Supplemental Material for LDET

A Experimental Details

In this appendix, we provide experimental details, additional analysis, and visualizations.

Data augmentation. Alg. 1 shows the pytorch-style pseudo-code for our data augmentation.

Implementation. We use the default hyper-parameters, provided by Detectron2, to train and test LDET, Mask RCNN, Mask RCNN$^{\mathrm{S}}$, and Mask RCNN$^{\mathrm{P}}$. Default configuration files are used for COCO [1] and Cityscapes [2] respectively. 2 GPUs of NVIDIA RTX A6000 with 48GB are used to train models.

One-Stage Detector. We use the same hyper-parameters for data augmentation as in Mask RCNN. Also, we follow the default hyper-parameters of RetinaNet and TensorMask. Since RetinaNet does not have mask head by default, we add the mask head on top of the feature pyramid following Mask RCNN. We will publish the code of one-stage detector upon accepntance.

Baselines.

1) Mask R-CNN. We do not make any change to the default training configuration.

2) Mask RCNN^{S}. We compute the area of intersection with ground truth boxes over the area of the proposal box, which we call *IoA*, and sample background boxes with a large value of this criterion. In both region proposal network and roi head, we pick background regions whose IoA is larger than 0.7.

3) Mask RCNN^{P}. Given the classification output (after softmax) from roi head, boxes confidently predicted as one of the foreground classes are choosen from background regions. The threshold to pick the pseudo-foreground is set as 0.9. The classification loss on the pseudo-foreground regions is incorporated to train the detector.

Experiments on texture dataset. To make a background using images of DTD [2], we crop the patch with the size of 256 x 256, and rescale it to the size of a detection training image. Then, we blend the foreground and background in the same way as LDET.

B Analysis

Precision-Recall curve. Fig. B (a) shows precision and recall curve measured on non-VOC classes. In most points, the precision of LDET is better than that of

[1] `detectron2/blob/main/configs/COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_1x.yaml`

[2] `detectron2/blob/main/configs/Cityscapes/mask_rcnn_R_50_FPN.yaml`

Algorithm 1: PyTorch-style pseudocode for our data augmentation

```
# scale=1/8: the size of background region to crop.
# M: mask of the foreground regions.
# Apply gaussian smoothing.
image = gaussian(image)
w, h = image.shape
# Randomly crop background with the specified size.
backg = randomcrop(image, w*scale, h*scale)
# Upscale to the size of the input.
backg = upscale(backg, scale)
# Downsample the input.
image = downsample(image, scale)
# Upsample to the original size.
image = upscale(image, scale)
# Paste foreground objects on the synthesized background.
image = M * image + (1 - M) * backg
# Apply smoothing.
image = smooth(image)
```

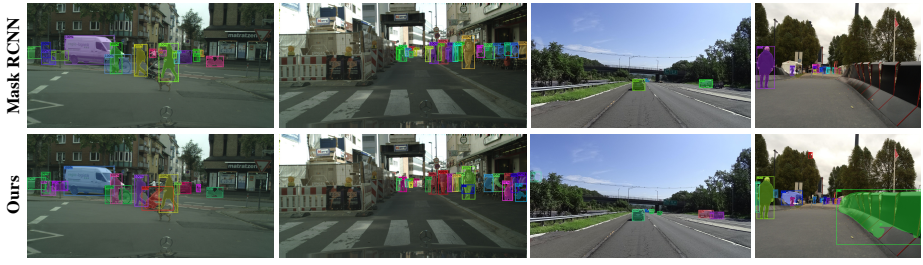
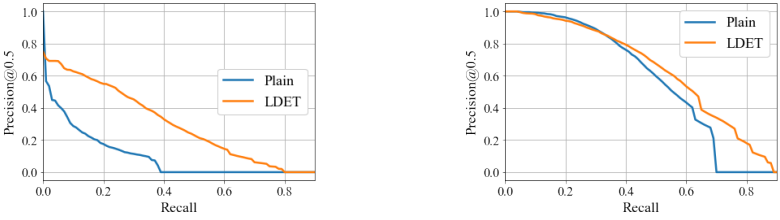


Fig. A: **Visualization for detectors trained on Cityscapes.** Leftmost two images are validation images of Cityscapes, rightmost two are from Mapillary.

the plain model, which means that LDET outputs more precise bounding boxes for novel objects.

