PointINS: Point-based Instance Segmentation

Lu Qi‡, Yi Wang‡, Yukang Chen‡, Ying-Cong Chen, Xiangyu Zhang* , Jian Sun, Jiaya Jia

Abstract—In this paper, we explore the mask representation in instance segmentation with Point-of-Interest (PoI) features. Differentiating multiple potential instances within a single PoI feature is challenging, because learning a high-dimensional mask feature for each instance using vanilla convolution demands a heavy computing burden. To address this challenge, we propose an instance-aware convolution. It decomposes this mask representation learning task into two tractable modules as instance-aware weights and instance-agnostic features. The former is to parametrize convolution for producing mask features corresponding to different instances, improving mask learning efficiency by avoiding employing several independent convolutions. Meanwhile, the latter serves as mask templates in a single point. Together, instance-aware mask features are computed by convolving the template with dynamic weights, used for the mask prediction. Along with instance-aware convolution, we propose PointINS, a simple and practical instance segmentation approach, building upon dense one-stage detectors. Through extensive experiments, we evaluated the effectiveness of our framework built upon RetinaNet and FCOS. PointINS in ResNet101 backbone achieves a 38.3 mask mean average precision (mAP) on COCO dataset, outperforming existing point-based methods by a large margin. It gives a comparable performance to the region-based Mask R-CNN [19] with faster inference.

Index Terms—Instance Segmentation, Single-Point Feature.

1 Introduction

Instance segmentation aims to detect all objects from images at the pixel-level, and involves both object detection [17], [18], [54], [36], [14] and semantic segmentation [43], [71], [9]. Instance segmentation is a critical task in computer vision research community, and plays an indispensable role in a variety of real-world applications, such as autonomous vehicles [49], robotics [44], video surveillance [57], etc. Considering its both academic and industrial values, increasing the effectiveness and efficiency of instance segmentation is an important challenge.

In formulation, instance segmentation requires the semantics, identities, and pixel-level locations of objects. Existing studies [19], [40], [24], [61], [10] demonstrate that learning a proper *mask representation* for characterizing objects' identities and locations remains a vital and open problem. Most state-of-the-art (SOTA) instance segmentation methods [19], [40], [8], [24], [31], [48] construct mask representations using Region-of-Interests (RoIs) features. The most representative method is Mask R-CNN [19], a region-based convolutional neural network that considers instance masks as refined forms from the corresponding bounding boxes. Specifically, it extracts RoIs features from regions containing potential instances [18], [54], [36] and then maps these features to detect objects' bounding boxes and to segment instances' masks. Although RoI features are sufficiently expressive to represent regions for final mask prediction, obtaining RoI features is a complicated process.

To achieve a simple and effective mask representation, several recent approaches, such as TensorMask [10], PolarMask [64] and MEInst [70], directly use single-point features, also known as points-of-interests (PoIs), to learn objects' masks. Based on the assumption that *each single-point feature corresponds to only one or two potential instances*, these approaches give a mask representation from each PoI by the *vanilla convolution* (top row of Fig. 1), building upon dense one-stage detectors, such as *e.g. RetinaNet [37] or FCOS [60]*.

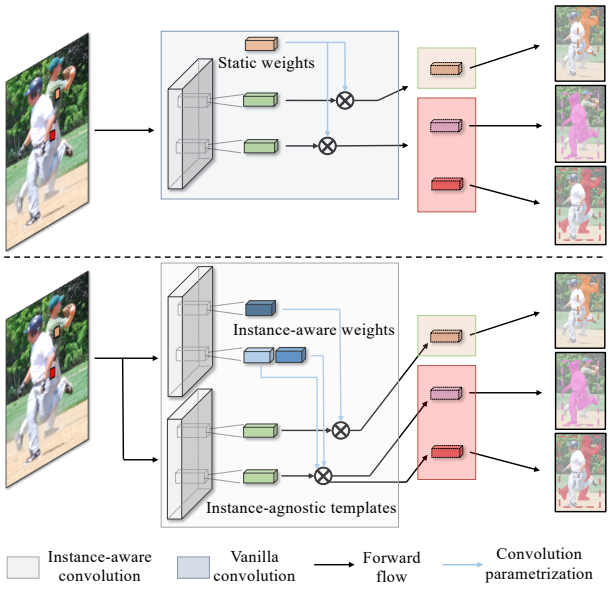


Fig. 1. The core module instance-aware convolution of the proposed PointINS framework in instance segmentation, along with the comparison with the vanilla convolution. By introducing the multiple instance-aware weights from the same point-of-interest (PoI) feature, we obtain different instance masks by instance-aware convolution.

Despite their competitive efficiency in instance segmentation, their performance and scalability are restricted by the adopted assumption and design. In contrast to their original assumption, *one PoI may correspond to multiple instances* (i.e.,

‡*Part of the work is done during the authors' internship in MEGVII Technology. They share the equal contributions.*
* *indicates the corresponding author.*

n instances where $n = 9$ in RetinaNet [37] and YOLO [53]). it is a more intuitive hypothesis due to the common aggregations and occlusions in the real world. If we switch to this more general assumption, these methods would meet the computational bottleneck. Because the assumption requires them to have around n times larger mask feature representations by employing a n times larger standard convolutions for predicting masks, which is empirically infeasible. A smaller n or a relatively low-dimensional mask presentation could avoid this issue, but their performance is compromised as mentioned.

To address the computational feasibility in the assumption that each PoI feature may cover multiple instances, we study to generate *high-dimensional instance-aware* mask representations from PoI features by our proposed *instance-aware convolution*. It decomposes the mask learning into two tractable modules: *instance-aware weights* and *instance-agnostic features* generation (as illustrated in the bottom of Fig. 1). Specifically, the former module *dynamically* generates n convolution weights, predicting n possible mask features. These dynamic weights are named as *instance-aware weights* as they correlate with the predicted instances. Also, the latter module learns a high-dimensional instance-agnostic feature, serving as a template for potential mask representations from a PoI feature. By convolving the instance-agnostic feature with the instance-aware weights as convolution parameters, we will get the instance-aware feature, aligning the template feature to the specific instance. In this way, it is easy to focus on predicting masks for those positive instance candidates of PoIs while maintaining a large mask feature size. Thus, our proposed instance-aware convolution can solve the computational burden existing in the standard convolution for PoI-based methods.

Along with the instance-aware convolution, we propose PointINS, a simple and practical approach to evolve one-stage detectors for instance segmentation. It provides compelling PoI features for final mask prediction both efficiently and effectively.

The contributions of this study are three-fold:

- We propose PointINS, a new pointed-based framework for instance segmentation equipped with the given instance-aware convolution. This module addresses the learning problem about computing proper mask representations for several instances with PoI features, concerning both effectiveness and efficiency.
- Our proposed method gives the state-of-the-art instance segmentation performance among existing PoI-based methods. This performance is also competitive compared with the region-based Mask R-CNN. Under the single-scale 1x training schedule, we obtained **34.5** mask mAP with ResNet101-FPN. Using data augmentation or with longer training time improved performance consistently to **38.3**, which is comparable to Mask R-CNN (38.3) yet with faster inference speed (14.9 fps vs. 13.5 fps (frame per second)).
- Our given PointINS is compatible with the most dense one-stage detectors (RetinaNet [37] and FCOS [60]), as its core module instance-aware convolution is decoupled from box heads. It serves as a general framework for point-based instance segmentation.

2 Related Work

In this section, we first revisit representative studies on object detection, as its close relationship to instance segmentation. Then, we describe methods related to instance segmentation and the core technique of dynamic weights employed in this study. Wrapped with dynamic weights, the proposed instance-aware convolution could evolve the dense one-stage detectors for instance segmentation with minimal modifications.

2.1 Object Detection

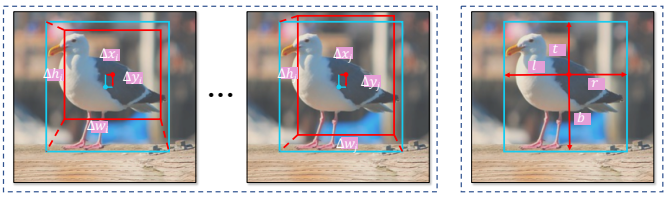
Current convolutional neural network (CNN)-based object detectors can be categorized into two-stage detectors or one-stage detectors based on whether they exploit region proposals. Two-stage detectors, such as a series of R-CNN [17], [18], [54] first select RoIs and then extract the region features to localize and classify objects. The following algorithms are proposed to improve the performance of two-stage detectors, including new architecture designs [6], [14], [7], [30], [34], context and attention mechanism [3], [55], [42], [4], [50], multi-scale exploration [58], [46], training strategy, and loss design [45], [56], [62], [21], feature fusion and enhancement [28], [27], [40], better proposal, and balance [59], [47]. We note that all these methods use the region feature extraction to detect objects. Due to the accurate alignment between region locations and features, RoI Align [19] has become the primary region feature extraction method in two-stage detectors.

Instead of using cropped region features, one-stage methods [41], [52], [37], [60] detect boxes only from point features. Owing to the focal loss [37], one-stage methods can effectively balance the training loss between positive and negative samples, thereby leading to satisfactory detection performance. With faster inference speed, one-stage detectors have become increasingly popular in the computer vision area. Among numerous one-stage detectors, some detectors [41], [52], [37], [60] generate bounding boxes from the single center point, as shown in Fig. 2. For simplicity, we refer to these center-point based one-stage detectors as *dense one-stage detectors*.

Like the box generation of dense one-stage detectors, our proposed PointINS framework generates instance masks from a single-point feature. Because both boxes and masks are different representations of instances, our method can be built upon these detectors regardless of whether they are anchor-based or anchor-free. This means that dense one-stage detectors can be quickly converted into instance segmentation frameworks by our method with minor modification.

