Pointly-Supervised Instance Segmentation

Bowen Cheng[1]* Omkar Parkhi[2] Alexander Kirillov[2]

[1]UIUC [2]Facebook AI

Abstract

We propose point-based instance-level annotation, a new form of weak supervision for instance segmentation. It combines the standard bounding box annotation with labeled points that are uniformly sampled inside each bounding box. We show that the existing instance segmentation models developed for full mask supervision, like Mask R-CNN, can be seamlessly trained with the point-based annotation without any major modifications. In our experiments, Mask R-CNN models trained on COCO, PASCAL VOC, Cityscapes, and LVIS with only 10 annotated points per object achieve 94%–98% of their fully-supervised performance. The new point-based annotation is approximately 5 times faster to collect than object masks, making high-quality instance segmentation more accessible for new data.

Inspired by the new annotation form, we propose a modification to PointRend instance segmentation module. For each object, the new architecture, called Implicit PointRend, generates parameters for a function that makes the final point-level mask prediction. Implicit PointRend is more straightforward and uses a single point-level mask loss. Our experiments show that the new module is more suitable for the proposed point-based supervision.[1]

1. Introduction

The task of instance segmentation requires an algorithm to locate objects and delineate them with pixel-level binary masks. Manual annotation of object masks for training is significantly more complex and time-consuming than other forms of image annotation like image-level categories [61, 1, 62, 9, 13, 34, 15] or per-object bounding boxes [23, 21, 2, 53]. For example, it takes on average 79.2 seconds per instance to create polygon-based object masks in COCO [33], whereas a bounding box for an object can be annotated ∼11 times faster in only 7 seconds [41].

Weakly-supervised methods, that use easier to acquire ground truth annotation forms, make instance segmentation more accessible for new categories or scene types, as the

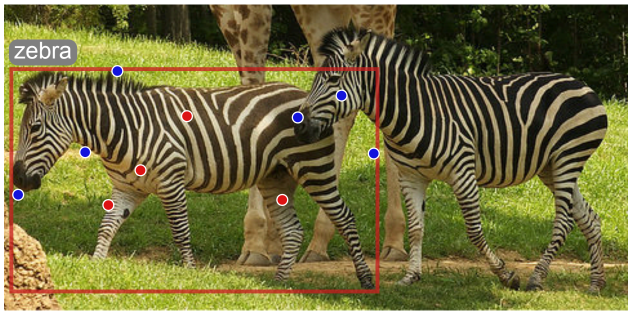


Figure 1. **Point-based instance annotation** combines object bounding boxes with points that are sampled randomly inside each box and annotated as the object (red) or background (blue). We demonstrate that 10 annotated points per instance is faster to collect than the standard object masks and such ground truth is sufficient to train a standard model like Mask R-CNN [19] to achieve 94%–98% of its fully-supervised performance on various datasets.

efforts required to collect the data are lower. Such models showed a great progress on smaller datasets under a fixed annotation time budget [4, 28], however, they are still considerably behind fully-supervised methods for large-scale datasets like COCO. The newest BoxInst model [53] significantly outperforms previous weakly-supervised approaches but achieves only 85% of its fully-supervised counterpart performance on COCO. A natural question emerges: Is object mask training data necessary to get closer to the fully-supervised performance? And is there an easier to collect annotation form for the instance segmentation task?

Beyond bounding boxes and image-level categories, point clicks and squiggles are the other time-efficient annotation forms most commonly used in interactive segmentation scenarios [35]. Several semantic and instance segmentation methods directly use them for training [4, 31]. In our paper we present a new instance segmentation annotation form that combines bounding boxes with point-based annotation (see Figure 1). Unlike previous works where points are clicked by annotators, we follow a different process where points are sampled randomly inside an object bounding box and an annotator is asked to classify each point as the object or background. Such point classification task is more straightforward and collected random points have better variability than clicked points [4].

*Work done during an internship at Facebook AI Research.

[1]Project page: https://bowenc0221.github.io/point-sup

With randomly sampled points, the annotation process can be easily simulated for datasets with existing instance segmentation ground truth. This property allows us to test the new annotation form and show its efficacy on multiple large-scale datasets without a major annotation effort that is usually a prerequisite for a new annotation format [4, 31].

The key advantage of the new point-based annotation is in its ability to seamlessly supervise instance segmentation models that directly predict object masks with no changes to their architectures or training pipelines. In our experiments, we train Mask R-CNN [19], PointRend [25], and CondInst [52] with the new point-based supervision. For each object, the standard mask loss is computed for ground truth points by interpolating mask predictions at these points. We show that the new point-based annotation form is applicable across different categories and scene types by testing it on COCO [32], PASCAL VOC [12], LVIS [17], and Cityscapes [11] datasets. Mask R-CNN models trained with only 10 annotated points per object achieve 94%–98% of their fully-supervised performance on these datasets. In addition, we propose a simple point-based data augmentation strategy and explore self-training paradigm for point supervision to show that the gap to the full mask supervision can be further reduced.

To analyze the performance/annotation time trade-off of the new annotation form, we created a simple labeling tool and measured that a trained human annotator classifies a point in 0.9 seconds on average. This means, together with a bounding box annotation that can be done in 7 seconds via extreme point clicking [41], the total annotation time for 10 points per object is 16 seconds ($7 + 10 \cdot 0.9$). This is 5 times faster than polygon-based mask annotation in COCO. For a new dataset, the point-based annotation allows the standard models to achieve performance close to full supervision at a fraction of the full data collection time.

In our experiments, we observe that PointRend [25] supervised with point-based annotations performs on par with Mask R-CNN [19] supervised in the same way, whereas with the standard full mask supervision it performs considerably better. Inspired by this finding, we propose a modification to the PointRend module which we name **Implicit PointRend**. Instead of a coarse mask prediction used in PointRend to provide region-level context to distinguish objects, for each object Implicit PointRend generates different parameters for a function that makes the final point-wise mask prediction. The new model is more straightforward than PointRend: (1) it does not require an importance point sampling during training and (2) it uses a single point-level mask loss instead of two mask losses. Implicit PointRend can be trained directly with point supervision without any intermediate prediction interpolation steps. Our experiments demonstrate that the new module significantly outperforms PointRend with point supervision.

2. Related Works

Fully-supervised instance segmentation models that currently dominate popular COCO benchmark [33] predict object masks directly. Mask R-CNN-based methods [19] make such predictions on a region level from features that correspond to detected bounding boxes, whereas methods like YOLACT++ [6] or CondInst [52] make image-level mask predictions. In both cases ground truth masks are directly used for supervision. Alternatively, bottom-up segmentation methods output masks indirectly, forming them from a set of predicted cues such as instance boundaries [24], energy levels [3], and offset vectors to center points [8]. In our work, we show that methods that use direct mask supervision can be trained with the new point-based annotation without any significant modifications.

Weakly-supervised instance segmentation methods usually use image-level category labels or bounding boxes. With image-level labels, such methods rely on segmentation proposals [54, 43] either re-ranking them [61] or generating pseudo-ground truth [1, 62, 2]. These methods show promising results on smaller datasets, but no competitive results have been shown on a large-scale dataset like COCO [33]. With bounding box supervision, SDI [23] proposes a multi-stage training procedure that generates pseudo-ground truth with GrabCut [48]. BBTP [21] trains Mask R-CNN [19] using a bounding box tightness prior. Recently proposed BoxInst [53] supervises the mask branch of CondInst [52] with a projection loss that forces horizontal/vertical lines inside bounding boxes to predict at least one foreground pixel and an affinity loss that forces pixels with similar colors to have the same label. BoxInst outperforms previous approaches, achieving an impressive 85% of fully-supervised performance on COCO [33]. In our work, we show that additional point-based supervision allows us to get much closer to the quality of fully-supervised models. In addition, our experiments suggest that the point-based annotation has a better performance/annotation time trade-off for a wide variety of annotation efforts.

