Sparse Instance Activation for Real-Time Instance Segmentation

Tianheng Cheng[1], Xinggang Wang[1], Shaoyu Chen[1], Wenqiang Zhang[1],

Qian Zhang[2], Chang Huang[2], Zhaoxiang Zhang[3], Wenyu Liu[1]

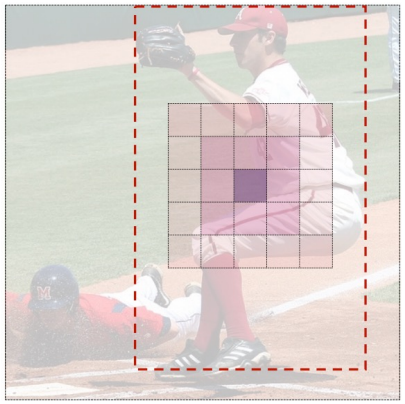
[1]Huazhong University of Science and Technology, [2]Horizon Robotics, [3]CASIA

Paper

Code

Motivation

- Object Representation in Object Detection



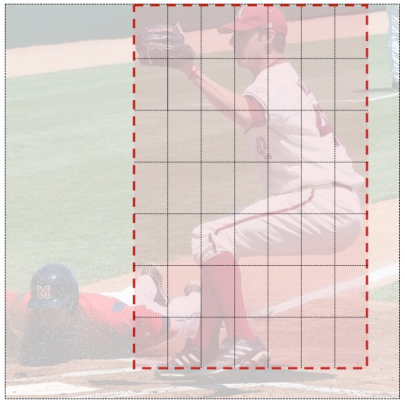
(a) center-based

Problems in center/anchor-based representation:

- requires dense anchors and hand-crafted assign rules,
- complex post-processing to remove duplicates,
- assigned targets might be wrong or confused, e.g., the positive anchors are beyond the object,
- lacking sufficient context for pixel-wise segmentation.

Motivation

- Object Representation in Object Detection



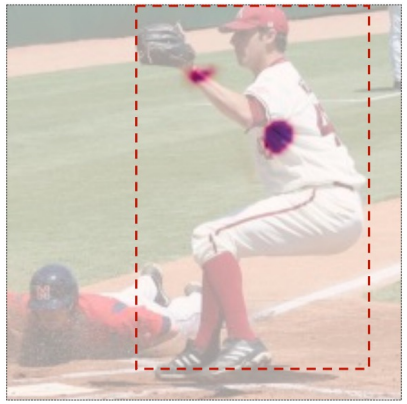
(b) region-based

Problems in region-based representation:

- requires special operation RoI-Align,
- includes undesired information of other objects or the background.

Motivation

- Object Representation in Object Detection



(c) instance activation map

Representing objects using a set of Instance Activation Maps (IAM):

(1) each object is highlighted by an activation map (IAM)

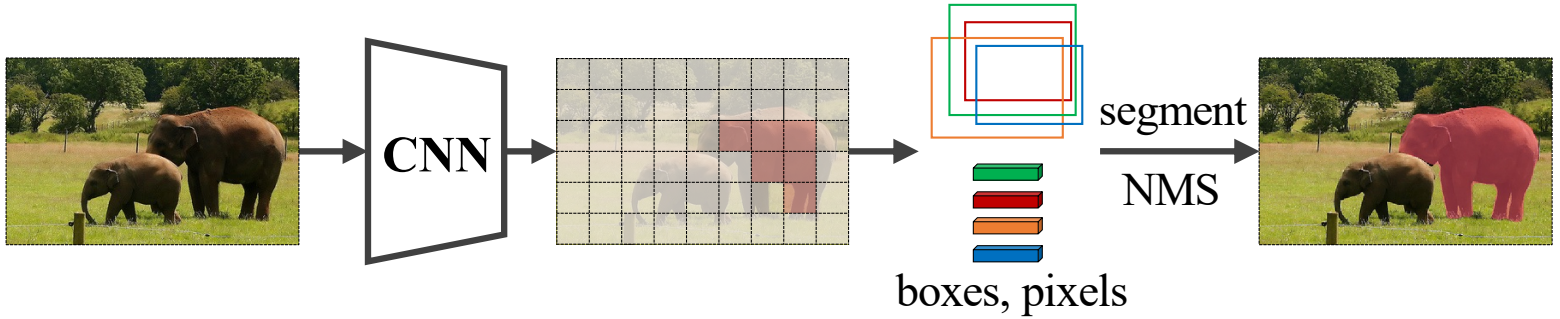
(2) object features are extracted from the activated regions.