2.2 Instance Segmentation

Current instance segmentation methods can be roughly classified into two categories: segmentation-based [39], [2], [61] and detection-based [19], [40], [8], [24], [31], [48] methods. Segmentation-based methods usually first predict a semantic map and then cluster points with similar semantic



(a) RetinaNet (Multiple Anchors) (b) FCOS (Anchor Free)

Fig. 2. Illustration of differences between RetinaNet [37] and FCOS [60]. Blue points and boxes represent the center and bound of an object, respectively, while red points and boxes represent the center and bound of anchor, respectively. (a) The single point in RetinaNet responds to several predefined anchors with various sizes and aspect ratios. Thus, RetinaNet regresses from the anchor box with four offsets. (b) FCOS has no anchors, and regresses from a center point with four distances. For simplicity, we refer to this center point as an anchor point.

feature embedding into instance masks. Despite the presented new problem formulation of instance segmentation, their expensive clustering produces prevent their real-world applications. Other types of methods encode the instance information into each channel of the generated semantic map. InstanceFCN [12] and FCIS [35] propose predicting position-sensitive score maps with vanilla fully convolutional networks (FCNs). SOLO [61] directly decodes each channel of the semantic map into instance masks without indicating whether the used regions are valid. The efficiency of these methods is a concern, as they segment numerous background areas containing no instances but usually covering a large portion of the images.

Popular instance segmentation frameworks are mainly detection-based [17], [18], [54], [14], [36], [48]. They generally predict a series of bounding boxes by two-stage detectors and then segment masks. Mask R-CNN is an representation for two-stage instance segmentation framework. Several methods, including PANet [40], HTC [8], and Mask Scoring R-CNN [24], refine structure details from different aspects, and all methods use the RoI-Align [19] to produce region features. Even RetinaMask [16], Centermask [31] and Embedmask [69], which are built upon one-stage detectors, such as RetinaNet [37] or FCOS [60], use RoI features to segment instances.

Instead of using RoI features, Tensormask [10], Polarmask [64], and MEInst [70] force a single-point feature to segment only one or two potential regions for exploring a more straightforward representation of instance masks. YOLACT [4] assembles the point responses to several prototypes for final mask prediction. However, the prototypes are predefined in numbers and involve additional hyper-parameters. In contrast to these methods, our framework generates a point-wise instance-agnostic feature, serving as a template for potential instance masks of generated bounding boxes.

Single-shot instance segmentation Single-shot instance segmentation has two popular definitions in the literature. The first depends on whether the proposed method directly generates masks without a box indicator, while the second depends on whether the proposed methods have fewer information processing steps than the popular Mask R-CNN.

SOLO [61] and Polarmask [64] belong to the former category, while Tensormask [10] and MEInst [70] belong to the latter category. Without boxes, SOLO and Polarmask have to segment many invalid regions. Although Tensormask [10] and MEInst [70] simultaneously generate boxes and masks, they still require box information to interpolate masks from fixed sizes to the box size, where the setting is the same as ours. The main difference between these methods and ours is our adopted sampling strategy, as described in subsection 3.3. We sample valid regions to segment masks in advance, following the process of Mask R-CNN: detection first, followed by instance segmentation. Our premise is that focusing on valid regions can reduce the computational cost of mask generation.

2.3 Dynamic Weight

As a specific strategy of meta-learning [1], [51], [63], weight prediction [32], [67] aims to generate dynamic parameters according to different inputs. In other words, the network parameters are input-dependent. It can significantly improve the flexibility of the used net structure at the cost of additional computation.

Cai et al. [5] used the predicted parameters of a classifier to discriminate new categories. These parameters were generated by the memory of the support set. In object detection, MetaAnchor [68] is a flexible anchor generator to produce higher quality boxes. The weight of the last regression layer is predicted with properties of customized prior anchors. To balance the instance annotation cost between box and mask, Hu et al. [22] proposed a semi-supervised method to segment zero-shot instances by re-weighting box features. Besides, MetaSR [23] uses a dynamic up-sampling module to super-resolve a single image with arbitrary scale factors.

In our proposed method, we use dynamic convolution weights to learn different instance representations from a single point feature, achieving alignment between an instance-agnostic feature and positive instances. With instance properties, we generate several unique instance-aware features for mask prediction while maintaining the high mask representation capacity. In this way, our method is effective and robust to one-stage dense detectors even with numerous anchors tiled at a single point.

3 Our Method

Instance segmentation aims to detect all instances from images at the pixel level, predicting the location of an object and the category that it belongs to. In this study, the proposed framework follows the design philosophy of Mask R-CNN, which splits the process of determining the class and location of objects into two independent sub-tasks: instance location regression and class prediction. In Mask R-CNN, both sub-tasks are predicted with RoI features extracted by RoI Align. In contrast, this paper studies how to exploit PoI features to learn mask representation for instances. Meanwhile, the classification component follows previous point-based detectors, such as RetinaNet [37] or FCOS [60].

Generally, our proposed PointINS framework has two components in its structure: a one-stage detector and instance-aware mask prediction. We use the dense one-stage

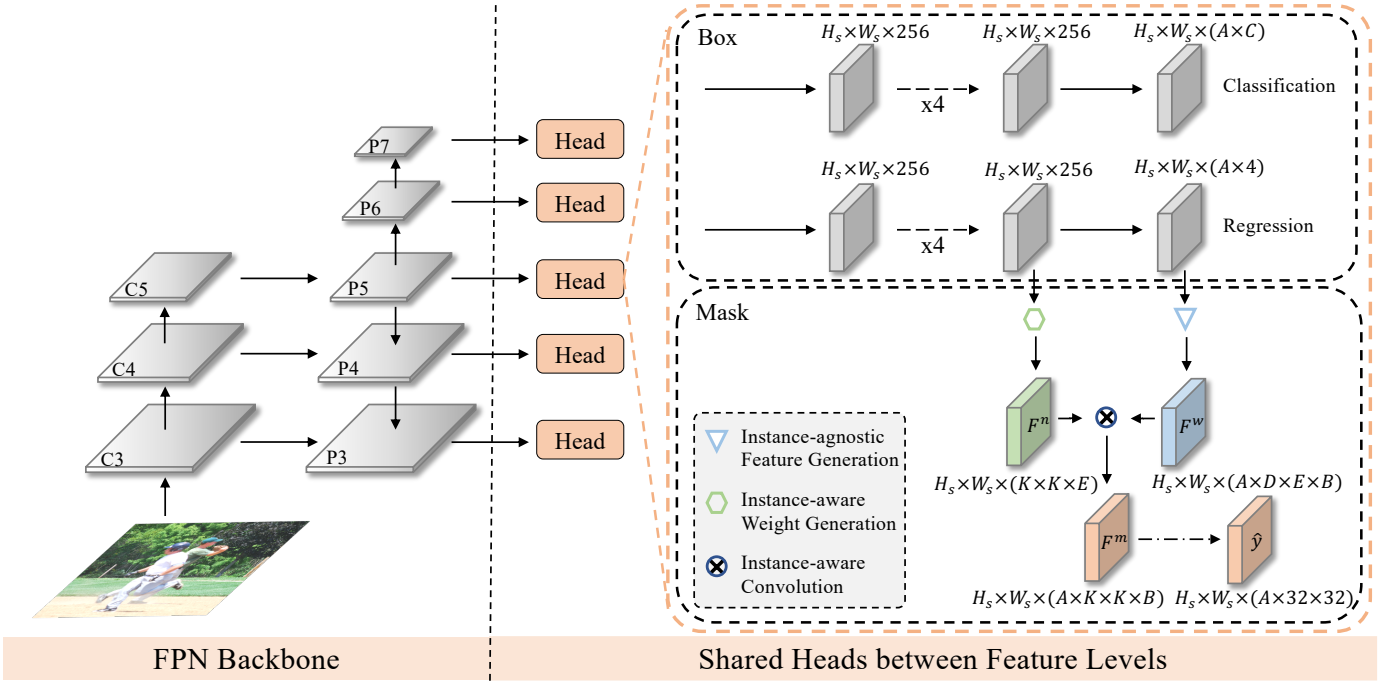


Fig. 3. Overall architecture of proposed PointINS framework. C_3, C_4 and C_5 are the feature maps of the backbone network (*e.g.*, ResNet50). P_3 to P_7 are the feature pyramid network (FPN) feature maps, as in [36], [37], [60]. For better illustration, each feature map is represented by three dimensions, including height H, width W and channel without considering the image batch. A is the number of anchors covered by a single point, with 9 and 1 in RetinaNet and FCOS respectively. C is the number of classes, such as 80 in COCO [38] dataset.

detector here, as it naturally offers pixel-level semantics via classification and our desired PoI features for mask estimation. More importantly, it generally runs faster than a two-stage detector. The employed dense one-stage detector has a classical structure, including a backbone (*e.g.* ResNet [20]), a feature pyramid structure (*e.g.* FPN [36]), and a box generation head. Our proposed instance-aware mask prediction module provides simple and effective mask estimation and is compatible with most dense one-stage detectors whatever they are anchor-based or anchor-free. It is plugged into the box generation head of the used detectors, transforming PoI features into instance masks with minor modifications.

In the remainder of this section, we first briefly revisit the applied one-stage detectors [37], [60]. Then, we describe our proposed instance-aware mask prediction module and demonstrate its flexibility for both RetinaNet [37] and FCOS [60] with two variants. Thereafter, we analyze the advantage of our proposed module in terms of computation cost, and introduce the final loss function for joint training with the utilized one-stage detectors. Finally, we discuss the relationship between our modules and some existing works, e.g., Tensormask [10].

3.1 Dense One-stage Detectors

The employed detector performs the pixel-level class prediction task and provides PoI features for mask prediction. Structurally, as illustrated in Fig 3 (not including the mask component), it contains an FPN backbone and a detection head, utilizing anchors (predefined sliding windows) for distinguishing possible multiple instances of varied forms.

The FPN backbone adopts a feature pyramid scheme to detect objects with different sizes. Specifically, the FPN backbone extracts feature maps $F_s \in \mathcal{R}^{H_s \times W_s \times 256}$ of different resolutions from the input image $I \in \mathcal{R}^{H \times W \times 3}$. Here, H, W, H_s, W_s denote the height and width of an image and generated pyramid feature maps, respectively, where $s \in \{8, 16, 32, 64, 128\}$ is the stride of the pyramid feature maps compared to the input image size.

