

# Beyond mAP: Towards better evaluation of instance segmentation

Rohit Jena<sup>1†</sup> Lukas Zhorniyak\*<sup>1</sup> Nehal Doiphode\*<sup>1</sup> Pratik Chaudhari<sup>1</sup>  
Vivek Buch<sup>2</sup> James Gee<sup>1</sup> Jianbo Shi<sup>1</sup>

<sup>1</sup> University of Pennsylvania

<sup>2</sup> Stanford University

{rjena, zhorniyak, lahen, pratikac, jshi}@seas.upenn.edu, vpbuch@stanford.edu, gee@upenn.edu

## Abstract

*Correctness of instance segmentation constitutes counting the number of objects, correctly localizing all predictions and classifying each localized prediction. Average Precision is the de-facto metric used to measure all these constituents of segmentation. However, this metric does not penalize duplicate predictions in the high-recall range, and cannot distinguish instances that are localized correctly but categorized incorrectly. This weakness has inadvertently led to network designs that achieve significant gains in AP but also introduce a large number of false positives. We therefore cannot rely on AP to choose a model that provides an optimal tradeoff between false positives and high recall. To resolve this dilemma, we review alternative metrics in the literature and propose two new measures to explicitly measure the amount of both spatial and categorical duplicate predictions. We also propose a Semantic Sorting and NMS module to remove these duplicates based on a pixel occupancy matching scheme. Experiments show that modern segmentation networks have significant gains in AP, but also contain a considerable amount of duplicates. Our Semantic Sorting and NMS can be added as a plug-and-play module to mitigate hedged predictions and preserve AP.*

## 1. Introduction

Tasks like classification and semantic segmentation have a fixed output space, i.e. the  $K$ -dimensional probability distribution of the classes and the per-pixel semantic class respectively. For classification, we can use the zero-one loss, and for semantic segmentation we can use a per-pixel cross entropy loss. On the other hand, instance segmentation is a challenging problem because the output is a set containing an arbitrary number of objects, and the network does not have knowledge of the number of objects in the scene *a priori*. Therefore, the model has to count the correct number of objects in the scene, localize them all and classify

them correctly. Deep learning for instance segmentation has two broad paradigms - top-down and bottom-up instance segmentation. In bottom-up instance segmentation, the image is converted into per-pixel features, and pixel features are aggregated to predict objects. This is typically done by grouping or clustering the pixels based on some similarity in the feature space [2, 7, 13, 30, 36, 41]. In top-down instance segmentation, a model proposes a set of candidate proposals, out of which proposals not containing an object are removed. This leaves us with a smaller set of proposals which are further passed into a localization and classification branch. This is typically followed by an NMS step, since an object may have multiple candidate proposals, so duplicates must be removed. Popular approaches are dominated by top-down methods where the network regresses a bounding box, mask, and category. Mask-RCNN [14, 16, 24] approaches it as a two-stage problem: localize the object, then predict the associated instance segmentation mask. SOLO [37, 38] builds on an anchor-free framework and directly regresses an object segmentation using a spatial grid feature as a probe. More recent work based on Transformers ([6, 12]) explicitly learn a query in the network memory, then refines this prediction. We can interpret all these top-down methods as implementing the query-key paradigm. Each uses different *query* designs: anchor box-based object proposal for Mask R-CNN, grid-cell for SOLO, or learnable latent features for DETR/QueryInst. The Query-Key interaction aims to extract different representations of the object: ROI pooled features for MaskRCNN, center-based convolution filters for SOLO, and cross-attention features in DETR.

In analyzing why top-down methods consistently perform better than bottom-up methods, we make an unusual observation. The qualitative performance of bottom-up methods is at par with that of top-down methods, but there is a significant gap in mAP. Upon further analysis of the precision-recall curves in top-down methods, we find that mAP can be increased by increasing the number of low-confidence predictions. We observe that recent design choices in the literature has exacerbated this problem. In

<sup>†</sup>Correspondence to: rjena@seas.upenn.edu

\*Equal contribution

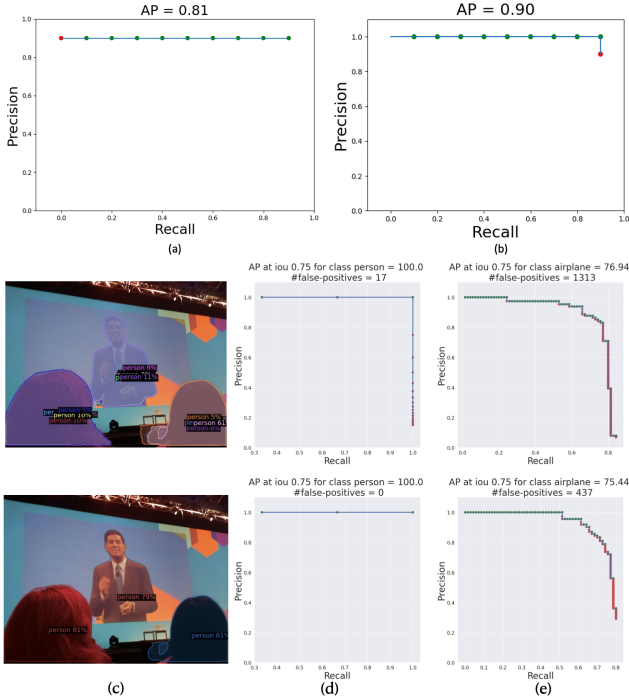


Figure 1. **Top:** Toy example demonstrating how AP changes with a reordering of the same set of detections (9 TPs, 1FP). Note that in (b) the last FP doesn’t contribute to AP. A detection that does not predict this example will also have the same AP. The last prediction in (b) is therefore a hedged prediction. **Middle, Bottom:** SOLOv2 with Matrix and Mask NMS respectively for the same network parameters. (c) shows the qualitative result and (d) is the corresponding P/R curve for the image. Note that hedged predictions do not penalize AP. (e) shows the P/R curve for *airplane* category over entire COCO val dataset. Note that AP increases by 1 point, but number of false positives increase 3-fold.

this work, we take a step back and analyze how mAP can be ‘gamed’ by increasing false positives, explore other metrics in the literature, and propose metrics that explicitly quantify this amount of false positives, both spatially and categorically. Furthermore, we propose a Semantic Sorting and NMS module to improve all metrics related to this excessive amount of prediction, only with a minimal dip in mAP.