Point-based supervision has been studied in a variety of tasks, including action localization [38, 37], object detection [42, 46], object counting [27], and semantic segmentation [4, 44]. For instance segmentation, point clicks are often used in interactive pipelines [59, 30, 29, 35, 5]. Such systems are trained with full supervision and require user input at test time, while we use the point-based annotation during training only. Laradji *et al.* [28] present a proposal-based instance segmentation method that uses a single point per instance as supervision. In contrast, we use multiple labeled points per instance and bounding box annotation together. Furthermore, instead of collecting annotator clicks, we randomly generate point locations for annotation which improves points variety as well as model performance [4].

3. Pointly-Supervised Instance Segmentation

3.1. Annotation format and collection

The new point-based instance segmentation annotation form is conceptually simple. In addition to the standard bounding box annotation, we assume N points within each object bounding box to be labeled as the object *vs.* background (Figure 1). We refer to this annotation form as $\mathcal{P}_N$.

Random points instead of clicks. Previous point-based annotation procedures collected annotators clicks [4, 44, 28, 38, 42, 27]. Such strategy, however, can lead to data with low variability as human clicks are often correlated [14, 10]. Bearman *et al.* [4] confirm this by showing that training with randomly located ground truth points is superior for their semantic segmentation model than training with clicks. Following these findings we randomly sample point locations within each object bounding box and an annotator is asked to classify each point as the object or background.

Collection and simulation. The annotation procedure for $\mathcal{P}_N$ is straightforward. First, bounding boxes for objects are collected using any off-the-shelf solution [41, 26, 18]. Next, for each object, N random point locations within its bounding box are sequentially presented to an annotator for a binary object/background classification given the bounding box and object category label.

Unlike click-based procedures where human is involved in the point selection process, the new annotation collection can be seamlessly simulated for datasets that have full instance segmentation ground truth by generating a bounding box for each instance mask and classifying randomly sampled points/pixels inside the box based on the corresponding positions in the ground truth mask. By simulating the new annotation format, we are able to ablate its design and verify our results on a diverse set of large-scale datasets without time consuming data collection efforts.

Annotation time. To measure the annotation time, we build a simple tool for point classification that shows a random point location inside a bounding box and an annotator performs the binary classification using two keyboard buttons (see the appendix for details). With this tool we annotated 100 objects from the COCO [33] and LVISv1.0 [17] datasets and found that it takes 0.9 seconds on average to classify a point. Note, that a bounding box can be annotated in 7 seconds using extreme points method [41]. Thus, the total annotation time of $\mathcal{P}_N$ per object is 7 (bounding box) $+ 0.9 \cdot N$ (binary classification for N points) seconds.

In our experiments we found that $\mathcal{P}_{10}$ provides a good trade-off between annotation time (16 seconds per object) and performance (94%-98% of fully-supervised performance). Such annotation is $\sim$5 times faster to label than a polygon-based instance mask in COCO that takes on average 79.2 seconds for a spotted object [33].

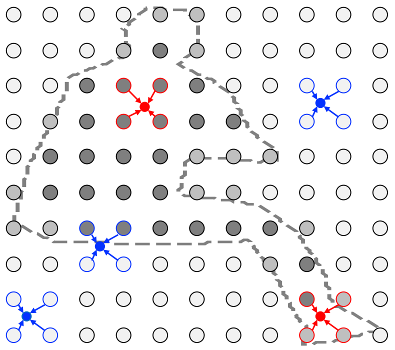


Figure 2. **Illustration of point supervision.** For a 10×10 prediction mask on the regular grid (darker color indicates foreground prediction), we get predictions at the exact locations of ground truth points (red and blue indicate foreground and background ground truth points respectively) with bilinear interpolation. Note, that the object contour line is only for the purpose of illustration.

Annotation quality. Collected point labels have 90% agreement with COCO ground truth masks. In most cases, the misclassification occurs very close to object boundaries or in regions where the ground truth polygons are imprecise. The agreement is around 95% when more accurate LVIS ground truth masks are used. Our experiments show no significant drop in performance if point labels are simulated with 95% accuracy. In real world scenarios, such errors are usually fixed with a verification step [33, 17, 26].

3.2. Training with points

The key advantage of the new point-based annotation is its ability to seamlessly supervise off-the-shelf instance segmentation models that make mask predictions on a regular grid (*e.g.*, a 28×28 RoI prediction of Mask R-CNN [19] or an image-level per-pixel prediction of CondInst [52]). In a fully-supervised setting, these models are trained by extracting a matching regular grid of labels from ground truth masks. In contrast, with point supervision, we approximate predictions in the locations of ground truth points from the prediction on the grid using bilinear interpolation (see Figure 2). Once we have predictions and ground truth labels at the same points, a loss can be applied in the same way as with full supervision and its gradients will be propagated through bilinear interpolation. Note, that such supervision does not require any changes to the architecture. In our experiments we use cross-entropy loss on points, however other losses like dice loss [39] can be used as well.

For region-based models, some ground truth points may lay outside of predicted boxes and we choose to ignore such points during training. Contrarily, for models that yield image-level masks, additional background points can be sampled from the outside of ground truth boxes. However, in this paper we study the most basic setup where only N annotated ground truth points are used to train all models.

Data augmentation during training is a crucial component of modern segmentation models. The new point-based annotation is compatible with all common augmentations like scale jitter, crop, or horizontal flip of an input image. In our experiments, we observe that the gap in performance between point and full mask supervision is larger for longer training (3× schedule [56]) and models with higher-capacity backbones (*e.g*., ResNeXt-101 [58]). We hypothesize that this is caused by a reduced variability of training data when only a few points are available and propose a extremely simple *point-based* data augmentation strategy: instead of using all available ground truth points for a box at each iteration, we subsample half of all available points at every training iteration (5 points for $\mathcal{P}_{10}$ ground truth). In the next section we show that our augmentation strategy improves performance for higher-capacity models.

3.3. Experiments with point supervision

In this section we first ablate the design of the new annotation form on the COCO dataset [33] using Mask R-CNN [19]. Then, we demonstrate the effectiveness of the point-based supervision across 4 different datasets and show that it is applicable to a diverse set of instance segmentation models. Finally, we explore the annotation time and performance trade-off of the new annotation form.

Implementation details. We use Mask R-CNN [19] with a ResNet-50-FPN [20, 32] backbone and the default training schedules and parameters from Detectron2 [56] for each dataset (3× schedule is used for COCO [33]). Apart from the mask branch loss, there are no other differences between full mask and the new point supervision in our experiments. Unless stated otherwise, we do not use point-based data augmentations described in section 3.2.

Datasets. The main dataset in our experiments is COCO [33] which has 118k training and 5k validation images with 80 common categories. We also conduct experiments using: PASCAL VOC dataset [12] which is smaller than COCO with ∼10k images and 20 categories; Cityscapes [11] ego-centric street-scene dataset that has 2975 train and 500 validation high-resolution images with high-quality instance-level annotations for 9 categories; LVISv1.0 [17] that uses the same set of images as COCO but has more than 1000 categories annotated in a federated fashion. We use the standard evaluation metrics: AP50 [12] for PASCAL VOC and AP [33] for all other datasets.