The detection head has two branches: the classification branch and regression branch. Each branch has four convolutional blocks, including the convolutional layer and rectified linear unit (ReLU) operation. These blocks are shared among all feature-level maps. The classification branch provides the classification prediction for every pixel of the feature map with a multi-class binary vector [37] form as $P_{s,i,a} \in \{0, 1\}^C$ where $P_s \in \mathcal{R}^{H_s \times W_s \times A \times C}$. A and C denote the anchor and object category number, respectively; and i and a denote the pixel location and anchor index, respectively. Also, the regression branch predicts the offset $R_s \in \mathcal{R}^{H_s \times W_s \times A \times 4}$ between the anchors and ground truth. We note that this offset regression is different when using different dense one-stage detectors. As illustrated in Fig. 2, RetinaNet [37] tiles multiple preset anchors for each point of the feature map. Thus, a single-point feature is used to regress the distances between anchors and nearest objects. FCOS [60] does not use anchors and uses a single-point feature to learn the distances between the coordinate of points and the nearest object. A point in FCOS is similar to an anchor without restrictions on the area size and aspect ratios. From this perspective, we can regard FCOS as an anchor-based detector with only one anchor. Unlike RetinaNet, FCOS has an additional centerness branch in box generation head to predict whether the anchor point is in the center of the ground truth bounding boxes.

In our method, we regard both the bounding boxes and pixel-level masks as the representations of instance locations. Thus, our proposed module utilizes the last feature map F_s^l of the regression branch for further instance segmentation. Given that RetinaNet and FCOS have different regression styles, we explore both of them to generate instance-aware weight in Section 3.2.2, and show how these regression styles affect performance empirically in the ablation study in Table 5.

3.2 Instance-aware mask prediction

TABLE 1
The illustration of key hyper-parameters (HP) using RetinaNet as backbone. A and U are 1 and 7, respectively when using FCOS as backbone. DCW is short for dynamic convolution weight.

HP	Meaning	Default
A	the number of anchors	9
K	the height of square mask feature	16
E	the input channel of DCW	9
B	the output channel of DCW	256
B	the kernel size of DCW	1
U	the input size to generate dynamic weight	10

For our proposed instance-aware mask prediction, each PoI feature $F_{s,i}^l \in \mathcal{R}^{1\times1\times256}$ is extracted from the regression branch of the one-stage detector. It produces a $K \times K$ class-agnostic binary mask for every anchor as $M_{s,i,a} \in \mathcal{R}^{K\times K}$, where $M_s \in \mathcal{R}^{H_s\times W_s\times A\times K\times K}$ is the regression result from the mask feature $F_s^m \in \mathcal{R}^{H_s\times W_s\times A\times K\times K\times B}$. $K \times K$ and B denote the mask resolution and mask feature dimension, respectively.

A main advantage of the proposed instance-aware mask prediction is that it supports a high-dimensional mask feature with sufficient characteristics from PoI features, thereby exhibiting the comparable instance segmentation performance for those methods using RoI features. For example, the mask feature size K^2B in Mask R-CNN is $50176 = 256 \times 14^2$. Our support for high-dimensional mask representation in PoI form results from our decoupled design of the mask feature computation. For each point, we decouple this computation into instance-agnostic features and instance-aware weights, in which we project the shared template to the unique mask feature. Specifically, with the feature maps from the regression branch of the dense detectors, we disentangle the mask feature computation into two parts: instance-agnostic features $F_s^n \in \mathcal{R}^{H_s\times W_s\times K^2E}$ and instance-aware weights $F_s^w \in \mathcal{R}^{H_s\times W_s\times A\times(D\times E)\times B}$. D is the weight kernel size, which is described in Section 3.2.3.

As the name suggests, every instance-agnostic feature point $F_{s,i}^n \in \mathcal{R}^{K\times K\times E}$ indicates a primitive template feature containing robust information on its receptive field. Thus, this feature can be responsible for all tiled anchors. Besides, every instance-aware weight point $F_{s,i}^w \in \mathcal{R}^{A\times B\times D}$ represents A mask feature embeddings whose embedding dimension is $B \times D$, derived from the anchors' property (such as anchor width/height or aspect ratio) and the regression offset between anchors and target boxes. Using dynamic convolution as the feature alignment, we transform the shared template feature to the unique feature of the potential instance.

$$F_{s,i}^m = \bigotimes(F_{s,i}^n, F_{s,i}^w), \tag{1}$$

Where $\bigotimes$ is the dynamic convolution, in which $F_{s,i}^n$ is input and $F_{s,i}^w$ is the dynamic convolution weight.

In structure, the instance-agnostic features and instance-aware weights are extracted from two different branches; however, both stem from the regression branch of the one-stage detector. They are illustrated in Fig. 4, and their specific designs are discussed below.

3.2.1 Instance-agnostic Feature Generation

This module aims to provide robust and informative template features for the numerous anchors. In our framework, a single-point feature $F_{s,i}^n$ is capable of producing several masks. Thus, the single-point feature should have a sufficiently large receptive field to cover all possible masks, and contain sufficient semantic information for subsequent mask prediction. This conjecture was verified in our ablation study, as illustrated in Table 2.

Let $F_{s,i}^n$ denote the transformed features from the last feature map in the regression branch of one-stage detectors. It is computed by an additional *channel up-scaling convolution* and a reshape operation to further aggregate location information. As such, our point feature is robust enough to capture the entire instance, even if this point is in the instance boundary.

Specifically, the aforementioned channel up-scaling operation is a convolution layer with input and output channel numbers as K^2 and K^2E, respectively. The convolution kernel size is $\sqrt{E} \times \sqrt{E}$. Since $K^2E = \sqrt{E} \times \sqrt{E} \times K^2$, the output channel increases E times with its $\sqrt{E} \times \sqrt{E}$ convolution kernel size. This operation is essential in practice, as the increased channels allow our tensor to encode more context information for mask prediction. We note that this operation can become relatively computationally cheap, as discussed in Section 3.3. On the other hand, the reshape operation is used to balance the template spatial resolution (spatial dimension) and feature representation (channel dimension). We transform a single-point feature into a three-dimensional tensor with size $(K\times K\times E)$. In this way, we can force each channel of our point feature to have explicit relative-location information[1]. For example, the first nine dimensions of features before the reshape operation respond to the information of the upper-left-most location in the template feature.

Note that the K and the input feature dimension C_{in} are correlated in this paper. The success of dense one-stage detectors is built on the premise that each point feature encodes the desired spatial information into the channel dimension. With this premise, K^2 should be equal/close to the feature dimension considering performance. Because $K^2 \gg C_{\text{in}}$ makes the prediction mask resolution go beyond the capacity of input feature, while $K^2 \ll C_{\text{in}}$ leads to coarse mask prediction. Regarding this, since the channel number of input features we used is 256, we suppose the maximum effective resolution of its leading mask resolution

1. It is similar to R-FCN [14] and FCIS [35], whose channels explicitly encode relative location scores of instances.

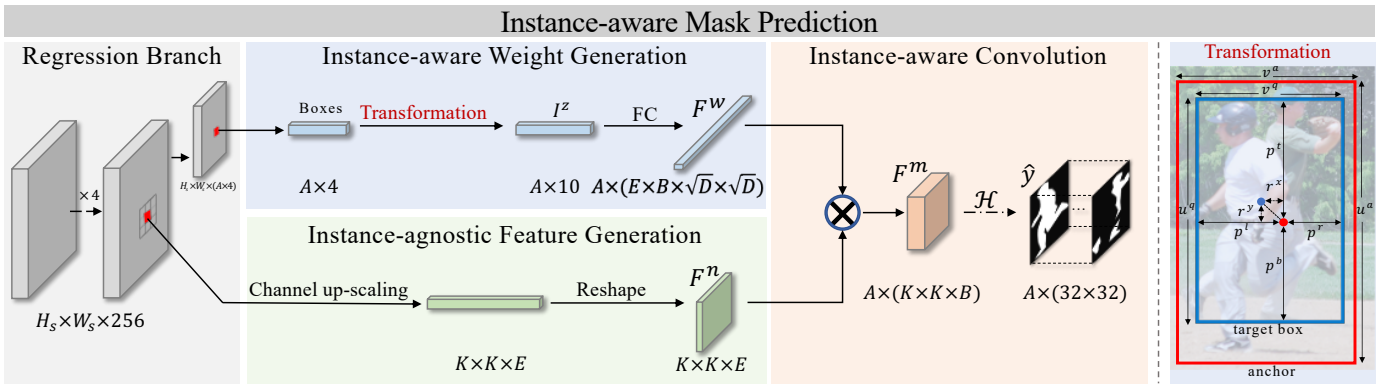


Fig. 4. Detailed structure of proposed instance-aware mask prediction module. For simplicity, we only sample a single-point feature generated from the 3×3 points feature in regression branch for illustration. The red point from regression branch is our sampled positive point. The key process in proposed module is instance-aware convolution, including the *instance-agnostic* feature and *instance-aware* weight. First, we generate the instance-agnostic feature F^n by a channel up-scaling convolution layer and a reshape operation. Meanwhile, the instance-aware weight F^w is obtained by transforming the explicit instance information I^z. Then the two features serve as the input and weight of the *instance-aware* convolution, in order to produce instance-aware feature F^m for final mask prediction. The right-most part is a geometric illustration of our transformation module.

is 16x16, and use $K^2 = C_{\text{in}}$ as the default setting unless other specified. For E being 9 by default, we share the same reason as for K that we want to prevent information loss in this channel up-scaling operator, so we ensure the input feature dimension equals to output feature dimension by relating the E with the convolution kernel size.

3.2.2 Instance-aware Weight Generation

This module targets to generate n dynamic convolution weights from a PoI feature, used for differentiating n possible instances. Its dynamics prevent from employing n different conventional convolutions. From this perspective, it notably reduces the used parameters and their corresponding footprints in the computation when n is relatively large, e.g., $n = 9$.