## 2. Related works

### 2.1. Instance segmentation

Top-down instance segmentation is often viewed as a localization task followed by pixelwise classification of foreground masks. Among such “detect then segment” strategies is FCIS [21], the first end-to-end fully convolutional work that considers position-sensitive score maps as mask proposals. The score maps are then assembled to produce classification agnostic instance masks and category likelihoods. Along the same line of strategies is MaskR-

CNN [16], a two-stage detector that predicts masks from proposed boxes after RoIAlign operation on feature maps. YOLACT [5] generates non-local prototype masks in an effort to learn and linearly combine them by predicting a set of mask coefficients. However, it relies on accurate bounding box predictions, and doesn’t learn to localize far-away instances. BlendMask [7] attempts to combine FCIS [21] and YOLACT [5] in a hybrid approach. Moving away from box-based object detection, SOLO [38] and CondInst [34] take an anchor-free approach and use position-sensitive *query* to extract object masks directly from the feature map. All these approaches are driven in a top-down manner, where a few query points (often object centers) are responsible for predicting the whole object shape. In contrast, bottom-up approaches focus on grouping pixels into an instance. These approaches, including Hough-voting [13,20], pixel affinity [18,25], Watershed methods [2], pixel embedding [19,26,27], can be thought of as ‘flow’ based: each pixel directly or indirectly learns to flow towards the object center. This ‘flow’ helps to group pixels to its object center, either in the image space or in a latent feature space, therefore localizing all objects simultaneously. However, bottom-up methods are generally worse at localizing smaller objects, dealing with occlusion and crowded objects, and require complex aggregation and post-processing techniques [7,23].

### 2.2. Evaluation of detection & segmentation

Average Precision (AP) [11] is the de-facto metric for measuring the performance of object detection and segmentation models. Hoiem *et al.* [17] provide a way to diagnose the effects of false positives and how they can be mitigated to improve mAP. TIDE [4] also provides a toolkit to identify and decompose the error (1 - mAP) into its constituent error components - such as classification, localization, duplication errors. This can allow a researcher to analyse the major shortcomings of a given detection and segmentation method. AP is therefore widely accepted in the community, and has remained unchallenged as a measure. However, recent works have pointed out different shortcomings in mAP as a reliable metric. Dave *et al.* [10] show that in a large vocabulary detection task, it is possible to gamify the AP metric by adding nonsensical predictions to a given prediction model. LRP [28] highlights two major problems with mAP: 1) different detectors having different P/R curves can have similar APs, but they have different underlying shortcomings, and 2) mAP is not sufficient to quantify localization. LRP also acts as a desirable performance measure in terms of setting an optical confidence score threshold per class, unlike mAP which is optimal at a confidence threshold of 0 for any given model. Our work is similar to [10], but we show that AP can be ‘gamed’ by adding low-confidence false positives, even with a moderate vocabulary task like

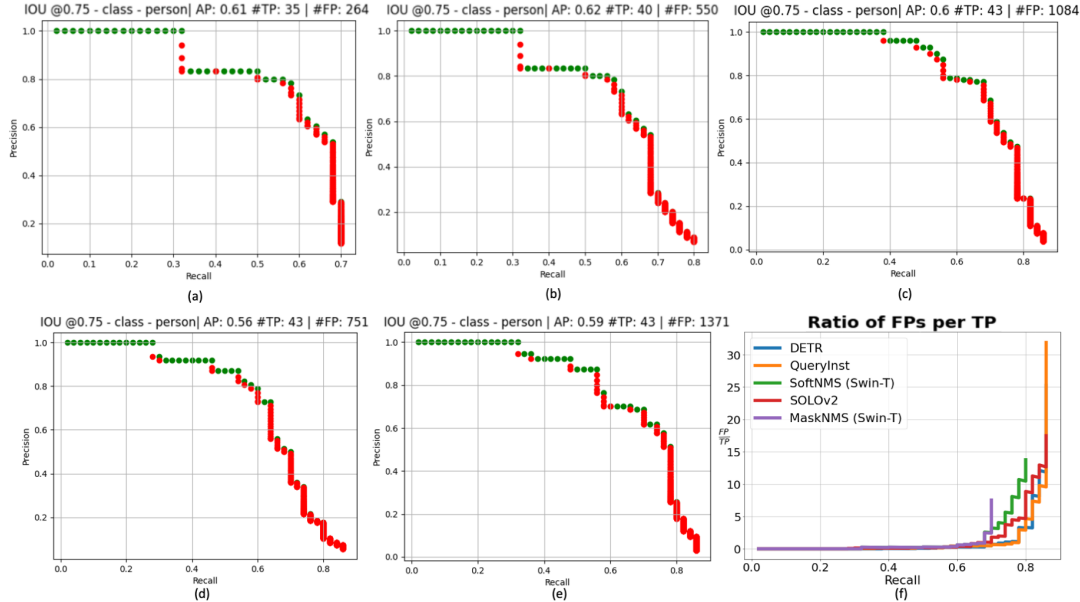


Figure 2. P/R curves for (a) Mask NMS, (b) SoftNMS, (c) SOLOv2, (d) DETR, and (e) QueryInst at IoU=0.75. Each point (red/green) on the P/R curve is a detection that is either a false positive or true positive respectively. (f) shows the ratio of FPs and TPs for (a-e). Note that modern segmentation networks have remarkably more hedging (as seen by the high FP:TP ratio).

