

---

# A High-Resolution Dataset for Instance Detection with Multi-View Instance Capture

---

Qianqian Shen<sup>1,\*</sup> Yunhan Zhao<sup>2,\*</sup> Nahyun Kwon<sup>3</sup> Jeeun Kim<sup>3</sup> Yanan Li<sup>1</sup> Shu Kong<sup>3,4</sup>  
<sup>1</sup>Zhejiang Lab    <sup>2</sup>UC-Irvine    <sup>3</sup>Texas A&M University    <sup>4</sup>University of Macau

*Dataset and open-source code*

## Abstract

Instance detection (InsDet) is a long-lasting problem in robotics and computer vision, aiming to detect object instances (predefined by some visual examples) in a cluttered scene. Despite its practical significance, its advancement is overshadowed by Object Detection, which aims to detect objects belonging to some predefined classes. One major reason is that current InsDet datasets are too small in scale by today’s standards. For example, the popular InsDet dataset GMU (published in 2016) has only 23 instances, far less than COCO (80 classes), a well-known object detection dataset published in 2014. We are motivated to introduce a new InsDet dataset and protocol. First, we define a realistic setup for InsDet: training data consists of multi-view instance captures, along with diverse scene images allowing synthesizing training images by pasting instance images on them with free box annotations. Second, we release a real-world database, which contains multi-view capture of 100 object instances, and high-resolution (6k×8k) testing images. Third, we extensively study baseline methods for InsDet on our dataset, analyze their performance and suggest future work. Somewhat surprisingly, using the off-the-shelf class-agnostic segmentation model (Segment Anything Model, SAM) and the self-supervised feature representation DINOv2 performs the best, achieving >10 AP better than end-to-end trained InsDet models that repurpose object detectors (e.g., FasterRCNN and RetinaNet).

## 1 Introduction

Instance detection (InsDet) requires detecting specific object instances (defined by some visual examples) from a scene image [12]. It is practically important in robotics, e.g., elderly-assistant robots need to fetch specific items (*my*-cup vs. *your*-cup) from a cluttered kitchen [42], micro-fulfillment robots for the retail need to pick items from mixed boxes or shelves [4].

**Motivation.** InsDet receives much less attention than the related problem of Object Detection (ObjDet), which aims to detect all objects belonging to some predefined classes [30, 39, 31, 50]. Fig. 1 compares the two problems. *One major reason is that there are not large-enough InsDet datasets by today’s standards.* For example, the popular InsDet dataset GMU (published in 2016) [16] has only 23 object instances while the popular ObjDet dataset COCO has 80 object classes (published in 2014) [30]. Moreover, *there are no unified protocols in the literature of InsDet.* The current InsDet literature mixes multiple datasets to simulate training images and testing scenarios [12]. Note that the training protocol of InsDet does not follow that of ObjDet, which has training images annotated with bounding boxes. Differently, for InsDet,<sup>2</sup> its setup should have profile images of instances (cf. right

---

\*co-first authors.

<sup>2</sup>In real-world applications (e.g., robot learning), it is infeasible to place objects in diverse scenes, take scene photos, then annotate instances using boxes towards training images (cf. training data in object detection).

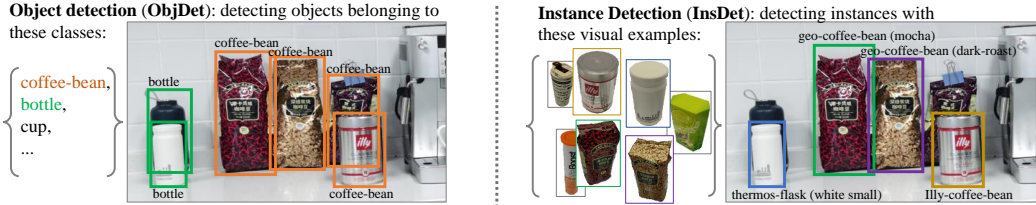


Figure 1: **Object detection (ObjDet) vs. instance detection (InsDet)**. ObjDet aims to detect all objects belonging to some predefined classes, whereas InsDet requires detecting specific object instances defined by some visual examples. Loosely speaking, InsDet treats a single object instance as a class compared to ObjDet. Please refer to Fig. 2-right for the challenge of InsDet, which is the focus of our work.

in Fig. 1) and optionally diverse background images not containing such instances [12]. We release a new dataset and present a unified protocol to foster the InsDet research.

**Overview of our dataset** is presented in Fig. 2. In our dataset, profile images (3072x3072) of object instances and testing images (6144x8192) are high-resolution captured by a Leica camera (commonly used in today’s cellphones). This inexpensive camera is deployable in current or future robot devices. Hence, our dataset simulates real-world scenarios, e.g., robotic navigation in indoor scenes. Even with high-resolution images, objects in testing images appear small, taking only a tiny region in the high-res images. This demonstrates a clear challenge of InsDet in our dataset. Therefore, our dataset allows studying InsDet methods towards real-time operation on high-res (as future work).

**Preview of technical insights.** On our dataset, we revisit existing InsDet methods [28, 12, 18]. Perhaps the only InsDet framework is cut-paste-learn [12], which cuts instances from their profile images, pastes them on random background images (so being able to derive “free” bounding boxes annotations), and trains InsDet detectors on such data by following that of ObjDet (e.g., FasterRCNN [39]). We study this framework, train different detectors, and confirm that the state-of-the-art transformer-based detector DINO [50] performs the best, achieving 27.99 AP, significantly better than CNN-based detector FasterRCNN (19.52 AP). Further, we present a non-learned method that runs off-the-shelf proposal detectors (SAM [25] in our work) to generate object proposals and use self-supervised learned features (DINO<sub>f</sub> [8]<sup>3</sup> and DINOv2<sub>f</sub> [35]) to find matched proposals to instances’ profile images. Surprisingly, this non-learned method resoundingly outperforms end-to-end learning methods, i.e., SAM+DINOv2<sub>f</sub> achieves 41.61 AP, much better than DINO (27.99 AP) [50].

**Contributions.** We make three major contributions.

1. We formulate the InsDet problem with a unified protocol and release a challenging dataset consisting of both high-resolution profile images and high-res testing images.
2. We conduct extensive experiments on our dataset and benchmark representative methods following the cut-paste-learn framework [12], showing that stronger detectors perform better.
3. We present a non-learned method that uses an off-the-shelf proposal detector (i.e., SAM [25]) to produce proposals, and self-supervised learned features (e.g., DINOv2<sub>f</sub> [35]) to find instances (which are well matched to their profile images). This simple method significantly outperforms the end-to-end InsDet models.

## 2 Related Work

**Instance Detection (InsDet)** is a long-lasting problem in computer vision and robotics [51, 12, 34, 3, 17, 23, 4], referring to detecting specific object instances in a scene image. Traditional InsDet methods use keypoint matching [36] or template matching [21]; more recent ones train deep neural networks to approach InsDet [34]. Some others focus on obtaining more training samples by rendering realistic instance examples [24, 23], data augmentation [12], and synthesizing training images by cutting instances as foregrounds and pasting them to background images [28, 12, 18]. Speaking of InsDet datasets, [16] collects scene images from 9 kitchen scenes with RGB-D cameras and defines 23 instances of interest to annotate with 2D boxes on scene images; [23] creates 3D models of 29 instances from 6 indoor scenes, and uses them to synthesize training and testing data; [4] creates 3D

<sup>3</sup>We add subscript  $f$  to indicate that DINO<sub>f</sub> [8] is the self-supervised learned feature extractor; distinguishing it from a well-known object detector DINO [50].