For its design, let $I^z_{s,i,a}$ denote the PoI feature with instance information, we predict its corresponding convolution weight $F^w_{s,i,a}$ by an FC layer, followed by a ReLU operation as:

$$F^w_{s,i,a} = ReLU(FC(I^z_{s,i,a})), \tag{2}$$

where the FC layer with input channel as U (10 for RetinaNet and 7 for FCOS) and output channel as $B \times D$. Usually I^z is computed from the PoI feature indicated by a regressed box. Since we can access both anchor and GT boxes in training, we can also compute I^z from these two types of boxes. This may be beneficial to the learning of instance-aware weights F^w in Eq. 5 by avoiding the error accumulation from regressed box locations. We study its training sampling strategies in Section 4.1.1.

Specifically, $I^z_{s,i,a} = \{I^o_{s,i,a}, I^q_{s,i,a}\}$ can be divided into an anchor indicator $I^o_{s,i,a}$ and an instance indicator $I^q_{s,i,a}$. $I^o_{s,i,a}$ contains information about which specific anchor at a specific feature level corresponds to the potential mask. $I^q_{s,i,a}$ has explicit information to characterize the target box, which could be regressed from an anchor box. These two indicators are detailed below.

For $I^o_{s,i,a}$, it has four components as follows:

$$I^o_{s,i,a} = \{\frac{1}{s}, \frac{u_{s,i,a}}{v_{s,i,a}}, \frac{u_{s,i,a}}{s}, \frac{v_{s,i,a}}{s}\}, \tag{3}$$

where s is the stride of the global feature level, and $u_{s,i,a}$ and $v_{s,i,a}$ are the height and width of the anchors, respectively. Therefore, the last three components represent the aspect ratio and scale of the anchor.

Moreover, $I^q_{s,i,a}$ represents the instance offset from the original anchors as

$$I^q_{s,i,a} = \{r^x_{s,i,a}, r^y_{s,i,a}, p^l_{s,i,a}, p^r_{s,i,a}, p^b_{s,i,a}, p^t_{s,i,a}\}, \tag{4}$$

where $r^x_{s,i,a}$ and $r^y_{s,i,a}$ represent the distances between the centers of the anchor and target box, respectively, and the other elements represent the distances between the box location and the center of the corresponding anchor in four directions. Each component in $I^q_{s,i,a}$ is calculated as

$$\begin{aligned} r^x_{s,i,a} &= \frac{x^q_{s,i,a} - x^o_{s,i,a}}{s}, \quad r^y_{s,i,a} = \frac{y^q_{s,i,a} - y^o_{s,i,a}}{s} \\ p^l_{s,i,a} &= \frac{0.5 * v^q_j}{s} - r^x_{s,i,a}, \quad p^r_{s,i,a} = \frac{0.5 * v^q_j}{s} + r^x_{s,i,a} \\ p^t_{s,i,a} &= \frac{0.5 * u^q_j}{s} - r^y_{s,i,a}, \quad p^b_{s,i,a} = \frac{0.5 * u^q_j}{s} + r^y_{s,i,a} \end{aligned} \tag{5}$$

where $x^q_{s,i,a}$ and $x^o_{s,i,a}$ are the x-axis center location of target box and anchors, respectively, and are the same as $y^q_{s,i,a}$ and $y^o_{s,i,a}$. $u^q_{s,i,a}$ and $v^q_{s,i,a}$ are the height and width, respectively, of the target box. The rightmost part of Fig. 4 presents the geometric illustration of our transformation module.

Note that $I^z_{s,i,a}$ only contains $I^q_{s,i,a}$ and $\frac{1}{s}$ for the anchor-free detectors. Using FCOS [60] as an example, we regard the points of each feature level as our anchors, in which x^o_{sia} and y^o_{sia} are the sampled-point locations of the images.

Effectiveness of instance-aware Weights Fig. 5 presents a visualization of an instance-aware feature by using different instance-aware weights for the same instance-agnostic feature. The dynamic weights can produce various instance-correlated features from the shared instance-agnostic feature. Moreover, it seems that some part of the person that is being occluded still shows up in the instance-aware mask feature in the left third sub-figure. We attribute it to the employed instance-aware features used for visualization contain negative samples. These potential negative samples

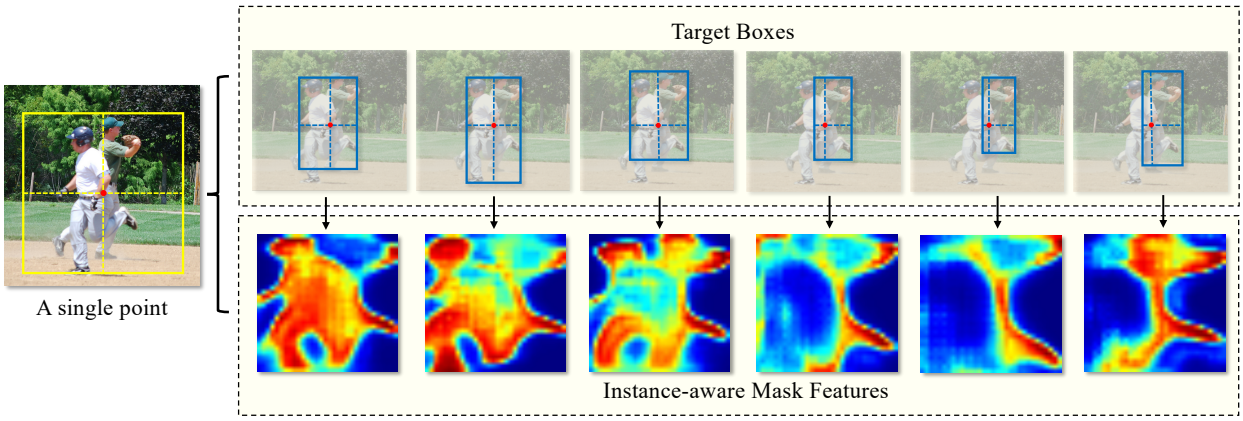


Fig. 5. Visualization of instance-aware features under the same point feature. With various instance information, instance-aware convolution generates distinct instance-aware features for subsequent mask prediction.

can segment the most salient part of instances like the head, arm, or leg. These samples do not consider the criterion that IoU between the regressed box and anchor box should be larger than 0.5, which are not used for both training and inference. For those negative samples, it may be reasonable to segment the most salient part of instances like the head, arm, or leg. This case could be solved if we degrade the sample IoU threshold and thus more highly overlapped samples could be trained.

3.2.3 *Instance-aware convolution*

The instance-agnostic feature and instance-aware weights are fused by pixel-wise convolution, leading to the mask features. We reshape the instance-aware weight $F^w_{s,i,a}$ with the $E \times B \times D$ dimension to the $E \times B \times \sqrt{D} \times \sqrt{D}$ dimensions. The reshaped weights can be regarded as convolution weight with input channel E, output channel B, and kernel size $\sqrt{D}$. For ath anchor of the point in location i, we obtain our instance-aware mask feature $F^m_{s,i,a}$ by the dynamic convolution as:

$$F^m_{s,i,a} = \bigotimes(F^n_{s,i}, F^w_{s,i,a}). \tag{6}$$

where the input of this convolution is the instance agnostic template $F^n_{s,i}$, and it is parametrized by the instance-aware weights $F^w_{s,i,a}$.

In this way, our instance-agnostic feature can be aligned to the corresponding mask feature with $K \times K \times B$ dimension. Then via the standard mask head of Mask R-CNN including four convolution blocks and a deconvolution block, it is prepared for final class-agnostic mask prediction. Defining the mask head operation as $\mathcal{H}$, we get the final mask prediction as:

$$\hat{y}_{s,i,a} = \mathcal{H}(F^m_{s,i,a}) \tag{7}$$

3.3 Computational Efficiency Optimization

In our proposed method, instance masks are only generated from a PoI feature. This generation method allows our approach to be easily improved for high efficiency. We can realize this by only sampling positive points and generating their corresponding instance-agnostic feature and instance-aware weight in parallel.

At first, a point is defined as positive if the Intersection of Union (IoU) between the ground truth and one of the anchors and target boxes is larger than 0.5. In the COCO dataset, there are 6.43 positive points for each ground truth instance following this sample principle with only considering the anchor condition. We randomly sample the maximum 128 points in training out of more than $22,000+$ points on all feature levels. For inference, only 100 detected points are used for final mask prediction. Note that the same point is likely to be sampled multiple times due to several of its covered anchors and target boxes satisfying our sample principle.

Thus, the subsequent processing is applied only on these points, which enables remarkable acceleration. Due to the channel up-scaling convolution with kernel size 3 in the instance-agnostic feature generation module, it is only necessary to sample 3×3 receptive fields from the feature map to produce our instance-agnostic features. For the instance-aware weights, it is only necessary to sample the corresponding four-dimensional regression feature for each point. Also, because these points are independent, they can be processed in parallel for further notable acceleration and memory usage reduction. Our inference achieves 15.8 FPS with ResNet50 backbone shown in Table 11.

3.4 Loss function

The overall optimization goal considers both instance mask segmentation and the employed detection as

$$\mathcal{L} = \lambda_{det} \cdot \mathcal{L}_{det} + \lambda_{mask} \cdot \mathcal{L}_{mask}, \tag{8}$$

where $\mathcal{L}_{det}$ and $\mathcal{L}_{mask}$ denote the detection loss and mask loss, respectively. The balance weights in λ_{det} and λ_{mask} are set to 1 and 2.

For $\mathcal{L}_{det}$, it relies on the one-stage detector used. For example, RetinaNet consists of $\mathcal{L}_{cls}$ for classification and $\mathcal{L}_{reg}$ for bounding box regression. FCOS also has an extra loss $\mathcal{L}_{cen}$ for center-ness. We adopt the default setting for these detection losses. $\mathcal{L}_{cls}$ and $\mathcal{L}_{cen}$ are the focal loss and binary cross-entropy loss, respectively. $\mathcal{L}_{reg}$ is the smooth-l1 loss in RetinaNet or the generalized IoU loss in FCOS.

