Concept-Aware LoRA for Domain-Aligned Segmentation Dataset Generation

Supplementary Material

Appendix

In the supplementary material, we provide additional analyses, experiments, qualitative and quantitative results, and implementation details, including pseudocode, due to space limitations in the main manuscript. Specifically, we describe implementation details, including PyTorch-like pseudocode, to ensure reproducibility (Appendix A). Next, we illustrate an implicit bias of concept gradients (Appendix B) and additional experimental studies of our design choice of CA-LoRA (Appendix C). We then demonstrate the generalizability of the proposed method using a general-domain dataset, Pascal VOC [11] (Appendix D). Finally, we discuss class-aware dataset generation, which could be a promising future research direction for handling long-tailed distributions by leveraging text prompts (Appendix E). Below is the table of contents for the supplementary material.

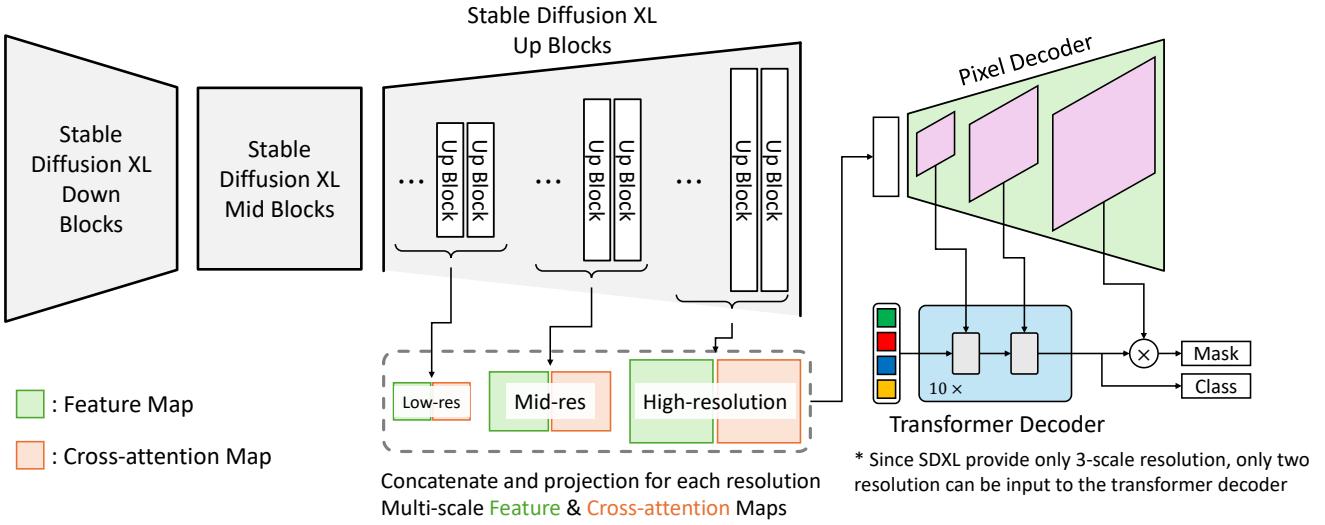


Figure 7. The detailed label generator architecture. The whole framework includes a text-to-image generation model (Stable Diffusion XL), pixel decoder, and transformer decoder, followed by DatasetDM [51]. Due to the change in the architecture of the text-to-image generation model, the following pixel decoder and transformer decoder minorly changed (*e.g.*, the number of input channels and the number of blocks).

A. Implementation Details

A.1. Label Generator Architecture

We build the label generator based on the recent segmentation dataset generation framework, DatasetDM [51]. The label generator in DatasetDM, called P-Decoder, is derived from the Mask2Former [5] decoder architecture. It takes intermediate features from the T2I model, including feature maps and cross-attention maps. The label generator then concatenates features of the same resolution and reduces the feature dimensions using predefined projection layers. The multi-resolution feature maps are passed through the pixel decoder and the transformer decoder sequentially, which outputs the segmentation predictions. Finally, we calculate the loss function of the label decoder, which mirrors that of Mask2Former, incorporating binary cross-entropy, dice loss, and classification loss. However, several modifications exist between the original DatasetDM P-Decoder and our label generator due to architectural differences between Stable Diffusion v1.5 [41] and Stable Diffusion XL [37].

Since DatasetDM is built on top of Stable Diffusion v1.5 [41], we simply adjust the feature dimensions in the projection layers to accommodate Stable Diffusion XL [37]. The detailed label generator architecture is illustrated in Fig. 7. However, if all feature maps and cross-attention maps are used, the total number of channels increases significantly, leading to an unmanageable number of parameters in the projection layers during concatenation and projection. In summary, the feature maps are extracted from the *last feature block* at each resolution of the up-sampling blocks, while cross-attention maps are sampled at equal intervals (every 7 blocks) from the total 36 up-sampling blocks (i.e., 1st, 8th, ... 29th, 36th).

Furthermore, as shown in Fig. 7, Stable Diffusion XL has only three resolution levels, compared to four resolution levels of the Stable Diffusion v1.5 architecture in the original DatasetDM. In the Mask2Former structure, feature maps from the pixel decoder, excluding the largest resolution, are fed into the transformer decoder. While the original design used three-resolution feature maps, only two were utilized in this case. Thus, while DatasetDM provides three-resolution feature maps three times for 9 transformer decoder blocks, we provide two-resolution feature maps five times, leading to a total of 10 transformer decoder blocks. *Importantly, to ensure a fair comparison, the reported scores for DatasetDM were obtained using a re-implemented version based on SDXL with the same modifications.*

A.2. Detailed Architecture of CA-LoRA

Overview (Fig. 8 (a)) The fundamental group of weights used to measure concept awareness is the linear projection layer. We selectively adapt the pretrained weights layer by layer within the projection layers. Practically, the projection layers in the multi-head attention block are integrated across heads, we split them to structurally distinguish the weights for each projection. Consequently, we attach LoRA layers projection-wise, as shown in Fig. 8.

Projection-wise CA-LoRA (Fig. 8 (b)) There are two types of CA-LoRA: output LoRA projection layers (Type A: Output (OUT)) and input LoRA projection layers (Type

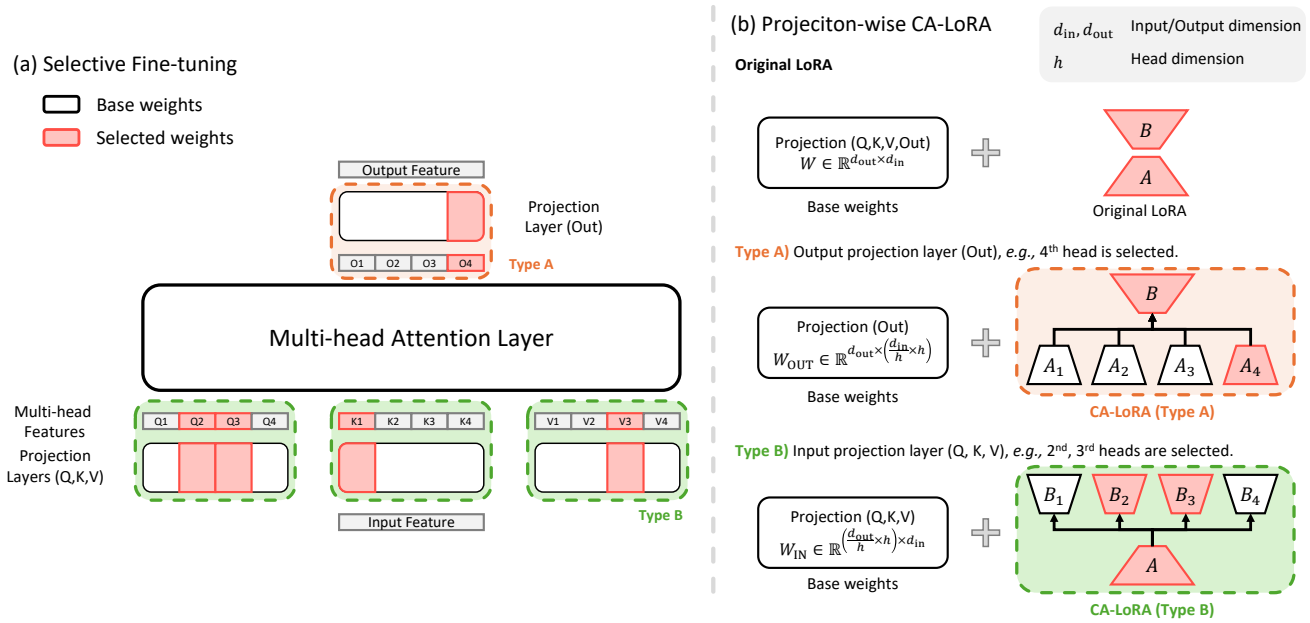


Figure 8. The detailed architecture of the CA-LoRA. We conduct projection-wise CA-LoRA that can attach the LoRA layer for each projection layer of multi-head self-attention.