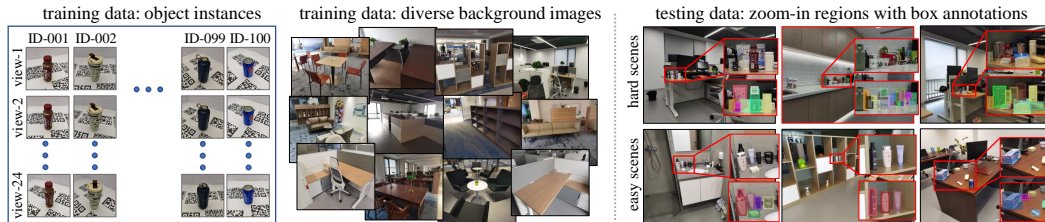


Figure 2: **Overview of our instance detection dataset.** **Left:** It contains 100 distinct object instances. For each of them, we capture 24 profile photos from multiple views. We paste QR code images beneath objects to allow relative camera estimation (e.g., by COLMAP [43]), just like other existing datasets [22, 5]. **Middle:** We take photos in random scenes (which do not contain any of the 100 instances) as background images. The background images can be optionally used to synthesize training data, e.g., pasting the foreground instances on them towards box-annotated training images [28, 12, 18] as used in the object detection literature [30]. **Right:** high-resolution ( $6k \times 8k$ ) testing images of clutter scenes contain diverse instances, including some of the 100 predefined instances and other uninterested ones. The goal of InsDet is to detect the predefined instances in these testing images. From the zoom-in regions, we see the scene clutters make InsDet a rather challenging problem.

mesh models of 100 grocery store objects, renders 80 views of images for each instance, and uses them to synthesize training data.

As for benchmarking protocol of InsDet, [12] synthesizes training data from BigBird [45] and UW Scenes [27] and tests on the GMU dataset [16]; [23] trains on their in-house data and test on LM-O [5] and Rutgers APC [40] datasets. Moreover, some works require hardware-demanding setups [4], some synthesize both training and testing data [23, 28], while others mix existing datasets for benchmarking [12]. Given that the modern literature on InsDet lacks a unified benchmarking protocol (till now!), we introduce a more realistic unified protocol along with our InsDet dataset, allowing fairly benchmarking methods and fostering research of InsDet.

**Object Detection (ObjDet)** is a fundamental computer vision problem [13, 30, 39], requiring detecting all objects belonging to some predefined categories. The prevalent ObjDet detectors adopt convolutional neural networks (CNNs) as a backbone and a detector-head for proposal detection and classification, typically using bounding box regression and a softmax-classifier. Approaches can be grouped into two categories: one-stage detectors [38, 32, 37, 48] and two-stage detectors [19, 6]. One-stage detectors predict candidate detection proposals using bounding boxes and labels at regular spatial positions over feature maps; two-stage detectors first produce detection proposals, then perform classification and bounding box regression for each proposal. Recently, the transformer-based detectors transcend CNN-based detectors [7, 53, 50], yielding much better performance on various ObjDet benchmarks. Different from ObjDet, InsDet requires distinguishing individual object instances within a class. Nevertheless, to approach InsDet, the common practice is to repurpose ObjDet detectors by treating unique instances as individual classes. We follow this practice and benchmark various ObjDet methods on our InsDet dataset.

**Pretrained Models.** Pretraining is an effective way to learn features from diverse data. For example, training on the large-scale ImageNet dataset for image classification [10], a neural network can serve as a powerful feature extractor for various vision tasks [11, 44]. Object detectors trained on the COCO dataset [30] can serve as a backbone allowing finetuning on a target domain to improve detection performance [29]. Such pretraining requires human annotations which can be costly. Therefore, self-supervised pretraining has attracted increasing attention and achieved remarkable progress [9, 20, 8, 35]. Moreover, the recent literature shows that pretraining on much larger-scale data can serve as a foundation model for being able to perform well across domains and tasks. For example, the Segment Anything Model (SAM) pretrains a class-agnostic proposal detector on web-scale data and shows an impressive ability to detect and segment diverse objects in the wild [25]. In this work, with our high-res InsDet dataset, we explore a non-learned method by using publicly available pretrained models. We show that such a simple method significantly outperforms end-to-end learned InsDet detectors.

### 3 Instance Detection: Protocol and Dataset

In this section, we formulate a realistic unified InsDet protocol and introduce the new dataset. We release our dataset under the MIT License, hoping to contribute to the broader research community.

### 3.1 The Protocol

Our InsDet protocol is motivated by real-world indoor robotic applications. In particular, we consider the scenario that assistive robots must locate and recognize instances to fetch them in a cluttered indoor scene [42], where InsDet is a crucial component. Realistically, for a given object instance, the robots should see it only from a few views (*at the training stage*), and then accurately detect it *in a distance* in any scenes (*at the testing stage*). Therefore, we suggest the protocol specifying the training and testing setups below. We refer the readers to Fig. 2 for an illustration of this protocol.

- **Training.** There are profile images of each instance captured at different views and diverse background images. The background images can be used to synthesize training images with free 2D-box annotations, as done by the cut-paste-learn methods [28, 12, 18].
- **Testing.** InsDet algorithms are required to precisely detect all predefined instances from real-world images of cluttered scenes.

**Evaluation metrics.** The InsDet literature commonly uses average precision (AP) at IoU=0.5 [12, 2, 34]; others use different metrics, e.g., AP at IoU=0.75 [23], mean AP [3, 17], and F1 score [4]. As a single metric appears to be insufficient to benchmark methods, we follow the literature of ObjDet that uses multiple metrics altogether [30].

- **AP** averages the precision at IoU thresholds from 0.5 to 0.95 with the step size 0.05. It is the *primary metric* in the most well-known COCO Object Detection dataset [30].
- **AP<sub>50</sub>** and **AP<sub>75</sub>** are the precision averaged over all instances with IoU threshold as 0.5 and 0.75, respectively. In particular, **AP<sub>50</sub>** is the widely used metric in the literature of InsDet.
- **AR** (average recall) averages the proposal recall at IoU threshold from 0.5 to 1.0 with the step size 0.05, regardless of the classification accuracy. AR measures the localization performance (excluding classification accuracy) of an InsDet model.

Moreover, we tag *hard* and *easy* scenes in the testing images based on the level of clutter and occlusion, as shown by the right panel of Fig. 2. Following the COCO dataset [30], we further tag testing object instances as *small*, *medium*, and *large* according to their bounding box area (cf. details in the supplement). These tags allow a breakdown analysis to better analyze methods.

### 3.2 The Dataset

We introduce a challenging real-world dataset of indoor scenes (motivated by indoor assistive robots), including high-resolution photos of 100 distinct object instances, and high-resolution testing images captured from 14 indoor scenes where there are such 100 instances defined for InsDet. Table 1 summarizes the statistics compared with existing datasets, showing that our dataset is larger in scale and more challenging than existing InsDet datasets. Importantly, object instances are located far from the camera in cluttered scenes; this is realistic because robots must detect objects in a distance before approaching them [1]. Perhaps surprisingly, only a few InsDet datasets exist in the literature. Among them, Grocery [4], which is the latest and has the most instances like our dataset, is not publicly available.

Our InsDet dataset contains 100 object instances. When capturing photos for each instance, inspired by prior arts [45, 22, 5], we paste a QR code on the tabletop, which enables pose estimation, e.g., using COLMAP [43]. Yet, we note more realistic scenarios can be hand-holding instances for capturing [26], which we think of as future work. Each instance photo is of 3072×3072 pixel resolution. For each instance, we capture 24 photos from multiple views. The left panel of Fig. 2 shows some random photos for some instances. For the testing set, we capture high-resolution images (6144×8192) in cluttered scenes, where some instances are placed in reasonable locations, as shown in the right panel of Fig. 2. We tag these images as *easy* or *hard* based on scene clutter and object occlusion levels. When objects are placed sparsely, we tag the testing images as *easy*; otherwise, we tag them as *hard*. Our InsDet dataset also contains 200 high-res background images of indoor scenes (cf. Fig. 2-middle). These indoor scenes are not included in testing images. They allow using

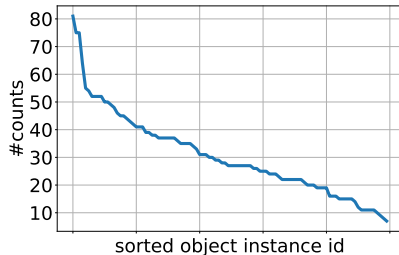
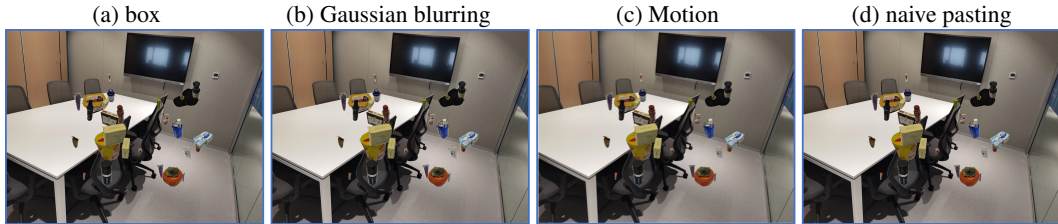


Figure 3: Imbalanced distribution of instances in test-set. Yet, instances have the same number of profile images in training and the metrics average over all instances. So, the evaluation is unbiased.

