Distilling Spectral Graph for Object-Context Aware Open-Vocabulary Semantic Segmentation (Supplementary Material)

Chanyoung Kim[1] Dayun Ju[1] Woojung Han[1] Ming-Hsuan Yang[1,2] Seong Jae Hwang[1]

[1]Yonsei University [2]University of California, Merced

{chanyoung, juda0707, dnwjddl, seongjae}@yonsei.ac.kr, mhyang@ucmerced.edu

Contents

A. Additional Material: Project Page & Presentation Video

We have described our results in an easily accessible manner on our project page, where a brief **presentation video** is also available. The link to the **project page** is as follows: `https://micv-yonsei.github.io/cass/`.

B. Detailed Method

B.1. Energy-based Low-rank Approximation

Algorithm 1 Optimal Rank k Selection with Low-Rank Eigendecomposition

```
Input: Adjacency matrix $A \in \mathbb{R}^{n \times n}$, energy threshold $\eta$, initial rank $q_0$, step size $\Delta q$
Output: Optimal rank $k$, eigenvectors $U$, eigenvalues $\Sigma$
Set $q_{\max} \leftarrow n$ ;            // Maximum allowable rank for symmetric matrices
Set $q \leftarrow q_0$
if $\|A - A^\top\|_F > \epsilon$ then
    Error: Input matrix is not symmetric
while $q \leq q_{max}$ do
    Approximate $A$ with rank-$q$ components:
        $A \approx U_q \Sigma_q U_q^\top,$
     where $U_q$ contains the top $q$ eigenvectors and $\Sigma_q = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_q)$
    Compute total energy: $E_{\text{total}} \leftarrow \sum_{i=1}^{q} \lambda_i$
    Compute cumulative energy: $E_{\text{cumulative}}(k) \leftarrow \sum_{i=1}^{k} \lambda_i$
    if $\frac{E_{cumulative}(k)}{E_{total}} \geq \eta$ for some $k \leq q$ then
        return $k, U_k, \Sigma_k$
    Increment $q \leftarrow q + \Delta q$
Set $q \leftarrow q_{\max}$ ;            // Fallback to maximum rank if threshold not met
return $q, U_q, \Sigma_q$
```

We leverage the Energy-based Low-rank Approximation method, as outlined in Algorithm 1, which efficiently captures key object-level contextual features within the VFM graph. The method approximates the i-th head attention adjacency graph A^i_{VFM} using the rank-q approximation as

$$A^i_{\text{VFM}} \approx U_q \Sigma_q U_q^\top, \tag{1}$$

where U_q contains the top q eigenvectors, and $\Sigma_q = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_q)$ consists of the top q eigenvalues. This selective focus on the most significant components highlights critical object relationships while discarding noise. By iteratively increasing the rank q, the algorithm efficiently captures the graph's essential structure with minimal complexity by examining cumulative energy. Our algorithm evaluates whether the retained energy satisfies the threshold η as

$$\frac{E_{\text{cumulative}}}{E_{\text{total}}} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{q} \lambda_i} \geq \eta, \quad k \leq q. \tag{2}$$

By adaptively identifying the smallest rank k that meets this criterion, the method ensures a compact and meaningful representation. Thus, our low-rank approximated VFM graph is expressed as

$$\tilde{A}^i_{\text{VFM}} = U_k \Sigma_k U_k^\top. \tag{3}$$

Our proposed combination of selective eigendecomposition, energy-based evaluation, and adaptive rank selection enables the algorithm to efficiently preserve object-level contextual features while minimizing computational costs. In the next section, we refine the computed low-rank components (i.e., the eigenvalues and eigenbasis), through dynamic eigenscaling. This approach further enhances their transformation into more object-centric representations, enabling improved precision and robustness in feature modeling.

B.2. Dynamic Eigenscaling

We propose a dynamic eigenscaling function ϕ to refine the eigenvalues Σ_k from Eq. (3). This function amplifies larger eigenvalues while suppressing smaller ones, highlighting dominant components in the graph structure and reducing the influence of noise. The scaled eigenvalues are computed as

$$\Sigma'_k = \phi(\Sigma_k) = \frac{\Sigma_k - \Sigma_{\min}}{\Sigma_{\max} - \Sigma_{\min}} \cdot \text{scaled\_range} + (2 - \epsilon) \cdot \Sigma_{\min}, \tag{4}$$

where $\Sigma_{\text{min}} = \min(\Sigma_k)$, $\Sigma_{\text{max}} = \max(\Sigma_k)$, and scaled_range $= \epsilon \cdot \Sigma_{\text{max}} - (2 - \epsilon) \cdot \Sigma_{\text{min}}$. We fix ϵ into 1.5. After applying our proposed dynamic eigenscaling function, our final tailored VFM graph is as

$$\ddot{A}^i_{\text{VFM}} = U_k \Sigma'_k U_k^\top := U_k \phi(\Sigma_k) U_k^\top. \tag{5}$$

This transformation dynamically adjusts the eigenvalues to retain the graph's most significant features, enhancing object-level contextual representation while minimizing the impact of less relevant components.

B.3. Hierarchical Grouping of Text Embeddings

Algorithm 2 Pseudo-code of hierarchical clustering in Python style.

```
from sklearn.metrics.pairwise import cosine_distances
from scipy.cluster.hierarchy import linkage, fcluster
from scipy.spatial.distance import squareform

def hierarchical_clustering(text_embeddings, h_threshold):

   # Compute cosine distance matrix
   distance_matrix = cosine_distances(text_embeddings)

   # Convert distance matrix to condensed form
   condensed_matrix = squareform(distance_matrix)

   # Perform hierarchical clustering using Ward's method
   Z = linkage(condensed_matrix, method='ward')

   # Form flat clusters based on the threshold
   clusters = fcluster(Z, t=h_threshold, criterion='distance') - 1

   return clusters
```

We provide details on grouping class prompts to identify semantically related object class prompts (e.g., `motorcycle` and `bicycle`) within the Object-Guided Text Embedding Adjustment module. As described in Algorithm 2, we compute the semantic distance between user-defined text prompts by measuring the cosine distance of their CLIP text embeddings in the CLIP latent space, forming a distance matrix. Next, the distance matrix is converted into a condensed form to facilitate hierarchical clustering using Ward's method, which minimizes variance within clusters. Finally, clusters are formed by applying a predefined height threshold to the hierarchical tree, ensuring that semantically similar class prompts are grouped together. This enables the identification of class prompt-level relationships that are leveraged to refine the text embeddings towards object-specific semantics.

