Looking Beyond the Window: Global-Local Aligned CLIP for Training-free Open-Vocabulary Semantic Segmentation

Supplementary Material

A. Boundary Error Rate (BER)

A.1. Definition of BER

In Fig. 1, to measure prediction inconsistencies caused by the sliding-window mechanism, we introduce the Boundary Error Rate (BER). BER measures how often adjacent pixel pairs across window boundaries, represented as $\{p, q\}$ in the equation, share the same ground-truth label but receive different predictions.

$$\text{BER} = \frac{\sum_{(p,q)\in\mathcal{B}} \mathbf{1}[(y_p = y_q) \wedge (\hat{y}_p \neq \hat{y}_q)]}{\sum_{(p,q)\in\mathcal{B}} \mathbf{1}[y_p = y_q]} \times 100 \quad (16)$$

Notation:

- p, q: A pair of adjacent pixels located across a sliding window boundary.
- $\mathcal{B} = \{(p, q)\}$: The set of all adjacent pixel pairs across window boundaries.
- y_p, y_q: Ground-truth labels of pixels p and q.
- $\hat{y}_p, \hat{y}_q$: Predicted labels of pixels p and q.
- $\mathbf{1}[\cdot]$: Indicator function.

The denominator counts the number of adjacent pixel pairs across window boundaries that share the same ground-truth label. The numerator counts how many of those are predicted as different classes. BER reflects the proportion of inconsistencies.

A.2. Complementarity of BER

BER captures sliding-window grid artifacts overlooked by mIoU. In Fig. 7, ours shows better sample mIoU than ProxyCLIP [24] & CASS [22] and also produces natural masks with fewer grid artifacts, which BER correctly reflects. At dataset-level, gains in both mIoU & BER shows accurate pixel classification and visually coherent segmentation.

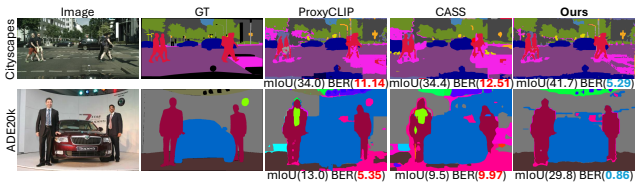


Figure 7. Qualitative Results with Sample-wise mIoU and BER.

B. Desciption of each setting

B.1. CLIP-DINOiser Setting

We follow CLIP-DINOiser [45] setting in Tab. 1 to conduct fair comparison. We resize input images with a short side of 448 and perform sliding-window inference with a 448 × 448 window and 224 stride for all dataset. We do not perform rename trick on each classnames and use only the standard ImageNet prompts following [46, 53]. Additionally, naive CLIP-DINOiser setting does not offer the background threshold unlike ClearCLIP [23, 24, 42], we follow the background threshold of ClearCLIP to identify background in Pascal VOC21 [17] , Pascal Context60 [32] , COCO-Object [4].

B.2. ClearCLIP Setting

We follow ClearCLIP [23] setting in Tab. 1 to conduct fair comparison. We resize input images with a short side of 448 and perform sliding-window inference with a 448 × 448 window and 224 stride for all dataset. We do not perform rename trick on each classnames and use only the standard ImageNet prompts following [46, 53]. The background threshold for Pascal VOC21 [17], Pascal Context60 [32], COCO-Object [4] is assigned to 0.5, 0.15, 0.4, for each.

B.3. ProxyCLIP Setting

In ProxyCLIP [24] setting in Tab 1, We resize the images to accommodate varying dataset specification: a shorter side of 336 pixels for PASCAL [17, 32] and COCO [4] datasets and 448 pixels for Cityscapes [12] and ADE20K [52] datasets. We adopt a sliding-window strategy with a 336 × 336 window and 112 × 112 stride. For the background class, rather than directly using the text prompt "background", we employ a renaming strategy in which multiple class names associated with background semantics are grouped and used as substitutes. This follows the official ProxyCLIP implementation. The background threshold for Pascal VOC21 [17], Pascal Context60 [32], COCO-Object [4] is assigned to 0.2, 0.15, 0.25, for each.

