# PrAda: Few-Shot Visual Adaptation for Text-Prompted Segmentation

## Supplementary Material

## Appendix

**Table of contents:**



## A. Implementation Details

### A.1. PrAda

To train our method we employ AdamW [8] optimizer with a weight decay of 0.01 with cosine learning rate scheduling. We use a batch size of 8 for all datasets. We adapt our method to different datasets using a consistent approach while adjusting key hyperparameters based on dataset characteristics. Table 1 summarizes the learning rate, number of iterations, and initial $\alpha$ value used across all datasets. For standard benchmarks (ADE20K [16], Cityscapes [2], Mapillary Vistas [10]), we use a learning rate of 0.008 with 1000 iterations and $\alpha_{\text{init}} = 80$. The ShowOrTell [11] benchmark follows a similar configuration with the same learning rate and $\alpha$ value, but we reduce the number of iterations to 500 for most datasets to balance adaptation efficiency and computational cost. Only UECFOOD [3], PASCAL VOC 2012 [4], and ZeroWaste [1] require 1000 iterations due to their larger domain shift. For the SegInW [19] benchmark, which presents significantly different visual domains, we adopt a more conservative approach using a lower learning rate of 0.0002 with 50 iterations and $\alpha_{\text{init}} = 50$. Notably, House-Parts and Strawberry benefit from a higher learning rate (0.002) and more iterations (100), while Trash requires extended adaptation with 800 iterations to handle its challenging characteristics.

### A.2. Baselines

For all baselines, we follow the same training protocol described for our method in the previous section. For their implementation and choice of hyperparameters, we followed their original publications and officially released code.

**CoOp [18].** For CoOp, we follow the original implementation and we set the context length to 16 with no initialization from hand-crafted prompts or class-specific prompts.

**CoCoOp [17].** For CoCoOp, we initialize the context vector with "*This is a photo of a large*" template and we keep the rest of the hyperparameters as in the original implementation.

| Dataset | LR | Iters | $\alpha_{\text{init}}$ |
|---|---|---|---|
| *Standard Benchmarks* | | | |
| ADE20K [16], Cityscapes [2], Mapillary Vistas [10] | 0.008 | 1000 | 80 |
| *ShowOrTell Benchmark [11]* | | | |
| House-Parts, LoveDA-Rural [12], LoveDA-Urban [12], MHPv1 [7], PIDray [14], Pizza, Toolkits, Trash, UAVid [9], ZeroWaste [1] | 0.008 | 500 | 80 |
| UECFOOD [3], PASCAL VOC 2012 [4] | 0.008 | 1000 | 80 |
| *SegInW Benchmark [19]* | | | |
| Airplane-Parts, Bottles, Brain-Tumor, Chicken, Cows, Electric-Shaver, Elephants, Fruits, Garbage, Ginger-Garlic, Hand, Hand-Metal, HouseHold-Items, Nutterfly-Squireel, Phones, Poles, Puppies, Rail, Salmon-Fillet, Tablets, Toolkits, Watermelon | 0.0002 | 50 | 50 |
| House-Parts, Strawberry | 0.002 | 100 | 50 |
| Trash | 0.0002 | 800 | 50 |

Table 1. Hyperparameters used for adaptation across different datasets. We report the learning rate (LR), number of iterations (Iters), and initial $\alpha$ value.

**CLIP-Adapter [5].** For CLIP-Adapter, we implement the method as described in the original paper, using a 4 times reduction ratio for the adapter's bottleneck layer. We insert the adapter after the MLP that generates the class embeddings.

**TipAdapter [15].** For TipAdapter, we follow the original implementation and we build the cache memory using the masked pooled features from the visual examples. For the scoring function, we set the hyperparameter $\alpha$ to 10.0 and $\beta$ to 1.0 for all the datasets.

## B. Additional ablation studies

**Alpha values.** Table 2 presents an analysis on the parameter $\alpha$, comparing different initialization values under two training strategies: trainable (where $\alpha$ is optimized during adaptation) and fixed (where $\alpha$ remains constant). Our design choice of using a trainable $\alpha$ initialized at 80 achieves the best overall performance, reaching 32.2 PQ and 38.7 mIoU on ADE20K, 50.1 PQ and 67.7 mIoU on Cityscapes, and 33.1 mIoU on ShowOrTell. This configuration outperforms all fixed $\alpha$ variants, demonstrating that allowing $\alpha$ to adapt during training is crucial for balancing the contribution between text-based and visual prototype-based predictions. When $\alpha$ is fixed, performance degrades across all initialization values, with the best fixed configuration ($\alpha = 60$) achieving only 31.9 PQ on ADE20K and 33.0 mIoU on ShowOrTell compared to 32.2 PQ and 33.1 mIoU respectively with trainable $\alpha = 80$. Interestingly, the results show that our method performs robustly across a broad range of initialization values ($30 \leq \alpha \leq 100$), with trainable $\alpha$ consistently achieving strong performance: values

