Sparse Instance Activation for Real-Time Instance Segmentation

Tianheng Cheng [1, 2] Xinggang Wang [1†] Shaoyu Chen [1, 2] Wenqiang Zhang [1]
Qian Zhang [2] Chang Huang [2] Zhaoxiang Zhang [3] Wenyu Liu [1]
[1] School of EIC, Huazhong University of Science & Technology [2] Horizon Robotics
[3] Institute of Automation, Chinese Academy of Sciences (CASIA)
{thch,xgwang,shaoyuchen,wq_zhang,liuwy}@hust.edu.cn {qian01.zhang,chang.huang}@horizon.ai
zhaoxiang.zhang@ia.ac.cn

Abstract

In this paper, we propose a conceptually novel, efficient, and fully convolutional framework for real-time instance segmentation. Previously, most instance segmentation methods heavily rely on object detection and perform mask prediction based on bounding boxes or dense centers. In contrast, we propose a sparse set of instance activation maps, as a new object representation, to highlight informative regions for each foreground object. Then instance-level features are obtained by aggregating features according to the highlighted regions for recognition and segmentation. Moreover, based on bipartite matching, the instance activation maps can predict objects in a one-to-one style, thus avoiding non-maximum suppression (NMS) in post-processing. Owing to the simple yet effective designs with instance activation maps, SparseInst has extremely fast inference speed and achieves 40 FPS and 37.9 AP on the COCO benchmark, which significantly outperforms the counterparts in terms of speed and accuracy. Code and models are available at https://github.com/hustvl/SparseInst.

1. Introduction

Instance segmentation aims to generate instance-level segmentation for each object in an image. Based on the advances in deep convolutional neural networks and object detection, recent works [4,9,14,18,40] have made tremendous progress in instance segmentation and achieved impressive results on large-scale benchmarks, *e.g.*, COCO [24]. However, developing real-time and efficient instance segmentation algorithms is still challenging and urgent, especially for autonomous driving and robotics.

Prevalent methods tend to adopt detectors [30, 37] to localize instances first and then segment through region-based convolutional networks [14], dynamic convolutions [36], *etc*. Those methods are conceptually intuitive and achieve great performance. However, when it comes to real-time instance segmentation, those methods suffer from some limitations. Firstly, most methods employ dense anchors (centers) to localize and then segment objects, *e.g.*, more than 5456 instances (given 512×512 input) in CondInst [36], which incur lots of redundant predictions and much computation burden. Besides, the receptive field of each pixel is limited and the contextual information is insufficient if we densely localize objects by centers or anchors [6, 12]. Secondly, most methods require multi-level prediction to handle the scale variation of natural objects, which inevitably increases the latency. Region-based methods [14] apply RoI-Align to acquire region features, making it difficult to deploy algorithms to edge/embedded devices. Finally, the post-processing also requires attention since the sorting and NMS as well as processing masks are time-consuming, es-

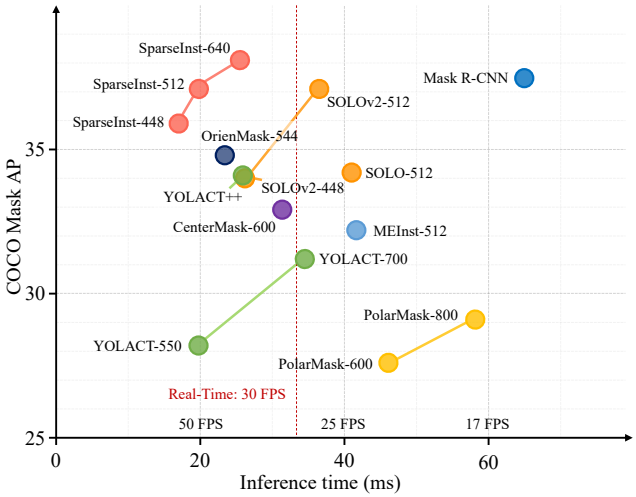
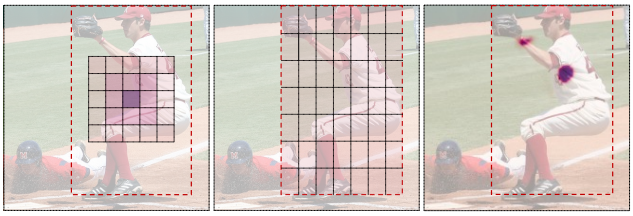


Figure 1. **Speed-and-accuracy Trade-off.** The proposed SparseInst outperforms most state-of-the-art methods in both speed and accuracy for real-time instance segmentation. Inference speeds are measured on one NVIDIA 2080Ti.

[†]Xinggang Wang is the corresponding author.



(a) center-based (b) region-based (c) instance activation map

Figure 2. **Object Representation.** (a) center-based representation may fail to hit the instance; (b) region-based representation may contain features from other instances and background; (c) instance activation map highlights instance-aware pixels.

pecially for dense predictions. It's worth noting that even improved NMS [1, 41] still takes $\sim$ 2ms, 10% of total time.

In this paper, we present a new *highlight to segment* paradigm for real-time instance segmentation. Instead of using boxes or centers to represent objects, we exploit a sparse set of *instance activation maps* (IAM) to highlight informative object regions, which is motivated by CAM [49] widely used in weakly-supervised object localization. Instance activation maps are instance-aware weighted maps and instance-level features can be directly aggregated according to the highlighted regions. Then, recognition and segmentation are performed based on the instance features. Figure 2 compares region-based, center-based, and IAM-based representations. In comparison, IAM has the following advantages: (1) it highlights discriminative instance pixels, suppresses obstructive pixels, and conceptually avoids the incorrect instance feature localization problems in center-/region-based methods; (2) it aggregates instance features from the whole image and offers more contexts; (3) computing instance features with activation maps is rather simple without extra operation like RoI-Align [14]. However, different from previous works [14, 37, 41] using spatial priors (*i.e.*, anchors and centers) to assign targets, instance activation maps are conditioned on the input and arbitrary for different objects and it is infeasible to assign targets with hand-crafted rules for training. To address that, we formulate the label assignment for instance activation maps as a bipartite matching problem, which is recently proposed in DETR [3]. Specifically, each target will be assigned to an object prediction as well as its activation map through Hungarian algorithm [31]. During training, the bipartite matching facilitates the instance activation maps to highlight individual objects and inhibit the redundant predictions, thus avoiding NMS during inference.

