Supplementary Materials for Robot Instance Segmentation with Few Annotations for Grasping

A. Technical details

Model The RISE framework begins with an image augmentation step that feeds into a feature extractor followed by an instance segmentation model, and ends at prediction heads for class, bounding box, mask and instance association. We use ResNet-50, ResNet-101 [15] and Swin-L transformer [29] as backbones throughout our experiments, followed by Deformable DETR [54] with 6 encoders and decoders, width of 256 and 300 fixed instance queries, converging on an FPN-like dynamic mask head (as in SeqFormer [44]). In our evaluation, we measure the contribution of the proposed approach to Deformable DETR which serves baseline, and all feature extractors are pretrained on COCO instance segmentation, as is common in Instance segmentation pretraining [58]. The proposed method incorporates a contrastive head (inspired by IDOL [45]) and introduces instance bank, self-supervision branch for non-labeled data, coupled prediction heads for stability (M2B) and label matching strategy during training (MLM). These, in aggregate, allow RISE to outperform both Deformable DETR and SAM, even when these are trained on $\times 10$ more data (1% vs 10%).

Hyperparameters Recall from Sec. 3.1 and Sec. 3.3 that the supervised loss $\mathcal{L}_{\mathrm{s}}$ and unsupervised loss $\mathcal{L}_{\mathrm{u}}$ (Eq. (1), Eq. (5), respectively) are a combination of the class loss $\mathcal{L}_{cls}$, bounding-box loss $\mathcal{L}_{box}$ weighted by λ_1, and the mask loss $\mathcal{L}_{mask}$ weighted by λ_2. We set the loss weights to be $\lambda_1 = 2.0$, $\lambda_2 = 1.0$. The total loss $\mathcal{L}_{\mathrm{total}}$ (in Eq. (9)) combines the supervised loss $\mathcal{L}_{\mathrm{s}}$ or unsupervised loss $\mathcal{L}_{\mathrm{u}}$ (depending on availability of label $\mathbf{y}$), with association loss $\mathcal{L}_{\mathrm{embed}}$ weighted by λ_3. Tab. 6 details an ablation of values of λ_3, showing that the $\mathcal{L}_{\mathrm{embed}}$ contributes to performance, with the best results attained for$\lambda_3 = 0.05$.

Augmentation Strategy The input images are downsampled and randomly cropped so that the longest side is at most 600 pixels, and so that the shortest side is at least 480 pixels. Recall from Sec. 3.2 that the instance bank contains object cutouts from the labeled portion of the dataset, inspired by [8]. We randomly insert K instances from the instance bank into the image to produce the "before" image x_1 and apply weak augmentations (e.g. slight rotation, translation, brightness etc.). However, we depart from previous approaches by having these K instances distributed according to $Beta(\alpha = 0.5, \beta = 0.5)$, depicted in Fig. 5, making it less likely for synthetically placed instances to occlude actual objects in the image. In addition, we also ensure that the K inserted instances don't overlap with one another beyond 85% since they form ground truth labels during self-supervision. We then generate the "after" image x_2 by randomly adding more instances from the instance bank, or alternatively removing (or transforming) already inserted instances, followed by another round of weak augmentations. The before and after frames serve toward learning through interaction, and we facilitate self-supervised learning by strongly augmenting x_1 to yield x_3 and treat $x_w = x_1$ and $x_s = x_3$ as an input a pair of weakly- and strongly-augmented images. We employ this approach in our evaluation of both ARMBench [35] and OCID [38] without the needing to tune its parameters specific datasets.

Training. We train our model for $12,000$ iterations, using AdamW [31] optimizer with learning rate of 10^{-4}, and weight decay of 10^{-4} and lr scheduler of StepLR that steps down an order of magnitude after $8,000$ iterations.

Table 6. Ablation of weight λ_3 applied to the sequence association loss $\mathcal{L}_{\mathrm{embed}}$ described in Eq. (4). This evaluation uses 10% of ARMBench labels (90% treated as unlabeled data) and the Swin-L as backbone. A value of $\lambda_3 = 0$ corresponds to a variant that ignores $\mathcal{L}_{embed}$. The best performance is obtained for $\lambda_3 = [0.05, 0.1]$.

λ_3	AP	AP_{50}	AP_{75}
0	74.7	84.9	75.9
0.02	74.4	84.5	75.6
0.05	**74.9**	**85.2**	76.0
0.1	74.5	83.8	**76.2**
0.5	74.3	84.2	75.3

B. Prediction Matching

The model predicts up to 300 instance labels, boxes and masks which are often far beyond the actual instance count in a given image. In order to compute the loss between valid predictions and ground-truth annotations, we compute the bipartite cost matrix which measures the IoU of each prediction against each ground-truth annotation (either based on box IoU or using the Mask-to-Box method detailed in Sec. 3.3). We then find the fitting assignment for each ground-truth annotation by solving an Optimal Transport (OT) Problem [10]. A similar approach described in Sec. 3.2 serves toward computing $\mathcal{L}_{\mathrm{embed}}$ which requires positive and negative views of an instance. We introduce a method inspired by IDOL [45], where the top-10 prediction matches of each ground-truth annotation are treated as positive views and the rest are considered negative views. The impact of matching is evident in the ablation study in $Tab.$ 3 where we use either OT or a more standard approach of using the top 0.7 IoU as positive and bottom 0.3 IoU as