| $\alpha$ | Trainable | ADE20K | | | Cityscapes | | SoT |
|---|---|---|---|---|---|---|---|
| | | PQ | mIoU | AP | PQ | mIoU | mIoU |
| *Trainable* $\alpha$ | | | | | | | |
| 10 | ✓ | 27.0 | 35.5 | 18.4 | 48.6 | 65.9 | 31.0 |
| 30 | ✓ | 30.4 | 37.9 | 18.8 | 49.8 | 67.0 | 32.8 |
| 60 | ✓ | 31.8 | 39.1 | 18.4 | 49.3 | 67.0 | 33.1 |
| **80** | ✓ | 32.2 | 38.7 | 18.3 | 50.1 | 67.7 | 33.1 |
| 100 | ✓ | 31.7 | 38.2 | 17.8 | 49.5 | 67.1 | 33.1 |
| *Fixed* $\alpha$ | | | | | | | |
| 10 | ✗ | 26.9 | 35.2 | 18.3 | 48.5 | 65.7 | 30.6 |
| 30 | ✗ | 30.8 | 38.3 | 18.6 | 49.6 | 66.9 | 32.9 |
| 60 | ✗ | 31.9 | 38.8 | 18.0 | 49.1 | 66.0 | 33.0 |
| 80 | ✗ | 31.6 | 38.0 | 17.6 | 49.3 | 67.2 | 33.0 |
| 100 | ✗ | 31.2 | 37.5 | 17.1 | 48.8 | 65.9 | 32.7 |

Table 2. **Ablation study on different values of $\alpha$.** We compare different initialization values for $\alpha$ parameter with two training strategies: trainable (*i.e.* $\alpha$ is learned during training) and fixed (*i.e.* $\alpha$ remains constant).

of 30, 60, 80, and 100 yield 30.4, 31.8, 32.2, and 31.7 PQ on ADE20K, and 32.8, 33.1, 33.1, and 33.1 mIoU on ShowOrTell respectively. However, initializing with very low values (*e.g.*, $\alpha = 10$) yields suboptimal performance (27.0 PQ on ADE20K and 31.0 mIoU on ShowOrTell), as the model struggles to properly weight the visual prototypes. This demonstrates that while training $\alpha$ is essential for optimal performance, the method is relatively insensitive to the specific initialization choice within a reasonable range.

**Alpha after training**. Table 3 and Figure 1 show the evolution of the $\alpha$ parameter during training across different benchmarks and individual datasets. We report both the initial value $\alpha_{\text{init}}$ used to start the adaptation process and the final value $\alpha_{\text{final}}$ after training converges. The results reveal that $\alpha$ adapts differently across benchmarks based on their specific characteristics. For standard benchmarks (ADE20K, Cityscapes, Mapillary Vistas), $\alpha$ consistently decreases from its initial value of 80, converging to values around 63 (62.5, 63.6, and 63.8 respectively). This reduction of approximately 20% suggests that these well-aligned datasets benefit from a more balanced combination of text-based and visual prototype-based predictions, with the model learning to place slightly less emphasis on visual prototypes while still maintaining their contribution. In contrast, for the SegInW [19] benchmark, where we initialize $\alpha$ at 50 due to the more diverse visual domains, the final value remains stable at 50.0. This stability is primarily due to the limited number of training iterations and the relatively small number of classes in many SegInW datasets, which constraints the extent to which $\alpha$ can be effectively optimized during the short adaptation phase. The ShowOrTell [11] benchmark shows an intermediate behavior, with $\alpha$ decreasing from 80 to 70.4 on average, a more

| | ADE20K | Cityscapes | Mapillary | SegInW | ShowOrTell |
|---|---|---|---|---|---|
| $\alpha_{\text{init}}$ | 80 | 80 | 80 | 50 | 80 |
| $\alpha_{\text{final}}$ | 62.5±0.0 | 63.6±0.1 | 63.8±0.0 | 50.0±0.0 | 70.4±3.3 |

Table 3. **Evolution of $\alpha$ parameter during training.** We report the initial value ($\alpha_{\text{init}}$) and the final value after training convergence ($\alpha_{\text{final}}$) across different benchmarks.

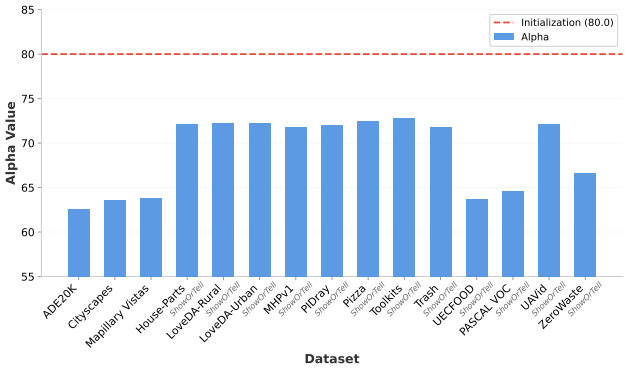


Figure 1. **Final $\alpha$ values across datasets.** We report the learned $\alpha$ values after training convergence for standard benchmarks (ADE20K, Cityscapes, Mapillary Vistas) and individual datasets in the ShowOrTell [11] benchmark. The red dashed line indicates the initialization value ($\alpha_{\text{init}} = 80$).

| imgs/class | Prompt Type | ADE20K | | | Cityscapes | | |
|---|---|---|---|---|---|---|---|
| | | PQ | mIoU | AP | PQ | mIoU | AP |
| 1 | V+T | 25.1±1.1 | 30.2±1.0 | 16.3±0.9 | 45.6±1.4 | 60.1±2.6 | 23.0±2.5 |
| 2 | V+T | 28.4±0.4 | 34.9±0.8 | 17.0±0.3 | 48.4±0.4 | 63.2±1.9 | 26.3±1.1 |
| 5 | V only | 29.3±0.3 | 36.6±0.5 | 16.1±0.4 | 47.7±0.8 | 62.8±1.0 | 23.2±0.9 |
| 5 | V+T | **31.4**±0.7 | **38.2**±0.5 | **18.1**±0.4 | **49.8**±0.5 | **66.2**±1.5 | **27.9**±0.7 |

