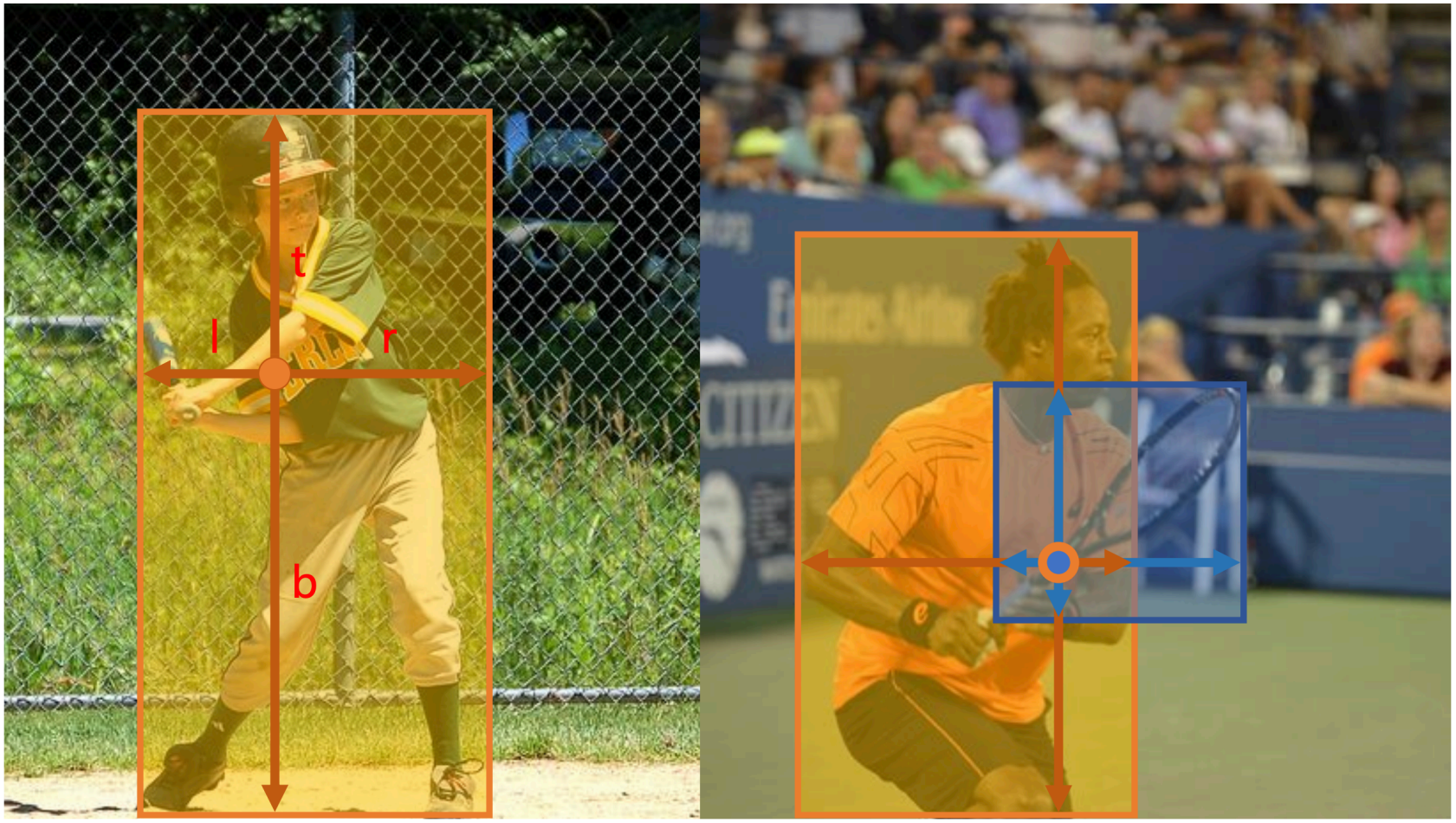
Single-shot Instance Segmentation

Chunhua Shen, June 2020
(majority of work done by my students: Zhi tian, Hao Chen, and Xinlong Wang)

FCOS Detector



Tian, Zhi, et al. "FCOS: Fully convolutional one-stage object detection." Proc. *Int. Conf. Comp. Vis*. 2019.

Overview of FCOS

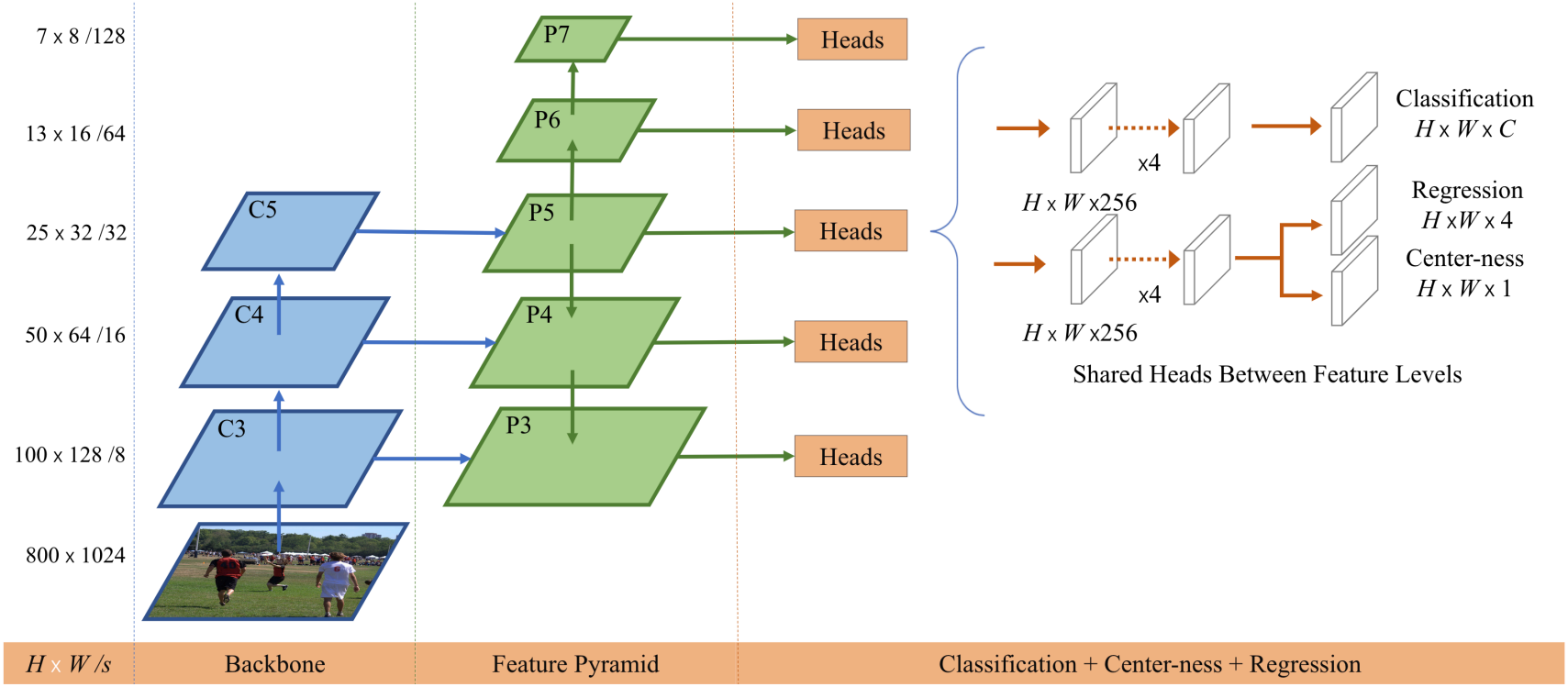


Fig. 2. **The network architecture of FCOS**, where C3, C4, and C5 denote the feature maps of the backbone network and P3 to P7 are the feature levels used for the final prediction. $H \times W$ is the height and width of feature maps. '/s' ($s = 8, 16, ..., 128$) is the down-sampling ratio of the feature maps at the level to the input image. As an example, all the numbers are computed with an 800×1024 input.