B.4. SCLIP Setting

In SCLIP [42] setting in Tab 1, We resize input images with a short side of 336 and perform sliding-window inference with a 224 × 224 window and 112 stride. Only for the Cityscapes, we resize the short side of 560. We use rename trick on both background class and several other classes, following official SCLIP code. The background threshold

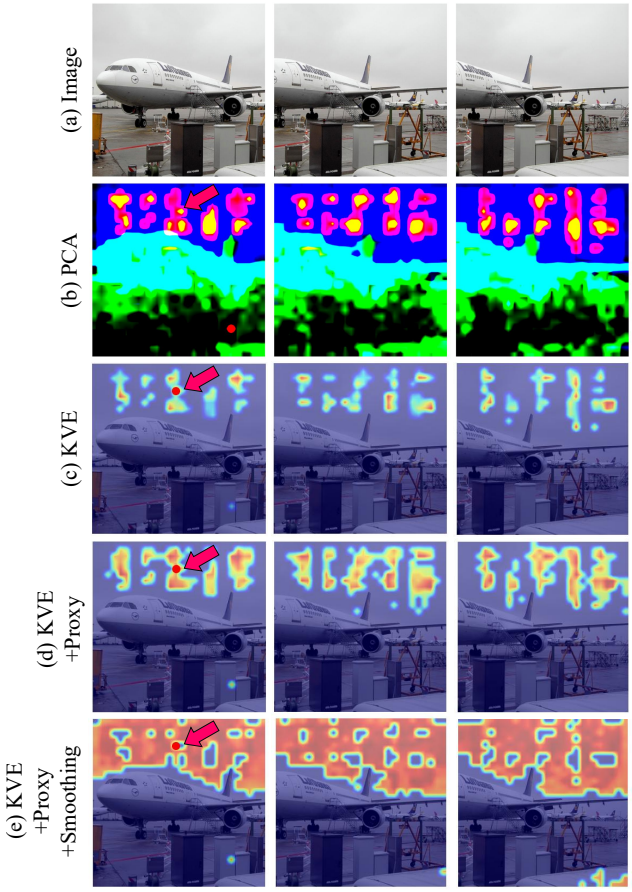


Figure 8. Effect of Smoothing on ClearCLIP attention map

for Pascal VOC21 [17], Pascal Context60 [32], COCO-Object [4] is assigned to 0.1, 0.1, 0.1, for each.

C. Query Smoothing for Proxy-Anchor Construction in CLIP features

As we mentioned in the Implementation Details (Sec. 4.1), our method incorporates a query smoothing step during proxy-anchor construction to ensure stable attention when generating attention maps from CLIP internal features, as in ClearCLIP [23]. This query smoothing is necessary because CLIP feature maps contain noisy high-norm patches and these high-norm patches lost its semantic [13]. By this reason high-norm patches do not have to be included in attention process. However, when the query token itself corresponds to a high-norm patch, the model tends to propagate this noise by repeatedly attending to other high-norm patches across windows, ultimately leading to incorrect semantic decisions. To mitigate this issue, we replace each raw query token with a smoothed token obtained by averaging it with neighbor patches. Specifically, we average each token with its eight spatial neighbors within the inner-window and with the patches at the same spatial position in adjacent overlapping windows:

$$\mathbf{Q}_i^{(0)} = \frac{\mathbf{Q}_i + \sum_{j\in N(i)} \mathbf{Q}_j + \sum_{k\in O(i)} \mathbf{Q}_k}{1 + |N(i)| + |O(i)|}. \tag{17}$$

Here, $N(i)$ denotes the local neighbors within the innner-window, and $O(i)$ represents the overlapping patches from adjacent outer-windows. This averaging suppresses high-norm activations of query token before the progressive positive mining step, allowing the model to aggregate meaningful positives across windows without reinforcing noisy high-norm responses.

Qualitatively, Fig. 8 illustrates this process. In Fig. 8(a-b), the cropped images and its pca visualization identify the location of the high-norm patches. When adapting Key-Value Extension, high-norm patches in background regions (e.g., sky) activate attention both within inner and outer windows (Fig. 8(c)), and even with proxy-anchor stabilization, these noisy patches can dominate the attention map (Fig. 8(d)). By replacing query tokens with their smoothed versions (Eq. 17), attention is redirected toward semantically meaningful tokens (Fig. 8(e)), demonstrating that our method effectively suppresses noisy high-norm patches and constructs stable attention maps.

