Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Will be included in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A WWbl algorithm without selective search

In this section, we present an algorithm for wwbl without the selective search method for describing the objects in the image.

The algorithm runs N iterations with a break option when no object was found. In line 2 of the listed algorithm, we obtain the caption of the input image $T_0 = BLIP(I)$.

Lines 4-16 of Alg. 2 describe the iterative algorithm used to extract objects from a single image. The first operation is to encode the text in the current iteration with the CLIP text encoder. The next step is to generate a map with g. From the output map M_i we extract the largest bounding box (B_i), using the algorithm described in section 3.1. For the extracted bounding box, the algorithm crops a relative patch from the image (P_i), and updates the image, such that the patch pixels are zero. The new image is used for generating the new caption $T_i^P = BLIP(P_i)$. Next, the algorithm calculates the similarity between $E_t(T_0)$ and $E_t(T_i^P)$. Only if the similarity is above $\tau = 0.6$ does the algorithm add this bounding box to the predicted output and continues with the iterations. Otherwise, it stops the routine and returns the predicted output.

Tab. 8 compares the performances of algorithm 1 and 2. Evidently, while Alg. 2 addresses WWbL to some degree, Alg. 1 is considerably better.

Algorithm 2 WWbL Inference

Require: Input image I
1: Load networks g, CLIP, BLIP
2: $T_0 = BLIP(I)$
3: $D \leftarrow \phi$ ▷ Empty output dictionary
4: **for** $i = 0....n-1$ **do**
5: $Z_i = E_t(T_i)$ ▷ CLIP text encoder
6: $M_i = g(I, Z_i)$ ▷ Generate map
7: $B_i \leftarrow C(M_i)$ ▷ Extract bounding box
8: $P_i \leftarrow I[B_i]$ ▷ Crop patch
9: $T_i^P = BLIP(P_i)$ ▷ Caption patch
10: $S = CLIP(T_i^P, T_0)$
11: **if** $S \geq 0.6$ **then**
12: $D = D \cup (B_i, T_i^P)$ ▷ Add object
13: $I[B_i] = 0$ ▷ Delete the object
14: $T_{i+1} = BLIP(I)$ ▷ New caption
15: **else**
16: break
17: return D

Table 8: WWbL results: "pointing game" accuracy on Visual Genome (VG), Flickr30K, and ReferIt

Method	Backbone	Training	Test Accuracy		
			VG	Flickr30K	ReferIt
Algorithm 2	CLIP+VGG	VG	35.02	42.57	37.56
Algorithm 1	CLIP+VGG	VG	**43.91**	**58.59**	**44.89**
Algorithm 2	CLIP+VGG	COCO	32.80	41.86	37.08
Algorithm 1	CLIP+VGG	COCO	**44.20**	**61.38**	**43.77**

B Visualization of baselines - task WSG

We present visualization of our network g for different scenarios, together with the GAE method, to emphasize the refinement of our network, as can be seen at Fig. 6. The proposed loss terms for training g encourage the output mask to capture the whole shape of the object instead of specific regions of GAE which tend to focus more on discriminate regions.

Figure 6: Sample phrase-grounding results for our method compared with GAE of CLIP [12] where (a) the input image (b) our output map (c) GAE of CLIP

C λ_3 sensitivity

The proposed loss terms have four coefficients that balance the final loss function. To avoid an excessive search for hyperparameters, we set λ_1, λ_2 and λ_4 to one and only set λ_3 at a different value.

In Tab. 9 we study the effect of λ_3 on the performance of the learned g for the WSOL task on the CUB benchmark [76] and for WSG on Filckr30K. Evidently, the value of $\lambda_3 = 4$ is optimal for both experiments (this is the value used throughout our work). However, performance is relatively stable in a wide range of values. As noted in the ablation experiments of the paper, a value of zero (canceling this term), leads to a drop of 6% for WSOL on CUB. The current results indicate an even larger drop for Phrase grounding.

Table 9: The effect of varying λ_3 on the accuracy of weakly supervised localization, as evaluated on CUB, and on that of phrase grounding, evaluated on Flickr30K.

