Supplementary

1 More Details of OA-MIL Deployment

Here we provide more details of OA-MIL deployment on FasterRCNN [34] and RetinaNet [27].

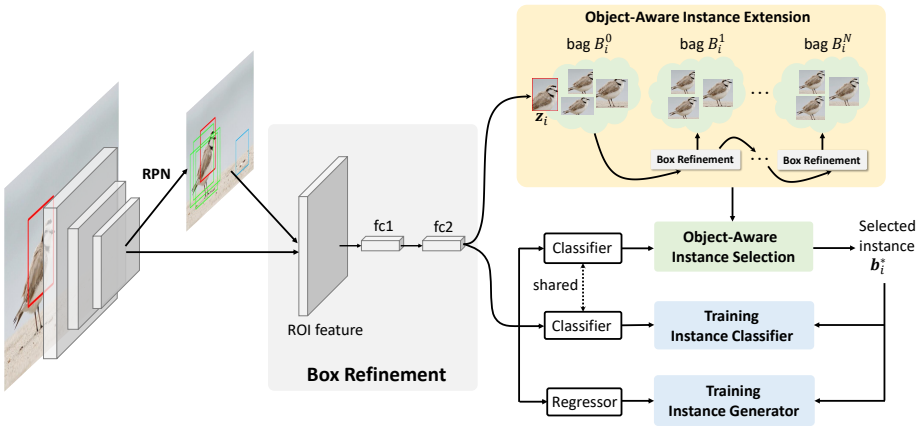


Fig. 1. An overview of our OA-MIL on FasterRCNN. RPN denotes region proposal network and Box Refinement denotes the second stage of FasterRCNN. For simplicity, we omit the bag construction process, where the object bags are constructed based on the outputs of the box refinement module. We perform object-aware instance extension in a multi-stage manner, which produces an extended object bag set $\{B_i^0, B_i^1, \ldots, B_i^N\}$. Then, we adopt object-aware instance selection to select the best instance $\mathbf{b}_i^*$, which is used to train the instance generator (regressor), the instance selector (classifier), and the instance classifier (classifier).

1.1 Deployment to FasterRCNN

The deployment includes two steps, the first step is to construct object bags and the second step is to apply OA-MIL on FasterRCNN. Fig. 1 depicts the pipeline of our method on FasterRCNN.

Bag Construction. We construct object bags based on the outputs of the second stage of FasterRCNN. Specifically, we treat each inaccurately annotated object as a positive bag, where instances are positive anchors (object proposals) corresponding to a specific object. In addition, we treat each negative anchor as a negative bag.

OA-MIL Deployment. As shown in Fig. 1, the instance selector and the instance classifier share the same classifier, *i.e.*, ω_f and ω_g share the same parameters. The regressor in the second stage is treated as the instance generator. We perform object-aware instance extension by multi-stage refinement, which produces a set of extended object bags $\{B_i^0, B_i^1, \ldots, B_i^N\}$. Then, object-aware instance selection is applied to select the best instance $\mathbf{b}_i^*$, where $\mathbf{b}_i^*$ is used to train the instance generator (regressor), the instance selector (classifier), and the instance classifier (classifier). We follow Eq. (11) to train the second stage of FasterRCNN. Note that the training objective of RPN is the same as [34]. In addition, we find that computing instance selection loss (Eq. (3)) across classes is better. Specifically, we first compute the loss of each object bag using Eq. (3) and then average the instance selection loss of each class.

Implementation Details. We implement our method on FasterRCNN [34] with ResNet50-FPN [19,26] backbone. Following common practices [17], the model is trained with "1×" schedule. The hyper parameters are set as $\gamma = 7.5$, $\theta = 0.85$, $N = 4$, and λ is selected from $\{0.01, 0.1\}$ (depending on datasets and noise levels). Our implementation is based on MMDetection toolbox [6].

1.2 Deployment to RetinaNet

The deployment process on RetinaNet is similar to FasterRCNN.

Bag Construction. Since RetinaNet is a one-stage detector, we construct the object bags based on the final outputs of RetinaNet. The bag construction process is the same as FasterRCNN.

OA-MIL Deployment. Similar to FasterRCNN, the instance selector and the instance classifier also share the same classifier. The regression branch is treated as an instance generator. We also use Eq. (11) to train RetinaNet, where the classification and regression loss are the same as [27]. Note that we perform instance extension by recursively constructing positive bags, because RetinaNet does not have a box refinement module.

