Deep Variational Instance Segmentation

Thanks for the constructive feedback. We note most reviewers support the novelty of our new paradigm (**R1**, **R3**, **R4**) and the results (**R1**, **R2**, **R4**). We are encouraged that they found our work can inspire future directions (**R2**, **R4**). **R3** recognizes our framework is simple and elegant. In the following we will address their specific comments.

Table S1. AP^r result on the PASCAL VOC and SBD *val.* set.

Method	backbone	split	mAP^r					AP^r_{avg}
			0.5	0.6	0.7	0.8	0.9	
DVIS	Resnet-50-FCN	VOC	68.4	63.3	58.1	49.1	33.7	54.5
DVIS	Resnet-50-FCN	SBD	70.0	67.0	61.0	49.1	27.8	55.0

Whether it's one-stage (R1, R2, R4) or bottom-up (R4) or should compare against two-stage approaches (R2) We agree with **R1** that being anchor-free/proposal-free is a significant advantage of DVIS, as even all the latest "one-stage" approaches verify/predict at least 100-200 masks per image, whereas we only verify less than 20 (Table 6 + refs from R4). We'll revise the paper to emphasize the proposal-free aspect instead of one-stage (a minor revision as all "two-stage" prior work were anchor-based). Based on this, we argue that our performance shouldn't be judged against two-stage work which required significantly more computation than ours (**R2**). We respectfully disagree with **R4** that DVIS is "bottom-up" whereas anchor-based methods are "top-down". Note all CNN papers include a bottom-up backbone, hence even anchor-based methods are not simply top-down. We'd rather note that anchor-based methods are more local since they crop/predict on local regions, whereas we directly predict on the entire image by upsampling from a **low-resolution** feature map which possesses less local information on small objects. We envision DVIS to be used as a "quick global scan" which quickly finds the most prominent instances (more similar to human vision) whereas anchor-based approaches make an exhaustive local search. We will incorporate more discussions in the final paper.

Ablations (Number of instances, effect of window size, unseen categories, individual loss functions (R1, R2, R3, R4)? Those ablations were already presented in supplementary material Sec. 1-5. We showed that (1) the number of instances DVIS predicts is usually adequate in COCO (Supl. Sec. 1); (2) DVIS can predict instances that do not belong to any training categories (Supl. Sec. 5); (3) a large window size is important (Supl. Sec. 2); and (4) the MS loss generally only affects performance at the boundary but not the IoU. The quantization loss has benefits both on the boundary (Supl. Sec. 3) and on the IoU. Note that on DAVIS, DVIS can predict instances from similar categories as PASCAL (e.g. other animals), but have difficulty on ones that are vastly different from the training (e.g. ropes). We believe this is a matter of training data, and indeed there is an objectness factor that the network learned.

The binary loss: Why use it (R1), why not a CE loss (R2), how important (R3). Note that a binary loss is simpler and much faster to compute than the permutation-invariant loss (**R1**). The common cross-entropy (CE) loss tend to penalize differently based on the predicted real-valued labels whereas the robust Huber loss is exactly 0 when the label value ≥ 2 on foreground and ≤ 0 on background. Hence it won't punish different predicted labels differently (**R2**). Binary loss is very important – without it we couldn't train the network properly hence there was no ablation (**R1**,**R3**).

Speed analysis (R1, R3). We don't possess a Titan-Xp hence can't report runtime consistent with literature. On our GTX 1080 Ti GPU, DVIS with ResNet-101-FCN is 20.5% faster than YOLACT-700 with ResNet-101-FPN. According to the YOLACT paper, this should correspond to 28.6 FPS which would be real-time on a Titan-Xp.

The permutation-invariant loss(R2, R4). The memory requirement will indeed be large if all pixel pairs were considered. Hence we use an approach similar to dilated convolution to sample exponentially less neighbors that are further away (Sec. 4 ln. 222-229). In the end, for each pixel only 160 neighbors were considered, which is computationally tractable. We appreciate **R2** for reminding us that Eq. (5) misses a *Relu* function which will be fixed.

Performance w.r.t. Instance size (R2, R4) AP_S, AP_M, and AP_L on COCO were reported in Table 3-4, which represent AP for small, medium, and large objects respectively. Note our performance discrepancy from Mask R-CNN is the highest on small instances, different from R4's intuition (also see answer to the first question). Suppl. Fig. 8 shows some failure cases from COCO *val.*, which were mostly on crowded scenes with small objects.

Different backbone in Table 1 and Table 2 (R2, R4). We ran one more experiment on PASCAL using ResNet-50-FCN as the backbone. As shown in Table S1, results are slightly worse than the DeepLab-v3 backbone, but it still outperforms all competitors in Table 1 and outperforms YOLACT at 0.7 mAP (Table 2), consistent with our claims.

References (R3, R4). We'll cite the related work. Note that none of the work R3 mentioned were about the instance segmentation task. And the work R4 mentioned are very recent, published around after our submission at CVPR/ECCV 2020. None of these work are very similar to our approach. PointRend is proposal-based, SOLO/CenterMask are similar to YOLO in dividing the image into a regular grid and detect the object that is centering in each cell. PolarMask predicts surrogates, extending [3,39,37]. None of them predict instance labels directly from an FCN as our algorithm.

The rounding operation in quantification loss (R2). The rounding operation is piecewise-constant hence we set its gradient to 0. Only gradient on the first f from the quantization term is back-propagated to train the FCN.

The AP^r_{avg} of DIN in Table 1 (R1). Upon inspection, we found that the average of the AP^r in DIN is averaged on IOU thresholds ranging from 0.1 to 0.9, while others range from 0.5 to 0.9. We'll fix this number.