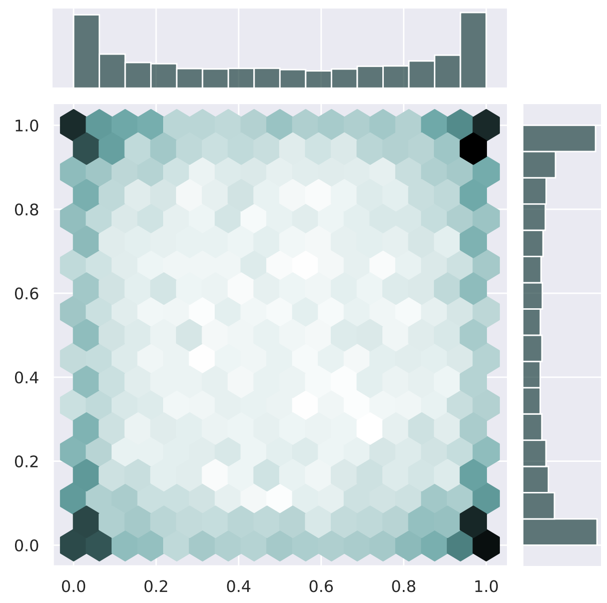


Figure 5. Two dimensional independent $Beta(\alpha = 0.5, \beta = 0.5)$ distribution representing the spread of instance-bank objects inserted into unlabeled images. The distribution favors placing inserted objects at the periphery of the image, since most images contain most of their information about their center (bright regions denote low probability).

negative.

This flow is similarly applied during the self-supervision phase, with the distinction of using pseudo- labels, boxes and masks instead of manually annotated ground truth. Here we also employ Multi-Label Matching (MLM, Sec. 3.3) to allow the model to learn from multiple pseudo-labels predicted from the weak augmentation x_w. The impact of MLM is demonstrated in $Tab.$ 3 and inspired by [2], where it further contributes to the framework's performance.

C. Thresholds

We use time-dependent thresholds [18, 53], whereby an initial threshold value increases every 1000 training steps. The class and mask thresholds start at $\gamma_t^{\text{cls}} = \gamma_t^{\text{mask}} = 0.85$ and peak at 0.98. For the Cascade approach (Sec. 3.3) which combines a lenient threshold followed by a quantile Q_t described in Eq. (6). We set the initial class and mask thresholds to be $\gamma_t^{\text{cls}} = \gamma_t^{\text{mask}} = 0.5$ and peak at 0.85. The quantile Q_t follows the schedule $p_t = a_0 \cdot (1 - {}^t\!/_T)$ where t is the training step, T denotes the total number of training steps, and $a_0 = 0.995$ is the quantile base value. Upon ranking the model's predicted instances by their class score, only the top p_t are retained, and the rest are discarded.

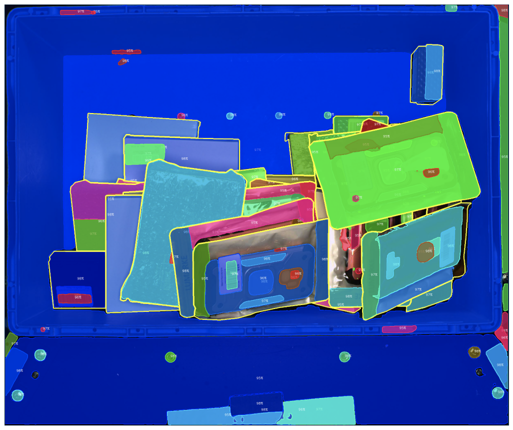


Figure 6. Fine-tuned SAM (1% ARMBench data) showing many fragmented masks and false positive artifacts.

D. Foundation Model Comparison

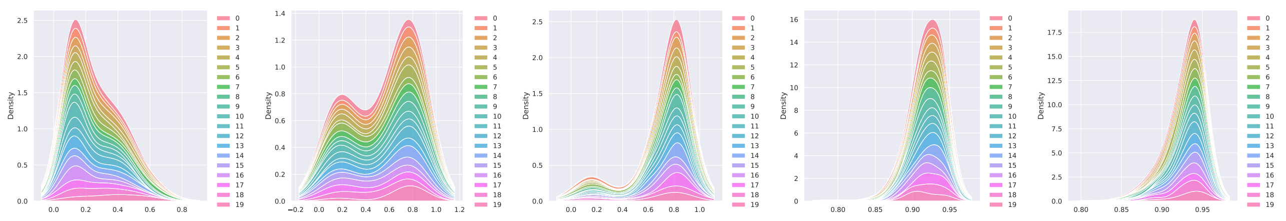
As an additional baseline we compare RISE with the "Segment Anything" (SAM) foundation model [20], fine-tuned on a subset of the ARMBench dataset. In Tab. 1 we demonstrate that despite SAM's unrivalled ability to *segment anything*, it is prone to over-segment and produce mask artifacts, even after fine-tuning on a small portion of domain-specific images. Fig. 6 shows an example where SAM, fine-tuned on 1% of the data still struggles with accurately discerning objects, resulting in fragmented and incomplete object masks and mask predictions that target less significant elements of the image (such as packaging features, rivets and shadows). The numerous false positive predictions impact the overall performance.