C. Additional Experiments

C.1. Additional Evaluation Results

In this section, we provide additional evaluation results including qualitative evaluation (C.1.1), real-world open-vocabulary semantic segmentation result (E.2), and scale-up version of *CASS* (C.1.2).

C.1.1. Additional Qualitative Evaluation

We present additional qualitative evaluation results: Fig. 3 illustrates results on PASCAL VOC (V20 & V21) [5], Fig. 4 on COCO (C-Stf & C-Obj) [2], Fig. 5 on PASCAL Context (PC59 & PC60) [11], and Fig. 6 on both ADE20K (ADE) [18] and Cityscapes (City) [4]. Note that all the results in this subsection are presented without mask refinement steps, such as PAMR [1] or DenseCRF [9], to ensure a fair comparison and to evaluate the capability of the model itself.

C.1.2. Scale-up Version

The zero-shot object classification performance of CLIP [14] improves as the capacity of CLIP increases. Unlike the main experiment, where all methods were standardized using CLIP ViT-B/16 for a fair comparison, here we compute the object presence prior using CLIP ViT-L/14 (i.e., the scaled-up version), as reported in Table 1. Since CaR [15] also classifies the proposed mask using CLIP's global image embedding for classification, we compared our method under the same conditions as CaR. The encoder used for patch-wise dense representation (i.e., F_{CLIP} in our main manuscript) is referred to as the Feature Encoder, while the encoder for encoding CLIP visual embedding vector (i.e., v_{CLIP}) is referred to as the `[CLS]` Encoder. The CLIP visual embedding vector serves as an object presence prior in our approach and as a mask proposal classifier in

Table 1. Performance of scale-up version of *CASS*.

Model		Feature Encoder	[CLS] Encoder	V21	PC60	C-Obj	V20	PC59	ADE	Avg
CaR [15]	CVPR'24	ViT-B/16	ViT-B/16	48.6	13.6	15.4	73.7	18.4	5.4	29.2
CASS		ViT-B/16	ViT-B/16	65.8	36.7	37.8	87.8	40.2	20.4	48.1
CaR [15]	CVPR'24	ViT-B/16	ViT-L/14	63.7	29.4	35.1	91.4	38.4	16.9	45.8
CASS		ViT-B/16	ViT-L/14	66.3	37.0	38.3	89.3	40.7	20.7	48.7

CaR. For this experiment, we use datasets that are reported in CaR and evaluate with the mIoU metric. Note that we exclude a mask refinement step (e.g., PAMR [1] or DenseCRF [9]) for a fair comparison.

When using CLIP ViT-L/14 as the [CLS] encoder in our *CASS*, we observe a noticeable performance improvement compared to the ViT-B/16 configuration. Furthermore, our approach consistently outperforms CaR across all benchmarks under both encoder settings. Notably, when comparing models fairly by using the same [CLS] encoder configuration as CaR (ViT-B/16), CaR experiences a significant drop in performance, particularly on datasets such as PC60 and ADE. In contrast, our method demonstrates robust performance in both settings, maintaining high accuracy and adaptability regardless of the encoder used. This highlights the stability and effectiveness of our method, especially in handling variations in encoder configurations, demonstrating its robustness compared to CaR.

C.2. Additional Ablation Study

For ablation studies, we use representative datasets for segmentation tasks: PASCAL VOC (V21), PASCAL Context (PC59), and COCO-Stuff (C-Stf), following [17].

C.2.1. Ablation: VFM Backbones

Table 2. Ablation results with different VFM backbones. For the CLIP visual encoder, ViT-B/16 is used.

	Model		V21		PC59		C-Stf		Avg. mIoU	Avg. pAcc
			mIoU	pAcc	mIoU	pAcc	mIoU	pAcc		
DINOv1 ViT-B/8	LaVG [7]	ECCV'24	62.1	89.3	34.7	58.9	23.2	39.1	40.0	62.4
	ProxyCLIP [10]	ECCV'24	59.1	86.6	38.8	63.4	26.2	43.4	41.4	64.5
	CASS		65.8	90.1	40.2	65.0	26.7	43.6	44.2	66.2
DINOv1 ViT-B/16	LaVG [7]	ECCV'24	61.5	88.9	34.6	58.9	22.8	38.8	39.6	62.2
	ProxyCLIP [10]	ECCV'24	56.6	85.8	37.4	62.0	25.1	42.2	39.7	63.3
	CASS		64.3	89.1	38.9	63.8	26.1	43.1	43.1	65.3
DINOv2 ViT-B/14	LaVG [7]	ECCV'24	25.3	75.3	25.1	49.3	17.3	34.4	22.6	53.0
	ProxyCLIP [10]	ECCV'24	57.1	85.2	37.3	61.4	25.3	42.3	39.9	63.0
	CASS		63.0	88.6	38.1	62.2	25.3	42.0	42.1	64.3

Table 2 presents the results with various VFM backbones [3, 12], including DINOv1 ViT-B/8, DINOv1 ViT-B/16, and DINOv2 ViT-B/14. Among these, DINOv1 ViT-B/8 serves as the backbone for our main results. In this experiment, we use ViT-B/16 as the CLIP visual encoder. To ensure a consistent and fair comparison, we include LaVG [7] and ProxyCLIP [10] as baselines, as both utilize DINO as their backbone model. Since LaVG does not report performance on DINOv1 ViT-B/16 and DINOv2 ViT-B/14, we report reproduced results. Notably, LaVG employs spectral techniques, particularly graph partitioning, making it directly comparable to our approach, which also builds on spectral methods. The results show that our method consistently outperforms the baselines across all backbones, demonstrating its robustness and adaptability to different VFMs. In particular, compared to LaVG, which uses graph partitioning for spectral techniques, our method more effectively leverages spectral methods by distilling low-rank components into CLIP via spectral-based graph matching, offering a more advanced solution for training-free OVSS.

C.2.2. Ablation: Feature Type

Table 3. Comparison of features used for constructing attention graphs.