For all datasets in our experiments, we simulate point-based ground truth by randomly sampling pixels inside each ground truth bounding box and selecting their labels based on the corresponding ground truth mask.

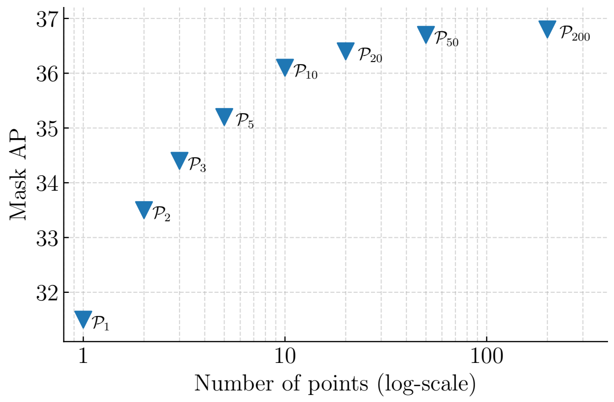


Figure 3. **Training with a different number of points.** Mask R-CNN [19] with a ResNet-50-FPN backbone trained on COCO with as few as 10 labeled points per instance ($\mathcal{P}_{10}$) achieves 36.1 mask AP with diminishing returns from more labeled points.

3.3.1 Ablation of the annotation design

Number of points. In Figure 3 we demonstrate the performance of a Mask R-CNN model trained with a varying number of labeled points per instance. The performance rapidly improves with the number of annotated points going up to tens with diminishing returns thereafter. While 20 points ($\mathcal{P}_{20}$) is approximately 0.3 AP better than 10 point supervision ($\mathcal{P}_{10}$), it takes twice as long to annotate. In what follows, we use the 10 points supervision which is ∼5 times faster than polygon-based mask annotation.

Sensitivity to point locations. We generate 5 different simulated versions of $\mathcal{P}_{10}$ ground truth annotation for the COCO `train2017` set using different random seeds for the uniform point sampling procedure. Training the same Mask R-CNN [19] model with these dataset versions, we observe very low 0.1 AP variation on the COCO `val2017` set. This result suggests that the new point-based annotation form is very robust with respect to the locations of sampled points. In what follows we conduct experiments with a single set of sampled points per dataset.

Sensitivity to the annotation quality. We study the sensitivity to the quality of point-based annotation by changing correct object/background labels for random 5% of points in a simulated ground truth $\mathcal{P}_{10}$ point annotation for COCO. The performance of a Mask R-CNN model trained on the tampered data is only 0.2 AP worse than the model trained on the clean data (35.9 AP *vs*. 36.1 AP). In a more challenging setup, instead of random points, we set wrong labels to 5% of points that are closest to object boundaries. In this case, the drop in performance is slightly larger – 0.4 AP (35.7 AP *vs*. 36.1 AP). Our experiment shows that the new point-based annotation does not require perfectly accurate point labeling procedure which reduces the need for additional verification steps in a real-world annotation pipeline.

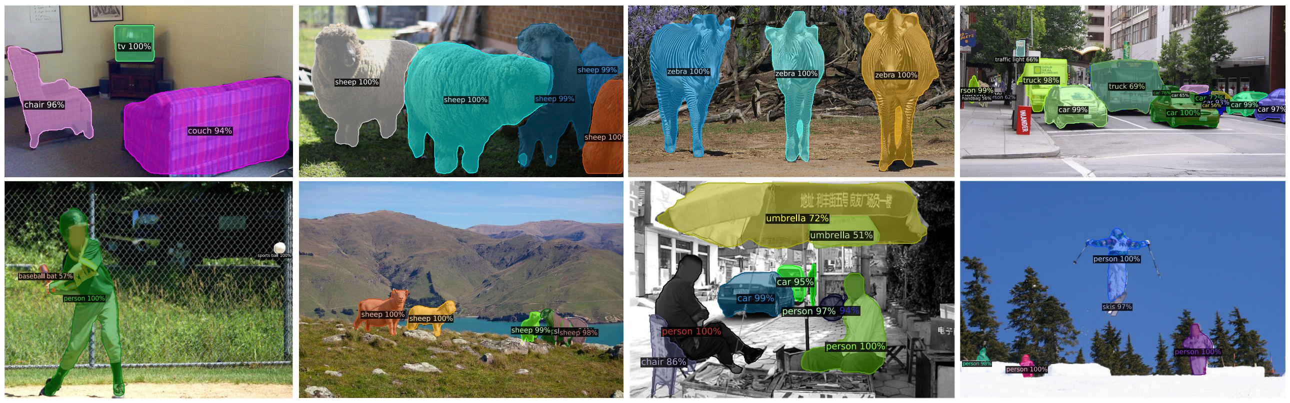


Figure 4. **Mask R-CNN trained with 10 points per object** ($\mathcal{P}_{10}$) on COCO. The model uses a ResNet-50-FPN [19, 32] backbone and is trained with 3× schedule [56] (36.1 AP). We show predictions with confidence scores greater than 0.5.

supervision	AP
$\mathcal{M}$	37.2
$\mathcal{P}_{10}$	36.1 (97%)

(a) COCO `val2017` [33].

supervision	AP_{50}
$\mathcal{M}$	66.3
$\mathcal{P}_{10}$	64.2 (97%)

(b) PASCAL VOC `val` [12].

supervision	AP
$\mathcal{M}$	32.7
$\mathcal{P}_{10}$	30.7 (94%)

(c) Cityscapes `val` [11].

supervision	AP
$\mathcal{M}$	22.8
$\mathcal{P}_{10}$	21.5 (94%)

(d) LVISv1.0 `val` [17].

Table 1. **Mask R-CNN with full mask ($\mathcal{M}$) *vs.* new point supervision ($\mathcal{P}_{10}$).** For each object only 10 labeled points are used. For all four datasets with different properties, pointly-supervised models are within 2 AP (94% – 97%) from their fully-supervised counterparts.

supervision	point aug.	R50 AP	R101 AP	X101 AP
$\mathcal{M}$	-	37.2	38.6	39.5
$\mathcal{P}_{10}$		36.1 (97%)	37.8 (98%)	38.1 (96%)
	✓	36.0 (97%)	37.8 (98%)	38.5 (97%)

Table 2. **Mask R-CNN with higher-capacity backbones and point-based augmentation** on COCO. We use ResNet-50 (R50), ResNet-101 (R101), and ResNeXt101-32×8 (X101) backbones [20, 58] with FPN [32]. The simple point-based data augmentation (*point aug.*) proposed in section 3.2 improves performance (+0.4 AP) for the higher-capacity X101 model.

supervis.	AP
$\mathcal{M}$	37.2
$\mathcal{P}_{10}$	36.1 (97%)

(a) Mask R-CNN [19].

supervis.	AP
$\mathcal{M}$	37.5
$\mathcal{P}^{*}_{10}$	35.7 (95%)

(b) CondInst [52].

supervis.	AP
$\mathcal{M}$	38.3
$\mathcal{P}^{*}_{10}$	35.7 (93%)

(c) PointRend [25].

Table 3. **Different models trained with point supervision** on COCO. All models use a ResNet-50-FPN backbone [20, 32]. "$*$": We train CondInst and PointRend models with point-based augmentation which improves their performance. While all three models achieve similar performance with point supervision, we observe that the gap between mask and point supervision is the largest for PointRend. We explore PointRend performance and propose an improved version of the module in the next section.