Advantages of using Instance Activation Maps:

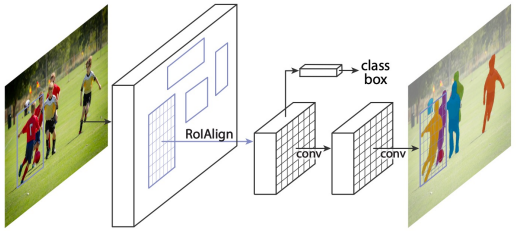
- adaptively highlight the informative regions of the object,
- avoid undesirable information from other objects or the background,
- get rid of (massive) anchors, post-processing is simple without NMS,
- implementation is simple, just matrix multiplications and convolutions.

Motivation

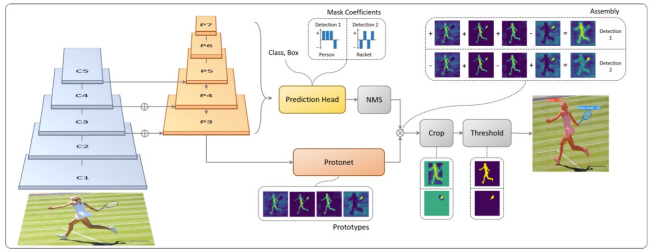
- Previous methods for Instance Segmentation