λ_3	0	0.01	0.1	1	2	4	8	16
CUB localization accuracy[%]	90.0	88.7	90.6	92.6	94.1	96.5	93.8	93.23
Phrase grounding accuracy [%]	41.9	41.3	44.5	61.8	73.1	75.6	75.4	75.1

D Visualization of bounding box extraction from localization map

In this section, we present visualization for bounding box extraction, which is used for WWbl, and also for WSG. We first set each pixel in the localization map that is below 50% under the maximum value in the map to zero. Next, we apply the algorithm of Suzuki et al. [70] to find the contours of all objects in the map. We consider the bounding box of each contour. We then calculate the sum of all values of the map g inside the bounding box, which we refer to as "energy". The final stage is to filter out bounding boxes via non-maximal suppression using the energy values, for all overlapping bounding boxes with an IOU larger than 0.3. The bounding boxes are then filtered further, keeping only the ones with energy values of at least 50% of that of the bounding box with the maximal energy.

Fig. 7 presents a visualization of the method where (a) the input image (b) the localization map (c) localization map after thresholding with bounding box proposals (d) localization map after thresholding with the final bounding boxes (e) the input image with the final bounding boxes. As can be seen, the first step of zeroing low-value pixels allows the method to focus on the relevant regions in the image based on the text. The importance of filtering non-relevant bounding boxes based on NMs and low energy also rises from the next examples.

