Evaluating Weakly Supervised Object Localization Methods Right

Junsuk Choe1,3∗ Sanghyuk Chun3 Seong Joon Oh2∗ Seungho Lee1

1School of Integrated Technology, Yonsei University 2Clova AI Research, LINE Plus Corp. 3Clova AI Research, NAVER Corp. 4University of Tuebingen

Yonsei University LINE Plus Corp. NA VER Corp.

Abstract

Weakly-supervised object localization (WSOL) has gained popularity over the last years for its promise to train localization models with only image-level labels. Since the seminal WSOL work of class activation mapping (CAM), the field has focused on how to expand the attention regions to cover objects more broadly and localize them better. However, these strategies rely on full localization supervision to validate hyperparameters and for model selection, which is in principle prohibited under the WSOL setup. In this paper, we argue that WSOL task is ill-posed with only image-level labels, and propose a new evaluation protocol where full supervision is limited to only a small held-out set not overlapping with the test set. We observe that, under our protocol, the five most recent WSOL methods have not made a major improvement over the CAM baseline. Moreover, we report that existing WSOL methods have not reached the few-shot learning baseline, where the full-supervision at validation time is used for model training instead. Based on our findings, we discuss some future directions for WSOL. Source code and dataset are available at https://github.com/clovaai/wsolevaluation.

1. Introduction

As human labeling for every object is too costly and weakly-supervised object localization (WSOL) requires only image-level labels, the WSOL research has gained significant momentum [60, 58, 59, 6, 26, 56] recently. Among these, class activation mapping (CAM) [60] uses the intermediate classifier activations focusing on the most discriminative parts of the objects to localize the objects of the target class. As the aim in object localization is to cover the full extent of the object, focusing only on the most discriminative parts of the objects is a limitation. WSOL techniques since CAM have focused on this limitation and have proposed different architectural [58, 59, 6] and data-augmentation [26, 56] solutions. The reported state-of-the-art WSOL performances have made a significant improvement over the CAM baseline, from 49.4% to 62.3% [6] and 43.6% to 48.7% [6] top-1 localization performances on Caltech-UCSD Birds-200-2011 [53] and ImageNet [42], respectively. However, these techniques have introduced a set of hyperparameters for suppressing the discriminative cues of CAM and different ways for selecting these hyperparameters. One of such hyperparameters is the operating threshold $\tau$ for generating object bounding boxes from the score maps. Among others, the mixed policies for selecting $\tau$ has contributed to the illusory improvement of WSOL performances over the years; see Figure 1.

Due to the lack of a unified definition of the WSOL task, we revisit the problem formulation of WSOL and show that WSOL problem is ill-posed in general without any localiza-
tion supervision. Towards a well-posed setup, we propose a new WSOL setting where a small held-out set with full supervision is available to the learners.

Our contributions are as follows. (1) Propose new experimental protocol that uses a fixed amount of full supervision for hyperparameter search and carefully analyze six WSOL methods on three architectures and three datasets. (2) Propose new evaluation metrics as well as data, annotations, and benchmarks for the WSOL task at https://github.com/clovaai/wsolevaluation. (3) Show that WSOL has not progressed significantly since CAM, when the calibration dependency and the different amounts of full supervision are factored out. Moreover, searching hyperparameters on a held-out set consisting of 5 to 10 full localization supervision per class often leads to significantly lower performance compared to the few-shot learning (FSL) baselines that use the full supervision directly for model training. Finally, we suggest a shift of focus in future WSOL research: consideration of learning paradigms utilizing both weak and full supervisions, and other options for resolving the ill-posedness of WSOL (e.g. background-class images).

2. Related Work

**By model output.** Given an input image, *semantic segmentation* models generate pixel-wise class predictions [11, 32], *object detection* models [11, 13] output a set of bounding boxes with class predictions, and *instance segmentation* models [28, 7, 18] predict a set of disjoint masks with class and instance labels. *Object localization* [42], on the other hand, assumes that the image contains an object of single class and produces a binary mask or a bounding box around that object coming from the class of interest.

**By type of supervision.** Since bounding box and mask labels cost significantly more than image-level labels, e.g. categories [2], researchers have considered different types of localization supervision: image-level labels [36], gaze [35], points [2], scribbles [27], boxes [8], or a mixture of multiple types [21]. Our work is concerned with the object localization task with only image-level category labels [34, 60].

**By amount of supervision.** Learning from a small amount of labeled samples per class is referred to as few-shot learning (FSL) [54]. We recognize the relationship between our new WSOL setup and the FSL paradigm; we consider FSL methods as baselines for future WSOL methods.

**WSOL works.** Class activation mapping (CAM) [60] turns a fully-convolutional classifier into a score map predictor by considering the activations before the global average pooling layer. Vanilla CAM has been criticized for its focus on the small discriminative part of the object. Researchers have considered dropping regions in inputs at random [26, 56] to diversify the cues used for recognition. Adversarial erasing techniques [58, 6] drop the most discriminative part at the current iteration. Self-produced guidance (SPG) [59] is trained with auxiliary foreground-background masks generated by its own activations. Other than object classification in static images, there exists work on localizing informative video frames for action recognition [38, 29, 55], but they are beyond the scope of our analysis.

**Relation to explainability.** WSOL methods share similarities with the model explainability [17], specifically the *input attribution* task: analyzing which pixels have led to the image classification results [16]. There are largely two streams of work on visual input attribution: variants of input gradients [51, 46, 43, 49, 44, 50, 25, 37] and counterfactual reasoning [40, 12, 61, 41, 15, 20]. While they can be viewed as WSOL methods, we have not included them in our studies because they are seldom evaluated in WSOL benchmarks. Analyzing their possibility as WSOL methods is an interesting future study.

**Our scope.** We study the WSOL task, rather than weakly-supervised detection, segmentation, or instance segmentation. The terminologies tend to be mixed in the earlier works of weakly-supervised learning [45, 14, 9, 48]. Extending our analysis to other weakly-supervised learning tasks is valid and will be a good contribution to the respective communities.

3. Problem Formulation of WSOL

We define and formulate the weakly-supervised object localization (WSOL) task as an image patch classification and show the ill-posedness of the problem. We will discuss possible modifications to resolve the ill-posedness in theory.

3.1. WSOL Task as Multiple Instance Learning

Given an image $X \in \mathbb{R}^{H\times W}$, *object localization* is the task to identify whether or not the pixel belongs to the object of interest, represented via dense binary mask $T = (T_{i1}, \ldots, T_{iHW})$ where $T_{ij} \in \{0, 1\}$ and $(i, j)$ indicate the pixel indices. When the training set consists of precise image-mask pairs $(X, T)$, we refer to the task as *fully-supervised object localization* (FSOL). In this paper, we consider the case when only an image-level label $Y \in \{0, 1\}$ for global presence of the object of interest is provided per training image $X$. This task is referred to as the *weakly-supervised object localization* (WSOL).

One can treat an input image $X$ as a bag of stride-1 sliding window patches of suitable side lengths, $h$ and $w$:
patches with a single label \( Y \) and then show that WSOL still cannot be perfectly solved.