VFM	CLIP	V21		PC59		C-Stf		Avg. mIoU	Avg. pAcc
		mIoU	pAcc	mIoU	pAcc	mIoU	pAcc		
FF	QQ	51.1	81.8	33.0	57.2	21.8	39.1	35.3	59.4
FF	QK	44.6	77.3	30.2	54.0	19.9	37.1	31.6	56.1
FF	KK	60.8	87.6	37.9	62.1	25.2	42.0	41.3	63.9
QQ	QQ	57.1	84.9	36.5	61.2	23.9	41.5	39.2	62.5
QQ	QK	50.9	53.1	33.7	43.6	21.9	30.2	35.5	42.3
QQ	KK	64.9	89.3	39.7	64.5	26.4	43.5	43.7	65.8
QK	QQ	33.7	58.1	25.2	45.7	17.3	32.4	25.4	45.4
QK	QK	30.7	53.1	23.2	43.6	15.5	30.2	23.1	42.3
QK	KK	29.0	49.1	22.1	41.7	15.0	28.6	22.0	39.8
KK	QQ	64.9	89.3	40.0	64.6	26.6	43.7	44.0	65.8
KK	QK	60.3	87.6	38.5	63.4	25.4	43.0	41.4	64.7
KK	KK	65.8	90.1	40.2	65.0	26.7	43.6	44.2	66.2

Table 3 presents the performance impact of various features employed in constructing A_ψ during VFM graph distillation. The VFM column represents the features utilized in the construction of A_{VFM}, while the CLIP column represents the features used for A_{CLIP}. In the table, F refers to visual features derived from VFM, and Q and K represent the attention query and key components from their respective models. Each row represents a specific combination of features from VFM and CLIP used to construct the attention graph (e.g., QK in the VFM column indicates that A_{VFM} is constructed as $A_{\text{VFM}} = QK$).

When using visual features F from VFM, the results show moderate performance overall; however, constructing A_{VFM} using attention components such as Q or K instead of F resulted in better performance. This indicates that attention components more effectively capture semantic relationships between patches, demonstrating the efficiency of using attention to extract object-level context, which aligns with the findings of prior study [8]. In terms of attention components, QQ performs better than QK across most datasets but is outperformed by KK. The QK configuration generally shows weaker performance compared to both QQ and KK, highlighting the importance of modifying the last layer of self-attention in CLIP instead of using vanilla attention settings. Overall, the best performance is achieved when KK is used for the attention graph consistently in both VFM and CLIP, as demonstrated by the highest average mIoU of 44.2 and average pAcc of 66.2. This suggests that the KK provides the most robust and object-centric representation for constructing attention graphs.

C.2.3. Ablation: Graph Matching Strategy

Table 4. Ablation results on different graph matching strategies.

Method	V21		PC59		C-Stf		Avg. mIoU	Avg. pAcc
	mIoU	pAcc	mIoU	pAcc	mIoU	pAcc		
Sequential	64.9	89.6	39.6	64.5	26.2	43.1	43.5	65.7
Similar	65.5	90.0	40.2	65.1	26.6	43.6	44.1	66.2
Complementary	65.8	90.1	40.2	65.0	26.7	43.6	44.2	66.2

In Table 4, we compare different graph matching strategies. "Sequence" refers to matching graphs head-to-head in a sequential manner without applying additional structural alignment techniques. "Similar" involves matching graphs that exhibit similar structures between VFM and CLIP. Lastly, "Complementary" uses a strategy that matches graphs with contrasting but complementary structures between VFM and CLIP to leverage diverse information. From the results, we observe that the "Sequential" method delivers competitive performance, achieving an average mIoU of 43.5 and pAcc of 65.7. The rows for

"Similar" and "Complementary" consider the attention graph structures from each head, matching either similar or dissimilar ones, respectively. The "Similar" strategy improves the results, with an average mIoU of 44.1 and pAcc of 66.2. However, the proposed "Complementary" method outperforms both matching methods, achieving the highest average mIoU of 44.2 and pAcc of 66.2. Overall, while sequential approaches are straightforward and offer reasonable performance, these results highlight the advantages of considering in matching attention heads.

C.2.4. Ablation: Number of Eigenvalues for Low-Rank Components in VFM Graph

Table 5. Ablation results on rank selection methods for low-rank components.

Method	Adaptive	V21		PC59		C-Stf		Avg. mIoU	Avg. pAcc
		mIoU	pAcc	mIoU	pAcc	mIoU	pAcc		
$k = 3$	✗	62.7	89.3	39.2	64.0	25.9	42.6	42.6	65.3
$k = 4$	✗	64.5	89.8	39.8	64.6	26.4	43.1	43.6	65.8
Eigengap	✓	64.9	89.6	39.8	64.6	26.3	43.3	43.7	65.8
Energy-based	✓	65.8	90.1	40.2	65.0	26.7	43.6	44.2	66.2

In this experiment, we compare methods for selecting the rank (number of eigenvalues) for low-rank components. Table 5 presents the results of applying several well-known methods alongside our proposed energy-based approach. Rows marked with ✗ represent the use of a fixed rank k across the entire images in datasets, whereas rows marked with ✓ indicate searching for the optimal rank on a per-image basis. The eigengap is computed as the index i^* where the difference between consecutive eigenvalues in the spectrum is maximized, typically defined as $i^* = \underset{i\in\{1,2,\ldots,n-1\}}{\arg\max}(\lambda_i - \lambda_{i+1})$, where λ_i and λ_{i+1} represent the i-th and $(i+1)$-th largest eigenvalues, respectively. This metric is commonly used to identify the point where the eigenvalue spectrum exhibits a significant drop, which helps in determining the optimal rank for low-rank approximations or clustering applications.

Fixed-rank methods maintain consistent ranks across all images but fail to adapt to varying data complexity within individual images. As a result, their performance is lower compared to adaptive approaches. The eigengap method, which adaptively selects the rank by identifying a significant drop in the eigenvalue spectrum, improves performance compared to fixed-rank strategies. However, the eigengap approach is limited by its sensitivity to small variations in the eigenvalue distribution, which can lead to inconsistent rank selections in certain cases. Our energy-based method further enhances performance by selecting ranks based on the cumulative energy of the eigenvalues, ensuring a more stable and data-driven approach to rank determination. This method achieves the best overall results and consistently outperforms all other methods across individual datasets. These results highlight the importance of adaptive rank selection strategies in capturing the intrinsic object-level structure of graphs.