Performance

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods:							
Faster R-CNN+++ [36]	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w/ FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [40]	Inception-ResNet-v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w/ TDM [42]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
One-stage methods:							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
YOLOv3 608×608 [3]	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9
DSSD513 [43]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [4]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
CornerNet [29]	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
FSAF [30]	ResNeXt-64x4d-101-FPN	42.9	63.8	46.3	26.6	46.2	52.7
CenterNet511 [44]	Hourglass-104	44.9	62.4	48.1	25.6	47.4	57.4
FCOS	ResNet-101-FPN	43.2	62.4	46.8	26.1	46.2	52.8
FCOS	ResNeXt-32x8d-101-FPN	44.1	63.7	47.9	27.4	46.8	53.7
FCOS	ResNeXt-64x4d-101-FPN	44.8	64.4	48.5	27.7	47.4	55.0
FCOS w/ deform. conv. v2 [45]	ResNeXt-32x8d-101-FPN	46.6	65.9	50.8	28.6	49.1	58.6
FCOS	ResNet-101-BiFPN [46]	45.0	63.6	48.7	27.0	47.9	55.9
FCOS	ResNeXt-32x8d-101-BiFPN	46.2	65.2	50.0	28.7	49.1	56.5
FCOS w/ deform. conv. v2	ResNeXt-32x8d-101-BiFPN	47.9	66.9	51.9	30.2	50.3	59.9
w/ test-time augmentation:							
FCOS	ResNet-101-FPN	45.9	64.5	50.4	29.4	48.3	56.1
FCOS	ResNeXt-32x8d-101-FPN	47.0	66.0	51.6	30.7	49.4	57.1
FCOS	ResNeXt-64x4d-101-FPN	47.5	66.4	51.9	31.4	49.7	58.2
FCOS w/ deform. conv. v2	ResNeXt-32x8d-101-FPN	49.1	68.0	53.9	31.7	51.6	61.0
FCOS	ResNet-101-BiFPN	47.9	65.9	52.5	31.0	50.7	59.7
FCOS	ResNeXt-32x8d-101-BiFPN	49.0	67.4	53.6	32.0	51.7	60.5
FCOS w/ deform. conv. v2	ResNeXt-32x8d-101-BiFPN	**50.4**	**68.9**	**55.0**	**33.2**	**53.0**	**62.7**

TABLE 8

FCOS vs. other state-of-the-art two-stage or one-stage detectors (*single-model results*). FCOS outperforms a few recent anchor-based and anchor-free detectors by a considerable margin.

Pros of FCOS

- Much Simpler
  - Much less hyper-parameters.
  - Much easy to implement (e.g., don't need to compute IOUs).
  - Easy to extend to other tasks such as keypoint detection/instance segmentation.
  - Detection becomes a per-pixel prediction task.
- Faster training and testing with better performance
  - FCOS achieves much better performance-speed tradeoff than all other detectors. A real-time FCOS achieves **46FPS/40.3mAP** on 1080Ti.
  - In comparison, YOLOv3, ~40FPS/33mAP on 1080Ti.
  - CenterNet, 14FPS/40.3mAP.

Instance segmentation

BlendMask

- Instance-level attention tensor
- Only four score maps (vs. 32 in YOLACT vs. 49 in FCIS)
- 20% faster than Mask-RCNN with higher performance under same training setting

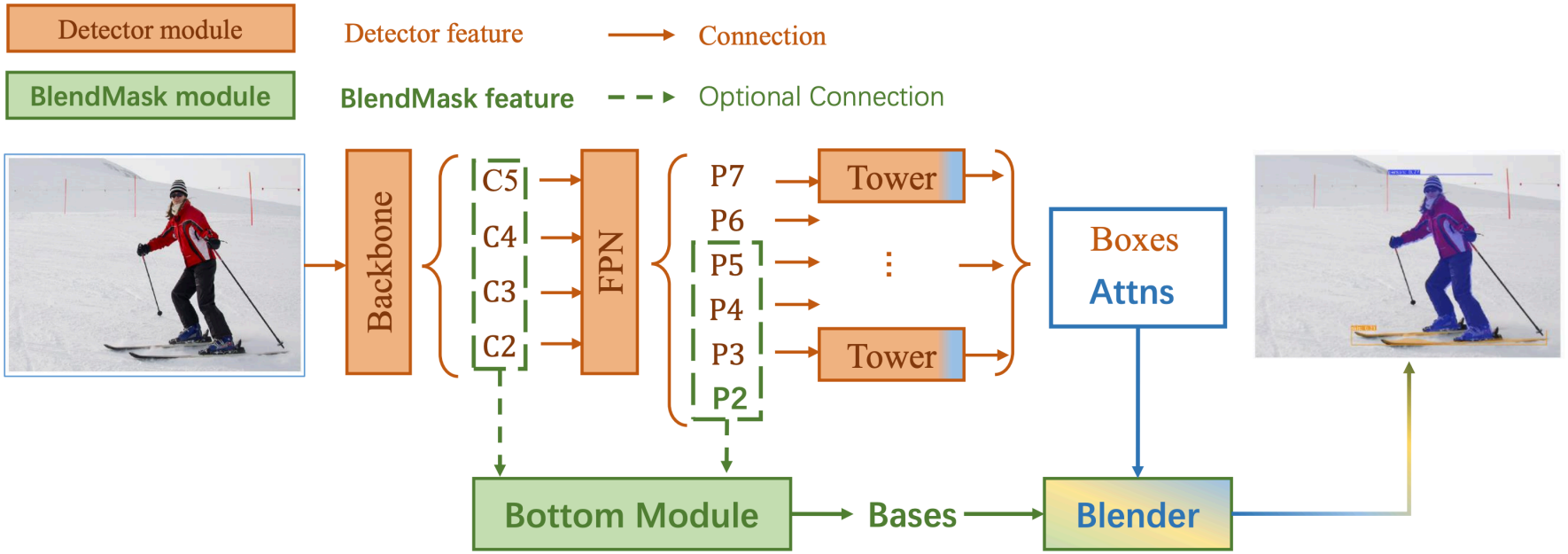


Figure 2 – BlendMask pipeline Our framework builds upon the state-of-the-art FCOS object detector [25] with minimal modification. The bottom module uses either backbone or FPN features to predict a set of bases. A single convolution layer is added on top of the detection towers to produce attention masks along with each bounding box prediction. For each predicted instance, the blender crops the bases with its bounding box and linearly combine them according the learned attention maps. Note that the Bottom Module can take features either from 'C', or 'P' as the input.

Blending

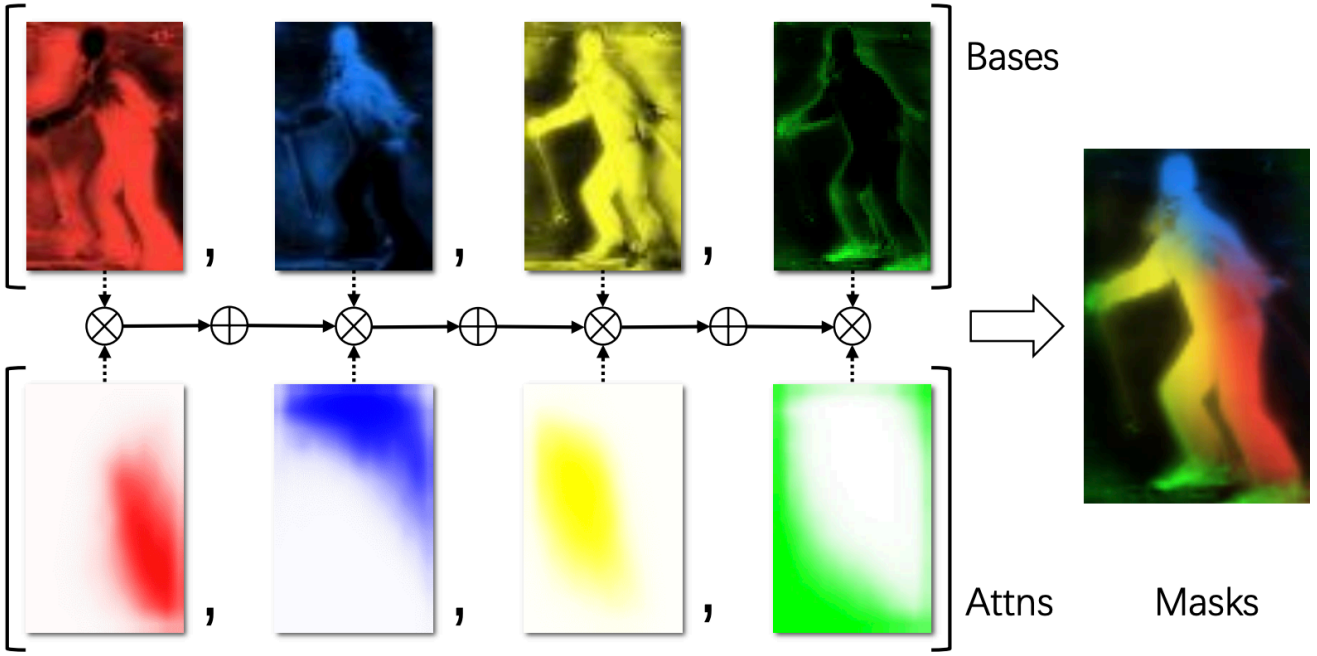


Figure 1 – Blending process. We illustrate an example of the learned bases and attentions. Four bases and attention maps are shown in different colors. The first row are the bases, and the second row are the attentions. Here $\otimes$ represents element-wise product and $\oplus$ is element-wise sum. Each basis multiplies its attention and then is summed to output the final mask.