Further, we materialize this paradigm and propose SparseInst, an extremely simple but efficient method for instance segmentation. SparseInst adopts single-level prediction and consists of a backbone to extract image features, an encoder to enhance the multi-scale representation for single-level features, and a decoder to compute the instance activation maps, perform recognition and segmentation, as shown in Figure 3. SparseInst is a pure and fully convolutional framework and independent from detectors. Benefiting from the facts: (1) the sparse predictions through the instance activation maps; (2) single-level prediction; (3) compact structures; (4) simple post-processing without NMS or sorting, SparseInst has extremely fast inference speed and achieves 37.9 mask AP on MS-COCO `test-dev` with 40.0 FPS on one NVIDIA 2080Ti GPU, outperforming most state-of-the-art methods for real-time instance segmentation. Given 448$\times$ input, SparseInst achieves 58.5 FPS with competitive accuracy, which is faster than previous methods. We hope the proposed SparseInst can serve as a general framework for (real-time) end-to-end instance segmentation.

2. Related Work

According to object representations, existing methods for instance segmentation can be divided into two groups, *i.e.* region-based methods and center-based methods.

Region-based Methods. Region-based methods rely on object detectors, *e.g.*, Faster R-CNN [30], to detect objects and acquire bounding boxes, and then apply RoI-Pooling [30] or RoI-Align [14] to extract region features for pixel-wise segmentation. Mask R-CNN [14], as the representative method, extends Faster R-CNN by adding a mask branch to predict masks for objects and offers a strong baseline for end-to-end instance segmentation. [9, 19, 35, 45] address the low-quality segmentation and coarse boundaries arising in Mask R-CNN and present several approaches to refine the mask predictions for high-quality masks. [2, 5] exploit cascade structures to progressively improve the object localization for more accurate mask prediction.

Center-based Methods. Recently, many approaches employ the single-stage detectors, especially the anchor-free detectors [37]. These approaches represent objects by center pixels instead of bounding boxes and segment using the center features. Several methods [43, 44] explores the object contours but show some limitations for objects having hollows or multiple parts. YOLACT [1] and maYOLACT [29] generate instance masks by the assembly of mask coefficients and prototype masks. MEInst [46] and CondInst [36] extend FCOS [37] by predicting the encoded mask vector or mask kernels for dynamic convolution [7] respectively. SOLO [40, 41], as a detector-free method, yet localize and recognize objects by centers as well as generating the mask kernels. The proposed SparseInst exploits sparse instance activation maps to represent objects with a simple pipeline and high efficiency.

Bipartite Matching for Object Detection. The bipartite matching has been widely explored for end-to-end object detection [3, 31–34, 39, 51], which avoids NMS in post-processing. Recently, SOLQ [10] and ISTR [17] exploit the

mask encodings for instance segmentation. QueryInst [13] extends [34] by adding dynamic mask heads. Besides, [8,21,38,47] employ transformers with instance and semantic queries to obtain panoptic segmentation results. However, our method aiming at fast speed is motivated by the instance activation maps as object representation for instance-level recognition and segmentation. And the concise yet effective representation drives the framework rather fast.

3. Method

In this section, we first investigate the instance activation maps for representing objects. Then we present a novel framework which exploits the sparse set of instance activation maps to highlight objects and aggregate instance features for instance-level recognition and segmentation.

3.1. Instance Activation Maps

Formulation. Intuitively, instance activation maps are instance-aware weighted maps which aim to highlight the informative regions for each object. And the features from the highlighted regions are semantically abundant and instance-aware for both recognizing and separating objects. Therefore, we directly aggregate the features according to the activation maps as the instance features. Given the input image features $\mathbf{X} \in \mathbb{R}^{D\times(H\times W)}$, instance activation maps can be formulated as: $\mathbf{A} = \mathcal{F}_{iam}(\mathbf{X}) \in \mathbb{R}^{N\times(H\times W)}$, where $\mathbf{A}$ is the sparse set of N instance activation maps and $\mathcal{F}_{iam}(\cdot)$ is a simple network with a sigmoid non-linearity. Then we can obtain the sparse set of instance features by gathering distinctive information from the input feature maps $\mathbf{X}$ with the instance activation maps through: $z = \bar{\mathbf{A}} \cdot \mathbf{X}^T \in \mathbb{R}^{N\times D}$, where $z = \{z_i\}^N$ are the feature representations for N potential objects in the image and $\bar{\mathbf{A}}$ is normalized to 1 for each instance map. The sparse instance-aware features $\{z_i\}^N$ are straightforwardly used for consequent recognition and instance-level segmentation.

Learning Instance Activations. Instance activation maps don't exploit explicit supervisions, *e.g.*, instance masks, for learning to highlight objects. Essentially, the subsequent modules for recognition and segmentation provide instance activation maps with indirect supervisions, which encourage the $\mathcal{F}_{iam}$ to discover informative regions. Additionally, the supervisions are instance-aware due to the bipartite matching, which further enforces the $\mathcal{F}_{iam}$ to discriminate objects and activate only one object per map. Consequently, the proposed instance activation maps are capable to highlight discriminative regions for individual objects.