We make some strong assumptions in favor of the learner, formulating WSOL task as a mapping from patches \((X_{ij}, \ldots, X_{HW})\) with \( X_{ij} \in \mathbb{R}^{h \times w} \). The object localization task is then the problem of predicting the object presence \( T_{ij} \) at the image patch \( X_{ij} \). The weak supervision imposes the requirement that each training image \( X \), represented as \((X_{11}, \ldots, X_{HW})\), is only collectively labeled with a single label \( Y \in \{0, 1\} \) indicating whether at least one of the patches represents the object. This formulation is an example of the multiple-instance learning (MIL) [22], as observed by many traditional WSOL works [36, 45, 14, 48].

Following the patch classification point of view, we formulate WSOL task as a mapping from patches \( X \) to the binary labels \( T \) (indices dropped). We assume that the patches \( X \), image-level labels \( Y \), and the pixel-wise labeling \( T \) in our data arise in an i.i.d. fashion from the joint distribution \( p(X, Y, T) \). See Figure 2 for an overview. The aim of WSOL is to produce a scoring function \( s(X) \) such that thresholding it at \( \tau \) closely approximates binary label \( T \). Many existing approaches for WSOL, including CAM [60], use the scoring rules based on the posterior \( s(X) = p(Y|X) \). See Appendix §A.1 for the interpretation of CAM as pixel-wise posterior approximation.

### 3.2. When is WSOL ill-posed?

We show that if background cues are more strongly associated with the target labels \( T \) than some foreground cues, the localization task cannot be solved, even when we know the exact posterior \( p(Y|X) \) for the image-level label \( Y \). We will make some strong assumptions in favor of the learner, and then show that WSOL still cannot be perfectly solved.

We assume that there exists a finite set of cue labels \( \mathcal{M} \) containing all patch-level concepts in natural images. For example, patches from a duck image are one of \{duck’s head, duck’s feet, sky, water, \ldots\} (see Figure 3). We further assume that every patch \( X \) is equivalently represented by its cue label \( M(X) \in \mathcal{M} \). Therefore, from now on, we write \( M \) instead of \( X \) in equations and examine the association arising in the joint distribution \( p(M, Y, T) \). We write \( M^{fg}, M^{bg} \in \mathcal{M} \) for foreground and background cues.

We argue that, even with access to the joint distribution \( p(Y,M) \), it may not be possible to make perfect predictions for the patch-wise labels \( T(M) \) (proof in Appendix §A.2).

#### Lemma 3.1. Assume that the true posterior \( p(Y|M) \) with a continuous pdf is used as the scoring rule \( s(M) = p(Y|M) \). Then, there exists a scalar \( \tau \in \mathbb{R} \) such that \( s(M) \geq \tau \) is identical to \( T \) if and only if the foreground-background posterior ratio \( \frac{p(Y=1|M^{fg})}{p(Y=1|M^{bg})} \geq 1 \) almost surely, conditionally on the event \{\( T(M^{fg}) = 1 \) and \( T(M^{bg}) = 0 \)\}.

In other words, if the posterior likelihood for the image-level label \( Y \) given a foreground cue \( M^{fg} \) is less than the posterior likelihood given background \( M^{bg} \) for some foreground and background cues, no WSOL method can make a correct prediction. This pathological scenario is described in Figure 3: Duck’s feet are less seen in duck images than the water background. Such cases are abundant in user-collected data (Appendix Figure 9).

This observation implies a data-centric solution towards well-posed WSOL: we can augment (1) positive samples \( Y = 1 \) with more less-represented foreground cues \( \{\text{duck images with feet}\} \) and (2) negative samples \( Y = 0 \) with more target-correlated background cues \( \{\text{non-duck images with water background}\} \). Such data-centric approaches are promising future directions for WSOL.

### 4. Evaluation Protocol for WSOL

We reformulate the WSOL evaluation based on our observation of the ill-posedness. We define performance metrics, benchmarks, and the hyperparameter search procedure.

#### 4.1. Evaluation metrics

The aim of WSOL is to produce score maps, where their pixel value \( s_{ij} \) is higher on foreground \( T_{ij} = 1 \) and lower on background \( T_{ij} = 0 \) (§3.1). We discuss how to quantify the above conditions and how prior evaluation metrics have failed to clearly measure the relevant performance. We then propose the MaxBoxAcc and PxAp metrics for bounding box and mask ground truths, respectively.

The localization accuracy [42] metric entangles classification and localization performances by counting the num-
number of images where both tasks are performed correctly. We advocate the measurement of localization performance alone, as the goal of WSOL is to localize objects (§3.1) and not to classify images correctly. To this end, we only consider the score maps $s_{ij}$ corresponding to the ground-truth classes in our analysis. Metrics based on such are commonly referred to as the *GT-known* metrics [26, 58, 59, 6].

A common practice in WSOL is to normalize the score maps per image because the score statistics differ vastly across images. Either max normalization (divide through by $\max_{ij}s_{ij}$) or min-max normalization (additionally map $\min_{ij}s_{ij}$ to zero) has been used; see Appendix §B.1 for the full summary. We always use the min-max normalization.

After normalization, WSOL methods threshold the score map at $\tau$ to generate a tight box around the binary mask $\{(i,j) \mid s_{ij} \geq \tau\}$. WSOL metrics then measure the quality of the boxes. $\tau$ is typically treated as a fixed value [60, 58, 56] or a hyperparameter to be tuned [26, 59, 6]. We argue that the former is misleading because the ideal threshold $\tau$ depends heavily on the data and model architecture and fixing its value may be disadvantageous for certain methods. To fix the issue, we propose new evaluation metrics that are independent of the threshold $\tau$.

**Masks: PxAP.** When masks are available for evaluation, we measure the pixel-wise precision and recall [1]. Unlike single-number measures like mask-wise IoU, those metrics allow users to choose the preferred operating threshold $\tau$ that provides the best precision-recall trade-off for their downstream applications. We define the *pixel precision and recall at threshold $\tau$* as:

$$\text{PxPrec}(\tau) = \frac{|\{s_{ij}^{(n)} \geq \tau \} \cap \{T_{ij}^{(n)} = 1\}|}{|\{s_{ij}^{(n)} \geq \tau\}|}$$

(1)

$$\text{PxRec}(\tau) = \frac{|\{s_{ij}^{(n)} \geq \tau \} \cap \{T_{ij}^{(n)} = 1\}|}{|\{T_{ij}^{(n)} = 1\}|}$$

(2)

For threshold independence, we define and use the *pixel average precision*, $\text{PxAP} := \sum_{\tau} \text{PxPrec}(\tau)\text{PxRec}(\tau)$, the area under curve of the pixel precision-recall curve. We use $\text{PxAP}$ as the final metric in this paper.

**Bounding boxes: MaxBoxAcc.** Pixel-wise masks are expensive to collect; many datasets only provide box annotations. Since it is not possible to measure exact pixel-wise precision and recall with bounding boxes, we suggest a surrogate in this case. Given the ground truth box $B$, we define the *box accuracy at score map threshold $\tau$ and IoU threshold $\delta$*, $\text{BoxAcc}(\tau, \delta)$ [60, 42], as:

$$\text{BoxAcc}(\tau, \delta) = \frac{1}{N} \sum_{n} \text{1}_{\text{IoU}(\text{box}(s(X^{(n)}), \tau), B^{(n)}) \geq \delta}$$

(3)

where $\text{box}(s(X^{(n)}), \tau)$ is the tightest box around the largest-area connected component of the mask $\{(i,j) \mid s(X_{ij}^{(n)} ) \geq \tau\}$. In datasets where more than one bounding box are provided (e.g. ImageNet), we count the number of images where the box prediction overlaps with at least one of the ground truth boxes with IoU $\geq \delta$. When $\delta = 0.5$, the metric is identical to the commonly-called *GT-known localization accuracy* [26] or CorLoc [10], but we suggest a new naming to more precisely represent what is being measured. For score map threshold independence, we report the box accuracy at the optimal threshold $\tau$, the *maximal box accuracy* $\text{MaxBoxAcc}(\delta) := \max_{\tau} \text{BoxAcc}(\tau, \delta)$, as the final performance metric. In this paper, we set $\delta$ to 0.5, following the prior works [60, 26, 58, 59, 6, 56].