Table 4. **Ablation on prompt type and number of support images.** We compare visual-only (V) and joint visual-textual (V+T) prompting strategies across varying numbers of support images per class on ADE20K [16] and Cityscapes [2].

| Model | ADE20K | | | Cityscapes | | |
|---|---|---|---|---|---|---|
| | PQ | mIoU | AP | PQ | mIoU | AP |
| MAFT+ [6] | 27.1 | 33.9 | 15.7 | 38.3 | 52.8 | 20.3 |
| **PrAda-L** (MAFT+) | 28.0±0.1 | 34.9±0.1 | 16.0±0.1 | 41.3±0.2 | 55.0±0.3 | 21.0±0.3 |

Table 5. **Application of PrAda to MAFT+ [6].** We report the performance of MAFT+ and our method when applied to MAFT+ across ADE20K [16] and Cityscapes [2].

moderate reduction compared to standard benchmarks. As illustrated in Figure 1, the final $\alpha$ values across individual ShowOrTell datasets exhibit notable variation, ranging from approximately 64 (PASCAL VOC, UECFOOD, ZeroWaste) to 73 (House-Parts, Toolkits, Pizza). This suggests that ShowOrTell's specialized domains (*e.g.*, aerial imagery, X-ray scans, waste management) require stronger visual prototype influence than standard scene parsing datasets but still benefit from adaptation. These adaptive behaviors demonstrate that allowing $\alpha$ to be trainable enables the model to automatically discover the optimal balance between text and visual prototypes for each specific domain.

**Prompt type and number of support images**. Table 4

ablates two key factors of our adaptation pipeline: the type of prompt used and the number of support images per class. Row (A) represents our full method using 5 images per class with combined visual and textual prompts (V+T), achieving the best performance on both ADE20K (31.4 PQ, 38.2 mIoU) and Cityscapes (49.8 PQ, 66.2 mIoU). Comparing rows (A) and (B) isolates the contribution of textual prompts: removing text and relying solely on visual prototypes (V only) consistently degrades performance by 2.1 PQ and 1.6 mIoU on ADE20K, confirming that combining visual and textual cues is beneficial. Rows (C) and (D) study the effect of reducing the number of support images to 1 and 2 respectively. Performance drops substantially with a single support image (row C), yielding 25.1 PQ and 30.2 mIoU on ADE20K, while using 2 images (row D) recovers most of the gap (28.4 PQ). These results show that our method scales gracefully with the number of support images, but benefits most from having at least 5 images per class to build reliable visual prototypes.

**Application to other methods**. PrAda can be applied to any open-vocabulary segmentation model built on M2F, including more recent methods like MAFT+ [6]. As depicted in Table 5, applying PrAda to MAFT+ yields consistent improvements across both ADE20K and Cityscapes, demonstrating the generality of our visual prototype learning strategy. On ADE20K, PrAda-L (MAFT+) achieves 28.0 PQ and 34.9 mIoU, improving over the original MAFT+ by 0.9 PQ and 1.0 mIoU. On Cityscapes, PrAda-L (MAFT+) reaches 41.3 PQ and 55.0 mIoU, outperforming MAFT+ by 3.0 PQ and 2.2 mIoU. These results confirm that our method can effectively enhance the adaptation capabilities of various open-vocabulary segmentation models by learning visual prototypes that complement their existing architectures.

## C. Additional studies on generalization

We extend the evaluation of zero-shot capabilities of FC-CLIP [13] on the SegInW benchmark presented in the main paper by reporting a complete comparison between zero-shot and oracle performance across all 25 datasets.

**Oracle analysis on SegInW**. Figure 2 presents a comprehensive comparison between zero-shot and oracle performance across all 25 datasets in the SegInW [19] benchmark. The oracle setting represents an upper bound where the model is evaluated using the classes present in the ground-truth masks, focusing purely on the model's ability to recognize and segment specific object categories. The results reveal significant heterogeneity in the zero-shot capabilities of FC-CLIP [13] across different domains. On one end of the spectrum, datasets like Hand, Chicken, and Fruits show relatively small gaps between zero-shot and oracle performance (less than 10 mAP difference), indicating that CLIP's pre-training already provides strong visual-semantic alignment for these common object categories. These datasets benefit from rich representation in web-scale training data, making few-shot adaptation less critical. Conversely, datasets such as Salmon-Fillet, Brain-Tumor, Electric-Shaver, and Watermelon exhibit substantial performance gaps exceeding 40-50 mAP, revealing severe domain misalignment. These specialized domains, featuring technical components, infrastructure elements, or objects with high visual ambiguity, are poorly represented in CLIP's pre-training data, making them prime candidates for few-shot adaptation. Datasets like Poles, Rail, and Cows show intermediate gaps (20-30 mAP), where zero-shot performance is moderate but substantial improvement is possible through adaptation.

## D. Efficiency Analysis

To better asses the efficiency of our method, we analyze the number of trainable parameters and training time required for adaptation across different datasets. Since we are training only the visual prototypes and the alpha parameter, the number of trainable parameters can be calculated as follows:

$$P_{\text{trainable}} = (N_{\text{classes}} + 1) \times D + 1 \tag{1}$$

where $(N_{\text{classes}} + 1)$ is the number of classes in the dataset plus the *void* class, $D$ is the dimensionality of the feature space (*i.e*. in our case 256).