3.2. SparseInst

As illustrated in Figure 3, SparseInst is a simple, compact, and unified framework which consists of a backbone network, an instance context encoder, and an IAM-based decoder. The backbone network, *e.g.*, ResNet [15], extracts multi-scale features from the given image. The instance context encoder is attached to the backbone to enhance more contextual information and fuse the multi-scale features. For faster inference, the encoder outputs single-level features of $\frac{1}{8}\times$ resolution *w.r.t.* the input image, and the features will be fed to subsequent IAM-based decoder to generate instance activation maps to highlight foreground objects for classification and segmentation.

3.3. Instance Context Encoder

Objects in natural scenes tend to have wide range of scales, which is prone to degrade the performance of detectors. Most approaches adopt multi-scale feature fusions, *e.g.*, feature pyramids [22], and multi-level prediction to facilitate the recognition for objects of different scales. Nevertheless, using multi-level pyramidal features increase the computation burden, especially for detectors using heavy heads [23, 37], as well as producing amounts of duplicate predictions. Conversely, our method aiming at faster inference leverages single-level prediction. Considering the limitations of the single-level features for objects of various scales, we reconstruct the feature pyramid networks and present an instance context encoder, as illustrated in Figure 3. The instance context encoder adopts a pyramid pooling module [48] after C_5 to enlarge the receptive fields and fuses features from P_3 to P_5 to further enhance the multi-scale representations for the output single-level features.

3.4. IAM-based Segmentation Decoder

Figure 3 illustrates the IAM-based segmentation decoder which contains an instance branch and a mask branch. The two branches are composed of a stack of 3×3 convolutions with 256 channels. The instance branch aims to generate instance activation maps and N instance features for recognition and instance-aware kernel. The mask branch is designed to encode instance-aware mask features.

Location-Sensitive Features. Empirically, objects are localized in different positions and the spatial locations can be used as cues to distinguish instances. Hence, we construct two-channel coordinate features which consists of normalized absolute (x, y) coordinates of spatial locations, which is similar to CoordConv [25]. Then we concatenate the output features from the encoder with coordinate features to enhance the instance-aware representation.

Instance Activation Maps $\mathcal{F}_{iam}$. We adopt a simple yet effective 3×3 convolution with sigmoid as the vanilla $\mathcal{F}_{iam}$, which highlights each instance with a single activation map. Accordingly, instance features $\{z_i\}$ are obtained through activation maps, in which each potential object is encoded into a 256-d vector. Then three linear layers are applied for classification, objectness score, and mask kernel $\{w_i\}^N$.

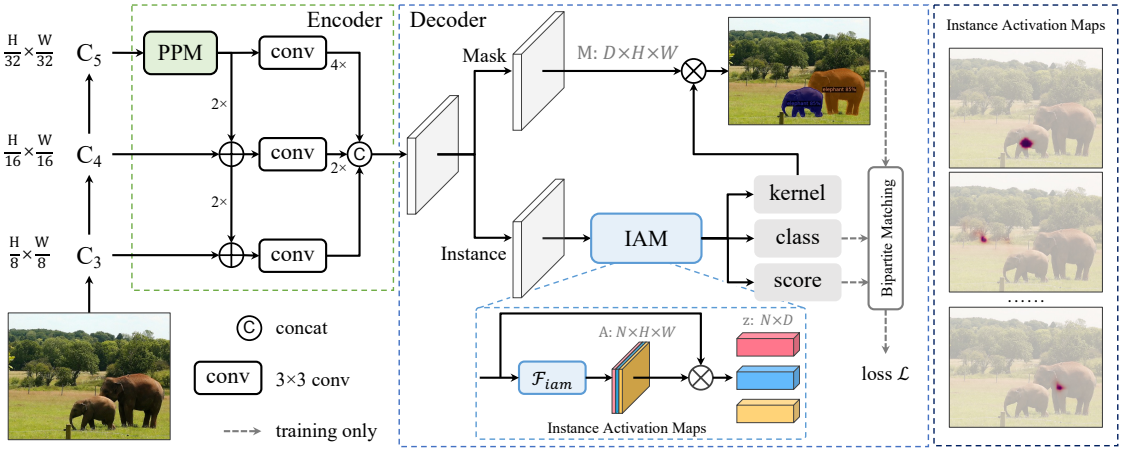


Figure 3. **The architecture of SparseInst.** SparseInst contains three main components: *backbone*, *encoder* and *IAM-based decoder*. Given the input image, the backbone extracts the multi-scale image features (*i.e.*, $\{C_3, C_4, C_5\}$). The encoder employs pyramid pooling module (PPM) [48] to enlarge the receptive field and fuses the multi-scale features. '4×'or '2×' denote the upsampling by a factor 4 or 2. The IAM-based decoder consists of two branches, *i.e.* an instance branch and a mask branch. In the instance branch, the 'IAM' module predicts the instance activation maps (shown in the right column) to acquire the instance features $\{z_i\}^N$ for recognition and mask kernels. The mask branch aims to provide mask features $\mathbf{M}$ and will be multiplied with the predicted kernels to generate segmentation masks.

Further, to obtain fine-grained instance features, we present the *group instance activation maps* (Group-IAM) to highlight a groups of regions for each object, *i.e.*, multiple activation maps per object. Specifically, we adopt a 4-group 3×3 convolution as the $\mathcal{F}_{iam}$ for Group-IAM and aggregate instance features by concatenating features from a group.