**Better box evaluation: MaxBoxAccV2.** After the acceptance at CVPR 2020, we have developed an improved version of $\text{MaxBoxAcc}$. It is better in two aspects. (1) $\text{MaxBoxAcc}$ measures the performance at a fixed IoU threshold ($\delta = 0.5$), only considering a specific level of fineness of localization outputs. We suggest averaging the performance across $\delta \in \{0.3, 0.5, 0.7\}$ to address diverse demands for localization fineness. (2) $\text{MaxBoxAcc}$ takes the largest connected component for estimating the box, assuming that the object of interest is usually large. We remove this assumption by considering the best match between the set of all estimated boxes and the set of all ground truth boxes. We call this new metric as $\text{MaxBoxAccV2}$. For future WSOL researches, we encourage using the $\text{MaxBoxAccV2}$ metric. The code is already available in our repository. We show the evaluation results under the new metric in the in Appendix §C.7.

### 4.2. Data splits and hyperparameter search

For a fair comparison of the WSOL methods, we fix the amount of full supervision for hyperparameter search. As shown in Table 1 we propose three disjoint splits for every dataset: $\text{train-weaksup}$, $\text{train-fullsup}$, and $\text{test}$. The $\text{train-weaksup}$ contains images with weak supervision (the image-level labels). The $\text{train-fullsup}$ contains images with full supervision (either bounding box or binary mask). It is left as freedom for the user to utilize it for hyperparameter search, model selection, ablative studies, or even model fitting. The $\text{test}$ split contains images with full supervision; it must be used only for the final performance report. For example, checking the $\text{test}$ results multiple times with different model configurations violates the protocol as the learner implicitly uses more full supervision than allowed.

As WSOL benchmark datasets, ImageNet [42] and

<table>
<thead>
<tr>
<th>Statistics</th>
<th>ImageNet</th>
<th>CUB</th>
<th>OpenImages</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Classes</td>
<td>1000</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>#images/class</td>
<td>$\sim$1.2K</td>
<td>$\sim$30</td>
<td>$\sim$300</td>
</tr>
<tr>
<td>$\text{train-weaksup}$</td>
<td>10</td>
<td>$\sim$5</td>
<td>25</td>
</tr>
<tr>
<td>$\text{train-fullsup}$</td>
<td>10</td>
<td>$\sim$29</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 1. Dataset statistics. "$\sim$" indicates that the number of images per class varies across classes and the average value is shown.
Caltech-UCSD Birds-200-2011 (CUB) [53] have been extensively used. For ImageNet, the 1.2M “train” and 10K “validation” images for 1000 classes are treated as our train-weaksup and test, respectively. For train-fullsup, we use the ImageNetV2 [39]. We have annotated bounding boxes on those images. CUB has 5994 “train” and 5794 “test” images for 200 classes. We treat them as our train-weaksup and test, respectively. For train-fullsup, we have collected 1000 extra images (~5 images per class) from Flickr, on which we have annotated bounding boxes. For ImageNet and CUB we use the oracle box accuracy BoxAcc.

We contribute a new WSOL benchmark based on the OpenImages instance segmentation subset [3]. It provides a fresh WSOL benchmark to which the models have not yet overfitted. To balance the original OpenImages dataset, we have sub-sampled 100 classes and have randomly selected 29,819, 2,500, and 5,000 images from the original “train”, “validation”, and “test” splits as our train-weaksup, train-fullsup, and test splits, respectively. We use the pixel average precision PxAP. A summary of dataset statistics is in Table 1. Details on data collection and preparation are in Appendix §B.2.

**Hyperparameter search.** To make sure that the same amount of localization supervision is provided for each WSOL method, we refrain from employing any source of human prior outside the train-fullsup split. If the optimal hyperparameter for an arbitrary dataset and architecture is not available by default, we subject it to the hyperparameter search algorithm. For each hyperparameter, its feasible range, as opposed to sensible range, is used as the search space, to minimize the impact of human bias.

We employ the random search hyperparameter optimization [4]; it is simple, effective, and parallelizable. For each WSOL method, we sample 30 hyperparameters to train models on train-weaksup and validate on train-fullsup. The best hyperparameter combination is then selected. Since running 30 training sessions is costly for ImageNet (1.2M training images), we use 10% of images in each class for fitting models during the search. We verify in Appendix §B.4 that the ranking of hyperparameters is preserved even if the training set is sub-sampled.

5. Experiments

5.1. Evaluated Methods

We evaluate six widely used WSOL methods published in peer-reviewed venues. We describe each method in chronological order and discuss the set of hyperparameters. The full list of hyperparameters is in Appendix §C.1.

**Class activation mapping (CAM)** [60] trains a classifier of fully-convolutional backbone with the global average pooling (GAP) structure. At test time, CAM uses the logit outputs before GAP as the score map $s_{ij}$. CAM has the learning rate and the score-map resolution as hyperparameters and all five methods below use CAM in the background.

**Hide-and-seek (HaS)** [26] is a data augmentation technique that randomly selects grid patches to be dropped. The hyperparameters are the drop rate and grid size.

**Adversarial complementary learning (ACoL)** [58] proposes an architectural solution: a two-head architecture where one adversarially erases the high-scoring activations in the other. The erasing threshold is a hyperparameter.

**Self-produced guidance (SPG)** [59] is another architectural solution where internal pseudo-pixel-wise supervision is synthesized on the fly. Three tertiary pixel-wise masks (foreground, unsure, background) are generated from three different layers using two thresholding hyperparameters for each mask and are used as auxiliary supervisions.

**Attention-based dropout layer (ADL)** [6] has proposed a module that, like ACoL, adversarially produces drop masks at high-scoring regions, while not requiring an additional head. Drop rate and threshold are the hyperparameters.

**CutMix** [56] is a data augmentation technique, where patches in training images are cut and pasted to other images during training. The target labels are also mixed. The hyperparameters are the size prior $\alpha$ and the mix rate $r$.

**Few-shot learning (FSL) baseline.** The full supervision in train-fullsup used for validating WSOL hyperparameters can be used for training a model itself. Since only a few fully labeled samples per class are available, we refer to this setting as the few-shot learning (FSL) baseline.

As a simple baseline, we consider a foreground saliency mask predictor [30]. We alter the last layer of a fully convolutional network (FCN) into a $1 \times 1$ convolutional layer with $H \times W$ score map output. Each pixel is trained with the binary cross-entropy loss against the target mask, as done in [5, 32, 33]. For OpenImages, the pixel-wise masks are used as targets; for ImageNet and CUB, we build the mask targets by labeling pixels inside the ground truth boxes as foreground [23]. At inference phase, the $H \times W$ score maps are evaluated with the box or mask metrics.