D. Semantic Collapse in Proxy Anchors

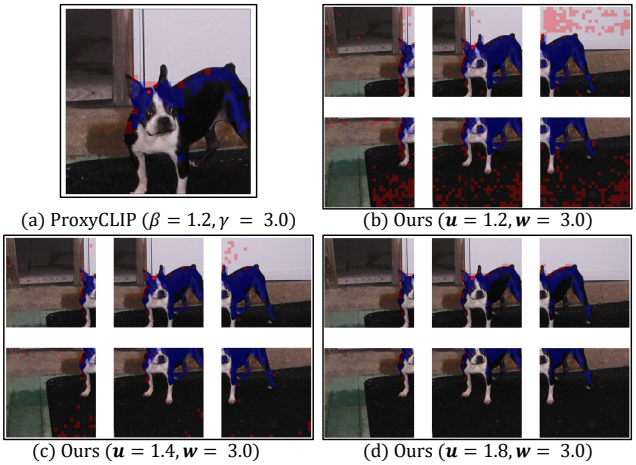
We examine the potential risk of semantic collapse during proxy-anchor construction. In this process, a small number of negative tokens may be mistakenly selected as high-confident tokens, potentially leading to semantic collapse of the proxy anchor. To assess the robustness of our high-confident token filtering, we perform a binary classification using GT labels. As shown in Tab. 5, the lowest precision is 93.9% on ADE20K [52], while all other datasets achieve even higher precision. These results indicate that semantic collapse rarely occurs, with at most 6% of negative tokens being incorrectly incorporated into the proxy anchor.

Metric	V21	PC59	C-Stf	City	ADE	Avg.
Precision (%)	98.7	96.3	94.2	95.6	**93.9**	96.2

Table 5. High confidence token classification precision.

E. Analysis for Attention Masking and Scaling

As discussed in Sec. 3.4, the Key–Value Extension can lead to the neglect of small objects due to the additional negative tokens from outer windows. Quantitatively, the total number of tokens increases by a factor of the number of windows L, substantially altering the statistics of the similarity map. For small objects, most of the newly introduced tokens correspond to background (negative) regions. This circumstance is clearly shown in Fig. 9(a-b), where background tokens (red) remain largely unmasked in Fig. 9(b) compared with Fig. 9(a), illustrating that low val-



(a) ProxyCLIP ($\beta = 1.2, \gamma = 3.0$) (b) Ours ($\boldsymbol{u} = 1.2, \boldsymbol{w} = 3.0$) (c) Ours ($\boldsymbol{u} = 1.4, \boldsymbol{w} = 3.0$) (d) Ours ($\boldsymbol{u} = 1.8, \boldsymbol{w} = 3.0$)

Figure 9. Effect of Key-Value Extension and $\mathbf{u}$.

ues of $\beta = 1.2$ and $\gamma = 3.0$ fail to suppress negative tokens from the outer windows.

While increasing fixed $\mathbf{u}$ and $\mathbf{w}$ (which correspond to β and γ in ProxyCLIP [24]) can quantitatively suppress this additional negative tokens, setting them too high can over-mask the attention. This over-masking removes parts of the object itself, restricting attention to local sub-regions and ultimately degrading performance (see Fig. 9c-d). Conversely, moderately stronger settings such as ($\mathbf{u} = 1.4, \mathbf{w} = 5.0$) achieve a balance: they suppress the negative tokens introduced by Key–Value Extension while preserving full object coverage, as evidenced qualitatively in Fig. 9.

These results directly support the normalization tightening strategy discussed in Sec. 3.4 of the main paper. Specifically, the "best" results reported in Tab. 1 and Tab. 2 are obtained by tightening the normalization parameters in this manner, compensating for the substantial increase of negative tokens introduced by Key–Value Extension. The observations from this results highlight a fundamental limitation of fixed masking and scaling: no single set of static hyperparameters can robustly handle both small and large objects across varying window configurations. This motivates the introduction of Dynamic Normalization, which adaptively adjusts masking and scaling to maintain consistent attention behavior regardless of object scale, window count, or dataset characteristics. Lastly, Dynamic Normalization removes the need for dataset-specific hyperparameters, providing a single formulation that generalizes well across datasets and window configurations.