IoU-aware Objectness. We discover that the one-to-one assignment will enforce most predictions to be background which may lower the classification confidence and cause misalignments between classification scores and segmentation masks. To alleviate the above issues, we introduce the IoU-aware objectness to adjust the classification outputs. We adopt the estimated IoU between predicted masks and ground-truth masks as the targets for foreground objects. The ground-truth objectness for instances is varied and can facilitate the network to separate instances. Different from [18] using an extra head to predict IoU score based on mask predictions, we only adopt IoUs as the objectness targets. At inference stage, we rescore the classification probability p_i with the IoU-aware objectness s_i and obtain the ultimate probability $\tilde{p_i} = \sqrt{p_i \cdot s_i}$, where i denotes the i-th instance.

Mask Head. With the instance-aware mask kernels $\{w_i\}^N$ generated by the instance branch, the segmentation mask for each instance can be directly produced by $m_i = w_i \cdot \mathbf{M}$, where m_i is the i-th predicted mask and its corresponding kernel is $w_i \in \mathbb{R}^{1\times D}$. $\mathbf{M} \in \mathbb{R}^{D\times H\times W}$ is the mask features. The final segmentation mask will be upsampled (via bilinear interpolation) to $1\times$ *w.r.t.* original resolution.

3.5. Label Assignment and Bipartite Matching Loss

The proposed SparseInst outputs a fixed-size set of predictions and it's difficult to assign ground-truth objects with hand-crafted rules. To tackle the end-to-end training, we formulate the label assignment as bipartite matching [3]. Firstly, we propose a pairwise dice-based *matching score* $\mathcal{C}(i,k)$ for i-th prediction and k-th ground-truth object in Eq. (1), which is determined by classification scores and dice coefficients of segmentation masks.

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

where α is a hyper-parameter to balance the impacts of classification and segmentation and empirically set to 0.8. c_k is termed as the category label for the k-th ground-truth object and p_{i,c_k} indicates the probability for the category c_k of i-th prediction. m_i and t_k are the masks of i-th prediction and k-th ground-truth object respectively. The dice coefficient is defined in Eq. (2).

$$\text{DICE}(m,t) = \frac{2\sum_{x,y} m_{xy} \cdot t_{xy}}{\sum_{x,y} m_{xy}^2 + \sum_{x,y} t_{xy}^2}, \tag{2}$$

where m_{xy} and t_{xy} denote the pixels at (x,y) in the predicted mask m and ground-truth mask t respectively. Then, we adopt Hungarian algorithm [31] to find the optimal match between K ground-truth objects and N predictions.

The training loss is defined in Eq. (3), involving losses for classification, objectness prediction, and segmentation.

$$\mathcal{L} = \lambda_c \cdot \mathcal{L}_{cls} + \mathcal{L}_{mask} + \lambda_s \cdot \mathcal{L}_s, \tag{3}$$

where $\mathcal{L}_{cls}$ is focal loss [23] for object classification, $\mathcal{L}_{mask}$ is the mask loss and $\mathcal{L}_s$ is the binary cross entropy loss for the IoU-aware objectness. Considering the severe imbalance problem between background and foreground in full-resolution instance segmentation, we adopt a hybrid mask

loss in Eq. (4) by combining the dice loss [27] and pixel-wise binary cross entropy loss for segmentation mask.

$$\mathcal{L}_{mask} = \lambda_{dice} \cdot \mathcal{L}_{dice} + \lambda_{pix} \cdot \mathcal{L}_{pix}, \tag{4}$$

where $\mathcal{L}_{dice}$ and $\mathcal{L}_{pix}$ are dice loss and binary cross entropy loss, λ_{dice} and λ_{pix} are corresponding coefficients.

3.6. Inference

The inference stage of SparseInst is much straightforward and concise. Forward the given images through the whole network and we can directly obtain N instances with classification scores $\{\tilde{p}_i\}^N$ and corresponding raw segmentation masks $\{m_i\}^N$. Then we can determine the category and confidence score for each instance and obtain the final binary mask by thresholding. Sorting and NMS are not needed, thus making the inference procedure very fast.

4. Experiments

In this section, we evaluate the accuracy and inference speed of our proposed SparseInst on the challenging MS-COCO dataset and provide detailed ablation studies about our framework as well as qualitative results.

Dataset and Evaluation Metrics. Our experiments are conducted on the COCO dataset [24] which consists of 118k images for training, 5k for validation and 20k for testing. All models are trained on `train2017` and evaluated on `val2017`. As for instance segmentation, we mainly report the AP for segmentation mask. For inference speed, we measure the frames per second (FPS) including the post-processing on one NVIDIA 2080Ti GPU. TensorRT or FP16 is not used for acceleration.

Implementation Details. SparseInst is built on Detectron2 [42] and trained over 8 GPUs with a total of 64 images per mini-batch. Following the training schedule in [33], we adopt AdamW [26] optimizer with a small initial learning rate 5×10^{-5} with weight decay 0.0001. All models are trained for 270k iterations and learning rate is divided by 10 at 210k and 250k respectively. The backbone is initialized with the ImageNet-pretrained weights with frozen batchnorm layers and other modules are randomly initialized. We adopt random flip and scale jitter in training. The shorter side of images are randomly sampled from 416 to 640 pixels, while the longer side is less or equal to 864. Unless specified, we evaluated the speed and accuracy with the shorter size 640. Loss coefficients λ_c, λ_{dice}, λ_{pix}, and λ_s are empirically set to 2.0, 2.0, 2.0, and 1.0 respectively. We adopt N=100 instances for each image. Besides, we provide a MindSpore [28] implementation of SparseInst.

