Exploring Orthogonality in Open World Object Detection

Supplementary Material

A. Experimental settings

A.1. Datasets

For open world object detection, we follow [49, 85] to adopt the superclass-mixed benchmark (M-OWODB) [30] and the superclass-separated benchmark (S-OWODB) [21], both consisting of 80 classes grouped into four sequential tasks, as summarized in Tab. 5. Specifically, M-OWODB is built on COCO [38] and PASCAL VOC [13], and it uses all VOC classes and data as the first task, and the remaining COCO classes as the three successive tasks. However, this can lead to data leakage across super-categories, *e.g.*, most vehicle-related classes belong to the first task, but the truck class is introduced in the second task. To address this, S-OWODB uses a stricter split of the COCO dataset that ensures a clear separation of super-categories across tasks, allowing for a fairer open-world evaluation.

For incremental object detection, which prioritizes the incremental learning capability, we adopt the class splits of PASCAL VOC 2007 proposed in [65]. It contains three two-stage incremental settings of 10 + 10, 15 + 5, and 19 + 1 classes. See the header of Tab. 3 for detailed class splits.

Task IDs ($\rightarrow$)	**Task 1**	**Task 2**	**Task 3**	**Task 4**
M-OWODB split	VOC [13] Classes	Outdoor, Accessories, Appliances, Truck	Sports, Food	Electronic, Indoor, Kitchen, Furniture
# classes	20	20	20	20
# training images	16551	45520	39402	40260
# test images	4952	1914	1642	1738
# training instances	47223	113741	114452	138996
# test instances	14976	4966	4826	6039
S-OWODB split	Animals, Person, Vehicles	Outdoor, Accessories, Appliances, Furniture	Sports, Food	Electronic, Indoor, Kitchen
# classes	19	21	20	20
# training images	89490	55870	39402	38903
# test images	3793	2351	1642	1691
# training instances	421243	163512	114452	160794
# test instances	17786	7159	4826	7010

Table 5. **Task composition in M-OWODB (top) and S-OWODB (bottom).** The semantics of each task split and the number of associated training and test images and object instances are displayed.

A.2. Metrics

We use the common evaluation metrics in [21, 71, 85]. Among them, mean average precision (mAP) and unknown class recall (U-Recall) at IoU threshold of 0.5 serve as the two main metrics. In addition to these, wilderness impact (WI) [9] at IoU threshold of 0.8 and absolute open-set error (A-OSE) [53] at IoU threshold of 0.5 are employed to measure unknown class confusion. Specifically, WI shows the change in precision due to unknown misclassifications:

$$\mathrm{WI} = \frac{\mathrm{P}_{\mathcal{K}}}{\mathrm{P}_{\mathcal{K}\cup\mathcal{U}}} - 1, \tag{8}$$

where $\mathrm{P}_{\mathcal{K}}$ is the precision on known classes, and $\mathrm{P}_{\mathcal{K}\cup\mathcal{U}}$ is the precision when unknown classes are included. On the other hand, A-OSE measures the number of unknown object instances that are misclassified into known classes.

A.3. Implementation details

This section describes some key implementation details, organized by model architecture, training, and test protocols. Most of these are consistent with RandBox [71].

Model architecture. We adopt a Fast R-CNN [17] like architecture, which uses a ResNet-50 [23] pretrained on ImageNet [62] to extract a feature map for each input image, and then applies RoI pooling with 500 random proposals. The resulting proposal features are forwarded to a cascade of detection heads that iteratively refine the detection results. Each head contains a self-attention module, followed by a regression head, an objectness head, and a classification head. The classification head is a linear classifier, while the objectness head uses Batch Normalization [27].

Training scheme. The training process is supervised by ground truth annotations and pseudo-labels estimated from the matching scores. We incorporate standard training objectives including focal loss [39] and regression loss, along with a decorrelation loss with a weight of 1.0. The model is optimized by the AdamW optimizer [46] with a batch size of 12 and an initial learning rate of 2.5×10^{-5}. The training iterations and learning rate schedules in open world and incremental object detection follow [31, 71]. It is worth noting that our Fast R-CNN based model is efficient to train, taking about 36 hours to finish training on M-OWODB with four NVIDIA 2080 Ti GPUs. In comparison, a DETR [4] based model such as CAT [49] takes almost 48 hours on the same benchmark with eight NVIDIA 3090 GPUs.