**Center-gaussian baseline.** The Center-gaussian baseline generates isotropic Gaussian score maps centered at the images. We set the standard deviation to 1, but note that it does not affect the MaxBoxAcc and PxAP measures. This provides a no-learning baseline for every localization method.

5.2. Comparison of WSOL methods

We evaluate the six WSOL methods over three backbone architectures, *i.e.* VGG-GAP [47, 60], InceptionV3 [52], and ResNet50 [19], and three datasets, *i.e.* CUB, ImageNet and OpenImages. For each (method, backbone, dataset) tuple, we have randomly searched the optimal hyperparameters over the train-fullsup with 30 trials, totalling about 9,000 GPU hours. Since the sessions are paralleliz-
able, it has taken only about 200 hours over 50 P40 GPUs to obtain the results. The results are shown in Table 2. We use the same batch sizes and training epochs to enforce the same computational budget. The checkpoints that achieve the best localization performance on train-fullsup are used for evaluation.

Contrary to the improvements reported in prior work (Appendix Table 4), recent WSOL methods have not led to major improvements compared to CAM, when validated in the same data splits and same evaluation metrics. On ImageNet, methods after CAM are generally struggling: only CutMix has seen a boost of +0.3pp on average. On CUB, ADL has attained a +2.0pp gain on average, but ADL fails to work well on other benchmarks. On the new WSOL benchmark, OpenImages, no method has improved over CAM, except for CutMix (+0.4pp on average). The best overall improvements over CAM (63.8% total mean) is a mere +0.2pp boost by HaS. In general, we observe a random mixture of increases and decreases in performance over the baseline CAM, depending on the architecture and dataset. An important result in the table to be discussed later is the comparison against the few-shot learning baseline (§5.5).

Some reasons for the discrepancy between our results and the reported results include (1) the confounding of the actual score map improvement and the calibration scheme, (2) different types and amounts of full supervision employed under the hood, and (3) the use of different training settings (e.g., batch size, learning rates, epochs). More details about the training settings are in Appendix §C.3.

Which checkpoint is suitable for evaluation? After the acceptance by CVPR 2020, we believe that it is inappropriate to use the best checkpoint for WSOL evaluation. This is because the best localization performances are achieved before convergence in many cases (Appendix §C.2). At early epochs, the localization performance fluctuates a lot, so the peak performance is noise rather than the real performance.

Hence, we recommend using the final checkpoint for future WSOL researchers. The evaluation results are shown in Appendix Table 6.

5.3. Score calibration and thresholding

WSOL evaluation must focus more on score map evaluation, independent of the calibration. As shown in Fig-
ACoL score maps show more distributed scores in body areas, demonstrating the effects of adversarial erasing during training. However, the maximal IoU performances do not differ as much. This is because WSOL methods exhibit different score distributions (Figure 5 and Appendix §C.4). Fixing the operating threshold \( \tau \) at a pre-defined value, therefore, can lead to an apparent increase in performance without improving the score maps.

Under our threshold-independent performance measures (MaxBoxAcc and PxAP) shown in Figure 5, we observe that (1) the methods have different optimal \( \tau^* \) on ImageNet and (2) the methods do not exhibit significantly different MaxBoxAcc or PxAP performances. This provides an explanation of the lack of improvement observed in Table 2. We advise future WSOL researchers to report the threshold-independent metrics.

5.4. Hyperparameter analysis

Different types and amounts of full supervision used in WSOL methods manifest in the form of model hyperparameter selection (§3). Here, we measure the impact of the validation on train-fullsup by observing the performance distribution among 30 trials of random hyperparameters. We then study the effects of feature-erasing hyperparameters, a common hyperparameter type in WSOL methods.

Performance with 30 hyperparameter trials. To measure the sensitivity of each method to hyperparameter choices, we plot the performance distribution of the intermediate models in the 30 random search trials. We say that a training session is non-convergent if the training loss is larger than 2.0 at the last epoch. We show the performance distributions of the converged sessions, and report the ratio of non-convergent sessions separately.

Our results in Figure 6 indicate the diverse range of performances depending on the hyperparameter choice. Vanilla CAM is among the less sensitive, with the smallest standard deviation \( \sigma = 1.5 \) on OpenImages. This is the natural consequence of its minimal use of hyperparameters. We thus suggest to use the vanilla CAM when absolutely no full supervision is available. ACoL and ADL tend to have greater variances across benchmarks (\( \sigma = 11.9 \) and 9.8 on CUB). We conjecture that the drop threshold for adversarial erasing is a sensitive hyperparameter.

WSOL on CUB are generally struggling: random hyperparameters often show worse performance than the center baseline (66% cases). We conjecture that CUB is a disadvantageous setup for WSOL: as all images contain birds, the models only attend on bird parts for making predictions. We believe adding more non-bird images can improve the overall performances (§3.2).

We show the non-convergence statistics in Figure 6d. Vanilla CAM exhibit a stable training: non-convergence rates are lowest on OpenImages and second lowest on ImageNet CUB. ACoL and ADL have the highest non-convergence rates on CUB (60% and 58% cases). This suggests that these methods are more sensitive to hyperparameter choices.
5.5. Few-shot learning baselines

Given that WSOL methods inevitably utilize some form of full localization supervision (§3), it is important to compare them against the few-shot learning (FSL) baselines that use it for model tuning itself.

Performances of the FSL baselines (§4.2) are presented in Table 2. Our simple FSL method performs better than the vanilla CAM at 10, 5, and 5 fully labeled samples per class for ImageNet, CUB, and OpenImages, respectively. The mean FSL accuracy on CUB is 92.0%, which is far better than that of the maximal WSOL performance of 70.8%.

We compare FSL against CAM at different sizes of train-fullsup in Figure 8. We simulate the zero-fully-labeled WSOL performance with a set of randomly chosen hyperparameters (§5.4); for FSL, we simulate the no-learning performance through the center-gaussian baseline.

FSL baselines surpass the CAM results already at 1-2 full supervision per class for CUB and OpenImages (92.4 and 70.9% MaxBoxAcc and PxAP). We attribute the high FSL performance on CUB to the fact that all images are birds; with 1 sample/class, there are effectively 200 birds as training samples. For OpenImages, the high FSL performance is due to the rich supervision provided by pixel-wise masks. On ImageNet, FSL results are not as great: they surpass the CAM result at 8-10 samples per class. Overall, however, FSL performances are strikingly good, even at a low data regime. Thus, given a few fully labeled samples, it is perhaps better to train a model with it than to search hyperparameters. Only when there is absolutely no full supervision (0 fully labeled sample), CAM is meaningful (better than the no-learning center-gaussian baseline).

6. Discussion and Conclusion

After years of weakly-supervised object localization (WSOL) research, we look back on the common practice and make a critical appraisal. Based on a precise definition of the task, we have argued that WSOL is ill-posed and have discussed how previous methods have used different types of implicit full supervision (e.g. tuning hyperparameters with pixel-level annotations) to bypass this issue (§3). We have then proposed an improved evaluation protocol that allows the hyperparameter search over a few labeled samples (§4). Our empirical studies lead to some striking conclusions: CAM is still not worse than the follow-up methods (§5.2) and it is perhaps better to use the full supervision directly for model fitting, rather than for hyperparameter search (§5.5).

We propose the following future research directions for the field. (1) Resolve the ill-posedness via e.g. adding more background-class images (§3.2). (2) Define the new task, semi-weakly-supervised object localization, where methods incorporating both weak and full supervision are studied.