3.3.2 Main results

In Table 1 we compare 10 points supervision ($\mathcal{P}_{10}$) to full mask supervision ($\mathcal{M}$) for a Mask R-CNN model with a ResNet-50-FPN [20, 32] backbone on four different datasets. Without any modifications to either the architecture or the default training algorithm/hyper-parameters set in Detectron2 [56], Mask R-CNN trained with $\mathcal{P}_{10}$ achieves 94% – 97% of the fully-supervised ($\mathcal{M}$) model performance. This result shows that the new point supervision is widely applicable to datasets at different scales (*e.g.*, from 20 categories in PASCAL VOC [12] to more than 1,000 categories in LVISv1.0 [17]) and in different domains (common objects of COCO [33] or street scenes of Cityscapes [11]). In Figure 4, we visualize predictions of a Mask R-CNN model supervised with points on COCO.

Higher-capacity backbones and point-based data augmentation. In Table 2 we compare full and 10 points supervision for Mask R-CNN with different backbones on COCO. The simple point-based augmentation strategy described in section 3.2 does not significantly change the performance of the smaller model. However, for higher-capacity ResNeXt-based model, the point-based augmentation effectively improves performance by 0.4 AP.

Different models. The new point-based supervision can be applied to a diverse set of models that use a per-pixel mask loss. In our experiments we use two methods in addition to Mask R-CNN: CondInst [52] that makes image-level predictions without an RoI pooling operation[2] and PointRend [25] that makes point-level prediction to iteratively achieve a high resolution. We use the same ResNet-50-FPN [20, 32] backbone for all models and report results on COCO. Unlike Mask R-CNN, both CondInst and PointRend benefit from point-based augmentation even

[2]As described in section 3.2, for CondInst [52] we use per-point cross entropy for given ground truth points and do not supervise predictions outside of the ground truth boxes. Therefore, we need to filter out any mask prediction outside of predicted boxes. A more sophisticated point selection strategy can be used to avoid it, however, this is not the focus of our paper.

with a smaller backbone; thus, we report their results using the augmentation. In Table 3 we observe that all three models achieve similar performance with point supervision. However, the gap between mask and point supervision for PointRend is significantly larger. The coarse mask head in PointRend makes prediction with low 7×7 resolution making direct point-based training inaccurate. Inspired by this observation, in section 4 we present a new streamlined version of PointRend module with a single mask loss which can be trained with either mask or point supervision.

Self-training with points. The gap between point and full mask supervision can be further decreased using self-training paradigm [49, 60, 47, 57, 7, 63]. After training a ResNet-50-FPN based Mask R-CNN with 10 points supervision on COCO (36.1 AP), we collect pseudo-ground truth masks by running inference on COCO train data with ground truth boxes. Mask R-CNN trained on these pseudo-ground truth masks without any changes in the training recipe, achieves 36.7 AP (+0.6 AP) or 98% of fully-supervised Mask R-CNN trained with 6× schedule that matches the training time of our self-training setup.

3.3.3 Annotation time and performance trade-off.

We compare the new point-based supervision with other forms of supervision for instance segmentation under *the same annotation budget* which we measure as the time required to label training data. For this comparison we use identical Mask R-CNN [19] models for full mask and point supervision. Whereas, for bounding box supervision we use BoxInst [53], a recently proposed instance segmentation model that shows the best performance on COCO among existing weakly-supervised methods with bounding box supervision. Note, that BoxInst is based on CondInst [52] which performs on par or better than Mask R-CNN with full mask and point supervision. For all models we use the standard ResNet-50-FPN backbone [20, 32].

COCO annotation timings. An annotation collection for instance segmentation is a multi-stage process. For example, in the COCO annotation pipeline [33], annotators first spot and categorize objects by pointing at them, and then the spotted objects are annotated with polygon-based masks. As discussed in section 3.1, for a spotted object in COCO, it takes on average 7 seconds to annotate its bounding box [41], 16 seconds to annotate the box and 10 points inside, and 79.2 seconds for polygon-based mask annotation [33]. For each annotation form we approximate the total time to label COCO as the time required for the categorization and spotting stages (reported by COCO creators [32], see the appendix) plus the time required to annotate all spotted objects with the corresponding annotation form. In Figure 5 we match the annotation times between different supervision forms by training a Mask R-CNN model using from 10% to 100% of COCO `train2017` data. For BoxInst [52] we use performance reported by its authors. Our analysis shows that under the same annotation budget, Mask R-CNN trained with points significantly outperforms models trained with the other supervision forms.

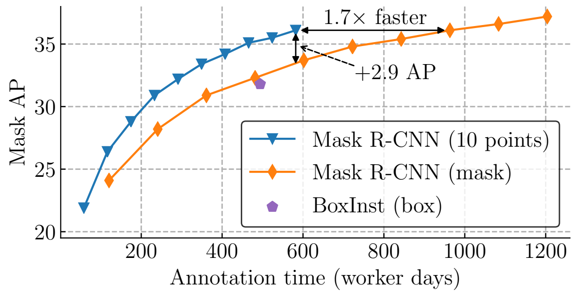


Figure 5. **AP *vs.* annotation time for different supervision types** on COCO `val2017`. To match annotation times between different supervision forms, we train a Mask R-CNN model using from 10% to 100% of COCO `train2017`. Observe that Mask R-CNN trained with the new point-based supervision significantly outperforms models trained with both full mask supervision and weak bounding box supervision under the same computation budget.

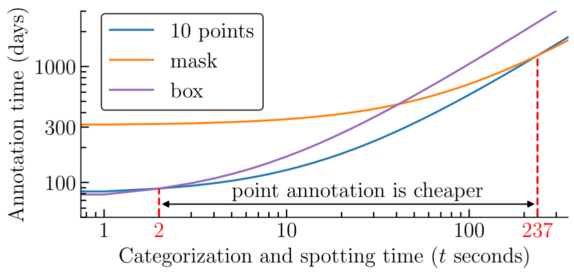


Figure 6. **Total annotation time to achieve 31.8 AP on COCO** depending on the speed of categorization and spotting stages. BoxInst [53] trained with boxes achieves this performance on full COCO `train2017`. Whereas, Mask R-CNN needs only ∼40% and ∼50% of the train set with mask and points supervision respectively. Using this data, we estimate that point-based annotation is more efficient for labeling pipelines where the spotting and categorization stages take between 2 and 237 seconds per instance.

Estimation for various annotation pipelines. Different annotation collection pipelines for instance segmentation use different protocols for object spotting and categorization stages [33, 17, 26]. Note, that time t spent on these stages is the same no matter what annotation form for instances is used. For the original COCO annotation pipeline, t is 43.2 seconds per instance, thanks to multiple additional verification steps [33]. In Figure 6 we compute total annotation time needed to achieve 31.8 AP on COCO[3] with different types of supervision for any $t \geq 0$. We observe, that the point-based annotation is more efficient for labeling pipelines where the spotting and categorization stages take between 2 and 237 seconds per instance on average.

[3]BoxInst [53] performance with full COCO `train2017` set.