**Table 1: Comparison of our dataset to existing ones.** Several datasets are used in the InsDet literature although they are designed for different tasks. For example, BigBird and LM are designed to study algorithms of object recognition and object pose estimation, hence they contain instances that are close to the camera. Naively repurposing them for InsDet leads to saturated performance, impoverishing the exploration space of InsDet. Instead, ours is more challenging as instances are placed far from the camera, simulating realistic scenarios where robots must detect instances at a distance. Importantly, our dataset contains far more instances than other publicly available InsDet datasets.

	for what task	publicly available	#instances	#scenes	published year	resolution
BigBird [45]	recognition	✓	100	N/A	2014	1280x1024
RGBD [28]	scene label.	✓	300	14	2017	N/A
LM [22]	6D pose est.	✓	15	1	2012	480x640
LM-O [5]	6D pose est.	✓	20	1	2017	480x640
RU-APC [40]	3D pose est.	✓	14	1	2016	480x640
GMU [16]	InsDet	✓	23	9	2016	1080x1920
AVD [1]	InsDet	✓	33	9	2017	1080x1920
Grocery [4]	InsDet	✗	100	10	2021	unknown
Ours	InsDet	✓	100	14	2023	6144x8192



**Figure 4: Synthetic training images for cut-paste-learn methods.** We use different blending methods to paste object instances on the same background. We recommend that interested readers refer to the supplement for an ablation study using different blending methods.

the cut-paste-learn framework to synthesize training images [28, 12, 18]. Following this framework, we segment foreground instances using GrabCut [41] to paste them on background images. It is worth noting that the recent vision foundation model SAM [25] makes interactive segmentation much more efficient. Yet, this work is made public after we collected our dataset. In Fig. 3, we plot the per-instance frequency in the testing set.

## 4 Methodology

### 4.1 The Strong Baseline: Cut-Paste-Learn

**Cut-Paste-Learn** serves as a strong baseline that synthesizes training images with 2D-box annotations [12]. This allows one to train InsDet detectors in the same way as training normal ObjDet detectors, by simply treating the  $K$  unique instances as  $K$  distinct classes. It cuts and pastes foreground instances at various aspect ratios and scales on diverse background images, yielding synthetic training images, as shown in Fig. 4. Cut-paste-learn is model-agnostic, allowing one to adopt any state-of-the-art detector architecture. In this work, we study five popular detectors, covering the two-stage detector FasterRCNN [39], and one-stage anchor-based detector RetinaNet [31], and one-stage anchor-free detectors CenterNet [51], and FCOS [47]; and the transformer-based detector DINO [50]. There are multiple factors in the cut-paste-learn framework, such as the number of inserted objects in each background image, their relative size, the number of generated training images and blending methods. We conduct comprehensive ablation studies and report results using the best-tuned choices. We refer interested readers to the supplement for the ablation studies.

### 4.2 The Simple, Non-Learned Method

We introduce a simple, non-learned InsDet method by exploiting publicly available pretrained models. This method consists of three main steps: (1) proposal generation on testing images, (2) matching proposals and profile images, (3) selecting the best-matched proposals as the detected instances.

**Proposal generation.** We use the recently released Segment Anything Model (SAM) [25] to generate proposals. For a proposal, we define a minimum bounding square box encapsulating the masked instance, and then crop the region from the high-resolution testing image. SAM not only achieves high recall (Table 3) on our InsDet dataset but detects objects not belonging to the instances of interest. So the next step is to find interested instances from the proposals.

**Feature representation of proposals and profile images.** Intuitively, among the pool of proposals, we are interested in those that are well-matched to any profile images of any instance. The well-matched ones are more likely to be predefined instances. To match proposals and profile images, we use off-the-shelf features to represent them. In this work, we study two self-supervised learned models as feature extractors, i.e.  $DINO_f$  [8], and  $DINOv2_f$  [35]. We feed a square crop (of a proposal) or a profile image to the feature extractor to obtain its feature representation. We use cosine similarity over the features as the similarity measure between a proposal and a profile image.

**Proposal matching and selection.** As each instance has multiple profile images, we need to design the similarity between a proposal and an instance. For a proposal, we compute the cosine similarities of its feature to all the profile images of an instance and use the maximum as its final similarity to this instance. We then filter out proposals and instances if they have similarities lower than a threshold, indicating that they are not matched to any instances or proposals. Finally, we obtain a similarity matrix between all remaining proposals and all remaining instances. Over this matrix, we study two matching algorithms to find the best match (hence the final InsDet results), i.e. Rank & Select, and Stable Matching [14, 33]. The former is a greedy algorithm that iteratively selects the best match (highest cosine similarity) between a proposal and an instance and removes the corresponding proposal until no proposal/instance is left. The latter produces an optimal list of matched proposals and instances, such that there exist no pair of instances and proposals which both prefer each other to their current correspondence under the matching.

## 5 Experiments

**Synthesizing training images for cut-paste-learn baselines.** Our baseline method trains state-of-the-art ObjDet detectors on data synthesized using the cut-paste-learn strategy [12]. For evaluating on our InsDet dataset, we generate 19k training examples and 6k validation examples. For each example, various numbers of foreground objects ranging from 25 to 35 are pasted to a randomly selected background image. The objects are randomly resized with a scale from 0.15 to 0.5. We use four blending options [12], including Gaussian blurring, motion blurring, box blurring, and naive pasting. Fig. 4 shows some random synthetic images. The above factors have a notable impact on the final performance of trained models, and we have conducted a comprehensive ablation study. We refer interested readers to the supplement for the study.

**Implementation details.** We conduct all the experiments based on open-source implementations, such as Detectron2 [49] (for FasterRCNN and RetinaNet), CenterNet [52], FCOS [46] and DINO [50]. The CNN-based end-to-end detectors are initialized with pretrained weights on COCO [30]. We fine-tune CNN-based models using SGD and the transformer-based model using AdamW with a learning rate of  $1e-3$  and a batch size of 16. We fine-tune all the models for 5 epochs (which are enough for training to converge) and evaluate checkpoints after each epoch for model selection. The models are trained on a single Tesla V100 GPU with 32G memory.

If applied, we preprocess object instance profile images and proposals. Specifically, for a profile image, we remove the background pixels (e.g., pixels of QR code) using foreground segmentation (i.e., GrabCut). For each proposal, we crop its minimum bounding square box. We also study whether removing background pixels by using SAM’s mask output performs better. We use  $DINO_f$  and  $DINOv2_f$  to compute feature representations.

### 5.1 Benchmarking Results

**Quantitative results.** To evaluate the proposed InsDet protocol and dataset, we first train detectors from a COCO-pretrained backbone following the cut-paste-learn baseline. Table 2 lists detailed comparisons and Fig. 5 plots the precision-recall curves for the compared methods. We can see that detectors with stronger architectures perform better, e.g. DINO (27.99% AP) vs. FasterRCNN (19.54% AP). Second, non-learned methods outperform end-to-end trained models, e.g., SAM+ $DINOv2_f$

Table 2: **Benchmarking results on our dataset.** We summarize three salient conclusions. (1) End-to-end trained detectors perform better with stronger detector architectures, e.g., the transformer DINO (27.99 AP) outperforms FasterRCNN (19.54 AP). (2) Interestingly, the non-learned method SAM+DINOv2<sub>f</sub> performs the best (41.61 AP), significantly better than end-to-end learned detectors including DINO (27.99 AP). (3) All methods have much lower AP on hard testing images or small objects (e.g., SAM+DINOv2<sub>f</sub> yields 28.03 AP on hard vs. 47.57 AP on easy), showing that future work should focus on hard situations or small instances.