Interpretation of Bases and Attentions

- Bases
  - Position-sensitive (Red & Blue)
  - Semantic (Yellow & Green)
- Attention
  - Instance poses
  - Foreground/background

(a) Bottom-Level Bases

(b) Top-Level attentions

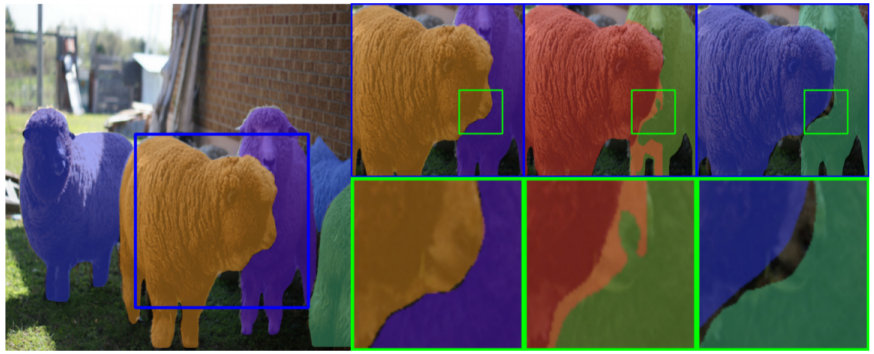
Quantitative Results

Method	Backbone	Epochs	Aug.	Time (ms)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [12]	R-50	12		97.0	34.6	56.5	36.6	15.4	36.3	49.7
Mask R-CNN*		72	✓	97+	36.8	**59.2**	39.3	17.1	38.7	52.1
TensorMask [7]		72	✓	400+	35.5	57.3	37.4	16.6	37.0	49.1
BlendMask		12		**78.5**	34.3	55.4	36.6	14.9	36.4	48.9
BlendMask		36	✓	**78.5**	**37.0**	58.9	**39.7**	**17.3**	**39.4**	**52.5**
Mask R-CNN	R-101	12		118.1	36.2	58.6	38.4	16.4	38.4	52.1
Mask R-CNN*		36	✓	118+	38.3	61.2	40.8	18.2	40.6	54.1
TensorMask		72	✓	400+	37.3	59.5	39.5	17.5	39.3	51.6
SOLO [24]		72	✓	-	37.8	59.5	40.4	16.4	40.6	54.2
+deform convs [24]		72	✓	-	40.4	62.7	43.3	17.6	43.3	58.9
BlendMask		36	✓	**101.8**	38.4	60.7	41.3	18.2	41.5	53.3
BlendMask*		36	✓	105.7	39.6	61.6	42.6	22.4	42.2	51.4
+deform convs (interval = 3)		60	✓	116.0	**41.3**	**63.1**	**44.6**	**22.7**	**44.1**	**54.5**

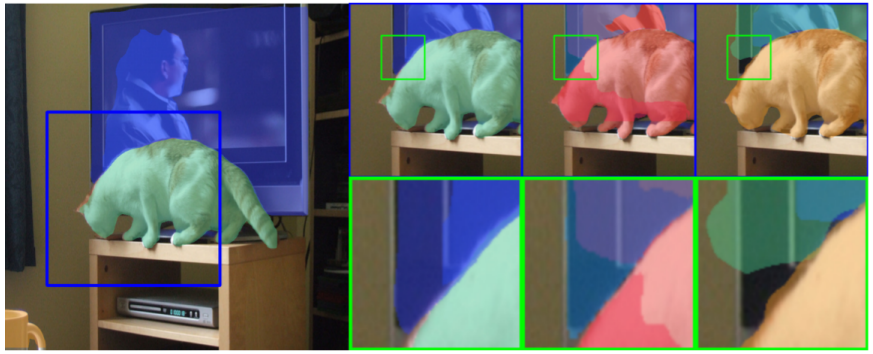
Speed on V100 (ms/image):

- BlendMask: 73
- Mask R-CNN: 90
- TensorMask: 380

ours ours YOLACT M-RCNN



ours ours YOLACT M-RCNN



ours ours YOLACT M-RCNN

ours ours YOLACT M-RCNN

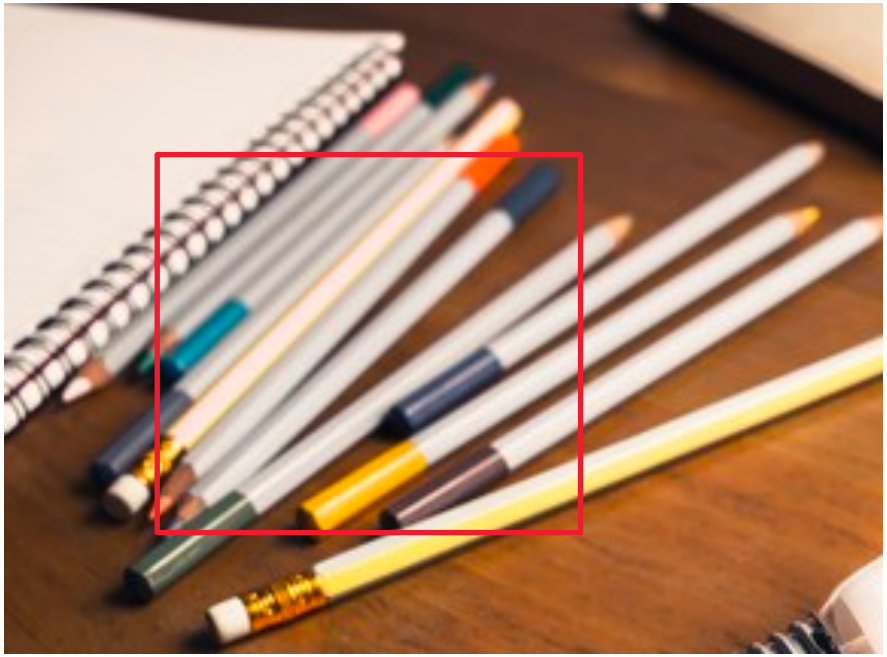
Easy to do Panoptic segmentation

Method	Backbone	PQ	SQ	RQ	PQ^{Th}	PQ^{St}	mIoU	AP^{box}	AP
Panoptic-FPN [16]	R-50	41.5	79.1	50.5	48.3	31.2	42.9	40.0	36.5
BlendMask		**42.5**	**80.1**	**51.6**	**49.5**	**32.0**	**43.5**	**41.8**	**37.2**
Panoptic-FPN [16]	R-101	43.0	80.0	52.1	49.7	32.9	44.5	42.4	38.5
BlendMask		**44.3**	**80.1**	**53.4**	**51.6**	**33.2**	**44.9**	**44.0**	**38.9**

Table 10 – Panoptic results on COCO `val2017`. Panoptic-FPN results are from the official `Detectron2` implementation, which are improved upon the original published results in [16].

- Can we remove bounding box (and related RoI align/pooling from Instance Segmentation?

Issues of Axis-aligned ROIs



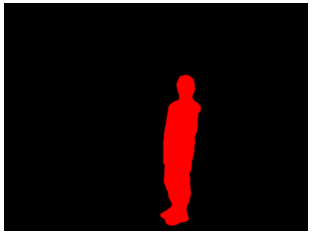
- Difficult to encode irregular shapes
- May include irrelevant background
- Low resolution segmentation results

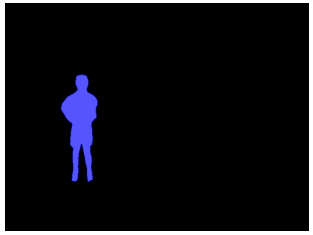
Conditional Convolutions for Instance Segmentation (ROI-free)

Main difference between instance & sematic segmentation: the same appearance needs different predictions, which standard FCNs fail to achieve.

