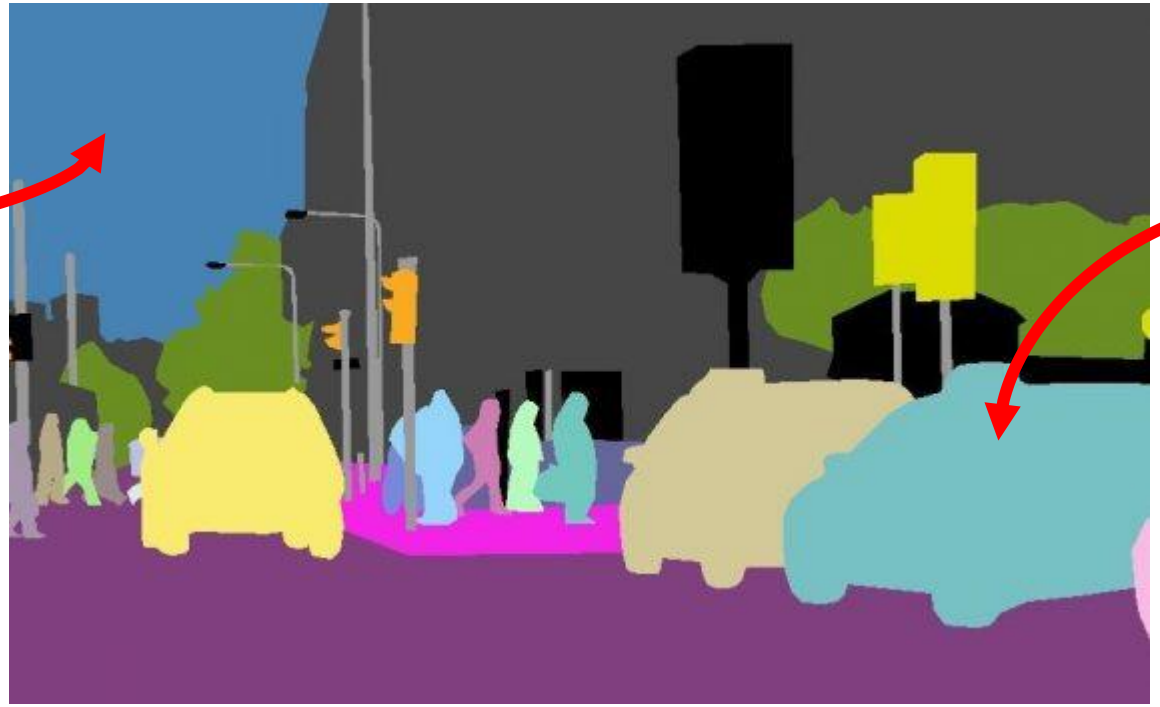
$$\mathcal{L} = \mathcal{L}^{St} \cup \mathcal{L}^{Th}$$
$$\mathcal{L}^{St} \cap \mathcal{L}^{Th} = \emptyset$$

Kirillov, A., He, K., Girshick, R., Rother, C. y Dollár, P. (2018). *Panoptic Segmentation. CoRR, abs/1801.00868.*

The task: Things vs Stuffs

If $l_i \in \mathcal{L}^{St} \Rightarrow$ the corresponding instance id z_i is irrelevant.

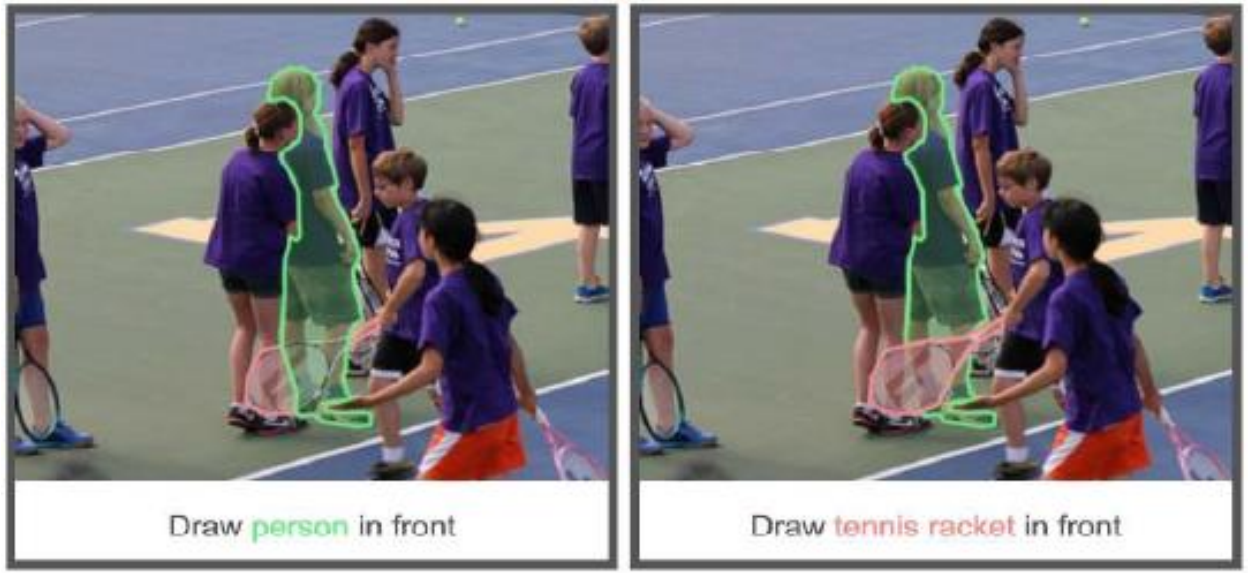
All pixels with the same (l_i, z_i): $l_i \in \mathcal{L}^{Th}$ belong to the same instance.



Kirillov, A., He, K., Girshick, R., Rother, C. y Dollár, P. (2018). *Panoptic Segmentation. CoRR, abs/1801.00868.*

Can there be overlapping?

Instance segmentation allows overlapping.



In PS each pixel has only one label!

Metric

What did they want the metric to have?

- Completeness.
- Interpretability.
- Simplicity.

Panoptic quality (PQ)

Involves 2 steps:

$i)$ Segment matching

$ii)$ PQ computation

Kirillov, A., He, K., Girshick, R., Rother, C. y Dollár, P. (2018). *Panoptic Segmentation. CoRR, abs/1801.00868.*

Metric: Segment matching

A predicted segment and a ground truth segment can match only if $IoU > 0.5$

Theorem 1. *Given a predicted and ground truth panoptic segmentation of an image, each ground truth segment can have at most one corresponding predicted segment with IoU strictly greater than 0.5 and vice verse.*

Kirillov, A., He, K., Girshick, R., Rother, C. y Dollár, P. (2018). *Panoptic Segmentation. CoRR, abs/1801.00868.*

Metric: PQ computation

Semantic segmentation metric (IoU) + Instance segmentation metric (AP)

$$PQ = \frac{\sum_{(p,g)\in TP} IoU(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

Kirillov, A., He, K., Girshick, R., Rother, C. y Dollár, P. (2018). *Panoptic Segmentation. CoRR, abs/1801.00868.*

Metric: PQ computation

Semantic segmentation metric (IoU) + Instance segmentation metric (AP)

$$PQ = \frac{\left[\sum_{(p,g)\in TP} IoU(p,g)\right]|TP|}{|TP|\left[|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|\right]}$$

Multiplying and dividing by TP.

Kirillov, A., He, K., Girshick, R., Rother, C. y Dollár, P. (2018). *Panoptic Segmentation. CoRR, abs/1801.00868.*

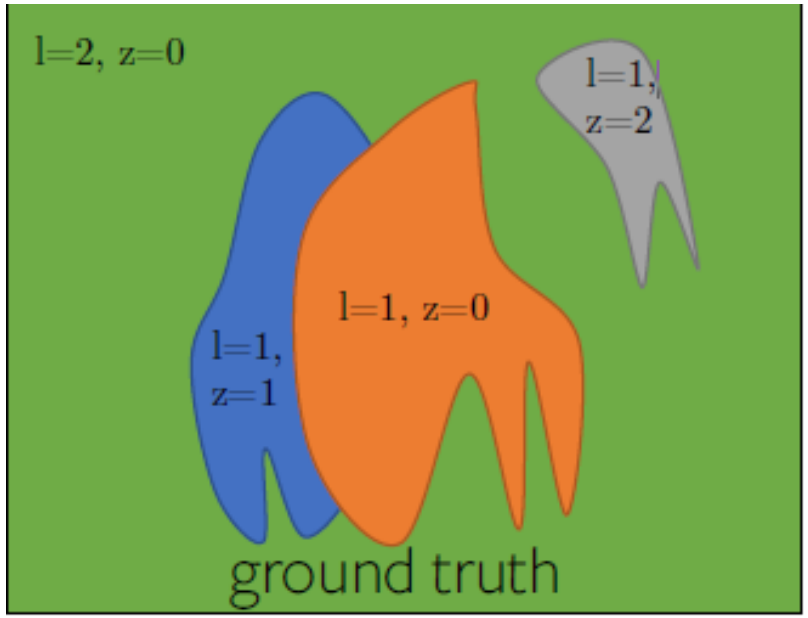
Metric: PQ computation

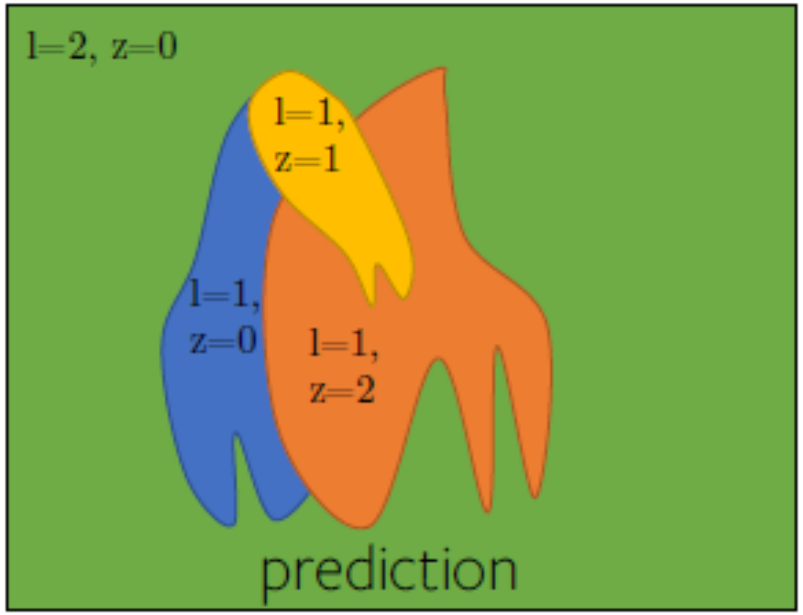
Semantic segmentation metric (IoU) + Instance segmentation metric (AP)

$$PQ = \underbrace{\frac{\sum_{(p,g)\in TP} IoU(p,g)}{|TP|}}_{\text{Segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{Recognition quality (RQ)}}$$

Kirillov, A., He, K., Girshick, R., Rother, C. y Dollár, P. (2018). *Panoptic Segmentation. CoRR, abs/1801.00868.*

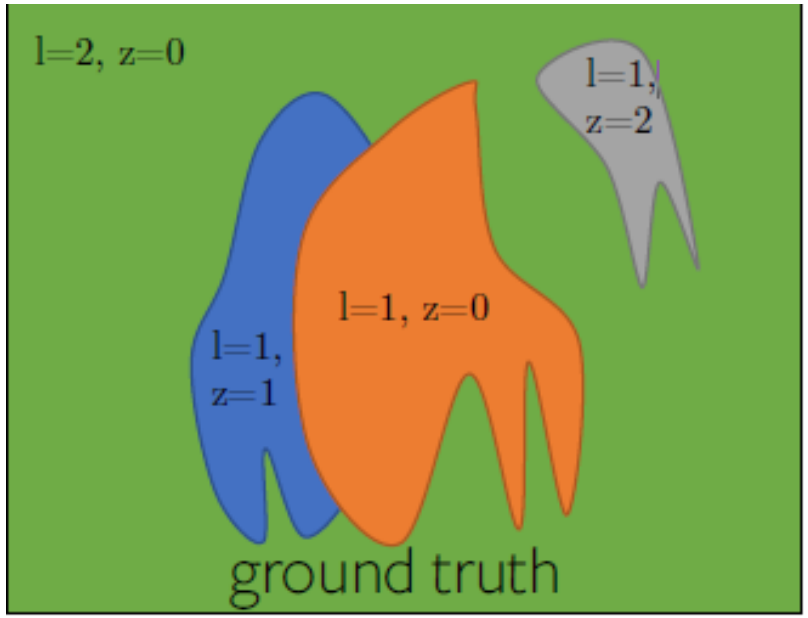
Metric: Example

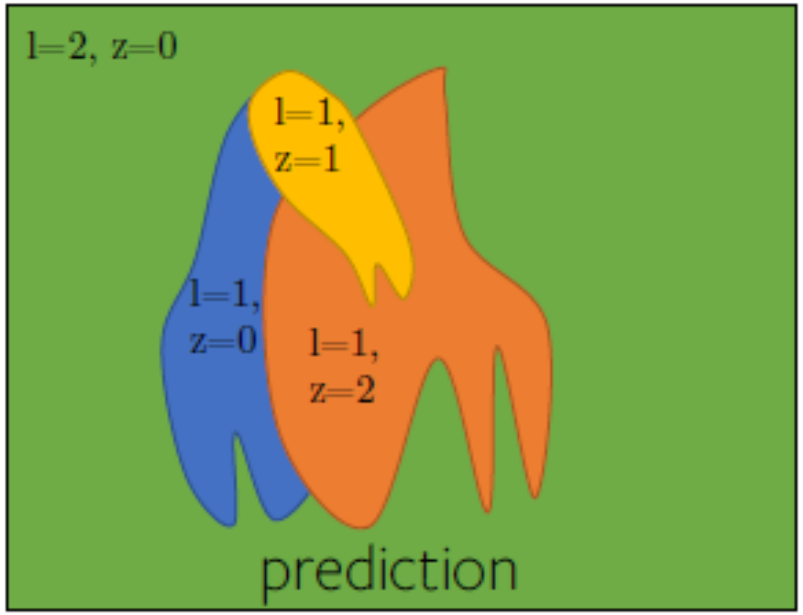




$$TP = \{(\boxed{}, \boxed{}), (\boxed{}, \boxed{})\},\ FN = \{\boxed{}\},\ FP = \{\boxed{}\}$$

Metric: Example





$$TP = \{(\boxed{}, \boxed{}), (\boxed{}, \boxed{})\},\ FN = \{\boxed{}\},\ FP = \{\boxed{}\}$$

$$PQ = \frac{IoU(\boxed{}, \boxed{}) + IoU(\boxed{}, \boxed{})}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