	AP						AP <sub>50</sub>	AP <sub>75</sub>
	avg	hard	easy	small	medium	large		
FasterRCNN [39]	19.54	10.26	23.75	5.03	22.20	37.97	29.21	23.26
RetinaNet [31]	22.22	14.92	26.49	5.48	25.80	42.71	31.19	24.98
CenterNet [51]	21.12	11.85	25.70	5.90	24.15	40.38	32.72	23.60
FCOS [47]	22.40	13.22	28.68	6.17	26.46	38.13	32.80	25.47
DINO [50]	27.99	17.89	32.65	11.51	31.60	48.35	39.62	32.19
SAM + DINO <sub>f</sub>	36.97	22.38	43.88	11.93	40.85	62.67	44.13	40.42
SAM + DINOv2 <sub>f</sub>	<b>41.61</b>	<b>28.03</b>	<b>47.57</b>	<b>14.58</b>	<b>45.83</b>	<b>69.14</b>	<b>49.10</b>	<b>45.95</b>

Table 3: **Benchmarking results w.r.t average recall (AR).** “AR@max10” means AR within the top-10 ranked detections. In computing AR, we rank detections by using the detection confidence scores of the learning-based methods (e.g., FasterRCNN) or similarity scores in the non-learned methods (e.g., SAM+DINO<sub>f</sub>). AR<sub>s</sub>, AR<sub>m</sub>, and AR<sub>l</sub> are breakdowns of AR for small, medium and large testing object instances. Results show that (1) the non-learned methods that use SAM generally recall more instances than others, and (2) all methods suffer from small instances. In sum, results show that methods yielding higher recall achieve higher AP metrics (cf. Table 2).

	AR@max10	AR@max100	AR <sub>s</sub> @max100	AR <sub>m</sub> @max100	AR <sub>l</sub> @max100
FasterRCNN [39]	26.24	39.24	14.83	44.87	60.05
RetinaNet [31]	26.33	49.38	22.04	56.76	69.69
CenterNet [51]	23.55	44.72	17.84	52.03	64.58
FCOS [47]	25.82	46.28	22.09	52.85	64.11
DINO [50]	29.84	54.22	<b>32.00</b>	59.43	72.92
SAM + DINO <sub>f</sub>	31.25	63.05	31.65	70.01	<b>90.63</b>
SAM + DINOv2 <sub>f</sub>	<b>40.02</b>	<b>63.06</b>	31.11	<b>70.40</b>	90.36

(41.61% AP) vs. DINO (27.99% AP). Third, all the methods perform poorly on *hard* and *small* instances, suggesting future work focusing on such cases.

Table 3 compares methods w.r.t the average recall (AR) metric. “AR@max10” means AR within the top-10 ranked detections. In computing AR, we rank detections by using the detection confidence scores of the learning-based methods (e.g., FasterRCNN) or similarity scores in the non-learned methods (e.g., SAM+DINO<sub>f</sub>). AR<sub>s</sub>, AR<sub>m</sub>, and AR<sub>l</sub> are breakdowns of AR for small, medium, and large testing object instances. Results show that (1) the non-learned methods that use SAM generally recall more instances than others, and (2) all methods suffer from small instances. In sum, results show that methods yielding higher recall achieve higher AP metrics (cf. Table 2).

**Qualitative results.** Fig. 6 visualizes qualitative results on two testing examples from the InsDet dataset. Stronger detectors, e.g., the non-learned method SAM+DINOv2<sub>f</sub>, produce fewer false negatives. Even so, all detectors still struggle to detect instances with presented barriers such as heavy occlusion, instance size being too small, etc. As shown in Fig. 5, the non-learned method SAM+DINOv2<sub>f</sub> outperforms end-to-end learned methods in a wide range of recall thresholds.

## 5.2 Ablation Study

Due to the space limit, we ablate the instance crop and stable matching in the main paper and put more (including ablation studies for the cut-paste-learn methods) in the supplement.

**Proposal feature extraction in the non-learned method.** Given a box crop (encapsulating the proposal) generated by SAM in the non-learned method, we study how to process the crop to improve InsDet performance. Here, we can either crop and feed its minimum bounding box to compute DINOv2<sub>f</sub> features, or we can use the mask to remove the background in the box. Table 4 shows the comparison. Clearly, the latter performs remarkably better in both “hard” and “easy” scenarios.

**Proposal-instance match in the non-learned method.** After generating proposals by SAM, we need to compare them with instance profile images to get the final detection results. We study the

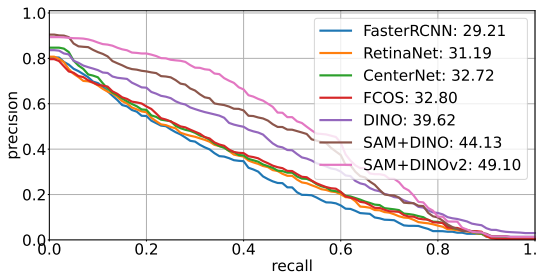


Figure 5: Precision-recall curves with IoU=0.5 (AP50 in the legend) on our InsDet dataset. Stronger detectors perform better, e.g., DINO, a transformer-based detector significantly outperforms FasterRCNN. Furthermore, even with a simple non-learned method, leveraging pretrained models, e.g., SAM+DINOv<sub>2f</sub>, outperforms end-to-end learned methods.

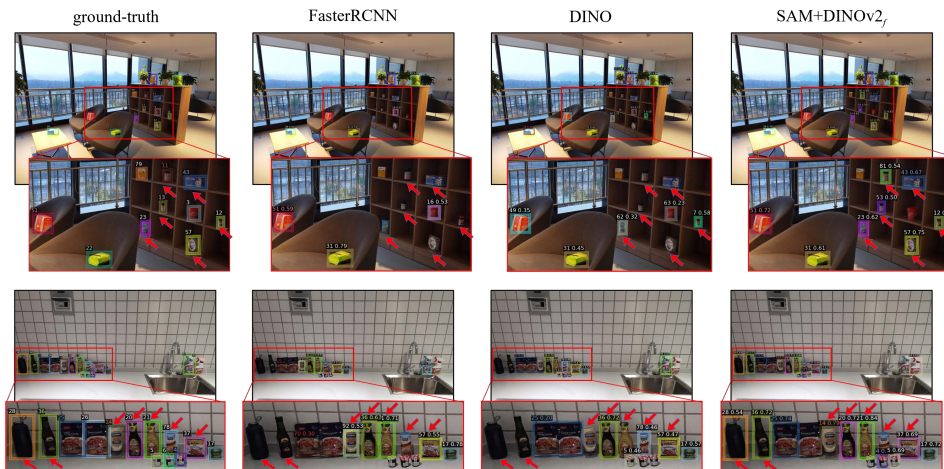


Figure 6: Visual results of FasterRCNN, DINO, and SAM+DINOv<sub>2f</sub> on our InsDet dataset. The top row illustrates the sparse placement of instances (i.e., easy scenario), while the bottom contains more cluttered instances (i.e., hard scenario). We drop predicted instance names for brevity. SAM helps localize instances with more precise bounding boxes, e.g., as arrows labeled in the upper row. DINOv<sub>2f</sub> provides more precise recognition of localized instances, e.g., five instances in the right of the bottom row. Compared with DINO, SAM+DINOv<sub>2f</sub> is better at locating occluded instances.

InsDet performance of the two matching algorithms. Rank & Select is a greedy algorithm that iteratively finds the best match between any proposals and instances until no instances/proposals are left unmatched; stable matching produces an optimal list of matched proposals and instances such that there does not exist a pair in which both prefer other proposals/instances to their current correspondence under the matching. Table 5 compares these two methods, clearly showing that stable matching works better.

### 5.3 Discussions

**Societal Impact.** InsDet is a crucial component in various robotic applications such as elderly-assistive agents. Hence, releasing a unified benchmarking protocol contributes to broader communities. While our dataset enables InsDet research to move forward, similar to other works, directly applying algorithms brought by our dataset is risky in real-world applications.