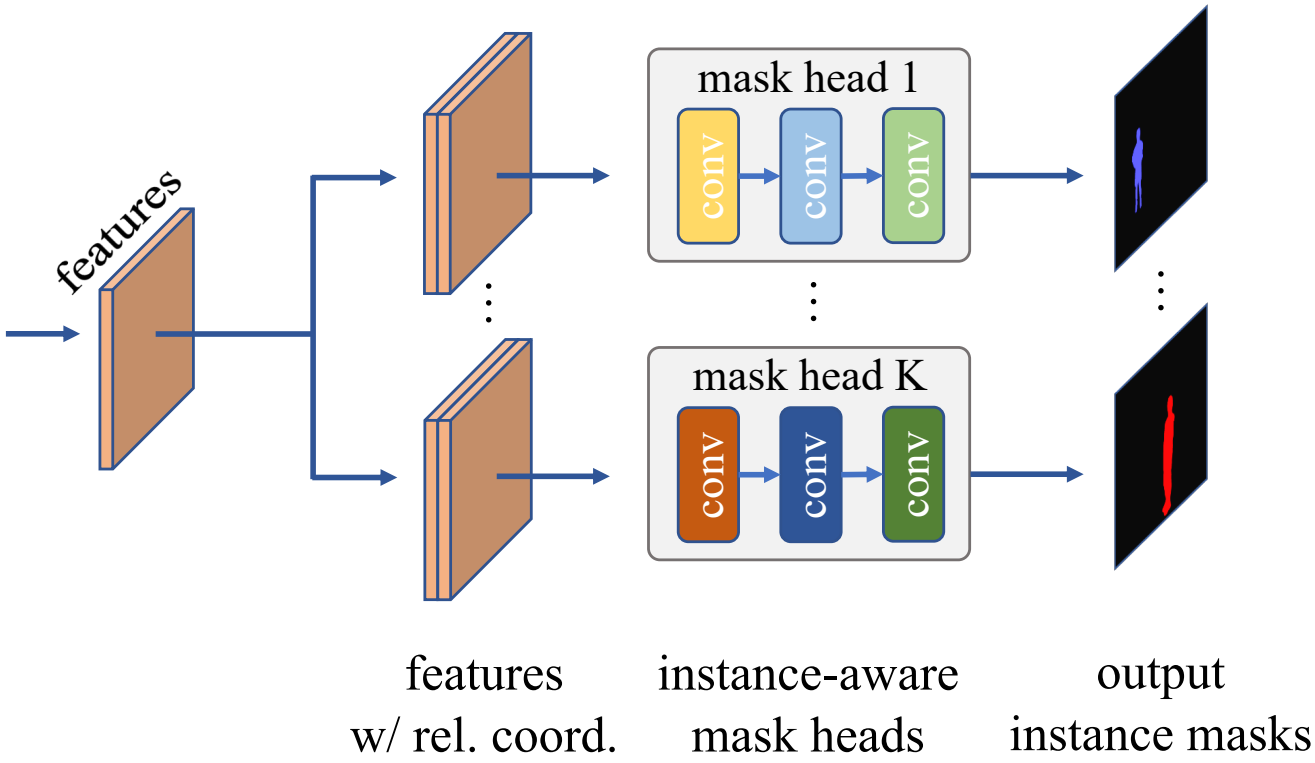
Semantic Segmentation

Instance Segmentation





Dynamic Mask Heads



Given input feature maps, CondInst employs different mask heads for different target, bypassing the limitation of the standard FCNs.

CondInst

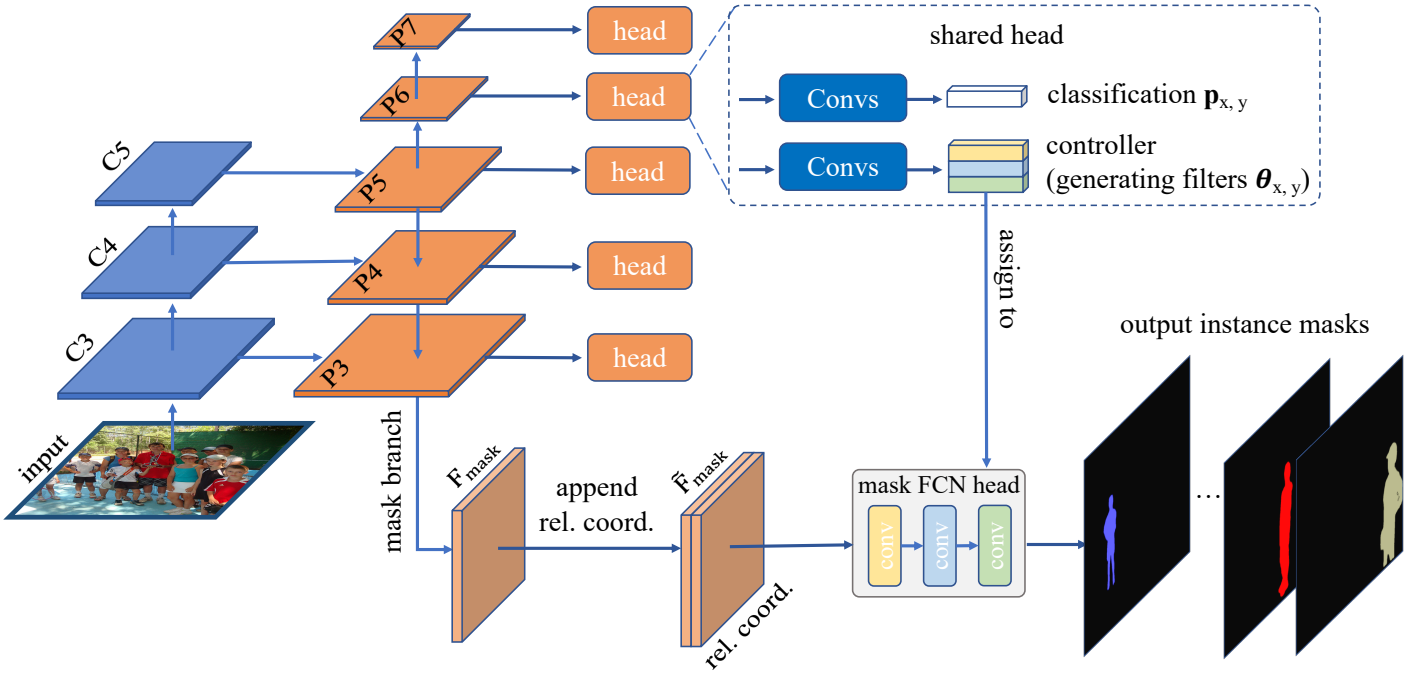


Figure 3. The overall architecture of CondInst. C_3, C_4 and C_5 are the feature maps of the backbone network (*e.g.*, ResNet-50). P_3 to P_7 are the FPN feature maps as in [8, 26]. $\mathbf{F}_{mask}$ is the mask branch's output and $\tilde{\mathbf{F}}_{mask}$ is obtained by concatenating the relative coordinates to $\mathbf{F}_{mask}$. The classification head predicts the class probability $\boldsymbol{p}_{x,y}$ of the target instance at location (x, y), same as in FCOS. Note that the classification and conv. parameter generating heads (in the dashed box) are applied to $P_3 \cdots P_7$. The mask head is instance-aware, whose conv. filters $\boldsymbol{\theta}_{x,y}$ are dynamically generated for each instance, and is applied to $\tilde{\mathbf{F}}_{mask}$ as many times as the number of instances in the image (refer to Fig. 1).

Comparisons with Mask R-CNN

- Eliminating ROI operations and thus being fully convolutional.
- Essentially, CondInst encodes the instance concept in the generated filters.
- Ability to deal with irregular shapes due to the elimination of axis-aligned boxes.
- High-resolution outputs (e.g., 400x512 vs. 28x28).
- Much lighter-weight mask heads (169 parameters vs. 2.3M in Mask R-CNN, half computation time).
- Overall inference time is faster or the same as the well-engineered Mask R-CNN in detectron2.

Ablation Study

depth	time	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1	**2.2**	30.9	52.9	31.4	14.0	33.3	45.1
2	3.3	35.5	56.1	37.8	17.0	38.9	50.8
3	4.5	**35.7**	**56.3**	37.8	17.1	**39.1**	50.2
4	5.6	**35.7**	56.2	**37.9**	**17.2**	38.7	**51.5**

(a) Varying the depth (width $= 8$).

width	time	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
2	**2.5**	34.1	55.4	35.8	15.9	37.2	49.1
4	2.6	35.6	**56.5**	**38.1**	17.0	**39.2**	**51.4**
8	4.5	**35.7**	**56.3**	37.8	17.1	39.1	50.2
16	4.7	35.6	56.2	37.9	**17.2**	38.8	50.8

(b) Varying the width (depth $= 3$).