4.1. Main Results

Since the SparseInst aims for real-time instance segmentation, we mainly compare SparseInst with the state-of-the-art methods towards real-time instance segmentation with respect to accuracy and inference speed. Results are evaluated on COCO `test-dev`. We provide SparseInst with group instance activation maps and different backbones to achieve the trade-off between speed and accuracy. We adopt ResNet-50 [15] to reach higher inference speed and its variant ResNet-d [16] to achieve better accuracy but with higher latency and aim for providing a stronger baseline for real-time instance segmentation. Additionally, we adopt a simple random crop and larger weight decay (0.05) to better compare with OrienMask [11] and YOLACT [1]. Table 1 shows that our SparseInst is superior to most real-time methods with better performance and faster inference speed. SparseInst outperforms the popular real-time approach YOLACT by a remarkable margin with faster speed. Figure 1 illustrates the speed-accuracy trade-off curve and the proposed SparseInst with R50-d and DCN [50] obtains better trade-off compared with the counterparts and achieves 58.5 FPS and 35.5 mask AP with 448× input, which is superior to most real-time methods ($\geq$ 30FPS).

4.2. Ablation Experiments

We conduct a series of ablations to investigate SparseInst, including experimental details about the components.

Instance Context Encoder. Table 2 shows the impacts of the modifications to the vanilla feature pyramids [22]. Adding the pyramid pooling module for larger receptive fields and more object contexts brings significant improvement by 1.5 AP and 2.2 AP for larger objects (AP_L) while incurs negligible latency. Moreover, fusing the multi-scale features from P_3 to P_5 further enhances the multi-scale feature representation and improves the performance by 0.7 AP and 2.0 AP_L. The context encoder is rather essential for single-level prediction to cope with the limited receptive fields and provide better multi-scale features, thus bridging the gap between multi-level and single-level methods.

Structure of the Decoder. In Table 3, we compare different structures of the two branches in the IAM-based Decoder. We adopt 4 conv layers with 256 channels as the basic setting for both branches and evaluate the performance of models with different depths or widths. Reducing width or reducing depth will lower the performance but increase the inference speed and it's worth noting that reducing channels to 128 performs worse. Increasing the depth from 4 to 6 brings 0.4 AP improvement. Considering the trade-off between speed and accuracy, we adopt width=256 and depth=4 in all experiments. Adding coordinate features improves the baseline by 0.5 AP with negligible time consumption, which indicates the effect of the explicit location-aware features as discussed in §3.4. Table 3 also shows the effects of replacing the last convolution of the two branches with a deformable convolution. Using deformable convolu-

method	backbone	size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MEInst [46]	R-50-FPN	512	24.0	32.2	53.9	33.0	13.9	34.4	48.7
CenterMask [20]	R-50-FPN	600	31.9	32.9	-	-	12.9	34.7	48.7
CondInst [36]	R-50-FPN	800	20.4†	35.4	56.4	37.6	18.4	37.9	46.9
SOLO [40]	R-50-FPN	512	24.4	34.2	55.9	36.0	-	-	-
SOLOv2-Lite [40]	R-50-FPN	448	38.2	34.0	54.0	36.1	10.3	36.3	54.4
SOLOv2-Lite [40]	R-50-DCN-FPN	512	28.2	37.1	57.7	39.7	12.9	**40.0**	**57.4**
PolarMask [43]	R-50-FPN	600	21.7†	27.6	47.5	28.3	9.8	30.1	43.1
PolarMask [43]	R-50-FPN	800	17.2†	29.1	49.5	29.7	12.6	31.8	42.3
YOLACT [1]	R-50-FPN	550	50.6	28.2	46.6	29.2	9.2	29.3	44.8
YOLACT [1]	R-101-FPN	700	29.0	31.2	50.6	32.8	12.1	33.3	47.1
YOLACT++ [1]	R-50-DCN-FPN	550	38.6	34.1	53.3	36.2	11.7	36.1	53.6
OrienMask [11]	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

Table 1. **COCO Instance Segmentation.** Comparisons with state-of-the-art methods for mask AP and speed on COCO `test-dev`. Inference speeds of all models are tested on our machine with one NVIDIA RTX 2080Ti except those marked with †, which are inherited from their publications.

w/ fusion	w/ PPM	t (ms)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
		22.0	29.8	48.7	31.0	12.0	31.8	44.1
	✓	22.2	31.3	50.8	32.4	**14.0**	33.2	46.2
✓		22.8	30.3	49.5	31.6	12.5	32.3	45.9
✓	✓	22.9	**32.0**	**52.0**	**33.3**	13.1	**34.5**	**48.2**

Table 2. **Ablation on the Instance Context Encoder.** The vanilla encoder [22] is incapable for single-level prediction. Leveraging PPM can enlarge the receptive fields and significantly improve the overall performance and adding multi-scale fusion further improves the accuracy, especially for AP_L. Notably, the extra latency of the improved encoder compared to the vanilla one is negligible.

depth	width	coord?	dconv?	AP	AP_S	AP_M	AP_L	t (ms)
4	256			31.5	13.4	33.5	47.9	22.9
4	256	✓		32.0	13.0	34.5	48.2	22.9
4	256	✓	✓	**32.6**	13.1	34.8	**49.2**	24.6
2	256	✓		31.0	12.9	33.2	47.0	20.6
6	256	✓		32.4	**13.7**	**35.4**	47.9	25.5
4	128	✓		30.6	12.4	32.5	46.2	**19.7**

Table 3. **Ablation on the structure of the decoder.** 'coord.' denotes coordinates and 'dconv.' denotes deformable convolution. Adding coordinates brings 0.5 AP improvement but with negligible latency. Replacing the last convolution with deformable convolution gives significant improvement on larger objects (AP_L). Reducing the width or depth improves the inference speed but lower the performance, while increasing the depth can further improve the accuracy but lower the speed.

tion [50] is optional and improves larger objects by enlarging the receptive field but consumes much time (+1.7ms).