**Limitations.** We note several limitations in our current work. First, while our work uses normal cameras to collect datasets, we expect to use better and cheaper hardware (e.g., depth camera and IMU) for data collection. Second, while the cut-paste-learn method we adopt does not consider geometric cues when synthesizing training images, we hope to incorporate such information to generate better and more realistic training images, e.g., pasting instances only on up-surfaces like tables, desks, and floors. Third, while SAM+DINOv<sub>2f</sub> performs the best, this method is time-consuming (see a run-time study in the supplement); real-world applications should consider real-time requirements.

**Future work.** In view of the above limitations, the future work includes: (1) Exploring high-resolution images for more precise detection on *hard* situations, e.g., one can combine proposals generated from multi-scale and multi-resolution images. (2) Developing faster algorithms, e.g., one can use multi-scale detectors to attend to regions of interest for progressive detection. (3) Bridging



Table 4: **Ablation study: whether to remove background in crops for feature computation.** Based on a proposal given by SAM, we can crop and feed its minimum bounding square to compute DINOv2<sub>f</sub> feature, or we can use the mask to remove the background in the square before computing the feature. Clearly, the latter performs remarkably better.

strategy	AP			AP <sub>50</sub>			AP <sub>75</sub>		
	avg	hard	easy	avg	hard	easy	avg	hard	easy
w/o background removal	36.04	23.04	42.37	43.84	29.12	51.00	39.59	25.74	46.13
w/ background removal	39.12	24.00	47.17	46.72	30.81	54.66	42.86	26.40	51.58

Table 5: **Ablation study: whether to generate unique proposal-instance match.** In contrast to Rank&Select, Stable Matching produces a unique match to proposal/instance for each instance/proposal, yielding better performance than Rank&Select.

strategy	AP			AP <sub>50</sub>			AP <sub>75</sub>		
	avg	hard	easy	avg	hard	easy	avg	hard	easy
Rank & Select	38.62	23.95	46.31	46.04	30.77	53.64	42.37	26.39	50.61
Stable Matching	39.12	24.00	47.17	46.72	30.81	54.66	42.86	26.40	51.58

end-to-end fast models and powerful yet slow pretrained models, e.g., one can train lightweight adaptors atop pretrained models for better InsDet.

## 6 Conclusion

We explore the problem of Instance Detection (InsDet) by introducing a new dataset consisting of high-resolution images and formulating a realistic unified protocol. We revisit representative InsDet methods in the cut-paste-learn framework and design a non-learned method by leveraging publicly-available pretrained models. Extensive experiments show that the non-learned method significantly outperforms end-to-end InsDet models. Yet, the non-learned method is slow because running large pretrained models takes more time than end-to-end trained models. Moreover, all methods struggle in hard situations (e.g., in front of heavy occlusions and a high level of clutter in the scene). This shows that our dataset serves as a challenging venue for the community to study InsDet.

## References

- [1] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Kosecka, and Alexander C. Berg. A dataset for developing and benchmarking active vision. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [2] Phil Ammirato, Cheng-Yang Fu, Mykhailo Shvets, Jana Kosecka, and Alexander C Berg. Target driven instance detection. *arXiv:1803.04610*, 2018.
- [3] Siddharth Ancha, Junyu Nan, and David Held. Combining deep learning and verification for precise object instance detection. *arXiv:1912.12270*, 2019.
- [4] Richard Bormann, Xinjie Wang, Markus Völk, Kilian Kleeberger, and Jochen Lindermayr. Real-time instance detection with fast incremental learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [5] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, 2014.
- [6] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.
- [7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.
- [8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

- [9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- [11] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR, 2014.
- [12] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *ICCV*, 2017.
- [13] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.
- [14] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.
- [15] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.
- [16] Georgios Georgakis, Md. Alimoor Reza, Arsalan Mousavian, Phi Hung Le, and Jana Kosecka. Multiview rgb-d dataset for object instance detection. *International Conference on 3D Vision (3DV)*, 2016.
- [17] Georgios Georgakis, Md Alimoor Reza, Arsalan Mousavian, Phi-Hung Le, and Jana Kovsecká. Multiview rgb-d dataset for object instance detection. In *International Conference on 3D Vision (3DV)*, 2016.
- [18] Georgios Georgakis, Arsalan Mousavian, Alexander C Berg, and Jana Kosecka. Synthesizing training data for object detection in indoor scenes. *Robotics: Science and Systems (RSS)*, 2017.
- [19] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.
- [20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.
- [21] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE transactions on pattern analysis and machine intelligence*, 34(5):876–888, 2011.
- [22] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary R. Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian Conference on Computer Vision*, 2012.
- [23] Tomávs Hodavn, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta N Sinha, and Brian Guenter. Photorealistic image synthesis for object instance detection. In *IEEE international conference on image processing (ICIP)*, 2019.
- [24] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *ICCV*, 2017.
- [25] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv:2304.02643*, 2023.
- [26] Ikki Kishida, Hong Chen, Masaki Baba, Jiren Jin, Ayako Amma, and Hideki Nakayama. Object recognition with continual open set domain adaptation for home robot. In *WACV*, 2021.

- [27] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.
- [28] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling. *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
- [29] Hengduo Li, Bharat Singh, Mahyar Najibi, Zuxuan Wu, and Larry S Davis. An analysis of pre-training on object detection. *arXiv:1904.05871*, 2019.
- [30] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
- [31] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.
- [32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.
- [33] David G McVitie and Leslie B Wilson. The stable marriage problem. *Communications of the ACM*, 14(7):486–490, 1971.
- [34] Jean-Philippe Mercier, Mathieu Garon, Philippe Giguere, and Jean-Francois Lalonde. Deep template-based object instance detection. In *WACV*, 2021.
- [35] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv:2304.07193*, 2023.
- [36] A Quadros, James Patrick Underwood, and Bertrand Douillard. An occlusion-aware feature for range images. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.
- [37] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*, 2018.
- [38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.
- [39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.
- [40] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgb-d-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016.
- [41] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.
- [42] Neil Savage et al. Robots rise to meet the challenge of caring for old people. *Nature*, 601(7893): 8–10, 2022.
- [43] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016.
- [44] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshops*, 2014.
- [45] Arjun Singh, James Sha, Karthik S. Narayan, Tudor Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

- [46] Zhi Tian, Hao Chen, Xinlong Wang, Yuliang Liu, and Chunhua Shen. AdelaiDet: A toolbox for instance-level recognition tasks. <https://git.io/adelaidet>, 2019.
- [47] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019.
- [48] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv:2207.02696*, 2022.
- [49] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [50] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Harry Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *International Conference on Learning Representations*, 2022.
- [51] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv:1904.07850*, 2019.
- [52] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Probabilistic two-stage detection. In *arXiv:2103.07461*, 2021.
- [53] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv:2010.04159*, 2020.

## Outline

This document supplements the main paper with more experimental results, more visualizations, further details of the InsDet dataset, and open-source code. Below is the outline of this document.

- **Section A.** We provide demo code for the non-learned method using Jupyter Notebook.
- **Section B.** We visualize object instance profile images captured in multiple views, instance proposals produced by SAM, and input crops to DINO<sub>f</sub> / DINOv2<sub>f</sub>.
- **Section C.** We display how to tag object instances as “small”, “medium” and “large”, and supplement the average recall of proposals under both “easy” and “hard” scenes.
- **Section D.** We conduct extensive ablation studies on the non-learned InsDet method and the traditional cut-paste-learn method.
- **Section E** We demonstrate runtime comparison between end-to-end learned detectors and non-learned detectors, and discuss their trade-off.
- **Section F** includes dataset documentation and intended uses.
- **Appendix** contains further details of dataset collection.

## A Open-Source Code

We release open-source code in the form of Jupyter Notebook plus Python files.

**Why Jupyter Notebook?** We prefer to release the code using Jupyter Notebook (<https://jupyter.org>) because it allows for interactive demonstration for education purposes. In case the reader would like to run Python script, using the following command can convert a Jupyter Notebook file `xxx.ipynb` into a Python script file `xxx.py`:

```
jupyter nbconvert --to script xxx.ipynb
```

**Requirement.** Running our code requires some common packages. We installed Python and most packages through Anaconda. A few other packages might not be installed automatically, such as Pandas, torchvision, and PyTorch, which are required to run our code. Below are the versions of Python and PyTorch used in our work.