D. Implementation Details

D.1. Prompt Templates

Table 6. Examples of the prompt templates used for text embedding generation.

Examples of Prompt Templates
"a photo of my {`prompt`}."
"a photo of a nice {`prompt`}."
"a cropped photo of a {`prompt`}."
"a photo of a large {`prompt`}."
"art of the {`prompt`}."

Following recent works [6, 7, 17], we employ a comprehensive set of prompt templates to enhance the diversity of text embeddings (e.g., "a photo of {`prompt`}"). Specifically, given a prompt query t, it is systematically inserted into a predefined list of 80 diverse templates designed to capture varying linguistic contexts. Each generated sentence is processed through the CLIP text encoder, resulting in 80 distinct text embeddings. To make these embeddings into a unified representation, we compute their average, yielding a single, contextually rich text embedding t_{CLIP} that corresponds to the input query t. The examples of prompt templates are listed in Table 6.

D.2. Hyperparameters

Table 7. Hyperparameters used in *CASS*.

V21		PC60		C-Obj		V20		PC59		C-Stf		City		ADE	
α	γ	α	γ	α	γ	α	γ	α	γ	α	γ	α	γ	α	γ
0.03	0.10	0.04	0.25	0.01	0.25	0.02	0.40	0.04	0.25	0.02	0.20	0.03	0.10	0.05	0.30

Here, we provide a detailed description of the hyperparameters listed in Table 7. The parameter α controls the balance between the original text embedding and the object-specific vector, while γ adjusts the trade-off between the patch-text similarity $\hat{S}$ and the object presence prior.

E. Application

E.1. Image Inpainting and Object Removal

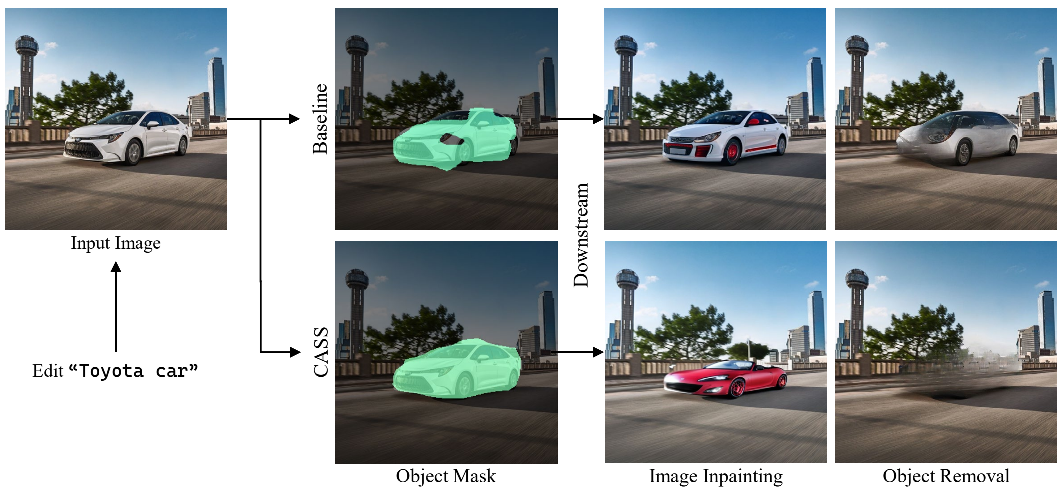

Figure 1. Visualization of image inpainting and object removal using our predicted mask. For image inpainting, we use "`red sports car with red wheels`" as an input prompt. Note that the mask refinement step is excluded when segmenting the object mask.

While our *CASS* can naturally be used for semantic segmentation directly based on user-provided prompts in real-world scenarios (see Sec. E.2 for diverse examples), its ability to be object-level context-aware allows it to group object components effectively. As a result, *CASS* produces *clean object masks* that can be effectively utilized in downstream tasks (e.g., image inpainting and object removal). Fig. 1 visualizes the application of our *CASS* in various downstream tasks. When provided with the object prompt `Toyota car`, our model generates a precise object mask, enabling tasks such as image inpainting and object removal. For image inpainting, we utilize Stable Diffusion XL (SDXL) [13], while for object removal, we employ LaMa [16]. The baseline model used for comparison is NACLIP [6]. Comparing the results to the baseline, it is evident that the baseline model struggles to correctly mask all the object components, such as the *rear wheels* and *headlights*. For example, the baseline fails to mask the rear wheel of the car, which limits the inpainting model from accurately rendering the red wheel in the edited image. Similarly, incomplete masking of headlights results in unedited regions (e.g., headlight) during the inpainting process. This limitation also affects object removal tasks. The baseline's failure to mask the entire object (i.e., `Toyota car`) limits object-level removal and instead focuses on specific internal components, such as side mirrors or door handles. In contrast, our method successfully captures all relevant object components, producing a complete and accurate object mask that enables the removal of the object as a whole. These results demonstrate the importance of generating object-level context-aware masks, as they enable downstream models to perform tasks such as image inpainting and object removal with significantly improved precision and quality.

E.2. Open-Vocabulary Semantic Segmentation in the Wild

Fig. 7 illustrates the application of *CASS* on real-world images, demonstrating its ability to adapt to diverse and uncontrolled settings. This showcases the method's robustness in handling the complexities and variations typically encountered in real-world scenarios, without requiring additional fine-tuning or preprocessing.

F. Discussion

F.1. Limitations

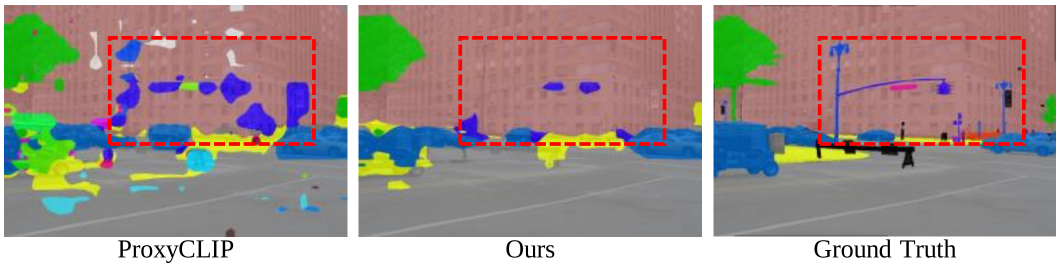

Figure 2. Visualization of limitation of *CASS*.