4. Implicit PointRend Model

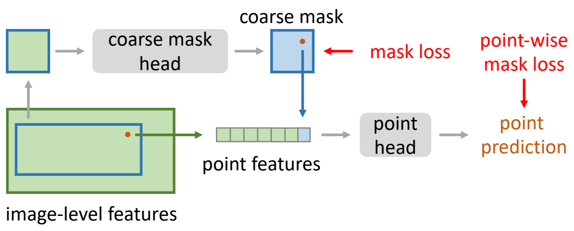
Intuitively, the new point-based supervision should suite well to the PointRend module [25] that yields a mask for a detected object using point-wise predictions. However, as we observed in Table 3, the gap between full mask and point supervision is larger for PointRend than for the other studied methods. While PointRend-based model trained with full mask supervision outperforms Mask R-CNN [19], it performs worse than the baseline with point supervision. We hypothesize that the gap in performance is due to the coarse mask head of the standard PointRend module that outputs a 7×7 mask prediction and has its own mask loss. With such low resolution if two ground truth points are close to each other but have different labels, then this head does not get a reliable training signal.

Inspired by this observation, we propose a simplified version of the PointRend module which we name Implicit PointRend. For each detected object, instead of a coarse mask prediction, the new architecture predicts parameters for a point head function that can make a point-wise object mask prediction for any point given its position and corresponding image features. The new module has a single point-level mask loss which simplifies its implementation in comparison with PointRend. In what follows we describe the new design in detail and compare it with PointRend using both full mask and point supervision.

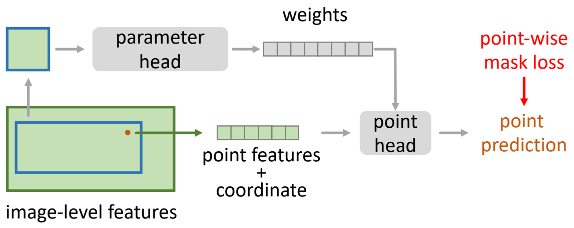
4.1. Point-wise representation and point head

PointRend [25] constructs feature representation for a point concatenating two feature types: (1) a fine-grained point representation extracted from an image-level feature map (e.g. an FPN [32] level feature map) in the exact position of the point via bilinear interpolation and (2) a coarse mask prediction for the point which provides region-specific information. The coarse mask prediction is made by a separate head that takes in RoI features and return a low resolution (7×7) mask prediction. Given this point representation, a small MLP network, called point head, makes mask prediction for the exact point location. Note, that in the PointRend model, the parameters of the point head are shared across all points and detected instances.

Implicit PointRend uses the point head to make point-wise mask predictions as well. However, instead of relying on coarse mask predictions to distinguish different instances, the new model generates different parameters of the point head for each instance similarly to the dynamic head approach of CondInst [52]. The parameters of the point head implicitly represent the mask of an objects and resembles implicit functions used by 3D community [22, 36, 16]. To make a mask prediction the point head takes in the coordinate of the point relative to the bounding box it belongs to. In addition, we use the same fine-grained point features as in



(a) PointRend [25]



(b) Implicit PointRend

Figure 7. **PointRend [25] *vs*. Implicit PointRend architectures.** Instead of a coarse mask prediction used in PointRend to provide region-level context to distinguish objects, Implicit PointRend generates different parameters of the point head for each detected object. The point head makes prediction at any location independently taking in fine-grained point features and the information about the coordinate of this location relative to the bounding box. The same subdivision mask rendering algorithm [25] is used in both to output high resolution mask with point-wise predictions.

the original PointRend design. Note, that the new Implicit PointRend module does not require a coarse mask prediction and, therefore, can be trained with a single mask loss applied to the output of the point head. We compare overall design of the new module with PointRend in Figure 7.

4.2. Point selection for inference and training

Inference. Implicit PointRend follows the same point selection strategy as PointRend during inference, *i.e.* adaptive subdivision [55]. We start from predictions generated by our point head using a 28×28 uniform grid of points. Then, we gradually upsample it by a factor of 2 to 224×224 resolution. In each subdivision step, we select N most uncertain points after interpolation and replace them with the predictions from the point head at these locations. Following the original PointRend implementation, we set N to 28^2.

Mask supervision ($\mathcal{M}$) training. During training, PointRend selects more points from the uncertain regions of the coarse mask prediction via an importance sampling strategy. In our experiments we find that the Implicit PointRend model performs on par with PointRend while using a much simpler uniform point sampling strategy.

Point supervision ($\mathcal{P}_{10}$) training. Implicit PointRend is naturally suited for point supervision, where we simply select points with ground truth annotation. Similarly to Mask R-CNN, we ignore points outside predicted boxes.

4.3. Implementation details

We set all hyper-parameters following the original PointRend setup [25]. The point head is an MLP with 3 hidden layers of 256 channels, ReLU activations [40] in hidden layers and the sigmoid activation applied to its output. To make a fair comparison, the head that generates parameters for the point head has the same architecture as the coarse mask head of PointRend [25]. We observe however that Implicit PointRend performs as well with a smaller parameter head such as the standard box head architecture [45]. We expect that a better design can bring further improvement.

As the input for the point head we use fine-grained features extracted from `p2` level of FPN [32] that has 256 channel dimension. We use the random Fourier positional encoding [51] to represent the point (x, y) location relative to the center of the bounding box. We found that the model that uses positional encoding performs better than the one that uses (x, y) coordinates directly. Please see the appendix for more detail and ablation experiments.

Implicit PointRend uses a single per-point mask loss (binary cross entropy) applied to the outputs of the point head. In addition, following the common setup for implicit function models [51], we add l_2 loss on predicted parameters of the point head to avoid predicted parameters become unbounded. This loss plays the role of weight decay which is otherwise absent for the dynamic parameters. In our experiments, we set the l_2 loss weight to 1e-5.

4.4. Experimental evaluation

To evaluate Implicit PointRend, we use COCO [33] and 3$\times$ schedule [56] to avoid underfitting. All hyper-parameters for the Implicit PointRend module are the same as in PointRend [25]. In all experiments we attach the modules as a mask head to Faster R-CNN [45]. Unless specified, we use a ResNet-50-FPN [20, 32] backbone.

In our experiments, we observe that Implicit PointRend performance significantly increases even for smaller models if the point sub-sampling augmentation described in section 3.2 is used (see the appendix for an ablation). We did not observe the same behavior with the standard PointRend module and hypothesize, that the implicit representation of an object mask is more prone to overfitting than the coarse mask-based representation due to its higher capacity. To make a fair comparison, in this section we use the point-based augmentation for all models supervised with points.

Main results. We compare the new Implicit PointRend and PointRend in Table 4 with both full mask and point supervision. Unlike PointRend, our new module does not use any importance point sampling strategy and has only one mask loss. The more straightforward Implicit PointRend module performs on par with PointRend trained with full mask supervision (AP ($\mathcal{M}$)). Moreover, the new module significantly outperforms the baseline in case of point supervision (AP ($\mathcal{P}_{10}$)). This observation underscores unique challenges of the point supervision and suggests that new model designs may be needed to fully uncover the potential of the point-based annotation form.

method	AP ($\mathcal{M}$)	AP ($\mathcal{P}_{10}$)
PointRend [25]	38.3	35.7
Implicit PointRend	**38.5** (+0.2)	**36.9** (+1.2)

Table 4. **PointRend *vs.* Implicit PointRend** with mask ($\mathcal{M}$) and our point supervision ($\mathcal{P}_{10}$) on COCO `val2017`. A ResNet-50-FPN [20, 32] backbone is used for both models. While Implicit PointRend performs on par with the standard PointRend with the full mask supervision (AP ($\mathcal{M}$)), it significantly outperforms the baseline in case of the point supervision (AP ($\mathcal{P}_{10}$)).

method	supervision	R50 AP	R101 AP	X101 AP
Mask R-CNN	$\mathcal{M}$	**37.2**	**38.6**	39.5
Mask R-CNN	$\mathcal{P}_{10}$	36.0 (-1.2)	37.8 (-0.8)	38.5 (-1.0)
Implicit PointRend	$\mathcal{P}_{10}$	36.9 (-0.3)	38.5 (-0.1)	**39.7** (+0.2)

Table 5. Implicit PointRend performance with 10 points supervision achieves the performance of the standard Mask R-CNN model with full mask supervision on COCO `train2017`. We use R50, R101, and X101 backbones [20, 58] with FPN [32].