Our work has implications in other tasks where learners are not supposed to be given full supervision, but are supervised implicitly via model selection and hyperparameter fitting. Examples include weakly-supervised vision tasks (e.g. detection and segmentation), zero-shot learning, and unsupervised tasks (e.g. disentanglement [31]).

Acknowledgements. We thank Dongyoon Han, Hyojin Park, Jaejun Yoo, Jung-Woo Ha, Junho Cho, Kyungju Baek, Muhammad Ferjad Naeem, Rodrigo Benenson, Youngjoon Yoo, and Youngjung Uh for the feedback. NAVER Smart Machine Learning (NSML) [24] has been used. Graphic design by Kay Choi. The work is supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the MSIP (NRF-2019R1A2C2006123) and ICT R&D pro-
gram of MSIP/IITP [R7124-16-0004, Development of Intelligent Interaction Technology Based on Context Awareness and Human Intention Understanding]. This work was also funded by DFG-EXC-Nummer 2064/1-Projektnummer 390727645 and the ERC under the Horizon 2020 program (grant agreement No. 853489).

References

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In CVPR, 2016. 2
[26] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In ICCV, 2017. 1, 2, 3, 4, 5, 6, 12, 14, 15, 16, 17
[31] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Scholkopf, and Olivier


Appendix

We include additional materials in this document. Each section matches with the main paper sections: §A to main paper §3, §B to main paper §4, and §C to main paper §5.

A. Problem Formulation of WSOL

A.1. CAM as patch-wise posterior approximation

In the main paper §3.1, we have described class activation mapping (CAM) [60] as a patch-wise posterior approximator trained with image-level labels. We describe in detail why this is so.

Equivalent re-formulation of CAM. Originally, CAM is a technique applied on a convolutional neural network classifier $h : \mathbb{R}^{3 \times H \times W} \rightarrow \mathbb{R}^C$, where $C$ is the number of classes, of the following form:

$$h_c(X) = \frac{1}{HW} \sum_{ij} W_{cd} g_{di,j}(X)$$

(4)

where $c, d$ are the channel-dimension indices and $i, j$ are spatial-dimension indices. In other words, $h$ is a fully convolutional neural network, followed by a global average pooling (GAP) and a linear (fully-connected) layer into a $C$-dimensional vector. We may swap the GAP and linear layers without changing the representation:

$$h_c(X) = \frac{1}{HW} \sum_{ij} \left( \sum_d W_{cd} g_{di,j}(X) \right)$$

(5)

$$\approx \frac{1}{HW} \sum_{ij} f_{cij}(X)$$

(6)

where $f$ is now a fully-convolutional network. Each pixel $(i, j)$ in the feature map, $(f_{1ij}(X), \cdots, f_{Cij}(X))$, corresponds to the classification result of the corresponding field of view in the input $X$, written as $X_{ij}$. Thus, we equivalently write

$$h_c(X) = \frac{1}{HW} \sum_{ij} f_c(X_{ij})$$

(7)

where $f$ is now re-defined as a image patch classifier with 1-dimensional feature output (not fully convolutional).

CAM as patch-wise posterior approximator. $h$ is now the average-pooled value of the patch-wise classification scores $f_c(X_{ij})$. CAM trains $f$ by maximizing the image-wide posterior of the ground truth class, where the posterior is defined as the softmax over $f(X_{ij})$:

$$\log p(Y | X) := \log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$$

(8)

In other words, CAM trains the network for patch-wise scores $f_c(X_{ij})$ to estimate the image-wide posterior $p(Y | X)$.

At inference time, CAM estimates the pixel-wise posterior $p(Y | X_{ij})$ approximately by performing $p(Y | X_{ij}) \approx f_{Y}(X_{ij}) / \max_{a \in \alpha} f_{Y}(X_{a \beta})$ (modulo some calibration to make sure $p(Y | X_{ij}) \in [0, 1]$).

A.2. Proof for the ill-posedness lemma

We first define an evaluation metric for our score map for WSOL as when the foreground-background posterior ratio

$$p(Y = 1 | M^{fg}) / p(Y = 1 | M^{bg})$$

Figure 1, written as $\log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$. The score map $f$ is defined as the softmax over $f(X_{ij})$:

$$\log p(Y | X) := \log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$$

(8)

In other words, CAM trains the network for patch-wise scores $f_c(X_{ij})$ to estimate the image-wide posterior $p(Y | X)$.

At inference time, CAM estimates the pixel-wise posterior $p(Y | X_{ij})$ approximately by performing $p(Y | X_{ij}) \approx f_{Y}(X_{ij}) / \max_{a \in \alpha} f_{Y}(X_{a \beta})$ (modulo some calibration to make sure $p(Y | X_{ij}) \in [0, 1]$).

A.2. Proof for the ill-posedness lemma

We first define an evaluation metric for our score map for WSOL as when the foreground-background posterior ratio

$$p(Y = 1 | M^{fg}) / p(Y = 1 | M^{bg})$$

Figure 1, written as $\log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$. The score map $f$ is defined as the softmax over $f(X_{ij})$:

$$\log p(Y | X) := \log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$$

(8)

In other words, CAM trains the network for patch-wise scores $f_c(X_{ij})$ to estimate the image-wide posterior $p(Y | X)$.

At inference time, CAM estimates the pixel-wise posterior $p(Y | X_{ij})$ approximately by performing $p(Y | X_{ij}) \approx f_{Y}(X_{ij}) / \max_{a \in \alpha} f_{Y}(X_{a \beta})$ (modulo some calibration to make sure $p(Y | X_{ij}) \in [0, 1]$).

A.2. Proof for the ill-posedness lemma

We first define an evaluation metric for our score map for WSOL as when the foreground-background posterior ratio

$$p(Y = 1 | M^{fg}) / p(Y = 1 | M^{bg})$$

Figure 1, written as $\log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$. The score map $f$ is defined as the softmax over $f(X_{ij})$:

$$\log p(Y | X) := \log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$$

(8)

In other words, CAM trains the network for patch-wise scores $f_c(X_{ij})$ to estimate the image-wide posterior $p(Y | X)$.

At inference time, CAM estimates the pixel-wise posterior $p(Y | X_{ij})$ approximately by performing $p(Y | X_{ij}) \approx f_{Y}(X_{ij}) / \max_{a \in \alpha} f_{Y}(X_{a \beta})$ (modulo some calibration to make sure $p(Y | X_{ij}) \in [0, 1]$).

A.2. Proof for the ill-posedness lemma

We first define an evaluation metric for our score map for WSOL as when the foreground-background posterior ratio

$$p(Y = 1 | M^{fg}) / p(Y = 1 | M^{bg})$$

Figure 1, written as $\log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$. The score map $f$ is defined as the softmax over $f(X_{ij})$:

$$\log p(Y | X) := \log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$$

(8)

In other words, CAM trains the network for patch-wise scores $f_c(X_{ij})$ to estimate the image-wide posterior $p(Y | X)$.

At inference time, CAM estimates the pixel-wise posterior $p(Y | X_{ij})$ approximately by performing $p(Y | X_{ij}) \approx f_{Y}(X_{ij}) / \max_{a \in \alpha} f_{Y}(X_{a \beta})$ (modulo some calibration to make sure $p(Y | X_{ij}) \in [0, 1]$).