Datasets

COCO (2014) + COCO-stuff(2017)

Mapillary Vistas (2017)

Cityscapes (2015)

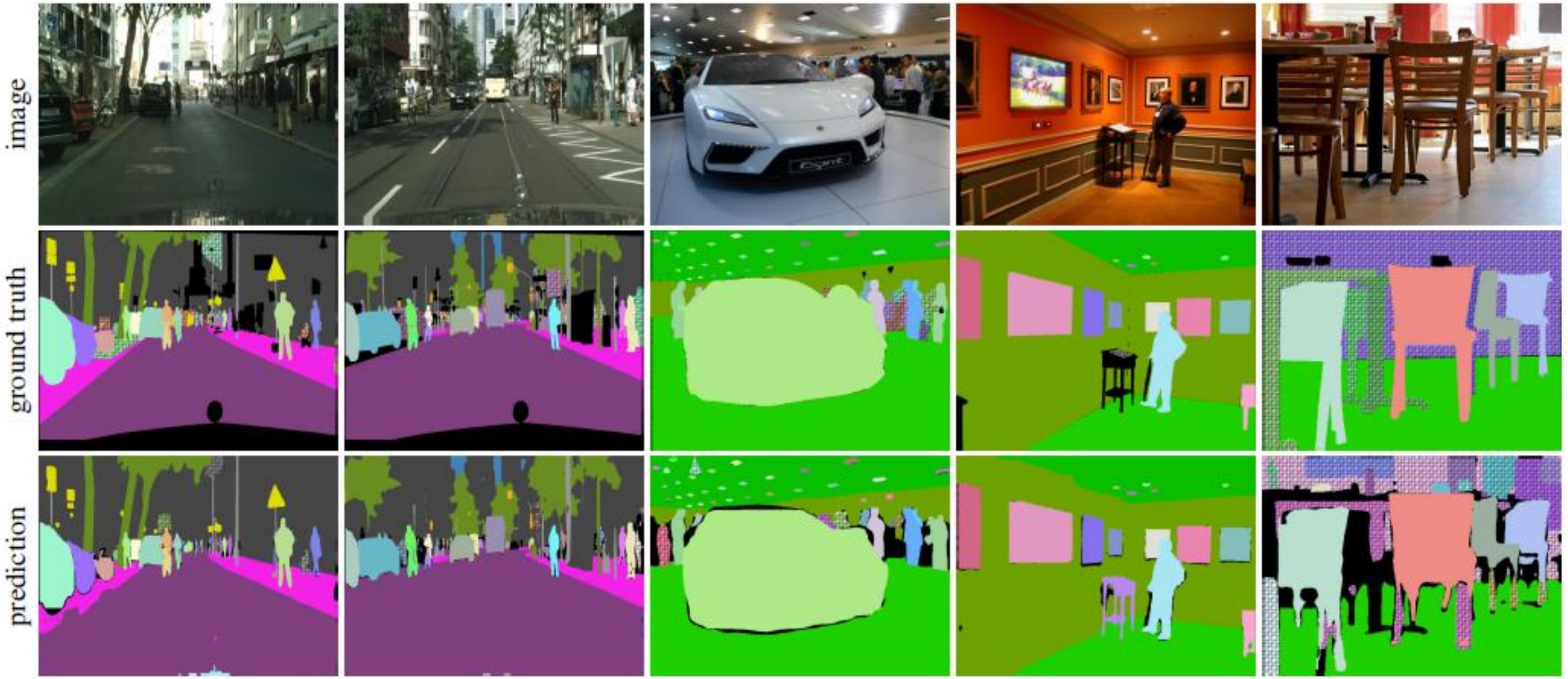
ADE20k (2016)

After proposing the task…

- There was an interest to perform a **heuristic combination** of the top performance algorithm on instance (**Mask R-CNN**) and semantic segmentation (**PSPNet**) .

- How does the panoptic quality (PQ) compare to existing metrics such as AP and IoU?

Implementation of PSP-Net with Mask R-CNN in Cityscapes and ADE20k .



Left two : Cityscapes , right three: ADE20k

Implementation of PSP-Net with Mask R-CNN in Cityscapes and ADE20k .

Cityscapes	AP	AP^{NO}	PQ^{Th}	SQ^{Th}	RQ^{Th}
Mask R-CNN+COCO [14]	**36.4**	**33.1**	**54.0**	**79.4**	**67.8**
Mask R-CNN [14]	31.5	28.0	49.6	78.7	63.0
ADE20k	AP	AP^{NO}	PQ^{Th}	SQ^{Th}	RQ^{Th}
Megvii [31]	**30.1**	**24.8**	**41.1**	**81.6**	**49.6**
G-RMI [10]	24.6	20.6	35.3	79.3	43.2

AP compared to PQ

Implementation of PSP-Net with Mask R-CNN in Cityscapes and ADE20k .

Cityscapes	IoU	PQ^{St}	SQ^{St}	RQ^{St}
PSPNet multi-scale [54]	**80.6**	**66.6**	**82.2**	**79.3**
PSPNet single-scale [54]	79.6	65.2	81.6	78.0
ADE20k	IoU	PQ^{St}	SQ^{St}	RQ^{St}
CASIA_IVA_JD [12]	**32.3**	**27.4**	**61.9**	**33.7**
G-RMI [11]	30.6	19.3	58.7	24.3

IoU compared to PQ

Results for the first approach

Cityscapes	PQ	PQ^{St}	PQ^{Th}
machine-separate	n/a	66.6	54.0
machine-panoptic	61.2	66.4	54.0
ADE20k	PQ	PQ^{St}	PQ^{Th}
machine-separate	n/a	27.4	41.1
machine-panoptic	35.6	24.5	41.1
Vistas	PQ	PQ^{St}	PQ^{Th}
machine-separate	n/a	43.7	35.7
machine-panoptic	38.3	41.8	35.7

Table 5: **Panoptic *vs*. independent predictions.** The 'machine-separate' rows show PQ of semantic and instance segmentation methods computed independently (see also Tables 3 and 4). For 'machine-panoptic', we merge the non-overlapping thing and stuff predictions obtained from state-of-the-art methods into a true panoptic segmentation of the image. Due to the merging heuristic used, PQ^{Th} stays the same while PQ^{St} is slightly degraded.

Break

Panoptic Feature Pyramid Networks

Alex Kirillov

Kaiming He

Ross Girshick

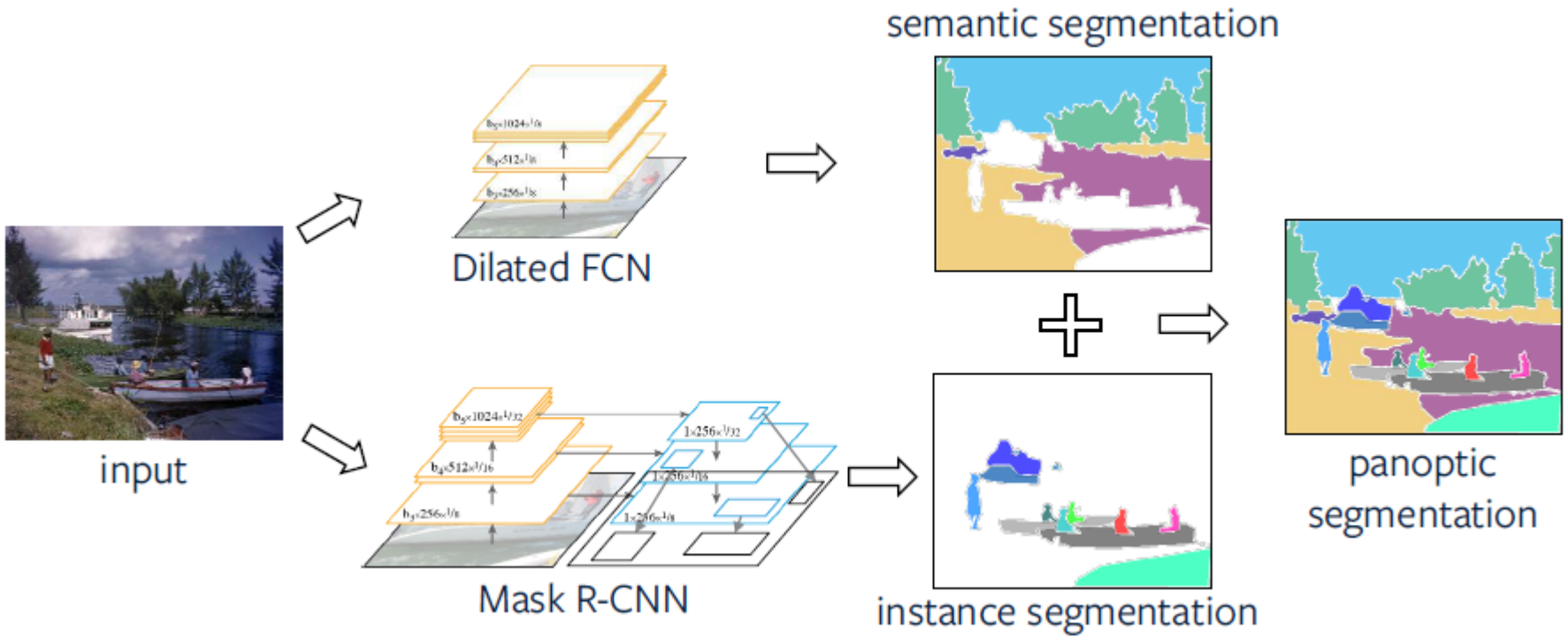
Piotr Dollár

UNIVERSITÄT HEIDELBERG

FACEBOOK AI RESEARCH

Basic approach

The most straightforward method to produce panoptic segmentation is combined two independent networks:

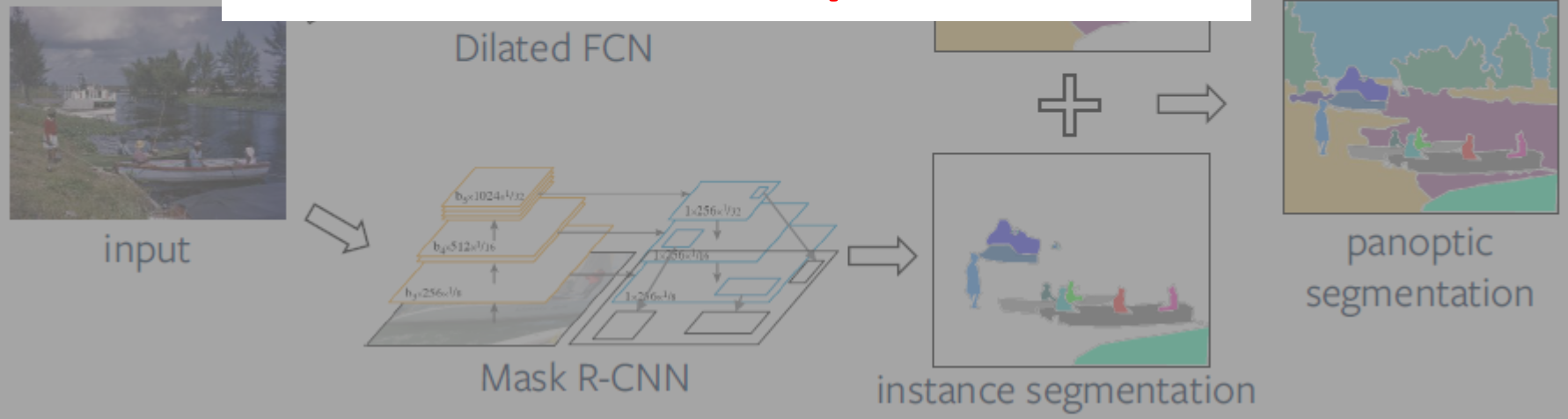


Basic idea

The most straightforward method to produce panoptic segmentation is combined two independent networks:

Inefficient!

Hard to improve!

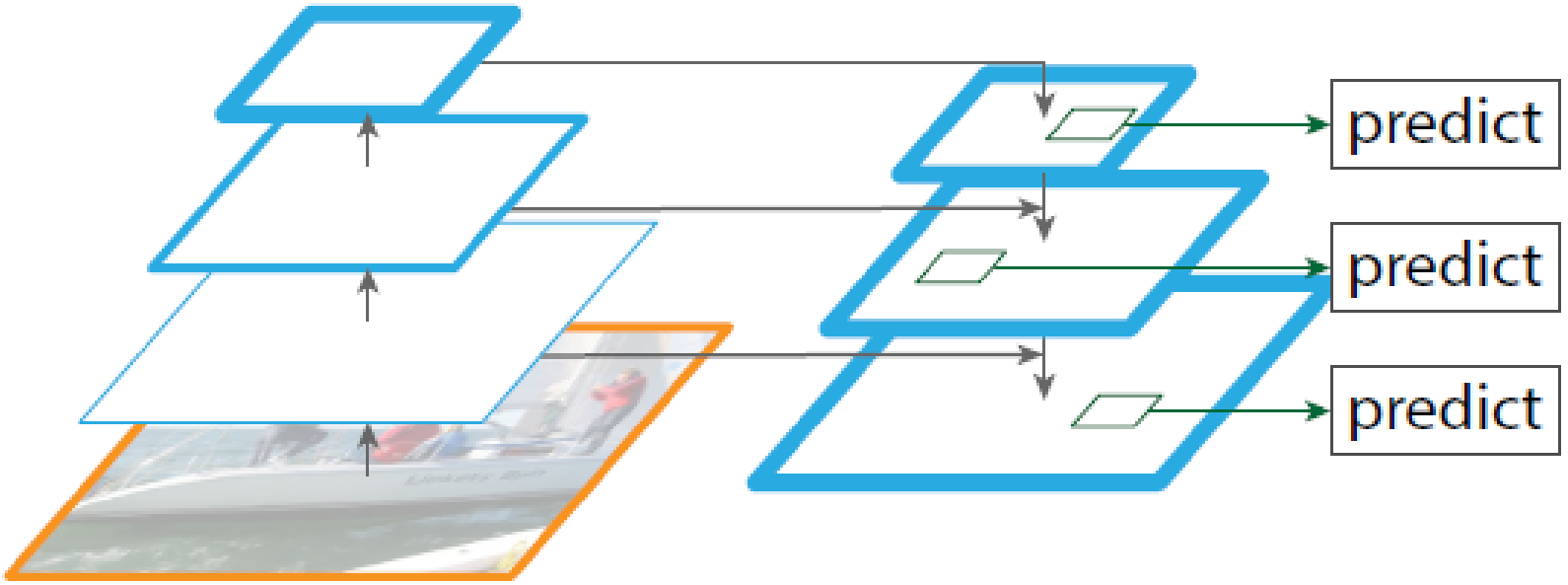


Motivation

- Propose a simple single-network baseline for the recently introduced joint task of *panoptic segmentation.*

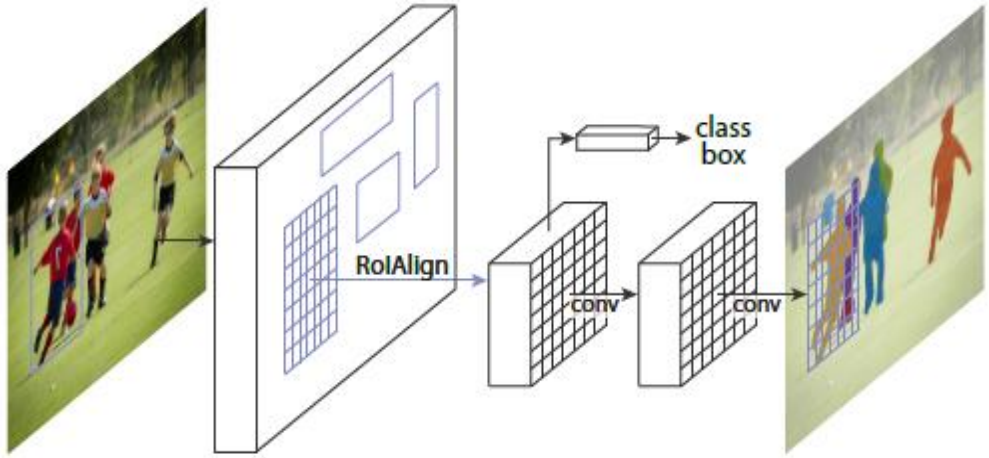
Simultaneously generating **region-based outputs** for *instance segmentation* and **dense-pixel outputs** for *semantic segmentation*.