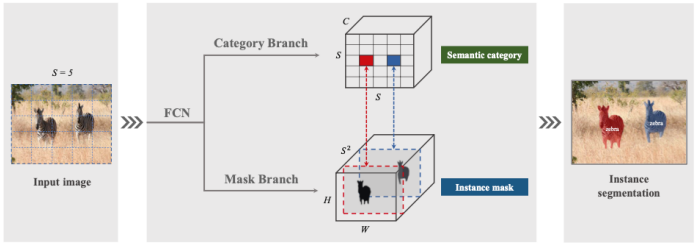
Region-/Pixel-based Method



Mask R-CNN



YOLACT

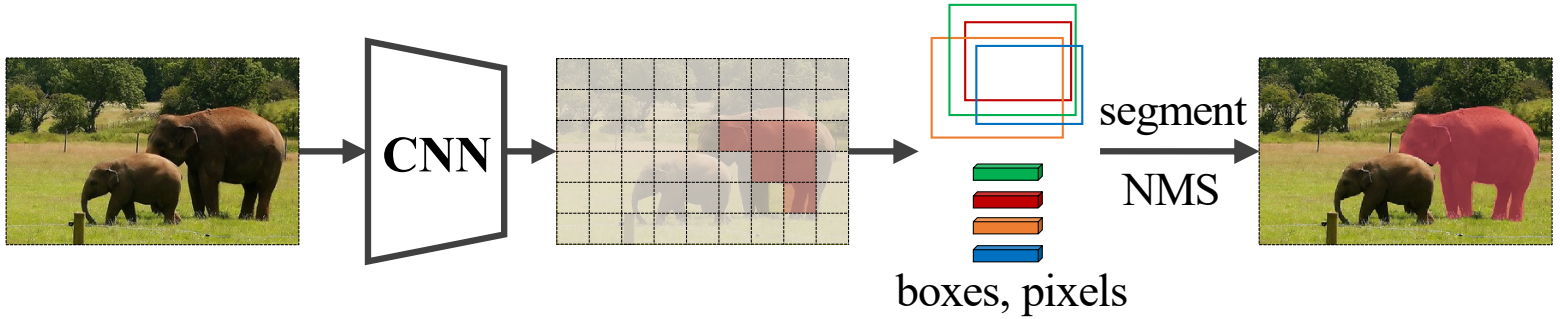


SOLO

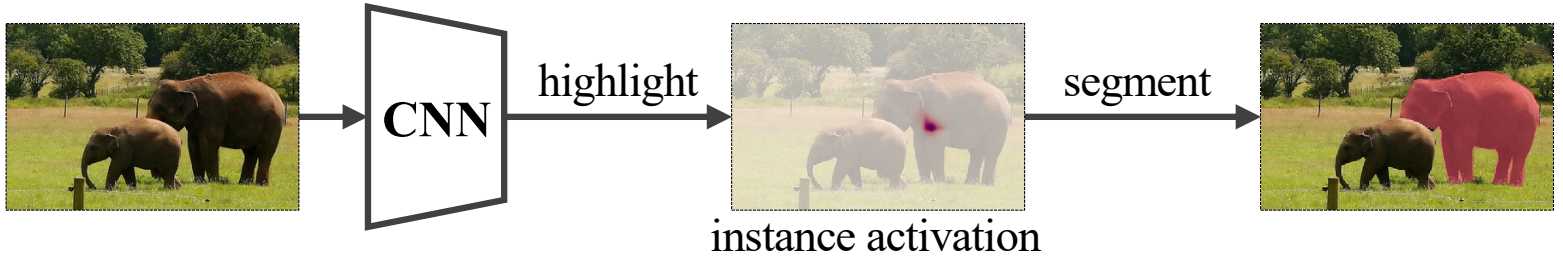
- rely on detectors or bounding boxes.
- require FPN and multi-level prediction.
- require post-processing procedures, e.g., NMS and sorting.

