LocLoc-Supplementary

A PROCEDURE DETAILS OF LOW-LEVEL CUES ON GRABCUT

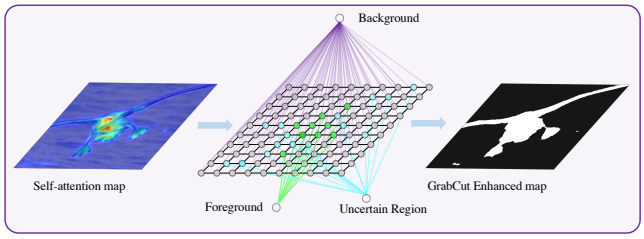

Figure 1: GrabCut algorithm in Low-level cues. In this diagram, GrabCut establishes a Gaussian model bases on the foreground and background pixels from original image, corresponding to the area of attention in self-attention map. Then identifies the uncertain pixels between foreground and background, ultimately producing a GrabCut Enhanced mask.

GrabCut is an image segmentation algorithm explained in Section 3.2 "GrabCut-Enhanced Mask". It utilizes pixel correlations to initialize the foreground and background and classify each pixel accordingly. However, inaccurate initialization of the foreground and background in GrabCut can impact the subsequent modeling. To address this, we propose leveraging the self-attention map generated by DINO [1] to guide GrabCut in identifying background pixels (gray circles in Figure 1(a)). This guidance leads to better segmentation.

Low-level cues refer to the pixel-level relationships in an image that facilitate the separation of foreground and background objects. In the middle figure, the green circles represent foreground pixels, the gray circles represent background pixels, and the blue circles represent uncertain pixels. The self-attention mechanism of DINO [1] allows us to identify foreground and background pixels. By leveraging the low-level cues from the foreground and background obtained through DINO, we determine whether the uncertain pixels belong to the foreground or background, resulting in a higher-quality mask.

B COMPARISON WITH BASELINE PPD

In this section, we compare our LocLoc approach to the baseline PPD [4] in terms of architecture, motivation, inference time and resource consumption.

B.1 Different architecture

As shown in code B.3, PPD requires four computations as images are loaded separately for the classifier and generator. In contrast, our approach (code B.3) utilizes the frozen ViT and performs only three computations, allowing the generation of Patch tokens and separate input into the classifier and generator, as mentioned earlier.

Method	Backbone	Classifier	Generator	Total
PPD	20.66M	0.30M	16.76M	37.72M
Ours	-	24.87M	3.86M	49.49M (+12M)

Table 1: Parameters of PPD and our method. we adopt DINO [1] as backbone and there approximately 12M parameters increase in our method.

B.2 First to use Low-level cues in WSOL

Previous methods [2, 3] found low-level features are highly effective for precise localization. However, these methods typically utilizes low-level features at the early layers of the network. Wei et al. [3] utilized the low-level features from shallow layers of the network to produce initial CAM. On the other hand, Jiang [2] discovered that the low-level layers contain more object regions and consequently generated LayerCAMs in these layers.

In contrast, we are the pioneers in considering low-level cues from the pixel-level relationships within the image itself, rather than solely relying on features extracted by the model, distinguishing our work from previous methods.

B.3 Less inference time and resource consumption

Table 1 shows the parameters of PPD and our method. Despite the increase of approximately 12M parameters in our models, we achieve a **31.58%** reduction in total inference time compared to PPD as shown in Table 2.

This is attributed to two factors.

- Our generator utilizes a lightweight UNet architecture (3 layers of down-sampling and 3 layers of up-sampling) with a total of 3.86M parameters, which is only **1/4** of PPD, which allows us to generate feature maps with smaller channels yet richer semantic information, effectively reducing parameter consumption.
- PPD requires separate image transformations for classification and localization, resulting in additional time spent on data loading. PPD performs **4** times computations to generate different features for classifier and generator, whereas our method only requires **3** times.

Meanwhile, we achieve a **82.01%** reduction in GFLOPS. In a segmentation network, the extraction of semantic information is known to be time-consuming. However, our lightweight architecture allows for significant reduction in model GFLOPS and inference time required for semantic information extraction.

Note: All the results are conducted on Nvidia Tesla V100 GPUs.

```
    #Using backbone to extract class tokens
    intermediate_output = model.get_intermediate_layers(
     cls_image, n_last)
    output = [x[:, 0] for x in intermediate_output]
    feat = [x[:, 1:] for x in intermediate_output]
    if avgpool:
        output.append(torch.mean(intermediate_output
     [-1][:, 1:], dim=1))
```

Method	Classifier	Generator	Total Inference Time
PPD	8s	12s	19s
Ours	9s	10s	13s (-6)

Table 2: Comparison of inference time on CUB-200-2011 test set (per batch with 32 images). Our proposed method achieve a decreasing of total inference time by 31.58%. The total inference time are calculate on one batch with 32 images, includes data loading time.