For instance, ADE20K [16] with 150 classes requires 38,657 trainable parameters, Mapillary Vistas [10] with 65 classes requires 16,897 parameters, and Cityscapes [2] with 19 classes requires only 5,121 parameters. Thanks to this minimal parameter footprint, our method enables rapid adaptation. Adapting to standard benchmarks such as ADE20K or Cityscapes takes less than *30 minutes* on a single NVIDIA A5000 GPU, making our approach practical for real-world scenarios requiring efficient domain adaptation.

## E. Detailed results

In the following sections, we provide comprehensive results for our method and all baselines across the ShowOrTell [11] and SegInW [19] benchmarks. We analyze performance on each individual dataset, highlighting strengths and weaknesses of our approach compared to few-shot adaptation baselines.

### E.1. Detailed results on SegInW

Table 6 presents detailed results for our method and all baselines across the 25 diverse datasets in the SegInW [19] benchmark. Our method achieves the best average performance (43.3 mAP), outperforming FC-CLIP (41.6 mAP), CLIP-Adapter (42.1 mAP), CoOp (41.2 mAP), and

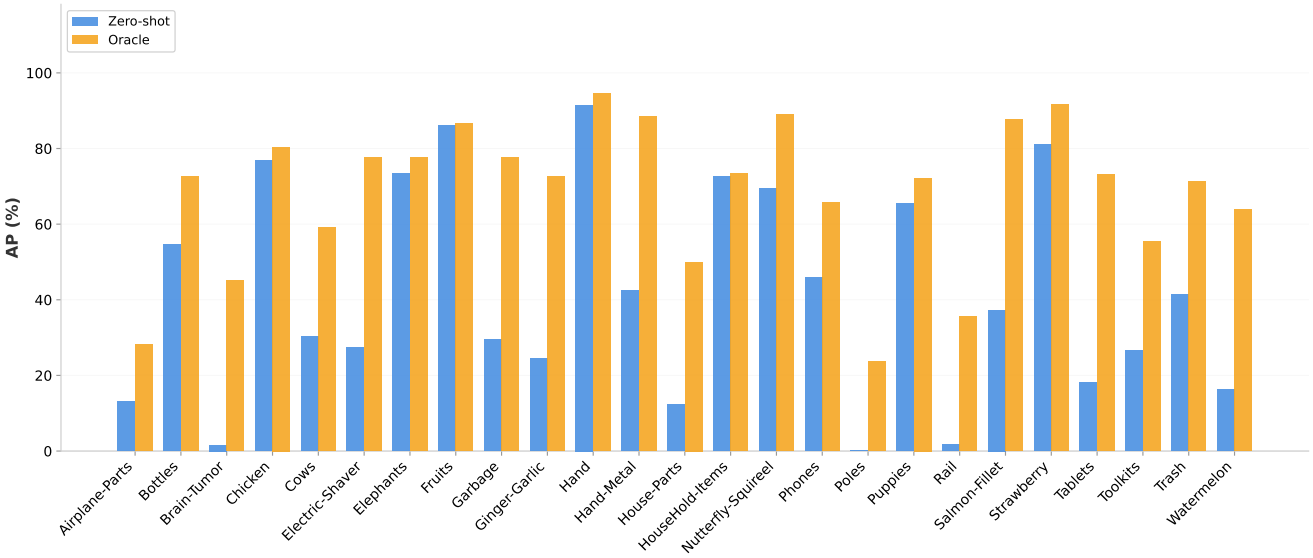


Figure 2. **Complete oracle results on SegInW [19] datasets.** We report zero-shot performance (mAP) of the original FC-CLIP [13] model across all 25 datasets in the SegInW benchmark, evaluated only on the classes present in the ground-truth masks. While some datasets show zero-shot performance close to the oracle upper bound, indicating strong pre-training alignment with those visual domains, other datasets exhibit significantly lower performance, revealing substantial domain gaps that highlight the need for adaptation.

CoCoOp (41.5 mAP). While no single method dominates across all datasets, our approach demonstrates particular strength in challenging scenarios with complex object categories, achieving the best performance on 8 datasets including Brain-Tumor (+3.0 mAP over FC-CLIP), Chicken (+0.8 mAP), Cows (+2.4 mAP), Phones (+3.6 mAP), Puppies (+5.0 mAP), Tablets (+24.0 mAP), and Toolkits (+8.9 mAP). Notably, our method shows robust performance across the highly varied domains in SegInW [19], from medical imaging (Brain-Tumor) to animals (Chicken, Cows, Puppies) to everyday objects (Phones, Tablets), highlighting the effectiveness of our visual prototype learning strategy for cross-domain adaptation. However, our method underperforms on certain datasets where the baseline FC-CLIP excels, particularly Bottles (32.8 mAP vs. 54.6 mAP), Fruits (47.4 mAP vs. 86.0 mAP), and Trash (23.3 mAP vs. 41.3 mAP). These cases suggest that when the frozen text embeddings already provide strong semantic alignment with the visual domain, our prototype adaptation may introduce unnecessary complexity. Additionally, datasets like Elephants (70.2 mAP vs. 73.3 mAP for FC-CLIP) and HouseHold-Items (61.4 mAP vs. 72.7 mAP) show that our method can struggle when few-shot examples are insufficient to capture the high intra-class variability present in these categories.