Table 1: Instance segmentation results with different architectures of the mask head on MS-COCO `val2017` split. "depth": the number of layers in the mask head. "width": the number of channels of these layers. "time": the milliseconds that the mask head takes for processing 100 instances.

Only cost ~5ms for even the maximum number of boxes!

w/ abs. coord.	w/ rel. coord.	w/ $\mathbf{F}_{mask}$	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR_1	AR_{10}	AR_{100}
		✓	31.4	53.5	32.1	15.6	34.4	44.7	28.4	44.1	46.2
	✓		31.3	54.9	31.8	16.0	34.2	43.6	27.1	43.3	45.7
✓		✓	32.0	53.3	32.9	14.7	34.2	46.8	28.7	44.7	46.8
	✓	✓	**35.7**	**56.3**	**37.8**	**17.1**	**39.1**	**50.2**	**30.4**	**48.8**	**51.5**

Table 3: Ablation study of the input to the mask head on MS-COCO `val2017` split. As shown in the table, without the relative coordinates, the performance drops significantly from 35.7% to 31.4% in mask AP. Using the absolute coordinates cannot improve the performance remarkably (only 32.0%), which implies that the generated filters mainly encode the local cues (*e.g.*, shapes). Moreover, if the mask head only takes as input the relative coordinates (*i.e.*, no appearance features in this case), CondInst also achieves modest performance (31.3%).

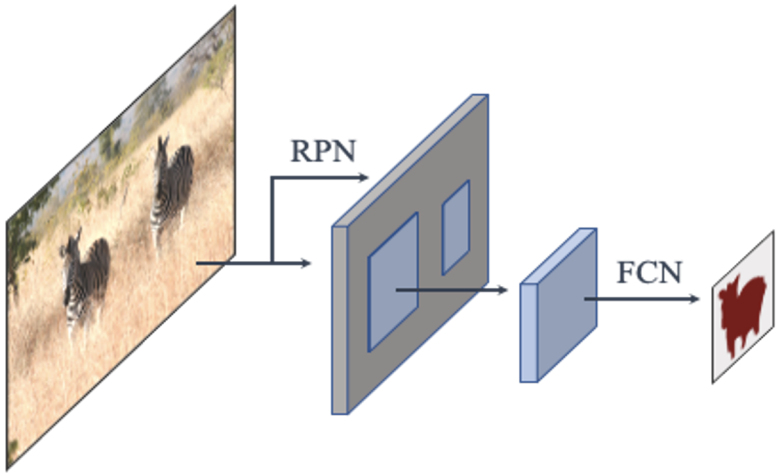
Experimental Results

method	backbone	aug.	sched.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [3]	R-50-FPN		1×	34.6	**56.5**	36.6	15.4	36.3	**49.7**
CondInst	R-50-FPN		1×	**35.4**	56.4	**37.6**	**18.4**	**37.9**	46.9
Mask R-CNN*	R-50-FPN	✓	1×	35.5	57.0	37.8	19.5	37.6	46.0
Mask R-CNN*	R-50-FPN	✓	3×	37.5	59.3	40.2	21.1	39.6	48.3
TensorMask [13]	R-50-FPN	✓	6×	35.4	57.2	37.3	16.3	36.8	49.3
CondInst	R-50-FPN	✓	1×	35.9	56.9	38.3	19.1	38.6	46.8
CondInst	R-50-FPN	✓	3×	37.8	59.1	40.5	21.0	40.3	48.7
CondInst w/ sem.	R-50-FPN	✓	3×	38.8	60.4	41.5	21.1	41.1	51.0
Mask R-CNN	R-101-FPN	✓	6×	38.3	61.2	40.8	18.2	40.6	**54.1**
Mask R-CNN*	R-101-FPN	✓	3×	38.8	60.9	41.9	21.8	41.4	50.5
YOLACT-700 [2]	R-101-FPN	✓	4.5×	31.2	50.6	32.8	12.1	33.3	47.1
TensorMask	R-101-FPN	✓	6×	37.1	59.3	39.4	17.4	39.1	51.6
CondInst	R-101-FPN	✓	3×	39.1	60.9	42.0	21.5	41.7	50.9
CondInst w/ sem.	R-101-FPN	✓	3×	**40.1**	**62.1**	**43.1**	**21.8**	**42.7**	52.6

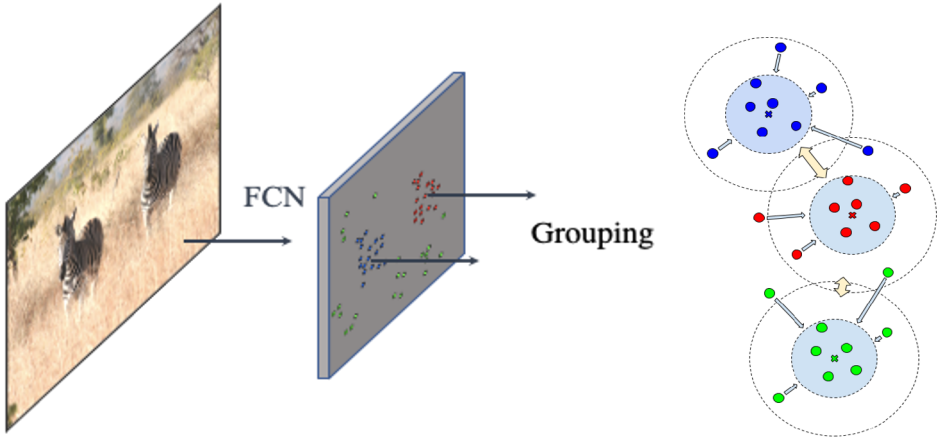
Table 6: Comparisons with state-of-the-art methods on MS-COCO `test-dev`. "Mask R-CNN" is the original Mask R-CNN [3] and "Mask R-CNN*" is the improved Mask R-CNN in `Detectron2` [35]. "aug.": using multi-scale data augmentation during training. "sched.": the used learning rate schedule. "1×" means that the models are trained with $90K$ iterations, "2×" is $180K$ iterations and so on. The learning rate is changed as in [36]. 'w/ sem": using the auxiliary semantic segmentation task.

SOLO: Segmenting objects by locations

Current Instance Segmentation methods



Detect-then-segment
e.g., Mask R-CNN



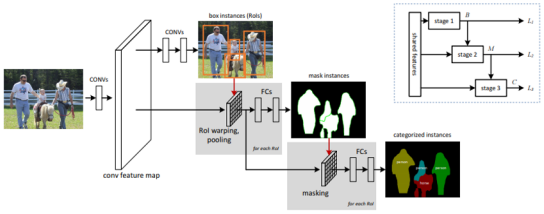
Label-then-cluster
e.g., Discriminative loss

Current Instance Segmentation methods

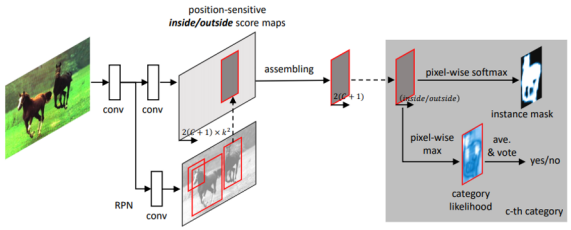
Detect-then-segment:

MNC, FCIS, Mask R-CNN,

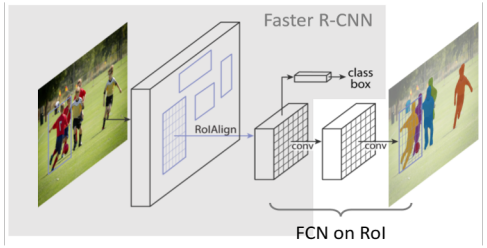
TensorMask



MNC, 2015

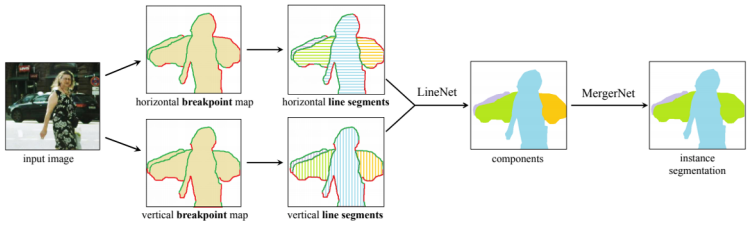


FCIS, 2016

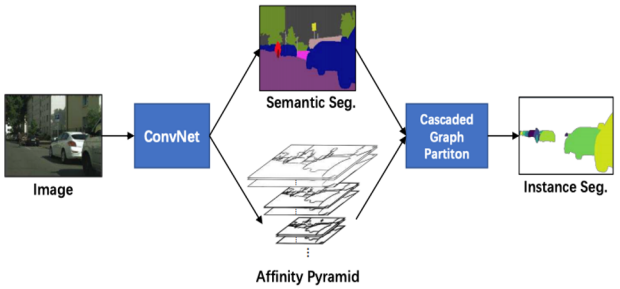


Mask R-CNN, 2017

Label-then-cluster:
SGN, SSAP, AE



SGN, 2017



SSAP, 2019

SOLO Motivation

Both the two paradigms are step-wise and indirect.