Implementation Details. We implement our method on RetinaNet [27] with ResNet50-FPN [19,26] backbone. We set the hyper parameters as $\gamma = 0.5$, $\theta = 0.5$, $N = 3$, and λ is selected from $\{0.01, 0.05, 0.15\}$ according to datasets.

2 Hyper-Parameter Sensitivity

2.1 Sensitiveness of Eq. (5)

Here we evaluate the sensitiveness of γ and θ in $\varphi(x)$ (Eq. (5)), because they determine the process of OA-IS. Table 1 shows the results of different γ's and θ's. With γ increases, the mAP improves. We find that $\gamma = 7.5$ works well in practice and is relatively robust to $\gamma \in [2.5, 7.5]$. Next, we fix $\gamma = 7.5$ and prune

Table 1. Sensitiveness of OA-IS under different γ's and θ's on the VOC 2007 test set. The evaluation metric is mAP@0.5(%)

Method	Type	γ	θ	Box Noise Level 20%	40%
Vanilla FasterRCNN	-	-	-	71.2	42.5
FasterRCNN with OA-IS	Varying γ	1.0	1.0	72.2	57.7
		2.5	1.0	73.1	62.7
		5.0	1.0	73.6	63.0
		7.5	1.0	**73.7**	**63.2**
	Varying θ	7.5	0.3	73.8	59.5
		7.5	0.5	**74.6**	61.7
		7.5	0.85	74.2	**63.3**
		7.5	1.0	73.7	62.8

Table 2. Sensitiveness of the loss-balancing weight λ in Eq. (11). Experiments are conducted on VOC dataset under 40% noise.

λ	0.01	0.05	0.1	0.2	0.5
mAP@0.5 (%)	51.0	60.1	**63.8**	62.1	51.2

Table 3. Sensitiveness of the number of instance extension N. Experiments are conducted on the VOC dataset under 40% noise.

N	0	2	4
mAP@0.5 (%)	63.3	63.5	**63.8**

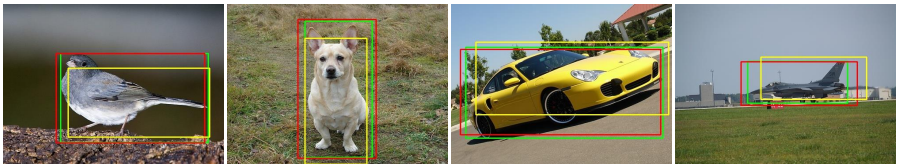
θ. We observe that low box noise (*i.e.*, 20%) favors small θ while high box noise (*i.e.*, 40%) favors large θ. An intuitive explanation is that the ground-truth boxes become more and more inaccurate when box noise increases, thus we may rely more on the selected instances under high box noise. Despite we use $\gamma = 7.5$ and $\beta = 0.85$ to balance different box noise levels, the performance is not necessarily the best.

2.2 Sensitiveness of λ in Eq. (11)

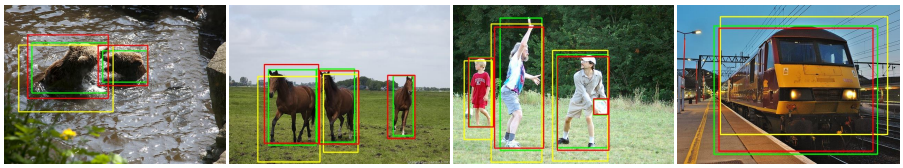
The results are shown in Table 2. The performance first increases and then drops. It reaches the peak when λ is 0.1. This suggests that our approach favors a modest value of λ.

2.3 Sensitiveness of the Number of Instance Extension N

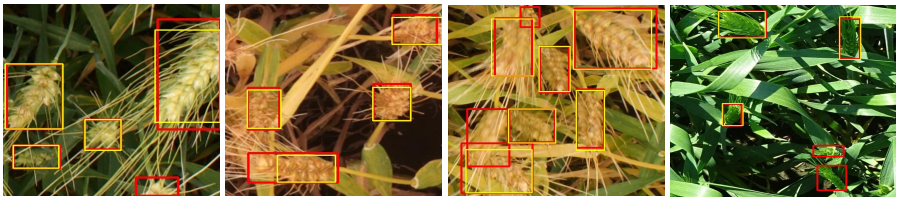
Table 3 shows the results. We can observe that the performance improves steadily when more instance extensions are executed. However, we shall note that a large N will bring extra computational cost during training.



(a) Qualitative results on the VOC 2007 dataset.