- Python version: 3.9.16 [GCC 7.5.0]
- PyTorch version: 2.0.0

We suggest assigning >30GB space to run all the files.

**License.** We release open-source code under the MIT License to foster future research in this field.

**Demo.** The Jupyter notebook files below demonstrate our non-learned method using SAM and DINOv2<sub>f</sub>. The masked instances generated by SAM are encapsulated by a minimum bounding square box and then cropped from the high-resolution testing image. We feed these proposals into DINOv2<sub>f</sub> for feature representation, just like how we represent profile images. We run Rank & Select and Stable Matching to determine “well-matched” proposals and instances.

- `demo_get_proposals.ipynb`  
Running this file crops a masked instance from the high-resolution testing image with/without background, and visualizes the cropped proposal regions on the testing image.
- `demo_eval_instance_detection_Stable_Matching.ipynb`  
Running this file extracts features of proposals and object instance profile images by DINOv2<sub>f</sub>, and implements Stable Matching to return matched proposals and instances.
- `demo_eval_instance_detection_Rank_Select.ipynb`  
Running this file extracts features of proposals and object instance profile images by DINOv2<sub>f</sub>, and implements Rank & Select to return matched proposals and instances.

## B More Visualizations

**Instance Profile Images.** Fig. 7 presents more visualizations of object instance profile images in our InsDet dataset. The InsDet dataset contains 100 object instances, including sauces, snack foods,

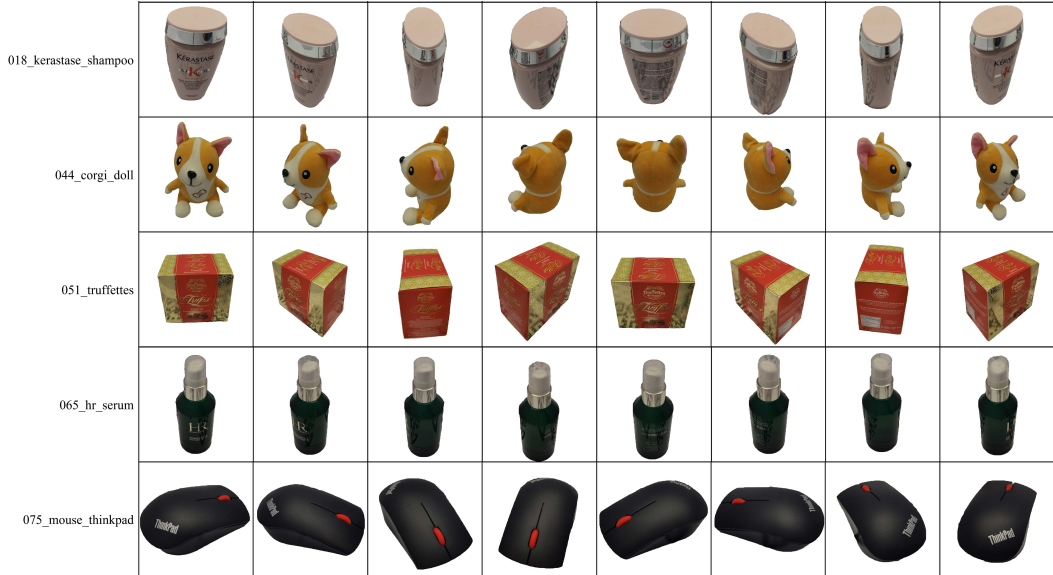


Figure 7: Examples of object instance profile images. We demonstrate profile images of five random object instances (after background removal using GrabCut).

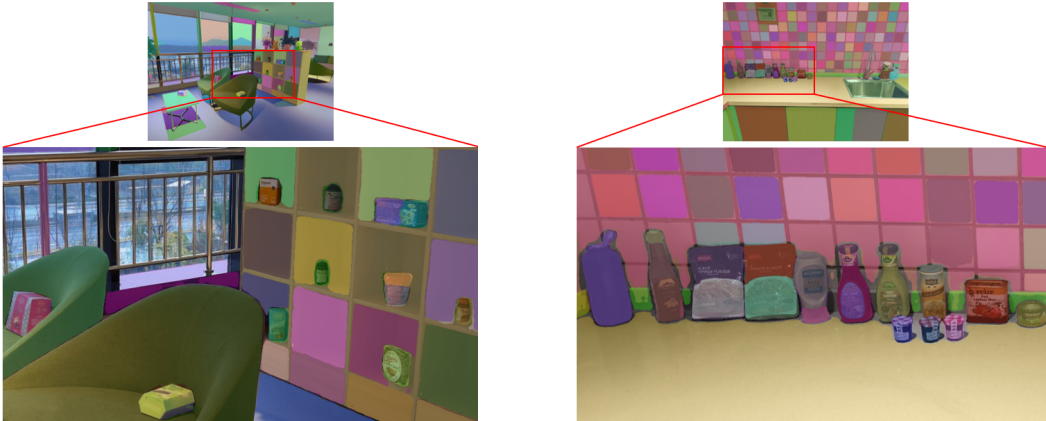


Figure 8: Instance proposals produced by SAM overlaid on the testing images. Note that at each location seed, SAM produces three proposals which might be object parts or the whole object. We use all of them as proposal detections and let the follow-up step of Stable Matching find the best-matched proposals to instances as the final InsDet results.

office stationery, cosmetics, toiletries, dolls, etc. Each instance is captured at 24 rotation positions (every  $15^\circ$  in azimuth) with  $45^\circ$  elevation view. Profile images are captured at  $3072 \times 3072$  pixel resolution (some are  $3456 \times 3456$ ). We use the GrabCut [41] toolbox to derive foreground masks of instances in profile images. This removes background pixels (such as QR code regions) in the profile images. In practice, we center-crop foreground instances from profile images and downsize the center-crops to  $1024 \times 1024$ . As shown in Fig. 7, we visualize object instances of various shapes and sizes in multiple rotation views.

**Instance Proposals Produced by SAM.** Fig. 8 shows the segmentation masks produced by SAM on two testing scene images (demonstrated on page 8 of the paper). We downsize the high-resolution testing images to low-resolution (e.g.,  $768 \times 1024$  or  $1536 \times 2048$ ) for more efficient and effective segmentation. We observe that SAM can produce high-quality segments, although it also over-segments. Note that at each location seed, SAM produces three proposals which might be object parts or the whole object. We use all of them as proposed detections and let the follow-up step of Stable Matching find the best-matched proposals to instances as the final InsDet results.



Figure 9: Examples of input proposals to DINOv2<sub>f</sub>. Based on a proposal given by SAM, we crop the instance using the minimum bounding square ((a) w/ background) or using the segmentation mask ((b) w/o background) before computing the features.

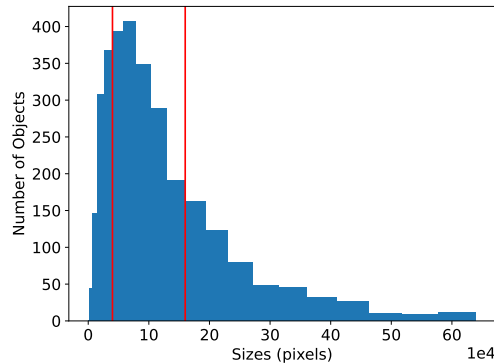


Figure 10: Distribution of objects w.r.t their bounding box area in testing images. We split them into small, medium, and large subgroups to allow breakdown analysis.

**Input Proposals to DINOv2<sub>f</sub>.** Fig. 9 presents two strategies of feeding proposals to DINOv2<sub>f</sub> for feature computation. One is to crop an instance proposal with its minimum bounding square which keeps the background from the testing image. Another is to use such with background removal (via the mask generated by SAM).

## C Dataset Statistics

In addition to tag testing images by *hard* and *easy* according to the level of scene clutter and object occlusion, we further tag testing instances based on their size. Specially, following the spirit of COCO dataset, we tag instances with *small*, *medium*, and *large*, as in Table 6. To determine their size tags, we plot the distribution of their sizes in Fig. 10, showing an intuitive way to tag them.

As a supplement to Table 3 in the main paper showing the average recall (AR) of *small*, *medium* and *large*, Table 7 further studies AR in *hard* and *easy* scenes. We can observe that: (1) the non-learned methods that use SAM recall more proposals than other competitors in both *easy* and *hard* scenes; (2) all methods basically suffer from *hard* scenes.