Our *CASS* introduces object-level contextual knowledge into training-free OVSS, enabling the effective grouping of object components into coherent semantic entities. By combining Energy-based Low-rank Approximation with spectral-based graph matching, our approach narrows the gap between pixel-level predictions and object-level understanding, contributing a significant advancement towards the primary objectives of OVSS.

However, *CASS* has limitations, particularly in computational efficiency. While Energy-based Low-rank Approximation reduces costs, the eigendecomposition step remains computationally demanding, especially for high-resolution images. Additionally, the Hungarian matching algorithm, which runs on the CPU, introduces further latency due to its lack of GPU support. These inefficiencies hinder real-time applicability. For instance, *CASS* processes at 5.6 FPS on an RTX A6000 GPU, which is insufficient for real-time segmentation. Optimizing computational efficiency is essential to expanding its practical use in time-sensitive tasks.

Moreover, as shown in Fig. 2, our emphasis on object-level context leads to stronger performance on large objects (e.g., building: 20.0 [10] to 34.7 mIoU) but can underperform on smaller objects, ultimately diminishing the overall mIoU improvement due to the averaging across all classes. As a result, *CASS* yields substantial improvements on datasets with predominantly large and distinct object classes (e.g., VOC), while showing relatively modest gains on datasets rich in small object categories (e.g., ADE, COCO). Nonetheless, *CASS* still secures a 7.25% relative improvement over the existing SoTA [10] on average across 8 datasets, underscoring its meaningful contribution to advancing training-free OVSS.

F.2. Future Works

Our proposed *CASS* is designed to perform semantic segmentation on individual images based on user-provided arbitrary prompts, achieving competitive performance in grouping object components into coherent entities. While effective for static scenarios, the current approach does not account for temporal consistency, limiting its applicability to video sequences and dynamic environments. To extend its use to real-time or sequential analysis tasks, future work will focus on incorporating temporal information to ensure coherence across frames. This enhancement will enable the method to handle video streams more effectively, supporting applications such as object tracking and dynamic scene understanding.

As mentioned in the limitations, the high computational time remains a challenge that must be addressed as part of our future work. Specifically, optimizing the eigendecomposition process and the Hungarian matching algorithm used for graph matching is crucial. These improvements will play a key role in enhancing the efficiency of our method and making real-time inference feasible for practical applications.

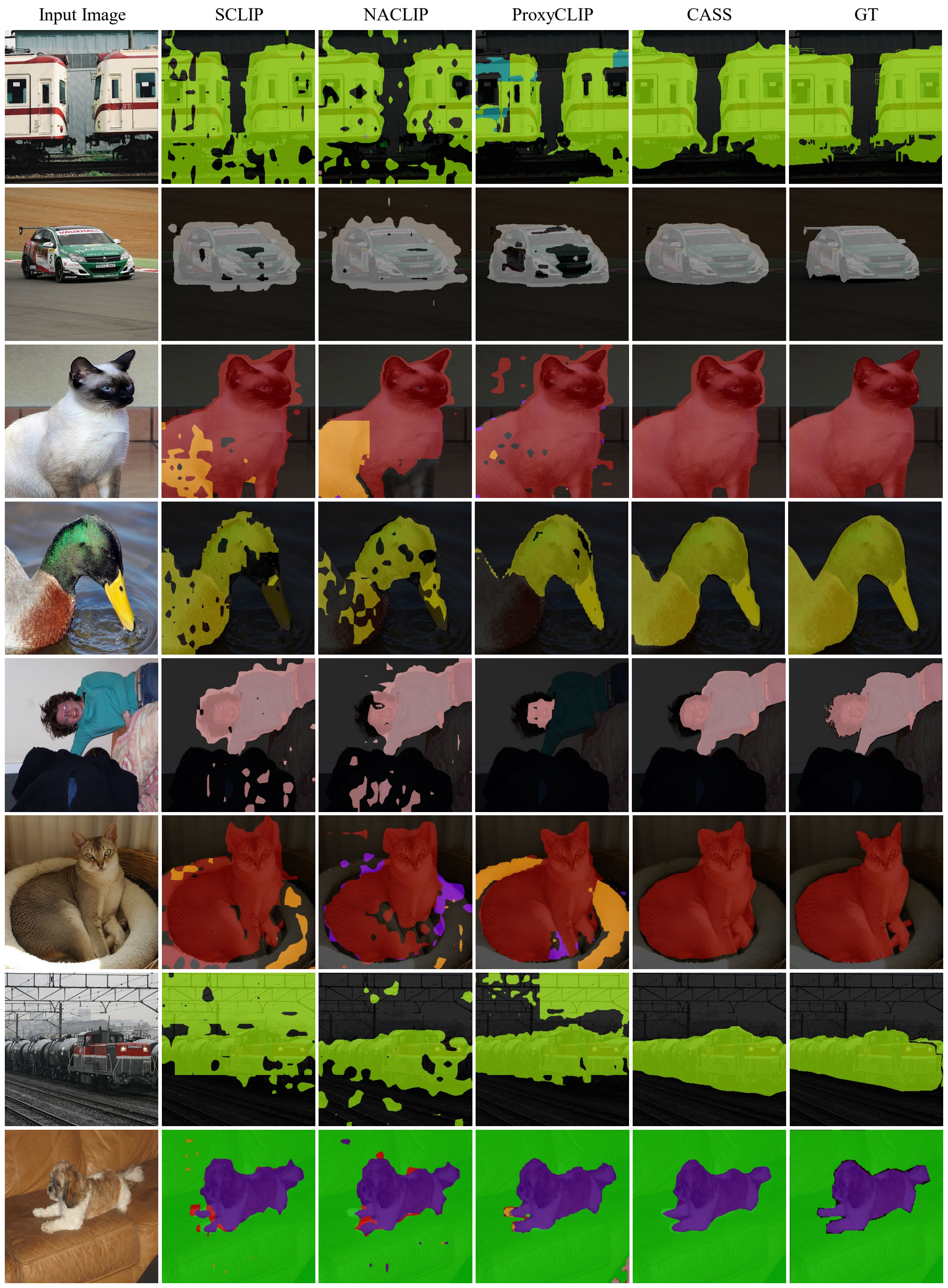

Figure 3. Additional qualitative comparison between recent state-of-the-art methods SCLIP [17], NACLIP [6], ProxyCLIP [10] and Our *CASS* using PASCAL VOC dataset [5].