Instance Activation Maps. $\mathcal{F}_{iam}$ is the key component for highlighting object regions, and we explore different designs for $\mathcal{F}_{iam}$ in Table 4. Using softmax or 1×1 conv brings 0.4 AP and 1.2 AP drop, respectively. Sigmoid (w/ norm) and softmax can be formulated as $s_i = \frac{f(x_i)}{\sum_k f(x_k)}$ where $f(x) = e^x$ for softmax and $f(x) = \frac{1}{1+e^{-x}}$ for sigmoid, which tends to saturate thus activate larger regions then softmax. Adding extra 3×3 conv brings no gain but increases the computation cost. Further, we evaluate the Group-IAM with different groups and Table 4 shows that using 4 groups improves the model by 0.7 AP.

$\mathcal{F}_{iam}$	act.	AP	AP_{50}	AP_{75}	t (ms)
3×3 conv	sigmoid	32.0	51.9	33.5	22.9
3×3 conv	softmax	31.6	51.4	32.9	22.9
1×1 conv	sigmoid	30.8	50.7	32.0	**22.4**
3×3 conv, ReLU, 3×3 conv	sigmoid	31.9	52.2	33.0	23.6
Group 3×3 conv (2 groups)	sigmoid	32.2	52.3	33.5	23.1
Group 3×3 conv (4 groups)	sigmoid	**32.7**	**53.1**	**34.0**	23.3

Table 4. **Ablation on $\mathcal{F}_{iam}$.** Using softmax or 1×1 conv brings 0.4 AP and 1.2 AP drop respectively, and using two 3×3 conv with ReLU brings no gain. However, Group-IAM with 4 groups obtains 0.7 AP improvement.

Hybrid Mask Loss. In Table 5, we analyze the effects of the hybrid mask loss. Notably, dice loss is the critical component for mask prediction and removing dice loss lead to the collapse (AP rapidly drops 8.1 points). Compared to RoI-based methods [14], full-resolution instance segmentation has severe imbalance problem between background and foreground, especially for small objects which may occupy less than 0.5% pixels. Dice loss is more robust to the foreground/background imbalance thus effective to handle the full-resolution segmentation. In Table 5, adding a pixel-wise classification loss can further improve the segmentation accuracy: using binary cross-entropy loss (BCE) or focal loss improves by 1.0 AP and 0.5 AP respectively. Moreover, we note that pixel-wise loss significantly improves AP_L (*e.g.*, +1.8 AP from BCE) for large objects. Addition-

Dice	Focal	BCE	AP	AP_{50}	AP_{75}	AP_L
		✓	23.9	40.2	24.3	40.8
✓			31.0	50.8	32.0	46.4
✓	✓		31.5	51.6	32.7	47.5
✓		✓	**32.0**	**52.0**	**33.3**	**48.2**

Table 5. **Ablation on the hybrid mask loss.** We evaluate the effects of the different hybrid mask loss. Dice loss is an essential component and adding extra BCE loss can further improve the performance (+1.0 AP) especially for larger objects (+1.8 AP_L).

w/ obj.	rescore?	loss	AP	AP_{50}	AP_{75}
✗	-	-	30.7	51.3	31.6
✓	✗	CE	31.4	**52.1**	32.2
✓	✓	CE	**32.0**	52.0	**33.3**
✓	✓	L1	31.5	51.3	32.7

Table 6. **Ablation on the IoU-aware objectness.** Adding objectness facilitates more instance-aware features and improves the performance even without rescoring. Using cross-entropy loss obtains better results than L1 loss.

$\mathcal{F}_{iam}$	AP	AP_{50}	AP_{75}	t (ms)
1×1 conv	30.8	50.7	32.0	**22.4**
3×3 conv	32.0	51.9	33.5	22.9
Group 3×3 conv	**32.7**	**53.1**	**34.0**	23.3
Cross Attention	31.8	51.7	33.1	23.4

Table 7. **Comparison with cross attention.** We evaluate the performance of directly using one 4-head cross attention [3] with 100 queries to segment objects. Notably, (Group-) IAM with 3×3 conv can offer better results

size	backbone	encoder	decoder	post
512	10.0 (54.3%)	2.5 (13.5%)	4.1 (22.2%)	1.8 (10.0%)
640	13.3 (55.6%)	2.9 (12.1%)	5.6 (23.4%)	2.1 (8.90%)

Table 8. **Inference time.** We report the inference latency of module of the SparseInst. The backbone consumes more than 50% of the total time.

ally, increasing the weight for pixel-wise loss (λ_{pix}), *e.g.*, 5.0, will bring some improvements.

IoU-aware Objectness. We further conduct ablations to investigate the effects of the proposed IoU-aware objectness method. In Table 6, employing the IoU-aware objectness can improve the baseline by 1.3 AP. Interestingly, we observe that adding objectness prediction without rescoring still brings 0.7 AP improvements, which has no direct impact to classification or segmentation. The targets for objectness differs among foreground instances and therefore the objectness loss can facilitate the instance branch to learn more instance-aware features for distinguishing objects as discussed in §3.4. We also compare different types of loss, *i.e.*, L1 loss and cross-entropy, for IoU-aware objectness and Table 6 shows the superiority of using cross-entropy.

4.3. Timing

Our framework achieves fast inference speed for since it saves much computation costs by using single-level prediction, highlighting a sparse set of instances, fully convolutional design, and adopting extremely simple post-processing without sorting or NMS. To better understand the efficiency of the proposed method, we measure the inference latency of each module (*i.e.*, backbone, encoder, decoder, and post-processing). We disable the asynchronous execution in GPU for accurately recording the time, which slows down the overall inference speed. Table 8 shows the inference latency (ms) of each module in SparseInst with different input resolutions. It's worth noting that the backbone (*i.e.*, ResNet-50) consumes most of the inference time and the post-processing inevitably requires nearly 2ms to process the final segmentation and recognition results for evaluation. The 3×3 convolutions in the decoder take much time and can be pruned for more efficient inference.