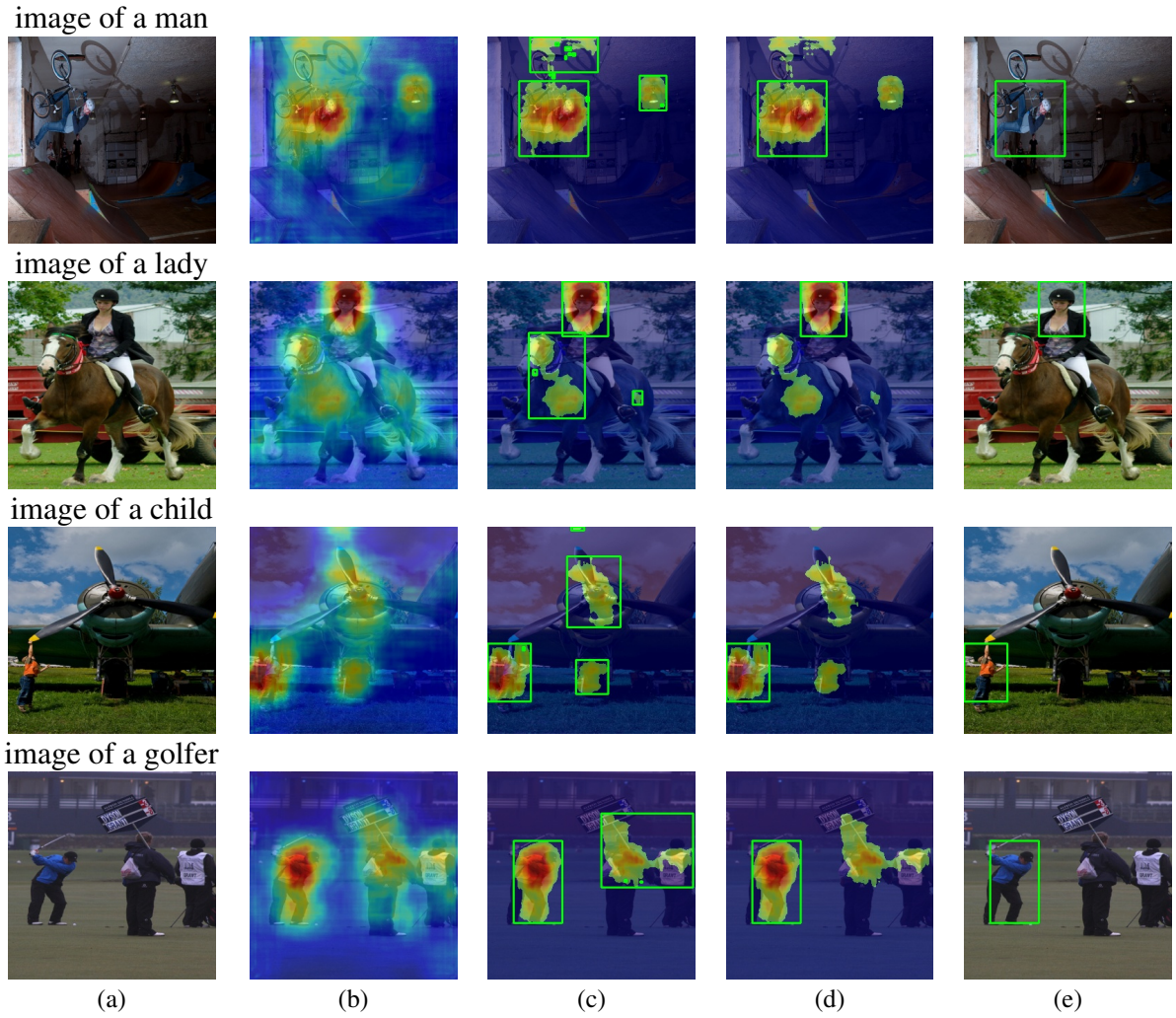

Figure 7: Visualization of bounding box extraction from localization map where (a) the input image (b) localization map (c) localization map after thresholding with bounding box proposals (d) localization map after thresholding with the final bounding boxes (e) the input image with the final bounding boxes

E WSOL visualization

In this section, we present a visualization of WSOL task for fine-grained datasets where the first row is the Stanford-cars dataset, the second row is the CUB dataset, and the last row is Stanford-dogs. Fig. 8 present our results together with GAE[12] which emphasizes our improvements for the relevant map of the GAE (this is used, during training only, as one of our losses) for the localization task.

F Multiple instances of the same object

In Fig. 9, we present our method's results for images with multiply instances of the same object, such as apples and dogs. On the left side, we present the results of the WWbL method. On the right side, we present results for grounding specific sentences that are provided to study the output of network g.

As can be seen, WWbL does not typically select sentences that distinguish between the various objects. However, network g can separate between different objects of the same class given specific captions. When there are multiple objects of the same type, e.g., multiple green apples, g marks all of them. We note that g's heatmap does peak at specific parts of the objects, which may facilitate instance separation (this direction is out of scope for the current study).

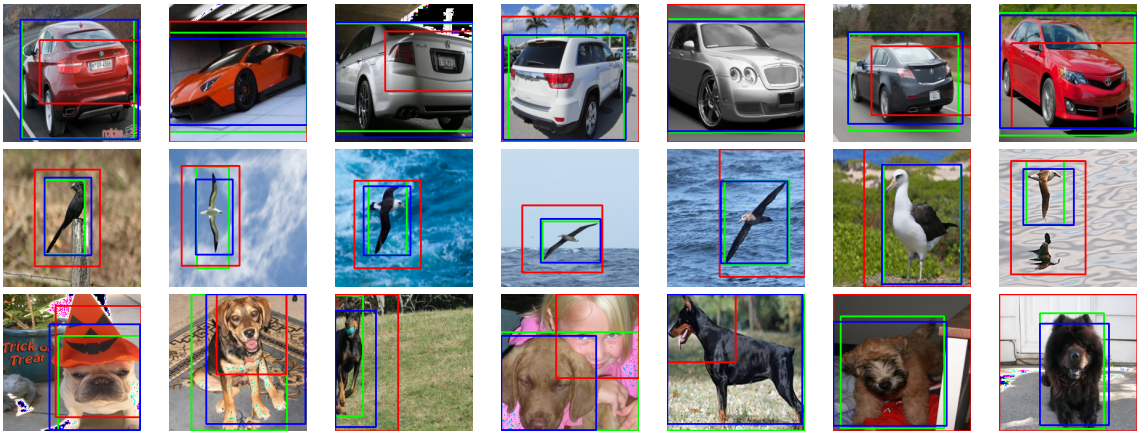

Figure 8: Sample localization results, where the first row is the visualization of the Stanford-cars dataset, the second row is the CUB dataset, and the last row is Stanford-dogs. The green bounding box is the ground-truth, the red is the bounding box of the CLIP explainability map, obtained wtih GAE, and the blue bounding box is the output of our algorithm.