B: Query (Q), Key (K), Value (V)). To split the original LoRA layer ($\Delta W = BA$) in projection-wise, the output projection LoRA layers (ΔW_{OUT}) split the A weights row-wise, while the input projection LoRA layers (ΔW_{IN}) split the B weights column-wise, as illustrated in the following equations.

$$\Delta W_{\text{OUT}} = B \begin{bmatrix} A_1 & A_2 & \cdots & A_h \end{bmatrix}, \tag{8}$$

$$\Delta W_{\text{IN}} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_h \end{bmatrix} A. \tag{9}$$

The projection-wise LoRA is represented in Fig. 8 and Algorithm 4.

A.3. Hyperparameters and Pseudocode

Hyperparameters (Tables 12, 13, 14 and 15) We provide all hyperparameters to support reproducibility. In the first stage, we fine-tune Stable Diffusion XL [37] using the HuggingFace Diffusers library [49]. The specific hyperparameters for fine-tuning Stable Diffusion XL on the Cityscapes dataset are listed in Tab. 12, while the training configurations for the label generator can be found in Tab. 13.

Next, we train segmentation models for both in-domain and domain generalization scenarios. The hyperparameters for in-domain fine-tuning are provided in Tab. 14,

Table 4. **Hyperparameter search of layer proportions.** Image-domain alignment (CMMD) and final segmentation performance.

Proportion	CMMD	Performance (mIoU)
0% (Pretrained)	5.063	42.82
1%	1.618	43.77
2%	1.420	**44.13**
3%	1.021	<u>43.94</u>
5%	1.105	43.81
10%	<u>0.686</u>	43.36
100% (Original LoRA)	**0.644**	42.97

while those for domain generalization, based on the DGInStyle [22] method, are included in Tab. 15. We hope these provided hyperparameters will facilitate reproducibility.

Pseudocode We also provide PyTorch-like pseudocode [36] for key algorithms to effectively support reproducibility. **Concept awareness (Algorithms 1 and 2)** The concept awareness algorithm is demonstrated in Algorithm 1. Conducting concept awareness requires several helper functions, as shown in Algorithm 2. **CA-LoRA (Algorithms 3 and 4)** The CA-LoRA algorithm is divided into two parts: the forward function and the declaration function. The forward pass of CA-LoRA is presented in Algorithm 3, while the declaration function, along with the selected layers, is illustrated in Algorithm 4.

While we provide the PyTorch-like pseudocode based on HuggingFace Diffusers library [49], HuggingFace PEFT-based implementation [32] can reduce the training time of the CA-LoRA.

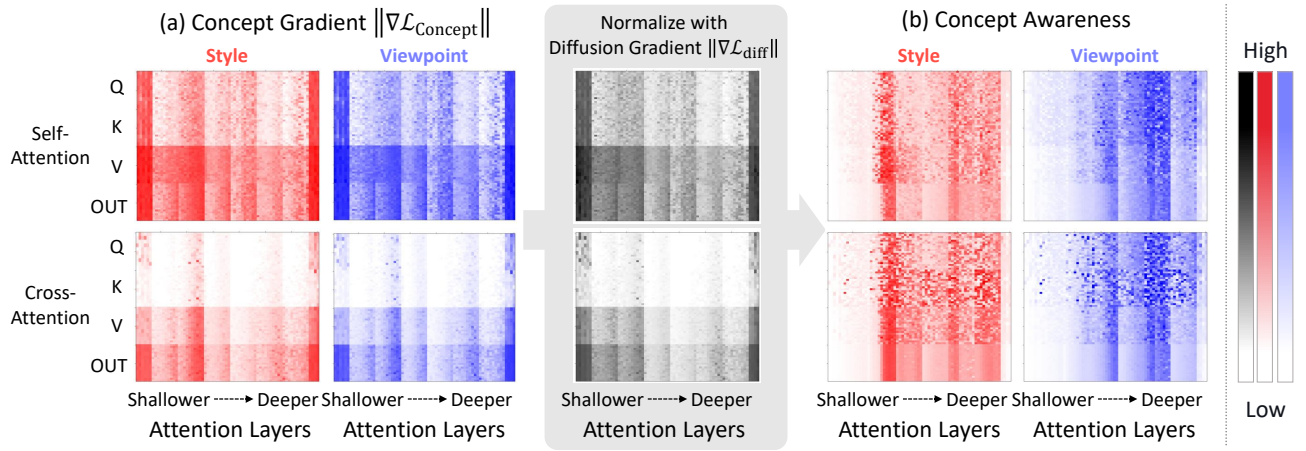


Figure 9. **Visualizing the gradients and concept awareness.** (a) Implicit gradient biases across network layers. (b) Concept awareness, computed by normalizing each concept gradient with respect to the original diffusion-loss gradient. Style-aware and viewpoint-aware weights are highlighted in red and blue, respectively. Although raw concept gradients alone cannot clearly separate different concepts, normalization with diffusion gradients yields a more distinguishable concept-awareness signal.

B. Visualizing Concept Awareness

B.1. The Implicit Bias of Gradients Across Layers

Observation (Fig. 9 (a)) We compute the awareness scores for each layer by using the norm of the gradient. However, the gradient norm cannot be uniformly scaled across different head types (Q, K, V, OUT), attention types (self, cross), and layers (shallow, deep). To address this, we construct a base gradient to scale the concept loss gradient by referencing the gradient of the original diffusion loss, as described in Section 3.2. We visualize the gradients of both the concept losses and the original diffusion loss in Fig. 9. In this visualization, we separate the self-attention and cross-attention layers to provide clearer distinctions, which differs from the approach in the main paper.

Normalizing Gradients (Fig. 9 (b)) Therefore, we normalize the gradients using the gradients calculated from the original diffusion loss, as discussed in Section 3.2 and shown in Fig. 3. As shown in Fig. 9 (a), the gradients calculated from the style concept loss and viewpoint concept loss are similar. However, the gradient increase ratio can differ significantly, as illustrated in Fig. 9 (b).

B.2. Concept Awareness per Prompt Augmentation

We conduct an additional analysis of the robustness of defining desired concepts based on prompt augmentation. As demonstrated in Section 3.2, we select several handcrafted prompt augmentations tailored to the desired concept. While this method is flexible and can be generalized to different concepts, it may introduce instability in prompt augmentation. To assess its robustness, we measure awareness across various prompt augmentations. Specifically, we provide five prompt augmentations for each desired concept (style, viewpoint) derived from the original prompts, as shown below.

$$c_{\text{Aug(Style)}} \in \left\{ \begin{array}{l} \text{“Sketch of first-person urban street view”,} \\ \text{“Watercolor of first-person urban street view”,} \\ \text{“Pop-art of first-person urban street view”,} \\ \text{“Line art of first-person urban street view”,} \\ \text{“Oil painting of first-person urban street view”} \end{array} \right\}$$

$$c_{\text{Aug(Viewpoint)}} \in \left\{ \begin{array}{l} \text{“Photorealistic urban street in top-down view”,} \\ \text{“Photorealistic urban street in high angle view”,} \\ \text{“Photorealistic urban street in low angle view”,} \\ \text{“Photorealistic urban street in eye-level view”,} \\ \text{“Photorealistic urban street in close-up view”} \end{array} \right\}$$

We then calculate the style and viewpoint awareness for each prompt augmentation, as shown in Fig. 10. As illustrated in the figure, our proposed method consistently demonstrates high awareness to similar regions across all prompt augmentations for styles. Similarly, for viewpoints, augmentations such as top-down, high-angle, and low-angle were applied, and the results indicate that our method highlights similar regions regardless of the specific viewpoint prompt. Based on these findings, we manually select the first three prompts for each desired concept. Nevertheless, developing an automated approach to search for prompt augmentations by leveraging recent LLMs could be a promising direction for enhancing concept awareness.