4.4. Comparison with Cross Attention

The proposed IAM has some connections with query-based methods [3, 8, 38, 47]. The cross attention between object queries $\mathbf{Q}$ and image features $\mathbf{X}$ can be briefly formulated by: $\mathbf{A} = \mathbf{QX}$ and $\mathbf{O} = \text{Softmax}(\mathbf{A})\mathbf{X}^T$, where $\mathbf{A}$ and $\mathbf{O}$ are attention maps and output queries. The cross attention has similar formulations with IAM in §3.1 especially for 1×1 conv, which can be viewed as 1-head cross attention. Differently, we adopt the 3×3 conv as $\mathcal{F}_{iam}$ to highlight object regions, which acts as a direct spatial object representation. Compared to queries or 1×1 conv, 3×3 conv perceives larger context and local patterns for instance recognition. Further, we replace IAM with a 4-head cross attention and 100 queries to generate instance features, and Table 7 shows that the 4-head cross attention drops 0.2 AP or 0.9 AP compared to IAM and Group-IAM, respectively.

4.5. Visualizations

Instance Activation Maps. Figure 4 provides the visualizations for instance activation maps and corresponding segmentation masks. Each instance activation map highlights a prominent region of the object. Segmentation masks are well-localized and aligned with the instance activation maps. Moreover, instance activation maps can highlight objects in despite of the scales, positions, categories and also perform well for crowd scenes.

For a better understanding of how the instance activation maps can discriminate objects, we further provide the visualizations of the instance activation maps from all images. Figure 6 illustrates 12 (of 100) instance activation maps by averaging the activation response over the 5,000 images from COCO `val2017`. Different instance activation maps highlight regions of different spatial locations, scales, and shapes, which contributes to separating the instances of the same or different categories.

Qualitative Results. Figure 5 shows the qualitative results of SparseInst. The proposed SparseInst can generate precise segmentation masks with fine boundaries. For crowd and dense scenes, SparseInst can also distinguish different instances well.

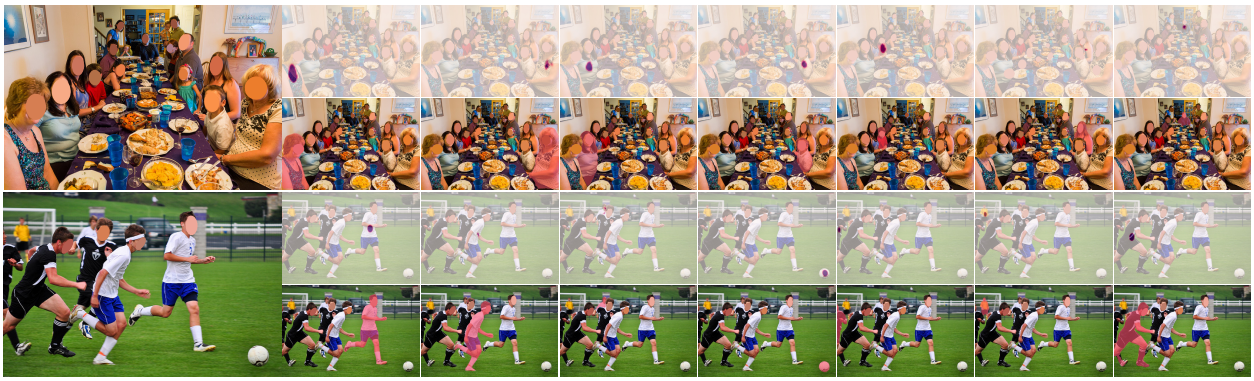


Figure 4. **Visualizations for Instance Activation Maps.** We present the visualizations of the instance activation maps and segmentation masks. For each input image, the upper row shows the instance activation maps and the bottom row shows the corresponding segmentation masks. The instance activation maps tend to highlight the discriminative regions of the objects regardless of the scales, occlusion, and poses. Best viewed on screen after zooming in.

Figure 5. **Visualizations for Instance Segmentation.** The results are obtained by SparseInst on COCO `val2017`. The confidence threshold is set to 0.4. We can observe that SparseInst can generate precise boundaries, highlight and segment well on the crowd scenes, and cope with the scale-variant segmentation.

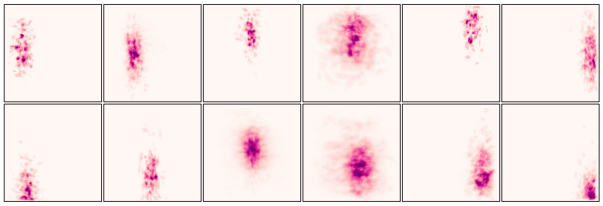


Figure 6. **Visualizations for Instance Activation Maps over the COCO dataset.** We gather the 100 instance activation maps over the 5,000 images from the COCO `val2017` by averaging the activation responses for each map. Instance activation maps from different images are resized to the same size 512×512. We provide 12 instance activation maps for visualization.

5. Conclusion

In this work, we have explored a novel object representation by instance activation maps, which are instance-aware weighted maps and aim to highlight informative regions of objects. Then we present a new *highlight to segment* paradigm to exploit a sparse set of instance activation maps to highlight objects and aggregate instance features according to the activation maps for instance-level recognition and segmentation. Following this paradigm, we propose SparseInst, a conceptually novel and efficient end-to-end framework, which achieves rather fast inference speed with highly competitive accuracy for real-time instance segmentation. Extensive experiments and qualitative results have demonstrated the effectiveness of the core idea and the superiority of the trade-off between speed and accuracy. Finally, we hope SparseInst can serve as a general framework for end-to-end real-time instance segmentation and be applied to practical scenes for its effectiveness and efficiency.