Motivation

- Highligh to Segment



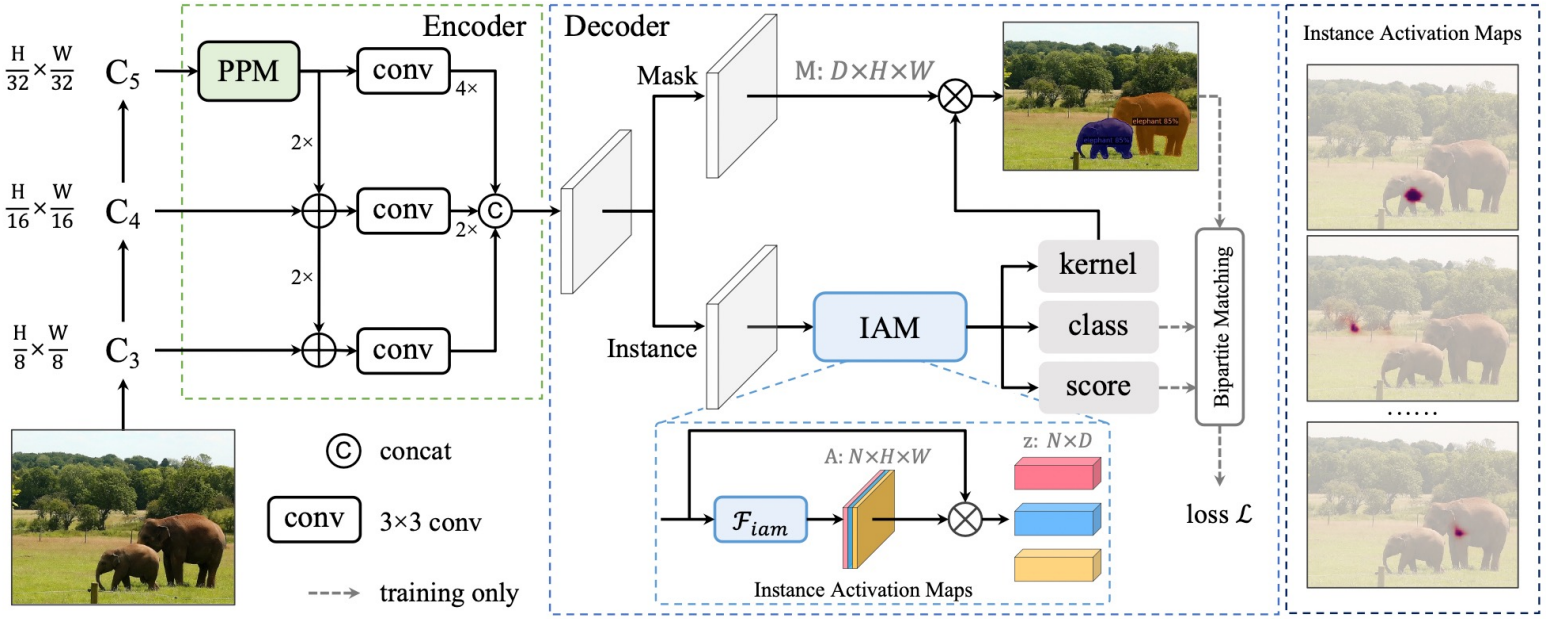
Region-/Pixel-based Method



SparseInst: Highlight to Segment

SparseInst

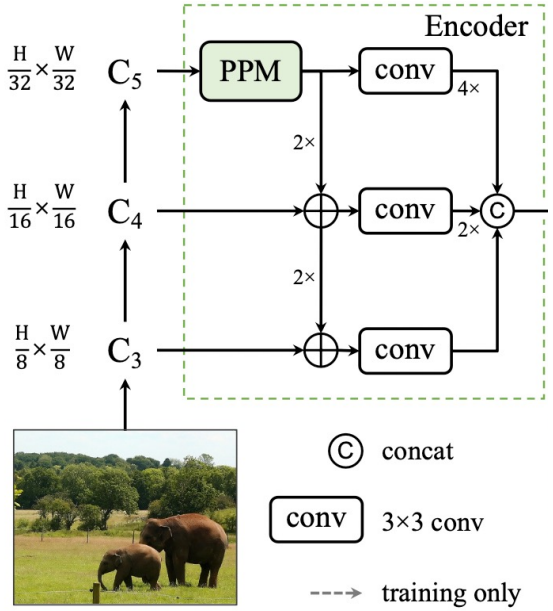
- Framework



- encoder-decoder architecute, fully convolutional framework
- single-level prediction
- no complex post-processing

SparseInst

- Context Encoder

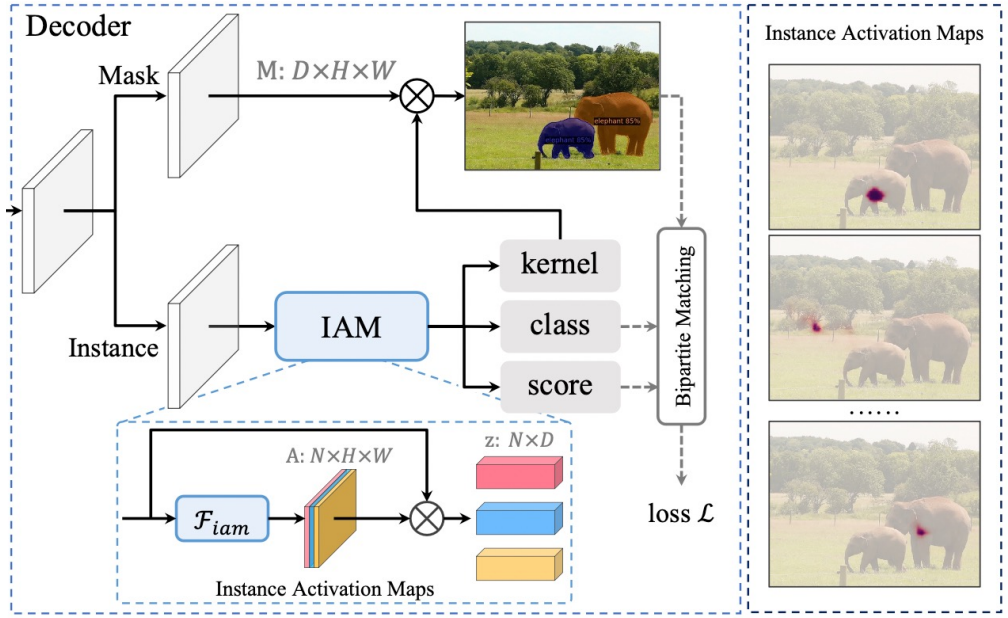


Context Encoder is derived from FPN:

- adopts Pyramid Pooling Module (PPM) for larger receptive fields,
- adopts multi-scale feature fusion,
- outputs single-level features for segmentation, making the inference very fast.

SparseInst

- IAM-based Decoder



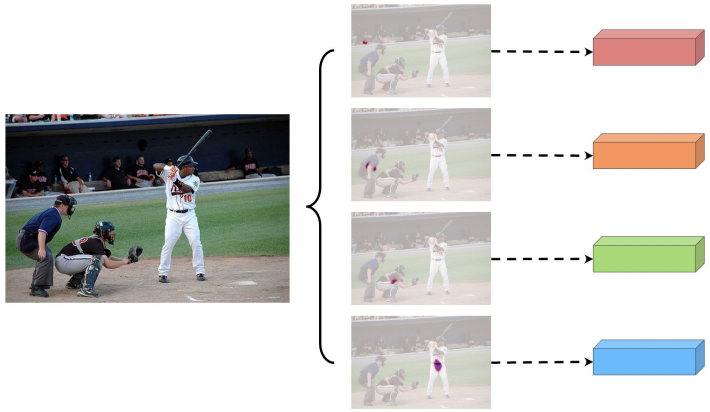
IAM-based Encoder contains:

- Instance Branch: predicting Instance Activation Maps of objects and obtain object features for classification and segmentation.
- Mask Branch: obtain mask features

SparseInst

- ## Instance Activation Maps

Basic Instance Activation Maps (IAM)



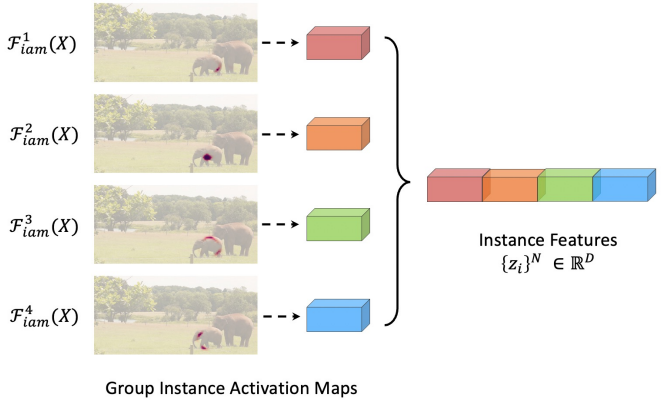
(1) Highlight objects: $A_i = \mathcal{F}_{iam}(X)$

(2) Instance features: $z_i = A_i \cdot X$

(3) Classification and segmentation

The $\mathcal{F}_{iam}$ can be simply implemented by a 3x3 convolution

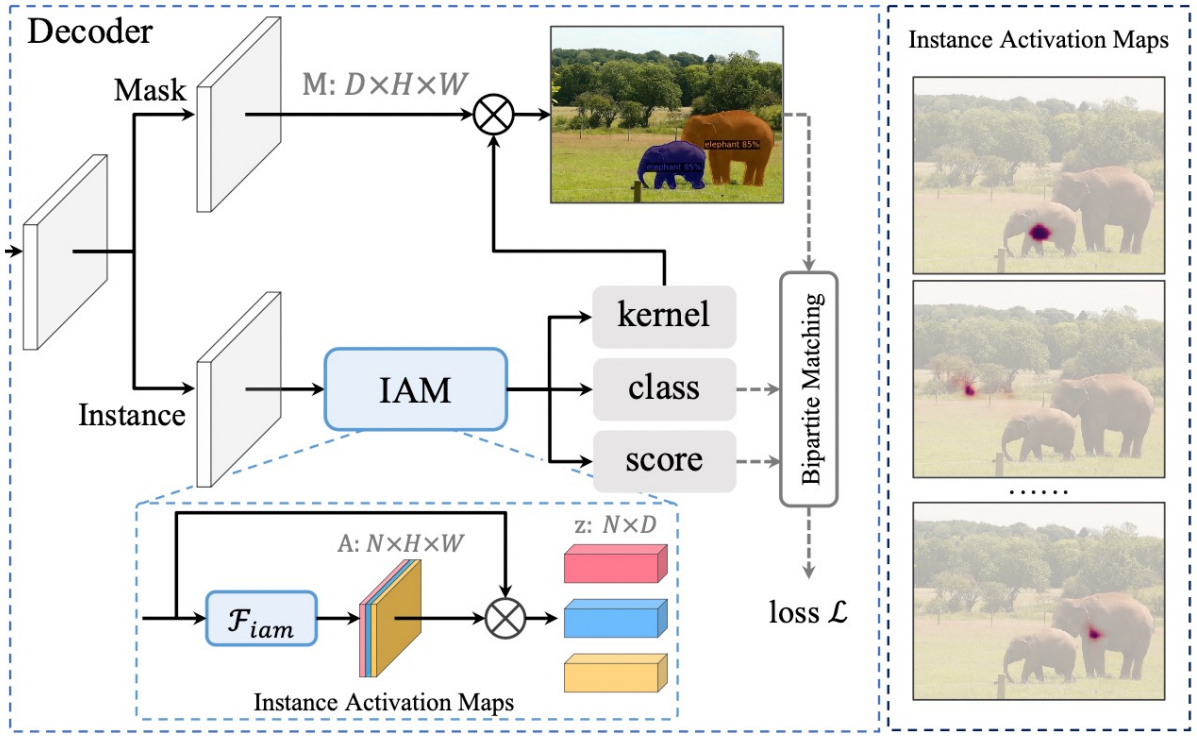
Group Instance Activation Maps (G-IAM)



Highlight one object with a group of instance activation maps and then aggregate features of the group together.