A.2. Proof for the ill-posedness lemma

We first define an evaluation metric for our score map for WSOL as when the foreground-background posterior ratio

$$p(Y = 1 | M^{fg}) / p(Y = 1 | M^{bg})$$

Figure 1, written as $\log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$. The score map $f$ is defined as the softmax over $f(X_{ij})$:

$$\log p(Y | X) := \log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$$

(8)

In other words, CAM trains the network for patch-wise scores $f_c(X_{ij})$ to estimate the image-wide posterior $p(Y | X)$. At inference time, CAM estimates the pixel-wise posterior $p(Y | X_{ij})$ approximately by performing $p(Y | X_{ij}) \approx f_{Y}(X_{ij}) / \max_{a \in \alpha} f_{Y}(X_{a \beta})$ (modulo some calibration to make sure $p(Y | X_{ij}) \in [0, 1]$).

A.2. Proof for the ill-posedness lemma

We first define an evaluation metric for our score map for WSOL as when the foreground-background posterior ratio

$$p(Y = 1 | M^{fg}) / p(Y = 1 | M^{bg})$$

Figure 1, written as $\log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$. The score map $f$ is defined as the softmax over $f(X_{ij})$:

$$\log p(Y | X) := \log \text{softmax}^Y \left( \frac{1}{HW} \sum_{ij} f(X_{ij}) \right)$$

(8)
We summarize that how prior works around the binary mask threshold the score map at a min-ping max

**B. Evaluation Protocol for WSOL**

search directions for turning WSOL into a well-posed task. background images containing confusing background cues. WSOL into a more well-posed task also by including many lake cues in non-duck images

\[ \alpha < 1 \text{ implies that lake patches are more abundant in duck images than are duck’s feet (see Figure 9) for an illustration.} \]

To increase \( \alpha \), two approaches can be taken. (1) Increase the likelihood ratio \( \frac{p(\text{duck|feet})}{p(\text{water|duck})} \). This can be done by collecting more images where duck’s feet have more pixels than lake does. (2) Decrease the prior ratio \( \frac{p(\text{feet})}{p(\text{water})} \). Note that the prior ratio can be written

\[
\frac{p(\text{feet})}{p(\text{water})} = \frac{p(\text{feet|duck})p(\text{duck}) + p(\text{feet|duck}^c)p(\text{duck}^c)}{p(\text{water|duck})p(\text{duck}) + p(\text{water|duck}^c)p(\text{duck}^c)}
\]

With fixed likelihoods \( p(\text{feet|duck}) \) and \( p(\text{water|duck}) \), one can decrease the prior ratio by increasing the likelihood of lake cues in non-duck images \( p(\text{water|duck}^c) \). We can alter WSOL into a more well-posed task also by including many background images containing confusing background cues.

Such data-centric approaches are promising future research directions for turning WSOL into a well-posed task.

**B. Evaluation Protocol for WSOL**

**B.1. Score map normalization**

A common practice in WSOL is to normalize the score maps per image because the maximal (and minimal) scores differ vastly across images. Prior WSOL papers have introduced either max normalization (dividing through by \( \max_{ij} \hat{s}_{ij} \)) or min-max normalization (additionally mapping \( \min_{ij} \hat{s}_{ij} \) to zero). After normalization, WSOL methods threshold the score map at \( \tau \) to generate a tight box around the binary mask \( \{ (i, j) | \hat{s}_{ij} \geq \tau \} \). \( \tau \) is typically treated as a fixed value \( \tau \geq 0.2 \) or a hyperparameter to be tuned \( [26, 58, 56] \). We summarize that how prior works calibrate and threshold score maps in Table 3. As discussed in §4.1 of main paper, we use the min-max normalization.

**B.2. Data preparation**

We present the following data-wise contributions in this paper (contributions **bolded**):

- **CUB**: **New data** (5 images per class) with **bounding box annotations**.
- **ImageNet**: ImageNetV2 \[39\] with **new bounding box annotations**.
- **OpenImages**: Processed a split for the WSOL training, hyperparameter search, and evaluation with ground truth object masks.

**B.2.1 ImageNet**

The test set of ImageNet-1k dataset \[42\] is not available. Therefore, many researchers report the accuracies on the validation set for their final results \[56\]. Since this practice may let models overfit to the evaluation split over time, ImageNetV2 \[39\] has been proposed as the new test sets for ImageNet-1k trained models. ImageNetV2 includes three subsets according to the sampling strategies, MatchedFrequency, Threshold0.7, and TopImages, each with 10,000 images (10 images per class). We use the Threshold0.7 split as our train-fullsup. Since ImageNetV2 does not contain localization supervision, we have annotated bounding boxes on those images, following the annotation protocol of ImageNet. The total number of annotated bounding boxes are 18,532.
B.2.2 CUB

We have collected 5 images for each of the 200 CUB fine-grained bird classes from Flickr. The overall procedure is summarized as:

1. Crawl images from Flickr.
2. De-duplicate images.
3. Manually prune irrelevant images (three people).
4. Prune with model classification scores.
5. Resize images.
6. Annotate bounding boxes.

Crawl images from Flickr. Since original CUB itself is collected from Flickr, we use Flickr as the source of bird images. We have used the Flickr API\textsuperscript{1} to crawl up to 400 images per class with class name as search terms. We have only crawled images under the following licenses:

- Attribution
- Attribution-NonCommercial
- Public Domain Dedication
- Public Domain Mark

De-duplicate images. We de-duplicate the images using the ImageHash library\textsuperscript{2} first among the crawled images themselves and then against the training and test splits of CUB.

Manually prune irrelevant images (three people). Since crawled images of each class contain negative-class images (non-birds and birds of wrong categories), three humans have participated in the prune-out process. We have only kept the images that all three have voted for “positive”.

Prune with model classification scores. To match with the original CUB data distribution, we have trained a fine-grained bird classifier using ResNet50 [19] on the CUB training split. Among crawled images, those with ground truth class confidence scores lower than 0.5 have been pruned out.

Achieving 5 images per class. At this point, most classes have more than 5 images per class and a few have less than 5. In the former case, we have randomly sampled 5 images per class.

Resize images. Photographic technologies have made significant advances over the decade since the time original CUB was collected. As a result, image size statistics differ. To fix this, we have resized the images appropriately.

Annotate bounding boxes. The evaluation on CUB is based on bounding boxes. Thus, the held-out set images are supplied with tight bounding boxes around birds (one per image).

B.2.3 OpenImages

There are three significant differences between OpenImages\textsuperscript{3} and CUB or ImageNet that make the OpenImages not suitable as a WSOL benchmark in its original form. (1) Images are multi-labeled; it is not sensible to train classifiers with the standard softmax cross-entropy loss assuming single label per image. (2) OpenImages has less balanced label distributions. (3) There are nice instance segmentation masks, but they have many missing instances.

We have therefore processed a subset of OpenImages into a WSOL-friendly dataset where the above three issues are resolved. The procedure is as follows:

1. Prune multi-labeled samples.
2. Exclude classes with not enough samples.
3. Randomly sample images for each class.
4. Prepare binary masks.
5. Introduce ignore regions.

Prune multi-labeled samples. We prune the multi-labeled samples in the segmentation subset of OpenImages. This process rejects 34.5% samples in OpenImages.

Exclude classes with not enough samples. After pruning multi-label samples, some classes are not available in validation and test sets. We have first defined the minimum number of samples per classes as (300, 25, 50) for (train, validation, test), and have excluded classes that do not meet the minimum requirement, resulting in 100 classes (out of 350 original classes).