B.3. Concept Awareness per Noise Timestep

The amount of added noise, defined by the timestep t, is critical hyperparameter for calculating concept awareness. We conduct comprehensive experiments to evaluate con-

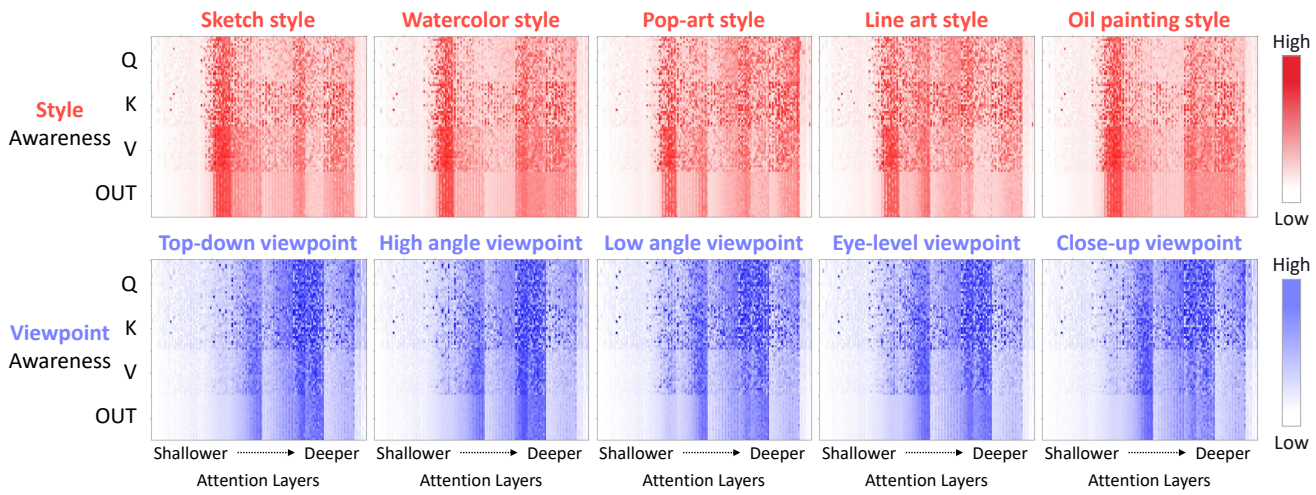


Figure 10. **Measured concept awareness according to the various prompt augmentation.** The highlighted concept-aware layers for each concept (style and viewpoint) remained largely consistent regardless of the prompt augmentation, demonstrating the robustness of concept awareness to variations in prompt design.

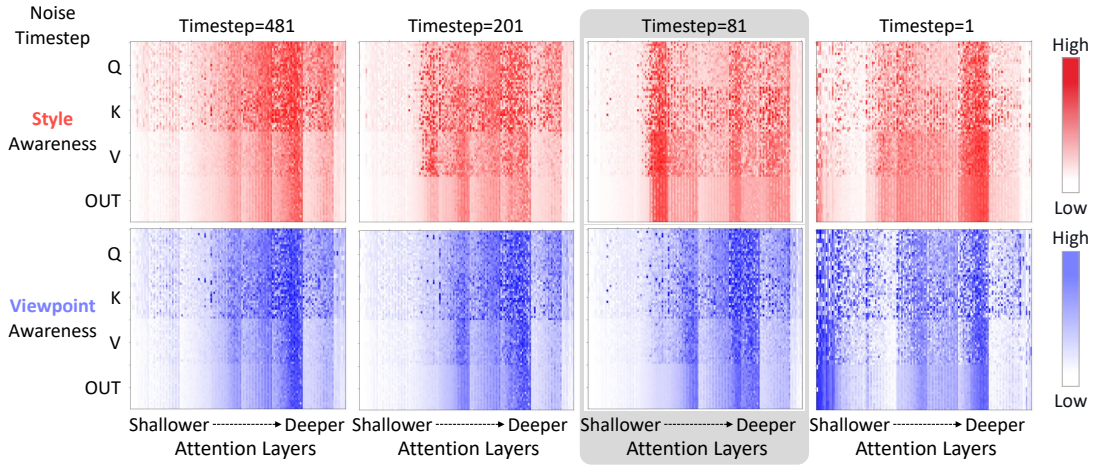


Figure 11. **Visualizing concept awareness across different noise timesteps.** It shows that the 81st timestep stands out with a significantly distinct concept awareness score between style and viewpoint awareness compared to the other timesteps.

cept awareness in relation to the noise timestep. Visualizations of concept awareness across different noise timesteps, along with qualitative and quantitative results, are provided. These experimental findings offer valuable insights into the behavior of concept awareness.

Visualization (Fig. 11) According to our experiment, calculating concept awareness at large timesteps (noisy images) does not yield meaningful information about concept awareness. For example, style and viewpoint sensitivities appear similar when the timestep is set to 481 out of 1000, as shown in the first column of Fig. 11. This occurs because concept-aware layers are less responsive to noisy inputs, which have a high potential to generate any image. Conversely, extremely small timesteps (*e.g.*, 1) also fail to capture concept awareness, as the loss from almost clean images does not provide sufficient generative information. Therefore, we explored intermediate timesteps (*e.g.*, 201, 81) and found that the 81st timestep reveals distinct concept sensitivities for style and viewpoint.

Qualitative Results (Fig. 12) Additionally, we fine-tuned 2% of the selected ratio using each concept awareness and generated images to qualitatively compare results across different noise timesteps. As shown in Fig. 12, the images generated using intermediate timesteps (201, 81) better align with the intended style and viewpoint.

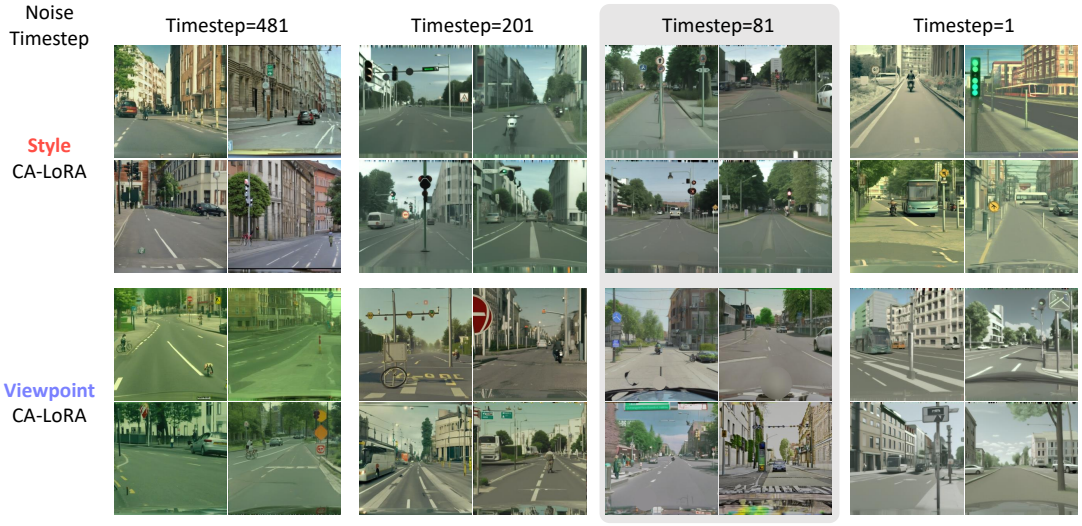


Figure 12. **Qualitative comparison across different noise timesteps.** According to the various noise timestep, 81st timestep represents the best concept awareness, qualitatively. The style of the generated images by Style CA-LoRA is well-aligned, while the generated images by Viewpoint CA-LoRA contain diverse styles.

Table 5. **Timestep ablation.** The CMMD (↓) and final segmentation performance of the CA-LoRA across the extracted timesteps.

Timestep	CMMD (↓)	Performance (↑)
1	2.383	40.49
81	**1.420**	**44.13**
201	<u>1.556</u>	<u>41.68</u>
481	1.920	40.80

Quantitative Results (Tab. 5) Finally, we quantitatively compared the intermediate timesteps using the image domain alignment metric, CMMD (↓) [21], to evaluate the 201st and 81st timesteps. The results indicate that the 81st timestep is the most effective for measuring concept awareness, as shown in Tab. 5. While our approach selects a single timestep to measure concept awareness, averaging multiple timesteps could improve the precision and robustness of concept awareness, which may be a promising direction for future research.

B.4. Layer selection strategy of CA-LoRA

Grouping Granularity of CA-LoRA (Fig. 13 and Tab. 6) As illustrated in Fig. 13, we explore the performance of CA-LoRA variants that partition the base weights differently. As shown in Tab. 6, these variants generally outperform the baselines. Among them, projection-wise CA-LoRA achieves the highest performance compared to other variants. This suggests that increasing the granularity of the base partition generally enhances performance, as it allows for more fine-grained selection of sensitive weights.

