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

Qianqian Shen¹ **Yuhan Zhao²** **Nahyun Kwon³** **Jeeeun Kim³** **Yanan Li¹** **Shu Kong³**
¹Zhejiang Lab ²UC-Irvine ³Texas A&M University

Dataset and open-source code in webpage

Abstract

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

20

1 Introduction

21 Instance detection (InsDet) requires detecting specific object instances (defined by some visual
22 examples) from a scene image [12]. It is practically important in robotics, e.g., elderly-assistant
23 robots need to fetch specific items (*my-cup* vs. *your-cup*) from a cluttered kitchen [41], micro-
24 fulfillment robots for the retail need to pick items from mixed boxes or shelves [4].

25 **Motivation.** InsDet receives much less attention than the related problem of Object Detection
26 (**ObjDet**), which aims to detect all objects belonging to some predefined classes [29, 38, 30, 49].
27 Fig. 1 compares the two problems. *One major reason is that there are not large-enough InsDet*
28 *datasets by today’s standards.* For example, the popular InsDet dataset GMU (published in 2016) [15]
29 has only 23 object instances while the popular ObjDet dataset COCO has 80 object classes (published
30 in 2014) [29]. Moreover, *there are no unified protocols in the literature of InsDet.* The current InsDet
31 literature mixes multiple datasets to simulate training images and testing scenarios [12]. Note that the
32 training protocol of InsDet does not follow that of ObjDet, which has training images annotated with

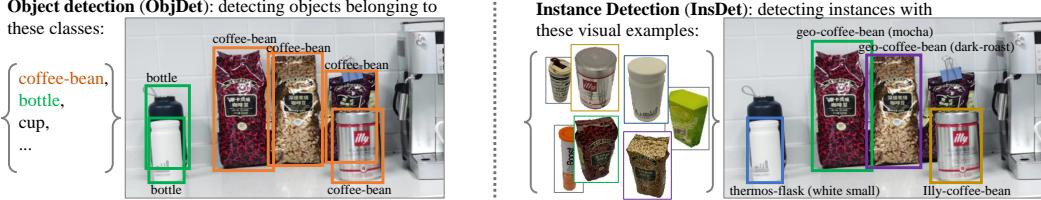


Figure 1: **Object detection (ObjDet) vs. instance detection (InsDet)**. ObjDet aims to detect all objects belonging to some predefined classes, whereas InsDet requires detecting specific object instances defined by some visual examples. Loosely speaking, InsDet treats a single object instance as a class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

33 bounding boxes. Differently, for InsDet,¹ its setup should have profile images of instances (cf. right
 34 in Fig. 1) and optionally diverse background images not containing such instances [12]. We release a
 35 new dataset and present a unified protocol to foster the InsDet research.

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

43 **Preview of technical insights.** On our dataset, we revisit existing InsDet methods [27, 12, 17].
 44 Perhaps the only InsDet framework is cut-paste-learn [12], which cuts instances from their profile
 45 images, pastes them on random background images (so being able to derive “free” bounding boxes
 46 annotations), and trains InsDet detectors on such data by following that of ObjDet (e.g., Faster-
 47 RCNN [38]). We study this framework, train different detectors, and confirm that the state-of-the-art
 48 transformer-based detector DINO [49] performs the best, achieving 27.99 AP, significantly better
 49 than CNN-based detector FasterRCNN (19.52 AP). Further, we present a non-learned method that
 50 runs off-the-shelf proposal detectors (SAM [24] in our work) to generate object proposals and use
 51 self-supervised learned features ($DINO_f$ [8]² and $DINOv2_f$ [34]) to find matched proposals to
 52 instances’ profile images. Perhaps surprisingly, this non-learned method resoundingly outperforms
 53 end-to-end learning methods, i.e., SAM+ $DINOv2_f$ achieves 41.61 AP, much better than DINO (27.99
 54 AP) [49].

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

- 56 1. We formulate the InsDet problem with a unified protocol and release a challenging dataset
 57 consisting of both high-resolution profile images and high-res testing images.
 58 2. We conduct extensive experiments on our dataset and benchmark representative methods
 59 following the cut-paste-learn framework [12], showing that stronger detectors perform better.
 60 3. We present a non-learned method that uses an off-the-shelf proposal detector (i.e., SAM [24])
 61 to produce proposals, and self-supervised learned features (e.g., $DINOv2_f$ [34]) to find
 62 instances (which are well matched to their profile images). This simple method significantly
 63 outperforms the end-to-end InsDet models.