SparseInst

- Model Training



- Dice-based Matching Score

$$\mathcal{C}(i,k) = p_{i,c_k}^{1-\alpha} \cdot \mathrm{DICE}(m_i, t_k)^{\alpha},$$

p_{i,c_k} denotes the probability of class c_k, m_i and t_k denotes the predicted mask and the ground-truth mask

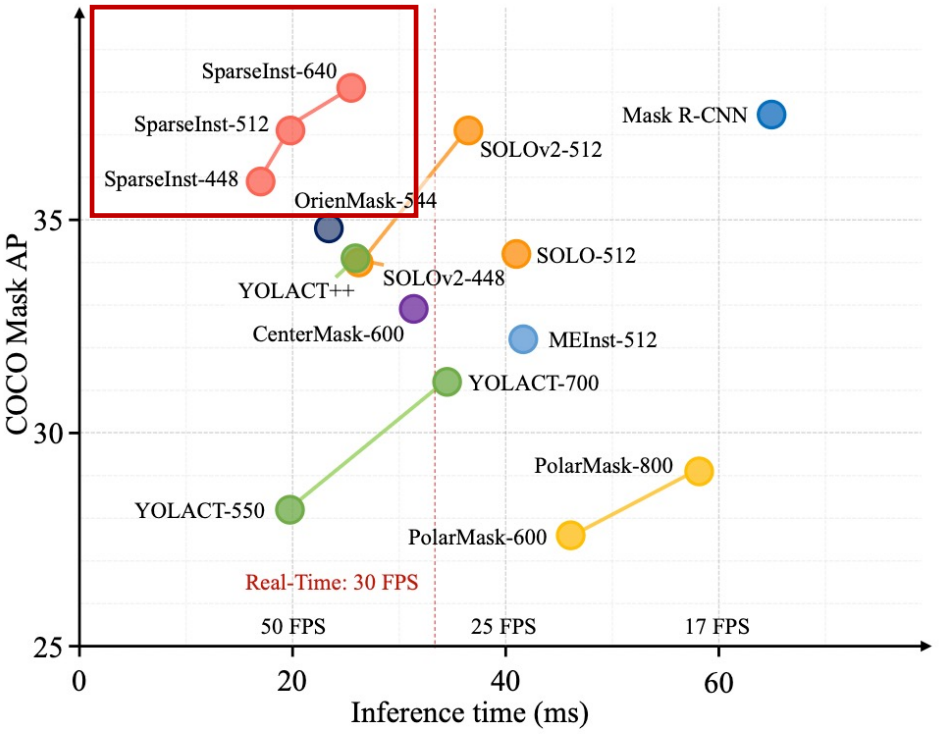
We adopt the **Hungarian Algorithm** to match the ground-truth objects with predicted objects.

Our Results

- Results on COCO test-dev

method	backbone	size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MEInst [44]	R-50-FPN	512	24.0	32.2	53.9	33.0	13.9	34.4	48.7
CenterMask [21]	R-50-FPN	600	31.9	32.9	-	-	12.9	34.7	48.7
CondInst [35]	R-50-FPN	800	20.4†	35.4	56.4	37.6	18.4	37.9	46.9
SOLO [38]	R-50-FPN	512	24.4	34.2	55.9	36.0	-	-	-
SOLOv2-Lite [38]	R-50-FPN	448	38.2	34.0	54.0	36.1	10.3	36.3	54.4
SOLOv2-Lite [38]	R-50-DCN-FPN	512	28.2	37.1	57.7	39.7	12.9	**40.0**	**57.4**
PolarMask [41]	R-50-FPN	600	21.7†	27.6	47.5	28.3	9.8	30.1	43.1
PolarMask [41]	R-50-FPN	800	17.2†	29.1	49.5	29.7	12.6	31.8	42.3
YOLACT [2]	R-50-FPN	550	50.6	28.2	46.6	29.2	9.2	29.3	44.8
YOLACT [2]	R-101-FPN	700	29.0	31.2	50.6	32.8	12.1	33.3	47.1
YOLACT++ [2]	R-50-DCN-FPN	550	38.6	34.1	53.3	36.2	11.7	36.1	53.6
OrienMask [12]	D-53-FPN	544	42.7	34.8	56.7	36.4	**16.0**	38.2	47.8
SparseInst	R-50	608	44.6	34.7	55.3	36.6	14.3	36.2	50.7
SparseInst	R-50-DCN	608	41.6	36.8	57.6	38.9	15.0	38.2	55.2
SparseInst	R-50-d	608	42.8	36.1	57.0	38.2	15.0	37.7	53.1
SparseInst	R-50-d-DCN	608	40.0	**37.9**	**59.2**	**40.2**	15.7	39.4	56.9

448x: 35.9 AP, 58.5 FPS, 🚀



Even using larger input images or larger backbones, SparseInst is still fast.