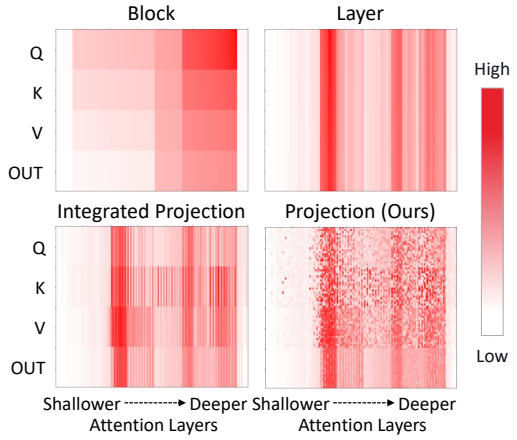


Figure 13. **Visualizing the concept awareness (style) according to various grouping granularity.**

Table 6. **Granularity ablation.** Number of trainable parameters and final segmentation performance by changing grouping granularity. Only 2% of projection-wise parameters can achieve the best results across all grouping granularity.

Grouping Granularity	% Params.	Performance (↑)
No Finetuning (DatasetDM)	0%	42.82
Block-wise	32%	<u>43.67</u> (+0.85)
Layer-wise	4.8%	43.26 (+0.44)
Integrated Projection-wise	5%	43.63 (+0.81)
Projection-wise (Ours)	2%	**44.13** **(+1.31)**

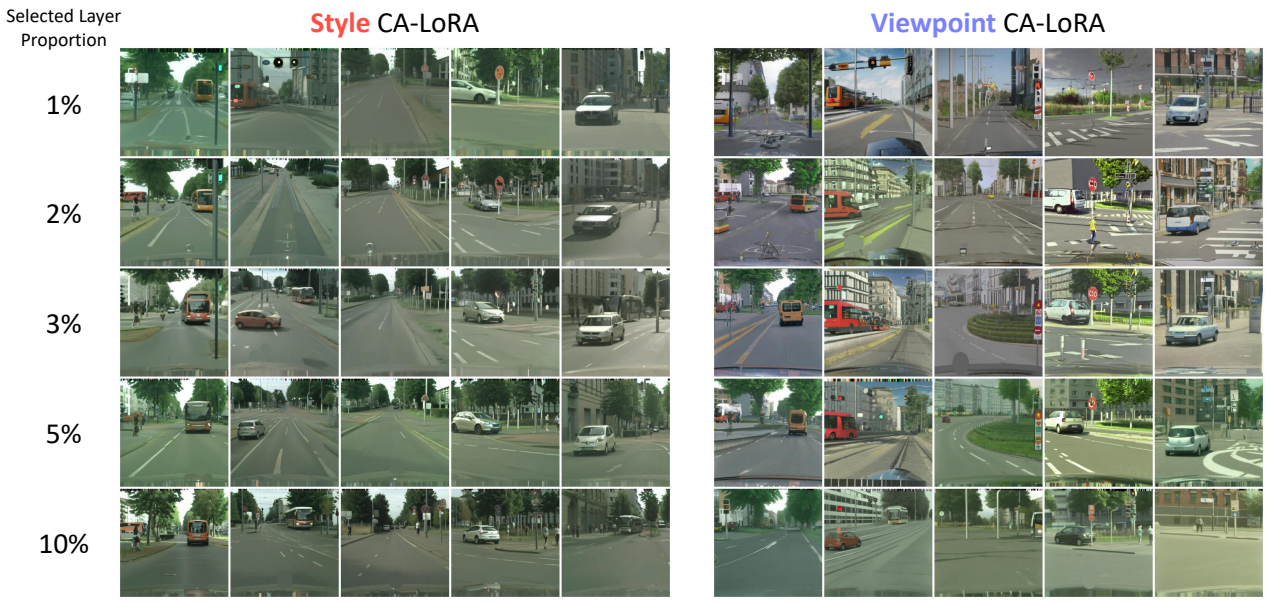


Figure 14. **Qualitative results of Style- and Viewpoint CA-LoRA across different layer proportions (1%, 2%, 3%, 5%, and 10%).** Selecting only 2% of the layers is already sufficient to learn the target concepts, while larger layer proportions increasingly introduce unintended concept leakage.

Layer Proportions of CA-LoRA (Fig. 14) We analyze the qualitative behavior of Style CA-LoRA and Viewpoint CA-LoRA when varying the proportion of selected layers (1%, 2%, 3%, 5%, and 10%). The results show that selecting only 2% of the layers is sufficient to learn the intended concepts: Style CA-LoRA produces images that accurately reflect the target style, whereas Viewpoint CA-LoRA generates images aligned with the desired viewpoint. However, as the layer proportion increases, the model gradually begins to encode unintended, non-target concepts, indicating growing interference between concepts at higher proportions.

C. CA-LoRA for Domain Generalization

In this section, we comprehensively compare the performance of CA-LoRA with baseline methods used for domain-generalized urban-scene segmentation. We apply both image-domain alignment and image-label alignment when constructing the domain-generalization dataset, following the procedure described in Section 4.3, and also provide the comparison with DGInStyle [22].

Image Domain Alignment (Tab. 7) For domain generalization in urban-scene segmentation, we generated urban-scene images under various adverse weather conditions (*e.g.*, "foggy", "night-time", "rainy", and "snowy"). In this section, we assess the domain gap between our generated adverse weather conditions and the real ACDC [45] dataset. Quantitatively, we used CMMD [21] to measure image domain alignment, and the results are presented in Tab. 7. These results show that the proposed generated dataset demonstrates a significant performance gap over DatasetDM and InstructPix2Pix. More importantly, it achieves competitive performance with DATUM, which requires training *individual models* separately for each weather condition *using a target domain image from the ACDC dataset*.

Table 7. **Image-domain alignment in domain generalization.** † DATUM utilizes one provided target-domain image per condition, whereas the other methods do not.

Method	Foggy	Night-time	Rainy	Snowy	Average
DATUM†	**2.41**	**2.46**	2.91	**2.10**	**2.47**
InstructPix2Pix	3.43	3.13	2.99	3.32	3.22
DatasetDM	4.90	5.52	5.34	4.96	5.18
Ours	2.43	2.55	**2.62**	2.63	2.56

Table 8. **Image-label alignment in domain generalization.** The first two rows report the segmentation performance of the pretrained and finetuned Mask2Former (M2F) [5] on the ACDC adverse-weather dataset [45]. The following four rows present a comparison of image-label alignment across baseline methods, evaluated using the finetuned M2F model.

Method	Foggy	Night-time	Rainy	Snowy	Average
Pretrained M2F	67.66	23.17	51.94	47.55	47.58
Finetuned M2F	**78.54**	**52.16**	**66.23**	**74.79**	**67.93**
DATUM	-	-	-	-	-
InstructPix2Pix	25.98	**48.60**	**63.04**	40.66	**44.57**
DatasetDM	40.84	35.90	47.43	**44.02**	42.05
Ours	**41.55**	43.07	48.69	39.47	43.20

Table 9. **Comparison with DGInStyle of generated datasets for domain generalization of urban-scene segmentation (mIoU).** We highlight the performance improvements are larger than DGInStyle, especially on HRDA backbone. *The gray color indicates the reported score from the DGInStyle authors.

DG Method	Method	ACDC	DZ	BDD	MV	Average
ColorAug	Baseline	52.38	23.00	53.33	60.06	47.19
	DGInStyle	55.19	26.83	55.18	59.95	49.29 (+2.10)
	Baseline	53.12	25.69	53.00	59.81	47.91
	CA-LoRA	56.07	29.75	54.35	61.40	**50.39** (+2.49)
DAFormer	Baseline	55.15	28.28	54.19	61.67	49.82
	DGInStyle	57.74	28.55	56.26	62.67	51.31 (+1.48)
	Baseline	53.98	27.82	54.29	62.69	49.70
	CA-LoRA	55.83	31.68	54.68	63.09	**51.32** (+1.63)
HRDA	Baseline	59.70	31.07	58.49	68.32	54.40
	DGInStyle	61.00	32.60	58.84	67.99	55.11 (+0.71)
	Baseline	58.48	29.46	56.12	64.27	52.08
	CA-LoRA	58.93	34.41	56.56	64.54	**53.61** (+1.53)

Image-Label Alignment (Tab. 8) We compare image-label alignment to assess how reliably label maps can be produced for domain-generalization datasets. Because the generated images do not have real ground-truth annotations, we instead use pseudo ground-truth produced by a highly accurate segmentation model. As no off-the-shelf urban-scene segmentation model achieves consistently strong performance across diverse domains, we manually finetune a reliable pretrained segmentor (Mask2Former [5]) on the ACDC dataset [45]. The first two rows in the figure show the segmentation performance on the ACDC validation set before and after finetuning, respectively.