### E.2. Detailed results on ShowOrTell

Table 7 reports comprehensive results across the 14 datasets in the ShowOrTell [11] benchmark, including 12 target datasets plus ADE20K and Cityscapes. Our method achieves the highest average performance (33.1 mIoU), significantly outperforming the second-best TipAdapter-F (27.7 mIoU) by 5.4 mIoU. The improvements are particularly pronounced on datasets with large domain shifts from natural images: House-Parts (30.8 mIoU vs. 15.0 mIoU for TipAdapter-F), Pizza (30.7 mIoU vs. 19.5 mIoU), Zero-Waste (28.5 mIoU vs. 14.3 mIoU), and PASCAL VOC (74.5 mIoU vs. 67.8 mIoU). Our method achieves the best performance on 8 out of 14 datasets, demonstrating consistent superiority over prompt-learning methods (CoOp, CoCoOp) and adapter-based approaches (CLIP-Adapter, TipAdapter-F). The results confirm that learning visual prototypes in the feature space provides more robust adaptation than text-based prompt tuning, especially when dealing with specialized visual domains like aerial imagery (UAVid), waste management (Zero-Waste, Trash), and food recognition (Pizza, UECFood). Nonetheless, our method shows weaker performance on certain outdoor scene parsing datasets where TipAdapter-F performs better, specifically LoveDA-Rural (25.5 mIoU vs. 30.9 mIoU) and LoveDA-Urban (30.6 mIoU vs. 40.1 mIoU). Similarly, on UECFood (20.4 mIoU vs. 21.9 mIoU for TipAdapter-F) and Cityscapes (66.2 mIoU vs. 64.3 mIoU for CLIP-Adapter), the performance gaps are minimal, suggesting that certain well-structured domains with consistent visual appearance may not fully benefit from our prototype learning approach.

## F. Qualitative results

In this section, we present qualitative results to complement the quantitative analysis provided in the main paper and previous sections. We visualize predictions from

| Method | Airplane-Parts | Bottles | Brain-Tumor | Chicken | Cows | Electric-Shaver | Elephants | Fruits | Garbage | Ginger-Garlic | Hand | Hand-Metal | House-Parts | HouseHold-Items | Nutterfly-Squireel | Phones | Poles | Puppies | Rail | Salmon-Fillet | Strawberry | Tablets | Toolkits | Trash | Watermelon | **AVG** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *FC-CLIP* | 13.2 | **54.6** | 1.6 | 76.9 | 30.2 | 27.3 | 73.3 | **86.0** | 29.4 | 24.5 | 91.5 | 42.5 | **12.4** | **72.7** | 69.5 | 45.9 | 0.2 | 65.5 | 1.8 | 37.3 | **81.1** | 18.2 | 26.6 | **41.3** | 16.2 | 41.6 |
| FC-CLIP + CoOp | 12.6 | 24.3 | 0.1 | 68.1 | 18.8 | 55.5 | **74.8** | 85.3 | 35.8 | 21.7 | **94.0** | **61.2** | 7.3 | **72.7** | 70.9 | 47.1 | 1.2 | 62.8 | 1.6 | 26.7 | 78.1 | 15.8 | 31.5 | 35.8 | 26.8 | 41.2 |
| FC-CLIP + CoCoOp | 12.9 | 19.7 | 0.3 | 78.2 | 29.8 | **73.4** | **74.8** | 79.0 | 21.7 | 30.9 | **94.0** | 55.0 | 9.0 | **72.7** | **72.7** | 48.5 | **2.7** | 65.5 | 2.1 | 31.6 | 78.5 | 21.1 | 15.5 | 32.2 | 14.6 | 41.5 |
| FC-CLIP + CLIP-Adapter | 12.1 | 51.6 | 0.6 | 74.8 | 35.9 | 70.6 | 74.1 | 55.7 | **40.4** | 23.4 | **94.0** | 50.0 | 7.6 | 62.7 | 71.8 | 49.0 | 0.5 | 62.9 | 1.2 | **37.3** | 60.6 | 20.9 | 28.8 | 39.5 | 25.8 | 42.1 |
| **PrAda** | **13.5** | 32.8 | **4.6** | **79.0** | **38.3** | 73.1 | 70.2 | 47.4 | 34.0 | **29.2** | **94.0** | 54.0 | 10.3 | 61.4 | 71.7 | **53.6** | 1.6 | **70.5** | **2.3** | 22.5 | 75.1 | **44.2** | **35.5** | 23.3 | **40.7** | **43.3** |

Table 6. **Results for few-shot adaptation methods across 25 SegInW datasets.** For each dataset we report the average across 5 random seeds. We report mean Average Precision (mAP).

| Method | House-Parts | LoveDA-Rural | LoveDA-Urban | MHPv1 | PIDray | Pizza | Toolkits | Trash | UECFood | PASCAL VOC | UAVid | Zero-Waste | ADE20K | City-scapes | **AVG** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *FC-CLIP* | 7.5 | 29.3 | 39.1 | 8.0 | 8.9 | 15.5 | 20.1 | 10.5 | 19.4 | 38.1 | 33.3 | 5.5 | 34.1 | 56.2 | 23.3 |
| FC-CLIP + CoOp | 12.3 | 27.0 | 32.5 | 2.6 | 10.7 | 13.2 | 7.9 | 12.7 | 8.9 | 39.0 | 30.0 | 5.0 | 31.6 | 59.3 | 20.9 |
| FC-CLIP + CoCoOp | 10.3 | 25.7 | 32.5 | 3.5 | 8.3 | 12.9 | 6.8 | 10.4 | 12.6 | 41.9 | 33.6 | 9.3 | 31.9 | 60.5 | 21.4 |
| FC-CLIP + CLIP-Adapter | 9.9 | 28.7 | 34.8 | 8.9 | 11.3 | 16.3 | **27.9** | 15.8 | 21.1 | 46.5 | 35.7 | 6.3 | 37.5 | **64.3** | 26.1 |
| FC-CLIP + TipAdapter-F | 15.0 | **30.9** | **40.1** | 11.4 | 18.1 | 19.5 | 11.6 | 13.5 | **21.9** | 67.8 | 37.7 | 14.3 | **38.5** | 48.0 | 27.7 |
| **PrAda** | **30.8** | 25.5 | 30.6 | **12.6** | **19.6** | **30.7** | 27.3 | **17.3** | 20.4 | **74.5** | **41.3** | **28.5** | 38.2 | **66.2** | **33.1** |