F. High confidence tokens

High confidence token count is not a universally accurate proxy for object type; our dynamic normalization serves as an effective heuristic. Beyond the class-wise analysis in Section 5.3, we group instances based on ground-truth (GT) object size and analyze the corresponding number of high confidence tokens in Pascal Context60 [32] and Cityscapes [12] (in Fig. 10). Specifically, for each image, we compute the number of high confidence tokens associated with each GT instance, assign it to a size bin according to its GT region size, and report the interquartile range (25th–75th percentile) and median within each bin. This analysis shows that the number of high confidence tokens provides a reliable proxy for object size, demonstrating its effectiveness in capturing scale information.

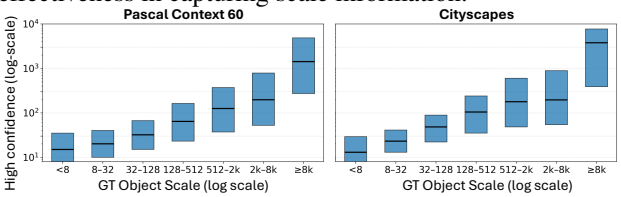


Figure 10. Object scale (GT vs high Confidence).

G. Adapting to SAM-based models

We conduct experiments to evaluate the adaptability of our method to SAM-based OVSS model [39]. Our method can be integrated into SAM-based models, which utilize components such as SAM masks and SAM refinement. While SAM-based models generally exhibit lower inference speed, they achieve superior performance. We aim to demonstrate that our method is compatible with SAM-based models and has the potential to achieve state-of-the-art performance in this setting. As shown in Tab. 6, our method attains 46.1% mIoU, representing a +0.3% improvement over the baseline, Trident, which is SAM-based model. This result confirms that our method can be effectively adapted to SAM-based models, and that our contribution is independent of existing approaches.

Model	V21	PC 60	C-Obj	V20	PC 59	C-Stf	City	ADE	Avg
Trident	67.1	38.6	41.1	84.5	42.2	**28.3**	42.9	**21.9**	45.8
Trident + GLA	**67.4**	**38.9**	**41.4**	**84.8**	**42.4**	**28.3**	**43.6**	21.7	**46.1**

Table 6. Trident adaptation with GLA.

H. Adapting to Multi-layer Feature Fusion

As we mentioned on Sec 5.5, we conduct experiments to evaluate the compatibility of our method with multi-layer feature fusion in the training-free OVSS task, which has been widely adopted recently. We also examine whether our model demonstrates performance improvements when combined with multi-layer feature fusion, which can validate that our method independently enhances performance and can be considered a standalone technique. In Tab. 7, we compare our method with and without multi-layer feature fusion. Following the approach of SC-CLIP [1], which leverages multi-layer feature fusion, we observe that our method combined with multi-layer feature fusion achieves an average mIoU of 44.9%, representing a +0.9% improvement over our baseline. This result demonstrates that our method independently improves baseline performance and can be effectively adapted to multi-layer feature fusion.

Model	V21	PC 60	C-Obj	V20	PC 59	C-Stf	City	ADE	Avg
Ours	66.3	36.1	37.7	**84.2**	39.9	26.9	40.8	20.0	44.0
Ours + Fusion	**67.9**	**37.0**	**38.1**	84.1	**40.9**	**27.5**	**43.2**	**20.3**	**44.9**

Table 7. Effect of multi-layer feature fusion.

I. Adapting to CASS for SOTA Performance

As mentioned in Sec. 5.5, we evaluate the compatibility of our method with CASS [22] to demonstrate that text prompt embedding modifications can be effectively integrated. We adapt our method to CASS by combining GLA-CLIP with two key components of CASS: (1) Object-Guided Text Embedding Adjustment, which refines text embeddings by fusing them with the mean of visual tokens, and (2) Object Perspective Patch-Text Similarity, which processes the entire image instead of individual windows and incorporates the full-image logits into the final prediction. Using the dataset-specific hyperparameters provided in the official CASS code, our approach achieves state-of-the-art performance, with an average mIoU of 44.7% in Tab. 8. One of our contribution is that ours can be combined with text embedding modification and logit fusion.