1. Top-down methods heavily rely on accurate bounding box detection.
2. Bottom-up methods depend on per-pixel embedding learning and the grouping processing.

How can we make it simple and direct?

SOLO Motivation

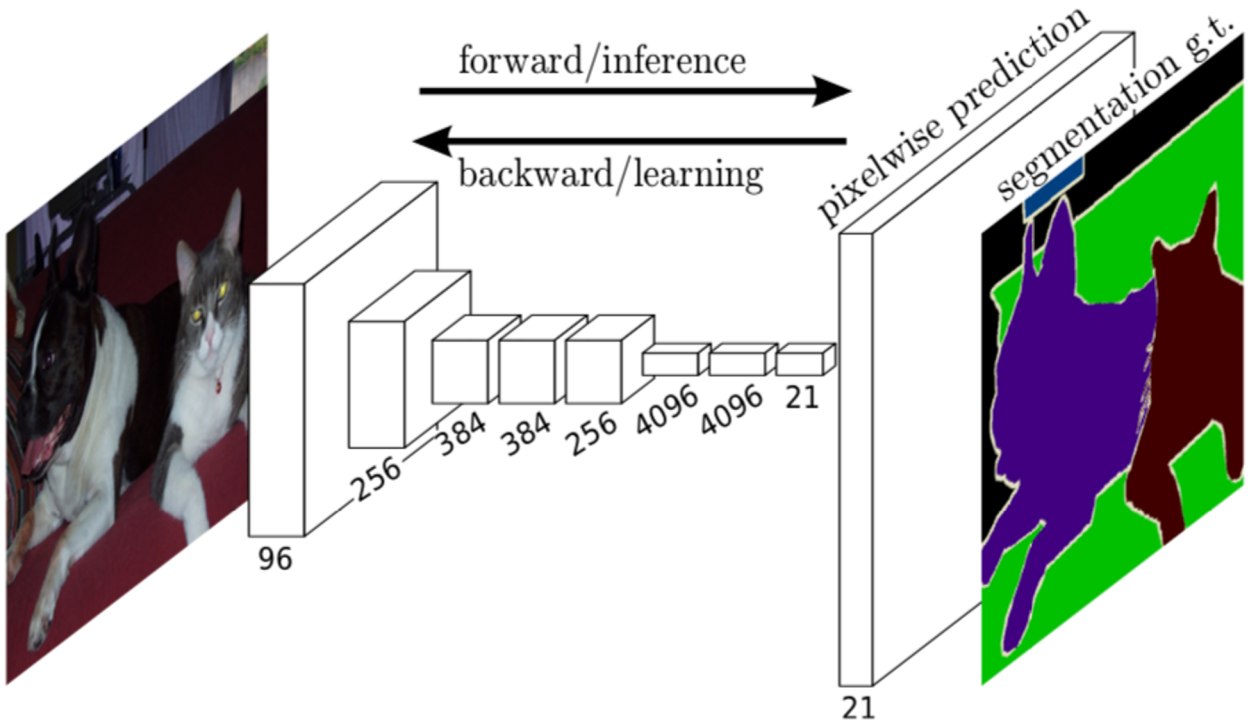


Figure credit: Long et al

Semantic segmentation: Classifying pixels into semantic categories.

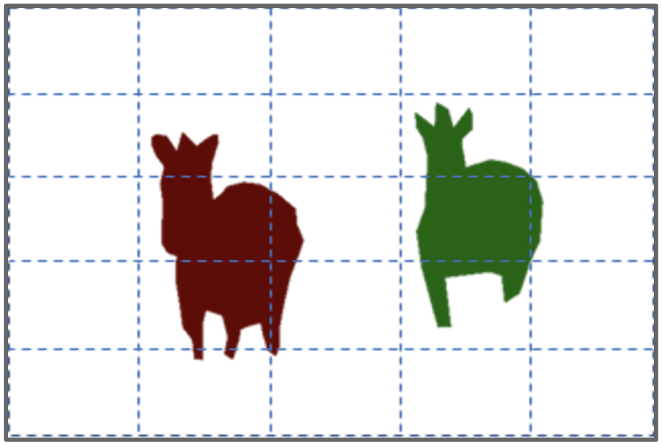
Can we convert instance segmentation into a per-pixel classification problem?

SOLO Motivation

How to convert instance segmentation into a per-pixel classification problem?

What are the fundamental differences between object instances in an image?

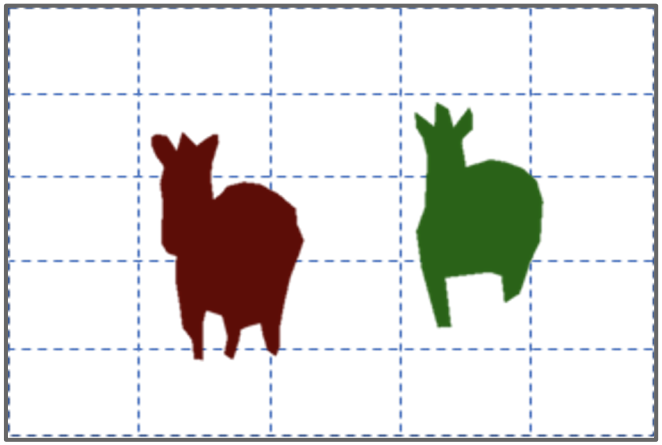
- **Instance location**
- **Object shape**