Comparison to unsupervised domain adaptation baseline. Although unsupervised domain adaptation (UDA) methods are tailored for a different problem, we conduct comparison to two baselines in the setting of Table 1 in the main paper. Due to the misalignment in the background contents, applying image-level feature alignment should be sub-optimal. We conduct experiments on aligning bounding-box level feature distributions. First, since both synthetic and real images are fully annotated, the use of paired images is natural to align real and synthetic box-level features. Then, we apply supervised contrastive loss [4] on the pair of synthetic and real foreground regions. Second, we follow [1] and apply adversarial training by training a domain classifier on bounding box-level features. Resulting losses include detection loss on synthetic data and adaptation loss on synthetic/real data. AR100 of the first baseline is 21.8, outperforming no-



(a) VOC to Non-VOC. (b) VOC to UVO.

Fig. B: **Precision-Recall Curve.** The precision is measured at the IoU threshold of 0.5. The comparison demonstrates that LDET detects novel objects more precisely than the plain model.

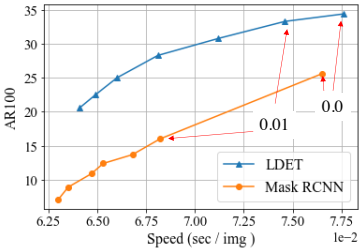
adapted baseline models (18.0), but underperforming LDET (30.8). Adversarial training (AR100 is 4.0.) performs significantly worse than all baselines probably due to the instability of training.

Study on the background. We choose to use the small region of the image as the background. An alternative way is to use a uniform pixel value as the background color. Specifically, we test using the mean pixel value or random pixel value as the background. The performance of using the mean pixel value and the random value are 29.8 and 30.0 respectively (AR100 in Table 1). The uniform pixel value is one option of the background canvas. Although we did not focus on improving the performance of LDET by the selection of background canvas, this result indicates a more room for improvements by it, which we leave for future work.

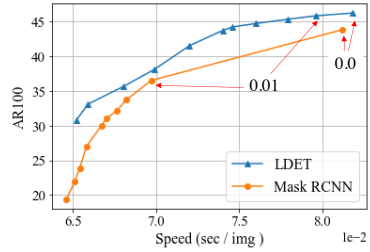
Number of training classes. We investigate the effect of the classes used for training, where 1 (person), 5, and 10 training classes are utilized in the setting of Table 1. AR in each setting is 1: 16.8, 5: 23.9, and 10: 29.5 respectively (All: 30.8). Although including the various categories is important to achieve better performance, using only 10 classes performs on par with using 20 classes.

Comparison to Copy-Paste baseline. Existing copy-paste [3] just cuts objects and paste them on other images without excluding the unlabeled objects while ours excludes the unlabeled objects. We conduct experiments on copy-paste augmentation in the setting of Table 1. AR100 in detection is 10.9 (plain model), 23.0 (copy-paste), and 30.8 (LDET). The augmentation partially solves the issue of suppressing unlabeled objects because pasted objects can hide the unlabeled objects. But, we can still see the significant advantage of LDET. **Study on the number of classes.**

Study on the confidence threshold. In Fig. C, we vary the confidence threshold used to remove unconfident bounding boxes in ROI classification head, where the value is set as 0.05 by default. Here, we vary thresholds starting from 0.0 (no thresholding) to 0.5. This result demonstrates that the baseline drops AR by applying a very small threshold value (Compare AR at 0.0 and 0.01), meaning that the baseline confuses many novel objects with the background.



(a) VOC to Non-VOC.



(b) VOC to UVO.

