Salvage of Supervision in Weakly Supervised Object Detection

Lin Sui¹    Chen-Lin Zhang¹,²    Jianxin Wu¹*

¹State Key Laboratory for Novel Software Technology, Nanjing University, China
²4Paradigm Inc., Beijing, China

{suilin0432, zclnjucs, wujx2001}@gmail.com

Abstract

Weakly supervised object detection (WSOD) has recently attracted much attention. However, the lack of bounding-box supervision makes its accuracy much lower than fully supervised object detection (FSOD), and currently modern FSOD techniques cannot be applied to WSOD. To bridge the performance and technical gaps between WSOD and FSOD, this paper proposes a new framework, Salvage of Supervision (SoS), with the key idea being to harness every potentially useful supervisory signal in WSOD: the weak image-level labels, the pseudo-labels, and the power of semi-supervised object detection. This paper proposes new approaches to utilize these weak and noisy signals effectively, and shows that each type of supervisory signal brings in notable improvements, outperforms existing WSOD methods (which mainly use only the weak labels) by large margins. The proposed SoS-WSOD method also has the ability to freely use modern FSOD techniques. SoS-WSOD achieves 64.4 mAP₅₀ on VOC2007, 61.9 mAP₅₀ on VOC2012 and 16.6 mAP₅₀:₉₅ on MS-COCO, and also has fast inference speed. Ablations and visualization further verify the effectiveness of SoS.

1. Introduction

Large-scale datasets with precise annotations are critical in developing detection algorithms, but are expensive to obtain. Thus, weakly supervised object detection (WSOD), which only needs image-level labels on training images, is popular these days. WSOD has borrowed ideas from fully supervised object detection (FSOD), such as object proposals [2, 33] and the Fast-RCNN framework [10]. But modern FSOD has discarded external object proposals and has developed better techniques like Faster-RCNN [24] and FPN [20]. Furthermore, current WSOD methods mostly use VGG16 [28] as the backbone and Fast-RCNN [10] as the detector, which confines both accuracy and speed. That is, due to the lack of detailed box-level annotations, WSOD cannot enjoy the progress from FSOD. In fact, it has been shown that modern FSOD techniques such as ResNet backbones and RoIAlign will even deteriorate WSOD detectors [27]. The weak image-level label is often the only supervisory signal utilized for object detection in WSOD, by resorting to a multi-instance recognition setup [4].

In this paper, we argue that WSOD must fight hard to harness every potential source of supervisory signal, and should find a way to utilize the progress in FSOD. The proposed Salvage of Supervision (SoS) framework (SoS-WSOD) is illustrated in Fig. 1, which has 3 stages. Stage 1 trains a detector with any WSOD method, and we propose an improved OICR [32] as our stage 1. Stage 2 is pseudo-FSOD, where the difficulty is to generate good pseudo box-level annotations in order to boost performance and adopt newer FSOD techniques (e.g., ResNet [13], RoIAlign [12], and FPN [20]), i.e., to salvage the supervision. This problem has been largely ignored in WSOD, for which we propose a simple but effective solution. Stage 3 is proposed by us, named as SSOD, in which we split the whole dataset into a “clean” and a “noisy” part, then treat the noisy part as unlabeled. That is, we salvage additional useful supervisory signals by creating a semi-supervised object detection (SSOD) problem. Hence, we have salvaged supervisory signals out of weak labels by designing novel algorithms to generate high-quality pseudo box-level labels and by creating a semi-supervised learning problem, respectively.

Compared to existing WSOD methods, our SoS-WSOD not only harnesses every potentially useful supervisory signal, but also enables WSOD to fully enjoy both accuracy and speed benefits of modern FSOD methods. Although pseudo FSOD has been tried [6,9,32,41], we will show that SoS-WSOD salvages pseudo-supervision of much higher quality. Hence, our contributions are:

- We propose SoS-WSOD, a new WSOD framework, showing that we must harness all potential supervisory signals in WSOD: generate high-quality pseudo-annotations for FSOD, and treat the generated pseudo-
We improve WSOD accuracy by large margins, with 64.4 mAP\textsubscript{50} on VOC2007, 61.9 mAP\textsubscript{50} on VOC2012, and 16.6 mAP\textsubscript{50:95} on MS-COCO. Besides, SoS-WSOD also has fast detection speed.