Model	V21	PC 60	C-Obj	V20	PC 59	C-Stf	City	ADE	Avg
CASS	65.8	36.7	**37.8**	87.8	40.2	26.7	39.4	20.4	44.4
ProxyCLIP + GLA	66.3	36.1	37.7	84.2	**40.8**	26.9	39.9	20.0	44.0
CASS + GLA	**67.2**	**36.8**	36.2	**88.5**	39.7	**27.2**	**40.6**	**21.0**	**44.7**

Table 8. CASS adaptation with GLA

J. Domain Shift: Remote Sensing Datset

J.1. Evaluation on remote sensing dataset

To demonstrate the generality of our GLA module beyond the eight web datasets in Tab. 1, we evaluate it on three remote sensing datasets following SegEarth-OV [25]. Since SegEarth-OV also adopts a sliding-window OVSS pipeline, our module can be directly integrated. We conduct experiments on Vaihingen[1], UAVid [31], and VDD [5]. As shown in Tab. 9, GLA improves the average mIoU from 38.8% to 39.6% (+0.8%), demonstrating strong cross-domain generalization. We expect our module to benefit any sliding-window-based setting.

Model	Vaih.	UAVid	VDD	Avg
ClearCLIP	27.3	36.2	39.3	34.3
ProxyCLIP	30.6	41.4	44.3	38.8
ProxyCLIP + GLA	**31.9** (+1.3)	**41.9** (+0.5)	**45.0** (+0.7)	**39.6** (+0.8)

Table 9. Performance comparison across remote sensing datasets.

J.2. Comparison with training-based model

Training-based OVSS models are vulnerable to domain shift due to reliance on fine-tuning. In domain-shift experiments (Tab. 10), ours and CLIP-DINOiser [45] (which is the representative Training-based model) show small gap on 8 web-image datasets (e.g. pascal voc [17], ade20k [52], cityscapes [12], etc), it becomes larger on 3 remote sensing datasets, indicating limited robustness and motivating training-free OVSS that preserves zero-shot capability of CLIP.

[1]https://www.isprs.org/education/benchmarks/UrbanSemLab

Model	Web (8)	Vaih.	UAVid	VDD	Avg (3)
Ours	**41.9**	31.9	41.9	45.0	**39.6**
CLIP-DINOiser	**40.3**	18.6	33.3	37.7	**29.9**

Table 10. Domain-shift evaluation on remote sensing datasets.

K. Computation cost.

We analyze the computational cost of our model to assess its practical adaptability. Due to the Key–Value Extension, our method must forward batched images to reference tokens from outer-window; forwarding each window independently is not feasible. In addition, the Key–Value Extension increases the number of tokens involved in the attention computation. While ProxyCLIP [24] requires a complexity of $O(LN^2D)$, our approach incurs $O(L^2N^2D)$ complexity on attention operation. Despite this theoretical increase, the actual latency and memory overhead remain modest. We measure the number of sliding windows, latency, and memory usage on the Pascal Context59 [32] following the SCLIP [42] setting (Tab 11). We conduct the experiments on a NVIDIA RTX A6000 GPU. By varying the sliding-window stride, increasing the number of windows leads to higher memory usage in ours. Though even with more windows, ours remains more efficient than CASS [22], using less memory while achieving competitive mIoU.

Context59 (336px×497px)	#Win	Lat.	Mem.	mIoU
CASS	1	5799 ms	7.6 GB	40.2
ProxyCLIP (Baseline)	8	429 ms	1.8 GB	38.8
Ours (Stride=224)	6	415 ms	1.7 GB	39.5
Ours (Stride=112)	8	431 ms	1.9 GB	39.9
Ours (Stride=98)	12	441 ms	2.4 GB	39.9

Table 11. Latency & memory usage comparison on PC59

L. Setting ablation.

We tune model hyperparameters for each setting while applying the same configuration consistently to both the baseline and our method. As shown in Tab. 12, our method consistently outperforms the ProxyCLIP [24] on Pascal VOC21 [17] across various configurations. The default setting is defined as {crop size: 224, stride: 112, image size: 336, background threshold: 0.1, rename trick: enabled} (following SCLIP [42] setting). We vary each parameter individually to demonstrate the robustness of our method.