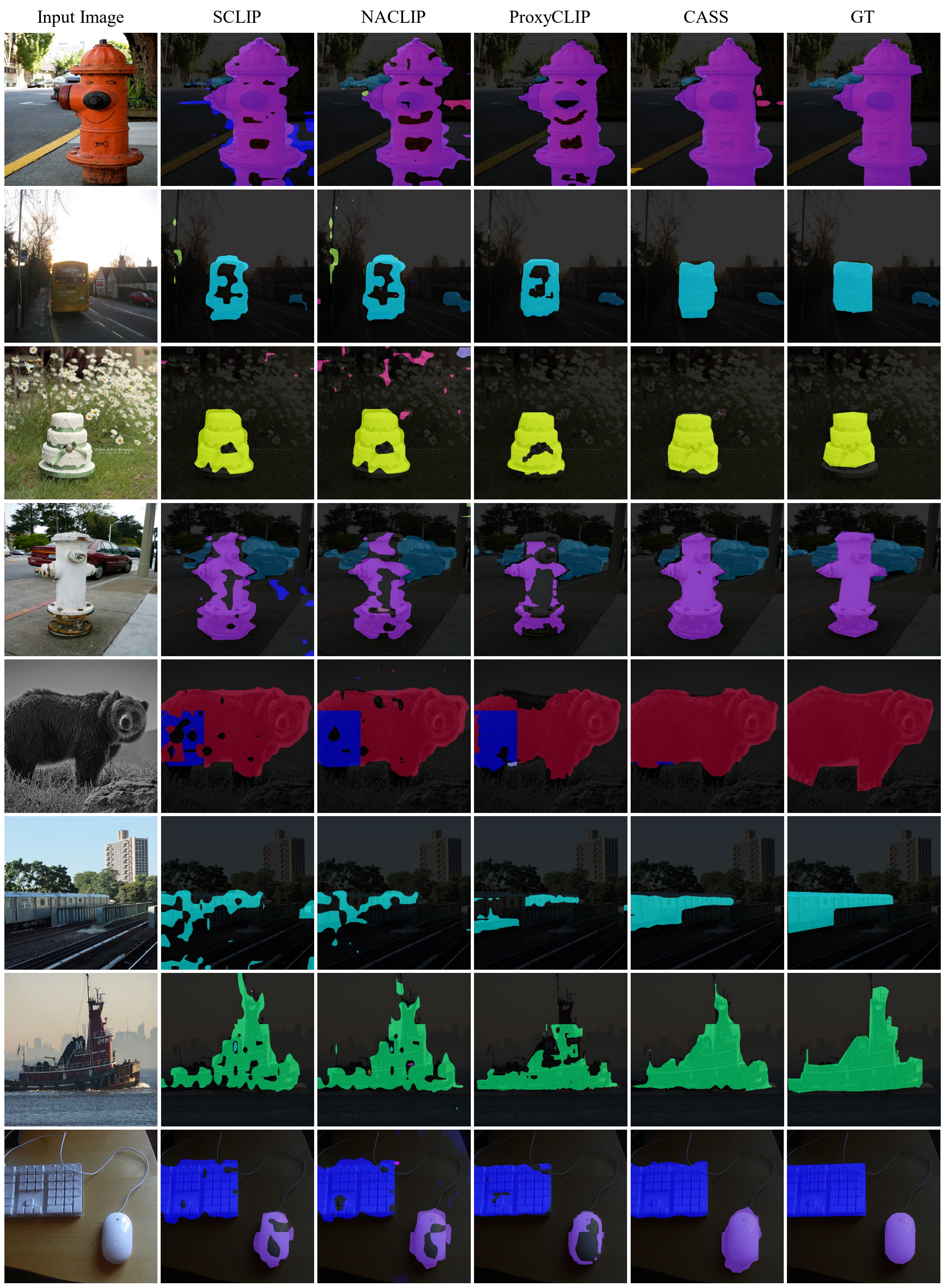

Figure 4. Additional qualitative comparison between recent state-of-the-art methods SCLIP [17], NACLIP [6], ProxyCLIP [10] and Our *CASS* using COCO dataset [2].