64 2 Related Work

65 **Instance Detection (InsDet)** is a long-lasting problem in computer vision and robotics [50, 12, 33, 3,
 66 16, 22, 4], referring to detecting specific object instances in a scene image. Traditional InsDet methods
 67 use keypoint matching [35] or template matching [20]; more recent ones train deep neural networks

¹In 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).

²We add subscript f to indicate that $DINO_f$ [8] is the self-supervised learned feature extractor; distinguishing it from a well-known object detector DINO [49].



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 [42]), just like other existing datasets [21, 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 [27, 12, 17] as used in the object detection literature [29]. **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.

68 to approach InsDet [33]. Some others focus on obtaining more training samples by rendering realistic
 69 instance examples [23, 22], data augmentation [12], and synthesizing training images by cutting
 70 instances as foregrounds and pasting them to background images [27, 12, 17]. Speaking of InsDet
 71 datasets, [15] collects scene images from 9 kitchen scenes with RGB-D cameras and defines 23
 72 instances of interest to annotate with 2D boxes on scene images; [22] creates 3D models of 29
 73 instances from 6 indoor scenes, and uses them to synthesize training and testing data; [4] creates 3D
 74 mesh models of 100 grocery store objects, renders 80 views of images for each instance, and uses
 75 them to synthesize training data.

76 As for benchmarking protocol of InsDet, [12] synthesizes training data from BigBird [44] and
 77 UW Scenes [26] and tests on the GMU dataset [15]; [22] trains on their in-house data and test on
 78 LM-O [5] and Rutgers APC [39] datasets. Moreover, some works require hardware-demanding
 79 setups [4], some synthesize both training and testing data [22, 27], while others mix existing datasets
 80 for benchmarking [12]. Given that the modern literature on InsDet lacks a unified benchmarking
 81 protocol (till now!), we introduce a more realistic unified protocol along with our InsDet dataset,
 82 allowing fairly benchmarking methods and fostering research of InsDet.

83 **Object Detection (ObjDet)** is a fundamental computer vision problem [13, 29, 38], requiring
 84 detecting all objects belonging to some predefined categories. The prevalent ObjDet detectors adopt
 85 convolutional neural networks (CNNs) as a backbone and a detector-head for proposal detection and
 86 classification, typically using bounding box regression and a softmax-classifier. Approaches can be
 87 grouped into two categories: one-stage detectors [37, 31, 36, 47] and two-stage detectors [18, 6].
 88 One-stage detectors predict candidate detection proposals using bounding boxes and labels at regular
 89 spatial positions over feature maps; two-stage detectors first produce detection proposals, then
 90 perform classification and bounding box regression for each proposal. Recently, the transformer-
 91 based detectors transcend CNN-based detectors [7, 52, 49], yielding much better performance on
 92 various ObjDet benchmarks. Different from ObjDet, InsDet requires distinguishing individual object
 93 instances within a class. Nevertheless, to approach InsDet, the common practice is to repurpose
 94 ObjDet detectors by treating unique instances as individual classes. We follow this practice and
 95 benchmark various ObjDet methods on our InsDet dataset.

96 **Pretrained Models.** Pretraining is an effective way to learn features from diverse data. For example,
 97 training on the large-scale ImageNet dataset for image classification [10], a neural network can
 98 serve as a powerful feature extractor for various vision tasks [11, 43]. Object detectors trained on
 99 the COCO dataset [29] can serve as a backbone allowing finetuning on a target domain to improve
 100 detection performance [28]. Such pretraining requires human annotations which can be costly.
 101 Therefore, self-supervised pretraining has attracted increasing attention and achieved remarkable
 102 progress [9, 19, 8, 34]. Moreover, the recent literature shows that pretraining on much larger-scale
 103 data can serve as a foundation model for being able to perform well across domains and tasks.
 104 For example, the Segment Anything Model (SAM) pretrains a class-agnostic proposal detector on

105 web-scale data and shows an impressive ability to detect and segment diverse objects in the wild [24].
106 In this work, with our high-res InsDet dataset, we explore a non-learned method by using publicly
107 available pretrained models. We show that such a simple method significantly outperforms end-to-end
108 learned InsDet detectors.