Table 7. Results for few-shot adaptation methods across ShowOrTell datasets. For each dataset we report the average across 5 random seeds. We report mean Intersection over Union (mIoU).

our method across different benchmarks: ADE20K [16] (Fig. 3), Cityscapes [2] (Fig. 4), Mapillary Vistas [10] (Fig. 5), and diverse domains from the ShowOrTell [11] benchmark (Fig. 6). These visualizations illustrate how our visual prototype learning approach adapts to varying visual domains and demonstrate the quality of both panoptic and semantic segmentation predictions. For each example, we show the input image alongside the predicted masks, highlighting the model's ability to accurately segment objects and stuff categories across different scenarios.

## References


[1] Dina Bashkirova, Mohamed Abdelfattah, Ziliang Zhu, James Akl, Fadi Alladkani, Ping Hu, Vitaly Ablavsky, Berk Calli, Sarah Adel Bargal, and Kate Saenko. Zerowaste dataset: Towards deformable object segmentation in cluttered scenes. In *CVPR*, 2022. 1

[2] 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. 1, 2, 3, 5, 8

[3] Takumi Ege, Wataru Shimoda, and Keiji Yanai. A new large-scale food image segmentation dataset and its application to food calorie estimation based on grains of rice. In *Proceedings of the 5th international workshop on multimedia assisted dietary management*, 2019. 1

[4] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88:303–338, 2010. 1

[5] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. CLIP-Adapter: Better vision-language models with feature adapters. *IJCV*, 132(2):581–595, 2024. 1

[6] Siyu Jiao, Hongguang Zhu, Jiannan Huang, Yao Zhao, Yunchao Wei, and Humphrey Shi. Collaborative vision-text representation optimizing for open-vocabulary segmentation. In *ECCV*, 2024. 2, 3

[7] Jianshu Li, Jian Zhao, Yunchao Wei, Congyan Lang, Yidong Li, Terence Sim, Shuicheng Yan, and Jiashi Feng. Multiple-human parsing in the wild. *arXiv preprint arXiv:1705.07206*, 2017. 1

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 1

[9] Ye Lyu, George Vosselman, Gui-Song Xia, Alper Yilmaz, and Michael Ying Yang. Uavid: A semantic segmentation dataset for uav imagery. *ISPRS journal of photogrammetry and remote sensing*, 165:108–119, 2020. 1

[10] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The Mapillary Vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 1, 3, 5, 9

[11] Gabriele Rosi and Fabio Cermelli. Show or tell? a benchmark to evaluate visual and textual prompts in semantic segmentation. In *CVPR*, 2025. 1, 2, 3, 4, 5, 10

[12] Junjue Wang, Zhuo Zheng, Ailong Ma, Xiaoyan Lu, and Yanfei Zhong. Loveda: A remote sensing land-cover dataset for domain adaptive semantic segmentation. In *NeurIPS*, 2021. 1

[13] Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Convolutions die hard: Open-vocabulary segmentation with single frozen convolutional clip. *NeurIPS*, 2023. 3, 4

[14] Libo Zhang, Lutao Jiang, Ruyi Ji, and Heng Fan. Pidray: A large-scale x-ray benchmark for real-world prohibited item detection. *IJCV*, 131(12):3170–3192, 2023. 1

[15] Renrui Zhang, Wei Zhang, Rongyao Fang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-Adapter: Training-free adaption of clip for few-shot classification. In *ECCV*, 2022. 1

[16] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 1, 2, 3, 5, 7

[17] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, 2022. 1

[18] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei

Liu. Learning to prompt for vision-language models. *IJCV*, 130(9):2337–2348, 2022. 1

[19] Xueyan Zou, Zi-Yi Dou, Jianwei Yang, Zhe Gan, Linjie Li, Chunyuan Li, Xiyang Dai, Harkirat Behl, Jianfeng Wang, Lu Yuan, et al. Generalized decoding for pixel, image, and language. In *CVPR*, 2023. 1, 2, 3, 4


Figure 3. **Qualitative results on ADE20K [16].** For each image, the first row contains *panoptic segmentation* predictions, while the second row contains *semantic segmentation* predictions.

