Supplementary Materials: Pointly-Supervised Instance Segmentation

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

[1]UIUC [2]Facebook AI

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

A. Introduction

We first provide more details for the point classification annotation tool in section B. In section C, 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 E discusses annotation time for COCO dataset [8] with different types of supervision. Finally, we conduct a thorough analysis of Implicit PointRend in section F.

B. Annotation Pipeline

To estimate the speed and quality of the point-based annotation scheme we developed a simple labeling tool. The screenshot of its interface is shown in Figure 1. 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 ∼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 2).

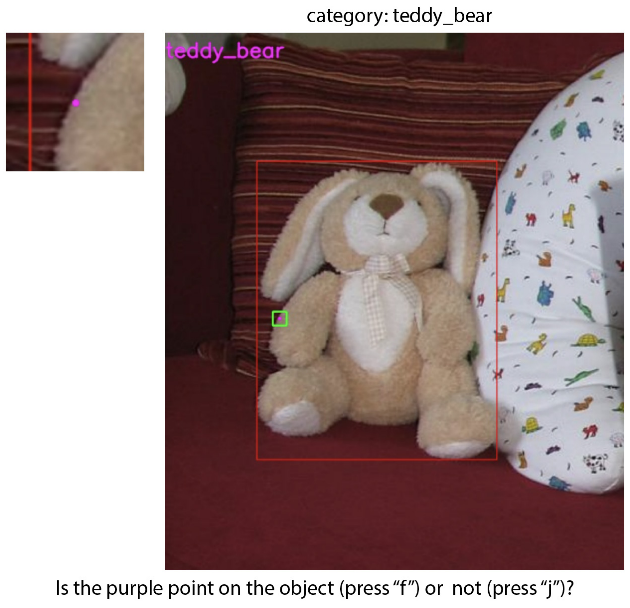
category: teddy_bear
teddy_bear
Is the purple point on the object (press "f") or not (press "j")?

Figure 1. **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.

Estimation for various annotation pipelines. Different annotation collection pipelines for instance segmentation use different protocols for object spotting and categorization stages [2, 6, 8]. 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 [8]. In Figure 3 we compute total annotation time needed to achieve 31.8 AP on COCO[1] with different types of supervision for any $t \geq 0$. We observe, that the point-based annotation is more efficient for label-

[*]Work done during an internship at Facebook AI Research.

[1]BoxInst [13] performance with full COCO `train2017` set.

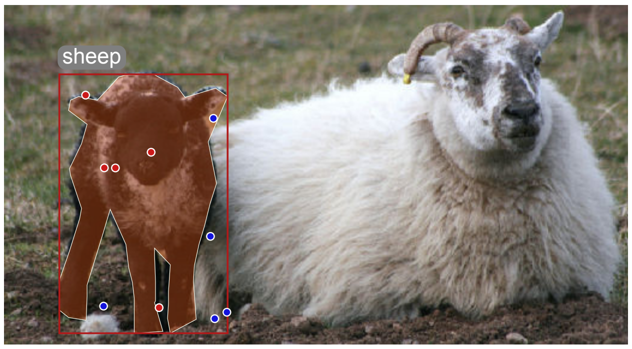

Figure 2. **Point annotation *vs*. polygon-based mask on COCO.** Red points were annotated as object and blue as background. Note, that 3 points (two near both ears and one between the front legs) have correct labels but do not match to the polygon-based mask annotation. We observe that on COCO most of disagreement between point labels and ground truth masks have similar nature.

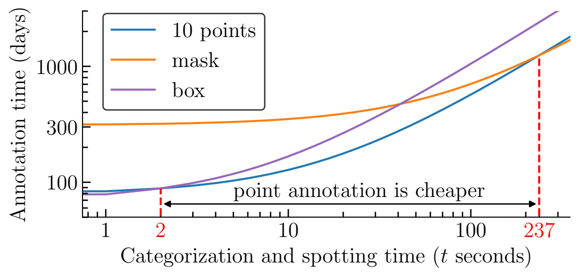

Figure 3. **Total annotation time to achieve 31.8 AP on COCO** depending on the speed of categorization and spotting stages. BoxInst [13] trained with boxes achieves this performance on full COCO `train2017`. Whereas, Mask R-CNN needs only ∼40% and ∼50% of the train set with mask and points supervision 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.

ing pipelines where the spotting and categorization stages take between 2 and 237 seconds per instance on average.

C. Overfitting and Point-based Augmentation

Overfitting with longer training schedules. In our experiments with the COCO dataset [7], we observe that the gap between Mask R-CNN models [3] trained with full mask and point-based (10 points) supervision increases for longer training schedules. For example, for a ResNet-50-FPN [4,7] backbone, the gap is 0.9 AP (35.2 AP *vs*. 34.3 AP) for 1× schedule, but grows to 1.1 AP (37.2 AP *vs*. 36.1 AP) for longer 3× 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× 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.