Acknowledgement. This work was in part supported by NSFC (No. 61876212 and No. 61733007) and CAAI-Huawei MindSpore Open Fund.

Limitations. SparseInst along with previous methods [1, 40, 41, 46] perform worse on small objects (AP_S) and we conjecture that the lack of high-resolution features (*e.g.*, P_2) or high-resolution input limits the performance on AP_S and will continue to tackle it in future research.

References

[1] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. YOLACT: real-time instance segmentation. In *ICCV*, 2019. 2, 5, 6, 8

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: high quality object detection and instance segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1483–1498, 2021. 2

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 2, 4, 7

[4] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. In *CVPR*, 2019. 1

[5] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. In *CVPR*, 2019. 2

[6] Qiang Chen, Yingming Wang, Tong Yang, Xiangyu Zhang, Jian Cheng, and Jian Sun. You only look one-level feature. In *CVPR*, 2021. 1

[7] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *CVPR*, 2020. 2

[8] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *arXiv preprint arXiv: 2107.06278*, 2021. 3, 7

[9] Tianheng Cheng, Xinggang Wang, Lichao Huang, and Wenyu Liu. Boundary-preserving Mask R-CNN. In *ECCV*, 2020. 1, 2

[10] Bin Dong, Fangao Zeng, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. SOLQ: segmenting objects by learning queries. In *NeurIPS*, 2021. 2

[11] Wentao Du, Zhiyu Xiang, Shuya Chen, Chengyu Qiao, Yiman Chen, and Tingming Bai. Real-time instance segmentation with discriminative orientation maps. In *ICCV*, 2021. 5, 6

[12] Yang et.al. Dense reppoints: Representing visual objects with dense point sets. In *ECCV*, 2020. 1

[13] Yuxin Fang, Shusheng Yang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Instances as queries. In *ICCV*, 2021. 3

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2, 6

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 5

[16] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. *arXiv preprint arXiv: 1812.01187*, 2018. 5

[17] Jie Hu, Liujuan Cao, Yao Lu, Shengchuan Zhang, Yan Wang, Ke Li, Feiyue Huang, Ling Shao, and Rongrong Ji. ISTR: end-to-end instance segmentation with transformers. *arXiv preprint arXiv: 2105.00637*, 2021. 2

[18] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring R-CNN. In *CVPR*, 2019. 1, 4

[19] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross B. Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 2

[20] Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In *CVPR*, 2020. 6

[21] Zhiqi Li, Wenhai Wang, Enze Xie, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, Tong Lu, and Ping Luo. Panoptic segformer. *arXiv preprint arXiv: 2109.03814*, 2021. 3

[22] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 3, 5, 6

[23] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 3, 4

[24] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014. 1, 5

[25] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *NeurIPS*, 2018. 3

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[27] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016. 5

[28] MindSpore. https://github.com/mindspore-ai/mindspore. 5

[29] Kemal Oksuz, Baris Can Cam, Fehmi Kahraman, Zeynep Sonat Baltaci, Sinan Kalkan, and Emre Akbas. Mask-aware iou for anchor assignment in real-time instance segmentation. In *BMVC*, 2021. 2

[30] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017. 1, 2

[31] Russell Stewart, Mykhaylo Andriluka, and Andrew Y. Ng. End-to-end people detection in crowded scenes. In *CVPR*, 2016. 2, 4

[32] Peize Sun, Yi Jiang, Enze Xie, Wenqi Shao, Zehuan Yuan, Changhu Wang, and Ping Luo. What makes for end-to-end object detection? In *ICML*, 2021. 2

[33] Peize Sun, Yi Jiang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. OneNet: Towards end-to-end one-stage object detection. *arXiv preprint arXiv: 2012.05780*, 2020. 2, 5

[34] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse R-CNN: end-to-end object detection with learnable proposals. In *CVPR*, 2021. 2, 3

[35] Chufeng Tang, Hang Chen, Xiao Li, Jianmin Li, Zhaoxiang Zhang, and Xiaolin Hu. Look closer to segment better: Boundary patch refinement for instance segmentation. In *CVPR*, 2021. 2

[36] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020. 1, 2, 6

[37] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: fully convolutional one-stage object detection. In *ICCV*, 2019. 1, 2, 3

[38] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan L. Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *CVPR*, 2021. 3, 7

[39] Jianfeng Wang, Lin Song, Zeming Li, Hongbin Sun, Jian Sun, and Nanning Zheng. End-to-end object detection with fully convolutional network. In *CVPR*, 2020. 2

[40] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. SOLO: segmenting objects by locations. In *ECCV*, 2020. 1, 2, 6, 8

[41] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. In *NeurIPS*, 2020. 2, 8

[42] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 5

[43] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *CVPR*, 2020. 2, 6

[44] Wenqiang Xu, Haiyang Wang, Fubo Qi, and Cewu Lu. Explicit shape encoding for real-time instance segmentation. In *ICCV*, 2019. 2

[45] Yuhui Yuan, Jingyi Xie, Xilin Chen, and Jingdong Wang. Segfix: Model-agnostic boundary refinement for segmentation. In *ECCV*, 2020. 2

[46] Rufeng Zhang, Zhi Tian, Chunhua Shen, Mingyu You, and Youliang Yan. Mask encoding for single shot instance segmentation. In *CVPR*, 2020. 2, 6, 8

[47] Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. K-net: Towards unified image segmentation. In *NeurIPS*, 2021. 3, 7

[48] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 3, 4

[49] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. In *CVPR*, 2016. 2

[50] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets V2: more deformable, better results. In *CVPR*, 2019. 5, 6

[51] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *ICLR*, 2021. 2