Fig. C: **Speed (sec /image) v.s. AR.** We vary the confidence threshold of the ROI head and see the changes of the speed and AR. Note that the speed changes due to the non-maximum supression after confidence thresholding. Points at confidence threshold at 0.0 and 0.01 are highlighted with red arrows. The baseline mask rcnn significantly drops performance between the points at 0.0 and 0.01, which indicates that the model suppresses many foreground objects at the confidence value of 0.01.

C Visualization

Cityscapes. Fig. A visualizes some qualitative results. Leftmost two images are from the validation set of Cityscapes, others are from Mapillary. We see that, as indicated by the quantitative results, LDET detects more objects, *e.g.*, *baby carriage* in the leftmost image. However, it is also true that LDET misses novel objects such as *dog* in the leftmost image, probably because there are no categories similar to dogs in the Cityscapes' 8 training categories. This fact indicates some room for improvement in our approach.

More visualizations in COCO. Fig. D and E are additional visualizations in VOC-COCO and COCO, respectively. Note that we add the results of Mask RCNN$^{\mathrm{S}}$, which are not visualized in the main paper due to a limited space. Mask RCNN$^{\mathrm{S}}$ locates many novel objects while generating many false positives. This is probably due to the imbalanced sampling of background regions. By contrast, LDET detects many novel objects, *e.g.*, *elephants, toilet paper, lizard, statue, toy*, *etc.*, with small number of false positives.

Demo on video. Fig. F and G are demo of applying LDET to UVO [5] videos. Click the images to play the videos.

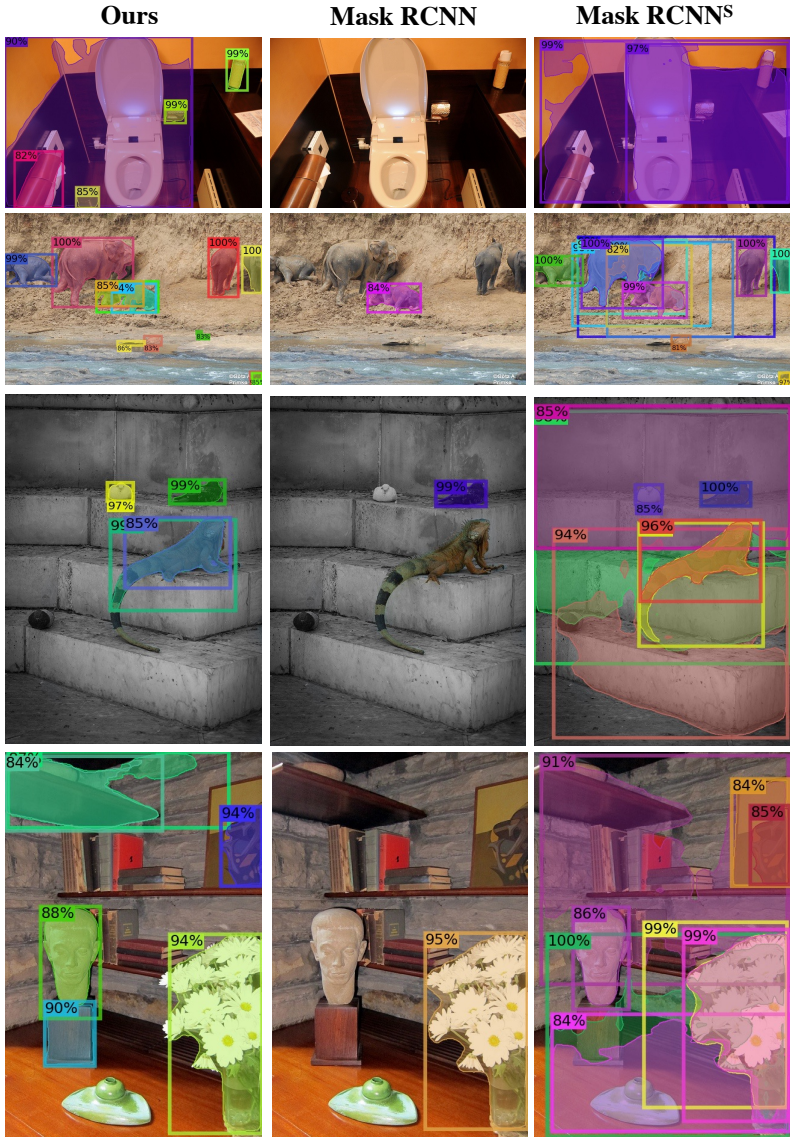


Fig. D: **Visualization in VOC-COCO to COCO setting.** Note that VOC-COCO does not contain objects such as lizard, toilet paper, and elephant.