We show that although existing WSOD methods lag far behind FSOD in terms of both accuracy and technique, it is very beneficial and feasible to fill this gap. Our pseudo-FSOD enjoys the benefits of all modern FSOD techniques in WSOD, and achieves both higher accuracy and faster speed.

We improve WSOD accuracy by large margins, with 64.4 mAP\textsubscript{50} on VOC2007, 61.9 mAP\textsubscript{50} on VOC2012, and 16.6 mAP\textsubscript{50:95} on MS-COCO. Besides, SoS-WSOD also has fast detection speed.

## 2. Related Work

**Weakly supervised object detection (WSOD).** Weakly supervised object detection (WSOD) seeks to detect the location and type of multiple objects given only image-level labels during training. WSOD methods often utilize object proposals and the multi-instance learning (MIL) framework. WSDDN [4] was the first to integrate MIL into end-to-end WSOD. OICR [32] proposed pseudo groundtruth mining and an online instance refinement branch. PCL [31] clustered proposals to improve pseudo groundtruth mining, and C-MIL [34] improved the MIL loss. Recently, MIST [25] changed the pseudo groundtruth mining rule of OICR, and proposed a Concrete DropBlock module. Zeng et al. [27] made the ResNet [13] backbones working in WSOD. CASD [14] proposed self-distillation along with attention to improve WSOD. Some methods [5, 7, 42] proposed to boost WSOD detector performance with the help of fully annotated COCO-60 dataset.

Some methods tried to adopt modern FSOD techniques into WSOD [26, 27]. Some methods have used the output of WSOD methods (pseudo box annotations) to retrain WSOD models with FSOD. W2F [41] proposed a pseudo groundtruth excavation and a pseudo groundtruth adaptation module to mine large and complete objects for retraining. However, they directly retrain WSOD models without considering any noise in the generated pseudo-labeled dataset which is bound to be very noisy. In contrast, we propose to reconsider the pseudo-labeled dataset with noisy label training perspective and harness the semi-supervised learning paradigm to squeeze better pseudo-labels.

**Semi-supervised object detection (SSOD).** SSOD trains a detector with a small set of images with box-level annotations plus many images without any labels. Compared to WSOD, fewer methods have been proposed for SSOD. SSM [35] stitched high-confidence patches from unlabeled to labeled data. CSD [15] used consistency and background elimination. Recently, STAC [29] used strong data augmentation for unlabeled data. Liu et al. [22] used a teacher-student framework, and ISMT [37] used mean teacher. However, these methods need an exact split of labeled and unlabeled data, and noisy-free box-level annotations for labeled images, but all these are not available in WSOD. We will generate them from the noisy output of the previous stage in SoS-WSOD.

**Learning with noisy labels.** As deep neural networks are annotation-hungry, training DNN with noisy labels also attracts much attention, especially in image classification. Some [30, 38] proposed iterative methods to relabel noisy samples by using network predictions. [1, 23] focused on reweighting. Besides, considering samples with smaller losses as clean ones is also commonly used in many works, such as in [11, 18]. In SoS-WSOD, we adopt the small loss idea to split the noisy output of stage 2 to split the data into “clean” and “noisy” parts.

## 3. Salvage of Supervision

**Notation.** We first define our notation. A training set $D_{w}$ consists of training images $I_{tr}$ and image-level annotations $L_{tr}$. Specifically, each image $x \in \mathbb{R}^{h \times w \times 3}$ in $I_{tr}$ has a corresponding label $y = [y_{1}, y_{2}, \cdots, y_{C}] \in [0, 1]^{C}$, where $C$ is the total number of object categories. We will train a detector $W_{final}$ without using any additional annotations.

**Overview.** Algorithm 1 is the pipeline of the proposed SoS-WSOD. We first train a WSOD detector $W_{wsod}$, which generates pseudo bounding boxes $b_{tr}$. These pseudo su-
Algorithm 1 Salvage of Supervision