Similar to the other methods studied in section 3.3, Implicit PointRend-based model supervised with points achieves 96% of its full mask supervision performance. In Table 5 we compare Implicit PointRend trained with points and the standard Mask R-CNN using the backbones of different capacities. As expected from a stronger method, Implicit PointRend significantly outperforms the standard Mask R-CNN when both trained with points. Moreover, we find that Implicit PointRend trained with only 10 labeled points per objects achieves the performance of the fully-supervised Mask R-CNN. Given that Mask R-CNN is still actively used in many real-world applications, this result suggests broad applicability of the new annotation form in scenarios with a constrained annotation budget.

In addition, in the appendix we report Implicit PointRend performance on COCO with mask supervision.

5. Conclusion

We present the new point-based supervision for instance segmentation which requires only a bounding box and several points annotated per instance. Unlike many other weak annotation forms, point-based supervision can be seamlessly applied to existing instance segmentation models without any modification to their architecture or training algorithm. The new point-based supervision provides the best trade-off between annotation time and accuracy among different supervision forms for instance segmentation. We further propose a simple but effective module named Implicit PointRend which tackles the unique challenges of point supervision with implicit mask representation.

Acknowledgments. We would like to thank Ross Girshick, Yuxin Wu, Piotr Dollár, Alex Berg, Tamara Berg, and Elisa Berger for useful discussions and advices.

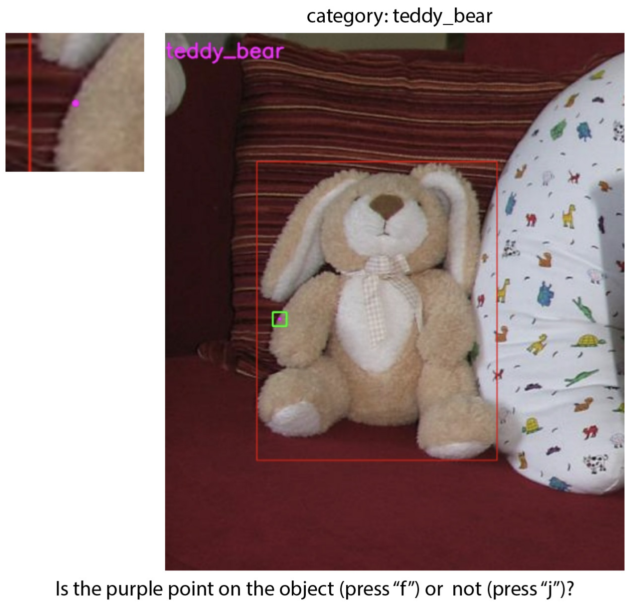
Appendix

We first provide more details for the point classification annotation tool in section A. In section B, we provide additional ablation experiments analyzing overfitting with point-based supervision as well as a more detailed study of the proposed point-based data augmentation. Section C discusses annotation time for COCO dataset [33] with different types of supervision. Finally, we conduct a thorough analysis of Implicit PointRend in section D.

A. Annotation Pipeline

To estimate the speed and quality of the point annotation we developed a simple labeling tool. The screenshot of its interface is shown in Figure 8. An annotator is presented with randomly sampled points one by one for each object. Our tool shows two views for a point. The first view contains the whole object together with its bounding box, category, and a point marker. The view is centered around the object bounding box with additional margins for context. Apart from the location of the point marker, this view does not change between different points. Note, that the point marker can be small and hard to spot for large objects. To make it more visible we add a green box around the point in this view. The second view shows zoomed in area centered around the point to help classify harder cases. Such two views system allows an annotator to classify points without the need to zoom in on them manually. Our experiments with COCO data show that a trained annotator labels a point in less than a second (0.8 – 0.9 seconds on average).

We estimate the quality of our point annotation by checking its labels against ground truth masks. We observed $\sim$90% agreement between points and instance masks on COCO. Upon closer analysis, we find that most of the errors are due to inaccurate boundaries in COCO polygon-based annotation (see Figure 9).


category: teddy_bear

Is the purple point on the object (press "f") or not (press "j")?

Figure 8. **Screenshot of the point annotation tool.** For each object, points are presented one by one. The tool uses two views of each point to simplify the task: (right) view shows the whole object annotated with a bounding box and a category. The point is depicted as a purple circle surrounded with a green box to simplify its spotting; For small objects, this view is cropped around the bounding box with margins for context; (left) view shows zoomed in patch of the image centered around the point to classify. This view helps to correctly classify points that are close to boundaries.

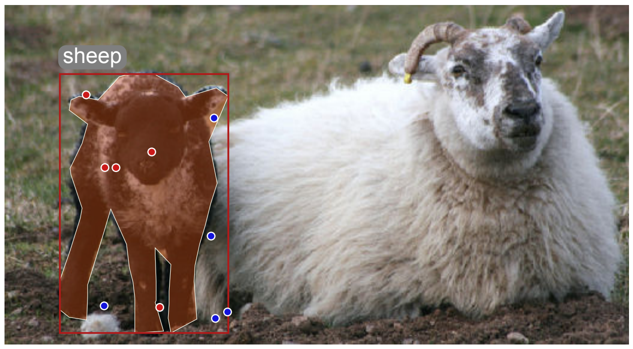


Figure 9. **Point annotation *vs*. polygon-based mask on COCO.** Red points were annotated as object and blue as background. Note, that 3 points (two near both ears and one between the front legs) have correct labels but do not match to the polygon-based mask annotation. We observe that on COCO most of disagreement between point labels and ground truth masks have similar nature.

B. Overfitting and Point-based Augmentation

Overfitting with longer training schedules. In our experiments with the COCO dataset [32], we observe that the gap between Mask R-CNN models [19] trained with full mask and point-based (10 points) supervision increases for longer training schedules. For example, for a ResNet-50-FPN [20, 32] backbone, the gap is 0.9 AP (35.2 AP *vs*. 34.3 AP) for 1$\times$ schedule, but grows to 1.1 AP (37.2 AP *vs*. 36.1 AP) for longer 3$\times$ schedule. We hypothesize that this is caused by a reduced variability of training data when only a few points are available and propose a simple *point-based* data augmentation strategy to counter the effect. Note, that in our paper we train all COCO models with 3$\times$ schedule.

Point-based data augmentation analysis. The proposed point-based augmentation randomly samples half of the points for a box at each iteration instead of using all of them. We vary the number of sampled points (with 1, 3, 5, 7, or 9 points) for 10 points ground truth and find that taking only 1 or 3 points is too aggressive while the results are similar when sampling 5, 7, or 9 points.

C. COCO Annotation Time

The annotation protocol of the COCO dataset [33] has 3 stages: (1) category labeling, (2) instance spotting and (3) mask annotation. The creators of the COCO dataset report detailed timing of each stage on all annotated images. Here, we summarize the annotation time *per instance* for different types of supervision.