(b) Qualitative results on the COCO dataset



(c) Qualitative results on the GWHD dataset

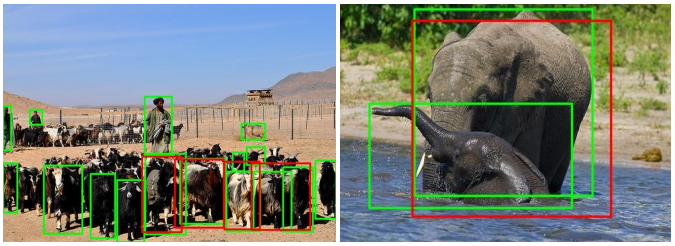
Fig. 2. Qualitative results of our OA-MIL FasterRCNN (red boxes) and vanilla FasterRCNN (yellow boxes) on the VOC, COCO, and GWHD datasets. The ground-truth boxes are in green. Best viewed in color with zoom in.

3 Qualitative Results

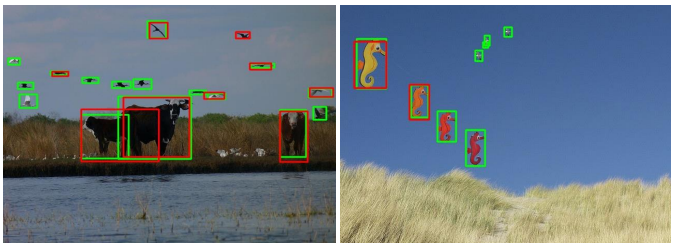
Fig. 2 shows the qualitative results of the vanilla FasterRCNN and our OA-MIL FasterRCNN on the VOC, COCO, and GWHD datasets. Note that the models are trained on VOC with 30% box noise, COCO with 40% box noise, and the "noisy" version of GWHD, respectively.

Specifically, Fig. 2(a) and Fig. 2(b) illustrate the results on the VOC and COCO datasets, we can observe that the vanilla FasterRCNN model tends to predict bounding boxes that cover object parts or include background areas. In addition, the vanilla model also suffers from missing detections in some cases (*e.g.*, it only predicts two horses instead of three in Fig. 2(b)). On the contrary, our method can predict more accurate bounding boxes.

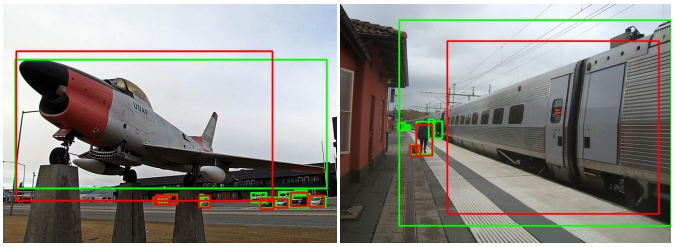
Different from the VOC and COCO datasets, the GWHD dataset only contains wheat head objects. As shown in Fig. 2(c), the vanilla model not only suffers from inaccurate localization but also produces false negatives (missing detections). Note that we omit the ground-truth boxes, because they may overlap with the predictions of our method. In addition, since the noisy GWHD dataset only contains around 20% inaccurate box annotations (*i.e.*, the impact



(a) Missing detections on overlapped objects



(b) Missing detections on small objects



(c) Inaccurate localization on objects with huge size

Fig. 3. Failure cases of our OA-MIL FasterRCNN (red boxes) on the COCO dataset. The ground-truth boxes are in green. Best viewed by zooming in.

of noisy labels is less severe than the VOC and COCO datasets), our method also predicts similar boxes against the vanilla model.

4 Failure Case Analysis

Despite our method achieves promising results with noisy box annotations, it still has some limitations. Fig. 3 illustrates some failure cases of our approach, where our model is trained under 40% box noise on the COCO dataset (red boxes). For comparison, we also visualize the ground-truth boxes (green boxes). First, overlapped objects could render detection failure. Second, our model may miss small objects, because small objects are sensitive to perturbation. Finally, our method can also be difficult to accurately localize objects when they occupy a large proportion of an image.

5 Implementation Details of the Compared Methods

We compare our method with several state-of-the-art approaches [18,20,48]. However, the experimental settings of the above methods are different from ours. For a fair comparison, we implement these methods using the same network architecture (FasterRCNN with ResNet50-FPN backbone) to construct baselines as follows:

1) KL Loss [20] proposes a novel bounding box regression loss for learning object detectors with uncertainty. Our reimplemented baseline achieves 39.0 AP on COCO `val2017`, which is comparable with the original paper that reports 39.2 AP.
2) Co-teaching [18] simultaneously trains two models where each model selects its small-loss samples to train the other. We adopt Co-teaching into Faster-RCNN in a dual-head fashion, *i.e.*, we maintain two box refinement heads for selecting "clean" data to train each other. Note that RPN is shared between the two detection heads and is trained on all samples.
3) SD-LocNet [48] proposes an adaptive sampling method to identify reliable object instances via measuring their localization stability scores. We assign higher weights to samples with higher classification scores and lower prediction variance over consecutive training epochs.