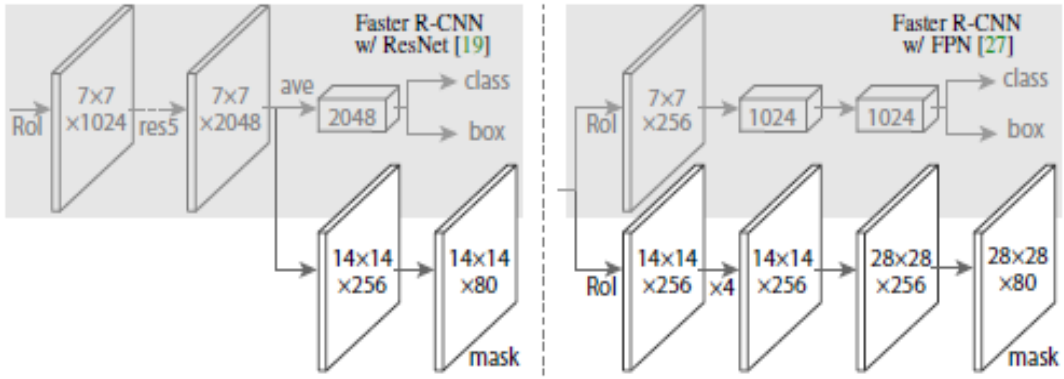
Backbone: Feature Pyramid Network



Instance segmentation branch

Generates bounding box regression , classification and a segmentation mask output.

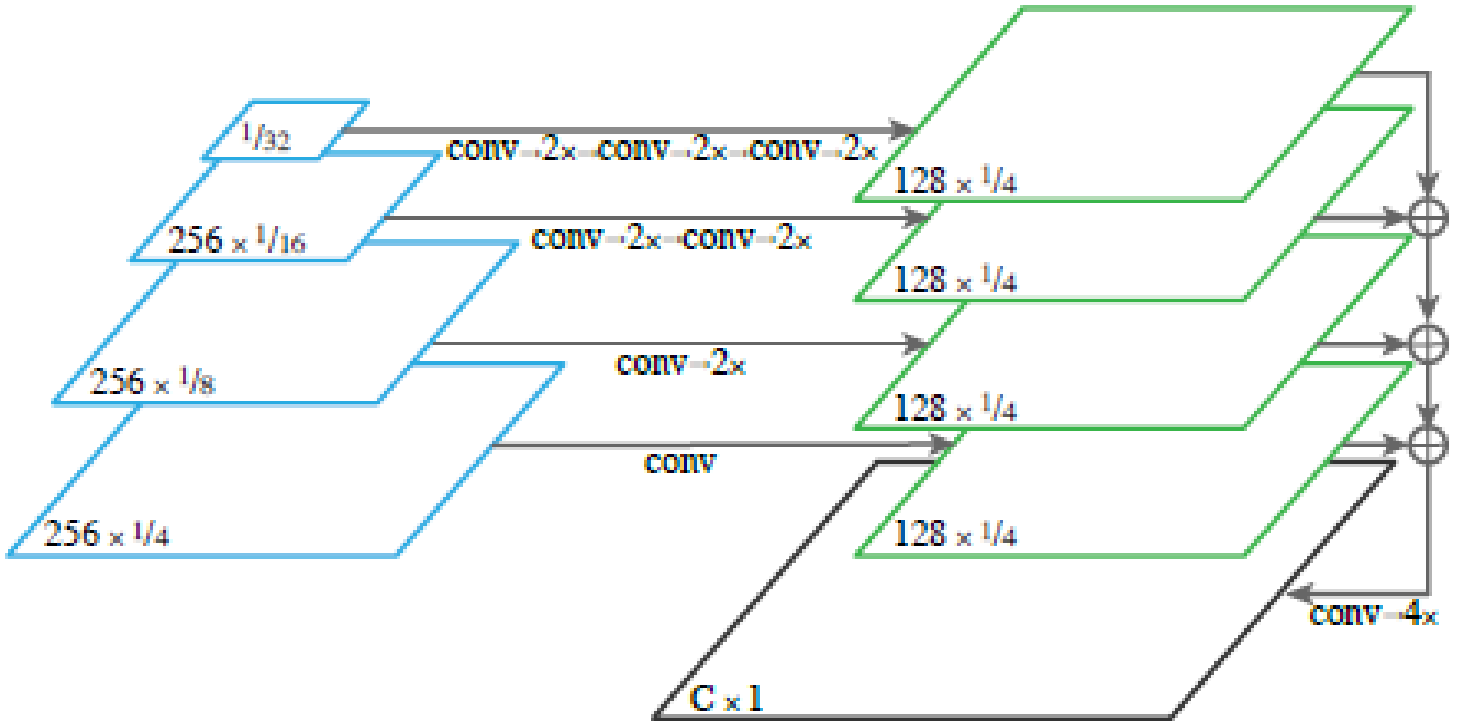




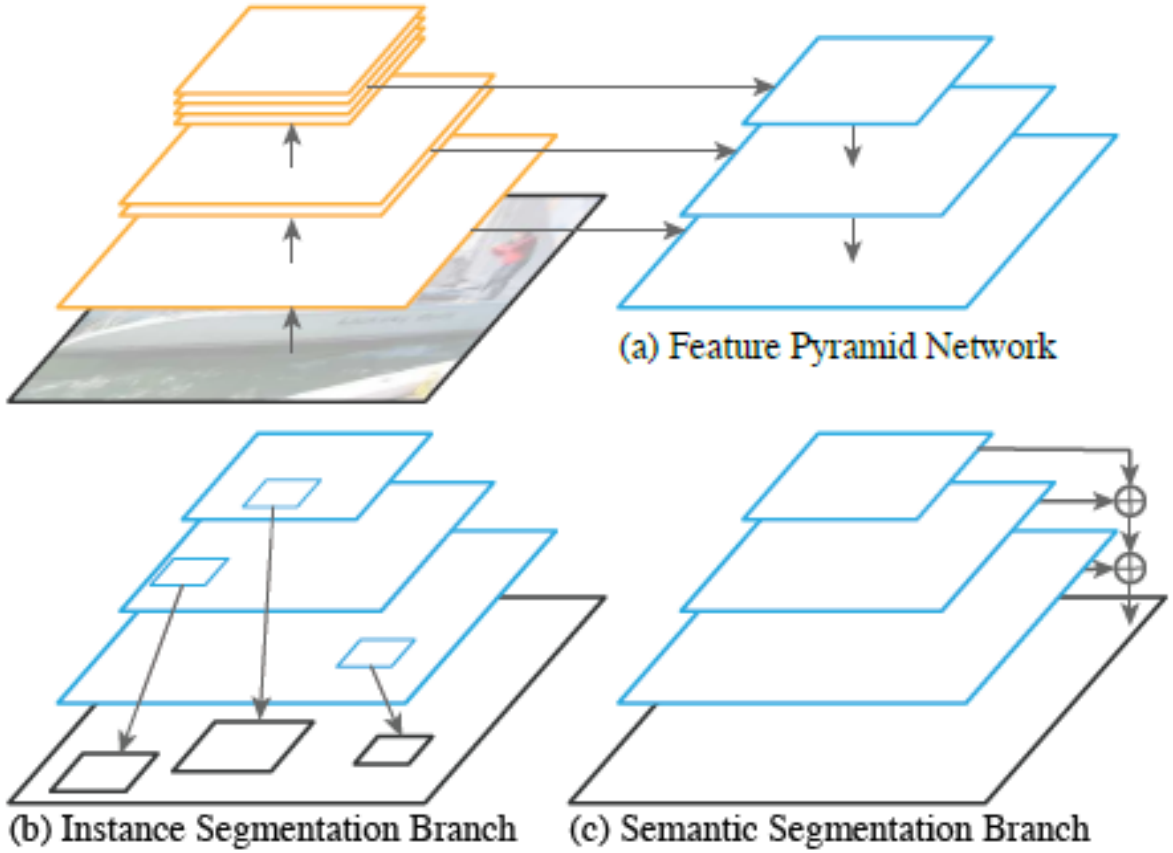
Instance segmentation head architecture

Semantic segmentation branch

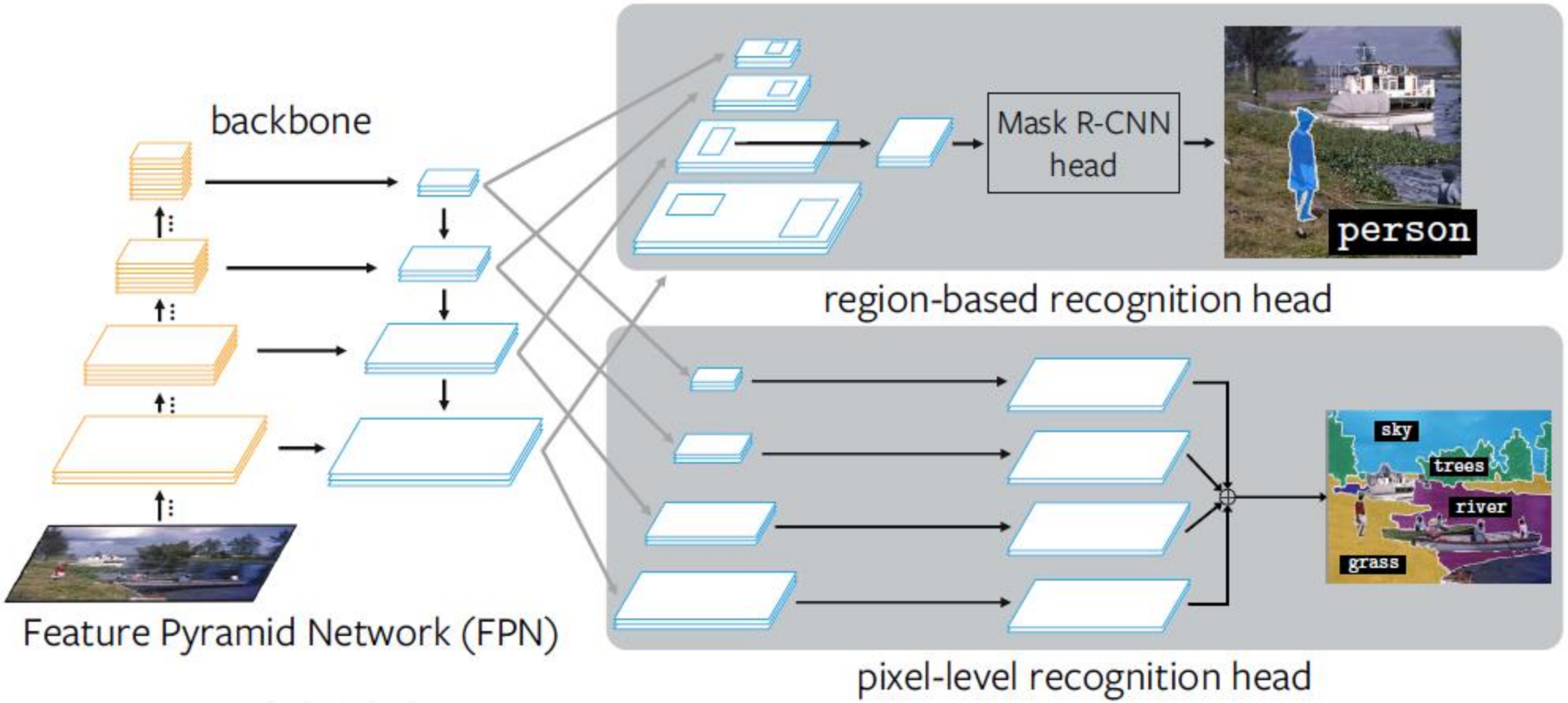
Merges the information from all levels of the FPN pyramid into a single output.



General architecture: Panoptic FPN



General architecture: Panoptic FPN



Joint training

- **Instance segmentation branch:**

$$L_c \rightarrow Classification\ loss$$
$$L_b \rightarrow Bounding\ box\ loss$$
$$L_m \rightarrow Mask\ loss$$

- **Semantic segmentation:**

$$L_s \rightarrow Semantic\ segmentation\ loss$$

- **Final Loss:**

$$L = \lambda_i(L_c + L_b + L_m) + \lambda_s L_s$$

Results for Semantic Segmentation

	backbone	mIoU	FLOPs	memory
DeeplabV3 [11]	ResNet-101-D8	77.8	1.9	1.9
PSANet101 [59]	ResNet-101-D8	77.9	2.0	2.0
Mapillary [5]	WideResNet-38-D8	79.4	4.3	1.7
DeeplabV3+ [12]	X-71-D16	79.6	0.5	1.9
Semantic FPN	ResNet-101-FPN	77.7	0.5	0.8
Semantic FPN	ResNeXt-101-FPN	79.1	0.8	1.4

Cityscapes semantic FPN

	backbone	mIoU	fIoU
Vllab [13]	Stacked Hourglass	12.4	38.8
DeepLab VGG16 [10]	VGG-16	20.2	47.5
Oxford [4]	ResNeXt-101	24.1	50.6
G-RMI [19]	Inception ResNet v2	26.6	51.9
Semantic FPN	ResNeXt-152-FPN	28.8	55.7

COCO-stuff 2017 challenge

Width	Cityscapes	COCO
64	74.1	39.6
128	74.5	40.2
256	74.6	40.1

Aggr.	Cityscapes	COCO
Sum	74.5	40.2
Concat	74.4	39.9

Ablation

Results for multi-task training

λ_s	mIoU	AP	AP_{50}	AP_{75}	PQ^{Th}
0.0	-	33.9	55.6	35.9	46.6
0.1	37.2	**34.0**	55.6	**36.0**	**46.8**
0.25	39.6	33.7	55.3	35.5	46.1
0.5	41.0	33.3	54.9	35.2	45.9
0.75	41.1	32.6	53.9	34.6	45.0
1.0	41.5	32.1	53.2	33.6	44.6
		+0.1	*+0.0*	*+0.1*	*+0.2*

(a) Panoptic FPN on COCO for **instance** segmentation ($\lambda_i = 1$).