Category labeling and instance spotting. Category labeling is the task of determining which object categories are present in each image. This step takes around 7,000 hours to annotate 118,287 images with 849,949 instances (`train2017` set), suggesting the time of category labeling is 28.8 seconds per instance. The instance spotting stage places a cross on top of each instance. This step takes 3,500 hours to annotate 118,287 images with 849,949 instances. Thus, instance spotting takes 14.4 seconds per instance. Note that, in order to achieve a high recall, both category labeling and instance spotting are performed with 8 workers for every image and the total annotation time mentioned before is the sum of 8 workers.

Bounding box annotation. Since the original COCO protocol does not annotate instances by bounding boxes, we approximate the time of bounding box annotation ($\mathcal{B}$) as 7 seconds for a spotted instance which can be achieved with extreme clicks technique [41]. Taken into account category labeling and instance spotting, bounding box annotation ($\mathcal{B}$) takes 28.8 (category labeling) + 14.4 (instance spotting) + 7.0 (box) = 50.2 seconds per instance.

Mask annotation. Polygon-based mask annoation in COCO takes 22 hours per 1,000 instances or $\sim$79.2 seconds per instance. Together with category labeling and instance spotting, mask annotation ($\mathcal{M}$) takes 28.8 (category labeling) + 14.4 (instance spotting) + 79.2 (mask) = 122.4 seconds per instance.

Our point-based annotation. Given the bounding box, it takes annotator 0.8 – 0.9 seconds on average to provide the binary label for each point. Thus, our point-based annotation with 10 points ($\mathcal{P}_{10}$) takes 50.2 (bounding box annotation) + 0.9 $\times$ 10 (10 points per instance) = 59.2 seconds per instance which is more than 2 times faster than mask annotation if the time of categorization and spotting stages is included. For datasets with bounding box annotations [50, 26], our point annotations takes only 0.9×10(10 points per instance) = 9 seconds per instance which is more than 8 times faster than the polygon-based mask annotation (79.2 seconds per instance). In Table 6, we report the annotation time for different annoation format on COCO `train2017` set, which contains 118,287 images and 849,949 instances.

supervision	notation	COCO annotation time (days)
bounding box	$\mathcal{B}$	493
mask	$\mathcal{M}$	1204
our point-based	$\mathcal{P}_{10}$	582

Table 6. **Annotation times for different types of supervision** on COCO `train2017`. We report the time to annotate 118,287 images with 849,949 instances. $\mathcal{B}$: bounding box supervision, $\mathcal{M}$: mask supervision, and $\mathcal{P}_{10}$: our point-based supervision with a bounding box and 10 points labels per instance.

point aug.	Mask R-CNN [19]	PointRend [25]	Implicit PointRend
	36.1	35.6	36.0
✓	36.0 (-0.1)	35.7 (+0.1)	36.9 (+0.9)

Table 7. **Point-based augmentation for various models** with a ResNet-50-FPN [20, 32] backbone. All model are trained with 10 points supervision on COCO `train2017` and mask AP on `val2017` is reported. The new augmentation is more effective for Implicit PointRend as the new module has higher capacity in representing object masks with its parameter head.

D. Implicit PointRend

In this section, we ablate the design of the Implicit PointRend module and report its performance with full mask supervision for reference.

D.1. Ablation study

For the ablation experiments we use a ResNet-50 [20] backbone with FPN [32], 3$\times$ schedule and point-based data augmentation.

Point-based data augmentation. We observe that point-based data augmentation does not significantly improve performance for Mask R-CNN [19] and PointRend [25] with a ResNet-50-FPN backbone (see Table 7). Whereas, Implicit PointRend shows 0.9 AP gain with the augmentation. We hypothesize that the augmentation is more effective for Implicit PointRend as the new module has a higher capacity in representing object masks with its parameter head.

Point-level feature representation. The point head of Implicit PointRend takes two types of features as input: point coordinate relative to the bounding box and image-level point features. Next, we ablate both components.

In Table 8a, we show that adding the relative coordinate w.r.t. the box is already effective, giving an improvement of 1.4 AP with point-based supervision and 0.8 AP with mask supervision. Encoding the coordinate with a positional encoding [51] further improves the mask prediction by 0.3 AP for point-based supervision.

Table 8b shows that Implicit PointRend can achieve reasonable performance with only the positional encoding as the input to its point head. Image-level point features from the `p2` level of FPN [32] backbone further improve the performance by 1.6 AP with point-based supervision and 1.4

coord.	AP ($\mathcal{P}_{10}$)	AP ($\mathcal{M}$)
none	35.2	37.7
rel.	36.6 (+1.4)	38.5 (+0.8)
p.e.	36.9 (+1.7)	38.5 (+0.8)

(a) **Coordinates.**

features	AP ($\mathcal{P}_{10}$)	AP ($\mathcal{M}$)
none	35.3	37.1
`p2`	36.9 (+1.6)	38.5 (+1.4)

(b) **Image-level features.**

Table 8. **Implicit PointRend ablations** on COCO `train2017` with ResNet-50-FPN [20, 32] backbone. AP ($\mathcal{P}_{10}$) is mask AP for point-based supervision and AP ($\mathcal{M}$) is mask AP for full mask supervision. Table 8a: Implicit PointRend utilizes the relative coordinate w.r.t. the box point-level information (*rel.*) to improve instance segmentation performance. Positional encoding representation for the coordinate (*p.e.*) [51] further improve results. Table 8b: Implicit PointRend can achieve reasonable performance with only the positional encoding as the input to its point head (*none*). Image-level point features from the `p2` level of FPN [32] backbone (*p2*) further improve the performance

AP with mask supervision. These experiments suggest that both coordinate and image-level features are essential for the overall performance of Implicit PointRend.

D.2. Mask supervision results

We compare Implicit PointRend with several instance segmentation approaches on the COCO dataset [33] with full mask supervision. The module design is motivated by our point-based supervision, however, Table 9 shows that Implicit PointRend is also a competitive method for fully-supervised instance segmentation as well. While both CondInst [52] and Implicit PointRend use an instance-dependent function to predict masks, Implicit PointRend outperforms CondInst by a large margin. Implicit PointRend is able to match the performance of PointRend [25] without the coarse mask prediction and importance sampling procedure during training. We note that the small gap (less than 0.3 AP) between Implicit PointRend and PointRend mostly comes from large objects. We hypothesize that the fixed-size feature map (*e.g.*, 14 × 14) extracted from `p2` FPN level is too coarse to generate point head parameters to accurately segment large objects. A simple fix could be dynamically choosing the FPN levels to pool feature like the box head [32]. In our experiments such design gave a small 0.1-0.2 AP boost for Implicit PointRend results. We expect a better design for the parameter head can further improve Implicit PointRend performance.

	backb.	LRS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [19]	R50	1×	35.2	56.3	37.5	17.2	37.7	50.3
	R50	3×	37.2	58.6	39.9	18.6	39.5	53.3
	R101	3×	38.6	60.4	41.3	19.5	41.3	55.3
	X101	3×	39.5	61.7	42.6	20.7	42.0	56.5
CondInst [52]	R50	1×	35.7	-	-	-	-	-
	R50	3×	37.5	-	-	-	-	-
	R101	3×	38.6	-	-	-	-	-
PointRend [25]	R50	1×	36.2	56.6	38.6	17.1	38.8	52.5
	R50	3×	38.3	59.1	41.1	19.1	40.7	55.8
	R101	3×	40.1	61.1	43.0	20.0	42.9	58.6
	X101	3×	41.1	62.8	44.2	21.5	43.8	59.1
Implicit PointRend	R50	1×	36.9	57.3	39.6	17.7	39.4	53.1
	R50	3×	38.5	59.4	41.4	18.9	41.1	55.0
	R101	3×	39.9	61.1	43.0	20.3	42.7	58.0
	X101	3×	40.8	62.6	43.9	21.5	43.5	57.3

Table 9. **Instance segmentation results with full mask supervision** on COCO `val2017`. LRS: learning rate schedule; a 1× learning rate schedule refers to 90,000 iterations. R50,101: ResNet-50,101 [20]. X101: ResNext-101 32 × 8d [58]. All models use FPN [32]. The proposed Implicit PointRend also performs better than Mask R-CNN [19] and comparable to PointRend [25] under full mask supervision.