For $\mathcal{L}_{mask}$, it is defined as

$$\mathcal{L}_{mask} = \sum_{s}^{S} \sum_{i}^{H \times W} \sum_{a}^{A} \mathbb{1}^{obj} d_{mask}(\hat{y}_{s,i,a}, y_{s,i,a}) \quad (9)$$

where $\mathbb{1}$ is the indicator function for positive samples. $\hat{y}$ and y denote the prediction and ground-truth vectors, respectively. The d_{mask} is the weighted binary cross-entropy loss. The edge of the mask is emphasized by assigning the larger weight as 4.

As our proposed method predict instance segmentation at each level of feature pyramid, we apply both detection loss and mask loss (as given in Eq. 8 and Eq. 9) to different levels of feature pyramid with equal weight for each level. This design shares the same idea with the loss for the mask head of Mask R-CNN, even though the sample Region of Interest (RoI) features are from different feature levels. Finally, the PointINS is jointly trained with the detectors it is built upon.

3.5 Relation to Existing Methods

As the PoI-based methods, e.g., Tensormask [10], we adopt PoI features to predict instance masks. However, we provide a new mask representation computation design with dynamic weights, and our mask representation is more effective and efficient than the existing works [10], [64], [70] due to its high capacity, as validated in the performance comparison presented in Table 12.

We obtain our instance-aware features by instance-aware convolution. The core of this step is the instance-aware weights derived from the anchor property and box regression feature. These weights dynamically interpolate the instance-agnostic feature to the instance-ware mask feature. Unlike bilinear interpolation used in Tensormask, dynamic convolution can flexibly align the instance-agnostic feature to a specific region by learnable weights. Thus, our module can be easily generalized to dense detectors with multiple anchors. With the sampling strategy, our module only focuses on PoIs while maintaining a high mask representation.

In a comparison regarding mask representation, our mask feature size is $65536 = 256 \times 16^2$ from PoI features with $K = 16$ and $B = 256$, outperforming the mask feature size (only 256) of Polarmask [64] and MEInst [70]. The low-dimensional mask feature of PolarMask and MEInst results in reduced performance regardless of how the final mask representation is modeled. The outlier is Tensormask [10] with a dynamic mask feature size from 15×15 to 480×480 by bilinear interpolation. This usage of bilinear interpolation depends on the aligned representation and tensor pyramid structure they proposed; however, this leads to a computationally intensive feature process.

Moreover, due to the proposed instance-aware convolution, our framework is robust to the anchor number each PoI feature corresponds to, making it applicable to the most types of dense one-stage detectors (both anchor-based and anchor-free), while existing methods only work in one specific type.

One-stage instance segmentation We categorize our method into one-stage framework by following the ritual in Tensormask [10] and MEInst [70], which are both considered as one-stage approaches. The definition of one-stage instance segmentation is controversial. There are two popular definitions. The first one is whether the proposed method directly generates masks without any box indicator, e.g. SOLO [4] and Polarmask [64]. The second is whether the proposed method has fewer intermediate process steps compared with the popular Mask R-CNN [19], like Tensormask and MEInst.

4 Experiments

4.1 Experiments on COCO Dataset [38]

We compare our method with other state-of-the-art approaches on the challenging COCO dataset [38]. Following common practice [19], [36], [40], [24], we train our models with 115,000 train images and reported results on the 5000 validation images for the ablation study. We also report results on 20,000 test-dev images for comparison.

Comprehensive ablation studies are conducted on this dataset with RetinaNet-ResNet50-based PointINS. We follow standard evaluation metrics, namely, average precision for IoU from 0.5 to 0.95 with a step size of 0.05 (AP), AP_{50}, AP_{75}, AP_S, AP_M and AP_L. The last three measure performance with respect to objects in different scales.

Training Details We train our network using batch size 16 for 12 epochs. The shorter and longer edge sizes of the images are 800 and 1333. Adam gradient descent with learning rate as 0.00005 , beta as $(0.9, 0.999)$, and eps as $1e^{-08}$ is used as the optimizer. We decay the learning rate with 0.1 at 8 and 11 epochs respectively. Using stochastic gradient descent leads to a performance reduction of approximately 0.3. We initialize our backbone networks with the pre-trained models on ImageNet. Moreover, we sum all the losses directly, *i.e.*, $\lambda_{det} = 1, \lambda_{mask} = 2$ in Eq. 8.

Inference Details The inference is kept the same as dense detectors, as we only append one additional mask prediction to the predicted boxes. Two strategies are used to obtain the final result on the condition of whether or not the samples have valid boxes. The former generates all the masks and then samples a maximum of 100 instances by box non-maximum suppression (NMS) following Tensormask [10]. The latter uses box NMS to sample a maximum of 100 boxes and then predict their masks, like MEInst [70]. Compared to the former strategy, the latter does not lead to performance reduction but avoids the unnecessary computational overhead for masks.

4.1.1 Ablation Study

All ablation studies are conducted by RetinaNet-based PointINS with ResNet50. Note that this baseline yields 32.5 AP for the validation split.

Instance-agnostic Feature Generation The representation of the template tensor is first studied. As illustrated in Table 2, we explore the influence of different shapes on the performance of the instance-agnostic feature. For a fair comparison, we fix the kernel size E of the channel-up scaling convolution layer to 3. That means, we only change the output channel of this convolution layer to match the size of (E, K, K) of Table 2. The default tensor shape is $(9, 16, 16)$, where 9 is the feature representation and 16 is the spatial resolution. The performance decreased when we reduced the number of feature representations. This

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3085295, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 2
Ablation study for instance-agnostic feature representation with a fixed kernel size of 3 of channel up-scaling convolution. The reshaped tensor' size is (E, K, K), where E denotes the feature representation and K denotes the spatial resolution. The Modification column lists the methods of changing the tensor size and shape by modifying the channel up-scaling output channel or performing the reshape operation.

(E, K, K)	modification	AP	AP_{50}	AP_{75}
$(9, 16, 16)$	default	**32.5**	**51.9**	**33.8**
$(1, 16, 16)$	Decrease Channels	30.4	49.2	31.6
$(4, 16, 16)$		31.6	50.1	32.4
$(36, 8, 8)$	Reshape Styles	31.8	51.0	32.4
$(1, 48, 48)$		29.6	48.5	30.1
$(16, 16, 16)$	Increase Channels	32.3	51.7	33.7
$(25, 16, 16)$		**32.5**	51.8	33.6

indicates that the feature representation dimension is vital for encoding information.

Intriguingly, as illustrated in the last two rows, further increasing the number of feature representation does not improve the results. This is resulted from our used template tensor. Regardless of how we transform the information, the template is reshaped from a single-point feature. In this ablation study, this single-point feature is generated by a 3×3 convolution with an input channel number of 256 by default. Therefore, the information capacity is $3 \times 3 \times 256$. The larger output channels of convolution do not further increase the capacity of valid messages.

This analysis also explains the performance reduction using different reshape styles. A 3×3 kernel encodes the 9 spatial locations in sequence. Therefore, reshaping to a $(9, 16, 16)$ tensor is more intuitive than the manner in which each channel represents the relative location response. We note that this is also the core principle of R-FCN [14] and FCIS [35]. We regard a tensor with size $(1, 16, 16)$ as the nature representation [10] proposed by the TensorMask study. It causes a reduction of approximately 3 mean average precision (mAP).

Table 3 outlines an ablation study using different kernel size E of channel up-scaling convolution with an output channel number of $256 \times E^2$. We note that this ablation study is different from that in Table 2, with various kernel size *e.g. 1, 3, 5 or 7*. Therefore, there is no capacity change via this convolution because its input channel number is always 256. We find that using a large kernel size would not lead to a noticeable performance increase. We conjecture that a 3×3 kernel size is sufficient to help the convolution layer capture the target instance regions, because its input has a large enough receptive field, which has been already used for bounding box prediction.

Instance-aware Weight Generation Table 4 summarizes an ablation study on the input used for generating instance-aware weight. We first explore the two extreme cases, presented in the first and last rows of Table 4. Our framework achieves the best performance when involving all three parts. In the first row, we directly use the features from the detection branch instead of our explicit box information. The performance reduction indicates that the explicit information of bounding boxes is more helpful in generating instance-aware weights than directly using the features. We suppose that the features directly generated by the network are insensitive to the stride of feature maps.

As alternatives to using this piece of information, only using the I^o or $\{r^x, r^y\}$ leads to lower AP than directly using the features of the detection head. This means that the offset between anchors and boxes is vitally important in our module. Our design makes it possible to determine the exact shift to guide fine-tuning the instance-agnostic template. This assumption is verified in Fig. 5. In our module, $\{p^l, p^r, p^b, p^t\}$ play vital roles, and I^o (and $\{r^x, r^y\}$) is peripheral.

Table 5 presents the performance with other heuristic designs for I^z. The results indicate that the unique region

TABLE 3
Ablation study for instance-agnostic feature representation with various kernel size of channel up-scaling convolution. The reshaped tensor' size is (E, K, K), where E denotes the feature representation and K denotes the spatial resolution.

(E, K, K)	AP	AP_{50}	AP_{75}
$(9, 16, 16)$	32.5	51.9	33.8
$(1, 16, 16)$	30.4	46.3	28.7
$(25, 16, 16)$	**32.7**	**52.0**	**33.9**
$(49, 16, 16)$	**32.7**	51.9	33.9

TABLE 4
Ablation study for the input to generate instance-aware weights. The inputs was split into three parts and explored. Each variable formulation is simplified by deleting its subscript. For example, I^o denotes I^o_{sia}. $\circ$ represents "not used" for this part, whereas ✓ represents used.