6 Discussion on the Significance of Object Bag

Our object bag distinct the bag in WSOD from two aspects: i) the concept of bag is established on the object instead of image, which explicitly encodes the object location information; ii) the object bag is dynamic during training rather than fixed, which unveils the performance upper bound. In what follows, we first discuss the necessity of our object bag by comparing it with the image bag in WSOD. Then, we investigate the significance of dynamic object bag construction.

Object Bag vs. Image Bag. Since an object is contained in an image, one may think that cropping generously around a noisy object box can form a cropped image bag, thus the existing WSOD methods can be used to tackle the noisy data. However, we argue that the object bag is significantly different from the image bag in two aspects. First, the object bag explicitly exploits the localization information and is capable of leveraging the context information of an object. On the contrary, the cropped image bag ignores the localization information, *i.e.*, we only know cropped object patch contains an object but do not know where is the object. In addition, the context is destroyed by cropping operation. Second, each cropped object forms a new training image, which significantly enlarges the training set and increases the training cost. This situation is more severe when the dataset contains a tremendous amount of objects. On the other hand, our object bag can easily cooperate with existing object detectors and does not change training images.

Table 4. Comparison of object bag and cropped image bag on the VOC 2007 test set

Method		Box Noise Level 20%	Box Noise Level 40%	Training Time
Copped Image Bag with C-MIL	padding=50	17.0	19.9	$\sim$ 12 hour
	padding=100	29.8	29.0	$\sim$ 16 hour
	padding=200	32.0	29.7	$\sim$ 22 hour
Object Bag with OA-MIL (Ours)		74.3	63.8	$<$ 1.5 hour

Table 5. Comparison of OA-MIL FasterRCNN that trained with dynamic object bag and fixed object bag on the VOC 2007 test set. The evaluation metric is mAP@0.5 (%)

Method	Box Noise Level 10%	20%	30%	40%
Fixed Object Bag using Selective Search [43]	72.9	69.8	61.9	48.2
Dynamic Object Bag (Ours)	77.4	74.3	70.6	63.8

To demonstrate the difference between the object bag and the image bag, we adopt C-MIL [45] to train the detection model with the cropped image bag (*i.e.*, object patches that cropped around the noisy bounding box annotations). Although there are various options to crop around an object, we simply crop object with different padding strides, *i.e.*, cropping object patch with padding equals 50, 100, and 200 pixels. Note that padding can preserve context information to some extend, otherwise the cropped object patch may only contain object parts because the ground-truth box is inaccurate. As shown in Table 4, our object bag is superior to the cropped image bag on both performance and training efficiency. Specifically, the cropped image bag requires $8\times \sim 14\times$ longer training time than our object bag and suffers from severe performance degradation, which indicates that it can not deal with noisy box annotations. In addition, the results also show that the cropped image bag performs worse than the image bag (40.7 reported in C-MIL [45]), which further suggests that cropping object patch is not an appropriate option. **We shall note that the comparison in Table 4 is not fair**, because a cropped image may contain few objects. We just aim to demonstrate that the cropped image bag is not an appropriate option.

Dynamic Object Bag vs. Fixed Object Bag. Typical MIL methods in WSOD use an off-the-shelf object proposal generator to construct the image bag, thus the image bag is fixed during training. Different from WSOD, our method involves training learnable instance generator, *i.e.*, the object bag is dynamically constructed during training. We argue that a dynamic object bag is essential for training accurate object detectors with inaccurate box annotations. The reasons are two folds: (i) a dynamic object bag enables flexible optimization while off-the-shelf proposal model limits the performance upper bound; (ii) it increases the diversity of training samples that is beneficial for training.

To elaborate the significance of the dynamic object bag, we first replace the RPN module in FasterRCNN with selective search [42], thus the object bag is fixed during training. Then, we separately train FasterRCNN with dynamic object bag and fixed object bag on the VOC dataset. Table 5 shows the performance comparison under 10% to 40% box noise, the dynamic object bag outperforms the fixed object bag by a large margin, which validates our claim. It is worth noting that the fixed object bag significantly limits the detection performance upper bound, especially under high box noise levels, *e.g.*, 40% box noise (48.2 vs. 63.8).