109 3 Instance Detection: Protocol and Dataset

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

112 3.1 The Protocol

113 Our InsDet protocol is motivated by real-world indoor robotic applications. In particular, we consider
114 the scenario that assistive robots must locate and recognize instances to fetch them in a cluttered
115 indoor scene [41], where InsDet is a crucial component. Realistically, for a given object instance,
116 the robots should see it only from a few views (*at the training stage*), and then accurately detect it
117 *in a distance in any scenes (at the testing stage)*. Therefore, we suggest the protocol specifying the
118 training and testing setups below. We refer the readers to Fig. 2 for an illustration of this protocol.

- 119 • **Training.** There are profile images of each instance captured at different views and diverse
120 background images. The background images can be used to synthesize training images with
121 free 2D-box annotations, as done by the cut-paste-learn methods [27, 12, 17].
- 122 • **Testing.** InsDet algorithms are required to precisely detect all predefined instances from
123 real-world images of cluttered scenes.

124 **Evaluation metrics.** The InsDet literature commonly uses average precision (AP) at IoU=0.5 [12, 2,
125 33]; others use different metrics, e.g., AP at IoU=0.75 [22], mean AP [3, 16], and F1 score [4]. As a
126 single metric appears to be insufficient to benchmark methods, we follow the literature of ObjDet
127 that uses multiple metrics altogether [29].

- 128 • **AP** averages the precision at IoU thresholds from 0.5 to 0.95 with the step size 0.05. It is
129 the *primary metric* in the most well-known COCO Object Detection dataset [29].
- 130 • **AP₅₀** and **AP₇₅** are the precision averaged over all instances with IoU threshold as 0.5 and
131 0.75, respectively. In particular, **AP₅₀** is the widely used metric in the literature of InsDet.
- 132 • **AR (average recall)** averages the proposal recall at IoU threshold from 0.5 to 1.0 with
133 the step size 0.05, regardless of the classification accuracy. AR measures the localization
134 performance (excluding classification accuracy) of an InsDet model.

135 Moreover, we tag *hard* and *easy* scenes in the testing images based on the level of clutter and
136 occlusion, as shown by the right panel of Fig. 2. **Following the COCO dataset [29]**, we further tag
137 **testing object instances as *small*, *medium*, and *large*** according to their bounding box area (cf. details
138 in the supplement). These tags allow a breakdown analysis to better analyze methods.

139 3.2 The Dataset

140 We introduce a challenging real-world dataset of in-
141 door scenes (**motivated by indoor assistive robots**),
142 including high-resolution photos of 100 distinct ob-
143 ject instances, **and high-resolution testing images cap-**
144 **tured from 14 indoor scenes where there are such 100**
145 **instances defined for InsDet.** Table 1 summarizes the
146 statistics compared with existing datasets, showing
147 that our dataset is larger in scale and more challeng-
148 ing than existing InsDet datasets. Importantly, object
149 instances are located far from the camera in cluttered
150 scenes; this is realistic because robots must detect



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 [44]	recognition	✓	100	N/A	2014	1280x1024
RGBD [27]	scene label.	✓	300	14	2017	N/A
LM [21]	6D pose est.	✓	15	1	2012	480x640
LM-O [5]	6D pose est.	✓	20	1	2017	480x640
RU-APC [39]	3D pose est.	✓	14	1	2016	480x640
GMU [15]	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

151 objects in a distance before approaching them [1]. Perhaps surprisingly, only a few InsDet datasets
152 exist in the literature. Among them, Grocery [4], which is the latest and has the most instances like
153 our dataset, is not publicly available.

154 Our InsDet dataset contains 100 object instances. When capturing photos for each instance, inspired
155 by prior arts [44, 21, 5], we paste a QR code on the tabletop, which enables pose estimation, e.g.,
156 using COLMAP [42]. Yet, we note more realistic scenarios can be hand-holding instances for
157 capturing [25], which we think of as future work. Each instance photo is of 3072×3072 pixel
158 resolution. For each instance, we capture 24 photos from multiple views. The left panel of Fig. 2
159 shows some random photos for some instances. For the testing set, we capture high-resolution images
160 (6144×8192) in cluttered scenes, where some instances are placed in reasonable locations, as shown
161 in the right panel of Fig. 2. We tag these images as *easy* or *hard* based on scene clutter and object
162 occlusion levels. When objects are placed sparsely, we tag the testing images as *easy*; otherwise,
163 we tag them as *hard*. Our InsDet dataset also contains 200 high-res background images of indoor
164 scenes (cf. Fig. 2-middle). These indoor scenes are not included in testing images. They allow using
165 the cut-paste-learn framework to synthesize training images [27, 12, 17]. Following this framework,
166 we segment foreground instances using GrabCut [40] to paste them on background images. It is
167 worth noting that the recent vision foundation model SAM [24] makes interactive segmentation much
168 more efficient. Yet, this work is made public after we collected our dataset. In Fig. 3, we plot the
169 per-instance frequency in the testing set.