The results of measuring image-label alignment using the fine-tuned M2F models are provided in Tab. 8. As shown in the table, our approach achieves superior image-label alignment compared to DatasetDM also on domain generalization setting. InstructPix2Pix, which directly reuses Cityscapes labels and applies only minor weather edits in image, exhibits an advantage in image-label alignment. However, despite this strong alignment, we previously highlighted its limited performance gains in Tables 1 and 2, which we attribute to the lack of scene diversity inherent to its use of fixed segmentation label maps (see Fig. 5).

Comparison with DGInStyle (Tab. 9) As shown in Tab. 9, our CA-LoRA achieves performance gains comparable to, and in most cases larger than, those reported for DGInStyle [22]. As their code and Cityscapes-generated data are unavailable, and their baseline accuracy is marginally higher than ours, we compare only the performance gains rather than the absolute numbers. We will update the comparison once the official code becomes available. Notably, our approach consistently exhibits substantial improvements when applied to the HRDA method. In contrast, DGInStyle generates images using fixed label maps, which limits its ability to construct semantically diverse scenes. We hypothesize that this restricted diversity contributes to the relatively small performance gains observed when DGInStyle is used with advanced DG methods such as HRDA.

D. Experiments in General Domain

Since our primary goal is to cover urban-scene segmentation, we focused on style and viewpoint as the desired concepts and conducted experiments exclusively on urban-scene datasets such as Cityscapes. However, the CA-LoRA methodology is not limited to urban-scene datasets. It can also be applied to general datasets for in-domain segmentation dataset generation. In this section, we demonstrate experiments on the Pascal-VOC dataset [11], showcasing how our approach improves few-shot semantic segmentation performance.

D.1. Experimental setup

In this experiment, we trained on a total of 100 real image-label pairs and evaluated the model using the 1,449 images in the Pascal-VOC validation set. For the text-to-image generation model, we applied the same style awareness score used in the Cityscapes experiment, setting the selected proportion to 10%. During the training of the text-to-image generation model, the prompt "a photo" was used. For training the label generator and generating the dataset, the prompt "a photo of a {class names}" was employed. The label generator was trained with a batch size of 4 for 90K iterations, ultimately producing 2,000 image-label pairs. When utilizing the generated dataset, the process was consistent with the in-domain semantic segmentation experiments. Specifically, Mask2Former was trained on the real dataset for 90K iterations (Baseline), followed by finetuning on the combined real and generated dataset for an additional 30K iterations. All other hyper-parameters remained identical to those used in the Cityscapes in-domain semantic segmentation experiment, as detailed in Tables 12 to 14. Notably, the diffusion timestep ($t = 81$) and the real-to-synthetic data ratio ($1 : 1$) are directly transferred from the Cityscapes experiment without re-tuning, demonstrating the robustness of our key hyperparameters across datasets.

D.2. Quantitative and Qualitative Results

As shown in Tab. 10, using Style CA-LoRA on the Pascal-VOC dataset resulted in a performance improvement of 0.93 mIoU. In contrast, DatasetDM, which omitted the fine-tuning process for the text-to-image generation model, showed a performance drop of 8.43 mIoU. This highlights the importance of concept-aware finetuning for style, even in general datasets beyond urban-scene datasets. Fig. 15 provides further insight into the role of style information. A significant image domain gap is evident between the images generated by the pretrained text-to-image generation model and the dataset generated using Pascal-VOC. This

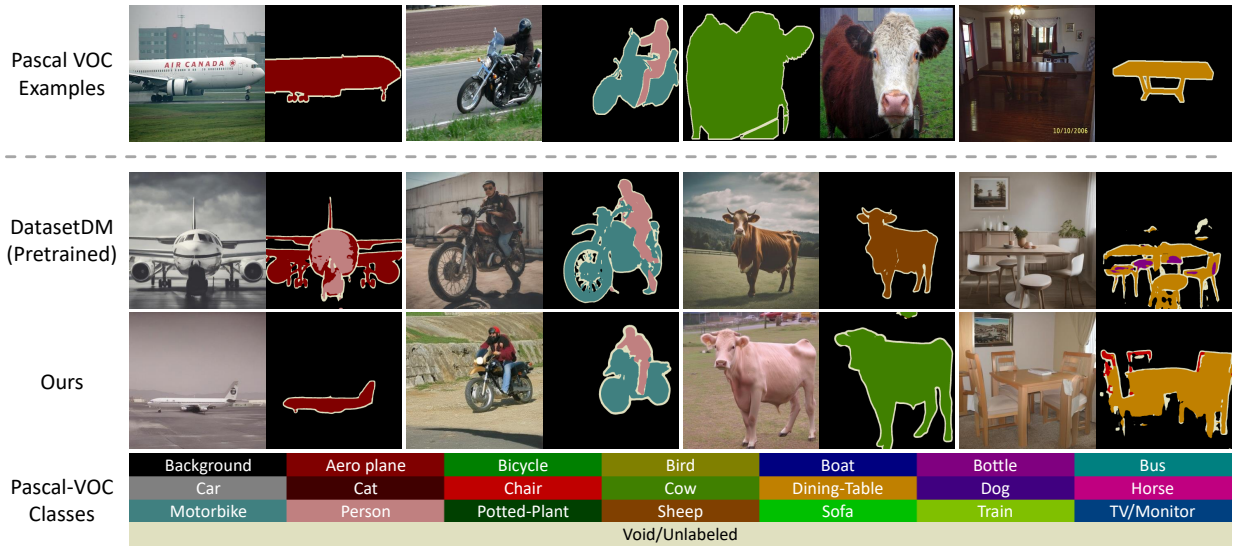


Figure 15. **Qualitative comparison for generating Pascal-VOC dataset.** Although both DatasetDM and ours are trained on the 100 labeled samples, our generated dataset shows better image domain alignment with the original Pascal-VOC examples and also shows better image-label alignment.

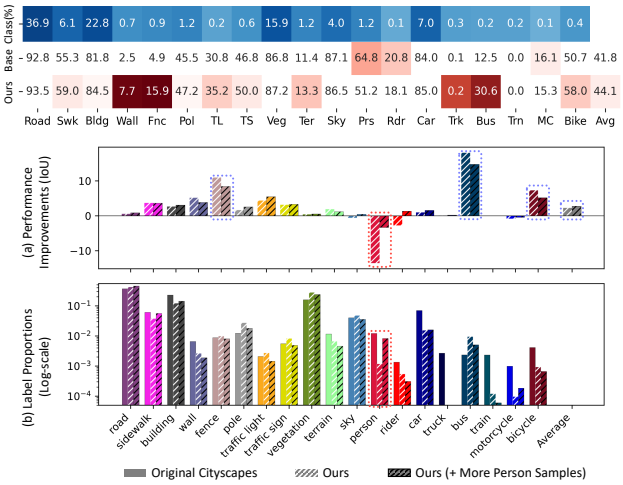


Figure 16. **Class-wise performance improvements and label proportions.** (a) Class-wise IoU improvements and (b) label proportions for Cityscapes, Ours, and Ours (+ More Person Samples). Adding 500 additional "person" samples balances label proportions. Rare classes such as "bus", "fence", and "bicycle" show clear gains, while decreases in "person" and "rider" arise from insufficient samples. Supplementing these classes further leads to more balanced improvements and a higher overall average.

demonstrates the impact of image domain alignment. Quantitatively, the CMMD, which was 1.46 for the pretrained model, decreased to 0.81 after alignment, illustrating the reduced domain gap and its contribution to performance improvement.

Table 10. **In-domain segmentation performance (mIoU) of the Pascal VOC dataset.** In the first row, we report the performance of Mask2Former trained on different fractions of the PASCAL VOC dataset (Baseline). While DatasetDM often degrades segmentation performance due to incorrect labels, CA-LoRA consistently improves the results.