λ_s	mIoU	AP	AP_{50}	PQ^{Th}
0.0	-	32.2	58.7	51.3
0.1	68.3	32.5	59.2	52.9
0.25	71.8	32.8	59.6	52.7
0.5	72.0	32.7	59.5	52.9
0.75	73.4	32.8	58.8	52.3
1.0	74.2	**33.2**	**59.7**	**52.4**
		+1.0	*+1.0*	*+1.1*

(b) Panoptic FPN on Cityscapes for **instance** segmentation ($\lambda_i = 1$).

λ_i	AP	mIoU	fIoU	PQ^{St}
0.0	-	40.2	67.2	27.9
0.1	20.1	40.6	67.5	28.4
0.25	25.5	41.0	67.8	28.6
0.5	29.2	41.3	68.0	28.9
0.75	30.8	41.1	68.2	28.9
1.0	32.1	**41.5**	**68.2**	**29.0**
		+1.2	*+1.0*	*+1.1*

(c) Panoptic FPN on COCO for **semantic** segmentation ($\lambda_s = 1$).

λ_i	AP	mIoU	iIoU	PQ^{St}
0.0	-	74.5	55.8	62.4
0.1	27.4	75.3	57.6	62.5
0.25	30.5	**75.5**	**58.3**	**62.5**
0.5	32.0	75.0	58.2	62.2
0.75	32.6	74.3	58.2	61.7
1.0	33.2	74.2	57.4	61.4
		+1.0	*+2.5*	*+0.1*

(d) Panoptic FPN on Cityscapes for **semantic** segmentation ($\lambda_s = 1$).

Results for panoptic segmentation in COCO and Cityscapes: **Panoptic R50-FPN vs. R50-FPN×2**

	backbone	AP	PQ^{Th}	mIoU	PQ^{St}	PQ
COCO	R50-FPN×2	33.9	46.6	40.2	27.9	39.2
	R50-FPN	33.3	45.9	41.0	28.7	39.0
		-0.6	*-0.7*	*+0.8*	*+0.8*	*-0.2*
Cityscapes	R50-FPN×2	32.2	51.3	74.5	62.4	57.7
	R50-FPN	32.0	51.6	75.0	62.2	57.7
		-0.2	*+0.3*	*+0.5*	*-0.2*	*+0.0*

th: thing , st : stuff, PQ : panoptic quality. ResNet-50 backbone

Results for panoptic segmentation in COCO and Cityscapes .

	backbone	AP	PQ^{Th}	mIoU	PQ^{St}	PQ
COCO	R50-FPN×2	33.9	46.6	40.2	27.9	39.2
	R101-FPN	35.2	47.5	42.1	29.5	40.3
		+1.3	*+0.9*	*+1.9*	*+1.6*	*+1.1*
Cityscapes	R50-FPN×2	32.2	51.3	74.5	62.4	57.7
	R101-FPN	33.0	52.0	75.7	62.5	58.1
		+0.8	*+0.7*	*+1.3*	*+0.1*	*+0.4*

th: thing , st : stuff, PQ : panoptic quality. ResNet-50 and ResNet -101 backbones.

Ablation study

	loss	AP	PQ^{Th}	mIoU	PQ^{St}	PQ
COCO	alternate	31.7	43.9	40.2	28.0	37.5
	combine	33.3	45.9	41.0	28.7	39.0
		+1.6	*+2.0*	*+0.8*	*+0.7*	*+1.5*
Cityscapes	alternate	32.0	51.4	74.3	61.3	57.4
	combine	32.0	51.6	75.0	62.2	57.7
		+0.0	*+0.2*	*+0.7*	*+0.9*	*+0.3*

Grouped FPN study

	FPN	AP	PQ^{Th}	mIoU	PQ^{St}	PQ
COCO	original	33.3	45.9	41.0	28.7	39.0
	grouped	33.1	45.7	41.2	28.4	38.8
		-0.2	*-0.2*	*+0.2*	*-0.3*	*-0.2*
Cityscapes	original	32.0	51.6	75.0	62.2	57.7
	grouped	32.0	51.8	75.3	61.7	57.5
		+0.0	*+0.2*	*+0.3*	*-0.5*	*-0.2*

Comparison to the state-of-the-art

COCO test-dev

	PQ	PQ^{Th}	PQ^{St}
Artemis	16.9	16.8	17.0
LeChen	26.2	31.0	18.9
MPS-TU Eindhoven [16]	27.2	29.6	23.4
MMAP-seg	32.1	38.9	22.0
Panoptic FPN	**40.9**	**48.3**	**29.7**

Cityscapes

	coarse	PQ	PQ^{Th}	PQ^{St}	mIoU	AP
DIN [1, 34]	✓	53.8	42.5	62.1	**80.1**	28.6
Panoptic FPN		**58.1**	**52.0**	**62.5**	75.7	**33.0**

Results in COCO and Cityscapes.

COCO – top , Cityscapes- bottom *ResNet -101 FPN network*

Results in COCO and Cityscapes.

COCO – top , Cityscapes- bottom ResNet -101 FPN network

Main contributions and conclusion

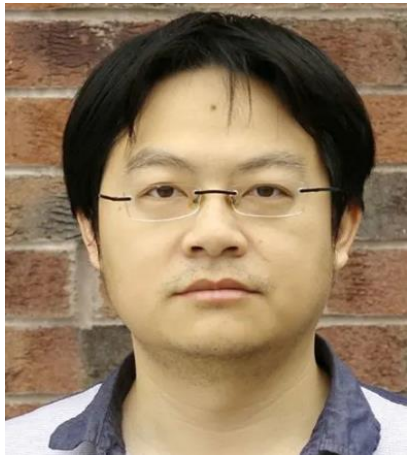
- They created a robust and accurate baseline for panoptic segmentation.
- The method starts with Mask R-CNN with FPN and adds to it a lightweight semantic segmentation branch for dense-pixel prediction.

UPSNet : A Unified Panoptic Segmentation Network.

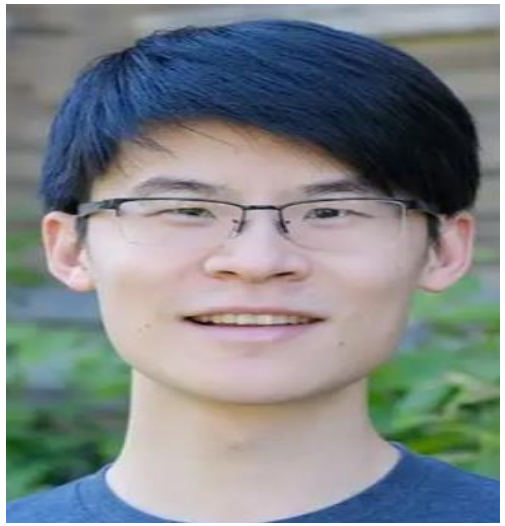
Yuwen Xiong

Renjie Liao

Hengshuang Zhao



Rui Hu



Min Bai

Ersin Yumer

Raquel Urtasun

Motivation

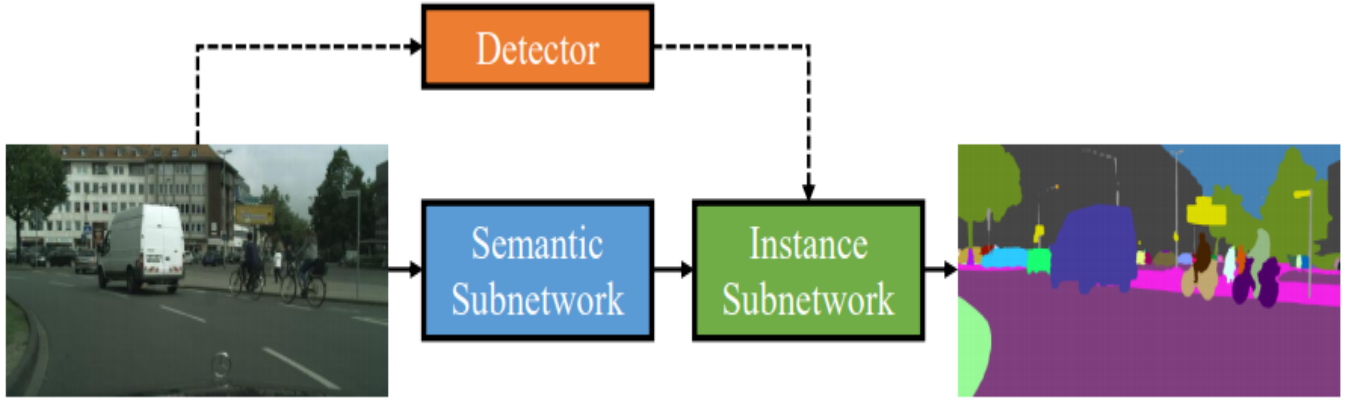
Due to the **new task** of **panoptic segmentation,** the autors propose a **unified network** which defers with some of the previous work for solving the panoptic segmentation task.

Related work.

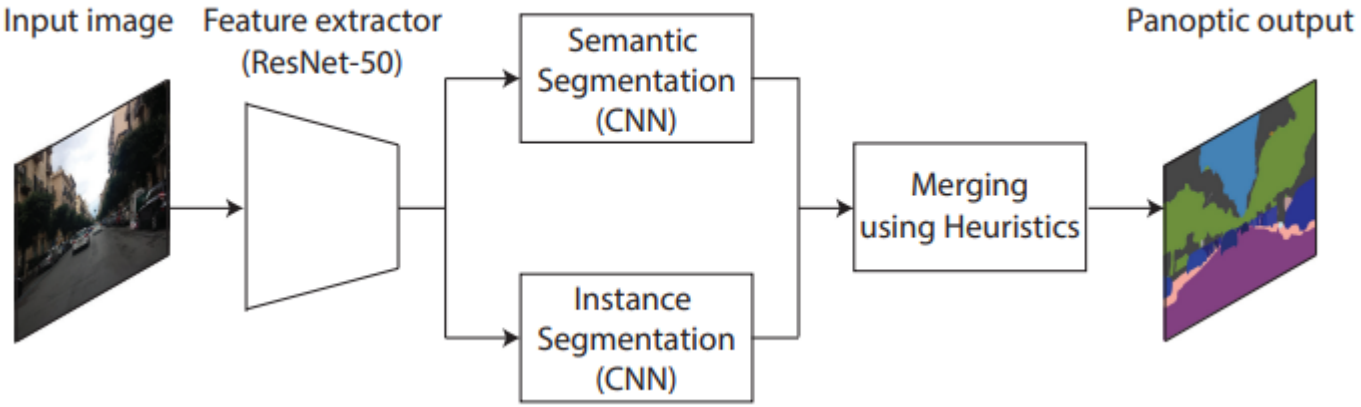
What is the problem with some methods that adress the panoptic segmentation task?

- The problem is that some methods solve the task using **two different networks**, one for semantic segmentation and the other for instance segmentation.

Related work: methods which use separated networks.

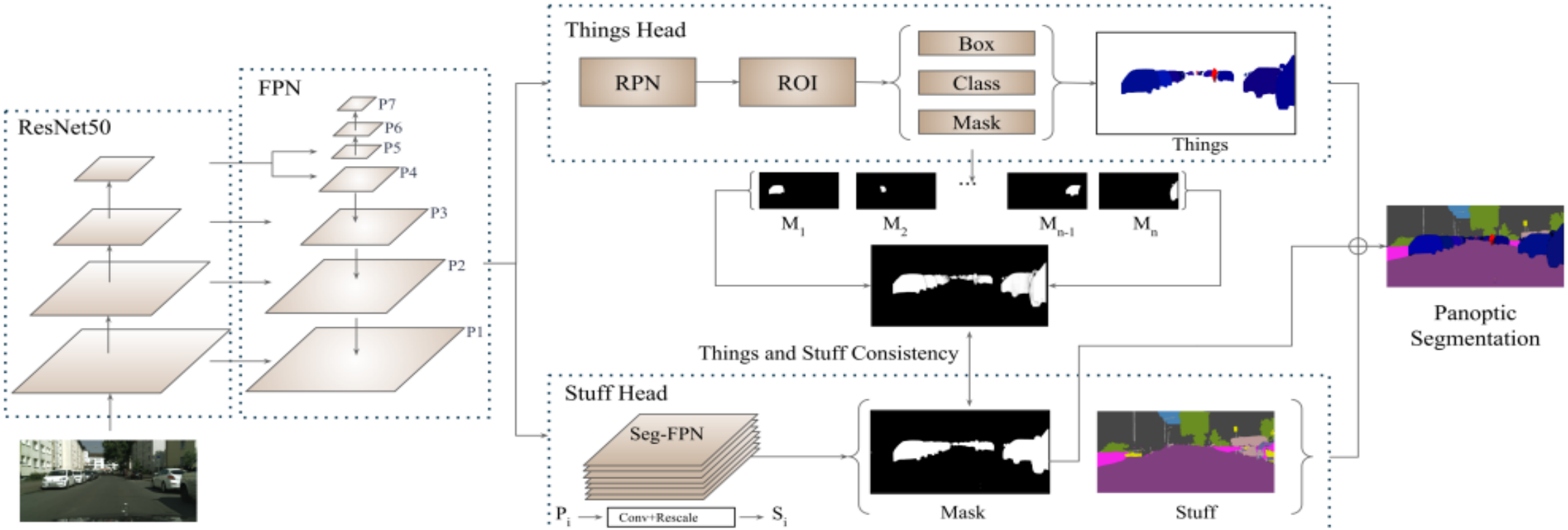


Qizhu Li et.al (2019). Weakly- and Semi-Supervised Panoptic Segmentation



Geus.D et.al (2019). Panoptic Segmentation with a Joint Semantic and Instance Segmentation Network.