I^o	$\{r^x, r^y\}$	$\{p^l, p^r, p^b, p^t\}$	AP	AP_{50}	AP_{75}
$\circ$	$\circ$	$\circ$	27.9	46.5	28.5
✓	$\circ$	$\circ$	25.5	44.9	26.1
$\circ$	✓	$\circ$	26.3	45.0	26.6
$\circ$	$\circ$	✓	28.9	47.1	29.3
✓	✓	$\circ$	30.4	49.1	31.2
$\circ$	✓	✓	31.7	50.6	32.8
✓	$\circ$	✓	31.3	50.5	32.6
✓	✓	✓	**32.5**	**51.9**	**33.8**

TABLE 5
Performance comparison with other heuristic designs for I_z. Each variable formulation is simplified as in Table 4. The first column describes the design method.

the heuristic I^z design	AP	AP_{50}	AP_{75}
Generating I^z from the feature map	27.9	46.5	28.5
I^q has FPN's box regression style	31.8	50.3	32.7
Log(I^o)	32.2	51.6	33.6
Sigmoid(I^z)	32.3	51.6	33.6
Tanh(I^z)	32.3	51.8	33.4
default Design	**32.5**	**51.9**	**33.8**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3085295, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 6
Ablation study for instance-aware weight prediction structure. FC and ReLU represent the fully connected layer and rectified linear unit, respectively. The "+" symbol represents cascading two operations in sequence. $2\times$ means repeating this operation module twice.

structure	AP	AP_{50}	AP_{75}
FC+ReLU	32.5	51.9	**33.8**
FC	31.7	51.0	33.4
2×(FC+ReLU)	**32.6**	**52.0**	33.6
FC+ReLU+FC	32.3	51.7	33.6

TABLE 7
Ablation study for information fusion between the instance-agnostic template and instance-aware weight. For feature addition and concatenation, we reshaped both our instance-agnostic feature and instance-aware weight to $16 \times 16 \times 9$.

combination	AP	AP_{50}	AP_{75}
addition	31.1	50.9	33.5
concatenation	30.9	50.6	33.6
3×3 convolution	**32.5**	51.8	33.7
1×1 convolution	**32.5**	**51.9**	**33.8**

indicators are critical to our performance. The design of the last row is our default setting. For other designs, regardless of how the hand-crafted features are designed, the final performance is relatively robust. We suppose this robustness is produced by our proposed weight prediction structure (FC Layer). This structure could handle the variations in our designed features and then produce instance-aware weights. The performance change among various element-wise operations, such as Log, Sigmoid, or Tanh, is marginal for I^o or I^q. Using the FPN's box regression style in I^q produces an AP lower by approximately 0.7 than using FCOS's style. The main difference between these two regression styles has been illustrated in Fig. 2. The performance comparison (Table 5) shows that the boundary distance between anchor and target box (Fig. 2 (b)) is more helpful to generate offset weight than width or height variance (Fig. 2 (a)).

Table 6 reveals the influence of using different weight prediction structures. This table indicates that there are no large differences when a single FC layer is used and not used. The performance of two layers is only 0.1 higher than a single layer. The reason is that our input only contains 10 elements. Using more layers does not greatly enhance information in this setting. Moreover, the ReLU operation guarantees non-linearity in our module. The reason is that the generated dynamic weights are used to convolve our instance-agnostic feature to obtain our target instance-aware feature. From this point of view, we could regard instance-agnostic features as templates and dynamic weights as warp operations. Therefore, some zero values should be guaranteed in our generated dynamic weights, similar to the affinity translation matrix.

Instance-aware Convolution Table 7 summarizes an ablation study on different fusion methods of the instance-agnostic template and instance-aware weight. It can be seen that directly adding or concatenating them leads to performance reduction. The reason is that these two features come from different sources and thus have a distinct meaning. Specifically, the instance-agnostic template has robust semantic information, whereas the instance-aware weight indicates shifting information between the template and proposals. As a result, using dynamic convolution is more appropriate for fusing them by interpolating semantic information with geometric offsets. With this consideration, our design is effective in aligning instance-agnostic mask features given different box information. Increasing the convolution kernel size of 1×1 to 3×3 does not further increase performance, so we choose 1×1 kernel, further reducing the computation cost of our instance-aware convolution.

TABLE 8
The ablation study on number of sampled positive points. "all" means we sample all the positive points once they satisfy our sample criterion.

Number of Positive Points	AP	AP_{50}	AP_{75}
32	29.6	48.9	31.8
64	31.7	51.1	33.0
128	**32.5**	**51.9**	**33.8**
256	32.4	51.7	**33.8**
all	**32.5**	**51.9**	33.7

Sampling Strategy We explore the effectiveness of the sampling strategy. As illustrated in Section 3.3 (Computation Optimization), we randomly sample 128 positive points by default. The influence about the number of positive points on the performance is initially ablated, as presented in Table 8. The mask improves as the number of positive points increases from 32 to 128. Further increasing the number of positive points do not improve performance because of the limited number of positive points in the COCO image.

Table 9 outlines an ablation study on the IoU threshold for positive points. The second row displays our default setting with which the IoU between the ground truth and both its anchors and regressed boxes is greater than 0.5. The performance dramatically decreases when we reduce the anchor's IoU threshold to 0.4. The reason for this is the sampling strategy adopted in the detection branch. Only anchors whose IoU are larger than 0.5 are used for box training. This means that the other anchors could not be trained; thus, the quality of their generated boxes is not guaranteed. However, increasing the IoU threshold of 0.5 to another value like 0.6 or 0.7 reduces the mask performance due to the network underfitting with fewer positive points.

Due to the heuristic design for our dynamic weight prediction input I^z, we can sample three types of target boxes for each point in training, including the anchor, ground truth, and regressed box. We ablate the influence of the sampling ratio of these three types, as illustrated in Table 10. We find using regressed boxes to compute I^z along with a few both anchor and ground truth boxes (with both sample ratio 0.05 in training) could make our instance-aware weight generation module be robust to more extreme

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3085295, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 9
The ablation study on the sample criterion for positive points. As illustrated in section 3.3, a point is defined positive if the Intersection of Union (IoU) between ground truth and one of the anchors and target boxes are larger than 0.5.

IoU(Anchor, GT)	IoU(Box, GT)	AP	AP_{50}	AP_{75}
0.4	0.5	30.2	50.7	32.7
0.5	0.5	**32.5**	**51.9**	**33.8**
0.5	0.6	32.0	51.2	33.1
0.5	0.7	31.0	50.1	32.2
0.6	0.6	31.7	50.9	32.9

TABLE 10
Ablation study on sample ratio for each target box type. The sample number is 128. Our default setting is presented in the last row. GT refers to ground truth.

Anchor	GT	Regressed Box	AP	AP_{50}	AP_{75}
1.0	0.0	0.0	24.5	42.7	25.3
0.0	1.0	0.0	32.4	51.8	**33.9**
0.0	0.0	1.0	31.9	51.4	33.4
0.05	0.05	0.9	**32.5**	**51.9**	33.8

cases. It achieves the best performance 32.5 compared with other sampling strategies. The performance of sampling the anchors' corresponding ground truth for all points is slightly lower than our default setting (32.4 vs 32.5). This indicates that our framework could be trained without the regressed boxes, and box information is required only to segment valid regions in inference. The performance of only using regressed boxes is inferior to our default setting. The reason for this is that there are no predicted boxes with high quality at the earlier iteration in training.

Inference Time Table 11 summarizes an ablation study on inference time with different scales in Nvidia V100 GPU. Clearly, in the scale of 800 and 12 training epochs, PointINS gives faster inference speed among all compared methods. In performance, it exhibits a notable gain compared to other PoI-based Polarmask (32.5 vs. 29.1) and MEInst (32.5 vs. 30.3), only inferior to RoI-based Mask R-CNN (32.5 vs. 34.4). We give the speed-accuracy trade-off curve in the Figure 6.

However, when using data augmentation and longer training time (72 epochs), our method holds its inference speed advantage and gives competitive performance compared with Mask R-CNN (36.7 vs. 36.8), as shown in Table 12. Additionally, when the input scale decreases (*e.g.*, 400), our model still achieves the practical performance (25.8) at real-time speed (27.3 fps). It indicates that PointINS can not only achieve high performance in mask AP, but also can be applied to real-time applications.

4.1.2 Comparison with State-of-the-Art Methods

Comparison among point-based methods We have implemented our method based on RetinaNet [37] and FCOS [60], and compare them with three state-of-the-art point-based instance segmentation frameworks: TensorMask [10], PolarMask [64], and MEInst [70], as described in Table 12. It is

TABLE 11
Ablation study on the inference time with different scales. All experiments are conducted with ResNet50 backbone.

Scale	Method	AP	AP_{50}	AP_{75}	FPS
800	PolarMask [64]	29.1	49.5	29.7	17.2
800	MEInst [70]	30.3	53.0	31.1	17.6
800	PointINS	32.5	51.9	33.8	18.2
600		30.3	49.4	31.3	22.4
400		25.8	44.7	26.4	**27.3**
800	Mask R-CNN [19]	**34.4**	**55.1**	**36.7**	16.1

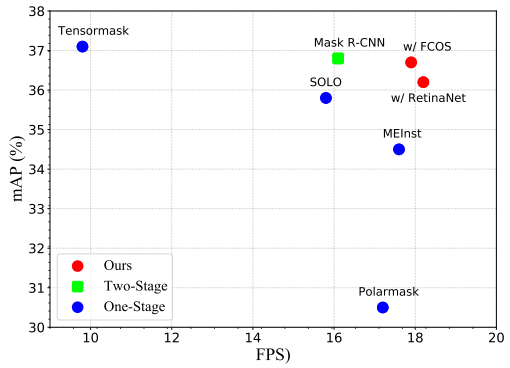


Fig. 6. The speed-accuracy trade-off curve of the state-of-the-art methods.

clear that our PointINS framework performs the best among all frameworks. In the following, We present a comparison with these approaches.