Test protocol. During inference, we remove the prediction randomness using 10000 pre-defined object proposals covering various locations, shapes, and scales. These proposals are then pruned by non-maximum suppression at IoU threshold of 0.6. The final detection results are selected by a score threshold default to 0.15, following the code of [71].

Task 1	U-Recall (↑)	K-mAP (↑)	WI (↓)	A-OSE (↓)
Base model	8.4	59.8	**0.0244**	5922
Polar	13.2	61.1	0.0254	5026
Unknown	14.6	60.2	0.0265	4862
Feature	**18.2**	**61.3**	0.0276	**4455**

Task 2	U-Recall (↑)	mAP (↑)		
		Previously known	Current known	Both
Base model	6.4	54.7	36.7	45.7
Feature	13.5	55.0	37.7	46.3
Feat + pred	17.2	54.6	37.3	45.9
Ours (sum)	23.5	55.3	**38.7**	47.0
Ours (max)	**26.3**	**55.5**	38.5	**47.0**

Table 6. **Ablation studies of detailed designs on M-OWODB.** Polar and unknown denote the two subdesigns in Sec. 3.3, namely polar coordinate based feature decomposition and unknown class discrimination. Whereas ours (sum) and ours (max) denote our full method with two different routing strategies related to Sec. 3.5.

B. Additional results

B.1. Ablation studies

This section presents additional ablation experiments on detailed designs (*e.g.*, the routing algorithm) and hyperparameter sensitivity, as a complement to Sec. 4.4.

Effectiveness of feature orthogonalization designs. To justify the two subdesigns in feature orthogonalization, we perform a set of ablation studies in Tab. 6. For polar coordinate based feature decomposition, it leads to clear improvements across three metrics (U-Recall, K-mAP, and A-OSE) with only a slight increase in WI. For the confidence-based unknown discrimination strategy, it significantly improves both U-Recall and A-OSE over the standard softmax-based approach (alternative methods like Gaussian modeling [85] are not compared because they do not directly apply to our spherical class feature space). In the meantime, it maintains competitive K-mAP and WI to existing baselines.

Effectiveness of orthogonal designs with incremental learning. The main text has only conducted ablation of the calibration layer (by comparing "Ours" and "– calibration") under the incremental setting. This is because the other designs (feature and prediction orthogonality) are curated for open-set problems rather than incremental learning, and thus can be tested on the first task. For comprehensiveness, we include an additional ablation study in Tab. 6, where both orthogonal designs continue to improve unknown class recall during incremental learning of the second task.

Effectiveness of the routing algorithm. To demonstrate the effectiveness of the maximum softmax probability [25] for routing calibration parameters, we compare it to a more straightforward alternative that sums all corresponding class probabilities as the task probability. Table 6 illustrates that while both methods significantly improve mAP, the maximum probability approach excels in U-Recall by incorporating prediction confidence. This can be understood intuitively with the following example: an object proposal with uniformly high probabilities, *i.e.*, high uncertainties on old tasks, should be considered new and unknown.

Method	DINO	Task 1		Task 4
		U-Recall (↑)	K-mAP (↑)	K-mAP (↑)
Base model	×	8.4	59.8	36.1
Ours	×	**24.6**	**61.3**	**37.9**
Base model	✓	10.1	59.6	38.7
Ours	✓	**23.5**	**61.9**	**40.0**

Table 7. **Influence of different backbones on M-OWODB.** We experiment with supervised or self-supervised (DINO) pretrained backbones on the initial and final task of M-OWODB. Our method remains effective with different pretrained backbones.

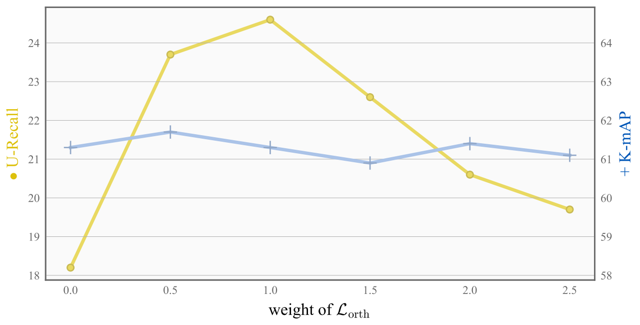

Figure 6. **Sensitivity to loss weight on M-OWODB.** We vary the weight of loss $\mathcal{L}_{\text{orth}}$ and report performance changes on Task 1.