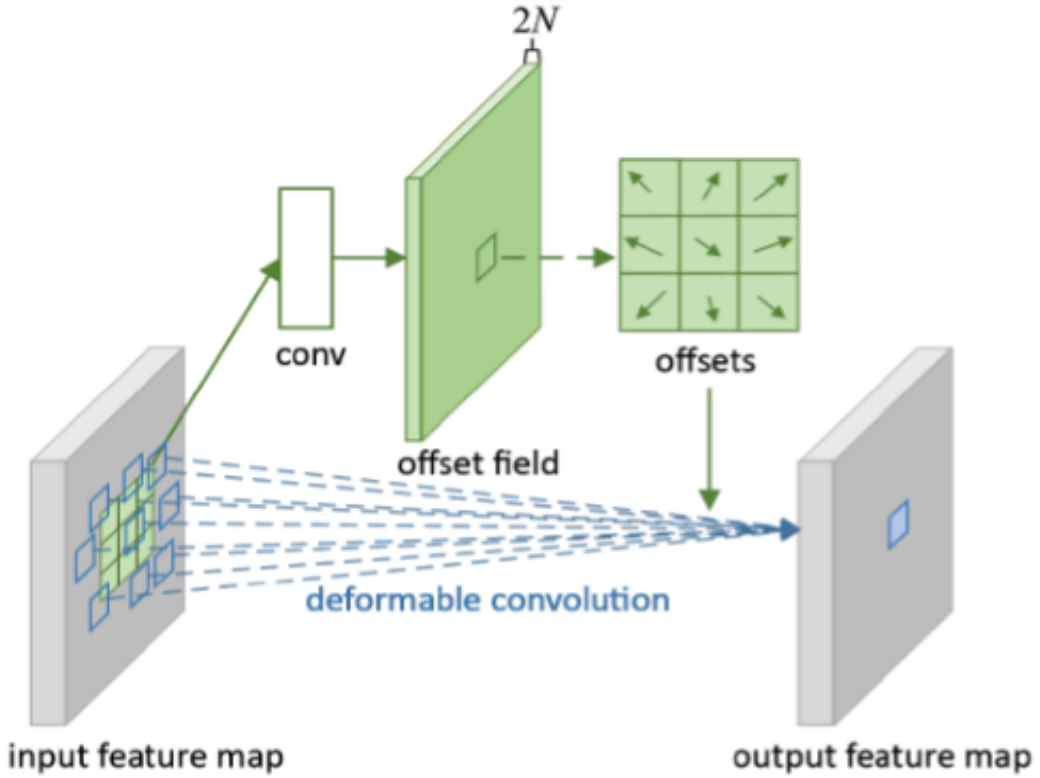
Related work: TASCNet



Qizhu Li et.al (2019). Learning to Fuse Things and Stuff

Important concept.

- **Deformable convolution**: developed by Microsoft Research Asia (MSRA)



Regular convolution

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n)$$

Deformable convolution

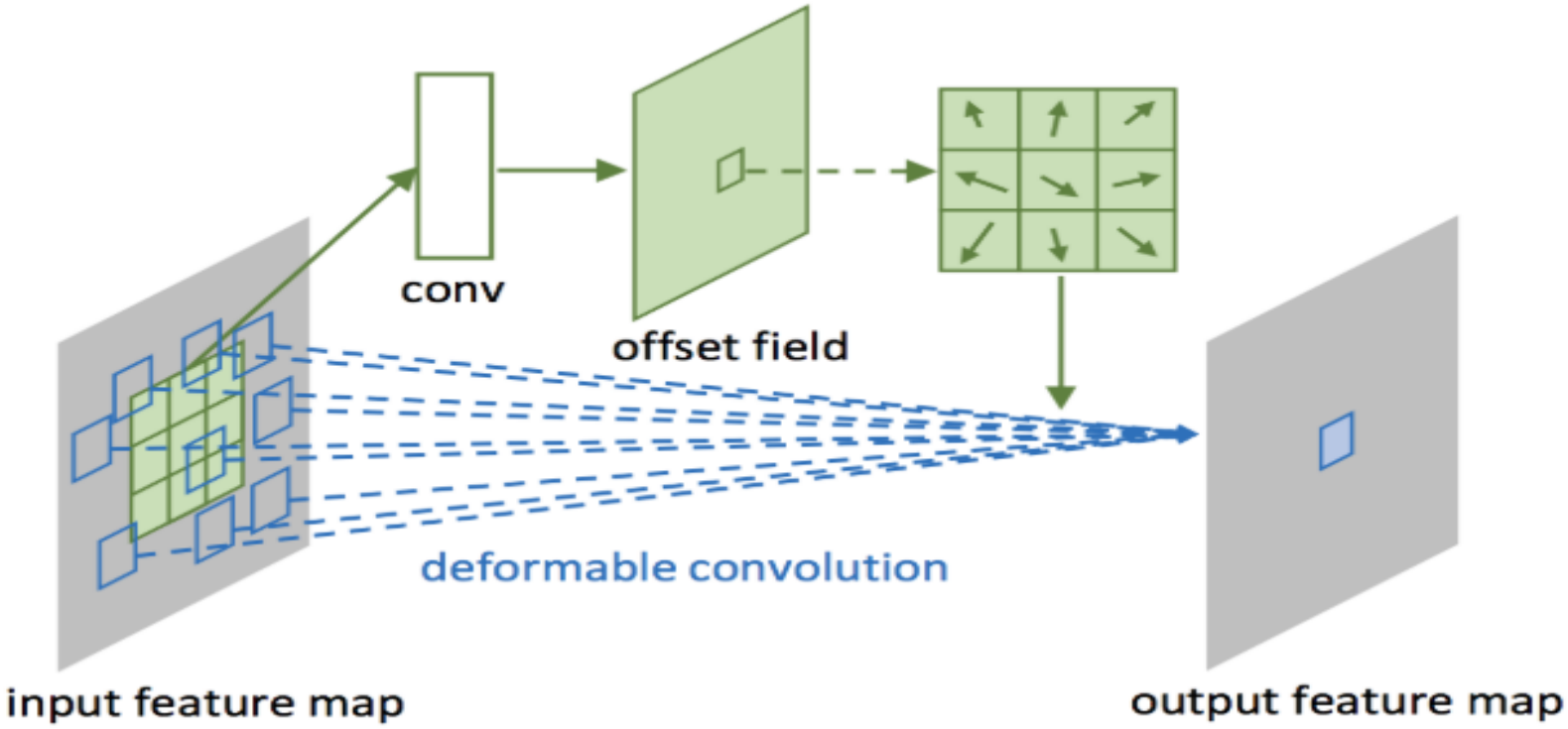
$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n)$$

where $\Delta\mathbf{p}_n$ is generated by a sibling branch of regular convolution

Deformable Convolution

Important concept.

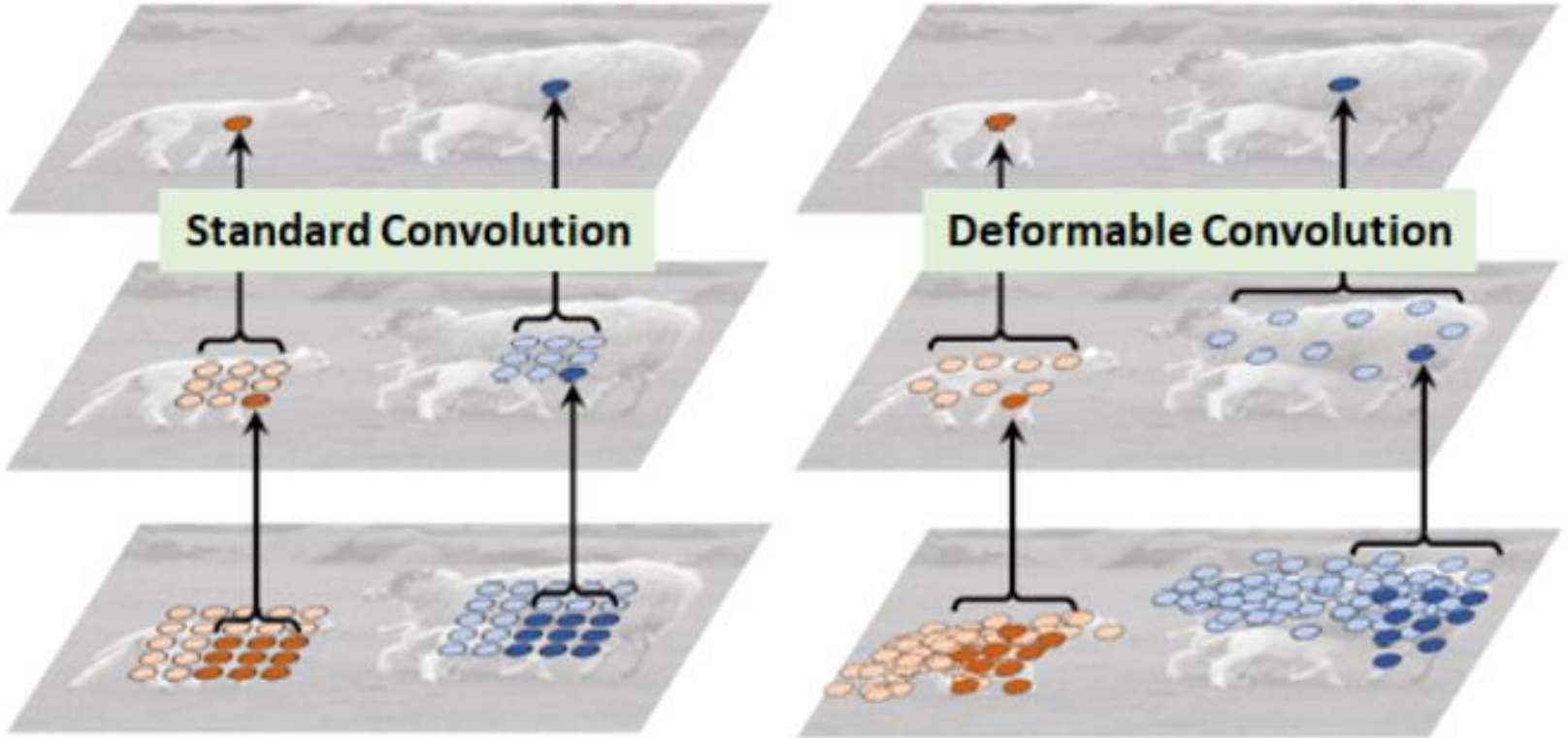
- **Deformable convolution** – example of a 3x3 deformable convolution.



Dynamic and learnable receptive field

Important concept.

- **Deformable convolution** – example.



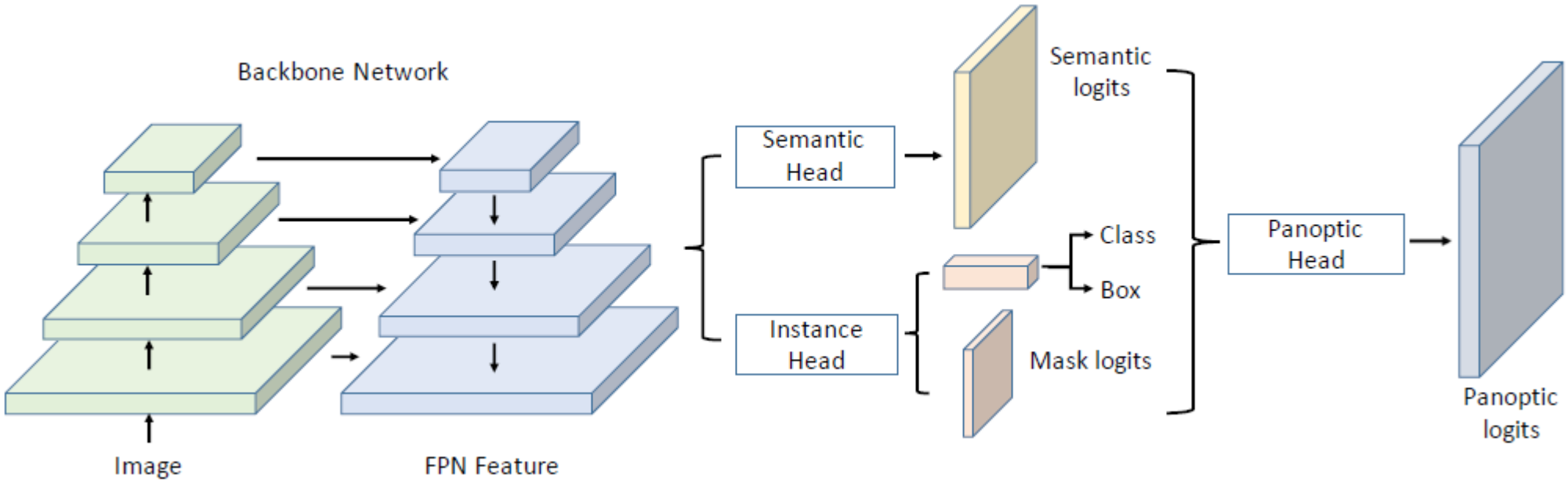
Standard Convolution (Left), Deformable Convolution (Right)

Dynamic and learnable receptive field

UPSNet Architecture

Backbone

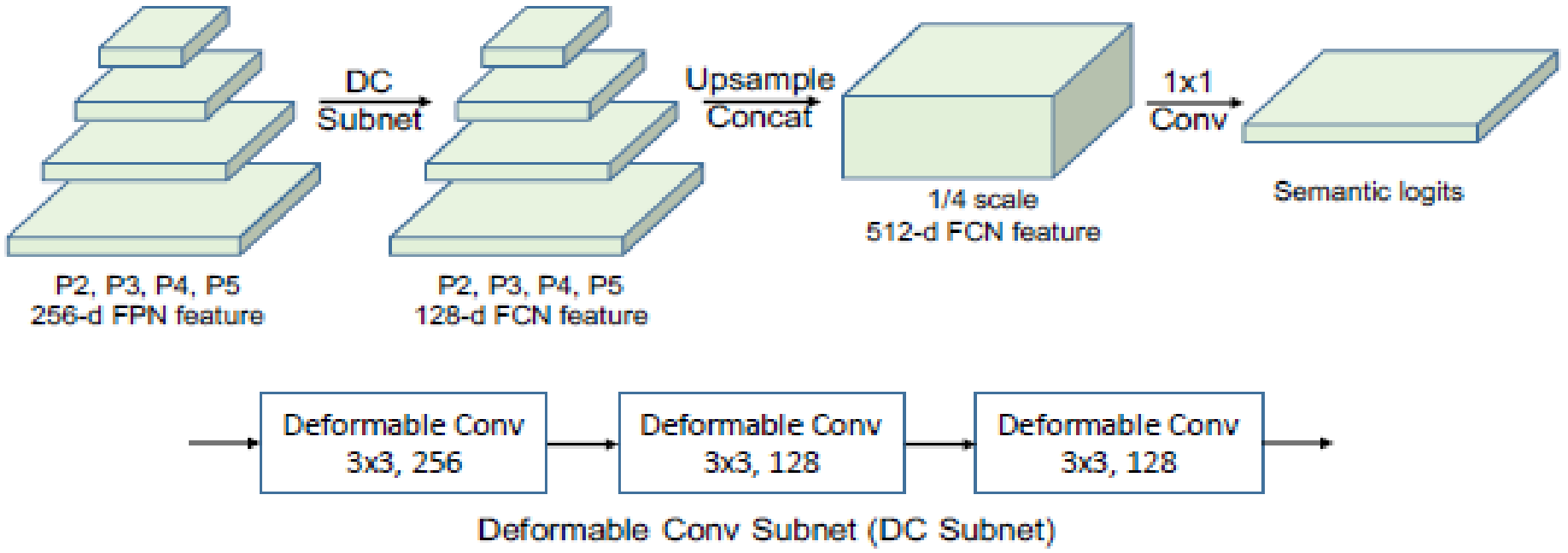
- Deep Residual network for feature extraction (ResNet)
- Feature Pyramid Network (FPN)



Overall architecture of UPSNet.