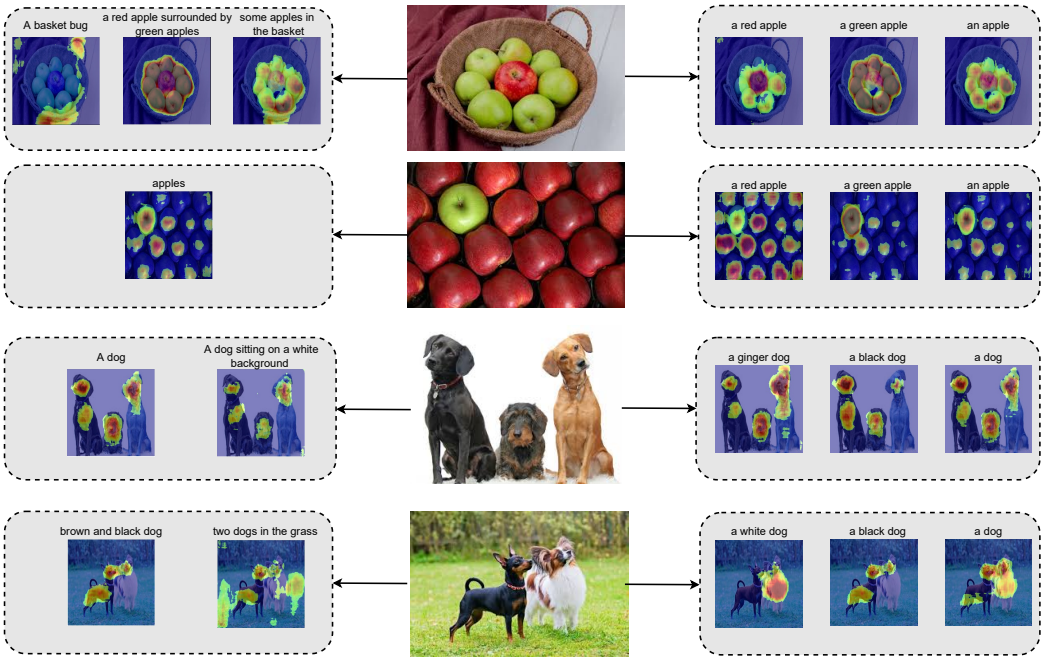

Figure 9: The heatmap obtained by network g for WWbL predictions (left side) and specific captions (right side).

G Employing different pre-trained CLIP models

In this section, we examine the ability of g to perform weakly supervised localization on CUB and Filckr30K for WSG task, when employing CLIP that is pre-trained on different datasets.

The models were taken from the OpenCLIP open-source repository [32], which provides both the pre-trained CLIP models and the obtained imagenet1K zero-shot accuracy.

As can be seen in Tab. 10, while the zero-shot classification varies considerably, the performance of g on the localization task varies much less. Also, there does not seem to be a strong correlation between the amount of training data used for training CLIP and the localization performance of g.

A similar experiment was conducted for the phrase grounding task, with similar results. As Tab. 10 indicates, the ResNet50 model slightly outperforms the results obtained by the ViT32 model, when both are trained on the openai dataset. Other datasets, both bigger and smaller, do not lead to better performance.

We note that the gap observed for both WSOL and Phrase Grounding is much narrower than the gap that we see in top-1 accuracy for zero-shot classification. This indicates that our method is relatively stable concerning the CLIP model used during training.

Table 10: Results obtained by network g in the task of (1) weakly supervised localization on the CUB benchmark, (2) phrase grounding on the Flicker benchmark, for different CLIP models. Also reported is the top-1 zero shot accuracy on ImageNet.

Clip version			Performance metrics		
Clip backbone	CLIP training dataset	Num. samples	CUB-Acc[%]	Flicker-Acc[%]	top1 ImageNet Acc[%]
RN50	openai	**400M**	89.3	**76.0**	**59.8**
RN50	yfcc15m	15M	88.1	72.9	32.4
RN50	cc12m	12M	**94.1**	73.3	35.9
RN101	openai	**400M**	**96.5**	**75.1**	**62.3**
RN101	yfcc15m	15M	87.2	69.5	34.0
ViT32	openai	400M	**96.5**	**75.6**	63.4
ViT32	laion2be16	**2B**	92.5	73.4	**65.6**
ViT32	laion400me32	400M	94.2	75.3	60.2