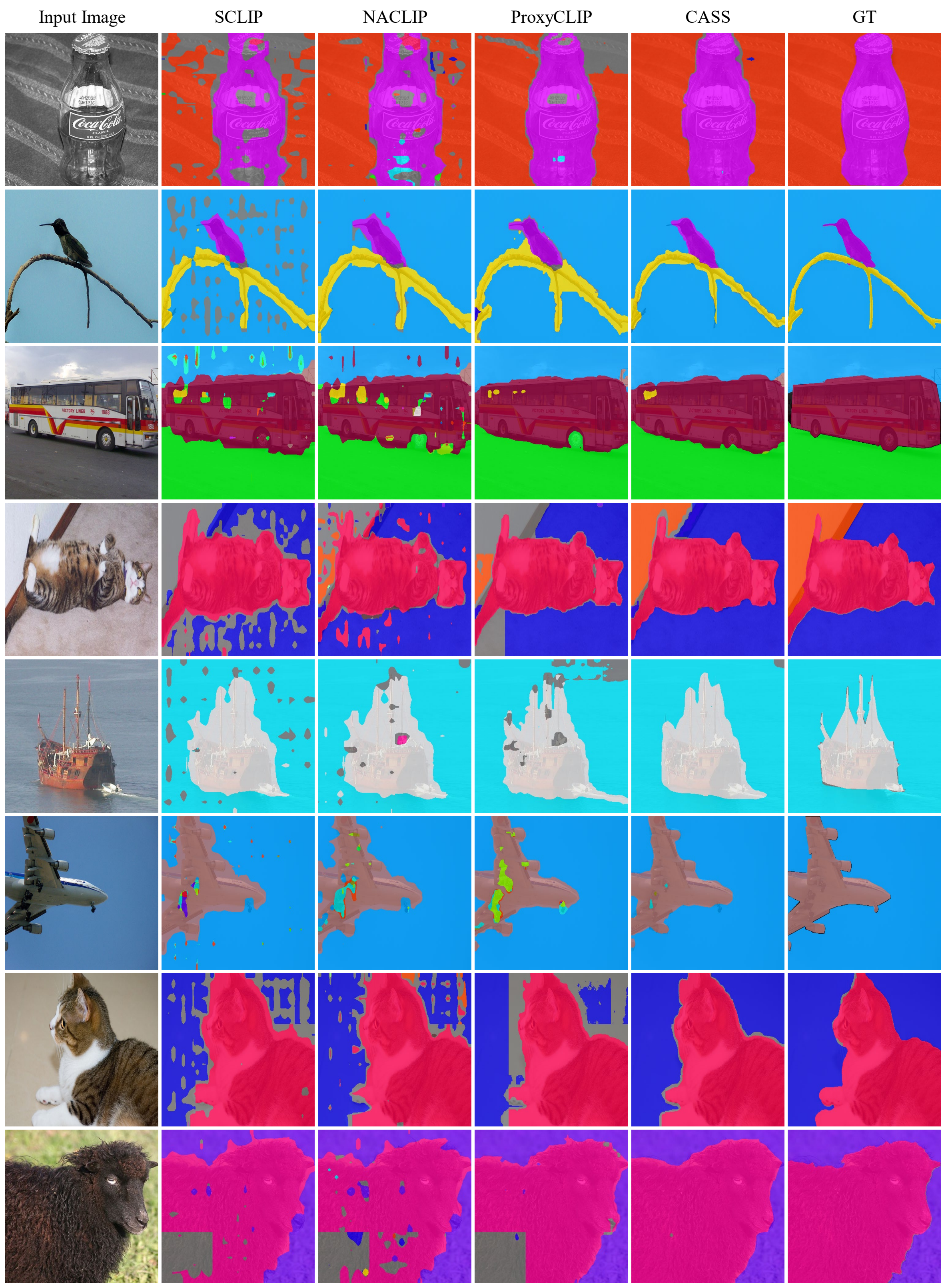

Figure 5. Additional quantitative comparison between recent state-of-the-art methods SCLIP [17], NACLIP [6], ProxyCLIP [10] and Our *CASS* using PASCAL Context [11].

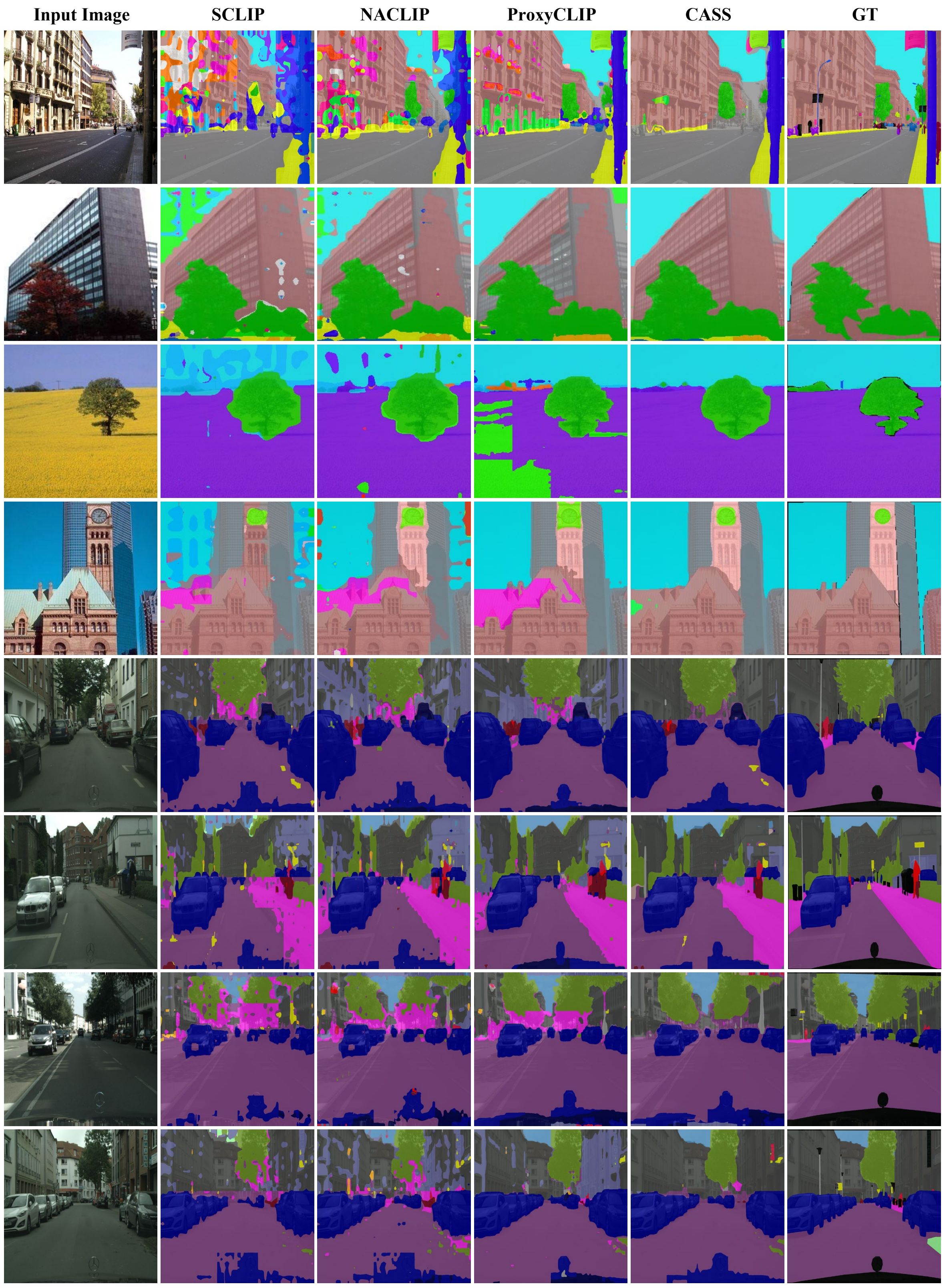

Figure 6. Additional qualitative comparison between recent state-of-the-art methods SCLIP [17], NACLIP [6], ProxyCLIP [10] and Our *CASS* using ADE20K [18] (top) and Citscapes [4] (bottom).