## D Ablation Study

We conduct more ablation studies on the non-learned InsDet method and the traditional cut-paste-learn method. In the cut-paste-learn group of methods, there are four factors influencing the final detection results, i.e., the number of objects inserted per image, the scales of inserted object instances, the blending methods when pasting instances on background images, and the total amount of synthesized training images. We ablate these factors by using the FasterRCNN architecture.

Table 6: Following the spirit of COCO dataset, we tag objects with different sizes by `small`, `medium`, and `large`, respectively.

size	bounding box area
small	$< 200^2$
medium	$200^2 - 400^2$
large	$> 400^2$

Table 7: **Benchmarking results w.r.t average recall (AR).** We added a breakdown analysis of testing images on `hard` and `easy` scenes. Results show that (1) the non-learned methods that use SAM generally recall more instances than others, and (2) all methods suffer from hard scenes.

	AR@max10			AR@max100		
	avg	hard	easy	avg	hard	easy
FasterRCNN [39]	26.24	12.92	32.33	39.24	16.91	49.43
RetinaNet [31]	26.33	15.38	31.33	49.38	29.00	58.69
CenterNet [51]	23.55	11.87	28.87	44.72	24.88	53.76
FCOS [47]	25.82	12.81	31.74	46.28	26.55	55.27
DINO [50]	29.84	16.63	35.84	54.22	36.46	62.30
SAM + DINO <sub>f</sub>	31.25	16.96	37.73	63.05	42.46	<b>72.41</b>
SAM + DINOv2 <sub>f</sub>	<b>40.02</b>	<b>27.64</b>	<b>45.36</b>	<b>63.06</b>	<b>43.47</b>	71.96

**Impact of different image/proposal resolutions.** We study the InsDet performance when using images of different sizes for SAM, and using different resolutions of crops fed into DINOv2<sub>f</sub>. For example, we can use SAM on images of size  $3072 \times 4096$  to generate proposals, we can resize crops of proposals to  $224 \times 224$  to feed into DINOv2<sub>f</sub> if they are larger than  $224 \times 224$  (otherwise, keep them unchanged). Table 8 lists detailed comparisons. We have two observations. (1) The InsDet performance generally increases with the image resolution but starts to drop when the input image is too large, i.e.,  $6144 \times 8192$ . This is because SAM tends to produce object parts as individual instances, resulting in more false positives. (2) When using larger proposals for DINOv2<sub>f</sub>, InsDet performance gets better, e.g., 41.61% ( $448 \times 448$ ) vs. 39.12% ( $224 \times 224$ ).

Table 8: **Ablation study: which image/proposal size to use for SAM and DINOv2<sub>f</sub>.** We notice that InsDet performance generally increases with the input image resolution, but starts to drop when the image is too large. When using larger proposals for DINOv2<sub>f</sub>, InsDet performance also gets better.

image resolution for SAM	input size for DINOv2 <sub>f</sub>	AP			AP <sub>50</sub>			AP <sub>75</sub>		
		avg	hard	easy	avg	hard	easy	avg	hard	easy
768×1024	224×224	36.46	21.67	43.88	46.22	29.52	54.11	41.53	24.95	49.99
1536×2048		39.12	24.00	47.17	46.72	30.81	54.66	42.86	26.40	51.58
3072×4096		39.17	24.08	46.60	45.71	30.34	53.07	41.90	26.12	49.71
6144×8192		38.74	23.39	46.29	45.24	29.24	52.78	40.81	25.14	48.65
1536×2048	112×112	26.46	16.52	31.32	30.83	20.94	36.26	28.89	18.81	33.73
	224×224	39.12	24.00	47.17	46.72	30.81	54.66	42.86	26.40	51.58
	448×448	41.61	28.03	47.57	49.10	36.64	54.84	45.95	31.41	52.03

**Number of objects inserted in each background image.** We study InsDet performance with different numbers of objects inserted in each background image in Table 9. We can see that inserting more objects helps train InsDet detectors and achieves better performance. Concretely, FasterRCNN yields 19.54% AP when trained on synthesized training images each of which has 25-35 object instances, better than 17.57% AP when trained on those that have 5-15 object instances per image. But inserting more is not necessarily increasing much further.

**Scales of inserted object instances in synthesizing training images.** We study the impact of the scales of inserted object instances when synthesizing training images in Table 10. The number in square brackets denotes the range of downsampling factors for instance profile images. For example, [0.1, 0.15] denotes that the original instance profile images (256x256 resolution) are randomly scaled by 0.1-0.15 before being pasted on background images. We can see that the scale significantly influences the final detection performance. For example, inserting objects that are too small (e.g. [0.1,



Table 9: **Ablation study: number of objects inserted in each background image.** Basically, inserting more objects helps train InsDet detectors and achieves better performance. Yet, inserting more is not necessarily increasing much further.

# of objects	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>l</sub>
[5, 15]	17.57	25.98	20.49	3.55	20.31	33.50
[15, 25]	18.20	27.76	21.09	4.45	20.66	35.51
[25, 35]	19.39	29.14	<b>23.09</b>	5.03	22.04	37.73
[35, 45]	<b>19.60</b>	<b>30.30</b>	22.82	<b>5.44</b>	<b>22.32</b>	<b>39.17</b>

Table 10: **Ablation study: scales of inserted object instances in synthesizing training images.** The scale significantly influences the final detection performance. Inserting objects that are too small (e.g. [0.1, 0.15]) or too large (e.g. [0.5, 1.0]) will not train detectors well. We think this is because the testing images contain more “medium” object instances.

scale of objects	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>l</sub>
[0.1, 0.15]	4.72	9.63	4.16	5.48	8.93	0.72
[0.15, 0.3]	16.66	26.82	18.55	<b>16.01</b>	<b>27.74</b>	9.88
[0.15, 0.5]	<b>19.39</b>	<b>29.14</b>	<b>23.09</b>	5.03	22.04	37.73
[0.5, 0.8]	5.43	8.16	6.08	1.79	18.72	<b>70.20</b>
[0.5, 1.0]	5.74	9.15	6.60	0.00	3.00	19.58

Table 11: **Ablation study: blending methods between objects and the background images.** We note that: (1) Naive pasting gives the worst performance, since directly pasting objects on background images creates boundary artifacts. (2) Although the other three blending modes do not yield visually perfect results, they could still improve the detection performance. (3) When using all four blending modes when mixing the same background image with the same object displacement, the InsDet performance could be further improved. This is because training on multiple images makes the algorithm less sensitive to these blending factors.

blending strategy	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>l</sub>
Gaussian	17.83	27.13	21.02	4.74	20.49	36.09
motion	17.92	27.57	20.85	4.69	20.76	34.78
box blurring	17.71	27.47	20.95	4.25	20.30	34.56
naive pasting	16.53	24.77	20.17	4.25	19.07	34.86
all	<b>19.39</b>	<b>29.14</b>	<b>23.09</b>	<b>5.03</b>	<b>22.04</b>	<b>37.73</b>

0.15]) or too large (e.g. [0.5, 1.0]) will not train detectors well. We conjecture this is because the testing images contain more “medium” object instances.

**Blending methods between objects and background.** We study four commonly-used blending methods when pasting objects into the background images in Table 11, i.e. Gaussian blurring, motion blurring, box blurring, and naive pasting. We note that: (1) naive pasting yields the worst performance, since this creates boundary artifacts; (2) the other three blending methods work better than naive pasting but do not show significant performance difference; (3) using all the four blending methods together leads to the best performance, significantly better than using any one of them alone.

**Total amount of synthesized training images.** We study InsDet performance by training on different amounts of synthesized images in Table 12. Perhaps surprisingly, using 5k synthesized training images is better than training on more images! We conjecture the reasons are that (1) more synthesized images do not bring new signals to help training, (2) domain gaps between synthesized images and real testing images are difficult to close by simply using more such synthetic data, otherwise, training will overfit to them and hence hurt the final InsDet performance.

## E Runtime Comparison

Table 13 compares the runtime of different methods, along with their InsDet performance. There is a trade-off between runtime and detection precision. For example, among the methods studied in our work, SAM+DINOv2<sub>f</sub> achieves the highest AP (41.61) but is four orders of magnitude slower than FasterRCNN (19.54 AP). Developing faster and better InsDet methods is future work.