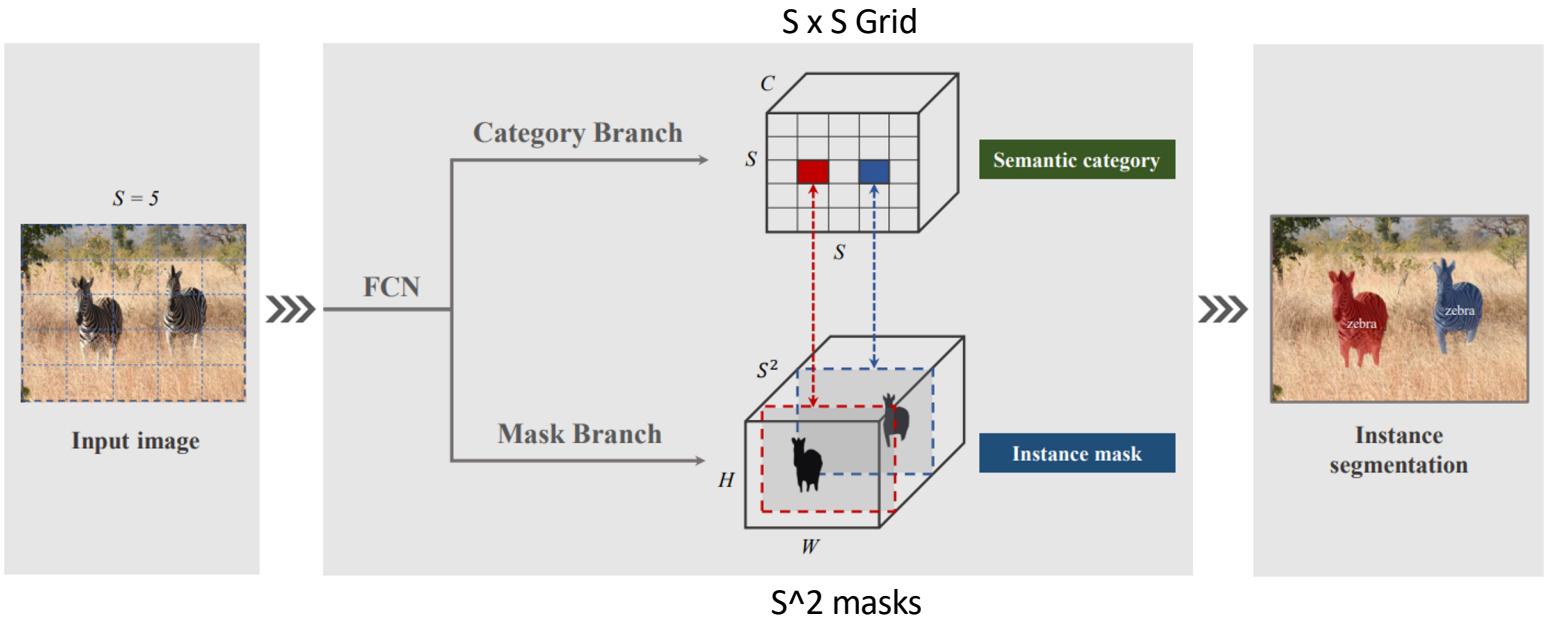
SOLO Motivation

SOLO: Segmenting Objects by Locations

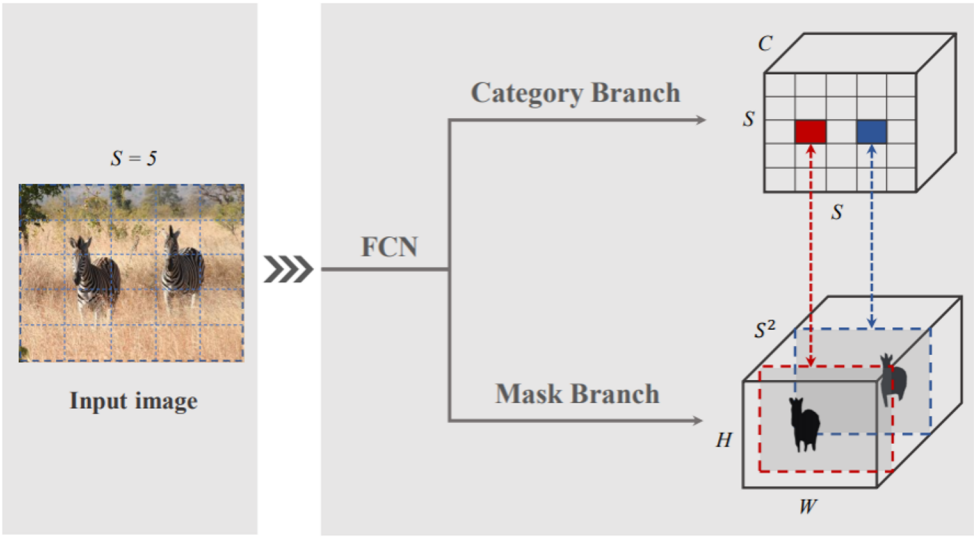
- Quantizing the locations -> mask category
- Semantic category



SOLO Framework



SOLO Framework



instance at grid (i, j)

mask at channel k, $k = i \times S + j$

Simple, fast to implement and train/test

SOLO Framework

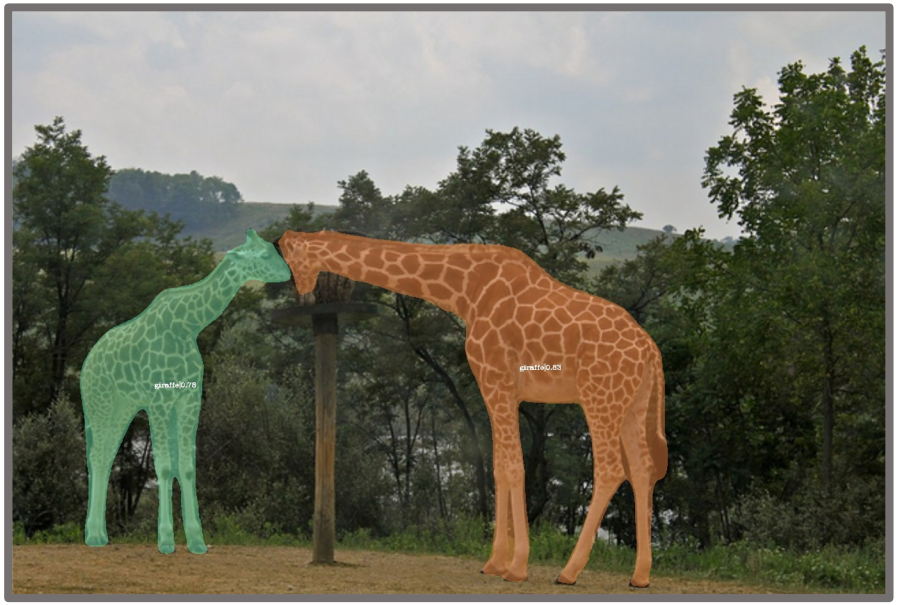
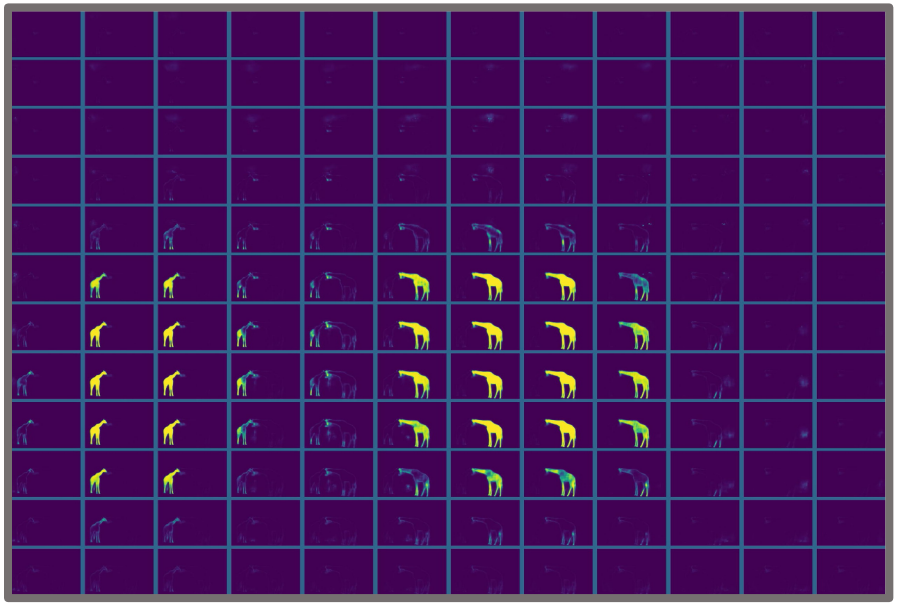
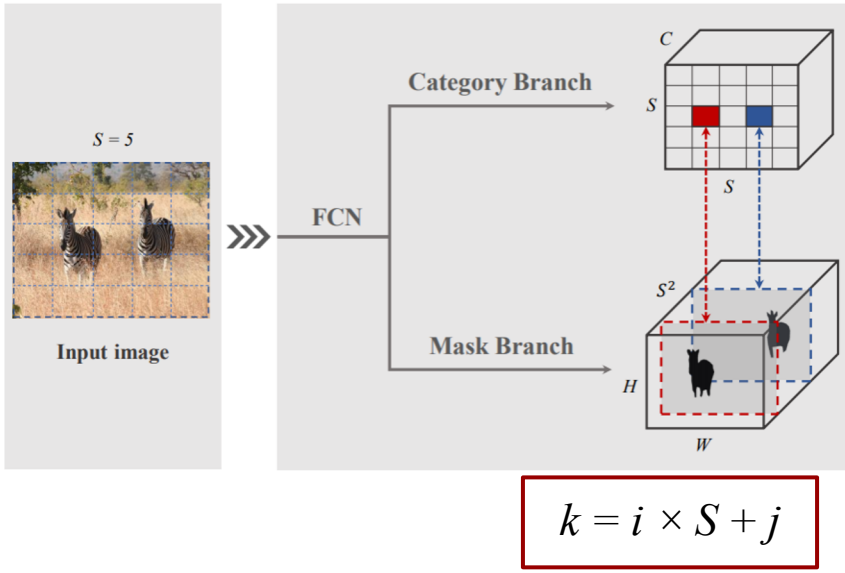


image and masks



masks with S = 12

SOLO Framework



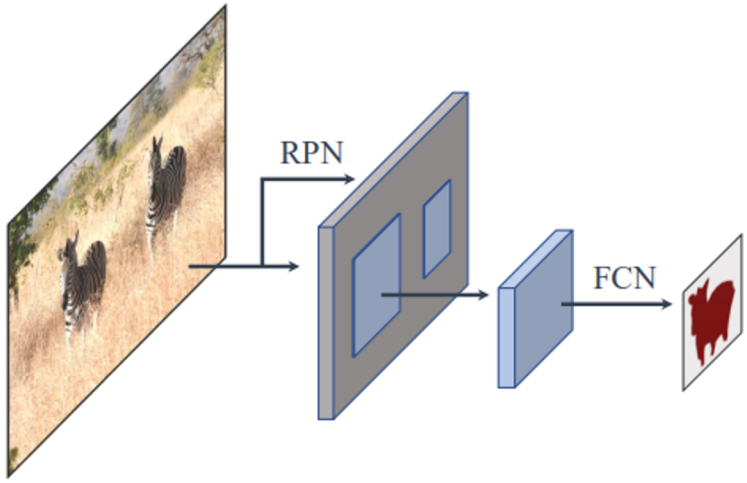
Loss Function $L = L_{cate} + \lambda L_{mask}$