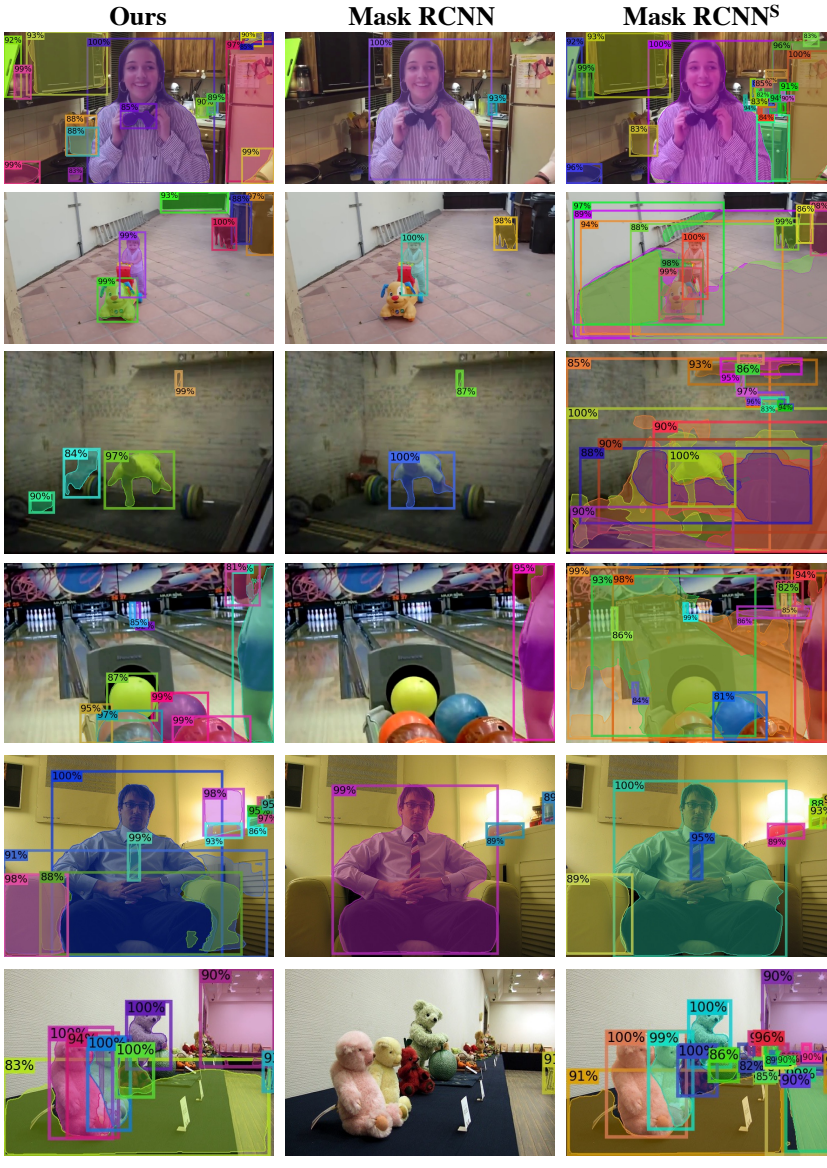


Fig. E: **Visualization of models trained on COCO.** The images are from COCO and UVO.