Randomly sample images for each class. We have randomly sampled (300, 25, 50) samples per class for (train, validation, test). This eventually results in 29 819

\textsuperscript{1}https://www.flickr.com/services/api/
\textsuperscript{2}https://pypi.org/project/ImageHash/
\textsuperscript{3}https://openimages.co
train-weaksup (train), 2,500 train-fullsup (validation), and 5,000 test (test) samples.

Prepare binary masks. WSOL methods are evaluated against binary masks of the foreground category in each image. Since OpenImages comes with instance-level masks of various categories, we have taken the union of instance masks of the class of interest in every image to build the binary masks.

Introduce ignore regions. OpenImages has instance-wise masks, but not all of them are annotated as masks. When the objects are difficult to be annotated instance-wise or there are simply too many, the “group-of” box annotations are provided over the region containing multiple instances (e.g., hundreds of books on bookshelves in an image taken at a library – See Figure 11). We have indicated the regions with the “group-of” boxes as “ignore” regions, and have excluded them from the evaluation during the computation of $P_x$Prec, $P_x$Rec, and $P_x$AP.

B.3. Transferability of rankings between train-fullsup and test

As detailed in main paper §4.2, we search hyperparameters on train-fullsup and test them on test. To validate if the found hyperparameter rankings do transfer well between the splits, we show the preservation of ranking statistics in Table 4 and visualize the actual rankings in Figure 14. We observe that the rankings are relatively well-preserved (with Kendall’s tau values $> 0.7$).

B.4. Hyperparameter search with proxy train-weaksup

We have reduced the training set size (train-weaksup) to 10% for ImageNet hyperparameter search for the interest of computational efficiency (§4.2). We examine how much this reduction affects the rankings of hyperparameters. We show a similar analysis as in §B.3: see Figure 12. We observe again that with Kendall’s tau value 0.743, the two rankings are largely preserved.

C. Experiments

C.1. Prior WSOL methods and hyperparameters

We describe each method in greater detail here. The list of hyperparameters for each method is in Table 5.

CAM, CVPR'16 [60]. Class Activation Mapping trains a classifier of fully-convolutional backbone with global average pooling structure, and at test time uses the logit outputs before the global average pooling for the scoring rule $s(X_{ij})$. See Appendix §A.1 for the interpretation of CAM as a posterior approximator. CAM has learning rate (LR) and the score-map resolution (SR) as hyperparameters. LR is sampled log-uniformly from $[10^{-5}, 10^0]$, where end points correspond roughly to “no training” and “training always diverges” cases. SR is sampled from Categorical $\{14, 28\}$, two widely used resolutions in prior WSOL methods. All five methods below use CAM technique in the background, and have LR and HR as design! choices.

HaS, ICCV’17 [26]. Hide-and-Seek (HaS) is a data augmentation technique that divides an input image into grid-like patches, and then randomly select patches to be dropped. The hyperparameters of HaS are drop rate (DR) and drop area (DA). Specifically, the size of each patch is decided by DA, and the probability of each patch to be selected for erasing is decided by DR. DA is sampled from a uniform distribution $U[0, 1]$, where 0 corresponds to “no grid” and 1 indicates “full image as one patch”.

ACoL, CVPR’18 [58]. Adversarial Complementary Learning (ACoL) adds one more classification head to backbone networks. From one head, ACoL finds the high-score region using CAM using erasing threshold (ET) and erases it from an internal feature map. The other head
### Table 4. In-distribution ranking preservation.

Kendall’s τ values for the hyperparameter ranking between train-fullsup and test are shown.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Hyperparameter</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Learning rate</td>
<td>LogUniform[$10^{-5}$, $10^9$]</td>
</tr>
<tr>
<td></td>
<td>Score-map resolution</td>
<td>Categorical[14, 28]</td>
</tr>
<tr>
<td>HaS [26]</td>
<td>Drop rate</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td></td>
<td>Drop area</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td>ACoL [58]</td>
<td>Erasing threshold</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td>SPG [59]</td>
<td>Threshold $\delta_{l1}$</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td></td>
<td>Threshold $\delta_{l1}$</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td></td>
<td>Threshold $\delta_{l2}$</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td></td>
<td>Threshold $\delta_{l2}$</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td></td>
<td>Threshold $\delta_{c}$</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td>ADL [6]</td>
<td>Drop rate</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td></td>
<td>Erasing threshold</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td>CutMix [56]</td>
<td>Size prior</td>
<td>Uniform[0, 1]</td>
</tr>
<tr>
<td></td>
<td>Mix rate</td>
<td>Uniform[0, 1]</td>
</tr>
</tbody>
</table>

### Table 5. Hyperparameter search spaces.

The widely-used “top-1 localization accuracy” for WSOL represents both the classification and localization performances. The metric can be misleading as a localization metric, as the increase in numbers can also be attributed to the improved classification accuracies. Thus, in the main paper, we have suggested using on the “GT-known” type of metrics like MaxBoxAcc and PxAP that measures the localization performances given perfect classification.

Here, we measure the classification accuracies of the models in Table 2 of the main paper, to complete the analysis. The performances are reported in Table 6. There are in general great fluctuations in the classification results (26.8% to 74.5% on CUB). This is because we have selected the hyperparameters for each model that maximize the localization performances on train-fullsup split. In many cases, the best localization performances are achieved at early epochs, before the classifiers are sufficiently trained (see also §C.6 and Figure 20). The result signifies that localization and classification performances may not necessarily correlate and, therefore, localization-only metrics like MaxBoxAcc and PxAP must be used for model selection and evaluation of WSOL methods.
Table 6. Classification performance of WSOL methods. Classification accuracies of the models in Table 2 of the main paper are shown. Hyperparameters for each model are optimally chosen for the localization performances on train-fullsup split. Classification performances may be sub-optimal when the best localization performance is achieved at early epochs.

<table>
<thead>
<tr>
<th>Methods</th>
<th>ImageNet</th>
<th>CUB</th>
<th>OpenImages</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VGG</td>
<td>Inception</td>
<td>ResNet</td>
<td>Mean</td>
</tr>
<tr>
<td>CAM [60]</td>
<td>63.8</td>
<td>68.7</td>
<td>75.9</td>
<td>69.5</td>
</tr>
<tr>
<td>HaS [26]</td>
<td>61.9</td>
<td>65.5</td>
<td>63.1</td>
<td>63.5</td>
</tr>
<tr>
<td>ACoL [58]</td>
<td>60.3</td>
<td>64.6</td>
<td>61.6</td>
<td>62.2</td>
</tr>
<tr>
<td>SPG [59]</td>
<td>61.6</td>
<td>65.5</td>
<td>63.4</td>
<td>63.5</td>
</tr>
<tr>
<td>ADL [6]</td>
<td>60.8</td>
<td>61.6</td>
<td>64.1</td>
<td>62.2</td>
</tr>
<tr>
<td>CutMix [56]</td>
<td>62.2</td>
<td>65.5</td>
<td>63.9</td>
<td>63.8</td>
</tr>
</tbody>
</table>

Table 7. Previously reported vs our results. The first six rows are reported results in prior WSOL papers. When there are different performance reports for the same method in different papers, we choose the greater performance. The last six rows are our re-implemented results under the new evaluation method (§4).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Top-1 localization accuracy</th>
<th>GT-known localization; MaxBoxAcc and PxAP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ImageNet</td>
<td>CUB</td>
</tr>
<tr>
<td>CAM [60]</td>
<td>42.8</td>
<td>46.3</td>
</tr>
<tr>
<td>HaS [26]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ACoL [58]</td>
<td>45.8</td>
<td>-</td>
</tr>
<tr>
<td>SPG [59]</td>
<td>-</td>
<td>48.6</td>
</tr>
<tr>
<td>ADL [6]</td>
<td>44.9</td>
<td>48.7</td>
</tr>
<tr>
<td>CutMix [56]</td>
<td>43.5</td>
<td>47.3</td>
</tr>
</tbody>
</table>

Table 8. Re-evaluating WSOL with MaxBoxAccV2. We re-evaluate six recently proposed WSOL methods with MaxBoxAccV2. The experimental setting is the same as that of Table 2 in the main paper.