COCO [22]. This deficiency of mAP has led to design choices that inadvertently increase mAP by exploiting this behavior [6, 12, 38]. We capture aspects of this deficiency very explicitly, how they influence AP, quantify it and propose a plug-and-play module to mitigate this problem.

### 3. Preliminaries

Average Precision (AP) measures the area under the precision-recall curve of the detector, where the predictions over the dataset are sorted by confidence. See [1, 11] for details on how AP is computed. LRP [28] points out that AP is not confidence-score sensitive. It is instead rank-sensitive.

Consider the following scenario. A detector has a set of 10 predictions for a category with 10 ground truths. Consider two cases where one of the predictions is a false positive (FP). If the highest confidence prediction is a FP, then the maximum precision is 0.9, and the AP is 0.81. If the lowest confidence prediction is a FP, then the precision stays at 1 throughout recall 0 to 0.9, leading to AP of 0.9. The AP curves are shown in Fig.1(top). This is a desirable property in a measure, to penalize a higher confidence FP more than a lower confidence FP. However, note that in the second case, the last FP did not contribute (negatively) to the AP. This is the basis of the shortcoming that we highlight.

#### 3.1. Hedged predictions

In this section, we introduce the terminology around *hedged predictions* in segmentation. Hedging occurs when

a segmentation framework outputs a lot of low-confidence *duplicate* predictions, which are marginally different versions of an ‘original’ high confidence prediction. This perturbation is both spatial and across object categories. We call these predictions *hedged predictions* because the network hedges these low-confidence predictions to increase mAP. Low-confidence FPs do not affect AP, but TPs in this low-confidence, high-recall regime can boost AP slightly.

**Spatial Hedging. (SH)** refers to hedged predictions which are spatially perturbed versions of each other. Spatial Hedging is done by design in top-down detection and segmentation frameworks. In Mask-RCNN [14, 16] the locations in the region proposal network (RPN) are trained for objectness by imposing a minimum IoU criteria with ground truth objects. A single ground-truth box/mask may assign positive labels to multiple anchors. This leads to a hedging scheme where nearby anchors are incentivized to predict the same object. In SOLO [37, 38], a grid cell is labeled as a positive grid if it falls in the *central region* of any ground truth object. As such, multiple grid cells may be assigned to predict the same ground truth object. HTC [8] uses the Mask-RCNN framework as a backbone, and shares the same RPN setup as MaskRCNN. FCOS [35] also assigns positive labels to spatial locations which fall into any ground truth box, which is used in PolarMask [40].

All these are examples of spatial hedging where neighboring spatial anchors are encouraged to predict the same object. Spatial hedging occurs to increase recall, in case

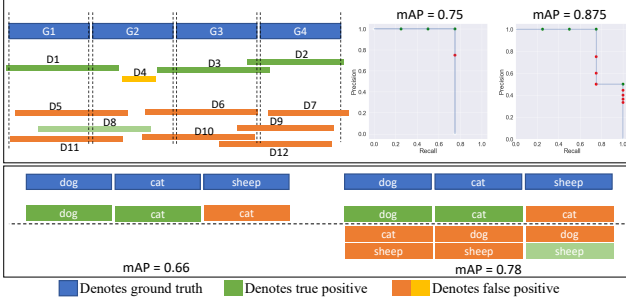


Figure 3. **Illustration of spatial and categorical hedging:** Top: Given ground truths G1-G4 (blue), predictions D1-D4 produce an AP of 0.75 (D4 is a FP). Spatially perturbing D1-D4 to produce objects D5-D12 results in a match of D8 with G2, leading to AP of 0.875.

the ‘central’ anchor fails to predict the object. Typically, NMS is employed to discard duplicate predictions. Recent works [3, 6, 38] have proposed alternate NMS that are faster than Mask NMS, and produce better mAP. A closer inspection reveals that the methods are very effective in decaying the confidence of duplicate predictions, but the post-NMS confidence threshold is chosen to be too low (as low as 0.05 for SOLOv2 and 0.001 for SoftNMS) which leads to lot of duplicates retained post-NMS contributing to hedging.

**Category Hedging. (CH)** occurs when a model predicts multiple object categories for a single object detection. This is a more subtle hedging, which seems to occur due to the way we do inference *i.e.* choosing top-k classes for an object mask [12], or selecting all classes that cross a particular threshold [38]. Note that selecting top-k classes during inference is inconsistent with the instance segmentation task - an object can only be of a single class. The other k-1 classes then end up being hedged predictions. Traditional NMS cannot mitigate this kind of hedging because NMS methods are class-independent. If there are no competing objects in the hedged categories (a car which is also hedged as a truck), then the hedged object will not be removed by NMS. This motivates the need for a category-aware NMS to suppress duplicate objects. This forms the basis for our Semantic NMS in Sec.5.