Classification Loss

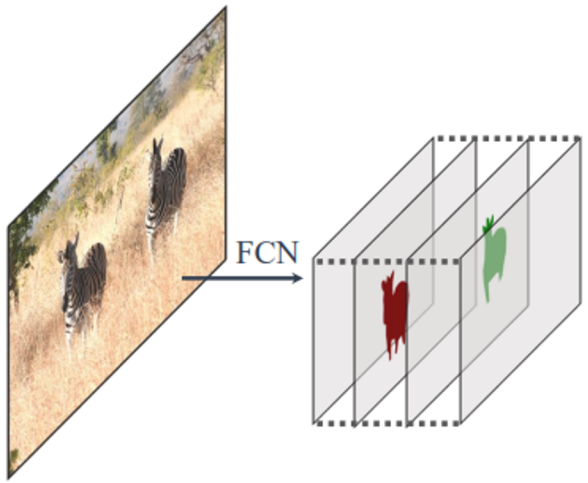
Dice Loss

$$k = i \times S + j$$

SOLO Framework



(a) Mask R-CNN



(b) SOLO

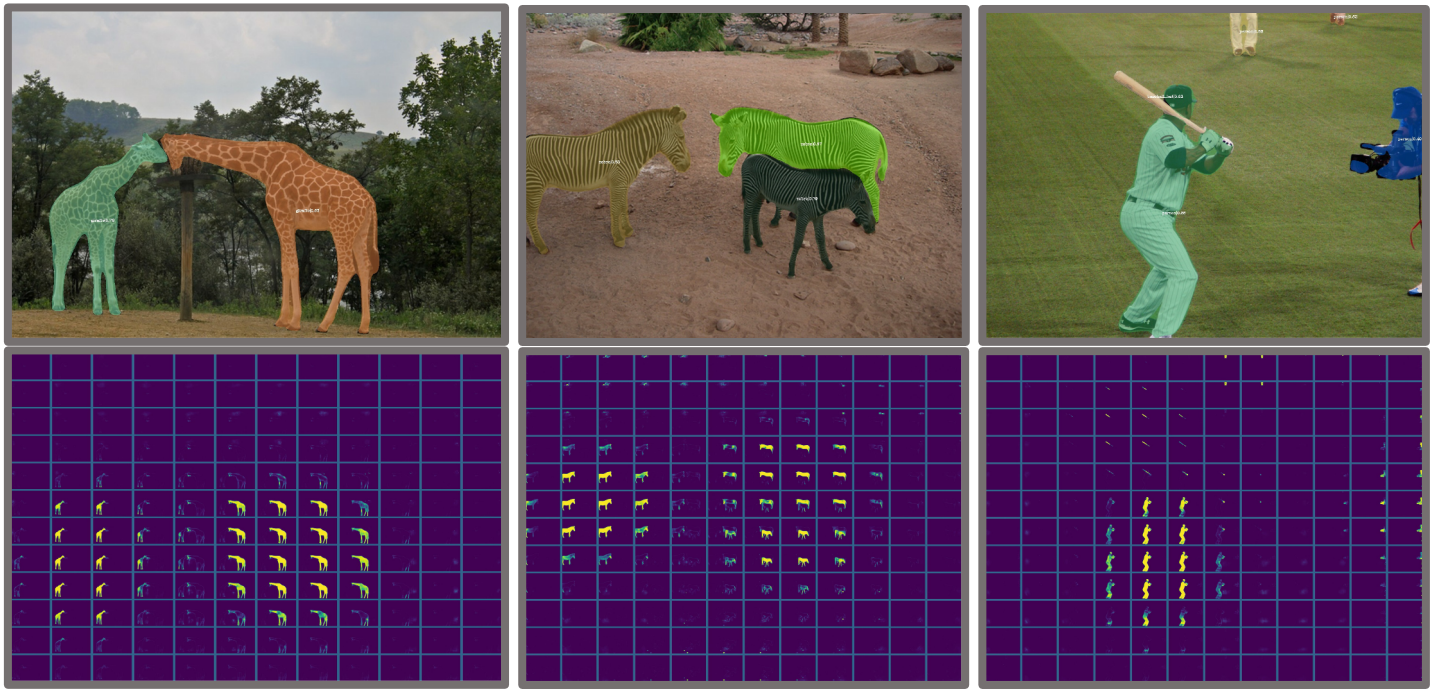
Main Results: COCO

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
two-stage:							
MNC [3]	Res-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [10]	Res-101-C5	29.2	49.5	–	7.1	31.3	50.0
Mask R-CNN [7]	Res-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN* [2]	Res-50-FPN	36.8	59.2	39.3	17.1	38.7	52.1
Mask R-CNN* [2]	Res-101-FPN	38.3	61.2	40.8	18.2	40.6	54.1
one-stage:							
TensorMask [2]	Res-50-FPN	35.4	57.2	37.3	16.3	36.8	49.3
TensorMask [2]	Res-101-FPN	37.1	59.3	39.4	17.4	39.1	51.6
YOLACT [1]	Res-101-FPN	31.2	50.6	32.8	12.1	33.3	47.1
PolarMask [27]	Res-101-FPN	30.4	51.9	31.0	13.4	32.4	42.8
ours:							
SOLO	Res-50-FPN	36.8	58.6	39.0	15.9	39.5	52.1
SOLO	Res-101-FPN	37.8	59.5	40.4	16.4	40.6	54.2
SOLO	Res-DCN-101-FPN	40.4	62.7	43.3	17.6	43.3	58.9

Table 1 – **Instance segmentation mask AP** on COCO `test-dev`. All entries are *single-model* results. Here we adopt the "6×" schedule (72 epochs) for better results. Mask R-CNN* is the improved version in [2].

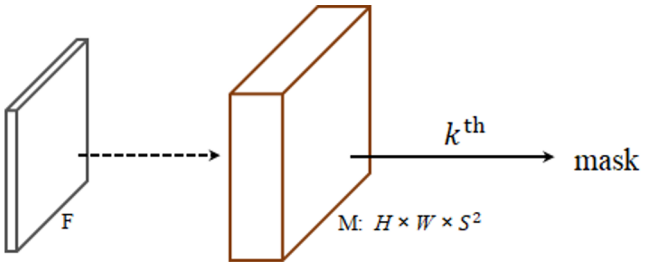
- comparable to Mask R-CNN
- 1.4 AP better than state-of-the-art one-stage methods

SOLO Behavior



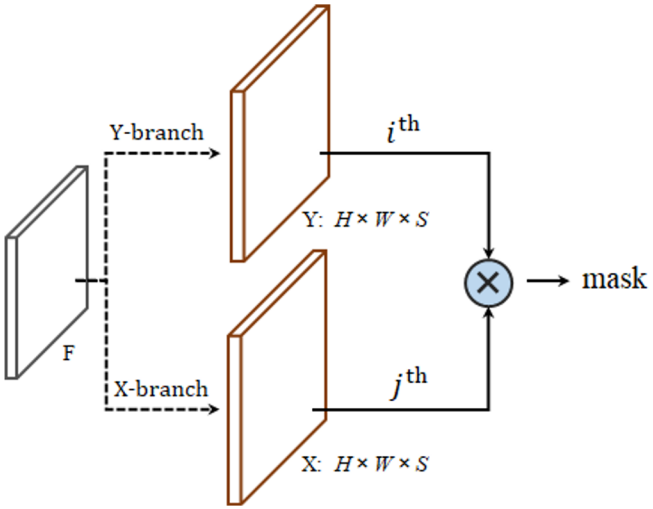
S = 12

From SOLO to Decoupled SOLO



Vanilla head

predict $p(k)$, where $k = i \times S + j$



Decoupled head

predict $p(i)$, $p(j)$, and $p(k) = p(i)p(j)$

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Vanilla SOLO	35.8	57.1	37.8	15.0	38.7	53.6
Decoupled SOLO	35.8	57.2	37.7	16.3	39.1	52.2

- an equivalent variant in accuracy
- considerably less GPU memory during training and testing


STOP

Thanks. That's all.

- All papers are available at arXiv. Code is available at

  https://git.io/AdelaiDet