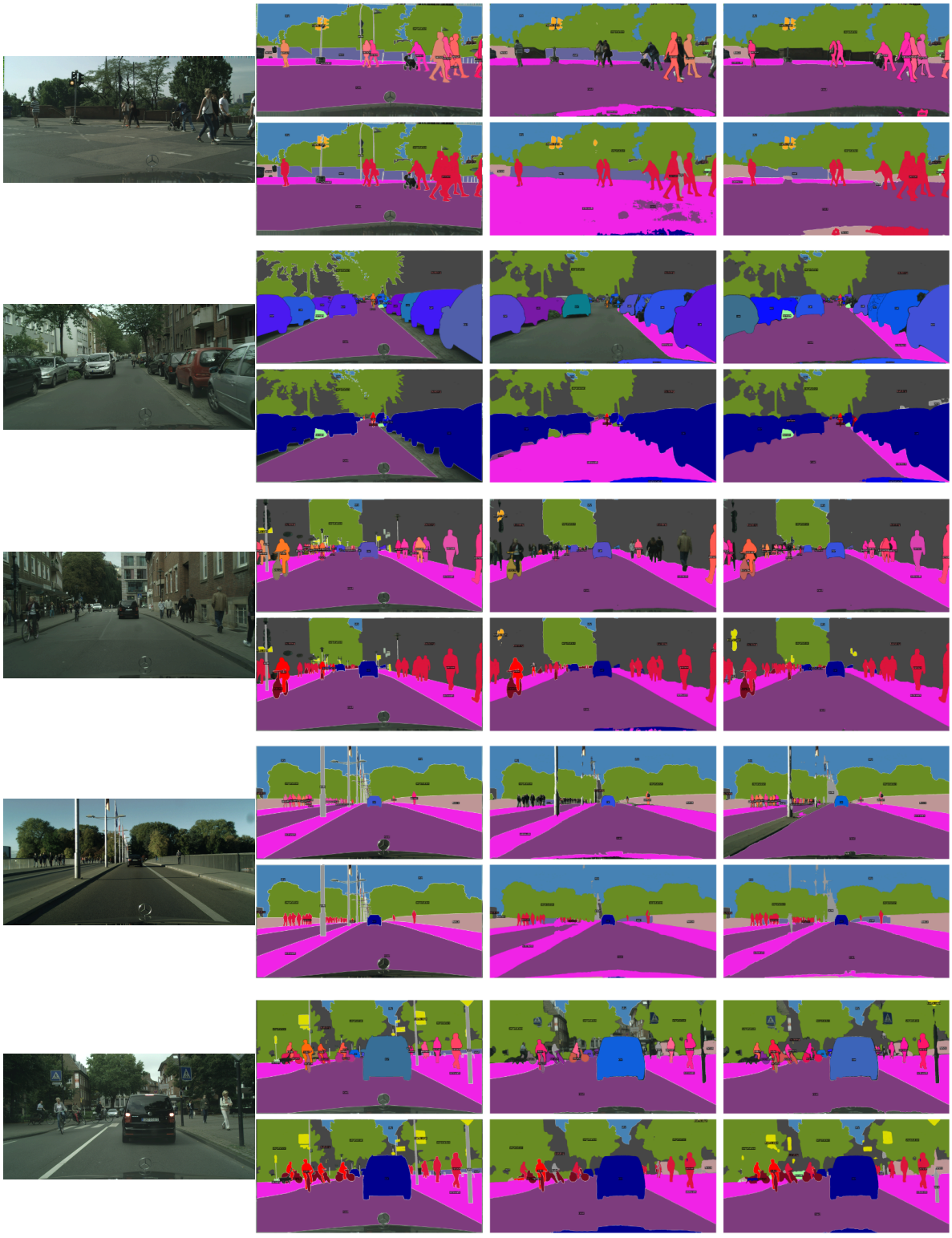


Figure 4. **Qualitative results on Cityscapes [2].** For each image, the first row contains *panoptic segmentation* predictions, while the second row contains *semantic segmentation* predictions.