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
<td>Training images $I_{tr}$ and image-level class labels $L_{tr}$, test images $I_{te}$</td>
</tr>
<tr>
<td>1:</td>
<td>Train a WSOD model $W_{wsod}$, and generate pseudo groundtruth bounding boxes $b_{tr}$ for training images</td>
</tr>
<tr>
<td>2:</td>
<td>Use $I_{tr}$ and $b_{tr}$ to train a fully supervised object detector $W_{full}$</td>
</tr>
<tr>
<td>3:</td>
<td>Divide $I_{tr}$ into a labeled subset $I_{tr}^<em>$ with pseudo boxes $b_{tr}^</em>$ and an unlabeled subset $I_{tr}'$</td>
</tr>
<tr>
<td>4:</td>
<td>Use $W_{full}$ to initialize, and learn a semi-supervised $W_{final}$ on $L_{tr}, I_{tr}$ (with $b_{tr}^*$) and $I_{tr}'$</td>
</tr>
<tr>
<td>5:</td>
<td><strong>Return:</strong> Use $W_{final}$ to predict the bounding boxes and their class labels for test images</td>
</tr>
</tbody>
</table>

Supervision signals are used to train an FSOD model $W_{full}$, which can use modern FSOD techniques. Then, we treat the generated pseudo-labeled dataset as a noisy one. With our proposed splitting rule, it is split into an unlabeled subset with $I_{tr}'$ and a “clean” labeled subset (those images with confident pseudo boxes) which consists of $I_{tr}^*$ and $b_{tr}^*$. Finally, we adopt an SSOD method to train the final detector $W_{final}$ on the pseudo labeled dataset.

3.1. Stage 1: Improved WSOD

A traditional WSOD detector starts the process. Besides the given image-level annotations $I_{tr}$, most WSOD methods use external object proposals $R$ as extra inputs. Among them, the pipeline of OICR [32] is widely used, which first selects a small number of most confident object proposals $R$ as foreground proposals and then refines them by filtering and adding bounding box regression branches.

We propose to improve OICR with two simple changes as our stage 1. First, recent works [19, 25, 31, 39] demonstrate that better proposal mining rules are critical in obtaining higher recall of objects, which are essential for WSOD detectors. For example, MIST [25] proposed to mine more proposals with low overlap between each other. We find that MIST can catch more objects but will also mine a large number of wrong proposals, while OICR is able to mine accurate proposals but ignores many groundtruth instances. Hence, we introduce a mining rule which strikes a balance between recall and precision. In addition, inspired by CASD [14], we find the multi-input technique is also helpful even without using inverted attention and CASD’s self-attention transfer. More details are in the appendix.

Our proposed WSOD (stage 1) is a strong baseline (cf. Sec. 4). However, we will also show that SoS-WSOD can achieve excellent performance by adopting a weaker WSOD baseline in stage 1, too.

3.2. Stage 2: High-quality pseudo boxes for FSOD

If we are able to output pseudo bounding boxes $b_{tr}$ from stage 1’s detector $W_{wsod}$ that are accurate to some extent, a subsequent FSOD using these boxes can further improve detection. [32] was the first to re-train a WSOD detector by selecting the top-one detection result per class as pseudo groundtruth label, but it will miss a large amount of objects, especially for complicated datasets such as MS-COCO. As will be shown in the ablations in Sec. 4, missed objects will be treated as backgrounds and will even deteriorate the detection accuracy. W2F [41] proposed pseudo groundtruth excavation (PGE) and pseudo groundtruth adaption (PGA) to generate pseudo groundtruth from WSOD output. However, W2F only dealt with the VOC datasets, which have a small number of objects per image and the objects are often large in size. Both modules in W2F are designed to mine large complete objects, and are not suitable for general detection. Figure 2 shows that W2F tends to cluster multiple objects into one pseudo-box. Instead, we propose a simple but effective algorithm called pseudo groundtruth filtering (PGF) to generate high-quality pseudo-boxes from stage 1’s WSOD model, whose pipeline is shown in Algorithm 2.

For each groundtruth class, we only keep the top-scored predictions and those with high confidence (≥ $t_{keep}$, line 6). Then, we remove tiny proposals which are mostly contained inside other proposals in the same category (lines 8-10). After PGF generates pseudo groundtruth $P$, in SoS-WSOD, we are able to use $P$ to supervise and train an FSOD detector $W_{full}$ using modern FSOD methods (e.g., Faster-RCNN [24] + FPN [20]). Please note that the impact of our pseudo-FSOD phase is two-fold. First, the retrained WSOD detector gets accuracy and speed gains from these salvaged supervisory signals. Besides, now we are able to use almost all modern FSOD techniques which are previously not applicable in WSOD. In other words, a WSOD detector now has the flexibility to select most backbones and architecture as needed in WSOD, without resorting to extensive efforts (e.g., as in [27]).
We intentionally designed PGF to be very simple in order to achieve both generality and simplicity. In practice, it is also possible to tailor the pseudo groundtruth mining algorithm to the characteristics of the dataset (e.g., as in [41]).