Table 12: **Ablation study: different amounts of synthesized training images.** Perhaps surprisingly, using 5k synthesized training images is better than training on more images! We conjecture the reasons are that (1) more synthesized images do not bring new signals to help training, (2) domain gaps between synthesized images and real testing images are difficult to close by simply using more such synthetic data, otherwise, training will overfit to them and hence hurt the final InsDet performance.

# of training images	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>l</sub>
5k	<b>19.93</b>	<b>30.60</b>	<b>23.21</b>	<b>5.62</b>	22.09	<b>38.92</b>
10k	19.08	29.50	21.91	4.67	21.65	37.21
20k	19.39	29.14	23.09	5.03	22.04	37.73
25k	19.19	29.13	22.33	4.46	<b>22.21</b>	36.92
30k	18.42	28.11	21.42	4.38	21.19	36.70

method	time (sec)	AP (%)
FasterRCNN [39]	0.00399	19.54
RetinaNet [31]	0.00412	22.22
CenterNet [51]	0.00376	21.12
FCOS [47]	0.00271	22.40
DINO [50]	1.90625	27.99
SAM + DINO <sub>f</sub>	15.10	36.97
SAM + DINOv2 <sub>f</sub>	14.70	41.61

Table 13: We compare the inference runtime (second/image) of different methods, along with their InsDet performance in AP (%). Clearly, there is a trade-off between runtime and detection precision. For example, among the methods studied in our work, SAM+DINOv2<sub>f</sub> achieves the highest AP (41.61) but is four orders of magnitude slower than FasterRCNN (19.54 AP). Developing faster and better InsDet methods is apparently future work.

## F Datasheet for our Dataset

We follow the datasheet proposed in [15] for documenting our InsDet dataset.

### 1. Motivation

- (a) For what purpose was the dataset created?  
This dataset was created to study the problem of Instance Detection, i.e., detecting individual object instances from every single-image of cluttered scenes.
- (b) Who created the dataset and on behalf of which entity?  
This dataset was mainly created by Qianqian Shen. Other authors help with logistics.
- (c) Who funded the creation of the dataset?  
[N/A]
- (d) Any other Comments?  
[No]

### 2. Composition

- (a) What do the instances that comprise the dataset represent?  
RGB images captured by a camera, and annotation files.
- (b) How many instances are there in total?  
There are 2,760 instances including 24\*100 profile images (100 objects with each of which having 24 profile images), 200 background images, and 160 testing images.
- (c) Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set?  
It contains all possible instances.
- (d) What data does each instance consist of?  
See 2.(a).
- (e) Is there a label or target associated with each instance?  
See 2.(a)
- (f) Is any information missing from individual instances?  
[No]
- (g) Are relationships between individual instances made explicit?  
[Yes] Images and annotations files are associated.
- (h) Are there recommended data splits?  
[Yes] We provide training and testing split for the dataset.

- (i) Are there any errors, sources of noise, or redundancies in the dataset?  
[No] We tried our best to manually annotate bounding boxes of object instances on the testing image, and we did not see visible noise or errors. In profile images of instances, there might be noise in camera pose estimation due to the QR code pasted on the table.
- (j) Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)?  
[Yes]
- (k) Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor-patient confidentiality, data that includes the content of individuals' non-public communications)?  
[No]
- (l) Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?  
[No]
- (m) Does the dataset relate to people?  
[No]
- (n) Does the dataset identify any subpopulations (e.g., by age, gender)?  
[No]
- (o) Is it possible to identify individuals (i.e., one or more natural persons), either directly or indirectly (i.e., in combination with other data) from the dataset?  
[No]
- (p) Does the dataset contain data that might be considered sensitive in any way (e.g., data that reveals racial or ethnic origins, sexual orientations, religious beliefs, political opinions or union memberships, or locations; financial or health data; biometric or genetic data; forms of government identification, such as social security numbers; criminal history)?  
[No]
- (q) Any other comments?  
[No]

### 3. Collection Process

- (a) How was the data associated with each instance acquired?  
For each instance, we capture 24 profile images. For each testing image, we manually annotate bounding boxes on the instances of interest as the ground-truth.
- (b) What mechanisms or procedures were used to collect the data (e.g., hardware apparatus or sensor, manual human curation, software program, software API)?  
We use a single Leica camera (embedded in a cellphone) to capture profile images and testing scene images. We use GrabCut to obtain foreground masks on the profile images. We manually draw bounding boxes on testing images as the ground-truth.
- (c) If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic, probabilistic with specific sampling probabilities)?  
[N/A]
- (d) Who was involved in the data collection process (e.g., students, crowdworkers, contractors), and how were they compensated (e.g., how much were crowdworkers paid)?  
Only authors are involved in the data collection.
- (e) Over what timeframe was the data collected?  
All images were collected between October 2022 to March 2023.
- (f) Were any ethical review processes conducted (e.g., by an institutional review board)?  
[No] No ethical review processes were conducted with respect to the collection and annotation of this data.
- (g) Does the dataset relate to people?  
[No]
- (h) Did you collect the data from the individuals in question directly, or obtain it via third parties or other sources (e.g., websites)?  
We collected the data by ourselves.
- (i) Were the individuals in question notified about the data collection?  
[Yes]

- (j) Did the individuals in question consent to the collection and use of their data?  
[Yes]
- (k) If consent was obtained, were the consenting individuals provided with a mechanism to revoke their consent in the future or for certain uses?  
[No]
- (l) Has an analysis of the potential impact of the dataset and its use on data subjects (e.g., a data protection impact analysis) been conducted?  
[N/A]
- (m) Any other comments?  
[No]

#### 4. Preprocessing, Cleaning and Labeling

- (a) Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)?  
[No] The only labeling activity is annotating bounding boxes for instances on testing images. Further annotation cleaning is not needed.
- (b) Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)?  
[Yes] We release the raw data on the website of this work.
- (c) Is the software used to preprocess/clean/label the instances available?  
[Yes] We use open-source software publicly available. We cite them already in the main paper or supplement.
- (d) Any other comments?  
[No]

#### 5. Uses

- (a) Has the dataset been used for any tasks already?  
[No]
- (b) Is there a repository that links to any or all papers or systems that use the dataset?  
[No] We will add future papers that use this dataset in the website of this work.
- (c) What (other) tasks could the dataset be used for?  
The dataset can also be used to study small object detection, 3D reconstruction from sparse views, real-time object detection, etc.
- (d) Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?  
[Yes] Capturing high-resolution images is cheap but requires more computation resources in subsequent tasks. This might impact future work related to how to capture high-resolution images, how to trade off performance and resolution, etc.
- (e) Are there tasks for which the dataset should not be used?  
The usage of this dataset should be limited to the scope of object instance detection.
- (f) Any other comments?  
[No]

#### 6. Distribution

- (a) Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created?  
[Yes] We expect other websites re-distribute our dataset.
- (b) How will the dataset be distributed (e.g., tarball on website, API, GitHub)?  
The dataset could be accessed on a GitHub webpage.
- (c) When will the dataset be distributed?  
The dataset will be released to the public upon acceptance of this paper. We provide some demo data and visualizations for the review process.
- (d) Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?  
We release our benchmark under MIT license.

- (e) Have any third parties imposed IP-based or other restrictions on the data associated with the instances?  
[No]
- (f) Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?  
[No]
- (g) Any other comments?  
[No]

#### 7. Maintenance

- (a) Who is supporting/hosting/maintaining the dataset?  
Qianqian Shen will be responsible for maintaining the dataset.
- (b) How can the owner/curator/manager of the dataset be contacted (e.g., email address)?  
E-mail addresses are at the top of the paper.
- (c) Is there an erratum?  
[No] When errors are discerned, we will announce erratum on our website.
- (d) Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?  
[Yes] We hope to expand this dataset with more instances and testing images.
- (e) If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were individuals in question told that their data would be retained for a fixed period of time and then deleted)?  
[N/A]
- (f) Will older versions of the dataset continue to be supported/hosted/maintained?  
[Yes]
- (g) If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?  
[No]
- (h) Any other comments?  
[No]