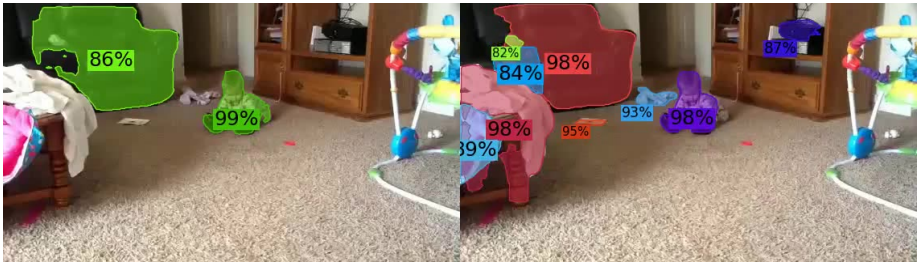


Fig. F: **Video demo of models trained on COCO. Left: Mask RCNN. Right: LDET. Click the image to play the video.**

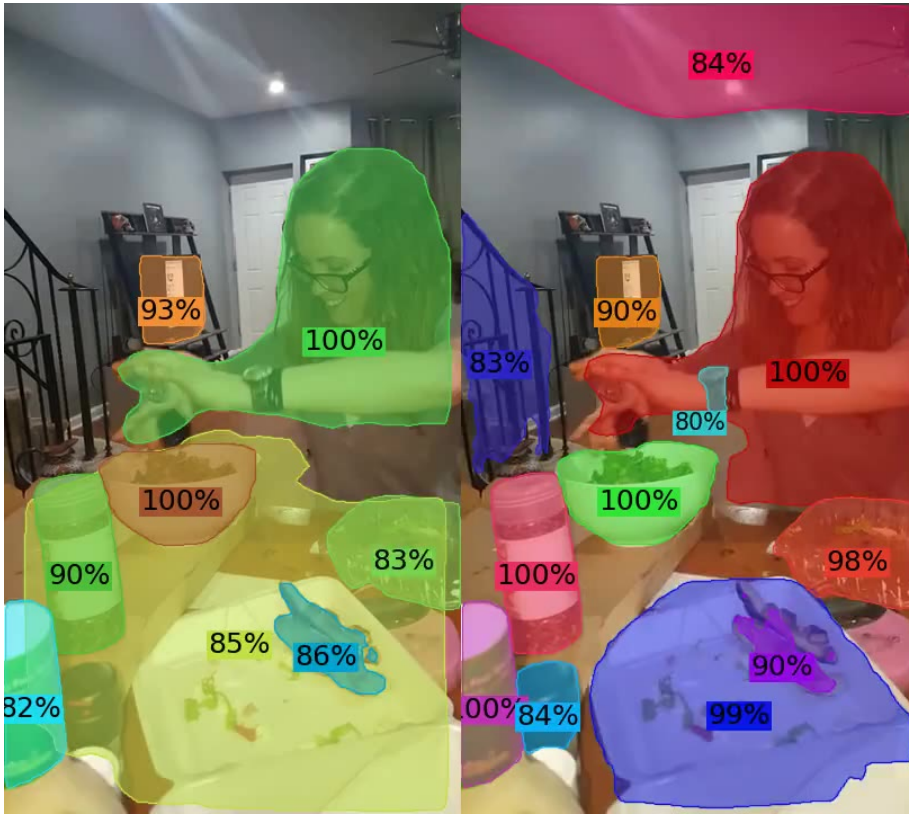


Fig. G: **Video demo of models trained on COCO. Left: Mask RCNN. Right: LDET.**

References

1. Chen, Y., Li, W., Sakaridis, C., Dai, D., Van Gool, L.: Domain adaptive faster r-cnn for object detection in the wild. In: CVPR (2018)
2. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: CVPR. pp. 3606–3613 (2014)
3. Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.Y., Cubuk, E.D., Le, Q.V., Zoph, B.: Simple copy-paste is a strong data augmentation method for instance segmentation. In: CVPR. pp. 2918–2928 (2021)
4. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. NeurIPS (2020)
5. Wang, W., Feiszli, M., Wang, H., Tran, D.: Unidentified video objects: A benchmark for dense, open-world segmentation. arXiv preprint arXiv:2104.04691 (2021)