Methods that are based on a fixed number of queries [6, 12] can perform both spatial and category hedging. QueryInst [12] and DETR [6] use a fixed number of object proposals per image, and a top-k category selection for each proposal can lead to both forms of hedging. DETR explicitly mentions that for all proposals where the top class prediction was ‘background’, replacing the class of those proposals to the second-most confident class increased mAP by 2 points. However, this change essentially predicts 100 objects for *each* image, showing that explicit hedging does improve mAP. However, from an end users’ perspective this

is not desirable, since the user may be more satisfied with a better FP-FN tradeoff than predicting a 100 objects for each image. We take a closer look at the P/R curves of pre-trained models that are used to evaluate mAP. Indeed, there is a tremendous amount of hedging that show up as false positives in the P/R curves. We show this in more detail in Fig. 2. We use the Swin-T model with NMS and SoftNMS in Fig.2(a,b) with a post-NMS threshold of 0.01 for SoftNMS. This is much higher than the default threshold of 0.001. Comparing Fig.2(a) and (b), we see that 5 new TPs are added, but they are all in the  $\geq 0.7$  recall range. Moreover, we add 286 false positives to add only 5 true positives (57.2 FPs added per TP). However, AP doesn’t penalize these newly added FPs. This problem also occurs in state-of-the-art frameworks *i.e.* DETR, SOLOv2, QueryInst (Fig.2(c-e)) where the FPs added per TP rate is upto 138x compared to (a). In Fig. 2(f), we plot the ratio of FPs and TPs with recall. Other methods are effective at having a lower FP/TP ratio at a medium recall range, but it quickly increases in the high recall range due to hedged predictions. Moreover, the AP values are very similar, indicating that hedging is not penalized by AP. More results are present in the Appendix. This motivates the need to quantify the amount of hedging in a segmentation framework.

## 4. Measuring hedging

We propose to quantify the amount of both spatial and categorical hedged predictions. The main idea we use to measure spatial hedging is to measure the amount of mask overlap between predicted instances, weighed by their confidence scores. For category hedging, the main idea is to match predictions and ground truth objects based on mask overlap alone, and then measure the mismatch in categories of matches.

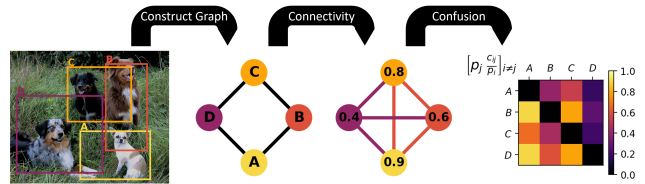


Figure 4. Illustration of duplicate confusion: It works by constructing a graph of objects based on IoU and then measuring a ‘max-flow-like’ measure shown in Eq.3.

### 4.1. Duplicate Confusion Error

Duplicate confusion (DC) measures how connected a given prediction is to all other predictions in terms of overlap, and aggregates this quantity for all predictions. To do this, we first consider an IoU threshold  $t$  and confidence threshold  $v$  and object category  $k$ . Now, consider a set of detections  $\{d_1, d_2 \dots d_m\}$  of category  $k$  for image  $I$ . Let

prediction  $d_i$  have a confidence score of  $\tau_i$ . We construct a graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$  where  $\mathcal{V} = \{d_1 \dots d_m\}$  is the set of predictions and  $\mathcal{E} = \{(i, j) : \text{IoU}(d_i, d_j) \geq t\}$  is the set of edges between these vertices. The connectivity of the graph indicates predictions that have an IoU of at least  $t$ . With slight abuse of notation, we refer to node  $i$  in graph  $\mathcal{G}$  to denote detection  $d_i$ . For two nodes  $i \neq j \in \mathcal{G}$ , consider a path  $\pi_{ij} = [u_1 = i, u_2, \dots, u_M = j]$  such that  $(u_a, u_{a+1}) \in \mathcal{E} \forall a$  and  $a \neq b \implies u_a \neq u_b$ . Consider the set of all such paths  $T_{ij} = \{\pi_{ij}\}_\pi$ . We define the ‘connectivity’ of  $i$  and  $j$  as

$$c_{ij} = \max_{\pi \in T_{ij}} \min_{k \in \pi} \tau_k \quad (1)$$

Roughly, the inner min calculates the ‘weakest link’ in terms of confidence score in the path  $\pi$ , which bottlenecks the (indirect) overlap between  $i$  and  $j$ . The max calculates the highest overlap among all such paths. We solve a variant of the *max-flow* problem to characterize the overlap of predictions  $i$  and  $j$  where the flow along the path  $\pi$  is bottlenecked by  $\min_{k \in \pi} \tau_k$ . If there are no paths between  $i$  and  $j$ , then  $c_{ij} = 0$ . Now, the total amount of overlap for a prediction  $i$  is simply equal to

$$\sum_{j \neq i} \tau_j c_{ij} \quad (2)$$

which is the weighted sum of max-flow to all other nodes  $j$  weighted by confidence  $\tau_j$ . However, for a lot of low-confidence false positives,  $c_{ij}$  will have very small absolute values and will be overshadowed by few high-confidence predictions. To alleviate this problem, we perform relative scaling on Eq.2 by weighing by  $1/\tau_i$ . We then aggregate this over all detections  $d_i$  to get the final connectivity score:

$$\text{DC}_{tv} = \frac{1}{m} \sum_i \sum_{j \neq i} \frac{\tau_j c_{ij}}{\tau_i} \quad (3)$$

Similar to mAP, we compute DC by averaging over IoU thresholds  $t$  and confidence thresholds  $v$ .

This can be interpreted as the confidence of a network in its own counting. This is not, however, a measure of how effectively a network can count instances - the ground truth is not considered when calculating DC. It is important to emphasize that DC by itself is not a good measure, *i.e.* producing no predictions trivially results in 0 DC. This is analogous to model calibration error [15], where 0 calibration error can be achieved trivially by predicting the overall label distribution for every input. However, DC captures spatial hedging effectively, since spatially perturbed predictions will form densely connected graphs, therefore accounting for a quadratic number of interactions among these hedged predictions.