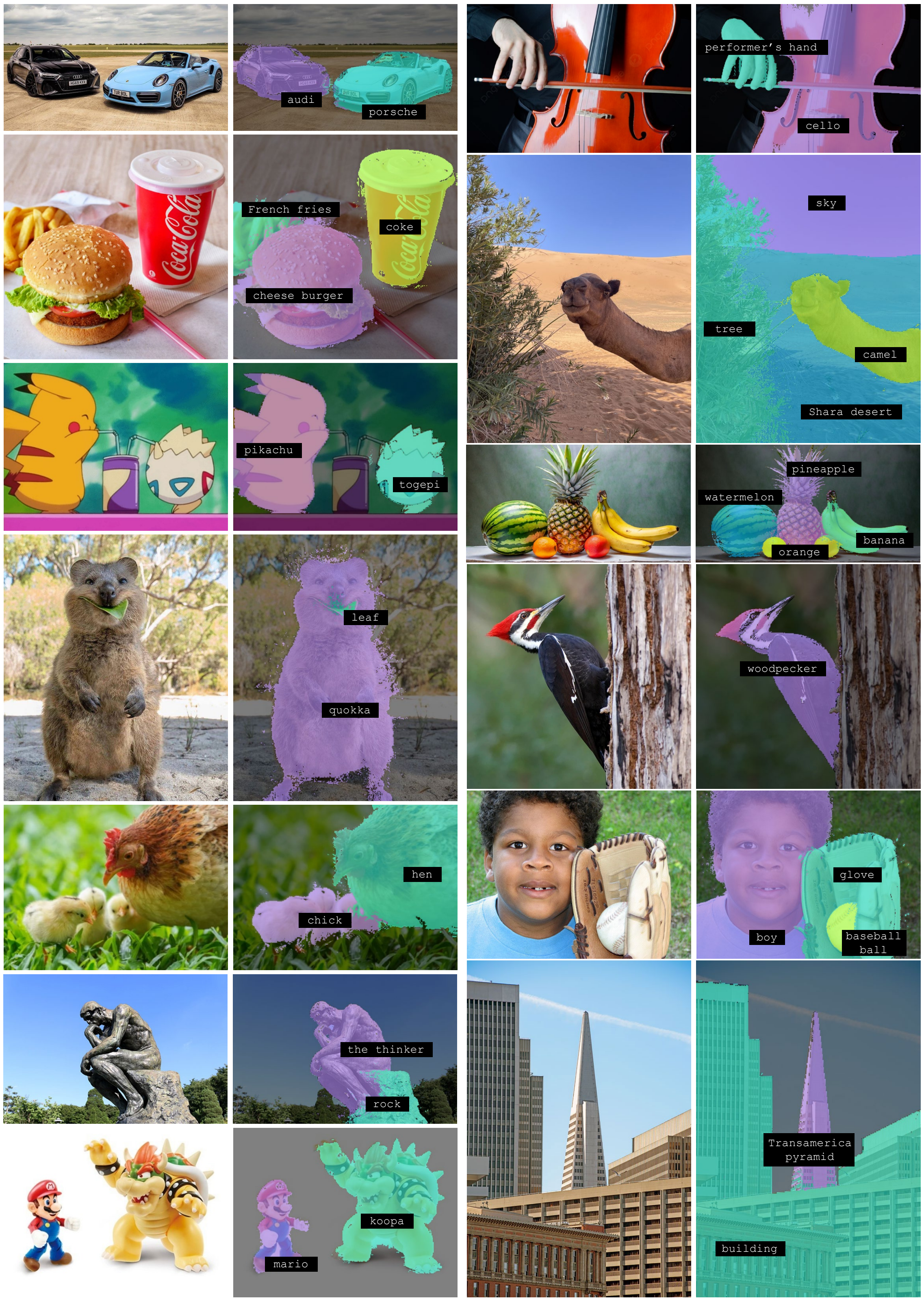

Figure 7. Real-world open-vocabulary semantic segmentation results of our model after applying mask refinement [1].

References

[1] Nikita Araslanov and Stefan Roth. Single-Stage Semantic Segmentation from Image Labels. In *CVPR*, 2020. 3, 4, 13

[2] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-Stuff: Thing and Stuff Classes in Context. In *CVPR*, 2018. 3, 10

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *ICCV*, 2021. 4

[4] 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. 3, 12

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, 2012. 3, 9

[6] Sina Hajimiri, Ismail Ben Ayed, and Jose Dolz. Pay attention to your neighbours: Training-free open-vocabulary semantic segmentation. In *WACV*, 2025. 6, 7, 9, 10, 11, 12

[7] Dahyun Kang and Minsu Cho. In defense of lazy visual grounding for open-vocabulary semantic segmentation. In *ECCV*, 2024. 4, 6

[8] Chanyoung Kim, Woojung Han, Dayun Ju, and Seong Jae Hwang. Eagle: Eigen aggregation learning for object-centric unsupervised semantic segmentation. In *CVPR*, 2024. 5

[9] Philipp Krähenbühl and Vladlen Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *NeurIPS*, 2011. 3, 4

[10] Mengcheng Lan, Chaofeng Chen, Yiping Ke, Xinjiang Wang, Litong Feng, and Wayne Zhang. Proxyclip: Proxy attention improves clip for open-vocabulary segmentation. In *ECCV*, 2024. 4, 8, 9, 10, 11, 12

[11] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The Role of Context for Object Detection and Semantic Segmentation in the Wild. In *CVPR*, 2014. 3, 11

[12] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 4

[13] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 7

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, 2021. 3

[15] Shuyang Sun, Runjia Li, Philip Torr, Xiuye Gu, and Siyang Li. Clip as rnn: Segment countless visual concepts without training endeavor. In *CVPR*, 2024. 3, 4

[16] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *WACV*, 2022. 7

[17] Feng Wang, Jieru Mei, and Alan Yuille. Sclip: Rethinking self-attention for dense vision-language inference. In *ECCV*, 2024. 4, 6, 9, 10, 11, 12

[18] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic Understanding of Scenes Through the ADE20K Dataset. *IJCV*, 2019. 3, 12