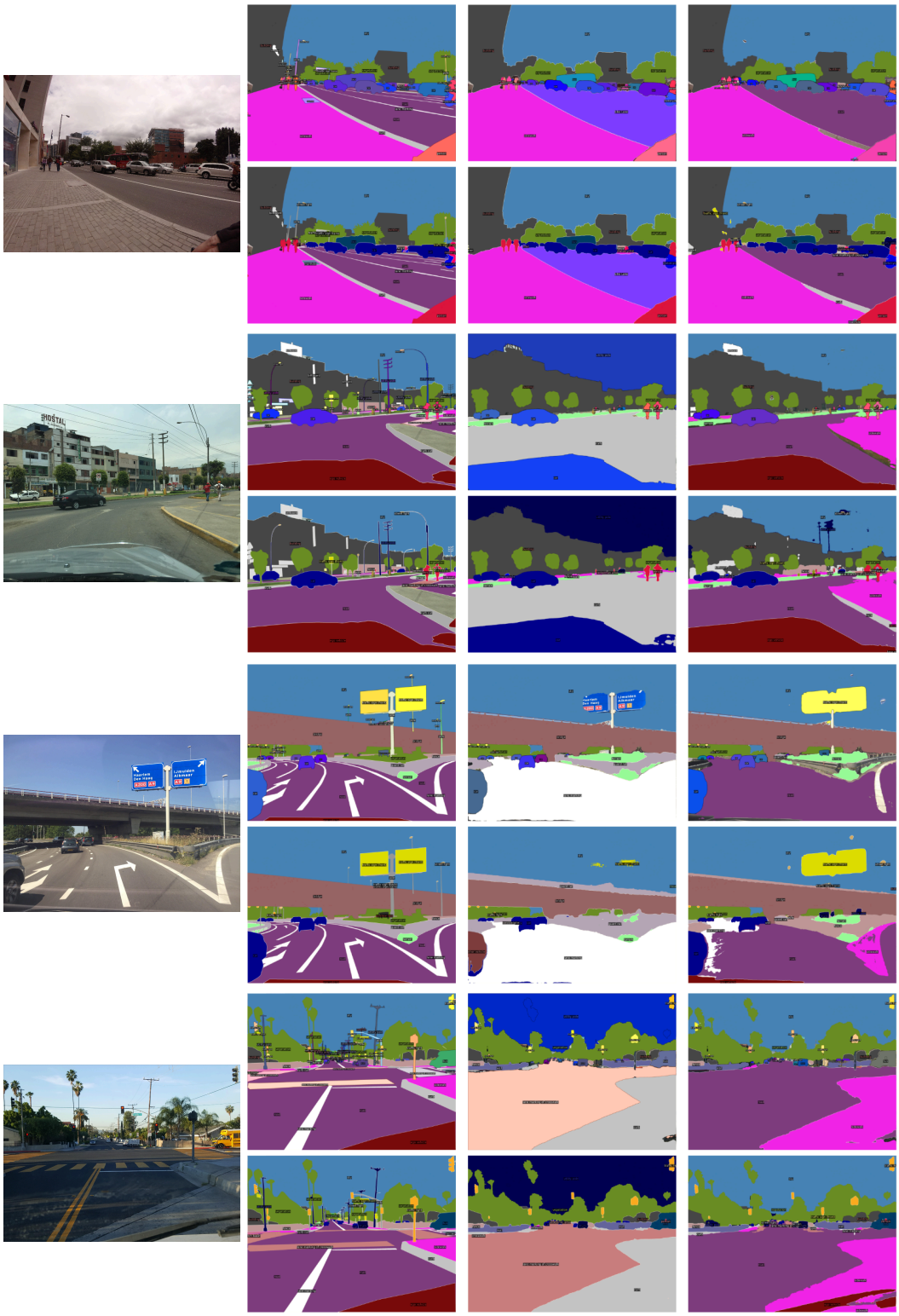


Figure 5. **Qualitative results on Mapillary Vistas [10].** For each image, the first row contains *panoptic segmentation* predictions, while the second row contains *semantic segmentation* predictions.

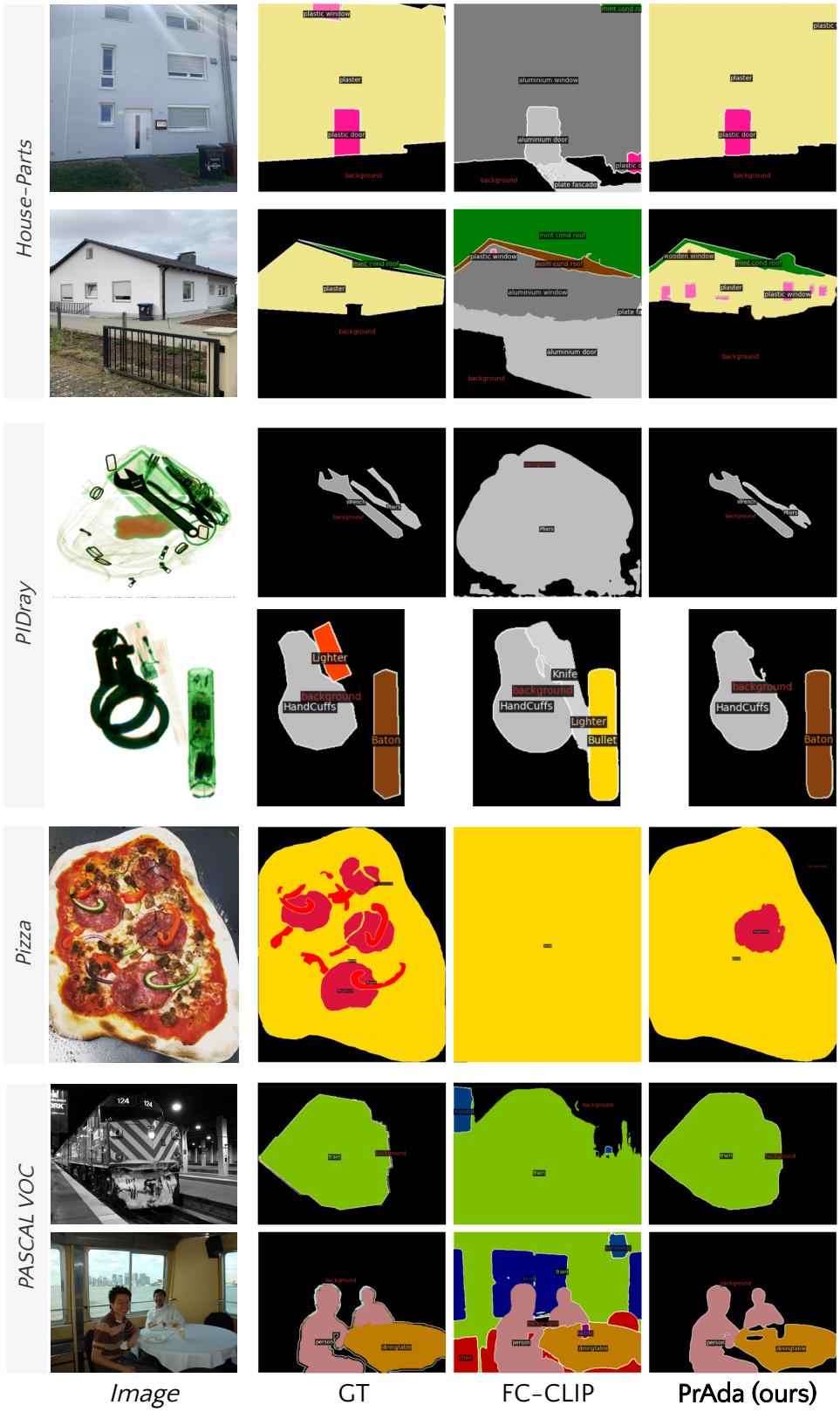


Figure 6. **Qualitative results on ShowOrTell [11] datasets.** We select a subset of datasets composed by: House-Parts, PIDray, Pizza and PASCAL VOC.