Method	Fraction of the PASCAL VOC Dataset	
	100 images (~7%)	1,464 images (100%)
Baseline	44.39	72.54
DatasetDM	36.16 (-8.43)	70.78 (-1.76)
CA-LoRA (Ours)	**45.52** (+0.93)	**73.72** (+1.18)

E. Class-Specific Dataset Generation

E.1. Class-wise Segmentation Performance

In this section, we present a detailed analysis of class-wise improvements, highlighting the effectiveness of the proposed method, particularly for rare classes. Additionally, we introduce a class-balanced performance improvement strategy tailored to specific classes.

Class-wise Performance Improvements (Fig. 16) Urban-scene segmentation has distinct challenges, including class imbalance and co-occurrence issues [25], making class-wise analysis particularly important. We present the class-wise IoU improvements in Fig. 16. As shown in Fig. 16 (a), our proposed dataset generation approach proves especially effective for rare classes such as "bus", "fence", and "bicycle". However, the generated dataset often fails to improve performance in certain classes, such

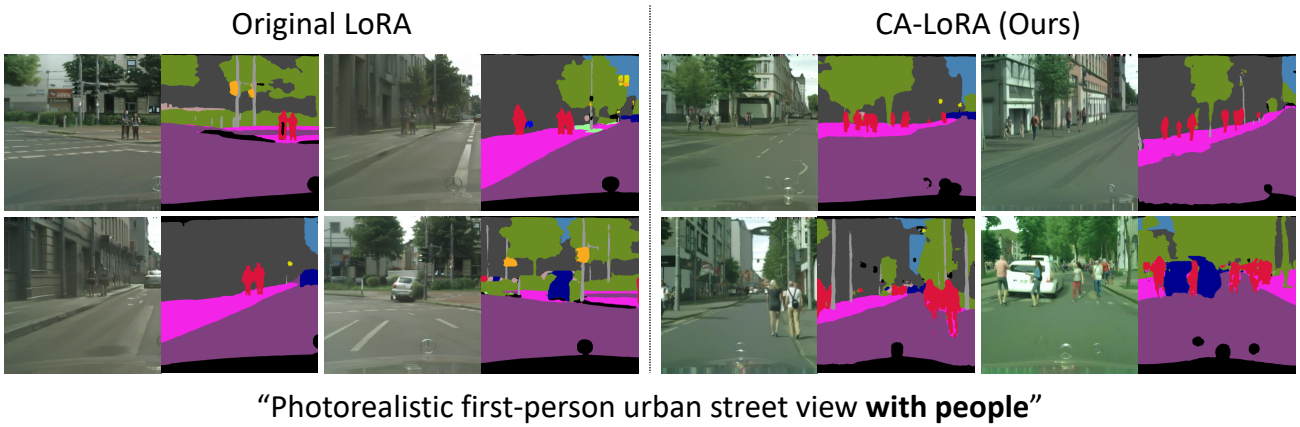


Figure 17. **Qualitative comparison between the original LoRA and CA-LoRA for generating “person” samples.** While the original LoRA produces limited variations resembling training images, CA-LoRA generates more diverse “person” scenes by learning only the style from the source dataset.

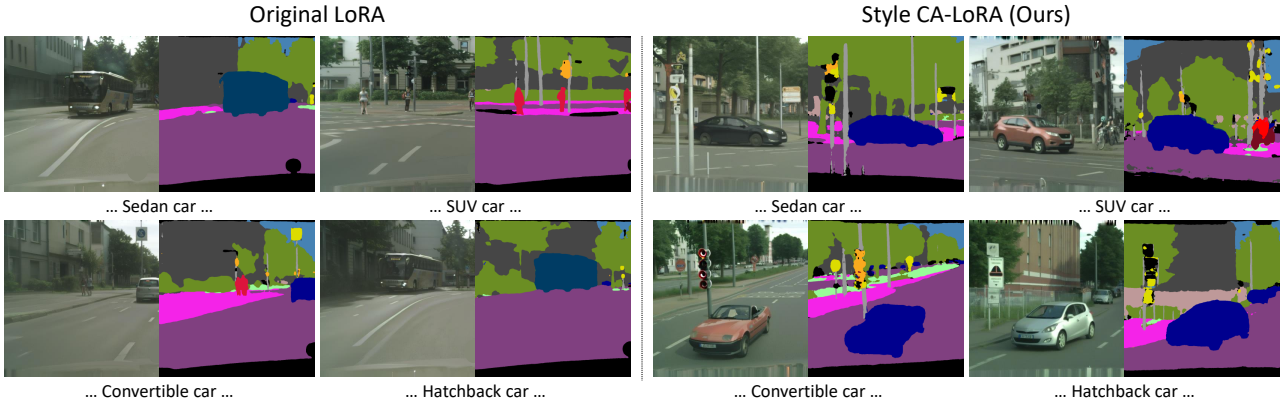


Figure 18. Generated image-label pairs showcasing various styles of cars, including sedan, SUV, convertible, and hatchback. Unlike the Original LoRA, since the Style CA-LoRA exclusively learned only the style from the Cityscapes, we can generate various types of cars in Cityscapes-style.

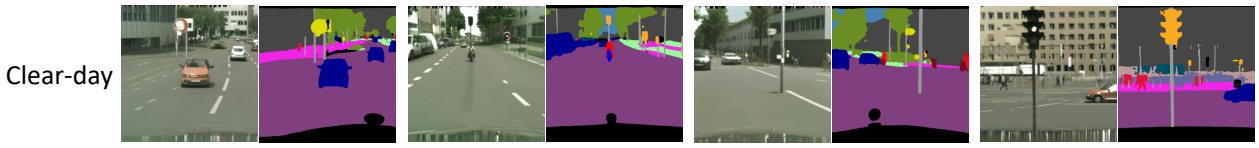
as “person” and “rider”. As illustrated in Fig. 16 (b), this degradation is primarily due to the insufficient number of generated samples for the “person” class. Since the synthetic dataset is generated randomly, disparities in label proportions can occur. To mitigate this, we propose a simple yet effective technique to increase the proportion of the target class.

Segmentation Dataset Generation Focused on a Specific Class (Figures 16 and 17) As detailed in Section 3.4 and Tab. 12, we generated the dataset using the prompt “photorealistic first-person urban street view with [Class names]”, where the class names were extracted from the label map of the training set by retrieving the names of all classes present in the label map. While the synthesized text prompt partially reflects the label proportions of the training set, it does not strictly enforce these proportions. As a result, the proposed generated dataset may exhibit misaligned label proportions, as illustrated in Fig. 16 (b).

Table 11. **In-domain segmentation performance of datasets incorporating the diverse cars dataset.** Incorporating the diverse cars dataset especially improved performance for vehicle classes such as “car”, “bus”, and “motorcycle”, leading to overall performance improvements.

Method	IoU			mIoU
	Car	Bus	Motorcycle	
Baseline	84.02	12.51	16.11	41.83
Ours	85.03	30.59	15.26	44.12
Ours (+ Diverse Cars)	**85.34**	**32.48**	**17.77**	**44.95**

To address this issue, we propose a class-specific generation approach that manually increases the target class by modifying the generation prompts. Specifically, we generated an additional 500 samples using the prompt “photoreal-



(a) Generated segmentation dataset for in-domain settings.

(b) Generated segmentation datasets for domain generalization.

Figure 19. Additional qualitative results

istic first-person urban street view with people" to increase the proportion of the "person" class.[3] Since we selectively fine-tuned the LoRA to learn only the style from Cityscapes, it enables effective manipulation using the text prompt, which the original LoRA cannot achieve, as demonstrated in Fig. 17. As illustrated in Fig. 16 (b), this approach successfully increased the proportion of the "person" class and mitigated its performance degradation. Furthermore, as shown in Fig. 16 (a), this adjustment led to additional performance improvements, increasing the average IoU from 44.12 to 44.59.

E.2. Generating Datasets with Diverse Class Names

Since the Style CA-LoRA selectively fine-tuned only the style from the in-domain Cityscapes dataset, it retains its generalization ability for text prompts such as objects. Leveraging this capability, we aim to generate a broader variety of images using more diverse class names beyond those provided in the dataset. In this experiment, we refined the prompts for generating images previously created with the simple class name "car" by subdividing them into "sedan car", "SUV car", "convertible car", and "hatchback car", as shown in Fig. 18. As illustrated in the figure, while the original LoRA fine-tuned text-to-image generation model struggles to produce diverse styles of cars, our approach reliably generates a wide variety of cars that align with the test prompts.

We then conducted an in-domain few-shot experiment (Cityscapes 0.3%) using the additional diverse cars dataset, following the experimental setup described in Section 4.1. As shown in Tab. 11, incorporating the diverse cars dataset significantly improves segmentation performance, particularly for vehicle classes. Beyond generating diverse cars, applying textual augmentations to other class names for dataset creation represents a promising direction for advancing segmentation dataset generation.