Our Results

- Instance Activation Maps

IAM	AP	AP_{50}	AP_{75}	t (ms)
1x1 Conv	30.8	50.7	32.0	**32.4**
3x3 Conv	32.0	51.9	33.5	22.9
Group 3x3 Conv (2 groups)	32.2	52.3	33.5	23.1
Group 3x3 Conv (4 groups)	**32.7**	**53.1**	**34.0**	23.3

➡ IAM

➡ Group-IAM

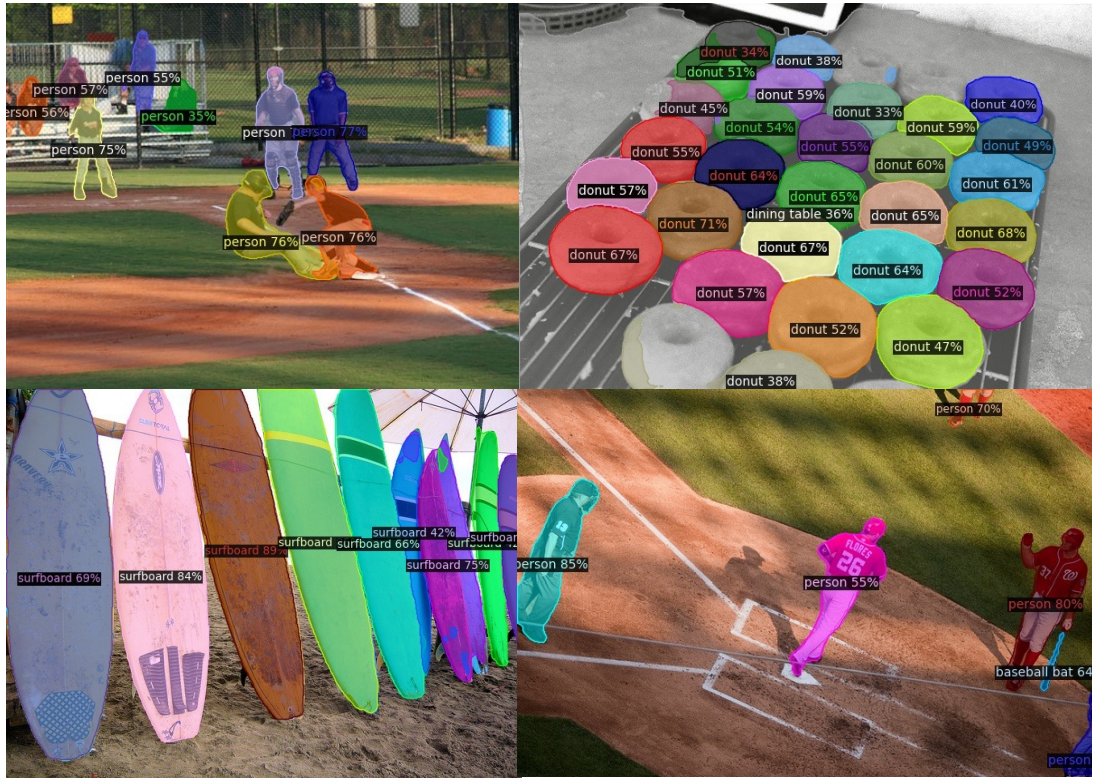
- 3x3 conv brings more locality and larger context than 1x1 conv, and achieves better results.
- G-IAM significantly improves the performance with negligible latency.

Qualitative Results

- Instance Activation Maps and Segmentation Masks

Qualitative Results

- Instance Segmentation



SparseInst can separate and segment objects well even in crowded scenes.

- Comparison with SOLOv2, YOLACT++

SparseInst can segment the objects occluded by other objects better!

Thanks

Tianheng Cheng

thch@hust.edu.cn

Paper

Code