VOC21	Default	Crop size 224 → 280	Stride 112 → 224	Img resize 336 → 448	Bg. th. 0.1 → 0.2	Rename O→ X
ProxyCLIP	63.3	64.4	59.7	61.8	61.0	60.4
Ours	**66.3**	**66.5**	**63.6**	**65.3**	**64.6**	**63.3**

Table 12. Setting ablation on VOC21 (mIoU).

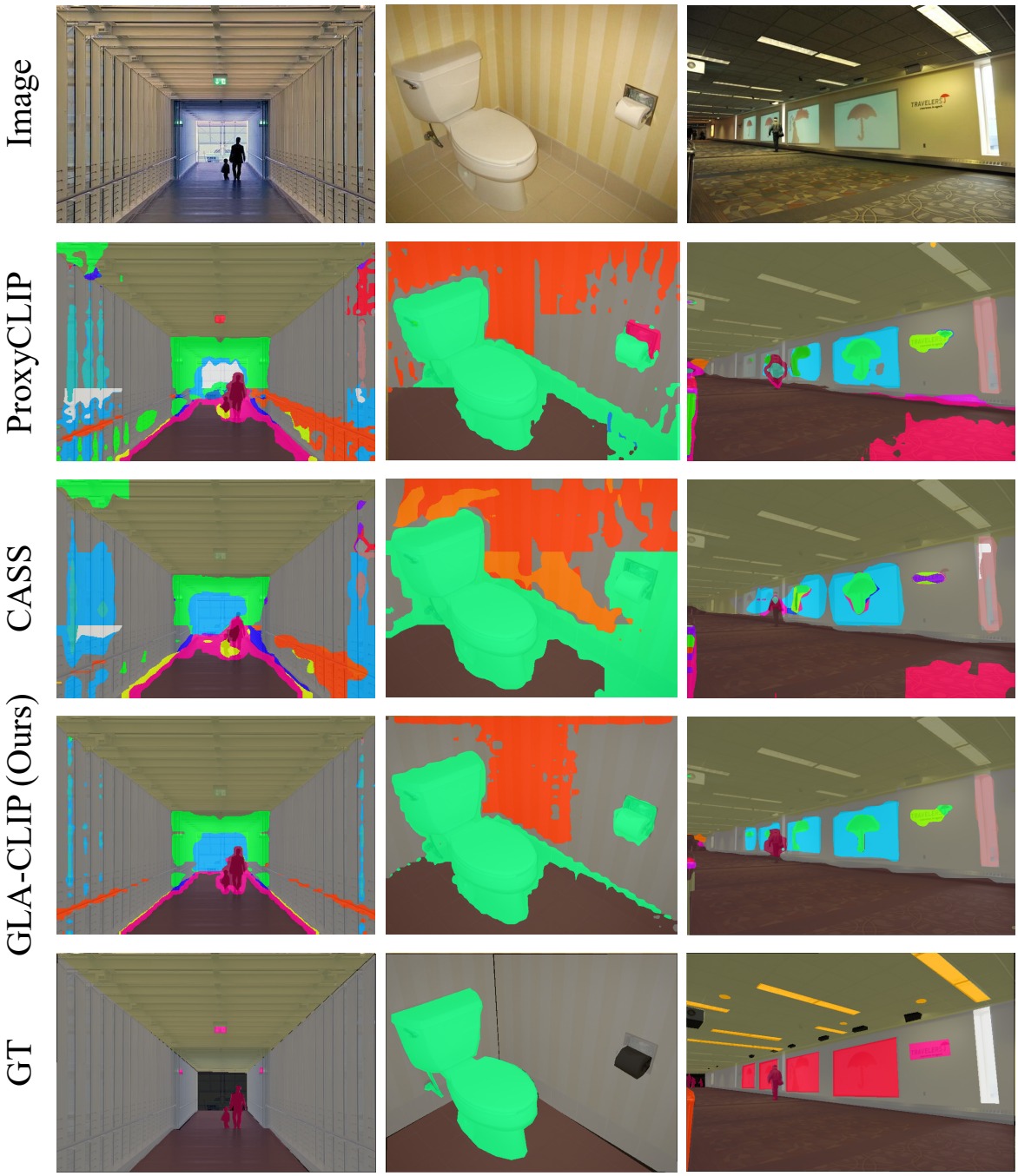


Figure 11. Qualitative results on ADE20K [52] with 150 categories. We compare GLA-CLIP with ProxyCLIP [24], CASS [22].