<table>
<thead>
<tr>
<th>Methods</th>
<th>ImageNet (MaxBoxAccV2)</th>
<th>CUB (MaxBoxAccV2)</th>
<th>OpenImages (PxAP)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAM [60]</td>
<td>60.0</td>
<td>63.4</td>
<td>63.7</td>
<td>62.4</td>
</tr>
<tr>
<td>HaS [26]</td>
<td>+0.6</td>
<td>+0.3</td>
<td>-0.3</td>
<td>+0.2</td>
</tr>
<tr>
<td>ACoL [58]</td>
<td>-2.6</td>
<td>+0.3</td>
<td>-1.4</td>
<td>-1.2</td>
</tr>
<tr>
<td>SPG [59]</td>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.4</td>
<td>-0.2</td>
</tr>
<tr>
<td>ADL [6]</td>
<td>-0.2</td>
<td>-2.0</td>
<td>+0.0</td>
<td>-0.7</td>
</tr>
<tr>
<td>CutMix [56]</td>
<td>-0.6</td>
<td>+0.5</td>
<td>-0.4</td>
<td>-0.2</td>
</tr>
</tbody>
</table>

Table C.3. Reproducing prior WSOL results

We also summarize reported results in prior WSOL papers [60, 26, 58, 59, 6, 56] along with our reproduced results in Table 7. While our main evaluation metrics are the classification-disentangled GT-known measures (§4.1 in main paper), we also report the top-1 localization metrics that also measure the classification scores to match the reported numbers. Note that we use oracle $\tau$ for Top-1 localization accuracy.

We experimentally observe that training epochs and batch sizes influence the localization accuracies (~10pp in MaxBoxAcc). However, the training details for each method are not given by the corresponding papers.
methods should share the same training budget for fair comparisons. Therefore, we fix our training epochs to (10, 50, 10) for (ImageNet, CUB, OpenImages) experiments, and fix the batch size to 32 for all datasets.

Following the prior methods [60, 26, 58, 59, 6, 56], we train all six methods by fine-tuning ImageNet pre-trained model. Specifically, we apply $10 \times$ scale of learning rate to higher-level layers of backbone networks during training. We consider the last two layers for VGG, last three layers for InceptionV3, and last three residual blocks, and fully connected layers for ResNet as the higher-level layers.

Note that with GT-known metrics, our re-implementations produce better performances than the previously reported results (e.g., 62.7 $\rightarrow$ 65.3 for CAM on ImageNet with Inception backbone). This is perhaps because MaxBoxAcc and PxAP are based on the best operating thresholds.

Top-1 localization accuracies are reproduced well in general, except for the ADL: e.g., 52.4 $\rightarrow$ 39.2 for ADL on CUB with VGG backbone. This is due to the reduced training epochs compared to the original paper [6], which results in a decreased classification accuracies.

### C.4. Score calibration and thresholding

The operating threshold $\tau$ for the score map $s$ is an important parameter for WSOL. We show additional plots and visualizations to aid understanding. In Figure 15, we show the BoxAcc and PxPrec-PxRec performances at different operating thresholds with diverse architectures and datasets. It extends the main paper Figure 5. We observe again that the optimal operating thresholds $\tau^*$ are vastly different across data and architectures for BoxAcc. The MaxBoxAcc in each method are relatively stable across methods especially on ImageNet. OpenImages PxPrec-PxRec curves do not exhibit big differences amongst WSOL methods, likewise.

In Figure 13, we visualize score distributions for different WSOL methods. We observe that methods have different distributions of scores; ACoL in particular tends to generate flatter score maps. Comparing dataset, we observe that OpenImages tends to have more peaky score distributions. It is therefore important to find the optimal operating point for each method and dataset for fair comparison.

We visualize more score maps in Figure 16, 17, 18; it extends the main paper Figure 4. We observe qualitatively different score maps in general across methods. However, the optimal IoU values are not as different and hard to predict just by visually looking at the samples. Again, it is important to have an objective way to set the thresholds $\tau$ in each case.

### C.5. Hyperparameter analysis

We observe some trends in score distribution as in the main paper. (1) Performances do vary according to the hyperparameter choice, so the hyperparameter optimization is necessary for the optimal performances. (2) CAM is among the more stable WSOL methods. (3) ACoL and ADL show greater sensitivity to hyperparameters in general. (4) CUB is a difficult benchmark where random choice of hyperparameters is highly likely to lead to performances worse than the center-Gaussian baseline.


We observe similar trends as in the main paper. (1) Performances do vary according to the hyperparameter choice, so the hyperparameter optimization is necessary for the optimal performances. (2) CAM is among the more stable WSOL methods. (3) ACoL and ADL show greater sensitivity to hyperparameters in general. (4) CUB is a difficult benchmark where random choice of hyperparameters is highly likely to lead to performances worse than the center-Gaussian baseline.

### C.7. Evaluating WSOL methods with MaxBoxAccV2

We re-evaluate six recently proposed WSOL methods with MaxBoxAccV2, and the results are shown in Table 8. All training configurations but evaluation metrics are the same as those of the main paper. We evaluate the last checkpoint of each training session. We obtain the same conclusions as with the original metric, MaxBoxAcc: (1) there has been no significant progress in WSOL performances beyond vanilla CAM [60] and (2) with the same amount of fully-supervised samples, FSL baselines provide better performances than the existing WSOL methods.
Figure 14. **Preservation of hyperparameter rankings.** Ranking of hyperparameters is largely preserved between train-weaksup and test. We only show the converged sessions (§5.4). Results on ResNet50, OpenImages.
Figure 15. **Performance by operating threshold $\tau$.** CUB and ImageNet: BoxAcc versus $\tau$, OpenImages: $P_xPrec$ versus $P_xRec$. ResNet, VGG, and Inception architecture results are used. This is the extension of Figure 5 in the main paper.
Figure 16. **ImageNet score maps.** Score maps of CAM, HaS, ACoL, SPG, ADL, CutMix from ImageNet.
Figure 17. **CUB score maps.** Score maps of CAM, HaS, ACoL, SPG, ADL, CutMix from CUB.
Figure 18. **OpenImages score maps.** Score maps of CAM, HaS, ACoL, SPG, ADL, CutMix from OpenImages.
Figure 19. All results of the 30 hyperparameter trials. CUB, ImageNet, OpenImages performances of all 30 randomly chosen hyperparameter combinations for each method. This is the extension of Figure 6 in the main content.
Figure 20. **Learning curves.** Results on three datasets with VGG, Inception and ResNet architectures.