[3]We also experimented with "photorealistic first-person urban street view with person", but using "people" as the test prompt proved to be more effective in increasing the label proportion for the "person" class.

Table 12. Hyperparameters to fine-tune Stable Diffusion XL [37]. The class names are extracted from the label map in the training set by retrieving the names of all classes that appear in the label map.

Hyperparameter	Value
Rank	64
Learning rate	1e-4
Batch size	1
Training iteration	10K
Data augmentation	Random horizontal flip, Random crop
Resolution	(1024, 1024)
Learning rate scheduler	constant
Optimizer	AdamW [31]
Adam beta1	0.9
Adam beta2	0.999
Adam weight decay	0.01
Training prompt	"photorealistic first-person urban street view"
Test-time hyperparameters	
Num. inference steps	25
Guidance scale	5.0
Test prompt augmentation (In-domain)	"... with [Class names]"[4]
Test prompt augmentation (DG)	"... in [Weather Condition] with [Class names]"

Table 13. Hyperparameters to train label generator followed by DatasetDM [51].

Hyperparameter	Value
Architecture	Mask2Former-shaped label generator [51]
Learning rate	1e-4
Batch size	2 for all few-shot, 8 for fully-supervised
Training iteration (few-shot)	12k, 24k, 24k, and 48k for 0.3%, 1%, 3%, and 10%, respectively
Training iteration (fully-supervised)	90K
Data augmentation	Random horizontal flip, Random resized crop (0.5, 2.0)
Resolution	(1024, 1024)
Learning rate scheduler	PolynomialLR(power=0.9)
Optimizer	Adam [26]
Adam beta1	0.9
Adam beta2	0.999
Adam weight decay	0.0

Table 14. Hyperparameters to fine-tune Mask2Former [5]. We modify the learning rate, batch size and training iteration from the original Mask2Former training configuration.

Hyperparameter	Value
Model Architecture	Mask2Former [5]
Num. generated images	500 for all few-shot, and 3,000 for fully-supervised
Learning rate	3e-6
Batch size	2 for all few-shot, and 8 for fully-supervised
Mixed batch	real:syn = 1:1
Training iteration	30K
Data augmentation	Random horizontal flip, Random resized crop (0.5, 2.0)
Resolution	(512, 1024)
Learning rate scheduler	PolynomialLR(power=0.9)
Optimizer	AdamW [31]
Adam beta1	0.9
Adam beta2	0.999
Adam weight decay	0.05

Table 15. Hyperparameters to train domain generalization in segmentation including ColorAug, DAFormer [18], and HRDA [19], followed by DGInStyle [22].

Hyperparameter	Value (ColorAug) [54]	Value (DAFormer) [18]	Value (HRDA) [19]
Model Architecture	SegFormer	DAFormer	HRDA
Backbone	MiT-B5 [54]		
Num. generated images	500 for each weather condition (clear, foggy, night-time, rainy, and snowy)		
Learning rate	6e-5		
Batch size	2		
Training iteration	40K		
Data augmentation for Gen.	Random horizontal flip, PhotoMetricDistortion		
Data augmentation for Real	Random horizontal flip, Random crop, DACS [48]		
Resolution	(512, 512)	(512, 512)	(1024, 1024)
Learning rate scheduler	PolynomialLR(power=0.9)		
Learning rate warmup	Linear		
Learning rate warmup iteration	1500		
Learning rate warmup ratio	1e-6		
Optimizer	AdamW [31]		
Adam beta1	0.9		
Adam beta2	0.999		
Adam weight decay	0.01		
SHADE	False	True	True
RCS [18, 19]	False		

Algorithm 1 PyTorch-like Pseudocode of Concept Awareness

```
# pipe: text-to-image generation diffusers pipeline
# c: str = "photorealistic first-person urban street view"
# c_augs: List[str] = List of the augmented prompts
# t: int = pre-defined timestep
# n_img: int = number of generated images for average

unet = pipe.unet
unet = unet.requires_grad_(True)

# optimizer for clear gradients
optimizer = torch.optim.AdamW(list(filter(lambda p: p.requires_grad, unet.parameters())))

imgs = [pipe(c).images[0] for _ in n_img] # generate images

awareness = []

for img in imgs: # average over generated images
    for c_aug in c_augs: # average over augmented captions
        latent = pipe.vae.encode(img)
        noise = torch.randn_like(latent)
        noisy_latent = pipe.scheduler.add_noise(latent, noise, t)

        prompt_embeds = encode_prompt(c)
        model_pred = unet(noisy_latent, t, prompt_embeds)

        gt_diff = noise

        with torch.no_grad():
            prompt_embeds_aug = encode_prompt(c_aug)
            gt_concept = unet(noisy_latent, t, prompt_embeds_aug)

        loss_diff = torch.nn.functional.mse_loss(model_pred, gt_diff)
        loss_concept = torch.nn.functional.mse_loss(model_pred, gt_concept)

        loss_diff.backward(retain_graph=True)
        grads_diff = get_unet_grads(unet) # Algorithm 2
        optimizer.zero_grad()

        loss_concept.backward()
        grads_concept = get_unet_grads(unet) # Algorithm 2
        optimizer.zero_grad()

        awareness.append(grads_concept / grads_diff)
awareness_avg = average_gradients(awareness) # Algorithm 2
```

Algorithm 2 PyTorch-like Helper Functions for Concept Awareness

```
def getattr_recursive(module, attrs: List[str]):
    target_module = module
    for attr in attrs:
        target_module = getattr(target_module, attr)
    return target_module

def get_unet_grads(unet):
    grads = {'to_q': [], 'to_k': [], 'to_v': [], 'to_out.0': []}
    for attn_name in unet.attn_processors.keys():
        attn_module = getattr_recursive(unet, attn_name.split('.')[:-1])

        for proj_name in grads.keys():
            proj = getattr_recursive(attn_module, proj_name.split('.'))
            head_dim = 1 if proj_name == 'to_out.0' else 0
            grads_chunk = torch.chunk(proj.weight.grad.cpu(), attn_module.heads, dim=head_dim)
            grads[proj_name].append([(grad ** 2).mean().sqrt().item() for grad in grads_chunk])

    return grads

    def average_gradients(grads):
        grad_avg = {'to_q': [], 'to_k': [], 'to_v': [], 'to_out.0': []}

        for key in grad_avg:
            for grad in grads:
                grad_avg[key].append(grad[key])
            grad_avg[key] = torch.mean(torch.tensor(grad_avg[key]), dim=0)

        return grad_avg
```

Algorithm 3 PyTorch-like Pseudocode of Modifying forward function of CA-LoRA

```
# F: torch.nn.functional

def modify_to_ca_lora(layer, reduced_layer):
    # layer: diffusers.models.LoRACompatibleLinear
    # reduced_layer: 'A' or 'B'

    def ca_lora_set_lora_layer(self: LoRACompatibleLinear):
        def set_lora_layer(lora_layer, indices):
            self.lora_layer = lora_layer
            if indices is not None:
                self.indices = indices
        return set_lora_layer

    def ca_lora_set_forward(self: LoRACompatibleLinear):
        def forward(hidden_states: torch.Tensor, scale: float = 1.0):
            if self.lora_layer is None:
                return F.linear(hidden_states, self.weight, self.bias)
            else:
                if self.indices is not None:
                    # CA-LoRA (start)
                    org = F.linear(hidden_states, self.weight, self.bias)
                    if reduced_layer == 'B':
                        org[:, :, self.indices] += scale * self.lora_layer(hidden_states)
                    else:
                        org += scale * self.lora_layer(hidden_states[:, :, self.indices])
                    return org
                    # CA-LoRA (end)
                else:
                    return F.linear(hidden_states, self.weight, self.bias) + scale *
                    self.lora_layer(hidden_states)
        return forward

    layer.set_lora_layer = ca_lora_set_lora_layer(layer)
    layer.forward = ca_lora_set_forward(layer)

    return layer
```