## 4.2. Naming Error

Measuring category hedging requires us to examine correctly localized object detections and analyse if their cate-

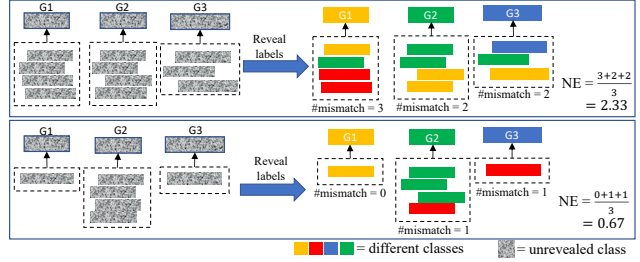


Figure 5. **Illustration of Naming Error (NE):** GTs and detections are matched in a category-agnostic manner, and then the labels are revealed. The average number of mismatches per GT is the NE.

gory is predicted correctly. Category hedging occurs when the model hedges on the category of the object with a single mask *i.e.* top-k category predictions. To measure category hedging, we need to treat object detections and ground truths in a category-agnostic manner.

Let  $\mathcal{G} = \{G_1, G_2 \dots G_n\}$  be the set of *all* ground truth objects in image  $I$  and  $l_{G_i}$  be the category of ground truth  $G_i$ . Similarly, let  $\mathcal{D} = \{D_1, D_2 \dots D_m\}$  be the set of *all* detections output from the network, and  $l_{D_j}$  be the category of detection  $D_j$ . The key intuition we use is that if detection  $D \in \mathcal{D}$  is a category hedged prediction, it will have a high mask overlap with some ground truth object  $G \in \mathcal{G}$ , but will have a label mismatch. For each detection  $D_j$ , we define its ground truth match as:

$$g(D_j) = \begin{cases} \arg \max_i \text{IoU}(D_j, G_i) & , \max_i \text{IoU}(D_j, G_i) \geq 0.5 \\ -1 & , \text{otherwise} \end{cases} \quad (4)$$

This matches each detection to a ground truth object or no object depending on its maximum overlap with the set of ground truth objects. Using the function  $g$  we define the *Naming Error* as:

$$\text{NE} = \frac{1}{N} \sum_{i=1}^N \sum_{j:g(D_j)=i} \mathbb{I}[l_{D_j} \neq l_{G_i}] \quad (5)$$

This gives us the average number of detections that have mismatched labels with a ground truth object.

## 5. Semantic Sorting and NMS

Techniques like NMS have been designed to resolve hedging in principle. However, newer NMS designs [3, 6, 38] use ‘softer’ penalties on duplicate predictions to decay their confidence scores based on overlap with other detections, and threshold the confidence score. Although these designs have led to significant improvements in mAP, we observe that they also contribute to the hedging problem. SoftNMS and SOLOv2 choose a very low post-NMS threshold (0.001 and 0.05 respectively), and DETR selects

---

**Algorithm 1:** Pseudocode for semantic sorting and NMS, given instances  $D_k$  with category  $c_k$  and confidence  $\tau_k$ , threshold  $thr$ , semantic masks  $M$

---

**Data:**  $\{D_k, c_k, \tau_k\}_{k=1\dots N}, \{M_c\}_{c=1\dots C}$

**Result:** Boolean array  $keep$  indicating preservation of instances

```

for  $k = 1 \dots N$  do
   $pr \leftarrow \text{precision}(D_k, M_{c_k});$ 
   $iou \leftarrow \text{IoU}(D_k, M_{c_k});$ 
   $\tau_k \leftarrow \tau_k + pr + (1 - iou);$ 
end
 $(D, c, \tau) = \text{sort}(D, c, \tau);$  // sort by decreasing  $\tau$ 
for  $k = 1 \dots N$  do
   $overlap \leftarrow \text{precision}(D_k, M_{c_k});$ 
  if  $overlap \geq thr$  then
     $keep[k] = True;$ 
     $M_{c_k} = M_{c_k} \setminus D_k$ 
  else
     $keep[k] = False$ 
  end
end

```

---

the next-best class for background predictions. We notice that these design choices lead to considerable spatial hedging (see Fig.2). Moreover, we notice that NMS does not consider suppression between different categories, which is important to mitigate category hedging. This is difficult to design based on detection masks and confidence scores alone, since neural networks are often miscalibrated [15], especially for long-tailed segmentation tasks [10, 29, 33].

**Semantic Sorting.** To alleviate this, we propose a Semantic Sorting algorithm, which re-scores each instance based on its ‘agreement’ with a semantic segmentation output, and a Semantic NMS which discards an instance if its ‘mask occupancy’ is not supported by the semantic mask. Given an image, we predict a semantic segmentation mask  $M$ . This can either be predicted from an off-the-shelf network, or can be learnt alongside instance segmentation [8]. For each instance  $D_k$  with category  $c_k$  and confidence  $\tau_k$ , we measure the fraction of pixels in  $D_k$  that match with semantic mask  $M_{c_k}$  (precision of  $D_k$ ). A low precision implies a lower-quality mask which doesn’t overlap well with the semantic segmentation. We also compute the IoU of the detection and semantic mask, and favor instances with smaller IoU, to retain smaller objects in case of crowded instances. These scores are added to  $\tau_k$  and averaged. This score is then used to reorder the detections for Semantic NMS (Alg.1).