170 4 Methodology

171 4.1 The Strong Baseline: Cut-Paste-Learn

172 **Cut-Paste-Learn** serves as a strong baseline that synthesizes training images with 2D-box anno-
173 tations [12]. This allows one to train InsDet detectors in the same way as training normal ObjDet
174 detectors, by simply treating the K unique instances as K distinct classes. It cuts and pastes fore-
175 ground instances at various aspect ratios and scales on diverse background images, yielding synthetic
176 training images, as shown in Fig. 4. Cut-paste-learn is model-agnostic, allowing one to adopt any
177 state-of-the-art detector architecture. In this work, we study five popular detectors, covering the two-
178 stage detector FasterRCNN [38], and one-stage anchor-based detector RetinaNet [30], and one-stage
179 anchor-free detectors CenterNet [50], and FCOS [46]; and the transformer-based detector DINO [49].
180 There are multiple factors in the cut-paste-learn framework, such as the number of inserted objects in
181 each background image, their relative size, the number of generated training images and blending
182 methods. We conduct comprehensive ablation studies and report results using the best-tuned choices.
183 We refer interested readers to the supplement for the ablation studies.

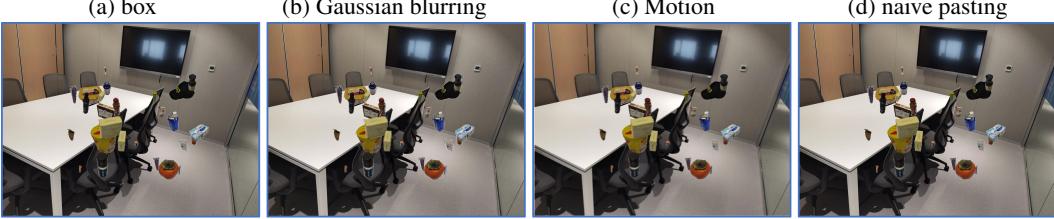


Figure 4: Synthetic training images for cut-paste-learn methods. We use different blending methods to paste object instances on the same background. We recommend that interested readers refer to the supplement for an ablation study using different blending methods.

184 4.2 The Simple, Non-Learned Method

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

188 **Proposal generation.** We use the recently released Segment Anything Model (SAM) [24] to generate
 189 proposals. For a proposal, we define a minimum bounding square box encapsulating the masked
 190 instance, and then crop the region from the high-resolution testing image. SAM not only achieves
 191 high recall (Table 3) on our InsDet dataset but detects objects not belonging to the instances of
 192 interest. So the next step is to find interested instances from the proposals.

193 **Feature representation of proposals and profile images.** Intuitively, among the pool of proposals,
 194 we are interested in those that are well-matched to any profile images of any instance. The well-
 195 matched ones are more likely to be predefined instances. To match proposals and profile images,
 196 we use off-the-shelf features to represent them. In this work, we study two self-supervised learned
 197 models as feature extractors, i.e. DINO_f [8], and DINOV2_f [34]. We feed a square crop (of a
 198 proposal) or a profile image to the feature extractor to obtain its feature representation. We use cosine
 199 similarity over the features as the similarity measure between a proposal and a profile image.

200 **Proposal matching and selection.** As each instance has multiple profile images, we need to design
 201 the similarity between a proposal and an instance. For a proposal, we compute the cosine similarities
 202 of its feature to all the profile images of an instance and use the maximum as its final similarity
 203 to this instance. We then filter out proposals and instances if they have similarities lower than a
 204 threshold, indicating that they are not matched to any instances or proposals. Finally, we obtain a
 205 similarity matrix between all remaining proposals and all remaining instances. Over this matrix, we
 206 study two matching algorithms to find the best match (hence the final InsDet results), i.e. Rank &
 207 Select, and Stable Matching [14, 32]. The former is a greedy algorithm that iteratively selects the best
 208 match (highest cosine similarity) between a proposal and an instance and removes the corresponding
 209 proposal until no proposal-instance is left. The latter produces an optimal list of matched proposals
 210 and instances, such that there exist no pair of instances and proposals which both prefer each other to
 211 their current correspondence under the matching.