Method	Classifier	Generator
PPD	0.30 MFLOPS	1.39 GFLOPS
Ours	7.50 MFLOPS	0.25 GFLOPS

Table 3: Comparison on resource consumption on CUB-200-2011 test set (one batch with 32). Our proposed method achieve a decreasing of total GFLOPS by 82.01%.

```
    output = torch.cat(output, dim=-1)
    feat = torch.cat(feat, dim=-1)

    #Using classifier for classification
    cls_logits = linear_classifier(output)
    _, topk_idx = cls_logits.topk(5, 1, True, True)
    topk_idx = torch.cat([topk_idx,target.unsqueeze(1)],
     dim=1)
    topk_idx = topk_idx.tolist()

    #Using backbone to extract patch tokens
    patch_embed = model.get_intermediate_layers(input,1)
     [0]
    hw = int((patch_embed.shape[1]-1)**0.5)
    patch_embed = patch_embed[:,1:,:].permute([0, 2, 1]).
     contiguous()
    patch_embed = patch_embed.reshape(bs,384,hw,hw)

    #Using generator for generator
    locmap = decov_regressor(torch.from_numpy(patch_embed
     .cpu().numpy()).cuda())
```

InferencePPD.py

```
    #Using backbone extract class tokens and patch tokens
    prepare_tokens, patch_embed = model_pre.
     forward_features(images)

    #Using classifier for classification
    output = model_classifier(prepare_tokens)
    _, pre_logits = output.topk(5, 1, True, True)

    #Using generator for generator
    hw = int((patch_embed.shape[1])**0.5) #14
    patch_embed = patch_embed.permute([0, 2, 1]).
     contiguous()
    patch_embed = patch_embed.reshape(batch,384, hw, hw)
    loc_maps = model_generator(torch.from_numpy(
     patch_embed.cpu().numpy()).cuda())
```

InferenceLocLoc.py

C ALGORITHM

Algorithm 1 Training algorithm for LocLoc

Require: Training data set $I = \{(I_i, y_i)\}_{i=0}^{M-1}$

 # GrabCut Enhanced Mask
Extracting self-attention map $\boldsymbol{F}^d \leftarrow DINO$
Obtain Grabcut Enhanced Mask $M^G \leftarrow (\boldsymbol{F}^d, GMM)$
 # Training GEG
while epoch < GEG epochs **do**
  Calculate patch embedding $\boldsymbol{F}^i$ from frozen model
  Learn global semantic features $\boldsymbol{F}^u \leftarrow f^u(\boldsymbol{F}^i)$ ▷ Eq (??).
  Activate the feature maps:
  $\boldsymbol{F}^a \leftarrow Sigmoid(BN(Conv(f^d(\boldsymbol{F}^u))))$ ▷ Eq (??).
end while
 # Training LFDM
while epoch < LFDM epochs **do**
  Calculate enhanced image $\tilde{I} \leftarrow (I, \tilde{M}^G)$ ▷ Eq (??).
  Learn local representations $\boldsymbol{F}^{\leftarrow} fc(f^t(\tilde{I}))$ ▷ Eq (??).
end while

REFERENCES

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging Properties in Self-Supervised Vision Transformers. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 9630–9640. https://doi.org/10.1109/ICCV48922.2021.00951

[2] Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. 2021. LayerCAM: Exploring Hierarchical Class Activation Maps for Localization. *IEEE Trans. Image Process.* 30 (2021), 5875–5888. https://doi.org/10.1109/TIP.2021.3089943

[3] Jun Wei, Qin Wang, Zhen Li, Sheng Wang, S. Kevin Zhou, and Shuguang Cui. 2021. Shallow Feature Matters for Weakly Supervised Object Localization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 5993–6001. https://doi.org/10.1109/CVPR46437.2021.00593

[4] Jingyuan Xu, Hongtao Xie, Chuanbin Liu, and Yongdong Zhang. 2022. Proxy Probing Decoder for Weakly Supervised Object Localization: A Baseline Investigation. In *MM '22: The 30th ACM International Conference on Multimedia, Lisboa, Portugal, October 10 - 14, 2022*, João Magalhães, Alberto Del Bimbo, Shin'ichi Satoh, Nicu Sebe, Xavier Alameda-Pineda, Qin Jin, Vincent Oria, and Laura Toni (Eds.). ACM, 4185–4193. https://doi.org/10.1145/3503161.3547945