**Semantic NMS.** To alleviate both spatial and category hedging, we propose the use of Semantic NMS. The key idea here is that NMS discards objects based on a minimum

Model	CoordConv	AP <sub>50</sub>	F1 <sub>0.5</sub>	LRP	LRP <sub>Loc</sub>
SOLOv2	✗	96.87	0.47	79.65	16.55
SOLOv2	✓	96.90	0.46	79.87	16.06
Ours	✗	<b>98.01</b>	<b>0.99</b>	33.46	15.87
Ours	✓	<b>98.01</b>	<b>0.99</b>	<b>33.37</b>	<b>15.75</b>

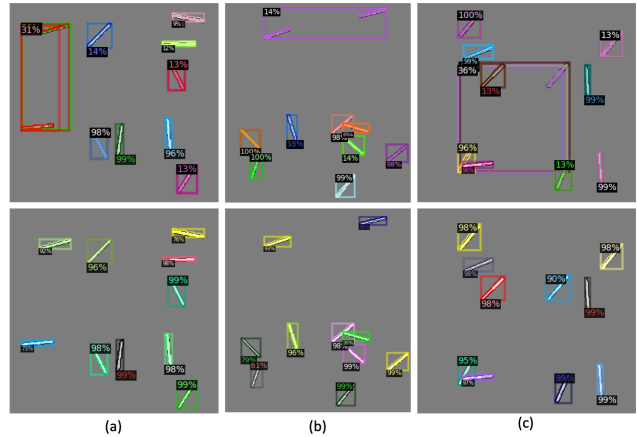


Table 1. Results on synthetic part counting dataset. Note that Semantic Sorting and NMS effectively resolves spatial hedging (as seen by the tremendous improvement in F1 and LRP). Figure (top) shows predictions by SOLOv2 and (bottom) shows our method.

IoU threshold with each other. However, we treat NMS as an occupancy problem *i.e.* an instance whose occupancy is not supported by the semantic mask is discarded. We do this by calculating the fraction of the pixels in  $D_k$  that are present in  $M_{c_k}$ . A substantial overlap indicates that the detection with its category  $c_k$  has occupancy in  $M_{c_k}$  and the detection is preserved. We update  $M_{c_k}$  by subtracting the mask  $D_k$ , deleting the occupancy of this region. Another object with a similar mask will have low overlap with this new  $M_{c_k}$  and will be discarded. This NMS provides us two other benefits: (1) it considers the predicted category to determine whether to keep or discard instances, (2) it runs in  $O(n)$  time unlike NMS which runs in  $O(n^2)$  time.

### 5.1. Implementation details

We run and evaluate all the baselines in the MMDet [9] and Detectron2 [39] frameworks. We implement our evaluation metrics within the *pycocotools* framework [1] to enable rapid adoption of the metrics into currently existing object detection and segmentation frameworks.

## 6. Experiments

We first analyze the effectiveness of Semantic Sorting and NMS on a part-counting toy problem, perform an ablation of different NMS by using SOLOv2 as an illustrative example, and compare different state-of-the-art architectures on their performance of mAP, LRP, and our pro-

Method	NMS	Spatial hedging			Mask quality		Category hedging	AP↑	LRP↓
		DC↓	LRP <sub>FP</sub> ↓	F1↑	b-IoU↑	LRP <sub>Loc</sub> ↓	NE↓		
SOLOv2	Matrix	55.6	92.61	18.46	43.0	22.43	1.93	26.34	95.88
SOLOv2	Mask	<b>16.0</b>	88.52	29.82	42.9	22.13	1.56	26.16	93.54
Ours	Matrix	63.5	91.99	17.87	44.1	22.44	1.68	<b>28.15</b>	95.56
Ours	Mask	17.4	<b>86.76</b>	<b>30.82</b>	<b>44.3</b>	<b>22.12</b>	<b>1.33</b>	<b>27.94</b>	<b>92.60</b>
Ours	Semantic	<b>2.3</b>	<b>79.29</b>	<b>36.05</b>	<b>44.7</b>	<b>21.84</b>	<b>0.98</b>	26.37	<b>89.25</b>

Indicates best result
  Indicates second best result

Table 2. **Ablation of NMS with SOLOv2:** We perform ablations on the type of NMS on coco-minitrain [32] dataset. Adding the semantic sorting and NMS modules leads to better resolution of hedging errors, both spatial and categorical. The only caveat is a slight drop in AP, which comes from pruning the long-tail of the P/R curve, which is generally acceptable from an end-user’s perspective.

posed measures. We also compare boundary IoU [31] and LRP<sub>Loc</sub> for localization quality, and F1-score of detections for an overall hedging measurement (F1 uses precision, which will be low with many hedged predictions). Unless otherwise mentioned, we use SOLOv2 as the base framework on which we implement our modules.

### 6.1. Part counting dataset

To isolate the *spatial hedging* problem, we construct a synthetic part counting dataset. Each image has 10 identical nails that are placed randomly in the image. The locations of the nails are sampled from a truncated random normal distribution around the image center and the nails are placed sequentially. Therefore, nails may be occluding each other, and the ground truth masks are constructed accordingly. Since there is only one class, we can isolate the effect of spatial hedging. Results are in Table.1. Note that SOLOv2’s over-reliance on appearance-based features in its kernel branch leads to instances that are pooled together based on convolution of a kernel feature with the entire mask feature. CoordConv doesn’t seem to resolve this issue either. Moreover, the AP is very high, which may mislead an user to think that the model is very strong. Our semantic sorting + NMS leads to drastic improvements in F1 score and LRP, showing better resolution of hedging.