212 5 Experiments

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

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_f 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_f 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 ₅₀	AP ₇₅
	avg	hard	easy	small	medium	large		
FasterRCNN [38]	19.54	10.26	23.75	5.03	22.20	37.97	29.21	23.26
RetinaNet [30]	22.22	14.92	26.49	5.48	25.80	42.71	31.19	24.98
CenterNet [50]	21.12	11.85	25.70	5.90	24.15	40.38	32.72	23.60
FCOS [46]	22.40	13.22	28.68	6.17	26.46	38.13	32.80	25.47
DINO [49]	27.99	17.89	32.65	11.51	31.60	48.35	39.62	32.19
SAM + DINO _f	36.97	22.38	43.88	11.93	40.85	62.67	44.13	40.42
SAM + DINOv2 _f	41.61	28.03	47.57	14.58	45.83	69.14	49.10	45.95

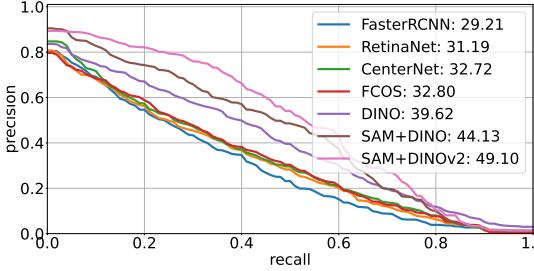


Figure 5: Precision-recall curves with IoU=0.5 (AP₅₀ in the legend) on our InsDet dataset. Stronger detectors perform better, e.g., DINO, a transformer-based detector significantly outperforms FasterRCNN. Furthermore, even with a simple non-learned method, leveraging pretrained models, e.g., SAM+DINOv2_f, outperforms end-to-end learned methods.

222 **Implementation details.** We conduct all the experiments based on open-source implementations,
223 such as Detectron2 [48] (for FasterRCNN and RetinaNet), CenterNet [51], FCOS [45] and DINO [49].
224 The CNN-based end-to-end detectors are initialized with pretrained weights on COCO [29]. We
225 fine-tune CNN-based models using SGD and the transformer-based model using AdamW with a
226 learning rate of 1e-3 and a batch size of 16. We fine-tune all the models for 5 epochs (which are
227 enough for training to converge) and evaluate checkpoints after each epoch for model selection. The
228 models are trained on a single Tesla V100 GPU with 32G memory.

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

234 5.1 Benchmarking Results

235 **Quantitative results.** To evaluate the proposed InsDet protocol and dataset, we first train detec-
236 tors from a COCO-pretrained backbone following the cut-past-learn baseline. Table 2 lists detailed
237 comparisons and Fig. 5 plots the precision-recall curves for the compared methods. We can see that de-
238 tectors with stronger architectures perform better, e.g. DINO (27.99% AP) vs. FasterRCNN (19.54%
239 AP). Second, non-learned methods outperform end-to-end trained models, e.g., SAM+DINOv2_f
240 (41.61% AP) vs. DINO (27.99% AP). Third, all the methods perform poorly on *hard* and *small*
241 instances, suggesting future work focusing on such cases.

242 **Table 3** compares methods w.r.t the average recall (AR) metric. “AR@max10” means AR within the
243 top-10 ranked detections. In computing AR, we rank detections by using the detection confidence
244 scores of the learning-based methods (e.g., FasterRCNN) or similarity scores in the non-learned
245 methods (e.g., SAM+DINO_f). AR_s, AR_m, and AR_l are breakdowns of AR for small, medium, and
246 large testing object instances. Results show that (1) the non-learned methods that use SAM generally
247 recall more instances than others, and (2) all methods suffer from small instances. In sum, results
248 show that methods yielding higher recall achieve higher AP metrics (cf. Table 2).