PolarMask [64] uses polar representation as an approximation to the instance mask. With 12 epochs of training, PolarMask achieves 29.1 mAP with R-50-FPN backbone, and 30.4 mAP with R-101-FPN. With RetinaNet, our PointINS achieves 32.2 mAP, with an improvement of 3.1 points. The margin is even larger when we use FCOS as the detection base, we achieve 33.4 and 34.5 with R-50-FPN and R-101-FPN backbones, respectively. Note that the polar representation is an approximation to the ground truth, which intrinsically leads to the precision loss. In contrast, our method uses a pixel-wise mask and does not suffer from this problem. Our method with ResNet50 with only single-scale $1\times$ training still outperforms the method with ResNet101 under multi-scale $2\times$ training. $1\times$ and $2\times$ training mean training the network in 12 and 24 epochs.

MEInst [70] encodes a mask into a compact representation by principal component analysis (PCA). The network responds to generate feature encodings and finally decodes them into instance masks. Like the polar representation in PolarMask, this representation is coarser than the original ground truth, thus affecting the final segmentation quality. With the ResNet-101 backbone, MEInst only obtains the performance of approximately 33.0, which is lower than ours with only ResnetNet-50 backbone. In particular, large objects in MEInst lead to poorer performance because the compact representation misses too many details of large

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3085295, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 12
Comparison with current point-based instance segmentation frameworks on test-dev split of COCO dataset. 'Aug' refers to data augmentation, including multi-scale and random crop. ✓(or ○) signifies that the network is trained with (or without) data augmentation. 'Epoch' indicates the training time. 12 epochs is a baseline widely used in most previous work [19], [40], [37], [60]. The X-101-FPN-DCN backbone is FPN [36] backbone with ResNeXt 101 [65] equipped with deformable convolution [15], [73].

Two-stage									
method	backbone	aug	epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [13]	R-101-C4	○	12	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [35]	R-101-C5-dilated	○	12	29.2	49.5	-	7.1	31.3	50.0
Mask R-CNN [19]	R-50-FPN	✓	72	36.8	59.2	39.3	17.1	38.7	52.1
	R-101-FPN	✓	72	38.3	61.2	40.8	18.2	40.6	54.1
	X-101-FPN	○	12	37.1	60.0	39.4	16.9	39.9	53.5
	X-101-FPN-DCN	✓	72	42.5	66.1	46.4	25.0	46.5	58.9
One-stage (Segmentation-based)									
method	backbone	aug	epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YoLACT [4]	R-50-FPN	✓	45	28.2	46.6	29.2	9.2	29.3	44.8
	R-101-FPN	✓	45	31.2	50.6	32.8	12.1	33.3	47.1
SOLO [4]	R-101-FPN	✓	36	37.8	59.5	40.4	16.4	40.6	54.2
EmbedMask [69]	R-101-FPN	✓	36	37.7	59.1	40.3	17.9	40.4	53.0
One-stage (Point-based)									
method	backbone	aug	epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ExtremeNet [72]	Hourglass-101	✓	100	18.9	44.5	13.7	10.4	20.4	28.3
TensorMask [10]	R-50-FPN	✓	72	35.4	57.2	37.3	16.3	36.8	49.3
	R-101-FPN	✓	72	37.1	59.3	39.4	17.4	39.1	51.6
PolarMask [64]	R-50-FPN	○	12	29.1	49.5	29.7	12.6	31.8	42.3
	R-101-FPN	○	12	30.4	51.1	31.2	13.5	33.5	43.9
	R-101-FPN	✓	24	32.1	53.7	33.1	14.7	33.8	45.3
	X-101-FPN-DCN	✓	24	36.2	59.4	37.7	17.8	37.7	51.5
MEInst [70]	R-101-FPN	○	12	33.0	56.4	34.0	15.2	35.3	46.3
	R-101-FPN	✓	36	33.9	56.2	35.4	19.8	36.1	42.3
	X-101-FPN-DCN	✓	36	38.2	61.7	40.4	**22.6**	40.0	49.3
RetinaNet + ours	R-50-FPN	○	12	32.2	51.6	33.4	13.4	34.4	48.4
	R-101-FPN	○	12	33.5	53.6	34.1	14.2	35.3	49.0
	R-50-FPN	✓	72	36.2	58.1	37.8	16.5	37.7	50.5
	R-101-FPN	✓	72	37.9	60.1	39.7	17.8	40.1	52.1
FCOS + ours	R-50-FPN	○	12	33.4	53.7	35.2	14.8	36.3	48.8
	R-101-FPN	○	12	34.5	54.5	36.6	15.4	37.0	49.4
	R-50-FPN	✓	72	36.7	58.2	38.1	16.8	38.0	50.9
	R-101-FPN	✓	72	38.3	60.3	40.0	18.1	40.3	52.4
	X-101-FPN-DCN	✓	72	**42.0**	**63.7**	**44.3**	21.0	**43.5**	**55.9**

objects.

Compared to the Tensormask [10], we achieve an improvement of 0.8 (37.1 $\rightarrow$ 37.9) and 1.2 (37.1 $\rightarrow$ 38.3) with RatinaNet and FCOS, respectively. The cause of our success is twofold. First, the instance-agnostic feature we use is more informative and robust than the aligned representation proposed by Tensormask. Second, we propose the instance-aware module, which allows our framework to capture accurate masks for the predicted instances.

Note that by applying data augmentation and training for a longer time, our performance further improves significantly. By training for as long as 72 epochs and using data augmentation, our model with RetinaNet achieves 37.9 mAP with R-101-FPN backbone, while with FCOS it achieves 38.3 mAP. For a fair comparison with the previous winner, TensorMask [10], we use the same hyperparameter setting.

Comparison with State-of-the-Arts Methods Table 12 also demonstrates that PointINS can obtain the comparable performance to Mask R-CNN [19] and SOLO [61]) by only using a single-point feature. This demonstrates that PointINS is a promising method that does not involve RoI feature extraction, and the point feature can be further improved with more elaborate designs.

4.1.3 Correlation to object detection.

Table 13 presents the associated detection performance of Tensormask and our PointINS framework on COCO test-dev set. Initially, the mask performance (36.2) of our RetinaNet-based PointINS is higher than that (35.4) of Ten-

TABLE 13
The associated detection performance of the Tensormask and PointINS with ResNet50 backbone. All the experiemnts are conducted with data augmentation and 6× training scheme.

method	backbone	type	AP	AP_{50}	AP_{75}
TensorMask	R-50-FPN	box	41.6	61.0	45.1
		mask	35.4	57.2	37.3
RetinaNet + ours	R-50-FPN	box	41.8	61.2	45.3
		mask	**36.2**	58.1	37.8
FCOS + ours	R-50-FPN	box	42.3	61.7	45.9
		mask	**36.7**	58.2	38.1

TABLE 14
The comparison between RetinaMask and RetinaNet ours with aligning all the training details including sample way and loss function. The AP^B and AP^M are the mAP of the bounding box and instance mask respectively.

method	AP^B	AP^M	FPS
RetinaMask [16]	36.7	31.9	15.9
RetinaNet+Ours	36.7	32.2	18.2

sormask with approximately the same box quality (41.6 vs 41.8). This shows that the mask improvement of the pointINS results from our designed modules, including instance-agnostic feature generation and instance-aware convolution. Both help our model capture more robust and aligned features for each specific box. Furthermore, the mask quality obtains consistent improvement with an effective one-stage detector such as FCOS.

4.1.4 Comparison to RetinaMask

Given the reason that the difference between RetinaMask [16] and RetinaNet [37] + Ours is the mask feature extracted from Region-of-Interest (RoI) or Point-of-Interest (PoI). We compared them in the Table 14 with aligning all the training details including sample way and loss function. We could see that our method is faster than RetinaMask with comparable or better performance. Our efficiency is brought by that we do not explicitly crop the RoI features with time-consuming RoI Alignment. Instead, ours use dynamic weight and instance-agnostic feature to produce final instance-aware features. This process is both effective and efficient by being optimized in parallel.

4.1.5 Visualization

Figure 7 shows a visualization of our predicted instances. Notably, our approach can easily detect small objects, as our employed dense one-stage detector excels at this. This is also verified in Table 12 with better AP_s than SOLO [61]. For large objects, the boundary is slightly worse than that of small ones. This signifies that a single-point feature in our PointINS compressed detailed information for large regions due to the pyramid features used in object detection and instance segmentation frameworks. Like RetinaNet and FCOS, large objects are detected with P6 or P7 features, which are highly abstract to estimate the location of large objects. However, they are not suitable for pixel-level instance segmentation because they lose details by downsampling. Longer training time and data augmentation can relieve this problem, as illustrated in Table 12.

4.2 Experiments on Cityscapes

Dataset and Metrics The Cityscapes [11] dataset contains street scenes captured by car-mounted cameras. There are 2,975 training, 500 validation, and 1,525 testing images with fine annotations. Here, we report our results on both validation and test subset. Eight semantic classes are annotated with instance masks, and each image had a size of 1024×2048. We evaluate results based on AP and AP50.

Hyper-parameters We use images with shorter edges randomly sampled from $\{800, 1024\}$ for training, and images with a shorter edge length of 1024 for inference. We train our model with a learning rate of 0.0005 for 18k iterations and with a rate of 0.00005 for other 6k iterations. Eight images (1 image per GPU) are in one image batch.

Results Table 15 presents the results of our PointINS framework with the FCOS-Res50 backbone. PointINS outperforms RoI and segmentation-based methods with a simple yet efficient structure.

5 Conclusion

In this study, we introduce PointINS to convert current dense one-stage detectors for instance segmentation by a single-point feature. The core module is instance-aware convolution, ensuring that single-point features are sufficiently expressive for instance masks. This module decomposes a single point feature into two tractable modules: an instance-agnostic feature and instance-aware weight generation modules. This design ensures that high-dimensional instance-aware mask features of several positive points are aligned (by convolving the template with dynamic weights) and are then used for mask prediction. The experiments show that the proposed framework achieves competitive accuracy and speed among point-based frameworks. Furthermore, the proposed framework is comparable to the popular Mask R-CNN framework.

In the future, we will explore new dynamic weight generation methods to make a single-point feature robust. In addition, we will study how to apply this concept to other instance-based downstream tasks like panoptic segmentation [26], [66], [33] or visual relationships prediction [29] or dense captioning [25].