UPSNet Architecture

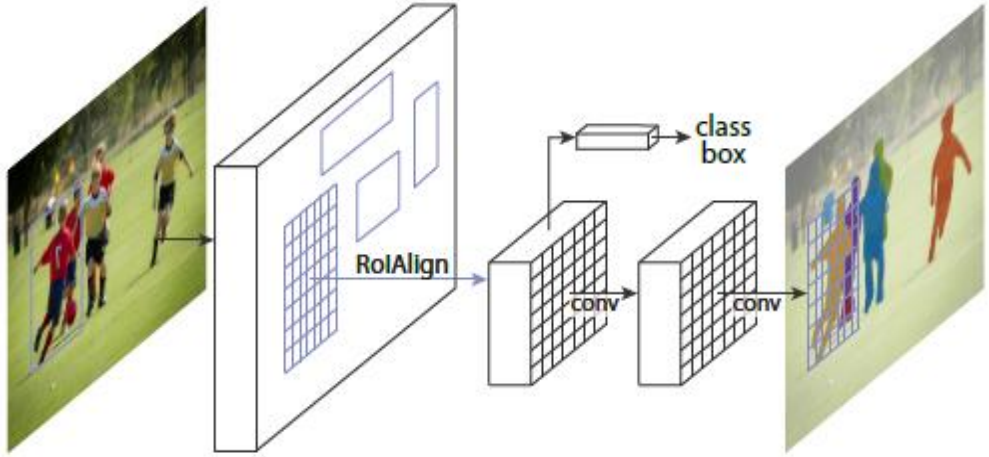
- **Semantic segmentation head** : Performs deformable convolution on the feature maps which came from the FPN.

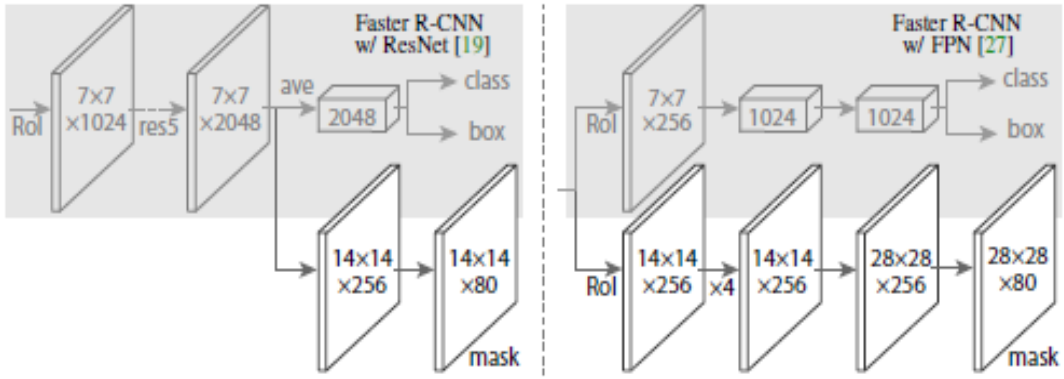


Semantic segmentation head architecture

UPSNet Architecture

Instance segmentation head: generates bounding box regression , classification and a segmentation mask output.

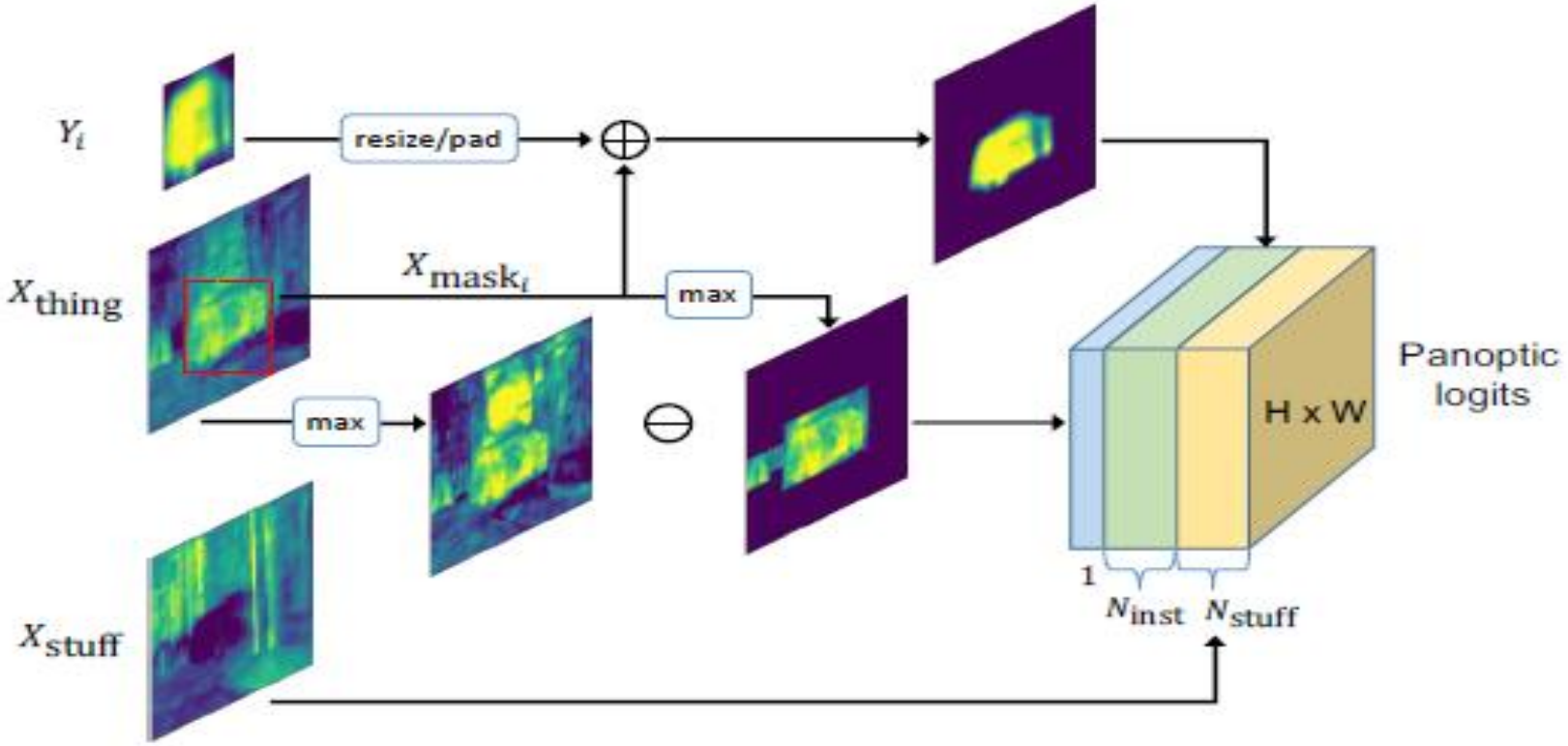




Instance segmentation head architecture

UPSNet Architecture

- **Panoptic segmentation head:** Takes the logits from the two previous heads to generate panoptic logits.



Panoptic segmentation head architecture.

No heuristics need, parameter free head .

Metric : Panoptic Quality.

$$PQ = \underbrace{\frac{\sum_{(p,g)\in TP} IoU(p,g)}{|TP|}}_{\text{Segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{Recognition quality (RQ)}}$$

UPSNet Architecture

- **Unknown prediction :** It´s purpose is to classify a pixel in a “unknown” class instead of making a wrong prediction of it.

- Takes in count the **false negatives (FN)** and the **false postives (FP)** which affect the Panoptic Quality.

- Increasing the **(FN)** or the **(FP)** decreases de Recognition quality which therefore decrease the Panoptic Quality.

$$\underbrace{\frac{|\mathrm{TP}|}{|\mathrm{TP}| + \frac{1}{2}|\mathrm{FP}| + \frac{1}{2}|\mathrm{FN}|}}_{\mathrm{RQ}}$$

Loss functions.

UPSNet model consider 8 loss functions which are:

- **Semantic segmentation head:**

1) Whole image pixel-wise classification loss.

2) RoI bassed pixel-wise classification loss.

- **Panoptic segmentation head**:

3) Whole image pixel-wise classification loss.

- **Instance segmentation head:**

4) Box classification loss.

5) Box regression loss.

6) Mask segmentation loss.

- **RPN:**

7) Box classification loss.

8) Box regression loss.

Inference

Most important points to take in count:

- “Mask pruning process” to determine **which mask** is going to be used for the **panoptic logits**.

- Non-maximum suppression , IoU treshold of 0.5 , why?

- To **filter** some of the **overlapping boxes** which came from the instance segmentation head.

- Finally they **sort the class probabilities** and keep those with a probability larger than 0.6 from the remaining bounding boxes from the step before.

Results for panoptic segmentation in COCO.

Models	PQ	SQ	RQ	PQ^{Th}	PQ^{St}	mIoU	AP
JSIS-Net [10]	26.9	72.4	35.7	29.3	23.3	-	-
RN50-MR-CNN	38.6	76.4	47.5	46.2	27.1	-	-
MR-CNN-PSP	41.8	**78.4**	51.3	47.8	32.8	53.9	34.2
Ours	**42.5**	78.0	**52.4**	**48.5**	**33.4**	**54.3**	**34.3**
Multi-scale	PQ	SQ	RQ	PQ^{Th}	PQ^{St}	mIoU	AP
MR-CNN-PSP-M	42.2	78.5	51.7	47.8	33.8	55.3	34.2
Ours-M	**43.2**	**79.2**	**52.9**	**49.1**	**34.1**	**55.8**	**34.3**

th: thing , st : stuff, PQ : panoptic quality , SQ: segmentation quality, RQ : recognition quality.

Results for panoptic segmentation in COCO.

Models	backbone	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	PQ^{St}	SQ^{St}	RQ^{St}
Megvii (Face++)	ensemble model	53.2	83.2	62.9	62.2	85.5	72.5	39.5	79.7	48.5
Caribbean	ensemble model	46.8	80.5	57.1	54.3	81.8	65.9	35.5	78.5	43.8
PKU_360	ResNeXt-152-FPN	46.3	79.6	56.1	58.6	83.7	69.6	27.6	73.6	35.6
JSIS-Net [10]	ResNet-50	27.2	71.9	35.9	29.6	71.6	39.4	23.4	72.3	30.6
AUNet [20]	ResNeXt-152-FPN	46.5	**81.0**	56.1	**55.9**	**83.7**	**66.3**	32.5	77.0	40.7
Ours	ResNet-101-FPN	**46.6**	80.5	**56.9**	53.2	81.5	64.6	**36.7**	**78.9**	**45.3**

th: thing , st : stuff, PQ : panoptic quality , SQ: segmentation quality, RQ : recognition quality.

Results for panoptic segmentation in Cityscapes.

Models	PQ	SQ	RQ	PQ^{Th}	PQ^{St}	mIoU	AP
Li *et al.* [19]	53.8	-	-	42.5	62.1	71.6	28.6
MR-CNN-PSP	58.0	79.2	71.8	52.3	62.2	75.2	32.8
TASCNet [18]	55.9	-	-	50.5	59.8	-	-
Ours	**59.3**	**79.7**	**73.0**	**54.6**	**62.7**	**75.2**	**33.3**
TASCNet-COCO [18]	59.2	-	-	56.0	61.5	-	-
Ours-COCO	**60.5**	**80.9**	**73.5**	**57.0**	**63.0**	**77.8**	**37.8**
Multi-scale	PQ	SQ	RQ	PQ^{Th}	PQ^{St}	mIoU	AP
Kirillov *et al.* [17]	61.2	80.9	74.4	54.0	**66.4**	**80.9**	36.4
MR-CNN-PSP-M	59.2	79.7	73.0	52.3	64.2	76.9	32.8
Ours-M	60.1	80.3	73.5	55.0	63.7	76.8	33.3
Ours-101-M-COCO	**61.8**	**81.3**	**74.8**	**57.6**	64.8	79.2	**39.0**

th: thing , st : stuff, PQ : panoptic quality , SQ: segmentation quality, RQ : recognition quality.

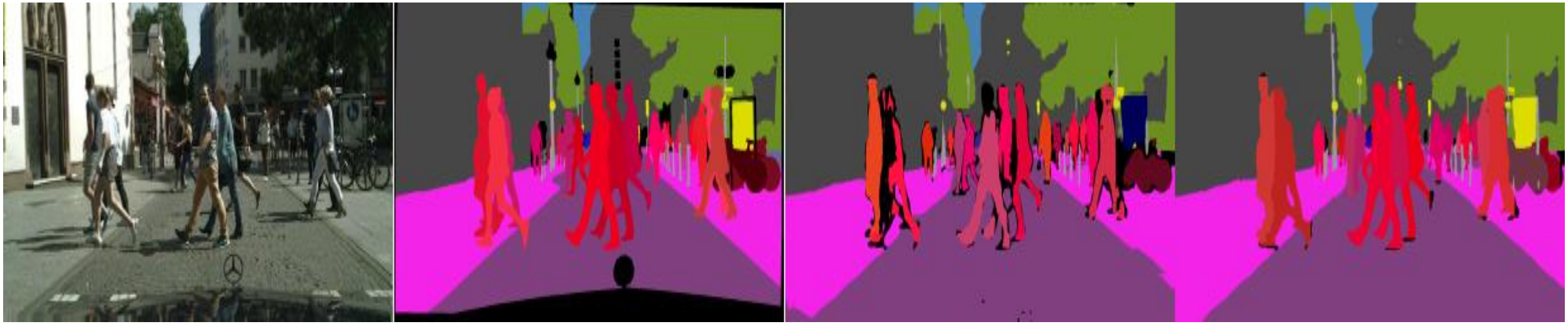
Results for panoptic segmentation in their dataset.