Analysis of backbone. We chose ImageNet-pretrained backbone following R-CNN based literature in [30, 50, 71]. Meanwhile, we recognize the concern with supervised pretraining, and experiment with a self-supervised DINO [5] pretrained backbone following [21, 49, 50, 85]. As shown in Tab. 7, our method delivers competitive open-set and incremental performance with new backbone, in line with the good transferability of self-supervised pretraining.

Hyperparameter sensitivity. Since most of our hyperparameters (*e.g.*, the score threshold for detection results) follow RandBox, we focus on the one newly introduced hyperparameter, the weight of the decorrelation loss $\mathcal{L}_{\text{orth}}$. Its sensitivity analysis is shown in Fig. 6. As can be seen, the inclusion of this new loss significantly improves U-Recall without affecting K-mAP, confirming its effectiveness. On the other hand, increasing the loss weight leads to a reduction in U-Recall, as the model may deviate from its main training objective. Nevertheless, the resulting U-Recall is still higher than the base one without the decorrelation loss.

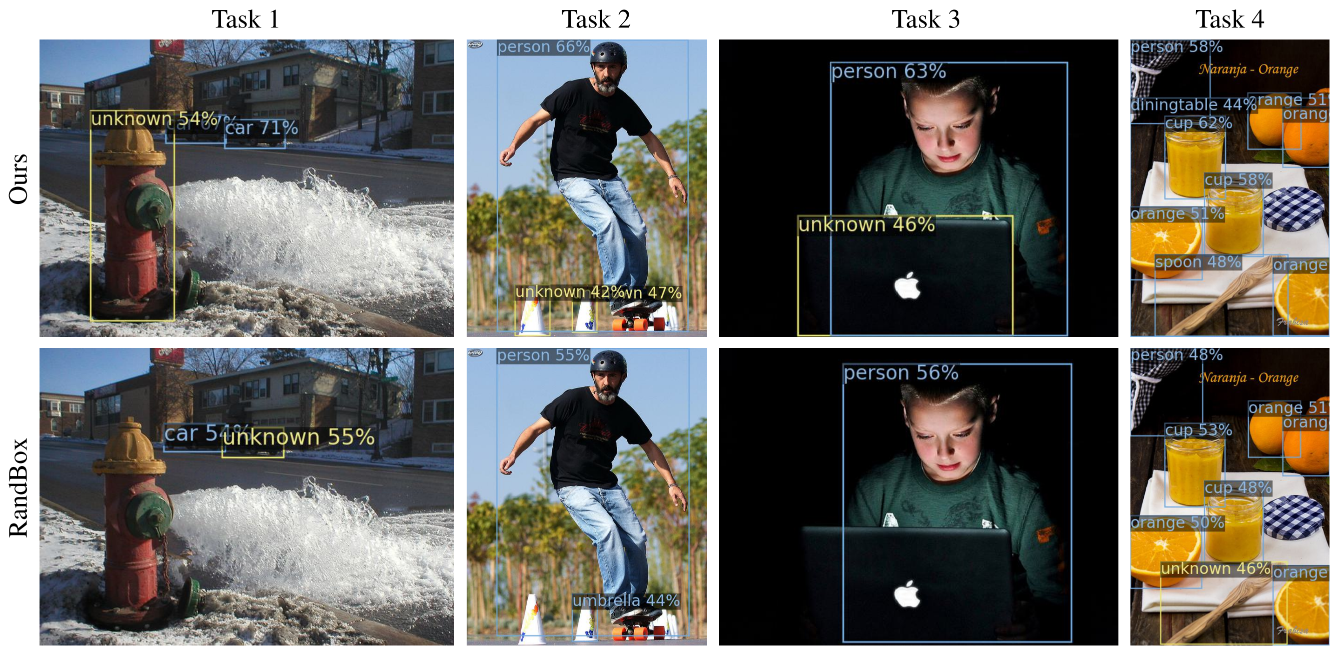

Figure 7. **Qualitative results after different tasks of M-OWODB.** Our method is compared with RandBox [71] in terms of known and unknown object detections after each stage of M-OWODB. Each image pair uses the same score threshold to ensure a fair comparison.