### 6.2. Ablation on coco-minitrain

We perform ablations on the coco-minitrain [32] dataset. We use coco-minitrain instead of the COCO-train-2017 set owing to similar data statistics as the full training set and to reduce the cost of running ablations. All hyperparameters used for SOLOv2 follow the experimental setup of [38]. The results are in Table 2. Overall, using semantic NMS provide at least a 86.8 % decrease in the duplicate confusion and a 15.4 % increase in the F1 score compared to Matrix and Mask NMS. Using Semantic NMS leads to a much better DC, F1, LRP<sub>FP</sub>, and NE showing better resolution of both spatial and category hedging.

### 6.3. Performance on COCOval dataset

We run our method on the COCO [22] training set. The results are in Table 3. We perform comparison with several SOTA methods to contrast the effect of Semantic NMS with Matrix NMS. Methods like QueryInst [12] use a fixed number of queries (*i.e.* 100) and produce predictions for each query. For R101, QueryInst has the highest AP value, but has the poorest performance in terms of LRP, F1, LRP<sub>FP</sub> and NE, showing that its predictions are prone to both spatial and category hedging. Modern instance segmentation methods perform very competitively in terms of AP, but also produce a noticeable quantity of hedged predictions. MaskRCNN has relatively lower spatial hedging because Mask NMS is effective at removing duplicates, albeit at a higher computational cost. MaskRCNN also uses one category prediction per mask, therefore has the lowest category hedging as well. Our method is built on SOLOv2, and we observe upto a 33x improvement in DC, and upto 10 points of LRP and 19 points of LRP<sub>FP</sub> showing significant resolution of spatial hedging. Notice that the mask quality of our final detections is not compromised, as shown by b-IoU and LRP<sub>Loc</sub> values. There is a slight drop in AP, which occurs because Semantic NMS essentially prunes the ‘tail-end’ of the P/R curve to delete hedged predictions.

### 6.4. Speedup

Semantic NMS has a  $O(n)$  running time complexity, making it faster than Mask NMS while improving all metrics quantifying hedging including LRP, F1, DC, NE and a slight improvement in boundary IoU. On coco-minitrain, Semantic NMS needs an average of  $0.028 \pm 0.027$  seconds, while Mask NMS needs an average of  $0.169 \pm 0.290$  seconds, achieving  $\sim 6.03\times$  speedup. Our NMS can thus prove beneficial when the number of objects is large, such as crowd segmentation.

### 6.5. Qualitative results

Finally, we analyse some qualitative examples comparing a SOLOv2 base model and SOLOv2 with semantic sort-

Method	Spatial hedging			Mask quality		Category hedging	AP $\uparrow$	LRP $\downarrow$
	DC $\downarrow$	LRP $_{FP}$ $\downarrow$	F1 $\uparrow$	b-IoU $\uparrow$	LRP $_{Loc}$ $\downarrow$	NE $\downarrow$		
<b>ResNet-50-FPN</b>								
Mask-RCNN	76.1	<b>80.3</b>	<b>38.4</b>	49.6	20.6	<b>0.63</b>	37.2	<b>88.4</b>
SOLOv2	64.1	90.4	20.8	49.8	20.6	1.13	<b>37.6</b>	94.4
HTC	62.3	93.9	23.3	49.9	<b>20.4</b>	2.19	37.4	96.3
QueryInst (100 queries)	<b>14.9</b>	95.1	17.1	16.9	20.6	2.78	<b>37.5</b>	97.1
CondInst	144.1	88.1	30.7	<b>50.2</b>	20.5	1.35	37.4	92.9
Ours	<b>2.0</b>	<b>78.1</b>	<b>43.3</b>	<b>50.5</b>	<b>20.1</b>	<b>0.94</b>	34.7	<b>87.6</b>
<b>ResNet-101-FPN</b>								
Mask-RCNN	62.6	<b>77.5</b>	<b>41.7</b>	50.4	20.0	<b>0.56</b>	38.6	<b>86.6</b>
SOLOv2	63.1	89.5	21.6	50.8	20.0	1.05	39.0	93.7
HTC	48.3	92.7	26.4	<b>51.1</b>	20.0	1.98	<b>39.6</b>	95.5
QueryInst (100 queries)	<b>10.9</b>	94.7	19.9	17.0	<b>19.6</b>	2.64	<b>41.0</b>	96.7
CondInst	126.2	86.1	33.5	50.9	20.2	1.17	38.5	91.6
Ours	<b>1.9</b>	<b>70.6</b>	<b>45.9</b>	<b>51.4</b>	<b>19.2</b>	<b>0.57</b>	37.4	<b>83.4</b>

Indicates best result
  Indicates second best result

Table 3. **Instance Segmentation on coco-val-2017**: Top and bottom rows are with ResNet-50-FPN and ResNet-101-FPN backbone respectively. Our method outperforms SOLOv2 on the F1 score by a factor of 2.08 and performs substantially better at all metrics, showing better resolution of spatial and category hedging, without compromising mask quality.