6 Acknowledgment

This work was supported by The National Key Research and Development Program of China (No. 2020AAA0105200) and Beijing Academy of Artificial Intelligence (BAAI).

References

[1] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016.
[2] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017.
[3] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016.

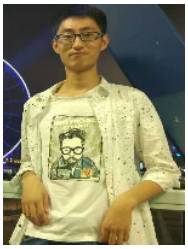
Fig. 7. Visualization of RetinaNet-based PointINS with ResNet101 on COCO dataset under $1\times$ (12 epochs) training. For all images, we set the mask confidence threshold to 0.5.

TABLE 15
Resuts on cityscapes dataset. All experiments are conducted with ResNet50. "Mcycle" is short for motorcycle.

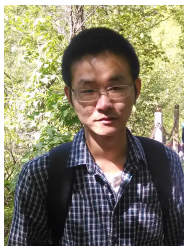
Method	AP_{val}	AP	AP_{50}	person	rider	car	truck	bus	train	mcycle	bicycle
SGN [39]	29.2	25.0	44.9	21.8	20.1	39.4	**24.8**	**33.2**	**30.8**	17.1	12.4
Mask R-CNN [19]	31.5	26.2	49.9	30.5	23.7	46.9	22.8	32.2	18.6	19.1	16.0
Ours	**32.4**	**27.3**	**50.5**	**34.1**	**26.1**	**51.9**	20.9	30.9	15.9	**20.5**	**19.2**

[4] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. Yolact: Real-time instance segmentation. In *ICCV*, 2019.
[5] Q. Cai, Y. Pan, T. Yao, C. Yan, and T. Mei. Memory matching networks for one-shot image recognition. In *CVPR*, 2018.
[6] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV*, 2016.
[7] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.
[8] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, et al. Hybrid task cascade for instance segmentation. In *CVPR*, 2019.
[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *PAMI*, 2016.
[10] X. Chen, R. Girshick, K. He, and P. Dollár. Tensormask: A foundation for dense object segmentation. In *ICCV*, 2019.
[11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.
[12] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016.
[13] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016.
[14] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: object detection via region-based fully convolutional networks. In *NeurIPS*, 2016.
[15] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017.
[16] C.-Y. Fu, M. Shvets, and A. C. Berg. Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free. *arXiv preprint arXiv:1901.03353*, 2019.
[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.
[18] R. B. Girshick. Fast R-CNN. In *ICCV*, 2015.
[19] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *ICCV*, 2017.
[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
[21] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang. Bounding box regression with uncertainty for accurate object detection. In *CVPR*, 2019.
[22] R. Hu, P. Dollár, K. He, T. Darrell, and R. Girshick. Learning to segment every thing. In *CVPR*, 2018.
[23] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun. Meta-sr: A magnification-arbitrary network for super-resolution. In *CVPR*, 2019.
[24] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang. Mask scoring r-cnn. In *CVPR*, 2019.
[25] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *CVPR*, 2016.
[26] A. Kirillov, R. Girshick, K. He, and P. Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019.
[27] T. Kong, F. Sun, C. Tan, H. Liu, and W. Huang. Deep feature pyramid reconfiguration for object detection. In *ECCV*, 2018.
[28] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*, 2016.
[29] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.
[30] H. Lee, S. Eum, and H. Kwon. Me r-cnn: multi-expert region-based cnn for object detection. In *ICCV*, 2017.
[31] Y. Lee and J. Park. Centermask: Real-time anchor-free instance segmentation. In *CVPR*, 2020.
[32] C. Lemke, M. Budka, and B. Gabrys. Metalearning: a survey of trends and technologies. In *Artificial intelligence review*, 2015.
[33] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang.

Attention-guided unified network for panoptic segmentation. In *CVPR*, 2019.
[34] Y. Li, Y. Chen, N. Wang, and Z. Zhang. Scale-aware trident networks for object detection. In *ICCV*, 2019.
[35] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, 2017.
[36] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.
[37] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.
[38] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
[39] S. Liu, J. Jia, S. Fidler, and R. Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *ICCV*, 2017.
[40] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018.
[41] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.
[42] Y. Liu, R. Wang, S. Shan, and X. Chen. Structure inference net: Object detection using scene-level context and instance-level relationships. In *CVPR*, 2018.
[43] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
[44] D. Morrison, A. W. Tow, M. McTaggart, R. Smith, N. Kelly-Boxall, S. Wade-McCue, J. Erskine, R. Grinover, A. Gurman, T. Hunn, et al. Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge. In *ICRA*, 2018.
[45] M. Najibi, M. Rastegari, and L. S. Davis. G-cnn: an iterative grid based object detector. In *CVPR*, 2016.
[46] M. Najibi, B. Singh, and L. S. Davis. Autofocus: Efficient multi-scale inference. In *ICCV*, 2019.
[47] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin. Libra r-cnn: Towards balanced learning for object detection. In *CVPR*, 2019.
[48] S. Peng, W. Jiang, H. Pi, H. Bao, and X. Zhou. Deep snake for real-time instance segmentation. In *CVPR*, 2020.
[49] L. Qi, L. Jiang, S. Liu, X. Shen, and J. Jia. Amodal instance segmentation with kins dataset. In *CVPR*, 2019.
[50] Z. Qin, Z. Li, Z. Zhang, Y. Bao, G. Yu, Y. Peng, and J. Sun. Thundernet: Towards real-time generic object detection on mobile devices. In *ICCV*, 2019.
[51] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2016.
[52] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.
[53] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. In *arXiv*, 2018.
[54] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.
[55] A. Shrivastava and A. Gupta. Contextual priming and feedback for faster r-cnn. In *ECCV*, 2016.
[56] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.
[57] G. Shu. Human detection, tracking and segmentation in surveillance video. 2014.
[58] B. Singh and L. S. Davis. An analysis of scale invariance in object detection snip. In *CVPR*, 2018.
[59] Z. Tan, X. Nie, Q. Qian, N. Li, and H. Li. Learning to rank proposals for object detection. In *ICCV*, 2019.
[60] Z. Tian, C. Shen, H. Chen, and T. He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019.
[61] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li. Solo: Segmenting objects by locations. In *arXiv*, 2019.
[62] X. Wang, A. Shrivastava, and A. Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *CVPR*, 2017.
[63] Y.-X. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016.
[64] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo. Polarmask: Single shot instance segmentation with polar representation. In *CVPR*, 2020.
[65] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.
[66] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun. Upsnet: A unified panoptic segmentation network. In *CVPR*, 2019.
[67] B. Yang, G. Bender, Q. V. Le, and J. Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *NeurIPS*, 2019.
[68] T. Yang, X. Zhang, Z. Li, W. Zhang, and J. Sun. Metaanchor: Learning to detect objects with customized anchors. In *NeurIPS*, 2018.
[69] H. Ying, Z. Huang, S. Liu, T. Shao, and K. Zhou. Embedmask: Embedding coupling for one-stage instance segmentation. In *arXiv*, 2019.
[70] R. Zhang, Z. Tian, C. Shen, M. You, and Y. Yan. Mask encoding for single shot instance segmentation. In *CVPR*, 2020.
[71] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.
[72] X. Zhou, J. Zhuo, and P. Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019.
[73] X. Zhu, H. Hu, S. Lin, and J. Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019.

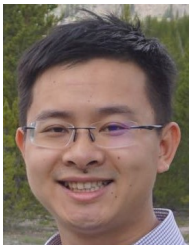


Lu QI received the M.S. degree from Institute of Automation, Chinese Academy of Sciences in 2017. He is currently working towards the Ph.D. degree in the department of computer science and engineering from the Chinese University of Hong Kong. He obtained the Hong Kong PhD Fellowship in 2017. His current research interests include object detection, instance segmentation and knowledge distillation.



Yi Wang received his M.Sc. degrees from Peking University in 2017. He is currently working towards the Ph.D. degree in the department of computer science and engineering from the Chinese University of Hong Kong. His research interests include image generation, computer vision, and machine learning.

Yukang Chen received the M.S. degree from Institute of Automation, Chinese Academy of Sciences in 2020. He is currently working towards the Ph.D. degree in the department of computer science and engineering from the Chinese University of Hong Kong. His current research interests include object detection, instance segmentation and neural architecture search.



Ying-Cong Chen received his PhD degree from the Chinese University of Hong Kong. He is currently a postdoctoral associate at the Computer Science and Artificial Intelligence Lab (CSAIL), Massachusetts Institute of Technology. He obtained the Hong Kong PhD Fellowship in 2016. He serves as a reviewer for IJCV, TIP, CVPR, ICCV, ECCV, BMVC, IJCAI, AAAI, etc. His research interest includes deep learning, image generation and editing, generative adversarial networks, etc.

Xiangyu Zhang received his PhD degree from Xi'an Jiaotong University in 2017. He is currently a senior researcher at MEGVII Research. His primary research interests focus on deep learning for computer vision, especially including CNN architecture design and acceleration. He got CVPR Best Paper Award in 2016. He also serves as a reviewer for CVPR, ICCV, ECCV, NeurIPS, ICLR, TPAMI, etc. The total number of his Google Scholar citations is over 100,000.

Jian Sun received the BS, MS, and PhD degrees from Xian Jiaotong University, in 1997, 2000, and 2003, respectively. He is currently a chief scientist in Megvii Technology. He worked with Microsoft Research Asia (MSRA) from July 2003 to July, 2016. he has been working in the fields of computer vision and computer graphics, with particular interests in solving fundamental research problems and building real-world working systems. His primary research interests are computational photography and deep learning based image understanding.

Jiaya Jia received the Ph.D. degree in Computer Science from Hong Kong University of Science and Technology in 2004 and is currently a full professor in Department of Computer Science and Engineering at the Chinese University of Hong Kong (CUHK). He is in the editorial boards of IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) and International Journal of Computer Vision (IJCV). He continuously served as area chairs for ICCV, CVPR, AAAI, ECCV, and several other conferences for organization. He was on program committees of major conferences in graphics and computational imaging, including ICCP, SIGGRAPH, and SIGGRAPH Asia. He is a Fellow of the IEEE.