Models	PQ	SQ	RQ	PQ^{Th}	PQ^{St}	mIoU	AP
MR-CNN-PSP	45.5	77.1	57.6	40.1	48.7	70.5	29.6
Ours	**47.1**	**77.1**	**59.4**	**43.8**	**49.0**	**70.8**	**30.4**

th: thing , st : stuff, PQ : panoptic quality , SQ: segmentation quality, RQ : recognition quality.

Results in COCO and Cityscapes.

Original image | Ground truth | Combined method from the Panoptic segmentation paper | Prediction



Original image | Ground truth | Combined method from the Panoptic segmentation paper | Prediction

Results in their dataset – similar to Cityscapes .

Original image

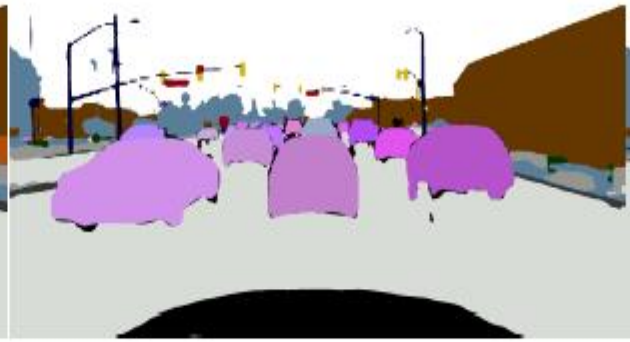
Ground truth

Combined method from the Panoptic segmentation paper

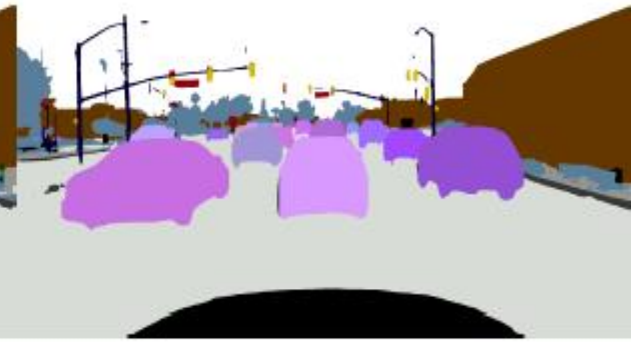
Prediction

Original image

Ground truth



Combined method from the Panoptic segmentation paper



Prediction

Main contributions and conclusion

- Ligthweight panoptic segmentation head
- **Unified network** for addressing the new panoptic segmentation task
- **Unknown class** for avoiding problems between instance and semantic segmentation
- Results from the **multitask learning**.

Tutorial

- Instance Segmentation
- Panoptic Segmentation

Tutorial

- Instance Segmentation

Tutorial – Instance segmentation

- Dataset

```
class PennFudanDataset(object):
    def __init__(self, root, transforms):
        self.root = root
        self.transforms = transforms
        # load all image files, sorting them to
        # ensure that they are aligned
        self.imgs = list(sorted(os.listdir(os.path.join(root, "PNGImages"))))
        self.masks = list(sorted(os.listdir(os.path.join(root, "PedMasks"))))

    def __getitem__(self, idx):
        # load images ad masks
        img_path = os.path.join(self.root, "PNGImages", self.imgs[idx])
        mask_path = os.path.join(self.root, "PedMasks", self.masks[idx])
        img = Image.open(img_path).convert("RGB")
        # note that we haven't converted the mask to RGB,
        # because each color corresponds to a different instance
        # with 0 being background
        mask = Image.open(mask_path)
        mask = np.array(mask)
        np.save('mask.npy',mask)
        # instances are encoded as different colors
        obj_ids = np.unique(mask)
        # first id is the background, so remove it
        obj_ids = obj_ids[1:]
```

Tutorial – Instance segmentation

- Dataset

```
num_objs = len(obj_ids)
boxes = []
for i in range(num_objs):
    pos = np.where(masks[i])
    xmin = np.min(pos[1])
    xmax = np.max(pos[1])
    ymin = np.min(pos[0])
    ymax = np.max(pos[0])
    boxes.append([xmin, ymin, xmax, ymax])

boxes = torch.as_tensor(boxes, dtype=torch.float32)
# there is only one class
labels = torch.ones((num_objs,), dtype=torch.int64)
masks = torch.as_tensor(masks, dtype=torch.uint8)

image_id = torch.tensor([idx])
area = (boxes[:, 3] - boxes[:, 1]) * (boxes[:, 2] - boxes[:, 0])
# suppose all instances are not crowd
iscrowd = torch.zeros((num_objs,), dtype=torch.int64)

target = {}
target["boxes"] = boxes
target["labels"] = labels
target["masks"] = masks
target["image_id"] = image_id
target["area"] = area
target["iscrowd"] = iscrowd

if self.transforms is not None:
    img, target = self.transforms(img, target)

return img, target
```

Tutorial – Instance segmentation

- Model.

```
def get_model_instance_segmentation(num_classes):
    # load an instance segmentation model pre-trained pre-trained on COCO
    model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)

    # get number of input features for the classifier
    in_features = model.roi_heads.box_predictor.cls_score.in_features
    # replace the pre-trained head with a new one
    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

    # now get the number of input features for the mask classifier
    in_features_mask = model.roi_heads.mask_predictor.conv5_mask.in_channels
    hidden_layer = 256
    # and replace the mask predictor with a new one
    model.roi_heads.mask_predictor = MaskRCNNPredictor(in_features_mask,
                                                       hidden_layer,
                                                       num_classes)

    return model
```

Tutorial – Instance segmentation

- Dataloader

```
def main():
    # train on the GPU or on the CPU, if a GPU is not available
    device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')

    # use our dataset and defined transformations
    dataset = PennFudanDataset('PennFudanPed', get_transform(train=True))
    dataset_test = PennFudanDataset('PennFudanPed', get_transform(train=False))
    #pdb.set_trace()
    # split the dataset in train and test set
    #torch.manual_seed(1)
    indices = torch.randperm(len(dataset)).tolist()
    dataset = torch.utils.data.Subset(dataset, indices[:-50])
    dataset_test = torch.utils.data.Subset(dataset_test, indices[-50:])

    # define training and validation data loaders
    data_loader = torch.utils.data.DataLoader(
        dataset, batch_size=2, shuffle=True, num_workers=4,
        collate_fn=utils.collate_fn)

    data_loader_test = torch.utils.data.DataLoader(
        dataset_test, batch_size=1, shuffle=False, num_workers=4,
        collate_fn=utils.collate_fn)
```

Tutorial – Instance segmentation

- Training

```
# get the model using our helper function
model = get_model_instance_segmentation(num_classes)

# move model to the right device
model.to(device)

# construct an optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005,
                            momentum=0.9, weight_decay=0.0005)
# and a learning rate scheduler
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
                                               step_size=3,
                                               gamma=0.1)


for epoch in range(num_epochs):
    # train for one epoch, printing every 10 iterations
    train_one_epoch(model, optimizer, data_loader, device, epoch, print_freq=10)
    # update the learning rate
    lr_scheduler.step()
    # evaluate on the test dataset
    evaluate(model, data_loader_test, device=device)
print("End")
```

Tutorial – Instance segmentation

- Single prediction

```
#Single prediction
# pick one image from the test set
img, _ = dataset_test[0]
# put the model in evaluation mode
model.eval()
with torch.no_grad():
    prediction = model([img.to(device)])
```

Homework

- Change the model to Mask-RCNN and Faster-RCNN using the detection models from torchvision.
- Report and discuss your qualitative results. Show the metrics and discuss based on them.

Tutorial

- Panoptic Segmentation

Tutorial- Panoptic Segmentation

- Installation & requirements:

  - Mask RCNN
  - apex
  - pycocotools

Requirements:

- PyTorch 10.0 from a nightly release. It **will not** work with 10.0 nor 10.0.1. Installation instructions can be found in https://pytorch.org/get-started/locally/. If you already have a diffent version, please unistall it befores re-install.
- torchvision from master
- cocoapi
- yacs
- scikit-image
- matplotlib
- GCC >= 4.9
- OpenCV version 3.4.3
  - You can run this line to install it: pip install opencv-python==3.4.3.18
- CUDA >= 9.0 (In case of running on the servers, the CUDA version is 10.0)

Tutorial- Panoptic Segmentation

- Imports

script: main.py

```
import torch
from torch import nn
import torch.nn.functional as Fx
import datetime
# Set up custom environment before nearly anything else is imported
# NOTE: this should be the first import (no not reorder)
from maskrcnn_benchmark.utils.env import setup_environment  # noqa F401 isort:ski
from maskrcnn_benchmark.data.build import *
from maskrcnn_benchmark.structures.bounding_box import BoxList
from maskrcnn_benchmark.structures.segmentation_mask import SegmentationMask
from maskrcnn_benchmark.modeling.detector import build_detection_model
from maskrcnn_benchmark.utils.checkpoint import DetectronCheckpointer
from maskrcnn_benchmark.structures.image_list import to_image_list
from maskrcnn_benchmark.modeling.roi_heads.mask_head.inference import Masker
from maskrcnn_benchmark import layers as L
from maskrcnn_benchmark.utils import cv2_util
from maskrcnn_benchmark.utils.miscellaneous import mkdir
from maskrcnn_benchmark.utils.logger import setup_logger
from maskrcnn_benchmark.utils.comm import synchronize, get_rank
from maskrcnn_benchmark.config import cfg
#from maskrcnn_benchmark.config import cfg
from maskrcnn_benchmark.data import make_data_loader
from maskrcnn_benchmark.solver import make_lr_scheduler
from maskrcnn_benchmark.solver import make_optimizer
from maskrcnn_benchmark.engine.inference import inference
from maskrcnn_benchmark.engine.trainer import do_train
from maskrcnn_benchmark.modeling.detector import build_detection_model
from maskrcnn_benchmark.utils.checkpoint import DetectronCheckpointer
from maskrcnn_benchmark.utils.collect_env import collect_env_info
from maskrcnn_benchmark.utils.comm import synchronize, get_rank
from maskrcnn_benchmark.utils.imports import import_file
```

Tutorial- Panoptic Segmentation

- Imports

  script: main.py

```
from maskrcnn_benchmark.data.datasets.evaluation import evaluate
from maskrcnn_benchmark.utils.comm import is_main_process, get_world_size
from maskrcnn_benchmark.utils.comm import all_gather
from maskrcnn_benchmark.utils.timer import Timer, get_time_str
from maskrcnn_benchmark.engine.inference import compute_on_dataset, _accumulate_predictions_
from maskrcnn_benchmark.data.datasets.evaluation.coco import coco_evaluation
from maskrcnn_benchmark.modeling.utils import cat
from maskrcnn_benchmark.structures.image_list import to_image_list
from maskrcnn_benchmark.modeling.backbone import build_backbone
from maskrcnn_benchmark.modeling.rpn.rpn import build_rpn
from maskrcnn_benchmark.modeling.roi_heads.roi_heads import build_roi_heads
from maskrcnn_benchmark.modeling.make_layers import make_conv3x3
from maskrcnn_benchmark.structures.image_list import to_image_list
from maskrcnn_benchmark.modeling.backbone import build_backbone
from maskrcnn_benchmark.modeling.rpn.rpn import build_rpn
from maskrcnn_benchmark.modeling.roi_heads.roi_heads import build_roi_heads
import torch.distributed as dist

from maskrcnn_benchmark.utils.comm import get_world_size
from maskrcnn_benchmark.utils.metric_logger import MetricLogger

from helper_functions import *
from dataset import *
from dataloader import *
```

Tutorial- Panoptic Segmentation

- Imports

  script: main.py

```
from PIL import Image
import json
import logging
import torch
import numpy as np
import skimage.draw as draw
import tempfile
from pycocotools.coco import COCO
import os
import sys
import random
import math
import re
import time
import cv2
import matplotlib
import matplotlib.pyplot as plt
from tqdm import tqdm

# Copyright (c) Facebook, Inc. and its affiliates. All Rights Reserved.
from torchvision import transforms as T
from torchvision.transforms import functional as F
```

Tutorial- Panoptic Segmentation

- Imports

  script: main.py

```
from PIL import Image
import json
import logging
import torch
import numpy as np
import skimage.draw as draw
import tempfile
from pycocotools.coco import COCO
import os
import sys
import random
import math
import re
import time
import cv2
import matplotlib
import matplotlib.pyplot as plt
from tqdm import tqdm

# Copyright (c) Facebook, Inc. and its affiliates. All Rights Reserved.
from torchvision import transforms as T
from torchvision.transforms import functional as F
```

Tutorial- Panoptic Segmentation

- Dataset

  script: dataset.py

```python
class ShapeDataset(object):

    def __init__(self, num_examples, transforms=None):

        self.height = 128
        self.width = 128

        self.num_examples = num_examples
        self.transforms = transforms # IMPORTANT, DON'T MISS
        self.image_info = []
        self.logger = logging.getLogger(__name__)

        # Class Names: Note that the ids start fromm 1 not 0. This repo uses the 0 index for background
        self.class_names = {"square": 1, "circle": 2, "triangle": 3}

        # Add images
        # Generate random specifications of images (i.e. color and
        # list of shapes sizes and locations). This is more compact than
        # actual images. Images are generated on the fly in load_image().
        for i in range(num_examples):
            bg_color, shapes = self.random_image(self.height, self.width)
            self.image_info.append({ "path":None,
                                    "width": self.width, "height": self.height,
                                    "bg_color": bg_color, "shapes": shapes
                                    })

        # Fills in the self.coco varibale for evaluation.
        self.get_gt()
```

Tutorial- Panoptic Segmentation

- Dataset

  script: dataset.py

```python
def random_image(self, height, width):
    """Creates random specifications of an image with multiple shapes.
    Returns the background color of the image and a list of shape
    specifications that can be used to draw the image.
    """
    # Pick random background color
    bg_color = np.array([random.randint(0, 255) for _ in range(3)])
    # Generate a few random shapes and record their
    # bounding boxes
    shapes = []
    boxes = []
    N = random.randint(1, 4)
    labels = {}
    for _ in range(N):
        shape, color, dims = self.random_shape(height, width)
        shapes.append((shape, color, dims))
        x, y, s = dims
        boxes.append([y-s, x-s, y+s, x+s])

    # Apply non-max suppression wit 0.3 threshold to avoid
    # shapes covering each other
    keep_ixs = non_max_suppression(np.array(boxes), np.arange(N), 0.3)
    shapes = [s for i, s in enumerate(shapes) if i in keep_ixs]

    return bg_color, shapes
```