3.3. Stage 3: Split noisy data for SSOD

FSOD detectors can bring performance gains to WSOD detectors if a high percentage of the generated pseudo groundtruth are correct. However, noisy or wrong pseudo groundtruth (e.g., missing instances, wrong classification results or inaccurate bounding boxes) are inevitable in WSOD. To deal with this issue, we propose to further salvage supervision by treating the generated pseudo-labeled dataset from the perspective of noisy label learning. After splitting the dataset into “clean” labeled part and unlabeled “noisy” part, the semi-supervised learning paradigm can be used.

Data split. Many works [11, 16] have shown that noisy annotations will deteriorate the performance. The pseudo groundtruth boxes \( \tilde{P} \) generated by PGF will inevitably have many noisy ones. As shown in [11, 17, 18], a deep network tends to fit clean data first, then gradually memorize noisy ones. Thus, we use the FSOD detector \( W_{f|\text{full}} \) (the detector before performing learning rate decay in the pseudo-FSOD stage) to divide training images \( I_{tr} \) into labeled \( I_{tr}^* \) (with relatively clean pseudo groundtruth boxes \( b_{tr}^* \)) and unlabeled ones \( I_{tr}^* \) (whose pseudo groundtruth boxes are more noisy). In a classification problem, the split is simple [11]: calculate the loss of each training image, and those with smaller loss values are “clean” ones. But, in object detection, it is hard to decide whether an image is clean simply based on the sum of all losses of all proposals. We follow the small loss idea but revise it for object detection.

Surely we want to focus on foreground objects, hence we propose the following simple splitting process. In Faster-RCNN, regions of interest (RoIs, denoted as \( R \)) are divided into foreground and background RoIs according to the IoU between RoIs and pseudo groundtruth boxes. Then, we do not calculate losses for background RoIs, and accumulate the RPN losses and RoI losses (both classification and regression branches) of different foreground RoIs. The aggregated loss is the split loss for an input image:

\[
L_{\text{split}}(I) = \frac{1}{N_{\text{pos}}} \sum_i I_{\text{fore}}^R L_{\text{split}}(R_i),
\]

\[
L_{\text{split}}(R_i) = L_{RPN_{\text{cls}}}(R_i) + L_{RPN_{\text{reg}}}(R_i) + L_{\text{RoI}_{\text{cls}}}(R_i) + L_{\text{RoI}_{\text{reg}}}(R_i),
\]

where \( N_{\text{pos}} \) is the number of foreground RoIs, \( I_{\text{fore}}^R \) is the indicator function for whether a proposal \( R_i \) belongs to foreground RoIs or not, \( L_{RPN} \) and \( L_{\text{RoI}} \) are RPN and RoI head losses, respectively, and \( \text{Cls} \) and \( \text{Reg} \) stand for classification and regression, respectively. We then rank all training images by \( L_{\text{split}} \) and choose images with small loss values as “clean” labeled data. For simplicity, we use a hard threshold \( K \) to decide sizes of each part. We believe there exist better but more complicated choices such as dynamically using GMM to fit the loss distribution to divide pseudo-labeled dataset. Fig. 3 shows our labeled “clean” part are indeed cleaner than the “noisy” unlabeled part.

The optimal choice of \( K \) varies depends on the size and difficulty of datasets, and we provide a rule-of-thumb for it. We find that traditional WSOD performs well for easy images which will be split as “clean”. Thus, we set \( K \) around the number of images which has a single class label (or few for datasets with many object categories). Ablation studies in Sec. 4 verify the effectiveness of such a rule-of-thumb.

Semi-supervised detection. Now we can perform semi-supervised detection. Unbiased Teacher [22] is an effective semi-supervised detector, whose key idea is a teacher-student pair updated by mutual learning. It first trains a detector using only labeled data (i.e., burn-in) and then uses it to initialize both the student and the teacher detectors. In the mutual learning phase, the teacher will dynamically generate pseudo labels for unlabeled data with weak data augmentation. The student will learn from both well-annotated labeled data and strong augmented unlabeled data with the generated pseudo labels. The teacher will receive updates from the student via exponential moving average.