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_f). AR_s, AR_m, and AR_l 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 _s @max100	AR _m @max100	AR _l @max100
FasterRCNN [38]	26.24	39.24	14.83	44.87	60.05
RetinaNet [30]	26.33	49.38	22.04	56.76	69.69
CenterNet [50]	23.55	44.72	17.84	52.03	64.58
FCOS [46]	25.82	46.28	22.09	52.85	64.11
DINO [49]	29.84	54.22	32.00	59.43	72.92
SAM + DINO _f	31.25	63.05	31.65	70.01	90.63
SAM + DINOv2 _f	40.02	63.06	31.11	70.40	90.36



Figure 6: Visual results of FasterRCNN, DINO, and SAM+DINOv2_f 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. DINOv2_f provides more precise recognition of localized instances, e.g., five instances in the right of the bottom row. Compared with DINO, SAM+DINOv2_f is better at locating occluded instances.

249 **Qualitative results.** Fig. 6 visualizes qualitative results on two testing examples from the InsDet
 250 dataset. Stronger detectors, e.g., the non-learned method SAM+DINOv2_f, produce fewer false
 251 negatives. Even so, all detectors still struggle to detect instances with presented barriers such as
 252 heavy occlusion, instance size being too small, etc. As shown in Fig. 5, the non-learned method
 253 SAM+DINOv2_f outperforms end-to-end learned methods in a wide range of recall thresholds.

254 5.2 Ablation Study

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

257 **Proposal feature extraction in the non-learned method.** Given a box crop (encapsulating the
 258 proposal) generated by SAM in the non-learned method, we study how to process the crop to improve
 259 InsDet performance. Here, we can either crop and feed its minimum bounding box to compute
 260 DINOv2_f features, or we can use the mask to remove the background in the box. Table 4 shows the
 261 comparison. Clearly, the latter performs remarkably better in both “hard” and “easy” scenarios.

262 **Proposal-instance match in the non-learned method.** After generating proposals by SAM, we
 263 need to compare them with instance profile images to get the final detection results. We study the
 264 InsDet performance of the two matching algorithms. Rank & Select is a greedy algorithm that
 265 iteratively finds the best match between any proposals and instances until no instances/proposals

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_f 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 ₅₀			AP ₇₅		
	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 ₅₀			AP ₇₅		
	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

266 are left unmatched; stable matching produces an optimal list of matched proposals and instances
267 such that there does not exist a pair in which both prefer other proposals/instances to their current
268 correspondence under the matching. Table 5 compares these two methods, clearly showing that stable
269 matching works better.

270 5.3 Discussions

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

275 **Limitations.** We note several limitations in our current work. First, while our work uses normal
276 cameras to collect datasets, we expect to use better and cheaper hardware (e.g., depth camera and IMU)
277 for data collection. Second, while the cut-paste-learn method we adopt does not consider geometric
278 cues when synthesizing training images, we hope to incorporate such information to generate better
279 and more realistic training images, e.g., pasting instances only on up-surfaces like tables, desks,
280 and floors. Third, while SAM+DINOv2_f performs the best, this method is time-consuming (see a
281 run-time study in the supplement); real-world applications should consider real-time requirements.

282 **Future work.** In view of the above limitations, the future work includes: (1) Exploring high-
283 resolution images for more precise detection on *hard* situations, e.g., one can combine proposals
284 generated from multi-scale and multi-resolution images. (2) Developing faster algorithms, e.g., one
285 can use multi-scale detectors to attend to regions of interest for progressive detection. (3) Bridging
286 end-to-end fast models and powerful yet slow pretrained models, e.g., one can train lightweight
287 adaptors atop pretrained models for better InsDet.