Tutorial- Panoptic Segmentation

- Dataset

  script: dataset.py

```
def random_shape(self, height, width):
    """Generates specifications of a random shape that lies within
    the given height and width boundaries.
    Returns a tuple of three valus:
    * The shape name (square, circle, ...)
    * Shape color: a tuple of 3 values, RGB.
    * Shape dimensions: A tuple of values that define the shape size
                        and location. Differs per shape type.
    """
    # Shape
    shape = random.choice(["square", "circle", "triangle"])
    # Color
    color = tuple([random.randint(0, 255) for _ in range(3)])
    # Center x, y
    buffer = 20
    y = random.randint(buffer, height - buffer - 1)
    x = random.randint(buffer, width - buffer - 1)
    # Size
    s = random.randint(buffer, height//4)
    return shape, color, (x, y, s)
```

  -
  -
  -

Tutorial- Panoptic Segmentation

- Dataset

  script: dataset.py

```
def __getitem__(self, idx):
    """Generate an image from the specs of the given image ID.
    Typically this function loads the image from a file, but
    in this case it generates the image on the fly from the
    specs in image_info.
    """
    image = Image.fromarray(self.load_image(idx))
    segmentation_mask, masks, labels, boxes = self.load_mask(idx)

    # create a BoxList from the boxes
    boxlist = BoxList(boxes, image.size, mode="xyxy")

    # add the labels to the boxlist
    boxlist.add_field("labels", torch.tensor(labels))

    # Add masks to the boxlist
    masks = np.transpose(masks, (2,0,1))
    masks = SegmentationMask(torch.tensor(masks), image.size, "mask")
    boxlist.add_field("masks", masks)

    # Add semantic segmentation masks to the boxlist for panoptic segmentation
    segmentation_mask = np.transpose(segmentation_mask, (2,0,1))
    seg_masks = SegmentationMask(torch.tensor(segmentation_mask), image.size, "mask")
    boxlist.add_field("seg_masks", seg_masks)

    # Important line! dont forget to add this
    if self.transforms:
        image, boxlist = self.transforms(image, boxlist)

    # return the image, the boxlist and the idx in your dataset
    return image, boxlist, idx
```

Tutorial- Panoptic Segmentation

- Dataset

  script: dataset.py

```
def get_gt(self):
    # Prepares dataset for coco eval
    images = []
    annotations = []
    results = []
    # Define categories
    categories = [ {"id": 1, "name": "square"}, {"id": 2, "name": "circle"}, {"id": 3, "name": "triangle"}
    i = 1
    ann_id = 0
    for img_id, d in enumerate(self.image_info):
      images.append( {"id": img_id, 'height': self.height, 'width': self.width } )
      for (shape, color, dims) in d['shapes']:
        if shape == "square":
          category_id = 1
        elif shape == "circle":
          category_id = 2
        elif shape == "triangle":
          category_id = 3

        x, y, s = dims
        bbox = [ x - s, y - s, x+s, y +s ]
        area = (bbox[0] - bbox[2]) * (bbox[1] - bbox[3])
```

Tutorial- Panoptic Segmentation

- Dataset

  script: dataset.py

```
            # Format for COCO
            annotations.append( {
                "id": int(ann_id),
                "category_id": category_id,
                "image_id": int(img_id),
                "area" : float(area),
                "bbox": [ float(bbox[0]), float(bbox[1]), float(bbox[2]) - float(bbox[0]) + 1, float(bbox[3])
                "iscrowd" : 0
            } )
            ann_id += 1
    # Save ground truth file
    with open("tmp_gt.json", "w") as f:
      json.dump({"images": images, "annotations": annotations, "categories": categories }, f)
    # Load gt for coco eval
    self.coco = COCO("tmp_gt.json")
```

Tutorial- Panoptic Segmentation

- Model

  script: base_config.yaml

```yaml
MODEL:
  META_ARCHITECTURE: "GeneralizedRCNN"
  WEIGHT: "catalog://Caffe2Detectron/COCO/35858933/e2e_mask_rcnn_R-50-FPN_1x"
  BACKBONE:
    CONV_BODY: "R-50-FPN"
  RESNETS:
    BACKBONE_OUT_CHANNELS: 256
  RPN:
    USE_FPN: True
    ANCHOR_STRIDE: (4, 8, 16, 32, 64)
    PRE_NMS_TOP_N_TRAIN: 2000
    PRE_NMS_TOP_N_TEST: 1000
    POST_NMS_TOP_N_TEST: 1000
    FPN_POST_NMS_TOP_N_TEST: 1000
  ROI_HEADS:
    USE_FPN: True
  ROI_BOX_HEAD:
    POOLER_RESOLUTION: 7
    POOLER_SCALES: (0.25, 0.125, 0.0625, 0.03125)
    POOLER_SAMPLING_RATIO: 2
    FEATURE_EXTRACTOR: "FPN2MLPFeatureExtractor"
    PREDICTOR: "FPNPredictor"
  ROI_MASK_HEAD:
    POOLER_SCALES: (0.25, 0.125, 0.0625, 0.03125)
    FEATURE_EXTRACTOR: "MaskRCNNFPNFeatureExtractor"
    PREDICTOR: "MaskRCNNC4Predictor"
    POOLER_RESOLUTION: 14
    POOLER_SAMPLING_RATIO: 2
    RESOLUTION: 28
    SHARE_BOX_FEATURE_EXTRACTOR: False
  MASK_ON: True
DATALOADER:
  SIZE_DIVISIBILITY: 32
```

Tutorial- Panoptic Segmentation

- Model

  script: shapes_config.yaml

```yaml
MODEL:
  META_ARCHITECTURE: "GeneralizedRCNN"
  WEIGHT: "base_model.pth"
  BACKBONE:
    CONV_BODY: "R-50-FPN"
  RESNETS:
    BACKBONE_OUT_CHANNELS: 256
  RPN:
    USE_FPN: True
    ANCHOR_STRIDE: (4, 8, 16, 32, 64)
    PRE_NMS_TOP_N_TRAIN: 2000
    PRE_NMS_TOP_N_TEST: 1000
    POST_NMS_TOP_N_TEST: 1000
    FPN_POST_NMS_TOP_N_TEST: 1000
  ROI_HEADS:
    USE_FPN: True
  ROI_BOX_HEAD:
    POOLER_RESOLUTION: 7
    POOLER_SCALES: (0.25, 0.125, 0.0625, 0.03125)
    POOLER_SAMPLING_RATIO: 2
    FEATURE_EXTRACTOR: "FPN2MLPFeatureExtractor"
    PREDICTOR: "FPNPredictor"
    NUM_CLASSES: 4 # background + num_classes : IMPORTANT dont forget to add this
  ROI_MASK_HEAD:
    POOLER_SCALES: (0.25, 0.125, 0.0625, 0.03125)
    FEATURE_EXTRACTOR: "MaskRCNNFPNFeatureExtractor"
    PREDICTOR: "MaskRCNNC4Predictor"
    POOLER_RESOLUTION: 14
    POOLER_SAMPLING_RATIO: 2
    RESOLUTION: 28
    SHARE_BOX_FEATURE_EXTRACTOR: False
  MASK_ON: True
```

Tutorial- Panoptic Segmentation

- Train

  script: main.py

```python
def do_train_panoptic(
    model,
    data_loader,
    optimizer,
    scheduler,
    checkpointer,
    device,
    checkpoint_period,
    arguments,
):
    logger = logging.getLogger("maskrcnn_benchmark.trainer")
    logger.error("Start training")
    meters = MetricLogger(delimiter="  ")
    max_iter = len(data_loader)
    start_iter = arguments["iteration"]
    model.train()
    start_training_time = time.time()
    end = time.time()

    for iteration, (images, targets, _) in enumerate(data_loader, start_iter):

        if any(len(target) < 1 for target in targets):
            logger.error(f"Iteration={iteration + 1} || Image Ids used for training {_} ||
            continue
        data_time = time.time() - end
        iteration = iteration + 1
        arguments["iteration"] = iteration

        scheduler.step()
```

Tutorial- Panoptic Segmentation

- Train

  script: main.py

```
images = images.to(device)
targets = [target.to(device) for target in targets]

loss_dict = model(images, targets)

losses = sum(loss for loss in loss_dict.values())

# reduce losses over all GPUs for logging purposes
loss_dict_reduced = reduce_loss_dict(loss_dict)
losses_reduced = sum(loss for loss in loss_dict_reduced.values())
meters.update(loss=losses_reduced, **loss_dict_reduced)

optimizer.zero_grad()
# Note: If mixed precision is not used, this ends up doing nothing
# Otherwise apply loss scaling for mixed-precision recipe
with amp.scale_loss(losses, optimizer) as scaled_losses:
    scaled_losses.backward()
optimizer.step()

batch_time = time.time() - end
end = time.time()
meters.update(time=batch_time, data=data_time)

eta_seconds = meters.time.global_avg * (max_iter - iteration)
eta_string = str(datetime.timedelta(seconds=int(eta_seconds)))

if iteration % 20 == 0 or iteration == max_iter:
    logger.info(
        meters.delimiter.join(
            [
```

Tutorial- Panoptic Segmentation

- Train

  script: main.py

```
    if iteration % 20 == 0 or iteration == max_iter:
        logger.info(
            meters.delimiter.join(
                [
                    "eta: {eta}",
                    "iter: {iter}",
                    "{meters}",
                    "lr: {lr:.6f}",
                    "max mem: {memory:.0f}",
                ]
            ).format(
                eta=eta_string,
                iter=iteration,
                meters=str(meters),
                lr=optimizer.param_groups[0]["lr"],
                memory=torch.cuda.max_memory_allocated() / 1024.0 / 1024.0,
            )
        )
    if iteration % checkpoint_period == 0:
        checkpointer.save("model_{:07d}".format(iteration), **arguments)
    if iteration == max_iter:
        checkpointer.save("model_final", **arguments)

total_training_time = time.time() - start_training_time
total_time_str = str(datetime.timedelta(seconds=total_training_time))
logger.info(
    "Total training time: {} ({:.4f} s / it)".format(
        total_time_str, total_training_time / (max_iter)
    ))
```

Tutorial- Panoptic Segmentation

- Train

  script: main.py

```
    if iteration % 20 == 0 or iteration == max_iter:
        logger.info(
            meters.delimiter.join(
                [
                    "eta: {eta}",
                    "iter: {iter}",
                    "{meters}",
                    "lr: {lr:.6f}",
                    "max mem: {memory:.0f}",
                ]
            ).format(
                eta=eta_string,
                iter=iteration,
                meters=str(meters),
                lr=optimizer.param_groups[0]["lr"],
                memory=torch.cuda.max_memory_allocated() / 1024.0 / 1024.0,
            )
        )
    if iteration % checkpoint_period == 0:
        checkpointer.save("model_{:07d}".format(iteration), **arguments)
    if iteration == max_iter:
        checkpointer.save("model_final", **arguments)

total_training_time = time.time() - start_training_time
total_time_str = str(datetime.timedelta(seconds=total_training_time))
logger.info(
    "Total training time: {} ({:.4f} s / it)".format(
        total_time_str, total_training_time / (max_iter)
    ))
```

Tutorial- Panoptic Segmentation

- Train

  script: main.py

```python
def train_panoptic(cfg, local_rank, distributed, dataset):
    model = build_panoptic_network(cfg)

    device = torch.device('cuda')
    model.to(device)

    optimizer = make_optimizer(cfg, model)
    scheduler = make_lr_scheduler(cfg, optimizer)

    # Initialize mixed-precision training
    use_mixed_precision = cfg.DTYPE == "float16"
    amp_opt_level = 'O1' if use_mixed_precision else 'O0'
    model, optimizer = amp.initialize(model, optimizer, opt_level=amp_opt_level)

    if distributed:
        model = torch.nn.parallel.DistributedDataParallel(
            model, device_ids=[local_rank], output_device=local_rank,
            # this should be removed if we update BatchNorm stats
            broadcast_buffers=False,
        )

    arguments = {}
    arguments["iteration"] = 0
```

Tutorial- Panoptic Segmentation

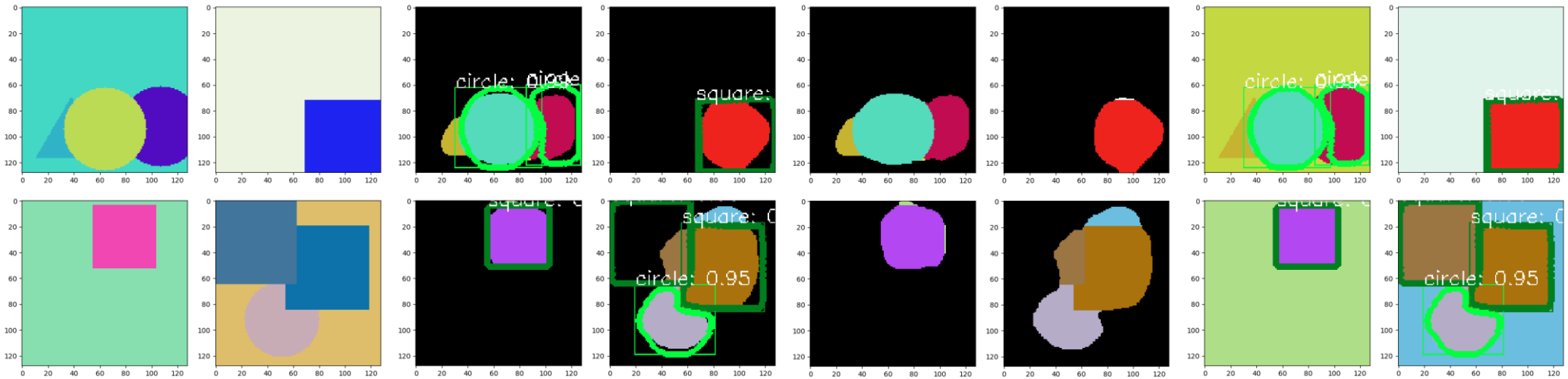
- Train

  script: main.py

```
output_dir = cfg.OUTPUT_DIR
save_to_disk = get_rank() == 0
checkpointer = DetectronCheckpointer(
    cfg, model, optimizer, scheduler, output_dir, save_to_disk
)
extra_checkpoint_data = checkpointer.load(cfg.MODEL.WEIGHT)
arguments.update(extra_checkpoint_data)


data_loader = build_data_loader(cfg, dataset)

checkpoint_period = cfg.SOLVER.CHECKPOINT_PERIOD

do_train_panoptic(
    model,
    data_loader,
    optimizer,
    scheduler,
    checkpointer,
    device,
    checkpoint_period,
    arguments,
)

return model
```

Tutorial- Panoptic Segmentation

- Results:



Input Instance Segmentation Semantic Segmentation Panoptic Segmentation

Homework

- Change the backbones of the architecture and compare the results obtained with the original code. Discuss over the quality of segmentations.
- (Optional) Implement the Panoptic Quality metric.

- Following the steps presented on Github, present the results of UPSNet using the COCO val2017 dataset, using pre-trained weights. Report the panoptic quality (PQ) and some qualitative results.