But, clean data is not available in WSOD. We use the Unbiased Teacher pipeline with a few changes and improvements. First, the pseudo-FSOD training is actually a burn-in process, and we do not need an additional burn-in stage.
Then, the student learns by minimizing
\[
L = L_{\text{sup}} + \lambda_u L_{\text{unsup}},
\]
where the student will learn from both labeled \((L_{\text{sup}})\) and unlabeled \((L_{\text{unsup}})\) data, and \(\lambda_u\) is the weight to balance the unsupervised and supervised loss terms. Specifically, \(L_{\text{sup}}\) and \(L_{\text{unsup}}\) are defined as follows:
\[
L_{\text{sup}} = \sum_i L_{\text{RPN}}(x_i^s, b_i^s) + L_{\text{RPN}}(x_i^u, b_i^u) + L_{\text{RoI}}(x_i^s, b_i^s) + L_{\text{RoI}}(x_i^u, b_i^u),
\]
\[
L_{\text{unsup}} = \sum_i L_{\text{RPN}}(x_i^u, b_i^u) + L_{\text{RoI}}(x_i^u, b_i^u),
\]
where \(x_i^s\) and \(b_i^s\) are images and pseudo groundtruth boxes in the “clean” subset \(I_i^c\). \(x_i^u\) and \(b_i^u\) are images in the unlabeled subset \(I_i^u\) and pseudo groundtruth dynamically generated by the teacher. \(L_{\text{sup}}\) is for labeled data only. For \(L_{\text{unsup}}\), the teacher generates pseudo labels for the student with weak data augmentations, then the student uses strong data augmentations along with pseudo labels to calculate it. We believe the predictions of the teacher are less accurate than annotations for the “clean” data, so \(L_{\text{unsup}}\) only contains the classification loss. In other words, all the regression branches are only learned with “clean” labeled data.

Pseudo boxes generated by the teacher for unlabeled images are not always accurate. Hence, different from regular SSOD, we utilize the image-level labels (i.e., another salvage of supervision) by filtering out false positive pseudo labels, which brings additional benefits to WSOD. Finally, the student detector updates its weights according to Eq. 3, and the teacher receives its update from the student through exponential moving average (EMA).

4. Experiments

We evaluated our SoS-WSOD on three standard WSOD benchmark datasets: VOC2007 [8], VOC2012 [8] and MS-COCO [21]. VOC2007 has 2501 training, 2510 validation and 4952 test images. VOC2012 contains 5717 training, 5823 validation, and 10991 test images. MS-COCO contains around 110,000 training and 5000 validation images. Following the common WSOD evaluation protocol, we use training and evaluation images to train our model on VOC2007 and VOC2012, and evaluate on the test images. For MS-COCO, we train our model on the training and evaluation images and evaluate on the validation images. We use \(m\text{AP}_{50:95}, m\text{AP}_{50}\) and \(m\text{AP}_{75}\) as evaluation metrics for both MS-COCO and VOC2007. For VOC2012, since labels for test images are not released, we report \(m\text{AP}_{50}\) results returned by the official evaluation server.

4.1. Implementation details

We use PyTorch on RTX3090 GPUs, and our code will be released soon. Backbone models are pretrained on ImageNet. It is worth noting that WSOD methods lag behind FSOD in terms of backbone and other techniques. For example, state-of-the-art WSOD methods still use VGG16 as the backbone, while FSOD methods use better architectures. Extra efforts are needed in order to adapt modern backbones to WSOD [27]. Instead, in stage 2 and 3 our SoS-WSOD has the freedom to choose backbones and detection architectures. For simplicity and efficiency, we use FPN with ResNet50 backbone as the FSOD detector in our main experiments without any extra handling. We also use VGG16 as backbone and remove FPN for fair comparisons.

Details of our improved OICR for the WSOD training stage (stage 1) are available in the appendix. In PGF (Algorithm 2), we set \(t_{\text{keep}} = 0.2\) and \(t_{\text{con}} = 0.85\) for all datasets. Although generating pseudo groundtruth labels with TTA (Test Time Augmentation) leads to higher accuracy, the high computational cost (1.5/3/33 hours on VOC2007/2012/MS-COCO) makes it hard to use in large-scale datasets. To keep the same setting in all experiments, we do not use TTA in Algorithm 2.