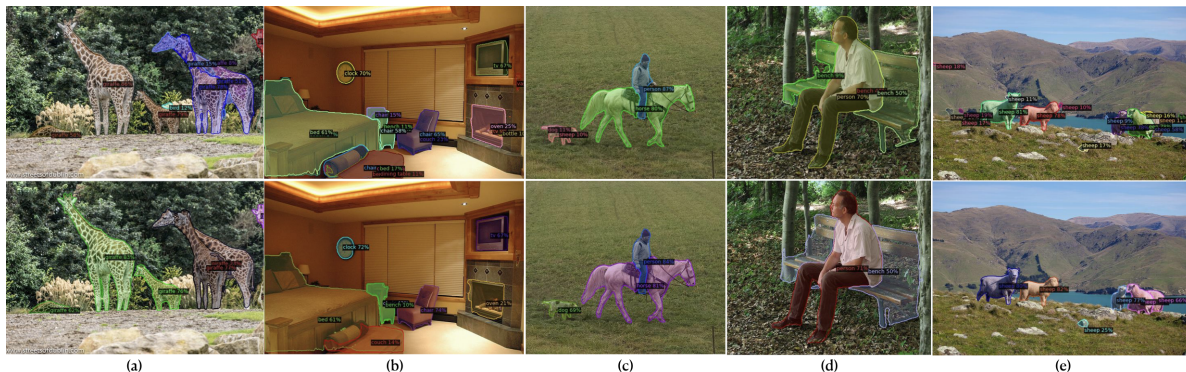


Figure 6. **Qualitative results on COCO val**: Top (SOLOv2) and bottom (Ours) show surprisingly different results with only a minimal difference in mAP. In (b,c) our method removes category hedging (chair  $\rightarrow$  couch, bed  $\rightarrow$  bench, couch  $\rightarrow$  dining table, dog  $\rightarrow$  sheep) and removes spatial hedging in (a,d,e). This drastic difference in hedging is reflected in F1 score, LRP, DC and NE measures (Table 3).

ing and NMS. Outputs are shown without any other post-processing, apart from that performed by the NMS. In Fig. 6 we note that our method resolves both spatial and category hedging, while Mask NMS can only resolve spatial hedging. More qualitative results are in the Appendix.

## 7. Conclusion

Average Precision has been a longstanding metric to evaluate instance detection and segmentation. However, AP has a few shortcomings. We show that AP does not penalize false positives near the tail-end of the precision-recall curve. Modern segmentation networks perform very competitively

in terms of AP, but also introduce *hedged predictions* which might be undesirable for a user. We review alternate metrics in the literature, and propose measures to explicitly quantify hedging. Modern segmentation networks turn out to have a considerable hedging problem, both spatial and categorical. To mitigate this, we also propose a semantic sorting and NMS module. Experiments on three datasets show that our method considerably prunes out hedged predictions without sacrificing mask quality, while being much faster than MaskNMS.



## References

- [1] Coco api. <https://github.com/cocodataset/cocoapi>. 3, 6
- [2] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017. 1, 2
- [3] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms—improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017. 4, 5
- [4] Daniel Bolya, Sean Foley, James Hays, and Judy Hoffman. Tide: A general toolbox for identifying object detection errors. In *European Conference on Computer Vision*, pages 558–573. Springer, 2020. 2
- [5] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9157–9166, 2019. 2
- [6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 1, 3, 4, 5
- [7] Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. Blendmask: Top-down meets bottom-up for instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8573–8581, 2020. 1, 2
- [8] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019. 3, 6
- [9] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 6
- [10] Achal Dave, Piotr Dollár, Deva Ramanan, Alexander Kirillov, and Ross Girshick. Evaluating large-vocabulary object detectors: The devil is in the details. *arXiv preprint arXiv:2102.01066*, 2021. 2, 6
- [11] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2, 3
- [12] Yuxin Fang, Shusheng Yang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Instances as queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6910–6919, 2021. 1, 3, 4, 7
- [13] Juergen Gall and Victor Lempitsky. Class-specific hough forests for object detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1022–1029, 2009. 1, 2
- [14] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. 1, 3
- [15] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017. 5, 6
- [16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1, 2, 3
- [17] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *European conference on computer vision*, pages 340–353. Springer, 2012. 2
- [18] Tsung-Wei Ke, Jyh-Jing Hwang, Ziwei Liu, and Stella X Yu. Adaptive affinity fields for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 587–602, 2018. 2
- [19] Shu Kong and Charless C Fowlkes. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9018–9028, 2018. 2
- [20] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, volume 2, page 7, 2004. 2
- [21] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2359–2367, 2017. 2
- [22] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 3, 7
- [23] Yiding Liu, Siyu Yang, Bin Li, Wengang Zhou, Jizheng Xu, Houqiang Li, and Yan Lu. Affinity derivation and graph merge for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 686–703, 2018. 2
- [24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 1
- [25] Michael Maire, Takuya Narihira, and Stella X Yu. Affinity cnn: Learning pixel-centric pairwise relations for figure/ground embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 174–182, 2016. 2
- [26] Michael Maire, Stella X. Yu, and Pietro Perona. Object detection and segmentation from joint embedding of parts and

- pixels. In *2011 International Conference on Computer Vision*, pages 2142–2149, 2011. 2
- [27] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, page 2274–2284, 2017. 2
- [28] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. Localization recall precision (lrp): A new performance metric for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2, 3
- [29] Tai-Yu Pan, Cheng Zhang, Yandong Li, Hexiang Hu, Dong Xuan, Soravit Changpinyo, Boqing Gong, and Wei-Lun Chao. On model calibration for long-tailed object detection and instance segmentation. *Advances in Neural Information Processing Systems*, 34:2529–2542, 2021. 6
- [30] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1
- [31] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 7
- [32] Nermin Samet, Samet Hicsonmez, and Emre Akbas. Houghnet: Integrating near and long-range evidence for bottom-up object detection. In *European Conference on Computer Vision (ECCV)*, 2020. 7
- [33] Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 6
- [34] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 282–298. Springer, 2020. 2
- [35] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 3
- [36] Weiyao Wang, Matt Feiszli, Heng Wang, Jitendra Malik, and Du Tran. Open-world instance segmentation: Exploiting pseudo ground truth from learned pairwise affinity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4422–4432, June 2022. 1
- [37] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In *European Conference on Computer Vision*, pages 649–665. Springer, 2020. 1, 3
- [38] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17721–17732. Curran Associates, Inc., 2020. 1, 2, 3, 4, 5, 7
- [39] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019. 6
- [40] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12193–12202, 2020. 3
- [41] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1