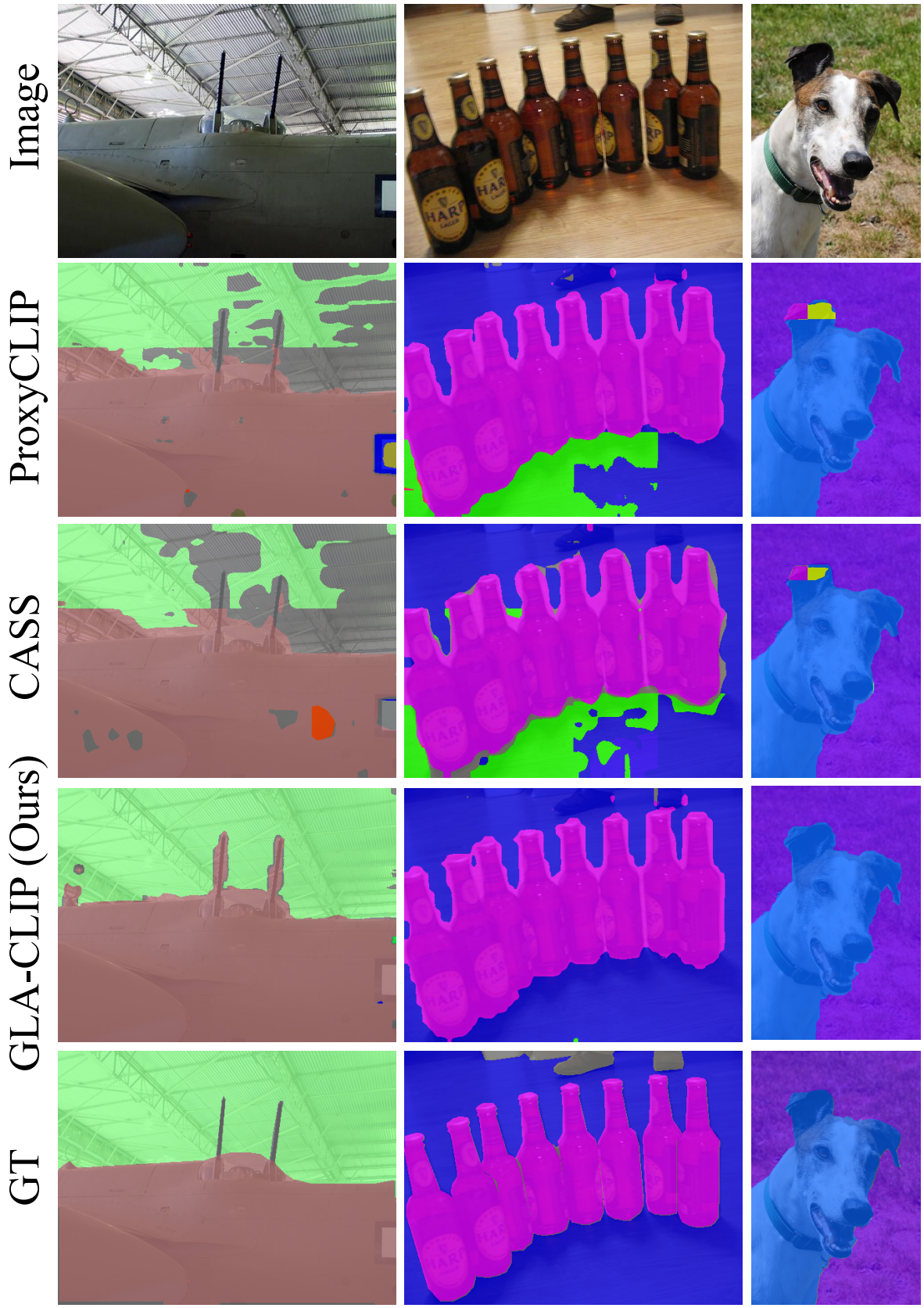


Figure 12. Qualitative results on Pascal Context60 [32] with 60 categories.. We compare GLA-CLIP with ProxyCLIP [24], CASS [22].