In both stages 2 and 3, we keep all hyperparameters except \(K\) the same as the default hyperparameters in [36] and [22], respectively. We reduce the total training iterations to reduce the training cost. As for \(K\) in the data splitting process, we use our rule-of-thumb to set it as 2000 and 30000 for VOC2007 and MS-COCO, respectively. More details can be found in the appendix.
line (stage 1 of SoS-WSOD) reaches results are reported with TTA. Our improved WSOD base-

15

with the help of ResNet50 backbone and FPN architecture, did not use modern FPN architecture in stages 2 and 3. MS-COCO, respectively, which is already pretty strong. SoS-WSOD further reaches on VOC2007, VOC2012 and MS-COCO, respectively. After pseudo-

16

accuracy is boosted to 16.6% mAP50:95, 32.8% mAP50 and 15.2% mAP75.

15

WSOD (stage 1+2+3) SoS-WSOD (stage 1+2+3) SoS-WSOD (stage 1+2+3) SoS-WSOD (stage 1+2+3)

15

Table 1. Comparison on PASCAL VOC.

15

Table 2. Comparison on MS-COCO.

4.2. Comparison with state-of-the-art methods

We compare our method with state-of-the-art WSOD methods, with the results reported in Tables 1 and 2. All results are reported with TTA. Our improved WSOD baseline (stage 1 of SoS-WSOD) reaches 55.0% mAP50, 52.5% mAP50 and 11.9% mAP50:95 on VOC2007, VOC2012 and MS-COCO, respectively, which is already pretty strong. For a fair comparison, we used VGG16 as backbone and did not use modern FPN architecture in stages 2 and 3. By harnessing all possible supervision signals, SoS-WSOD finally reaches 60.3% and 57.7% mAP50 on VOC2007 and VOC2012, which outperforms previous methods by large margins. On MS-COCO, SoS-WSOD reaches 15.5% mAP50:95, 30.5% mAP50 and 14.3% mAP75, which outperforms previous methods significantly, too.

When further adopting modern techniques in FSOD, with the help of ResNet50 backbone and FPN architecture, SoS-WSOD further reaches 64.4% and 61.9% mAP50 on VOC2007 and VOC2012, respectively. On MS-COCO, accuracy is boosted to 16.6% mAP50:95, 32.8% mAP50 and 15.2% mAP75.

Recently, some methods [5, 7, 42] leverage the well-annotated MS-COCO-60 dataset (removing the 20 cate-
gories in VOC). As shown in Table 1, they have higher accuracy than pure WSOD methods because of the additional cross-domain data. However, SoS-WSOD achieves higher accuracy than them without resorting to these additional data.

4.3. Ablation studies and visualization

Are salvaged supervision signals useful? Tables 1 and 2 already show that both pseudo boxes (stage 2) and semi-supervised detection (stage 3) notably improve detection accuracy on all 3 datasets. Furthermore, Tables 3 to 5 show results on VOC2007, VOC2012 and MS-COCO, respectively. Our improved WSOD (stage 1 of SoS-WSOD) reaches 54.1%, 51.8% and 23.6% mAP50 on VOC2007, VOC2012 and MS-COCO, respectively. After pseudo-FSOD (stage 2), mAP50 is improved by 3.5%, 2.1% and 3.9%, respectively. Finally, another 5.1%, 5.7% and 3.1% higher mAP50 are boosted by stage 3, respectively. For the stricter mAP50:95 metric on MS-COCO, stage 2 and 3 bring 18.1% and 15.5% relative improvements, respectively.

Compatibility with other WSOD methods. SoS-WSOD is compatible with various WSOD methods in stage 1 and can also improve a weaker WSOD method. We tested the original OICR method in stage 1, and results with TTA are in Table 6. The basic OICR model gets 50.2 mAP50 on VOC2007. With SoS-WSOD, such a model finally reaches 59.9 mAP50. These results demonstrate the flexibility and effectiveness of our SoS-WSOD.

Details about accuracy gains in stage 3. As stated in [22, 29], a mixture of weak and strong data augmentation and EMA updating are essential for SSOD methods. As shown in Table 7, in addition to gains from strong augmentation and EMA updating, our method always brings significant gains. As for models use EMA and strong data augmentation but do not adopt the splitting rule (i.e., the third stage), we use the same (mix weak and strong) data augmentation used in the student branch of stage 3 and maintain another model by EMA in stage 2. SoS-WSOD works well for VGG16 w/o FPN, even when strong augmentation and EMA updating have minor improvements.