References

[1] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. Weakly supervised learning of instance segmentation with inter-pixel relations. In *CVPR*, 2019. 1, 2

[2] Aditya Arun, CV Jawahar, and M Pawan Kumar. Weakly supervised instance segmentation by learning annotation consistent instances. In *ECCV*, 2020. 1, 2

[3] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017. 2

[4] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision. In *ECCV*, 2016. 1, 2, 3

[5] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *CVPR*, 2019. 2

[6] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. YOLACT++: Better real-time instance segmentation. *PAMI*, 2020. 2

[7] Liang-Chieh Chen, Raphael Gontijo Lopes, Bowen Cheng, Maxwell D Collins, Ekin D Cubuk, Barret Zoph, Hartwig Adam, and Jonathon Shlens. Naive-student: Leveraging semi-supervised learning in video sequences for urban scene segmentation. In *ECCV*, 2020. 6

[8] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020. 2

[9] Hisham Cholakkal, Guolei Sun, Fahad Shahbaz Khan, and Ling Shao. Object counting and instance segmentation with image-level supervision. In *CVPR*, 2019. 1

[10] Herbert H Clark. Coordinating with each other in a material world. *Discourse studies*, 2005. 3

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes

dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 4, 5

[12] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes challenge: A retrospective. *IJCV*, 2015. 2, 4, 5

[13] Ruochen Fan, Qibin Hou, Ming-Ming Cheng, Gang Yu, Ralph R Martin, and Shi-Min Hu. Associating inter-image salient instances for weakly supervised semantic segmentation. In *ECCV*, 2018. 1

[14] Chaz Firestone and Brian J Scholl. “please tap the shape, anywhere you like” shape skeletons in human vision revealed by an exceedingly simple measure. *Psychological science*, 2014. 3

[15] Weifeng Ge, Sheng Guo, Weilin Huang, and Matthew R Scott. Label-PEnet: Sequential label propagation and enhancement networks for weakly supervised instance segmentation. In *ICCV*, 2019. 1

[16] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3D shape. In *CVPR*, 2020. 7

[17] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *ICCV*, 2019. 2, 3, 4, 5, 6

[18] Michael Gygli and Vittorio Ferrari. Efficient object annotation via speaking and pointing. *IJCV*, 2019. 3

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2, 3, 4, 5, 6, 7, 9, 10, 11

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4, 5, 6, 8, 9, 10, 11

[21] Cheng-Chun Hsu, Kuang-Jui Hsu, Chung-Chi Tsai, Yen-Yu Lin, and Yung-Yu Chuang. Weakly supervised instance segmentation using the bounding box tightness prior. In *NIPS*, 2019. 1, 2

[22] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3D scenes. In *CVPR*, 2020. 7

[23] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *CVPR*, 2017. 1, 2

[24] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. InstanceCut: from edges to instances with multicut. In *CVPR*, 2017. 2

[25] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. In *CVPR*, 2020. 2, 5, 7, 8, 10, 11

[26] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020. 3, 6, 10

[27] Issam H Laradji, Negar Rostamzadeh, Pedro O Pinheiro, David Vazquez, and Mark Schmidt. Where are the blobs: Counting by localization with point supervision. In *ECCV*, 2018. 2, 3

[28] Issam H Laradji, Negar Rostamzadeh, Pedro O Pinheiro, David Vazquez, and Mark Schmidt. Proposal-based instance segmentation with point supervision. In *ICIP*, 2020. 1, 2, 3

[29] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Interactive image segmentation with latent diversity. In *CVPR*, 2018. 2

[30] JunHao Liew, Yunchao Wei, Wei Xiong, Sim-Heng Ong, and Jiashi Feng. Regional interactive image segmentation networks. In *ICCV*, 2017. 2

[31] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. ScribbleSup: Scribble-supervised convolutional networks for semantic segmentation. In *CVPR*, 2016. 1, 2

[32] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2, 4, 5, 6, 7, 8, 9, 10, 11

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1, 2, 3, 4, 5, 6, 8, 9, 10, 11

[34] Yun Liu, Yu-Huan Wu, Pei-Song Wen, Yu-Jun Shi, Yu Qiu, and Ming-Ming Cheng. Leveraging instance-, image- and dataset-level information for weakly supervised instance segmentation. *PAMI*, 2020. 1

[35] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018. 1, 2

[36] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019. 7

[37] Pascal Mettes and Cees GM Snoek. Pointly-supervised action localization. *IJCV*, 2019. 2

[38] Pascal Mettes, Jan C Van Gemert, and Cees GM Snoek. Spot on: Action localization from pointly-supervised proposals. In *ECCV*, 2016. 2, 3

[39] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *International conference on 3D vision (3DV)*, 2016. 3

[40] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 8

[41] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *ICCV*, 2017. 1, 2, 3, 6, 10

[42] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Training object class detectors with click supervision. In *CVPR*, 2017. 2, 3

[43] Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *PAMI*, 2016. 2

[44] Rui Qian, Yunchao Wei, Honghui Shi, Jiachen Li, Jiaying Liu, and Thomas Huang. Weakly supervised scene parsing with point-based distance metric learning. In *AAAI*, 2019. 2, 3

[45] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 8

[46] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Alexander G. Schwing, and Jan Kautz. UFO^2: A unified framework towards omni-supervised object detection. In *ECCV*, 2020. 2

[47] Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *EMNLP*, 2003. 6

[48] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. “GrabCut” interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 2004. 2

[49] H Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 1965. 6

[50] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *ICCV*, 2019. 10

[51] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NIPS*, 2020. 8, 10, 11

[52] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020. 2, 3, 5, 6, 7, 11

[53] Zhi Tian, Chunhua Shen, Xinlong Wang, and Hao Chen. BoxInst: High-performance instance segmentation with box annotations. In *CVPR*, 2021. 1, 2, 6

[54] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013. 2

[55] Turner Whitted. An improved illumination model for shaded display. In *ACM Siggraph 2005 Courses*, 2005. 7

[56] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 4, 5, 8

[57] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *CVPR*, 2020. 6

[58] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 4, 5, 8, 11

[59] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. Deep interactive object selection. In *CVPR*, 2016. 2

[60] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, 1995. 6

[61] Yanzhao Zhou, Yi Zhu, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Weakly supervised instance segmentation using class peak response. In *CVPR*, 2018. 1, 2

[62] Yi Zhu, Yanzhao Zhou, Huijuan Xu, Qixiang Ye, David Doermann, and Jianbin Jiao. Learning instance activation maps for weakly supervised instance segmentation. In *CVPR*, 2019. 1, 2

[63] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D Cubuk, and Quoc V Le. Rethinking pre-training and self-training. In *NIPS*, 2020. 6