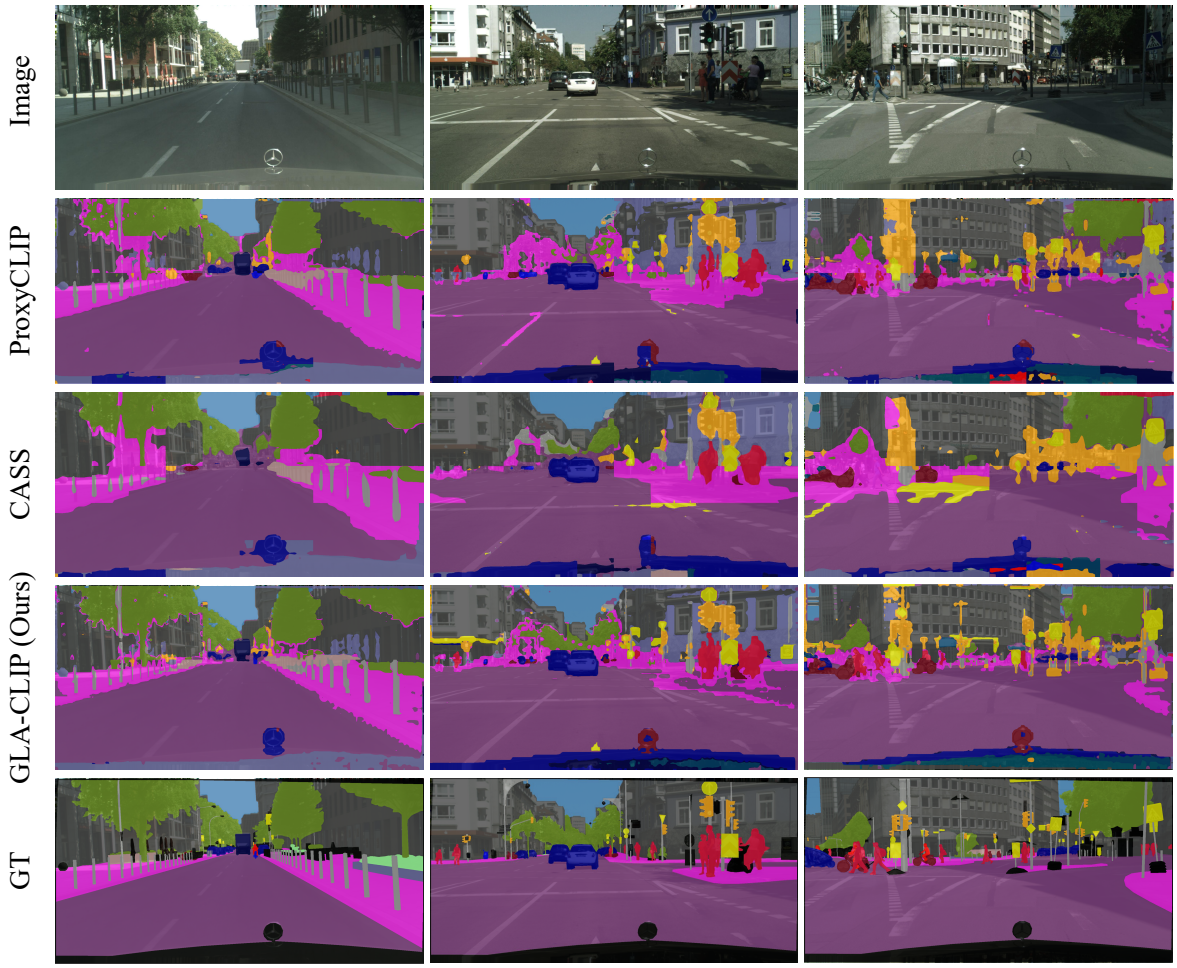


Figure 13. Qualitative results on Cityscapes [12] with 19 categories. We compare GLA-CLIP with ProxyCLIP [24], CASS [22].