Effectiveness of PGF. As shown in Table 1, some
WSOD | PGF | SSOD | mAP50:95 | mAP50 | mAP75 | mAP5 | mAP1 | mAPL
---|---|---|---|---|---|---|---|---
✓ | | | 11.6 | 23.6 | 10.4 | 2.5 | 11.9 | 20.2
✓ | ✓ | | 13.7 | 27.5 | 12.2 | 3.8 | 13.1 | 23.0
✓ | ✓ | ✓ | 15.5 | 30.6 | 14.4 | 5.4 | 16.8 | 24.6

Table 5. Ablations of SoS-WSOD stages on MS-COCO.

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP50:95</th>
<th>mAP50</th>
<th>mAP75</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline (OICR)</td>
<td>24.1</td>
<td>50.2</td>
<td>19.5</td>
</tr>
<tr>
<td>baseline (OICR+)</td>
<td>27.1</td>
<td>55.0</td>
<td>24.8</td>
</tr>
<tr>
<td>SoS-WSOD (OICR)</td>
<td>28.5</td>
<td>59.9</td>
<td>23.8</td>
</tr>
<tr>
<td>SoS-WSOD (OICR+)</td>
<td>32.6</td>
<td>64.4</td>
<td>29.6</td>
</tr>
</tbody>
</table>

Table 6. Ablations of adopting different WSOD model in stage 1. OICR+ is our improved OICR.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>FPN</th>
<th>EMA &amp; Aug</th>
<th>Split Rule</th>
<th>mAP50:95</th>
<th>mAP50</th>
<th>mAP75</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>26.8</td>
<td>56.2</td>
<td>22.3</td>
</tr>
<tr>
<td>VGG16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>27.3</td>
<td>56.6</td>
<td>22.8</td>
</tr>
<tr>
<td>ResNet50</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>28.8</td>
<td>59.0</td>
<td>24.3</td>
</tr>
<tr>
<td>ResNet50</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>27.3</td>
<td>57.6</td>
<td>22.3</td>
</tr>
<tr>
<td>ResNet50</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>29.6</td>
<td>59.6</td>
<td>25.7</td>
</tr>
<tr>
<td>ResNet50</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>31.6</td>
<td>62.7</td>
<td>28.1</td>
</tr>
</tbody>
</table>

Table 7. Detailed accuracy gains in stage 3. The columns mean whether FPN, strong data augmentation, EMA, and the proposed data splitting rule are used or not.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
<th>mAP50:95</th>
<th>mAP50</th>
<th>mAP75</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-One [32]</td>
<td>VOC07</td>
<td>25.8</td>
<td>53.4</td>
<td>22.2</td>
</tr>
<tr>
<td>W2F* [41]</td>
<td>VOC07</td>
<td>27.2</td>
<td>57.0</td>
<td>22.4</td>
</tr>
<tr>
<td>PGF (Ours)</td>
<td>VOC07</td>
<td>27.3</td>
<td>57.6</td>
<td>22.5</td>
</tr>
<tr>
<td>W2F*</td>
<td>MS-COCO</td>
<td>12.6</td>
<td>24.1</td>
<td>11.7</td>
</tr>
<tr>
<td>PGF (Ours)</td>
<td>MS-COCO</td>
<td>13.7</td>
<td>27.5</td>
<td>12.2</td>
</tr>
</tbody>
</table>

Table 8. Results of different pseudo groundtruth mining algorithms. * means that we implemented W2F because its code is not available.

WSOD methods tried to add a re-train stage. Most of them follow [32] to re-train a WSOD detector by selecting the top-one detection result per class as pseudo groundtruth labels. However, the performance of the retrained detector model starts to saturate with the increasing performance of the WSOD model, and pseudo-label generation method is one of the most important reasons. Experimental results in Table 8 show the effectiveness of the proposed PGF. Besides, the widely adopted method (Top-One in Table 8) failed to get benefits from modern techniques. When training with RPN and FPN, missed objects will be treated as backgrounds, which will deteriorate localization. Besides, conspicuous objects also have more chance to become the top score proposal which would cause imbalanced anchor allocation and inadequate training in FPN. As for W2F [41], we have slightly better results on VOC07, and are far superior than it on the more complicated MS-COCO, as W2F is designed for large complete objects in VOC.

**Splitting rule vs. random splitting.** To demonstrate the effectiveness of the proposed splitting rule in stage 3, we compare it with the random splitting strategy. As shown in Table 9, adopting random splitting is notably worse than our proposed method. However, random splitting can still surpass simply adopting EMA update and strong data augmentations in stage 2 by a clear margin, which demonstrates the importance of salvaging useful supervisory signals.