E. Study of Cascade Threshold

We study the behavior of pseudo-label and pseudo-mask Cascade filter strategy (Eq. (6)). We evaluate the per-instance prediction score of the model using different base values for the quantile Q_t of the Cascade threshold. In Fig. 7, each color band represent 1000 iterations. The figure shows that setting the base value of the quantile too low would allow in more false-negatives as pseudo- labels and masks. Alternatively, setting it too high would discard valuable predictions as they don't meet the ranking requirement of the qunatile. Following this evaluation we set the quantile base value to $a_0 = 0.995$, which leads to the most balanced behavior of discarding false-positive predictions while allowing through true-positives (even when their score would be considered too low by a standard scheduled threshold).

F. Failure Cases

In both the supervised and self-supervised stages, we randomly draw instance-bank objects and distribute them



(a) a_0 = 0.7 (b) a_0 = 0.9 (c) a_0 = 0.95 (d) a_0 = 0.99 (e) a_0 = 0.995

Figure 7. Confidence density over time using different quantile values for the cascade threshold (Eq. (6)). The x-axis represent the score of all samples, the y-axis the valid instance-count (instance density), and the color band correspond to training iterations in increments of 1000. The cascade threshold applies both a time-dependent threshold (which tightens over time) and a time-dependent quantile Q_t (which loosens over time). The base quantile value a_0 is detailed for each subfigure, showing that the best initial value for the quantile is 0.995.

in the image according to a 2d $Beta(\alpha, \beta)$ distribution (Eq. (2)), and prevent object overlap that exceeds 85% by resampling from the distribution in case of such overlap. In the supervised phase, we also ensure that inserted objects do not overlap existing (ground-truth) objects by more than 85%, whereas in the self-supervised phase, the $Beta(\alpha, \beta)$ distribution (Eq. (2)) helps reduce the likelihood of inserted memory-bank objects overlapping actual objects in the image (since no ground-truth is available). Despite these precautions, failure cases still occur, particularly at very low annotation rates. Since our method incorporate noisy pseudo–labels in low annotated data regime, we will follow improvements in noisy spatial labels [13] for combating with noise and improve pseudo–labels. Fig. 8 shows how a model trained on 1% of the labeled data (99% treated as unlabeled) accurately predicts the masks of all objects in the "before" image x_1 (and ignores the background). However, in the "after" image x_2 (post-interaction), which contains an additionally inserted object (bottom row), the model fails to produce masks for the occluded cardboard box.

Acknowledgment: Bin illustrations in figure Fig. 2 are attributed and modified from [1].

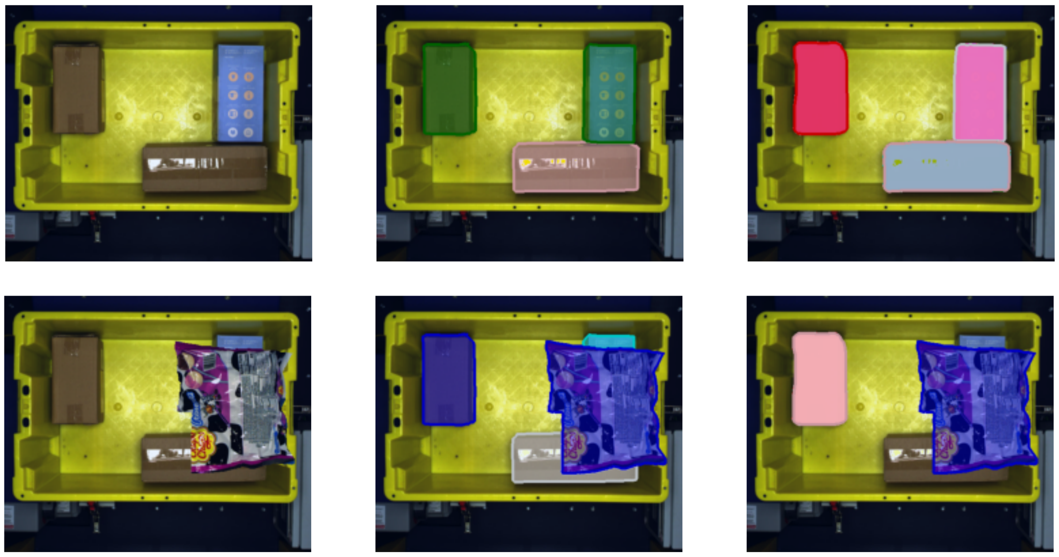


Figure 8. **Failure cases.** Mask prediction of a model trained on 1% of annotated data and 99% unlabeled (ResNet-50 backbone). The top row shows that the model accurately predicts the three objects in the tote. The bottom row includes an additional item inserted from the instance-bank, which partially overlaps several objects. Although the model correctly segments the inserted object, it completely misses one occluded object.