288 6 Conclusion

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

297 **References**

- 298 [1] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Kosecka, and Alexander C. Berg. A
299 dataset for developing and benchmarking active vision. In *IEEE International Conference on*
300 *Robotics and Automation (ICRA)*, 2017.
- 301 [2] Phil Ammirato, Cheng-Yang Fu, Mykhailo Shvets, Jana Kosecka, and Alexander C Berg. Target
302 driven instance detection. *arXiv:1803.04610*, 2018.
- 303 [3] Siddharth Ancha, Junyu Nan, and David Held. Combining deep learning and verification for
304 precise object instance detection. *arXiv:1912.12270*, 2019.
- 305 [4] Richard Bormann, Xinjie Wang, Markus Völk, Kilian Kleeberger, and Jochen Lindermayr.
306 Real-time instance detection with fast incremental learning. In *IEEE International Conference*
307 *on Robotics and Automation (ICRA)*, 2021.
- 308 [5] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten
309 Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, 2014.
- 310 [6] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection.
311 In *CVPR*, 2018.
- 312 [7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and
313 Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.
- 314 [8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski,
315 and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.
- 316 [9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework
317 for contrastive learning of visual representations. In *International conference on machine*
318 *learning*, pages 1597–1607. PMLR, 2020.
- 319 [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale
320 hierarchical image database. In *CVPR*, 2009.
- 321 [11] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor
322 Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In
323 *International conference on machine learning*, pages 647–655. PMLR, 2014.
- 324 [12] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy
325 synthesis for instance detection. In *ICCV*, 2017.
- 326 [13] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection
327 with discriminatively trained part-based models. *IEEE transactions on pattern analysis and*
328 *machine intelligence*, 32(9):1627–1645, 2009.
- 329 [14] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The*
330 *American Mathematical Monthly*, 69(1):9–15, 1962.
- 331 [15] Georgios Georgakis, Md. Alimoor Reza, Arsalan Mousavian, Phi Hung Le, and Jana Kosecka.
332 Multiview rgb-d dataset for object instance detection. *International Conference on 3D Vision*
333 *(3DV)*, 2016.
- 334 [16] Georgios Georgakis, Md Alimoor Reza, Arsalan Mousavian, Phi-Hung Le, and Jana Kovsecká.
335 Multiview rgb-d dataset for object instance detection. In *International Conference on 3D Vision*
336 *(3DV)*, 2016.
- 337 [17] Georgios Georgakis, Arsalan Mousavian, Alexander C Berg, and Jana Kosecka. Synthesizing
338 training data for object detection in indoor scenes. *Robotics: Science and Systems (RSS)*, 2017.

- 339 [18] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.
- 340 [19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for
341 unsupervised visual representation learning. In *CVPR*, 2020.
- 342 [20] Stefan Hinterstoesser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua,
343 and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects.
344 *IEEE transactions on pattern analysis and machine intelligence*, 34(5):876–888, 2011.
- 345 [21] Stefan Hinterstoßer, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary R. Bradski, Kurt
346 Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less
347 3d objects in heavily cluttered scenes. In *Asian Conference on Computer Vision*, 2012.
- 348 [22] Tomáš Hodavn, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro
349 Urbina, Sudipta N Sinha, and Brian Guenter. Photorealistic image synthesis for object instance
350 detection. In *IEEE international conference on image processing (ICIP)*, 2019.
- 351 [23] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d:
352 Making rgb-based 3d detection and 6d pose estimation great again. In *ICCV*, 2017.
- 353 [24] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson,
354 Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything.
355 *arXiv:2304.02643*, 2023.
- 356 [25] Ikki Kishida, Hong Chen, Masaki Baba, Jiren Jin, Ayako Amma, and Hideki Nakayama. Object
357 recognition with continual open set domain adaptation for home robot. In *WACV*, 2021.
- 358 [26] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view
359 rgb-d object dataset. In *IEEE International Conference on Robotics and Automation (ICRA)*,
360 2011.
- 361 [27] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling.
362 *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
- 363 [28] Hengduo Li, Bharat Singh, Mahyar Najibi, Zuxuan Wu, and Larry S Davis. An analysis of
364 pre-training on object detection. *arXiv:1904.05871*, 2019.
- 365 [29] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan,
366 Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*,
367 2014.
- 368 [30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense
369 object detection. In *ICCV*, 2017.
- 370 [31] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu,
371 and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.
- 372 [32] David G McVitie and Leslie B Wilson. The stable marriage problem. *Communications of the
373 ACM*, 14(7):486–490, 1971.
- 374 [33] Jean-Philippe Mercier, Mathieu Garon, Philippe Giguere, and Jean-Francois Lalonde. Deep
375 template-based object instance detection. In *WACV*, 2021.
- 376 [34] Maxime Oquab, Timothée Darret, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov,
377 Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning
378 robust visual features without supervision. *arXiv:2304.07193*, 2023.
- 379 [35] A Quadros, James Patrick Underwood, and Bertrand Douillard. An occlusion-aware feature for
380 range images. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