Algorithm 4 PyTorch-like Pseudocode of selecting projection layers for CA-LoRA

```
def apply_ca_lora(unet, selected_layers, rank):
    for attn_processor_name in unet.attn_processors.keys():
        _selected_layers = [selected_layer for selected_layer in selected_layers if
        '.'.join(attn_processor_name.split('.')[:-1]) in selected_layer]
        if len(_selected_layers) == 0:
            continue
        attn_module = getattr_recursively(unet, attn_processor_name.split('.')[:-1])
        # getattr_recursively: Algorithm 2
        dim_head = attn_module.out_dim // attn_module.heads
        for layer_type in ('to_q', 'to_k', 'to_v', 'to_out.0'):
            selected_layers_proj = [selected_layer for selected_layer in _selected_layers if layer_type in
            selected_layer] is_out = layer_type == 'to_out.0'
            if len(selected_layers_proj) == 0:
                continue
            projection_layer = getattr_recursively(attn_module, layer_type.split('.'))
            # getattr_recursively: Algorithm 2

            # Projection-wise CA-LoRA (start)
            head_indices = sorted([int(selected_layer.split('.')[-1][1:]) for selected_layer in
            selected_layers_proj])
            indices = sum([list(range(dim_head * head_idx, dim_head * (head_idx + 1))) for head_idx in
            head_indices], [])
            # Indices are split grouped by the dim_head
            # Projection-wise CA-LoRA (end)

            projection_layer = modify_to_ca_lora_linear(projection_layer, reduced_layer='A' if is_out else
            'B')
            # modify_to_ca_lora_linear: Algorithm 3

            if is_out:
                projection_layer.set_lora_layer(
                  LoRALinearLayer(
                    in_features=len(indices),
                    out_features=projection_layer.out_features,
                    rank=rank),
                  indices)
            else:
                projection_layer.set_lora_layer(
                  LoRALinearLayer(
                    in_features=projection_layer.in_features,
                    out_features=len(indices),
                    rank=rank),
                  indices)
    return unet
```

References

[1] Dmitry Baranchuk, Andrey Voynov, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. In *International Conference on Learning Representations*, 2022. 1, 3

[2] Samyadeep Basu, Keivan Rezaei, Priyatham Kattakinda, Vlad I Morariu, Nanxuan Zhao, Ryan A Rossi, Varun Manjunatha, and Soheil Feizi. On mechanistic knowledge localization in text-to-image generative models. In *Forty-first International Conference on Machine Learning*, 2024. 2

[3] Yasser Benigmim, Subhankar Roy, Slim Essid, Vicky Kalogeiton, and Stéphane Lathuilière. One-shot unsupervised domain adaptation with personalized diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 698–708, 2023. 2, 3, 5

[4] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023. 2, 5

[5] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022. 5, 7, 13, 18, 19, 24

[6] Sungha Choi, Sanghun Jung, Huiwon Yun, Joanne T Kim, Seungryong Kim, and Jaegul Choo. Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11580–11590, 2021. 2

[7] Sungha Choi, Seunghan Yang, Seokeon Choi, and Sungrack Yun. Improving test-time adaptation via shift-agnostic weight regularization and nearest source prototypes. In *European Conference on Computer Vision*, pages 440–458. Springer, 2022. 2

[8] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[9] Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. *arXiv preprint arXiv:2311.11696*, 2023. 2

[10] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024. 2

[11] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010. 6, 12, 19

[12] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022. 2

[13] Rui Gong, Martin Danelljan, Han Sun, Julio Delgado Mangas, and Luc Van Gool. Prompting diffusion representations for cross-domain semantic segmentation. *arXiv preprint arXiv:2307.02138*, 2023. 3

[14] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 2

[15] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024. 2

[16] Shwai He, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. *arXiv preprint arXiv:2210.04284*, 2022. 2

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2

[18] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9924–9935, 2022. 2, 5, 6, 24

[19] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Hrda: Context-aware high-resolution domain-adaptive semantic segmentation. In *European conference on computer vision*, pages 372–391. Springer, 2022. 2, 5, 6, 24

[20] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 1, 2, 4, 5

[21] Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9307–9315, 2024. 7, 17, 18

[22] Yuru Jia, Lukas Hoyer, Shengyu Huang, Tianfu Wang, Luc Van Gool, Konrad Schindler, and Anton Obukhov. Dginstyle: Domain-generalizable semantic segmentation with image diffusion models and stylized semantic control. In *Synthetic Data for Computer Vision Workshop@ CVPR 2024*, 2023. 1, 2, 5, 6, 14, 18, 19, 24

[23] Ugur Ali Kaplan, Margret Keuper, Anna Khoreva, Dan Zhang, and Yumeng Li. Domain-aware fine-tuning of foundation models. *arXiv preprint arXiv:2407.03482*, 2024. 3

[24] Aliasghar Khani, Saeid Asgari Taghanaki, Aditya Sanghi, Ali Mahdavi Amiri, and Ghassan Hamarneh. Slime: Segment like me. *arXiv preprint arXiv:2309.03179*, 2023. 5

[25] Dongseob Kim, Seungho Lee, Junsuk Choe, and Hyunjung Shim. Weakly supervised semantic segmentation for driving

scenes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2741–2749, 2024. 20

[26] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 23

[27] Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*, 2023. 2

[28] Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. *arXiv preprint arXiv:2210.11466*, 2022. 2

[29] Daiqing Li, Huan Ling, Seung Wook Kim, Karsten Kreis, Sanja Fidler, and Antonio Torralba. Bigdatasetgan: Synthesizing imagenet with pixel-wise annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21330–21340, 2022. 1, 3

[30] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024. 2

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 23, 24

[32] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022. 14

[33] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999, 2017. 5

[34] Quang Nguyen, Truong Vu, Anh Tran, and Khoi Nguyen. Dataset diffusion: Diffusion-based synthetic data generation for pixel-level semantic segmentation. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[35] Minho Park, Jooyeol Yun, Seunghwan Choi, and Jaegul Choo. Learning to generate semantic layouts for higher text-image correspondence in text-to-image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7591–7600, 2023. 1, 3

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 14

[37] 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. 1, 2, 5, 13, 14, 23

[38] 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 *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 7

[39] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1 (2):3, 2022. 2

[40] Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. Tied-LoRA: Enhancing parameter efficiency of LoRA with weight tying. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024. 4

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1, 2, 4, 13

[42] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023. 2

[43] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 1, 2

[44] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7374–7383, 2019. 5

[45] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10765–10775, 2021. 5, 18, 19

[46] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022. 1, 2

[47] Joonghyuk Shin, Minguk Kang, and Jaesik Park. Fill-up: Balancing long-tailed data with generative models. *arXiv preprint arXiv:2306.07200*, 2023. 1

[48] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. Dacs: Domain adaptation via cross-domain mixed sampling. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1379–1389, 2021. 24

[49] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffu-

sion models. https://github.com/huggingface/diffusers, 2022. 14

[50] Haofan Wang, Qixun Wang, Xu Bai, Zekui Qin, and Anthony Chen. Instantstyle: Free lunch towards style-preserving in text-to-image generation. *arXiv preprint arXiv:2404.02733*, 2024. 2

[51] Weijia Wu, Yuzhong Zhao, Hao Chen, Yuchao Gu, Rui Zhao, Yefei He, Hong Zhou, Mike Zheng Shou, and Chunhua Shen. Datasetdm: Synthesizing data with perception annotations using diffusion models. *Advances in Neural Information Processing Systems*, 36:54683–54695, 2023. 1, 2, 3, 4, 5, 13, 23

[52] Weijia Wu, Yuzhong Zhao, Mike Zheng Shou, Hong Zhou, and Chunhua Shen. Diffumask: Synthesizing images with pixel-level annotations for semantic segmentation using diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1206–1217, 2023. 1, 3, 5

[53] Yichao Wu, Yafei Xiang, Shuning Huo, Yulu Gong, and Penghao Liang. Lora-sp: Streamlined partial parameter adaptation for resource-efficient fine-tuning of large language models, 2024. 4

[54] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34: 12077–12090, 2021. 6, 24

[55] Peng Xing, Haofan Wang, Yanpeng Sun, Qixun Wang, Xu Bai, Hao Ai, Renyuan Huang, and Zechao Li. Csgo: Content-style composition in text-to-image generation. *arXiv preprint arXiv:2408.16766*, 2024. 2

[56] Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. Open-vocabulary panoptic segmentation with text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2955–2966, 2023. 5

[57] Lihe Yang, Xiaogang Xu, Bingyi Kang, Yinghuan Shi, and Hengshuang Zhao. Freemask: Synthetic images with dense annotations make stronger segmentation models. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[58] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020. 5

[59] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023. 4, 5

[60] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10145–10155, 2021. 1, 3, 5

[61] Hongbo Zhao, Bolin Ni, Junsong Fan, Yuxi Wang, Yuntao Chen, Gaofeng Meng, and Zhaoxiang Zhang. Continual forgetting for pre-trained vision models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28631–28642, 2024. 4