**Longer training schedule for WSOD.** Counting all 3 stages in, SoS-WSOD does require more training iterations. Hence, we double the training iteration of the WSOD stage for a further fair comparison. However, we find that mAP50 will drop from 54.1% to 52.7% due to overfitting.

**Size of the labeled subset in SoSOD.** In the SSOD stage (stage 3), we split a dataset into labeled and unlabeled subsets. The number of pseudo labeled images, $K$, is a hyperparameter. When we treat a small number of images as “clean” labeled ones, severe class imbalance will deteriorate the performance. However, when splitting most images as labeled, the performance will collapse using fully pseudo annotated labels. As shown in Table 10, $K = 2000$ is a suitable choice for VOC2007. These results also demonstrate the effectiveness of the rule-of-thumb we proposed. We use $K = 2000$ in all our experiments on VOC2007, and double the size to 4000 on VOC2012. Following the proposed rule-of-thumb, for MS-COCO, we use $K = 30000$.

**Hyperparameters in PGF.** Figure 5 shows the effects of hyperparameters $t_{keep}$ and $t_{con}$ introduced in PGF (Algorithm 2). These two hyperparameters are robust and $t_{keep} = 0.2$, $t_{con} = 0.85$ works best for mAP50 on VOC2007. We tune these two hyperparameters on the smallest VOC2007 dataset and keep them fixed on all other datasets following previous works [14, 31, 34].

**Inference speed.** SoS-WSOD also enjoys speed benefits from modern FSOD methods. We compare the inference speed in Table 11 (on single RTX3090 GPU). Please note
Figure 4. Visualization of SoS-WSOD results on MS-COCO. Top row: groundtruth annotations. 2nd to 4th rows: detection results from stages 1, 2 and 3, respectively. Last column: a failure case.

Figure 5. Effect of hyperparameters in PGF.

Table 11. Inference speed comparison. “Reg” means the bounding box regression branch.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pro. Time (s / img)</th>
<th>Inf. Time (s / img)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OICR (+Reg.) [32]</td>
<td>&gt; 0.2</td>
<td>0.101</td>
</tr>
<tr>
<td>SoS-WSOD</td>
<td>0</td>
<td>0.031</td>
</tr>
</tbody>
</table>

that the time for generating proposals is always far longer than 0.2 seconds per image, e.g., 8.3 s/img for Selective Search [33], while our SoS-WSOD does not need to generate external proposals. Hence, SoS-WSOD not only is significantly faster than baseline WSOD methods, but also eliminates the time to generate external proposals.

Finally, we provide visualization of detection results on MS-COCO in Fig. 4. These results show that SoS-WSOD can mine more correct objects even in complicated environments. Additional visualization results on VOC2007 and MS-COCO are shown in the appendix.

5. Conclusions and Remarks

In this paper, we proposed a new three-stage framework called Salvage of Supervision for the weakly supervised object detection task (SoS-WSOD). SoS-WSOD tackles the WSOD problem from a new perspective, which advocates harnessing all potentially useful supervisory signals (i.e., salvage of supervision) and successfully adopted modern fully supervised detection techniques in WSOD.

The first stage is a WSOD training stage, in which we train a detector with any WSOD method. Pseudo-FSOD, the second stage, improves the WSOD detector by harnessing the pseudo groundtruth generated by PGF and then freely using techniques from modern FSOD. Finally, stage 3 treats the generated pseudo-labeled dataset as a dataset with noisy labels and proposes a novel criterion to split images into labeled and unlabeled subsets, so semi-supervised detection can be used to squeeze useful supervisory signals to further improve the detection performance. Extensive experiments and visualization on VOC2007, VOC2012 and MS-COCO proved both the effectiveness of our SoS-WSOD and extra supervision signals. By successfully utilizing modern FSOD methods, SoS-WSOD can also have faster detection speed than previous WSOD methods.

As for the limitation, SoS-WSOD still suffers from a large performance gap compared to FSOD, especially on COCO. Due to the lack of fully annotated box-level annotations, we need to salvage more supervisory signals in the future. And, SOS-WSOD still suffers problems like part domination, missing instances and clustered instances, which are widely occurred in WSOD. In the future, we will continue to explore to solve the common WSOD problem and develop better rules to split datasets and stronger SSOD methods for the WSOD task.
References