- 381 [36] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*,
382 2018.
- 383 [37] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified,
384 real-time object detection. In *CVPR*, 2016.
- 385 [38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time
386 object detection with region proposal networks. In *Advances in Neural Information Processing
387 Systems*, 2015.
- 388 [39] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved
389 rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics
390 and Automation Letters*, 1(2):1179–1185, 2016.
- 391 [40] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground
392 extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314,
393 2004.
- 394 [41] Neil Savage et al. Robots rise to meet the challenge of caring for old people. *Nature*, 601(7893):
395 8–10, 2022.
- 396 [42] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*,
397 2016.
- 398 [43] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features
399 off-the-shelf: an astounding baseline for recognition. In *CVPR Workshops*, 2014.
- 400 [44] Arjun Singh, James Sha, Karthik S. Narayan, Tudor Achim, and P. Abbeel. Bigbird: A
401 large-scale 3d database of object instances. *IEEE International Conference on Robotics and
402 Automation (ICRA)*, 2014.
- 403 [45] Zhi Tian, Hao Chen, Xinlong Wang, Yuliang Liu, and Chunhua Shen. AdelaiDet: A toolbox for
404 instance-level recognition tasks. <https://git.io/adelaidet>, 2019.
- 405 [46] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object
406 detection. In *ICCV*, 2019.
- 407 [47] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-
408 freebies sets new state-of-the-art for real-time object detectors. *arXiv:2207.02696*, 2022.
- 409 [48] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2.
410 <https://github.com/facebookresearch/detectron2>, 2019.
- 411 [49] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Harry
412 Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In
413 *International Conference on Learning Representations*, 2022.
- 414 [50] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv:1904.07850*,
415 2019.
- 416 [51] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Probabilistic two-stage detection. In
417 *arXiv:2103.07461*, 2021.
- 418 [52] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr:
419 Deformable transformers for end-to-end object detection. *arXiv:2010.04159*, 2020.

420 **Checklist**

421 The checklist follows the references. Please read the checklist guidelines carefully for information on
422 how to answer these questions. For each question, change the default [TODO] to [Yes] , [No] , or
423 [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing
424 the appropriate section of your paper or providing a brief inline description. For example:

- 425 • Did you include the license to the code and datasets? [Yes] See Section 3.
- 426 1. For all authors...
 - 427 (a) Do the main claims made in the abstract and introduction accurately reflect the paper's
428 contributions and scope? [Yes] See the last paragraph in Section 1.
 - 429 (b) Did you describe the limitations of your work? [Yes] See the second paragraph in
430 Section 5.3
 - 431 (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the
432 first paragraph in Section 5.3
 - 433 (d) Have you read the ethics review guidelines and ensured that your paper conforms to
434 them? [Yes]
- 435 2. If you are including theoretical results...
 - 436 (a) Did you state the full set of assumptions of all theoretical results? [N/A]
 - 437 (b) Did you include complete proofs of all theoretical results? [N/A]
- 438 3. If you ran experiments (e.g. for benchmarks)...
 - 439 (a) Did you include the code, data, and instructions needed to reproduce the main ex-
440 perimental results (either in the supplemental material or as a URL)? [Yes] We are
441 constructing a website for this work, and will release open-source code.
 - 442 (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were
443 chosen)? [Yes] See Implementation details in Section 5. We (will) release open-source
444 code for further details and reproduction.
 - 445 (c) Did you report error bars (e.g., with respect to the random seed after running experi-
446 ments multiple times)? [N/A]
 - 447 (d) Did you include the total amount of compute and the type of resources used (e.g., type
448 of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.
- 449 4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
 - 450 (a) If your work uses existing assets, did you cite the creators? [Yes] See implementations
451 in Section 5
 - 452 (b) Did you mention the license of the assets? [No] We use multiple open-source GitHub
453 repositories which have different licenses but are free to use for non-commercial and
454 research purposes.
 - 455 (c) Did you include any new assets either in the supplemental material or as a URL? [No]
 - 456 (d) Did you discuss whether and how consent was obtained from people whose data you're
457 using/curating? [N/A]
 - 458 (e) Did you discuss whether the data you are using/curating contains personally identifiable
459 information or offensive content? [N/A]
- 460 5. If you used crowdsourcing or conducted research with human subjects...
 - 461 (a) Did you include the full text of instructions given to participants and screenshots, if
462 applicable? [N/A]
 - 463 (b) Did you describe any potential participant risks, with links to Institutional Review
464 Board (IRB) approvals, if applicable? [N/A]
 - 465 (c) Did you include the estimated hourly wage paid to participants and the total amount
466 spent on participant compensation? [N/A]