D. More Analysis on Point-based Annotation

Point sampling schemes. The advantage of using *one* random point over *one* click has been shown in [1]. We are not able to make direct comparison with multiple clicks since the reliable simulation of clicks is impossible and collecting such annotation for large-scale datasets used in our draft is prohibitively expensive for an ablation. We observe that random points are already close to full supervision while being more efficient to collect and simulate. We further explore non-uniform sampling scheme by sampling more points close to boundaries on COCO. We observe the performance of a Mask R-CNN model decreases more with a heavier bias – uniform: 36.1 AP, mildly biased to boundaries: 35.5 AP, heavily biased to boundaries: 27.5 AP.

Point-based annotation for rare objects. We re-generate $\mathcal{P}_{10}$ 3 times for LVIS dataset [2] which has a lot of rare categories (occur in $<$ 10 images). For each version, we train a Mask R-CNN 3 times and the mean and std of AP for rare classes are: 10.8±0.85, 10.2±0.30 and 10.7±0.40. The std over 3 dataset versions is 0.26 which is smaller than the std (0.78) for model trained with mask supervision[2]. This suggests that sampling 10 random points is sufficiently robust for rare categories.

E. COCO Annotation Time

The annotation protocol of the COCO dataset [8] 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

[2]https://www.lvisdataset.org/bestpractices

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

Table 1. **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.

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 [9]. 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 [6, 10], our point annotations takes only 0.9 $\times$ 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 1, we report the annotation time for different annoation format on COCO `train2017` set, which contains 118,287 images and 849,949 instances.

F. Implicit PointRend

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

point aug.	Mask R-CNN [3]	PointRend [5]	Implicit PointRend
	36.1	35.6	36.0
✓	36.0 (-0.1)	35.7 (+0.1)	36.9 (+0.9)

Table 2. **Point-based augmentation for various models** with a ResNet-50-FPN [4, 7] 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.

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 3. **Implicit PointRend ablations** on COCO `train2017` with ResNet-50-FPN [4, 7] backbone. AP ($\mathcal{P}_{10}$) is mask AP for point-based supervision and AP ($\mathcal{M}$) is mask AP for full mask supervision. Table 3a: 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.*) [11] further improve results. Table 3b: 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 [7] backbone (*p2*) further improve the performance

F.1. Ablation study

For the ablation experiments we use a ResNet-50 [4] backbone with FPN [7], 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 [3] and PointRend [5] with a ResNet-50-FPN backbone (see Table 2). 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 3a, 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 [11] further improves the mask prediction by 0.3 AP for point-based supervision. Without the coordinates, the mask head of Implicit PointRend is translation invariant and cannot distinguish two points with similar appearance in the same bounding box. PointRend [5] breaks the invariance by taking coarse mask prediction as input.

Table 3b 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 [7] backbone further improve the performance by 1.6 AP with point-based supervision and 1.4 AP with mask supervision. These experiments suggest that both coordinate and image-level features are essential for the overall performance of Implicit PointRend.

F.2. Mask supervision results

We compare Implicit PointRend with several instance segmentation approaches on the COCO dataset [8] with full mask supervision. The module design is motivated by our point-based supervision, however, Table 4 shows that Implicit PointRend is also a competitive method for fully-supervised instance segmentation as well. While both CondInst [12] 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 [5] 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 [7]. 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. Moreover, PointRend is trained with a more complex importance sampling while much simpler uniform sampling is used for Implicit PointRend, which may also lead to the small drop when larger backbones are used.

	backb.	LRS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [3]	R50	$1\times$	35.2	56.3	37.5	17.2	37.7	50.3
	R50	$3\times$	37.2	58.6	39.9	18.6	39.5	53.3
	R101	$3\times$	38.6	60.4	41.3	19.5	41.3	55.3
	X101	$3\times$	39.5	61.7	42.6	20.7	42.0	56.5
CondInst [12]	R50	$1\times$	35.7	-	-	-	-	-
	R50	$3\times$	37.5	-	-	-	-	-
	R101	$3\times$	38.6	-	-	-	-	-
PointRend [5]	R50	$1\times$	36.2	56.6	38.6	17.1	38.8	52.5
	R50	$3\times$	38.3	59.1	41.1	19.1	40.7	55.8
	R101	$3\times$	40.1	61.1	43.0	20.0	42.9	58.6
	X101	$3\times$	41.1	62.8	44.2	21.5	43.8	59.1
Implicit PointRend	R50	$1\times$	36.9	57.3	39.6	17.7	39.4	53.1
	R50	$3\times$	38.5	59.4	41.4	18.9	41.1	55.0
	R101	$3\times$	39.9	61.1	43.0	20.3	42.7	58.0
	X101	$3\times$	40.8	62.6	43.9	21.5	43.5	57.3

Table 4. **Instance segmentation results with full mask supervision** on COCO `val2017`. LRS: learning rate schedule; a $1\times$ learning rate schedule refers to 90,000 iterations. R50,101: ResNet-50,101 [4]. X101: ResNext-101 32×8d [14]. All models use FPN [7]. The proposed Implicit PointRend also performs better than Mask R-CNN [3] and comparable to PointRend [5] under full mask supervision.

References

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

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

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

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

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

[6] 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.

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

[8] 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.

[9] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *ICCV*, 2017.

[10] 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.

[11] 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.

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

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

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