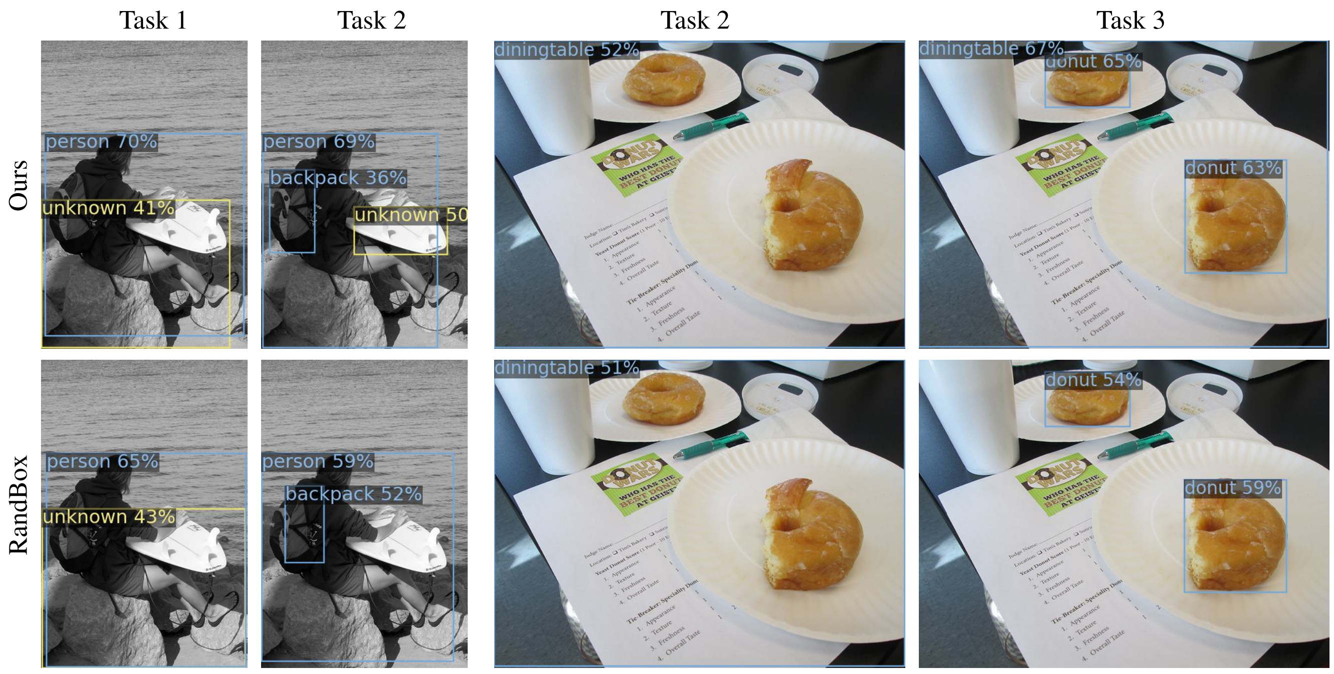

Figure 8. **Illustrations of the incremental learning capability on M-OWODB.** We compare with RandBox [71] in terms of known and unknown object detections across two successive stage of M-OWODB. Each image uses the same score threshold for a fair comparison.

B.2. Visualization

Figure 7 compares our method with RandBox [71] on all four tasks of M-OWODB. As can be observed, our model successfully detects various unknown objects, including the fire hydrant in the first image and the laptop in the third image. Meanwhile, its performance remains advantageous for known objects such as the dining table in the fourth image. Note that there are also some common failures between the two models (*e.g.*, the skateboard in the second image and the knife in the fourth image) that could be improved.

Additionally, our incremental learning capability is illustrated in Fig. 8, where we test two checkpoints before and after incremental learning on the same image. The left two columns show that our model can continuously incorporate new class information (the backpack) to facilitate unknown object discovery (the surfboard). The last two columns suggest that, compared to RandBox, our model forgets less of the old class (dining table) and even exhibits some degree of knowledge consolidation as its object score increases.

C. Limitations

We would like to further discuss the limitations of our work: (1) This work is limited to existing open world object detection datasets [21, 30] to allow for more controllable experiments, while there is a growing need for research on scalability with larger models trained on more data. (2) The problem formulation assumes that the current task identifier is specified during training, which is not compatible with setups with blurry task boundaries or unknown task identifiers. (3) Our proposed method is scoped to the traditional supervised training scheme and needs adaptation for recent pre-training methods such as GLIP [34] and Detic [83].